Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Physics

submitted by

Christof Sauer

born in Fürth

2019

Towards a Data-Driven Simulation of QCD Radiation with Generative Models utilizing Machine Learning Methods.

This Master thesis has been carried out by Christof Sauer
at the

Physikalisches Institut Heidelberg

under the supervision of

Herrn Prof. Dr. André Schöning

Comment

The present document is a revised version of the original work that has been submitted on August 30, 2019 as part of the fulfillment for the degree of Master of Science in Physics at the University of Heidelberg. Further sugestions for improvements as well as hints concerning mistakes of any kind are mostly welcome. Regarding this matter, please contact: csauer@physi.uni-heidelberg.de.

CERN, Friday, 22th November 2019,

Christof Sauer

Acknowledgements

This Master's thesis owes its existence to the support, help, and inspiration of several people who should be acknowledged at this occasion.

First and foremost, I want to express my sincere gratitude to my advisor, Prof. Dr. André Schöning, for the great opportunity to complete my Bachelor's (2017) as well as my Master's degree under his supervision in the ATLAS group at the Physikalisches Institut at the Ruprecht-Karls-Universität, Heidelberg. Furthermore, I would like thank him for his support regarding my future academic career in his group for years to come. I must not forget to give thanks to Prof. Dr. Monica Dunford for her willingness to be the second reviewer of this report.

Besides my advisors, I want to articulate my deepest sense of gratitude and appreciation to Dr. Danilo Enoque Ferreira de Lima for his marvellous encouragement, guidance – without patronizing! –, and patience throughout the entire duration of my stay. Our fruitful, professional, and personal conversations were of inestimable value and will remain cherished memories. Today's science always is a joint effort; in my daily work in our research group, I have been blessed with friendly and cheerful colleagues who heartily welcomed me in their community among them and made my entire stay a very pleasant and enjoyable experience. My thanks go to (in alphabetical order): Anjali Krishnan, Arthur Bolz, Dr. Christoph Falk Anders, Dr. Louis Helary, Marta Czurylo, Dr. Mathis Kolb, Dr. Mykhailo Lisovy, and Tamasi Kar.

I would like to thank the former school principal, Ursula Engelberger, for her support in difficult phases of life and my maths teacher Alfons Frink. I also want to express special thanks to my erstwhile physics teacher at the Gymnasium am Römerkastell, Gudrun Hattemer, for her encouragement and confirmation to study physics – a decision I have never regretted to the present day.

In my private surrounding, I want to thank Marius-Hergen and Björn-Arved Rauch from the bottom of my heart for their long-standing loyal friendship and mental support for more than eighteen years. In addition, my gratitude goes to my friends and fellow students Alexander Uth, Markus Kardorff and my recent acquaintance Nina Alisa Laura Oser for their support and friendship.

Last, but not least, I take the opportunity to express my profound, boundless gratitude and love to my parents, Ursula Elaine and Hubertus Udo Sauer, who supported me throughout my entire life, spiritually and materially, and encouraged me in all of my life decisions up to this very point.

Dedicated to my beloved parents, Hubertus Udo and Ursula Elaine Sauer.

Zusammenfassung

Im Kontext dieser Masterthesis wurde die Applikabilität sog. »erzeugender« bzw. »genererierender Modelle« (aus dem Englischen *Generative Models*) unter Verwendung gegenwartsnaher Methoden des maschinellen Lernens – als wissenschaftlicher Teilbereich der »künstlichen Intelligenz« – zur Simulation von QCD resp. Gluonen induzierter Bremsstrahlung untersucht. Abseits der prinzipiellen Anwendbarkeit der o. g. Methoden lag das Hauptaugenmerk auf der direkten Gegenüberstellung der beiden prominentesten Ansätze zu deren praktischen Realisierung: *(Gauß-) Variational Auto-Encoders* (VAEs) sowie die auf der Earth Mover's Metrik basierenden, jüngsthin ersonnenen *Wasserstein Generative Adversarial Networks* (WGANs). Die Vergleichsstudie offenbarte eine deutliche Überlegenheit der Letztgenannten gegenüber den VAE im Bereich Qualität und Diversität der simulierten Daten, und deckt sich somit in konsistenter Weise mit ähnlichen Beobachtungen in anderen Anwendungsgebieten. Weiter wurden sowohl Wasserstein GANs als auch VAEs mit Rekurrenten Neuronalen Netzwerken (RNN) in Form von Long-Short-Term-Memory-Netzen (LSTM) verquickt, um den sequentiellen Charakter des zugrundeliegenden Teilchenschauers möglichst wirklichkeitsgetreu zu Modellieren. Wie sich jedoch herausstellte, konnte durch die Hinzunahme von LSTM-Netzen keine nennenswerte Verbesserung erzielt werden. Indes zeigte sich jedoch, dass die Faktorisierung der generativen Modelle in Teilchenschauer und Matrix-Element qua bedingter neuronaler Netzwerke der Leistung zuträglich ist, wobei die Netzwerke auf die Energie und die Pseudorapidität des Teilchen-Jets konditioniert wurden.

Abstract

This master's thesis studied the possible employment of generative models – using state-of-the-art machine learning methods – concerning the simulation of QCD- respectively gluon-induced radiation. Besides the principle applicability and feasibility of the methods presented in the course this report, the main focus of attention was on the direct comparison of the two most prominent approaches regarding the implementation of generative models, i.e., (Gaussian) Variational Autoencoders (VAEs) and the novel Wasserstein Generative Adversarial Networks (WGANs) that are based on the earth mover's distance. The comparative study of the two paradigms revealed clear superiority of WGANs compared to VAEs regarding not only the quality but also the diversity of the generated data. This outcome is consistent with the results that have been obtained in similar studies in other fields of application. Furthermore, both models have been combined with Recurrent Neural Networks (RNNs) using Long Short-Term Memory (LSTM) cells to mimic the underlying sequential character of the particle shower. Yet, it turned out that the combination of generative models and RNNs is disadvantageous and unsuited to model the actual splitting process that is not directly present in the training data. However, it became apparent that the conditioning of the model on the energy and the pseudorapidity of the jet, i.e., the marginalization of the matrix element information has beneficial effects on the model's overall performance.

Οἶδα οὐδὲν εἰδώς.

– Plato (Apol. 21d), “Apology of Socrates (Απολογία Σωκράτους)”

Contents

Page

List of Figures xvii

List of Plots xxi

Nomenclature xxviii

Introduction 1

Part I

1 Theory 5

1.1 The Standard Model of particle physics 6
1.1.1 History of the Standard Model 6
1.1.2 Elementary particle content 8
1.1.3 The Standard Model as a QFT 10
1.1.4 The QCD Lagrangian 11

1.2 Perturbative QCD 13
1.2.1 Renormalization and running coupling 13
1.2.2 Asymptotic freedom and colour confinement 14
1.2.3 Soft and collinear limits of QCD 16
1.2.4 Factorization theorem and DGLAP equations 17
1.2.5 Parton shower and Sudakov form factors 20

1.3 QCD phenomenology 22
1.3.1 Jets in particle physics 23
1.3.2 Infrared and collinear safety 24
1.3.3 Cone based algorithms 25
1.3.4 Sequential recombination algorithms 26
The k_t algorithm 27
The Cambridge-Aachen algorithm 27
The anti-k_t algorithm 27
1.3.5 Jet-related observables 28
Jet-shape 28
Jet-width 28
N-subjettiness 29

2 Event Simulation in HEP **31**

2.1 The "Monte Carlo" method 31

2.1.1 Monte Carlo integration 32

2.1.2 Monte Carlo simulation 33

2.2 Monte Carlo event simulation 34

2.2.1 The larger picture 34

2.2.2 Fixed-order matrix element expansion 35

2.2.3 Parton shower simulation 36

2.2.4 Event simulation and event topology 37

2.3 Types and examples of MC event generators 39

2.4 Challenges in MC event generation 41

2.4.1 Double-counting 41

Part II

3 Machine Learning **45**

3.1 A brief history of neural networks 45

3.2 Trinity of machine learning 49

3.3 Artificial neural networks 49

3.3.1 Feed-forward neural networks 51

3.3.2 Recurrent neural networks 54

3.4 Training of neural networks 56

3.4.1 The loss function 57

3.4.2 Gradient descent and optimizers 57

3.4.3 The backpropagation algorithm 60

3.4.4 Training stability, regularization and normalization 60

3.5 Variational Autoencoders 62

3.5.1 Latent variable models 63

3.5.2 Gaussian Variational Autoencoders 65

3.5.3 Implementation 67

3.5.4 Conditional variational autoencoders 68

3.6 Generative Adversarial Networks 69

3.6.1 GANs according to Ian Goodfellow *et al.* 69

3.6.2 Wasserstein GANs 75

Conditional Wasserstein GANs 80

3.6.3 A kaleidoscope of GANs 81

4 Training Data and Preprocessing **83**

4.1 Event simulation 83

4.1.1 Underlying hard subprocess and parton shower 84

4.2 Data preprocessing 87

4.3 "Invertible preprocessing" 92
4.4 Training data 95
4.4.1 Average jet image 95
4.4.2 Information loss 98

Part III

5 Jets with Gaussian Variational Autoencoders 107
5.1 Model architecture specification 107
5.2 Latent space and hyperparameter configuration 112
5.3 Unconditioned VAEs 115
5.3.1 Samples and average jet images 116
5.3.2 Kinematic distributions and jet observables 120
5.4 Conditional VAEs 124
5.4.1 Conditioning the model 125
5.4.2 Samples and average jet images 126
5.4.3 Conditioning of the model 129
5.4.4 Kinematic distributions 130
5.4.5 Other jet observables 134
5.4.6 Why the model "fails" 137
5.5 Conditioned VAEs with RNNs 138
5.5.1 Why generative models with RNNs? 138
5.5.2 Combining VAEs with RNNs 139
5.5.3 Samples, average jet image and jet observables 142
5.6 Final notes 148

6 Jets with Wasserstein Generative Adversarial Networks 149
6.1 Architecture specification 149
6.2 Hyperparameter configuration 151
6.3 Unconditioned WGAN – another attempt 155
6.3.1 Linear versus logarithmic scale 156
6.3.2 Samples and average jet images 158
6.3.3 Kinematic distributions 159
6.3.4 Other jet observables 161
6.4 Conditional WGAN 163
6.4.1 Training and model convergence 163
6.4.2 Samples and average jet images 166
6.4.3 Kinematic distributions 170
6.4.4 Other jet observables 173
6.5 Conditioned WGANs with RNNs 178
6.5.1 Training and convergence 178

6.5.2 Samples and average jet image 180
6.5.3 Kinematic distributions 182
6.5.4 Jet observables 183
6.6 Final notes 185

7 Conclusion 189

7.1 Discussion 189
7.2 Future research suggestions 192
7.2.1 Bayesian networks – “I know what I don’t know” 192
7.2.2 Image-to-image or jet-to-jet translation 193
7.2.3 RNNs for higher jet multiplicities 195

List of Figures

1.1 The basic building blocks of QCD Feynman diagrams: (*left*) quark-gluon vertex (*middle*) three-gluon vertex (*right*) four-gluon vertex. 12
1.2 Visualization of the factorization theorem for hadron-hadron interactions (1.2a) and deep inelastic scattering (1.2b) with the respective flavour-dependent PDFs $f_{i/h}$. 18
1.3 Pictorial representation of the evolution of the parton distribution functions via the DGLAP equations (adapted from Höche, 2014, Fig. 1, p. 3). 19
1.4 Parton distribution functions (CTEQ6M) for different partons at the energy scale $Q = 2\,\mathrm{GeV}$ (1.4a) and $Q = 100\,\mathrm{GeV}$ (1.4b) (adapted from Pumplin *et al.*, 2002, Fig. 1, p. 8). 19
1.5 The collinear factorization theorem for the example of a matrix element $\mathcal{M}_{n+2}$ of a given process with $n + 2$ external legs and a final state a that branches (splits) two times with ordered splitting angles θ. 20
1.6 Left: a two-jet event with clear separation of the final state particles; right: a three-jet event with significant overlap among radiation from two different sources. The dashed, blue line indicated the direction of the respective underlying parton. 23
1.7 Visualization of infrared safety: the configuration of reconstructed jets must not change with the emission of another *soft* particle. . 24
1.8 Visualization of collinear safety: the configuration of reconstructed jets must not change with one particle replace by two collinear particles. 25

2.1 NLO Feynman diagrams contributing to the $q\bar{q}$ annihilation cross-section. 35
2.2 Individual components in the simulation chain of proton-proton collisions: incoming parton densities heading towards each other with the *hard* interaction indicated by the central red blob and a second hard scattering (purple). The tree-like structure surrounding the hard interaction is bremsstrahlung simulated through parton showers. The non-perturbative regime, i.e., the transition from partons to hadrons is depicted in light-green, which is followed by the formation of excited states that further decay to final-state particles measured in the detector (adapted from Höche [2014], Fig. 3, p. 6). 37

2.3 Double counting at leading order. Along the horizontal axis, the parton shower algorithm adds radiation; in doing so, it generates diagrams which have an equivalent in leading order processes with higher multiplicities. 42

3.1 The perceptron according to Frank Rosenblatt with bias term. . 46
3.2 "Shallow" feed-forward neural network with n inputs, one hidden layer with k neurons, and m outputs. The trainable weights of the networks are represented by the connections between the individual neurons. 51
3.3 A recurrent neural network as a loop diagram (*left*) and its unfolding in time sequences (*right*). 55
3.4 An LSTM cell and its weighted gates. 55
3.5 Fictional contour of a complicated non-convex loss function $\mathcal{L}(\boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \mathbb{R}^N$ for a two-dimensional subspace spanned by the weights ($\mathbb{R} \times \mathbb{R} \subset \mathbb{R}^N$). The trajectories represent two possible paths of the gradient descent algorithm (different learning rates and/or initial conditions) to reach the (local) minimum in this subspace. 58
3.6 Feature map tensor with N batches, C channels and a spatial dimension of H (height) and W (width) (adapted from [Wu and He, 2018, Fig. 2, p. 3]). 62
3.7 Graphical model representation of a general latent space model (*left*), a variational autoencoder (*middle*, section 3.5.2) and a conditional variational autoencoder (*right*, section 3.5.4). In the graphs, N refers to the number of times $\boldsymbol{z}$ and $\boldsymbol{x}$ are sampled, while the parameters/weights $\boldsymbol{\theta}(\boldsymbol{\phi})$ remain fixed. The solid lines indicate the path of the generative process, while the dashed lines denote the variational approximation. 64
3.8 A generative model pictured as a smooth map g_θ from a low-dimensional coordinate space/representation $\mathcal{Z}$ to a high-dimensional manifold $\mathcal{X}$ (adapted from Shao *et al.* [2017]). 64
3.9 The author being used as "guinea pig" to demonstrate the operating principle of a GVAE. The encoder network (inference network) receives a set of images $\boldsymbol{x}$ and learns a hidden and compressed representation of the data. In a GVAE, the latent space is constrained to be Gaussian (second term in Equation 3.40). The decoder network (generative model) samples from a Gaussian $\boldsymbol{z} \sim \mathcal{N}(0, \mathbb{1})$ and tries to reconstruct the input as close as possible (regression task). Note: the array of images on the right side has actually been generated by a neural network. 68
3.10 Graphical illustration of the training of generative adversarial networks as proposed by Goodfellow *et al.* The discriminator model/network f_ϕ alternately receives real data from the training set and fake data produced by the generator network g_θ. The loss function is evaluated, and the weights (trainable parameters) of the networks are updated accordingly. The procedure is repeated till convergences or termination of the training. 71

3.11 An two-dimensional example for a possible transportation plan γ that corresponds to a joint probability distribution with the marginal distributions $\mathbb{P}_r$ and $\mathbb{P}_g$. 76
3.12 Graphical illustration of the training of generative adversarial networks with the Wasserstein metric and gradient penalty as introduced by Arjovsky *et al.*, 2017. The critic model/network f_ϕ measures the Earth mover's distance between the generated distribution $\mathbb{P}_g$ and the real distribution of the training data $\mathbb{P}_r$. 80
3.13 Graphical illustration of the training of conditional generative adversarial networks with the Wasserstein metric and gradient penalty. Both models, the generator and the critic, additionally receive a list of conditioning label $\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$. 81

4.1 Overview of several cross-section measurements of selected Standard Model processes compared to the corresponding theoretical expectations (adapted from Gemme, 2016, Fig. 1, p. 3). 84
4.2 Small selection of leading order tree-level Feynman diagrams for the processes $pp \to gq$+0jets, $pp \to q\bar{q}$+0jets, and $pp \to gg$+0jets for strong interactions solely (see Section 1.2). 85
4.3 Example of the leading order Feynman diagram(s) for the hadronic final state of the $t\bar{t}$ system for different initial states and interactions. 85
4.4 First and second preprocessing step: a Lorentz boost as well as a rotation in the transverse plane are applied to transform the leading p_T (sub)jet to the origin of the detector $(0, 0)$ in the η-ϕ grid. Afterwards, a small region of the detector is selected according to condition 4.11 (adapted from Thaler and Van Tilburg, 2011, Fig. 1 (c,d), p. 4). 89
4.5 Third preprocessing step: the image is rotated by $\alpha_\mathrm{rot}^\mathrm{PC}$ such that the first principal component axes points at at 12 o'clock (adapted from Thaler and Van Tilburg, 2011, Fig. 1 (c,d), p. 4). 90
4.6 Overlaying pixels between the rotated and non-rotated image. . . 91
4.7 A parity transformation $\boldsymbol{P}$ is applied such that the $E_r^\mathrm{img} > E_l^\mathrm{img}$, i.e., the largest energy fraction is always on the right-half of the plane (adapted from Thaler and Van Tilburg, 2011, Fig. 1 (c,d), p. 4). 92

5.1 Network architecture of the variational autoencoder. 108
5.2 A simple residual block. 109
5.3 Convolutional and fully-connected layers. 110
5.4 The decoder combined with an RNN. As an input, the LSTM layer receives two conditioning labels, jet energy E^jet and the pseudorapidity η^jet, as well as the seed $\boldsymbol{z}_t$, which is different for each time step t. The output for each time step, i.e., each loop cycle, is a normalized vector and a scalar that corresponds to the energy at time t. 140

7.1 Image-to-image translation with the condition of cycle-consistency (adapted from Zhu *et al.* [2017] Fig. 1, p. 1). 194
7.2 Extrapolation from one class to several classes (adapted from Zhu *et al.* [2017] Fig. 1, p. 1) 194

List of Plots

1.1 Overview of several measurements of the running (strong) coupling $\alpha_s(Q)$ (adapted from Khachatryan and others, 2015, Fig. 7, p. 12). 15

1.2 N-subjettiness distribution τ_2 (1.2a) and τ_2 (1.2b) for QCD and boosted W jets and a mass window $145\,\text{GeV} < m^{\text{jet}} < 205\,\text{GeV}$. The discrimination power of this variable manifests it self in the ratio $\tau_{32} := \tau_3/\tau_2$ (1.2c). (adapted from (Thaler and Van Tilburg, 2011, Fig. 2(b,c), 3(b), p. 8). 29

3.1 The most frequently used activation functions σ in machine learning: sigmoid, hyperbolic tangent, and Rectified Linear Units. . . 51

3.2 Discrimination boundary, which was learned by the neural network, for the raw data (a) and its new representation in the transformed coordinate system of the second hidden layer (b). 53

3.3 Failed classification due to inopportune set of hyperparameters. . 53

3.4 VAE trained on the MNIST data set. $\mathbb{R}^2$ latent space representation (*left*) where each different color corresponds to a digit in the set $\{0, \ldots, 9\}$ and randomly generated data (*right*) (Taken from Davidson *et al.* [2018] Fig. 2a, p. 5; Fig. 10. p. 18). 68

4.1 Distribution of pixel intensity values E^{pix} for different scales. . . 93

4.2 Reconstructed average leading p_{T} QCD jet for minimal preprocessing (*left*) and full preprocessing (*right*) for $200,000$ events. 96

4.3 Three random QCD events. 96

4.4 Reconstructed average leading p_{T} W jet for minimal preprocessing (*left*) and full preprocessing (*right*) for $200,000$ events and a p_{T} cut $p_{\text{T}}^{\text{jet}} \geq 200\,\text{GeV}$. 97

4.5 Three random W events. 97

4.6 Distribution of rotation angles $\alpha_{\text{rot}}^{\text{PC}}$ obtained from the PCA (4.2) (*left*), parity values P as defined in Section 4.16 (*middle*) and the correlation coefficient $\rho_{\alpha_{\text{rot}}^{\text{PC}},\text{P}}$ between $\alpha_{\text{rot}}^{\text{PC}}$ and P (*right*). 98

4.7 Loss of information for the reconstructed jet mass for different preprocessing configurations. 99

4.8 Comparison of the reconstructed jet mass in the Full Detector (FD) and the Cut Window (CW). 100

4.9 Loss of information for the reconstructed τ_1 for different preprocessing configurations. 101

4.10 Loss of information for the reconstructed number of active pixels N_0^{pix} for different preprocessing configurations. 102

4.11 Bin-to-bin-migration as a function of the energy threshold $E_{\text{th}}^{\text{pix}}$. . 103

4.12 Number of active pixels $N^{\mathrm{pix}}_{E^{\mathrm{pix}}>E^{\mathrm{pix}}_{\mathrm{th}}}$ for different threshold cuts. . 103

5.1 The ELBO and the KL-divergence (see Equation 3.41) versus the number of training iterations for different dimensions of the latent space dim($\mathcal{Z}$). 113
5.2 The ELBO versus the number of training iterations for different learning rates α_l (5.2a) and various optimization algorithms (5.2b). 114
5.3 Three randomly simulated QCD jets. 116
5.4 Three randomly simulated W jets. 117
5.5 Occupancy according to Equation 5.2. 117
5.6 Average jet image for $50,000$ events and $40,000$ iterations. 118
5.7 nth statistical moment for generated QCD and W jets. 119
5.8 Reconstructed energy E^{img} for $50,000$ events. 120
5.9 Energy distribution in a two-dimensional projection of the 100-dimensional latent space $\mathcal{Z}$. 121
5.10 Correlation between $\tilde{z}$ and the reconstructed energy E^{img}. 122
5.11 The ELBO and its components. 123
5.12 Reconstructed mass spectrum m^{img} for $50,000$ events. 123
5.13 Correlation between the reconstructed average mass m^{img} and the transverse momentum $p_{\mathrm{T}}^{\mathrm{img}}$ for $50,000$ events. 124
5.14 Three randomly simulated QCD jets. 126
5.15 Three randomly simulated W jets. 127
5.16 Occupancy according to Equation 5.2. The reported errors $\delta\bar{o}$ are statistical uncertainties based on "propagation of errors". 127
5.17 Average jet image for $50,000$ events and $40,000$ iterations. 128
5.18 The ELBO (see Equation 3.39) and its components. 128
5.19 Input energy E^{jet} and reconstructed jet energy E^{img}. 129
5.20 Correlations in the latent space. 130
5.21 Reconstructed jet mass m^{img} for $50,000$ events. 131
5.22 Pixel activation values on an event-on-event base for $50,000$ images/events and $40,000$ iterations. 132
5.23 Logarithmic representation of the pixel activation values on an event-on-event base for $50,000$ events. 132
5.24 n^{th} statistical moment for generated QCD and W jets for a *logarithmic scale* (*cf.* Figure 5.7). 133
5.25 1-subjettiness for generated QCD and W jets. 134
5.26 2-subjettiness for generated QCD and W jets. 135
5.27 21-subjettiness for generated QCD and W jets. 135
5.28 Correlation between the reconstructed mass m^{img} and the transverse momentum $p_{\mathrm{T}}^{\mathrm{img}}$ for $50,000$ events. 136
5.29 Correlation between the reconstructed mass m^{img} and the average momentum per constituent $\frac{p_{\mathrm{T}}^{\mathrm{img}}}{N_0^{\mathrm{pix}}}$ for $50,000$ events. 137
5.30 The softmax activation gives a discrete probability distribution (in contrast to the continuous spectrum depicted) over the pixel positions while the Gaussian activation selects one particular pixel that corresponds to the highest probability – all other pixels are highly suppressed (distributions not to scale). 141
5.31 The ELBO and the KL-divergence (see Equation 3.41) versus the number of iterations parameterized for different numbers time steps n_T. 142

5.32 Three randomly simulated QCD jets. 143
5.33 Three randomly simulated W jets. 143
5.34 Average jet image for $50,000$ events and $40,000$ iterations. 143
5.35 Reconstructed jet mass m^{img} for $50,000$ events. 144
5.36 Reconstructed jet mass m^{img} for $50,000$ events. 145
5.37 Reconstructed jet mass m^{img} for $50,000$ events. 145
5.38 Pixel activation values on an event-on-event base for $50,000$ events. 146
5.39 n^{th} statistical moment for generated QCD and W jets. 146
5.40 Correlation between the reconstructed mass m^{img} and the transverse momentum $p_{\mathrm{T}}^{\text{img}}$ for $50,000$ events. 147
5.41 Correlation between the reconstructed 1-subjettiness τ_1 and the transverse momentum $p_{\mathrm{T}}^{\text{img}}$ for $50,000$ events. 147

6.1 Modulus of the Wasserstein loss for different network architectures. 150
6.2 The non-negative Wasserstein loss (6.2a) (earth mover's distance) and the gradient penalty term (6.2b) for different dimensionality $\dim(\mathcal{Z})$ of the latent space $\mathcal{Z}$ for the same architecture. 152
6.3 The non-negative Wasserstein loss (6.3a) and its components (6.3b) according to $\sup_{\|f_\phi\|_L \leq 1} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))]$ for different learning rates α_l with the constraint $\alpha_l := \alpha_c = \alpha_g$. . 153
6.4 The non-negative Wasserstein loss for different "penalty factors" λ_{GP} of the gradient penalty term (6.4a) and for different optimizers (weights' update rule) used in the gradient descent algorithm (6.4b). 154
6.5 Pixel activation values E^{pix} on an event-on-event base for $50,000$ events. 156
6.6 Reconstructed jet mass m^{img} for a linear (6.6a) and a logarithmic scale (6.6b) for $50,000$ events. 157
6.7 Average jet image for a linear (6.7a) and a logarithmic scale (6.7b). 158
6.8 Three randomly simulated QCD jets after $40,000$ iterations. . . . 158
6.9 Three randomly simulated W jets after $40,000$ iterations. 158
6.10 Average jet image for $50,000$ events and $40,000$ iterations. 159
6.11 Reconstructed transverse momentum $p_{\mathrm{T}}^{\text{img}}$ for $50,000$ events. . . . 160
6.12 Reconstructed jet mass m^{img} for $50,000$ events. 160
6.13 Correlation between the reconstructed average jet mass m^{img} and the transverse momentum $p_{\mathrm{T}}^{\text{img}}$ for $50,000$ events. 161
6.14 Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the transverse momentum $p_{\mathrm{T}}^{\text{img}}$ for $50,000$ events. 162
6.15 Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the mass m^{img} for $50,000$ events. 162
6.16 The Wasserstein loss (6.16a) and its components (6.16b) for QCD jets. 164
6.17 The Wasserstein loss (6.17a) and its components (6.17b) for W jets. 164
6.18 The distribution of the critic's weights (without biases $\mathcal{W} = \mathcal{P} \setminus \mathcal{B}$) for different iterations (6.18a) and the distribution of biases (6.18b). 165
6.19 The distribution of the generator's weights (without biases $\mathcal{W} = \mathcal{P} \setminus \mathcal{B}$) for different iterations (6.19a) and the distribution of biases (6.19b). 165
6.20 Three randomly simulated QCD jets after $40,000$ iterations. . . . 166
6.21 Three randomly simulated W jets after $40,000$ iterations. 166

6.22 Average jet image for $50,000$ events and $40,000$ iterations. 167
6.23 "Energy correlation" between pixel no. 312 and pixel no. 337 for the training data (6.23a) and the generated distribution (6.23b). The correlated pixels have been highlighted in the subfigure in the left corner. 168
6.24 The first nine statistical moments according to Equation 5.3 for the training data and the generated distribution of the generative model. In case of QCD, the two distributions almost perfectly coincide, which is why other instances of the neural network (besides the one corresponding to $40,000$ iterations) have been included to illustrate the progression of the model. 168
6.25 Correlation between the unique seed $\widetilde{z}$ and the reconstructed energy E^{img} for an unconditioned (6.25a) and an conditioned (6.25b) Wasserstein GAN. 170
6.26 Correlation between the reconstructed jet energy from the generated image E^{img} and the conditioning label E^{jet} for a *logarithmic* energy scale and $50,000$ events. 170
6.27 Correlation between the reconstructed jet energy from the generated image E^{img} and the conditioning label E^{jet} for a *linear* energy scale and $50,000$ events. 171
6.28 Pixel activation values E^{pix} on an event-on-event base for $50,000$ events. 171
6.29 Reconstructed transverse momentum $p_{\text{T}}^{\text{img}}/E^{\text{img}}$ for $50,000$ events. 172
6.30 Reconstructed jet mass m^{img} for $50,000$ events. 173
6.31 Reconstructed leading energy fraction f_1^{img} for $50,000$ events. . . 174
6.32 Reconstructed subleading f_2^{img} for $50,000$ events. 174
6.33 Correlation between the reconstructed average mass m^{img} and the transverse momentum $p_{\text{T}}^{\text{img}}$ for $50,000$ events. 175
6.34 Correlation between the reconstructed average mass m^{img} and the mean transverse momentum per constituent $p_{\text{T}}^{\text{img}}/N_0^{\text{pix}}$ for $50,000$ events. 175
6.35 Reconstructed 1-subjettiness τ_1^{img} for $50,000$ events. 176
6.36 Reconstructed 21-subjettiness τ_{21}^{img} for $50,000$ events. 176
6.37 Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the transverse momentum $p_{\text{T}}^{\text{img}}$ for $50,000$ events. 177
6.38 Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the mass m^{img} for $50,000$ events. 177
6.39 The Wasserstein loss (6.39a) and its components (6.39b) for QCD jets. 179
6.40 The Wasserstein loss (6.40a) and its components (6.40b) for W jets. 179
6.41 Three randomly simulated QCD jets after $40,000$ iterations. . . . 180
6.42 Three randomly simulated W jets after $40,000$ iterations. 180
6.43 Average jet image for $50,000$ events and $40,000$ iterations. 181
6.44 The first nine statistical moments according to Equation 5.3 for the training data and the generated distribution of the generative model. 181
6.45 Pixel activation values E^{pix} on an event-on-event base for $50,000$ events. 182
6.46 Reconstructed jet mass m^{img} for $50,000$ events. 182
6.47 Reconstructed leading energy fraction f_1^{img} for $50,000$ events. . . 183

6.48 Correlation between the reconstructed average mass m^{img} and the transverse momentum $p_{\text{T}}^{\text{img}}$ for $50,000$ events. 184
6.49 Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the transverse momentum $p_{\text{T}}^{\text{img}}$ for $50,000$ events. 184
6.50 Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the mass m^{jet} for $50,000$ events. 185
6.51 "Benchmark test" of the performance of several generative models on CPUs and GPUs compared to the standard method of event simulation through Monte Carlo generators for a different number of CPU cores. 187

Nomenclature

Reserved Greek letters

ϵ	Small non-negative real number.
λ	Penalty factor, e.g., λ_{GP} for Wasserstein GANs.
ϕ	Weights of the discriminator/critic (GAN) or endcoder (VAE).
θ	Weights of the generator (GAN) or decoder (VAE).

Reserved Other Symbols

$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and standard deviation σ.
$\mathbb{P}_g$, $\mathbb{P}_\theta$	Underlying probability distribution of generated data.
f_ϕ, g_θ	Two functions implemented via neural networks.
$\mathbb{P}_r$	Underlying probability distribution of the training data.
$\mathcal{U}(x_0, x_1)$	Uniform distribution between x_0 and x_1.

Reserved Roman Letters

$\mathcal{B}$	Set of trainable biases.
$\mathcal{P}$	Total set of trainable parameter $\mathcal{P} = \mathcal{W} \cup \mathcal{B}$.
$\mathcal{W}$	Set of trainable weights.
$\mathcal{D}$	Data set for training with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$.
$\mathbb{P}$	A probability distribution.
$\mathcal{X}, \mathcal{Y}$	Space (manifold) of input/output data.
$\mathcal{L}$	Meaning context-dependent: *Lagrangian density* in particle physics; *loss-, cost-* or *objective function* for neural networks..
$\boldsymbol{x}$	Input data.
$\boldsymbol{y}$	Label associated with $\boldsymbol{x}$ (see $\mathcal{D}$).
$\hat{\boldsymbol{y}}$	Prediction of neural netwok for $\boldsymbol{x}$.
$\hat{\boldsymbol{x}}$	Generated data point (e.g. a jet image).
$\boldsymbol{W}$	Weight matrix (neural network's parameters).
$\boldsymbol{z}$	Random latent space/noise vector.

Reserved Subscripts

t	Time step in RNNs.

Reserved Superscripts

img	Indicates that the respective quanity/figure of merit (e.g. E^{img}, η^{img}, τ_N^{img} etc.) has been evaluated for data simulated with the trained model.

pix	Indicates that the respective quanity/figure of merit has been evaluated for each pixel individually (e.g. E_{ij}^{pix} denotes the energy deposition in pixel $(i, j) \in \mathbb{N} \times \mathbb{N}$).
jet	Indicates that the respective quanity/figure of merit (e.g. E^{jet}, η^{jet}, τ_N^{jet} etc.) has been evaluated for the training data, i.e., the data generated with `MadGraph5_aMC@NLO` and `Pythia8.2`.

Roman and Greek letters

$\mathbb{M}$	This is a set.
$\boldsymbol{M}$	Bold, capital letters are matrices.
$\boldsymbol{m}, \boldsymbol{\mu}$	Bold, minuscle letters are vectors.
m, μ	Non-emphasized letters are scalars.
M, N, n	A non-negative integer number denoting the quantity or cardinality of something to be specified by the sub- resp. superscript.

General Comments

First of all, the nomenclature introduced above is not set in stone but is considered to be a guiding principle, which might be abandoned if the situation is opportune without the likelihood of confusion.

It is very important to clearly distinguish between *probability densities* and *probabilities*. The probability of a random variable X with particular realization x is given by $p(x) := P(X \in [x, x + \mathrm{d}x)) = \mathbb{P}(X \in [x, x + \mathrm{d}x)) = f_X(x)\,\mathrm{d}x$, whereby f_X denotes the respective probability *density* function.

Throughout this report, the two notation $\mathbb{P}_g$ and $\mathbb{P}_\theta$, which denote the probability distribution that has been learned by the generative model $\mathbb{P}_g := \mathbb{P}_\theta = g_\theta(\mathbb{P}_z)$, are used interchangeability based on phonetic and esthetic criteria.

The choice of whether an abbreviation, an acronym or the full form of a word, term or expression is used, is based on esthetic criteria with a view to smooth reading fluency. This means in particular that abbreviation and full form should be treated as such and read accordingly.

All logarithms in this report (without exception) are the *natural logarithm*, the logarithm to the base e.

Furthermore, *natural units* are used ($c = 1$ and $\hbar = 1$) through this report such that $[E] = [p] = [m] = \mathrm{GeV}$ and $[T] = \mathrm{GeV}^{-1}$ for time.

Introduction

With “big data analysis” and so-called “artificial intelligence” continuously taking on greater significance, today’s society gradually starts to experience what will likely be an enormous upheaval of their innermost structure that has virtually no equal in more recent contemporary history. Indeed, it might be reasonably to assume that almost all decisive spheres of human life will be affected in some way or another by the inexorable forthcoming *change of paradigm* – for the better or for the worse.

Machine learning is one possible approach to bring artificial intelligence into being, however, it has already been firmly established among many natural sciences as a powerful tool that allows researchers to handle the steadily increasing amount of data available. With successively growing understanding of the underlying mechanisms that govern neural networks over the past few decades, the wandering “spectre” of what was long disreputable known as the so-called “black box” (which still haunts many discussions though) has gained acceptance to a great extent.

A matter of particular interest over the past few years is the implementation of *generative models* using machine learning methods or, to be more precise, *deep (structured) learning*. The two most prominent representatives of (deep) generative models are the already well-established *autoencoders* based on *variational Bayesian inference* and the more recent generative models through *adversarial networks*; both methods differ significantly concerning their underlying mathematical principles. Especially Generative Adversarial Networks (GANs) have proven themselves capable of generating data of excellent quality and diversity, however, they are also dreaded for their notorious instabilities and ubiquitous convergence problems. Those issues are addressed by a novel generation of GANs that introduce a new metric based on the earth mover’s distance, also known as Wasserstein metric, to measure the similarity between two probability density functions.

All deep learning models have in common that they rely on rather large data volumes in order to deliver meaningful results and to correctly learn the underlying structure of the training data. Fortunately, high-energy particle physics provides a highly suitable environment for the utilization of neural networks, for which reason they have been in use for many years, e.g., for the purpose of event selection or within high-level triggers. With an abundance of recorded data available, one trend is going towards the application of generative models in the context of particle physics. This thesis is part of this recent tide and dedicated to the implementation of state-of-the-art machine learning methods in the context of event simulation/generation in high-energy particle physics through the two aforementioned approaches to generative models.

Nowadays, event generation and background estimation in particle physics mostly is a hybrid between Markov Chain Monte Carlo simulation and their adjustments to data. This particular approach has proven itself in many applications over many years, however, its precision is limited primarily by the fixed-order matrix element calculation of the hard subprocess in the perturbative expansion and the double-logarithmic approximation of the Sudakov form factors in the parton shower model. On the contrary, the employment of generative models could possibly allow for a solely data-driven approach to event simulation in high-energy particle physics that is – at least theoretically – precise to all orders in perturbation theory and, furthermore, provides an almost perfect detector simulation for the respective measuring apparatus. This idea is particularly appealing within the field of particle searches that look for rare processes and hence rely on very precise background predictions/estimations.

The methods presented within the scope of this thesis are still beyond any practical application in a "real-world analysis"; notwithstanding, it might be yet another step "towards a data-driven simulation of QCD radiation through generative models." The objective as part of this report is to examine the principle applicability and feasibility of Gaussian Variational Autoencoders and Wasserstein GANs. Moreover, both methods are confronted with the same tasks, i.e., the simulation of QCD and W initialized jets and subsequently compared with respect to their performance. As part of the study, the event generation has been factorized into matrix element and the simulation of QCD radiation with the aid of *conditioned* generative models through the marginalization of the matrix element information. Additionally, in an attempt to model the underlying sequential character of the particle shower, which is not directly present in the time-projected training data, a combination of generative models with *recurrent neural networks* through standard LSTM units was used.

With regards to the thesis' structure, the written report is subdivided into three superordinate parts consisting of six chapters in total: *first* (I), a rather superficial introduction into the foundations of the Standard Model of particle physics (1.1), perturbative QCD (1.2) and its phenomenological aspects (1.3) (chap. 1) as well as the basics of the "Monte Carlo method" and event simulation in high-energy particle physics (chap. 2); *second* (II), a methodical introduction into machine learning and (deep) neural networks (chap. 3) with an emphasis on generative models, i.e., Gaussian VAEs (3.5) and Wasserstein GANs (3.6) followed by a step-by-step explanation of preprocessing procedure applied to the training data (chap. 4); *third* (III) and finally, the presentation of the actual results that have been obtained with Gaussian VAEs (chap. 5) and Wasserstein GANs (chap. 6) as well as their foregoing mentioned variations.

Part I

Chapter 1

Theory

The Standard Model (SM) of particle physics is the fundamental theoretical model with an underlying, spontaneously broken $SU(3)_c \times SU(2)_L \times U(1)_Y$ symmetry that encapsulates the gathered knowledge of elementary particle physics in a fertile interplay between theory and experiment. It provides an incredibly precise description of the elementary particles known to mankind (see, e.g., Grunewald [2006]) and their mutual interactions that are governed by the fundamental forces occuring in nature: the *electromagnetic*, the *weak*, and the *strong force* – with the gravitational force withstanding a consistent quantum field-theoretical description and, therefore, not being incorporated into the theory so far. The tumultuous history of the Standard Model, a name that has been given in the '70s, is rich and goes back to the origin of modern particle physics at the beginning of the previous century. As the hour of birth, one could take the unification of the electromagnetic and the weak force by Sheldon Glashow, Abdus Salam, and Steven Weinberg in the '60s [Glashow, 1961, Salam, 1968, Weinberg, 1967]. Ever since, the Standard Model has been tested in numerous experiments around the world, providing an accurate description of the microscopic world of elementary particles (making it the most accurate scientific theory known to humankind to the present day). This culminated in the successful discovery of the Higgs boson at the Large Hadron Collider (LHC) near the French-Swiss border in 2012 [Aad and others, 2012]; finally, making the Standard Model a self-consistent (albeit inherently incomplete [Ellis, 2002]) theory of nature.

The purpose of this very first chapter is to serve as a general introduction into the topic area of the Standard Model of particle physics. A comprehensive introduction into the subject is, of course, beyond the scope of this work; hence, the focus lies on the fundamental aspects of the theory as well as the milestones in its history.

The first section (1.1) gives a outline of the Standard Model's history (1.1.1) as well as its elementary particle content (1.1.2). This is followed by a very brief introduction into the fundamental concepts behind Quantum Field Theory (QFT) (1.1.3). Equipped with the necessary foundations, the Lagrangian of the Standard Model along with its individual terms is introduced with a focus on the strong sector of the theory (1.1.4). The subsequent section is dedicated to the domain of perturbative Quantum Chromodynamics (pQCD) and key concepts like the running coupling constant (1.2.1) and factorization theorems in QCD (1.2.2, 1.2.4). The second section also introduces the so-called Sudakov form factors (1.2.5) that are the essential ingredient in the simulation of parton showers, which are the subject of chapter 2. This chapter closes with a careful look at some phenomenological aspects of QCD by means of the important concept of a particle jet (1.3.1, 1.3.2) and its reconstruction using different reconstruction algorithms (1.3.3, 1.3.4).

1.1 The Standard Model of particle physics

This section roughly sketches the historical development of the Standard Model with the objective to embed the theoretical concepts, which are introduced in the following section 1.2, into its relative historical context and background. Furthermore, it introduces the known elementary particles and their associated properties that are described by the Standard Model.

1.1.1 History of the Standard Model

He who encounters the state-of-the-art Standard Model of particle physics for the first time might be dazzled and overwhelmed by its complexity and the diverse (partially counter-intuitive) physical phenomena it is able to accurately describe. However, to understand and appreciate this highly advanced theory of nature, it is vital to reflect its historical development over the last decades, starting from the '30s of the previous century until today. Therefore, it is even more surprising that the history of the Standard Model is virtually never part of its introduction.

The Standard Model's history is an adventurous story full of misconceptions and embroilments, all of which represent steps on the path to higher knowledge. It's a beautiful story; but unfortunately, too long to tell. Within the context of this thesis, only an incomplete overview of the milestones in the history of the Standard Model – according to the author's personal view! – is given.

It is quite simple to set a starting point for the historical development of the Standard Model. At the beginning of the 20th century, the world was more or less *classical* and described by classical fields as in Maxwell's laws of electromagnetism [Maxwell, 1865] and Einstein's theory of gravitation [Einstein, 1916] that replaced Netwon's theory of gravity [Newton *et al.*, 1729]. Both theories at that time accurately described the two fundamental forces that are daily experienced by human beings in their macroscopic world – and one of them still does. This was the situation roughly up to the year 1930, when new discoveries and insights had changed the fundamental understanding of nature, with an enhanced understanding of Quantum Mechanics (QM) leading the charge. With the discovery of the neutron [Bothe, 1930] – which was initially mistakenly assumed to be γ-radiation –, the postulation of weak interaction by Fermi to solve the puzzle of the continuous energy spectrum of the electron emitted in the beta-decay in his revolutionary essay "[t]entativo di una teoria dellaemissione di raggi β" (Fermi [1933]) ("tentative theory of beta-decay") as well as the invention of Quantum Electrodynamics (QED) [Kramers, 1938] by quantizing Maxwell's equations, the foundations of the Standard Model were laid. With the procedure of renormalization – invented in 1947 –, QED agreed with astonishing accuracy with the experiments (*cf.*, e.g., measurement of the anomalous magnetic dipole moment) and quickly became the most well-tested theory in physics. After World War II, when many scientists ended their rendezvous with the military and returned to their actual studies, the number of known particles significantly increased, creating a zoo of hundreds of apparently elementary particles; it was necessary to make sense out of this mess. This ordering was partially achieved by the observation of apparent similarities between the different particles (e.g. mass, spin, electric charge, etc.) and the discoveries of (approximate) symmetries such as the famous "eight-fold way" [Gell-Mann, 1961] to name only one out of many. It was a major crisis, when it became clear that most symmetries only represent an approximation. The existence of numerous approximate symmetries confronted physicist with a daunting problem of interpretation.

A brilliant idea, which was independently developed by Murray Gell-Mann and Georg Zweig in 1961 [Gell-Mann, 1964, Zweig, 1964], gradually lifted the curtain of confusion: the so-called *parton* model pictures hadrons as a collection of point-like, i.e., elementary particles, which today are associated with the elementary quarks and gluons. It revealed an underlying structure of known (composite) particles and reduced those to a small number of elementary constituents of *material being* – the actual *atomos*. The parton model played a similar role in the development of modern particles physics like Pauli's exclusion principle did for chemistry in explaining the ordering of the elements in the periodic table.

Alongside the aforementioned parton model, the systematic development of gauge symmetry (also known as local gauge symmetry) was yet another important step in the evolution of the Standard Model. Although classical Electrodynamics may also be regarded as a gauge theory that is based on the $U(1)_Q$ symmetry group. The first methodical contribution in this matter came from Chen Ning Yang (楊振寧/杨振宁) and Robert Laurence Mills [Yang and Mills, 1954] who constructed a theory that was based on the non-Abelian group $SU(2)$ with non-commuting "charges". The objective was to build a theory of the strong interaction; however, it was later applied to weak interactions that were already known to be a mixture of vector and axial-vector interactions.

The '50s, as well as the '60s, were eventful times full of revolutionary ideas and discoveries. Another notable one was the concept of intermediate or mediator particles, respectively, vector bosons to be more precise. This idea was independently developed by several scientists like Julian Seymour Schwinger [Schwinger, 1957], Sheldon Lee Glashow [Glashow, 1961] and in joint work between Abdus Salam (عبد السلام) and Ward John Clive [Salam and Ward, 1964]. It had a tremendous impact on our understanding and interpretation of the fundamental nature of forces in nature as being transmitted by vector (and possibly tensor) bosons in contrast to the rather "vague" concept of (classical) potentials.

The Standard Model at that time was already able to describe a large range of phenomena; especially QED had proven as most accurate theory in physics, with the theoretical predicted electromagnetic fine-structure constant agreeing with the measured one within *ten-parts-per-billion* (1ppb= 10^{-9}) [Hanneke *et al.*, 2008]. But despite these resounding successes and the improved understanding of the subatomic world, the Standard Model (at that time) had an apparent flaw: the requirement of local gauge invariance culminated in a theory with solely massless particles – which is in clear contradiction to the obvious massive particles measured in the experiments. This situation was a highly unsatisfactory since it caused inconsistencies in the Standard Model between predictions and observations – the merciless guillotine in physics that without hesitation beheads even the most elegant theories if they disagree with observations. Full of despair, physicist had to put in masses by hand; but, in doing so they accepted reduced predictability and predictive power of the theory and, additionally, made the theory non-renormalizable! A "novel" idea finally brought the long-awaited breakthrough with the desired way out of the dilemma: the concept of Spontaneous Symmetry Breaking (SSB), which is the final part of this summary.

The path to SSB was paved with obstacles. The problem was that it was proven by Goldstone, Salam and Weinberg that for each exact symmetry that is spontaneously broken there must be a massless and spinless (scalar- resp. pseudoscalar) particle. At this time, a large number of approximate symmetries have been known, e.g., isospin symmetry or the aforementioned eight-fold way. But, besides the photon, no massless particle had been discovered (the gluon was yet unknown). In the end, one problem was replaced by another.

At that time, Peter Ware Higgs [Higgs, 1964] arrived on stage. He (and others) tried to find a way out of the Goldstone theorem and its ominous mass- and spinless particles. He realized that the Goldstone theorem does not apply for local gauge symmetries, which are spontaneously broken. In this case, the Goldstone bosons do not manifest as real particles but remain in the theory and turn into the helicity-zero component of the gauge bosons (see Higgs-Kibble-Dinner). This not only gets rid of the non-observed Goldstone particles but simultaneously results in a mass for the gauge bosons and hence creates a theory of massive mediators (Higgs' idea was contemporaneously also discovered by the collaboration between Englert and Brout [Englert and Brout, 1964] and Guralnik, Hagen and Kibble [Guralnik *et al.*, 1964] – therefore also the somewhat cumbersome name Englert-Brout-Higgs-Guralnik-Hagen-Kibble mechanism, usually just called Higgs mechanism). With the spontaneous breaking of the symmetry $SU(2)_L \times U(1)_Y$, the Higgs mechanism could also explain the masses of the fundamental fermions in the Standard Model; finally making it a consistent and (even more) predictive theory. Besides explaining the masses of massive vector bosons and fermions, the Higgs mechanism also predicted the presence of a new particle that is the quantum excitation of the respective Higgs field. This particle was predicted to be massive; hence, it could have escaped detection due to limited energies in particle colliding experiments at that time (e.g. at LEP at CERN). This was indeed the case. It took more than 50 years to finally discover the Higgs boson in 2012. A great example of human willpower.

Besides the discovery of the Higgs, the Standard Model has proven its validity and predictive power in a variety of other experiments. For example, the Standard Model predicted the existence of the $W^{\pm}$ and Z bosons in electroweak interactions, with the first observations of neutral currents in 1973 at the Gargamelle bubble chamber at CERN [Hasert and others, 1973, Hasert *et al.*, 1973] and the discovery of the actual particles back in 1983 that have been observed in 1986 for the very first time at the Super Proton Synchrotron also at CERN [Watkins, 1986]. Or, the discovery of the gluon in 1979 with the PLUTO and TASSO experiment at DESY [Barber and others, 1979, Berger and others, 1979, Brandelik and others, 1979]. And, not to forget, the several quark flavours predicted by the Standard Model all that have been observed later in the experiment, like the *charm* and *top* quark [Archambault *et al.*, 2003, Campagnari and Franklin, 1997].

The Higgs mechanism finishes the journey through the history of the Standard Model. Of course, there would be far more to tell. However, it would be a mistake to think that these concepts have matured in an "ivory tower". On the contrary, the evolution of the Standard Model is a perfect example of how the interaction and mutual pollination of theoretical and experimental physics leads to a path of higher knowledge and a better understanding of the world around us. [Weinberg, 2004].

1.1.2 Elementary particle content

As a quantum field theory, all the fundamental particles in the Standard Model are associated with a certain quantum field, more precisely, the quantum excitations that manifest as fundamental particles of the respective field. Roughly speaking, the elementary particles in the Standard Model can be subdivided into fermions (half-integer spin particles that obey the Fermi-Dirac statistic), vector (tensor) bosons (integer spin particles that mediate the fundamental forces), and one pseudoscalar particle, the Higgs, whose field plays an important role in the essential Higgs mechanism and the Yukawa interaction. The known fundamental fermions in the Standard Model are summarized in Table 1.1.

	Flavour	Mass	Q_{el}	T_3^L	Discovery
Leptons	ν_e	$< 2.2\,\mathrm{eV}$	0	$+\frac{1}{2}$	1956
	e	$0.511 \pm 10^{-8}\,\mathrm{MeV}$	-1	$-\frac{1}{2}$	1897
	ν_μ	$< 0.19\,\mathrm{MeV}$	0	$+\frac{1}{2}$	1962
	μ	$105.7 \pm 4 \times 10^{-6}\,\mathrm{MeV}$	-1	$-\frac{1}{2}$	1936
	ν_τ	$< 18.2\,\mathrm{MeV}$	0	$+\frac{1}{2}$	2000
	τ	$1776.86 \pm 0.12\,\mathrm{MeV}$	-1	$-\frac{1}{2}$	1975
Quarks	u	$2.3 \pm 0.7\,\mathrm{MeV}$	$+\frac{2}{3}$	$+\frac{1}{2}$	1968
	d	$4.8 \pm 0.5\,\mathrm{MeV}$	$-\frac{1}{3}$	$-\frac{1}{2}$	1968
	c	$1.275 \pm 0.025\,\mathrm{GeV}$	$+\frac{2}{3}$	$+\frac{1}{2}$	1974
	s	$95 \pm 5\,\mathrm{MeV}$	$-\frac{1}{3}$	$-\frac{1}{2}$	1968
	t	$173.2 \pm 0.9\,\mathrm{GeV}$	$+\frac{2}{3}$	$+\frac{1}{2}$	1995
	b	$4.18 \pm 0.03\,\mathrm{GeV}$	$-\frac{1}{3}$	$-\frac{1}{2}$	1977

Table 1.1: Fermions (subdivided into leptons and quarks) in the Standard Model (masses have been rounded) [Eidelman *et al.*].

According to Table 1.1, the fermions in the Standard Model may be further subdivided into *leptons* and *quarks*. A lepton is an elementary spin-1/2 particle that does not experience the strong force. The known elementary leptons are the electron, the muon, the tauon, and the corresponding electrically neutral neutrino flavours with their respective antiparticles. The second category of fermions are the quarks that differ from leptons insofar as they carry a fractional electric charge as well as *color charge*, which is the subject of quantum chromodynamics. Usually, fermions are associated with matter, with quarks being the key building blocks of composite particles (hadrons) such as protons and neutrons. With leptons and quarks, the Standard Model knows twelve different fermion flavours and 24 fermions in total – including the corresponding antiparticles. Furthermore, a distinction is made between fermions and bosons. Bosons (the name refers to the Indian physicist Satyendra Nath Bose (সত্যেন্দ্রনাথ বসু)) are described by the Bose-Einstein statistics. In the Standard Model, the elementary *vector bosons* ($s = 1$) are responsible for the mediation of the fundamental forces of nature. These elementary vector bosons are the photon γ, the gluon(s) g, as well as the neutral Z and the charged $W^\pm$ bosons.

	Boson	Interaction	Mass	Q_{el}
Scalar	higgs H_0	none	$125.18 \pm 0.16\,\mathrm{GeV}$	0
Vector	photon γ	QED	$< 10^{16}\,\mathrm{eV}$	0
	gluons g	QCD	0	0
	Z	EW	$91.1876 \pm 0.0021\,\mathrm{GeV}$	0
	$W^\pm$		$80.379 \pm 0.012\,\mathrm{GeV}$	± 1
Tensor	graviton[1]G	gravity	$< 6 \cdot 10^{-32}\,\mathrm{eV}$	0

Table 1.2: Bosons in the Standard Model [Eidelman *et al.*].

It should be mentioned that in a hypothetical quantum field theory of gravity there would be an elementary tensor boson with spin two, the graviton, that is the force-carrier of the gravitational force. However, to be a consistent theory, the Standard Model requires the existence of (at least) one scalar particle ($s = 0$) that is associated with the mechanism that generates the mass of the fermions and bosons. This is the aforementioned Higgs boson. The bosons in the Standard Model are summarized in the Table 1.2.

The appearance of gauge bosons in the theory is a consequence of the required local gauge symmetry of the Standard Model Lagrangian. This concept is important in order to understand how elementary particles interact with each other through the exchange of force-carrying particles, which are quantum fields by themselves.

1.1.3 The Standard Model as a QFT

The underlying mathematical-physical framework – to put in a very simplified manner: the language with its grammatical regularities – of the Standard Model of particle physics is Quantum Field Theory (QFT). It is therefore only appropriate to spend a moment on this subject to convey a vestigial idea of the underlying principles behind one of the most prominent physical theories of all time. With the Standard Model being a QFT it is (by construction) simultaneously consistent with Quantum Mechanics (QM) as well as the theory of Special Relativity (SR) [Einstein, 1905]. While in non-relativistic QM the number of particles is conserved (due to the structure of the underlying Hilbert space $\mathcal{H} = \otimes_{i=1}^{n} \mathcal{H}_i$ that prohibits the appearance of non-particle-number-conserving terms in the Hamiltonian), in QFT However, the number of particles is not fixed, which allows for the creation and annihilation of particles. Furthermore, in QFT the central *quantum field* is an operator (*cf.* canonical quantization) in Fock space while the *particle* (this may also be a quasi-particle like the phonon in condensed matter physics) is considered to represent a state of the respective field.

The central object of interest in QFT is the quantized field $\phi(\boldsymbol{x})$ and its dynamics that is encoded in the Lagrangian $\mathcal{L}$ (which can be used to derive the equations of motion), e.g. the Standard Model $\mathcal{L}_{\mathrm{SM}}$, that results in predictable and physical *measurable* quantities such as, e.g., the scattering amplitude. In general, observables are derived from the so-called (n-point, time-ordered) correlation function

$$\left\langle 0 \middle| \prod_{i=1}^{n} \mathbf{T}\hat{\phi}(\boldsymbol{x}_i) \middle| 0 \right\rangle = \frac{\int \mathcal{D}\phi\, e^{-S[\phi]} \prod_{i=1}^{n} \phi(\boldsymbol{x}_i)}{\int \mathcal{D}\phi\, e^{-S[\hat{\phi}]}}, \tag{1.1}$$

which gives the amplitude for a field configuration (a particle of a certain kind) to propagate from one point in space-time $\boldsymbol{x}$ to another (two-point correlation function). Equation 1.1, given in the path integral formulation, can be written as a perturbative expansion in several orders. Richard Feynman introduced a pictorial representation, which later was named after him, of the individual terms in the expansion that correspond to higher-order corrections to the Born approximation, i.e., lowest-order approximation [Feynman, 1949].

In particle physics, each type of particle is associated with a certain quantum field[2] that describes and encodes its properties, e.g. mass, spin etc., in the Lagrangian of the free fields.

[1]Hypothetical elementary particle that mediates the force of gravity.

[2]Although the discipline of QFT has its historical origins the study of particles and their interactions, it is not limited to this field of research. Over the years, the fundamentals QFT has been successfully adapted to other areas in physic with a positive retroactive effect on particle physics (*cf.*, e.g., SSB and renormalization in condensed matter physics).

For instance, the Lagrangian $\mathcal{L}$ of the Dirac or fermion field, which describes a spin-$1/2$ particle such as leptons or quarks (see Table 1.1), is given by $\mathcal{L} = \bar{\psi}(i\gamma^\mu \partial_\mu - m)\psi$ with the resulting equation of motion $(i\gamma^\mu \partial_\mu - m)\psi = 0$, whereby ψ is the associated quantum field of a particle with mass m and half-integer spin. In QFT the fields are interpreted as operators; hence, in analogy to QM one imposes certain *commutator* resp. *anticommutator* relations on the fields. In case of fermionic particles their fields have to obey the anticommutation relation $\{\psi_a(\boldsymbol{x}), \psi_b(\boldsymbol{y})\} = \delta^{(3)}(\boldsymbol{x} - \boldsymbol{y})\delta_a$ that imply the Fermi-Dirac statistic and impose Pauli's exclusion principle. On the other hand, if the fields A_μ obey the Bose-Einstein statistic they satisfy the canonical commutator relation $[A_a(\boldsymbol{x}), A_b(\boldsymbol{y})] = \delta^{(3)}(\boldsymbol{x} - \boldsymbol{y})\delta_a$ and therefore describes quanta that exhibit an integer spin. In the Standard Model, the boson fields are the interacting particles that mediate the respective fundamental force – therefore the term "force-carrier particles". These mediator particles naturally emerge in quantum field theories in which the respective Lagrangian is invariant under local *gauge transformations*, hence also the name *gauge particles*, that are covered in the next section of this chapter. Last but not least, the simplest class of fields are the so-called scalar fields ϕ that are invariant under any Lorentz transformation and do not involve polarization effects. In its most simple form, the Lagrangian of a complex scalar field $\phi \in \mathbb{C}$ with $\phi := \phi_1 + i\phi_2$ with $\phi_1, \phi_2 \in \mathbb{R}$ is given by $\mathcal{L} = (\partial_\mu \phi^*)(\partial^\mu \phi) - m^2 \phi^* \phi$ with the equations of motions given by $(\Box + m^2)\phi^{(*)} = 0$ (in QFT each component of all quantum fields additionally must satisfy the free Klein-Gordon equation due to energy-momentum conservation). The only (complex) scalar field in the Standard Model that has been observed in nature so far is the pseudoscalar Higgs boson. This pseudoscalar particle is the essential component of the Higgs mechanism and the concept of SSB.

1.1.4 The QCD Lagrangian

The Standard Model of particle physics is a gauge-invariant quantum field theory that is based on the (spontaneously broken) unitary, non-Abelian symmetry group[3]

$$G_{\mathrm{SM}} := SU(3)_c \times SU(2)_L \times U(1)_Y, \tag{1.2}$$

whereby each of the compact (Lie) subgroups $SU(3)_c$, $SU(2)_L$ and $U(1)_Y$ introduces associated *gauge fields* that defines the nature of the interactions allowed in the theory. The gauge fields in the Standard Model ensures the underlying Lagrangian to be invariant under local gauge transformations concerning the group G_{SM}. The requirement of a local gauge symmetry is non-trivial and comes with a long history. First, it was introduced in analogy to QED and has then later been adopted to other theories. However, today there exists a deeper understanding of gauge transformations as the connection between equivalent coordinate bases for the same mathematical object (i.e. the field).

With the underlying group structure of the Standard Model introduced, it is about time to present the actual Lagrangian of the theory that encodes the dynamic of the fundamental quantum fields. The Lagrangian (density) $\mathcal{L}_{\mathrm{SM}}$ of the Standard Model (reduced and in simplified notation) in a legible form is given by:

$$\mathcal{L}_{\mathrm{SM}} = -\frac{1}{4}\mathrm{tr}\left[B_{\mu\nu}B^{\mu\nu}\right] - \frac{1}{4}\mathrm{tr}\left[W_{\mu\nu}W^{\mu\nu}\right] \tag{1.3}$$

$$+ \bar{\Psi}_q \gamma^\mu D_\mu \Psi_q - -\frac{1}{4}\mathrm{tr}\left[G_{\mu\nu}G^{\mu\nu}\right], \tag{1.4}$$

[3]The actual gauge group of the Standard Model is the reduced $SU(3)_c \times SU(2)_L \times U(1)_Y \,/\, \mathbb{Z}_6$ symmetry group [Bakker *et al.*, 2004]

$$+ (D_\mu \Phi)^\dagger (D^\mu \Phi) + \mu^2 \Phi^\dagger \Phi - \frac{1}{2}\lambda(\Phi^\dagger \Phi)^2 \tag{1.5}$$

$$+ \bar{\Psi}_L \gamma^\mu D_\mu \Psi_L + \frac{1}{2}\Psi_L^T C \Phi H \Psi_L + h.c. \tag{1.6}$$

Phenomena surrounding the strong force are described by Equation 1.4 in the expression above.

The strong force was (first phenomenologically) introduced in the 1930s to explain the binding of the constituents that form the nuclei. Today, quantum chromodynamics (from Greek χρῶμα, "colour") is the theory of the strong interaction with the underlying non-Abelian symmetry group $SU(3)_c$, whereby the subscript c refers to the so-called *color charge*, which remains unbroken in the Standard Model. In the Lagrangian of the Standard Model 1.6, QCD manifests itself by the term

$$\mathcal{L}_{\text{QCD}} = \sum_q \bar{\Psi}_{q,a}(\gamma^\mu \partial_\mu \delta_{ab} - i g_3 \gamma^\mu G_\mu^\alpha t_\alpha)\Psi_{q,b} - \sum_q \bar{\Psi}_{q,a}\Psi_{q,b} m_q \delta_{ab} - \frac{1}{4}\text{tr}\left[G_{\mu\nu}G^{\mu\nu}\right], \tag{1.7}$$

with t_α bein related to the Hermitian and traceless Gell-Mann matrices (which are generators of the $SU(3)_c$ group) and $\alpha \in \{1, ..., 8\}$ that correspond to eight gluons with different linear independent colour charge combinations. The field strength tensor $G_{\mu\nu}^a$ is given by $G_{\mu\nu}^a = \partial_\mu G_\nu^a - \partial_\nu G_\mu^a - g_3 f_{abc} G_\mu^b G_\nu^c$, which describes the dynamic of the gluon fields.

The quarks acquire their mass via SSB of the subgroup $SU(2)_L \times U(1)_Y \to U(1)_Q$ that also acts on the quark doublets Ψ_q. The gluons, which are considered to be in the adjoint representation of the gauge group and an octet under $SU(3)_c$, remain massless though (which agrees with experimental observations) since the symmetry group $SU(3)_c$ is unbroken in the Standard Model.

Quantum chromodynamics is a non-Abelian gauge theory, i.e., the generators t_α of the underlying symmetry group $SU(2)_c$ do not commute with each other $[t^a, t^b] = i f_{abc} t^c$. This property causes all kind of beautiful effects and makes QCD a rich theory. A well-known phenomenon in quantum chromodynamics, as it is also known for the weak interaction, is the so-called *self coupling* of the gauge bosons (see gluon self-interaction in Figure 1.1).

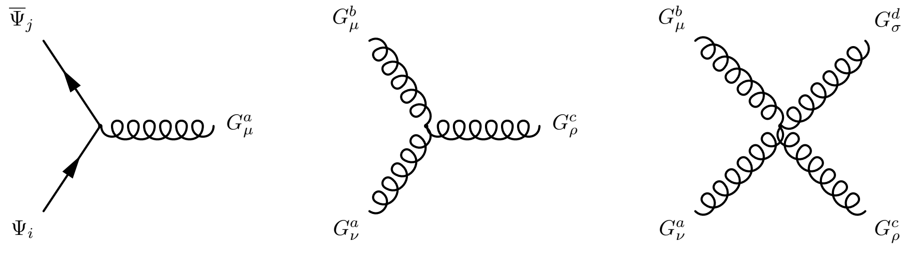


Fig. 1.1: The basic building blocks of QCD Feynman diagrams: (*left*) quark-gluon vertex (*middle*) three-gluon vertex (*right*) four-gluon vertex.

Contrary to QED, the mediator bosons in QCD carry a (color) charge by themselves, which induces self coupling. These terms can easily be identified by expanding and rearranging the kinetic term of the gluon field strength tensor $\mathcal{L}_{\text{QCD}} \supset \frac{1}{4}\text{tr}\left[G_{\mu\nu}G^{\mu\nu}\right]$.

The self-coupling of the gluon fields has further consequences with regards to the *effective charge* or *coupling* of the theory that is related to the β-function, $\beta(g) = \mu\frac{dg}{d\mu}$ of the theory, which encodes the dependency of the "coupling constant" on the energy scale μ. In case of the non-Abelian gauge theory QCD, the situation is different because the

β-function is negative (antiscreening) [Foundation, 2004, Gross and Wilczek, 1973], which means that the effective strong coupling becomes small a short distances leading to the concept of *running coupling.* This is a characteristic feature of the strong force which leads to *asymptotic freedom*, which will be further discussed in Section 1.2.2.

Asymptotic freedom in QCD results in small couplings for sufficient large energy scales that allows to consider quarks and gluons quasi-free particles. In this limit, perturbation theory is appropriate to compute the matrix element of the respective process down to a characteristic *energy scale* Λ_{QCD} where non-perturbative dynamics dominates. For small energies however, the coupling becomes large and perturbation theory is no longer applicable; therefore, predictions in this domain mostly relies on phenomenological model such as, e.g., the Lund string model [Andersson *et al.*, 1983a]. Thus, the name *non-perturbative* regime. This phenomenon where the underlying partons can not be isolated from the hadrons anymore is known as *confinement.*

1.2 Perturbative QCD

As already stated above, the Standard Model of particle physics is the most accurate fundamental theory of the smallest building blocks of matter and their interaction by the fundamental forces of nature except gravity. The previous section provided a brief outline of the Standard Model's history and introduced some most basic concepts as well as its particle content. This section directly joins the previous one and serves as a basic introduction into the theoretical framework of perturbative QCD. Furthermore, the very important phenomenon of the running coupling constant and its physical and practical implications are discussed in more detail. This section also lays the foundation for the following chapter 2 by introducing the concept of a parton shower and its implementation based on the Sudakov form factors.

1.2.1 Renormalization and running coupling

Historically – soon after its initial success – QFT faced a serious problem: the naïve computation of higher-order terms in the perturbative expansion, such as loop diagrams in the correlation functions, results in divergent integrals over the particle's (intermediate) momentum. As a consequence, most of the terms in the perturbative expansion are indeed infinite, contradicting the finite observables measured in the experiments. It was a long-lasting development process until this peculiar phenomenon and its physical meaning were finally understood. Important contributions to a better understanding of renormalization in particular came from Sin-Itiro Tomonaga (朝永 振一郎), Julian Schwinger and Richard Feynman, who all were awarded the Nobel Price in 1965 "for their fundamental work in quantum electrodynamics, with deep-ploughing consequences for the physics of elementary particles[.]" (Foundation [1965]).

In a nutshell, the renormalization process aims to remove divergencies in the computations of the physical observables. It usually starts with the *regularization scheme* by introducing an additional (non-physical) parameter μ that allows the isolation of the part that gives rise to divergencies. This parameter may be a cutoff (*cf.* cutoff regularization) or a modification of the dimensionality of the respective integral (*cf.* dimensional regularization). The divergencies are removed ("discarded") by the redefinition, which is the actual *renormalization step*, of the parameters, e.g. the mass, the fields or the charge, in the Lagrangian. This is an extraordinary step that requires reflection since it changes our perspective on the actual parameters occurring in the Lagrangian 1.3–1.6. Due to the regularization procedure, the physical quantities are finite without any divergencies; however, they are now a function of the artificially introduced regularization parameter μ. The procedure described above, consisting of regularization and renormalization, allows to obtain finite (renormalized) states if the theory (like, e.g., QCD) is inherently renormalizable. Unfortunately, there is a certain degree of arbitrariness attributed to this technique, since the individual regularization and renormalization steps are not unique. As a consequence, the resulting QFT depends on the respective renormalization scheme. Hence, regularization and renormalization result in a *family* of QFTs that depend on the specific choice of the scheme and the regularization parameter. To obtain a consistent procedure, the resulting quantum field theory must be independent of those particular choices. This requirement induces the so-called *renormalization group equations* (*cf. Callan-Symanzik equations* in QED) that ensures scale invariance of the physical observables if the associated beta-function vanishes. If the renormalization group equations are satisfied, the different regularization and renormalization schemes are guaranteed to result in an equivalence class of QFTs.

Based on the renormalization group equation, the regularization parameter μ is absorbed into the coupling of the theory, giving rise to the famous *running coupling*, i.e., the functional dependence of α_S on μ, which is the subject of the following section.

1.2.2 Asymptotic freedom and colour confinement

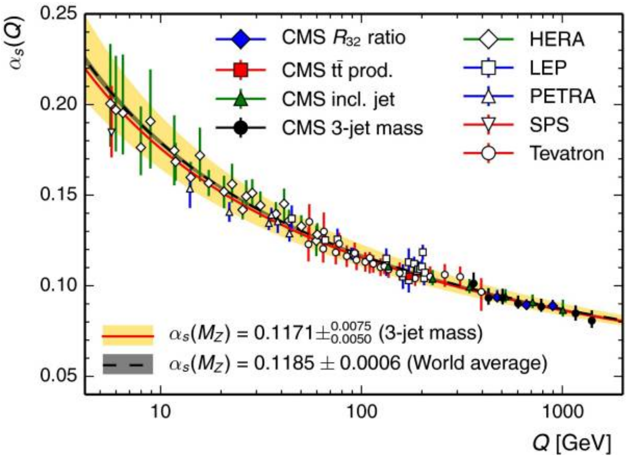
The renormalization procedure roughly outlined in Section 1.2.1 results in a dependence of the coupling α_S on the regularization parameter μ that is governed by the underlying renormalization group equation(s). Of particular interest is the aforementioned *beta-function* of QCD

$$\beta(\alpha_S) := 4\pi\mu^2 \frac{\partial \alpha_S}{\partial \mu^2} = 4\pi \sum_{k=0}^{\infty} \beta_k \left(\frac{\alpha_S}{4\pi}\right)^{k+2} \approx -\frac{\alpha_S^2}{4\pi}\left(\frac{11}{3}N_c - \frac{2}{3}N_f\right), \tag{1.8}$$

which encodes the information of how the "coupling constant" runs with the energy scale μ. The approximation of Equation 1.8 to $\mathcal{O}(\alpha_S^2)$, with $\beta_0 = \frac{2}{3}N_f - \frac{11}{3}N_c$, is the so-called *one*-loop beta-function (one loop in the perturbative expansion), whereby N_c denotes the number of colors and N_f the number of quark flavours in the theory. For three colours $N_c = 3$ and $N_f = 6$ quark flavors – as observed in nature –, the beta-function is negative $\beta_0 < 0$, which is a characteristic feature of QCD contrary to QED where β_0 is strictly positive. Solving the differential equation 1.8 for the integral boundaries $[Q, \mu]$ within the one-loop approximation and rewriting the resulting expression in terms of the energy scale $\Lambda_{\mathrm{QCD}} := \mu^2 \exp \frac{4\pi}{\beta_0 \alpha_S(\mu^2)}$ gives

$$\alpha_S(Q^2) = -\frac{1}{\frac{\beta_0}{4\pi}\log\left(\frac{Q^2}{\Lambda_{\mathrm{QCD}}^2}\right)}. \tag{1.9}$$

Due to $\beta_0 < 0$, the coupling α decreases with the energy scale Q^2. This phenomenon is known as *asymptotic freedom* since the coupling vanishes in the limit of very large energies $\lim_{Q^2\to\infty} \alpha_S(Q^2) = 0$; therefore, QCD appears to be a *free* theory in the ultraviolet limit (interestingly, according to Equation 1.8, theories based on the $SU(3)$ are only asymptotically free if $N_f < \frac{11}{2}N_c$). Figure 1.1 shows several measurements of the strong coupling constant α_S for different energy scales Q^2 at various experiments that nicely illustrate the aforementioned mechanism. The current world average value of α_S evaluated at the mass of the Z boson $Q^2 = M_Z$ is given by $\alpha_S(Q^2 = M_Z^2) = 0.1172 \pm 0.0059$ [Eidelman *et al.*].



Plot 1.1: Overview of several measurements of the running (strong) coupling $\alpha_s(Q)$ (adapted from Khachatryan and others, 2015, Fig. 7, p. 12).

The characteristic scale Λ_{QCD} is, by construction, invariant under the renormalization group and corresponds roughly to the energy scale at which $\alpha_S(\Lambda_{\mathrm{QCD}}) \sim \mathcal{O}(1)$ where perturbation theory is no longer applicable. Hence, Λ_{QCD}, which is in the order of hadron masses, determines the energetic boundary between the perturbative and the non-perturbative regime in QCD. So, perturbative QCD is able to provide meaningful results if $E \gg \Lambda_{\mathrm{QCD}}$.

Another characteristic property of QCD besides asymptotic freedom is the so-called *colour confinement* that accounts for the experimental evidence that quarks only exist within bound states that are a colour-singlet state under $SU(3)_c$ transformations. This is the reason why particles with fractional electric charge have never been *directly* observed in any experiment to the present day. Colour confinement is phenomenologically well-established and unquestioned; however, it still is remains a hypothesis in QCD because until now there is no general mathematical proof of this property for non-Abelian gauge theory based on first principles in QFT[4]. Due to the lack of precise theoretical description, colour confinement needs to be approximately described by a potential $V(r)$ (which can actualy be computed in lattice QCD (see, e.g., [Bornyakov and others, 2003])) between two quarks in a colour-singlet state

$$V(r) = -\frac{4}{3}\frac{\alpha_S}{r} + kr, \tag{1.10}$$

with $V(r) \to \infty$ for $r \to \infty$ (confinement) and $V(r) \to -\frac{4}{3}\frac{\alpha_S}{r}$ for large energies (asymptotic freedom). This potential is extensively used in phenomenological hadronization models such as, e.g., the already mentioned Lund string model [Andersson, 1986].

[4]The proof of confinement is equivalent to show that the quantum Yang-Mills theory exists and has a mass gap [Jaffe and Witten, 2000], which is one of the Millenium Problems advertised by the Clay Mathematics Institute and awarded with US$1 million for a righteous solution to the problem.

1.2.3 Soft and collinear limits of QCD

The previous section introduced the effect of asymptotic freedom at short distances (large energies) and color confinement in the infrared regime (low energies), both of which are a characteristic feature of QCD that follows from the dependence of the strong coupling $\alpha_S(Q^2)$ on the momentum transfer Q^2, i.e., the running coupling which in turn follows from the self-coupling of the gluon fields. Perturbative QCD takes advantage of this property by performing a perturbation expansion of, e.g., the cross-section around the strong coupling if the characteristic momentum transfer Q of the process under consideration is in the appropriate domain $Q \gg \Lambda_{\mathrm{QCD}}$. The term *perturbation expansion* refers to a power series that divides a problem, which can not be solved exactly, into several (usually infinitely many) subproblems that in turn can be solved analytically. The quality of the approximation is determined by the number of terms that are included in the power expansion. Formally, the perturbative expansion of the cross-section of a given process $i \to f_n$ with n particles in the $f$inal state is given by

$$\mathrm{d}\sigma^{if_n} = \sum_{k=0}^{\infty} \alpha_S^{k+n} \mathrm{d}\widetilde{\sigma}_k^{if_n}, \tag{1.11}$$

whereby $\widetilde{\sigma}_k^{if_n} := \sigma_k^{if_n}/\alpha_S^{k+n}$ denotes kth expansion of σ^{if_n} factorized for the coupling α_S. For the method to be valide and give meaningful results, the expansion coefficient, i.e, the coupling must be sufficiently small ($\alpha_S < 1$) such that the contributing corrections to the matrix element decrease for higher-orders in α_S. In practice, however, the perturbative series 1.11 has to be terminated prematurely.

In particle physics, the first term of the expansion $\mathrm{d}\widetilde{\sigma}_0^{if_n}$ is called *Leading-Order* (LO) (also known as Born level) cross-section. All of the following terms are named with respect to the LO term, for instance, $d\widetilde{\sigma}_1^{if_n}$ is referred to as the *Next-to-Leading-Order* (NLO) cross-section.

$$\mathrm{d}\sigma^{if_n} = \alpha^n \underbrace{\mathrm{d}\widetilde{\sigma}_0^{if_n}}_{\mathrm{LO}} + \alpha^{n+1} \underbrace{\mathrm{d}\widetilde{\sigma}_1^{if_n}}_{\mathrm{NLO}} + \alpha^{n+2} \underbrace{\mathrm{d}\widetilde{\sigma}_2^{if_n}}_{\mathrm{NNLO}} + \sum_{k=3}^{\infty} \alpha^{n+k} \mathrm{d}\widetilde{\sigma}_k^{if_n}. \tag{1.12}$$

The individual terms in the expansion 1.12 can be computed by means of *Feynman rules* and *Feynman diagrams.*

Feynman diagrams are a graphical representation of the individual terms in the S-matrix expansion. Along with the Feynman rules, which are derived from the Lagrangian $\mathcal{L}$ of the underlying theory, the diagrams can be used to derive mathematical expressions for the matrix element of a given process under consideration.

It is instructive to study some general properties of the matrix element in QCD in the soft and collinear limit. The cross-section, which is proportional to the squared amplitude, of the process $i \to f_n$ can be computed according to

$$\begin{aligned} \mathrm{d}\sigma^{if_n} &= \frac{(2\pi)^4}{4E_f E_i J_\alpha} \frac{1}{n!} \prod_{j=1}^{n} \int \frac{\mathrm{d}^3 k_n}{(2\pi)^3} \frac{1}{2E_j} \delta^{(4)} \left(\boldsymbol{p}_f + \boldsymbol{p}_i + \sum_i k_l \right) \left| \mathcal{M}^{if_n} \right|^2 && (1.13) \\ &= \mathrm{d}\Phi_n \left| \mathcal{M}^{if_n} \right|^2 , && (1.14) \end{aligned}$$

with n partons in the final state. Equation 1.14 comprised the kinematic information of the available differential phase space $d\Phi_n$ as well as the information of the underlying physical process that is encoded in the matrix element $\mathcal{M}^{if_n}$. If the n^{th} particle is a collinear and soft gluon – i.e. radiated Bremsstrahlung – the cross-section 1.14 can be simplified (in anticipation of the factorization theorem to be introduced in Section 1.2.5) according to

$$\lim_{\substack{\theta\to 0\\ E_n\to 0}} \mathrm{d}\Phi_n \left|\mathcal{M}^{if_n}\right|^2 = \mathrm{d}\Phi_{n-1}\left|\mathcal{M}^{if_{n-1}}\right|^2 \frac{\alpha_S C_i}{\pi}\frac{\mathrm{d}\theta^2}{\theta^2}\frac{\mathrm{d}E_n}{E_n}. \tag{1.15}$$

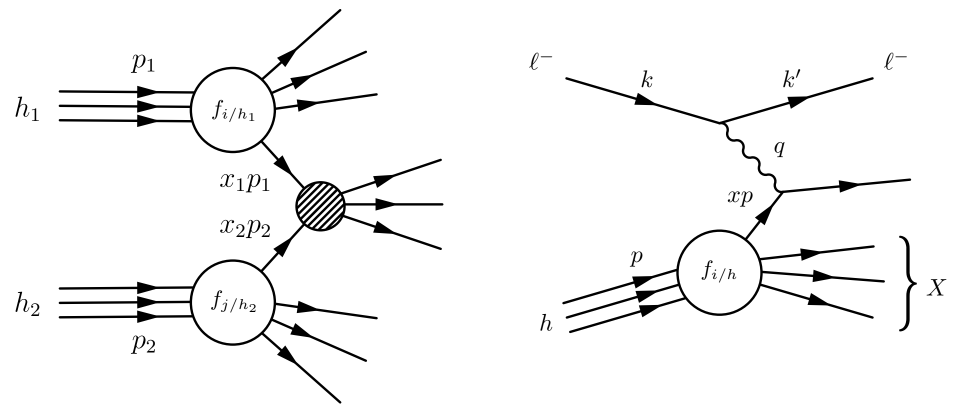
The factorized cross-section 1.15 has a non-integrable divergence if the radiated gluon is very soft, i.e., $E_n \to 0$ and/or the radiation angle is very small $\theta \to 0$. This apparent contradiction to the finite observables measured in the experiments is explained by the circumstance that the partonic cross-section is not an actual physical observable. Due to the colour confinement explained in Section 1.2.2, quarks and gluons are not observed as freely propagating partciles but as a compound states, i.e., hadrons that are a singlet under rotations in colour space. This problem can be solved by "absorbing" the unpleasant divergencies in the (renormalized) *parton distribution functions* (PDFs), which are the subject of the following section.

1.2.4 Factorization theorem and DGLAP equations

In general, a QCD process includes both short- and long-distant behavior. Therefore, perturbation theory alone is not sufficient to provide reliable predictions. Especially at lepton-hadron and hadron-hadron colliding experiments, most QCD processes live in the low energetic regime where the formation of colour singlet states, i.e., bound hadrons takes place. The *factorization theorem* divides the cross-section into a hard process – that comes with large momentum transfer Q^2 and hence can be computed with perturbation theory (see section 1.2.3) – and a long-distant part that describes processes with low momentum transfer, which is mostly based on empirical models. Figure 1.2 provides a visualization of the factorization theorem for the example of hadron-hadron interaction (1.2a) and deep inelastic scattering (1.2b) through hadron-lepton interaction. In Figure 1.2, the structure of the hadrons (such as protons) is described by the PDFs, while the hard subprocess is given by the scattering cross-section. According to the factorization theorem, the differential cross-section $\mathrm{d}\sigma^{h_1h_2\to f}$ of a physical process $h_1h_2 \to f$ (read: interaction between hadron h_1 and h_2 with the final state f) is given by the convolutional integral

$$\mathrm{d}\sigma^{h_1h_2\to f} = \int_{[0,1]}\int_{[0,1]} \mathrm{d}x_1\mathrm{d}x_2 \sum_{i,j} f_{i/h_1}(x_1,\mu_F) f_{j/h_2}(x_2,\mu_F)\mathrm{d}\sigma^{ij\to f}\left(x_1, x_2, \left(\frac{Q}{\mu_F}\right)^2\right), \tag{1.16}$$

with x_1, x_2 being the *momentum fraction* carried by the respective parton, $\sigma^{ij\to f}$ being the partonic cross-section and μ_F denoting the so-called *factorization scale* that can be thought of the scale that separates long and short distance contributions. Emission below the factorization scale (long distance effect with small energy scales Q^2) are described by the PDFs. Equation 1.16 divides the computation of the cross-section into two parts: the hard scattering cross-section $\mathrm{d}\sigma^{h_1h_2\to f}$ at some order of perturbation theory and non-perturbative contribution that accounts for the complex internal structure of the hadrons involved in the interaction. Loosely speaking, the computation of the cross-section has been separated in an "analytical" and an "empirical" part. Equation 1.16 also provides experimental instruction on how to measure the PDFs in Deep Inelastic Scattering (DIS).



(a) Hadron-hadron interactions *(b)* Deep inelastic scattering

Fig. 1.2: Visualization of the factorization theorem for hadron-hadron interactions (1.2a) and deep inelastic scattering (1.2b) with the respective flavour-dependent PDFs $f_{i/h}$.

The precision of the cross-section computation according to the factorization theorem 1.2 depends on the uncertainty of the hard scattering cross-section and the PDFs. While the former depends on the order of perturbation theory, i.e., the number of terms that are included in the perturbative expansion of the cross-section, the latter is dominated by experimental uncertainties in the measurement of the PDFs. Therefore, a precise measurement of $f_{i/h}(x, \mu_F)$ is indispensable and utterly important to obtain precise predictions of the cross-section.

The PDFs as introduced above are used to provide a non-perturbative description of the internal structure of hadrons. More specifically, at leading-order the parton distribution function $f_{i/h}(x, Q^2)$ of a hadron h corresponds to the probability to find a parton of type i (a quark of a certain flavor or a gluon) with a longitudinal momentum fraction x of the compound object at an energy scale of Q^2. While (at present) the PDFs itself cannot be derived from first principles in QFT, their evolution with the energy scale Q^2, which can be thought of as the energy transfer between the hadron and its scattering partner, is accurately predicted by the so-called DGLAP (Dokshitzer–Gribov–Lipatov–Altarelli–Parisi) equations [Altarelli and Parisi, 1977, Dokshitzer, 1977, Gribov and Lipatov, 1972] which follows from the requirement that the observable structure functions are independent of the non-physical factorization scale. The DGLAP equations, which describe the PDFs' dependence on μ_F and momentum fraction x, are given by

$$\mu_F^2 \frac{\partial f_{i/h}(x, \mu_F^2)}{\partial \ln \mu_F^2} = \sum_j \frac{\alpha_S}{2\pi} \int_x^1 \frac{\mathrm{d}z}{z} P_{ij}(z) f_{j/h}\left(\frac{x}{z}, \mu_F^2\right), \tag{1.17}$$

where P_{ij} are the (leading-order) spin-averaged, regularized *splitting functions/kernels* that describe the probability of a daughter parton i splitting from a parent parton [Höche, 2014]. A pictorial representation of the DGLAP equations is given in Figure 1.3.

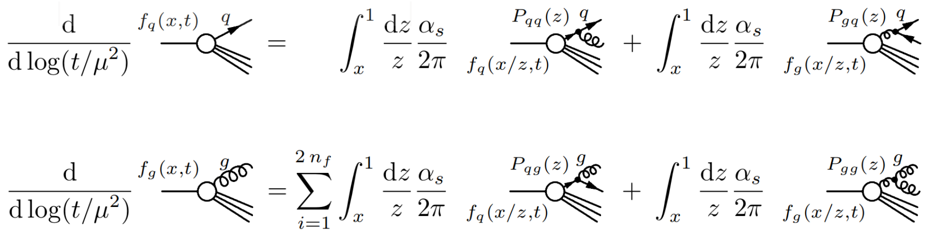
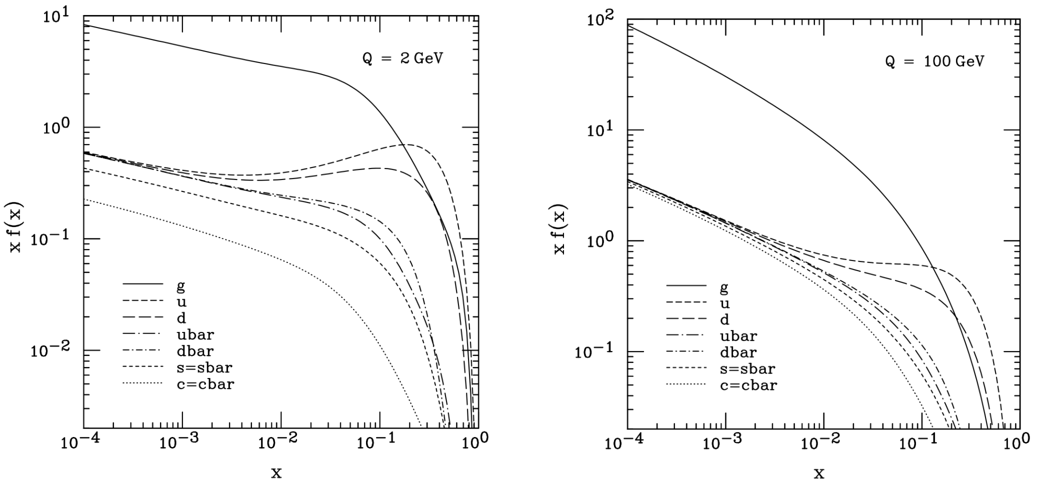


Fig. 1.3: Pictorial representation of the evolution of the parton distribution functions via the DGLAP equations (adapted from Höche, 2014, Fig. 1, p. 3).

With the DGLAP equation at our disposal, the PDFs measured at some lower energy scale Q^2 – as it has been extensively done at the HERA and the Tevatron collider in DIS – can be evolved to higher energy scales required by more recent colliders like the LHC. In this sense, the PDFs are *universal.*



(a) $Q = 2\,\mathrm{GeV}$ *(b)* $Q = 100\,\mathrm{GeV}$

Fig. 1.4: Parton distribution functions (CTEQ6M) for different partons at the energy scale $Q = 2\,\mathrm{GeV}$ (1.4a) and $Q = 100\,\mathrm{GeV}$ (1.4b) (adapted from Pumplin *et al.*, 2002, Fig. 1, p. 8).

Figure 1.4 gives an example of two parton distribution functions (CTEQ6M) published by the CTEQ Collaboration [Pumplin *et al.*, 2002]. The two plots illustrate the PDFs for different energy scales $Q = 2\,\mathrm{GeV}$ and $Q = 100\,\mathrm{GeV}$ for different partons in the hadron. The up-type quarks are the dominant distribution for small values of Q and large momentum fractions x. For smaller values of x, however, the distribution is mostly dominated by gluons; hence, they carrying small momentum fractions. The distribution changes significantly for higher energy scales. For higher values of Q, the distribution becomes progressively flavor-independent for small values of x, a flavor symmetry that is not present in the PDF for $Q = 2\,\mathrm{GeV}$.

The parton distribution functions are, broadly speaking, experimentally determined by fitting a large number of measured cross-sections in deep inelastic scattering in a

(Q^2, x)-grid. The precise and well controlled experimental conditions in deep nuclear-lepton scattering (see Figure 1.2b) cross-section measurements along with the accurate predictions in perturbative QED provide an ideal environment for probing the internal structure of hadrons.

1.2.5 Parton shower and Sudakov form factors

The previous section introduced the very important concept of factorization in perturbative QCD. The principle idea behind factorization is the decomposition of some measurable quantity, e.g. the cross-section of some process, into two (or possibly more) independent factors, whereby each factor only depends on its dedicated energy scale [Collins, 2003]. This Paragraph introduces yet another factorization theorem that plays a crucial role in the computation of the Sudakov form factors and hence in the simulation of parton showers: the so-called (soft-)collinear factorization or *collinar approximation* theorem.

The hard subprocess in the interaction involves large momentum transfers Q^2 between the interacting partons – which is why perturbation theory is applicable in the first place – which again cause acceleration and hence the emission of additional radiation in form of photons and/or gluons, depending on whether the accelerated particles carry an electric charge (QED) and/or a colour charge (QCD). Contrary to the gauge boson in QED, the photon, the gluons in QCD carry a (colour) charge by themselves and hence emit further radiation if accelerated in form of additional gluons. Within the picture of perturbative QCD, the additional radiation of gluons can be interpreted as higher-order corrections to the hard subprocess.

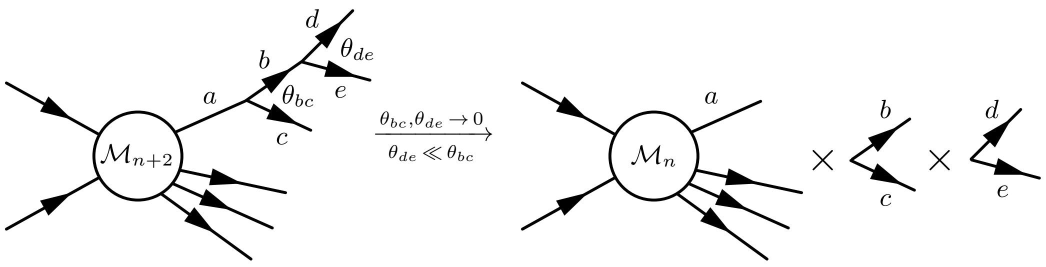


Fig. 1.5: The collinear factorization theorem for the example of a matrix element $\mathcal{M}_{n+2}$ of a given process with $n+2$ external legs and a final state a that branches (splits) two times with ordered splitting angles θ.

However, these corrections to the hard subprocess cannot be computed exactly for all radiation angles and energy fractions. The problem can be bypassed by using an approximation scheme that includes only the dominant contributions to all orders of perturbation theory in the limit of soft and collinear radiation. This approximation scheme is the so-called *collinear factorization theorem* (see Figure 1.5). The collinear factorization theorem states that the $(n+1)$-parton differential cross-section with an additional soft and collinear gluon can be factorized into the n-parton differential cross-section before splitting and the (Dokshitzer-Gribov-Lipatov-Altarelli-Parisi) *splitting functions* $P_{i,jk}(z, \phi)$ that gives the distribution of the fraction of energy of parton i carried by j:

$$\mathrm{d}\sigma_{n+1} \approx \mathrm{d}\sigma_n \frac{\alpha_S}{2\pi} \frac{\mathrm{d}\theta^2}{\theta^2} \mathrm{d}z \, \mathrm{d}\phi \, P_{i,jk}(z, \phi). \tag{1.18}$$

Equation 1.18 is at leading-order of perturbation theory and holds under the assumption of an *almost collinear* splitting of the parton of type i to jk. The similarity of Equation 1.18 and the DGLAP equation (1.17) introduced in Section 1.2.4 is not a coincidence: averaging the splitting functions over ϕ results in the same splitting functions as those appearing in the DGALP equation.

The sequential application of Equation 1.18 by using the Monte Carlo method (see Chapter 2) to generate "random numbers" of z, t and ϕ (see section 2.1) defines a Markov process that can be used to produce an arbitrary number of parton splittings and hence an arbitrary number of particles [Höche, 2014, Webber, 2011]. This procedure is a probabilistic approach to parton shower simulation. It is important to emphasise again that the aforementioned splitting procedure is a Markov process (or a Markov chain); therefore, the splitting of the partons does not depend on the previous splittings (the history) of the system and hence neglects (quantum) interference between the radiation that is produced. Furthermore, the collinear factorization theorem requires the splittings to be ordered according to some *evolution variable.* There are different choices for the evolution variable. In Equation 1.18 the splittings are ordered according to θ^2; however, alongside angular-ordered parton showers there are also ordering schemes according to the vitality q^2 and the p_T of the process (in the collinear limit all definitions of the evolution variable are equivalent $\mathrm{d}\theta^2/\theta^2 = \mathrm{d}k_\perp^2/k_\perp^2 = \mathrm{d}q^2/q^2$). The evolution variable allows defining a cut at which the system falls into the hadronization scale and hence the perturbative splitting process is terminated.

The differential probability of a parton i to split into jk in the interval $[q^2, q^2 + \mathrm{d}q^2]$ according to Equation 1.18 is given by:

$$\mathrm{d}\mathcal{P}_{i\to jk} = \frac{\alpha_S}{2\pi}\frac{\mathrm{d}q^2}{q^2}\sum_{jk}\int_z\int_\phi \mathrm{d}z'\,\mathrm{d}\phi'\,P_{i,jk}(z',\phi'). \tag{1.19}$$

The requirement of unitarity gives the probability that a parton i does *not* split into $j+k$ by $\mathcal{P}_{i\nrightarrow jk} = 1 - \mathcal{P}_{i\to jk}$. The probability that a parton does not emits additional radiation within the interval $[q^2, Q_0^2]$, whereby Q_0^2 is some resolution criterion, is given by the so-called *Sudakov form factor*

$$\begin{aligned}
\Delta_i(Q_0^2, q^2) &:= \lim_{n\to\infty}\prod_{k=0}^{n-1}\mathcal{P}(z_k,\phi)_{i\nrightarrow jk} \\
&= \lim_{n\to\infty}\prod_{k=0}^{n-1}\left[1 - \frac{\alpha_S}{2\pi}\int_{\frac{Q_0^2}{q^2}}^{1-\frac{Q_0^2}{q^2}}\frac{\mathrm{d}q'^2}{q'^2}\sum_{jk}\int_z\int_\phi \mathrm{d}z'\,\mathrm{d}\phi'\,P_{i,jk}(z',\phi')\right] \\
&= \exp\left\{-\frac{\alpha_S}{2\pi}\int_{q^2}^{Q_0^2}\frac{\mathrm{d}k^2}{k^2}\int_{\frac{Q_0^2}{k^2}}^{1-\frac{Q_0^2}{k^2}}\frac{\mathrm{d}q'^2}{q'^2}\sum_{jk}\int_z\int_\phi \mathrm{d}z'\,\mathrm{d}\phi'\,P_{i,jk}(z',\phi')\right\} \\
&= \exp\left\{-\int_{q^2}^{Q_0^2}\frac{\mathrm{d}k^2}{k^2}\mathcal{P}_{i\to jk}\right\},
\end{aligned} \tag{1.20}$$

whereby the index i in Δ_i refers to the i^{th} parton that splits into jk [Buckley and others, 2011]. Since the calculation of the Sudakov form factors is based on the probability of a parton *not* to split, it includes not only the collinear-enhanced real parton emissions, but also virtual quantum loop corrections to all orders of perturbation theory. The value of the resolution variable Q_0^2 is a question of definition, with the integration limits of z defining a range in which the splitting process is experimental resolvable. Splittings with too soft, i.e., to small values of z are not included in the parton shower.

Obviously, the function $\Delta_i(Q_0^2, q^2)$ given by Equation 1.20 is the solution to the following linear first-order differential equation

$$-\frac{\mathrm{d}\Delta_i(Q_0^2, q^2)}{\mathrm{d}q^2} = \Delta_i(Q_0^2, q^2)\frac{\mathrm{d}\mathcal{P}_i}{\mathrm{d}q^2}, \tag{1.21}$$

which accounts for the fact that the change of probability ${}^{\mathrm{d}\Delta_i(Q_0^2,q^2)}/_{\Delta_i(Q_0^2,q^2)}$ is proportional to the branching probability $\mathcal{P}_i$. This is in close analogy to the law of radioactive decay. The actual implementation of Equation 1.21 is rather straightforward and discussed in Chapter 2.

It is important to note that in the limit of soft and collinear radiation the higher-order corrections and the leading-order matrix element factorize *in all orders of perturbation theory* due to the factorization theorem 1.18. This can be seen by rewriting Equation 1.20 $\Delta_i(Q_0^2, Q^2) \propto \exp\left\{-C_F \frac{\alpha_S}{2\pi}\log^2\frac{Q^2}{Q_0^2}\right\}$ (see Buckley and others, 2011, Eq. 17, p. 27) and expressing the exponential function in terms of its series representation

$$\Delta_i(Q_0^2, Q^2) \propto \lim_{N\to\infty}\sum_{k=0}^{N}\left(\frac{\alpha_S}{2\pi}\right)^k\left(\log^2\frac{Q}{Q_0}\right)^k. \tag{1.22}$$

Equation 1.22 is an expansion in terms of the strong coupling α_S; hence, the Sudakov form factors are the sum of *all* leading collinear logarithms to all orders in perturbation theory. However, it should be emphasised that this statement is only true in the limit of collinear and soft radiation. Therefore, the parton shower through Sudakov form factors provides meaningful result in the soft regions of phase space; the radiation of high-energy and wide-angle particles is not well described. This will be the main motivation for the joint use of matrix element generators and shower Monte Carlos in Chapter 2.

1.3 QCD phenomenology

The previous chapter 1.2 reviewed the theoretical basics of perturbative QCD and how it can be used to compute the hard-scattering matrix elements for given processes as well as the showering approximation (parton shower) to simulate all dominant collinear QCD radiation. As it was shown, the cascade of splittings caused by the initial parton results in a large number of additional particles that all carry a fraction of the primary parton's four-momentum. To reconstruct the four-momentum of the original parton, it is therefore necessary to measure and reconstruct the additional particles that are a consequence of this showering process. However, a particle detector is by no means a sterile environment. Usually, there are several interactions involved that cause pile-up contamination of the environment as well as higher particle multiplicities in the final state. As a result, the difficulty lies in the identification of active regions in the detector that are assumed to originate from the same process and the same initializing particle. This is – as one can imagine – a very sophisticated task. Over the years many different concepts and techniques have been developed to solve this problem in a reasonable manner. The purpose of this chapter is therefore to provide a short introduction into the extensive field of jet physics, i.e., a phenomenological approach to QCD whereby it is limited to the main aspects of this area.

1.3.1 Jets in particle physics

The abstract object that particle physicists refere to as *jet* is the remaining signature of quarks and gluons that are produced in high-energy particle collisions but escape direct detection in the experiment. Therefore, in many instances, the reconstruction of jets is the only way to gain insight into the underlying process of an event. Furthermore, hadron colliders such as the LHC are dominated by QCD, which is why the reconstruction of jets often is the only way to extract structures and make sense out of an event in the first place. As a result, studying jets and its properties is utterly important and the key element for many studies – in particular for the search of new physics beyond the Standard Model. Figure 1.6 shows to different jet typologies as they are steady observed in experiments.

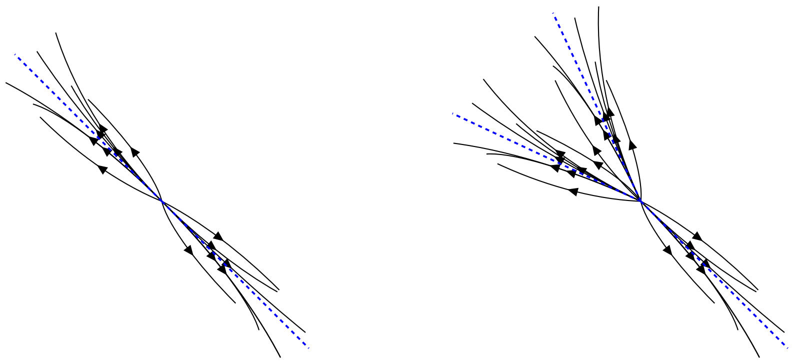


Fig. 1.6: Left: a two-jet event with clear separation of the final state particles; right: a three-jet event with significant overlap among radiation from two different sources. The dashed, blue line indicated the direction of the respective underlying parton.

Each solid line in Figure 1.6 represents the path or track of a particle that originates from the interaction point of the event. The tracks of charged particles are measured in the inner detector regions, which are usually equipped with semiconductor detectors, while the energy deposition of the particles is measured in the so-called calorimeter (derived from Latin *calor*, "heat", and Greek *metron* (μέτρον), "measure") of the detector. The left Figure in 1.6 is a somewhat simplified illustration of a two-jet event, i.e. two separated collection of particles in the final state to balance energy-momentum. The right hand side of the Figure, however, shows a three-jet event with three final state particles. As seen in this topology, there is the possibility of significant overlap between the particles measured in the detector, making the assignment to the underlying process difficult. This becomes even worse for higher jet multiplicities in the final state. For many years, theorists and experimentalists alike worked on efficient methods that allow the reconstruction of jets through clustering particles. In the course of these efforts, a variety of different algorithms has been developed to fulfill this task. It is therefore evident that the definition of a jet is based on the respective jet reconstruction algorithm that are used since different reconstruction methods will result in non-identical jets with different particle content. Therefore the following section introduces the most widely used jet reconstruction algorithms in experimental particle physics along with a rather broad overview of the basic requirements for a jet algorithm.

1.3.2 Infrared and collinear safety

Jet reconstruction is the task to group particles into orthogonal categories that are assumed to originate from the same parton of the underlying hard scattering process. To organize the particles into groups, the jet reconstruction algorithm usually comes with a distant measure, as well as recombination scheme that specifies how particles are (re-)combined into a single four-vector for the next iterative step [Glover and Kosower, 1996]. The most simple recombination scheme would be the four-vector sum of the constituents.

There exists a large variety of different jet algorithms since, historically, each experiment used its own definition of a jet. In 1990, there was an attempt to specify a set of criteria based on experimental and theoretical considerations that each jet reconstruction algorithm should meet. This set of rules became later known as the "Snowmass Accord", which was the first step "[t]oward a standardization of jet definitions[.]" (Huth and others, 1990). The CDF Collaboration at the Tevatron, Illinois, USA, was the first collaboration that tried to implement this "accord" [Abe, 1992].

In the Snowmass accord from 1990 the Authors proposed "[s]everal important properties that should be met by a jet definition[.]" (Huth and others, 1990). Among other things, the jet definition should "[y]ield[.] finite cross-sections at any order of perturbation theory" as well as "[...] a cross-section that is relatively insensitive to hadronization" (Huth and others, 1990, p. 6, property 4 & 5). Those properties are guaranteed if the definition of a jet is *infrared* and *collinear* (IRC) safe. Formally, infrared safeness can be defined as:

An observable is infrared safe if, for any parton configuration,adding an infinitely soft parton does not affect the observable at all (Seymour, 1998, Def. 4, p.5).

Similar, one defines collinear safeness:

An observable is collinear safe if, for any parton configuration, replacing any massless parton by an exactly collinear pair of massless partons does not affect the observable at all (Seymour, 1998, Def. 5, p.5).

As described in Section 1.2, the parton in the hard interaction undergoes numerous soft and collinear splittings as part of the fragmentation process that finally leads to stable hadrons in the final state. A consistent jet definition should therefore be insensitive to those effects, i.e., the radiation of additional soft and collinear partons shall not change the jet. Figure 1.7 and 1.8 show an idealized example of a fictional jet algorithm that is neither infrared nor collinear safe.

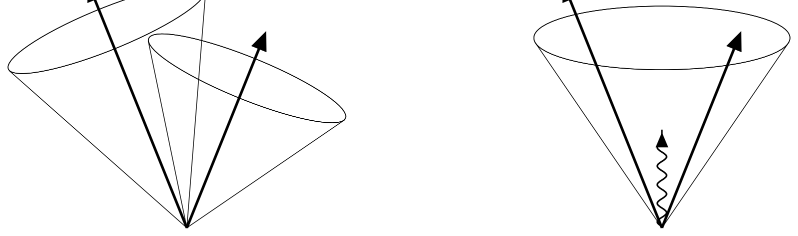


Fig. 1.7: Visualization of infrared safety: the configuration of reconstructed jets must not change with the emission of another *soft* particle.

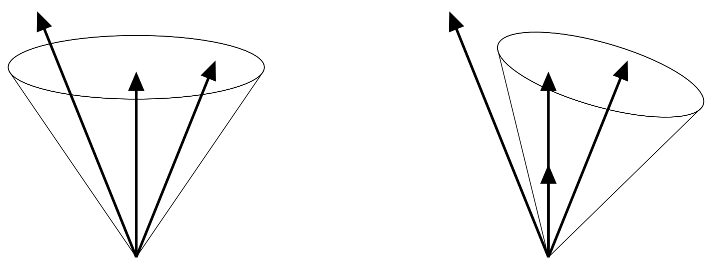


Fig. 1.8: Visualization of collinear safety: the configuration of reconstructed jets must not change with one particle replace by two collinear particles.

Apart from that, fixed-order calculations in perturbative QCD soft radiation and collinear splittings are accompany by divergent matrix elements already at tree-level. Those divergences are guaranteed to cancel with higher order loop diagrams that enter the calculation with an opposite sign. This can easily be seen by means of the leading order cross-section of process $\ell^+\ell \rightarrow q\bar{q}$. The dominant contribution to the next-to-leading order correction comes from additional QCD radiation $\ell^+\ell \rightarrow q\bar{q}g$. The double-differential cross-section for this process is then given by

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}x_q \mathrm{d}x_{\bar{q}}} = \sigma^{LO} \frac{\alpha_s}{2\pi} C_F \frac{x_q^2 + x_{\bar{q}}^2}{(1 - x_{\bar{q}})(1 + x_q)}, \tag{1.23}$$

with x_q and $x_{\bar{q}}$ being the momentum fraction carried by the quark respectively the antiquark. The cross-section according to Equation 1.23 is obviously ill-defined for collinear radiation of a gluon (i.e. $x_q \rightarrow 0$ or $x_{\bar{q}} \rightarrow 0$) as well as a very soft gluon (i.e. $x_g \rightarrow 0 \Rightarrow (x_q, x_{\bar{q}}) \rightarrow (1,1)$). First, consider the radiation of a very soft or collinear gluon that is unresolvable in the experiment. This situation correspond to the two-jet cross-section that is then given by $\sigma_{\text{two-jet}}(T) = \sigma^{LO}(1 + \alpha_s f(T) + \mathcal{O}(\alpha_s^2))$, with T being the separation between two- and three-jet in the region (*cf.* Dalitz plot) to be integrated over. The function f is still divergent for $T \rightarrow 1$. However, if one performs the integration over the three-jet region $\sigma_{\text{three-jet}}(T) = \sigma^{LO}\alpha_s g(T) + \mathcal{O}(\alpha_s^2)$, where the gluon is resolvable in the experiment, and computes the total inclusive cross-section $\sigma^{\text{tot}} = \sigma_{\text{two-jet}} + \sigma_{\text{three-jet}} + \cdots = \sigma^{LO}(1 + \alpha_s [f(T) + g(T)] + \mathcal{O}(\alpha_s^2))$ it turns out that the infinities precisely cancel each other in the limit $\lim_{T\rightarrow 1} [f(T) + g(T)] = 0$. Hence, the cross-section $\sigma^{\text{tot}} = \sigma^{LO} \left(1 + \frac{3}{4} C_F \frac{\alpha_s}{2\pi} + \mathcal{O}(\alpha_s^2)\right)$ is finite [Chiochia *et al.*, 2010].

This simple sample calculation shows that in the inclusive scenario the collinear and soft infinities cancel each other out. This, however, is not necessarily the case for exclusive kinematic measurements in which only a selected group of particles is used to compute certain observables. If the respective observable is inherently IRC safe, though, the calculation will yield finite results.

1.3.3 Cone based algorithms

For a long time, the standard jet reconstruction algorithms have been so-called cone algorithms that are based on the geometric definition of a cone (the first jet algorithm has been developed by Sterman and Weinberg in 1977 [Sterman and Weinberg, 1977]). Even though this definition might appear straightforward it still allows for numerous different implementations of reconstruction algorithms. However, with the Snowmass Accord as guidance regarding meaningful jet definitions, most cone algorithms seek to extremize the hadronic energy flow through a cone with a fixed radius R, with $R^2 = (\Delta\eta)^2 + (\Delta\phi)^2$, in

$\eta - \phi$ space. According to this definition, the transverse energy of the jet is simply given by the scalar sum of the constituent's transverse energy within the cone

$$E_{\mathrm{T}} = \sum_{i \in \{R_i < R\}} E_{\mathrm{T},i}, \tag{1.24}$$

while the pseudorapidity and azimuthal angle of the jet is the weighted sum of the constituent's position $\eta - \phi$ space

$$\eta = \frac{1}{E_{\mathrm{T}}} \sum_{i \in \{R_i < R\}} E_{\mathrm{T},i}\eta_i, \qquad \phi = \frac{1}{E_{\mathrm{T}}} \sum_{i \in \{R_i < R\}} E_{\mathrm{T},i}\phi_i. \tag{1.25}$$

Those very simple definitions [Seymour, 2000] already allow to reconstruct jets.

Most cone algorithms are seed-based, e.g. the particle with the largest momentum, and use of an iterative procedure to successively combine particles until stable configurations are found. Within the group of iterative cone algorithms, the most prominent methods are the iterative cone algorithm with progressive removal (IC-PR) – which uses the hardest cell as a seed – and the iterative cone algorithm with the split merge procedure – where all cells above a certain energy threshold are seeds. In contrast to iterative cone algorithms, fixed cone algorithms use a fixed geometry around the seed direction. Although cone algorithms are comparatively simple – or perhaps exactly for this reason –, most of them suffer a serious problem: they are soft (IC-SM) and collinear *un*safe (IC-PR) [Atkin, 2015]. The aforementioned problems have their origin in the seed that is used by the algorithm. Therefore, the seedless cone algorithm, SISCone, (Seedless Infrared Safe Cone [Salam and Soyez, 2007]) have been developed, which is the only representative of its family that is infrared as well as collinear safe. The algorithm SIScone uses to identify stable pseudo-jets is rather complicated since it involves several nested iterations and depends on several parameters. Hence, it would not be very conducive to discuss it at this point (a description of the complete algorithm can be found in Salam and Soyez, 2007, Algorithm 2, p. 12). The problems regarding cone based algorithms were one main motivation for the development of so-called sequential recombination algorithms for jet clustering, which are the subject of the next section.

1.3.4 Sequential recombination algorithms

The second family of jet reconstruction algorithms are the so-called *sequential recombination algorithms* that have been introduced in the '80s by the JADE collaboration [Bartel *et al.*, 1986]; hence, have their origin in studies of e^+e^- collisions at PETRA at DESY, Hamburg. Within this class, there exists a large variety of different algorithms that are based on different assumptions such as the splitting functions in QCD for the k_t algorithm [Catani *et al.*, 1993, Ellis and Soper, 1993], angular ordering for Cambridge-Aachen [Wobisch and Wengler, 1998] and collimated jet cores as used by anti-k_t [Cacciari *et al.*, 2008]. The first two are especially suited for studies regarding the substructure of jets, while the latter one is often used to study single-parton jets.

Sequential recombination algorithms are inherently (by construction) infrared and collinear safe. This property makes them theoretical superior to cone based reconstruction of jets which is why those are generally favoured by theorists. The problem was, however, that this class of algorithms suffered from a poor computational performance. This issue has mostly been resolved by the efficient implementation of the FastJet package [Cacciari *et al.*, 2011] in C++ that is used by the experiments at the LHC.

The k_t algorithm

When k_t algorithm was introduced by the JADE experiment at DESY in 1993, it was designed in accordance with studies at lepton colliders. The (modified) distance measure of the algorithm is given by

$$y_{ij} = \frac{2\min(E_i^2, E_j^2)}{Q^2}(1 - \cos\theta_{ij}), \tag{1.26}$$

whereby Q is the total energy of the even. Equation 1.26 is evaluated for each pair of particles from which the minimum $y_{\min}$ is determined. If $y_{\min}$ below some threshold value y_{cut}, then i and j are recombined into a new particle (referred to as a "pseudo-jet"); afterwards the procedure is repeated until the iteration terminates. Due to the definition 1.26 this algorithms favors soft particles, which results in somewhat "diffuse" shapes.

In the limit $\theta \ll 1$, y_{ij} is reduced to $y_{ij} \overset{\theta \lll 1}{=} (\min(E_i, E_j)\theta_{ij}/Q)^2$ which is the squared normalized transverse momentum – hence the name k_t algorithm. As mentioned in Section 1.3.4, the k_t algorithm can be related to the splitting functions in QCD by considering a soft and collinear branching $k \to ij$. In this case, the differential splitting is given by $\partial_{E_i,\theta_{ij}} P_{k\to ij} \sim (\min(E_i, E_j)\theta_{ij})^{-1}$.

Equation 1.26 has been constructed with regard to the application in lepton colliders. In hadron colliders, however, the total energy of an event is not well defined along with other complications. An adapted version of the k_t algorithm for hadron collisions is given by:

$$d_{ij} = \min(p_{\text{T},i}^2, p_{\text{T},j}^2)\Delta R_{ij}^2, \quad d_{iB} = p_{\text{T},i}^2, \tag{1.27}$$

with each particle being assigned to either a beam-jet (d_{iB}) or a final state-jet [Salam, 2010].

The Cambridge-Aachen algorithm

The Cambridge-Aachen algorithm uses a distant measure similar to Equation 1.26 and 1.27 along with a second distant measure defined as $v_{ij} = 2(1 - \cos\theta_{ij})$. The rationale behind this method was to combine the k_t algorithm with angular ordering that is related to the ordering in multiple gluon emissions. The actual algorithm is simple: Find the pair of particles with minimum v_{ij} and recombine those particles to a new pseudo-jet if the corresponding condition $y_{ij} < y_{\text{cut}}$ is satisfied and repeat the process. As it was the case for the k_t algorithm, the Cambridge-Aachen algorithm must be modified to meet the requirements of hadron colliders.

Like the k_t algorithm the Cambridge-Aachen's approach tends to result in irregular jet shapes. This problem is, *inter alia*, addressed by the next sequential recombination algorithms – the famous and widespread anti-k_t algorithm.

The anti-k_t algorithm

The so-called anti-k_t algorithm – which is the one used in this thesis – can be considered as being a generalization of the aforementioned (inclusive) k_t and Cambridge-Aachen algorithm. The distant measures of the anti-k_t algorithm is given by:

$$d_{ij} = \min(p_{\text{T},i}^{2p}, p_{\text{T},j}^{2p})\frac{\Delta R_{ij}^2}{R^2}, \quad d_{iB} = p_{\text{T},i}^{2p}, \tag{1.28}$$

where p is an additional parameter that allows to recover the k_t algorithm for $p = 1$ and Cambridge-Aachen for $p = 0$. For a value $p = -1$, the algorithm preferably clusters hard particles ($\min(p_{\mathrm{T},i}^{-2}, p_{\mathrm{T},j}^{-2}) \rightarrow \max(p_{\mathrm{T},i}^{2}, p_{\mathrm{T},j}^{2})$). An interesting "side-effect" of this algorithm is that it tends to produce very circular and hard jets, making it an attractive alternative for certain cone-type algorithms.

With the (anti-)k_t, Cambridge-Aachen and SIScone algorithm the four most commonly used jet reconstruct algorithms have been introduced.

1.3.5 Jet-related observables

The jet reconstruction algorithms introduced in Section 1.3.3 and 1.3.4 cluster final state hadrons measured in the detector into an object called *jet* whose four-momentum, which is assumed to roughly corresponding to the parton's involved in the hard subprocess, is then used for subsequent analysis. Out of the jet's kinematic variables, i.e., E_T^{jet}, ϕ^{jet}, η^{jet}, m^{jet} (which already completely defines the four-momentum of a particle), it is possible and reasonable to construct other observables that can be measured in the experiment and later compared to expectations from theory.

Jet-shape

The so-called *Jet-shape* Ψ_J is a jet observable which has been in use for a long time [Ellis *et al.*, 1992] and which is still in use to this day. Unlike global event shapes, the jet-shape is individually defined for each jet in an event. Since this thesis only considers the *leading jet* in an event, it is a perfect figure of merit to evaluate the quality of the generated data.

The (conventional) jet-shape is defined by:

$$\Psi(r) = \int_0^r \mathrm{d}r' \, \frac{p_\mathrm{T}(r')}{p_\mathrm{T}^{\mathrm{jet}}} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p_{\mathrm{T},i}}{p_\mathrm{T}^{\mathrm{jet}}}, \tag{1.29}$$

where the continuous integration has been approximated by a discrete sum over the pixel cells in the detector [Chien and Vitev, 2016]. As can be seen from Equation 1.29, the range of values of Ψ is limited to $\Psi \in [\Psi(r=0)=0, \Psi(r=R)=1]$. With this definition of the jet-shape, the interpretation is very simple: it measures the fraction of the total transverse momentum $p_\mathrm{T}^{\mathrm{jet}}$ that falls in the region defined by r. This observable has some discrimination power regarding the distinction of gluon or quark initialized jet; however, the identification efficiencies is rather poor. Notwithstanding, this variable is very useful as it allows to study the distribution of particles or jet's constituents as well as their momenta within the scope of this thesis. This is much more powerful than, e.g., just considering the transverse momentum of the jet.

Jet-width

The *jet-width* or *jet-broadening* as it was used by ALEPH and OPAL (e.g. [Ackerstaff and others, 1999], is yet another jet observable that can be used to distinguish between gluon and quark jets. It is defined by the simple equation

$$B_{\mathrm{jet}} = \frac{\sum_{i=1}^{N} |\mathbf{p}_i \times \mathbf{n}_{\mathrm{jet}}|}{\sum_{i=1}^{N} p_{\mathrm{T},i}^{\mathrm{jet}}}. \tag{1.30}$$

This definition, however, cannot be used with hadron colliders. ATLAS' definition of the jet-width is therefore modified accordingly

$$w_{\text{jet}} = \frac{1}{E_{\text{T}}^{\text{jet}}} \sum_{i=1}^{N} \Delta R_i \times E_{\text{T},i}, \tag{1.31}$$

with $E_{\text{T}}^{\text{jet}}$ being the scalar sum of the constituent's p_{T} [and, 2011]. This definition of the jet-width is quite natural, straightforward and has an easy interpretation: it gives the average distance between the clusters inside a jet. This definition is, by the way, very similar to N-subjettiness τ_1 for one jet (see Equation 1.32).

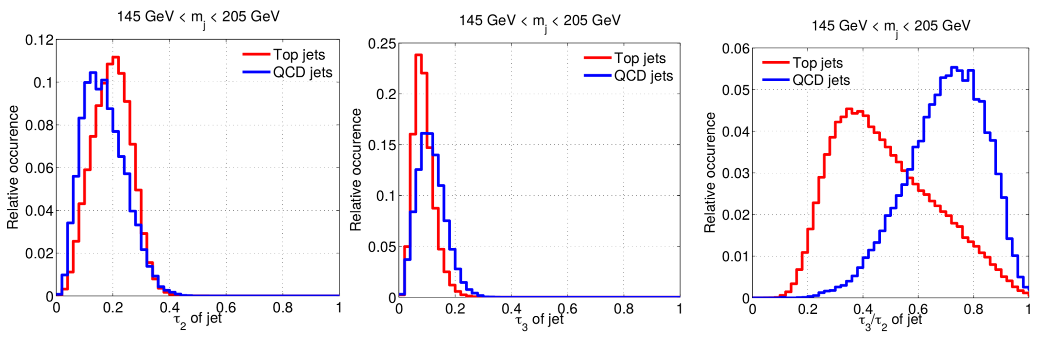
N-subjettiness

Another variable to study the substructure of a jet is the inclusive jet shape referred to as *N-subjettiness* [Thaler and Van Tilburg, 2011] that is based on N-jettiness [Stewart *et al.*, 2010]; although the latter one being an event shape, while the former one is defined for each jet individually. The N-subjettiness of a jet with an assumed number of N *subjets* is calculated based on the following equation:

$$\tau_{N,\beta} = \frac{1}{d_0} \sum_{k=0}^{N} p_{\text{T},i} \min\left\{\Delta R_{1,k}^{\beta}, \Delta R_{2,k}^{\beta}, \cdots \Delta R_{N,k}^{\beta}\right\}, \tag{1.32}$$

with $d_0 = R^{\beta} \sum_{k=1}^{N} p_{\text{T},i}$ and $\beta \in \mathbb{R}^{>0}$ (in this work, without exception, $\beta = 1$ is used. henceforth, unless specified differently, the simplified notation τ_N is synonymous with $\tau_N := \tau_{N,\beta=1}$). As mentioned in Section 1.3.5, for $N = 1$ and $R = 1$ the jet-width $w_{\text{jet}} = \tau_1^{R=1} = 1/p_{\text{T}}^{\text{jet}} \sum_{i=1}^{N} \Delta R_i p_{\text{T},i}$ if the clusters are taken to be massless. With this definition, it is easy to see that N-subjettiness provides an information about to which degree a jet can be regarded as being composed out of N subjets.

The following three plots in Figure 1.2 give an impression of the distribution of this variable for QCD and boosted W jets.



(a) 2-subjettiness *(b)* 3-subjettiness *(c)* 32-subjettiness

Plot 1.2: N-subjettiness distribution τ_2 (1.2a) and τ_2 (1.2b) for QCD and boosted W jets and a mass window $145\,\text{GeV} < m^{\text{jet}} < 205\,\text{GeV}$. The discrimination power of this variable manifests it self in the ratio $\tau_{32} := \tau_3/\tau_2$ (1.2c). (adapted from (Thaler and Van Tilburg, 2011, Fig. 2(b,c), 3(b), p. 8).

As can be seen in Figure 1.2, N-subjettiness itself does not provide a good discrimination between QCD and boosted W jets; however, the ratio τ_3/τ_2 does give a significant separation between the different processes.

In this thesis N-subjettiness is not used to discriminate between jets originating from different processes but to probe the substructure of the generated jets and compare them with expectations from the training data. It is considered to be another figure of merit that provides different perspective on the *manifold* learned by the neural network and hence allows to quantitatively estimate its performance. This statement holds true for all figures of merit and obervables used throughout this thesis.

Chapter 2

Event Simulation in HEP

Until now the focus was largely on theoretical and conceptional considerations. However, in physics – as in natural sciences in general –, each theory must prove itself valid in the experiment; otherwise, it will be buried in the vast cemetery of disproven theories.

In this chapter, both threads, theory and experiment, are wed together to provide an introduction into the broad topic of *event generators* along with their practical applications in experimental particle physics. A full description of this extensive subject is beyond the scope of this work; therefore, it is limited to merely convey an idea of the underlying principles that are common to most event generators.

The title of this chapter already reflects parts of its structure. Starting from the Monte Carlo method (2.1), the very general concept of Monte Carlo integration (2.1.1) and simulation (2.1.2) is introduced (which will also be relevant for the subsequent chapter), as both methods are fundamental and frequently used in High Energy particle Physics (HEP). The next section 2.2 then builds a bridge to the subject matter of the previous chapter by introducing the objective of Monte Carlo event generators (2.2.1) in the context of high-energy particle physics. Furthermore, special attention is devoted to the fixed-order computation of the matrix element of the underlying hard subprocess (2.2.2) as well as the simulation of parton showers utilizing the Sudakov form factors (2.2.3) that have been introduced in the previous chapter. Section 2.2.4 combines and summarizes all accumulated insights to give a short overview of the individual steps common to most gevent generators. This is followed by the introduction of different types of event generators and their most prominent representatives (2.3). Finally, the chapter is closed by a brief consideration of challenges arising in the combination of matrix element generators and and shower Monte Carlos (2.4).

2.1 The "Monte Carlo" method

Generally speaking, the Monte Carlo method – a term first introduced by the Greek-American physicist Nicholas Metropolis (Νικόλαος Μητρόπουλος) and an innuendo to the eponymic Monégasque city – is, in its broadest sense, a class of numerical algorithms to solve complex mathematical problems employing probabilistic principles. The application of this method can be broadly divided into Monte Carlo *integration* (2.1.1) and *simulation* (2.1.2).

It was first made public in a theoretical essay by Nicholas Metropolis and the Polish-American scientist Stanłslaw Ulam in 1949, mainly to take "[...] a statistical approach to the study of differential equations [...]" (Metropolis and Ulam, 1949). However, the proposed method was already invented several years before by Ulam when he took part in the nuclear weapons program at the Los Alamos National Laboratory, USA (where he was involved in the Manhattan Project during World War II). The Hungarian-American scientist John von Neumann, who was involved in the same project at that time, quickly recognized the importance of the novel method suggested by his colleague. In a first practical application of the method, Ulam and von Neumann studied, under strict secrecy self-evidently, the problem of neutron diffusion, which is essential to build and design nuclear weapons [Richtmyer *et al.*, 1947].

Although the Monte Carlo method has a turbulent history that goes back to a period of geopolitical tension and military confrontation, it has become an indispensable tool in a wide spectrum of scientific applications: from industrial engineering; the simulation of physical, chemical and biological processes of all different kinds; to economics and finance. Everywhere people "play dice".

2.1.1 Monte Carlo integration

Formally, Monte Carlo computations are equivalent to an approximate integration in higher dimensional space. Consider, for instance, the d-dimensional integral

$$I = \int_{[0,1]^d} \mathrm{d}\boldsymbol{x}\, f(x) = \int_{[0,1]} \mathrm{d}x_1 \cdots \int_{[0,1]} \mathrm{d}x_d\, f(x_1, \ldots, x_d), \tag{2.1}$$

whereby $f : [0,1]^d \to \mathbb{R}^d$ is a real-valued function over (for reasons of simplicity and without loss of generality) the normalized hypercube $[0,1]^d = \Pi_{i=1}^d [0,1]$. Equation 2.1 may be interpreted as an expectation value $\mathbb{E}_{X \sim \mathcal{U}^d(0,1)}\left[f(X)\right]$ of the function f over the d-dimensional uniform probability distribution $\mathcal{U}^d(0,1)$, with X being an vector of *i.i.d.* random variables $X \sim \mathcal{U}^d(0,1)$, meaning $X_i \in [0,1]$ with $i \in \mathbb{N}^{\leq d} \setminus \{0\}$.

The Monte Carlo approximation of expression 2.1 is then given by

$$S_n = \frac{1}{n} \sum_{i=1}^{n} f(x_i), \tag{2.2}$$

where $\{x_i\}_{i=1}^n$ are n independent samples of the random variable X. The convergence of the Monte Carlo approximation 2.2 to the integral 2.1 is guaranteed by *the law of large numbers*, i.e., the sample mean of a sequence of variables $\overline{S}_n = \frac{1}{n}\sum_{i=1}^n S_i$ converges to the expectation value $\overline{S}_n \to \mu := I$ for $n \to \infty$ *iff* the first two moments $\mathbb{E}[\overline{S}_n] = \mu \in \mathbb{R}$ and $\mathbb{E}\left[(S_i - \mu)^2\right] = \sigma^2 \in \mathbb{R}^{>0}$ exist. The circumstance that the sample mean converges to a fixed number justifies the approximation 2.2 of 2.1.

The statements above vindicates the estimation of integrals by means of the Monte Carlo approximation. However, what are the benefits of Monte Carlo integration compared to other numerical integration methods available (e.g. rectangular integration or Simpson's rule)? The advantage lies in the statistical nature of the method that not only connects it to the law of large numbers but also to *the central limit theorem*[1] in probability theory.

According to the central limit theorem, the sample mean $\overline{S}_n$ of a sequence $\{S_i\}_{i=1}^n$ of with $\mathbb{E}[S_n] = \mu$ and $\mathrm{Var}(S_i) = \sigma^2$ follows approximately a normal distribution $\mathcal{N}(\mu, \sigma^2/n)$. This implies that the standard score $\sqrt{n}\frac{\overline{S}_n - \mu}{\sigma}$ converges to a stochastic variable that is

[1]For a proof of the central limit theorem see [Feller, 1945].

normal distributed with zero mean and unity variance, i.e. $\sqrt{n}\frac{\overline{X}_n-\mu}{\sigma} \overset{n\to\infty}{\sim} \mathcal{N}(0,1)$.

To relate the statements above with Monte Carlo integration, consider the approximation error $S_n - I(f)$; according to the central limit theorem this quantity is (approximately) normal distributed $S_n - I(f) \dot{\sim} \mathcal{N}(\mu, \sigma^2/n)$. Then, the following statement is true within the limits of finite statistical precision: $P\left(a\frac{\sigma}{\sqrt{n}} < S_n - I(f) < b\frac{\sigma}{\sqrt{n}})\right) \approx \Phi(b) - \Phi(a)$, with $\Phi(\cdot)$ being the cumulative distribution function. From this simple consideration, it can be concluded that the approximation error of the Monte Carlo method behaves like $\mathcal{O}\left(1/\sqrt{n}\right)$. The statement holds – and this is the decisive aspect – regardless of the dimensionality $d \in \mathbb{N}$ of the integral 2.1. This property, the approximation error being independent of the dimensionality of the integral to be calculated, makes the Monte Carlo method superior when it comes to evaluate *higher dimensional* integrals (for $d \leq 2$ there are numerical methods that give lower or similar errors with $\mathcal{O}\left(1/n^{1/d}\right)$, e.g. the general trapezoid approximation).

The line of arguments above explain the success of Monte Carlo integration in many scientific fields, which usually deal with high-dimensional integrals. The same argument applies to high energy particle physics due to the appearance of high-dimensional phase-space integrals in, e.g., cross-section computations. This makes the Monte Carlo method (integration) a natural method of choice in particle physics and hence in the event generators discussed in Section 2.2.

2.1.2 Monte Carlo simulation

As it has been mentioned, the code name “Monte Carlo” is derived from the city with the same name in Monaco, situated along the French Riviera. It is well known for its numerous casinos, which makes Monte Carlo the world’s largest gambling centre. The term is by no means arbitrarily chosen: Nicholas Metropolis suggested the name as an allusion to Ulam’s uncle who used to borrow money from its relative, only to squandering it for gambling [Metropolis and Aspray, 1987].

A typical problem in statistical physics (as well as in many other fields) is to estimate the mean value of some function h of a random variable $\boldsymbol{x}$ with respect to a probability density function $f(\boldsymbol{x})$ (PDF)

$$\langle h \rangle := \mathbb{E}_{\boldsymbol{x}\sim f(\boldsymbol{x})}\left[h(\boldsymbol{x})\right] = \int_\Omega \mathrm{d}\boldsymbol{x}\, h(\boldsymbol{x}) f(\boldsymbol{x}). \tag{2.3}$$

A typical example in experimental particle physics would be, for instance, the expected mean energy $\langle E \rangle$ deposition in some region and part of the detector system. The probability of some particle to deposit energy in the interval $[E, E + dE]$ is given by $f(E)dE$. However, the PDF might be a very complicated function that is *a priori* unknown. This is usually the case. The Monte Carlo method provides a procedure to estimate the unknown PDF, f, through simulation of the actual (physical) processes involved and hence is referred to as *Monte Carlo simulations*. The objective is to numerically mimic and reproduce the basic processes as accurately as possible to get a good estimation of the underlying PDF that have generated the data. This requires precise theoretical and experimental models that accounts for the stochastic nature of the respective process under consideration.

Monte Carlo simulations are used extensively in particle event generators to estimate the expected number of events in some regions of phase space, and therefore it is an indispensable tool in HEP.

2.2 Monte Carlo event simulation

From a philosophical perspective, theoretical concepts in physics are studied concerning their ontological assumptions and implications in nature. Hence, the interplay between theory and experiment in physics is an essential concept – one is not possible without the other. From an epistemological point of view, the agreement of theoretical predictions and corresponding observations in nature does not only allow for a description of the process under consideration of a specific language but also reveals a deep insight in the structure of the surrounding reality [Hartmann, 2000]. This fundamental concept applies to all physics questions and fields. Indeed, one could say that for a theory to be called physical it needs to describe measurable processes. A theory that protects itself by being non-verifiable and hence non-falsifiable is not valid in a physical sense; it is part of metaphysics.

To apply those concepts to particle physics, it is necessary to compare experimental observations with theoretical predictions. Monte Carlo event generators have been developed to satisfy exactly this need. Their purpose is to incorporate the current knowledge of elementary particle physics into a tool that allows to make predictions of known processes. However, they are also used to study and predict processes beyond the current Standard Model based on theoretical considerations. Thus, Monte Carlo event generators are the bridge that links theory and experiment in the broad field of elementary particle physics.

This section intends to provide a brief introduction to Monte Carlo event generators and their applications. By doing so, it refers to the physical fundamentals introduced in chapter 1.

2.2.1 The larger picture

As state above, the objective of MC event simulation is to give accurate predictions of some physical *observable* $\mathscr{O}$ that can finally be compared with data obtained in an experiment. Generally, this observable is computed from the reconstructed four-momenta $p_1, \dots, p_n$ in the final state f, i.e., $\mathscr{O}_f := \mathscr{O}_f(p_1, \dots, p_n)$, with n being the number of "particles" (this may also be a jet or missing transverse energy) in the final state. Due to the underlying statistics of quantum mechanics, the probability of a specific final state configuration to occur is given by the differential cross-section $\mathrm{d}p_f(p_1, \dots, p_n) \propto \frac{\mathrm{d}\sigma_f(p_1,\dots,p_n)}{\mathrm{d}^3p_1\dots\mathrm{d}^3p_n}$. The cross section σ of a physical process is a detector independent quantity; however, it is related to the number of observed events N in an experiment/particle collider through the integrated luminosity $\mathcal{L}$ as per $N(t) = \sigma \int_t \mathrm{d}t'\mathcal{L}'$. Therefore, the frequent repetition of an experiment allows to measure the expectation value $\langle \mathscr{O}_f \rangle$ of the observable $\mathscr{O}_f$ based on the likelihood and the integrated luminosity according to

$$\langle \mathscr{O}_f \rangle_{\mathcal{L}} = \mathcal{L} \sum_{i \in I_f} \int_{\mathcal{V}} \mathrm{d}^3p_1 \dots \mathrm{d}^3p_n \, \frac{\mathrm{d}\sigma_f(p_1, \dots, p_n)}{\mathrm{d}^3p_1 \dots \mathrm{d}^3p_n} \mathscr{O}_f(p_1, \dots, p_n), \tag{2.4}$$

(*cf.* Equation 2.3). The dimensionality of the complicated integral in Equation 2.5 increases with the number of final state particles in an event. As a result, one relies on Monte Carlo methods to get a reasonable estimation of the expectation value. The Monte Carlo approximation of Equation 2.5 according to Equation 2.2 is therefore given by:

$$\langle \mathscr{O}_f \rangle_{\mathcal{L},N} \approx \mathcal{L} \frac{|\mathcal{V}|}{N} \sum_{j=0}^{N} \sum_{i \in I_f} \mathrm{d}^3p_1 \dots \mathrm{d}^3p_n \, \frac{\mathrm{d}\sigma_f(p_1, \dots, p_n)}{\mathrm{d}^3p_1 \dots \mathrm{d}^3p_n} \mathscr{O}_f(p_1, \dots, p_n), \tag{2.5}$$

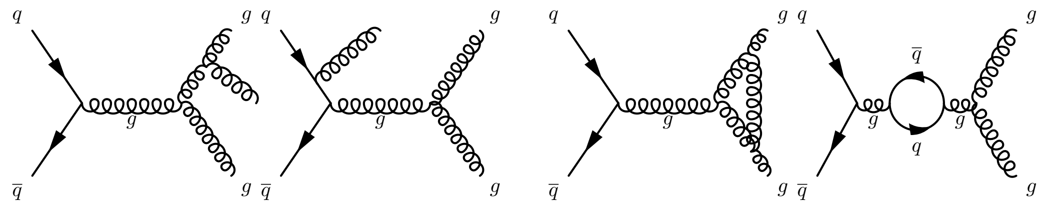
with $|\mathcal{V}|$ being the integration volume and I_f denoting the index set of final state particles. Based on the statements in Section 2.1.1, the sample mean $\langle \mathscr{O}_f \rangle_{\mathcal{L},N}$ converges to the true expectation $\langle \mathscr{O}_f \rangle_{\mathcal{L}} = \lim_{N\to\infty} \langle \mathscr{O}_f \rangle_{\mathcal{L},N}$ for infinite statistics. Following Equation 2.5, the expectation value for each observable may be computed based on the measured differential cross-section.

Now, to get a prediction for $\langle \mathscr{O}_f \rangle^{\mathrm{MC}}_{\mathcal{L},N}$ that can be compared with the experiment, the probability, i.e., the differential cross-section of a process, must be accurately predicted by theory. Within the framework of pQCD (or perturbation theory in general), the precision of the prediction is improved by including higher-order corrections into the expansion of the matrix element (see Section 1.2.3). This is known as *fixed-order expansion* in the simulation chain of MC event generators, which is the subject of the following section.

2.2.2 Fixed-order matrix element expansion

All commonly used multi-purpose event generators come with a comprehensive list of pre-implemented leading-order matrix elements equipped with the corresponding phase-space parameterization for processes up to three particles in the final state [Buckley and others, 2011] (see Equation 1.12). However, since the number of Feynman diagrams growths roughly factorial(!) with the number of final-state particles – resulting in a considerable number of terms for the squared amplitude –, the computation of higher particle multiplicities is usually done by dedicated matrix-element and phase-space generators such as, for instance, AlpGen [Mangano *et al.*, 2003], Comix [Gleisberg and Hoeche, 2008] or `MadGraph5_aMC@NLO` [Alwall *et al.*, 2014b].

There is another complication related to higher-order corrections to Equation 1.12 besides of the large number of additional diagrams that are related to whether the radiated gluon is resolvable or not – the corresponding NLO Feynman diagrams are illustrated in Figure 2.1a.



(a) Real contributions $\mathcal{R}$ (FS- and ISR)   *(b)* Virtual $\mathcal{V}$ contributions

Fig. 2.1: NLO Feynman diagrams contributing to the $q\bar{q}$ annihilation cross-section.

This problem was already addressed in Section 1.3.2 in the context of collinear and infrared safety. If the radiated gluon can not be resolved, e.g. due to finite detector granularity, the phase space integrals in Equation 1.12 are identical for LO and NLO. As it was shown before, the infrared and collinear divergences cancel if the contributions are combined. In practice, however, this requires certain *subtraction techniques* or *phase space slicing* since the divergent contributions to the NLO cross-section $\sigma_{\mathrm{NLO}} = \int_{N+1} \mathrm{d}\sigma^{\mathcal{R}} + \int_N \mathrm{d}\sigma^{\mathcal{V}}$ live in different regions of phase space, which can not be simultaneously solved by Monte Carlo.

If the partonic cross-section $\sigma^{ij\to f}_{\mathrm{N}^n\mathrm{LO}}$ has been evaluated through perturbative QCD to a fixed order n in the perturbative expansion of the matrix element, the cross-section for the scattering process can be computed according to the factorization theorem 1.16. In doing so, one has to chose the (non-physical) renormalization μ_R and factorization scales

μ_F as well as a parameterization of the PDF that matches the desired accuracy of the final cross-section calculation [Buckley and others, 2011].

2.2.3 Parton shower simulation

The theoretical background of parton shower simulation through the Sudakov form factors has been introduced in Section 1.2.5; this short paragraph focuses on the actual implementation of the showering algorithm using Monte Carlo methods. As it was shown before, the Sudakov form factor $\Delta_i(t,T)$ give the probability that a parton does *not* undergo a branching between two scales t and T

$$\Delta_i(T,t) = \exp\left\{-\sum_j \int_T^t \frac{\mathrm{d}t'}{t'} \int_z \mathrm{d}z' \, \frac{\alpha_S}{2\pi} P_{ji}(z',t')\right\} = \exp\left\{-\int_T^t \mathrm{d}\log t' \mathcal{P}_i(t')\right\}, \quad (2.6)$$

whereby Equation 1.20 has been expressed in terms of a general scale t and the marginalized splittings functions $P_{ji}(z',t') = \int_\phi^{\prime} \mathrm{d}\phi' \, P_{ji}(z',t',\phi')$. This Equation was derived based on the assumption that the splitting probability is unconditioned – which is not true due to quantum interference effects –; hence, making the simulation of parton showers a Markov process. The probability $\mathcal{P}_i(T)$ that a splitting of i occurs at scale T after the parton did not branch in the interval $(T,t]$ is given by

$$\frac{\mathrm{d}\mathcal{P}_i(T)}{\mathrm{d}t} = \frac{\mathrm{d}\mathcal{P}_{i\to jk}(T)}{\mathrm{d}t} \exp\left\{-\int_T^t \mathrm{d}\log t' \mathcal{P}_i(t')\right\} \overset{1.21}{=} -\frac{\mathrm{d}\Delta_i(T,t)}{\mathrm{d}t}, \quad (2.7)$$

whereby the first factor $\mathcal{P}_{i\to jk}(T)$ denotes the naïve probability for a splitting to occur at scale T. Conveniently, the distribution corresponds to the Sudakov form factors. The objective now is to solve Equation 2.7 for a new scale T. Formally, this can be done by sampling a random number $R \sim \mathcal{U}(0,1)$ with $\int_T^t \mathrm{d}t' \frac{\mathrm{d}\mathcal{P}_i(T)}{\mathrm{d}t} = 1 - R$ and solving the equation

$$-\int_T^t \mathrm{d}t' \frac{\mathrm{d}\Delta_i(t',t)}{\mathrm{d}t} = \underbrace{\Delta_i(t,t)}_{=0} - \Delta_i(T,t) = 1 - R, \quad (2.8)$$

so $\Delta_i(T,t) = R$. If the function $\mathcal{P}_i$ has an analytical and invertible primitive $\mathscr{P}_i$ with $\mathscr{P} = \int_T^t \mathrm{d}\log t' \mathcal{P}_i(t')$, then the solution to Equation 2.8 is simply given by $T = \mathscr{P}_i^{-1}(\mathscr{P}_i(t) - \log R)$. However, usually $\mathcal{P}$ is a complicated function such that there is no analytical solution to this problem. Therefore, the integral equation in must be solved by Monte Carlo methods. Nowadays, most shower Monte Carlos use to so-called *veto algorithm* [Buckley and others, 2011] – which is a variant of the well-known hit-or-miss Monte Carlo – to generate a sequence of evolution variables of a parton shower.

The descending order of the evolution variable t that has been used above implies final state radiation, i.e., the showering cascade continues until the energy scale of the partons is at the order of Λ_{QCD} where non-perturbative effects (the formation of hadrons) take over. However, the parton shower model and the Sudakov form factors can also be used to simulate initial state radiation (see Figure 2.1). However, in case of initial state radiation, one has to account for the fact that the additional radiation before the hard scattering process changes the energy scale of the event and thus also the parton density functions according to the DGLAP equation 1.17. The corresponding *backwards-evolution algorithm* [Sjöstrand, 1985] starts from the hard scattering process and considers the case were no collinear emission takes place $\mathrm{d}\sigma_i \propto |M_q(x)|^2 \mathrm{d}x f_{i/h}(x,t)$ as well as the opposite case, where

i does emit a collinear parton $\mathrm{d}\sigma_i \propto |M_q(x)|^2 \mathrm{d}x \frac{\alpha_S(t)}{2\pi} \sum_{jk} f_{j/h}(x/z,t) P_{j,ik}(z) \mathrm{d}z \frac{\mathrm{d}\phi}{2\pi}\frac{\mathrm{d}t}{t}$. So, the relative probability for the gluon to be *unresolved* is given by the ratio

$$\mathrm{d}\mathcal{P}_i = \frac{\mathrm{d}\sigma_{q\to qg}}{\mathrm{d}\sigma_{q\nrightarrow qg}} = \frac{\alpha_S(t)}{2\pi}\frac{f_{q/h}(x/z,t)}{f_{q/h}(x,t)} P_{q,qg}(z)\mathrm{d}z\frac{\mathrm{d}\phi}{2\pi}\frac{\mathrm{d}t}{t}, \tag{2.9}$$

whereby x/z is the momentum fraction of the incoming quark before the emission of a gluon. In analogy to the steps in Section 1.2.5, Equation 2.9 can be used to derive the Sudakov form factors for ISR

$$\Delta_i^{\mathrm{ISR}}(t,t') = \exp\left\{-\int_{t'}^{t}\frac{\mathrm{d}t''}{t''}\frac{\alpha_S(t'')}{2\pi}\int_x^1\frac{\mathrm{d}z}{z}\sum_{jk} P_{j,ik}(z)\frac{f_{j/h}(t'',x/z)}{f_{i/h}(t'',x)}\right\}. \tag{2.10}$$

2.2.4 Event simulation and event topology

The purpose of Monte Carlo event generators is to simulate high energy collisions in all process steps to the stable final-state particles that are measured in the detector. This process chain and its individual component follows physical principles and should therefore mimic the actual process that produced the event in the real world.

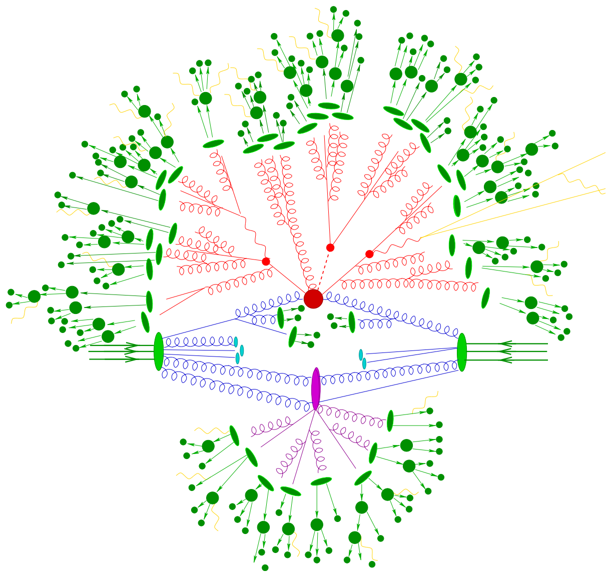


Fig. 2.2: Individual components in the simulation chain of proton-proton collisions: incoming parton densities heading towards each other with the *hard* interaction indicated by the central red blob and a second hard scattering (purple). The tree-like structure surrounding the hard interaction is bremsstrahlung simulated through parton showers. The non-perturbative regime, i.e., the transition from partons to hadrons is depicted in light-green, which is followed by the formation of excited states that further decay to final-state particles measured in the detector (adapted from Höche [2014], Fig. 3, p. 6).

The generation of an event starts with the computation of the matrix element at fixed order of perturbation theory for the underlying hard subprocess (2.2.2) with some dedicated matrix-element-generator like `POWHEG` or `aMC@NLO`. This very first step only computes the partonic final-state without any hadronization involved (*parton-level events*). To simulate the emissions of QCD radiation, the information of the parton-level event is processed by a Shower Monte Carlo, such as `Herwig6`, `Herwig++`, `Pythia6`, `Pythia8` or `Sherpa`, that simulates a cascade of showering particle up to the hadronization scale where colourless hadrons are finally formed (see "colour confinement" 1.2.2), which may further decay to stable final-state particles. With the simulation of the so-called *particle-level events*, the event generation process is finished. However, due to the imperfection of measuring devices at our disposal and the resulting limitations in the reconstruction of particles, the further simulation of detector effects is utterly important to get realistic predictions that can be compared to the measurementd at the actual detector. This simulation of detector effects is based on Monte Carlo (see Section 2.1.2). A well-known tool for detector simulations is `Herwig` that actually "[...] is a multi-purpose particle physics event generator[.]" (Bellm and others [2016]). After all those steps, the generated *reco-level events* are comparable with the measurement in the respective detector and only this detector.

Following now is a short description of the individual steps of the event generation process and how they are related to the theoretical fundamentals introduce in this chapter.

1. Hard subprocess

The first step is the aforementioned simulation of the hard subprocess – whereby the term "hard" implies processes, whose energies are large enough for perturbation theory to be applicable. 1.2.2). It starts with the simulation of hadron-hadron collisions, the underlying interaction of the fundamental partons whose momentum fraction depends on the PDF (see Figure 1.4) and the underlying energy scale of the event.

The calculation of the matrix element is a fixed-order calculation, i.e. the perturbative expansion of the scattering matrix is terminated at some fixed order. The order of perturbation, which is taken into account to compute the hard subprocess, defines the precision as well as the order of the whole event.

It should be noted, that the assumption of only one underlying hard subprocess that evolves to a final-state measured in the detector is overly-simplistic. Usually, the actual hard event is accompanied by a large number of *secondary interactions* called *pile-up*, which needs to be simulated as well.

2. Parton shower simulation

The calculation of the hard subprocess is followed by the simulation of additional radiation. Since the hard subprocess usually is associated with high energies of the final-state particles at parton-level, the accelerated particles that carry a colour charge tend to emit radiation in form of gluons (similar to the emitted photons in case of accelerated electric charges). In contrast to QED, the non-Abelian nature of QCD causes self interaction of the gluons (see Figure 1.1) which carry a colour charge by themselves. This causes QCD radiation to further emit radiation and so forth, causing a cascade of splittings until the energy of the radiated particles reaches the hadronization threshold where the particles become "soft" and the formation of colour singlet hadrons sets in.

In high energy particle physics, there are different schemes available that try to provide approximate modelling of the process described above based on the Sudakov form factors. As it has been described in Section 1.2.5, the Sudakov form factors sum the leading contribution to all orders in perturbation theory, but only in the limit of soft and collinear splittings, otherwise, the approximation fails and diverges. This in particular means, that the showering algorithms preferably add soft and collinear radiation to the partons from the hard subprocess. On the one hand, this is beneficial since the computation of the matrix element fails for soft and collinear particles; hence, the parton shower algorithm allows to fill regions in phase space that are inaccessible for the matrix element. On the other hand, the emission of hard radiation with wide angles is suppressed. This is also related to the problem of matching and merging between matrix element and parton shower since both create and overlap in the phase space and hence double count events.

3. Hadronization and decay to final-state particles

In each subsequent splitting step the respective partons decrease their energy until they enter the hadronization phase. At this energy scale (Λ_{QCD}), the perturbative approach is not valid anymore due to the large coupling constant $\alpha(\Lambda_{\mathrm{QCD}}) \sim 1$ as a consequence of the running coupling and colour confinement. Therefore, the soft or hadronic regime can not be described by first principles but relies mostly on empirical hadronization models. Usually, those hadronization models are based on the already mentioned Lund string model or the so-called cluster model.

Most of the hadrons created after the showering process are in an excited state for very short lifetime. Therefore, they will decay to stable final-state configurations with several decay products before finally being registered in the detector. This adds a layer of complexity since the enormous amount of hadron decays need to be modelled, with some of them not being well understood.

4. Detector Simulation and reconstruction

A detector that has been build to measure the final-state particles produced in hadron-hadron interactions does not measure the particles directly but their "trace" left in the material due to energy deposition in calorimeters or ionization in (Silicon) pixel cells (hits and tracks). From the measured energy, transverse momentum, track curvature, position in η-ϕ etc. the four-momentum of the particle(s) may be reconstructed. However, the measurement of the detector is not perfect since the detector has a finite granularity as well as limited acceptance in its materials. Furthermore, a detector does not cover the entire region $(\eta, \phi) \in \mathbb{R} \times [0, 2\pi)$ and therefore potentially misses some decay products.

To account for the detector effects, the actual physical processes occurring in its subcomponents are simulated with Monte Carlo methods (see Section 2.1.2). This includes, e.g., the simulation of energy deposition, ionization and much more. After this step the generated data is valid only for this particular measuring device since those simulations depend on the actual detector architecture (number of layers, material etc.) that is used in the experiment

The last step is the identification and/or reconstruction of particles in the final state for further analysis.

2.3 Types and examples of MC event generators

Loosely speaking, there are two types of event generators in high-energy particle physics: the so-called General-Purpose Monte Carlo (GPMC) event generators and the generators that are based on Matrix Element and Parton Shower matching (ME+PS). The former ones include some LO or NLO matrix elements up to a final-state multiplicity of three particles and use the parton shower algorithms to compute the cross-section for the underlying hard process including all dominant collinear radiation. As the name suggests, GPMC event generators are able to simulate the entire event including hadronization, i.e., the formation of color-singlet states as well as their decay to final-state particles through various decay modes. There are plenty GPMC event generators available. The most most prominent ones are listed in the following enumeration [Buckley and others, 2011]:

- Ariadne [Lönnblad, 1992] was the first shower Monte Carlo that implemented a *dipole cascade* to model coherent gluon emission through two colour-connected partons, which by now is used by most other GPMC event generators.
- Herwig++ [Bahr and others, 2008] is an acronym for Hadron Emission Reactions With Interfering Gluons (re)written in C++. Its unique selling point is the angular ordering of parton showers to take colour coherence effects into account. Furthermore, it provides matching at NLO as well as elaborate hadronic decay models.
- Pythia(8) [Andersson *et al.*, 1983b, Sjostrand *et al.*, 2006, 2008] is the most widely used general-purpose event generator in HEP that has been extensively used at LEP and HERA. Its main features are its enormous list of pre-implemented hard subprocesses up to three final-state partilces (higher multiplicities are archived through parton showers) as well as its interface to other matrix element generators via the so-called Les Houches interface. The parton shower simulation is based on Ariadne's dipole-approach.
- Sherpa is yet another general-purpose event generator "[...] for the Simulation of High-Energy Reactions of Particles in lepton-lepton, lepton-photon, photon-photon, lepton-hadron and hadron-hadron collisions[.]" (Gleisberg *et al.* [2008]) (the name also is a reference to the ethnic groups native to Nepal). Sherpa's philosophy is to model the actual underlying physical processes as close as possible (*bottom-up approach*).

The latter one, ME+PS, combines matrix element generator and parton shower Monte Carlo to take advantage of their particular strengths. As it was shown before, the parton shower is based on the collinear factorization that is only valid within the limit of soft and collinear radiation. The matrix element exhibits singularities in this regime where the parton shower is valide, but provides excellent results for wide-angled, hard partons in the final state. Hence, the combination of matrix element and parton shower aims to bring together the best of two worlds: the precise calculation of the matrix element at fixed order of perturbation theory that accounts for quantum interference and it allows for higher multiplicities and the parton shower in the collinear and soft limit. However, the complementary use of both methods and their naïve combination may give rise in an overlap of the phase space and therefore may cause double-counting of events – even though they operate in different regions of phase space. This is a very serious issue since the overlap distorts the predicted cross-section. The problem gets even worse if higher-order corrections (such as NLO or NNLO) are taken into account. To remove the overlap in phase space, matching and merging schemes between matrix element and parton shower must be applied.

There are several matrix event generators available. One famous representative is `MadGraph5_aMC@NLO` that has emerged from the fusion of `MadGraph` [Alwall *et al.*, 2014a, Stelzer and Long, 1994] (later `MadGraph5` [Alwall *et al.*, 2011]), which was invented in 1994, as well as `MadLoop` [Hirschi *et al.*, 2011, Hirschi, 2011], `MadFKS` [Frederix *et al.*, 2009], and `aMC@NLO` [Frixione *et al.*, 2010]. The great advantage of `MadGraph5_aMC@NLO` lies in its flexibility, i.e., the fully automated computation of Born-level and one-loop amplitudes for arbitrary processes under consideration. Furthermore, it provides a simple interface to GPMCs via Les Houche files for parton shower shower simulation and matching to NLO. Other matrix element generators are CalcHEP/CompHEP [Boos *et al.*, 2004, Pukhov *et al.*, 1999, Pukhov, 2004] or ALPGEN [Mangano *et al.*, 2003].

2.4 Challenges in MC event generation

This Chapter aimed to provide a general introduction into the topic of parton showers as they are used by the general-purpose Monte Carlo event generators in experimental particle physics. The application of parton showers has proven to be very successful in the prediction of background processes and hence is an integral part of most particle searches. Despite their unquestionable success, however, the implementation of parton showers in event generators is an approximation that is based on the requirement of collinear factorization of the splitting functions. Moreover, the computation of the matrix element of the hard subprocess is at a fixed order of perturbation theory and hence comes with uncertainties due to the premature termination of the expansion series. Furthermore, due to the combination of matrix element calculations at fixed order of perturbation theory for resolvable separations (large angles) and the collinear approximation (small angles) used for parton showers causes additional problems such as multiple counting of diagrams.

2.4.1 Double-counting

As explained above, the fixed-order matrix element events generators are based on perturbation theory and hence provide very precise results as long as the respective process is hard and the partons in the final state are well separated. In this case, the limiting factor is the premature termination of the perturbative expansion. However, the computation of the matrix element is expensive not only due to higher orders but also because of the large number of diagrams for larger multiplicities in the final state – which growths factorially with the number of external particles.

The shower Monte Carlo event generators introduced are computational cheap; however, they rely on the assumption of soft and collinear partons and do not take into account quantum interference.

It can be seen that both methods complement each other; hence, it is only reasonable to combine both in the event generation process. To combine matrix element and parton shower, one must take into account the *overlap* of the phase space of matrix element calculation and parton shower simulation that will result in double counting events. This issue, which is illustrated in Figure 2.3, already occurs in the simple case of leading order processes. This problem is addressed by the so-called merging schemes that aim to remove the overlap in phase space. A comparatively simple technique is the so-called MLM matching [Mangano *et al.*, 2002] for LO processes. However, the situation is far more complicated for NLO processes where a further overlap (besides double-counting) occurs due to the additional real emission, i.e., initial- and final state radiation. A merging scheme that accounts for this effect would be FxFx merging [Frederix and Frixione, 2012].

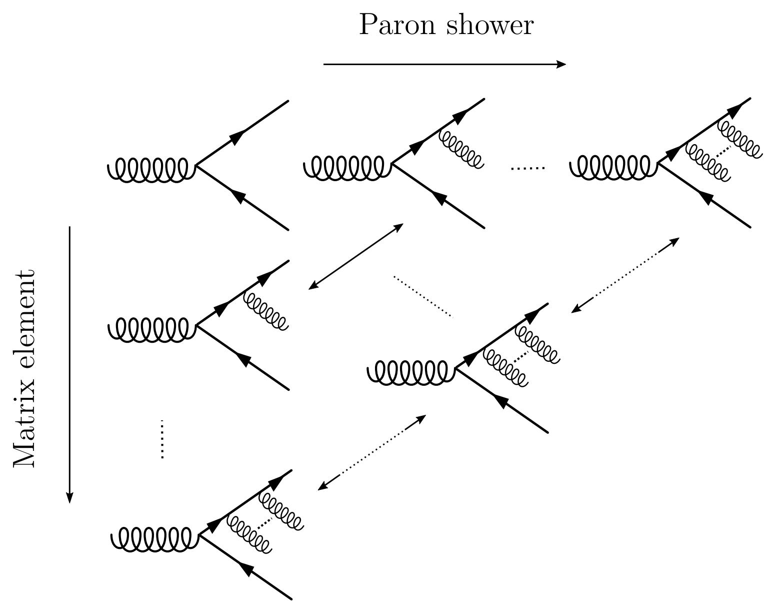


Fig. 2.3: Double counting at leading order. Along the horizontal axis, the parton shower algorithm adds radiation; in doing so, it generates diagrams which have an equivalent in leading order processes with higher multiplicities.

Part II

Chapter 3

Machine Learning

Nomen est omen – the objective of this chapter is to provide a short yet substantive introduction into the highly topical area of machine learning with an emphasis on *(artificial) neural networks* – more specifically, *deep neural networks*. The first section (3.1) serves as a description as well as a recapitulation of the historical development of neural networks and machine learning: from its modest beginnings in the mid-twentieth of the previous century to highly complex, state-of-the-art generative models that are in the focus of today's research. By doing so, the section is limited to "milestones" (according to the author's subjective opinion) that caused a significant step forward in the area of machine learning or even – to cite the words of the physicist and philosopher Thomas S. Kuhn [Kuhn, 1970] –, a *change of paradigm*. After the historical introduction, the three paradigms of machine learning, *supervised*, *unsupervised* and *reinforcement learning*, are quickly explained (3.2). The subsequent section is more specific and introduces the fundamental concepts behind artificial neural networks (3.3) as well as the two most commonly used network topologies: feed-forward (3.3.1) and recurrent neural networks (3.3.2). The fourth section 3.4 is again rather technical in its character; alongside some fundamental concepts, it introduces the essential basics to train neural networks such as the crucial loss function 3.4.1, gradient descent to optimize the network's parameters with respect to the cost 3.4.2, the backpropagation algorithm 3.4.3, and some popular regularization and normalization techniques 3.4.4 to stabilize the training routine. All the preliminaries serves as a focal point for the gist of this chapter, i.e., the introduction of generative models and their realization employing *Variational AutoEncoders* (VAEs) (3.5) Generative *Adversarial* and (neural) Networks (GANs) (3.6.2) as well as several variations. Now, equipped with the necessary tools to grasp the topic at hand, the successive topic is dedicated to a detailed discussion and comparison of the best-known adversarial models including different metrics, such as the *Jensen-Shannon* (f-)divergence or the recently proposed *Wasserstein* a.k.a. "*marth mover's*" distance.

3.1 A brief history of neural networks

The highly advanced and complex computer models, which more and more find their way into the daily life of modern societies, come with a long and rich history that goes back even further than the appearance of the very first computers – at least conceptually. The principle idea is/was to model the functionality of the (human) brain with its enormous amount of interconnections between individual neurons and the activation potential within these cells. The growing understanding of neurobiology and the processes that govern the human brain in the last three centuries therefore greatly influenced the early concepts of artifical neural networks.

At the beginning of the eighteenth century, biologists and physicists alike studied the effect of electricity on the nervous system. The Italian naturalist Luigi Galvani [Galvani, 1791] first demonstrated the presence of electrical currents in animal tissues. Later in 1871, Julius Bernstein, with the help of Emil du Bois-Reymond, deepened the knowledge of the processes in the nervous system with his "Membrane Theory of Electrical Potentials" [Bernstein, 1871], which is considered to be the first description of the action potential. Shortly thereafter, the Spanish neuroscientist and pathologist Santiago Ramón y Cajal [Ramón y Cajal, 1888] reported the first detailed anatomy of the nervous system, which later inspired scientists to adopt and apply those novel insights to mathematical and computer models.

The idea to model the functionality of the brain with its interconnections of neurons – a term that was first introduced by Heinrich W. G. von Waldeyer-Hartz [von Waldeyer-Hartz, 1891] – and hence the beginning of artificial neural networks is often taken to be the research article "A Logical Calculus of Ideas Immanent in Nervous Activity" of Warren McCulloch and Walter Pitts in 1943 [Mcculloch and Pitts, 1943]. With the starting gun being fired, it did not take long until the concepts introduced by McCulloch and Pits received the attention of the scientific community. The '40s of the previous century also had a profound influence on brain theory, whereby special emphasis should be placed on Donald Hebb's "Organization of Behavior: A Neuropsychological Theory" in particular the therein introduced "[.] Neurophysiological Postulate" (also known as *Hebb's rule*) that describes the pre-post synaptic enhancement that is essential for the process of "learning" [Hebb, 1949]. This concept was then translated to the artificial McCulloch-Pitts neuron by weighting each input. The gathered knowledge in neurobiology and the (at this time) hypothetical neural network models can be considered as important preparation work for the fast development of the field in the 1950s that was driven by the advancement of computers with increased computational power as well as significantly increased memory capacities.

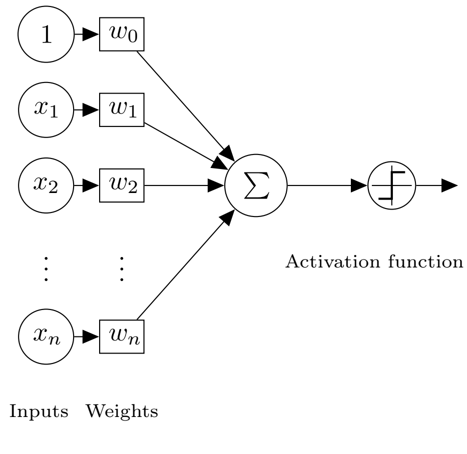


Fig. 3.1: The perceptron according to Frank Rosenblatt with bias term.

The discoveries and scientific research by Warren McCulloch and Walter Pitts, Hebbs *et al.*

culminated in Frank Rosenblatt's *perceptron* in 1962, which can be considered as the first artificial neural network in a modern sense. Frank Rosenblatt introduced the perceptron in his famous book "Principles of Neurodynamics" [Rosenblatt, 1962]. The perceptron is a very simple model that consists of a fixed number of inputs $\{x_1, x_2, \dots x_n\}$ (Rosenblatt only used five at the time) that are weighted with a real number $\{w_1, w_2, \dots w_n\}$ and results a weighted input $\{w_1x_1, w_2x_2, \dots w_nx_n\}$ with $x_k, w_k \in \mathbb{R}$ to the node (artificial neuron) where the inputs are added together $\sum_{k=1}^{n} w_kx_k$. Inspired by McCulloch and Pits' work, the artificial neuron had a binary activation according to the threshold value $\theta \in \mathbb{R}$ with $\sigma(x) = \Theta(x - \theta)$, where Θ is the Heaviside step function.
This simple model introduced by Rosenblatt was already quite successful; though, it was limited to a restricted family of functions it could approximate. For example, Rosenblatt perceptron is incapable representing a simple logical XOR (eXclusive-OR) or XNOR (eXclusive-NOR) gate due to its limitation to only learn a single decision boundary. Rosenblatt was not aware of these limitations; he – overly enthusiastically – proclaimed: "Given an elementary [..]perceptron, a stimulus world W, and any classification $C(W)$ for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in a finite time; then beginning from an arbitrary initial state, an error correction procedure will always yield a solution to $C(W)$ in a finite time [. . .]" (Rosenblatt, 1962). This initial elation caused the machine learning community to completely exaggerate the potential and applicability of neural networks at that time – disappointment and disillusionment was the consequence. In 1969, seven years after Rosenblatt's publication, however, Marvin Minsky and Seymour Papert published the essay "An Introduction to Computational Geometry" [Minsky and Papert, 1969], which drew the attention to the inherent limitations of the perceptron and its restrictions to only learn linearly separable functions (the aforementioned XOR and XNOR gate is a simple example for a non-linearly separable function). The paper by Minsky and Papert had put the breaks on the initial euphoria of the novel method introduced by Rosenblatt and almost caused a complete standstill of research in the area until the beginning of the '80s.

After the initial hype in the '60s, the discipline of neural networks fell into a deep sleep. It was not until the year 1982, that the area awoke from the dark ages; rose out of the ashes of its eventful past in manifold dazzling colours, like the phoenix or the Egyptian Bennu. The age of machine learning is yet to come. The person triggering the renaissance of neural networks was the American physicist John Hopfield of Caltech. In his directive essay "Neural networks and physical systems with emergent collective computational abilities" [Hopfield, 1988], he provided a structured and systematic analysis of of the potential as well as the limitations of a neural network at that time. In 1985, he, Hopfield, in cooperation with D. W. Tank published another important paper: "'Neural' computation of decisions in optimization problems" [Hopfield and Tank, 1985].

The 80's mark a turning point and a rapid development of the field of machine learning boosted by the essay of Hopfield and the political circumstances of the cold war and the fear of the USA to be outperformed by other countries in this promising technology, the development in machine learning accelerated rapidly. In 1986 another method experienced its revival: a process called *backpropagation*. The backpropagation (automatic differentiation) algorithm was first introduced by Seppo Linnainmaa in 1970 [Linnainmaa, 1970]; however, the method was not appreciated until the year 1986 when David Rumelhart, Geoffrey Hinton, and Ronald Williams published a paper that used "[l]earning representations by back-propagating errors" [Rumelhart *et al.*, 1986]. This algorithm was the prerequisite to train the large complex networks that emerged at that time.

Also, the theoretical understanding of the mechanisms and the mathematical fundamentals grew, enabeling great steps forward. The paper by George Cybenko "Approximation by Superpositions of a Sigmoidal Function" from 1989 [Cybenko, 1989] is considered to be the first proof of the so-called *Universal Approximation Theorem* (UAT) for the special case on sigmoid activation functions solely.

Theorem (**UAT**; Cybenko, 1989, Thm. 1, p. 306)**.** *Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T + \theta_j)$$

are dense in $C(I_n)$ [space of real-valued continuous functions]. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \epsilon \quad \textit{for all } x \in I_n.$$

(Proof: see Cybenko, 1989, p. 306)

Two years later, Kurth Hornik provided another, more general proof of the universal approximation theorem [Hornik, 1991]. The UAT can be considered as the theoretical basis of neural networks that tells us that – in principle – there are no limitations in potential applications – except the bounds of reality of course. Under the assumption of an arbitrary complex model, a neural network can approximate, respectively, learn *any* continuous function. The continuous growth in complexity of today's networks come with an enormous amount of trainable parameters (weights) goes hand in hand with an increased demand on computer performance regarding the requirements of CPU power and RAM storage. For this reason, the complexity and the performance of neural network models scale with the enhancement of computing power, similar to parallelization of mathematical operations on modern GPUs.

With the universal approximation theorem at the beginning of the '90s, neural networks have established themselves. In the following decades, the field of machine learning and artificial intelligence have made considerable progress and numerous discoveries. Today, artificial intelligence is an integral part of our daily life – but not without controversy. It is hardly possible to foresee the implications of the increasing presence of machine learning methods and big-data analysis within the context of the progressive digitization and its repercussions on our daily life. However, it is very likely that this age marks the beginning of the era of "artificial intelligence".

The increased interest at the beginning of the '90s caused a cascade of numerous discoveries. One milestone was followed by another one so that it becomes difficult to distil the major steps that caused significant progress of the field as a whole. The most significant of these were – according to the suthor's opinion –: the "discovery" of recurrent artificial neural networks by John Hopfield [Hopfield, 1988] and recurrent neural networks derived therefrom, the Long Short-Term Memory (LSTM) (recurrent neural) networks introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997 [Hochreiter and Schmidhuber, 1997a] (the development of LSTM, which are much better at capturing long-term dependencies, were the main driver for the great success of RNNs in the applications of speech recognition); the invention of support vector machines as an algorithm for optimal margin classifiers [Bos, 1992]; reinforcement learning based on Markov decision processes [Watkins and Dayan, 1992] (for an overview, see [Li, 2017]); the realization of generative models by means of generative *adversarial* neural networks [Goodfellow *et al.*, 2014] and autoencoders (conceptually introduced by Ballard [Ballard, 1987]).

3.2 Trinity of machine learning

The field of machine learning is usually subdivided into three conceptually very different learning paradigms: *supervised* learning, *unsupervised* learning, and *reinforcement* learning.

Supervised learning is the oldest paradigm in machine learning used to train neural networks. In a supervised learning task, the input $\boldsymbol{x}$ as well as the expected output $\boldsymbol{y}$ are both available $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$ to train the network. Hence, the predictions of the neural network $\hat{\boldsymbol{y}}$ after a certain number of training steps are compared with the (true) label known from the data set. The discrepancy between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ is then used as a measure of similarity to update the network's weights accordingly to improve the prediction in the next iteration. Well-known examples for supervised learning would be simple linear and logistic regression, non-linear regression, (multi-class, multi-label) classification etc.

The second pillar of machine learning is *un*supervised learning. As the name suggests, in case of unsupervised learning the true label $\boldsymbol{y}$ is withheld or more likely inaccessible during training. Due to the absence of this information, the network has to learn and identify patterns in the data unshephered and unsupervised (an example for an unsupervised learning task would be, for instance, the identification of substructure within a large jet). A popular example would be a clustering problem where the neural network is supposed to learn the inherent structure in the data and group the elements accordingly.

The last paradigm in machine learning is reinforcement learning, which is conceptually very different from the two concepts introduced above. Reinforcement learning labels a methode, in which an *agent* in a certain *state* that performs *actions* within an *environment* according to some *policy* with the objective to maximize its *reward*.

Within the scope of this thesis, only the first two concepts are relevant. However, reinforcement learning could also have some possible applications in the context of the problems studied in this work (a proposal is given in at the end of this report).

3.3 Artificial neural networks

The basic building blocks of an *artificial neural network* have already been introduced in Theorem 3.1. More Formally, a neural network can be described in terms of a triple $(\mathcal{N}, \mathcal{V}, w)$, whereby $\mathcal{N}$ corresponds to the set of *neurons*, whereat the *connection* between the pairs of neurons $(i, j) \in \mathcal{V}$ is given by the set $\mathcal{V} := \{(i,j) | i, j \in \mathbb{N}^{\geq |\mathcal{N}|}\}$. The *weights* are defined by the function $w : \mathcal{V} \to \mathbb{R}$ that gives the strength $w_{ij} := w((i,j))$ of the connection $(i,j) \in \mathcal{V}$. Hence, a weight $w_{i'j'} = 0$ cuts of the connection between neuron i' and j' (no stimulus). Furthermore, neural networks need a *propagation function* T_w that receives the output of several neurons $\{i_1, i_2, \cdots i_n\}$ with the output $(o_{i_1}, o_{i_2}, \cdots o_{i_n})$ connected via weights $\{w_{i_1,j}, w_{i_2,j} \cdots w_{i_n,j}\}$ to one neutron j and transforms it to $T_{w_j} := T_w(o_{i_1}, \cdots o_{i_n}, w_{i_1,j}, \cdots w_{i_n,j}) \in \mathbb{R}$. In principle, there is no restriction on T_w besides differentiability (which is required by the backpropagation algorithm). In practice, however, in almost all practical applications, the so-called *weighted sum* is used. This weighted sum is an affine transformation between outputs of the neurons and the weights

$$T_{w_j} = \sum_{i \in \{1, \cdots, n\}} o_i w_{i,j} + b_j, \tag{3.1}$$

where b_j is the so-called learnable *bias* $b_j \in \mathcal{B}$ term [Kriesel, 2007][1].

The neural network f, i.e., the function it learns defined above, is still rather limited in its applications since it is restricted to affine functions solely due to the chosen form of the propagation function. Hence, the network is not able to approximate non-linear functions since the concatenation of linear ($b = 0$) functions is linear again (homomorphism).

$$f = \bigcirc_{j\in\mathcal{N}} T_{w_j}^{b=0} = \bigcirc_{j\in\mathcal{N}} \left(\sum_{i\in\{1,\cdots n\}} o_i w_{i,j} \right) = \sum_{i\in\{1,\cdots n\}} \left(o_i \bigcirc_{j\in\mathcal{N}} w_{i,j} \right) \tag{3.2}$$

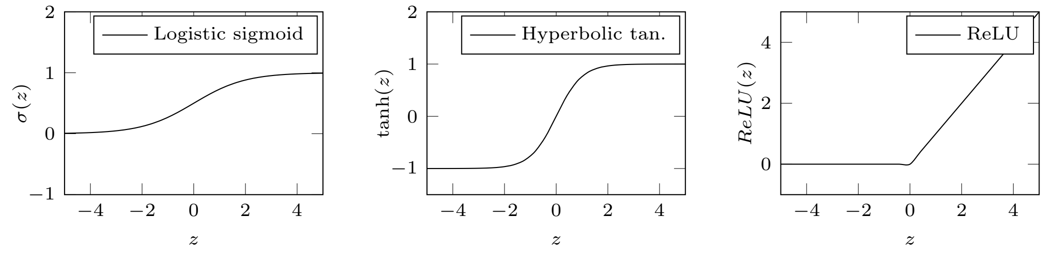
A neural network defined like Equation 3.2 would be nothing but a simple linear regression model. To improve the approximation power of general *feed-forward* neural networks, the output of the propagation function T_{w_j} is further processed by what is called the *activation function* (also known as *transfer function*) σ

$$\sigma_j = \sigma(T_{w_j}, \alpha_1, \cdots, \alpha_n), \tag{3.3}$$

with $\alpha_1, \cdots, \alpha_n \in \mathbb{R}$ being some adjustable *hyperparameters* (i.e. fixed parameters not learned by the network). The output of the activation function may also be time dependent $\sigma_j(t)$ and hence depend on previous activation (see Section 3.3.2). There exists a huge variety of different activation function and choosing the "most appropriate" one is part of the model's adjustement. For example, Rosenblatt's perceptron introduced in Section 3.1 had a threshold function Θ. The choice of the activation function also depend on the network's task whether it is, e.g. a regression or a classification task. Thus the latter one expects probabilities which requires an appropriate activation function that maps the network's output to the interval $[0, 1]$. The most popular activation functions used today are the *sigmoid*, *Fermi logistic function* $\sigma(z) = \frac{1}{1+e^{-z}}$ (as it used for the proof of the UAT 3.1), the *hyperbolic tangent* $\cosh(z) = \frac{1}{2}\left(e^z + e^{-z}\right)$, and the so-called *Rectified Linear Units* (ReLU) $\mathrm{ReLU}(z) = \max(0, z)$ [Maas, 2013, Nair and Hinton, 2010], whose graphs can be seen in Figure 3.1 (for a comprehensive list of activation functions see, e.g., Nwankpa *et al.*, 2018). ReLUs are the most frequently encountered activation functions in today's neural networks although it only has a non-linearly in one point. Compared to other saturating activation functions, however, ReLUs do not suffer from the vanishing or exploding gradient problem. Furthermore, the derivations needed for the backpropagation algorithm are simple to compute and hence significantly improve performance. On the other hand, ReLUs tends to result in sparse gradients. This effect can be reduced by using so-called Leaky ReLUs $\ell\mathrm{ReLU}(z) = \max(-\epsilon z, z)$ that adds a small slope $\epsilon \in \mathbb{R}^{>0}$ for negative arguments [Xu *et al.*, 2015].

Summarizing the statements above, it can be said that neural networks are a composition of simple linear or affine transformations T and non-linear (activation) functions a with trainable weights w which are the interconnections between the network's neurons. These are the main components that *all* neural networks have in common. However, there exists a large number of different topologies or design concepts, i.e., the "geometrical" arrangement

[1] I felt rather uncomfortable with this approach from the very first time; not because it lacks any logic but because it imposes a strong constraint on the complexity of the neural network and hence limits the space of possible functions from the start. From a historical perspective, this definition is reasonable since it allows for effective calculation of derivatives in the backpropagation algorithm (see Section 3.4.3) and hence meets the limited computer resources available at that time. It would be interesting – insofar this has not already been done – to study the effect of different propagation functions that possibly even take into account vector fields.



Plot 3.1: The most frequently used activation functions σ in machine learning: sigmoid, hyperbolic tangent, and Rectified Linear Units.

of the weights and its interconnections in the network. The following section introduces some of the most known and widespread network topologies.

3.3.1 Feed-forward neural networks

The by far most common form of a network topology is the so-called *feed-forward* design. In this conservative topology, the network's neurons are grouped into different layers. A distinction is made between the *input* layers, *hidden* layers. and the *output* layer. The input layer is the outermost level of a neural network that does not perform any computations on the data but serves as the "interface" between the outside world and the network. The hidden layers are the enclosed level of the neural network that is not directly connected to the outer world. All layers besides the input and the output layers are referred to as *hidden*. Within the hidden layers, the actual mathematical operations are performed that transform the data from layer to layer. Last but not least, the final layer is the output of the neural network that performs some final computations and returns the processed input data.

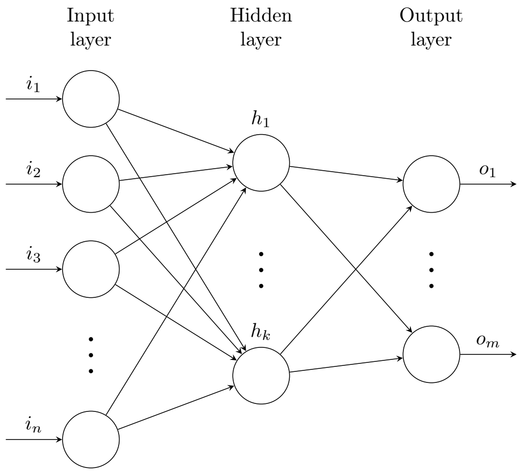


Fig. 3.2: "Shallow" feed-forward neural network with n inputs, one hidden layer with k neurons, and m outputs. The trainable weights of the networks are represented by the connections between the individual neurons.

The arrangement of weights and nodes (respectively neurons) can be summarized in what is known as a *directed graph.* Figure 3.2 shows a directed graph for a very simple feed-forward neural network with only one hidden layer (also referred to as "shallow" neural network.). As explained above, a feed-forward neural network is a composition of simple linear or affine transformations $T_w^{(l)} : \mathbb{R}^n \to \mathbb{R}^m$ and non-linear (activation) functions $\sigma^{(l)} : \mathbb{R}^n \to \mathbb{R}^n$ (with l denoting the lth layer in the network) that are arranged in layers without any feedback (loops), i.e., the output of any layer does not affect that very same layer. With all those elements and ingredients, a feed-forward neural network can be written as:

$$f(\mathbf{x}) = \underbrace{\left[\sigma^{(L)} \circ T_w^{(L)}\right]}_{\text{output layer}} \circ \underbrace{\left[\sigma^{(L-1)} \circ T_w^{(L-1)}\right] \circ \cdots \circ \left[\sigma^2 \circ T_w^{(2)}\right]}_{\text{hidden layers}} \circ \underbrace{\left[\sigma^{(1)} \circ T_w^{(1)}\right]}_{\text{output layer}} (\boldsymbol{x}), \tag{3.4}$$

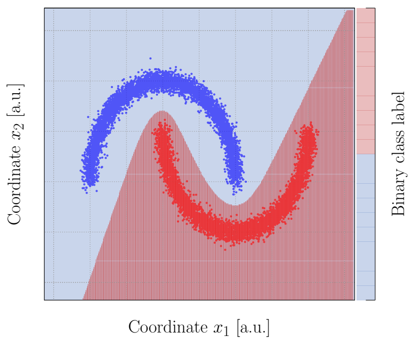
with $\boldsymbol{x}$ denoting the input to the input layer. So the output $f^{(l)}(\mathbf{x})$ of the lth layer is given by

$$f_w^{(l)}(\mathbf{x}) = \begin{cases} \boldsymbol{x} & : l = 0 \\ \left(\left[\sigma^{(l)} \circ T_w^{(l)}\right] \circ f^{(l-1)}\right)(\boldsymbol{x}) & : l > 0 \end{cases}. \tag{3.5}$$

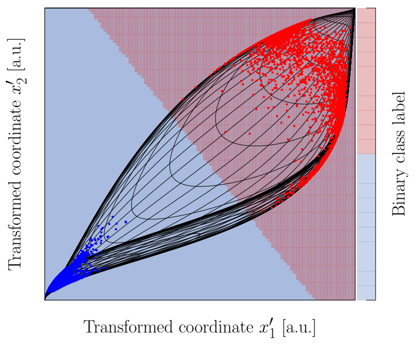
With the definition of a feed-forward neural network according to Equation 3.4, the system is mathematically well defined by means of its trained weights (see Section 3.4).

In case of discrimination or a multi-class classification task, the network needs to assign a probability to each label that maps it to class with a certain probability. This adds the constrain $\sigma^{(L)} : \mathbb{R}^n \to [0,1]^{n_c}$ on the image set of the last layer, whereby n_c denotes the number of classes (in practice this restriction is enforced, e.g., by a *softmax* layer $\sigma(\mathbf{x})_j = e^{x_j} / \sum_{k=1}^{n_c} e^{x_k}$).

Considering a very simple neural network with two input nodes, two hidden layers (three and two nodes each) and two outputs that correspond to the classification probability of two classes (binary classification: red and blue). Due to the choice of the propagation function to be linear or affine, the classification network tries to learn a new representation of the data (by means of a (homotopic) coordinate transformation) in each layer that allows for a separation of the transformed input by a hyperplane (linear separation). In order to visualize the new representation of the data in the second hidden layer of the network, the input data is forward passed form the first to the second layer $x^{(2)} = \sigma^{(2)}(T_w^{(2)}(\sigma^{(1)}(T_w^{(1)}(\boldsymbol{x}))))$. (The architecture of the network is very primitive; hence, its complexity and capability is highly limited. It was chosen such that the transformed coordinate system can be visualized easily.)



(a) Raw data

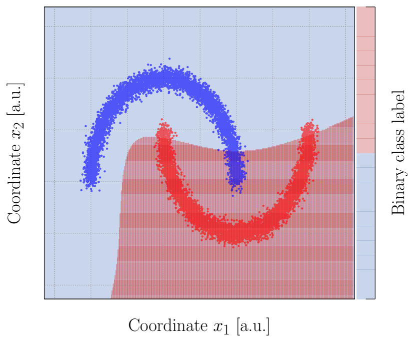


(b) Transformed data

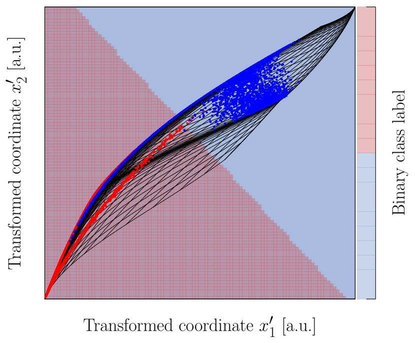
Plot 3.2: Discrimination boundary, which was learned by the neural network, for the raw data (a) and its new representation in the transformed coordinate system of the second hidden layer (b).

Figure 3.2 shows the learned classification boundary of the neural network (the training has been terminated after the classification accuracy reached 100 %). In configuration space $\boldsymbol{x}$, the data points are separated by a complex, non-linear decision boundary. In the transformed representation $\boldsymbol{x}'$, however, the data points are transformed in a complicated way such that they can be separated by a simple hyperplane. So, the network learns a new representation of the data (through a complicated coordinate transformation) in which the classification is simple.

The network may very well fail to learn an appropriate transformation that allows for a linear separation (e.g. due to the architecture that limits the complexity of the network, the amount of training data available, training steps or (most likely) an inopportune configuration of hyperparameters.) Figure 3.3 gives an example of a failed attempt to learn a decision boundary due to a purposely il-chosen setup.



(a) Raw data



(b) Transformed data

Plot 3.3: Failed classification due to inopportune set of hyperparameters.

As shown in the this section, discriminative models (or feed-forward neural networks with softmax output) try to learn a complex transformation that allows for a linear separation of the transformed data in the representation of the penultimate layer. This property does not come as a surprise since it is imposed by the requirement alternating sequence of a linear (or affine) transformation T_w followed by a non-linear activation σ – as critically commented[1]. One may think of a more general transformation than T that allows for a better transformation of the underlying manifold.

3.3.2 Recurrent neural networks

While feed-forward neural networks assume time-*independent* data, many of today's learning tasks, actually have to deal with sequential data that changes over time. A natural example would be predicting stock prices, composing a piece of music or, to mention a more prominent and relevant example, (natural) language processing as it is needed for translation, speech recognition or text generation. In general terms, it can be said that the feed-forward neural networks reach their limits in cases where the input data is varying in its length and highly time-correlated as they learn relationships between input features. To this end *recurrent neural networks* have been invented in the 1980s of the previous century (see Section 3.1).

Between their invention in the '80s and their breakthrough years later lies a long period of hibernation. This lack of interest was caused by serious problems regarding the stability as well as computational costs due to a large number of weights of the model. But, let's start from the beginning.

Classical Recurrent Neural Networks (RNNs) and feed-forward neural networks are similar in many aspects regarding the composition of weights as well as the arrangement of nodes in levels of layers. In case of recurrent neural networks, however, the input $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ and/or the output $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T)$ of the network is a sequence in time, e.g. a list of words, notes, particles etc. In the case of feed-forward neural networks, the output of the model was solely given by the input (for the current configuration of weights). Recurrent neural networks, however, add another level of complexity by including the network's previous state(s) $\boldsymbol{h}_{t-1}$. Hence, the current state $\boldsymbol{h}_t$ is a function of the previous one as well as the input and is defined as

$$\boldsymbol{h}_t = a\left(\boldsymbol{W}^{\mathrm{hx}}\boldsymbol{x}_t + \boldsymbol{W}^{\mathrm{hh}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{h}}\right), \tag{3.6}$$

with $\boldsymbol{w}^{\mathrm{hh}}$ being a matrix that gives weights to the previous state (recurrent weight). For $\boldsymbol{w}^{\mathrm{hh}} = 0$ the "memory" is removed and the well-known feed-forward network is recovered (see Equation 3.1). The output of the network for time step t is then given by

$$\hat{\boldsymbol{h}}_t = \mathrm{softmax}\left(\boldsymbol{W}^{\mathrm{yh}}\boldsymbol{h}_t + \boldsymbol{y}\right). \tag{3.7}$$

The two Equations 3.6 and 3.7 specify all necessary computations for each time step of the network. The dynamics of the network is shown in Figure 3.3 for different time steps t. Figure 3.3 depicts an example of a very simple recurrent neural network with input $\boldsymbol{x}_t$, output $\boldsymbol{y}_t$ and several hidden nodes $\boldsymbol{h}_t^1, \ldots, \boldsymbol{h}_t^n$ that itself build a feed-forward network. As it can be seen in Equation 3.6 and Figure 3.3, the information of the previous state of the network $\boldsymbol{h}_{t-1}$ is passed forward in time and hence affected the current state $\boldsymbol{h}_t$. To train the weights of this network, the aforementioned backpropagation algorithm (see Section 3.4.3) is applied across the time steps. This modification is known as *backpropagation-through-time* [Werbos, 1990].

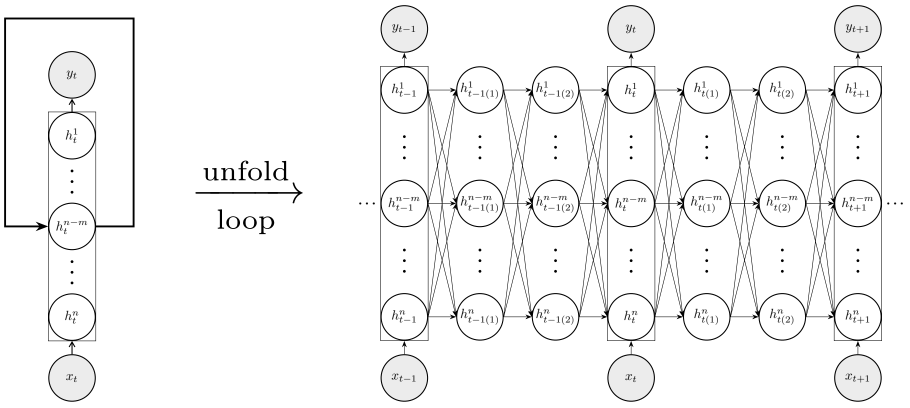


Fig. 3.3: A recurrent neural network as a loop diagram (*left*) and its unfolding in time sequences (*right*).

The "naïve" recurrent network introduced above is conceptually very simple and intuitive, but, unfortunately, it is doomed to fail. The instability of this family of networks troubled scientist for years. The inherent problem lies in the already mentioned training by backpropagation-through-time that requires the unfolding of the network in time. Depending on the number of time steps, those unfolded networks can be very deep. If gradients are passed back through many time steps, they tends to grow (exploding gradient) or vanish (vanishing gradient). This phenomenon is also known for very deep feed-forward neural networks. The solution to this problem, which leveraged this feed-back topology, was the *Long Short-Term Memory* (LSTM) RNNs [Hochreiter and Schmidhuber, 1997b].

The basic ingredient is a conventional recurrent network as explained above; however, with each node in the hidden layers being replace by what is called a *memory cell* as one is shown by Figure 3.4.

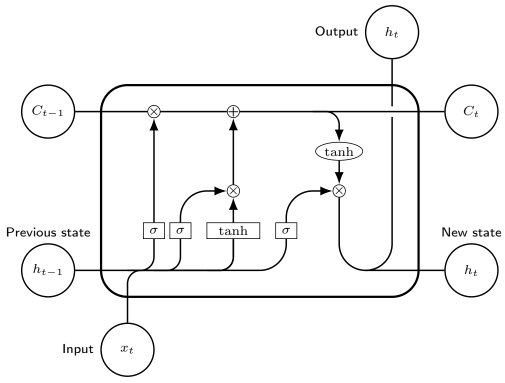


Fig. 3.4: An LSTM cell and its weighted gates.

Simple recurrent neural networks only possess an long-term memory in form of their weights. The LSTM network, however, introduces a Short-term memory utilizing ephemeral activations. The heart of the LSTM network, the memory cell, consists of simple nodes and a specific patterns that build *gates* that serve a particular purpose, which are explained below. To gain a better understanding of LSTM cells and the internal mechanism at work, the following list give as brief step-by-step introduction to the individual components shown in Figure 3.4.

- In the very first step the current input $\boldsymbol{x}_t$ and the previous state $\boldsymbol{h}_{t-1}$ are concatenated and passed through what is called the *forget gate* (introduced by Gers *et al.* [2000])

$$f_t = \sigma\left(\boldsymbol{W}\boldsymbol{x}_t \circ \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right). \tag{3.8}$$

- In a second step, the network "decides" which information to store within the internal memory of the cell. This is done in two parts: the so-called *input gate* decides which values to update (gives weight to features). A second layer utilizes hyperbolic tangent to assign positive or negative weights

$$i_t = \sigma\left(\boldsymbol{W}_i\boldsymbol{x}_t \circ \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right), \tag{3.9}$$

$$\widetilde{C}_t = \tanh\left(\boldsymbol{W}_C\boldsymbol{x}_t \circ \boldsymbol{h}_{t-1} \circ + \boldsymbol{b}_C\right). \tag{3.10}$$

- To update the state of the cell C_t, the previous state C_{t-1} is multiplied by the output of the forget gate f_t element wise (remove features); furthermore, the product $\widetilde{C}_t \otimes i_t$ is computed, i.e, how the new state is updated (no update if $i_t = 0$ or $\widetilde{C}_t = 0$)

$$C_t = \left(C_{t-1} \otimes f_t\right) \oplus \left(\widetilde{C}_t \otimes i_t\right). \tag{3.11}$$

- The final step is to produce the actual (filtered) output that is the input to the subsequent cells. The concatenated data $[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}]$ is passed through a sigmoid layer that filters the output (see Plot 3.1 in the leftmost position). The next operation is performed on the updated cell state C_t that is passed through a hyperbolic tangent. The two component are then multiplied

$$o_t = \sigma\left(\boldsymbol{W}_o[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o\right), \tag{3.12}$$

$$h_t = o_t \otimes \tanh\left(C_t\right), \tag{3.13}$$

  and the described procedure repeats for the next LSTM cell [Lipton *et al.*, 2015].

The LSTM layer described above represents a conventional architecture. Besides of this one, a wide variety of different topologies exists such as the one proposed by Gers and Schmidhuber [2000], bidirectional RNNs [Schuster and Paliwal, 1997], the Gated Recurrent Unit (GRU) [Cho *et al.*, 2014] or the so-called Deep Gated RNNs [Yao *et al.*, 2015], to name but a few.

3.4 Training of neural networks

section 3.3 introduced the basic concepts of neural networks regarding their mathematical formulation in terms of weights, activation and propagation functions, as well as different topologies in particular feed-forward (3.3.1) and recurrent neural networks (3.3.2). This section gives a short introduction to the actual training procedure, in particular, on how to update the model's parameters through gradient descent (3.4.2) and how to efficiently compute gradients (3.4.3). Furthermore, the elementary concept of a loss function in supervised and unsupervised learning is covered in this section.

3.4.1 The loss function

In order to quantitatively measure the inconsistency between the predicted values by the neural network $\hat{\boldsymbol{y}} = f_\theta(\boldsymbol{x})$ (now denoting the network's weights by $\boldsymbol{\theta}$) and the actual *label* $\boldsymbol{y}$ (available only in case of (semi-)supervised learning) for the training data $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$ one needs to define a *loss function* $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$. This loss function provides an estimate of the network's approximation error; hence, it should be defined to be a non-negative quantity ($\mathcal{L} \geq 0$), with a decreasing loss corresponding to reduced approximation error. Furthermore, the loss must be a differentiable function to get meaningful derivatives needed in the backpropagation and gradient descent algorithm to update the network's weights (the conceptually very different paradigm of reinforcement learning also allows for non-differentiable loss functions).

There are plenty of different loss functions available, most of them serving a particular purpose, e.g. whether the objective is a regression or a discrimination task. A prominent example for a loss function in a simple regression task would be the *Mean Squared Error* (MSE), $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$, while in a classification task the usual choice would be the *binary cross-entropy* (multi-class cross-entropy for multi-class classification), $\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})\right]$.

At this point, there is nothing to be gained from providing a long enumeration of different loss functions and their properties. Furthermore, the loss functions that are used in this thesis are much more complicated and therefore need a dedicated paragraph. But one should keep the loss function and its role in the training process of neural networks in mind: the loss function is a measure of similarity between predictions of the neural network and the training data and hence is utterly important in the training process (in case of generative models, the loss is a measure of similarity between two probability density functions). It is the fundamental quantity to compute the corrections to the network's weights using gradient descent and backpropagation, which is subject to the next sections.

3.4.2 Gradient descent and optimizers

With the loss function at hand, the learning objective of neural networks is to find a configuration of weights/parameters, which define the model, that minimize $\mathcal{L}$. If $\mathcal{L}$ is non-convex, there is no guarantee that the optimization procedure will yield a *global* minimum of the loss function. Besides that, it is not expected that the minimization will result in a perfect configuration of weights since exact optimization is NP-hard.

Most optimization algorithms used in machine learning today are based on a *first-order* iterative optimization algorithm called *gradient descent.* The objective is to update the models parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, i.e., the weight, in the opposite direction of the gradient – which points in the direction of the steepest slope – of the cost function[2] $\nabla_\theta \mathcal{L}(\theta)$. The most simple form of such an algorithm requires one additional, adjustable parameter, i.e., the so-called learning rate $\alpha \in \mathbb{R}^{>0}$ that corresponds to the step size taken towards the (local) minimum.

[2]Through this thesis the terms *loss*, *cost* and *objective* function are used interchangeably.

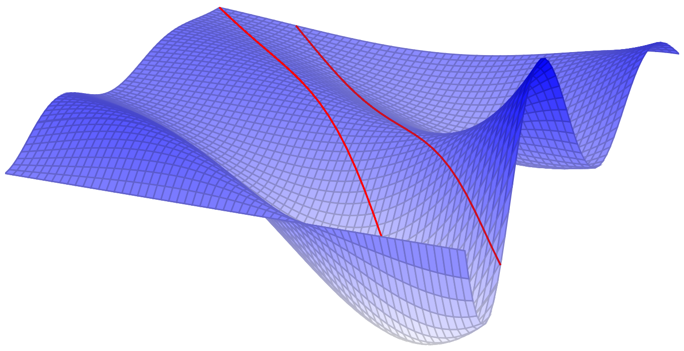


Fig. 3.5: Fictional contour of a complicated non-convex loss function $\mathcal{L}(\boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \mathbb{R}^N$ for a two-dimensional subspace spanned by the weights ($\mathbb{R} \times \mathbb{R} \subset \mathbb{R}^N$). The trajectories represent two possible paths of the gradient descent algorithm (different learning rates and/or initial conditions) to reach the (local) minimum in this subspace.

It is important to internalize that all optimization algorithms based on gradient descent are at *first-order*. While the gradient of the loss function only measures the change of the slope, the second-order derivative provides information about the direction based on the curvature of the cost's error surfaces. So, the restriction to first-order gradients is a significant limitation. An example of an optimization algorithm that also uses second-order derivatives would be Newton's method. However, computing second derivatives is "expensive" and often computationally intractable as for evaluating the Hessian matrix one needs to compute N_θ^2 derivatives, whereby N_θ is the number of weights in the network. The networks presented in this thesis have a total number of parameters in the order of approximately one million $N_\theta \sim \mathcal{O}(10^6)$. In other words, the second-order optimization algorithm would have to compute, roughly, one *trillion* ($\mathcal{O}(10^{12})$) derivatives for **one** iteration step! This is unfeasible for high-dimensional data sets; hence, the algorithms rely on first-order approximations.

Furthermore, a distinction is made between three different versions of gradient descent based on the information that is used to compute the gradients. *Batch gradient descent* computes the gradients for the entire data section and updates the parameters weighted by the constant and global learning rate

$$\boldsymbol{\theta} \to \boldsymbol{\theta} - \alpha \nabla_\theta \mathcal{L}(\theta). \tag{3.14}$$

This optimization method might be intractable for large data sets that does not fit into memory at once. The second variant is known by the name *stochastic gradient descent.* It is the other extreme and performs an update of the training parameters for each training example

$$\boldsymbol{\theta} \to \boldsymbol{\theta} - \alpha \nabla_\theta \mathcal{L}(\theta; x^{(i)}, y^{(i)}), \tag{3.15}$$

which might cause redundant computations and usually results in very volatile losses. Finally, *mini-batch gradient descent* reaches a compromise among the two previous approaches by performing an update of the model's weights for each *mini-batch* consisting of n_{mb} samples

$$\boldsymbol{\theta} \to \boldsymbol{\theta} - \alpha \nabla_\theta \mathcal{L}\left(\theta; x^{(i:i+n_{\text{mb}})}, y^{(i:i+n_{\text{mb}})}\right). \tag{3.16}$$

This one usually provides more stable convergence and hence is the method of choice – also in this thesis.

As has been described above, gradient descent only uses the information of first-order derivatives. However, some methods allow improving the estimation of the direction in the loss' error surface by taking into account the information of previous iterations. This is important since it has a profound impact on the performance of the optimization. There exists an unmanageable and somewhat confusing variety of extensions to the vanilla gradient descent that takes this approach; however, they can be broadly subdivided into methods that use *momentum* respectively *moving averages* and *adaptive learning rates.*

Momentum [Qian, 1999] is a way to accelerate gradient descent by including previous gradients ("moving average" weighted by some hyperparameter γ) with the effect of damped oscillations of the geodesic path

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} + \alpha \nabla_\theta \mathcal{L}(\theta), \tag{3.17}$$

$$\boldsymbol{\theta} \to \boldsymbol{\theta} - \boldsymbol{v}_t. \tag{3.18}$$

An evident extension of this method is the *nesterov accelerated gradient* algorithm [Nesterov, 2011] that additionally approximates the next position of the parameters by computing $\nabla_\theta \mathcal{L}(\theta - \gamma \boldsymbol{v}_{t-1})$. (For this approximation to be reasonable, it is important to choose small values for γ.)

Adagrad [Duchi *et al.*, 2011] is an example for an optimizer that comes with an adaptive learning rate. The updating rule for the respective weight $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} \to \boldsymbol{\theta} - \frac{\alpha}{\sqrt{G_{ii} + \epsilon}} \nabla_\theta \mathcal{L}(\theta), \tag{3.19}$$

whereby the matrix' diagonal elements G_{ii} depend on previous gradients.

The network presented in this thesis have been trained with the so-called *RMSprop* [Tieleman and Hinton, 2012] optimization algorithm. The algorithm has been developed to correct for Adagrad's radically diminishing learning rates by incorporating the exponentially decaying average of squared gradients (exponential moving average). With $\boldsymbol{g} = \nabla_\theta \mathcal{L}(\theta)$ RMSprop is given by:

$$\boldsymbol{\theta} \to \boldsymbol{\theta} - \frac{\alpha}{\sqrt{0.9 \cdot \mathbb{E}[\boldsymbol{g}^2]_{t-1} + 0.1 \cdot \mathbb{E}[\boldsymbol{g}^2]_t + \epsilon}} \boldsymbol{g}_t. \tag{3.20}$$

This algorithm is particular well suited for sparse data.

This list would be incomplete without mentioning the most prominent and best-known representative of its kind: the so-called *Adaptive Moment Estimation* or *Adam*, for short [Kingma and Ba, 2014]. Adam has an adaptive learning rate for each parameter and not only store the exponentially decaying average of past squared gradients but also accounts for exponentially decaying average of past gradients, which can be seen as akind of momentum. However, Adam has proven not to be the appropriate choice in case of Wasserstein GANs with gradient penalty (see Section 3.6.2), but has been used for Gaussian VAEs.

Other optimization algorithms available are, e.g., *Adadelta* [Zeiler, 2012], *AMSGrad* [Reddi *et al.*, 2019] etc.

3.4.3 The backpropagation algorithm

Gradient descent provides rules to update the weights based on the gradient of a predefined error function. Backpropagation is the numerical implementation of gradient descent that provides a method for performing *automatic differentiation* of complex, nested functions (such as multi-layer neural networks) by successive application of the chain and power rule.

The backpropagation algorithm consists of a *forward-pass* and a *backward-pass* (in this order). In the forward-pass step, the training data is propagated forward through the entire network that produces an output/prediction $\hat{\boldsymbol{y}}$. Now, the loss function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is evaluated for the network's prediction (or a batch of predictions) and the actual label $\boldsymbol{y}$ of the data. (This procedure, of course, is only reasonable in the context of supervised learning where the *truth* information is available during training.) The starting point of backpropagation is the very intuitive and natural definition of the "error"

$$\delta_k^{(l)} = \frac{\partial \mathcal{L}}{\partial z_k^{(l)}} = \frac{\partial \mathcal{L}}{\partial \sigma_k^{(l)}} \frac{\partial \sigma_k^{(l)}}{\partial z_k^{(l)}} = \sum_{i=1} \delta_i^{(l+1)} w_{ik}^{(l+1)} \frac{\partial \sigma}{\partial z_k^{(l)}}, \tag{3.21}$$

with $z_k^{(l)}$ referring to the output of the propagation function of the kth node/neuron in layer $l \in \{1, \ldots, L\}$. In Equation 3.21, the error has been (re-)express using the chain rule. Since backpropagation starts from the output layer, the first step is to compute $\delta_k^{(L)}$ for neurons in this layer. In this case $\delta_k^{(L)}$, is simply given by $\delta_k^{(L)} = \frac{\partial \mathcal{L}}{\partial a_k^{(L)}} \sigma' \left(z_k^{(L)} \right)$, whereby $\sigma^{(L)}$ denotes the activation function in the output layer. Now, one needs to compute the gradients of $\mathcal{L}$ with respect to the weights $w_{kj}^{(l)}$. Using the chain rule again, this is given by

$$\frac{\partial \mathcal{L}}{\partial w_{kj}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(l)}} \frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l)}} = \delta_k^{(l)} \frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l)}} \tag{3.22}$$

The second multiplicand can be written as $\frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l)}} = \sigma_j^{(l-1)}$ so that $\frac{\partial \mathcal{L}}{\partial w_{kj}^{(l)}} = \delta_k^{(l)} \sigma_j^{(l-1)}$. (Analogously, the very same can be done for the biases $\frac{\partial \mathcal{L}}{\partial b_k^{(l)}} = \delta_k^{(l)}$.) By subsequently applying the chain rule tp Equation 3.22

$$\frac{\partial \mathcal{L}}{\partial w_{kl}^{(l)}} = \sum_{\alpha\beta\gamma\ldots\omega} \frac{\partial \mathcal{L}}{\partial z_\alpha^{(L)}} \frac{\partial z_\alpha^{(L)}}{\partial z_\beta^{(L-1)}} \frac{\partial z_\beta^{(L-1)}}{\partial z_\gamma^{(L-2)}} \cdot \ldots \cdot \frac{\partial z_\gamma^{(l+1)}}{\partial z_\omega^{(l)}} \sigma_j^{(l-1)}, \tag{3.23}$$

it becomes clear why the algorithm is called backpropagation: with the chain rule, the "error" from the very first layer L is gradually backtracked to the respective layer l. The sum in Equation 3.23 is complicated and goes over all connected paths in the neural network.

With an equation for $\frac{\partial \mathcal{L}}{\partial w_{kl}^{(l)}}$ and gradient descent (see Section 3.4.2) at our disposal, the weights of the neural network can be iteratively updated until the cost function is justifiable minimized and the model provides a reasonable approximation of the training data.

3.4.4 Training stability, regularization and normalization

Since their emergence, neural networks have been cursed with stability issues. To counteract this serious problem, an incredible multitude of different regularization and normalization techniques have been developed over the recent years, to make the training of those algorithms more stable and less prone to an inopportune configuration of hyperparameters. This section only introduces a very selected subset of methods that are relevant in this work.

L_1 and L_2 regularization

Neural networks are models that usually have thousands or even millions of adjustable parameters. This is both a blessing and a curse – a blessing because it allows to potentially fit very complicated data sets; a curse since it makes them prone to overfitting the data to minimize the error function. The L_1 (Lasso regression) and L_2 (Ridge regression) are regularization methods that aim to prevent overfitting by adding a constraint on the model's parameters.

$$\mathcal{L} \supset \lambda \sum_{w \in \mathcal{W}} |w| \quad (L_1 \text{ regularization}) \tag{3.24}$$

$$\mathcal{L} \supset \lambda \sum_{w \in \mathcal{W}} |w|^2 \quad (L_2 \text{ regularization}) \tag{3.25}$$

Due to the two regularization terms above, the network prefers to learn "small weights" and hence is not able to arbitrarily vary its weights to fit the data set; hence, it helps to reduce overfitting. The two regularization techniques according to Equation 3.24 and 3.25 are basically the enforcement of constraints through Lagrange multipliers.

Dropout regularization

Dropout regularization [Srivastava *et al.*, 2014] is inspired by the observation that only parts of the neurons in the human brain are active at the same time. This effect is imitated in neural networks by randomly dropping (disabling) nodes and through this disconnecting connections between nodes according to $w \to rw$ with $r \sim \mathbb{P}_{\{0,1\}}$. This will result in a slightly different architecture for each training step. Hence, the neural network has to learn weights for each configuration of different network architectures. With dropout applied, neural networks are much less susceptible to statistical noise in the training data and therefore reduce the risk of overfitting the data by fitting noise.

Batch normalization

Batch normalization [Ioffe and Szegedy, 2015] is a technique to significantly speed up training of the neural networks by reducing the effect of the internal *covariate shift* due to frequent parameter updates in the network's layers. Additionally, it acts as a regularisation method (hence reduce overfitting) of the weights and allows for higher learning rates. Conceptually, batch normalization introduces two new trainable parameters γ, β *per layer* to the neural network. For each training iteration, the mean $\mu_j^{\mathrm{MB}} = \frac{1}{N^{\mathrm{MB}}} \sum_{i=1}^{N^{\mathrm{MB}}} x_{ij}$ and the variance $\sigma_j^{\mathrm{MB}} = \frac{1}{N^{\mathrm{MB}}} \sum_{i=1}^{N^{\mathrm{MB}}} (x_{ij} - \mu_j^{\mathrm{MB}})^2$ of a mini-batch are calculated and then used to normalize the data $\hat{x}_{ij} = \frac{x_{ij} - \mu_j^{\mathrm{MB}}}{\sqrt{\left(\sigma_j^{\mathrm{MB}}\right)^2 + \epsilon}}$. In the subsequent steps the data is scaled and shifted $\hat{x}'_{ij} = \gamma \hat{x}_{ij} + \beta =: \mathrm{BN}_{\gamma,\beta}(x_{ij})$ according to the "learnable" parameters γ, β.

Layer normalization

Batch normalization helps to significantly reduce the training time in feed-forward neural networks; however, it depends on the size of the mini-batch and therefore can not be used in combination with RNNs. Furthermore, the batch size can not be changed later after the parameters of the model are fixed. These drawbacks where the main motivations for the invention of another data normalization method called *Layer normalization* (LN) [Ba *et al.*, 2016]. Layer normalization is conceptually very similar to batch normalization. Like in batch normalization the mean μ_i^{LN} and variance σ_i^{LN} are computed, however, this time for one batch i and not for a feature (x_{ij}) along a batch (see Figure 3.6). Then, the data is normalized $\hat{x}_{ij} = \frac{x_{ij} - \mu_i^{\mathrm{MB}}}{\sqrt{\left(\sigma_i^{\mathrm{MB}}\right)^2 + \epsilon}}$ (note the different axis).

Other normalization techniques

Besides batch and layer normalization, which are the most commonly used ones, there exist other normalization techniques like *instance normalization* [Ulyanov *et al.*, 2016] or *group normalization* [Wu and He, 2018] all of which are summarized in the Figure below.

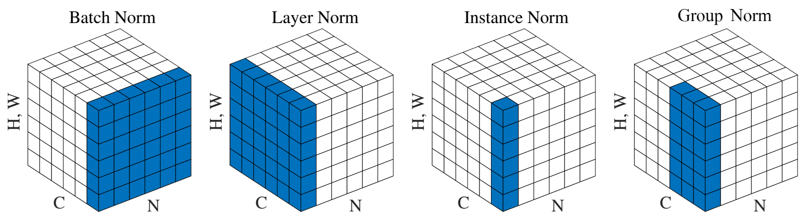


Fig. 3.6: Feature map tensor with N batches, C channels and a spatial dimension of H (height) and W (width) (adapted from [Wu and He, 2018, Fig. 2, p. 3]).

3.5 Variational Autoencoders

For a long time, the major area of application of machine learning was to construct models that learn decision boundaries between different classes. This is what is generally termed as *discriminative models* (an example of such a classification task in particle physics would be the discrimination between background and signal events). From a probabilistic point of view, a discriminative model learns the *conditional probability* distribution $p(\boldsymbol{y}|\boldsymbol{x})$, i.e., the probability that the data point $\boldsymbol{x}$ corresponds to the class or category $\boldsymbol{y}$ with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$. The information provided by discriminative models is therefore satisfactory to categorize data; but it is not sufficient to generate new data point $\hat{\boldsymbol{x}}$ according to some class or label $\boldsymbol{y}$ whereby $(\hat{\boldsymbol{x}}, \boldsymbol{y}) \notin \mathcal{D}$. The underlying mechanism that generated the data remains hidden.

To generate "unseen" data, it is necessary to model the *joint probability distribution*, i.e., the probability of a data points *and* its label $p(\boldsymbol{y}, \boldsymbol{x})$. However, in order to compute the joint distinction from conditional probability, one needs the probability distribution of the data $p(\boldsymbol{x})$ (prior). If the prior of the data is known, new data can be generated according to $p(\boldsymbol{y}, \boldsymbol{x}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$. The question which immediately comes to mind is how to determine the underlying distribution of the data that is *a priory* unknown in most of the realistic cases? This question opens the gate to a completely different world – the colourful world of *generative models* and will be our guide for the next sections to come.

With this problem being formulated, it is about time to systematically develop concepts to solve it by means of machine learning methods. The first generative models in this series – which is historically also one of the earliest – are the so-called *(Gaussian) Variational Autoencoders* (VAEs) which is a Bayesian model implemented via variational inference. The objective of this section is to give a comprehensive introduction into the topic and, at the same time, to provide an intuitive understanding of the mechanism at work.

3.5.1 Latent variable models

The basic assumption of *all* generative models is that the training data has its origin in some (unknown) probability distribution $\mathbb{P}_r$. The objective is to learn a parameterized distribution $\mathbb{P}_g := \mathbb{P}_\theta$ (see nomenclature and conventions) that approximates the *real* distribution $\mathbb{P}_r$ as close as possible $\mathbb{P}_g \approx \mathbb{P}_r$. The "naïve" way of doing so would be to *directly* learn a distribution $\mathbb{P}_g$ through of some function g_θ such that $\int_{\boldsymbol{x}} \mathrm{d}\boldsymbol{x}'\, P_\theta(\boldsymbol{x}') = 1$ with $P_\theta(\boldsymbol{x}) = g_\theta(\boldsymbol{x}) \geq 0\ \forall \boldsymbol{x} \in \mathcal{X}$. This means in particular that g_θ needs to be optimized directly by means of maximum likelihood estimation

$$\max_{\boldsymbol{\theta}\in\mathbb{R}^N} \frac{1}{M}\sum_{i=1}^{M} \log P_\theta(\boldsymbol{x}_i). \tag{3.26}$$

It can easily be seen that the optimization problem 3.26 corresponds to learning the underlying distribution of the data $\boldsymbol{x} \in \mathcal{X}$ in the limit of infinite statistics (continuous limit):

$$\begin{aligned}\lim_{M\to\infty} \max_{\boldsymbol{\theta}\in\mathbb{R}^N} \frac{1}{M}\sum_{i=1}^{M} \log P_\theta(\boldsymbol{x}_i) &= \min_{\boldsymbol{\theta}\in\mathbb{R}^N} \int_{x'} \mathrm{d}\boldsymbol{x}\, P_r(\boldsymbol{x}) \log \frac{1}{P_\theta(\boldsymbol{x})} \\ &= \min_{\boldsymbol{\theta}\in\mathbb{R}^N} \int_{x'} \mathrm{d}\boldsymbol{x}\, P_r(\boldsymbol{x}) \log \frac{P_r(\boldsymbol{x})}{P_\theta(\boldsymbol{x})} \\ &=: \min_{\boldsymbol{\theta}\in\mathbb{R}^N} D_{\mathrm{KL}}(\mathbb{P}_r||\mathbb{P}_\theta), \end{aligned} \tag{3.27}$$

where D_{KL} is the so-called *Kullback-Leibler divergence* (KL-divergence) which is a measure of similarity between the two distributions $\mathbb{P}_r$ and $\mathbb{P}_\theta$. The penultimate step is valid since the additional term does not depend on θ, hence, the solution to the minimum of the function remains unchanged. Thus, maximum likelihood estimation corresponds to minimizing the KL-divergence (in the limit of endless number of samples) – and that's the crux. The KL-divergence for $P_\theta(\boldsymbol{x}) = 0$ which often is the case if $\mathbb{P}_\theta$ lies on a low-dimensional support compared to $\mathbb{P}_r$. Besides, even if one succeeds to learn a distribution $\mathbb{P}_\theta$ with $\mathbb{P}_\theta \approx \mathbb{P}_r$, it is still necessary to sample from this model after the network has been optimized to generate new data. This requires the Metropolis–Hastings algorithms – which is a Markov chain Monte Carlo – to sample new data from $\hat{\boldsymbol{x}} \overset{\mathrm{MCMC}}{\sim} \mathbb{P}_g$, which might be quite expensive and inefficient.

This motivates an alternative approach that is introduced in the following paragraph.

Variational autoencoders – as well as generative adversarial networks that are the subject of the next section 3.6 – are examples of so-called *latent variable models* that relates observables $\boldsymbol{x}$ to a set of latent variables $\boldsymbol{z}$ parameterized function f_θ (see leftmost Figure 3.7).

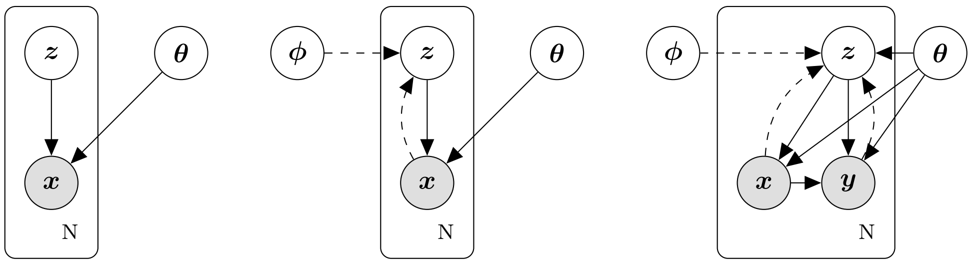


Fig. 3.7: Graphical model representation of a general latent space model (*left*), a variational autoencoder (*middle*, section 3.5.2) and a conditional variational autoencoder (*right*, section 3.5.4). In the graphs, N refers to the number of times $\boldsymbol{z}$ and $\boldsymbol{x}$ are sampled, while the parameters/weights $\boldsymbol{\theta}(\boldsymbol{\phi})$ remain fixed. The solid lines indicate the path of the generative process, while the dashed lines denote the variational approximation.

The latent vector is an element in the high-dimensional latent space $\boldsymbol{z} \in \mathcal{Z}$ whose elements are distributed according to some *known* (ideally simple) probability distribution $\mathbb{P}_z$. The fully deterministic function f_θ then defines a map from the latent space $\mathcal{Z}$ to the space $\mathcal{X}$ according to $f : \mathcal{Z} \times \boldsymbol{\theta} \to \mathcal{X}$. Since $\boldsymbol{z}$ is a random variable, the image of the function $f_\theta(\boldsymbol{z})$ is a random variable, which is distributed according to some unknown distribution $f_\theta(\boldsymbol{z}) \sim f_\theta(\mathbb{P}_z)$ with $\boldsymbol{z} \sim \mathbb{P}_z$. In concrete terms, this means that the function f_θ learns a transformation from a known to an unknown probability distribution $\mathbb{P}_\theta = f_\theta(\mathbb{P}_z)$.

The objective now is to fit the generated marginal distribution $p_\theta(\boldsymbol{x})$ to the data set $\{\boldsymbol{x}\}_{i=1}^N$ that is available for training such that $\max_{\boldsymbol{\theta}}\ p_\theta(\boldsymbol{x})$ is maximized (maximum likelihood) with respect to the models parameters. Herein, instead of learning $p_\theta(\boldsymbol{x})$ directly, i.e., without a latent space model, the problem is factorized according to

$$p_\theta(\boldsymbol{x}) = \int \mathrm{d}\boldsymbol{z}\, f_\theta(\boldsymbol{z})p(\boldsymbol{z}) = \int \mathrm{d}\boldsymbol{z}\, p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}), \tag{3.28}$$

whereby the function f_θ has been replaced by the joint probability distribution of the observables and the latent space vector. The joint probability distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the actual generative model of interest.

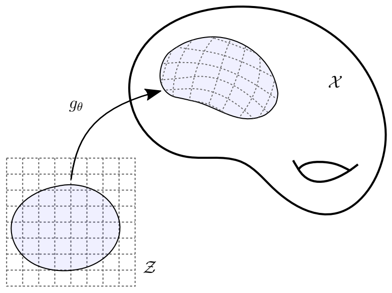


Fig. 3.8: A generative model pictured as a smooth map g_θ from a low-dimensional coordinate space/representation $\mathcal{Z}$ to a high-dimensional manifold $\mathcal{X}$ (adapted from Shao *et al.* [2017]).

The latent space $\mathcal{Z}$ may also be considered as a low-dimensional coordinate space. This Interpretation provides a connection between *generative models* and *manifolds*, which are central object of study in the domain of differential geometry. In this context, the generative model g_θ represents a smooth map $g : \mathcal{Z} \times \boldsymbol{\theta} \to \mathcal{X}$ from a coordinate space to a manifold $\mathcal{X}$. This situation is illustrated in Figure 3.8

3.5.2 Gaussian Variational Autoencoders

Gaussian variational autoencoders, although conceptually very different from classical autoencoders, try to approximately maximize Equation 3.28 under the assumption of a deep latent Gaussian model $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|f_\theta(\boldsymbol{z}), \sigma^2\mathbb{1})$. However, the optimization problem

$$\max_{\boldsymbol{\theta}} \int \mathrm{d}\boldsymbol{z}\, p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}), \tag{3.29}$$

is quite difficult, since it involves a possibly high-dimensional, intractable integral over the latent variables $\boldsymbol{z}$. One possible approach to evaluate the integral in Equation 3.29 would be to utilize Monte Carlo methods as introduced in Chapter 2.1.1, with the approximation of the integral given by $p_\theta(\boldsymbol{x}) \approx \frac{1}{N}\sum_{i=1}^{N} p_\theta(\boldsymbol{x}|\boldsymbol{z}_i)$. Since $\mathbb{P}_z$ is "simple" and known, the sampling of the latent variable $\boldsymbol{z} \sim \mathbb{P}_z$ does not require expensive Markov chain Monte Carlos. Nonetheless, this approach, is not practical since in a high-dimensional space the number of samples N drawn from $\mathbb{P}_z$ must be very large ($N \gg \dim(\mathcal{Z})$) to get a good estimation of the integral 3.28. This is because $p_\theta(\boldsymbol{x}|\boldsymbol{z}_i)$ will be close to zero for most values of $\boldsymbol{z}$ and hence contribute little to the estimate of $p_\theta(\boldsymbol{x})$. Therefore, the evaluation of Equation 3.28 by means of Monte Carlo integration is very inefficient – but possible. To solve the optimization problem 3.29, another approach is needed.

Variational autoencoders try to solve exactly this problem by efficiently sample only values of the latent variable $\boldsymbol{z}$ that are likely, i.e., with a high probability to have generated the data $\boldsymbol{x}$, and then compute $p_\theta(\boldsymbol{x})$ solely for those dominant contributions. This corresponds to the conditional probability distribution $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ (a new function/network parameterized by $\boldsymbol{\phi}$), i.e., the probability of the latent variable given the data. To rephrase the crucial statement above: to efficiently solve Equation 3.29, one samples the "most likely" values of $\boldsymbol{z}$ and based on those evaluates the integral 3.28 by means of Monte Carlo integration $\mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_\phi}[p_\theta(\boldsymbol{x}|\boldsymbol{z})]$.

How does this small detour solve the problem? At this stage – not at all. The challenge is now to perform posterior inference. According to Bayes' theorem the posterior is given by

$$p_\phi(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{\int \mathrm{d}\,\boldsymbol{z} p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}, \tag{3.30}$$

which still involves an intractable integral in the denominator. Furthermore, the distribution $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ is unknown; hence, to sample from it would again require Markov chain Monte Carlo techniques. To finally solve the issue, the posterior inference problem 3.31, which involves an intractable integral, is converted into an optimization problem by means of *variational inference* – which gave the method its name. In general, variational inference seeks to find an distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ that approximates the posterior probability. More formally, variational inference solves the following optimization problem

$$\min_{\boldsymbol{\phi}} D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) = \min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}\right], \tag{3.31}$$

where D_{KL} is the so-called *Kullback-Leibler divergence* that is a measure of similarity between two distributions q, p and given by $D_{\mathrm{KL}}(q||p) = \int_x q(x)\log\frac{q(x)}{p(x)}$. Still, the problem remains the same since the Kullback-Leibler divergence involves an integral over the latent space. However, the systematic evaluation of Equation 3.31 by applying Bayes' rule gives:

$$D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) = \int \mathrm{d}\boldsymbol{z}\, q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})} \tag{3.32}$$

$$= \int \mathrm{d}\boldsymbol{z}\, q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})} \tag{3.33}$$

$$= \int \mathrm{d}\boldsymbol{z}\, q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})} + \int \mathrm{d}\boldsymbol{z}\, q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log p(\boldsymbol{x}) \tag{3.34}$$

$$= \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})}\right] + \log p(\boldsymbol{x}). \tag{3.35}$$

The last step is possible since $p(\boldsymbol{x})$ is independent of $\boldsymbol{z}$ and $\mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}[1] = 1$. Rearranging Equation 3.35 allows to find a lower bound on the log-probability of observed data

$$\log p(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})}\right] \tag{3.36}$$

$$\geq -\mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})}\right] \tag{3.37}$$

$$= \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right]. \tag{3.38}$$

The statement above is true because $D_{\mathrm{KL}}(q||p) \geq 0$ and $\log p(\boldsymbol{x}) \leq 0$. The last Equation 3.38

$$\mathcal{L}^{\mathrm{ELBO}}(\theta,\phi;\boldsymbol{x},\boldsymbol{z}) := \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right] \tag{3.39}$$

is the solution to the problem and hence deserves a special name; it is called the *Evidence Lower BOund* (ELBO), since it gives a lower bound for the computational intractable evidence $p(\boldsymbol{x})$.

Equation 3.39 is a clear instruction on how to proceed: instead of directly extremize the evidence $\max_\theta\ p_{\boldsymbol{\theta}}(\boldsymbol{x})$, the ELBO is optimized $\mathcal{L}^{\mathrm{ELBO}}$ with respect to $\boldsymbol{\theta}, \boldsymbol{\phi}$ instead. The generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ – referred to as the *decoder* network – and the inference model – known as the *encoder* network – $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ are both neural networks with the set of weight $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ respectively (see Figure 3.7).

The evidence lower bound according to Equation 3.39 has two terms that have a simple interpretation. The first term, $\mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$, is a regression task. This can easily be seen by substituting $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ by the Gaussian constraint $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|f_\theta(\boldsymbol{z}, \sigma^2\mathbb{1})$. (The assumption of a Gaussian model corresponds to the assumption of Gaussian distributed reconstruction errors.) Hence, the ELBO for a *Gaussian* Variational Autoencoder (GVAE) is given by:

$$\mathcal{L}^{\mathrm{ELBO}}(\theta,\phi;\boldsymbol{x},\boldsymbol{z}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[||\left(\boldsymbol{x} - f_\theta(\boldsymbol{z})\right)||^2\right] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right]. \tag{3.40}$$

Thus, the reconstruction loss for the regression task of the GVAE is a simple MSE between the actual training data $\boldsymbol{x}$ and the data form the generative model $\hat{\boldsymbol{x}} = f_\theta(\boldsymbol{z})$. The second term is what distinguishes VAEs from Classical autoencoders. The term $\mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right]$ defines a constrain on the "shape" of latent space since the KL-divergence measures the similarity between the distribution $\mathbb{Q}_\phi$ learned by the inference network and the imposed distribution over the random noise vectors $\boldsymbol{z}$. If $\mathbb{Q}_\phi$ differs significantly from $\mathbb{P}_z$, the KL-divergence is large and penalizes the network's weights to correct the error. This constrain of the shape of the latent space is utterly important; it allows to sample from the known distribution P_z to generate new data via the trained decoder network f_θ. Without the restriction of the shape, the system would be just an autoencoder and therefore no generative model.

3.5.3 Implementation

The previous section derived the objective function for variational autoencoders, i.e., the ELBO. The optimization problem that is addressed by VAEs is:

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{p(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right], \tag{3.41}$$

whereby $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ denote the weights of the *decoder* resp. *encoder* network. As discussed in the context of gradient descent (3.4.2) and backpropagation (3.4.3), the weights of the networks are updated based on the gradients of the loss function. So, in order to perform gradient descent, one needs to compute the gradients of Equation 3.41 with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

The integral in Equation 3.41 can simply be evaluated by Monte Carlo integration $\mathcal{L}^{\text{ELBO}} \approx \frac{1}{L}\sum_{k=1}^{L}\log p_\theta(\boldsymbol{x},\boldsymbol{z}_k) - \log q_\phi(\boldsymbol{z}_k|\boldsymbol{x})$ with $\boldsymbol{z}_k \sim \mathbb{Q}_\phi$. Now, in contrast to the previous situation, Monte Carlo integration now is appropriate since the inference network samples $\boldsymbol{z}$ values with a large contribution to $p_\theta(\boldsymbol{x})$ (in fact, in most situations it is appropriate to set $L = 1$). The computation of the gradient ∇_θ is straightforward since the expectation value in Equation 3.41 is with respect to $\boldsymbol{\phi}$. Therefore, the gradient ∇_θ is given by $\nabla_\theta\mathcal{L}^{\text{ELBO}} \approx \frac{1}{L}\sum_{k=1}^{L}\nabla_\theta \log p_\theta(\boldsymbol{x},\boldsymbol{z}_k)$. In case of ∇_ϕ there is an additional obstacle due to $\nabla_\phi\mathbb{E}_{q_\phi}[f(z)] \neq \mathbb{E}_{q_\phi}[\nabla_\phi f(z)]$. This problem can, *inter alia*, be solved by making use of the so-called *reparameterization trick* which is a simple reparameterization of $q_\phi(\boldsymbol{z}_k|\boldsymbol{x})$ such that $\boldsymbol{z}_k$ is given by $\boldsymbol{z}_k = \boldsymbol{\mu}_\phi + \boldsymbol{\epsilon}\otimes\boldsymbol{\sigma}_\phi(\boldsymbol{x})$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbb{1})$.

Now, after the fundamental prelude, it is about time to breath life into the dry theory, and make it come alive.

To illustrate the functional principle, a very simple VAE is constructed. The two networks, encoder and decoder, are a sequence of fully connected layers as they have already used in Section 3.3.1 (see Figure 3.2). The data set consists of only one image with 200×200 pixels, so the dimensionality of the input and the output vector is $\dim(\boldsymbol{x}) = \dim(\hat{\boldsymbol{x}}) = 40,000$ pixels. Compared with the dimensionality of the latent space, which is only 50, this resembles an extremely narrow "bottleneck" with a compression factor of ${}^{\dim(\boldsymbol{x})}\!/_{\dim(\hat{\boldsymbol{x}})} = 0.00125$. This is only possible since the network is trained on only one image, so there is no diversity in the data (vanishing entropy $H(\boldsymbol{X}) = \mathbb{E}_{\boldsymbol{X}\sim\mathbb{P}_r}[I(\boldsymbol{X})] = 0$). In this simple setup with only one data point, even a latent space dimension of 1 would be appropriate if the complexity of the networks is sufficient.

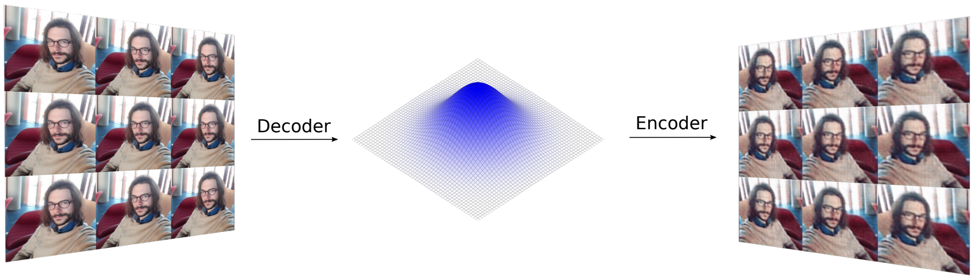
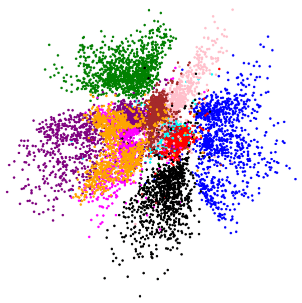


Fig. 3.9: The author being used as "guinea pig" to demonstrate the operating principle of a GVAE. The encoder network (inference network) receives a set of images $\boldsymbol{x}$ and learns a hidden and compressed representation of the data. In a GVAE, the latent space is constrained to be Gaussian (second term in Equation 3.40). The decoder network (generative model) samples from a Gaussian $\boldsymbol{z} \sim \mathcal{N}(0, \mathbb{1})$ and tries to reconstruct the input as close as possible (regression task). Note: the array of images on the right side has actually been generated by a neural network.

As it can be seen in Figure 3.9, the GVAE does a good job in reconstructing the input data.

3.5.4 Conditional variational autoencoders

This thesis focus on Conditional (Gaussian) Variational Autoencoders (C(G)VAEs) which is an extension of the VAEs introduced in Section 3.5.2. In the generative model introduced in the previous section, there is no mechanism that allows controlling the data generation process. So, to generate new, i.e., unseen data $\hat{\boldsymbol{x}}$ one samples a random noise vector $\boldsymbol{z} \sim \mathbb{P}_z$ and produce new data points via the decoder network $\hat{\boldsymbol{x}} = f_\theta(\boldsymbol{z})$. However, since the class, category or, more generally, the label $\boldsymbol{y}_i$ of the data $\boldsymbol{x}_i$ was not explicitly provided during training, the VAE learns to encode this information in a non-trivial way. For instance, a VAE that has been trained on the MNIST data set [LeCun and Cortes, 2010] to generate "handwritten digits"; the Result can be seen in the Figure below.



(a) Latent space

(b) Generated data

Plot 3.4: VAE trained on the MNIST data set. $\mathbb{R}^2$ latent space representation (*left*) where each different color corresponds to a digit in the set $\{0, \ldots, 9\}$ and randomly generated data (*right*) (Taken from Davidson *et al.* [2018] Fig. 2a, p. 5; Fig. 10. p. 18).

As it can be seen in Figure 3.4, the network (unsupervised) clusters the compressed representation of the data into groups in the hidden space. By sampling $\boldsymbol{z}$, one also randomly samples a category/label $\boldsymbol{y}$ from $\mathbb{P}_y$. To get a data point associated with a certain label, it would be necessary to perform a scan of the latent space, which is infeasible if its dimensionality is large. Conditional variational autoencoders, for instance, provide a solution to generate data associated with a specific label, e.g. a specific digit, a jet with a certain energy or a certain process like QCD or W initialized jets. This label might be discrete (e.g. QCD $\mapsto y = 0$ and $W \mapsto y = 1$), like in the example above, or even continuous such as the reconstructed energy of a jet.

More formally, while a plain VAE learns the distributions $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ (encoder) and $q_\theta(\boldsymbol{x}|\boldsymbol{z})$ (decoder), a VAE that is conditioned on a label $\boldsymbol{y}$ learns the distributions $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ and $q_\theta(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{y})$ that are additionally conditioned on $\boldsymbol{y}$. All arguments and statement from section 3.5.2 remain valid in case of a CVAE, with the addition that now the probability distributions are conditioned on $\boldsymbol{y}$. So, e.g., the ELBO in case of CVAEs is given by:

$$\mathcal{L}^{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \frac{p(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{y})}{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})}\right]. \tag{3.42}$$

Practically, the additional information must be provided during training along with the data $\boldsymbol{x}$. This is done (in this thesis) by concatenating $\boldsymbol{x} \in \mathbb{R}^{N_{\boldsymbol{x}}}$ and $\boldsymbol{y} \in \mathbb{R}^{N_{\boldsymbol{y}}}$ with $\boldsymbol{x} \oplus \boldsymbol{y} \in \mathbb{R}^{N_{\boldsymbol{x}}+N_{\boldsymbol{y}}}$ (there is a large range of different approaches to incorporate the addition information into the architecture of the networks).

3.6 Generative Adversarial Networks

The variational autoencoders discussed in the previous section are simple examples of generative models that are based on machine learning methods and variational Bayes. From a theoretical point of view, VAEs are well defined in terms of their probabilistic interpretation (log-likelihood estimated by lower-bounded evidence). Also, their practical implementation utilizing machine learning methods is no obstacle – on the contrary, those generative models have proven to be very stable with good convergence characteristics. However, the Gaussian VAEs introduced in Section 3.5 are based on a rather strong assumption regarding the probabilistic distribution of the latent space, which is constrained to be a multivariate normal distribution, and the reconstruction error, which assumed to be Gaussian distributed as well. So, the distribution of the data is learned by fitting the data via a multi-dimensional Gaussian distribution. Furthermore, the dimensionality of the latent space is usually small compared to the data set one's; hence, creating a "tight" bottleneck that may give rise to data loss. This usually results in noisy data produced by the generative model. This section, therefore, introduces another approach to generative models.

3.6.1 GANs according to Ian Goodfellow *et al.*

The development of Generative Adversarial Networks (GANs) by Ian Goodfellow *et al.* in 2014 [Goodfellow *et al.*, 2014] may indeed be regarded as a real paradigm shift in the field of machine learning – almost unprecedented in this area of research. The heart of the revolution is simply the introduction of an *adversarial framework* in which "[...] the generative model is pitted against an adversary[.]" (Goodfellow *et al.* [2014]); two networks that play a *non-cooperative* game against each other. The actual generative model is implemented by the so-called *generator* (neural) network g_θ, whereby $\boldsymbol{\theta}$ are the trainable parameters of the model. The objective of the generator is to learn a mapping through a *continuous* function g_θ from a known distribution $\mathbb{P}_z$ to a *generated* probability distribution $\mathbb{P}_g$ that approximates the distribution of the underlying (training) data $\mathbb{P}_r$ as close as possible $\mathbb{P}_g \approx \mathbb{P}_r$. If this approximation is met to a high degree of agreement, new, unseen data points $\hat{\boldsymbol{x}} \sim \mathbb{P}_g \approx \mathbb{P}_r$ can be generated by sampling a *seed* $\boldsymbol{z}$ from the latent space $\mathscr{Z}$ which is then transformed by the model g_θ with fixed parameters $\hat{\boldsymbol{x}} \sim g_\theta(\boldsymbol{z})$. Based on this argument, the generative model in GANs is – quite similar to variational autoencoders – a latent variable model (see Section 3.5.1) that transforms a known distribution $\mathbb{P}_z$ to a more complicated one $\mathbb{P}_g := \mathbb{P}_\theta = g_\theta(\mathbb{P}_z)$ which is still unknown but indirectly accessible via the learned transformation g_θ. The second network is referred to as the *discriminator* f_ϕ, i.e., the adversarial network in the system. The discriminator network is a binary classifier whose task is to discriminate between data points that originate from the underlying distribution of the real data $\boldsymbol{x} \sim \mathbb{P}_r$ and those that are generated ("faked") by the generator network $\hat{\boldsymbol{x}} \sim \mathbb{P}_g$. The objective of the generator, on the other hand, is to learn a distribution $\mathbb{P}_g$ such that the discriminator fails its discrimination task, i.e., it assigns the probability $f_\phi(\boldsymbol{x}_g) \approx f_\phi(\boldsymbol{x}_r) \approx \frac{1}{2}$ for both the real and the generated samples (state of minimum confidence).

Theoretical considerations

As described above, GANs use an adversarial game between two competing networks, g_θ and f_ϕ, while, from a probabilistic point of view, variational autoencoders maximize the likelihood of the data employing the ELBO instead (see Section 3.5). The objective of the generator network is to learn an approximation $\mathbb{P}_g$ of the data such that the discriminator can not distinguish it from the underlying distribution of the training data $\mathbb{P}_r$ anymore. Generally speaking, this means that the discriminator represents a family of functions $\mathscr{L}$ that measures the dissimilarity between $\mathbb{P}_r$ and $\mathbb{P}_g$ [Roth *et al.*, 2017]. The objective of the generator-discriminator system is therefore given by the *minimum* of the *supremum* over $\mathscr{L}$ (i.e. extremize the discrimination between real and generated distribution):

$$\min_{\boldsymbol{\theta}} \left[\sup_{\mathcal{L} \in \mathscr{L}} \mathcal{L}(\mathbb{P}_r, \mathbb{P}_g := \mathbb{P}_\theta) \right], \tag{3.43}$$

whereby $\boldsymbol{\theta}$ are the parameters of the generated distribution (Equation 3.43 taken from Roth *et al.*, 2017, Eq. 1, p. 1 (notation has been modified)). Equation 3.43 represents a *saddle point problem*; hence, the solution – which is known as Nash equilibrium (named after the American mathematician John Forbes Nash) – is likely to be unstable. The optimization problem is implemented by representing $\mathcal{L}$ through a family of parameterized functions f_ϕ that are realized by a neural network, i.e., the aforementioned discriminator model. In classical GANs as proposed by Goodfellow *et al.*, the objective or loss function $\mathcal{L}$ is assumed to be a two-player *min-max* game with a *(log-)logistic classification* task according to

$$\mathcal{L}(\mathbb{P}_r, \mathbb{P}_\theta; \phi) := \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[\log f_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_\theta}[\log(1 - f_\phi(\boldsymbol{x}))], \tag{3.44}$$

respectively explicitly expressed in terms of the generator's weights $\boldsymbol{\theta}$ and the distribution of the latent space $\mathbb{P}_z$

$$\mathcal{L}(\mathbb{P}_r, g_\theta(\mathbb{P}_z); \phi, \theta) := \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[\log f_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_z}[\log(1 - f_\phi(g_\theta(\boldsymbol{z})))]. \tag{3.45}$$

Consequently, generative adversarial networks solve the following non-cooperative two-player min-max game:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathcal{L}(\mathbb{P}_r, g_\theta(\mathbb{P}_z); \phi, \theta). \tag{3.46}$$

Equation 3.43 gives two opposing optimization problems that define the loss function/objective for the discriminator

$$\mathcal{L}^f_\theta(\boldsymbol{\phi}) := \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log f_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\log(1 - f_\phi(g_\theta(\boldsymbol{z})))] \tag{3.47}$$

and the generator model respectively

$$\mathcal{L}^g_\phi(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\log(1 - f_\phi(g_\theta(\boldsymbol{z})))], \tag{3.48}$$

whereby the last Equation derives from the fact that $\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log f_\phi(\boldsymbol{x})]$ does not depend on $\boldsymbol{\theta}$.

According to Equation 3.46 and 3.47, the objective of the discriminator model is the maximization of the classification probability, i.e., the correct assignment of samples to either the real or the generated distribution. An almost perfect discriminator would assign $f_\phi(\boldsymbol{x}) \approx 1$ for $\boldsymbol{x} \sim \mathbb{P}_r$ and $f_\phi(\hat{\boldsymbol{x}}) \approx 0$ for a generated data point $\hat{\boldsymbol{x}} \sim \mathbb{P}_g$. This is usually the situation for the initial training period since the generator mostly produces random output that can easily be categorized as being fake. This is true even though both models, the generator as well as the discriminator, are initialized randomly. However, the classification task of the discriminator is fundamentally that the generation task of the generator (as in real life, it is always easier to criticize than to be creative). Therefore, it frequently happens during training that the discriminator becomes too powerful. In this case, the output of the discriminator is $f_\phi(g_\theta(\boldsymbol{z})) \approx 0$ for most of the generated samples $\hat{\boldsymbol{x}} = g_\theta(\boldsymbol{z})$; hence, $\mathcal{L}^g_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ vanishes, as a result of which the generator's weights do not get update anymore – the model configuration is frozen. This problem is known as *saturation*. Therefore, one often replaces the optimization problem to maximize $\mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\log f_\phi(g_\theta(\boldsymbol{z}))]$ instead, which leads to the very same fix points but reduces the risk of saturating gradients. Finally, the training of generative adversarial networks is summarized in Figure 3.10.

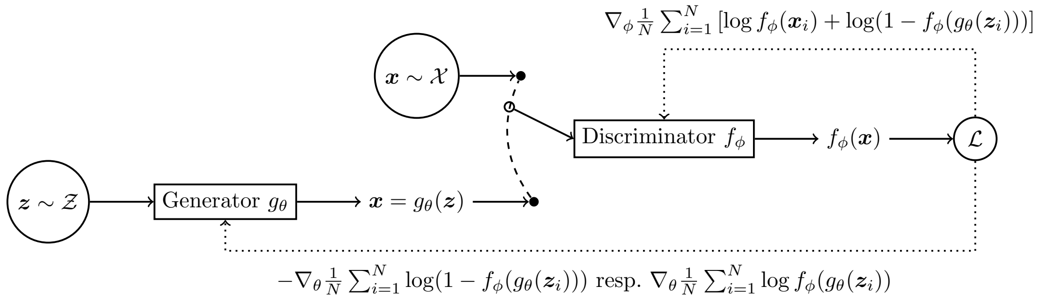

$\nabla_\phi \frac{1}{N}\sum_{i=1}^N [\log f_\phi(\boldsymbol{x}_i) + \log(1 - f_\phi(g_\theta(\boldsymbol{z}_i)))]$
$\boldsymbol{x} \sim \mathcal{X}$
Discriminator f_ϕ
$f_\phi(\boldsymbol{x})$
$\mathcal{L}$
$\boldsymbol{z} \sim \mathcal{Z}$
Generator g_θ
$\boldsymbol{x} = g_\theta(\boldsymbol{z})$
$-\nabla_\theta \frac{1}{N}\sum_{i=1}^N \log(1 - f_\phi(g_\theta(\boldsymbol{z}_i)))$ resp. $\nabla_\theta \frac{1}{N}\sum_{i=1}^N \log f_\phi(g_\theta(\boldsymbol{z}_i))$

Fig. 3.10: Graphical illustration of the training of generative adversarial networks as proposed by Goodfellow *et al.* The discriminator model/network f_ϕ alternately receives real data from the training set and fake data produced by the generator network g_θ. The loss function is evaluated, and the weights (trainable parameters) of the networks are updated accordingly. The procedure is repeated till convergences or termination of the training.

To better understand the fundamental mechanisms at work, it is reasonable to study the best discriminator f_{ϕ^*} given any generator g_θ. According to Equation 3.47, the optimal discriminator is the network whose parameter configuration results in $\nabla_\phi \mathcal{L}_\phi \overset{!}{=} 0$. First, the expectation value in Equation 3.47 is expressed in terms of an integral

$$\mathcal{L}_\phi := \int_{\boldsymbol{x}'} \mathrm{d}\boldsymbol{x} \underbrace{\left(p_r(\boldsymbol{x}) \log f_\phi(\boldsymbol{x}) + p_g(\boldsymbol{x}) \log(1 - f_\phi(g_\theta(\boldsymbol{z})))\right)}_{=L_\phi}, \tag{3.49}$$

whereby $\boldsymbol{x}$ is sampled over all possible values. Now, consider the gradient with respect to $\boldsymbol{\phi}$

$$\begin{aligned} \nabla_\phi L_\phi &= p_r(\boldsymbol{x}) \nabla_\phi L_\phi \log f_\phi(\boldsymbol{x}) + p_g(x) \nabla_\phi L_\phi \log(1 - f_\phi(\hat{\boldsymbol{x}})), & (3.50) \\ &= p_r(\boldsymbol{x}) \frac{\nabla_\phi f_\phi(\boldsymbol{x})}{f_\phi(\boldsymbol{x})} - p_g(\boldsymbol{x}) \frac{\nabla_\phi f_\phi(\hat{\boldsymbol{x}})}{1 - f_\phi(\hat{\boldsymbol{x}}))}, & (3.51) \\ &= \frac{\nabla_\phi f_\phi(\boldsymbol{x})}{f_\phi(\hat{\boldsymbol{x}})(1 - f_\phi(\hat{\boldsymbol{x}})} \left(p_r(\boldsymbol{x}) - f_\phi(\hat{\boldsymbol{x}})(p_r(\boldsymbol{x}) + p_g(\boldsymbol{x}))\right). & (3.52) \end{aligned}$$

Hence, according to Equation 3.52 the perfect discriminator f_{ϕ^*} is given by $p_r(\boldsymbol{x}) - f_\phi(\hat{\boldsymbol{x}})(p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})) \overset{!}{=} 0$ with $f_\phi(\hat{\boldsymbol{x}}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. Consistently, the output of a perfect discriminator for an optimal generator, i.e., $p_g(\boldsymbol{x}) = 1$ is – as expected – $f_{\phi^*}(\boldsymbol{x}) = \frac{1}{2}$ (random guessing).

Under the assumption of an optimal discriminator, the loss function 3.49 provides some insights into the actual mechanisms at work. Replacing f_ϕ by f_{ϕ^*} in Equation 3.49 gives

$$\begin{aligned} \mathcal{L}_{\phi^*} &= \int_{\boldsymbol{x}'} \mathrm{d}\boldsymbol{x} \left(p_r(\boldsymbol{x}) \log \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})} + p_g(\boldsymbol{x}) \log \frac{p_g(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right), & (3.53) \\ &= D_{\mathrm{KL}}\left(p_r || \frac{p_r + p_g}{2}\right) + D_{\mathrm{KL}}\left(p_r || \frac{p_r + p_g}{2}\right) - 2\log 2, & (3.54) \\ &= 2D_{JS}\left(p_r || p_g\right) - 2\log 2, & (3.55) \end{aligned}$$

whereby D_{JS} is the so-called *Jensen–Shannon (JS-)divergence*, which can be thought of as a symmetrized Kullback–Leibler divergence. (In contrast to the KL-divergence, the JS-divergence does represent a proper metric.) Equation 3.55 represents an important result: the minimization of the loss function 3.49 essentially corresponds to the minimization of the Jensen Shannon Divergence that measures the *similarity* between $\mathbb{P}_r$ and $\mathbb{P}_g$. Therefore, the objective function of generative adversarial networks is an *f-divergence.*

Problems with classical GANs

As explained in the previous paragraph, GANs are a tempting alternative to variational autoencoders since they minimize an f-divergence between the generated and the underlying distribution of the training data instead of maximizing the likelihood (ELBO) of the data. The direct modelling of the target distribution should in principle provide better results than VAEs; this is usually the case. However, finding the Nash equilibrium of the non-convex loss function 3.45 in a very high-dimensional space that is spanned by the trainable weights of the model is quite challenging. Consequently, GANs have been plagued with non-convergence since the very beginning of their existence. One reason for the instability of GANs is the utilization of the gradient descent algorithm (see Section 3.4.2) to update the parameters of the neural network(s). Since the solution to the non-cooperative two-player min-max game 3.46 is a saddle point, the model is inherently unstable regarding perturbations around this equilibrium. This can nicely be illustrated by using the simple example of the following popular min-max game $\min_x \max_y xy$. Using gradient descent, the updated values of x_{i+1} and y_{i+1} for the $(i+1)$th iteration are given by $x_{i+1} = x_i - \gamma_n \partial_x(xy) = x_i - \alpha_i y_i$ and $y_{i+1} = y_i - \gamma_n \partial_y(xy) = y_i + \alpha_i x_i$ – whereby the learning rate α_i depends on the number of iterations as well (see optimizer and non-constant learning rates section 3.4.2). Hence, the new state vector $\boldsymbol{x}_{i+1}^T = (x_{i+1}, y_{i+1})^T$ can be expressed by a matrix multiplication $\boldsymbol{x}_{i+1} = \boldsymbol{M}\boldsymbol{x}_i$ with $\boldsymbol{M} = \left(\begin{smallmatrix} 1 & -\alpha_i \\ \alpha_i & 1 \end{smallmatrix}\right) = \sqrt{1+\alpha_i}\left(\begin{smallmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{smallmatrix}\right)$, whereby $\boldsymbol{M}$ has been expressed in terms of a simple rotation with $\beta_i = \arccos((1+\alpha_i^2)^{-\frac{1}{2}})$[3]. To reach a stable orbit, the path radius ρ_i must be finite $\rho_i < \infty \; \forall i \in \mathbb{N}$. The radius after N iterations is given by $\rho_N = \prod_{i=0}^{N} \sqrt{1+\alpha_i^2}$, with the length of the curve being $\sum_{i=1}^{N} \alpha_i$. For each choice of α_i the gradient descent algorithm will result in a stable orbit around the origin or will diverge rather than converging to the Nash equilibrium $x = y = 0$ with $\rho_\infty = 0$ since $1 \leq \prod_{i=0}^{\infty} \sqrt{1+\alpha_i^2} = \rho_\infty$. Consequently, gradient descent is incapable to find the desired stable point in this particular (admittedly slightly pathological) example. In general, to use gradient descent in combination with generative adversarial networks, the learning rate must be chosen very small, e.g. $\alpha_l \sim \mathcal{O}(10^{-4})$ or below, to get convergence. This, of course, goes hand in hand with a significantly increased number of training iterations.

The problem described above is a consequence of the GANs' loss function and the optimization through gradient descent. This, however, is only one out of many problems regarding the stability of generative adversarial networks. One of the most serious threats is a phenomenon that is commonly known as *mode collapse*. In case of mode collapse, the generative model only learns a small subset of the underlying distribution of the training data, with the effect of completely underestimating the *entropy* (a measure of encoded information) of the target distribution. So, the generative model might produce high-quality data – but with less or even without any diversity. It is intuitively clear that adversarial models are vulnerable to this effect: assumed the discriminator model does not get updated anymore (because it reached the limit of its complexity of experienced vanishing gradients), in this case, the optimal point is given by $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}} f_\phi(\boldsymbol{x})$ to a high precision, which corresponds to a single *mode*. The point $\boldsymbol{x}^*$ is then learned by the generator $\boldsymbol{x}^* \approx g_\theta(\boldsymbol{z})$. Hence, the generator maps all seeds to the same point and is therefore independent of $\boldsymbol{z}$, with the consequence of vanishing gradients with respect to $\boldsymbol{z}$. If this happens, the generator has collapsed to a single point without any hope of recovery; the training of the model must be terminated for good and started from the very beginning.

There are further obstacles besides the aspects mentioned above, for instance, the high sensitivity to the complexity and the architecture of the generator and the discriminator network. The two models must be precisely balanced to avoid that one outperforms the other. This is related to yet another problem when it comes to training GANs, i.e., the tuning of the model's hyperparameters. An inopportune configuration of hyperparameters quickly leads to non-convergence – as well as frustration and resignation.

Last but not least, there is an inherent problem based on the fact that the loss function corresponds to the JS-divergence which does not converge under a sequence of distributions with disjoint support. This circumstance is the main motivation for the use of alternative metrics such as e.g., the Wasserstein distance – which will be the topic of the following section.

There exists a considerable weaponry for classical GANs in the battle against afore-

[3]The factor $\sqrt{1+\alpha_i^2}$ follows from the relation $\sin\arccos x = \sqrt{1-x^2}$, hence, $\pm\sin\arccos((1+\alpha_i^2)^{-\frac{1}{2}}) = \pm\sqrt{1-(1+\alpha_i^2)^{-1}} = \pm\alpha_i/\sqrt{1+\alpha_i^2}$.

mentioned enemies, many of which are heuristically. Examples include mini-batch discrimination, historical averaging, one-side label smoothing, virtual batch normalization etc. (for a comprehensive list along with implementation details see Salimans *et al.* [2016a]).

Sources of instabilities – the loss function

As already mentioned before, some stability issues when training GANs are a consequence of the incapability of the gradient descent (first-order) optimization algorithm to converge to the Nash equilibrium of the non-convex cost function. There are – in principle – algorithms that do not suffer the same problem; however, all these methods are not feasibly optimized functions in very high-dimensional spaces, such as those spanned by the weights of neural networks. The other category of problems, however, is more fundamental since it is directly related to the loss function (more precisely, the underlying f-divergence).

An important contribution towards a better understanding of the dynamics of generative adversarial networks was provided by Martin Arjovsky and Léon Bottou in 2017. In their work, they identify several sources of instabilities. They recognized that many problems regarding the instability in GANs are caused by a perfect discriminator network that is, in turn, related to the *support* of the real and the generated distribution – whereby the support of a probability distribution f (or function in general) $\text{supp}(f)$ is the set of points $\boldsymbol{x} \in \mathcal{X}$ for which f is non-zero, i.e., $\text{supp}(f) = \{\boldsymbol{x} \in \mathcal{X} | f(\boldsymbol{x}) \neq 0\}$ (non-zero probability mass). In their publication, they showed that if the support of two distributions $\mathbb{P}_g$ and $\mathbb{P}_r$ do not perfectly align and one of them lies on a lower-dimensional manifold, then there exists an optimal discriminator f_{ϕ^*} that perfectly discriminates between $\mathbb{P}_g$ and $\mathbb{P}_r$ (see Theorem 2.2 in Arjovsky and Bottou, 2017). In this case, $\nabla_{\boldsymbol{x}} f_{\phi^*}(\boldsymbol{x})$ will be zero almost everywhere; hence, the generator does not get updated anymore due to vanishing gradients (see Theorem 3.6.1). This is a serious problem since the generated manifold usually lies on low-dimensional support due to the "low" dimension of $\mathcal{Z}$ compared to $\mathcal{X}$. This insight led to the following theorem.

Theorem (Arjovsky and Bottou, 2017, Thm. 2.3, p. 5)**.** *Let $\mathbb{P}_r$ and $\mathbb{P}_g$ two distributions whose support lies in two manifolds $\mathcal{M}$ and $\mathcal{P}$ don't have full dimension and don't perfectly align. We further assume that $\mathbb{P}_r$ and $\mathbb{P}_g$ are continuous in their respective manifolds. Then*

$$\begin{aligned} JSD(\mathbb{P}_r||\mathbb{P}_g) &= \log 2 \\ KL(\mathbb{P}_r||\mathbb{P}_g) &= +\infty \\ KL(\mathbb{P}_g||\mathbb{P}_r) &= -\infty \end{aligned}$$

(Proof: see definition of Kullback-Leibler and Jensen-Shannon divergence)

From Theorem 3.6.1 follows that the Kullback-Leibler divergence may not be a reasonable measure of similarity between the two distributions $\mathbb{P}_g$ and $\mathbb{P}_r$. This leads to a problem of interpretation regarding the actual values of the loss function that do not necessarily correspond to the performance of the model. This problem will, *inter alia*, lead to an interpretable loss function in the next section.

So, the problem of instability is related to the discriminator becoming too powerful compared to the generator, which leads to vanishing gradients with respect to the generator's weights. This is summarized in Theorem 3.6.1.

Theorem (**Vanishing gradients on the generator**; Arjovsky and Bottou, 2017, Thm. 2.3, p. 5)**.** *Let $g_\theta : \mathcal{Z} \to \mathcal{X}$ be a differentiable function that induces a distribution $\mathbb{P}_g$. Let $\mathbb{P}_r$ be the* *real data distribution. Let D be a differentiable discriminator $[f_\phi]$. If the conditions of Theorems 2.1 or 2.2 are satisfied, $||D-D^*|| < \epsilon$ $[D^*$ is a perfect discriminator], and $\mathbb{E}_{z\sim p(z)}\left[||J_\theta g_\theta(z)||\right]_2^2 \leq M^2$,[.] then*

$$||\nabla_\theta \mathbb{E}_{z\sim p(z)}[\log(1-D(g_\theta(z)))]||_2 < M\frac{\epsilon}{1-\epsilon}$$

(Proof: see Arjovsky and Bottou, 2017, p. 6)

If the discriminator becomes perfect – which according to the statements above is likely –, i.e., $f_\phi \to f_{\phi^*}$ $(D \to D^*)$, then the gradients of the generator become zero.

Corollary (Arjovsky and Bottou, 2017, Cor. 2.1, p. 6). *Under the same assumptions of Theorem 2.4*

$$\lim_{||D-D^*||\to 0} \nabla_\theta \mathbb{E}_{z\sim p(z)}[\log(1-D(g_\theta(z)))] = 0$$

To sum it up, it can be said that many instabilities in GANs are a result of non-aligned, low-dimensional support between two distributions that allows for a perfect discriminator with the result of vanishing gradients.

There is an obvious "brute force method" to solve the problem related to the disjoint support of the probability distributions, i.e., to add noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\mathbb{1})$ to the data before feeding it to the discriminator (see Corollary 3.2 in Arjovsky and Bottou, 2017). The convolution of $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$ (summing random variables $\widetilde{\boldsymbol{x}} \sim \mathbb{P}*\mathcal{N}(0, \sigma^2\mathbb{1})$) significantly increases the support of the generated distribution; hence, reduced the risk of disjoint supports between $\mathbb{P}_r$ and $\mathbb{P}_g$. The implementation is very simple and only requires some minor modifications of the training scheme shown in Figure 3.10. Adding (explicit) continuous noise to the inputs of the discriminator significantly stabilizes the training of generative adversarial networks, with the drawback, of course, that the quality of the data reduces due to increased sampling variance.

3.6.2 Wasserstein GANs

The previous section introduced GANs as they have been proposed in 2014 by Goodfellow *et al.*, as well as some superficial considerations of its mathematical properties. Despite their unquestionable success, however, GANs are famously known for being notoriously unstable and hard to train. This instability is mainly due to the loss function (see Equation 3.47) and the underlying f-divergence that is minimized by the model during training, which corresponds to the Jensen-Shannon divergence.

This very problem is addressed in this section by introducing a new metric $W(\mathbb{P}_r, \mathbb{P}_g)$ to measure the similarity between the two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$, which does not represent an f-divergence but an *Integral Probability Metric* (IPM) instead; therefore, does not suffer the aforementioned vanishing (or exploding) gradient problem. This alternative cost function is the so-called *Wasserstein loss* (also known as *earth mover's distance*), which was first applied to GANs by Martin Arjovsky *et. al.* at the end of 2017 [Arjovsky *et al.*, 2017]. Since then, Wasserstein GANs (WGANs) have taken the machine learning community by storm. In line with this trend, this section provides a detailed introduction into Wasserstein GANs – which will be used through this thesis – along with their mathematical properties and, most importantly, the practical implementation through machine learning methods.

The Earth Mover's distance

This section provides a short derivation of the above mentioned Earth Mover's Distance (EMD) instead of just showing the final equation with a shallow explanation of its components. The first objective is to give an intuition of the underlying idea behind this metric without going into too much detail.

As it was shown in Section 3.5.1, the minimization of an f-divergence can be (loosely speaking) though of as maximum likelihood estimation of the generated distribution. From a probabilistic point of view, this is very intuitive. However, there are other ways to measure the similarity between distributions.

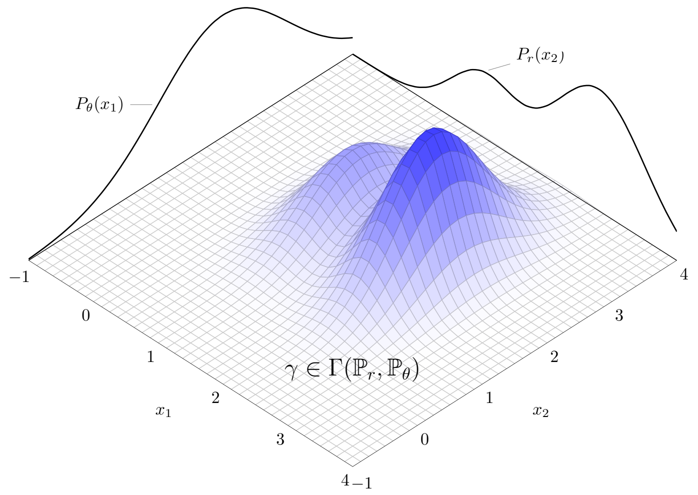


Fig. 3.11: An two-dimensional example for a possible transportation plan γ that corresponds to a joint probability distribution with the marginal distributions $\mathbb{P}_r$ and $\mathbb{P}_g$.

One possible approach, for instance, would be to define a measure of how much (probability) mass must be moved to transform one probability distribution into another one based on some cost function $c : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^{\geq 0}$, $(\boldsymbol{x}_1, \boldsymbol{x}_2) \mapsto c(\boldsymbol{x}_1, \boldsymbol{x}_2)$ with $c(\boldsymbol{x}, \boldsymbol{x}) = 0$ that gives the cost to "move" mass from $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$. The less mass needs to be moved, the closer the two distributions are; or, in other words: the lower *total cost* to transform the distribution $\mathbb{P}_\theta$ into $\mathbb{P}_r$, the more they resemble each other. The rule for moving mass from one distribution to the other is given by a *transportation plan* or a *coupling* γ. Formally, for this transportation plan to be reasonable, it must fulfill the requirement of continuity, i.e., the total change of mass in the system must be zero. Therefore, the infinitesimal mass $P_r(\boldsymbol{x}_1)\mathrm{d}\boldsymbol{x}$ removed from $P_r(\boldsymbol{x}_1)$ according to the transportation plan $\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2)$ must be equal to the mass $P_g(\boldsymbol{x}_2)\mathrm{d}\boldsymbol{x}$ added to $P_g(\boldsymbol{x}_2)$. This defines the requirements $P_r(\boldsymbol{x}) = \int \mathrm{d}\boldsymbol{x}'\, \gamma(\boldsymbol{x}', \boldsymbol{x})$ and $P_g(\boldsymbol{x}) = \int \mathrm{d}\boldsymbol{x}'\, \gamma(\boldsymbol{x}, \boldsymbol{x}')$; therefore, the transportation plan γ is a *joint probability* distribution whose univariate marginal distributions correspond to $\mathbb{P}_r$ and $\mathbb{P}_g$ respectively. Accordingly, the infinitesimal mass moved from $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$ is given by $\mathrm{d}\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2) = \gamma(\boldsymbol{x}_1, \boldsymbol{x}_2)\mathrm{d}\boldsymbol{x}_1\mathrm{d}\boldsymbol{x}_2$ with the associated infinitesimal cost $\mathrm{d}w(\boldsymbol{x}_1, \boldsymbol{x}_2) = c(\boldsymbol{x}_1, \boldsymbol{x}_2)\mathrm{d}\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2)$. The total cost of the respective transportation plan γ under consideration is then given by

$$C(\gamma) = \int\int \mathrm{d}\boldsymbol{x}_1\mathrm{d}\boldsymbol{x}_2\, c(\boldsymbol{x}_1, \boldsymbol{x}_2)\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{R}. \tag{3.56}$$

This is for solely one particular transportation plan; however, there is and infinite number of possible plans γ. The objective is to find the plan (the joint probability distribution) that corresponds to the *minimal* total cost. This is equivalent to finding the infimum over the possible space of *all* joint probability distributions $\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)$ with the univariate marginal distributions $\mathbb{P}_r$ and $\mathbb{P}_g$ with minimum total cost

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)} \int \mathrm{d}\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2)\, c(\boldsymbol{x}_1, \boldsymbol{x}_2), \tag{3.57}$$

or a more compressed and intuitive expression in terms of an expectation value

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} \left[c(\boldsymbol{x}_1, \boldsymbol{x}_2) \right], \tag{3.58}$$

with $W(\mathbb{P}_r, \mathbb{P}_\theta) \in \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)$. There are several possible options for the cost function c; a common choice is the standard Euclidean norm $c(\boldsymbol{x}_1, \boldsymbol{x}_2) := \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$. Finally, the Wasserstein metric is given by

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} \left[\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \right]. \tag{3.59}$$

The EMD and the Kantorovich-Rubinstein duality

The Wasserstein distance according to Equation 3.58 has an easy interpretation; however, it is highly intractable due to the infimum that requires the evaluation of the expectation value for *all* possible joint probability distributions between $\mathbb{P}_r$ and $\mathbb{P}_g$. Apart from that, the number of possible states scales *exponentially* with the input dimension of the data, making the calculation of γ NP-hard and therewith impossible. Furthermore, the solution to Equation 3.58 is a (discrete) probability distribution, whereas it would be aspirational to get a single number instead to measures the cost[4]. A more convenient representation of Equation 3.58, which is suited to be used in combination with backpropagation, is given by its *dual representation* resp. *dual form.* The transformation from the primal form to the dual form is not straightforward. The argument goes like this: the constrain $\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)$ in Equation 3.58 is removed by adding an optimization over a function $f : \mathbb{R}^N \to \mathbb{R}$ that removes all $\gamma \notin \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)$ (this insight is very important to understand the critic[5]/discriminator network in Wasserstein GANs)

$$\begin{aligned} W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma} & \mathbb{E}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} \left[\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \right] \\ & + \sup_f \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[f(\boldsymbol{x})] - (f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)), \end{aligned} \tag{3.60}$$

whereby the infimum is taken over the supremum as well. The second term is 0 if $\gamma \notin \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)$ and ∞ otherwise; hence, it selects all γ with $\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_\theta)$. It can be shown

[4]Hypothetically, the optimal transportation plan $\gamma^* := W(\mathbb{P}_r, \mathbb{P}_\theta)$ may be transformed to a scalar by some matrix norm $\|\gamma^*\|$. This norm must be differentiable to allow for the computation of gradients in the backpropagation algorithm. Furthermore, it must be unique!

[5]In case of Wasserstein GANs the discriminator is referred to as the *critic* network. The change of terminology is justified, since contrary to classical GANs the network f does not classify the data to "real" of "fake" samples but "filters" correct transportation plans. Therefore, it measures the Wasserstein distance between $\mathbb{P}_r$ and $\mathbb{P}_g$.

that in Equation 3.60 the order of the infimum and the supremum can be inverted without changing the outcome (this step is non-trivial)

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_f \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[f(\boldsymbol{x})]$$
$$\inf_\gamma \mathbb{E}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma}[\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|] - (f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)). \tag{3.61}$$

The second term defines a convexity condition. It can be shown that all Lipschitz continuous functions f provide the same solution. Therefore, the second term can be absorbed into a condition on f, i.e., the function must be Lipschitz continuous $f \in \text{Lip}_1$

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[f(\boldsymbol{x})]. \tag{3.62}$$

This result is known as the so-called *Kantorovich-Rubinstein duality* [Bonsall, 1966, Kantorovich and G.Sh.Rubinstein, 1958]. Writing Equation 3.62 in terms of the *generator* g_θ and the *critic* f_ϕ network gives the loss function for Wasserstein generative adversarial networks

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f_\phi\|_L \leq 1} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[f_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))], \tag{3.63}$$

with the following optimization task being solved

$$\min_{\boldsymbol{\theta}} \sup_{\|f_\phi\|_L \leq 1} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[f_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))]. \tag{3.64}$$

This form can “easily” be implemented using neural networks. The only difficulty here is to ensure that the network of the critic only learns a family of Lipschitz steady functions.

There is an entire smorgasbord of methods to ensure $f_\phi \in \text{Lip}_1$. In the original paper the authors clamped the weights of the critic's network to a compact space $[w_{\min}, w_{\max}]^N$ which implies global Lipschitz continuity. However, they admitted that “[w]eight clipping is a [...] terrible way to enforce a Lipschitz constraint[.]” (Arjovsky *et al.* [2017]), since the performance of the network is highly sensitive to the clipping interval. A more suitable way to ensure Lipschitz continuity is the so-called *Gradient Penalty* (GP), i.e., “[...] penalize the norm of gradient of the critic with respect to its input [...]” (Gulrajani *et al.* [2017]). This method as well as the training of WGANs is the subject of the next section.

Beforehand, it is worth to point out the relation between the integral probability metric 3.63 and the JS-divergences D_{JS} introduced in the previous section. This was later generalized to any f-divergence D_f (of which the JS-divergence is only one representative) in the so-called f-GANs [Nowozin *et al.*, 2016] by introducing a convex function f [Roth *et al.*, 2017]. It can be shown that the D_f divergence has an lower bound that can be seen by expressing $D_f(\mathbb{P}_r||\mathbb{P}_g)$ in terms of the *Radon-Nikodym derivative* $\frac{\mathrm{d}\mathbb{P}_r}{\mathrm{d}\mathbb{P}_g}$ and the Fenchel dual f^c of f [Nguyen *et al.*, 2008, Reid and Williamson, 2011] – which was already used to write Equation 3.59 in its dual representation

$$D_f(\mathbb{P}_r||\mathbb{P}_g) := \mathbb{E}_{\mathbb{P}_g}\left[f \circ \frac{\mathrm{d}\mathbb{P}_r}{\mathrm{d}\mathbb{P}_g}\right] = \int_{\mathcal{X}} \mathrm{d}\mathbb{P}_g \sup_u \left(u\frac{\mathrm{d}\mathbb{P}_r}{\mathrm{d}\mathbb{P}_g} - f^c(u)\right). \tag{3.65}$$

The lower bound on D_f in terms of an arbitrary class of statistics $\psi \in \Psi$ [Roth *et al.*, 2017] is given by

$$D_f(\mathbb{P}_r||\mathbb{P}_g) \geq \int_{\mathcal{X}} \mathrm{d}\mathbb{P}_g \sup_u \left(u\frac{\mathrm{d}\mathbb{P}_r}{\mathrm{d}\mathbb{P}_g} - f^c(u)\right) = \sup_\psi \{\mathbb{E}_{\mathbb{P}_r}[\psi] - \mathbb{E}_{\mathbb{P}_g}[f^c \circ \psi]\}. \tag{3.66}$$

This result is interesting and requires some reflection. On the right-hand side of Equation 3.66, a more general form of the Wasserstein distance can be identified (*cf.* Equation 3.62). Hence, the Wasserstein loss does represents a lower bond on the f-divergence that is minimized by classical GANs.

Furthermore, the Wasserstein distance is much "weaker" than, e.g., the Jensen-Shannon divergence, i.e., every sequence of distributions that convergence under the JS-divergence also converges under the EMD.

It should also be mentioned that the EMD is closely related to the so-called Maximum Mean Discrepancy (MMD) that is given by $\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \left\| \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_g}[\phi(\boldsymbol{x})] \right\|_{\mathcal{H}}$ with $\phi : \mathcal{X} \to \mathcal{H}$ being a feature map and $\mathcal{H}$ denoting the so-called reproducing kernel Hilbert space. Writing the MMD in an alternative representation $\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \sup_{f\in\mathcal{H}, \|f\|_{\mathcal{H}}\leq 1} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_g}[\phi(\boldsymbol{x})]$ with $\langle f, \phi(\boldsymbol{x})\rangle_{\mathcal{H}} = f(\boldsymbol{x})$, reveals the close connection to the Wasserstein distance. This relation will become important later when the statistical moments of the generated jets are studied.

Gradient penalty and training of WGANs

Gradient penalty is a method to ensure *local* Lipschitz continuity of f. It is based on the fact that if f^* is a 1-Lipschitz steady function which is the optimal solution (optimal critic) to $\max_{\|f\|_L\leq 1} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_g}[f(\boldsymbol{x})]$ with the optimal coupling $\pi \in \Gamma$, then $\mathbb{P}_{(\boldsymbol{x}_1,\boldsymbol{x}_1)\sim\pi}\left[\nabla f^*(\boldsymbol{x}_t) = \frac{\boldsymbol{x}_2-\boldsymbol{x}_t}{\|\boldsymbol{x}_2-\boldsymbol{x}_t\|}\right] = 1$ with $\boldsymbol{x}_t = t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2$ and $t \in [0,1]$. So, "[...] f^* has gradient norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$[.]" (Arjovsky *et al.*, 2017, Prop. 1, Cor. 1, p. 3). According to this statement, one has to sample points between $\mathbb{P}_r$ and $\mathbb{P}_g$ along a parameterized line and ensure the gradient norm of the critic to be close to one along this path – therefore, the Lipschitz constrain is only locally enforced[6]. The norm condition can be incorporated in the optimization task by adding an additional term to the loss 3.63 that penalizes the weights of the critic network if the gradient norm differs significantly from 1. This term is referred to as gradient penalty term

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))] + \lambda_{GP}\mathbb{E}_{\boldsymbol{x}'\sim\mathbb{P}_{\boldsymbol{x}'}}\left[\left(\|\nabla_{\boldsymbol{x}'} f_\phi(\boldsymbol{x}')\|_2 - 1\right)^2\right], \qquad (3.67)$$

whereby λ_{GP} is a hyperparameter that weights the contribution of the gradient penalty term [Arjovsky *et al.*, 2017]. With Equation 3.67 at our disposal, everything is well defined. The implementation does not represent a problem and cam mostly be adopted from classical GANs.

As one can see by comparing Figure 3.12 with Figure 3.10, the training scheme and the actual implementation of WGANs and GANs are very similar. One difference – besides the loss function – is the multiple updates of the critic's weights, indicated by the number of critic updates N_c in the Figure above. The hyperparameter N_c gives the number of backpropagation steps for the critic per generator update. It is necessary to train the critic several times before updating the model of the generator to get a precise approximation of the Wasserstein distance; otherwise, the Lipschitz constraint might not be fulfilled with the consequence of poorly estimated gradients. In the paper the authors proposed $N_c = 5$

[6] In 2018, a new weight normalization technique, called *spectral normalization*, was introduced [Miyato *et al.*, 2018] that allows to globally enforce the Lipschitz constrain. Within the context if this thesis, it was tried to combine spectral normalization with Wasserstein GANs. However, the resulting networks turned out to be highly unstable with significant oscillations in the loss function. This behavior is not yet understood; therefore, spectral normalization is not part of this record.

[Arjovsky *et al.*, 2017]; however, for the networks in this thesis, $N_c = 10$ have been found to be more appropriate.

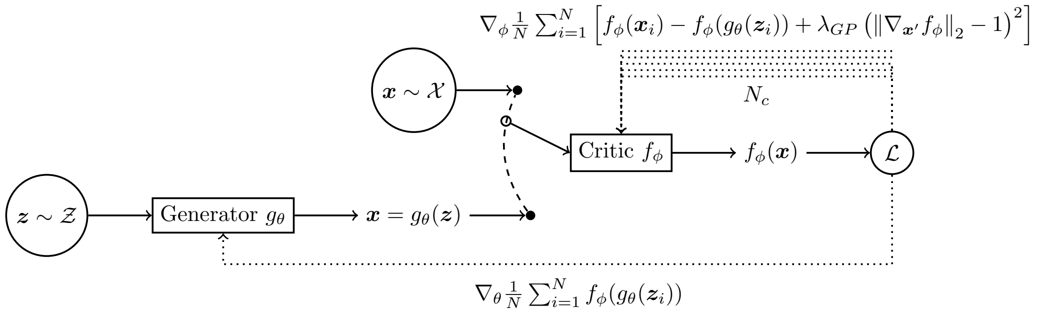

$\nabla_\phi \frac{1}{N}\sum_{i=1}^{N}\left[f_\phi(\boldsymbol{x}_i) - f_\phi(g_\theta(\boldsymbol{z}_i)) + \lambda_{GP}\left(\|\nabla_{\boldsymbol{x}'} f_\phi\|_2 - 1\right)^2\right]$
$\boldsymbol{x} \sim \mathcal{X}$
N_c
Critic f_ϕ
$f_\phi(\boldsymbol{x})$
$\mathcal{L}$
$\boldsymbol{z} \sim \mathcal{Z}$
Generator g_θ
$\boldsymbol{x} = g_\theta(\boldsymbol{z})$
$\nabla_\theta \frac{1}{N}\sum_{i=1}^{N} f_\phi(g_\theta(\boldsymbol{z}_i))$

Fig. 3.12: Graphical illustration of the training of generative adversarial networks with the Wasserstein metric and gradient penalty as introduced by Arjovsky *et al.*, 2017. The critic model/network f_ϕ measures the Earth mover's distance between the generated distribution $\mathbb{P}_g$ and the real distribution of the training data $\mathbb{P}_r$.

Despite all obvious similarities, it is important to be aware of the substantial difference. First, the output of the critic (the discriminator in case of GANs) does *not* correspond to a probability; hence, the image set is not restricted to $[0, 1]$ but can take on any real number. Moreover, the Wasserstein distance is the subtraction between two terms instead of an addition; this minus sign changes the whole world: it reveals a close connection between the EMD and the MMD as already mentioned in the previous section.

A serious issue with classical GANs was the problem of vanishing or exploding gradients due to a discriminator network that is too powerful compared to the generator. Hence, the balance between the discriminator and the generator network regarding the complexity of the model must be carefully tuned. This problem does not occur in case of Wasserstein GANs due to the properties of the earth mover's distance. On the contrary, in WGANs the critic model should be as complex as possible to provide precise information of the gradients for the generator and to get a good approximation of the Wasserstein distance.

Conditional Wasserstein GANs

Like it was done in case of variational autoencoders (see Section 3.5.4), the generative adversarial networks used through this thesis are conditioned on external labels $\boldsymbol{y}$ (if not explicitly motioned otherwise). This requires some minor modifications of the loss function as well as of the architecture. The overall objective of this step is – again – to control certain properties of the data that is produced by the generative model. Thus, for instance, a model that is conditioned of the energy E^{jet} and pseudorapidity η^{jet} of the jet allows to generate jets in certain regions of phase space at specific locations in the detector. From a probabilistic perspective, conditioning the generative model on a set of *features* $\{\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ *owned* by the data corresponds to a factorization of the (joint) probability density function

$$\begin{aligned} p(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N|\boldsymbol{z}) &= \frac{p(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N, \boldsymbol{z})}{p(\boldsymbol{z})}, \\ &= \frac{1}{p(\boldsymbol{z})} p(\boldsymbol{x}|\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N, \boldsymbol{z}) p(\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N, \boldsymbol{z}), \end{aligned}$$

$$= p(\boldsymbol{x}|\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N, \boldsymbol{z}) \cdot \prod_{i=1}^{N} p\left(\boldsymbol{y}_i \Big| \bigcap_{j=1}^{i-1} \boldsymbol{y}_j \cap \boldsymbol{z}\right), \tag{3.68}$$

which also takes correlations between the various labels into account (e.g. $p(\eta^{\text{jet}}|p_T^{\text{jet}})$ etc.). Equation 3.68 now corresponds to $N+1$ (generative) models. In order to condition the generative model $p(\boldsymbol{x}|\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N, \boldsymbol{z})$ on the additional information, the conditioning labels $\{\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ must be provided to the generator and the discriminator/critic network during training – as it was done for the decoder and encoder model in case of variational autoencoders in Section 3.5.4.

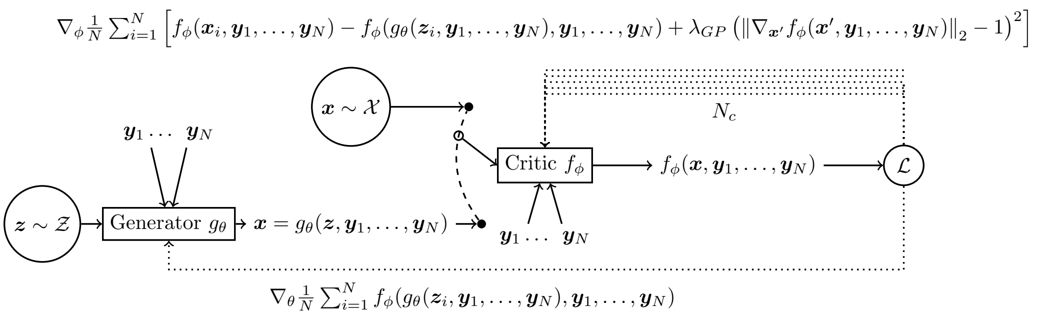

$\nabla_\phi \frac{1}{N} \sum_{i=1}^N \left[f_\phi(\boldsymbol{x}_i, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) - f_\phi(g_\theta(\boldsymbol{z}_i, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N), \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) + \lambda_{GP}\left(\|\nabla_{\boldsymbol{x}'} f_\phi(\boldsymbol{x}', \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)\|_2 - 1\right)^2\right]$
$\boldsymbol{x} \sim \mathcal{X}$
N_c
$\boldsymbol{y}_1 \cdots \boldsymbol{y}_N$
Critic f_ϕ
$f_\phi(\boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$
$\mathcal{L}$
$\boldsymbol{z} \sim \mathcal{Z}$
Generator g_θ
$\boldsymbol{x} = g_\theta(\boldsymbol{z}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$
$\boldsymbol{y}_1 \cdots \boldsymbol{y}_N$
$\nabla_\theta \frac{1}{N} \sum_{i=1}^N f_\phi(g_\theta(\boldsymbol{z}_i, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N), \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$

Fig. 3.13: Graphical illustration of the training of conditional generative adversarial networks with the Wasserstein metric and gradient penalty. Both models, the generator and the critic, additionally receive a list of conditioning label $\boldsymbol{y}_1, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$.

The training of Conditional Wasserstein GANs (CWGANs) is illustrated in Figure 3.13. The labels itself are sampled from the respective probably distribution $\boldsymbol{y}_i \sim p\left(\boldsymbol{y}_i \big| \bigcap_{j=1}^{i-1} \boldsymbol{y}_j \cap \boldsymbol{z}\right)$, $\boldsymbol{y}_i \sim p\left(\boldsymbol{y}_i \big| \bigcap_{j=1}^{i-1} \boldsymbol{y}_j\right)$ or $\boldsymbol{y}_i \sim p(\boldsymbol{y}_i)$ if independent of the latent space $\boldsymbol{z}$ or uncorrelated.

3.6.3 A kaleidoscope of GANs

Generative models that use an "adversary" for training have received a tremendous amount of attention not only by the machine learning community but from all kind of scientific fields. It is therefore scarcely surprising that, since their introduction in 2014, generative adversarial (neural) networks have undergone rapid development in many directions, such that it becomes quite difficult to retain an overview over the complex as a whole. To date, however, it appears that the curve of new publications and contributions starts to flatten off; nonetheless, the area remains very dynamic and full of verve.

The two previous sections only introduced two kinds of generative adversarial networks: first, GANs as proposed by Goodfellow *et al.* [Goodfellow *et al.*, 2014] as the pioneering paper that opened the "floodgates"; second, Wasserstein GANs proposed by Martin Arjovsky *et. al.* at the end of 2017 [Arjovsky *et al.*, 2017]. However, one and a half years are a donkey's year in the tremendously fast-moving field of machine learning. But, at some point, one has to stop chasing the current trends, fix the methods, with the disadvantage of never using true state-of-the-art techniques. However, "recent" developments in the field should not be left unmentioned.

As already mentioned above, there exists a large variety of different versions of GANs. They mostly differ concerning their objective or loss function to measure the similarity between the generated distribution and the underlying probability density of the training data. We've already encountered two classes of GANs: those that minimize an f-divergence between two distributions (e.g. Jensen-Shannon or KL-divergence) and the above-mentioned ones that minimize an integral probability metric (e.g. the earth mover's distance or, more generally, the MMD). Besides the aforementioned GANs [Goodfellow *et al.*, 2014], WGANs [Arjovsky *et al.*, 2017], improved WGANs with gradient penalty [Gulrajani *et al.*, 2017] or even the "[i]mproving [of] the [i]mproved [t]raining of Wasserstein GANs" by means of consistency regularization[7] (Wei *et al.* [2018]), the following versions are frequently encountered: MMD-GANs that complement generative moment matching networks by an adversary [Li *et al.*, 2017], Cramer GANs that introduce a new distance to correct for biased sample gradients in the Wasserstein loss [Bellemare *et al.*, 2017], Fisher GANs – that use an IPM as well – [Mroueh and Sercu, 2017], LSGANs (Least Squares GANs) that minimize the Pearson χ^2-divergence [Mao *et al.*, 2016], Energy-based GANs "[...] which views the discriminator as an energy function that attributes low energies to the regions near the data manifold [...]" (Zhao *et al.* [2016]), and much, much more.

This little foray into the jungle of different versions of generative adversarial networks not only finishes this chapter regarding machine learning but the introduction in this thesis as a whole. It is now about time to breathe life into the theoretical concepts discussed in the last chapters and to amalgamate physics with concepts from machine learning.

[7]This method has been tried as well within the scope of this thesis – with limited success though. This is unfortunate since I appreciate the idea. In their paper, the authors proposed to extend the loss according to Equation 3.67 by another term called *consistency regularization*. The principle idea is to additionally enforce the 1-Lipschitz constraint by $\mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{x}_2\sim\mathbb{P}_{\boldsymbol{x}}}\left[\max\left(0,\frac{\mathrm{d}(f_\phi(\boldsymbol{x}_1),f_\phi(\boldsymbol{x}_2))}{\mathrm{d}(\boldsymbol{x}_1,\boldsymbol{x}_1)}\right)-M'\right]$ which is based on the property $\mathrm{d}(f_\phi(\boldsymbol{x}_1),f_\phi(\boldsymbol{x}_2))\leq M\mathrm{d}(\boldsymbol{x}_1,\boldsymbol{x}_2)$ common to all Lipschitz continuous functions. However, the method, despite being very well motivated, introduces several new hyperparameters that need to be tuned.

Chapter 4

Training Data and Preprocessing

The previous chapter provided a rather extensive insight into the thematic field of machine learning starting from the historical development of the discipline with its modest beginnings in the first half of the 20th century to highly advanced state-of-the-art concepts seen today. However, notwithstanding the complexity of contemporary machine learning methods, fundamental underlying principles have changed very little over time. Still, a model that is based on a neural network – regardless of being supervised or unsupervised – is supposed to learn a set of trainable parameters based on a finite subset of points sampled from a data space with an underlying distribution that might be *a priory* unknown. In the case of generative models, the distribution of the data ought to be modelled by a parameterized model implemented through a neural network. Thus, it is evident that the selection as well as the preparation of data used to train the model, is of great significance, which may be the decisive factor for success or failure of the model. It is therefore important to be aware of the underlying nature of data. This chapter intends to acknowledge this circumstance and to give account for the data that is used to train the neural networks that are described in this report. To this end, the first part of this chapter introduces the basic physical processes that are supposed to be modelled. This report is restricted to QCD (strong force only) and W initialized jets due to the clear and distinct difference in their respective jet substructure. Once the simulation and reconstruction of the events have been introduced, the second part focuses on the conversion of the energy distribution in the calorimeter cells of an idealized detector to an "image", which is then used to train the generative models. To improve the performance of the training routine, the data undergoes a number of further preparation steps, i.e., the preprocessing of the images to exploit as many inherent symmetries as possible with the objective to remove redundancies in the training set, which is a common practice in data analysis. The preprocessing of the data is important to speed up the convergence of the model (especially if the data has intrinsically complex structures), however, it is inevitably accompanied by "information loss". This loss of information will be investigated in the last part of this chapter – even though not of importance in the scope of this feasibility study.

4.1 Event simulation

To study and evaluate the methods presented in this thesis, a sterile environment is required that allows to perform tests under well-controlled "laboratory conditions". For this purpose, the data sets used to train neural networks have been generated with Monte Carlo event generators and shower Monte Carlo in preference of real data. As previously described in Chapter 2, the simulation of an event in high energy particle physics usually consists of (at least) two steps: the calculation of the matrix element of the underlying hard subprocess at fixed order in perturbation theory (e.g. LO or NLO) and, afterwards, the modeling of QCD radiation, i.e., gluon emission based on some dedicated parton shower algorithm (see Section 1.2.5).

At the LHC (like at all hadron-hadron colliding experiments) QCD is the dominant process with the largest contribution to the total cross-section σ_{tot}^{pp} for proton-proton interactions (see “jets” (R=0.4) in Figure 4.1).

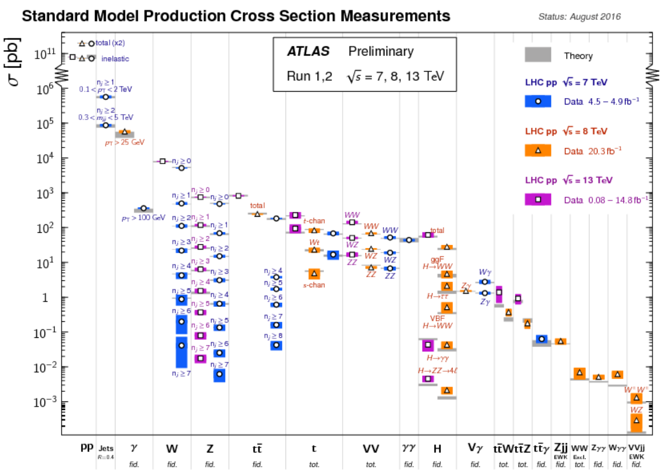


Fig. 4.1: Overview of several cross-section measurements of selected Standard Model processes compared to the corresponding theoretical expectations (adapted from Gemme, 2016, Fig. 1, p. 3).

Due to the large “contamination” of QCD events at hadron colliders, it is utterly important to provide accurate predictions of this omnipresent background to improve the sensitivity to potential signals whose cross-sections are usually orders of magnitude smaller. Therefore, with QCD being by far the most dominant contribution to the total cross section, it is only reasonable to study this process in more detail in the course of this thesis.

4.1.1 Underlying hard subprocess and parton shower

This section introduces the underlying physical processes of the simulated data that is used to train the neural networks presented in this report. Besides pure QCD – i.e. strong interactions exclusively –, $t\bar{t}$ events in the *fully-hadronic* final state have been simulated.

Pure QCD

The simulation of the underlying hard subprocess of the generated events is done with the latest version (2.6.6) of `MadGraph5_aMC@NLO` (MG5) [Alwall *et al.*, 2014b] for proton-proton interactions at leading order (to avoid complications in the matching and merging procedure) with a center-of-mass energy of $\sqrt{s} = 14\,\text{TeV}$, disregarding all interactions but QCD for the time being (MG5 command interface: `MG5_aMC>generate p p > j j`). Figure 4.2

depicts the essential LO tree-level s-channel Feynman diagrams for the simulated processes, ignoring t- and u-channel as well as time inverted diagrams.

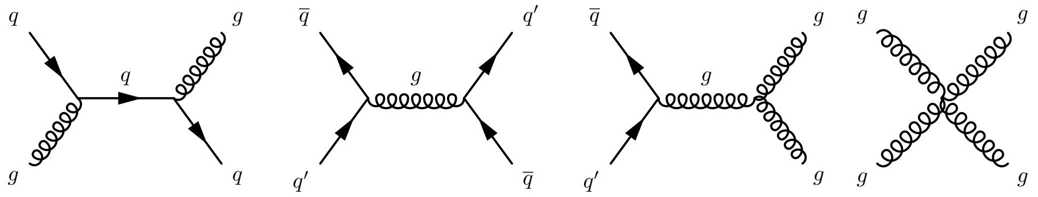


Fig. 4.2: Small selection of leading order tree-level Feynman diagrams for the processes $pp \to gq$+0jets, $pp \to q\bar{q}$+0jets, and $pp \to gg$+0jets for strong interactions solely (see Section 1.2).

The simulation of the hard interaction accounts for all quark flavors u, d, c, s, t and b; this result in a total number of 65 processes with 112 different diagrams that contribute to the total cross section σ_{jets}. Furthermore, an arbitrary p_{T}-cut $p_{\text{T,cut}}^{\text{jet}} = 100\,\text{GeV}$ and a pseudorapidity-cut $|\eta_{\text{cut}}^{\text{jet}}| \leq 5$ (which is MG's default) has been applied.

Electroweak

Additionally to the aforementioned simulated QCD processes (see Section 4.1.1), a LO $t\bar{t}$ sample has been generated for the fully-hadronic final-state of the decaying W^+ and W^- bosons[1] (MG5 command interface: `MG5_aMC>generate p p > t t , (t > W+ b, W+ > j j), (t  > W- b , W- > j- j))`) in order to evaluate the performance of the generative models for a jet topology that differs from pure QCD jets' ones. The LO Feynman diagram for $q\bar{q} \to b\bar{q}q'\bar{b}\bar{q}''q'''$ is shown in the figure below.

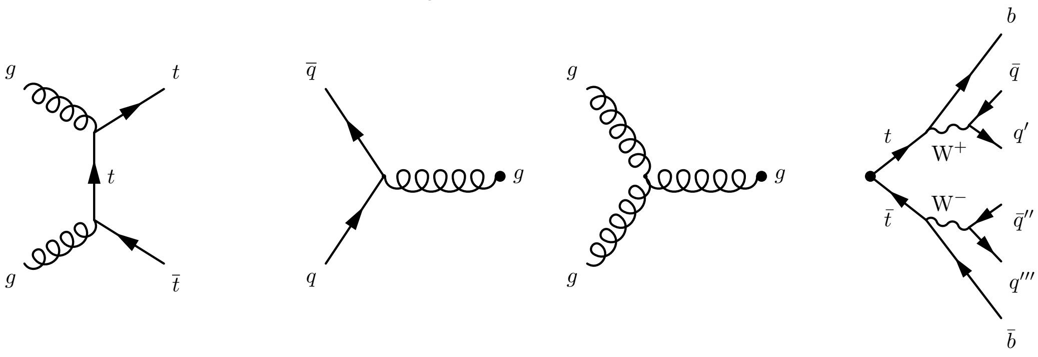


Fig. 4.3: Example of the leading order Feynman diagram(s) for the hadronic final state of the $t\bar{t}$ system for different initial states and interactions.

While QCD jets usually result in a diffuse spray of radiation that is distributed over the reconstructed jet radius R without an immediately apparent substructure (see Figure 1.2), the hadronic decay of the W into two quarks may result in a very different jet structure, depending on whether a resolved or a boosted topology is present. In a topology that is completely resolved, the number of jets in the final state it at least six. However, in the

[1]This is the dominant decay with a branching fraction of $\text{BR}(W \to q'\bar{q}) \approx 67.60$ [Eidelman *et al.*]).

boosted topology, where the transverse momentum of the W bosons is sufficiently large ($R \geq \frac{2m_W}{p_\mathrm{T}}$), the decay products of the W may be reconstructed within one single *large* jet (usually $R = 1$).

Parton shower simulation

The parton-level event information (see 2.2.4) from the matrix-elements-based calculation is saved in a file that based on the Les Houches format (Les Houches Event Files resp. Les Houches Accord (LHA)). `Pythia8.2` provides various interfaces to external matrix element event generators with LO and NLO matching available. To simulate QCD radiation by means of a parton shower model, the Les Houches file(s) from `MadGraph5_aMC@NLO` with parton-level event information are processed by `Pythia8.2` via the Les Houches interface. It is worthwhile noting that `Pythia8.2`, contrary to its previous version uses a dipole shower to simulate parton showers which also accounts for gluon coherence [Sjostrand *et al.*, 2008].

"Detector simulation" and event reconstruction

The next step is the "reconstruction" of the parton-level events with QCD radiation being simulated by Pythia. This done through an extremely simplified "detector simulation" that consists of nothing other than a (detector) grid with finite granularity in the η-ϕ plane, as well as a threshold cut $E_\mathrm{th}^\mathrm{pix}$, which accounts for the finite acceptance of the detector in each pixel. A cell in the detector is therefore associated with a position in $\eta_i^\mathrm{pix}, \phi_j^\mathrm{pix}$ as well as an energy deposition $E_{ij} \geq E_\mathrm{th}^\mathrm{pix}$ in that particular cell by one *or* more particles with $E_{ij}^\mathrm{pix} = 0$ if below $E_\mathrm{th}^\mathrm{pix}$. Furthermore, there is no magnetic field involved in the simulation. Hence, the parton-level information is used to populate the detector layer.

This idealized detector has some spatial extensions $\Delta\eta, \Delta\phi$ in the *eta*-ϕ space as well as a predefined number of pixels $n_\eta^\mathrm{pix}, n_\phi^\mathrm{pix}$ in each dimension. With the spatial extension and the given number of pixels the *granularity* of the detector is given by

$$\Delta\eta^\mathrm{pix}(\phi^\mathrm{pix}) = \frac{\Delta\eta(\phi)}{n_{\eta(\phi)}^\mathrm{pix}}. \tag{4.1}$$

In this case, the number of pixels in η and ϕ direction is not fixed but depends on the radius R that is used in the jet reconstruction algorithm to avoid artifacts in the reconstructed distributions. With this information available, the detector tower $(\eta^\mathrm{pix}, \phi^\mathrm{pix}, E^\mathrm{pix})$ is an element of the following set

$$\mathbb{D} := \left\{\tfrac{2|k|-1}{2}\mathrm{sgn}(k)\Delta\eta^\mathrm{pix}\right\}_{k=-\frac{n_\eta^\mathrm{pix}-1}{2}}^{\frac{n_\eta^\mathrm{pix}-1}{2}} \times \left\{\tfrac{2|\ell|-1}{2}\mathrm{sgn}(\ell)\Delta\phi^\mathrm{pix}\right\}_{\ell=-\frac{n_\phi^\mathrm{pix}-1}{2}}^{\frac{n_\phi^\mathrm{pix}-1}{2}} \times \mathbb{R}^{\geq E_\mathrm{th}^\mathrm{pix}}, \tag{4.2}$$

with $(\eta_i^\mathrm{pix}, \phi_j^\mathrm{pix}, E_{ij}^\mathrm{pix}) \in \mathbb{D}$ and the signum function $x = \mathrm{sgn}(x)|x|$ whereby $\mathrm{sgn}(0) \overset{!}{=} 0$. In the set 4.2, one recognizes that the description of this oversimplified detector is given by two contributions: the information which pixel/cell is active ($E^\mathrm{pix} \geq E_\mathrm{th}^\mathrm{pix}$) is given by the tuple $(\eta_i^\mathrm{pix}, \phi_j^\mathrm{pix})$ and a *continuous* energy value E_{ij}^pix that represents a regression task for the neural network. The detector defined by Equation 4.2 does only exhibit one individual layer, i.e., one layer of "radiation sensitive material" that absorbs all energy of the passing particle. An obvious extension that is closer to reality would be to include several detector layers. This, however, would require the simulation of the actual energy deposition in the material in each layer and is therefore beyond the scope of this thesis.

The four-momentum of the particles associated with the respective particle-level event is available and associated with one of the detector's η-ϕ cells according to the criterion $\left|(\eta_i^{\text{pix}} - \eta)/\Delta\eta^{\text{pix}}\right| < \frac{1}{2}$ and $\left|(\phi_j^{\text{pix}} - \phi)/\Delta\phi^{\text{pix}}\right| < \frac{1}{2}$, whereby η_i^{pix} and ϕ_j^{pix} give the *central* position of each pixel in the η-ϕ grid. The energy in the respective detector cell $(i,j) \cong (\eta_i^{\text{pix}}, \phi_j^{\text{pix}})$ is then simply given by $E_{ij}^{\text{pix}} = \Theta(E_{\text{th}}^{\text{pix}} - E_{ij})E_{ij}$, whereby $E_{ij} = \sum_{k\in=_{ij}} E_{ij,k}$ denotes the summed energy of all generated particles in the event that fall into the same pixel cell $(\eta_i^{\text{pix}}, \phi_j^{\text{pix}})$ according to the aforementioned resolution criterion. This procedure is repeated until all generated particles are processed and the detector is populated. After this step the event is fully discretized and ready for the actual reconstruction, i.e., the reconstruction of the jet.

The event information (position and energy of the particles in the detector) is now distributed over a grid with finite granularity according to Equation 4.2. The task now is to reconstruct the event from the detector information, i.e., cluster the energy distribution over the η-ϕ grid to jet candidates based on some sequential recombination algorithm as described in Chapter 1.3. For this purpose, the anti-k_t jet reconstruction algorithm (see 1.3.4) with a reconstruction radius of $R = 0.4$ (for QCD jets) and $R = 1.0$ (for W jets) was used. The actual reconstruction is done with the implementation of the anti-k_t algorithm in the software package FastJet [Cacciari *et al.*, 2012].

The (inclusive) algorithm returns a list of jet candidates with an associated register of constituent particles that constitute the respective jet. Within the scope of this study, only the *leading* p_T jet will be used for further analysis; all other reconstructed jets in the event are discarded for the sake of simplicity.

4.2 Data preprocessing

The previous Section gave a brief description of the data generation process as it is usually done in high-energy physics. After populating the idealized detector, reconstructing the event using FastJet's implementation of the anti-k_t algorithm, and discarding all identified jets except the leading p_T one[2], the result is only one jet in the final state that is used for further analysis. This reconstructed jet is associated with a collection of constituent p_i particles and a four-vector p^{jet} given by

$$p^{\text{jet}} = E^{\text{jet}} \left(1, \frac{\cos\phi^{\text{jet}}}{\cosh\eta^{\text{jet}}}, \frac{\sin\phi^{\text{jet}}}{\cosh\eta^{\text{jet}}}, \frac{\sinh\eta^{\text{jet}}}{\cosh\eta^{\text{jet}}}\right) \tag{4.3}$$

$$= \sum_{i\in I_{\text{jet}}} E_i^{\text{pix}} \left(1, \frac{\cos\phi_i^{\text{pix}}}{\cosh\eta_i^{\text{pix}}}, \frac{\sin\phi_i^{\text{pix}}}{\cosh\eta_i^{\text{pix}}}, \frac{\sinh\eta_i^{\text{pix}}}{\cosh\eta_i^{\text{pix}}}\right), \tag{4.4}$$

whereby the summation is taken over all constituent particles of the jet that have been clustered by the reconstruction algorithm.

In principle, the discretized leading p_T jet (discrete values of η and ϕ) is the information used to train the generative models in this report. However, as it is commonly done in data science, the preparation of the data is utterly important and may decide over success or failure of the respective method – neural networks certainly are no exception to this rule.

[2]Unless specified differently, the term "jet" within the scope of this report always refers to the leading p_T jet.

Hence, this Section introduces and explains the individual preparation or preprocessing steps of the data that are applied before providing it into the neural network. The objective is to exploit as many symmetries as possible that are encoded in the data. This procedure makes the learning task of the neural networks much more efficient since it does not have to learn redundant symmetries but may focus on the main features of the data. Furthermore, taking symmetries into account results in faster convergence and allows for a smaller data set to be used. But one should consider that preprocessing of data often is accompanied by information loss to a certain extent, a topic discussed after the introduction of the individual preprocessing steps.

In the next paragraphs the following preprocessing steps are discussed: Lorentz transformation (boost) of the leading p_{T} jet, a cut window of the detector, rotation and interpolation of the pixelated image, scaling of the energy, and parity transformation.

Step 1: Lorentz boost and rotation

The reconstructed jet according to Equation 4.3 (as well as its constituents) is associated with some position in the detector, i.e., the discretized η-ϕ grid. To prepare the training set for the neural network, it is advisable to "standardize" the data to a certain extent. Therefore, the four-vector of the reconstructed jet p^{jet} is transformed via a Lorentz transformation $\Lambda = (\Lambda^{\alpha}_{\beta})$ (strictly speaking a Lorentz boost) followed by a simple rotation R in Euclidean space such that the spatial position of its leading p_{T} subjet[3] is located at $(0, 0)$ in the η-ϕ grid. To this end, the four-momentum p_i of the jet's constituent particles must be transformed accordingly. With this in mind, the transformation is given by

$$\begin{aligned}
p^{\mathrm{jet}\prime} &= R\Lambda\, p^{\mathrm{jet}} && (4.5)\\
&= \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & \cos\phi^{\mathrm{rot}} & -\sin\phi^{\mathrm{rot}} & 0\\ 0 & \sin\phi^{\mathrm{rot}} & \cos\phi^{\mathrm{rot}} & 0\\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \gamma^{\mathrm{b}} & 0 & 0 & -\beta^{\mathrm{b}}\gamma^{\mathrm{b}}\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -\beta^{\mathrm{b}}\gamma^{\mathrm{b}} & 0 & 0 & \gamma^{\mathrm{b}} \end{bmatrix} \cdot \begin{bmatrix} E^{\mathrm{jet}}\\ p_x^{\mathrm{jet}}\\ p_y^{\mathrm{jet}}\\ p_z^{\mathrm{jet}} \end{bmatrix}, && (4.6)\\
&= \begin{bmatrix} \cosh\eta^{\mathrm{b}} & 0 & 0 & -\sinh\eta^{\mathrm{b}}\\ 0 & \cos\phi^{\mathrm{rot}} & -\sin\phi^{\mathrm{rot}} & 0\\ 0 & \sin\phi^{\mathrm{rot}} & \cos\phi^{\mathrm{rot}} & 0\\ -\sinh\eta^{\mathrm{b}} & 0 & 0 & \cosh\eta^{\mathrm{b}} \end{bmatrix} \cdot \begin{bmatrix} E^{\mathrm{jet}}\\ p_x^{\mathrm{jet}}\\ p_y^{\mathrm{jet}}\\ p_z^{\mathrm{jet}} \end{bmatrix}, && (4.7)\\
&\overset{(4.3)}{=} \sum_{i\in I_{\mathrm{jet}}} (R\Lambda\, p_i) = \sum_{i\in I_{\mathrm{jet}}} p_i', && (4.8)
\end{aligned}$$

whereby $\gamma^{\mathrm{b}} = \tanh\eta^{\mathrm{b}}$ and $\gamma^{\mathrm{b}}\beta^{\mathrm{b}} = \sinh\eta^{\mathrm{b}}$ have been used. It remains to determine the boost parameter η^{b} and the rotation angle ϕ^{rot} to fully define the transformation above. From the matrix multiplication in Equation 4.6 one gets the following condition on the energy $E^{\mathrm{jet}\prime}$ and the z component $p_z^{\mathrm{jet}\prime}$ of the jet after the transformation

$$p_z^{\mathrm{jet}} \cosh\eta^{\mathrm{b}} - E^{\mathrm{jet}} \sinh\eta^{\mathrm{b}} = p_z^{\mathrm{jet}\prime} \overset{!}{=} 0. \qquad (4.9)$$

Equation 4.9 defines the boost parameter $\tanh\eta^{\mathrm{b}} = \beta^{\mathrm{b}} = p_z^{\mathrm{jet}}/E^{\mathrm{jet}} = \tanh\eta^{\mathrm{jet}}$. Similarly, the defining condition for the rotation angle is $p_x^{\mathrm{jet}}\cos\phi^{\mathrm{rot}} + p_y^{\mathrm{jet}}\sin\phi^{\mathrm{rot}} = 0$; hence, the vector in the transverse plane must be rotated by $\phi^{\mathrm{rot}} = -\arctan\left(p_y^{\mathrm{jet}}/p_x^{\mathrm{jet}}\right)$.

[3]If there is no *subjet*, then the barycenter of the reconstructed jet is transformed to the origin.

The transformation according to Equation 4.8 defined by η^{b}, ϕ^{rot} is applied to *all* constituents that make up the jet. Hence – by construction – the reconstructed object, i.e., the leading p_{T} subjet will be centered in the η-ϕ plane.

Instead of transforming the leading p_{T} subjet of the reconstructed leading p_{T} jet to the origin of the detector, one could decide to only transform the barycenter of the jet irregardless of the substructure. In fact, this should be the preferred solution since it guarantees that all radiation remains within the radius R around the center. In case of QCD, the two transformation schemes, leading p_{T} subjet and barycenter, are approximately the same due to the structure of the jet that has most of its radiation cumulated in the center. However, the situation is very different for W initialized jets. In this case, the leading p_{T} subjet and barycenter of the jet differs significantly; hence, most of the radiation is not necessarily contained in the center of the detector.

Step 2: cut window

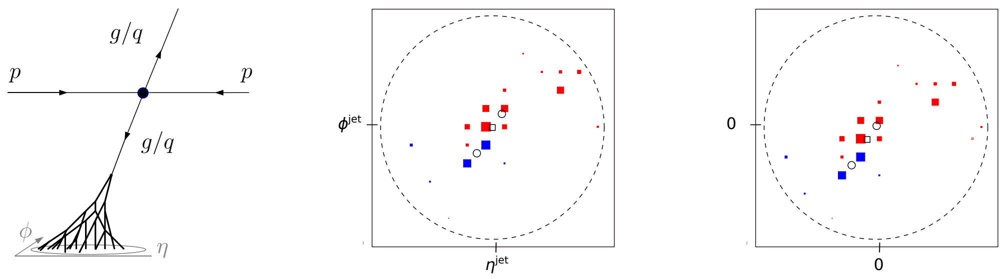


Fig. 4.4: First and second preprocessing step: a Lorentz boost as well as a rotation in the transverse plane are applied to transform the leading p_{T} (sub)jet to the origin of the detector $(0, 0)$ in the η-ϕ grid. Afterwards, a small region of the detector is selected according to condition 4.11 (adapted from Thaler and Van Tilburg, 2011, Fig. 1 (c,d), p. 4).

After the transformation of the leading p_{T} (sub)jet using a Lorentz boost and a spatial rotation, the jet is centered in the detector at position $(0, 0)$. Nevertheless, the discretized "image" of the event still expands over the entire detector region in η and ϕ direction. However, using the complete detector would not be very conducive since the transformation displaces the entire structure into a cone with radius R around the origin; hence, leaving almost all cells outside this range empty (remember: all other jets are discarded). Furthermore, the entire detector area comes with a large number of pixels $(n_{\eta}^{\mathrm{pix}} \cdot n_{\eta}^{\mathrm{pix}} \sim \mathcal{O}(10^4))$ in each dimension $n_{\eta(\phi)}^{\mathrm{pix}}$. Training neural networks on a data set that is based on the entire detector region would, therefore, be computationally very expensive. It would also be accompanied by significant memory consumption, limiting the applicability of batch training (3.4.2), which is very important to obtain precise estimates of the gradients. Cutting out a selected region of interest in the detector is therefore unavoidable.

Generally, care should be taken in this step to prevent spoiling the jet definition given in Chapter 1.3. To ensure infrared and collinear safety of the jet definition, the cut-out window must not be smaller than $2R$. The window (zoomed version of the event) should have $n_{\eta}^{\mathrm{pix}\prime}$ and $n_{\phi}^{\mathrm{pix}\prime}$ pixels in η respectively ϕ direction. To avoid artifacts in the image (boundary or discretization effects), it is naturally required that the granularity of the detector $\Delta\eta^{\text{pix}}(\phi^{\text{pix}})$ and the cut-out window $\Delta\eta^{\text{pix}\prime}(\phi^{\text{pix}\prime})$ are identical $\Delta\eta^{\text{pix}}(\phi^{\text{pix}}) \overset{!}{=} \Delta\eta^{\text{pix}\prime}(\phi^{\text{pix}\prime})$. This requirement defines two relations for η and ϕ that must be met

$$\frac{\Delta\eta(\phi)}{n^{\text{pix}}_{\eta(\phi)}} = \frac{\Delta\eta'(\phi')}{n^{\text{pix}\prime}_{\eta(\phi)}}. \tag{4.10}$$

The full detector range $\Delta\eta(\phi)$, the section $\Delta\eta'(\phi')$, as well as the number of pixels $n^{\text{pix}\prime}_{\eta(\phi)}(= 25)$ of the window is fixed; hence, the number of detector pixels in each dimension is given by

$$n^{\text{pix}}_{\eta(\phi)} = n^{\text{pix}\prime}_{\eta(\phi)} \cdot \frac{\Delta\eta(\phi)}{\Delta\eta'(\phi')} = n^{\text{pix}\prime}_{\eta(\phi)} \cdot \frac{\Delta\eta(\phi)}{2R+\delta}, \tag{4.11}$$

with $\delta \geq 0$ being some offset to extend the view. With this definition the cut-window is a subset of the set 4.2, i.e., the detector.

Step 3: image rotation, interpolation and scaling

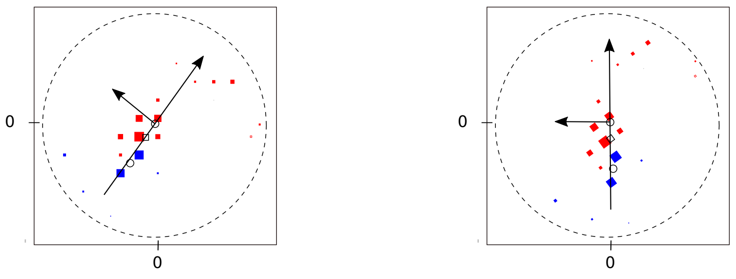


Fig. 4.5: Third preprocessing step: the image is rotated by $\alpha^{\text{PC}}_{\text{rot}}$ such that the first principal component axes points at at 12 o'clock (adapted from Thaler and Van Tilburg, 2011, Fig. 1 (c,d), p. 4).

After the preprocessing steps one and two, the result is a section of the detector's center that contains the transformed constituents of the leading p_{T} jet. The third preprocessing step serves the purpose of taking advantage of the underlying rotational symmetry of the image by conducting a *Principal Component Analysis* (PCA) that finds the principal components of the data – the jet image in this case – on an *event-on-event* base.

The PCA is a technique that is frequently used in data science. Its objective is to find the axis of maximum variance in a distribution of data points by solving an eigenvalue problem. Often, a PCA is used to reduce the dimensionality of the data by projecting it on the principal axis; here, however, the PCA defines a "special axis" that is used to rotate each jet image on an event-on-event base such that the principal component(axis) (PC) of this particular image points at 12 o'clock (see Figure 4.5). The PCA then provides the angle $\alpha^{\text{PC}}_{\text{rot}}$ by which the image must be rotated counterclockwise.

The eigenvalue problem that must be solved in context of the PCA is a very simple one. As it has been described above, after the preprocessing steps one and two the jet is given by a an image with a total number of $N^{\text{pix}} = n^{\text{pix}\prime}_{\eta} \cdot n^{\text{pix}\prime}_{\phi}(= 625)$ pixels. Each particle has a position $(\eta^{\text{pix}}_i, \phi^{\text{pix}}_i)$ in the grid (see Section 4.1) as well as an associated energy value E_i. To simplify the notation, the labeling of the detector towers has been temporarily changed from $(\eta_i^{\text{pix}}, \phi_j^{\text{pix}}, E_{ij}^{\text{pix}})$ with $(i,j) \in \mathbb{N}^{\geq n_\eta^{\text{pix}}} \times \mathbb{N}^{\geq n_\phi^{\text{pix}}} \setminus \{(0,0)\}$ to the equivalent representation $(\eta_i^{\text{pix}}, \phi_i^{\text{pix}}, E_i^{\text{pix}})$ with $i \in \mathbb{N}^{\geq n_\eta^{\text{pix}} \cdot n_\phi^{\text{pix}}} \setminus \{0\}$. The mean position/barycenter in the image is given by the weighed sum $\mu_{\eta(\phi)} = \sum_{i=1}^{N^{\text{pix}}} E_i \eta_i(\phi_i) / \sum_{i=1}^{N^{\text{pix}}} E_i$. The next step is to compute the entries of the correlation matrix Σ_{ij} which is given by $\Sigma_{ij} = \mathbb{E}[E_i^{\text{pix}} E_j^{\text{pix}}] - \mu_i \mu_j$ with $i,j \in \{\eta, \phi\}$. To get the fist principal axis, one needs to compute the eigenvalues λ^{PC} and eigenvectors $\boldsymbol{x}^{\text{PC}}$ of the covariance matrix $\det|\Sigma - \lambda^{\text{PC}} \mathbb{1}^{2\times 2}| = 0$. So, the characteristic polynomial is given by

$$\det \left| \begin{bmatrix} \Sigma_{\eta\eta} - \lambda^{\text{PC}} & \Sigma_{\eta\phi} \\ \Sigma_{\phi\eta} & \Sigma_{\phi\phi} - \lambda^{\text{PC}} \end{bmatrix} \right| = \left(\lambda^{\text{PC}}\right)^2 - 2\lambda^{\text{PC}} \Sigma_{\eta\phi} + \Sigma_{\eta\eta}\Sigma_{\phi\phi} - \Sigma_{\eta\phi}^2 = 0, \tag{4.12}$$

whereby the symmetry of the covariance matrix $\Sigma_{\eta\phi} = \Sigma_{\phi\eta}$ was used. The polynomial 4.12 can easy be solved for the eigenvalues $\lambda_{1,2}^{\text{PC}} = \Sigma_{\eta\phi} \pm \sqrt{2\Sigma_{\eta\phi}^2 - \Sigma_{\eta\eta}\Sigma_{\phi\phi}}$. On that basis, the kernel of Σ is computed; the elements of the kernel $\boldsymbol{x}^{\text{PC}} \in \ker(\Sigma)$ are

$$(\Sigma_{\eta\eta} x_1^{\text{PC}} - \lambda^{\text{PC}}) + \Sigma_{\eta\phi} x_2^{\text{PC}} = 0, \tag{4.13}$$

$$(\Sigma_{\phi\phi} x_2^{\text{PC}} - \lambda^{\text{PC}}) + \Sigma_{\phi\eta} x_1^{\text{PC}} = 0. \tag{4.14}$$

Based the solution of the system of linear Equations 4.13 and 4.14, the rotation angle for the image is $\alpha_{\text{rot}}^{\text{PC}} = \frac{\pi}{2} - \arctan\left(\frac{x_2^{\text{PC}}}{x_1^{\text{PC}}}\right)$.

The rotation causes a problem that is related to the finite granularity of the digitized detector. The grid of the rotated and the non-rotated images do not necessarily lie on top of each other (see Figure 4.6); hence, it is necessary to perform some kind of interpolation between neighboring pixels.

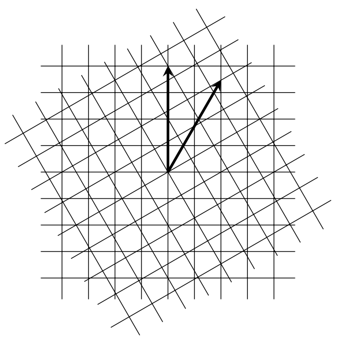


Fig. 4.6: Overlaying pixels between the rotated and non-rotated image.

There are several interpolation techniques available such as the (k-)nearest neighbors algorithm or spline interpolation. In this thesis, a cubic spline interpolation was used to smoothly interpolate between neighboring pixels. The interpolation – apart from the nearest neighbors interpolation – obviously causes another complication, since it changes ("smears") the energy content, i.e., the total energy sum $E_i^{\text{img}} = \sum_{i=1}^{N^{\text{pix}}} E_i^{\text{pix}}$ of the image. To avoid this problem, the image is normalized and scaled by the initial energy after the rotation has been performed.

Step 4: party transformation

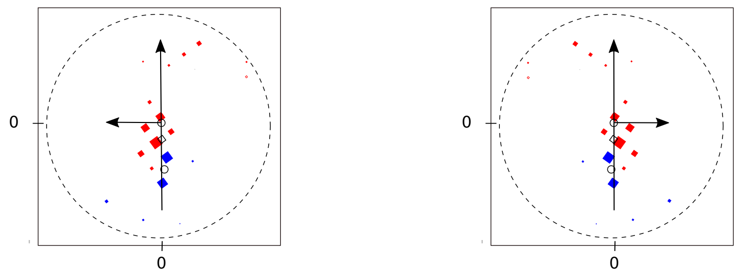


Fig. 4.7: A parity transformation $\boldsymbol{P}$ is applied such that the $E_r^{\text{img}} > E_l^{\text{img}}$, i.e., the largest energy fraction is always on the right-half of the plane (adapted from Thaler and Van Tilburg, 2011, Fig. 1 (c,d), p. 4).

After the rotation of the image, there is one trivial symmetry left: spatial inversion or parity transformation (this can already be seen in the previous paragraph since the basis given by the PCA is only defined up to a sign). To avoid this underlying symmetry of the data to be learned by the neural network, the image is transformed such that the *largest fraction* of energy is always placed on the *right* half of the image $\boldsymbol{M}^{\text{img}} \in \mathbb{D}$. The energy fraction on the left E_l^{img} and the right E_r^{img} half of the image is given by

$$E_{l(r)}^{\text{img}} = \sum_{\substack{1 \le i < \lfloor n_\eta^{\text{pix}}/2 \rfloor \\ \left(\lfloor n_\eta^{\text{pix}}/2 \rfloor \le i \le n_\eta^{\text{pix}}\right)}} \sum_{1 \le j < \lfloor n_\phi^{\text{pix}}/2 \rfloor} E_{ij}^{\text{pix}} . \tag{4.15}$$

Based on Equation 4.15, the parity operation is defined as

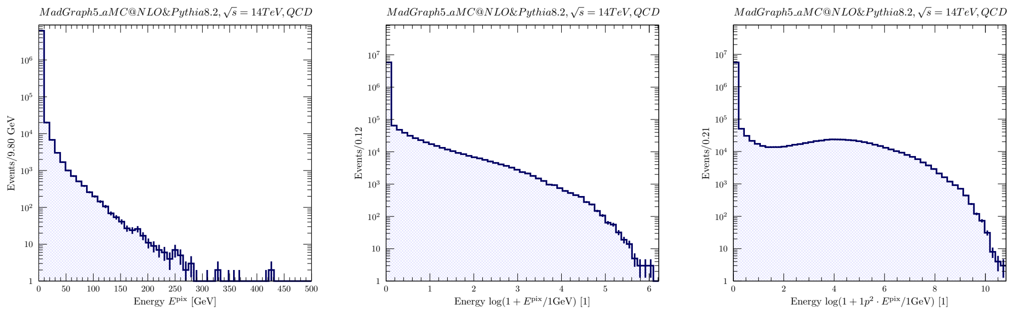
$$P = \begin{cases} 1 & \text{if } E_r^{\text{img}} \ge E_l^{\text{img}} \\ -1 & \text{otherwise} \end{cases} , \tag{4.16}$$

with the parity transformation matrix $\boldsymbol{P} = \frac{1}{2}(1+P)\mathbb{1}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}} + \frac{1}{2}(1-P)\mathbb{1}_{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}}$ (whereby $\mathbb{1}_{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}}$ denotes the anti-diagonal unit matrix).

4.3 "Invertible preprocessing"

The previous Section introduced a set of preprocessing steps that are applied to the data *before* the training routine. However, besides this preparation procedure, there are further modifications of the data prior to the actual training of the networks. These data manipulations differ from those defined in Section 4.2 through their property to be *exactly* invertible on an event-on-event base, while the others can only statistically be inverted by sampling rotation angles $\alpha_{\text{rot}}^{\text{PC}} \sim \mathbb{P}_\alpha$ and parity values $\mathrm{P} \sim \mathbb{P}_{\mathrm{P}}$ if both quantities are uncorrelated, i.e., $P(\alpha_{\text{rot}}^{\text{PC}}, \mathrm{P}) = P(\alpha_{\text{rot}}^{\text{PC}})P(\mathrm{P})$. Those data transformations, henceforth referred to as *invertible preprocessing*, serve the main purpose to scale the data such that the information can be well processed by the respective algorithms, "[s]ince networks prefer small numbers[...] between 0 and 1" (Kasieczka *et al.* [2017]). Besides the scaling, the functions φ ϑ are applied to the data whose characteristics and purpose are discussed in this Section.

The jet images obtained after the aforementioned preprocessing steps presents a challenge to neural networks because: first, the data is *extremely* sparse (see the universal approximation theorem 3.1 and Section 4.4.2), and second, the neural network has to learn a large range of energy values in the pixels that extend over several orders of magnitude. As a matter of fact, the neural network has to learn an energy distribution for each individual pixel as well as their correlations among each other. These two characteristics alone put a challenge on all state-of-the-art neural networks. The situation is illustrate in Figure 4.1 that shows the distribution of energy values in the pixel E_{ij}^{pix} with $i, j \in \mathbb{N}^{\leq 25} \setminus \{0\}$ for a *linear scale* (4.1a) and a *logarithmic scale* with $E_{ij}^{\text{pix}} \to \log(1 + E_{ij}^{\text{pix}})$ (4.1b) and $E_{ij}^{\text{pix}} \to \log(1 + 10^2 E_{ij}^{\text{pix}})$ (4.1c).



(a) Linear *(b)* $\log(1 + E^{\text{pix}})$ *(c)* $\log(1 + 10^2 E^{\text{pix}})$

Plot 4.1: Distribution of pixel intensity values E^{pix} for different scales.

The pronounced, dominant peak in the very first bin reflects the already mentioned low occupancy of the image with most pixels being empty. Besides that, the log-scale (Plot 4.1b and 4.1c) reveals some structure that is not immediately visible in the linear representation where most of the structure is hidden in the very first bins. It will be very difficult for the neural network to resolve the low energy contributions and learn the entire range of energy values that roughly covers four orders of magnitude. Therefore, in order to simplify the learning task of the network, the transformation φ is applied to *each* individual pixel in the image and its energy content E_{ij}^{pix} such that approximately $\widetilde{E}_{ij}^{\text{pix}} := \varphi(E_{ij}^{\text{pix}}) \sim \mathcal{O}(1)$ or below.

On the other hand, the conditional networks also receive two labels besides the latent space vector $\boldsymbol{z} \sim \mathcal{N}(0, \mathbb{1})$, i.e., E^{jet} and η^{jet} (see conditioned neural networks Section 3.5.4 and 3.6.2). However, usually the desired energy of the reconstructed jet E^{jet} will not be at the same order as the seed components z_i; therefore, a *linear* transformation ϑ is applied such that at least $\widetilde{E}^{\text{jet}} := \vartheta(E^{\text{jet}}) \sim \mathcal{O}(1)$.

The general transformations φ and θ introduced above create an “aesthetical flaw” that concerns the relation between the energy values in the pixels of the image E_{ij}^{pix} as well as the conditioning label E^{jet}. Since the network is conditioned, it should be able to reconstruct the energy of the jet – which is externally provided by E^{jet} in form of the conditioning label – from the pixel activations over the image. More precisely, there must be a function $\psi : \mathbb{D} \to \mathbb{R}$ with $E^{\text{img}} = \psi(\boldsymbol{M}^{\text{img}})$, whereby $\boldsymbol{M}^{\text{img}}$ is one instant, i.e., one event of the detector $\boldsymbol{M}^{\mathrm{img}} \in \mathbb{D}$. In the *non*-transformed system, this is simply $E^{\mathrm{img}} = \sum_{i=1}^{n_\eta^{\mathrm{pix}}} \sum_{j=1}^{n_\phi^{\mathrm{pix}}} E_{ij}^{\mathrm{pix}}$ (see Equation 1.24). The function ψ is learned by the network and might be non-trivial after all. The situation is simple if the transformation that is applied to the pixel values φ and the one applied to the conditioning energy ϑ are both homomorphisms, i.e., linear functions. In this case, the relation is given by

$$E^{\mathrm{img}} = \sum_{i,j} E_{ij}^{\mathrm{pix}}, \tag{4.17}$$

$$\vartheta^{-1}\left(\widetilde{E}^{\mathrm{img}}\right) = \sum_{i,j} \varphi^{-1}\left(\widetilde{E}_{ij}^{\mathrm{pix}}\right), \tag{4.18}$$

$$= \varphi^{-1}\left(\sum_{i,j} \widetilde{E}_{ij}^{\mathrm{pix}}\right), \tag{4.19}$$

whereby in the last step the linearity of φ was used. So the relation between the conditioning label and the pixel values in the transformed system is given by

$$\widetilde{E}^{\mathrm{img}} = \vartheta \circ \varphi^{-1}\left(\sum_{i,j} \widetilde{E}_{ij}^{\mathrm{pix}}\right) = \sum_{i,j} \vartheta \circ \varphi^{-1}\left(\widetilde{E}_{ij}^{\mathrm{pix}}\right). \tag{4.20}$$

This means, especially, that the natural relation between the energy of the jet and the constituents is preserved, with $\widetilde{E}^{\mathrm{img}} = \sum_{i,j} \widetilde{E}_{ij}^{\mathrm{pix}}$ if $\vartheta = \varphi$. However, a linear scale comes along with the aforementioned problems regarding the large range of energy values in the pixels; therefore, other, non-linear transformations might be more beneficial in respect of their impact on the training performance. One particular transformation – which is used in this thesis – alongside the linear one, is the logarithm with $\varphi(E_{ij}^{\mathrm{pix}}) = \log(1 + \varrho E_{ij}^{\mathrm{pix}})$ whereby $\varrho \in \mathbb{R}$. This transformation significantly helps regarding the possible range of energy values E_{ij}^{pix}, *but* breaks the simple relation between jet and pixel energy in Equation 4.20 due to the non-linear nature of the logarithm

$$\widetilde{E}^{\mathrm{img}} = \vartheta\left(\sum_{i,j} \varphi^{-1}\left(\widetilde{E}_{ij}^{\mathrm{pix}}\right)\right). \tag{4.21}$$

This is, as already mentioned above, more an aesthetical than a fundamental problem. Eventually, the network is supposed to be conditioned on the jet energy E^{jet}, i.e., $E^{\mathrm{jet}}/E^{\mathrm{img}} \approx 1$ – this relation needs to be verified later.

In order to enforce a more "intuitive" relation between jet-image and its corresponding label(s), one may consider to use an exponential transformation with an appropriate exponent. If both ϑ and φ are exponential, Equation 4.21 can be written as

$$\widetilde{E}^{\mathrm{img}} = \prod_{i,j} \vartheta \circ \varphi^{-1}\left(\widetilde{E}_{ij}^{\mathrm{pix}}\right), \tag{4.22}$$

with $\widetilde{E}^{\mathrm{img}} = \prod_{i,j} \widetilde{E}_{ij}^{\mathrm{pix}}$ if $\vartheta = \varphi$. This is an interesting structure; however, after several experiments, it has become clear that an exponential transformation is numerically highly unstable despite antecedent scaling of the exponent or using a logarithm. Hence, this idea was discarded.

Last but not least, the pseudorapidity η^{jet} of the jet, which is the second conditioning label of the model, is simply transformed according to $\eta^{\mathrm{jet}} \to \frac{1}{2}\eta^{\mathrm{jet}}$.

4.4 Training data

The preprocessing scheme introduced and explained in Section 4.2 is a standard choice in machine learning when dealing with data that exhibit an underlying rotational symmetry (see for instance de Oliveira *et al.* [2017] or Kasieczka *et al.* [2017], whereby the latter one used the preprocessing steps in the context of a classification task; hence, inversion is not required subsequently).

After the preprocessing steps have been applied, the preparation of the data is finished and ready to be used to train neural networks. The reasons for the preprocessing in the first place was to make the learning task of the network simpler by utilizing use of all trivial symmetries encoded in the data beforehand. This – hopefully – will have a positive impact on the performance (although it has a measurable effect) and the convergence behaviour and result in a significant speedup of the time needed to train the model. But, everything comes at a price: the loss of *information* to a certain extent. Of course, already the finite granularity of the measuring device will result in an inevitable loss of information since the resolution, as well as the acceptance of the detector, is fundamentally limited by the bounds of reality. This is not a hindrance since it reflects the ubiquitous imperfection of the measuring devices at our disposal in real-world experiments. Similarly, the training data for the neural network will be an image with a finite granularity that corresponds to some region of an idealized calorimeter with one layer. The rotation of the image, however, is quite problematic, precisely because of its limited resolution (see Figure 4.6). The interpolation between pixels in the rotated and non-rotated system changes the "activation value" E_{ij}^{pix} (energy tower) in the pixels as well as the number of pixels that are considered to be "active" $N_{E^{\text{pix}}>E_{\text{th}}^{\text{pix}}}^{\text{pix}}$ (hereinafter, the following abbreviation will be used: $N_0^{\text{pix}} := N_{E^{\text{pix}}>0}^{\text{pix}}$), i.e., number of "constituents" of the jet. This poses a serious problem – although not in the context of this thesis[4] – since many observables take the relative position between particles into account. Therefore, changing the number of constituents of the jet is very likely to spoil the reconstructed observables.

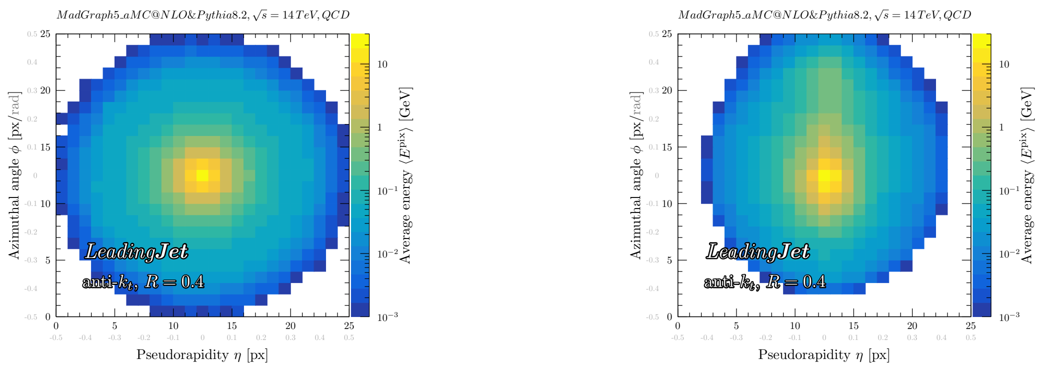
Since machine learning is a (solely) data-driven method, understanding the training data is very important. Therefore, the purpose of this Section is to provide a closer examination of the data that is used to train the neural networks presented in this report. Here the main issue is to study the effect of the individual steps in the preprocessing chain regarding the data and the loss of information.

4.4.1 Average jet image

Since the discretization of the event is unavoidable, it – along with the Lorentz boost that transforms the four-momentum of the jet to the center of the detector (furthermore referred to as "minimal preprocessing" or indicated by the first italic Roman numeral *I*) is the starting point to study the implications of the individual preprocessing steps in Section 4.2. As peviously discussed, training the model on the entire detector region is, in principle, possible but not recommended due to limited memory resources.

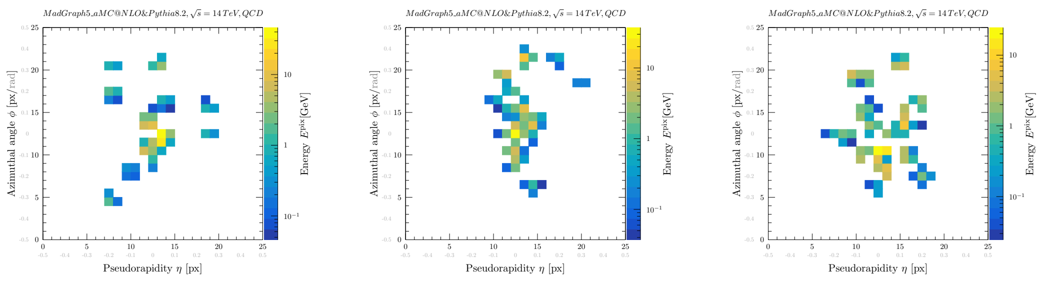
First of all, consider the *average reconstructed jet image* for the leading p_{T} QCD jet in Figure 4.2.

[4]In the context of this thesis, the information loss is not relevant since the generated samples are exclusively compared with the respective data from the actual training set. Therefore, everything is consistent (incidentally, this also applies to collinear and infrared safety of the jet).



Plot 4.2: Reconstructed average leading p_{T} QCD jet for minimal preprocessing (*left*) and full preprocessing (*right*) for 200,000 events.

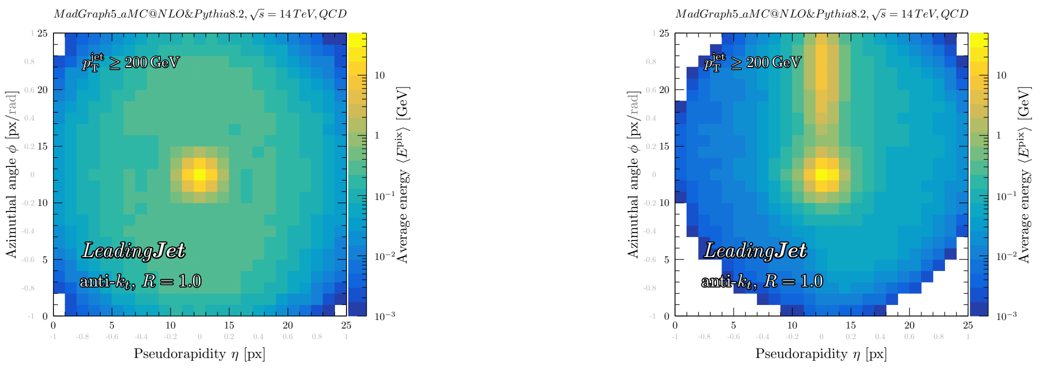
The jet images in Figure 4.2 correspond to the average of $N^{\text{event}} = 200,000$ individual images (single events like illustrated in Figure 4.3) whereby the energy entry $\bar{E}^{\text{pix}}_{ij}$ in each pixel in the average jet-image is given by the sample mean $\bar{E}^{\text{pix}}_{ij} = \frac{1}{N^{\text{event}}} \sum_{k=1}^{N^{\text{event}}} E^{\text{pix}}_{ij,k}$. One should keep in mind that the model learns a different energy distribution for *each* pixel whose shape will differ significantly for different pixels. Hence, the average energy $\bar{E}^{\text{pix}}_{ij}$ provides the sample mean approximation of the expectation value $\bar{E}^{\text{pix}}_{ij} \approx \mathbb{E}_{E \sim \mathbb{P}_{ij}}[E]$ of the energy distribution $\mathbb{P}_{ij}$ in pixel (i, j).



Plot 4.3: Three random QCD events.

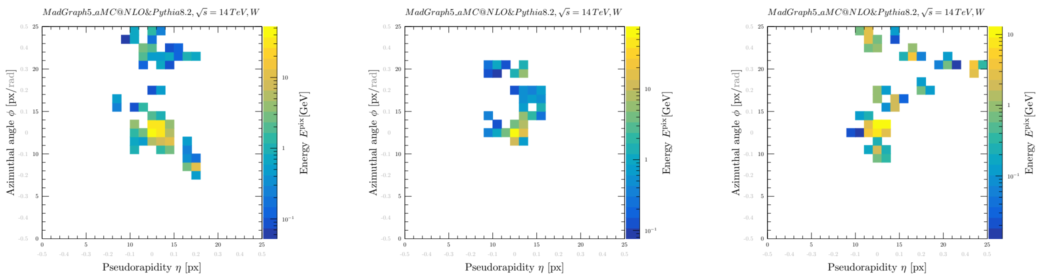
The images in Figure 4.3 show one characteristic of QCD initialized jets, i.e., the "diffuse" spray of radiation within the radius of the jet without any immediately obvious substructure (*cf.* N-subjettiness for QCD jets in Chapter 1.3.5). Due to this lack of substructure, the implications of the preprocessing steps are relatively small – though they are clearly visible. Therefore, the preprocessing just barely influences the training performance of the networks, and might as well be dropped (nonetheless, the preprocessing is applied since it still positively affects training time and reduces the size of the data set needed). Furthermore, Figure 4.3 and 4.2 show the large range of energy values in the individual pixels of the image as well as the aforementioned small occupancy in the periphery of the image.

The situation, however, is very different in case of W jets as can be seen in the Feynman diagram 4.3. The decay products are reconstructed in one large jet (provided that the radius R of the jet is sufficiently large). This results in a clear substructure within the jet's radius that consists of two isolated energy accumulations originating from the two quarks enclosed by soft radiation from the parton shower. Figure 4.4 shows again the reconstructed average jet-image for W jets and the two preprocessing configurations evaluated for $200,000$ events.



Plot 4.4: Reconstructed average leading p_T W jet for minimal preprocessing (*left*) and full preprocessing (*right*) for $200,000$ events and a p_T cut $p_T^{jet} \geq 200$ GeV.

The average image for W jets shows a clear substructure, which is expected for W jets, with two distinct contributions. As it can be shown based on simple kinematic considerations, for the decay products to be reconstructed within a jet of radius R, the p_T of jet should meet the condition $R \gtrsim 2m_W/p_T^{jet}$. Therefore, a p_T cut of 200 GeV has been applied on the reconstructed jet, justified by the reconstruction radius $R = 1.0$ used for the anti-k_t algorithm 1.3.4. The structure that is visible in Figure 4.4 does only manifest itself on average; for single events, however, it is barely visible.



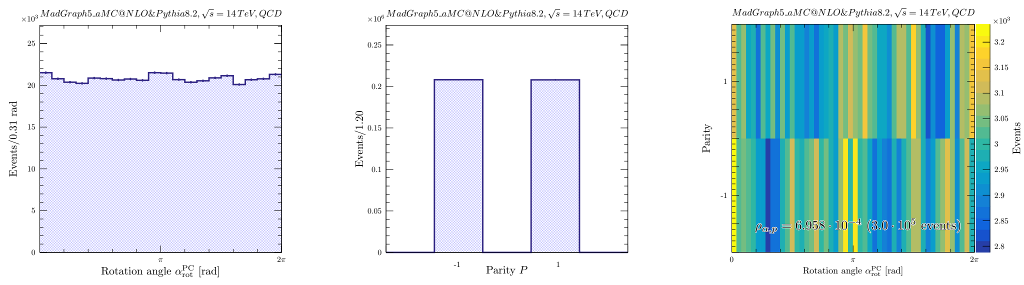
Plot 4.5: Three random W events.

While due to the lack of structure, QCD jets do not necessarily require preprocessing, the W initialized jets do. The abandonment of the preprocessing steps for the W data set results in a significant increase of the training time compared to the situation where preprocessing applied. This effect can be compensated with an enlarged data set and/or an increased number of training iterations. Due to the limited computer resources available, though – unless specifically stated otherwise –, the full preprocessing chain (steps 1 to 4 in Section 4.2) is applied for all data sets used in this report.

4.4.2 Information loss

The aforementioned image preprocessing inevitably gives rise to data loss mainly due to the interpolation between the pixels in the course of the rotation of the PC. This Section briefly studies the impact of the individual preprocessing steps on the data relative to the minimal configuration I, i.e., Lorentz boost and a zoomed detector view only.

First and foremost, with the preprocessing steps being applied, all the jet images generated by the neural network will be biased. A neural network that was trained on a data set with the full preprocessing chain applied will only generate jets with the principal component axis pointing at 12 o'clock (4.2) and the major energy fraction on the right side of the image $E_r^{\text{img}} > E_l^{\text{img}}$ (4.2). This does not represent a problem in the scope of this thesis since the preprocessing can easily be inverted by sampling from the distribution of the rotation angles $\alpha_{\text{rot}}^{\text{PC}}$ for the PCA and the parity values P to mirror the jet image. Both distributions can always be generated from the training set as shown in Figure 4.6.



Plot 4.6: Distribution of rotation angles $\alpha_{\text{rot}}^{\text{PC}}$ obtained from the PCA (4.2) (*left*), parity values P as defined in Section 4.16 (*middle*) and the correlation coefficient $\rho_{\alpha_{\text{rot}}^{\text{PC}},\text{P}}$ between $\alpha_{\text{rot}}^{\text{PC}}$ and P (*right*).

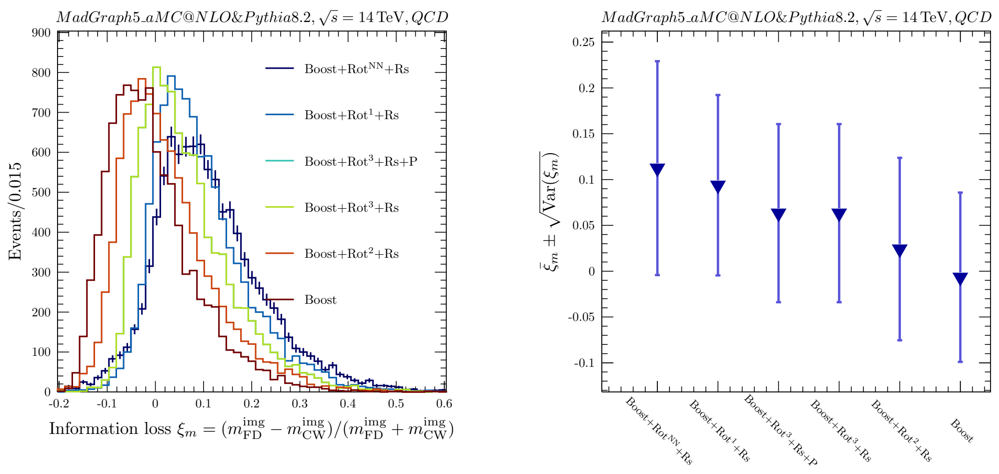
The distribution of the rotation angles and parity values in Figure 4.6 look as expected. Furthermore, the two quantities can be considered uncorrelated (though not necessarily independent) with a Pearson correlation coefficient of $\rho_{\alpha,P} = 0.6958‰$ (for 300,000 events) as can be seen in the rightmost correlation plot in Figure 4.6. So, to invert the preprocessing chain – if required at all –, all one has to do is to sample random variables from the two leftmost distributions in Figure 4.6. However, it shall not be left unmentioned that this approach exhibits some characteristic problems. For instance, while the inversion of the preprocessing is easily done for the leading p_{T} jet, it does not take int account the correlation between the leading and the subleading p_{T} jet that is discarded at this point. This is an inherent problem of the method, since the neural networks in this thesis are exclusively trained on the leading p_{T} jet. Therefore, all models are predestined to fail to learn correlation between different jets in an event like, e.g., color connection/flow [Collaboration, 2018]. Subsequent studies should aim to account for those effects which are encoded in the actual training data by allowing to train models on different jet multiplicities. But this is, unfortunately, beyond the scope of this thesis. Possible strategies of how to train neural network for a variable number of jets through RNNs are provides at the end of this report.

The effect of the individual preprocessing steps can clearly be seen in the change in the distribution of the jet observables such as, e.g., the mass, N-subjettiness etc. (the jet energy remains unchanged by construction if the image is (re-)scaled (Rs) after the rotation took place). The normalized distance

$$\xi^i_{O^{\mathrm{img}}} = \frac{O^{\mathrm{img}}_{\mathrm{FD}} - O^{\mathrm{img}}_{\mathrm{CW},i}}{O^{\mathrm{img}}_{\mathrm{FD}} + O^{\mathrm{img}}_{\mathrm{CW},i}} \in \mathbb{R}, \tag{4.23}$$

is defined as a proxy for the "information loss", whereby O^{img}_i refers to the reconstructed jet observables under consideration of the respective preprocessing steps introduced in Section 4.2. O^{img}_i on the other hand denotes the observable that was reconstructed based on the boosted jet with the full detector instead of a zoomed section.

The information loss in the reconstructed jet mass is summarized in Figure 4.7.



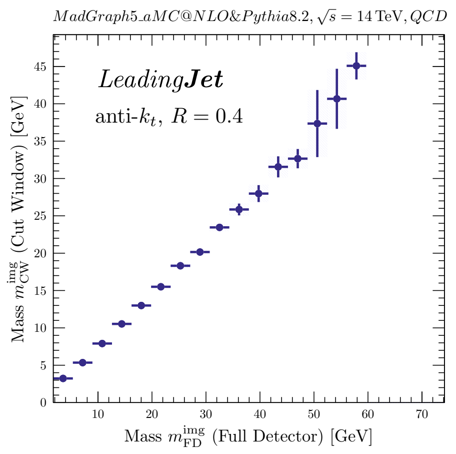
(a) ξ_m distribution *(b)* Average "information loss"

Plot 4.7: Loss of information for the reconstructed jet mass for different preprocessing configurations.

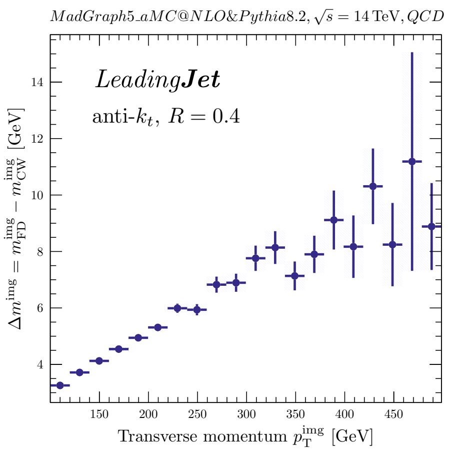
Figure 4.7a shows the distribution[5] ξ_m for different preprocessing steps while the graph in Figure 4.7b gives the mean as well as the Root Mean Square (RMS) error of the respective distribution. Rot$^{\mathrm{INTRPL}}$ indicates that the image has been rotated with the INTeRPoLation method Nearest Neighbor (NN) or spline interpolation like (bi)linear (1), quadratic (2) or cubic spline interpolation (3). The parity transformation (4.2) does not affect the distribution. This is expected since the parity operation preserves *all* relative angles between the constituents. However, it is hardly surprising that all image modifications that involve rotations significantly affect the shape of the mass distribution. The interpolation between pixels ((re-)pixelation) distributes/smears the energy content of one pixel-cell in the initial frame over several pixels in the rotated system; hence, it affects the relative angles and thus also the reconstructed mass. This effect is a consequence of the finite granularity of the detector and vanishes in the continuous limit $(\Delta\eta^{\mathrm{pix}}, \Delta\phi^{\mathrm{pix}}) \to (0,0)$ if $(n^{\mathrm{pix}}_\eta, n^{\mathrm{pix}}_\phi) \to (\infty, \infty)$. It may come as a surprise that the Lorentz boost gives rise to information loss in the mass spectrum, since the mass is a Lorentz invariant quantity and

[5]The bin with w_{bin} of the histogram was computed according to Freedman–Diaconis' rule [CIS, 1981] $w_{\mathrm{bin}} = 2\frac{\mathrm{IQR}(x)}{\sqrt[3]{n}}$, whereby IQR$(x)$ denotes the interquartile range of the data set x.

therefore remains unchained under boosts in beam direction. The apparent discrepancies can be explained by the first two preprocessing components in Section 4.2. According to the first step, not the leading p_{T} jet itself, i.e. its barycenter is centered in the detector but its leading p_{T} subjet (if it exists). This serves a particular purpose: centering the leading p_{T} subjet ensures that the highest region of activity is always shifted to center of the image. This step was crucial when working with *classical GANs* (according to Goodfellow *et. al.*); however, it is probably obsolete with the more robust WGANs. Nonetheless, this preprocessing step was used for all the data through this thesis. The difference in the mass for the reconstructed jet in the full and the selected detector region can be seen in Figure 4.8.



(a) m_{FD}^{img} versus m_{CW}^{img}

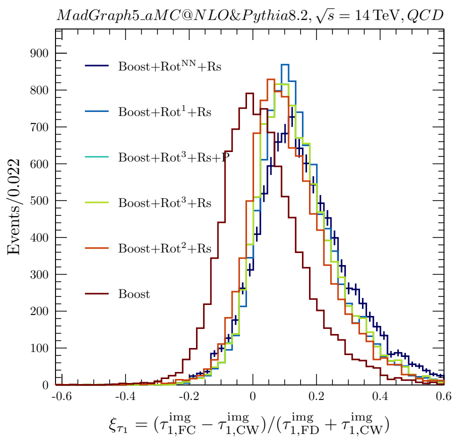


(b) $\Delta m = m_{FD}^{\mathrm{img}} - m_{CW}^{\mathrm{img}}$ versus $p_{\mathrm{T}}^{\mathrm{img}}$

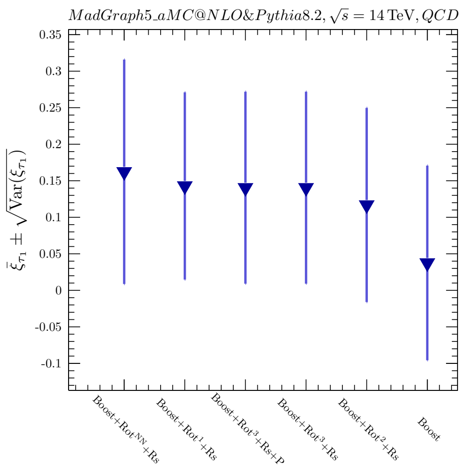
Plot 4.8: Comparison of the reconstructed jet mass in the Full Detector (FD) and the Cut Window (CW).

As it can be seen in Figure 4.8a, $m_{\mathrm{FD}}^{\mathrm{img}}$ and $m_{\mathrm{CW}}^{\mathrm{img}}$ are strongly positively correlated whereby $m_{\mathrm{FD}}^{\mathrm{img}} > m_{\mathrm{CW}}^{\mathrm{img}}$. This can be understood with the information given above. Since the leading subjet is centered, it might happen that other constituents of the reconstructed jet lay outside of the window after the transformation is applied. Therefore, the mass reconstructed from the image systematically underestimates the actual mass of the jet. The right Figure 4.8b on the other hand shows the difference of the reconstructed mass in the full detector and the selected region $\Delta m^{\mathrm{img}} = m_{\mathrm{FD}}^{\mathrm{img}} - m_{\mathrm{CW}}^{\mathrm{img}}$ versus the transverse momentum $p_{\mathrm{T}}^{\mathrm{img}}$ of the jet. In this case, a positive correlation can be observed too. This might be caused by the increasing amount of additional QCD radiation due to acceleration that is not contained within the window after the Lorentz transformation. Although, this effect is – of course – undesirable, it does not present a problem within the context of this feasibility study but, on the contrary, was important for convergence of GANs.

The effect is very similar for τ_1 (1-subjettiness for one assumed subjet) although less pronounced as illustrated in Figure 4.9.



(a) ξ_{τ_1} distribution



(b) Average "information loss"

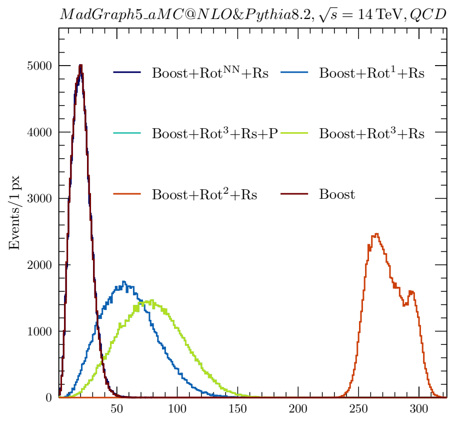
Plot 4.9: Loss of information for the reconstructed τ_1 for different preprocessing configurations.

Compared to the mass in Figure 4.7 the differences between the various interpolation techniques is rather 'marginal".

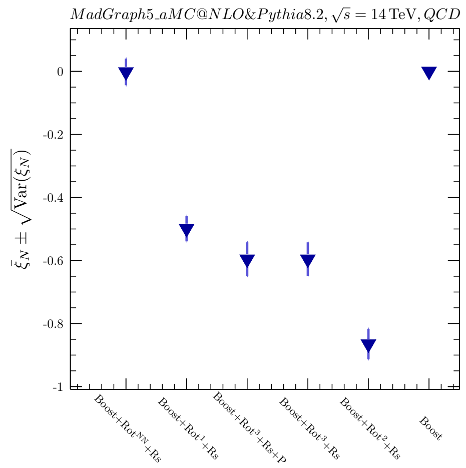
However, the situation is very different if one studies the information loss in the number of "constituents" or active pixels, i.e., the number of pixels N_0^{pix} in a jet-image with an energy value $E_{ij}^{\mathrm{pix}} > 0$. Since the effect is significant, Figure 4.10a directly shows the distributions of N_0^{pix} on an event-on-event base instead of $\xi_{N^{\mathrm{pix}}}$. Figure 4.10b, however, still gives the average and RMS of the $\xi_{N^{\mathrm{pix}}}$ distributions since it is centered around zero and hence easier to interpret. As was to be expected, the impact on the number of active pixels in an image is considerable. This effect is caused by the rotation of the image and therefore is strongly dependent on the respective interpolation method that is used. As it can be seen in Figure 4.10a, the distribution of active pixels for the (1-)nearest neighbour interpolation coincides with the distribution from the boosted data set that did not undergo any rotation and interpolation. This is line with expectations since the (1)-nearest neighbour interpolation is simply defined by

$$E(i,j) = \begin{cases} E(\lfloor i \rfloor, \lfloor j \rfloor) & \text{for } i/j - \lfloor i/j \rfloor < \frac{1}{2} \\ E(\lfloor i \rfloor + 1, \lfloor j \rfloor + 1) & \text{otherwise} \end{cases},$$

with $(i,j) \in \mathbb{N}^{n_\eta^{\mathrm{pix}\prime}} \times \mathbb{N}^{n_\phi^{\mathrm{pix}\prime}}$. So, the number of active pixels remains unchanged, however, not their position in the image. The other interpolation techniques in Figure 4.10b use several pixels for interpolation; therefore, the number of active pixels is significantly increases. So, the rotation as well as the following interpolation (except nearest neighbor) result in a distorted number of active pixels. This, however, does not mean that the nearest neighbor method is the appropriate choice to interpolate between pixels. As it can be seen in Figure 4.7 and 4.9, the nearest neighbor interpolation results in a significant information loss in the mass as well as τ_1 where the relative position between pixels is crucial. This information, the distance between pixels in the η-ϕ grid, is not well preserved (anyway, the number of active pixels is not a good observable since it is not IRC safe (see Section 1.3.2)).



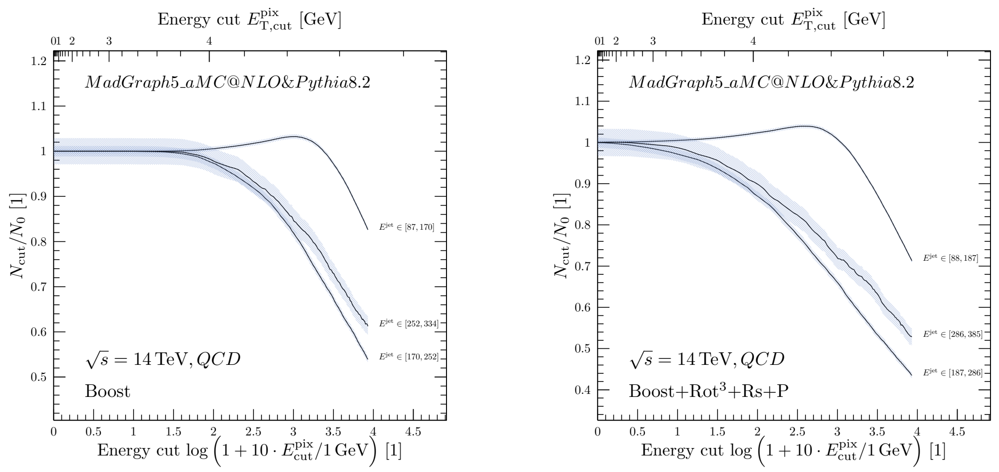
(a) Number of pixels with $E^{\text{pix}} > 0$



(b) Average "information loss"

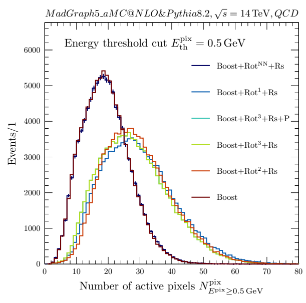
Plot 4.10: Loss of information for the reconstructed number of active pixels N_0^{pix} for different preprocessing configurations.

Taking into account many other distributions (which are not shown), it was decided that the *cubic spline interpolation* is the method of choice even though it results in a significant increase in the number of active pixels in an image. The increased occupancy in the image due to the interpolation is actual beneficial from the point of view of the training stability of neural networks (see stabilizing GANs through noise Section 3.6.1): as evident from Figure 4.10a, as well as visually observed in the generated samples in Figure 4.3, the training data is extremely sparse. For non-rotated QCD events/images, the average occupancy is roughly 3 % while the rest stay inactive. It is well known that machine learning algorithms (especially generative models) usually show poor performance if applied to sparse data. The reason for this follows the fact that for sparse data a significant amount of weights stay inactive and hence are not updated (sparse gradients). This corresponds to an effective reduction of the size of the training set. If the training set includes $N^{|\mathcal{D}|}$ training samples $\{\boldsymbol{x}_k\}_{k=1}^{N^{|\mathcal{D}|}}$ with an average occupancy $f_{\text{eff}} = \frac{1}{N^{|\mathcal{D}|}\cdot n_{\boldsymbol{x}}} \sum_{i,k} \Theta(x_k^{(i)}) \in [0.1]$ – counting non-zero entries over the entire data set –, the "effective size" of the data set $N_{\text{eff}}^{|\mathcal{D}|}$ is actually just $N_{\text{eff}}^{|\mathcal{D}|} \sim \mathcal{O}(f_{\text{eff}} N^{|\mathcal{D}|})$. This means, for instance, that the effective size of the boosted data set with 200, 000 events effectively is reduced to only 6, 000 events compared to a data set with full occupancy in each image. This effect is confirmed by several experiments and has often been observed. Normally, a counteractive measure to reduce the impact of the aforementioned effect is to convolute the data with a known distribution (usually taken to be a Gaussian), i.e., adding noise to increase the occupancy and to expand the intersecting set of the support. This is a common regularization scheme to improve the performance of generative models. Likewise, the rotation and interpolation acts as a regularization since it increases the number of active pixels in the image. This turned out to be very important for the training of *classical GANs*. In case of the cubic spline interpolation, the average occupancy significantly increases to roughly 13 % – which is still very sparse though. A possible countermeasure would be to reduce the number of pixels $n_{\eta(\phi)}^{\text{pix}\prime}$ with the cost of less structure in the image. Furthermore, an adjustment of the number of active pixels in the image can be done by applying an energy threshold cut $E_{\text{th}}^{\text{pix}}$ on each pixel. Figure 4.11 shows the fraction of events in one out of three bins for the reconstructed jet energy parameterized by the energy threshold $E_{\text{th}}^{\text{pix}}$ applied on each pixel (migration of events from one bin to another).

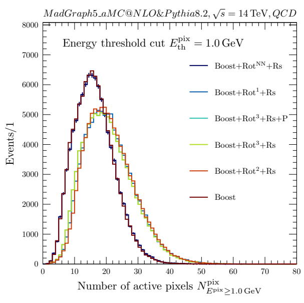


Plot 4.11: Bin-to-bin-migration as a function of the energy threshold $E_{\text{th}}^{\text{pix}}$.

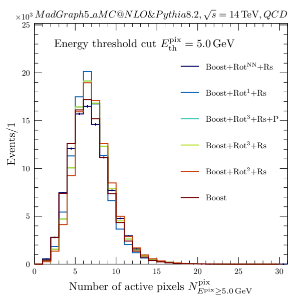
The direct effect of the energy cut on the distribution of active pixels for the different interpolation methods is exemplary illustrated in Figure 4.12 for $E_{\text{th}}^{\text{pix}} \in \{0.5\,\text{GeV}, 1.0\,\text{GeV}, 5.0\,\text{GeV}\}$.



(a) $E_{\text{th}}^{\text{pix}} = 0.5\,\text{GeV}$



(b) $E_{\text{th}}^{\text{pix}} = 1.0\,\text{GeV}$



(c) $E_{\text{th}}^{\text{pix}} = 5.0\,\text{GeV}$

Plot 4.12: Number of active pixels $N_{E^{\text{pix}} > E_{\text{th}}^{\text{pix}}}^{\text{pix}}$ for different threshold cuts.

There is already a significant change in the shape of the distributions for an energy-cut of only $E_{\text{th}}^{\text{pix}} = 0.5\,\text{GeV}$ (the distribution for the boosted data set remains unchanged due to the absence of any interpolation and the 500 MeV cut in the "detector simulation" at parton level (see Section 4.1.1)). For a threshold cut of $E_{\text{th}}^{\text{pix}} = 5.0\,\text{GeV}$ all distributions are (more or less) in good agreement.

Part III

Chapter 5

Jets with Gaussian Variational Autoencoders

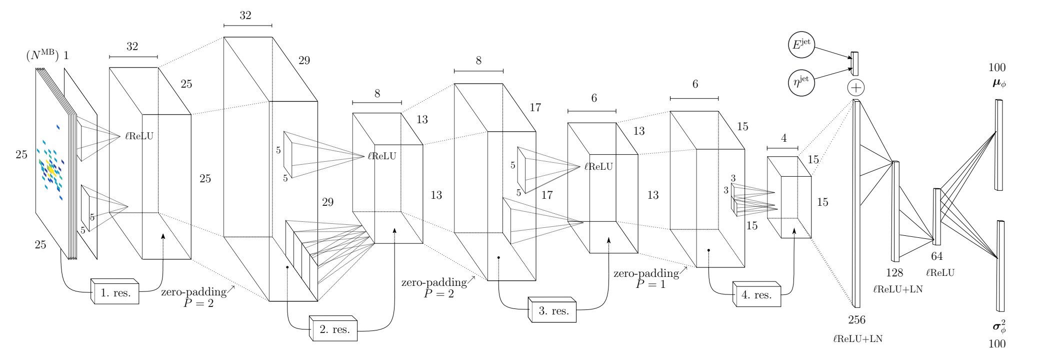
The last chapter provided an introduction into the topic of machine learning, motivated and compiled the data, which consists of QCD and W initialized jet and introduced the individual preprocessing steps that are applied to the jet images to remove redundancies present in the actual data set to improve the performance of the training routine.

With a standardized and preprocessed data set at one's disposal, it is time to turn the attention to the actual subject matter of this thesis, i.e., the analysis of generative models through machine learning based on deep learning. This opens the third and final part conducted within the framework of this report that studies Gaussian VAEs as well as Wasserstein GANs, following the chronological order introduced in Chapter 3.

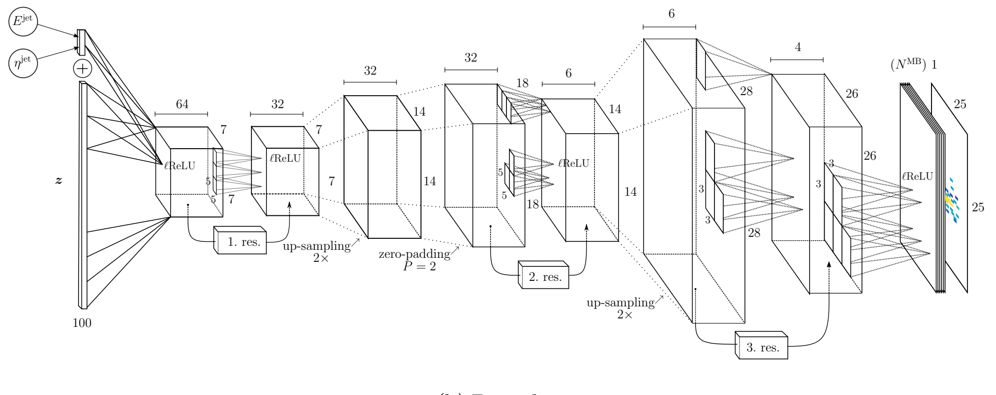
This chapter starts with a brief description of the architecture that was used for the neural networks of the *encoder* and the *decoder* model (5.1) (the architecture of the critic and the generator model in case of adversarial networks, which are the subject of the subsequent Chapter 6, will remain mostly unaltered compared to the ones introduced in the context of variational autoencoders). Section 5.2 introduces and justifies the configuration of hyperparameters, i.e., the dimension of the latent space $\dim(\mathscr{Z})$, the learning rate α_l, and the gradient-descent-based optimization algorithm that have been used to train the composite model. The successive section (5.3) is the first attempt to generative models and studies unconditioned Gaussian variational autoencoders. In Section 5.4, the information of the matrix element is marginalized through a Gaussian VAE that is conditioned on the energy E^{jet} and the pseudorapidity η^{jet} of the jet. The penultimate section of this chapter (6.5) combines conditional Gaussian VAEs with RNNs with a view to model the underlying sequential nature of the splitting sequence in the parton shower. This chapter finally ends with a very brief summary and conclusion based on the gathered insights before the attention is directed to Wasserstein GANs in Chapter 6.

5.1 Model architecture specification

An important part of the necessary preperation when dealing with machine learning methods is, in general, to develop a particular design for the architectures of the neural networks used to implement the respective models, e.g., the non-linear activations, the number of hidden layers and weights used in a feed-forward topology (see Section 3.3.1) or, for instance, the number of time-steps n_T in a recurrent network. At present, there are no *general* rules available regarding the choice and the design of a network architecture for a specific purpose (even though there are attempts towards a fully automatic construction of deep neural networks); therefore, the optimization of the *unable* hyperparameters of the respective model mostly relies on empirical insights, rules of thumb that have been acquired over time and is therefore often one recurring point of criticism. Within the scope of this thesis, the design of the network architecture does by no means aim for the best possible performance of the respective deep learning models, but to reach an acceptable compromise between performance and complexity – fully aware that different architectures might provide occasionally better results.



(a) Encoder



(b) Decoder

Fig. 5.1: Network architecture of the variational autoencoder.

All architectures that are used through this thesis – also in case of the Wasserstein GANs (see Section 6.1) – are based on the famous and pioneering design of the *Deep Convolutional Generative Adversarial Networks* (DCGAN) introduced by Radford *et al.* [2015] as well as the modified architecture proposed by de Oliveira *et al.* [2017]. The corresponding geometry of the decoder (Figure 5.1a) and encoder (Figure 5.1b) neural network used in this report is schematically illustrated in Figure 5.1[1]. The two basic architecture geometries that are shown in Figure 5.1 come in different variations that either utilize fully-connected dense layers (`FcNet`), classical convolutional layers (`ConvNet`) [Springenberg *et al.*, 2015], locally-connected layers (`LcNet`) or locally-connected layers with *residual blocks* (`ResNet`). Furthermore, the suffix "`-BN`" resp. "`-LN`" implies that either batch or layer normalization (see Section 3.4.4) was used (e.g. `ResNetBN/LN`), while the suffix "`-Res`" indicates that the number of trainable parameters have been reduced (usually approximately by a factor of one-half $N^{\mathrm{X}} : N^{\mathrm{XRed}} \approx 2 : 1$) for the specific architecture under consideration (for instance, the network `LcNet` comprises significantly more trainable parameters than `LcNetRed`).

The aforementioned concept of *residual learning* was first introduced by He *et al.* [2015] in Google's "Deep Residual Learning for Image Recognition" and has become increasingly popular among the machine learning community over the recent years. Today, many state-of-the-art models use residual networks that are, in turn, based on the aforementioned *residual blocks*. Conceptually, this method is very intuitive and simple to understand yet very effective. The basic operating principle of a residual block in its simplest form – as it is used in this report – is summarized in Figure 5.2 below.

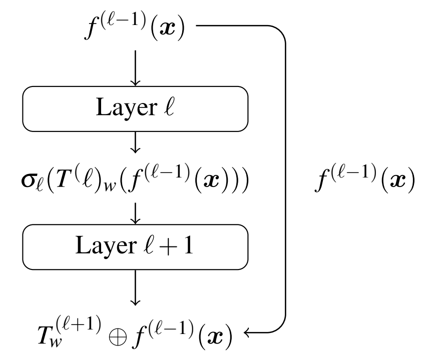

$f^{(\ell-1)}(\boldsymbol{x})$
Layer ℓ
$\boldsymbol{\sigma}_\ell(T^(\ell)_w(f^{(\ell-1)}(\boldsymbol{x})))$
Layer $\ell+1$
$f^{(\ell-1)}(\boldsymbol{x})$
$T_w^{(\ell+1)} \oplus f^{(\ell-1)}(\boldsymbol{x})$

Fig. 5.2: A simple residual block.

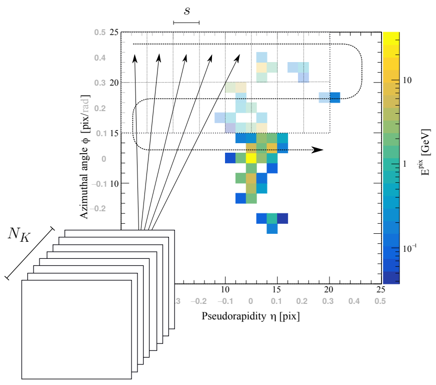
In conventional feed-forward neural networks (as they have been introduced in Section 3.3.1) the output of each (hidden) layer is *exclusively* used as an input for the *subsequent* one. Hence, the current state of layer ℓ does only depend on $\ell - 1$ and not *explicitly* on layer $\ell - 2$. For models with at least one residual block, however, the input into one layer contemporaneously possibly serves as a "non-transformed"[2] input into many other layers somewhere in the network (see Figure 5.2). This allows for the construction of very deep and complicated models without running into known characteristic problems of deep neural networks like vanishing or exploding gradients in the backpropagation algorithm due to the subsequent application of the chain-rule and/or the dreaded "curse of dimensionality" [Keogh and Mueen, 2017]. If the complexity of the network is gradually increased, the "goodness of the fit", i.e., the loss/cost function will begin to saturate and possibly start to degrade again if more and more parameters are added to the model (*cf.* degradation

[1]Important note: The geometrical dimensions of the shapes in Figure 5.1 are *not* drawn to scale! They only intend to provide an intuitive display of the architectures used in this work.

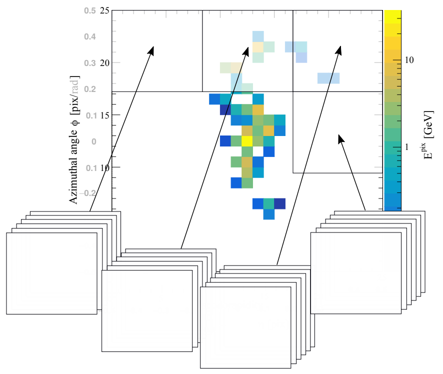
[2]Occasionally, it is necessary to correct the "shape" of the residual such that the concatenation $T_w^{(\ell+1)} \oplus f^{(\ell-1)}$ is a meaningful operation.

problem in deep learning). With residual blocks being part of the model, however, sections of the neural network may be bypassed (see Figure 5.2) or even completely disconnected; hence, the effective number of parameters is reduced. This makes the model significantly less vulnerable and more robust against a suboptimal design of the network's architecture, which is often the root of poor performance.

Besides the residual blocks, the rather unconventional locally-connected convolutional layers (with unshared parameters) need some explanation. Convolutional neural networks have been developed simultaneously with the increasing relevance of image processing and image analysis in machine learning. The need for *shared weights* – as commonly used by convolutional networks – is immediately apparent since fully-connected layers do not scale well with the increasing dimensionality of the input, which grows quadratically with the size of the image. A simple fully-connected network that consists of only one shallow layer with N_n nodes/neuron that takes as input an image with dimension $N_x \times N_y \times N_c$, with N_c being the number of color channels in the image (e.g. $N_c = 1$ for 8-bit (monochrome) grayscale or $N_c = 3$ for RGB of the 24-bit standard), will already have $N_x \times N_y \times N_c \times N_n$ learnable weights (excluding biases). Usually, the dimensions N_x and N_y of the image are the same; hence, the number of weights grows quadratic, which makes them again prone to overfitting. Convolutional neural networks solve this problem by using the same set of weights (*cf.* shared-weight filters) for the convolution with the data and therefore also accompanied by some regularization effect that reduces overfitting (see Figure 5.3a). Besides the advantage of a considerably reduced number of parameters, convolutional layers are translational invariant [Lecun, 1989], i.e., a shift in the input data will cause a proportional shift in the output of the operation. This property is especially desirable for data with a high degree of symmetry.



(a) Shared weights in a convolutional layer with N_K filters (kernels) of size $K \times K$ convoluted with an image of size $L \times L$ with a stride (step) size of S. If one includes zero-padding P, the output size of the convolution is given by $W \times W \times N_K$ with $W = \frac{L-K-2P}{S} + 1$.



(b) Operating principle of a locally-connected layer. A *locally*-connected layer has groups N *unique* filters that are applied to each section of the image individually; hence, each stack of filters learns a segment of the image, whereby the filters may partially overlap.

Fig. 5.3: Convolutional and fully-connected layers.

The operating principle of a locally-connected layer, on the other hand, differs significantly from convolutional layers. As it is shown in Figure 5.3b, locally-connected layers do not use the same set of kernels for each region of the image but a group or family of filters associated with a local region of the matrix that is fixed; hence, the weights are not shared. Usually, convolutional layers are favoured over their locally-connected counterpart due to the considerable smaller number of parameters in the model and the aforementioned translational invariance – which is not present in case of locally-connected layers. However, it is also known that convolutional layers often perform poorly when applied to very sparse data. Furthermore, there are complications if the data has regions of very small variance. As it has been demonstrated by de Oliveira *et al.* [2017], locally-connected layers are indeed beneficial and show better performance if applied to data with low occupancy such as the jet images that are sparse as shown in Chapter 4.4.2.

Following now is a detailed description of the architecture of the decoder and the encoder model.

The *encoder* network in Figure 5.1a, which tries to learn a new representation of the data, receives as an input a batch of (jet-)images with dimension $25 \times 25 \times 1$ and two labels, $p_{\mathrm{T}}^{\mathrm{jet}}$ and η^{jet}, if the model is additionally conditioned. The first layers of the network only process the image information (no conditioning labels) with a conventional convolutional layer (see Figure 5.3a) with 32 filters/kernels of size 5×5 and a stride size $s_x = x_y = 1$. The *stride* size is the distance between spatial locations of the image where the convolutional kernel is applied (see Figure 5.3). Proximately, the output of the convolution passes a `ReLU` activation function (see Figure 3.1) and is expanded by "appending" two rows and columns of zeros (zero-padding), which serves as a preperatory step for the subsequent operation (as it has been proposed in the "Architecture guidelines for stable Deep Convolutional GANs" by Radford *et al.* [2015], the first layer remains always non-normalized). The zero-padded image is then further processed by the first locally-connected layer (unshared weights) with a group size of 8 filters with a spatial extension 5×5 and a stride size of $s_x = s_y = 2$. The output is again handled by a `ReLU` activation function and possibly normalized employing layer or batch normalization (see Section 3.4.4) indicated by the respective suffix. The procedure is repeated exactly from the point of zero-padding but with "only" 6 filters in the following locally-connected layer and a stride size of $s_x = s_y = 1$. The subsequent block is the last locally-connected layer in the network with 4 filters of size 3×3 and $s_x = s_y = 2$. It receives the zero-padded image with pad size of $P_x = P_y = 1$ and transfers the output to a `ReLU` (no normalization layer). The image is then flattened ($\mathbb{R}^{N_1} \times \mathbb{R}^{N_2} \times \mathbb{R}^{N_3} \to \mathbb{R}^{N_1 \times N_2 \times N_3}$), and the resulting vector is concatenated (if conditioned) with the conditioning value for the energy $E^{\mathrm{jet}} \in \mathbb{R}$ and the pseudorapidity $\eta^{\mathrm{jet}} \in \mathbb{R}$ of the jet. From now on, the following layers are three fully-connected *dense* layers with 256, 128 and 64 neurons (the GPU can utilize optimizations when working with powers of two) and with `ReLU` activation functions and layer normalization in between[3]. In case of variational autoencoders, there are two additional dense layers for the mean $\boldsymbol{\mu}_\phi$ and the variance $\boldsymbol{\sigma}_\phi^2$ of the latent space representation of the data with an output dimension of $N_z := \dim(\mathcal{Z})$ (the dimensionality of the output in case of the critic/discriminator for Wasserstein GANs is just 1). Not yet mentioned in the description of the architecture above are the residual layers that are displayed by the paths that connect certain layers in Figure 5.1 (*cf.* Figure 5.2). Those residual blocks are only active if the network is a residual network as explicitly indicated by the "`-Res`" suffix otherwise they are discarded.

The architecture of the *decoder* network in Figure 5.1b is quite similar to have a sym-

[3]The combination of convolutional and dense layers is a standard architecture in deep learning. This is justified by the observation that the convolutional kernels learn location-associated structures of the image while the dense layers connect respectively correlate those learned structures.

metric system. However, it receives as an input a latent-space vector $\boldsymbol{z}$ of random variables of size N_z sampled from a standard multivariate normal distribution $\boldsymbol{z} \sim \mathcal{N}(0, \mathbb{1})$ that is simply concatenated[4] with the conditioning labels $p_{\mathrm{T}}^{\mathrm{jet}}$ and η^{jet} to $\boldsymbol{z} \oplus p_{\mathrm{T}}^{\mathrm{jet}} \oplus \eta^{\mathrm{jet}} \in \mathbb{R}^{N_z+1+1}$ (if conditioned). To further process the data via convolutional and/or fully-connected layers, the data must be reshaped by a dense layer to $\mathbb{R}^{N_z+1+1} \to \mathbb{R}^{64} \times \mathbb{R}^{7} \times \mathbb{R}^{7}$. This operation gets more expensive if the dimensionality of the latent-space vector is increased (the number of trainable parameters is given by $[N_z + 2] \cdot 64 \cdot 7 \cdot 7$). So, after this operation, the result is an "image" with dimension $64 \times 7 \times 7$. This image serves as input for the subsequent convolutional operation with again 32 and a kernel size 5×5 filters with a stride size of $s_x = s_y = 1$ followed by a `ReLU` activation and normalization. The next processing step is an up-sampling of the image by a factor of two. This layer does not come with any trainable parameters but only scales the image such that $W \to 2W$. This is then once again subsequently followed by zero-padding with $P = 2$. Those steps are necessary to ensure that the subsequent operations on the intermediate image will finally result in an output image with size $25 \times 25 \times 1$ (compare dimension of the encoder's input). The resulting image is then handled by a locally-connected layer with 6 filter per group of size 5×5 and $s_x = s_y = 1$. After the activation function and the normalization layer, the same block is repeated but without zero-padding and with only 4 kernels for the locally-connected layers with size 3×3. The last layer is again a locally-connected layer with only one filter per group (since the output image only has one "channel" that corresponds to the continuous energy associated with the pixels E^{pix}) and a kernel size of 2×2, as well as a disabled bias term. Finally, the output is passed through a very last `ReLU` function, which is quite uncommon, since they are likely to cause *sparse gradients*. However, the activation function was chosen to avoid saturation effects for large ranges of energy values that span over several orders of magnitude (*cf.* activation functions in Plot 3.1). The bias $\boldsymbol{b}$ term in the rectified linear unit $\texttt{ReLU}(\boldsymbol{x}) = \theta(\boldsymbol{x} + \boldsymbol{b})$ has been disabled from the start to prevent the network from learning non-physical negative energy values in the pixels.

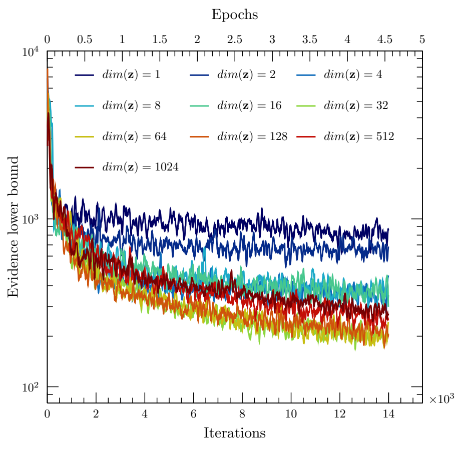
The architecture of the encoder and the decoder network described above are purposely designed to be very similar such that their total number of trainable parameters is in the same order of magnitude. This is intended since both networks should have more or less the same complexity such that both networks are able to compete with one another. This is especially important in case of classical GANs as introduced in Section 3.6.1 to keep the discriminator from becoming too powerful; regarding Wasserstein GANs, however, this restriction does not exist.

5.2 Latent space and hyperparameter configuration

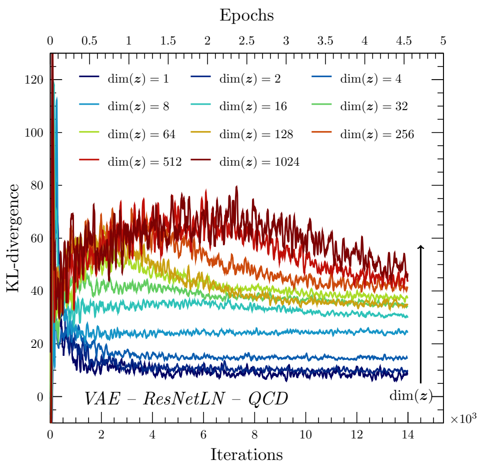
The latent space is one of the crucial components likewise in a variational autoencoder as in a classical autoencoder. It is the space of the compressed representation of the training

[4]This is the most simple way to incorporate additional information into the neural network. However, it should be noted that there is an ongoing debate on how to integrate conditioning labels into the architecture of a neural network. The approaches range from adding the additional information only at the beginning of the network, to only in the last layer(s), to all layers (see, e.g., Miyato and Koyama [2018]). In this thesis, the conditioning labels are simply concatenated with the input to the decoder and with the input to the dense layers in the encoder. However, there is a small modification: the position of the labels in the respective vector is not fixed but randomly chosen according to a discrete uniform probability distribution, with the probability of a certain position $k \leq N_z + 2$ given by $p(X = k) = \frac{1}{N_z+2}$. This is a regularization technique that ensures that the network does not simply disconnect the additional information by setting certain weights to zero.

data and simultaneous serves as the "source space" for the seeds $\boldsymbol{z}$ from which new, i.e., unseen samples are generated by the generative model $\hat{\boldsymbol{x}} = f_\theta(\boldsymbol{z})$. If the latent space $\mathcal{Z}$ is inappropriately chosen (regarding the shape of the underlying distribution and/or its number of dimensions), the encoder network f_ϕ is likely to fail to learn an appropriate map from the data manifold $\mathcal{X}$ to $\mathcal{Z}$. Consequently, the encoder (the generative model) will necessarily fail to reconstruct the original data from the hidden space $f_\theta \circ f_\phi \not\approx \text{id}$, as well as to generate authentic samples that are not elements of the training set $f_\theta(\boldsymbol{z}) \notin \mathcal{X}$. Usually, the dimension on the latent space is much smaller than the dimension of the training data $\dim(\mathcal{Z}) \ll \dim(\mathcal{X})$ which gives rise to a narrow "bottleneck" that may result in data loss, i.e., the encoder is not able to embed all characteristic features of the data in the hidden space. Therefore, for the encoder to learn a decent hidden representation of the data, the dimensionality of the latent space must be sufficiently large to avoid data loss and hence mismodelling of the underlying distribution. However, there is no general law that relates the size of the latent space and, for instance, the dimensionality of the input data or the number of parameters in the networks. Therefore, it must be determined in several experiments such that there is an acceptable trade-off between performance and complexity, i.e., the number of parameters and the quality of the ELBO according to Equation 3.41. Figure 5.2 shows the ELBO as well as the KL-divergence $\text{KL}(\mathbb{Q}_\phi||\mathbb{P}_z)$ that measures the similarity between the distribution learned by the inference network $\mathbb{Q}_\phi = f_\phi(\mathbb{P}_r)$ and the Gaussian prior $\mathbb{P}_z$ parameterized for different dimensions of the latent space $\dim(\mathcal{Z}) \in \{2^k\}_{k\leq 10}$.



(a) ELBO

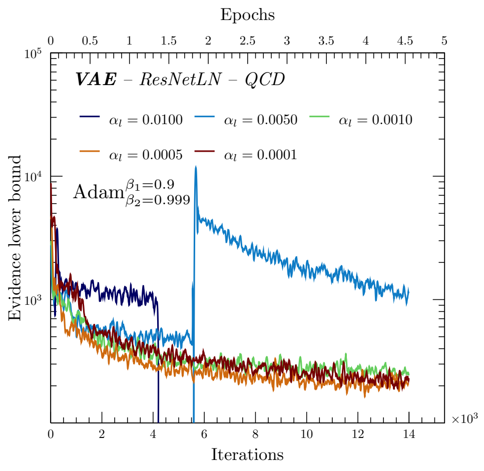


(b) KL-divergence

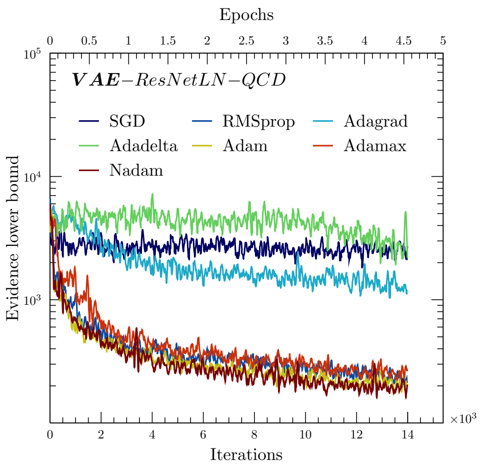
Plot 5.1: The ELBO and the KL-divergence (see Equation 3.41) versus the number of training iterations for different dimensions of the latent space $\dim(\mathcal{Z})$.

The strong dependence of the ELBO and the KL-divergence on the dimensionality of the latent space is evident from the two plots above. In Figure 5.1a, the ELBO decreases up to roughly $\dim(\mathcal{Z}) \leq 128 \sim \mathcal{O}(\dim(\mathcal{X}))$. Subsequently, the reconstruction loss and the KL-divergence gradually deteriorates. In principle, an increased number of dimensions for the latent space allows to capture more features of the training data. However, the encoder (and/or decoder) network might not be able to learn a well-suited function that achieves a transformation from the data to the latent space or *vice versa*. So, to make use of the more powerful latent space, the complexity of the encoder and/or the decoder network must be increased accordingly by adding more parameters to the model(s). But since the computer resources available in the scope of this thesis are rather limited, the architecture and complexity of the encoder and the decoder remain unchanged (see Section 5.1). Hence, the dimension of the latent space was chosen to be $\dim(\mathcal{Z}) = 100$ (also concerning the comparability with the latent space in case of Wasserstein GANs). The curve progression of the KL-divergence in Figure 5.1b may come as a surprise: it grows with the increased complexity of the latent space. However, this is exactly the desired curve characteristics. The family of curves indicates that the entropy $H(\boldsymbol{Z}) = \mathbb{E}_{\boldsymbol{Z}\sim\mathbb{Q}_\phi}[-\log(P(\boldsymbol{Z}))]$ of the distribution grows and hence encodes more intrinsic features of the data into the latent space for an increased number of dimensions.

Another important hyperparameter is the learning rate α_l that, along with the respective optimizer, determines the weight of the correction to the model's parameters with respect to the gradients of the loss function after each iteration step (see Section 3.4.2). Tuning this parameter is painful – especially for generative models – since the model is very sensitive to an inopportune configuration of α_l. A learning rate that is chosen too small may quickly result in extremely long training times and a complete standstill, while a learning rate that is too large may easily cause non-convergence. This is illustrated in Figure 5.2 that shows the ELBO for different learning rates $\alpha_l (:= \alpha_l^e = \alpha_l^d)$, whereby both networks, the encoder and the decoder, experience the same learning rate.



(a) ELBO versus learning rate



(b) ELBO versus optimizer

Plot 5.2: The ELBO versus the number of training iterations for different learning rates α_l (5.2a) and various optimization algorithms (5.2b).

As it can be seen in Figure 5.2a, a learning rate with $\alpha_l = 0.01$ or $\alpha_l = 0.005$ is to large; the models do not converge. Especially the (blue) curve corresponding to $\alpha_l = 0.05$ nicely illustrates how the configuration of trainable parameters (which span a very high-dimensional space with $\dim(\mathcal{W}) \approx \mathcal{O}(10^6)$) gets "kicked out" of a local minimum on the optimization hyper-surface $\mathcal{L}^{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ if the learning rate is chosen too large (see Figure 3.4.2 for a conceptional portrayal of the situation). Based on Figure 5.2a, a learning rate of $\alpha_l = 0.0002$ in combination with Adam (see Figure 5.2b) was deduced, which was used for training the encoder as well as the decoder network.

The rightmost Plot in Figure 5.2 shows a family of curves whereby each graph corresponds to a Gaussian variational autoencoder that has been trained with a different, commonly used first-order optimization algorithm (see Section 3.4.2). All additional parameters besides the learning rate α_l are kept at their respective default value (default values given by Chollet [2015]). For the (initial) learning rate, the previously determined value $\alpha_l = 0.0002$ according to Figure 5.2a was used for the different optimization schemes except for the "primitive" stochastic gradient descent optimizer, which required $\alpha_l \leq 0.00005$ in order to "converge" due to a missing damping mechanism of the learning rate. Based on the curve progression of the ELBO for different optimization algorithms in Figure 5.2b, it was decided to use Adam [Kingma and Ba, 2014] in the further course of this study. Note: Adam is only used in combination with Gaussian VAEs in this chapter. As it turned out, Adam is not at all an appropriate choice in case of Wasserstein GANs due to gradient penalty (see Figure 6.3a in Chapter 6).

5.3 Unconditioned VAEs

The data that is used to train the neural network in this thesis (see Chapter 4) basically encodes two contributions: the matrix element information for the hard subprocess and the parton shower simulation for soft and collinear QCD radiation. If the generative model is unconditioned and exclusively trained on jet-images (the energy E^{jet} and the pseudorapidity η^{jet} are withheld), it is intended to learn the underlying distributions of the matrix-element *and* the parton shower in the same model. In probabilistic terms, the neural network models the joint probability distribution $p_\theta(\boldsymbol{x}, E^{\text{jet}}, \eta^{\text{jet}})$. This scenario is obviously more complicated for the respective model compared to the situation where the network is conditioned because the entropy of the target distribution is much larger. However, despite being more complicated, the study of variational autoencoders in this section starts with the case where the model is unconditioned.

An unconditioned model is quite easy to implement and does not require any material changes in the architecture introduced in Section 5.1. All one has to do is to replace the conditioning labels $x_{c_1} := p_{\text{T}}^{\text{jet}}$ and $x_{c_2} := \eta^{\text{jet}}$ by $x_{c_1} = x_{c_2} = 0$ such that those particular connections in the directed graph are disabled

$$\sigma_j\left(\sum_{i=1}^{N} w_{ji}x_i\right) = \sigma_j\left(\sum_{i=1}^{N-2} w_{ji}x_i + w_{c_1}\underbrace{x_{c_1}}_{=0} + w_{c_2}\underbrace{x_{c_2}}_{=0}\right) = \sigma_j\left(\sum_{i=1}^{N-2} w_{ji}x_i\right).$$

For this purpose, the stochastic placement of the labels described in Footnote 4 in the random seed vector is abandoned and replaced by a fixed position for the time being.

The networks in this section have been trained on QCD (Section 4.1.1) and W (Section 4.1.1) samples for 200,000 data points in each case. A mini-batch of data $\{\boldsymbol{x}_i\}_{i=1}^{N_{\text{MB}}}$ contains – if not mentioned otherwise – $N_{\text{MB}} = 64$ jet images. This information is important to compare the ELBO of two networks that have been trained with different mini-batch sizes since the ELBO for mini-batch training is given by:

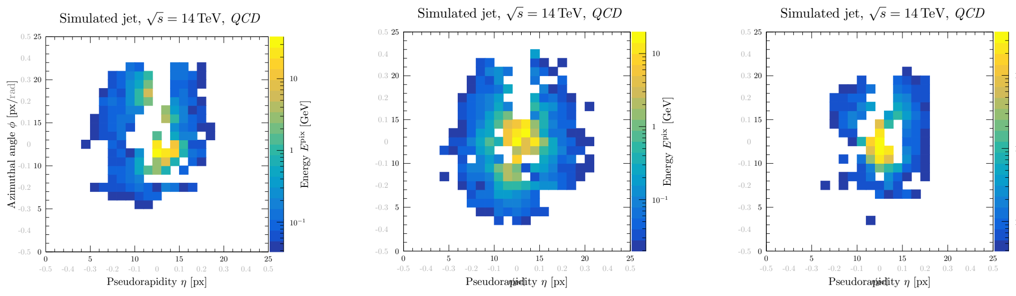
$$\mathcal{L}^{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \{\boldsymbol{x}_i\}_{i=1}^{N_{\text{MB}}}) \approx \frac{N}{N_{\text{MB}}} \sum_{i=1}^{N_{\text{MB}}} \mathcal{L}^{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_i). \tag{5.1}$$

According to Figure 5.2b, Adam was used as the stochastic gradient-based optimization of the loss function with a learning rate $\alpha_l = 0.0002$ (see Plot 5.2a and 5.2b) with the default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ (the default values have been taken from [Kingma and Ba, 2014, Fig. 4, p. 8]). The QCD and W samples are generated by two different generative models P_θ^{QCD}, P_θ^{W} in order to study both processes in detail and completely independently from each other. However, it only requires minor modifications to train one generative model on both processes at the same time by additionally conditioning the model on the process by a multi-class label (e.g. $0 \mapsto$ QCD, $\pm1 \mapsto W^\pm$, $2 \mapsto Z$ etc.), which should be the preferred solution.

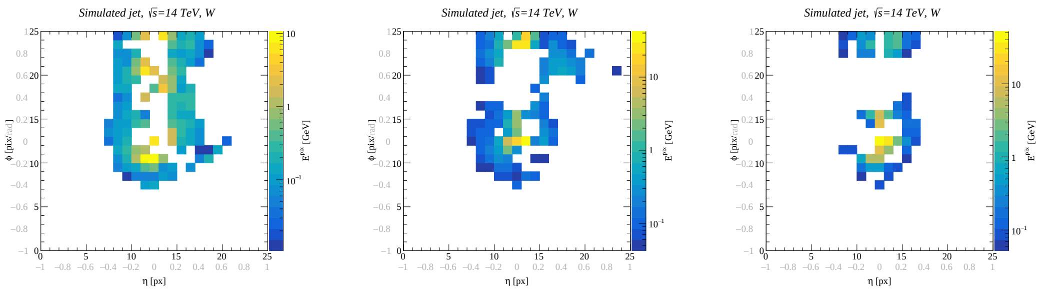
As it is demonstrated as part of this section, an unconditioned variational autoencoder does show quite a bad performance compared to its conditioned counterpart. (Already anticipating one result of the subsequent chapter: this statement does not apply to unconditioned Wasserstein GANs that, on the contrary, show an excellent performance.) Therefore, it is not the intention of this section to dwell upon this subject for an unnecessarily long time, but to highlight the characteristic problems and to move on to conditioned models.

5.3.1 Samples and average jet images

The performance of variational autoencoders is directly measured by the ELBO – contrary to generative adversarial networks based on an f-divergence, whose loss function is less conclusive. The regression loss provides information about the *quality* of the generated data while the KL-divergence measures the *information* (cross-entropy) of the latent space. Even though the ELBO provides a good measure to evaluate the performance of VAEs, the visual inspection of the generated data is indispensable to draw conclusions regarding the reliability of the simulated samples. This is the main purpose of this section, i.e., to provide an impression the extend to which the unconditioned variational autoencoder can produce jet-images that are visually indistinguishable from the training data.



Plot 5.3: Three randomly simulated QCD jets.

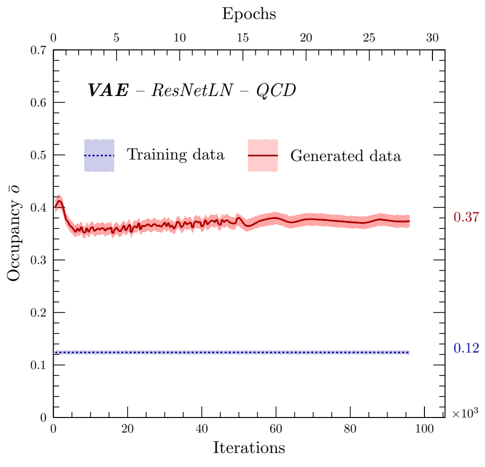


Plot 5.4: Three randomly simulated W jets.

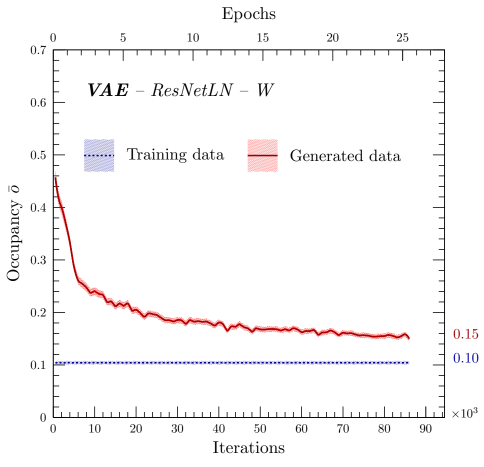
First, let's have a look at some individual generated images for QCD and W initialized jets generated by the Gaussian VAE. The set of images in Figure 5.3 and 5.4, which have been generated from random seeds $\boldsymbol{z} \in \mathcal{Z}$ already show some conspicuous features that are characteristic for variational autoencoders. If one compares the generated samples with the training data (*cf.* Figure 4.3 and 4.5), the aforementioned characteristic "blurring" or noising effect becomes clearly recognizable (see 3.6). This effect is particularly apparent when comparing the *occupancy* of the generated data and the training data. The mean occupancy $\bar{o}$ simply gives the average number of active pixels (number of pixels N^{pix} with $E^{\text{pix}} > E^{\text{pix}}_{\text{th}}$) for a total number of N^{event} images

$$\bar{o} = \frac{1}{N^{\text{event}}} \sum_{i=1}^{N^{\text{event}}} o_i = \frac{1}{N^{\text{event}}} \frac{1}{n_\eta^{\text{pix}} n_\phi^{\text{pix}}} \sum_{i=1}^{N^{\text{event}}} \sum_{j=1}^{n_\eta^{\text{pix}}} \sum_{k=1}^{n_\phi^{\text{pix}}} \Theta(E^{\text{pix}}_{jk,i} - E^{\text{pix}}_{\text{th}}), \tag{5.2}$$

whereby the Heaviside step function just counts the number of active pixels. Evaluating Equation 5.2 for the two processes under consideration results in quite different curve characteristics as it is illustrated in Figure 5.5.



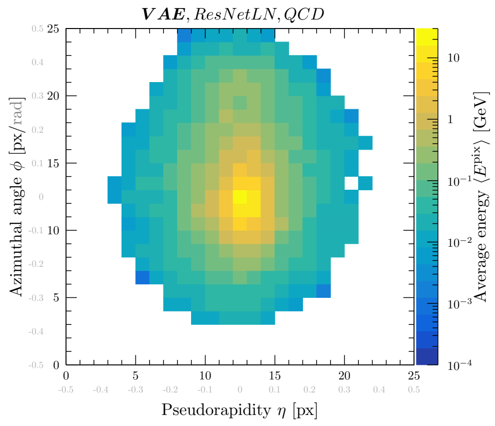
(a) QCD



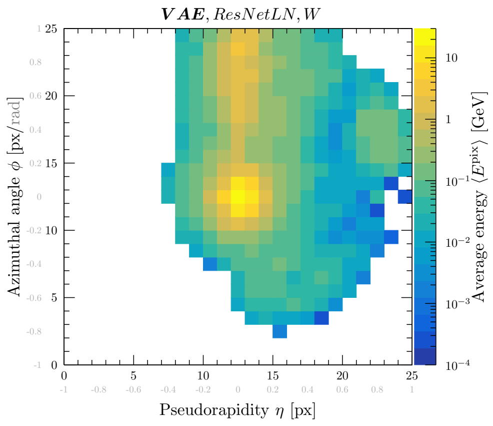
(b) W

Plot 5.5: Occupancy according to Equation 5.2.

In Figure 5.5, the "blurring" of the image clearly manifests itself by an significantly reduced mean sparseness $1 - \bar{o}$ or an increased occupancy $\bar{o}$ of the generated images. This effect is much more pronounced in case of the QCD initialized jets, although, the sparseness for QCD and W images is more or less the same ($\bar{o}_r^{\mathrm{QCD}} \approx 12\,\%$, $\bar{o}_r^{W} \approx 10\,\%$); hence, the distribution, or the average occupancy of active pixels in the image, does play a role as well. Therefore, this behavior can only be explained based on the very different jet substructure for the two different processes that can be seen in Figure 5.3 to some extend and 5.4, but becomes even more visible if one considers the average jet images shown below.



(a) QCD



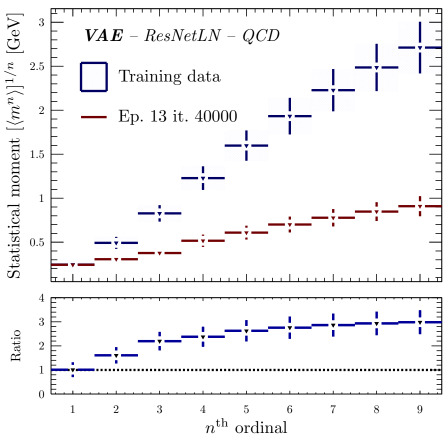
(b) W

Plot 5.6: Average jet image for 50,000 events and 40,000 iterations.

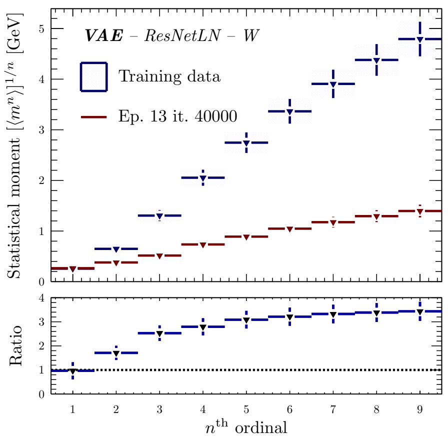
Figure 5.6 demonstrates that the Gaussian variational autoencoder rudimentarily learns the different characteristic features, i.e., one highly active center with surrounding soft radiation (which is blurred) for QCD jets and two well-separated regions of increased activity for W jets from the decay products (note: there is no p_{T}-cut involved in Figure 5.6b). For the latter one, the implications of the preprocessing (see Section 4.2) is quite apparent. The approximate correspondence between the generated average jet images in Figure 5.6a and 5.6b with the training data in Figure 4.2 (QCD) and 4.4 (W) demonstrates that the network correctly learns the *average* correlation between the different pixels. However, the agreement in the very first moment (the mean) is not sufficient to finally conclude that the VAE has learned the distribution of the underlying data since two distributions only completely match iff they agree in all statistical moments (*cf.*, e.g., maximum mean discrepancy). In case of the unconditioned VAE, however, significant deviations can already be observed in the first statistical moments. Figure 5.7 shows a comparison of the first nine (central) statistical moments of the training data and the data produced by the Gaussian variational autoencoder; thus, allows to gain a deeper insight into the distribution learned by the generative model. The n^{th} *statistical moment* $\boldsymbol{m}_{(n)} \in \mathbb{R}^M$ with $M := \dim(\mathcal{X})$ measures the "shape" of a distribution and is simply defined by

$$\boldsymbol{m}_{(n)} := \begin{cases} \int_{\mathbb{R}^M} \mathrm{d}\boldsymbol{x}\, p(\boldsymbol{x})\, (\boldsymbol{x} - \boldsymbol{\mu})^n \approx \frac{1}{N} \sum_{i=1}^N (\boldsymbol{x}_i - \boldsymbol{\mu})^n & \text{if } n > 1 \\ \int_{\mathbb{R}^M} \mathrm{d}\boldsymbol{x}\, p(\boldsymbol{x}) \boldsymbol{x} \approx \frac{1}{N} \sum_{i=1}^N \boldsymbol{x}_i & \text{if } n = 1 \end{cases}, \tag{5.3}$$

whereby the integrals are numerically evaluated by Monte Carlo integration (see Section 2.1.1). The definition of the statistical moment according to Equation 5.3 gives a moment for *each* pixel in the image, i.e., $\boldsymbol{m}_{(n)} \in \mathbb{R}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}}$. Comparing the moments for each pixel individually would be very tedious; therefore, the *nth moment per image* $m_{(n)}^{\text{img}} \in \mathbb{R}$ is defined by $m_{(n)}^{\text{img}} = \boldsymbol{m}_{(n)}^T \boldsymbol{m}_{(n)} / n_\eta^{\text{pix}} \cdot n_\phi^{\text{pix}}$. As it can clearly be seen in Figure 5.7, the distribution learned by the unconditioned variational autoencoder differs *significantly* from the training data.



(a) QCD

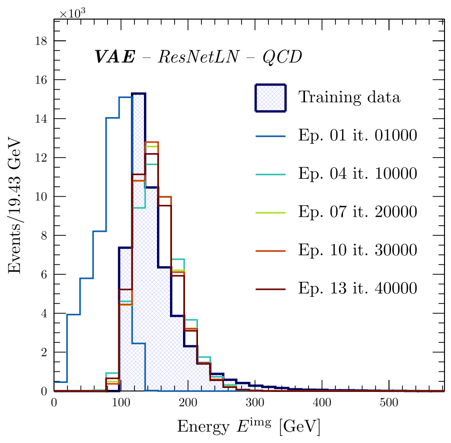


(b) W

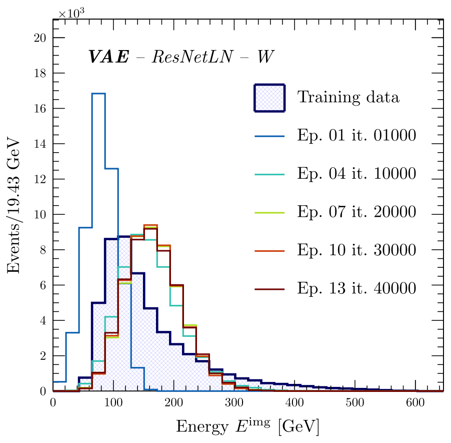
Plot 5.7: nth statistical moment for generated QCD and W jets.

A good agreement only exists for the very first moment – which can be visually confirmed by the comparison with the average images in Figure 5.6 –, i.e., the mean of the respective distribution; higher moments like the variance $\sigma^2 = m_{(2)}$, the *skewness* $m_{(3)}$, the *kurtosis* $m_{(4)}$ or higher moments without an apparent corresponding geometric or intuitive interpretation are not well described by the model. This means that for both processes, QCD (Figure 5.7a) and W (Figure 5.7b) jets, the generative model fails to properly learn the correlation between the individual pixels.

5.3.2 Kinematic distributions and jet observables



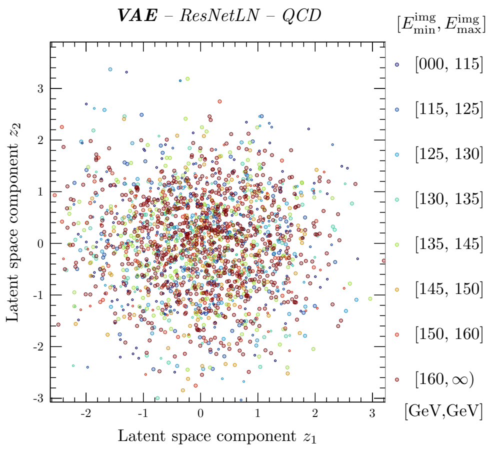
(a) QCD



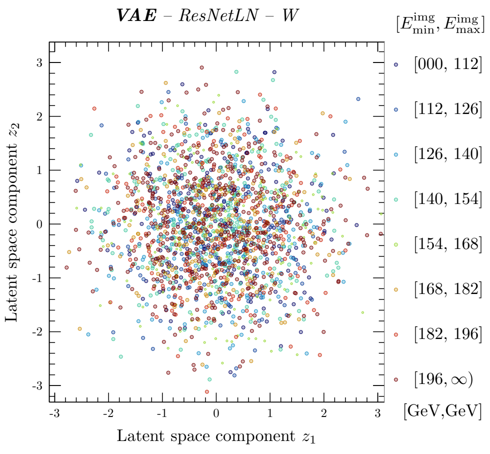
(b) W

Plot 5.8: Reconstructed energy E^{img} for $50,000$ events.

This section studies different kinematic distributions, such as the mass, energy etc., that have been reconstructed from distributionslearned by the generative model. Based on the previous section, it is to be expected that the network will suffer from poor performance. This can already be seen from the reconstructed energy spectrum. Since the model is unconditioned according to $p_\theta(\boldsymbol{x}, E^{\text{jet}}, \eta^{\text{jet}}|\boldsymbol{z})$, the energy distribution $p_\theta(E^{\text{jet}})$ is not factorized; hence, the reconstructed energy spectrum in Figure 5.8 is an indicator of to what extend the neural network has managed to learn the information of the underlying matrix element and the parton shower. So, by sampling a seed vector $\boldsymbol{z} \sim \mathbb{P}_z$ one also randomly samples a jet-energy E^{img} from the underlying energy distribution learned by the network that should correspond to the true distribution. This is only possible if the inference network has managed to encode the distribution of jet-energies E^{jet} into the latent space (similar to the discrete example for MNIST illustrated in Figure 3.4). One possible explanation for the severe deviations in Figure 5.8 is that the information of the energy is not correctly embedded in the hidden space, i.e., $P_\theta(E^{\text{img}}) \not\approx P(E^{\text{jet}})$. To study the generated space of the inference network $\mathbb{Q}_\phi$ in more detail, consider the distribution of Reconstructed jet-energies in Figure 5.9b for a two-dimensional subspace of the latent space.



(a) QCD

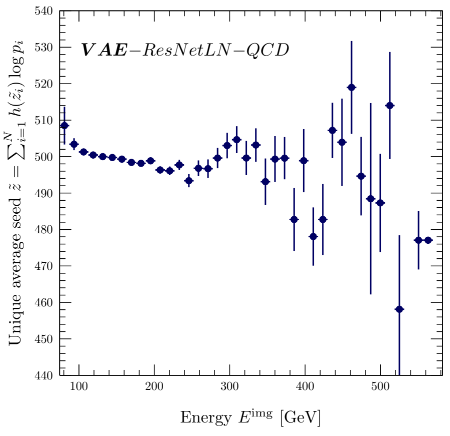


(b) W

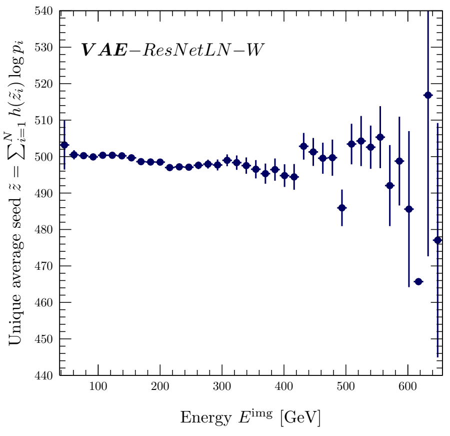
Plot 5.9: Energy distribution in a two-dimensional projection of the 100-dimensional latent space $\mathcal{Z}$.

In Figure 5.9, the range of individual energy intervals/classes are computed such that all bins contain the same number of events; otherwise the distribution would not be very conclusive. The plots are created by evaluating the generative model for a seed $\boldsymbol{z} \sim \mathbb{P}_z$ that gives a jet-image via $\hat{\boldsymbol{x}} = f_\theta(\boldsymbol{z})$. The reconstructed energy $E^{\text{img}} = E^{\text{img}}(\hat{\boldsymbol{x}})$ is then color-coded according to its respective class and drawn in a two dimensional subspace spanned by $z_i, z_j \in \boldsymbol{z} \in \mathcal{Z}$. Obviously, the model does not cluster the latent space to regions of different energy labels (as it was the case for the MNIST example in Chapter 3.4). It is important to remember that the distribution in Figure 5.9 only represents a two-dimensional projection of an actually 100-dimensional latent space; therefore, this plot cannot be used to conclude that there isnt a structure present in other correlations. There could be a correlation between the hidden space and the energy of the jet that is not visible in Figure 5.9. In order to uncover possible correlations, it is necessary to consider the entire latent space vector $\boldsymbol{z}$ and the associated energy E^{img} reconstructed from the image $\hat{\boldsymbol{x}} = f_\theta(\boldsymbol{z})$.

For a simple visualization that does not require complex illustration techniques for higher-dimensional data, it is necessary to construct a function that *uniquely* maps the high-dimensional latent space vector to a single number. But, such a bijective map does not exist since the entries $z_i \in \mathbb{R}$ of the seed vector are real numbers with $z_i \sim \mathcal{N}(0,1)$. However, a possible workaround is to multiply (scale) all entries of $\boldsymbol{z} \in \mathbb{R}^N$ with $N = \dim(\mathcal{Z})$ globally by a real number $\gamma \in \mathbb{R}^{\neq 0}$ and to round off the entries such that $\widetilde{z}_i := \lfloor \gamma z_i \rfloor \in \mathbb{Z}$. This means, of course, that the precision is reduced depending on the choice of γ. With $\widetilde{\boldsymbol{z}} \in \mathbb{Z}^N$, a function $q : \mathbb{Z}^N \to \mathbb{Q}$ that uniquely maps $\widetilde{\boldsymbol{z}}$ to some rational number can be constructed based on the unique-prime-factorization theorem. This function is defined by $q(\widetilde{\boldsymbol{z}}) = \prod_{i=1}^N p_i^{h(\widetilde{z}_i)}$, whereby p_i denotes the ith prime number and h is some affine function that shifts $\widetilde{z}$ by some constant to avoid problems if $\widetilde{z}_i = 0$ (for reasons of simplicity, $h(\widetilde{z}_i) = \widetilde{z}_i + 1$ is used if $\widetilde{z}_i \geq 0$, otherwise $h(\widetilde{z}_i) = \widetilde{z}_i$). The quantity $q(\widetilde{\boldsymbol{z}})$ is unique, but impractical regarding its numerical implementation. Therefore, for reasons of numerical stability, the monotonic logarithm is applied. As a result, the function that maps $\widetilde{\boldsymbol{z}} \in \mathbb{Z}^N$ to $\widetilde{z} \in \mathbb{Q}$ is given by $\widetilde{z} = \sum_{i=1}^N h(\widetilde{z}_i) \log p_i$. Now, that there is a $\widetilde{z}$ for each $\boldsymbol{z}$, the reconstructed energy of a jet generated from $\boldsymbol{z}$ can be plotted against the average $\widetilde{z}$; this situation is illustrated in Figure 5.10.



(a) QCD

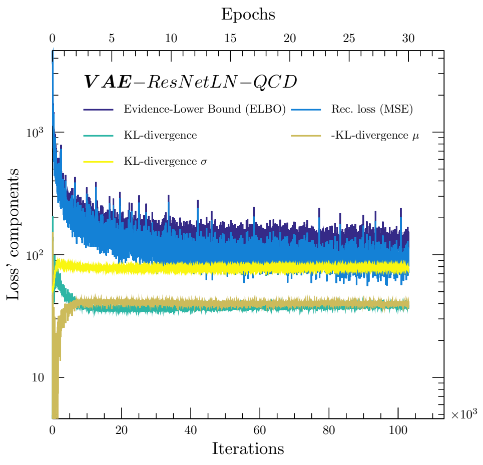


(b) W

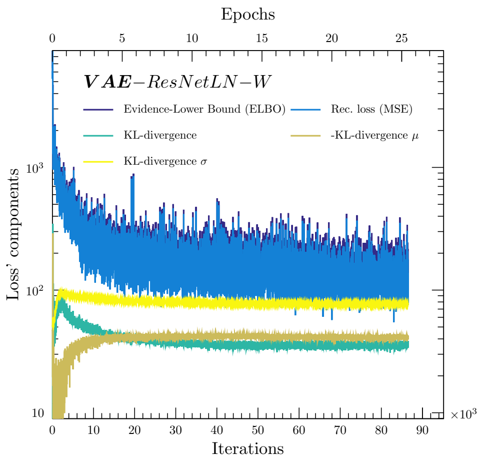
Plot 5.10: Correlation between $\widetilde{z}$ and the reconstructed energy E^{img}.

In case of QCD jets in Figure 5.10a, there appears to be a small negative correlation between the unique seed $\widetilde{z}$ and the reconstructed jet energy E^{img}. Regarding W jets however, the distribution is almost flat, which might be an explanation for why the discrepancy between the generated and expected energy spectra is much more pronounced in case of W than for QCD jets (compare Figure 5.10a and 5.8a, as well as Figure 5.10b and 5.8b).

As it is clear from Figure 5.8a and 5.8b, the network struggles to learn and reconstruct the energy distribution. The situation is worse for W initialized jets – probably due to the long tail towards higher energies (less Gaussian). Equally, the situation does not improve any further for a larger number of iterations; this can moreover be seen in the evolution of the ELBO in Figure 5.8.



(a) QCD

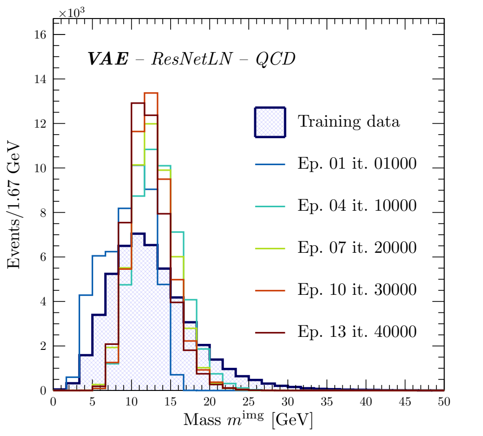


(b) W

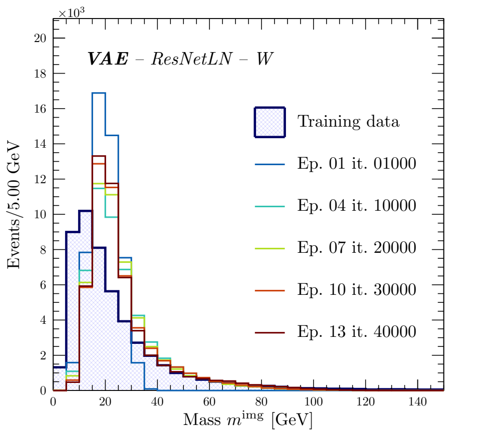
Plot 5.11: The ELBO and its components.

Of particular interest in Figure 5.11 is the Kullback-Leibler divergence since it is a direct measure of the *information* the inference network has encoded in the latent space. The KL-divergence (green curve) remains nearly static onward $10,000$ training iterations, which explains the behavior witnessed in Figure 5.11.

A similar problem can be observed in the spectrum of the reconstructed invariant mass of the jet (see Figure 5.12). Compared to the reconstructed energy of the image, the mass contains more information since it also accounts for the distribution of active pixels ("constituents") over the image/detector and their relative positions.



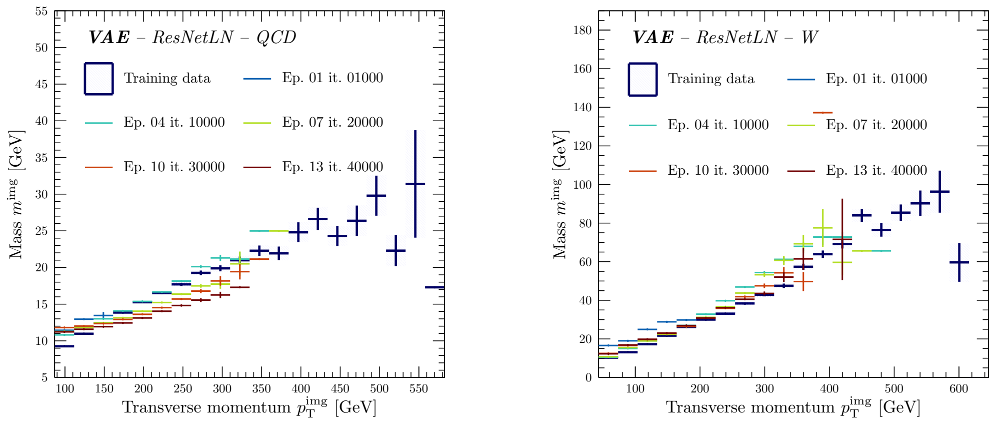
(a) QCD



(b) W

Plot 5.12: Reconstructed mass spectrum m^{img} for $50,000$ events.

Both distributions show significant deviations from the expectation based on the training data. The same trend can be observed for the correlation between the energy resp. the transverse momentum and the *average* reconstructed mass in Figure 5.13. As shown below, the unconditioned Gaussian VAE fails to learn the complete range of energy values for both processes. Therefore, the problems furthermore appears to be related to the large range of energy values, which is not well modeled.



(a) QCD *(b)* W

Plot 5.13: Correlation between the reconstructed average mass m^{img} and the transverse momentum $p_{\text{T}}^{\text{img}}$ for $50,000$ events.

The study of unconditioned GANs in the subsequent chapter will reveal a very different behaviour in case of unconditioned models. This result, already anticipated here, indicates that the problems observed in this section are specifically related to the unconditioned VAE. To further investigate Gaussian VAEs, the generative model should be factorized into two tasks: first, a model that is responsible for the parton shower simulation; and second, another model that learns the matrix element information. These are two generative models resembling the two simulation steps of the hard subprocess and the parton shower (see Chapter 2). The next section of this chapter therefore studies variational autoencoders that are conditioned on the jet energy and the pseudorapidity. This procedure will be repeated for Wasserstein GANs.

5.4 Conditional VAEs

The previous section studied the classical approach to a Gaussian variational autoencoder. The implemented model has been trained on a large set of jets images (that could correspond, e.g., to some measured jet in a control region in some particular analysis) with the learning objective not only to visually reconstruct the jet images (some distribution of energy in the detector), but also their associated characteristic energy spectrum. This is a complicated task: the network has to learn the distribution of pixels in the $n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}$ grid, the correlation between all those pixels and their energy. Furthermore, since the energy is continuous and many images may result in roughly the same reconstructed jet energy, the neural network has to theoretically learn an infinite number of classes, i.e., it has to learn a very complicated boundary between the different images in high-dimensions. This is one motivation for the investigation of the conditioned variational autoencoders that are the subject of this section since the neural network is relieved of its duty to classify the images according to their energy in the hidden space. In a conditioned variational autoencoder (see Section 3.5.4) the model receives additional inputs besides the latent space noise/seed vector $\boldsymbol{z}$. However, while the latent space vector is randomly sampled from a known distribution $\mathbb{P}_z$, the conditioning labels are directly associated with the training data – or the generated data after the training process. In this thesis, all conditioned neural networks (whether variational autoencoders or generative adversarial models) are conditioned on the energy of the *reconstructed* jet E^{jet} as well as the pseudorapidity η^{jet} before the Lorentz boost (see Section 4.2) but before pixelation (see Section 4.2). Through this measure, the model does not depend on any parton level information that would not be available in a down-to-earth scenario anyway.

The advantage of conditioned neural networks is that the model is provided with additional information regarding the training data. This makes the learning task much easier compared to the unconditioned case. One of the reasons why this is the case is because the network does not have to learn all classes associated with the data by itself. If the model uses with the energy and pseudorapidity information, it will be easier to categorize the data, i.e. the jet images according to those labels. This may also be considered as a beneficial regularization effect that helps to lower the risk of overfitting and reduce the venture of mode collapse.

This section follows the previous one regarding its structure. The configuration of hyperparameters (optimizer, learning rate α_l, dimension of the latent space etc.) is – if not explicitly mentioned otherwise – equivalent to the setup used for the unconditioned variational autoencoders. However, due to the additional information that needs to be incorporated into the encoder and decoder model (see Figure 5.1), the architecture is different – albeit only marginally – since the supplementary connections increase the model's number of weights. To exclude the possibility that the differences between the unconditioned and the conditioned model are caused by the "increased complexity" of the network and not because of its conditioning, the unconditioned VAE has been systematically scanned for increased model complexity. However, as it has become apparent, the results in the prior section are consistent within small variations for an increased number of weights. Hence, it is fair to assume that the difference between unconditioned and conditioned variational autoencoder is indeed exclusively caused by the additional information provided to the network.

5.4.1 Conditioning the model

This paragraph starts with the difference between conditioned and unconditioned VAE regarding their probabilistic interpretation. The unconditioned model learns the probability function $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ which is, as it is the case for all latent space models (Section 3.5.1), conditioned on the latent space variable $\boldsymbol{z}$. The output of the generative model, i.e., its prediction $\hat{\boldsymbol{x}} = f_\theta(\boldsymbol{z})$ therefore depends solely on the latent space that, hopefully, encodes all information of the data. However, the jet image is associated with an energy E^{img} as well as a pseudorapidity η^{img} as fundamental kinematic variables. The distribution of the energy and the pseudorapidity is process-specific and hence depend on the underlying event that is given by the matrix element $\mathcal{M}$ of the hard subprocess. Therefore, the neural network does not only need to encode the jet images into the latent space but also the information of the matrix element. More formally, E^{img} and η^{img} are conditioned on $\boldsymbol{z}$ as well with $p_\theta(E^{\text{img}}, \eta^{\text{img}}|\boldsymbol{z})$. The last probability distribution can be factorized

$$p_\theta(E^{\text{img}}, \eta^{\text{img}}|\boldsymbol{z}) = \frac{p_\theta(E^{\text{img}}, \eta^{\text{img}}, \boldsymbol{z})}{p(\boldsymbol{z})}, \tag{5.4}$$

$$
\begin{aligned}
&= \frac{p_\theta(E^{\text{img}}, \boldsymbol{z})}{p(\boldsymbol{z})} \cdot \frac{p_\theta(\eta^{\text{img}}|E^{\text{img}}, \boldsymbol{z})}{p_\theta(E^{\text{img}}, \boldsymbol{z})} && (5.5)\\
&= p_\theta(E^{\text{img}}|\boldsymbol{z}) \cdot p_\theta(\eta^{\text{img}}|E^{\text{img}}, \boldsymbol{z}) && (5.6)\\
&\overset{E^{\text{img}} \perp\!\!\!\perp \eta^{\text{img}}}{=} p_\theta(E^{\text{img}}|\boldsymbol{z}) \cdot p_\theta(\eta^{\text{img}}|\boldsymbol{z}), && (5.7)
\end{aligned}
$$

whereby the last step used the fact that the energy and the pseudorapidity are (mostly) uncorrelated[5]. In summary, the generative model in the unconditioned case correspond to the following probability distribution

$$
p(\boldsymbol{x}, E^{\text{img}}, \eta^{\text{img}}|\boldsymbol{z}) = p(\boldsymbol{x}|E^{\text{img}}, \eta^{\text{img}}, \boldsymbol{z}) \cdot p(E^{\text{img}}|\boldsymbol{z}) \cdot p(\eta^{\text{img}}|\boldsymbol{z}), \tag{5.8}
$$

which basically rephrases the problem explained above. Obviously, the learning task can be simplified by making the networks *not* learn $p(E^{\text{img}}|\boldsymbol{z})$ and $p(\eta^{\text{img}}|\boldsymbol{z})$ and this is precisely where conditioned networks come into play. The idea is to train the neural network on the jet images, sample the energy and the pseudorapidity of the jets from the distributions $\mathbb{P}_E$ and $\mathbb{P}_\eta$ known from the matrix element and provide those as an additional information to the neural network. From a probabilistic point of view, this procedure corresponds to the marginalization of Equation 5.8 with respect to E^{jet} and η^{jet}

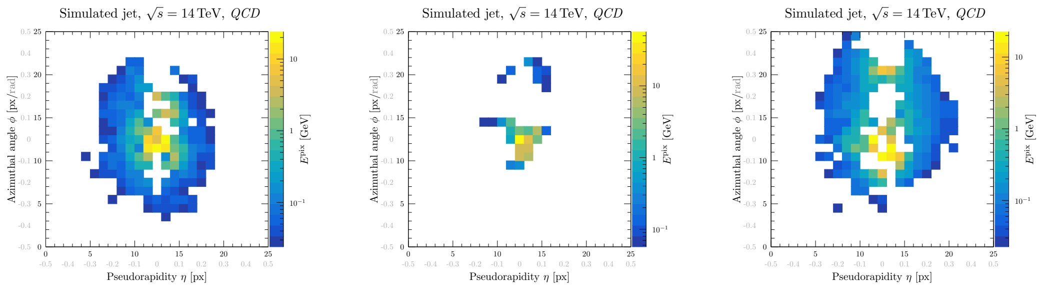
$$
p(\boldsymbol{x}|\boldsymbol{z}) = \int_{E^{\text{jet}}} \int_{\eta^{\text{jet}}} \mathrm{d}E^{\text{jet}\prime} \mathrm{d}\eta^{\text{jet}\prime}\, p(\boldsymbol{x}, E^{\text{jet}\prime}, \eta^{\text{jet}\prime}|\boldsymbol{z}). \tag{5.9}
$$

After this detailed discussion of the underlying probabilistic principles at work, it be expected that the unconditioned variational autoencoder will perform better since it is liberated from the heavy burden of learning the matrix element information as well.

The conditioning labels E^{jet} and η^{pix} in Equation 5.8 are sampled from the probability distribution of the energy $p(E^{\text{jet}})$ and the pseudorapidity $\mathbb{P}_\eta$ directly given by the matrix element as it is associated with the respective jet image

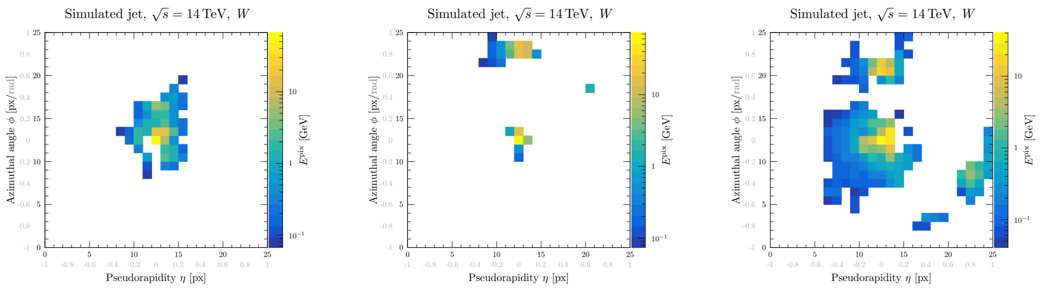
5.4.2 Samples and average jet images

As it was done in the previous section, the first step is to visually inspect the generated data that has been generated by the neural network and to look at the mean jet image to make sure that the network has learned the *average* correlation between the pixels.



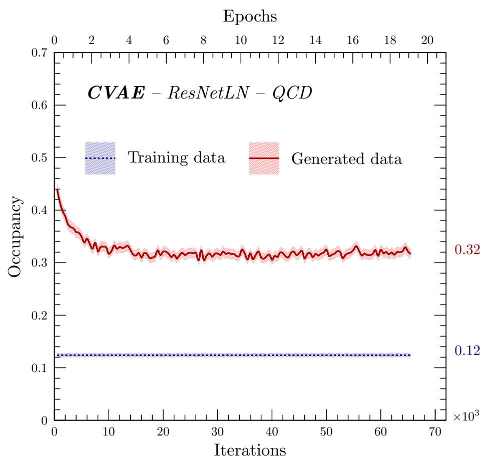
Plot 5.14: Three randomly simulated QCD jets.

[5]This was a main motivation why choosing the energy in favor of the jet's transverse momentum $p_{\text{T}}^{\text{jet}}$ since it allows to independently samples values for E^{jet} and η^{jet}. However, the approximation $p(E^{\text{img}}, \eta^{\text{img}}|\boldsymbol{z}) \approx p(p_T^{\text{img}}|\boldsymbol{z}) \cdot p(\eta^{\text{img}}|\boldsymbol{z})$ is still valid for pseudorapidities within a certain range where the correlation is negligible: in case of QCD jets this tuned out to be roughly the case for $\eta^{\text{jet}} \in [-2, 2]$.

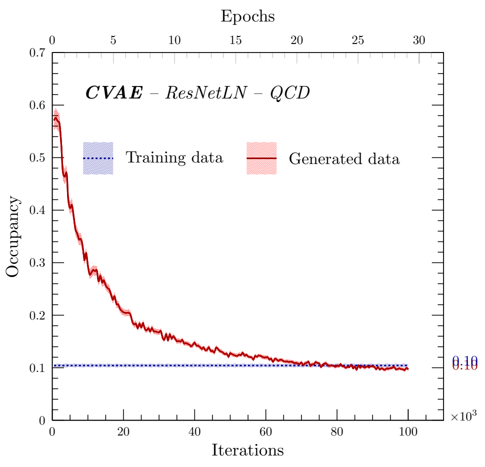


Plot 5.15: Three randomly simulated W jets.

Compared to the QCD (Figure 5.3) and W (Figure 5.4) jet images simulated by the unconditioned variational autoencoder, the conditioned model appears to generate data that is less blurred (this is by no means a general statement since the random sample only envelopes three data points and thus statistically not significant). This impression is further supported by the reduced occupancy of the images that corresponds to an increased sparseness as illustrated in Figure 5.16.



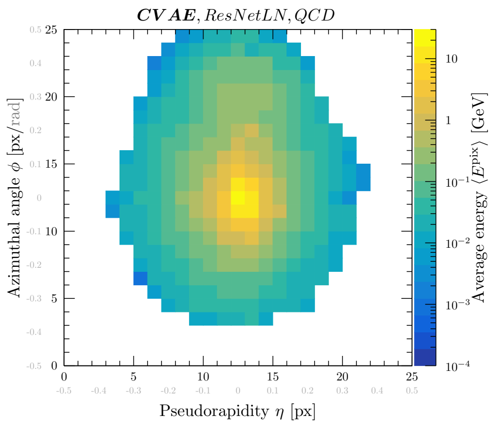
(a) QCD



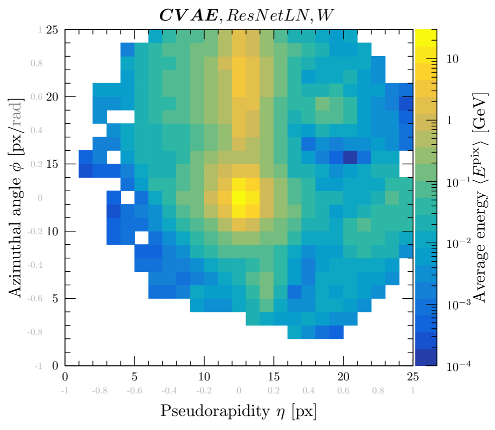
(b) W

Plot 5.16: Occupancy according to Equation 5.2. The reported errors $\delta\bar{o}$ are statistical uncertainties based on "propagation of errors".

The direct comparison of the average occupancy of the images generated by the conditioned VAE in Figure 5.16a and 5.16b ($\bar{o}^{\mathrm{QCD}} \approx 37\,\%$, $\bar{o}^{W} \approx 13\,\%$), as well as the one in Section 5.3.1 (Figure 5.5a and 5.5b) ($\bar{o}^{\mathrm{QCD}} \approx 32\,\%$, $\bar{o}^{W} \approx 10\,\%$) show a considerable improvement. For both processes, QCD and W jets, the average occupancy is reduce roughly by 5% – in case of W jets, the average occupancy now coincides with the one of the training data within statistical uncertainties. This, along with the visual improvements, is a first indication that the arguments in Section 5.4.1 regarding the factorization of the matrix element (Equation 5.8) information are indeed correct and the performance of the model is improved. This is also observed in the average jet image in Figure 5.34.



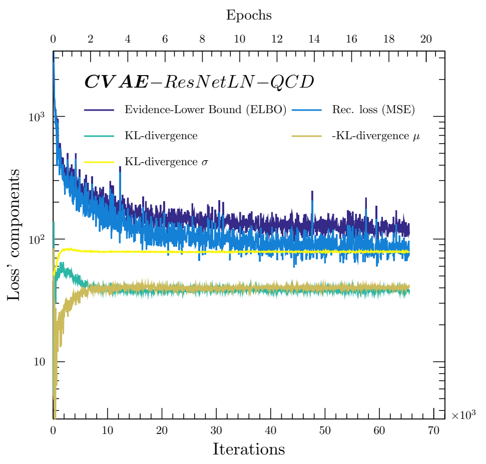
(a) QCD



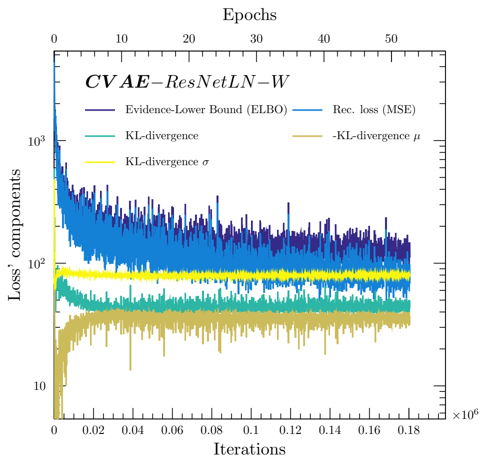
(b) W

Plot 5.17: Average jet image for 50,000 events and 40,000 iterations.

While the difference between the mean jets in case of QCD (*cf.* Figure 5.34a and 5.6a) is rather negligible, the average W jet shows a clear improvement compared to Figure 5.6b and is generally much closer to the one of the training data according to Figure 4.4. This means that the network has improved in learning the correlation between the individual pixels – at least on average. Interestingly, the improvements are not substantially reflected in the agreement between the different statistical moments (not shown at this occasion), which roughly corresponds to Figure 5.7. Nonetheless, the positive effect of conditioning the neural network is indisputable. Finally, this can also be seen in the reconstructed loss as well as the KL-divergence.



(a) QCD



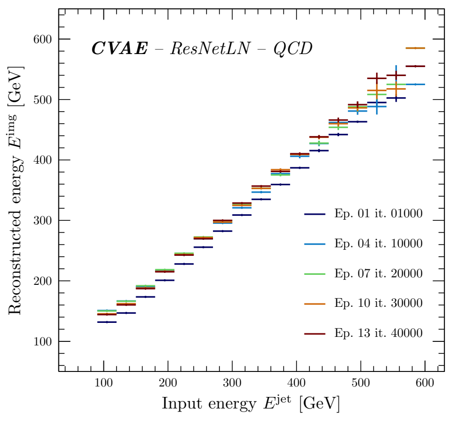
(b) W

Plot 5.18: The ELBO (see Equation 3.39) and its components.

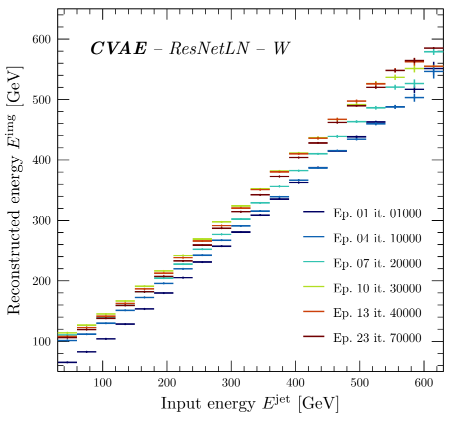
5.4.3 Conditioning of the model

As it has been done before, this section dives deeper into the manifold generated by the conditioned variational autoencoder.

The first step is, of course, to verify that the neural network is indeed conditioned on the jet energy and the pseudorapidity. The "stochastic insertion" of the conditioning label described in Footnote 4 ensures that the model does not simply disconnect the additional information by setting the associated weights to zero (like it was purposely done for the unconditioned VAE). However, the network hypothetically could – although very unlikely – learn to average over the conditioning labels to soften the constraint. To verify whether the network is conditioned on the energy or not, the conditioning energy label is plotted against the reconstructed energy from the generated image. If the network is conditioned, one expects a strong, positive correlation between E^{jet} and $E^{\text{img}} = \sum_{i=1}^{n_\eta^{\text{pix}}} \sum_{j=1}^{n_\phi^{\text{pix}}} E_{ij}^{\text{pix}}$. This correlation is shown in Figure 5.19 for QCD and W jets.



(a) QCD



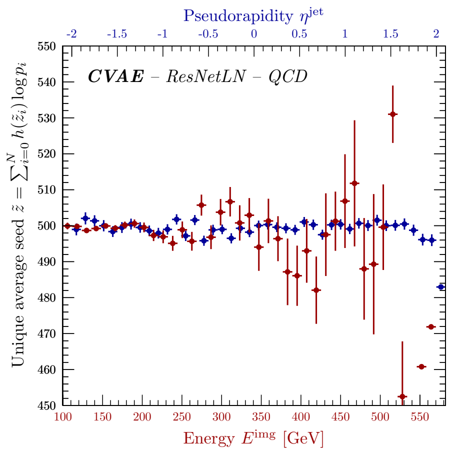
(b) W

Plot 5.19: Input energy E^{jet} and reconstructed jet energy E^{img}.

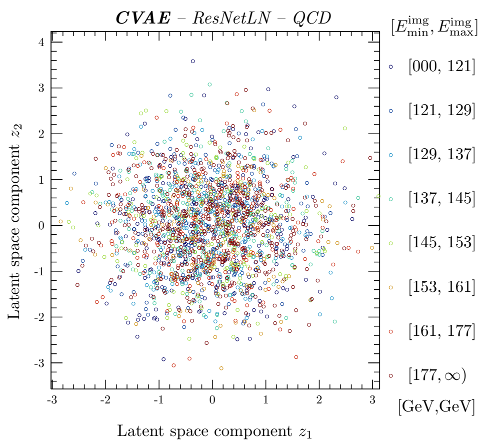
The expected strong correlation between the reconstructed jet energy E^{img} and the conditioning label is visible in the two diagrams above. Therefore, as a matter of fact, the conditioning energy E^{jet} input allows controlling the reconstructed energy of the jet! In the case of QCD jets, the correlation is almost one-to-one while for W jets the relation appears to be non-linear on the edge of low and high energies. An additional curve, corresponding to $70,000$ training iterations, have been included to see if this effect is reduced. That does seem to be the case (see last bin). Interestingly, the network first learns a linear relation that underestimates the input-energy. However, it develops a non-linear relation while trying to improve the reconstruction. This behaviour is not fully understood at the time of writing.

Now that it is guaranteed that the energy of the jet is controlled by the input label, it is about time to study to what extent the trained model has learned to simulate parton showers. To this end, different figures of merit, as well as their corrections among each other are closely examined. Each quantity is supposed to provide a different perspective on the generated manifold. For example, the reconstructed *energy* that is position-independent (scalar sum) and therefore only measures the quality of the regression task, the reconstructed *mass* that depends on the energy values in the pixel cells as well as their relative position, the *N-subjettiness* that probes the substructure of the jet etc. The more aspects of the generated manifold are aggregated, the more they form an overall picture of the data's structure.

For completeness and for QCD only, consider again a two-dimensional subset of the latent space, categorized into energy bins of equal statistics, as well as the "unique" latent space variable $\widetilde{z}$ as defined in Section 5.3.2 for a conditioned Gaussian variational autoencoder.



(a) Unique $\widetilde{z}$ versus reconstructed jet energy E^{img} and input pseudorapidity η^{jet}.



(b) Energy-correlation in a two-dimensional subspace of $\mathcal{Z}$.

Plot 5.20: Correlations in the latent space.

While a small negative correlation between the energy of the generated jet and the associated unique latent space number $\widetilde{z}$ in Figure 5.10 in case of the unconditioned model was observed, the latent space vector and the respective energy are uncorrelated according to Figure 5.20. This result is in accordance with the previously explained effect concerning the classification of images based on their associated energy. The neural network does not learn the energy distribution of the jet; hence, this information does not have to be encoded in the latent space since it is provided externally[6].

5.4.4 Kinematic distributions

All figures of merit that can be derived from the jet images are either based on the energy values in the pixels E_{ij}^{pix} and/or the position of the active pixels in the η-ϕ grid, i.e., the cells in the detector with an activation above a certain acceptance threshold. The most simple quantity one may reconstruct is the energy spectrum. To reconstruct the energy spectrum of a given process, all that needs to be done is to calculate the scalar sum of the energy values over the pixels on an event-on-event basis according to $E^{\text{img}} = \sum_{i=1}^{n_\eta^{\text{pix}}} \sum_{j=1}^{n_\phi^{\text{pix}}} E_{ij}^{\text{pix}}$.

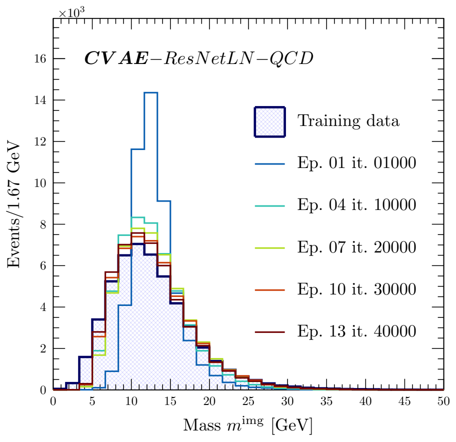
[6]The latent space itself remains a "black box" to some extent since the actual encoding is inaccessible. This is an unsatisfactory situation – to say the least. Future studies should, therefore, aim to imbue the latent space with some physical meaning.

This quantity does not take into account any positional information. Theoretically, the neural network could learn to perfectly reconstruct the energy spectrum just by mapping the conditioning energy input to one pixel in the image. This does not happen since the KL-divergence penalizes the network if it underestimates the entropy of the underlying distribution. Since the model is conditioned, the reconstructed energy spectrum should simply correspond to the distribution used to sample the conditioning label. This is the case since, as it was shown in Section 5.19 Figure 5.19a and 5.19b, the correlation between the input and the reconstructed output energy is roughly one-to-one. To gain more insight into the manifold learned by the generative model, it is necessary to account for the relative position of the active pixel cells as well. An obvious quantity that satisfies this need is the jet mass. The mass of the jet is given by the squared sum of the jet's constituents four momenta according to

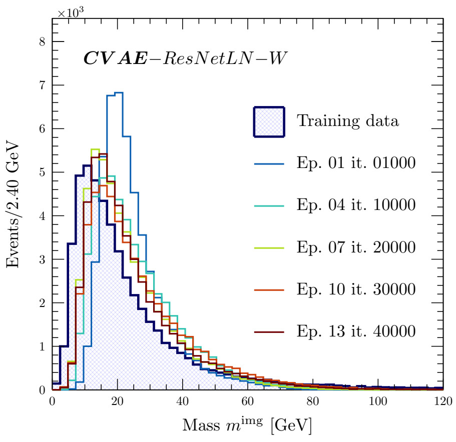
$$\left(m^{\text{jet}}\right)^2 = \left(\sum_{k=0}^{N-1} p_k\right) = \underbrace{\sum_{k=0}^{N-1} p_k^2}_{=0} + \sum_{i=0}^{N-1}\sum_{i\neq j}^{N-1} p_i p_j \tag{5.10}$$

$$= \sum_{i=0}^{N-1}\sum_{i\neq j}^{N-1} p_{\mathrm{T},i}\, p_{\mathrm{T},j} \left(\cosh\left(\eta_i - \eta_j\right) - \cos\left(\phi_i - \phi_j\right)\right), \tag{5.11}$$

whereby the constituents are assumed to be massless $p_k^2 = 0$. According to Equation 5.11, the mass probes not only the energy/transverse momentum values of the individual pixels cells by $p_{\mathrm{T},i}$ and $p_{\mathrm{T},j}$, but also their position in the η-ϕ grid.



(a) QCD

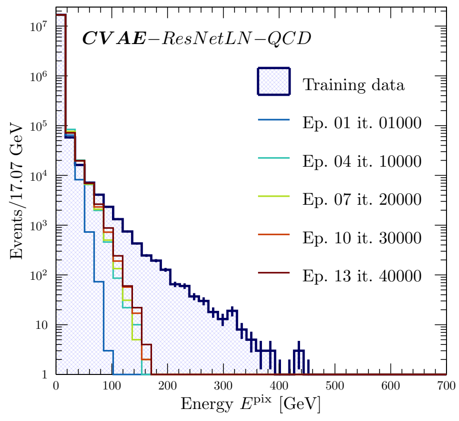


(b) W

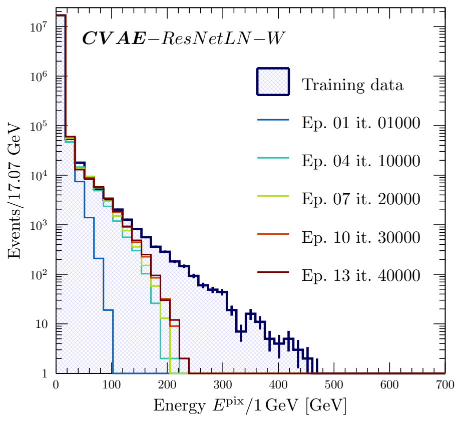
Plot 5.21: Reconstructed jet mass m^{img} for $50,000$ events.

From the comparison the reconstructed jet mass for the conditioned variational autoencoder (Figure 5.36) for QCD and W jets with the unconditioned model (Figure 5.12), it is obvious that the description has significantly improved. This is due to the factorization of the learning task, as already mentioned before.

It is only reasonable to study the individual contributions to Equation 5.11 in more detail. Although, the energy of the jet is provided as an input to the network and therefore not learned, the energy spectrum of the constituents E_{ij}^{pix} is certainly not. Therefore, the following plots show the distribution of the energy values (energy of the constituents) in the pixels – including empty cells $E_{ij}^{\text{pix}} = 0$ (for each image on an event-on-event base).

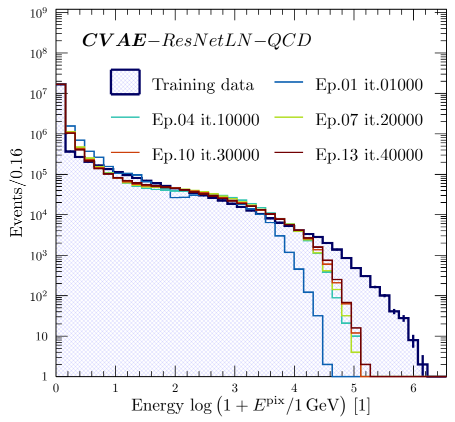


(a) QCD

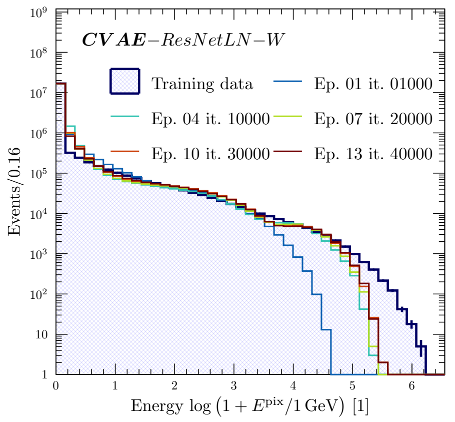


(b) *W*

Plot 5.22: Pixel activation values on an event-on-event base for $50,000$ images/events and $40,000$ iterations.



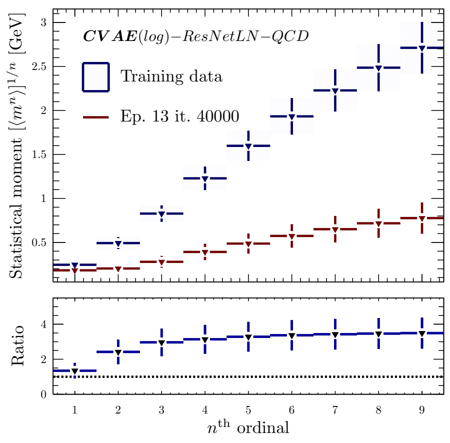
(a) QCD



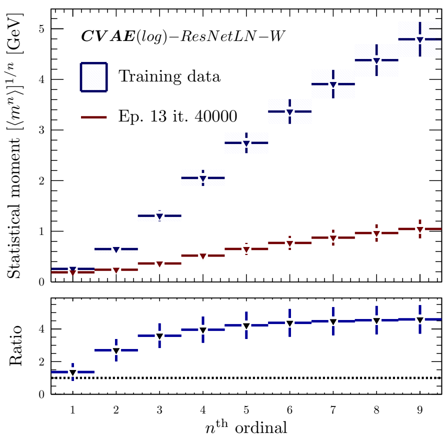
(b) *W*

Plot 5.23: Logarithmic representation of the pixel activation values on an event-on-event base for $50,000$ events.

Figure 5.22 emphasizes once again one challenge that needs to be mastered by the neural networks: the pixel values span a wide range of energy values. As the images are very sparse, most cells are empty with an energy value of $E_{ij}^{\text{pix}} = 0$. Due to the large amount of statistics for pixels with no activation, the VAE will focus mostly on accurately describing this bin at the expense of learning high-energy values. This is the source of all kind of problems when dealing with sparse data in general. So, it is all the more astonishing that the model provides a good description of the mass despite those large discrepancies in Figure 5.22. The reason is that the linear scale in Figure 5.22 mediates a wrong impression of the constituent's energy. Since the parton shower simulation described in Section 1.2.5 mostly adds soft radiation, the linear representation hides a lot of substructure in the lower bins. A logarithmic scale provides a different picture. According to Figure 5.23 the description is not so bad after all. The network learns an approximation of the spectrum and by doing so it focuses on the low energy contributions with large statistics. As it has been described in Section 4.3, the network was also directly trained on a data set whose elements (images) feature a logarithmic scale like in Figure 5.23. However, against all odds, the performance of the network deteriorates significantly compared to the linear scale. This is surprising since in case of Wasserstein GANs, which are covered later in the next chapter, the opposite behaviour could be observed. The discrepancy between training data and the simulated jets is particular pronounced regarding different statistical moments, which can be seen in Figure 5.24.



(a) Statistical moments QCD



(b) Statistical moments W

Plot 5.24: n^{th} statistical moment for generated QCD and W jets for a *logarithmic scale* (*cf.* Figure 5.7).

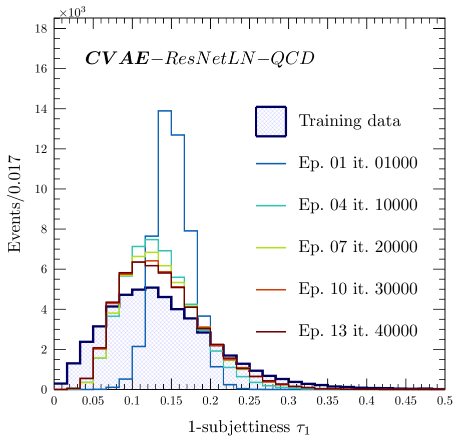
The deterioration compared to the corresponding diagram for the unconditioned VAE in Figure 5.7 is particularly evident in the ratio between the two graphs. Even for the very first moment, i.e., the mean of the distribution the deviation from the training data is significant. The origin of this effect is not entirely understood. A possible cause might be a modification of the reconstruction component in the ELBO (see Equation 3.40) in case of a log-scale. With the transformation $\boldsymbol{x}' = \log(1+\boldsymbol{x})$ and $f'_\theta(\boldsymbol{z}) = \log(1+f_\theta(\boldsymbol{z}))$, the reconstruction part is given by $\mathcal{L}^{\text{ELBO}} \supset -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\| (\boldsymbol{x}' - f'_\theta(\boldsymbol{z})) \|^2\right] = \mathbb{E}_{\boldsymbol{z}\sim\mathbb{Q}_\phi}\left[\log \prod_{i=1}^{n_\eta^{\text{pix}}\times n_\phi^{\text{pix}}} \frac{1+f_\theta^i(\boldsymbol{z})}{1+x^i}\right]$. The last Equation poses a problem since with this transformation the basic assumption of Gaussian distributed errors is replaced by something less intuitive and hence spoils the basic probabilistic assumptions. Furthermore, the expression is numerically problematic because the gradient may give rise to vanishing gradients. However, this is pure speculation at this point. Based on those observations, the linear scale was used.

5.4.5 Other jet observables

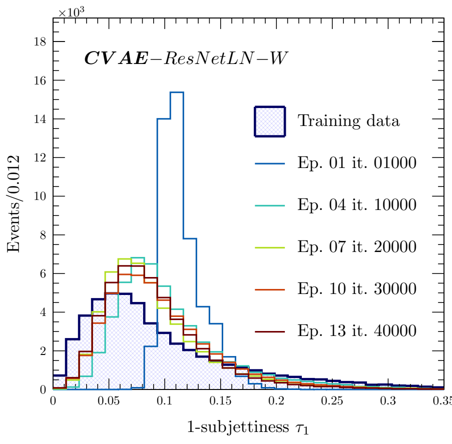
The "good" description of the mass – which still has space for improvement – indicated that the variational autoencoder not only learns to reconstruct the energy values of the pixelsm but also their relative position in the image/grid (see Equation 5.11). The next logical step is therefore to study a quantity that specially probes the *substructure* within the reconstructed jet as well. N-subjettiness (see Section 1.3.5) fulfills exactly this need. This paragraph studies 1-, 2 and 21-subjettiness (τ_1, τ_2 and $\tau_{21} := \tau_2/\tau_1$) all three of which have some interpretations. According to Equation 1.32, τ_1 is given by

$$\tau_1 = \sum_{i=1}^{N} \left(\frac{\Delta R_i}{R}\right) \left(\frac{p_{\mathrm{T},i}^{\mathrm{pix}}}{\sum_{j=1}^{N} p_{\mathrm{T},j}^{\mathrm{pix}}}\right). \tag{5.12}$$

The somewhat cumbersome representation of 1-subjettiness in Equation 5.12 provides a possible interpretation: it is the weighted sum of the relative distances from reconstructed jet in the η-ϕ grid. Hence, it gives the average distance of the constituent from the jet's reconstructed barycenter. This is, intuitively, closely related to the width of the jet (see Equation 1.31).



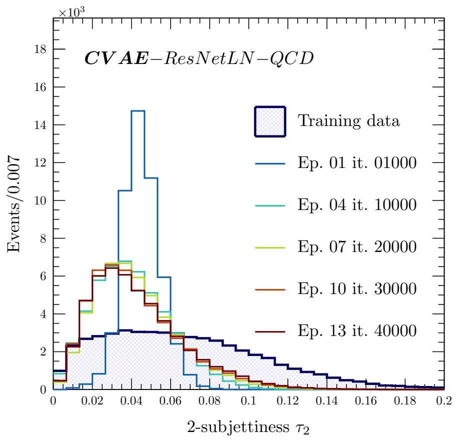
(a) QCD



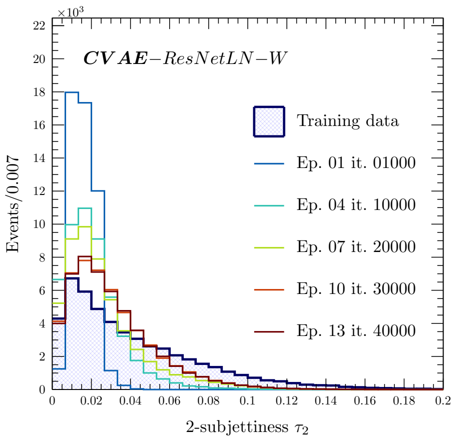
(b) W

Plot 5.25: 1-subjettiness for generated QCD and W jets.

Here the conditioned variational autoencoder fails. It comes as little surprise that τ_1 is rather poorly described with non-insignificant deviations from the true distribution for both processes. Interestingly, the mean value of the learned and the true distribution are consistent (which is in agreement with the good description of the first statistical moment), but *full width at half maximum* is not (the mean values are roughly consistent with the average jet image 5.16a,5.16a). This already reveals an apparent limitation of the model proposed in this section: the more complicated the substructure of the jets, the worse is the performance of the network. This is likely caused by an "undersizing" of the dimensionality of the latent space $\mathcal{Z}$. To reiterate: the dimensionality of the latent space is 100 while the training data has 625 dimensions (pixels). This narrow bottleneck with compression by a factor of roughly 0.16 will probably cause information loss. So, it is to be expected that there will deviations between the generated and the true distribution. Figure 5.26 and 5.27 show the distributions for τ_2 and τ_{21}.

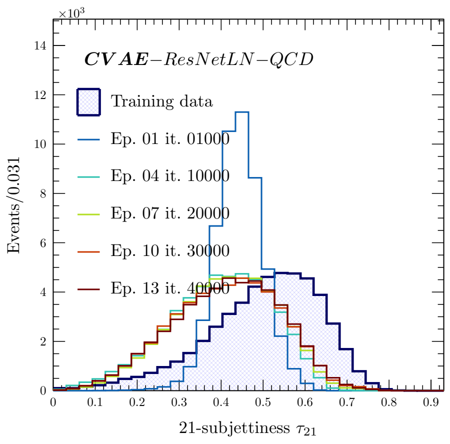


(a) QCD

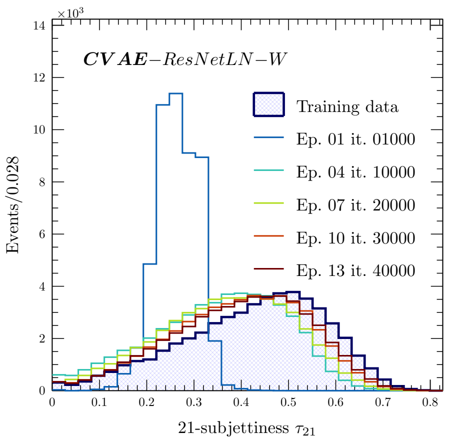


(b) W

Plot 5.26: 2-subjettiness for generated QCD and W jets.



(a) QCD



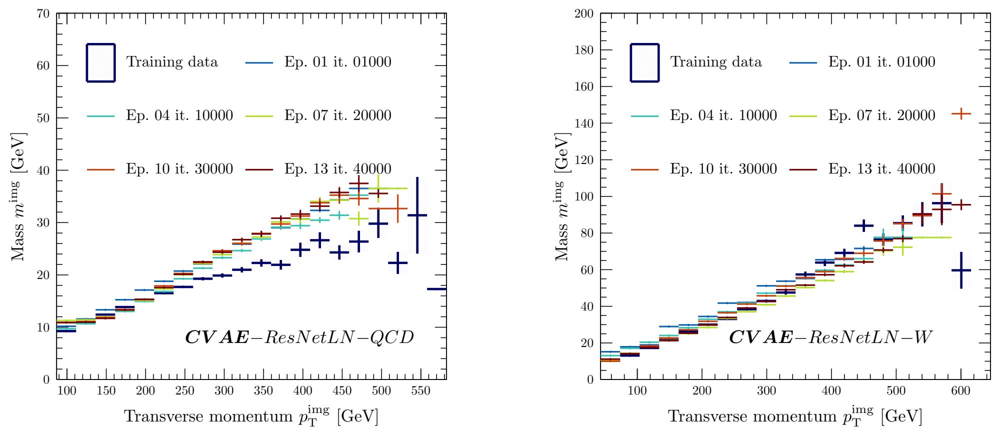
(b) W

Plot 5.27: 21-subjettiness for generated QCD and W jets.

In the case of QCD in Figure 5.26a, the description of the spectrum significantly deteriorates – which is to be expected. QCD jets usually cause a diffuse spray of radiation with less substructure. Therefore, τ_1, which corresponds to the assumption of just one (or fewer) subjets, is rather well described. The situation is different for W initialized jets. W jets, which decay to hadrons in the final state, normally cause two spatial regions of higher activity originating from the respective quarks. Hence, in case of W jets, τ_2 should be better approximated by the network compared to τ_1 since 2-subjettiness corresponds to the main feature of W jets, i.e., at least two subjets. This can *de facto* be observed in Figure 5.26b and 5.27b. Following this logic (without further demonstration), τ_3 should get worse for both processes QCD and W jets.

Until now, mostly "counting experiments" have been studied to check to what extent the network manages to fit the true distributions of the training data. However, those distributions provide little to no information about, e.g., the performance of the network in different mass or p_{T} bins. Therefore, it is only reasonable (with the mass and N-subjettiness being introduced) to study the correlation between different quantities. This provides more insight into the manifold that is learned by the Gaussian VAE and allows to systematically determine situations where the model fails.

From Figure 5.19 and 5.36 it is clear that the energy as well as the mass are rather well modeled. This observation is qualified by studying the correlation between those two figures of merit.

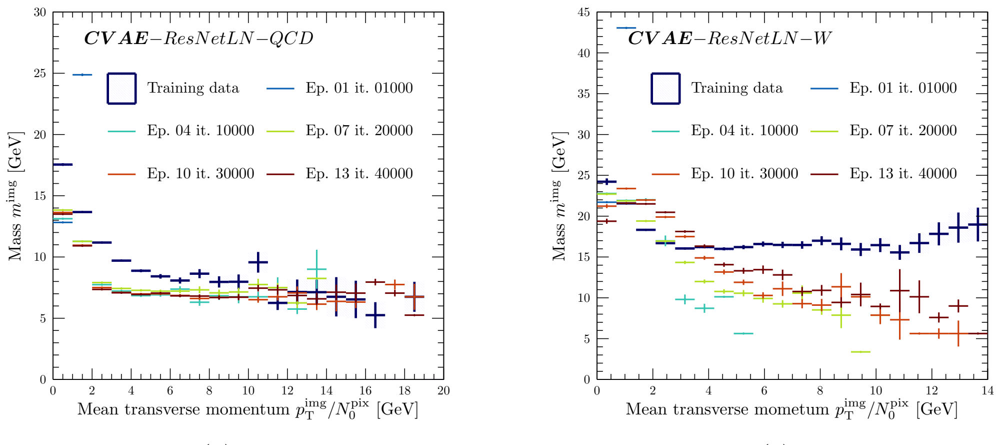


(a) QCD

(b) W

Plot 5.28: Correlation between the reconstructed mass m^{img} and the transverse momentum $p_{\mathrm{T}}^{\mathrm{img}}$ for $50,000$ events.

Both plots above show deviations for higher $p_{\mathrm{T}}^{\mathrm{img}}$ values – while the effect is much more pronounced for QCD jets. From Figure 5.28 it is furthermore expected that the correlation between the mass m^{jet} and the average transverse momentum per constituent $p_{\mathrm{T}}^{\mathrm{img}}/N_0^{\mathrm{pix}}$ will not be modeled well, which might also explain the deviation in Figure 5.28.



(a) QCD *(b) W*

Plot 5.29: Correlation between the reconstructed mass m^{img} and the average momentum per constituent $\frac{p_{\mathrm{T}}^{\mathrm{img}}}{N_0^{\mathrm{pix}}}$ for 50, 000 events.

Plot 5.29 finally reveals the actual Achilles' heel of the model, which is consistent with the observations in Figure 5.23 that shows the distribution of energy values in the pixels. The mismodelling in case of QCD jets in Figure 5.29a can be explained by the "blurring" and the accompanying increased occupancy (see Figure 5.16a) that cause a bias towards low values of $p_{\mathrm{T}}^{\mathrm{img}}/N_0^{\mathrm{pix}}$. For W jets on the other hand this argument is not valid.

5.4.6 Why the model "fails"

The title is somewhat provocative. However, based on the insights that have been gained up to this points, it is fair to say that the model (to a certain extent) fails to learn the underlying probability distribution of the training data. As it has already been described in Section 3.5, variational autoencoders are known to "blur" the data, or rather, to produce noisy samples. Many publications investigate this behaviour and several solutions have been proposed over the recent years such as e.g, Variational Lossy Autoencoder [Chen *et al.*, 2016], Improving Variational Inference with Inverse Autoregressive Flow [Kingma *et al.*, 2016] etc. Many reasons facilitate the apparent blurriness of the generated data. One factor that encourages the model to generate noisy data is the mean squared error loss used for the regression task, which corresponds to the assumption of a Gaussian distributed errors. In combination with a small bottleneck, i.e., low dimensionality of the latent space, the network might not be able to reproduce all features of the data precisely and therefore "smears" the generated samples to provide a good description on average. Subsequent studies should therefore further investigate the effect of the latent space as well as other state-of-the-art inventions surrounding variational autoencoders or, more preferably, utilizes adversarial models instead.

5.5 Conditioned VAEs with RNNs

The previous sections 5.3 and 5.4 studied conditioned and unconditioned variational autoencoders with a Gaussian constraint on the latent space. As it was shown, the description of the data improves significantly if the learning task is factorized into the actual shower algorithm and the matrix element of the underlying subprocess. However, the evident deviation between the generated and the real manifold strongly indicates that the generative model (without any assessment) "failed" – at least to some extend – to learn the underlying distribution of the training data.
Nonetheless, this section introduces another modification to Gaussian variational autoencoders with the objective to later apply this method to generative adversarial networks in the next chapter as well. Even though significant improvement compared to the previous configurations is rather unlikely, it is nevertheless useful to investigate the effect of combining variational autoencoders with recurrent neural networks in the context of parton shower simulation. These insights gained in this section, will be very helpful when combining RNNs with the more complicated GANs.

5.5.1 Why generative models with RNNs?

The introduction of recurrent networks represents yet another complication of the problem compared to the previous setup. Thus, the question why the combination of recurrent neural networks and generative models, in general, is considered to be beneficial is justified. To appreciate the argument, one has to remember that recurrent neural networks have been developed for the purpose of modelling time-sequential data (see Section 3.3.2), for instance, a sequence of words or letters. So, contrary to classical feed-forward neural networks without any feedback (loops), recurrent neural networks also have to learn temporal correlations between the data. To make this statement clear, consider again the example of a generative model that has been trained to generate English sentences out of a random seed $\boldsymbol{z}$. If the first word represents a noun, it is very unlikely to be directly followed by yet another noun; more likely, the network has to place a verb that describes the action in the sentence. More generally, the network has to learn the close connection between nouns, verbs, adjectives etc. in a sentence. In respect thereof, it has to account for the *previous* state of the system in the current decision-making process. This requires the network to learn temporal correlations; something that can not easily be done with simple feed-forward networks, which only learn correlations between the data at one instant. Another example would be the composition of music. The choice of cords in parallel harmonies and melodies strongly depends on previous notes, musical motifs and figures in the score. Therefore, for a neural network to compose meaningful music that appears pleasing to human beings, it must account for the previous states of the piece. Otherwise, the composition will sound random and most likely disharmonious most of the time (just like Arnold Schönberg's atonal music or twelve-note compositions) (for a generative adversarial neural network that has been trained to compose music see Yang *et al.* [2017]). So, recurrent neural networks are indispensable when temporal correlations play a crucial role in modelling the data.

As described in Chapter 2 and 1.3, the jets observed in the calorimeter of a detector are the result of a (temporal) sequence of emitted radiation that causes a cascade of secondary particles. In the simulation of parton showers, this process is sequentially modelled by a Markov chain and the probability that a parton does *not* split, given by the Sudakov form factors (see Section 1.2.5). With the actual underlying fundamental process in mind – the sequential emission of QCD radiation –, it seems only natural to model this process with recurrent neural networks that have been designed exactly for this area of application. However, there is a major limitation: the jet images used to train neural networks do not contain any sequential information anymore since the image itself represents a projection along the time dimension (the accumulation of particles over some time interval). It is not possible to reconstruct the order of the showering in the calorimeter of a detector; therefore, this information is not accessible and hence can not be used for training of the model. It is more than doubtful that the neural network will learn a meaningful shower sequence if the timestamp of the constituents is withheld. However, by using the recurrent neural network the model is "enforced" to model the data sequentially by any means necessary – whether it is actual sequential or not. It will be interesting to study how the neural network accomplishes this task if trained on jet images introduced in Chapter 4.

5.5.2 Combining VAEs with RNNs

How does one combine variational autoencoders with recurrent neural networks introduced in Section 3.3.2? The actual implementation is straightforward. The architecture of the encoder according to Figure 5.1a remains unchanged since only the decoder (the actual generative model) is supposed to be time-sequential. The choice of RNNs introduces several new hyperparameters as part of the LSTM layer (see Figure 3.4) that need to be tuned. Furthermore, the length of the time sequence n_T must be defined that gives the number of individual images $\boldsymbol{x}_t$ for each time step $t \in \{1, 2, \ldots, n_T\}$[7]. In the end, the model should return a single jet image $\boldsymbol{x} := h(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ that is composed out of the individual time sequences $\{\boldsymbol{x}_t\}_{t=1}^{n_T}$ by some function $h : \prod_{i=1}^{n_T} \mathbb{R}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}} \to \mathbb{R}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}}$. The simplest choice of h would simply be the sum of the individual images generated in each time step that also mostly corresponds to the underlying physical process of subsequent splittings

$$h\left(\{\boldsymbol{x}_t\}_{t=1}^{n_T}\right) = \sum_{t=1}^{n_T} \boldsymbol{x}_t. \tag{5.13}$$

The definition of h according to Equation 5.15 has a potential weakness: the neural network could learn to bypass the sequential model by (re-)producing the same output $\boldsymbol{x}_t$ exactly n_T times with $\boldsymbol{x}_1 = \boldsymbol{x}_2 = \ldots = \boldsymbol{x}_T$. In this case, each image $\boldsymbol{x}_t$ with time stamp t just represents output scaled by $1/n_T$

$$h\left(\{\boldsymbol{x}_t\}_{t=1}^{n_T}\right) = \sum_{t=1}^{n_T} \boldsymbol{x}_t = \boldsymbol{x}_t \sum_{t=1}^{n_T} = \boldsymbol{x}_t n_T = \boldsymbol{x}. \tag{5.14}$$

Empirical tests (not presented in this report) showed that this kind of risk can be reduced by introducing an explicit ordering in the series Equation 5.15 according to

$$h\left(\{\boldsymbol{x}_t\}_{i=1}^{n_T}\right) = \sum_{t=1}^{n_T} \boldsymbol{x}_t i. \tag{5.15}$$

Of course this can theoretically be bypassed by generating the same image scaled by $n_T(n_T + 1)/2$.

[7]In this thesis the length of the sequence of images is fixed for reasons of simplicity. However, based on fundamental arguments, the number of time steps n_T that generate an image should be a function of the energy since the average number of emitted gluons strongly depends on the energy via $n_T \sim \langle N_g \rangle \propto \exp\left(\sqrt{\frac{4C_A}{\pi b}\ln\frac{Q}{\Lambda}}\right)$.

The architecture of the decoder network is kept very simple; it only consists of a sequence of LSTM and dense layers as shown in Figure 5.4.

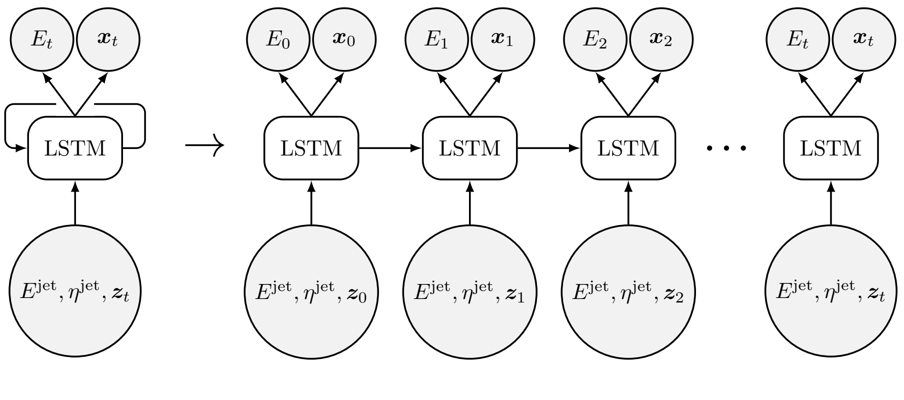

E_t
$\boldsymbol{x}_t$
LSTM
$E^{\text{jet}}, \eta^{\text{jet}}, \boldsymbol{z}_t$
$\rightarrow$
E_0
$\boldsymbol{x}_0$
E_1
$\boldsymbol{x}_1$
E_2
$\boldsymbol{x}_2$
E_t
$\boldsymbol{x}_t$
LSTM
LSTM
LSTM
LSTM
$E^{\text{jet}}, \eta^{\text{jet}}, \boldsymbol{z}_0$
$E^{\text{jet}}, \eta^{\text{jet}}, \boldsymbol{z}_1$
$E^{\text{jet}}, \eta^{\text{jet}}, \boldsymbol{z}_2$
$E^{\text{jet}}, \eta^{\text{jet}}, \boldsymbol{z}_t$

Fig. 5.4: The decoder combined with an RNN. As an input, the LSTM layer receives two conditioning labels, jet energy E^{jet} and the pseudorapidity η^{jet}, as well as the seed $\boldsymbol{z}_t$, which is different for each time step t. The output for each time step, i.e., each loop cycle, is a normalized vector and a scalar that corresponds to the energy at time t.

For each time-step, the LSTM layer returns an energy value E_t as well as a vector $\widetilde{\boldsymbol{x}} \in \mathbb{R}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}}$ with $\sum_{t=1}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}} x_t = 1$, i.e., output is a discrete probability distribution. This is achieved by using a dense layer with $n_\eta^{\text{pix}} \cdot n_\phi^{\text{pix}}$ output nodes with a softmax function $\sigma : \mathbb{R}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}} \to \left\{ \widetilde{\boldsymbol{x}} \in \mathbb{R}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}} | x_t \geq 0, \sum_{t=1}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}} x_t = 1 \right\}$ with $\sigma(\widetilde{\boldsymbol{x}})_t = e^{x_i} / \sum_{t=1}^{n_\eta^{\text{pix}} \times n_\phi^{\text{pix}}} \widetilde{x}_t$. The actual final jet image that is returned by the decoder network is then given by

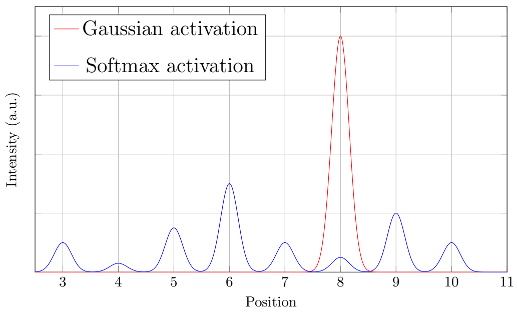
$$\boldsymbol{x} := h\left(\{\boldsymbol{x}_i\}_{t=1}^{n_T}\right) = \sum_{t=1}^{n_T} E_{(i)} \sigma(\widetilde{\boldsymbol{x}})_i i^\beta, \tag{5.16}$$

with $\beta \in \mathbb{R}$ being a tunable (hyper)parameter that introduces “time-ordering”. Henceforth, we refer to the operation defined by Equation 5.16 as *image generation layer* or *time projection layer*.

Using a softmax activation was not the first choice. The original intention was to only add one parton for each time step t, in connection the number of emitted gluons according to $n_T \sim \langle N_g \rangle \propto \exp\left(\sqrt{\frac{4C_A}{\pi b} \ln \frac{Q}{\Lambda}}\right)$. In doing so, each cycle in the LSTM layer would have a clear physical interpretation, i.e., the addition of a new gluon to the shower cascade for each time step. However, the implementation is difficult for several reasons. To add only one gluon per time t, the output of the LSTM layer must be modified such that the probability distribution given by the softmax activation is replaced by an operation σ' that assigns 1 corresponding to the position of highest probability and 0 otherwise

$$x'_t := \sigma(\boldsymbol{x})'_t = \begin{cases} 1 & \text{if} \quad x_i = \arg\min_{\boldsymbol{x}}(\sigma(\boldsymbol{x})) \\ 0 & \text{otherwise} \end{cases} . \tag{5.17}$$

Equation 5.17, although very intuitive, has a serious problem, i.e.m it is not *continuously differentiable*. Backpropagation (Section 3.4.3) and gradient descent (Section 3.4.2), however, require continuously differentiable functions to compute gradients of the loss function (Section 3.4.1) and to update the model's weights accordingly. Therefore, Equation 5.17 can not be used in the context of supervised or unsupervised learning[8]. To work around this problem, it was tried to approximate Equation 5.17 by a Gaussian $\mathcal{N}(\mu_{t^{\max}}, \sigma^2)$ whereby $\arg\min_{\boldsymbol{x}}(\sigma(\boldsymbol{x})) = x_{t^{\max}}$ and $\sigma = 1/n$ with $n \in \mathbb{N} \setminus \{0\}$. The conceptual difference between the Gaussian and the softmax activation is shown in Figure 5.30.



Plot 5.30: The softmax activation gives a discrete probability distribution (in contrast to the continuous spectrum depicted) over the pixel positions while the Gaussian activation selects one particular pixel that corresponds to the highest probability – all other pixels are highly suppressed (distributions not to scale).

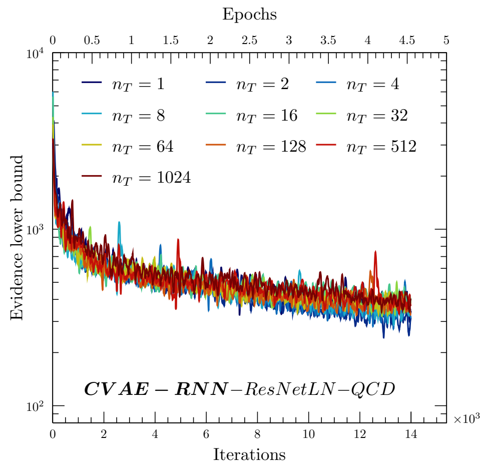
Even though the Gaussian approximation is continuously differentiable, it does suffer from vanishing gradients due to the small width σ that is required by the resolution criterion $\left(\frac{\mu_{t^{\max}} - x_t}{\sigma}\right)^2 \ll 1$ if $x_i \neq \mu_{i^{\max}}$. Hence, the model does not get updated anymore and thus remains static. In conclusion, the softmax activation was used as a compromise solution[9].

The actual number of LSTM cells and dense layers is part of the tuning of the model. Fortunately, in case of variational autoencoders, the ELBO and/or the KL-divergence provide an ideal measure of the performance of the model in terms of its reconstruction loss and/or the *information* encoded in the latent space as it was done before. Therefore, both quantities, the ELBO and the KL-divergence, are evaluated for different sequence length n_T as it was already done for the dimensionality of the latent space $\dim(\mathcal{Z})$ (Figure 5.2) and the learning rate α_l (Figure 5.2).

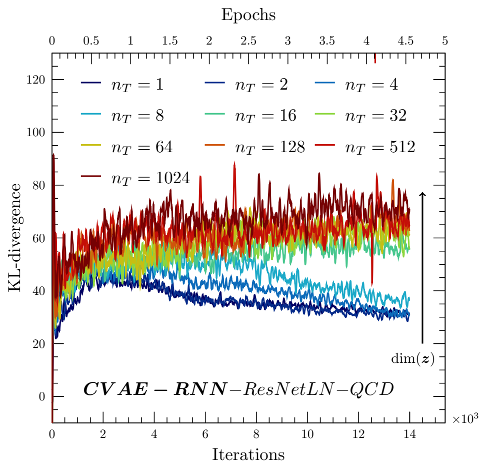
Figure 5.31a and 5.31b show the ELBO as well as the Kullback-Leibler divergence for an increasing number of time steps, successively increasing by powers of two.

[8]Reinforcement learning (see Section 3.2) does not have the restriction of a continuously differentiable cost function but allows to learn non-differentiable functions as well. It might therefore be worth trying to translate the problem to this paradigm of machine learning. Of cause, this would be a significant change at all levels with an uncertain outcome.

[9]This is still rather unsatisfactory since the softmax activation gives rise to the aforementioned problems regarding the replication of the same image for each time step. By construction, this issue does not occur if only one particle is added per cycle. Consequently, the recommended long-term scenario definitely would be to get rid of the softmax activation for good.



(a) ELBO



(b) KL-divergence

Plot 5.31: The ELBO and the KL-divergence (see Equation 3.41) versus the number of iterations parameterized for different numbers time steps n_T.

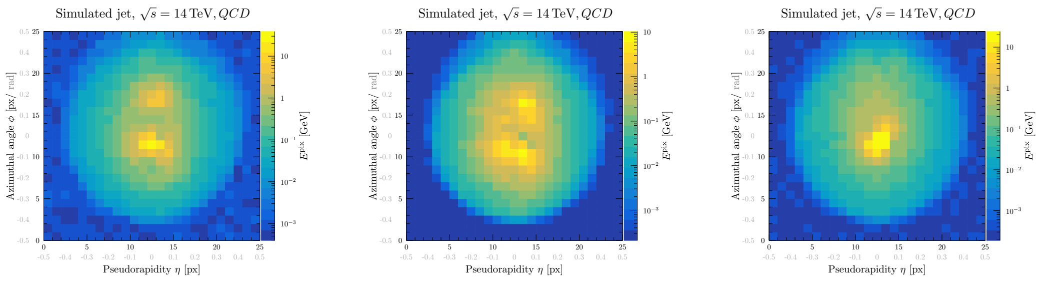
Again, the KL-divergence is significantly affected. This effect is caused by the increased complexity of the inference network that is accompanied by an increased number of time steps rather than a direct effect of the number of time steps. Contrary to the variational autoencoders in the previous section, the RNN requires an additional dimension that corresponds to the time axis. Therefore, the output of the encoder network is of dimension $\dim(\mathcal{Z}) \cdot n_T$, which corresponds to the number of neurons in the last layer of the network. Hence, the complexity of the encoder scales linearly with the number of time steps n_T. According to Figure 5.31, n_T was chosen to be 64.

The architecture of the generative model is very simple and only consists of one single LSTM layer that receives a vector of size $\dim(\mathcal{Z}) \cdot n_T$ as an input and has 512 output nodes followed by two (time distributed) dense layers with 512 nodes each. As illustrated in Figure 5.4, the network has two outputs: first, a single value that is the energy E_t at each time step; second, the flattened image $\boldsymbol{x}_t \in \mathbb{R}^{n_\eta^{\text{pix}} \cdot n_\phi^{\text{pix}}}$ at time t that is given by a dense layer $n_\eta^{\text{pix}} \cdot n_\phi^{\text{pix}}$. Normalization layers have been waived in experiments concerning VAEs with RNNs.

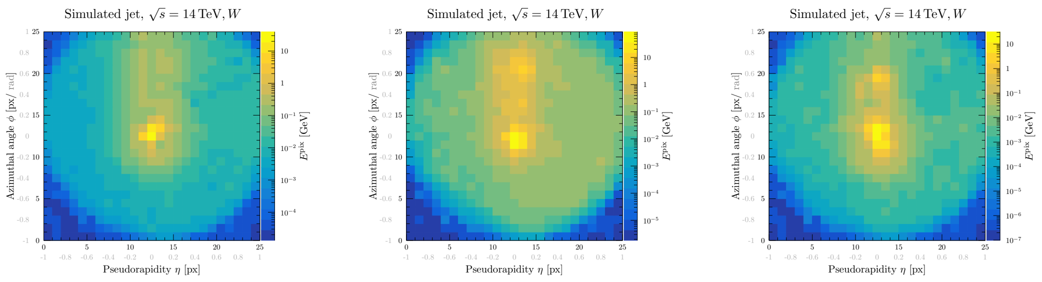
Comment: In case of variational autoencoders combined with recurrent neural networks each time step t gets its own seed z_t that is sampled from the latent space $\mathcal{Z}$. It was later realized, that this approach is conceptually very similar to increasing the number of samples in the Monte Carlo approximation as described in Section 3.5.3.

5.5.3 Samples, average jet image and jet observables

Like it was done in the previous two sections, the first step is to visually examine the simulated samples from the generative model combined with recurrent neural networks. Figure 5.32 and 5.33 show again three samples generated from a random seed $\boldsymbol{z}_t \sim \mathbb{P}_z$ for QCD and W initialized jets respectively. Recall, that in case of VAEs combined with RNNs according to the image generation layer (see Equation 5.16), each image is generated out of n_T seeds, i.e., one latent space vector for each time step.

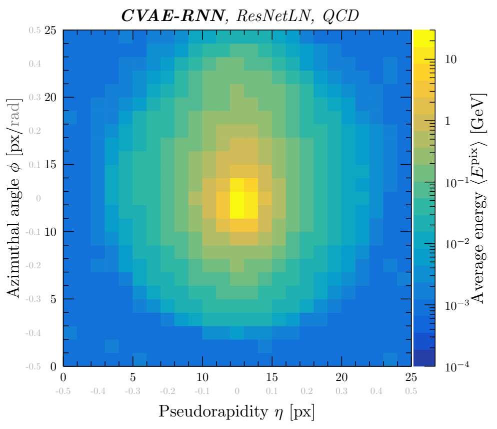


Plot 5.32: Three randomly simulated QCD jets.

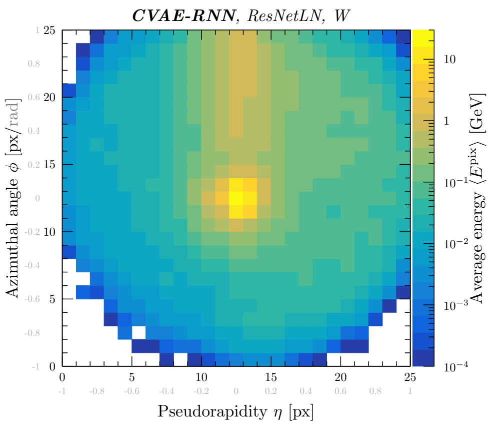


Plot 5.33: Three randomly simulated W jets.

The comparison between the simulated samples for VAEs in the previous sections (see Figure 5.3, 5.4 and 5.14, 5.15) and the ones that utilizes recurrent neural networks reveal an entirely different picture. The first feature that immediately strikes the eye in Figure 5.32 and 5.33 is the very high activity in the image, i.e., a significant increased occupancy compared to the previous examples. In fact, the occupancy is almost 100 % for each image on an event-on-event base. What is moreover conspicuous is that the individual samples separately are already quite close to the respective average jet image shown in Figure 5.34a and 5.34b.



(a) QCD

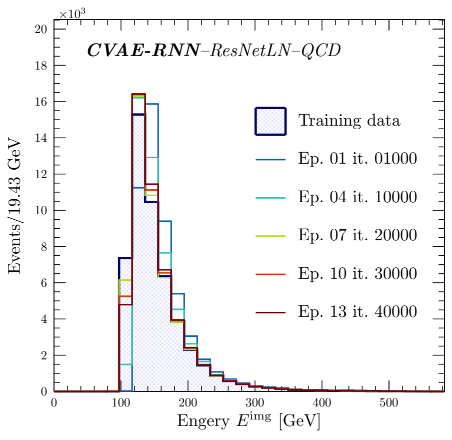


(b) W

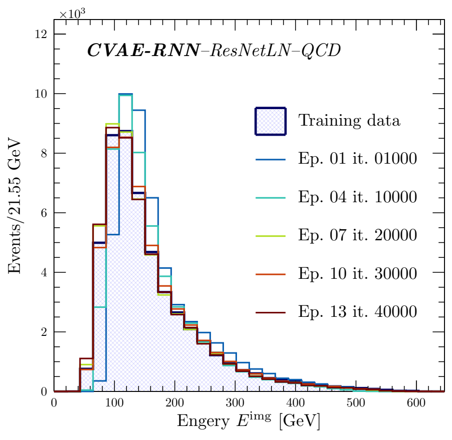
Plot 5.34: Average jet image for 50, 000 events and 40, 000 iterations.

As one can clearly see from the Figures above, the average jet image for 50,000 samples and the individual jet images (Figure 5.32 and 5.33) are in surprisingly close proximity to each other. This phenomenon is owed to the definition of the image generation layer $\boldsymbol{x} = \sum_{t=1}^{n_T} E_t \sigma(\widetilde{\boldsymbol{x}})_t t^\beta$ that merges several images to one final jet image by the summation over the individual images $\boldsymbol{x}_t = E_t \sigma(\widetilde{\boldsymbol{x}})_t$ generated at each time step. This is a strong indication that the neural network generates mostly random jet images for each time step t; hence, does not learn any temporal correlations between the particular samples. Again, it is hardly surprising that the network does not learn temporal correlations since this information is not provided during training. The training data itself is nothing but a two-dimensional projection, which does not simply allow to reconstruct the time-sequential character of the underlying physical process that generated the final pattern observed in the detector.

Surprisingly, the significantly increased occupancy does not adversely affect the reconstructed energy spectrum as illustrated in Figure 5.35.



(a) QCD

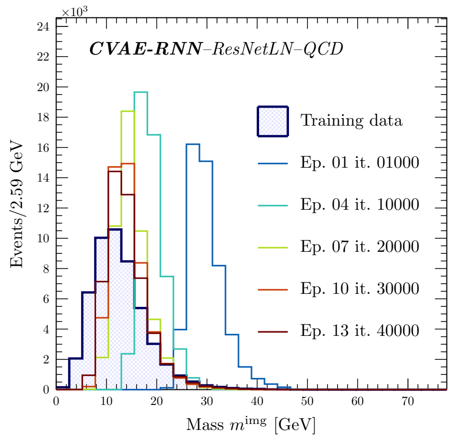


(b) *W*

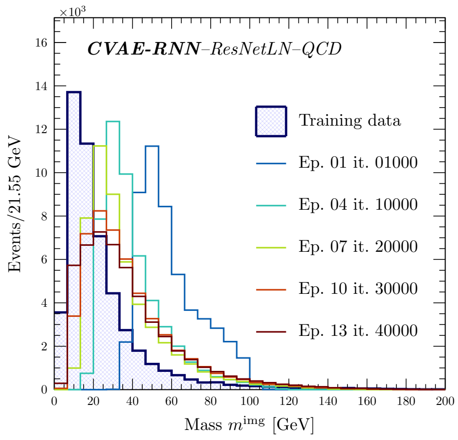
Plot 5.35: Reconstructed jet mass m^{img} for 50,000 events.

However, according to Figure 5.3 and 5.4, most pixel activations in the periphery of the image are very soft. Hence, the energy E^{img} is composed of two contributions: the structuring component of the image (high-energy activations primary in the center) that does come with higher energies and the softer component that is an artifact from the summation in the image generation layer. However, the soft contribution to the total energy is orders or magnitudes smaller compared to the highest pixel activation values and can therefore be neglected.

The situation is different if the respective jet observable takes into account the relative position between the individual pixels such as, e.g., the reconstructed mass (see Equation 5.11) that is shown in Figure 5.36.



(a) QCD

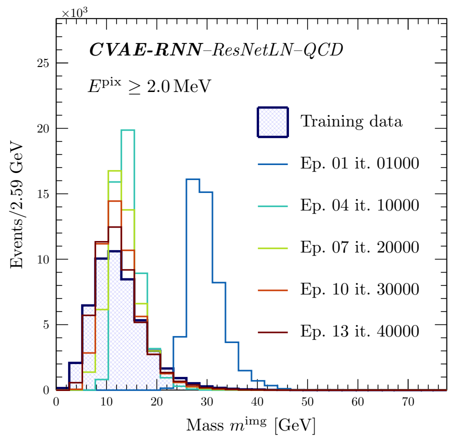


(b) W

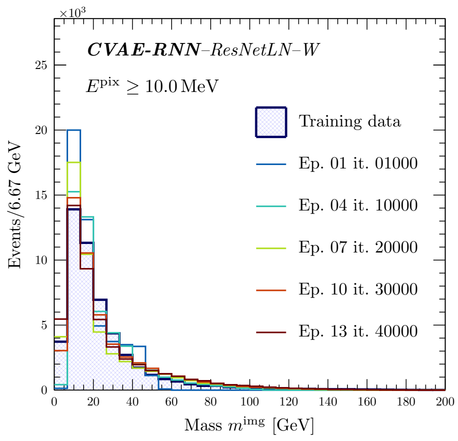
Plot 5.36: Reconstructed jet mass m^{img} for $50,000$ events.

The significant shift of the reconstructed distributions towards higher masses for both processes is a consequence of the increased occupancy in the image. Furthermore, the shift is positive since the increased activity is not uniformly distributed over the image but slightly shifted to the right hand side due to the preprocessing. Hence, the additional contribution to the reconstructed mass from the soft energy values $\left(m^{\text{img}}\right)^2 = \sum_{i=0}^{N-1} \sum_{i\neq j}^{N-1} p_{\mathrm{T},i} p_{\mathrm{T},j} \left(\cosh \delta\eta_{ij}^{\text{img}} - \cos \delta\phi_{ij}^{\text{img}}\right)$ does not completely average out.

To improve the consistency between the generated and the real mass spectrum, an energy-cut $E_{\text{th}}^{\text{pix}}$ is applied to each individual pixel. The result is shown in Figure 5.37.



(a) QCD



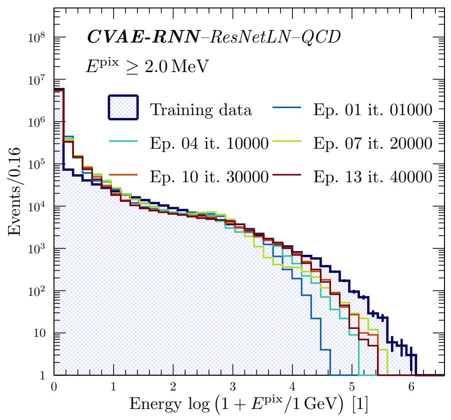
(b) W

Plot 5.37: Reconstructed jet mass m^{img} for $50,000$ events.

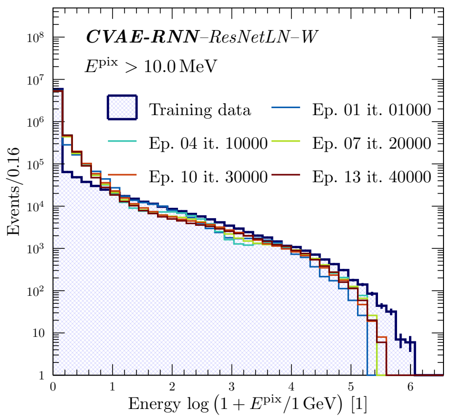
The energy threshold cut $E_{\text{th}}^{\text{pix}}$ has been varied to find the value that corresponds to the best agreement between two distributions (there are certainly other distributions that are rather suitable to determine the energy cut). In case of QCD, this threshold is at roughly

$E_{\text{th}}^{\text{pix}} \approx 2\,\text{MeV}$ while for W jets with $E_{\text{th}}^{\text{pix}} \approx 10\,\text{MeV}$ the energy cut must be chosen much larger to get a tolerable result. This threshold cut is of course quite unsatisfactory since it introduces a certain degree of arbitrariness.

The impact of this threshold cutting of the distribution of pixel activation values is shown in the Figure below.



(a) QCD

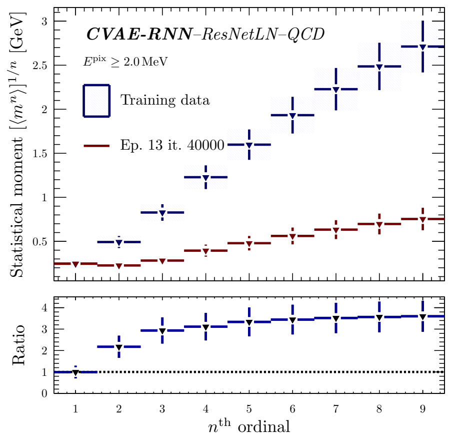


(b) W

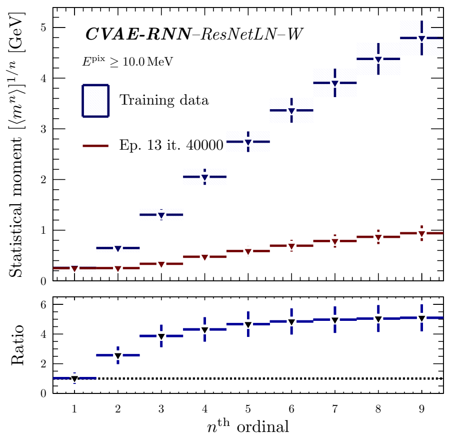
Plot 5.38: Pixel activation values on an event-on-event base for 50,000 events.

Still, even after the energy cut, the shift towards lower energy values is significant (*cf.* Figure 5.22) in the distribution of pixel activation values.

The deterioration of the description of the underlying distribution of the training data can also be observed in the significant deviations between the statistical moments of the generated and the expected distribution.



(a) Statistical moments QCD

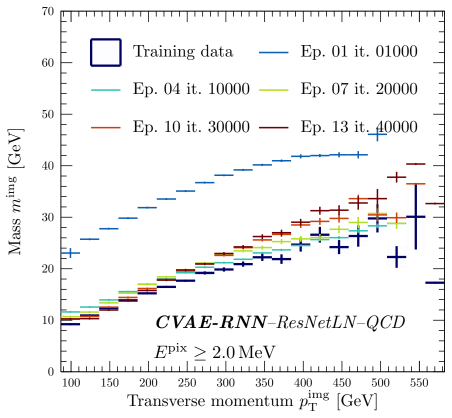


(b) Statistical moments *W*

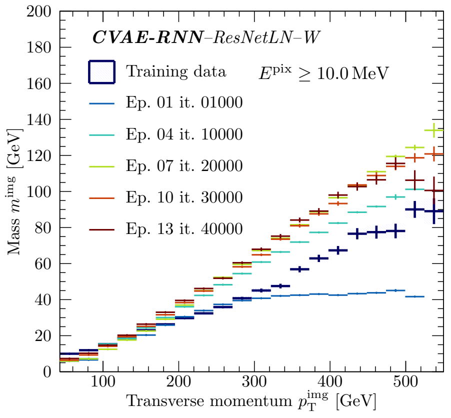
Plot 5.39: n^{th} statistical moment for generated QCD and W jets.

Compared to the variational autoencoder without recurrent neural network in Figure 5.24, the approximation of the target distribution has measurable deteriorated – especially in case of W jets.

If the correlation between the reconstructed mass and the transverse momentum is studied instead, one unexpectedly observes an improvement – dependend on the aforementioned energy cut – for QCD jets compared to the corresponding Plot 5.28. This, however, does already significantly changes in case of W jets as can be seen in Figure 5.40b.



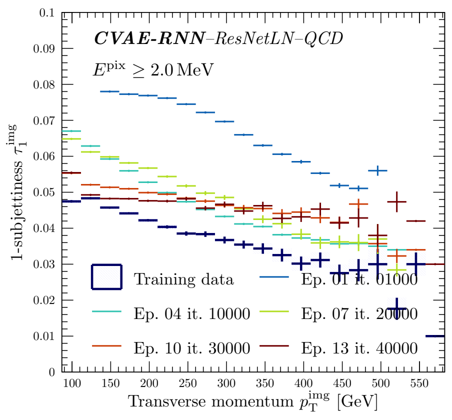
(a) QCD



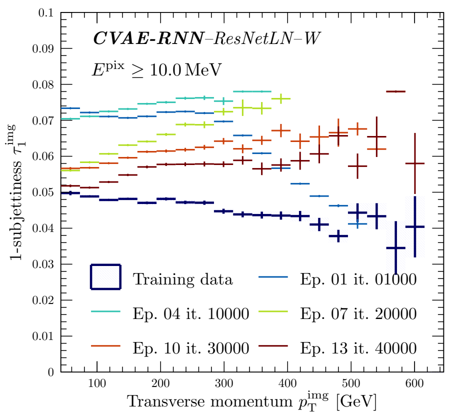
(b) W

Plot 5.40: Correlation between the reconstructed mass m^{img} and the transverse momentum p_T^{img} for $50,000$ events.

Finally, the model completely fails to reconstruct the substructure of the jet images compared to the training data, which is lost within the high occupancy in the image. This becomes particularly apparent in the reconstructed correlation between the 1-subjettiness τ_1 and the transverse momentum p_T^{img}. The large discrepancy for the processes is clearly visible in Figure 5.41.



(a) QCD



(b) W

Plot 5.41: Correlation between the reconstructed 1-subjettiness τ_1 and the transverse momentum p_T^{img} for $50,000$ events.

Based on Figure 5.41, it is obvious that the generative model does indeed fail to learn the relation between the relative position of the consistent jets and the transverse momentum of the jet. The description does not improve for other choices of the energy threshold cut. Under these circumstances, it is not reasonable to further investigate the variational autoencoders in more detail – although many other jet observables have been studied that provide a different perspective on the manifold learned by the generative model. The following brief paragraph is supposed to quickly summarize some insights that have been gained in the previous sections before giving attention to Wasserstein GANs in the subsequent Chapter.

5.6 Final notes

In this chapter, the applicability of variational autoencoders in the context of the simulation of QCD radiation has been studied. The first Section 5.1 introduced the architecture of the encoder as well as the decoder neural network. In the second Section 5.2 the hyperparameter configuration regarding the dimensionality of the latent space, the learning rate α_l and the optimizer used for gradient descent was motivated. With the technical questions being clarified, the *unconditioned* variational autoencoder has finally been trained on the generated QCD and W data as explained in the previous chapter. As it turned out, the unconditioned model shows a poor performance regarding the modelling of the training data, i.e., matrix element and parton shower. The reason for the significant discrepancies that have been observed between the generated and the real data is due to the fact that the neural network has to learn two problems in one model: the underlying information of the hard subprocess encoded in the matrix element as well as QCD radiation by means of parton shower simulation (see Section 1.2.5). Therefore, in the subsequent Section 5.3 the encoder and the decoder have been *conditioned* on the energy as well as on the pseudorapidity of the jet. From a probabilistic point of view, this step corresponds to the factorization into two generative models one of which models the matrix element and one that learns the shower simulation. Instead of learning the hard subprocess through a generative model, the information of the matrix element is directly taken from the training data (e.g. the distribution of the energy or the transverse momentum). Compared to the unconditioned VAE, conditioning the model on the energy and the pseudorapidity had a measurable positive effect on the performance regarding the quality of the generated samples. Nonetheless, significant deviations in several figures of merit could be observed, indicating that the generative model does indeed fail to learn the actual underlying distribution that generated the training data (particularly apparent in the progression of the statistical moments). This was one main motivation for the combination of variational autoencoders and recurrent neural networks, which have been introduced in Section 6.5. Using RNNs in this context is very tempting since the underlying process that generated the data is a time sequence of parton splittings; hence, recurrent networks seem to be a natural choice. However, the sequential nature of the actual process is not included in the training data; as a result, the network fails to use the options provided by RNNs and does not learn any temporal correlation. As a matter of fact, the description of the data noticeably degrades if VAEs are combined with RNNs – probably due to the definition of the time-projection layer. In general, it can be said that in the scope of this thesis the VAE reached its limit. The generative model mastered to superficially learn the approximate distribution of the data. However, a deeper look into the manifold (see Figure 3.8) reveals significant deviations, which are especially apparent if the 1-subjettiness is studied that probes the substructure of the jets. The results obtained so far are not satisfactory. The following chapter therefore takes a completely different approach to the problem through Wasserstein GANs.

Chapter 6

Jets with Wasserstein Generative Adversarial Networks

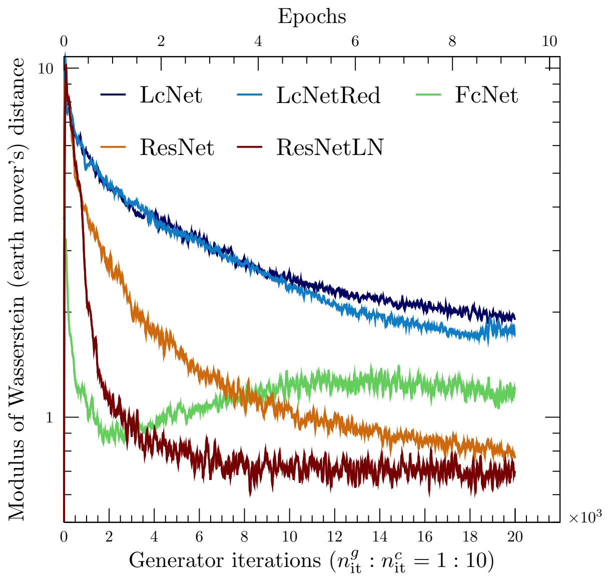
The previous chapter has clearly shown that the application of (Gaussian) variational autoencoders in the context of matrix element and/or parton shower simulation is rather limited. As a matter of fairness, it must be mentioned that this conclusion does by no means represent a general statement, but is restricted to the scope of this thesis. As always, the performance of the respective machine learning model under consideration strongly depends on the configuration of hyperparameters and the architecture of the network(s). Hence, it is very well possible that a different model configuration performs significantly better (or worse) than those presented in this report.

In this chapter, the variational autoencoders and their various variations are left behind to make room for generative models that are based on the Wasserstein generative adversarial networks, which have been introduced in Section 3.6. The structure of this chapter closely follows the previous one to allow for a simple comparison between the two different approaches to generative models. The first two sections are rather technical, explaining and motivating the architecture of the generator and the critic network (6.1) as well as the configuration of hyperparameters used in the training routine (6.2). Hereafter, unconditioned (6.3) and conditioned WGANs (6.4) are studied one by one, followed by the combination of WGANs and RNNs (6.5) like it was done in the previous chapter for VAEs. The chapter finally concludes with a summary of the acquired insights (6.6).

6.1 Architecture specification

On the whole, the networks' architecture used for the critic and the generator of the adversarial model is the same as the one introduced in Chapter 5 in the context of VAEs – with some minor but important changes. In the case of Gaussian variational autoencoders, the *encoder* has two outputs that correspond to the mean μ_ϕ and the variance σ_ϕ^2 of the Gaussian approximation by the inference network. Accordingly, the *decoder* network receives two inputs as well (disregarding the conditioning labels) to reconstruct the data from the latent space $\mathcal{Z}$. In the case of Wasserstein GANs on the contrary, the output of the critic network f_ϕ is a single number, i.e., a *score* in the range $[-\infty, \infty]$ (see Section 3.6.2) contrary to classical GANs in which the discriminator maps the input to a (classification) probability (see Section 3.6.1). The image set is of particular importance. Otherwise, the function f_ϕ is not able to rule out invalid transportation plans $\gamma \notin \Gamma(\mathbb{P}_r, \mathbb{P}_g)$. Hence, it fails to learn the earth mover's distance from the start. To ensure that the critic network indeed maps the input to the entire range of real numbers $f_\phi : \mathbb{R}^{n_\eta^{\text{pix}\prime}} \times \mathbb{R}^{n_\phi^{\text{pix}\prime}} \to \mathbb{R}$, it must *not* have an activation function in the last layer[1]. The input to the network of the critic is the same as for the encoder: an image from the training set $\boldsymbol{x} \in \mathcal{X}$ or generated by the generative model $g_\theta(\boldsymbol{z}) = \hat{\boldsymbol{x}} \in \mathcal{F}$ and, if the model is conditioned on external labels too, the energy E^{jet} as well as the pseudorapidity η^{jet} of the jet.

The situation is similar for the generator network g_θ, i.e., the actual generative model. Contrary to VAEs, the generator in WGANs only receives one input, i.e., the noise vector or seed $\boldsymbol{z} \in \mathcal{Z}$ samples from the distribution $\mathbb{P}_z$ over the latent space $\mathcal{Z}$ that is constrained to resemble a multivariate Gaussian distribution with mean 0 and variance $\mathbb{1}$. If the generative model is conditioned it also receives E^{jet} and η^{jet} as an input that is concatenated to $\boldsymbol{z}$. Except for the changes mentioned above, the architecture regarding the number of layers, residual blocks (see Figure 5.2), number filters etc. remains the same.



Plot 6.1: Modulus of the Wasserstein loss for different network architectures.

It should also be noted that in contrast to VAEs, (Wasserstein) GANs show a significant dependence on the underlying network architecture. Figure 6.1 shows the modulus of the earth mover's distance for several architectures with roughly the same number of trainable parameters (Locally-connected Network (LcNet), Fully-connected Network (FcNet), and Residual Networks (ResNet) with Layer Normalization (ResNetLN)) (*cf.* Section 5.1). The family of graphs show notable differences. Especially the layer normalization has proven to be very beneficial for the reduction of training time.

[1]In fact, the critic network might have an activation function $\sigma : \mathbb{R}^{n_\eta^{\text{pix}\prime}} \times \mathbb{R}^{n_\phi^{\text{pix}\prime}} \to \mathbb{R}$ with $\sigma \neq \text{id}_{\boldsymbol{x}}$ in the last layer as long as the image set $\text{Im}(f_\phi) = \mathbb{R}$ is guaranteed to be the entire range of real numbers. This has been tried as well. However, a non-linear activation function in the last layer causes additional complications and reduces the training performance due to the costlier calculations of gradients.

6.2 Hyperparameter configuration

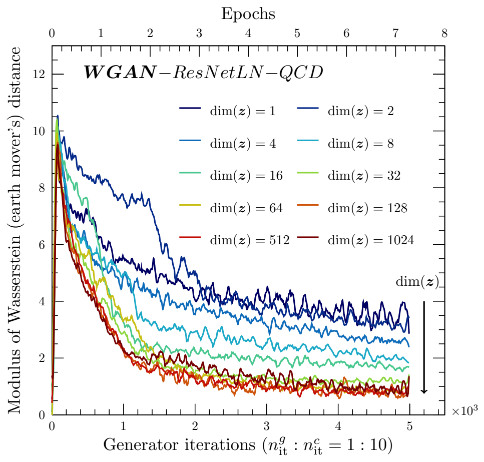
After the network architecture of the critic and the generator has been explained, it is once again about time to justify the configuration of hyperparameters that was used to train the adversarial neural network in this report. Doing this is crucial because Wasserstein GANs – more specifically: GANs in general – are *very* sensitive to the setting of hyperparameters compared to, for instance, variational autoencoders that are more robust in this regard. Hence, an inopportune configuration of the model's tunable parameters might quickly result in chaotic behaviour and non-convergence. In fact, the results shown here represent the fundamental prerequisite for the outcomes and insights of the sections to follow. (Admittedly, it is an unpleasant work and certainly not the most interesting part, but it's certainly worth the effort.) Furthermore, the findings acquired in this section may serve others who work with generative adversarial networks as a decent starting point for their model optimization. In spite of every effort, the tuning and optimization of the model are still considered to be rather superficial due to the large dimensionality of the hyperparameter space. There are promising approaches towards an automated hyperparameter tuning besides performing a random grid search, like Bayesian optimization methods or the utilization of Gaussian processes. However, all those novel methods currently require small networks with rather simple cost functions. Hence, none of these methods is suited to be used in combination with generative models for the time being.

In order to adequately estimate the performance of the neural network, a *figure of merit* is required that reflects the improvement or deterioration for different configurations of hyperparameters. In case of variational autoencoders, the loss function, i.e., the ELBO (see Equation 3.39) unequivocally is the quantity of choice since it directly gives a lower bound on the object of interest, i.e., the evidence of the data $P_r(\boldsymbol{x})$. Generally, however, there is no easy way to measure the performance of (classical) generative adversarial networks regarding the *quality* and the *entropy* (diversity) of the generated distribution $\mathbb{P}_g$. This is due to the "lack of interpretability" of the objective function (see Section 3.6.1) since an increasing loss may correspond to an improved performance or vice versa. This makes it very difficult to systematically rate the effectiveness of the model or to define some criterion of early stopping for the training routine. Therefore, it is established common practice in many publications to just present visual results[2]. Fortunately, the situation is different in case of Wasserstein GANs. As it has been explained in Section 3.6.2, as one representative on an integral probability metric the Wasserstein distance gives a lower bound on the Kullback-Leibler divergence $D_{\mathrm{KL}}(\mathbb{P}_r||\mathbb{P}_g)$ between the real data $\mathbb{P}_r$ and the generated distribution $\mathbb{P}_g$ (see Equation 3.66) similar to the lower bound on $P_r(\boldsymbol{x})$ in case of VAEs. Hence, a smaller value of $W(\mathbb{P}_r, \mathbb{P}_g)$ corresponds to an improved approximation of the real

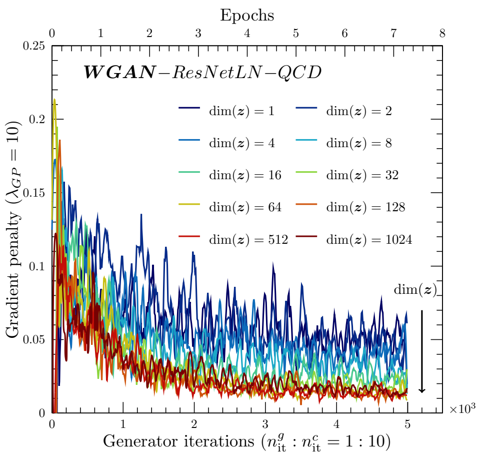
[2]A good figure of merit or "GAN evaluation measure" should measure the *quality* of the simulated samples as well as their *diversity* of the generated distribution, i.e., the entropy of the underlying probability distribution. To meet both criteria, an additional neural network is required: the so-called image classification network or *inception classifier* (that is provided by Google). The task of the inception network is to measure $p(\boldsymbol{y}|\boldsymbol{x})$, i.e., to classify the generated data $\boldsymbol{x} \sim \mathbb{P}_g$ to some label $\boldsymbol{y}$, for instance, the p_{T} of the jet. This measures the quality of the generated data. To estimate the diversity of the data, the entropy of the generated distribution $\mathbb{P}_g$ must be calculated according to the marginalization $\int_z \mathrm{d}\boldsymbol{z}'\, p(\boldsymbol{y}|g_\theta(\boldsymbol{z}'))$ that should be close to the true distribution $p(\boldsymbol{y})$. Both criteria combined give the so-called inception score $\mathrm{IS}(\mathbb{P}_g) = \exp\left(\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_g}[D_{\mathrm{KL}}(p(\boldsymbol{y}|\boldsymbol{x})||p(\boldsymbol{y}))]\right)$ [Barratt and Sharma, 2018, Brock *et al.*, 2018, Salimans *et al.*, 2016b]. Unfortunately, the inception score can not be used in the context of his thesis since Google's classification network has not been trained to classify jet images.

distribution. This makes perfect sense, as the integral probability metric is related to the maximum mean discrepancy by $\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \sup_{f \in \mathcal{H}, \|f\|_{\mathcal{H}} \leq 1} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[\phi(\boldsymbol{x})]$ and hence directly measures the similarity between two distributions with respect to their statistical moments. This is not necessarily true for GANs that use an f-divergence instead. Therefore, just as it was done for variational autoencoders, the *approximation* of the earth mover's distance by the critic network is used as a figure of merit to estimate the performance of the model. Furthermore, the gradient penalty term, $\mathcal{L} \supset \mathbb{E}_{\boldsymbol{x}' \sim \mathbb{P}_{\boldsymbol{x}'}} \left[\left(\|\nabla_{\boldsymbol{x}'} f_\phi\|_2 - 1 \right)^2 \right]$, is used to measure the quality of the required Lipschitz condition and thereby the reliability of the estimation of the earth mover's distance.

The first step is to specify the dimensionality of the latent space that serves as a random seed from which the jet-images are generated. For this purpose, the Wasserstein loss and the gradient penalty term are evaluated for different dimensions of the latent space $\mathcal{Z}$ for $\dim(\mathcal{Z}) \in \{2^k\}_{k \leq 9}$. The results are illustrated in Figure 6.2.



(a) Wasserstein loss

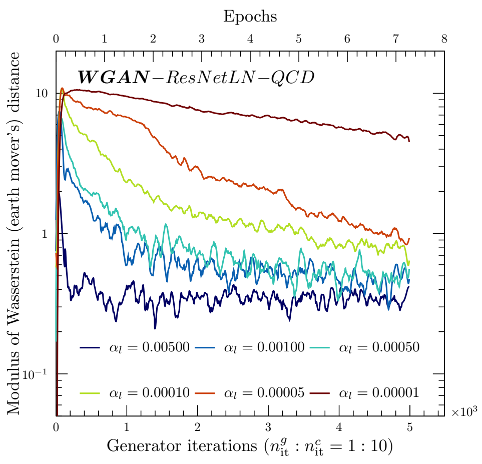


(b) Gradient penalty contribution

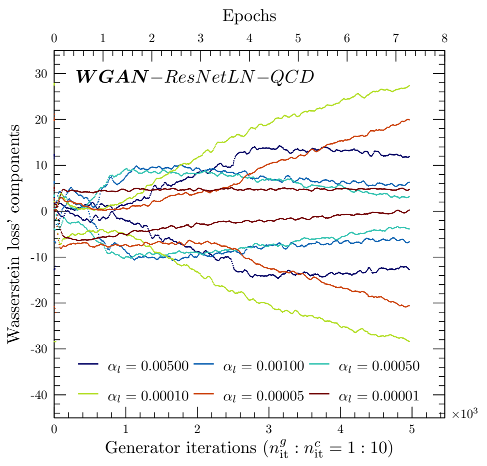
Plot 6.2: The non-negative Wasserstein loss (6.2a) (earth mover's distance) and the gradient penalty term (6.2b) for different dimensionality $\dim(\mathcal{Z})$ of the latent space $\mathcal{Z}$ for the same architecture.

Both family of curves in Plot 6.2a and 6.2b show some saturation from $\dim(\mathcal{Z}) \approx 64$ onwards, with the loss still decreasing slowly. It is also noteworthy that the variance of the contribution from the gradient penalty is significantly reduced for increased dimensions of $\mathcal{Z}$. Since it is hard to tell whether $\dim(\mathcal{Z}) = 32$ or $\dim(\mathcal{Z}) = 128$ is the appropriate choice, a dimension of $\dim(\mathcal{Z}) = 100$ has been assigned to $\mathcal{Z}$, which turned out to provide reasonable results. It is important to keep in mind the improvements in Figure 6.2 are not exclusively due to the larger dimension of $\mathcal{Z}$ since increasing $\dim(\mathcal{Z})$ always is accompanied by an increased complexity of the network regarding the number of trainable parameters (although this effect is small compared to the total number of weights in the network $N^{\text{tot}}_{g,w} \sim \mathcal{O}(10^6)$). This is obvious, since the noise vector $\boldsymbol{z} \in \mathcal{Z}$ is processed by fully connected layers in the first layer of the generator. Hence, the number of weights in the first layer of the generator scales linearly with the dimension of $\mathcal{Z}$.

After the dimensionality of the latent space $\dim(\mathcal{Z}) = 100$ has been fixed, the next step is to determine the learning rate α_l that weights the updates to the parameters in the gradient descent algorithm. At this point there is an additional difficulty compared to VAE: in VAEs, the decoder and the encoder network are trained *simultaneously*, while in GANs, the discriminator/critic and the generator are trained *alternatingly*. Therefore, a priori, there is no argument why the learning rate of the critic α_l^c the learning rate of the generator α_l^g should be identical; on the contrary, there are reasonable arguments why this should indeed not be the case. However, evaluateing the model on a grid of $[\alpha_l^{c,\min}, \alpha_l^{c,\max}] \times [\alpha_l^{g,\min}, \alpha_l^{g,\max}]$ is unfeasible in light of the complexity of the used model (the generation of the plots in Figure 6.3 already took several days), which is why the constraint $\alpha_l := \alpha_l^c = \alpha_l^g$ is applied in all models presented in this report. Figure 6.3a shows the approximation of the earth mover's distance for different learning rates.



(a) Wasserstein loss

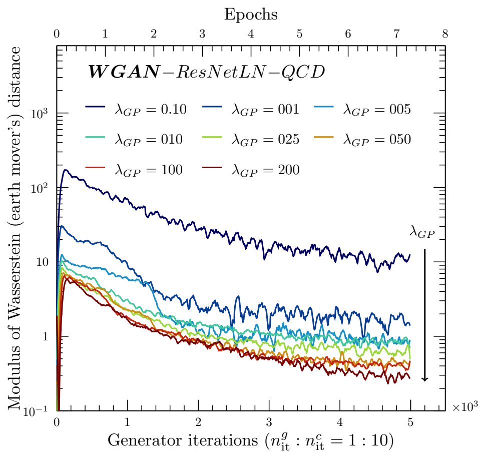


(b) Wasserstein loss' components

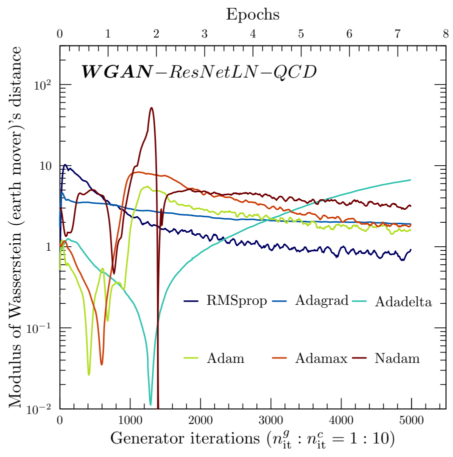
Plot 6.3: The non-negative Wasserstein loss (6.3a) and its components (6.3b) according to $\sup_{\|f_\phi\|_L \leq 1} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))]$ for different learning rates α_l with the constraint $\alpha_l := \alpha_c = \alpha_g$.

Judging by the graphs shown in Plot 6.3a, all learning rates except $\alpha_l = 0.005$ result in a converging model – obviously, the convergence is slower for smaller learning rates (in case of $\alpha_l = 0.005$, the model quickly collapsed). To facilitate the decision regarding the learning rate to use for training the model, consider Figure of 6.3b that shows the individual terms of the earth mover's distance without gradient penalty term $W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f_\phi\|_L \leq 1} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))]$. Plot 6.3b reveals some interesting behavior: the difference between the *absolute value* of $\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f_\phi(\boldsymbol{x})]$ and $\mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))]$ is the largest for $\alpha_l = 0.0001$. This is generally desirable because it means that the critic improves in order to assign a score to the respective samples. Furthermore, the variance for $\alpha_l = 0.0001$ is smaller than for $\alpha_l > 0.0001$, which positively affects the stability of the training routine. Based on those examinations, the learning rate for the critic and the generator was chosen to be $\alpha_l = 0.0001$, which corresponds roughly to the default value that is commonly used to train generative models. The gradient penalty contribution has not been evaluated for different learning rates since the respective term in the loss function experiences a different effective learning rate that also depends on λ_{GP} via $\alpha_l^{\text{GP}} = \alpha_l \lambda_{\text{GP}}$.

The next step is therefore to investigate the effect of the penalty coefficient λ_{GP} on the loss function, which is shown in Figure 6.4a.



(a) Wasserstein loss



(b) Wasserstein loss

Plot 6.4: The non-negative Wasserstein loss for different "penalty factors" λ_{GP} of the gradient penalty term (6.4a) and for different optimizers (weights' update rule) used in the gradient descent algorithm (6.4b).

As one can see, the penalty coefficient λ_{GP} is of great influence and has a strong impact on the curve progression. This is expected since it basically controls the quality of the Lipschitz approximation required by the Kantorovich-Rubinstein duality (see Section 3.6.2) and thus the validity of the critic's approximation of the earth mover's distance. However, if λ_{GP} is chosen too large, the gradient penalty term becomes too dominant in the optimization, with the consequence that the network only focuses on satisfying the constraint $\mathbb{E}_{\boldsymbol{x}'\sim\mathbb{P}_{\boldsymbol{x}'}}\left[\left(\|\nabla_{\boldsymbol{x}'} f_\phi\|_2 - 1\right)^2\right] \approx 0$, letting the earth mover's distance slide. This situation is, however, not present in Figure 6.4a. As indicated in the graph, the array of curves saturates roughly at $\lambda_{\mathrm{GP}} \approx 50$, which is why this value was henceforth used to train the neural network of the critic.

Last but not least, an optimizer, i.e., an algorithm that performs the update of the model's weights for each training step/iteration must be established. The same complication as in the case of the learning rate arises since both networks – critic and generator – could in principle get different optimizers. And, again, there are good reasons to use different optimizers for both networks due to the complicated gradient penalty term in the critic's loss that gives rise to all kind of problems in the optimization procedure. The modulus of the Wasserstein loss for different standard optimizers is summarized in Figure 6.4b. As one can see, the learning performance of the model is *highly* sensitive to the choice of the optimization algorithm. Especially noticeable is the glaring differences in the curve's progression of *RMSprop* and *ADADELTA* [Zeiler, 2012] (see Section 3.4.2). This is all the more surprising given the fact that both optimization algorithms have been invented to address the same problem, i.e., reduce the aggressive, monotonically decreasing learning rate of *ADAGRAD* [Ruder, 2016], i.e., an "'[a]daptive subgradient method[.] for online learning and stochastic optimization" (Duchi *et al.* [2011]) (this effect can be seen in Figure 6.4b: the Wasserstein loss decreases very slowly in case of ADAGRAD). So, both methods serve the same purpose, but behave very differently. In order to understand this phenomenon, it is recommended to closer examine the updating rule for the weights of the respective algorithm. In case of RMSprop, the update of the weights $w_i \to w_i + \alpha_l \delta w_i$ is given by $\delta w_i = -\frac{g_i}{\mathrm{RMS}[g]_i}$, whereby $\mathrm{RMS}[g]_i$ denotes the "root mean squared" error of the gradient g of the loss function $\mathcal{L}$ for the i^{th} iteration. The RMS in case of RMSprob is given by $\mathrm{RMS}[g]_i = \sqrt{\mathbb{E}[g^2]_i + \epsilon}$ with $\mathbb{E}[g^2]_i = 0.9 \cdot \mathbb{E}[g^2]_{i-1} + 0.1 \cdot \mathbb{E}[g^2]_i$, i.e., a moving average of the gradients $g (= \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}))$ and $\epsilon > 0$. ADADELTA's update rule for the weight is very similar. The correction to the weights for the i^{th} training step is $\delta w_i = -\frac{\mathrm{RMS}[\delta w]_{i-1}}{\mathrm{RMS}[g]_i} g_i$, whereby $\mathrm{RMS}[g]_i = \sqrt{\mathbb{E}[g^2]_i + \epsilon}$ with $\mathbb{E}[g^2]_i = \gamma \mathbb{E}[g^2]_{i-1} + (1-\gamma)\mathbb{E}[g^2]_i$. For the default value of $\gamma = 0.9$, RMSprop and ADADELTA do not differ in this regard. The difference is the RMS of the *previous weights* (the weights for the i^{th} iteration are unknown) $RMS[\delta w]_{i-1} = \sqrt{\mathbb{E}[\delta w^2]_{i-1} + \epsilon}$ with $\mathbb{E}[\delta w^2]_{i-1} = \gamma \mathbb{E}[\delta w^2]_{i-2} + (1-\gamma)\mathbb{E}[\delta w_{i-1}^2]$; so, $\delta w^{\text{ADADELTA}} / \delta w^{\text{RMSprop}} = \mathrm{RMS}[\delta w]_{i-1}$. Therefore, the problem that is present in Figure 6.4b must be caused by the RMS of the previous weights or, more generally, by the inclusion of many weighted previous model configurations into the current state of the network through the moving average of the weights. Something similar occurs in case of Adam and its variations – though Adam does not average over the model's weights directly but computes the moving average of the previous gradients. The incorporation of previous weights into the calculation of the new state is problematic with respect to the gradient penalty term since it creates correlations between different model configurations. This spoils some basic assumptions in the computation of the gradients with respect to the sample interpolation $\hat{\boldsymbol{x}} = \boldsymbol{x}_r \epsilon + (1-\epsilon)\boldsymbol{x}_g$ (see Section 3.6.2). This is also the reason why batch normalization (see Section 3.4.4) can *not* be used in combination with Wasserstein GANs that uses gradient penalty to enforce the Lipschitz constraint because it creates correlation between the data over several batches. Based on those insights,, RMSprop – besides plain SGD and ADAGRAD that both converge too slowly – is the only appropriate choice. It is worth underlining again that all the results shown in Plot 6.4b are obtained for the *same* optimization algorithm *for both* networks (critic and generator). However, the aforementioned problems regarding the induced correlations between different states of the model only concerns the critic; the generator does not come with a gradient penalty term (see objection function of the generator according to Equation 3.67).

To summarize: the generative adversarial neural network (more precisely: the generative model) uses a latent space with dimensionality $\dim(\mathscr{Z}) = 100$ (6.2); both networks are trained with the same learning rate $\alpha_l = 0.0001$ (6.3) and the same optimizer: RMSprop (6.4b); finally, the gradient penalty factor is $\lambda_{\text{GP}} = 50$ (6.4a). Now, it is time to proceed and to simulate some QCD radiation.

6.3 Unconditioned WGAN – another attempt

As observed in the previous chapter of this report, the *Gaussian* variational autoencoder (see Section 3.5) is "incapable" of learning the information of the underlying subprocess, i.e, the matrix element information and the parton shower simulation in one model. This means that the model is not able to correctly reproduce the probability distributions given by the matrix element such as the energy of the "final state" particles. The exact underlying cause of those problems are rather difficult to identify (from a theoretical perspective); however, based on the empirical evidence obtained in Chapter 5, there is little to no doubt that VAEs are unsuited for this task. Only if the learning task is factorized into matrix element (provided as a piece of external information) and parton shower, the model can provide (more or less) reasonable results. These are the events so far.

However, this chapter provides a new chance for the aforementioned learning task by using Wasserstein generative adversarial networks instead. As it has been explained in

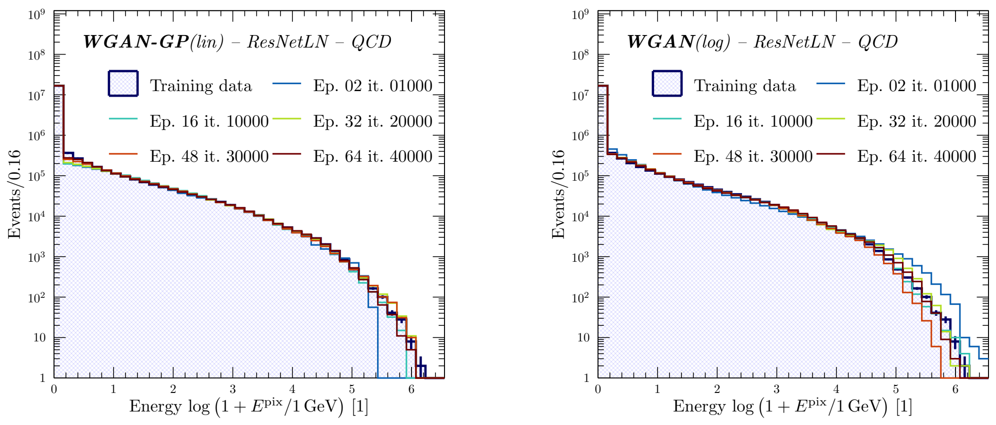
Section 3.6.2, Wasserstein GANs optimize an *integral probability metric* instead of the log-likelihood of the data. Therefore, both methods differ significantly regarding their implementation, as well as their fundamental underlying principles. Due to those essential differences between both approaches to generative models, it is *not* possible to draw any premature conclusion for the Wasserstein GANs based on the result obtained in case of VAEs. The procedure must be repeated from the beginning.

An additional note regarding the characteristics of this section. As it turned out, the usage of Wasserstein GANs will result in a *significant* improvement compared to Gaussian VAEs for unconditioned models as will be shown in the next section below. However, the focus of this thesis is on the simulation of parton showers and hence on models that are conditioned on the jet energy and pseudorapidity. For this reason, the considerable improvements observed in this section are presented mostly uncommented. As it turns out, the results further improve in the subsequent section that studies the conditioned case. Therefore, most explanations and/or interpretations regarding the reasons behind the improvement are postponed to the next section 6.4.

6.3.1 Linear versus logarithmic scale

In case of Wasserstein GANs, a strong difference in the performance was observed depending on whether a linear or logarithmic energy scale was used for the pixel activation values in the image (see “invertible preprocessing” 4.3). Both scales result in converging models – contrary to the Gaussian VAE where the section logarithmic scale spoiled the probabilistic assumptions regarding the Gaussian distribution of reconstruction errors. This, however, is not the case for Wasserstein GANs or GANs in general.

To study the effect of a linear $E^{\text{pix}} \mapsto \varrho E^{\text{pix}}$ scale, with $\varrho \in \mathbb{R}$, and a logarithmic scale $E^{\text{pix}} \mapsto \log\left(1 + \varrho E^{\text{pix}}\right)$, the distribution of pixel activations E^{pix} is evaluated for two generative models that have been trained on both scales respectively (for reasons of clarity and brevity, only QCD jets are shown. However, it should be noted that all results also apply for the W training set). The results for the different scales are summarized in Figure 6.5.



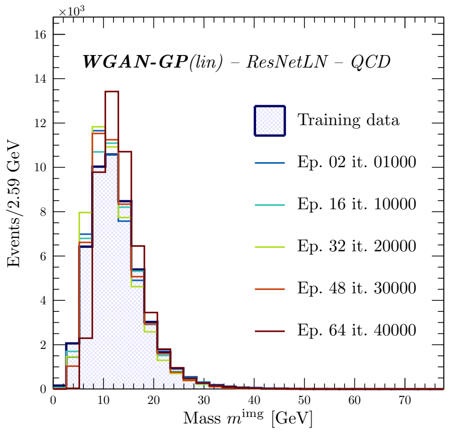
(a) Linear scale

(b) Logarithmic scale

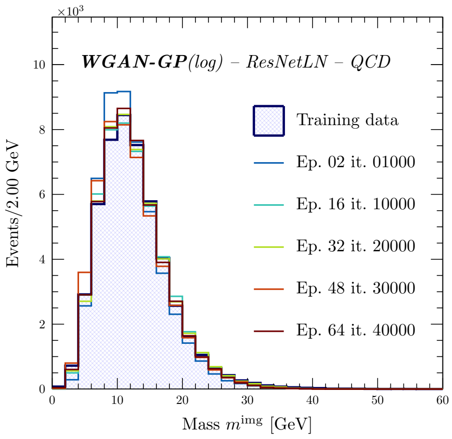
Plot 6.5: Pixel activation values E^{pix} on an event-on-event base for $50,000$ events.

At first glance, both networks provide a descent description of the energy spectra of the pixel values. In case of a linear scale (Figure 6.5a), however, there are a *drastic deviations* from the expected curve at low energies. This difference might appear small, but it is actually considerable due to the logarithmic scale of the ordinate axis that hides discrepancies. In case of a logarithmic scale (Figure 6.5b), the situation has improved significantly, and the discrepancy has almost completely vanished. This effect is easily explained. In case of a linear scale, the neural network actually "sees" an energy spectrum as illustrated in Figure 4.1a. There is one pronounced peek at zero that "absorbs" most of the structure. In case of a logarithmic scale on the other hand, more structure is resolved, whereby the sensitivity to low-energy values can be controlled by ϱ (in this thesis a $\varrho = 0.1$ for lin- and $\varrho = 1$ for log-scale was used).

The effect is even more pronounced in case of the reconstructed invariant mass of the jet (again only for QCD jets).



(a) Linear scale

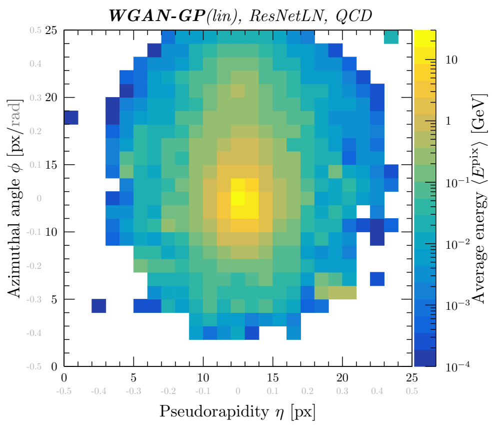


(b) Logarithmic scale

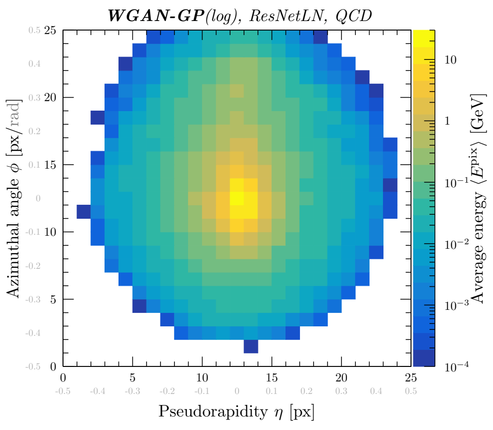
Plot 6.6: Reconstructed jet mass m^{img} for a linear (6.6a) and a logarithmic scale (6.6b) for 50,000 events.

The difference between the two mass spectra in Figure 6.6 nicely illustrates the beneficial impact of the logarithmic scale on other jet observables. This is expected too since the increased sensitivity to low energy contributions in the image is automatically accompanied by an increased "awareness" of their position, i.e., the relative position of active pixels in the image, which is an essential information used by many observables.

Finally, the same beneficial effect can be observed for the average jet image as can be seen in Plot 6.7a and 6.10a.



(a) Linear scale



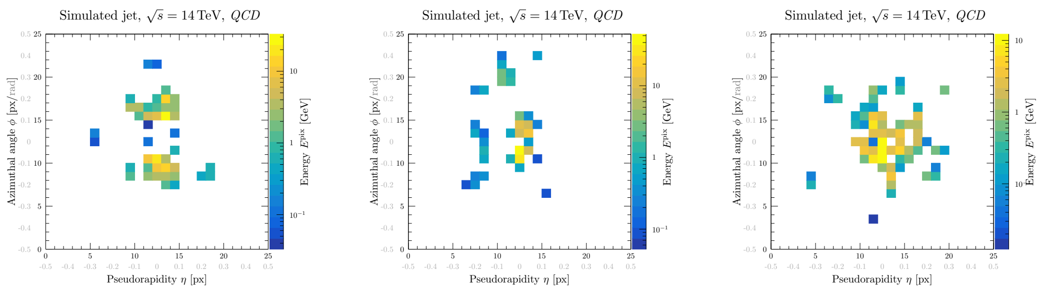
(b) Logarithmic scale

Plot 6.7: Average jet image for a linear (6.7a) and a logarithmic scale (6.7b).

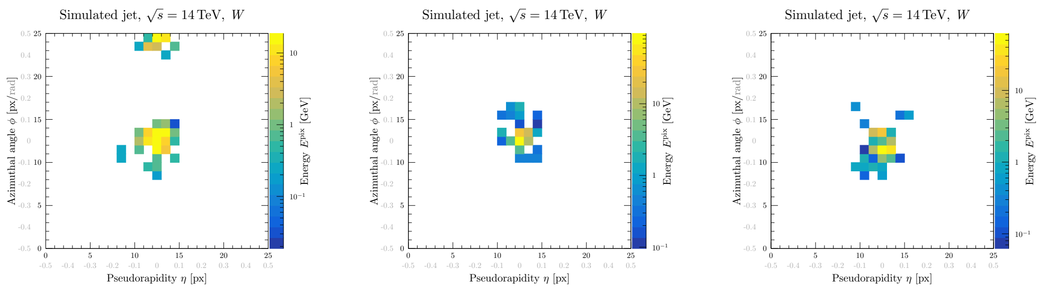
Figure 6.7 once more illustrates the effect of the logarithmic energy scale. The structure of the jet much more resembles the true image according to Figure 4.2. Based on those observations, the decision was made in favor of a logarithmic scale that is used for all Wasserstein GANs in the following sections if not specifically mentioned otherwise.

6.3.2 Samples and average jet images

As it has been done in the previous chapter, the first step is a visual comparison of the generated data for QCD and W jets for a generative model that has been trained for 40,000 iterations.

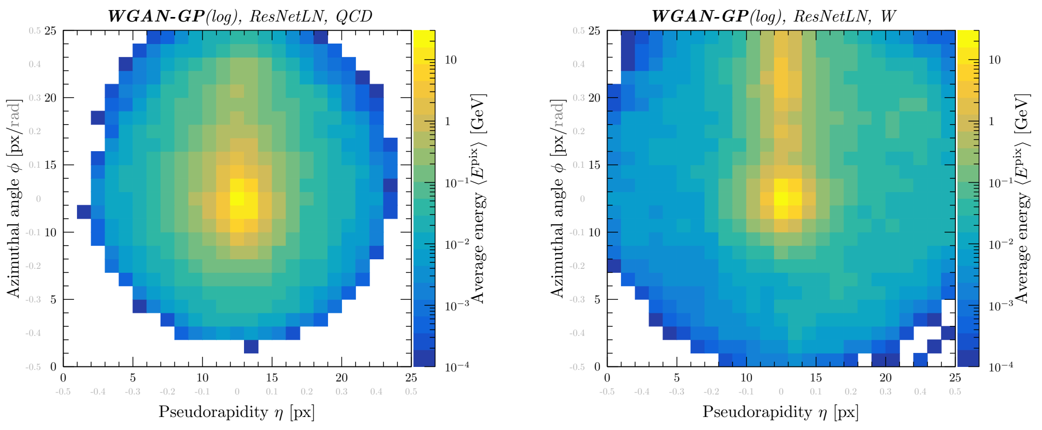


Plot 6.8: Three randomly simulated QCD jets after 40,000 iterations.



Plot 6.9: Three randomly simulated W jets after 40,000 iterations.

The results displayed above show a significant improvement compared to the *unconditioned* variational autoencoders in Figure 5.3 and 5.4. Apparently, the occupancy of the images is well modelled (compared to the VAE) and equally so the activation of the individual pixels. The same is observed for the average jet images in the Figure below.



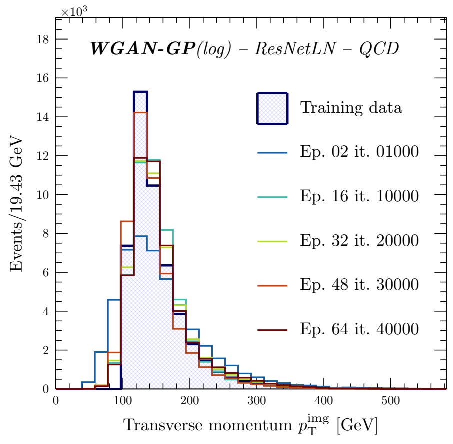
(a) QCD *(b)* *W*

Plot 6.10: Average jet image for $50,000$ events and $40,000$ iterations.

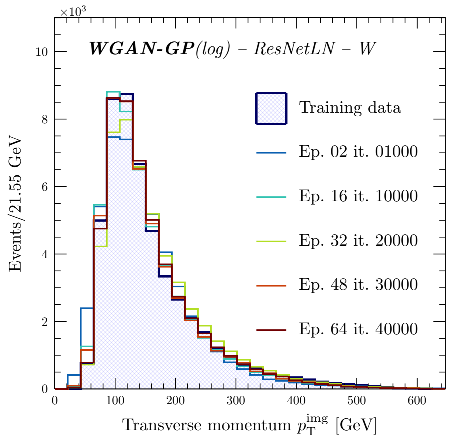
The generated average jet images almost perfectly resemble the true ones shown in Figure 4.2 and 4.4 (note: there is no p_{T}-cut involved in Figure 6.10b).

6.3.3 Kinematic distributions

The following Figure 6.11 shows the reconstructed spectrum of the jet's transverse momentum $p_{\mathrm{T}}^{\mathrm{img}}$. Since the model is unconditioned, the energy or the transverse momentum distribution (which is a piece of information provided by the matrix element) must be entirely learned by the neural network. As it was shown in Section 5.3, the unconditioned Gaussian variation autoencoder failed to distil this underlying information from the training set, i.e., the jet images. In the case of unconditioned Wasserstein GANs, the model is able to learn both the underlying matrix element and the parton shower information as can be seen in Figure 6.11.



(a) QCD

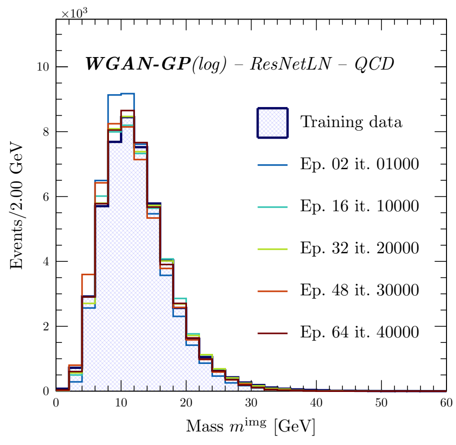


(b) W

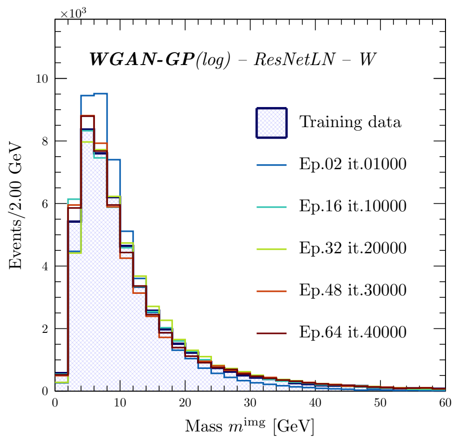
Plot 6.11: Reconstructed transverse momentum p_T^{img} for $50,000$ events.

The improvement compared to the corresponding Figure 5.8 for the Gaussian VAE is remarkable. It can thus be concluded that the generative model indeed has learned the underlying information of the matrix element as well as the parton shower.

The same holds true for the reconstructed invariant mass of the jet, which is shown in Figure 6.12.



(a) QCD



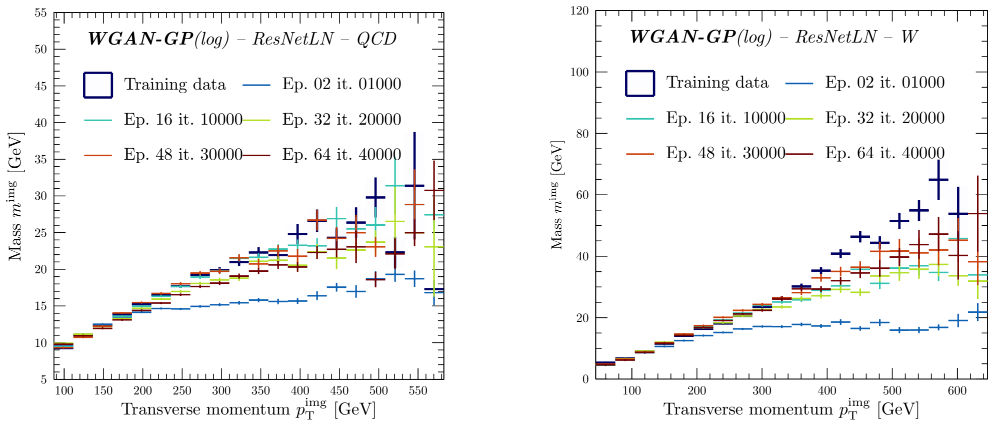
(b) W

Plot 6.12: Reconstructed jet mass m^{img} for $50,000$ events.

The corresponding distributions for the Gaussian VAE are illustrated in Figure 5.9. The explanations for this significant degree of improvement is given in the subsequent section in the context of conditioned Wasserstein GANs. Without giving too much away, the improvement is closely related to the fundamental connection between the MMD and the EMD.

6.3.4 Other jet observables

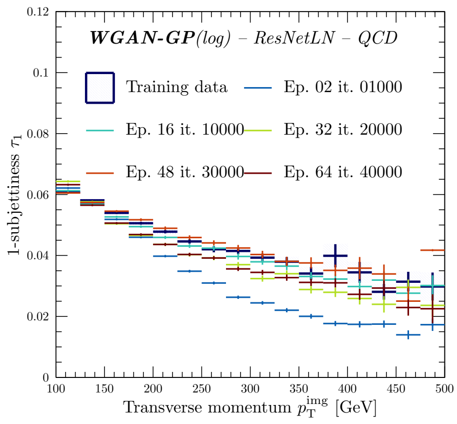
The previous results already demonstrated a significant improvement. Apparently, the generative adversarial network can provide a much better approximation $\mathbb{P}_g$ of the underlying probability distribution of the training data $\mathbb{P}_r$. To verify this first impression on a deeper level, the following plots show the correlation between different jet-observables as well as their description in different regions of phase space.



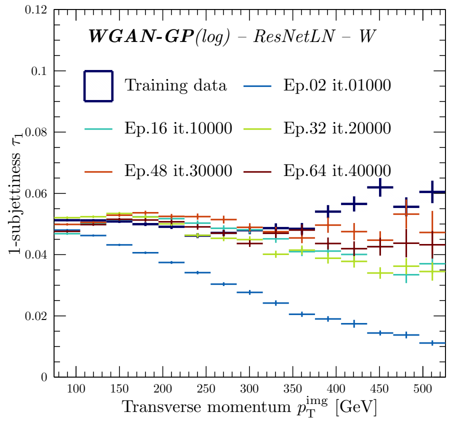
(a) QCD *(b)* W

Plot 6.13: Correlation between the reconstructed average jet mass m^{img} and the transverse momentum $p_{\text{T}}^{\text{img}}$ for $50,000$ events.

As it was shown in Figure 6.11 and 6.12, the spectrum of the transverse momentum and the invariant mass is well described for both processes. The same holds true in case of the average invariant mass of the jet in different $p_{\text{T}}^{\text{img}}$ regions. The situation has significantly improved compared to the corresponding Figure 5.13 for VAEs. In the case of unconditioned Gaussian VAEs, the investigation was terminated at this point due to the poor modelling of the training data. However, since the Wasserstein GAN provides excellent results, it is worthwhile to study jet observables that also probe the substructure of the jet such as, for instance, N-subjettiness. The following figure therefore shows the reconstructed 1-subjettiness for different regions of the transverse momentum of the jet.



(a) QCD

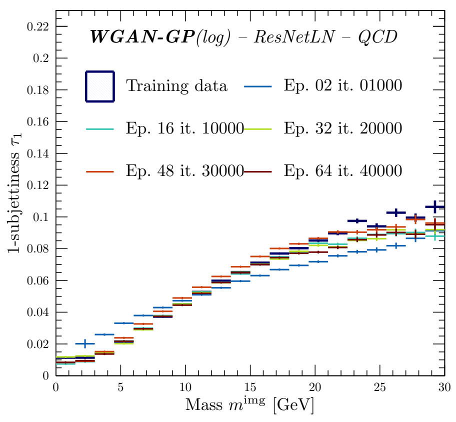


(b) W

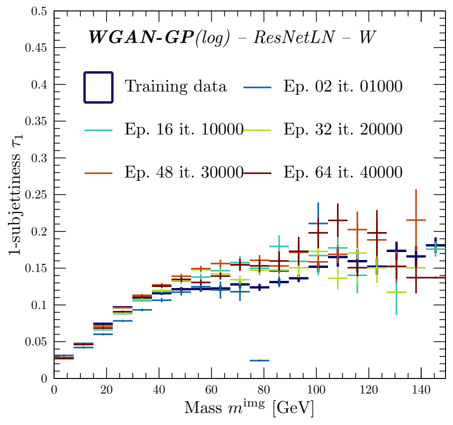
Plot 6.14: Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the transverse momentum $p_{\text{T}}^{\text{img}}$ for $50,000$ events.

As it can be seen in Figure 6.14, the graphs that have been reconstructed from the generated distribution provide a very good description of the training data. Furthermore, the strong correlation between the number of iterations (updates of the generator's weights with $n_{\text{it}}^c : n_{\text{it}}^g = 1 : 10$) and the goodness of the reconstruction can be observed.

This also applies to the correlation between the 1-subjettiness and the invariant mass of the reconstructed jet, which is shown in Figure 6.15.



(a) QCD



(b) W

Plot 6.15: Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the mass m^{img} for $50,000$ events.

As expected, the 1-subjettiness τ_1 increases with larger jet masses. In the case of W jets, some deviations from the expected curve characteristics can be observed.

This section could easily be extended by many more figures of merit and plots that all together convey the same message, i.e., the excellent agreement between the generated distribution and the training data. The results achieved with unconditioned Wasserstein GANs are indeed very satisfactory. Contrary VAE, WGANs allow to model the matrix element as well as the parton shower.

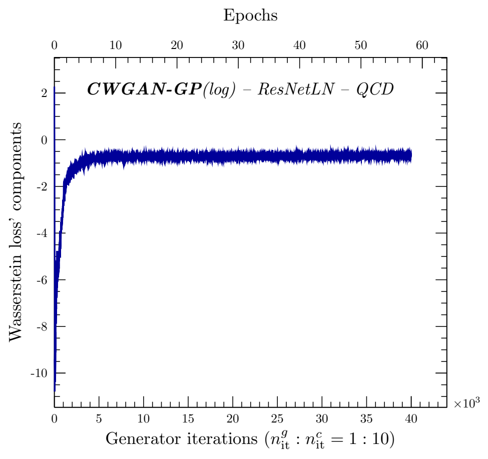
6.4 Conditional WGAN

This section introduces the generative model that – out of all networks presented in this report – showed the best performance and provided the finest results (albeit very close to the ones obtained in the previous Section 6.3). The model under consideration is the Wasserstein generative adversarial network as it was introduced previously but *conditioned* on the energy E^{jet} and the pseudorapidity η^{jet} of the jet (following the footsteps of the variational autoencoder in Chapter 5). Taking into account the positive effect of the factorization of the generative model into the matrix element and the parton shower simulation for the VAE, the conditional WGAN is expected to benefit from this step as well. Like it was done in the previous chapter, the distributions of conditioning energy $\mathbb{P}_E$ and pseudorapidity $\mathbb{P}_\eta$ are not learned by a generative model,but directly approximated through the training data that correctly accounts for the correlation between the two labels.

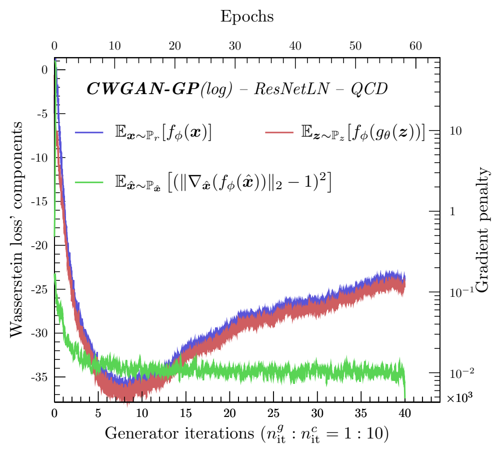
6.4.1 Training and model convergence

It is important to closely monitor the training routine of the networks to correctly estimate the performance of the generative model. As noted previously, the earth mover's distance (as an integral probability metric) provides a good figure of merit not only to estimate the performance of the model, but to monitor the training progress as well. If the Wasserstein loss $W(\mathbb{P}_r, \mathbb{P}_g)_i < W(\mathbb{P}_r, \mathbb{P}_g)_{i-1}$ for iteration i has reduced compared to the previous step $i-1$, it is guaranteed that the approximation of $\mathbb{P}_r$ by means of $\mathbb{P}_g = g_\theta(\mathbb{P}_z)$ has improved (see MMD and Equation 3.66 and lower bound 3.66). Therefore, the Wasserstein loss does not only allow to directly follow the progression of the model, but it also allows to define a "stopping rule" of when to terminate the training of both models. This is an incredible advantage compared to classical GANs (see Section 3.6.1) and enables one to systematically work with generative adversarial models. However, the number itself, i.e., the actual value of the Wasserstein loss is much harder to interpret – the smaller, the better.

The Wasserstein loss (6.16a) and its components (6.16b) for the model trained on QCD samples are shown in Figure 6.16.



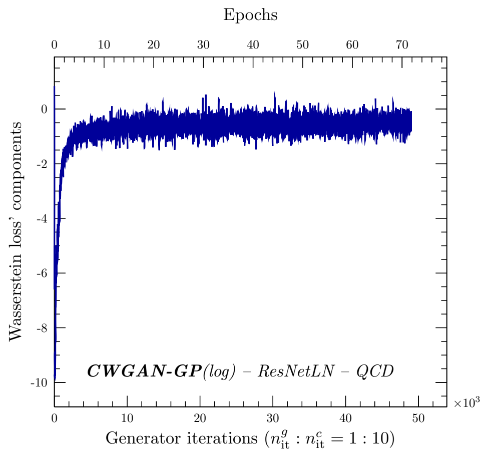
(a) Wasserstein loss



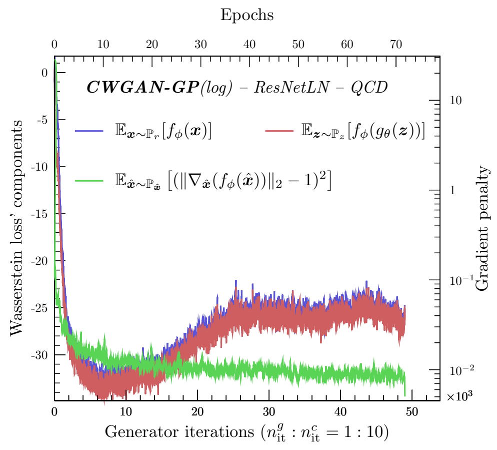
(b) Wasserstein loss' components

Plot 6.16: The Wasserstein loss (6.16a) and its components (6.16b) for QCD jets.

As one can seen in Figure 6.16a, the model converges since the curve it is monotonically decreasing (the small fluctuations are a consequence of mini-batch training (see Section 3.4.2)). The model is still improving after 40,000 iterations; however, the training has been terminated due to a very small rate of changes. The respective stopping criterion was defined as $\left|\frac{\text{MA}[W(\mathbb{P}_r,\mathbb{P}_g)]_i - \text{MA}[W(\mathbb{P}_r,\mathbb{P}_g)]_{i-N}}{\text{MA}[W(\mathbb{P}_r,\mathbb{P}_g)]_{i-N}}\right| < \delta^{\text{stop}} \approx 1‰$, whereby $\text{MA}(x)_i$ is a moving average over the first i samples with a higher weight for most recent iterations (exponential moving average). However, for reasons of comparability (and limited amount of time), all result shown in this report are up to 40,000 training iterations if not explicitly mentioned otherwise. The right Plot 6.16b in Figure 6.17 shows the components of the loss including the gradient penalty term normalized to λ_{GP} (see Equation 3.67). The curve(s) show the desired characteristic as it is expected from the optimization task according to Equation 3.64. The same accounts for the gradient penalty term, i.e., the Lipschitz condition $\mathbb{E}_{\boldsymbol{x}'\sim\mathbb{P}_{\boldsymbol{x}'}}\left[\left(\left\|\nabla_{\boldsymbol{x}'} f_{\phi^{40k}}\right\|_2 - 1\right)^2\right] \sim \mathcal{O}(1\,\%)$ is sufficiently met at the one-percent level. The very same is done for the model that has been trained on W samples instead.



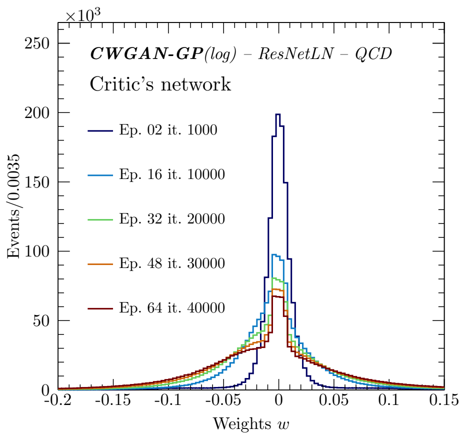
(a) Wasserstein loss



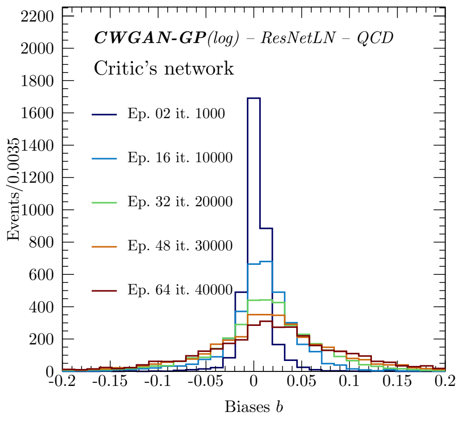
(b) Wasserstein loss' components

Plot 6.17: The Wasserstein loss (6.17a) and its components (6.17b) for W jets.

According to Figure 6.17, the results in case of W initialized jets are very similar to QCD's. The curve progression of the loss' components shown in Figure 6.17b are even closer to the optimum outcome since the components even out at roughly $\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f_\phi(\boldsymbol{x})] \approx \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))] \approx -27$, which means that both networks f_ϕ and g_θ are well balanced. Nonetheless, better networks should be able to archive larger values.



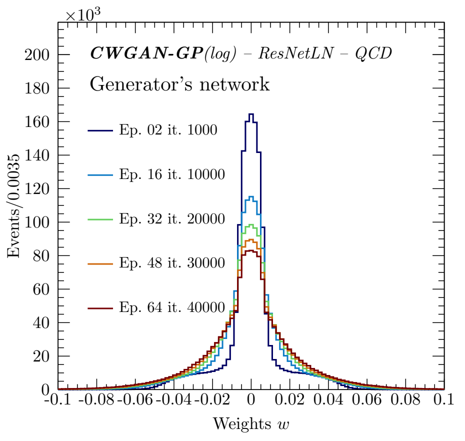
(a) Weights $w \in \mathcal{W} \subset \mathcal{P}$



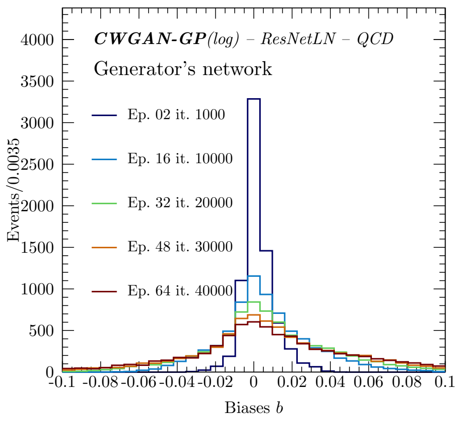
(b) Biases $b \in \mathcal{B} \subset \mathcal{P}$

Plot 6.18: The distribution of the critic's weights (without biases $\mathcal{W} = \mathcal{P} \setminus \mathcal{B}$) for different iterations (6.18a) and the distribution of biases (6.18b).

Besides of studying the loss and its components, it is revealing to look at the evolution of the distribution of the model's trainable parameters for different training iterations (configuration of the model of the generator network). For this purpose, the values of the respective parameters (weights) are filled separately into a histogram for the generator and the critic. Furthermore, the values for the weights and the biases (both of which are trainable model parameters) are presented separately due to the significant difference in their number in the network ($N_w^{c,g} \sim \mathcal{O}(10^6)$ and $N_b^{c,g} \sim \mathcal{O}(10^3)$).



(a) Weights $w \in \mathcal{W} \subset \mathcal{P}$



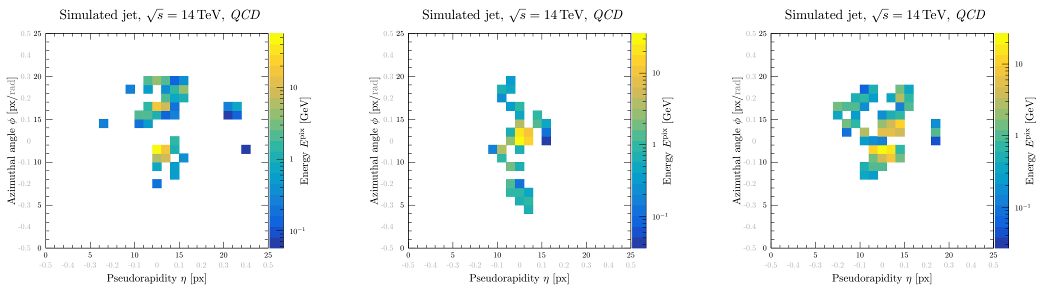
(b) Biases $b \in \mathcal{B} \subset \mathcal{P}$

Plot 6.19: The distribution of the generator's weights (without biases $\mathcal{W} = \mathcal{P} \setminus \mathcal{B}$) for different iterations (6.19a) and the distribution of biases (6.19b).

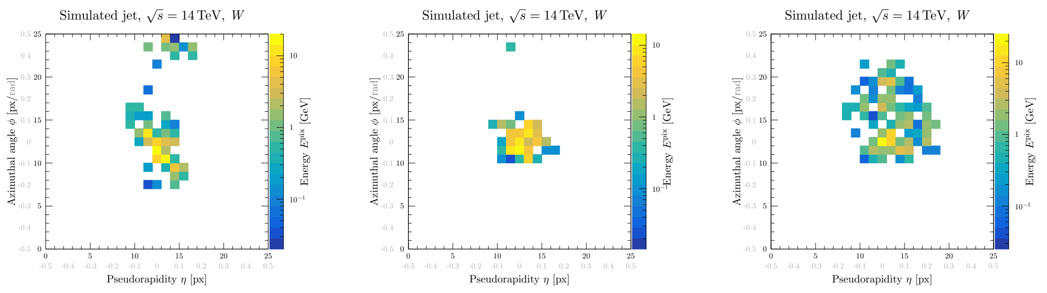
The distributions for the critic are shown in Figure 6.18. Similarly, Figure 6.19 shows the distribution of parameter values at different instances of the neural network trained on QCD samples (the respective distributions for the model trained on W samples are not shown since they mostly correspond to the one shown in Figure 6.18 and 6.19). The distribution of weights in case of the critic (Figure 6.18a) shows some interesting features since there is a significant change in the *shape* of the distribution compared to the early stage of the network; the target distribution becomes apparent. From the peak around zero it can be concluded that the network has not yet reached its optimum (compare the significant change from 30,000 to 40,000 iterations) and/or the model is too complex regarding its number of weights, with the network correcting for this fact by setting a large fraction of weights to zero in order to disable interconnections between nodes in the graph. In fact, this is precisely what residual network have been invented for (see Figure 5.2). The distributions of the generator's trainable parameters are less conspicuous. The "suction-bell-like" shape at the early stage of the network is a consequence of the initialization of the weights in the network that is a mixture of Gaussian and uniform priors – whether this initialization scheme is reasonable or not is certainly a worthwhile discussion.

6.4.2 Samples and average jet images

As previously done, the first step is to ensure that the generative model produces reasonable samples that visually resemble the training data. Figure 6.20 and 6.21 shows three randomly simulated jet images (random seed $\boldsymbol{z} \sim \mathbb{P}_z$ and labels $E^{\text{jet}} \sim \mathbb{P}_E$, $\eta^{\text{jet}} \sim \mathbb{P}_\eta$ given by the squared matrix element) for QCD and W initialized jets.

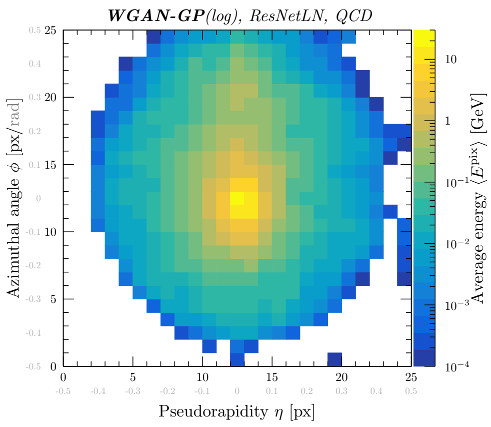


Plot 6.20: Three randomly simulated QCD jets after 40,000 iterations.

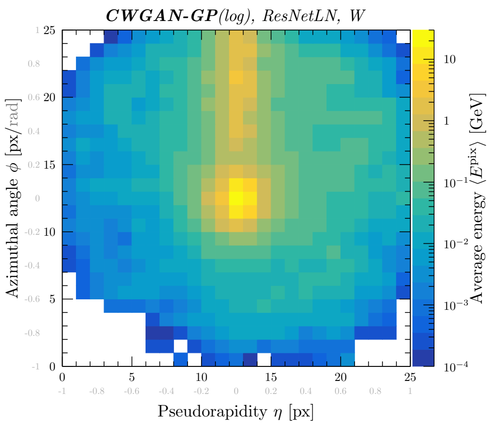


Plot 6.21: Three randomly simulated W jets after 40,000 iterations.

Like in the previous section, the results shown above vary significantly from the images that have been generated with the Gaussian variational autoencoders and its variations (*cf.* Figure 5.3, 5.4; 5.14, 5.15; and 5.32, 5.33). The simulated jet images are quite in accordance with the training data. Indeed, it can be stated that the generated data is *visually indistinguishable* compared with points sampled from the training set as in Figure 4.3 and 4.5. Of cause, it would be foolish to solely rely on the visual comparison of images on an event-on-event base. Therefore, the following two Plots 6.22a and 6.22b show once again the average jet image for QCD and W initialized jets.



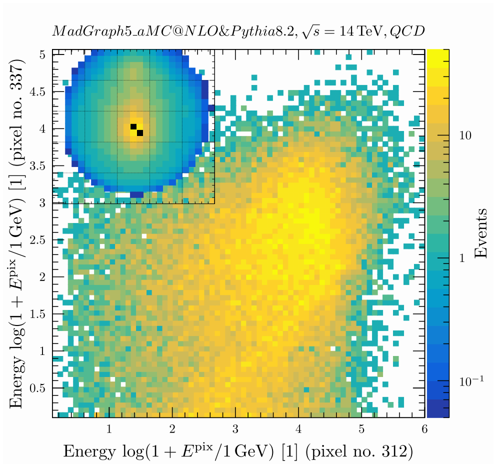
(a) QCD



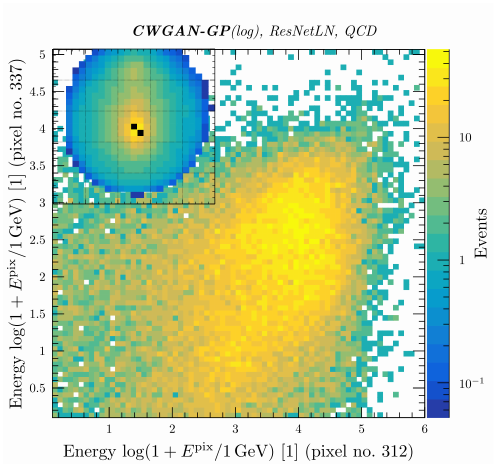
(b) W

Plot 6.22: Average jet image for 50, 000 events and 40, 000 iterations.

Also, the average images do look much closer to the training data (*cf.* Figure 4.2 and 4.4). Particularly in the case of the W jets in Figure 6.22b, the improvement compared to the VAEs is clearly visible. This can be seen (among other things) by the distinct two-peak structure that is much more pronounced than it was the case for VAEs. However, this result must be interpreted with caution as it was pointed out in the previous chapter since the average jet only gives the sample mean of the energy distributions in each pixel of the image. This does not mean that the respective distributions are learned correctly. On the contrary, it is easy to think of different distributions that have exactly the same mean value but completely different shapes. In total, there are $n_\eta^{\text{pix}\prime} \cdot n_\phi^{\text{pix}\prime}$ *individual* energy distribution that are highly correlated among each other. For the purpose of an example, Figure 6.23 shows the correlation between two selected pixels close to the center of the high-active image for the Monte Carlo based training data 6.23a and the generated distribution 6.23b for QCD.



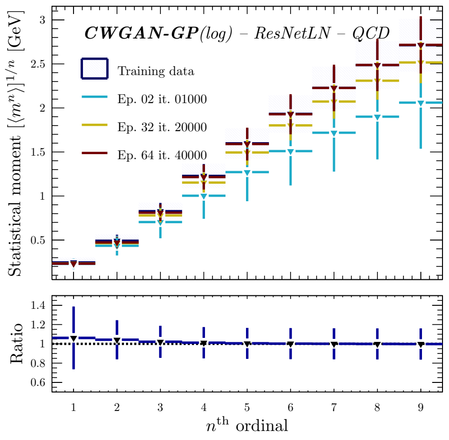
(a) Generated from training data



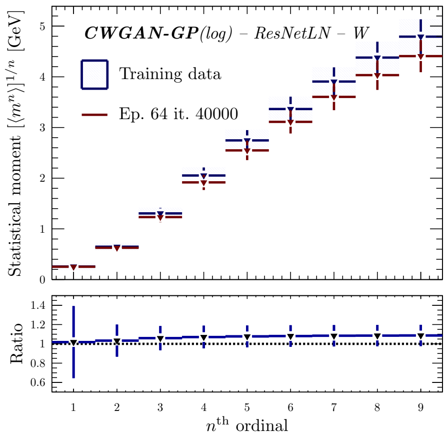
(b) Generated after $40,000$ iterations

Plot 6.23: "Energy correlation" between pixel no. 312 and pixel no. 337 for the training data (6.23a) and the generated distribution (6.23b). The correlated pixels have been highlighted in the subfigure in the left corner.

Figure 6.23 is only one possible correlation among a total number of $n_\eta^{\rm pix\prime} \cdot n_\phi^{\rm pix\prime} \cdot (n_\eta^{\rm pix\prime} \cdot n_\phi^{\rm pix\prime} - 1)(= 390,000)$ similar diagrams; it is therefore hardly representative but rather educational in its nature. Nonetheless, the plots do provide some valuable insights. As can be seen in Figure 6.23a, the correlation of the energy values in the individual pixels is non-trivial. The neural network does provide a good description of the distribution. However, compared to the real data, the sharp edges are smoothed out with the overall structure being rather well described.



(a) Statistical moments QCD

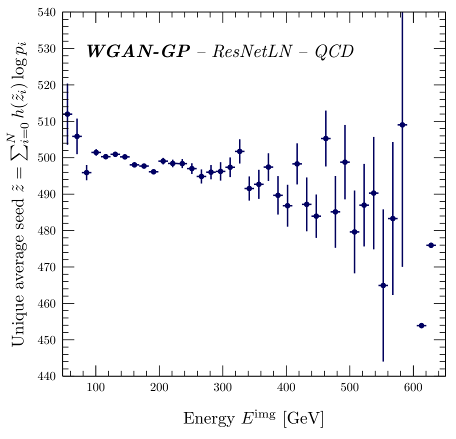


(b) Statistical moments W

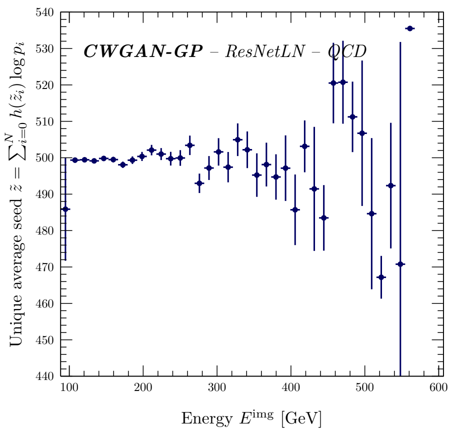
Plot 6.24: The first nine statistical moments according to Equation 5.3 for the training data and the generated distribution of the generative model. In case of QCD, the two distributions almost perfectly coincide, which is why other instances of the neural network (besides the one corresponding to $40,000$ iterations) have been included to illustrate the progression of the model.

A more reliable estimation of the similarity between the real and the generated distribution is provided by the direct comparison of the statistical moments according to Equation 5.3. The results for QCD (6.24a) and W jets (6.24b) are shown in Figure 6.24. After studying the similarity of the statistical moments of the two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$, there is no longer any doubt that the Wasserstein generative adversarial networks are superior to the Gaussian variational autoencoders in Chapter 5 (compare Figure 6.24 with Figure 5.7 and 5.24) regarding their description of the underlying distribution of the training data. This statement claims validity since any probability distribution $f_X(x)$ (for reasons of simplicity only defined on $\mathbb{R}$) for which the series of statistical moments $\mathbb{E}[X^n] = \int_{\mathbb{R}} \mathrm{d}x \, x^n f_X(x)$ is defined can be expanded in terms of the inverse Fourier transform $f_X(x) = \int_{\mathbb{R}} \mathrm{d}s \, e^{2i\pi xs} \left(\sum_{k=0}^{\infty} \frac{(-2i\pi s)^2}{n!} \mathbb{E}[X^n] \right)$. Hence, the probability distribution $f_X(x)$ can be expressed as an infinite series over the distribution's statistical moments $\mathbb{E}[X^n]$. This Furthermore means that two distributions $f(x)$ and $g(x)$ coincide if they agree on all statistical moments $f(x) - g(x) = \int_{\mathbb{R}} \mathrm{d}s \, e^{2i\pi xs} \left\{ \sum_{k=0}^{\infty} \frac{(-2i\pi s)^2}{k!} \left(\mathbb{E}[X^k]_f - \mathbb{E}[X^k]_g \right) \right\} = 0 \Rightarrow$ $\mathbb{E}[X^k]_f = \mathbb{E}[X^k]_g$ for all orders $k \in \mathbb{N}$. This property – two distributions being identical if they agree on all moments – is used by the aforementioned generative moment matching networks [Li *et al.*, 2017] and GANs based on the (kernel) maximum mean discrepancy (see Section 3.6.3). The MMD between the two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$ was given by $\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \left\| \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[\phi(\boldsymbol{x})] \right\|_{\mathcal{H}}$ with $\phi : \mathcal{X} \to \mathcal{H}$ being a feature map and $\mathcal{H}$ denoting the so-called reproducing kernel Hilbert space. For the most simple example possible with $\mathcal{X} = \mathbb{H} = \mathbb{R}^N$ and $\phi = \mathrm{id}$, the MMD is just given by the Euclidean distance between the mean of the respective distributions $\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \left\| \mu_{\mathbb{P}_r} - \mu_{\mathbb{P}_g} \right\|$. This quantity is simple and was also well described in case of VAEs. However, the MMD is much stronger if ϕ maps to a general space $\mathcal{H}$, which allows its application to the so-called *kernel trick* that leads to a MMD that is zero iff the distributions under considerations are completely identical. As described in Section 3.6.2 of Chapter 3, the Wasserstein distance as a representative of an integral probability metric, is closely related to the MMD as is directly apparent from the alternative representation $\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \sup_{f \in \mathcal{H}, \|f\|_{\mathcal{H}} \leq 1} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[\phi(\boldsymbol{x})]$. Due to this relationship, the Wasserstein loss will learn the underlying distribution of the training data by "fitting" the statistical moments, which explains the astonishing accordance in Figure 6.24 (deviations can be observed for much higher moments). The mechanism described above differs fundamentally from the operating principle of Gaussian VAEs (or VAEs in general) that optimize the network's parameters by optimizing the log-likelihood of the data based on probabilistic assumptions. This is an important result: Wasserstein GANs provide a significantly better description of the data because they fit the statistical moments and hence directly model the underlying distribution of the data. Consequently, this should imply an improved description of the introduced jet observables compared to the VAEs in Chapter 5. Reviewing this assumption is the objective of this section.

Finally, Figure 6.25 shows the correlation between the energy and the unique latent-space seed $\widetilde{z}$ as defined in Section 5.3.2 for QCD jets only. Like in case of conditioned and unconditioned VAEs, the unique seed shows a clearly visible dependence on the reconstructed energy of the jet. Since the model in Figure 6.25a is unconditioned, the network has to learn a hidden structure that relates the images and their associated reconstructed energy (similar to the (discrete) classification illustrated in Figure 3.4). Regardung conditioned GANs in Figure 6.25b, the unique seed and the energy are obviously independent. Hence, the latent vector $\boldsymbol{z} \in \mathcal{Z}$ serves indeed exclusively as a seed for the structure (shower) of the image and not its energy.



(a) Unconditioned (Section 5.3)

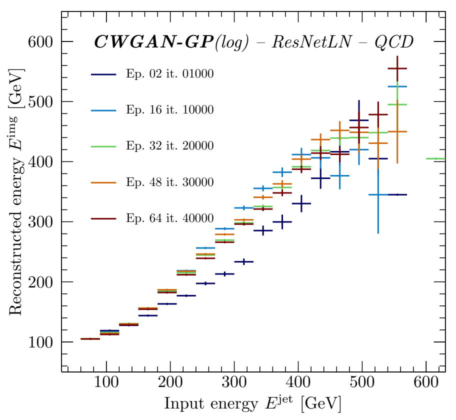


(b) Conditioned

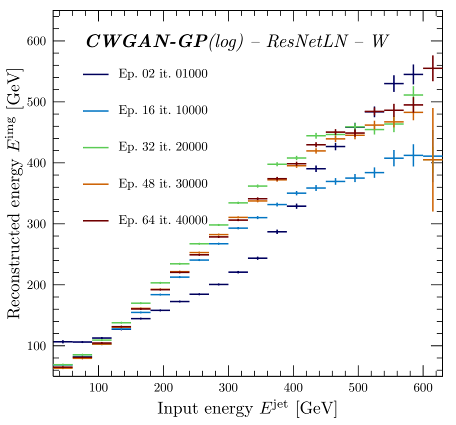
Plot 6.25: Correlation between the unique seed $\tilde{z}$ and the reconstructed energy E^{img} for an unconditioned (6.25a) and an conditioned (6.25b) Wasserstein GAN.

6.4.3 Kinematic distributions

Before studying some kinematic distributions in this section and other jet observables in the following section (6.4.4), one has to verify that the neural network is indeed conditioned using the energy of the jet. Like it was done in 5.4, the input to the model, i.e., the energy E^{jet} is plotted against the reconstructed jet energy $E^{\text{img}} = \sum_{k=1}^{N} E^{\text{pix}}_{\theta,k}(\boldsymbol{z}, E^{\text{jet}}, \eta^{\text{jet}})$ that is the scalar sum of all pixels in an image.



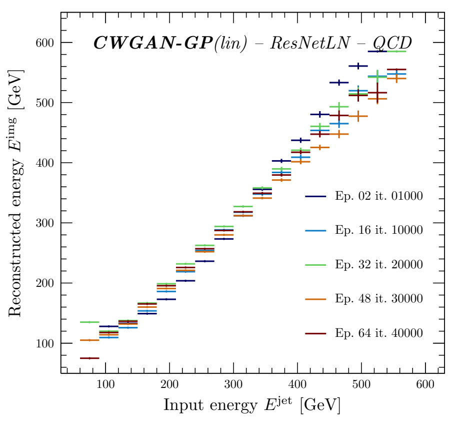
(a) QCD



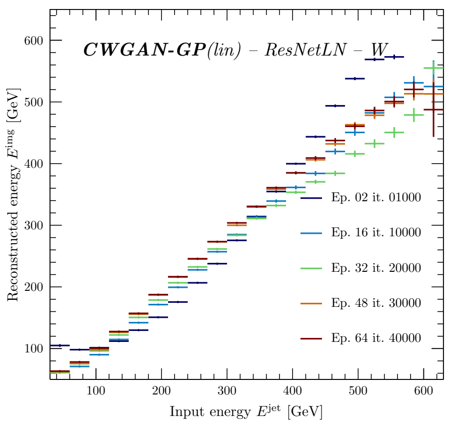
(b) *W*

Plot 6.26: Correlation between the reconstructed jet energy from the generated image E^{img} and the conditioning label E^{jet} for a *logarithmic* energy scale and 50,000 events.

The neural network has mostly learned to correctly reproduce the jet energy; however, the correlation is not exactly linear (as it is desired), but shows some non-linear boundary effects that are certainly non-negligible. In order to understand the cause of this undesirable effect, it is enlightening to show the very same plot for a network that has been trained on a *linear* energy scale ($E^{\text{pix}} \to \varrho E^{\text{pix}}$) instead.



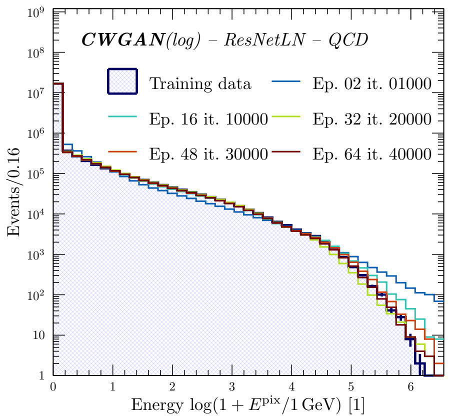
(a) QCD



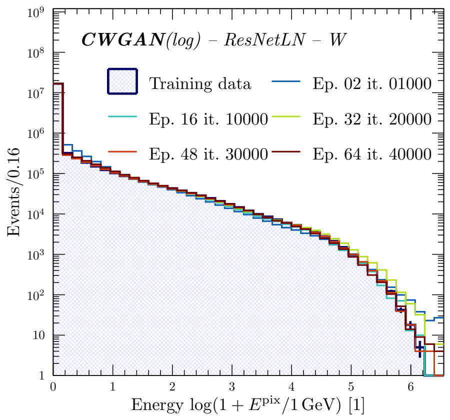
(b) W

Plot 6.27: Correlation between the reconstructed jet energy from the generated image E^{img} and the conditioning label E^{jet} for a *linear* energy scale and $50,000$ events.

In Figure 6.27 the effect is significantly reduced compared to the corresponding Figure 6.26; though, it is still present. So, the problem is most likely related to the logarithmic energy scale as described in Section 4.3 concerning the “invertible preprocessing”. At this point, this problem must remain unsolved. Nonetheless, one possible attempt to solve the issue could be to use a customized input/output layer in the critic/generator network that implements the transformation $T_\varrho(\boldsymbol{x}) = \log(1 + \varrho x)$ and its inverse $T_\varrho^{-1}(x)$. By doing so, the transformation is hard coded into the network's architecture.



(a) QCD

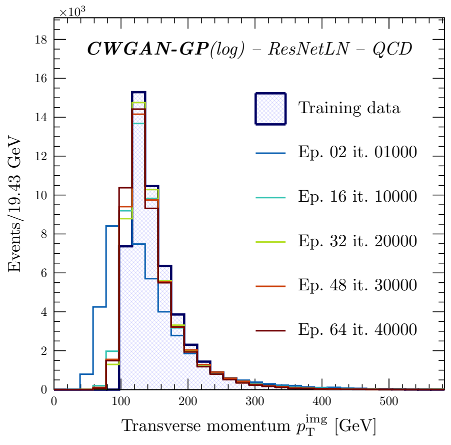


(b) W

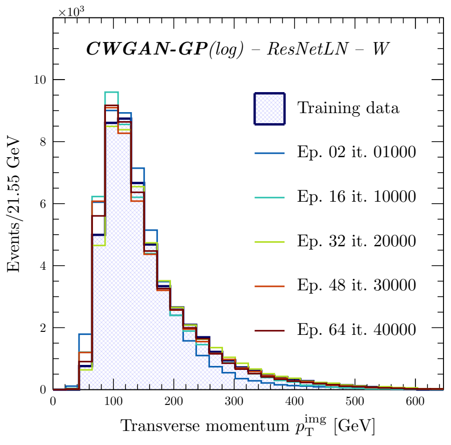
Plot 6.28: Pixel activation values E^{pix} on an event-on-event base for $50,000$ events.

Before the generated manifold is further explored in more detail, the distribution of the pixel activation values E^{pix} is analyzed again since it provides important information about the average description of the energy spectra learned the generative model (note that the distribution of energy activations is less conclusive than studying all energy spectra in each pixel individually since the position information is marginalized). The comparison of Plot 6.28a and 6.28b and the corresponding histograms in Figure 5.22 reveals a significant improvement in the agreement of the description. There is almost a perfect correspondence between the generated distribution and the one that is based on the training data. Even for regions with high "pixel energies" – with comparatively low statistics –, the generative model shows a good performance within statistical uncertainties. The same holds true for rather low pixel activations despite of the obvious dominance of empty pixels with $E^{\text{pix}} = 0$; this was not the case for Gaussian variational autoencoders (see, e.g., Figure 5.22a for $\log(1 + x) < 1$). The indisputable enhancement that can be seen in Figure 6.28 is a consequence of the improved correspondence between the generated and the real distribution at the level of their respective statistical moments. Furthermore, the figures above once again nicely illustrate the strong correlation between reduction of the earth mover's distance for larger number of training iterations and the *visual* improvement in the description of the data (compare the curve for 1000 training steps with corresponding graph at $40,000$ iterations).

From the pixel activations, it is just a small step to the total energy E^{img} and the transverse momentum of the jet, i.e., the scalar sum of E^{pix} on an event-on-event base[3].



(a) QCD



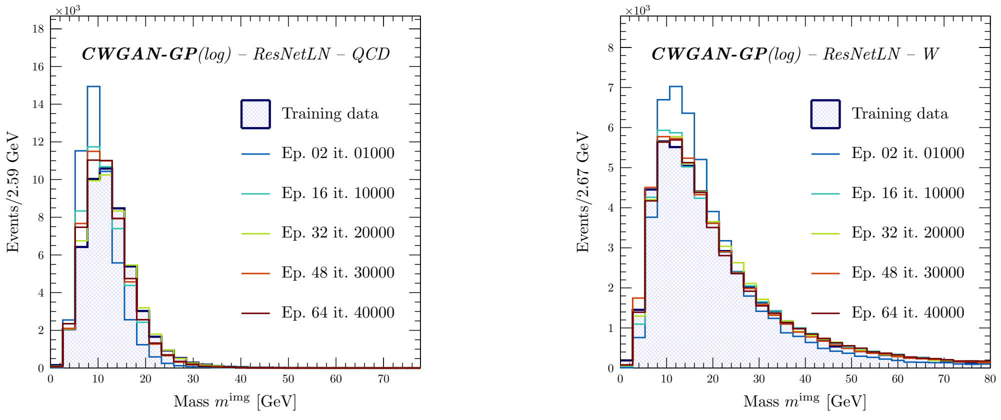
(b) W

Plot 6.29: Reconstructed transverse momentum $p_T^{\text{img}}/E^{\text{img}}$ for $50,000$ events.

[3]This is in general not true, since the p_T^{jet} of the jet is given by $\left(p_T^{\text{jet}}\right)^2 = \left(p_x^{\text{jet}}\right)^2 + \left(p_y^{\text{jet}}\right)^2 = \left(\sum_i p_{x,i}^{\text{pix}}\right)^2 + \left(\sum_i p_{y,i}^{\text{pix}}\right)^2$; hence, the transverse momentum in terms of the constituent's/pixel's position in the η-ϕ grid is $\left(p_T^{\text{jet}}\right)^2 = \left(\sum_i E_i^{\text{pix}} \frac{\cos\phi_i^{\text{pix}}}{\cosh\eta_i^{\text{pix}}}\right)^2 + \left(\sum_i E_i^{\text{pix}} \frac{\sin\phi_i^{\text{pix}}}{\cosh\eta_i^{\text{pix}}}\right)^2$. Due to the Lorentz boost and rotation (see 4.2), the azimuthal angle as well as the pseudorapidity of the constituents are restricted to a small window that roughly corresponds to $2R$ in which the "small-angle" approximations $\cosh\eta^{\text{pix}} \approx 1$, $\cos\phi^{\text{pix}} \approx 1$ and $\sin\phi^{\text{pix}} \approx 0$ are mostly valid. So, within the scope of this approximation, the transverse momentum corresponds to a simple scalar sum $p_T^{\text{jet}} \approx E^{\text{jet}} = \sum_i E_i^{\text{pix}}$.

The overall spectra are well described – which was expected from the results in Figure 6.26. The remaining discrepancies are a consequence of the aforementioned small non-linearity between the conditioning label and the reconstructed output; thus, this spectrum is better described by a linear scale.

The distribution of the mass as a representative of a Lorentz invariant and a "non-trivial" jet observable (in the sense that is also accounts for the relative position of the pixels/particles) is shown in Figure 6.30.



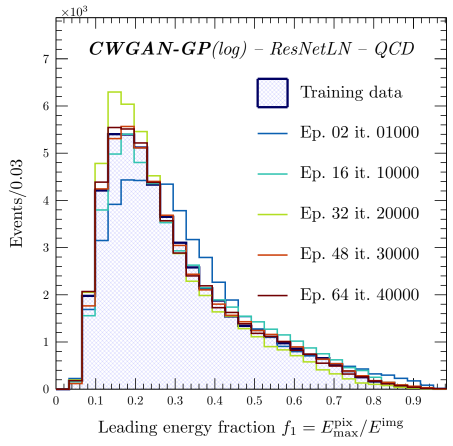
(a) QCD *(b)* W

Plot 6.30: Reconstructed jet mass m^{img} for $50,000$ events.

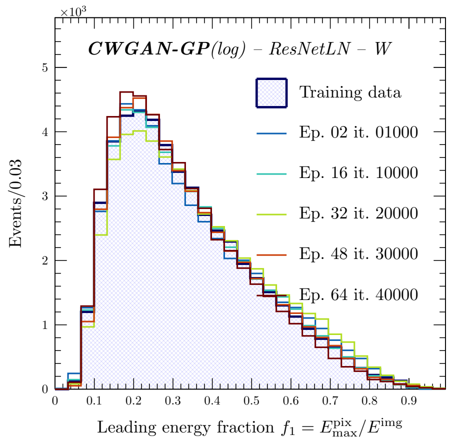
The distributions are in good agreement. The small bias towards lower mass values in case of the QCD spectrum is a consequence of the reduced occupancy of the images. The average occupancy of the images for the training set in case of QCD is roughly 12 %; the generative model, however, only populates approximately 8 % of the pixels on average. In case of W jets, the effect is less pronounced.

6.4.4 Other jet observables

This section studies further jet observables and their correlations among each other. The first quantity, which was not shown for VAEs, is the so-called *energy fraction* f_n^{img}. This very simple observable gives the energy ratio $f_n^{\text{img}} = E_n^{\text{pix}}/E^{\text{img}}$ of the largest $E_1^{\text{pix}} = \max_{i,j} E_{ij}^{\text{pix}}$, second largest E_2^{pix} etc. constituent/pixel in the jet/image and the total energy E^{img}. (The observable f_n^{img} is obviously not IRC safe, however, in this instance this circumstance does not represent a serious problem.) The results for the largest energy fraction $f_{\max}^{\text{img}} := f_1^{\text{img}}$ are shown in Figure 6.31.



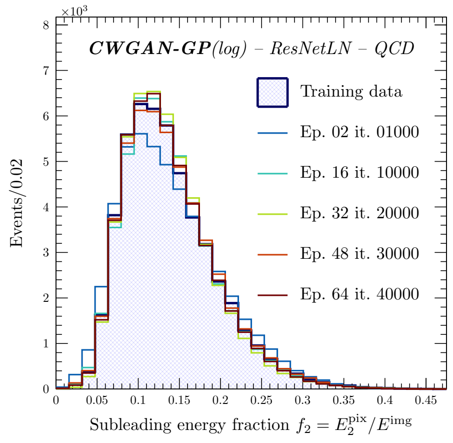
(a) QCD



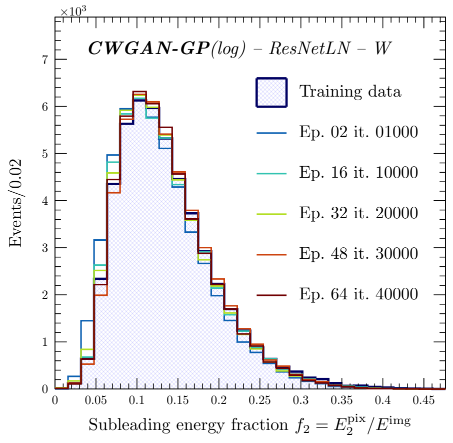
(b) W

Plot 6.31: Reconstructed leading energy fraction f_1^{img} for 50,000 events.

The very same is done for the second largest energy fraction in the image f_2^{img}.



(a) QCD

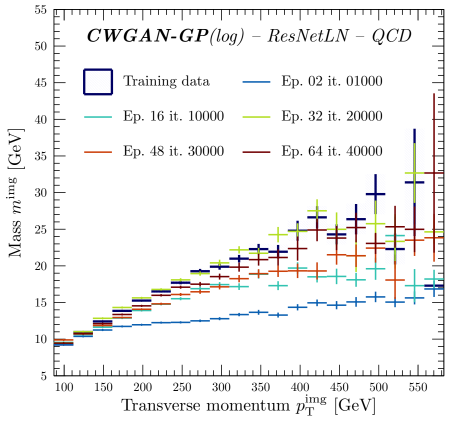


(b) W

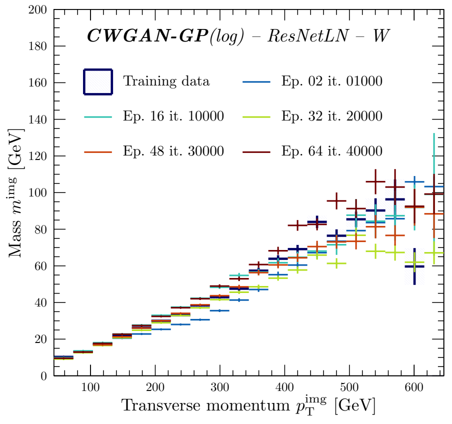
Plot 6.32: Reconstructed subleading f_2^{img} for 50,000 events.

The "beauty" of this jet observable lies in its simplicity, notwithstanding of its already mentioned flaw of not being IRC save. But, this is of no importance in this context because the comparison of the training data and its approximation by the generative model is consistent in every way. Both observables f_1^{img} and f_2^{img} are well described for both processes.

In order to estimate the performance of the neural network in different regions of phase space, the mass of the reconstructed jet is correlated with its respective transverse momentum. The results are summarized in Figure 6.33.



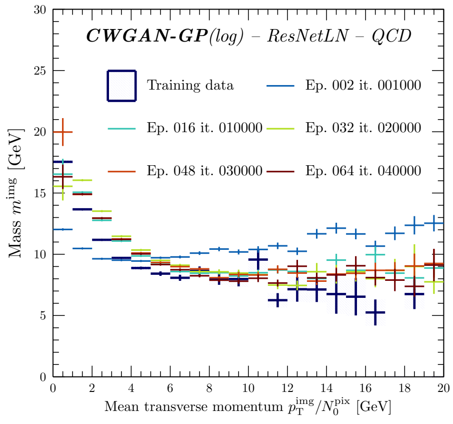
(a) QCD



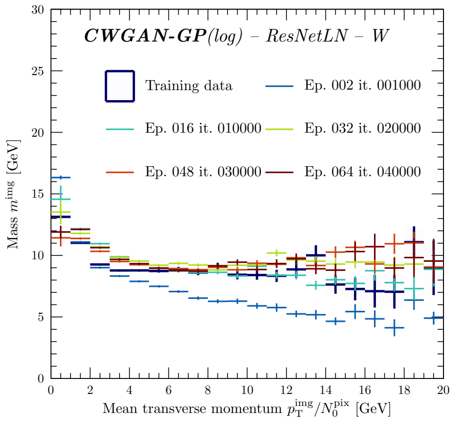
(b) W

Plot 6.33: Correlation between the reconstructed average mass m^{img} and the transverse momentum p_T^{img} for $50,000$ events.

The improvements of the generated data in case of conditional Wasserstein GANs are again obvious compared to the corresponding results in Figure 5.28 obtained with the CVAE. This statement holds true for the entire range of p_T^{img} values. The same applies for the correlation between the invariant mass and the transverse momentum normalized to the number of active pixels in the image $p_T^{\text{img}}/N_0^{\text{pix}}$.



(a) QCD

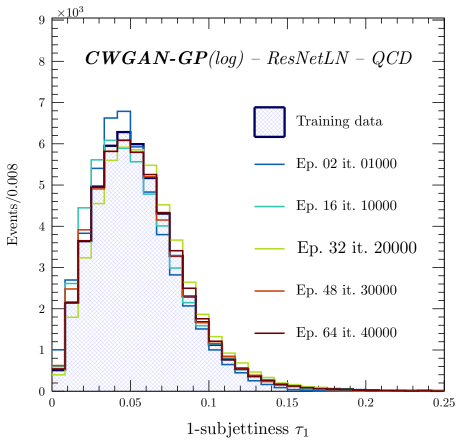


(b) W

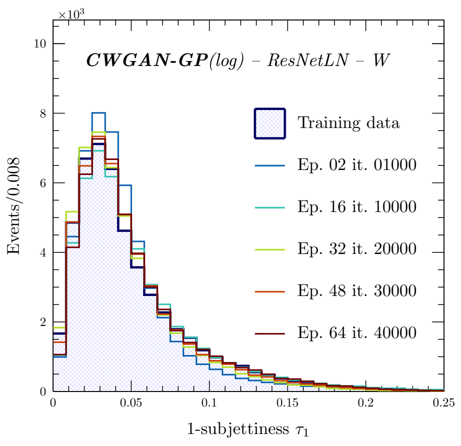
Plot 6.34: Correlation between the reconstructed average mass m^{img} and the mean transverse momentum per constituent $p_T^{\text{img}}/N_0^{\text{pix}}$ for $50,000$ events.

The systematic discrepancies that can be observed in Figure 6.34 can be explained by the underestimated occupancy in the generated jet samples. However, as already stated before, this result should be treated with caution since the number of constraints N_0^{pix} is not IRC.

A deeper look into the data's manifold is once more provided by the N-subjettiness that probes the substructure of the jet while being collinear and infrared safe at the same time. The results for 1-subjettiness, i.e., one subjet are shown in Figure 6.35.



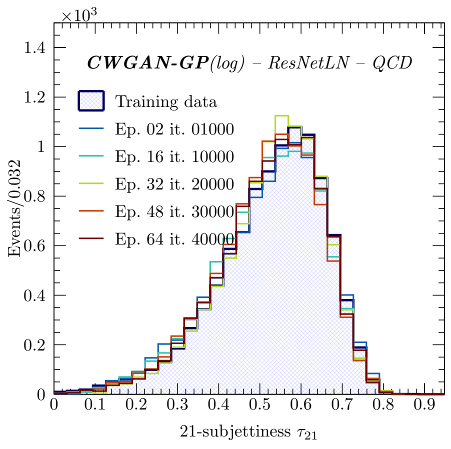
(a) QCD



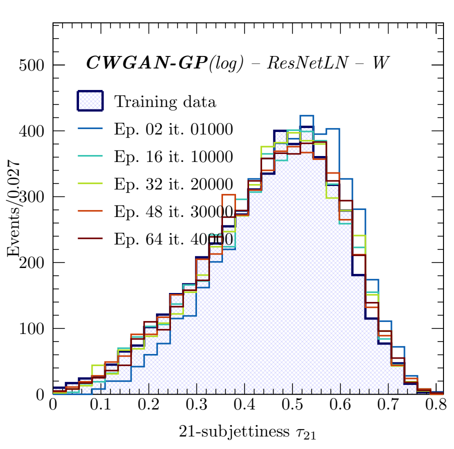
(b) W

Plot 6.35: Reconstructed 1-subjettiness τ_1^{img} for 50,000 events.

The same is done for τ_{21} that is the ratio of τ_2 and τ_1.



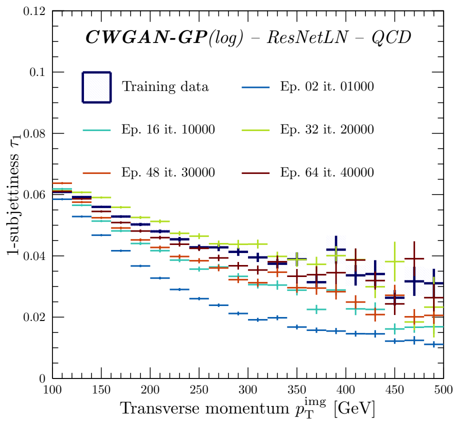
(a) QCD



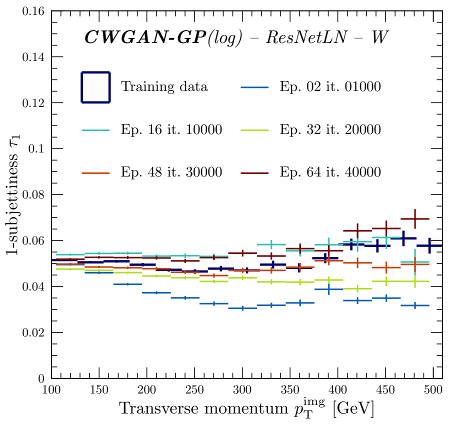
(b) W

Plot 6.36: Reconstructed 21-subjettiness τ_{21}^{img} for 50,000 events.

All spectra are sufficiently well described by the generative model. The same accounts for the correlation between τ_1 and the transverse momentum p_T^{img} of the reconstructed jet.



(a) QCD

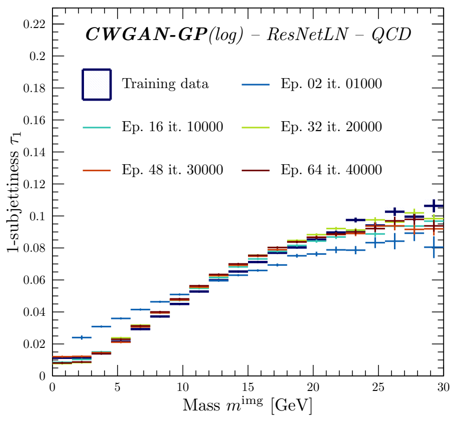


(b) W

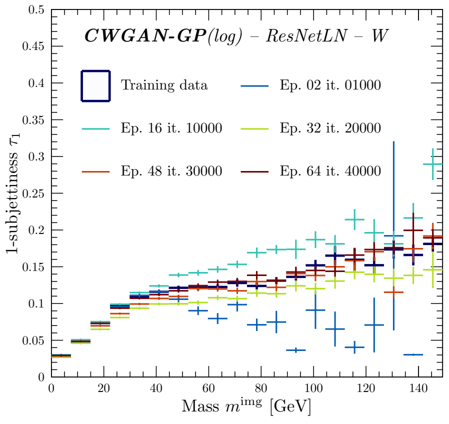
Plot 6.37: Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the transverse momentum p_T^{img} for 50,000 events.

Since according to Equation 5.12 the 1-subjettiness can be regarded as the mean distance (in the η-ϕ grid) from the *barycenter* of the jet $\tau_1 = \sum_{i=1}^{N} (\Delta R_i/R)) \left(p_{T,i}^{\text{pix}} / \sum_{j=1}^{N} p_{T,j}^{\text{pix}} \right)$, the generative model has correctly learned the correlation between the energy of the jet and its width concerning its opening angle. If the system is boosted, the radiated bremsstrahlung from the parton shower is more likely to be collinear to the initial parton. This effect can be directly observed in Figure 6.37. For higher transverse momenta of the jet, τ_1 decreases monotonically, i.e., the mean distance between the constituents is reduced and so is the (total) width of the jet (see Equation 1.31). This "hidden" information was not externally provided, but has been solely reconstructed by the neural neural learning the respective correlations that are encoded in the data.

Something similar can be done for the reconstructed invariant mass and τ_1, which is shown in Figure 6.38.



(a) QCD



(b) W

Plot 6.38: Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the mass m^{img} for 50,000 events.

As expected, τ_1 increases with the mass of the jet. Again, the degree of agreement between the generated histogram(s) and the distribution associated with the training data is excellent.

6.5 Conditioned WGANs with RNNs

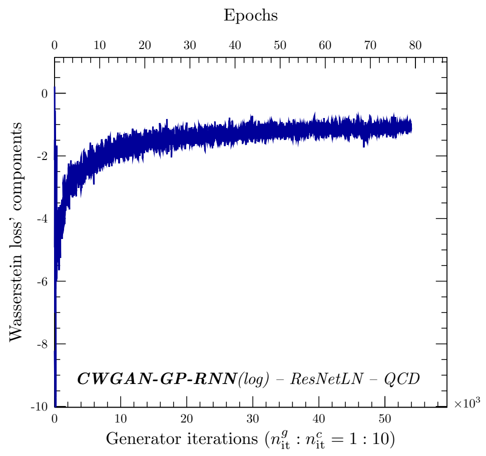
This is the final section in this chapter and contemporaneously in this report that will present actual results. As it was done in case of Gaussian variational autoencoders in section 6.5, the conditioned GAN with Wasserstein metric of the previous sections is extended by an RNN utilizing LSTM layers (see Figure 3.4). To allow for a fair comparison between VAEs and WGANs with RNNs, the architecture of the critic network in the adversarial system remain unchanged concerning the network introduced in section 6.1. Furthermore, the architecture of the generator is the same as the one of the decoder network (including the image generation layer (see Equation 5.16) with $\beta = 0$) in the VAE with LSTM layers introduced in section 5.5.2 – aside from the aforementioned (see Section 6.1) modifications that are required concerning the number of inputs to the neural network and the activations. Besides striking resemblance between the sequential model in case of WGAN and VAE, the hyperparameters of the LSTM layers (like gate weights, number of parameters etc.) and the number of time steps $n_T = 64$ remains unchanged too (see Figure 5.31). Based on the observations in Section 6.5, a significant deterioration of the general performance of the Wasserstein GAN compared to excellent results in Section 6.4 is to be expected. Nonetheless, the comparison is reasonably taking into account the quite different operating principles of WGANs and VAEs.

The structure of this section is inspired by the previous one. In the beginning, the convergence of the model must be confirmed, i.e., the monotone decline of the earth mover's distance with an increasing number of generator iterations. This is again followed by the visual inspection of some generated samples as well as the average image for both processes under consideration. Afterwards, the generated data is used to construct a small subset of the already familiar jet observables which are then compared to the results obtained in the previous section 6.5 and the VAE with RNN in Section 5.5.2.

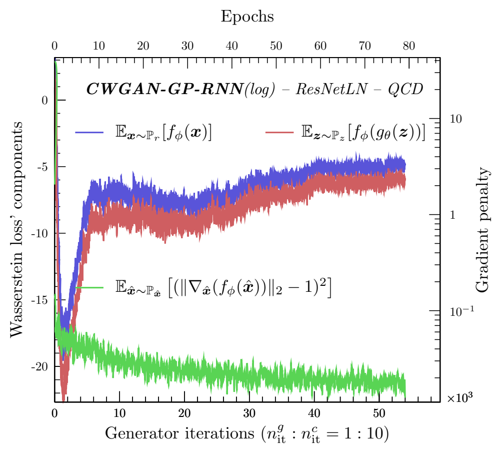
An additional note on the characteristics of this section. All plots presented here have already been introduced – in one way or another – at some point in this report. Therefore, most results presented in this section are left uncommented, with a reference made to the corresponding figures for comparison.

6.5.1 Training and convergence

The combination of Wasserstein GANs and RNNs give rise to some interesting training- and convergence behaviour that can bee seen in Figure 6.39 and 6.40.



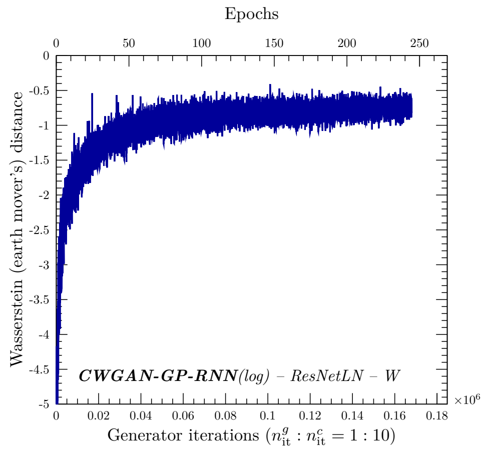
(a) Wasserstein loss



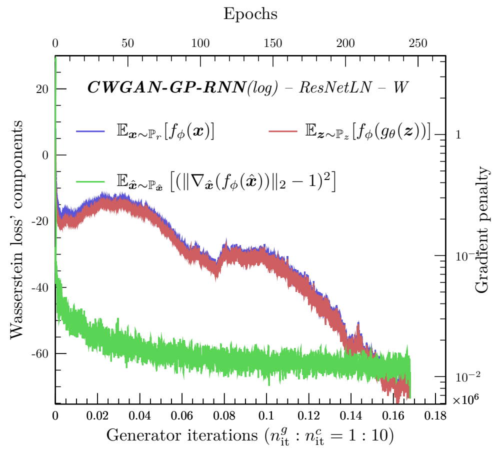
(b) Wasserstein loss' components

Plot 6.39: The Wasserstein loss (6.39a) and its components (6.39b) for QCD jets.

As apparent from Figure 6.39, the difference between the two expectation values $\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f_\phi(\boldsymbol{x})]$ and $\mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f_\phi(g_\theta(\boldsymbol{z}))]$, the approximation of the earth mover's distance is rater "large". This "gap", which ideally should be zero, increases with the number of time steps n_T in the recurrent neural network (not shown). This is effect is also present for the model that was trained on W samples. However, as can be seen in Figure 6.40, the discrepancy shrinks for larger number of iterations (this also applies to the QCD model); hence, it can be reduced by increasing the number of training steps.



(a) Wasserstein loss



(b) Wasserstein loss' components

Plot 6.40: The Wasserstein loss (6.40a) and its components (6.40b) for W jets.

Again, this effect as well as the somewhat slower convergence (*cf.*, for instance, Figure 6.16 and 6.17) is a consequence of the definition of the image generation layer accordingly to Equation 5.16 that maps an array of images to one single image (hence the name time projection layer). With the image generation layer, the gradients for the generator network

g_θ are roughly given by

$$\nabla_\theta f_\phi(g_\theta^\beta(\boldsymbol{z})) = \nabla_\theta f_\phi\left(\sum_{i=1}^{n_T} E_\theta^{(i)} \hat{g}_\theta(\boldsymbol{z})_i i^\beta\right) \tag{6.1}$$

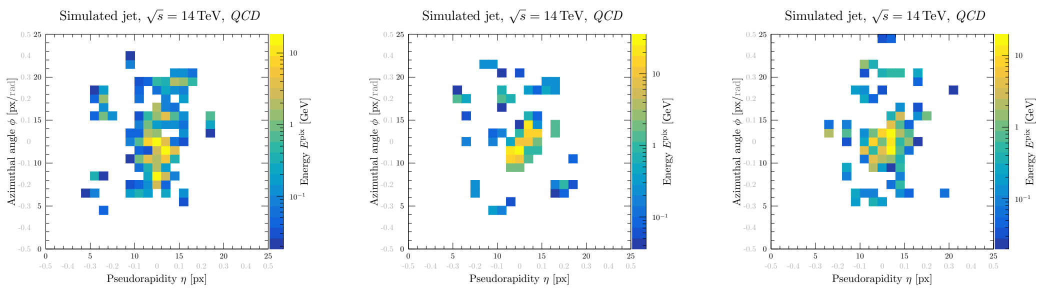
$$= \sum_{i=1}^{n_T} i^\beta \nabla_\theta \left(E_\theta^{(i)} \hat{g}_\theta(\boldsymbol{z})_i\right) \nabla_{\widetilde{\boldsymbol{x}}_i} f_\phi(\widetilde{\boldsymbol{x}}_i) \tag{6.2}$$

$$= \sum_{i=1}^{n_T} i^\beta \left(\hat{g}_\theta(\boldsymbol{z})_i \nabla_\theta E_\theta^{(i)} + E_\theta^{(i)} \nabla_\theta \hat{g}_\theta(\boldsymbol{z})_i\right) \nabla_{\widetilde{\boldsymbol{x}}_i} f_\phi(\widetilde{\boldsymbol{x}}_i). \tag{6.3}$$

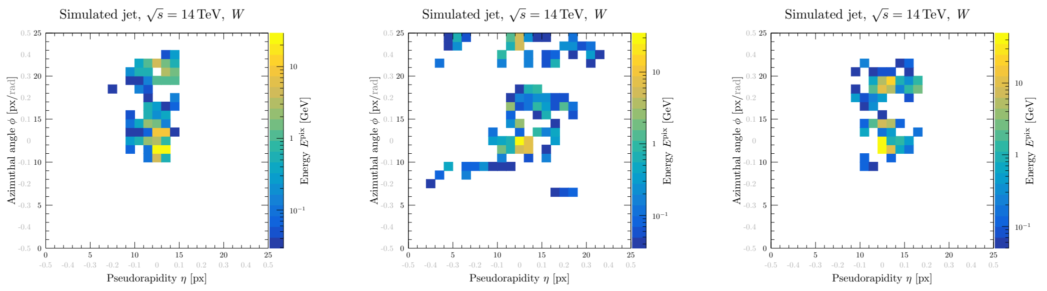
Equation 6.3 shows the modification of the gradients if the time projection layer is used. The factorization of the image into a scalar energy value $E_\theta^{(i)}$ and a normalized image, e.g., a discrete probability distribution $\hat{g}_\theta(\boldsymbol{z})_i$ gives rise to additional contributions to the gradients. This is potentially dangerous, which is why it would be reasonable to aim for a long term alternative to the image generation layer if this path should be continued.

6.5.2 Samples and average jet image

Compared to the results obtained with the combination of VAE and RNNs (*cf.* Figure 5.32 and 5.33) and the real samples (*cf.* Figure 4.3 and 4.5), the generated samples for the generative model with RNNs trained in a Wasserstein GAN look far more reasonable.

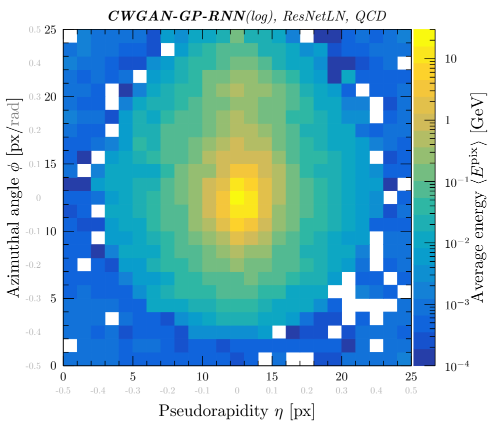


Plot 6.41: Three randomly simulated QCD jets after 40, 000 iterations.

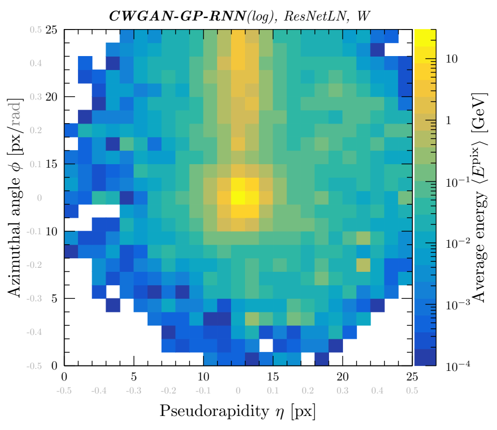


Plot 6.42: Three randomly simulated W jets after 40, 000 iterations.

Still – like it was the case for the CVAE with RNNs –, the occupancy in the image is increased (QCD samples in particular) compared to the previous setup. This becomes especially apparent in the average jet image.



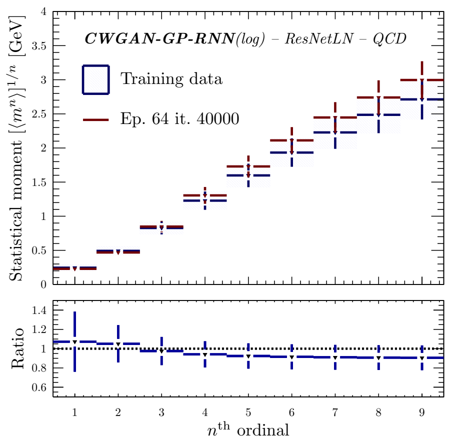
(a) QCD



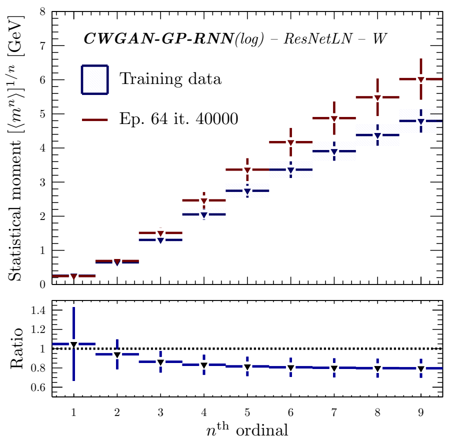
(b) W

Plot 6.43: Average jet image for 50,000 events and 40,000 iterations.

This is once more a consequence of the time projection layer (to be more precise: the sum that merges the individual images). However, unlike in Figure 5.34, the additional radiation in the image's periphery is *very soft* and almost completely disappears for more numbers of training iterations.



(a) Statistical moments QCD



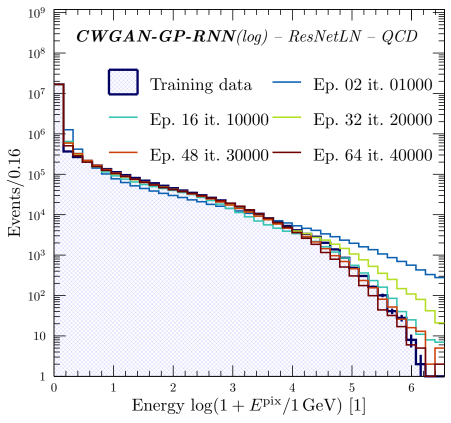
(b) Statistical moments W

Plot 6.44: The first nine statistical moments according to Equation 5.3 for the training data and the generated distribution of the generative model.

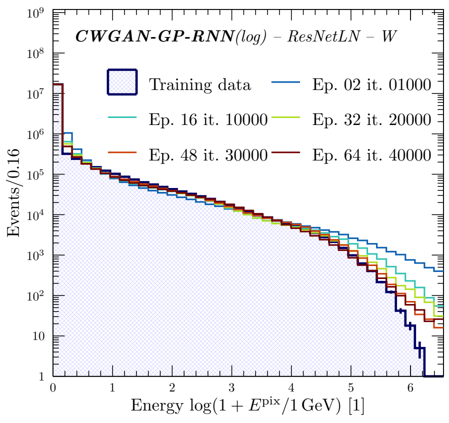
The high degree of agreement of the statistical moments between the underlying distribution of the training data and the generated distribution was one of the milestones of the previous section. It is therefore particularly interesting to study the changes in this distribution for the new setup. The trend does not continue as can be seen in Figure 6.44. For the same number of training iteration (40,000), the agreement between the statistical moments is significantly worse compared to Figure 6.24 but still considerably better than the results based on the VAE in Figure 5.39.

6.5.3 Kinematic distributions

The deterioration of the agreement between the statistical moments according to Figure 6.44 already suggests that the same tendency is to be anticipated in case of the other jet observables. This expectation is immediately confirmed by the distribution of pixel activation values/energies in Figure 6.45.



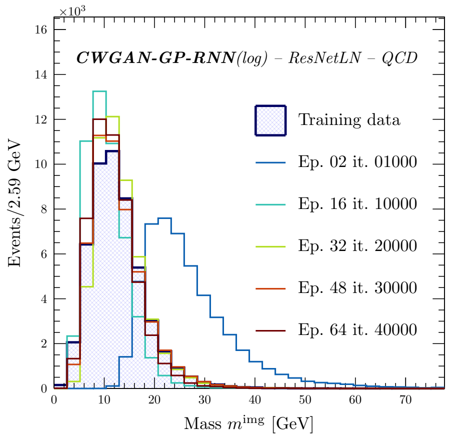
(a) QCD



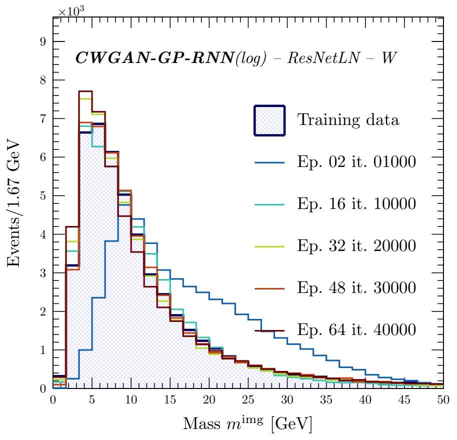
(b) W

Plot 6.45: Pixel activation values E^{pix} on an event-on-event base for 50,000 events.

The large discrepancy (which is hidden by the logarithmic representation) mostly affects small energy values. This mismodelling of the energy propagates to other jet observables like the mass which is shown in the following figure.



(a) QCD



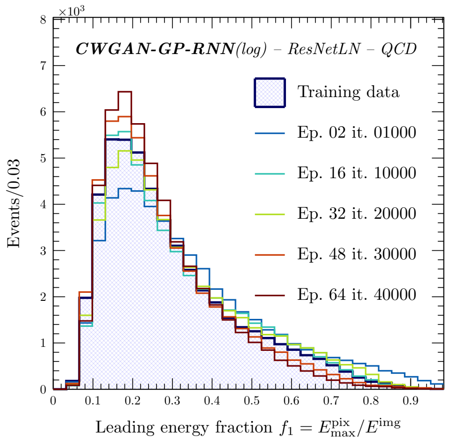
(b) W

Plot 6.46: Reconstructed jet mass m^{img} for 50,000 events.

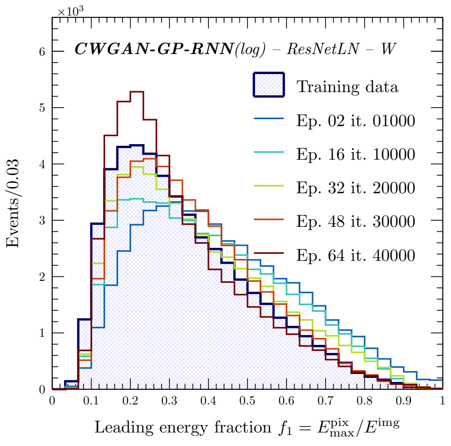
Still, the approximation of the underlying distribution of the data is of rather high quality compared to Figure 5.40. Furthermore, contrary to the Gaussian variational autoencoder with recurrent networks in the previous chapter, no energy cut needs to be applied in order to archive this degree of agreement, making this method much more consistent and hence less arbitrary.

6.5.4 Jet observables

The trend continuous for the other jet observables. Interestingly, the apparent simple leading energy fraction $f_{\max}^{\text{img}}$ (see Section 6.4.4) is badly described by the generative model in the case of QCD (Figure 6.47a) as well as for W (Figure 6.47b) initialized jets.



(a) QCD

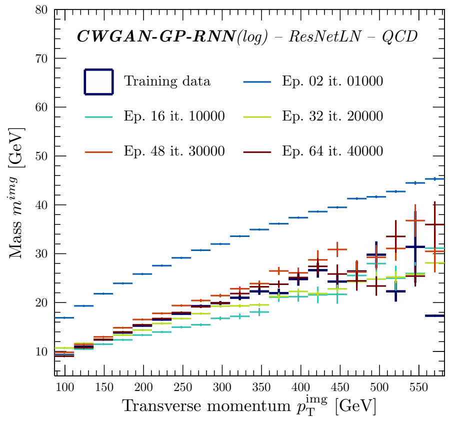


(b) W

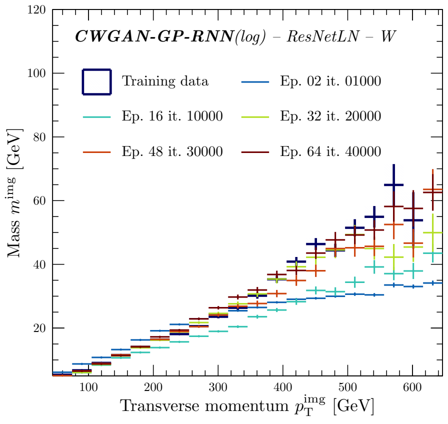
Plot 6.47: Reconstructed leading energy fraction f_1^{img} for $50,000$ events.

This discrepancy is caused by the denominator, i.e., the total (jet/image) energy that is given by $E^{\text{img}} = \sum_{i=1,j=1,k=1}^{n_T, n_\eta^{\text{pix}\prime}, n_\phi^{\text{pix}\prime}} i^\beta \left(E_\theta^{(i)} \hat{g}_\theta(\boldsymbol{x})_i\right)_{kj}$. The increased occupancy in the image then causes an overestimation of the jet's energy and hence a bias towards smaller values of $f_{\max}^{\text{img}}$.

Surprisingly, the correlation between the mass of the jet and the transverse momentum/energy is comparatively well described.



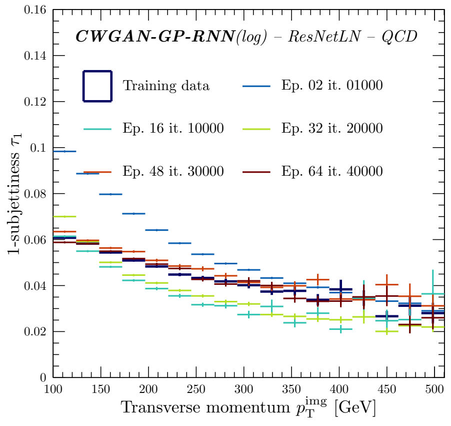
(a) QCD



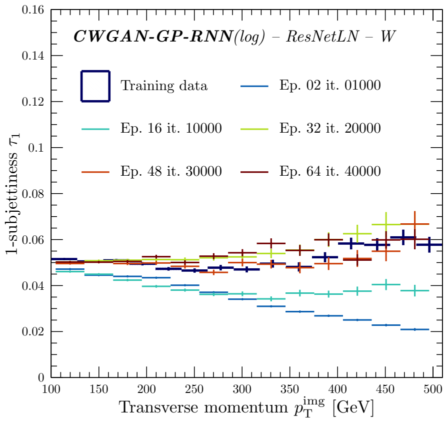
(b) W

Plot 6.48: Correlation between the reconstructed average mass m^{img} and the transverse momentum p_T^{img} for $50,000$ events.

The same goes for the correlation between the 1-subjettiness τ_1 and the transverse momentum p_T^{img} (Figure 6.49) and the mass (Figure 6.50) of the jet.

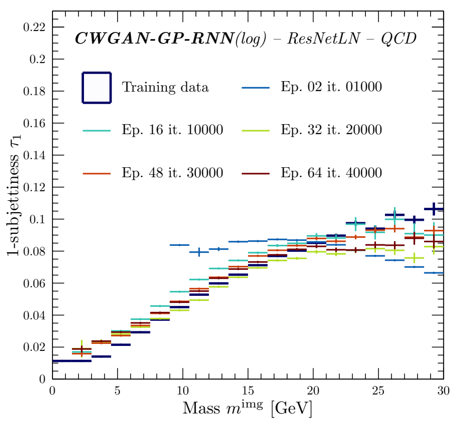


(a) QCD

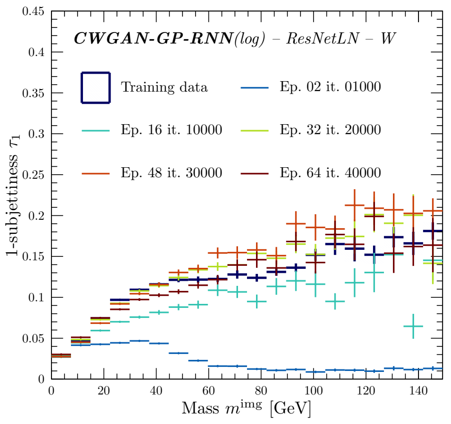


(b) W

Plot 6.49: Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the transverse momentum p_T^{img} for $50,000$ events.



(a) QCD



(b) W

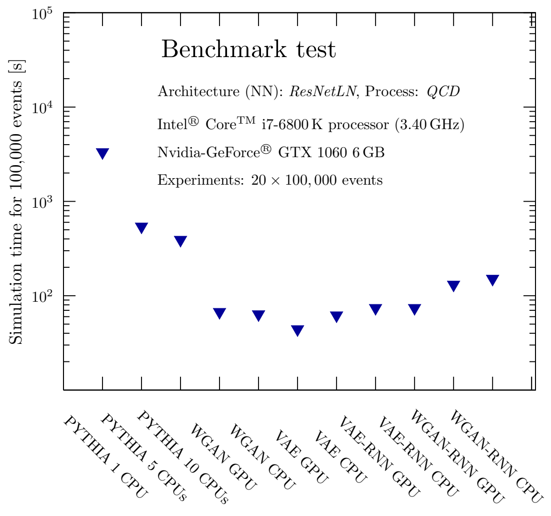
Plot 6.50: Correlation between the reconstructed average 1-subjettiness τ_1^{img} and the mass m^{jet} for $50,000$ events.

All in all – based on the results presented in this section –, it can be said that unlike in the case of VAEs with RNNs, the combination of recurrent networks with Wasserstein GANs can produce reasonable results. Nonetheless, compared with the conditioned Wasserstein GANs in Section 6.4 without RNNs, a clear deterioration can be observed – especially in the description of the statistical moments of the generated distributions. Hence, the combination of generative models and recurrent neural network for the modelling of the underlying sequential particle shower has not proven to be particularly effective (at least in combination with the simple time projection layer).

6.6 Final notes

This chapter studied the application of generative adversarial (deep neural) networks based on the earth mover's distance in the context of "full" event generation, i.e., matrix element and particle shower as well as the exclusive simulation of parton showers. The first section 6.1 quickly summarized all necessary changes that are required to apply the network architectures introduced in the context of VAEs in Section 5.1 to Wasserstein GANs with gradient penalty. Subsequently, the configuration of hyperparameters (such as the learning rate α_l, the optimization algorithm, and the dimensionality of the latent space $\dim(\mathcal{X})$) was motivated in Section 6.2. It has ben shown, that, especially the choice of the optimization algorithm that is used for gradient descent is crucial. The optimization algorithm should not rely on momentum to avoid problems with the gradient penalty term in the loss function. Besides this observation, however, the training of Wasserstein GANs has turned out to be very stable and reliable with little to no mode collapse as it was constantly the case with classical GANs. In fact, without Wasserstein GANs, a systematic study of the subject would have been impossible since the application of classical GANs (in the context of event simulation) turned out to be a disaster. (Weeks were passing and have quickly turned into months without any stable configuration found that would allow for replicable results.) Almost all training sessions ended up in a collapsed mode of the data set. For the rare occasions that, by chance, the training did not result in mode collapse, vanishing gradient maliciously sprayed its poisonous breath and brought all progress to a sudden halt. Furthermore, the hard-to-interpret loss function required close monitoring of the training and early stopping. The combination of plenty methodes had been tried in hope to somehow stabilize the training – without success. It was not until Wasserstein GANs caught the attention of the author that this thesis was possible in the first place. After the rather technical but important details, in Section 6.3 the Wasserstein generative adversarial network (with gradient penalty) has been used to generate events in high energy particle physics, i.e., matrix element at leading order with parton shower simulation for some processes. Compared to the results obtained for Gaussian VAE for the same scenario, the generative model has learned to simulate very realistic jets images that are visually indistinguishable from the training data. Also, the closer inspection of the generated manifold using other jet observables that probe deep substructure of the jets could not reveal any significant deviations. Furthermore, this section showed that the use of a logarithmic energy scale for the activation values in the pixels of the image is beneficial – although it results in a non-linear relation between the energy input and the output of the model. in Section 6.4, the information of the matrix element has been marginalized by conditioning the previously introduced model regarding the energy as well as the pseudorapidity of the jet; thus, the model is reduced to the simulation of parton showers. Out of all models in this report, this one gave the best results. The generated probability distribution, $\mathbb{P}_g$, agreed with the training data $\mathbb{P}_r$ in several statistical moments to a high degree of precision. This positively affected the description of the kinematic distributions and other jet-like observables. Now all of them are in excellent agreement with the expectations from the training data. The cause of this significant improvement is a result of the earth mover's distance; more precisely: its fundamental connection to the maximum mean discrepancy as an integral probability metric. Based on the results in Section 6.4, small deviations still present in the approximation of the training data can likely be improved by increasing the number of generator iterations and/or increased complexity of the network(s). Encouraged by the results, the (conditioned) model was combined with recurrent neural networks – LSTM cells to be more precise – in Section 6.5. After the first experiences with this method in Section 6.5, the worst was to be expected. But, contrary to the previous chapter, the combination of Wasserstein GANs and RNNs provided reasonable results – although some deterioration in the description of the observables could be seen. Based on this results, it has to be stated that recurrent neural networks are not suited to model the sequential nature of the particle shower if this information is not directly extractable from the training data (since the image represents a time projection). However, there is another possible application of RNNs and generative models in the context of the simulation of higher jet multiplicities in an event, which is proposed in Section 7.2.3 of Chapter 7 in the context of future research suggestions.

Now that all networks have been evaluated, it remains to evaluate the speed performance of the generative models compared to the conventional approach (i.e. event simulation based on Monte Carlo) in the form of a "quick" benchmark test. For this purpose, 100,000 QCD events have been simulated for twenty repetitions to average out fluctuations in the processor's load. The generative models are, in turn, assessed on a CPU (Intel$^{®}$ Core$^{\text{TM}}$ i7-6800 K processor (3.40 GHz)) and a GPU (Nvidia-GeForce$^{®}$ GTX 1060 6 GB) whereas the performance of the MC event generator (Pythia8.2) is evaluated on a different number of CPU cores. The result of the benchmark test is shown in Figure 6.51.



Plot 6.51: "Benchmark test" of the performance of several generative models on CPUs and GPUs compared to the standard method of event simulation through Monte Carlo generators for a different number of CPU cores.

As can be seen, the generative models based on machine learning methods allow for much faster event simulation compared to the standard MC approach once the model has been trained. Moreover, the time requires to simulate an event for higher orders through Monte Carlo increases significantly while it remains unchanged in case of the generative model. (Admittedly, this comparison is not entirely fair since the time required to train the networks – which can easily take several days – has not been taken into account. But, once the network has been trained, event simulation is very fast.)

Chapter 7

Conclusion

This chapter concludes this master thesis and thereby contemporaneously the adventurous journey through the world of machine learning in the context of the simulation of QCD radiation. Habitually, the last chapter provides a brief recapitulation of the objective, the methods used, as well as the results gained through this report. Afterwards, the discussion is followed by some personal suggestions concerning future research in this field that, hopefully, serves as a inspirational seed for further projects and studies.

7.1 Discussion

This thesis is concluded by a brief summary and a subsequent assessment of the obtained results as well as the insights gathered within the context of this report.

The objective of this Master's thesis was to study the applicability and the feasibility of event simulation in high-energy particle physics through generative models utilizing machine learning methods. The main focus of attention was on variational autoencoders with a Gaussian constraint on the latent space as well as generative adversarial models based on the earth mover's distance that are currently the two most prominent approaches to generative models. This idea is appealing since the precision of Monte Carlo event-simulation is limited by the fixed-order expansion of the matrix element calculation and the approximation scheme in the parton shower simulation that is a Markov Chain process and hence completely neglects quantum interference. Apart from that, the simulation of the acceptance of the measuring apparatus that is used in the experiment is limited. A solely data-driven approach through neural networks addresses both problems at once by directly reproducing the data that is precise (to all orders in perturbation theory) by definition, as well as going along with a complete detector simulation since the model is trained on data that has been recorded with a particular particle detector.

Gaussian VAEs have been extensively investigated in Chapter 5. The study revealed that VAEs are only able to provide a decent description of the training data for the very first statistical moment of the underlying distribution (see Figure 5.7, 5.24 and 5.39). Thus, the generative model inevitably misses characteristic features of the data that are not embedded in the latent space representation of the data. This circumstance becomes particular apparent in the various distributions that have been studied during the course of Chapter 5. The origin of this apparent mismodelling of the data is most likely caused by the underlying probabilistic assumption of Gaussian VAEs, i.e., the Gaussian distribution of reconstruction errors in the ELBO. More generally, the constraint on the shape of the underlying probability distribution of the latent space, which is the absolute prerequisite to use autoencoders to train generative models in the first place, gives rise to blurred and noisy data (see, for instance, the individual jet images in Figure 5.3, 5.14, 5.32 and image occupancy in Figure 5.5, 5.16b). This phenomenon does by no means come as a surprise; on the contrary, it is a characteristic albeit undesirable property of VAEs, which is the subject of many ongoing studies. Despite all efforts, this problem is yet unsolved and hence remains a limiting factor in the application of variational autoencoders in the context of image generation in general. Nonetheless, the performance of the Gaussian VAE showed significant differences depending on whether the model is conditioned (Section 5.4) on the energy and the pseudorapidity of the jet or not, i.e., the factorization of the model into matrix element calculation and parton shower simulation. In case of an unconditioned model (Section 5.3), the network has to construct a complicated representation of the data that categorizes the images according to its associated *continuous* energy label. This task is highly non-trivial, and the model apparently struggles to learn an appropriate representation in the latent space. In case of conditioned VAEs, however, the performance of the neural network significantly improves since the model focuses on the parton shower and doesnt have to learn the information of the matrix element. Nevertheless, the agreement of the statistical moments is still limited to the very first order (see Figure 5.24). This indicates that the conditioned VAEs does not learn a proper approximation of the underlying distribution that generated the training data. Hence, the problem is most likely related to the fundamental probabilistic assumptions that are fundamental to Gaussian VAEs. Afterwards, recurrent neural networks and VAEs have been combined with the objective to model the underlying sequential nature of the splitting process that in the end gives rise to the particle jets observed in the calorimeter of the detector. This modification of the model resulted in no improvement, but, on the contrary, in a deterioration of the overall description. This is hardly surprising since the sequential character is not directly present in the training data that represents a time projection of the subsequent splittings of the partons (the next section suggest an alternative application of RNNs in combination with generative models in the context of HEP).

Chapter 5 revealed evident limitations in the application of Gaussian VAEs. This finding was, among other aspects, a decisive motive to study a completely different approach to generative models through adversarial networks, which was the main topic of Chapter 6. In doing so, this report omitted a substantial portion of the actual study, which was the investigation of classical GANs (see Section 3.6.1) that are based on the Jensen-Shannon divergence. As it turned out, the notorious instabilities with regard to mode collapse and vanishing gradients in particular ruled out any systematic analysis concerning those kind of generative adversarial models. In fact, the somewhat cumbersome and excessive preprocessing procedure of the data introduced in Section 4 actually directly emerged from the effort to stabilize the training in the context of classical GANs – with moderate success though. It was not until Wasserstein GANs or rather integral probability metrics caught the attention of the author that a methodical study was within the realms of possibility in the first place. After the aforementioned steep initial learning curve, Wasserstein GANs quickly turned out to be very well suited for the application in the context of event simulation in high-energy particle physics. This becomes particularly apparent in the excellent agreement between the statistical moments of the generated distribution and the distribution associated with the training data (see Figure 6.24b and 6.44b). This is an unmistakable indication that the generative model indeed learned the actual distribution of the training data since two probability distributions agree iff they coincide with each other on all statistical moments. The origin of the good accordance between the generated and the expected distribution is the aforementioned deep fundamental connection between the earth mover's distance and the maximum mean discrepancy that measures the similarity of two distributions with respect to their statistical moments (see Section 6.4.2), while the VAE directly optimizes the log-likelihood of the data. The high quality of the model manifests itself in the accurate description of various figures of merit. Thus, for instance, the generated samples are visually completely indistinguishable from the training data (*cf.*, e.g., Figure 6.8 and Figure 4.3). Contrary to VAEs, the the overall performance of the unconditioned (Section 6.3) and the conditioned model (Section 6.4) is almost identical. This is an interesting observations since the factorization of matrix element and parton shower was crucial in case of VAEs. Only in the event of Wasserstein GANs combined with RNNs, a noticeable deterioration of the performance could be observed as it was the situation for the corresponding setup in case of VAEs. One problem that remains to be solved is the insufficient modelling of the number of active pixels in the generated images, which can exemplary be seen by the deviations in Figure 6.34. First, it has to be clarified whether this non-conformance in the number of active cells (which is related to the number of "constituents" in a jet) poses a problem after all since this quantity is not IRC save. As it turned out, the observed deviations are primarily due to *very* soft activity in the image/jet that barely affects other jet obervables. Hence, the problem may be avoided by applying a low cut on the energy threshold in the image from the beginning. However, this minuscule flaw is bearable in view of the excellent agreement between the generated distribution and the expectations from the training data. Additionally, with the robust Wasserstein GANs available, the preprocessing of the data might be obsolete to a certain extent since it does not affect the training stability. However, the utilization of intrinsic symmetries in the data still reduces the training time as well as the necessary size of the training set. But since according to Section 4.3 the preprocessing of the images is accompanied by information loss (particularly due to to the rotation and the concomitant interpolation between neighbouring pixels), it might be more reasonable to accept longer training times in favour of unbiased or distorted data. In consideration of the aspect of consistency, the entire preprocessing chain was applied to the data within the scope of this thesis.

There is another, far more serious problem associated with this method: the problem of double-counting. This issue was already discussed in Section 2.4 in the context of Monte Carlo event generation; however, it is also relevant in the scope of the machine learning method presented in this thesis. Solving this problem will be an important aspect concerning further studies in this field.

To finally summarize: Wasserstein generative adversarial networks have proven to be the method of choice over classical GANs (based on f-divergences) and Gaussian variational autoencoders. They are superior with respect to the stability of the training routine (no mode collapse nor vanishing or exploding gradients), the quality (good agreement of the statistical moments) as well as the diversity (high entropy) of the generated data and its associated probability distribution. Only with respect to the training performance, i.e., the overall time required to train the model, Wasserstein GANs using gradient penalty have a considerable disadvantage compared to the other methods since the model requires very small learning rates and the computation of the complicated gradient penalty term is computationally quite expensive and consequently time consuming.

Last but not least, the two generative models, Wasserstein GANs and Gaussian VAEs, have been compared with the conventional approach to event simulation through GPMC event generators with respect to their time required to simulate $100,000$ QCD events by means of a quick benchmark test (see Figure 6.51). The models are alternately evaluated on a CPU and GPU, while the Monte Carlo simulations are performed on 1, 5 and 10 CPU(s) respectively. After the networks have been trained and the model configuration is fixed, the generative models allow for fast event simulation that is orders of magnitude lower compared to the same task done with Monte Carlo. It should be emphasized again that the benchmark test in Figure 6.51 was evaluated based on LO samples. The simulation of NLO or even NNLO events through generative models however remains unaltered. With the final discussion of the obtained results and the gathered insights, this Master's thesis finally reached its end. The last section is supposed to provide some further ideas, proposals and research suggestions that are related to thematic framework covered in this thesis.

7.2 Future research suggestions

This master's thesis is nothing but a small and insignificant piece in a large picture that slowly starts to emerge. Hopefully, this work will contribute to a further development of the field and help to build a bridge between machine learning and particle physics even if only certain aberrations are not repeated by subsequent studies. Within the scope of this project, many ideas have come to mind on how to proceed. Of course, all of those suggestions are at a very early stage of development and have not been completely thought through. However, the concepts proposed in this last chapter are not supposed to be the answer to any question but to provide some inspiration for future research.

7.2.1 Bayesian networks – "I know what I don't know"

There are two interpretations of probability distributions or probabilities in statistics: the *frequentist* and the *Bayesian* approach. According to the frequentist interpretation, the probability of an event to occur corresponds to the relative frequency of this event to the total number of trials in the limit of infinite statistics. According to the Bayesian interpretation on the other hand, the probability reflect the "degree of belief" and so represent an "uncertainty" over unknown parameters.

A neural network can be considered as a probabilistic model that tries to model some random process based on a finite random sample of *events* $\{x\}_{i=1}^{N}$ from a data set $\mathcal{D}_N$ with size N through a parameterized probability distribution of $p(\mathcal{D}_N|w)$. The objective to learn the parameters of the network w to provide a good description of the data set $\mathcal{D}_N$ based one a likelihood function $\mathcal{L}(w|\mathcal{D})$ that gives the probability of the data based on the unknown parameters w. The frequentist approach to the problem is to approximate the unknown parameters of the distribution w by what is called an *estimator* $\hat{w}$ by means of maximum likelihood optimization such that $p(w|\mathcal{D}_N) \approx p(\hat{w}|\mathcal{D}_N)$. The law of large numbers – on which the frequentist philosophy is founded on – says that the estimator will converge to the true parameter in the limit of infinite statistics. So, the parameters are fixed but unknown. The situation is fundamentally different in the Bayesian picture that is based on Bayes' rule and hence assumes a probability distribution $p(w|\mathcal{D}_N)$ (posterior) over the parameters of the model, which corresponds to probability of parameters given the data $p(\mathcal{D}_N|w)$ (likelihood) and some state of knowledge $p(w)$ (prior). According to Bayes' rule, the posterior distribution is given by $p(w|\mathcal{D}) = \frac{p(\mathcal{D}_N|w)}{p(w)}$. As a result of the prior, the posterior is a probability distribution as well. Translated to neural networks (or any parameterized distribution), this means that instead of an estimation of some unknown parameter, Bayesian methods provide a probability distribution over weights instead. Therefore, in *Bayesian neural networks* [MacKay, 1992, Neal, 1996], the weights of the model are random variables associated with some probability distribution in place of an estimated (according to the frequentist interpretation) parameter (point estimate). This may sound trivial, but it represents a considerable change. In practice, this means that the neural network has to learn a distribution of each parameter in the model. This distribution is unknown beforehand and needs to be determined by means of Bayesian inference, which is numerically intractable. Therefore, the posterior distribution $p(w|\mathcal{D})$ is usually approximated by a variational posterior $q_\theta(w|\mathcal{D})$ that optimizes the *variational free energy*, i.e., the ELBO $\mathcal{L}(\mathcal{D}, \theta) = \mathbb{E}_{q_\theta(w|\mathcal{D})} \log q_\theta(w|\mathcal{D}) - \mathbb{E}_{q_\theta(w|\mathcal{D})} \log p(w) - \mathbb{E}_{q_\theta(w|\mathcal{D})} \log p(\mathcal{D}|w)$ by means of, e.g., "Bayes by Backprop" [Blundell *et al.*, 2015]. Furthermore, for most practical applications the variational posterior of the weights is constrained to be a multivariate Gaussian with mean $\mu(x)$ and variance $\sigma(x)$. This procedure is conceptually very similar to the Gaussian constraint of the latent space in case of variational autoencoders; but, applied to all weights in the network. Despite being tractable, this still represents doubling of the numbers of trainable parameters in the model compared to classical neural networks; hence, significantly increase the training time. Nonetheless, with a distribution over each parameter, the neural network does not represent a deterministic model anymore but a random process. The probabilistic nature of the model now allows to associate uncertainties with the prediction(s) $\hat{y} = f_{w \sim q_\theta}(x)$ of the neural network that now is a random number. So, the mean $\mu(x)$ and the variance $\sigma(x)$ for a repeated number of experiments $\{\hat{y_i}\}_{i=1}^N$ is simply given by $\mu(x) = \frac{1}{N} \sum_{i=1}^N f_{w_i \sim q_\theta}(x)$ and $\sigma(x)^2 = \frac{1}{N-1} \sum_{i=1}^N \left(f_{w_i \sim q_\theta}(x) - \mu(x)\right)^2$. This is a very promising approach since it allows to associate an uncertainty with the prediction $\mu(x) \pm \sigma(x)$ of the model that depends on the number of data/training points the network has seen in a *neighborhood* of x.

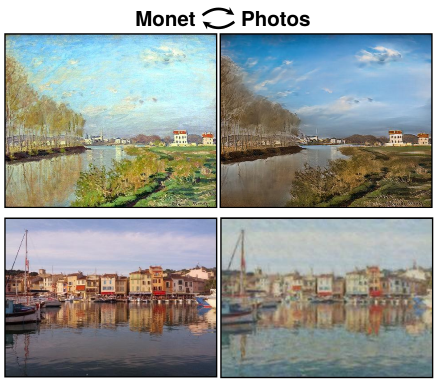
The combination of Bayesian neural networks and generative models (Wasserstein GANs in particular) is therefore very appealing since it allows for the estimation of model uncertainties. This becomes particular important if the generative model is supposed to interpolate to "regions" where training data is rare, non-existent, or even purposely withhold. This is, for instance, the situation in many particle searches in experimental particle physics where control, validation and signal regions are defined. So, the network could be trained to reproduce a control region as good as possible and thereafter interpolate to one of the validation regions. In this case, the Bayesian neural network would assign a higher model uncertainty to the unseen data in the validation/signal region. The uncertainties provided by the neural network should be taken with care since their *physical* interpretation appears to be non-trivial and requires critical reflection. Nevertheless, the combination of model uncertainties and generative adversarial network appears to be very promising and is definitely worth trying. To summarize, Bayesian neural networks are like Sokrates once said: "I know that I know nothing" (Οἶδα οὐδὲν εἰδώς) – except for what I've seen.

Proposal (Bayesian Neural Networks)**.** *Combine generative models (GANs and VAEs) with Bayesian neural networks based on (Gaussian) variational posterior approximation to incorporate uncertainties into model's predictions.*

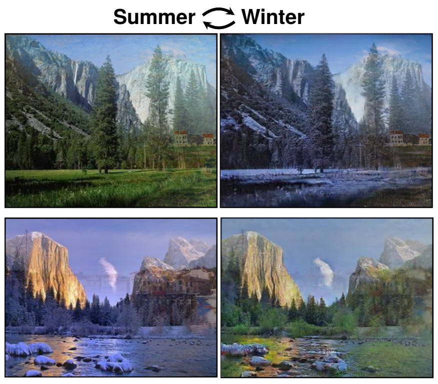
7.2.2 Image-to-image or jet-to-jet translation

The objective of *unpaired* image-to-image translation is to learn a transformation G that allows to map images between different classes or categories X, Y "in the absence of paired [training] examples" (Zhu *et al.* [2017]). This transformation is realized through neural network. The goal is achieved if the transformed data $G(X)$ is indistinguishable from the data in class Y, i.e., $G(X) \approx Y$. Furthermore, the transformation must be *cycle-consistent*:

the map $F : Y \to X$ applied to $G(X)$ must result in the original data $F(G(X)) \approx X$. The crucial aspect, however, is the fact that the method can be trained on *un*-paired examples. This means that the network actually learns the features of the different classes. The paper by Zhu *et al.* [2017] provides some nice examples that shows the performance of the network and illustrates the principle concept. Figure 7.1 shows some translated images for the example of only four classes; furthermore, it demonstrates cycle-consistency.



(a) Monet $\rightleftharpoons$ photo



(b) Summer $\rightleftharpoons$ winter

Fig. 7.1: Image-to-image translation with the condition of cycle-consistency (adapted from Zhu *et al.* [2017] Fig. 1, p. 1).

Since the model has been trained on unpaired examples, the translation from one class to another (e.g. from a Monet painting to the corresponding photo-realistic image) is an extrapolation between features of the respective categories. The next logical step is to translate one image (or data in general) into several classes. To give a simple example, take a model that has been trained on actual photos, the paintings of several artists (Monet, Van Gogh, Cezanne), as well as Ukiyo-e paintings as an example of genre of art with very distinctive features. If the model has been trained on unpaired examples of photos and paintings, the learned function allows to transform a photo to several works of art each one reflecting the art style of the respective artist or genre. A visual example of the aforementioned procedure is given by Figure 7.2

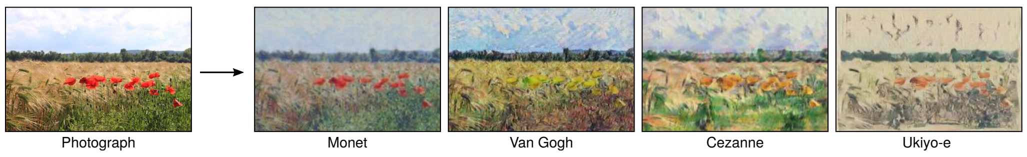


Fig. 7.2: Extrapolation from one class to several classes (adapted from Zhu *et al.* [2017] Fig. 1, p. 1)

He, who is familiar with the period of impressionism or with traditional Japanese art will immediately associate the paintings in Figure 7.2 with the respective artist or genre. Not because he knows the paintings – these can not be found in any museum on earth –, but because he recognizes the unique style that has been learned by the neural network. This method has far more potential than pure entertainment. On the contrary, it provides a possible application in experimental particle physics – especially in the context of particle searches.

As described in 7.2.1, a common strategy in particle searches is to divide the recorded data into several regions (or classes) $\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}$ that are *pairwise disjoint* $\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha} \cap \mathcal{C}^{\mathcal{R}'}_{i_\beta j_\beta \ldots k_\beta} = \emptyset$, i.e., no pairs of classes should – ideally – contain the same event. The indices $i_\alpha j_\alpha \ldots k_\alpha$ are the features of the respective class, e.g. b-tags, region of phase space, mass etc., while $\mathcal{R} \in \{\text{CR}, \text{VR}, \text{SR}\}$ denotes the type of class, i.e., Control Region (CR), Validation Region (VR) and Signal Region (SR). The objective is to get an accurate background prediction for the SR estimated from the CR and validated in the VR. However, to get a prediction of the background in $\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}$ based on the background measured in $\mathcal{C}^{\mathcal{R}'}_{i_\beta j_\beta \ldots k_\beta}$, one needs (loosely speaking) a transformation $TT^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}}$ with $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}}\left(\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}\right) \approx \mathcal{C}^{\mathcal{R}'}_{i_\beta j_\beta \ldots k_\beta}$. The usual procedure is to make assumption regarding the distributions in the individual control regions. It could be argued, for instance, that the shape of the distributions remains unchanged (i.e. the statistical moments are identical up to a global factor) but the number of events differ. Hence, the transformation would be given by a normalization factor. This is just a very simple example and there are far more elaborate methods besides global normalization. However, already this simple example reveals the close connection to the concept of circle-consistent image-to-image translation introduced previously. So, instead of making assumptions regarding the transformation $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}}$, train a neural network with unpaired examples from different classes $\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}$ and learn a continuous map that allows for the extrapolation between different regions through cycle-consistent image-to-image translation, i.e., $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}}\left(\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}\right) \approx \mathcal{C}^{\mathcal{R}'}_{i_\beta j_\beta \ldots k_\beta}$ and $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}} T^{\mathcal{R}'\mathcal{R}}_{i_{\beta\alpha} j_{\beta\alpha} \ldots k_{\beta\alpha}}\left(\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}\right) \approx \mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}$. The combination of cycle-consistent image-to-image translation with Bayesian neural network (see 7.2.1) would furthermore allow to get an extrapolation error from one class into another.

Proposal (Class extrapolation through image-to-image translation)**.** *Given a set of classes $\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}$ associated with region $\mathcal{R} \in \{CR, VR, SR\}$ that are pairwise disjoint $\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha} \cap \mathcal{C}^{\mathcal{R}'}_{i_\beta j_\beta \ldots k_\beta} = \emptyset$ if $\alpha \neq \beta$. Learn a transformation $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}}$ through image-to-image translation utilizing adversarial networks such that $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}}\left(\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}\right) \approx \mathcal{C}^{\mathcal{R}'}_{i_\beta j_\beta \ldots k_\beta}$ and the transformation is cycle consistent, i.e., $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}} T^{\mathcal{R}'\mathcal{R}}_{i_{\beta\alpha} j_{\beta\alpha} \ldots k_{\beta\alpha}}\left(\mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}\right) \approx \mathcal{C}^{\mathcal{R}}_{i_\alpha j_\alpha \ldots k_\alpha}$. Furthermore, combine the generative model with Bayesian neural networks to get extrapolation uncertainties associated with $T^{\mathcal{R}\mathcal{R}'}_{i_{\alpha\beta} j_{\alpha\beta} \ldots k_{\alpha\beta}}$.*

7.2.3 RNNs for higher jet multiplicities

Recurrent neural networks have turned out to be rather inappropriate for modeling the sequential character of the underlying particle shower if this information is not directly extractable from the training data. However, there are plenty other possible applications of RNNs in particle physics.

All results given in this thesis are based on the leading p_T jet of an event; all other jets are ignored for the time being. This is fine within the context of this feasibility study but of course totally inconceivable in a real application. A possible next step, therefore, would be to allow for higher jet multiplicities in an event. In principle this is already possible at this point: given a generative model conditioned on the transverse momentum of the respective jet p_{T}, an event with a jet multiplicity N (itself being a random variable $N \sim \mathbb{P}_N$) can be subsequently generated according to the following factorization:

$$p(\boldsymbol{x}_N, \boldsymbol{x}_{N-1}, \dots, \boldsymbol{x}_1, p_{\mathrm{T}_N}, p_{\mathrm{T}_{N-1}}, \dots p_{\mathrm{T}_1}) \approx \prod_{i=1}^{N} p\left(\boldsymbol{x}_i \middle| \bigcap_{j=1}^{i} p_{\mathrm{T}_j}\right). \tag{7.1}$$

This factorization, however, is only an approximation since it does only account for the correlations between the transverse momentum of the different jets in an event but discards the correlation between the different jet images that would be

$$p(\boldsymbol{x}_k, \dots \boldsymbol{x}_1, p_{\mathrm{T}_k}, \dots, p_{\mathrm{T}_1}) = p\left(\boldsymbol{x}_k \middle| \bigcap_{\substack{i=1\\ i\leq k-1}} (\boldsymbol{x}_i \cap p_{\mathrm{T}_i}) \cap p_{\mathrm{T}_k}\right) \cdot p\left(\bigcap_{\substack{i=1\\ i\leq k-1}} (\boldsymbol{x}_i \cap p_{\mathrm{T}_i}) \cap p_{\mathrm{T}_k}\right). \tag{7.2}$$

Equation 7.2 is exact and should therefore allow the neural network to learn all kinds of effects like, for instance, colour flow between jets [Collaboration, 2018] since the generated jet $\boldsymbol{x}_k$ depends on all previous jets. However, by now the situation is starting to become complicated. To account for the full correlation between all jets in an event, one needs to include the correlation between the conditioning labels *and* to provide the previously generated jet image to the neural network, i.e., the previous state. This, however, is precisely the reason why recurrent neural networks have been invented in the first place. Therefore, a possible application of RNNs combined with generative models would be to learn correlations between different jets in an event. Contrary to the parton shower, the "sequential" information is available on an event-on-event basis; hence, can be used for training. Furthermore, RNNs – unlike feed-forward neural networks – allow for a variable number of inputs and outputs which is essential to simulate different jet multiplicities in an event.

Proposal (Higher jet multiplicities with generative RNNs)**.** *Combine generative models (GANs and VAEs) with Recurrent neural networks to allow for a variable number of in- and outputs to model the distribution of jet multiplicities* $\mathbb{P}_{N^{jet}}$.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 30.08.2019 ....

Bibliography

Georges Aad et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys.Lett.*, B716:1–29, 2012.

F. *et. all* Abe. Topology of three-jet events in $\overline{p}p$ collisions at $\sqrt{s} = 1.8$ tev. *Phys. Rev. D*, 45:1448–1458, Mar 1992.

K. Ackerstaff et al. Production of K0(S) and Lambda in quark and gluon jets from Z0 decay. *Eur. Phys. J.*, C8:241–254, 1999.

Guido Altarelli and G. Parisi. Asymptotic Freedom in Parton Language. *Nucl. Phys.*, B126:298–318, 1977.

Johan Alwall, Michel Herquet, Fabio Maltoni, Olivier Mattelaer, and Tim Stelzer. Madgraph 5 : Going beyond, 2011. arxiv:1106.0522.

J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. 2014.

J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *JHEP*, 07:079, 2014.

Peter Loch and. Jet measurements in ATLAS. *Journal of Physics: Conference Series*, 323:012002, nov 2011.

Bo Andersson, G. Gustafson, G. Ingelman, and T. Sjostrand. Parton Fragmentation and String Dynamics. *Phys. Rept.*, 97:31–145, 1983.

Bo Andersson, G. Gustafson, and B. Soderberg. A General Model for Jet Fragmentation. *Z. Phys.*, C20:317, 1983.

Bo Andersson. THE LUND STRING MODEL. In *7th European Symposium on Antiproton Interactions: From LEAR to the Collider and Beyond Durham, England, July 9-13, 1984*, pages 447–462, 1986.

L Archambault, Beaulieu, et al. Overview of geant4 applications in medical physics. pages 1743 – 1745 Vol.3, 11 2003.

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

Ryan Atkin. Review of jet reconstruction algorithms. *J. Phys. Conf. Ser.*, 645(1):012008, 2015.

Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

M. Bahr et al. Herwig++ Physics and Manual. *Eur. Phys. J.*, C58:639–707, 2008.

B. L. G. Bakker, A. I. Veselov, and M. A. Zubkov. Internal structure of discretized Weinberg-Salam model. *Phys. Lett.*, B583:379–382, 2004.

Dana H. Ballard. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, pages 279–284. AAAI Press, 1987.

D. P. Barber et al. Discovery of Three Jet Events and a Test of Quantum Chromodynamics at PETRA Energies. *Phys. Rev. Lett.*, 43:830, 1979.

Shane Barratt and Rishi Sharma. A note on the inception score, 2018.

Wulfrin Bartel, Lizandra Becker, Rolf Felst, D Haidt, G Knies, H Krehbiel, P Laurikainen, N Magnussen, R Meinke, B Naroska, J Olsson, D Schmidt, G Dietrich, Timothy Greenshaw, J Hagemann, G Heinzelmann, H Kado, C Kleinwort, M Kuhlen, and S Yamada. Experimental studies on multijet production ine+e− annihilation at petra energies. *Zeitschrift für Physik C*, 33:23–31, 03 1986.

Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. *CoRR*, abs/1705.10743, 2017.

Johannes Bellm et al. Herwig 7.0/Herwig++ 3.0 release note. *Eur. Phys. J.*, C76(4):196, 2016.

Christoph Berger et al. Jet Analysis of the Υ (9.46) Decay Into Charged Hadrons. *Phys. Lett.*, 82B:449–455, 1979.

J. Bernstein. *Untersuchungen über den Erregungsvorgang im Nerven- und Muskelsysteme.* Carl Winter's Universitätsbuchhandlung, Heidelberg, 1871.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org, 2015.

F. F. Bonsall. L. v. kantorovich, g. p. akilov functional analysis in normed spaces, translated from the russian by d. e. brown edited by a. p. robertson (pergamon press, 1964), xiii 771 pp., 140s. *Proceedings of the Edinburgh Mathematical Society*, 15(1):80–81, 1966.

E. Boos, V. Bunichev, M. Dubinin, L. Dudko, V. Ilyin, A. Kryukov, V. Edneral, V. Savrin, A. Semenov, and A. Sherstnev. CompHEP 4.4: Automatic computations from Lagrangians to events. *Nucl. Instrum. Meth.*, A534:250–259, 2004.

V. Bornyakov et al. Heavy quark potential in lattice QCD at finite temperature. In *Quark confinement and the hadron spectrum. Proceedings, 5th International Conference, Gargnano, Italy, September 10-14, 2002*, pages 294–296, 2003.

A Training Algorithm for Optimal Margin Classifiers, COLT '92, New York, NY, USA, 1992. ACM.

Becker Bothe. Künstliche erregung von kern-γ-strahlen. *Z. Phys.*, 66:289, 1930.

R. Brandelik et al. Evidence for Planar Events in e+ e- Annihilation at High-Energies. *Phys. Lett.*, 86B:243–249, 1979.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018.

Andy Buckley et al. General-purpose event generators for LHC physics. *Phys. Rept.*, 504:145–233, 2011.

Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_t jet clustering algorithm. *JHEP*, 04:063, 2008.

Matteo Cacciari, Gavin P Salam, and Gregory Soyez. FastJet user manual. *Eur. Phys. J. C*, 72(arXiv:1111.6097. CERN-PH-TH-2011-297):1896. 69 p, Nov 2011. Comments: 69 pages. FastJet 3 is available from http://fastjet.fr/.

Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet user manual. *Eur.Phys.J.*, C72:1896, 2012.

Claudio Campagnari and Melissa Franklin. The Discovery of the top quark. *Rev. Mod. Phys.*, 69:137–212, 1997.

S. Catani, Yuri L. Dokshitzer, M. H. Seymour, and B. R. Webber. Longitudinally invariant K_t clustering algorithms for hadron hadron collisions. *Nucl. Phys.*, B406:187–224, 1993.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *CoRR*, abs/1611.02731, 2016.

Yang-Ting Chien and Ivan Vitev. Towards the understanding of jet shapes and cross sections in heavy ion collisions using soft-collinear effective theory. *JHEP*, 05:023, 2016.

Vincenzo Chiochia, Günther. Dissertori, and Thomas Gehrmann. Lecture notes in phenomenology of particle physics i, September 2010.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

On the histogram as a density estimator:l2 theory. *Probability Theory and Related Fields*, 57(4):453–476, 1981.

ATLAS Collaboration. Measurement of colour flow using jet-pull observables in $t\bar{t}$ events with the atlas experiment at $\sqrt{s} = 13$ tev. 2018.

John C. Collins. Sudakov form factors. 2003.

G. Cybenko. Approximation by superpositions of a sigmoidal function, 1989.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. 2018.

Luke de Oliveira, Michela Paganini, and Benjamin Nachman. Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis. *Comput. Softw. Big Sci.*, 1(1):4, 2017.

Yuri L. Dokshitzer. Calculation of the Structure Functions for Deep Inelastic Scattering and e+ e- Annihilation by Perturbation Theory in Quantum Chromodynamics. *Sov. Phys. JETP*, 46:641–653, 1977. [Zh. Eksp. Teor. Fiz.73,1216(1977)].

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

S. Eidelman, K.G. Hayes, K.A. Olive, M. Aguilar-Benitez, and journal = "Physics Letters B" year = 2004 volume = "592" pages = 1+ url = http://pdg.lbl.gov others, title = "Review of Particle Physics".

Albert Einstein. On the electrodynamics of moving bodies. *Annalen Phys.*, 17:891–921, 1905. [Annalen Phys.14,194(2005)].

A. Einstein. Die Grundlage der allgemeinen Relativitätstheorie. *Annalen der Physik*, 354:769–822, 1916.

Stephen D. Ellis and Davison E. Soper. Successive combination jet algorithm for hadron collisions. *Phys. Rev.*, D48:3160–3166, 1993.

Stephen D. Ellis, Zoltan Kunszt, and Davison E. Soper. Jets at hadron colliders at order $\alpha - s^{3:}$ A Look inside. *Phys. Rev. Lett.*, 69:3615–3618, 1992.

John R. Ellis. Limits of the standard model. In *PSI Zuoz Summer School on Exploring the Limits of the Standard Model Zuoz, Engadin, Switzerland, August 18-24, 2002*, 2002.

F. Englert and R. Brout. Broken Symmetry and the Mass of Gauge Vector Mesons. *Phys. Rev. Lett.*, 13:321–323, 1964. [,157(1964)].

W. Feller. The fundamental limit theorems in probability. *Bull. Amer. Math. Soc.*, 51(11):800–832, 11 1945.

Enrico Fermi. Tentativo di una teoria dell'emissione dei raggi beta. *Ric. Sci.*, 4:491–495, 1933.

R. P. Feynman. Space-time approach to quantum electrodynamics. *Phys. Rev.*, 76:769–789, Sep 1949.

Nobel Foundation. The nobel prize in physics, 1965.

Nobel Foundation. The nobel prize in physics, 2004.

Rikkert Frederix and Stefano Frixione. Merging meets matching in MC@NLO. *JHEP*, 12:061, 2012.

Rikkert Frederix, Stefano Frixione, Fabio Maltoni, and Tim Stelzer. Automation of next-to-leading order computations in qcd: the fks subtraction. 2009.

Stefano Frixione, Fabian Stoeckli, Paolo Torrielli, Bryan R. Webber, and Chris D. White. The mc@nlo 4.0 event generator, 2010.

L. Galvani. De viribus electricitatis in motu musculari: commentarius. 1791.

M. Gell-Mann. The eightfold way: A theory of strong interaction symmetry. 3 1961.

Murray Gell-Mann. A Schematic Model of Baryons and Mesons. *Phys. Lett.*, 8:214–215, 1964.

Claudia Gemme. Latest ATLAS results from Run 2. Technical Report arXiv:1612.01987, CERN, Geneva, Dec 2016. 9th International Worshop on top quark physics Olomouc, Czech Republic, September 19–23, 2016.

Felix A. Gers and Juergen Schmidhuber. Recurrent nets that time and count. Technical report, 2000.

Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471, 2000.

S. L. Glashow. Partial Symmetries of Weak Interactions. *Nucl. Phys.*, 22:579–588, 1961.

Tanju Gleisberg and Stefan Hoeche. Comix, a new matrix element generator. *JHEP*, 12:039, 2008.

T. Gleisberg, S. Hoeche, F. Krauss, M. Schoenherr, S. Schumann, F. Siegert, and J. Winter. Event generation with sherpa 1.1. 2008.

E. W. Nigel Glover and David A. Kosower. Recombination methods for jets in p anti-p collisions. *Phys. Lett.*, B367:369–376, 1996.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

V. N. Gribov and L. N. Lipatov. Deep inelastic e p scattering in perturbation theory. *Sov. J. Nucl. Phys.*, 15:438–450, 1972. [Yad. Fiz.15,781(1972)].

David J. Gross and Frank Wilczek. Ultraviolet behavior of non-abelian gauge theories. *Phys. Rev. Lett.*, 30:1343–1346, Jun 1973.

Martin W. Grunewald. Precision tests of the standard model. *PoS*, HEP2005:306, 2006.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble. Global Conservation Laws and Massless Particles. *Phys. Rev. Lett.*, 13:585–587, 1964. [,162(1964)].

D. Hanneke, S. Fogwell, and G. Gabrielse. New measurement of the electron magnetic moment and the fine structure constant. *Physical Review Letters*, 100(12), 3 2008.

Stephan Hartmann. James t. cushing, philosophical concepts in physics. the historical relation between philosophy and scientific theories. *Erkenntnis*, 52:133–137, 01 2000.

F. J. Hasert et al. Observation of Neutrino Like Interactions Without Muon Or Electron in the Gargamelle Neutrino Experiment. *Phys. Lett.*, B46:138–140, 1973. [,5.15(1973)].

F J Hasert, Helmut Faissner, W Krenz, J Von Krogh, and D Lanske. Search for elastic muon neutrino electron scattering. *Phys. Lett. B*, 46:121–124, 1973.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Donald O. Hebb. *The organization of behavior: A neuropsychological theory.* Wiley, New York, June 1949.

Peter W. Higgs. Broken Symmetries and the Masses of Gauge Bosons. *Phys. Rev. Lett.*, 13:508–509, 1964. [,160(1964)].

Valentin Hirschi, Rikkert Frederix, Stefano Frixione, Maria Vittoria Garzelli, Fabio Maltoni, and Roberto Pittau. Automation of one-loop QCD corrections. *JHEP*, 05:044, 2011.

Valentin Hirschi. New developments in MadLoop. 2011. [PoSRADCOR2011,018(2011)].

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

J. J. Hopfield and D. W. Tank. "neural" computation of decisions in optimization problems. *Biol. Cybern.*, 52(3):141–152, July 1985.

J. J. Hopfield. Neurocomputing: Foundations of research. chapter Neural Networks and Physical Systems with Emergent Collective Computational Abilities, pages 457–464. MIT Press, Cambridge, MA, USA, 1988.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Cuthbert C. Hurd. A note on early Monte Carlo computations and scientific meetings. 7(2):141–155, April/June 1985. Includes typeset reprint of Richtmyer *et al.* [1947].

John E. Huth et al. Toward a standardization of jet definitions. In *1990 DPF Summer Study on High-energy Physics: Research Directions for the Decade (Snowmass 90) Snowmass, Colorado, June 25-July 13, 1990*, pages 0134–136, 1990.

Stefan Höche. Introduction to parton-shower event generators, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

Arthur M. Jaffe and Edward Witten. Quantum Yang-Mills theory. 2000.

L.V. Kantorovich and G.Sh.Rubinstein. On the space of completely additive functions. *Vestnic Leningrad Univ., Ser. Mat. Mekh. i Astron.*, 13(7):52–59, 1958. In Russian.

Gregor Kasieczka, Tilman Plehn, Michael Russell, and Torben Schell. Deep-learning Top Taggers or The End of QCD? *JHEP*, 05:006, 2017.

Eamonn Keogh and Abdullah Mueen. *Curse of Dimensionality*, pages 314–315. Springer US, Boston, MA, 2017.

Vardan Khachatryan et al. Measurement of the inclusive 3-jet production differential cross section in proton–proton collisions at 7 TeV and determination of the strong coupling constant in the TeV range. *Eur. Phys. J.*, C75(5):186, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Diederik P. Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016.

H.A. Kramers. *Die grundlagen der quantentheorie: Quantentheorie des elektrons und der strahlung*. Number Bd. 2 in Die grundlagen der quantentheorie: Quantentheorie des elektrons und der strahlung. Akademische verlagsgesellschaft m.b.h., 1938.

David Kriesel. *A Brief Introduction to Neural Networks*. 2007.

Thomas S. Kuhn. *The structure of scientific revolutions*. University of Chicago Press, Chicago, 1970.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

Yann Lecun. *Generalization and network design strategies*. Elsevier, 1989.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: towards deeper understanding of moment matching network. *CoRR*, abs/1705.08584, 2017.

Yuxi Li. Deep reinforcement learning: An overview. *CoRR*, abs/1701.07274, 2017.

Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's thesis, Univ. Helsinki, 1970.

Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

Leif Lönnblad. : A program for simulation of QCD cas-cades implementing the colour dipole mode. pages 15–31, 1992.

Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, May 1992.

Michelangelo L. Mangano, Mauro Moretti, and Roberto Pittau. Multijet matrix elements and shower evolution in hadronic collisions: $Wb\bar{b} + n$ jets as a case study. *Nucl. Phys.*, B632:343–362, 2002.

Michelangelo L. Mangano, Mauro Moretti, Fulvio Piccinini, Roberto Pittau, and Antonio D. Polosa. ALPGEN, a generator for hard multiparton processes in hadronic collisions. *JHEP*, 07:001, 2003.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

James Clerk Maxwell. A dynamical theory of the electromagnetic field. *Philosophical Transactions of the Royal Society of London*, 155:459–513, 1865.

Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

N. Metropolis and William Aspray. Oral history interview with Nicholas Metropolis. Audio recording, 1987. The Charles Babbage Institute.

Nicholas Metropolis and S. Ulam. The Monte Carlo method. 44(247):335–341, September 1949.

Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *CoRR*, abs/1802.05637, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018.

Youssef Mroueh and Tom Sercu. Fisher GAN. *CoRR*, abs/1705.09675, 2017.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

Yurii Nesterov. Random gradient-free minimization of convex functions. CORE Discussion Papers 2011001, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011.

I. Newton, A. Motte, and J. Machin. *The Mathematical Principles of Natural Philosophy*. Number Bd. 1 in The Mathematical Principles of Natural Philosophy. B. Motte, 1729.

XuanLong Nguyen, Martin J Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1089–1096. Curran Associates, Inc., 2008.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 271–279. Curran Associates, Inc., 2016.

Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018.

A. Pukhov, E. Boos, M. Dubinin, V. Edneral, V. Ilyin, D. Kovalenko, A. Kryukov, V. Savrin, S. Shichanin, and A. Semenov. CompHEP: A Package for evaluation of Feynman diagrams and integration over multiparticle phase space. 1999.

A. Pukhov. CalcHEP 2.3: MSSM, structure functions, event generation, batchs, and generation of matrix elements for other packages. 2004.

J. Pumplin, D. R. Stump, J. Huston, H. L. Lai, Pavel M. Nadolsky, and W. K. Tung. New generation of parton distributions with uncertainties from global QCD analysis. *JHEP*, 07:012, 2002.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Netw.*, 12(1):145–151, January 1999.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. cite arxiv:1511.06434Comment: Under review as a conference paper at ICLR 2016.

Santiago Ramón y Cajal. Revista trimestral de histología normal y patológica, año 1, n. 1. 1888.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *CoRR*, abs/1904.09237, 2019.

Mark D. Reid and Robert C. Williamson. Information, divergence and risk for binary experiments. *J. Mach. Learn. Res.*, 12:731–817, July 2011.

Robert D. Richtmyer, Stanisław Ulam, and John von Neumann. Statistical methods in neutron diffusion. Technical Report LAMS-551, April 1947. Republished in typeset form in Hurd [1985].

F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.

Kevin Roth, Aurélien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. *CoRR*, abs/1705.09367, 2017.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–, October 1986.

Gavin P. Salam and Gregory Soyez. A Practical Seedless Infrared-Safe Cone jet algorithm. *JHEP*, 05:086, 2007.

Abdus Salam and John Clive Ward. Electromagnetic and weak interactions. *Phys. Lett.*, 13:168–171, 1964.

Abdus Salam. Weak and Electromagnetic Interactions. *Conf. Proc.*, C680519:367–377, 1968.

Gavin P. Salam. Towards Jetography. *Eur. Phys. J.*, C67:637–686, 2010.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681, 1997.

Julian S. Schwinger. A Theory of the Fundamental Interactions. *Annals Phys.*, 2:407–434, 1957.

M. H. Seymour. Jet shapes in hadron collisions: Higher orders, resummation and hadronization. *Nucl. Phys.*, B513:269–300, 1998.

Michael H. Seymour. Jets in hadron collisions. In *Deep inelastic scattering. Proceedings, 8th International Workshop, DIS 2000, Liverpool, UK, April 25-30, 2000*, pages 27–41, 2000. [,21(2000)].

Hang Shao, Abhishek Kumar, and P. Thomas Fletcher. The riemannian geometry of deep generative models. *CoRR*, abs/1711.08014, 2017.

Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. PYTHIA 6.4 Physics and Manual. *JHEP*, 05:026, 2006.

Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. A Brief Introduction to PYTHIA 8.1. *Comput. Phys. Commun.*, 178:852–867, 2008.

Torbjörn Sjöstrand. A model for initial state parton showers. *Physics Letters B*, 157(4):321 – 325, 1985.

J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

T. Stelzer and W. F. Long. Automatic generation of tree level helicity amplitudes. 1994.

George F. Sterman and Steven Weinberg. Jets from Quantum Chromodynamics. *Phys. Rev. Lett.*, 39:1436, 1977.

Iain W. Stewart, Frank J. Tackmann, and Wouter J. Waalewijn. N-Jettiness: An Inclusive Event Shape to Veto Jets. *Phys. Rev. Lett.*, 105:092002, 2010.

Jesse Thaler and Ken Van Tilburg. Identifying Boosted Objects with N-subjettiness. *JHEP*, 03:015, 2011.

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

W. von Waldeyer-Hartz. *Ueber einige neuere Forschungen im Gebiete der Anatomie des Centralnervensystems.* Wochenschrift, Deutsche Medicinische 1891, No. 44 u. ff. Georg Thieme, 1891.

Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Machine Learning*, 8:279–292, 1992.

P. M. Watkins. DISCOVERY OF THE W AND Z BOSONS. *Contemp. Phys.*, 27:291–324, 1986.

B. Webber. Parton shower Monte Carlo event generators. *Scholarpedia*, 6(12):10662, 2011. revision #128236.

Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. Improving the improved training of wasserstein gans: A consistency term and its dual effect. *CoRR*, abs/1803.01541, 2018.

Steven Weinberg. A model of leptons. *Phys. Rev. Lett.*, 19:1264–1266, Nov 1967.

Steven Weinberg. The Making of the standard model. *Eur. Phys. J.*, C34:5–13, 2004. [,99(2005)].

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

M. Wobisch and T. Wengler. Hadronization corrections to jet cross-sections in deep inelastic scattering. In *Monte Carlo generators for HERA physics. Proceedings, Workshop, Hamburg, Germany, 1998-1999*, pages 270–279, 1998.

Yuxin Wu and Kaiming He. Group normalization. *CoRR*, abs/1803.08494, 2018.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

Chen-Ning Yang and Robert L. Mills. Conservation of Isotopic Spin and Isotopic Gauge Invariance. *Phys. Rev.*, 96:191–195, 1954. [,150(1954)].

Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *ArXiv*, abs/1703.10847, 2017.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. Depth-gated lstm, 2015.

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial network. *CoRR*, abs/1609.03126, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.

G. Zweig. An SU(3) model for strong interaction symmetry and its breaking. Version 2. In D.B. Lichtenberg and Simon Peter Rosen, editors, *DEVELOPMENTS IN THE QUARK THEORY OF HADRONS. VOL. 1. 1964 - 1978*, pages 22–101. 1964.