Introduction to Machine Learning

Elke Kirschbaum

Image Analysis and Learning Lab (IAL)
Interdisciplinary Center for Scientific Computing (IWR)
Heidelberg University

Overview

Part I:

Introduction to Machine Learning and Deep Learning

Part II:

Unsupervised Deep Learning Models

Part I: Introduction to Machine Learning and Deep Learning

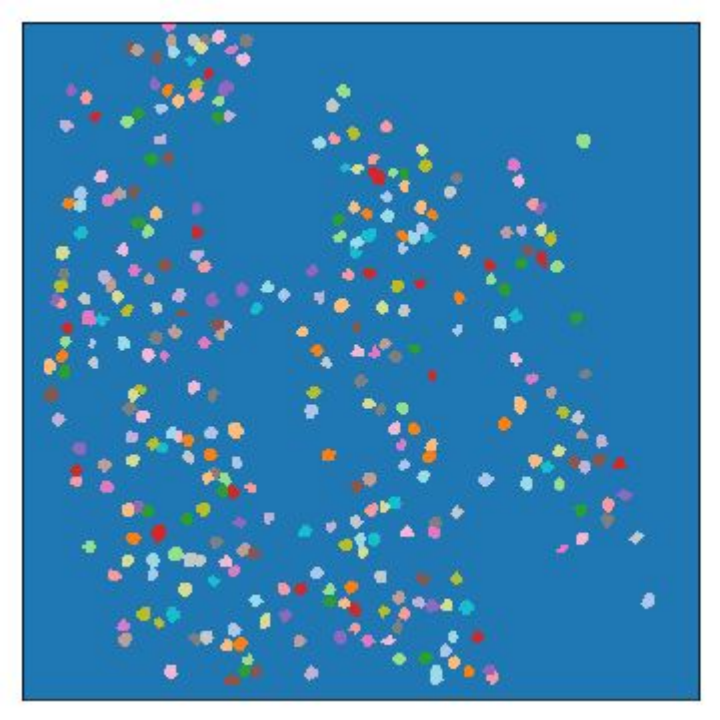
1. About Me and the IAL Group
2. And What About You?
3. What is Machine Learning?
4. What is Deep Learning?
   a. What is a Neural Network?
   b. Common Architectures and Loss Functions
   c. Applications in Particle Physics
   d. Tools for Deep Learning

About Me and the IAL Group

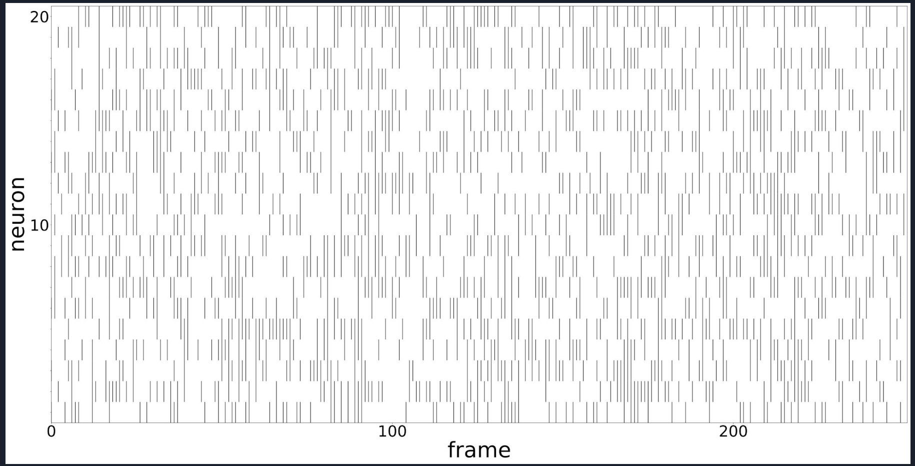
About me

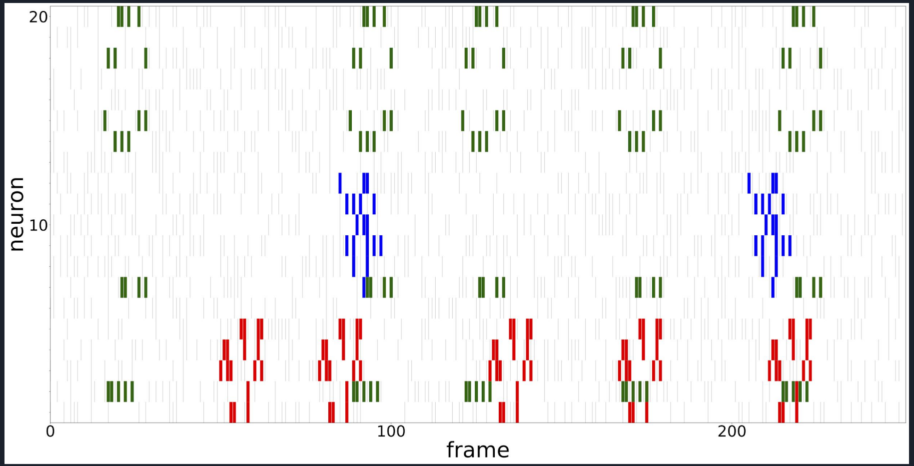
- studied Physics in Tübingen
- since 2015: PhD student in Image Analysis and Learning (IAL) group
- developing machine learning algorithms to analyse neurological data

We do...
Cell Segmentation from Videos

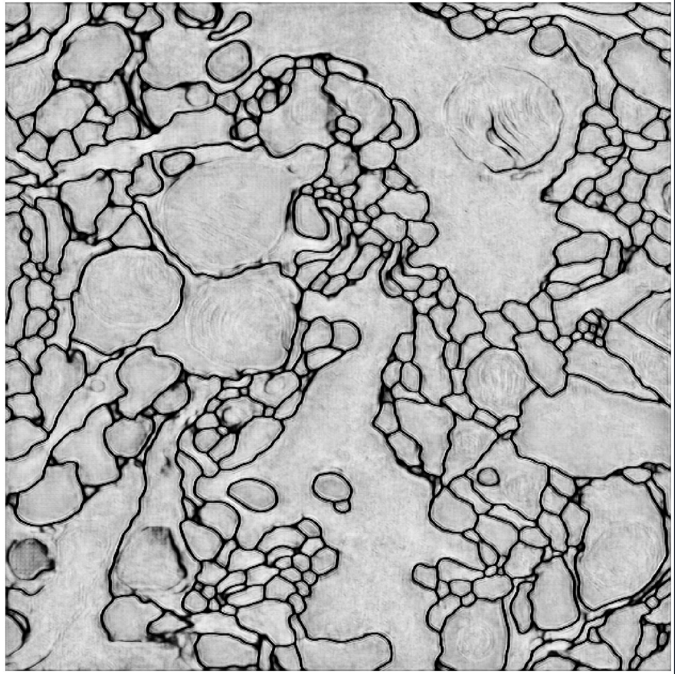


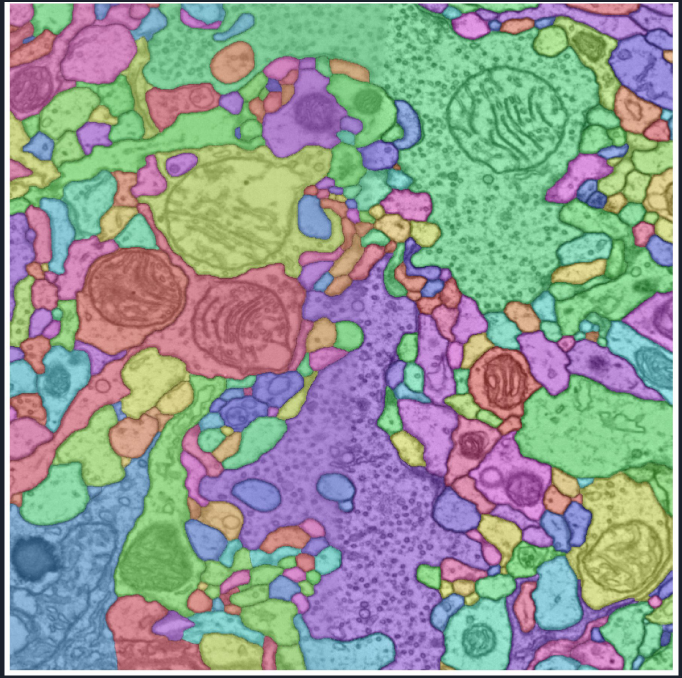
We do…
Motif Detection





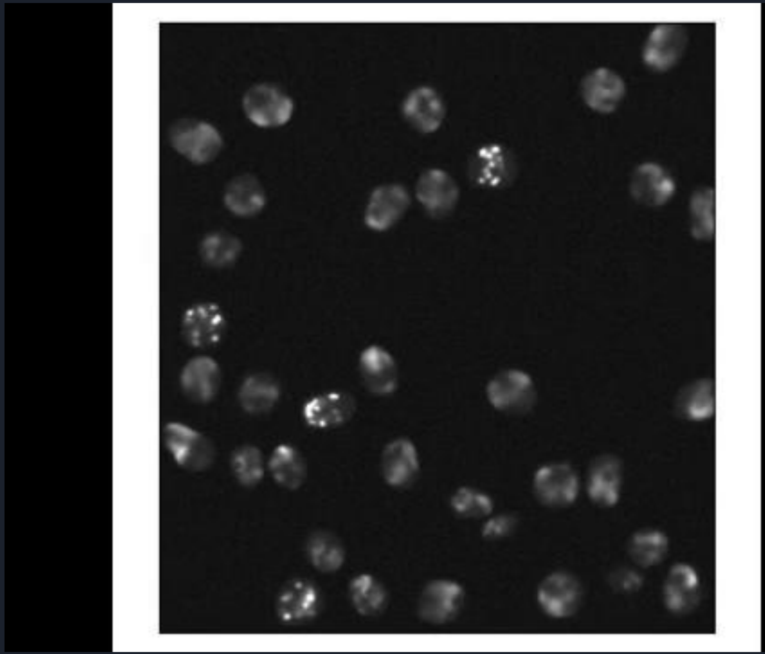
We do...
Neuron Segmentation

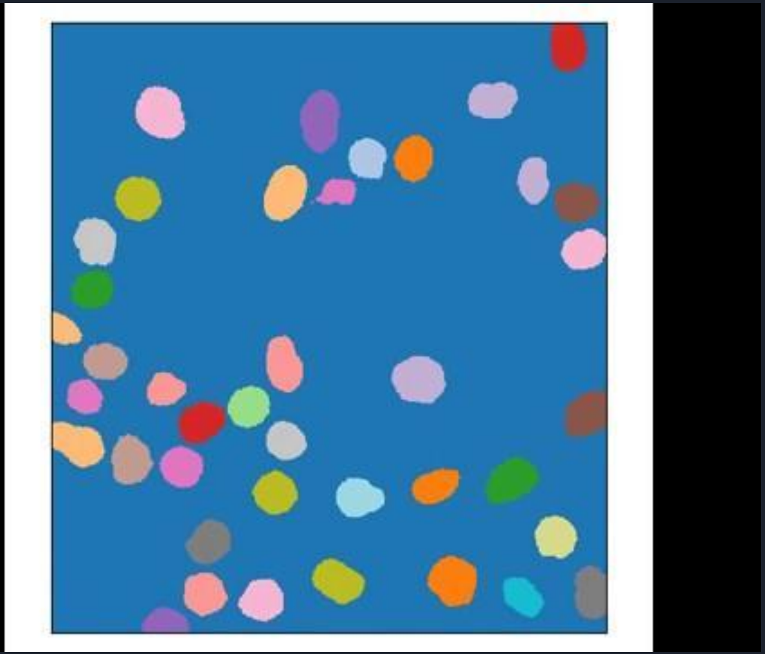




Figures taken from Cerrone et al. (2019), “End-to-End Learned Random Walker for Seeded Image Segmentation”

We do...
Cell Tracking

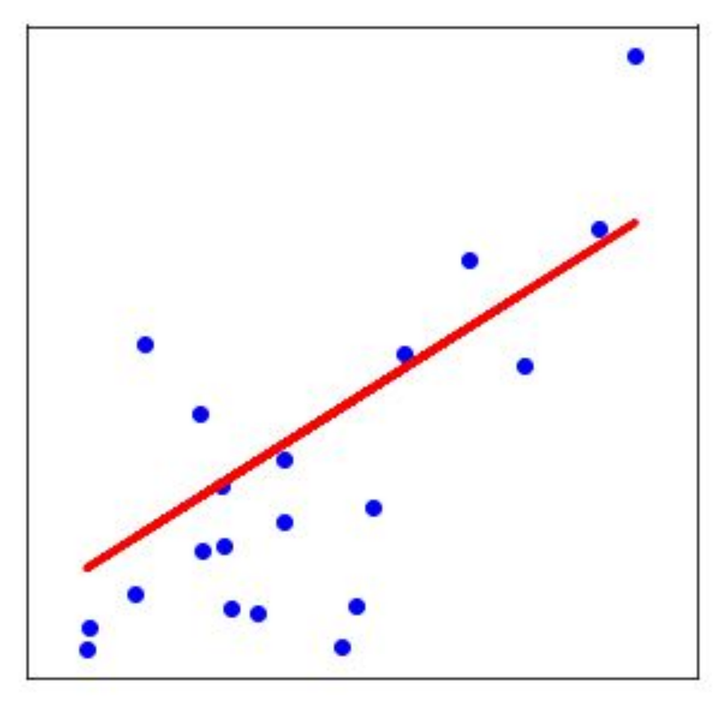




And What About You?

What is Machine Learning?

Machine Learning is...



Regression

find parameters *a* and *b* such that

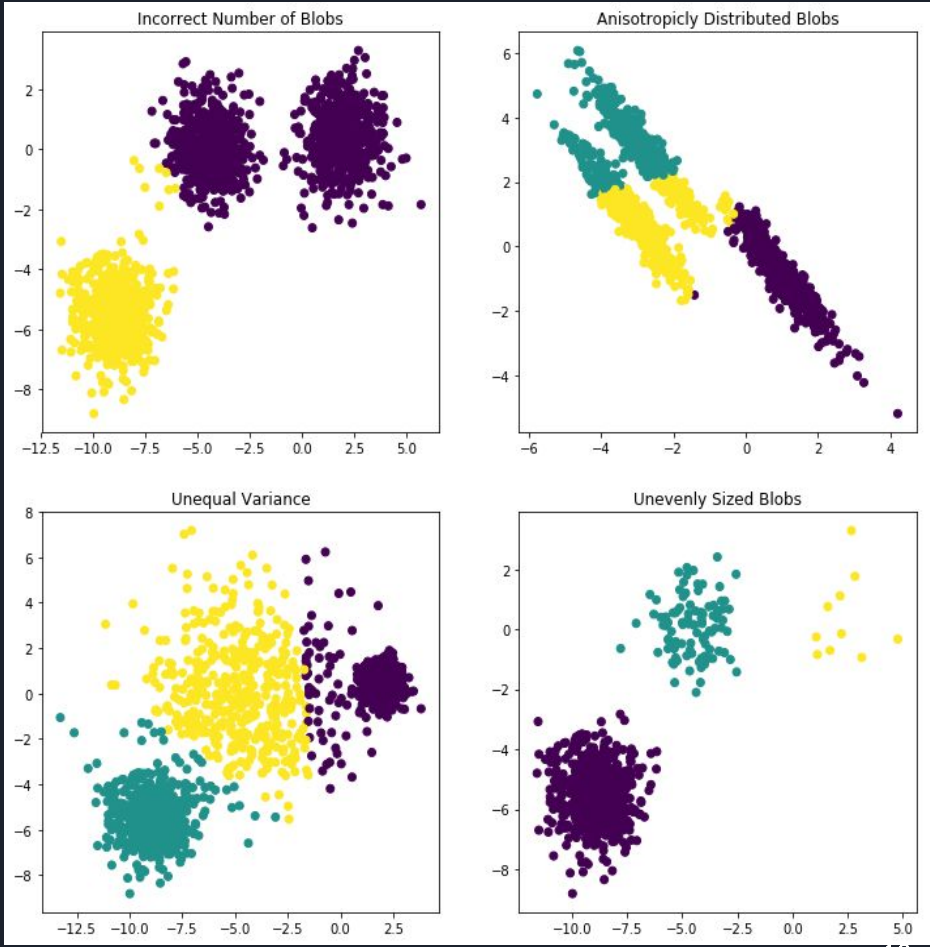
$$a, b = \arg\min_{a,b} \sum_i \|y_i - (a \cdot x_i + b)\|^2$$

Machine Learning is...

Clustering

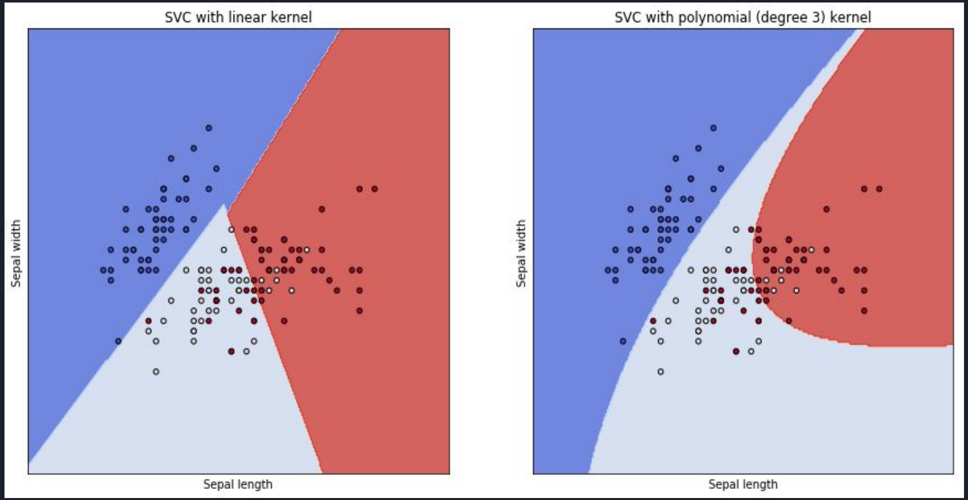
partition the n observations into k sets S to minimize the within-cluster sum of squares

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$



Machine Learning is...

Classification



Soft-margin SVM:

$$\min_{b,\vec{w}} \left[\frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)) + \lambda \|\vec{w}\|^2 \right]$$

Machine Learning is...

Classification

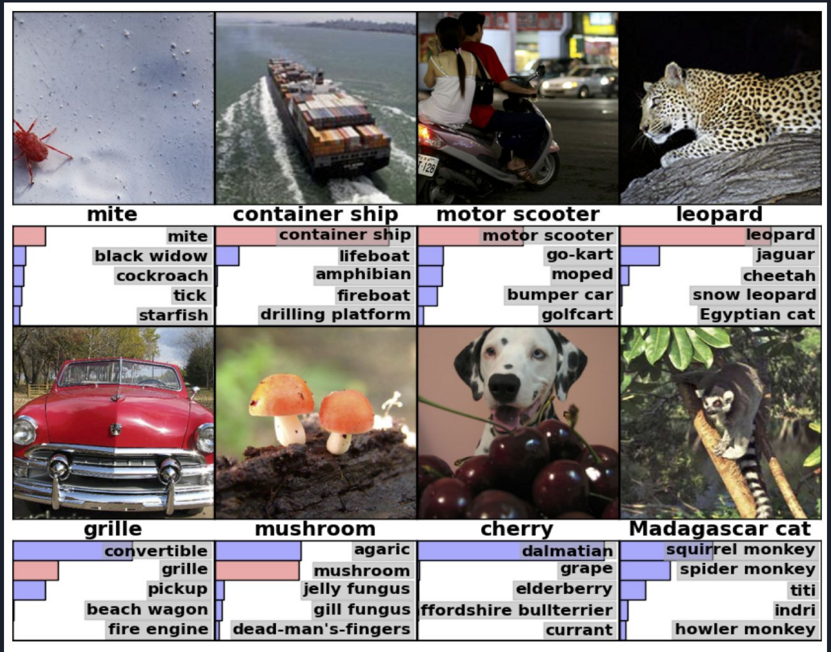


Figure taken from Krizhevsky et al. (2012), “ImageNet Classification with Deep Convolutional Neural Networks”

Machine Learning is... Segmentation

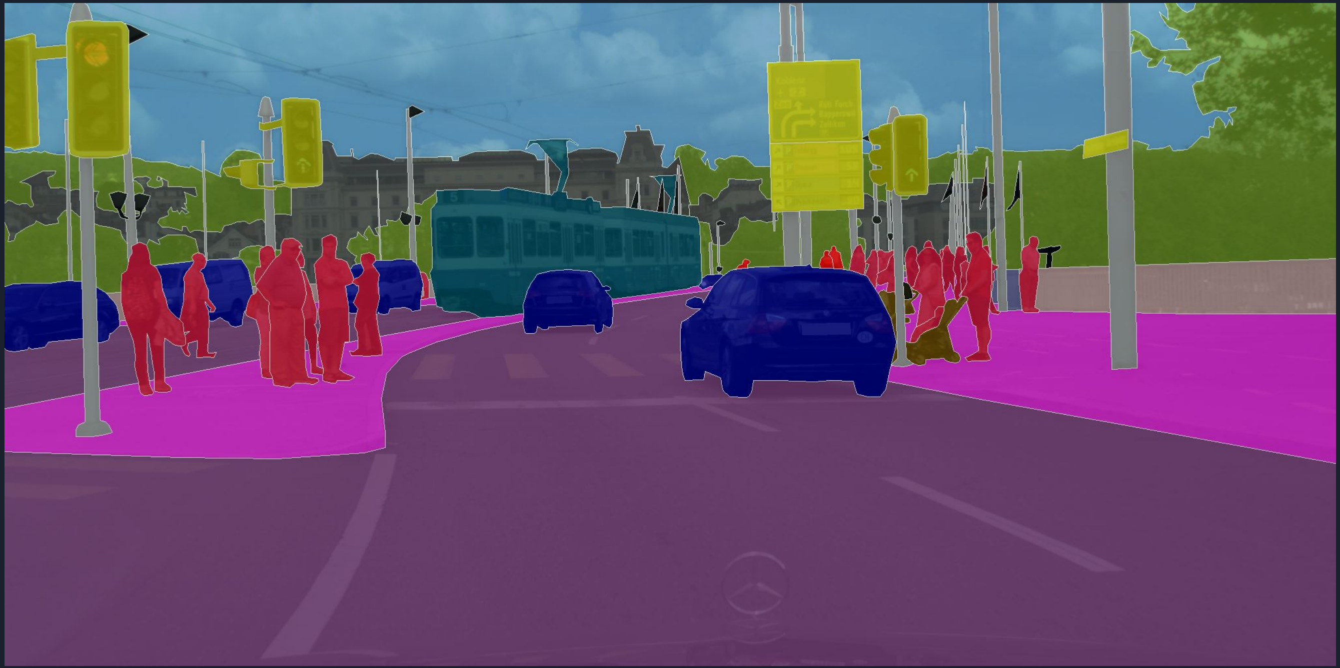


Figure taken from https://www.cityscapes-dataset.com/examples/

Machine Learning is...

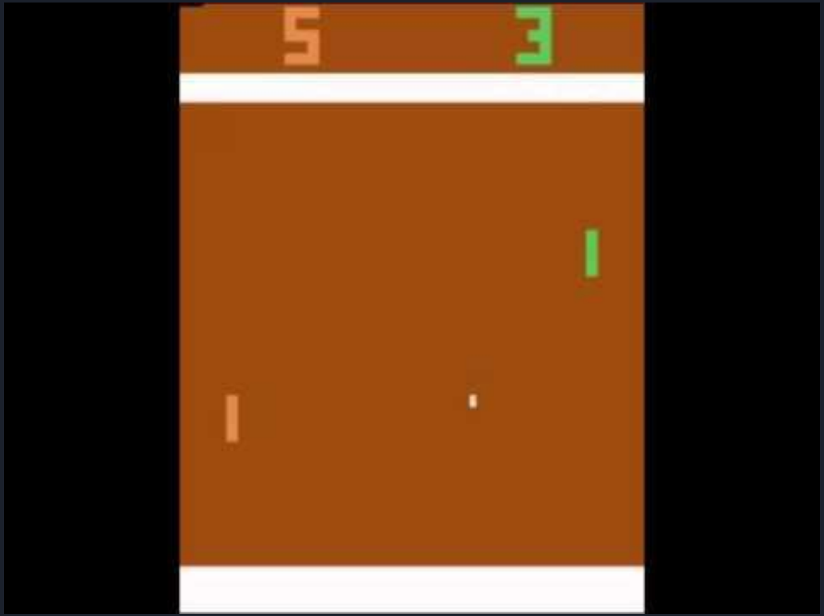
- “a computer program learning from experience”
- algorithms that build a mathematical model based on training data to make predictions or decisions
- model parameters are usually learned by finding an (approximate) solution to an optimization problem
- name was coined in 1959

Machine Learning includes...

- Supervised Learning
  - training data = input data + output labels
  - e.g. SVM, linear regression
- Unsupervised Learning
  - training data = only input data
  - e.g. clustering, feature learning, dimensionality reduction, pattern or anomaly detection
- Semi-Supervised Learning
  - only parts of the training data include output labels

Machine Learning includes...

- Active Learning
  - training labels for a limited input set are accessed based on a budget
  - choice of inputs for which labels are used is learned
  - can be trained interactively with a human labeller
- Reinforcement Learning
  - actions get positive or negative feedback
  - goal is to maximize reward



Why is Machine Learning suddenly so “hot”?

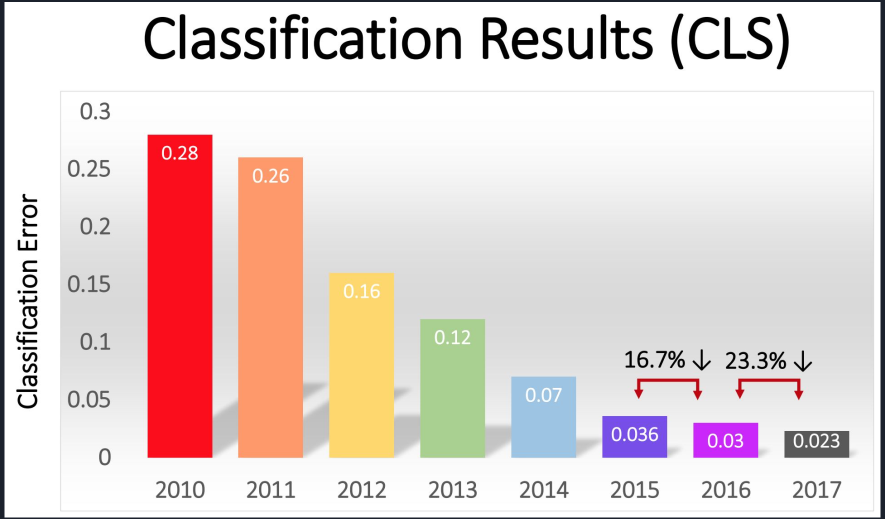


Figure taken from https://arstechnica.com/science/2018/12/how-computers-got-shockingly-good-at-recognizing-images/

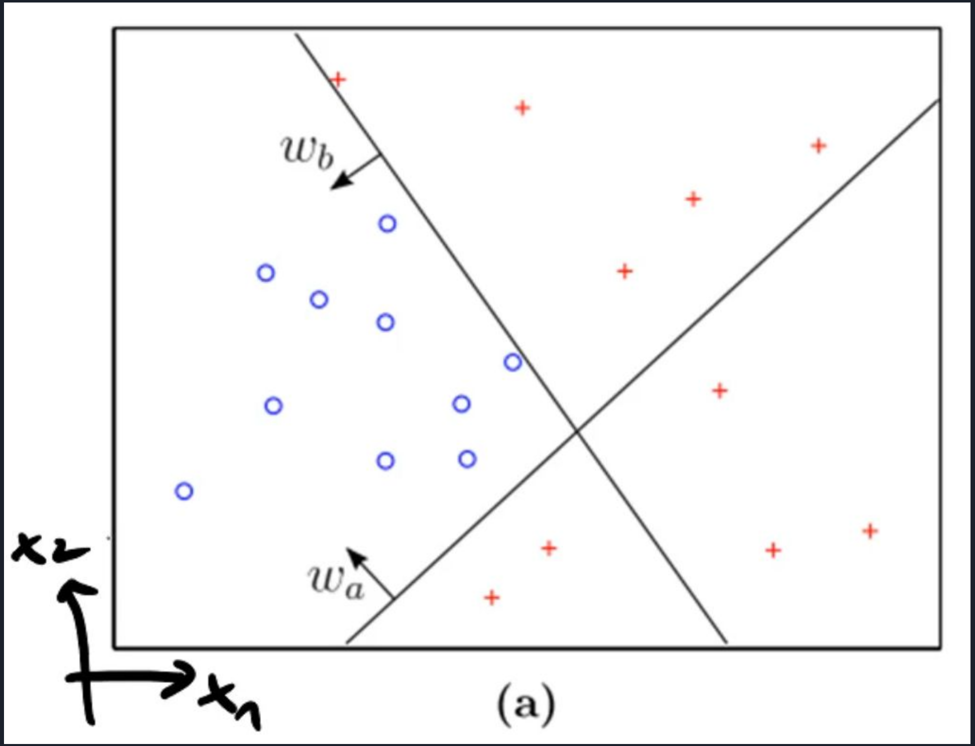
What is Deep Learning?

Deep Learning is...

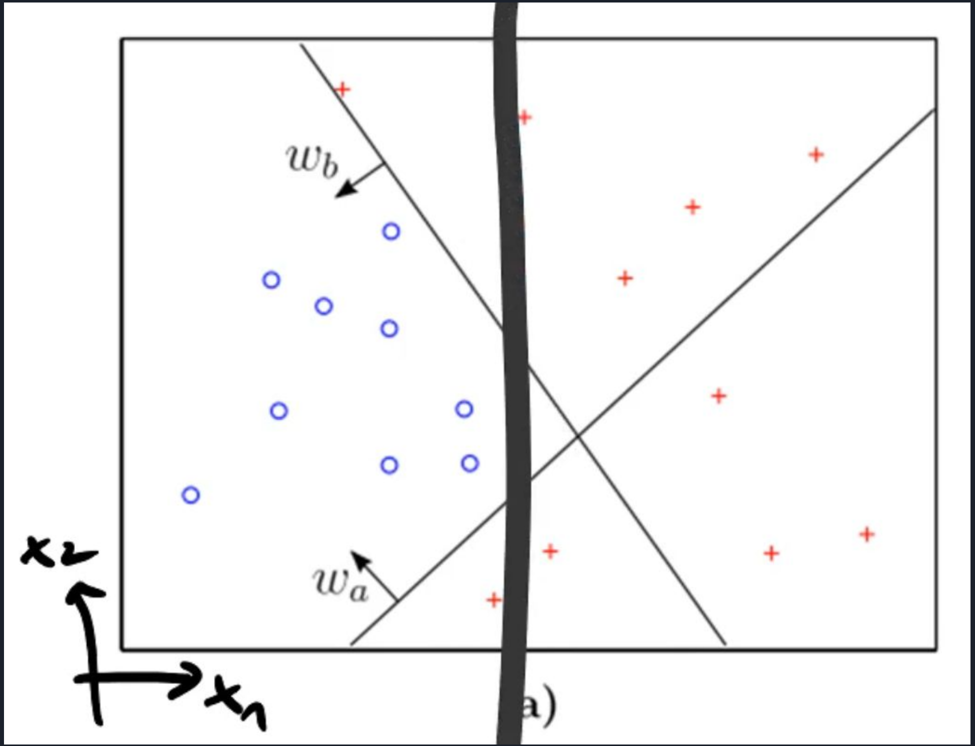
- a specific class of machine learning models
- widely applicable to a huge range of different datasets and tasks
  - computer vision
  - natural language processing
  - speech recognition
  - machine translation
  - bio and medical image analysis
  - material inspection
- machine learning with deep neural networks

What is a Neural Network?

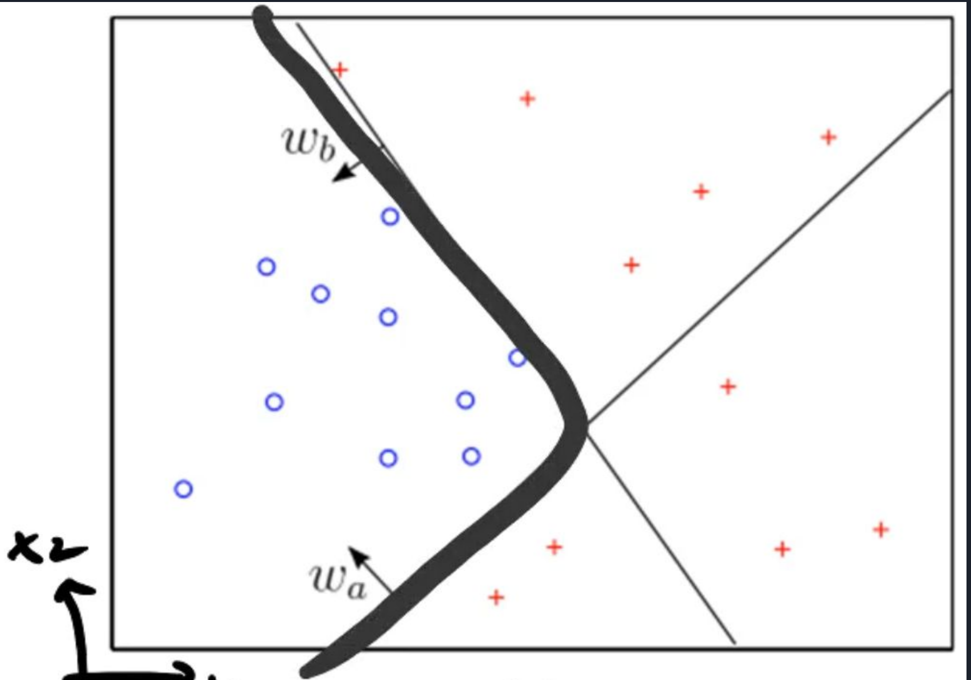
Example: Classification in 2D



Linear Decision Boundary

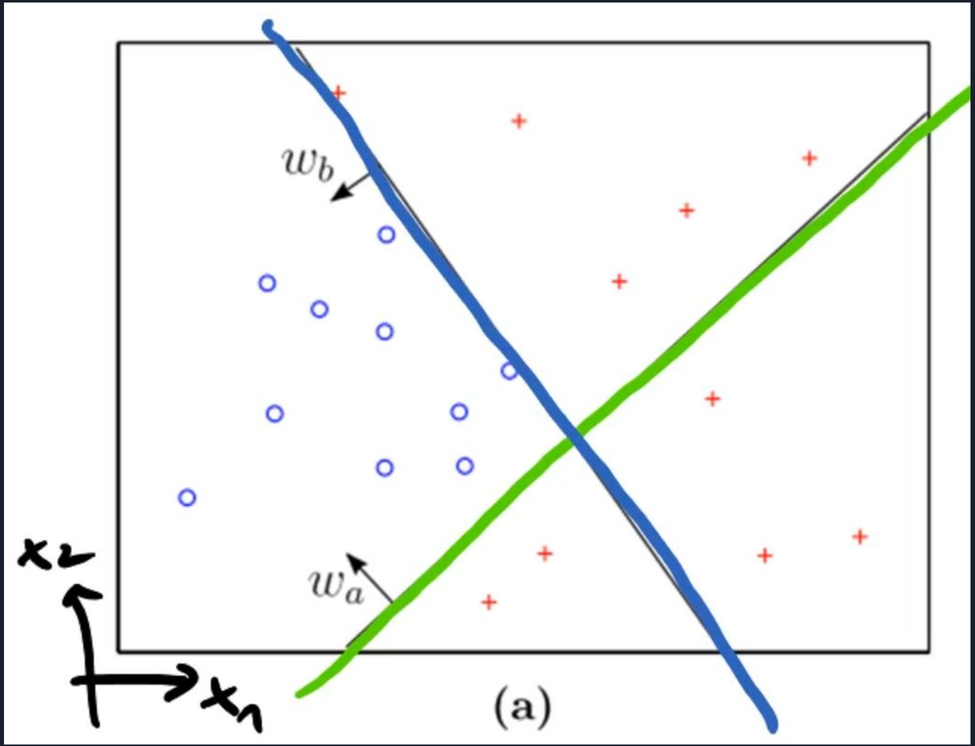


Non-Linear Decision Boundary

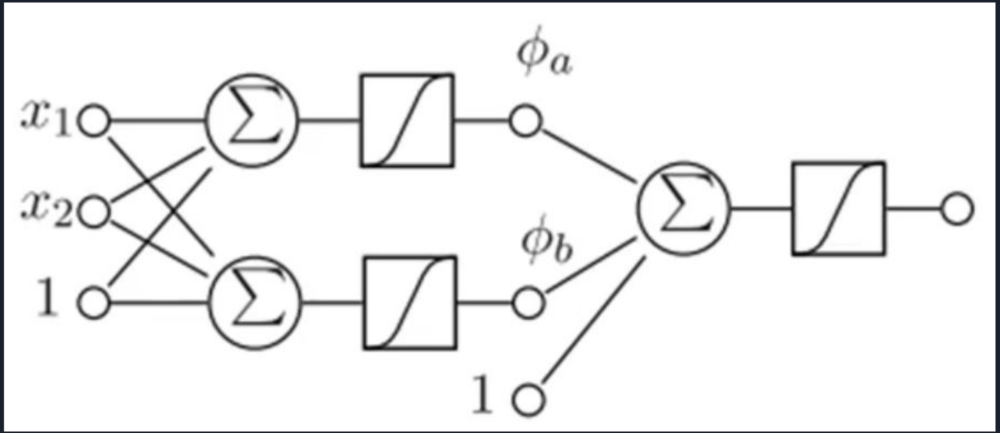


(a)

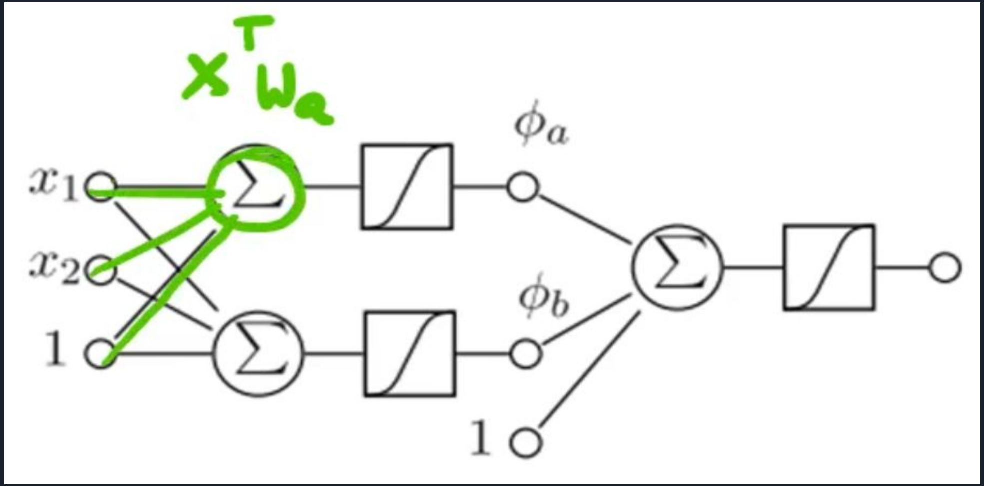
Non-Linear Mapping of the Data + Linear Decision Boundaries



Multi-Layer Perceptron (MLP)

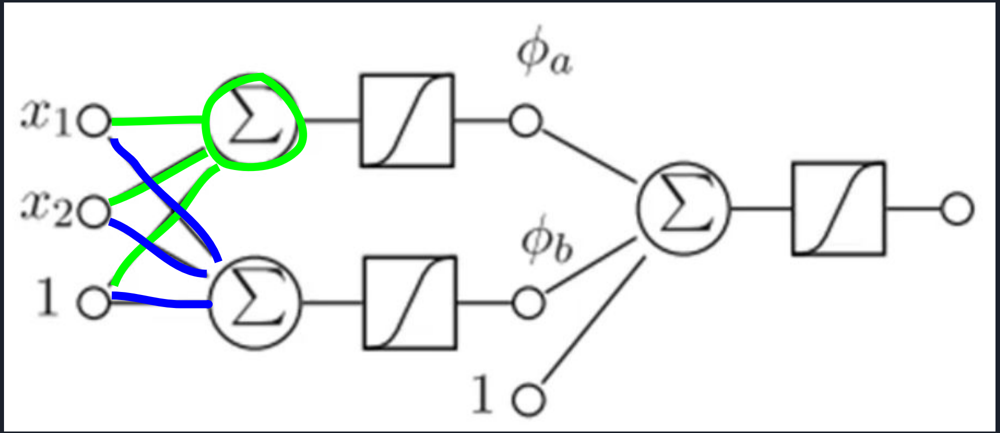


Multi-Layer Perceptron (MLP)



$$\vec{x}^T \cdot \vec{w}_a = x_1 \cdot w_{a,1} + x_2 \cdot w_{a,2} + w_{a,3} = \sum_{i=1}^{3} \tilde{x}_i \cdot w_{a,i}$$

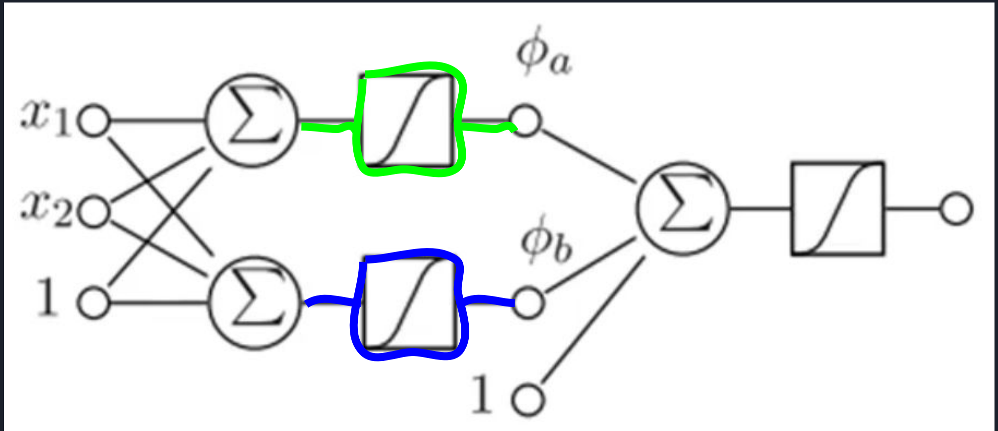
Multi-Layer Perceptron (MLP)



$$\vec{x}^T \cdot \vec{w}_a = x_1 \cdot w_{a,1} + x_2 \cdot w_{a,2} + w_{a,3} = \sum_{i=1}^{3} \tilde{x}_i \cdot w_{a,i}$$

$$\vec{x}^T \cdot \vec{w}_b = x_1 \cdot w_{b,1} + x_2 \cdot w_{b,2} + w_{b,3} = \sum_{i=1}^{3} \tilde{x}_i \cdot w_{b,i}$$

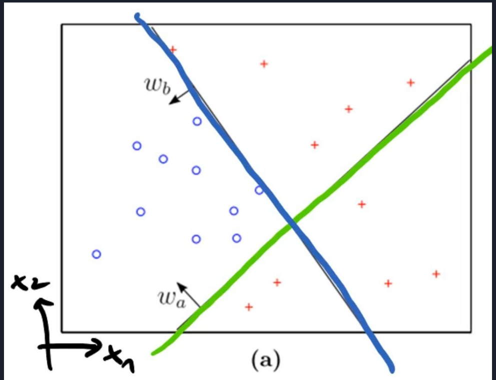
Multi-Layer Perceptron (MLP)

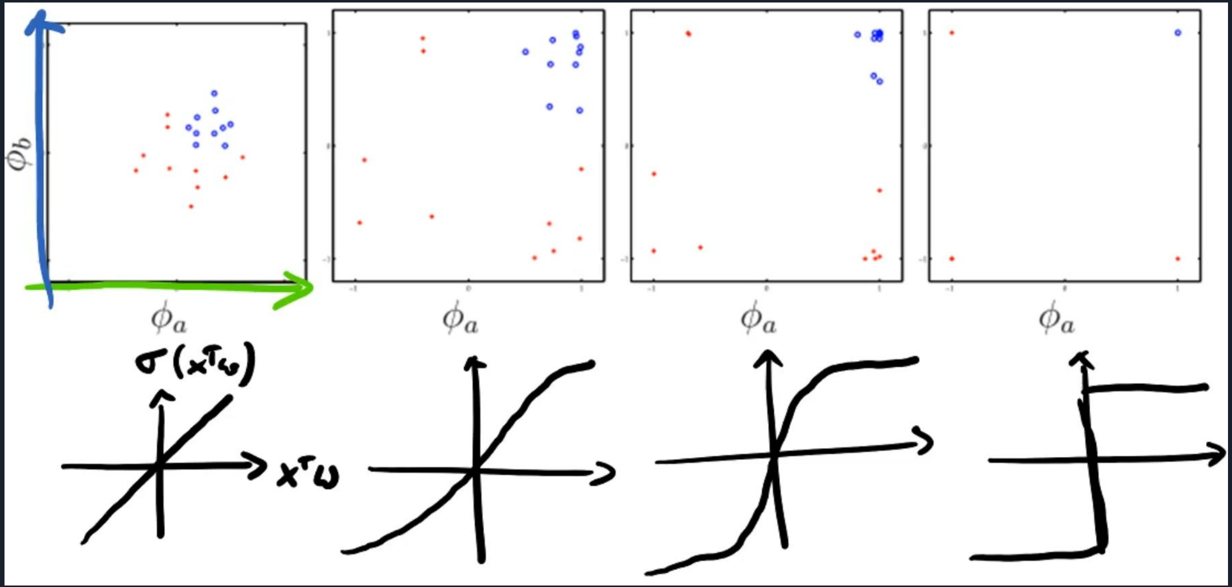


$$\phi_a = \sigma(\vec{x}^T \cdot \vec{w}_a)$$

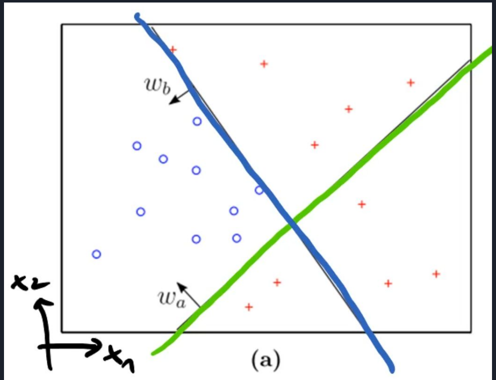
$$\phi_b = \sigma(\vec{x}^T \cdot \vec{w}_b)$$

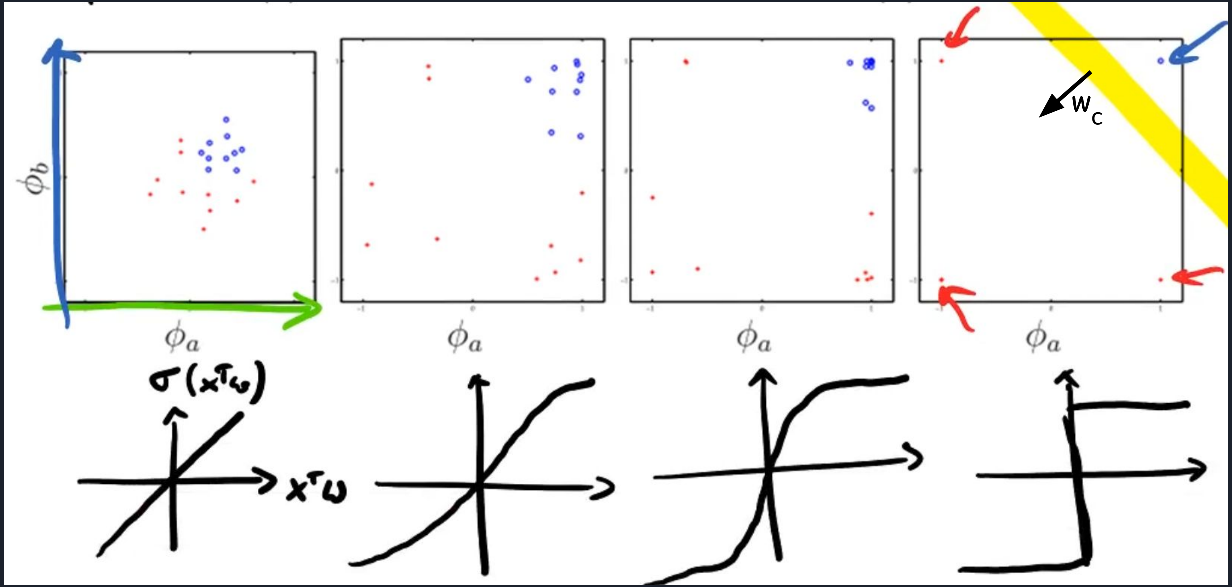
Multi-Layer Perceptron (MLP)

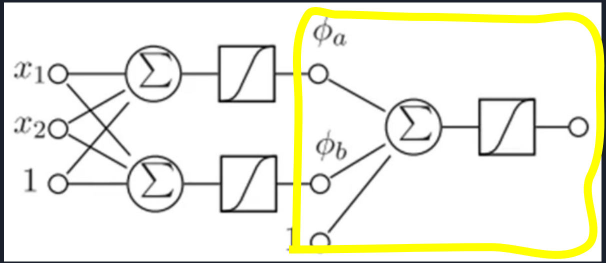




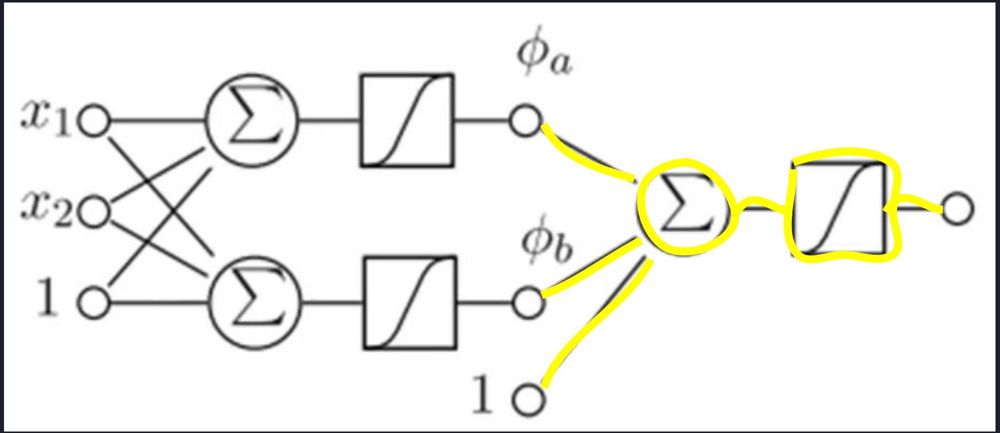
Multi-Layer Perceptron (MLP)





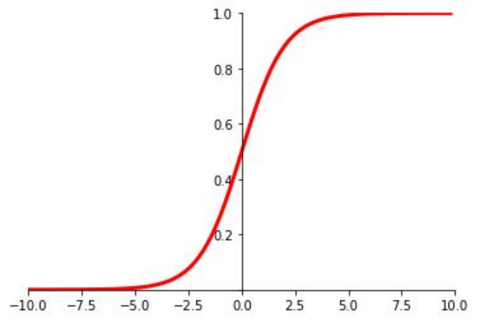


Multi-Layer Perceptron (MLP)



$$\vec{\phi}^T \cdot \vec{w}_c = \phi_a \cdot w_{c,1} + \phi_b \cdot w_{c,2} + w_{c,3}$$

$$p = \tilde{\sigma}(\vec{\phi}^T \cdot \vec{w}_c)$$



Training

- minimize the loss function

$$\ell = -\sum_{o} \left(-y_o \log(p_o) + (1 - y_o) \log(1 - p_o)\right)$$

y_o = true label for observation *o*

p_o = predicted label for observation *o*

Training

- learn weights using gradient descent

$$w_a^{\text{new}} = w_a^{\text{old}} - \lambda \frac{\partial \ell}{\partial w_a}$$

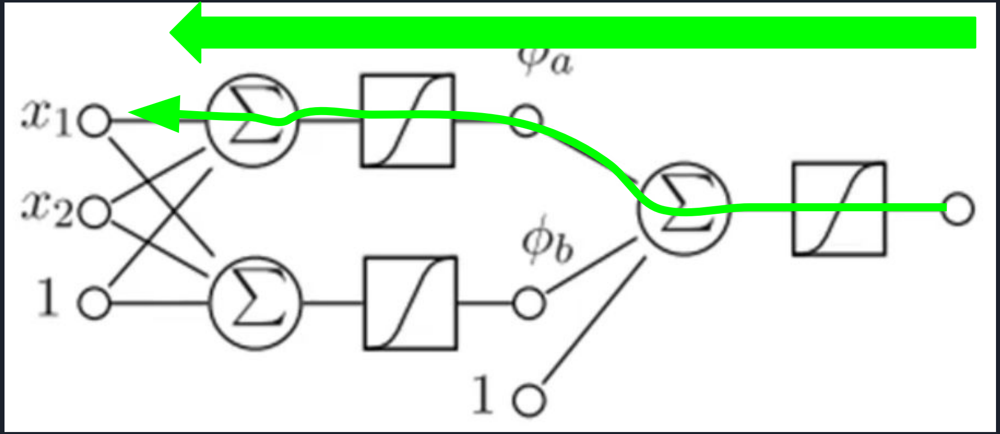
$$w_b^{\text{new}} = w_b^{\text{old}} - \lambda \frac{\partial \ell}{\partial w_b}$$

$$w_c^{\text{new}} = w_c^{\text{old}} - \lambda \frac{\partial \ell}{\partial w_c}$$

Training

- use backpropagation to compute gradients efficiently

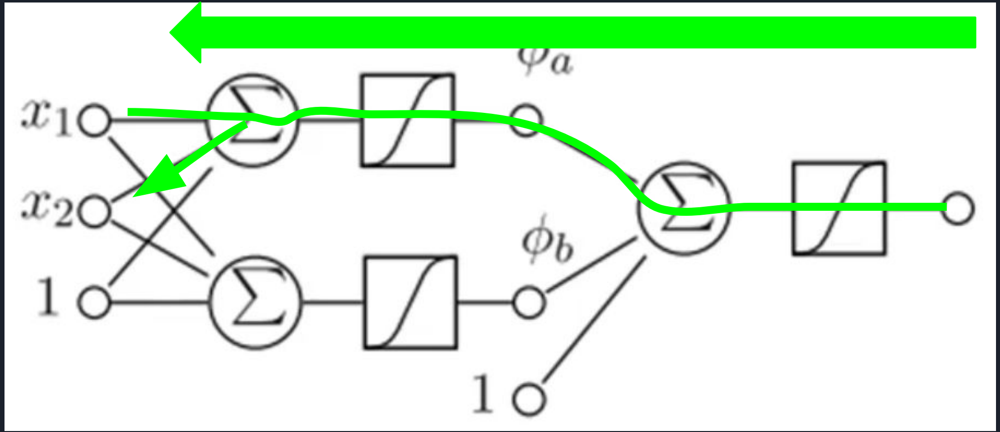
$$\frac{\partial \ell}{\partial w_{a,1}} = \frac{\partial \ell}{\partial p} \cdot \frac{\partial p}{\partial (\vec{\phi}^T \cdot \vec{w}_c)} \cdot \frac{\partial (\vec{\phi}^T \cdot \vec{w}_c)}{\partial \phi_a} \cdot \frac{\partial \phi_a}{\partial (\vec{x}^T \cdot \vec{w}_a)} \cdot \frac{\partial (\vec{x}^T \cdot \vec{w}_a)}{\partial w_{a,1}}$$



Training

- use backpropagation to compute gradients efficiently

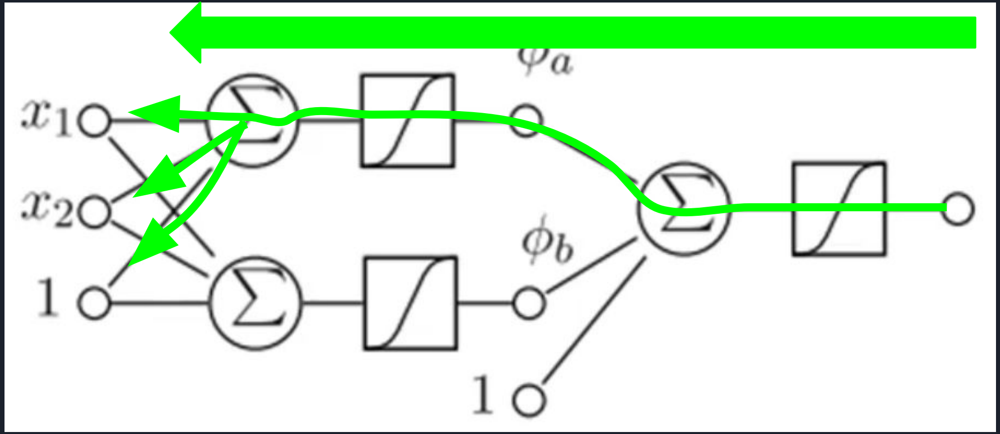
$$\frac{\partial \ell}{\partial w_{a,2}} = \frac{\partial \ell}{\partial p} \cdot \frac{\partial p}{\partial(\vec{\phi}^T \cdot \vec{w}_c)} \cdot \frac{\partial(\vec{\phi}^T \cdot \vec{w}_c)}{\partial \phi_a} \cdot \frac{\partial \phi_a}{\partial(\vec{x}^T \cdot \vec{w}_a)} \cdot \frac{\partial(\vec{x}^T \cdot \vec{w}_a)}{\partial w_{a,2}}$$



Training

- use backprojection to compute gradients efficiently

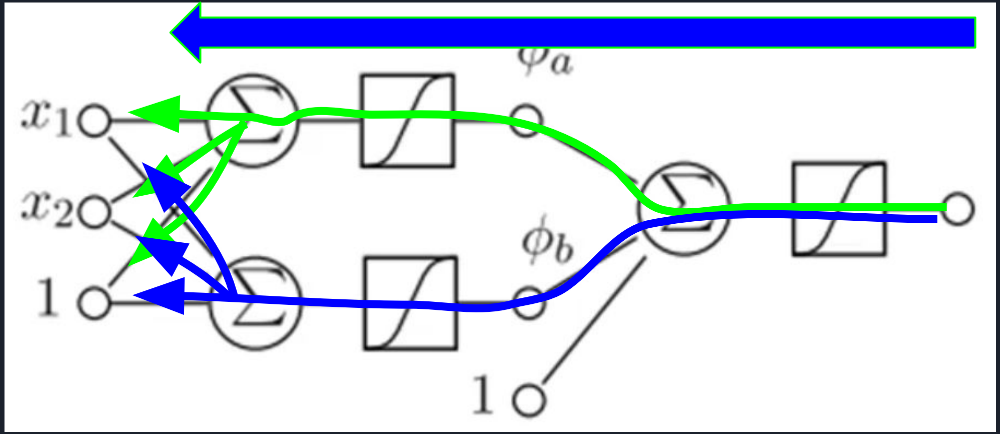
$$\frac{\partial \ell}{\partial w_{a,3}} = \frac{\partial \ell}{\partial p} \cdot \frac{\partial p}{\partial(\vec{\phi}^T \cdot \vec{w}_c)} \cdot \frac{\partial(\vec{\phi}^T \cdot \vec{w}_c)}{\partial \phi_a} \cdot \frac{\partial \phi_a}{\partial(\vec{x}^T \cdot \vec{w}_a)} \cdot \frac{\partial(\vec{x}^T \cdot \vec{w}_a)}{\partial w_{a,3}}$$



Training

- use backpropagation to compute gradients efficiently

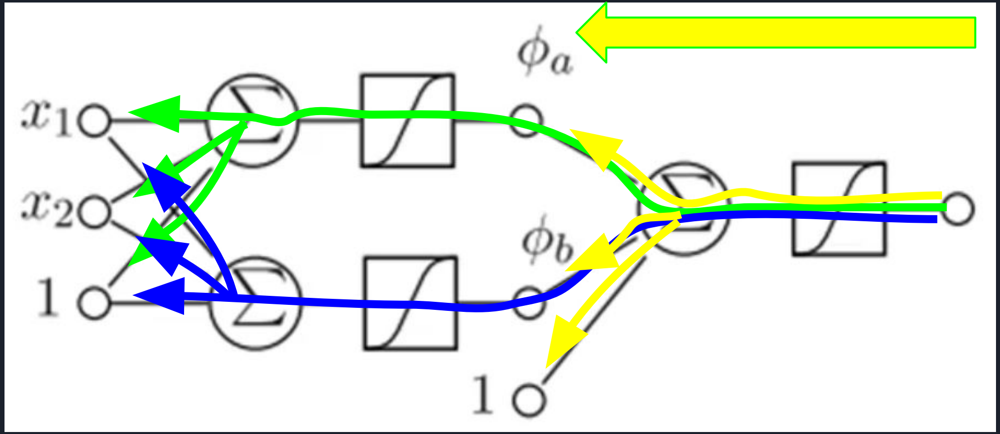
$$\frac{\partial \ell}{\partial w_{b,i}} = \frac{\partial \ell}{\partial p} \cdot \frac{\partial p}{\partial (\vec{\phi}^T \cdot \vec{w}_c)} \cdot \frac{\partial (\vec{\phi}^T \cdot \vec{w}_c)}{\partial \phi_b} \cdot \frac{\partial \phi_b}{\partial (\vec{x}^T \cdot \vec{w}_b)} \cdot \frac{\partial (\vec{x}^T \cdot \vec{w}_b)}{\partial w_{b,i}}$$



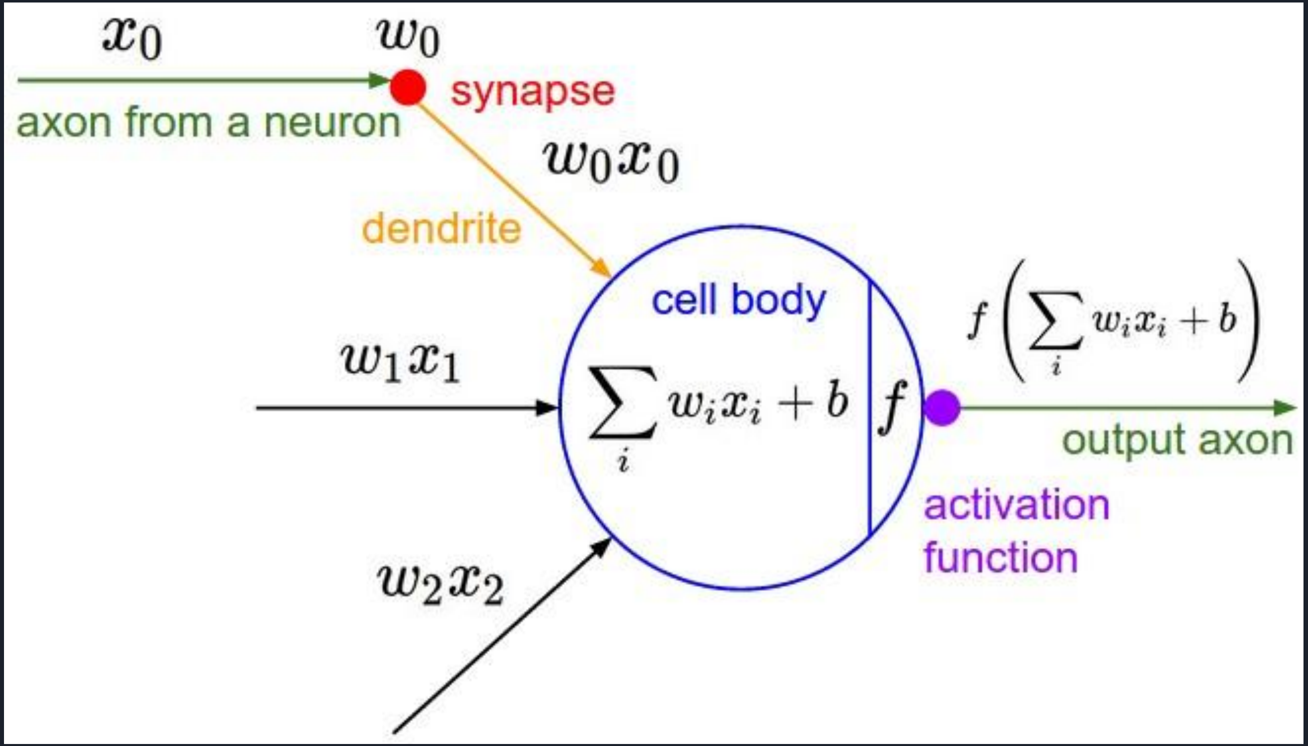
Training

- use backpropagation to compute gradients efficiently

$$\frac{\partial \ell}{\partial w_{c,i}} = \frac{\partial \ell}{\partial p} \cdot \frac{\partial p}{\partial (\vec{\phi}^T \cdot \vec{w}_c)} \cdot \frac{\partial (\vec{\phi}^T \cdot \vec{w}_c)}{\partial w_{c,i}}$$



Neural Network



Figures taken from http://cs231n.github.io/convolutional-networks/

Fully Connected Neural Network

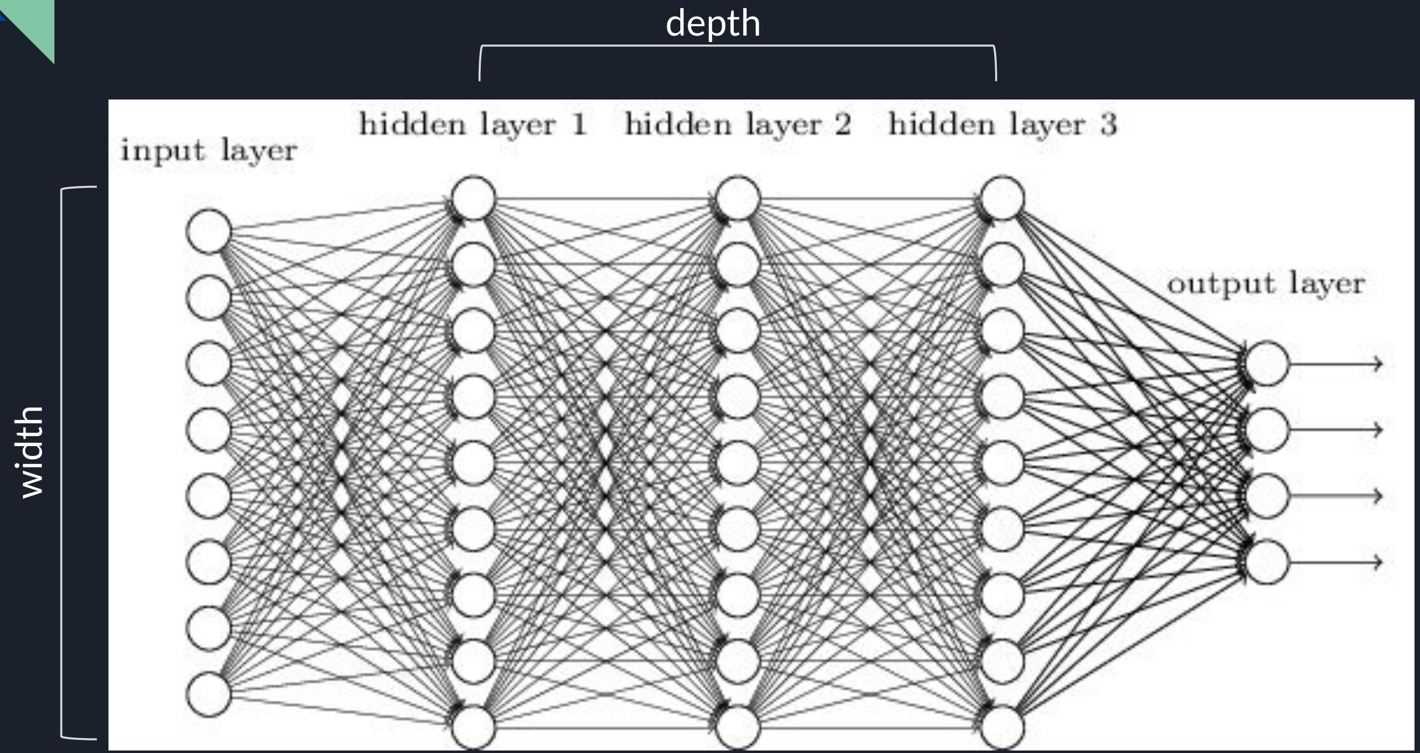
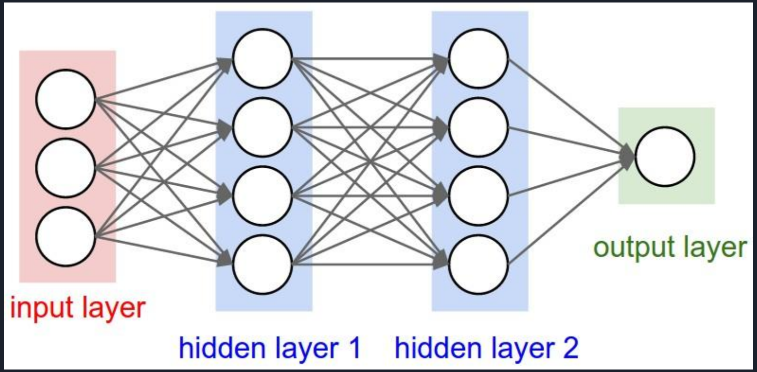
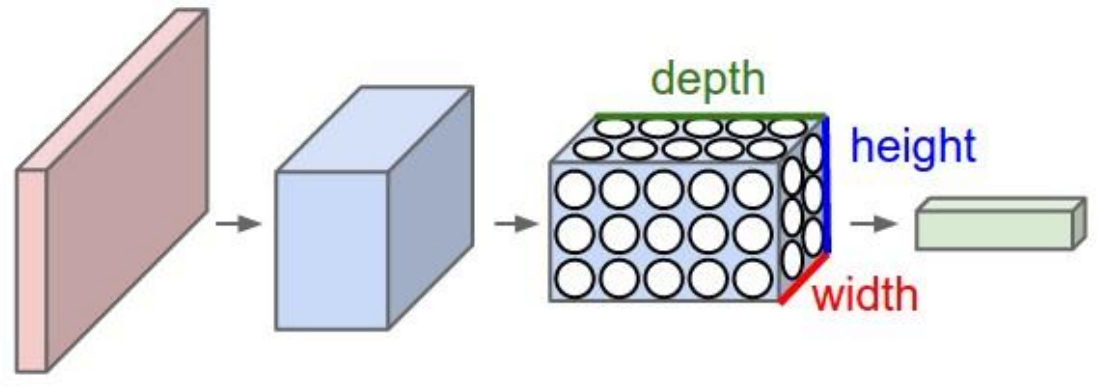


Figure taken from
https://de.mathworks.com/matlabcentral/fileexchange/64247-simple-neural-network

Convolutional Neural Network (CNN)





Figures taken from http://cs231n.github.io/convolutional-networks/

Convolutional Neural Network (CNN)

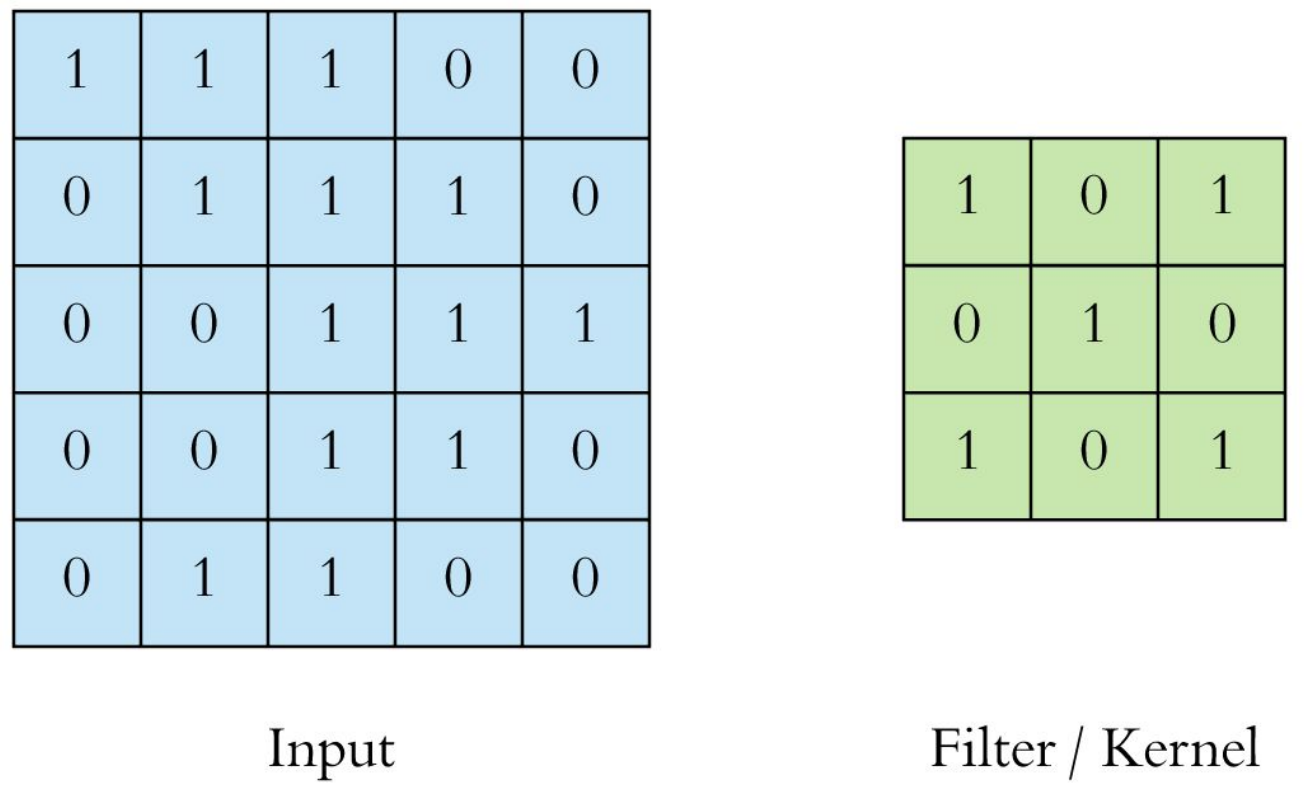


Figure taken from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolutional Neural Network (CNN)

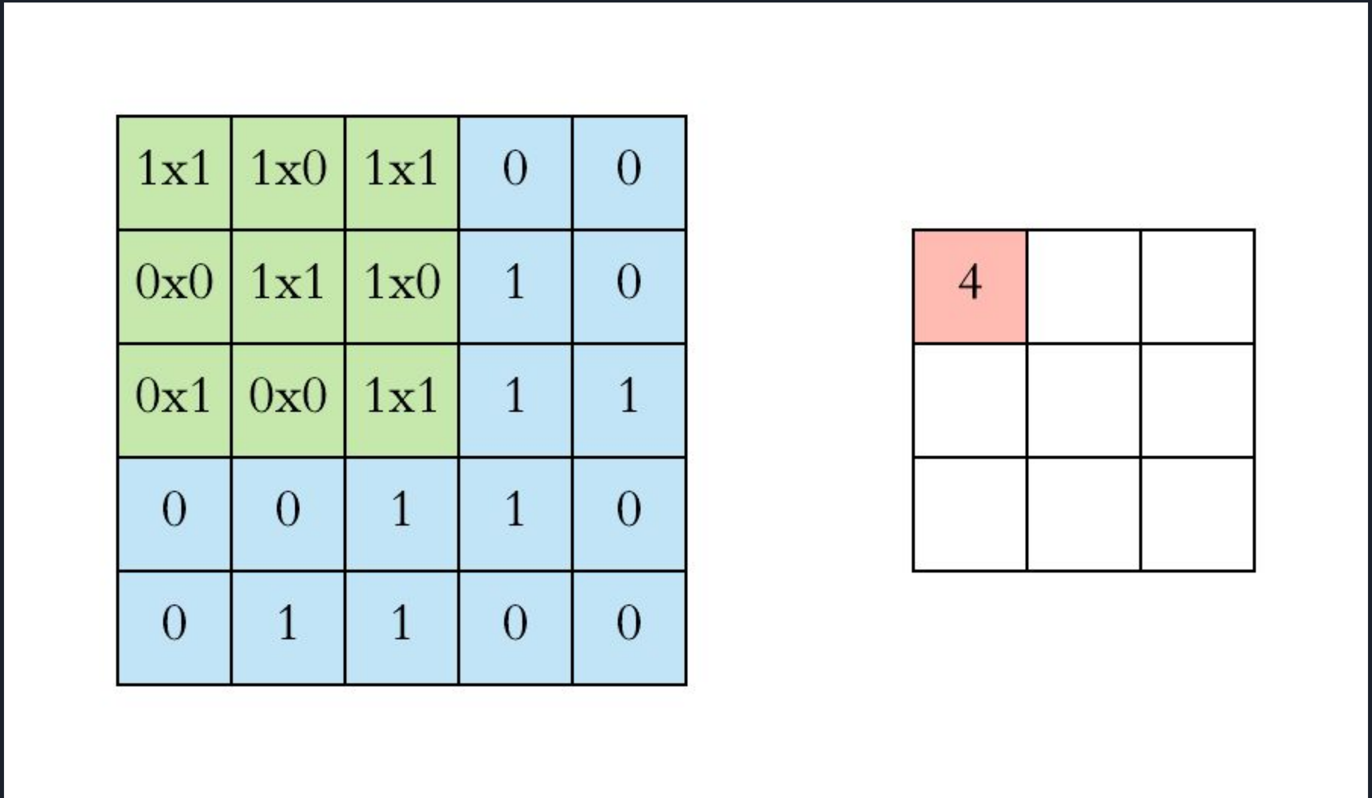


Figure taken from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolutional Neural Network (CNN)

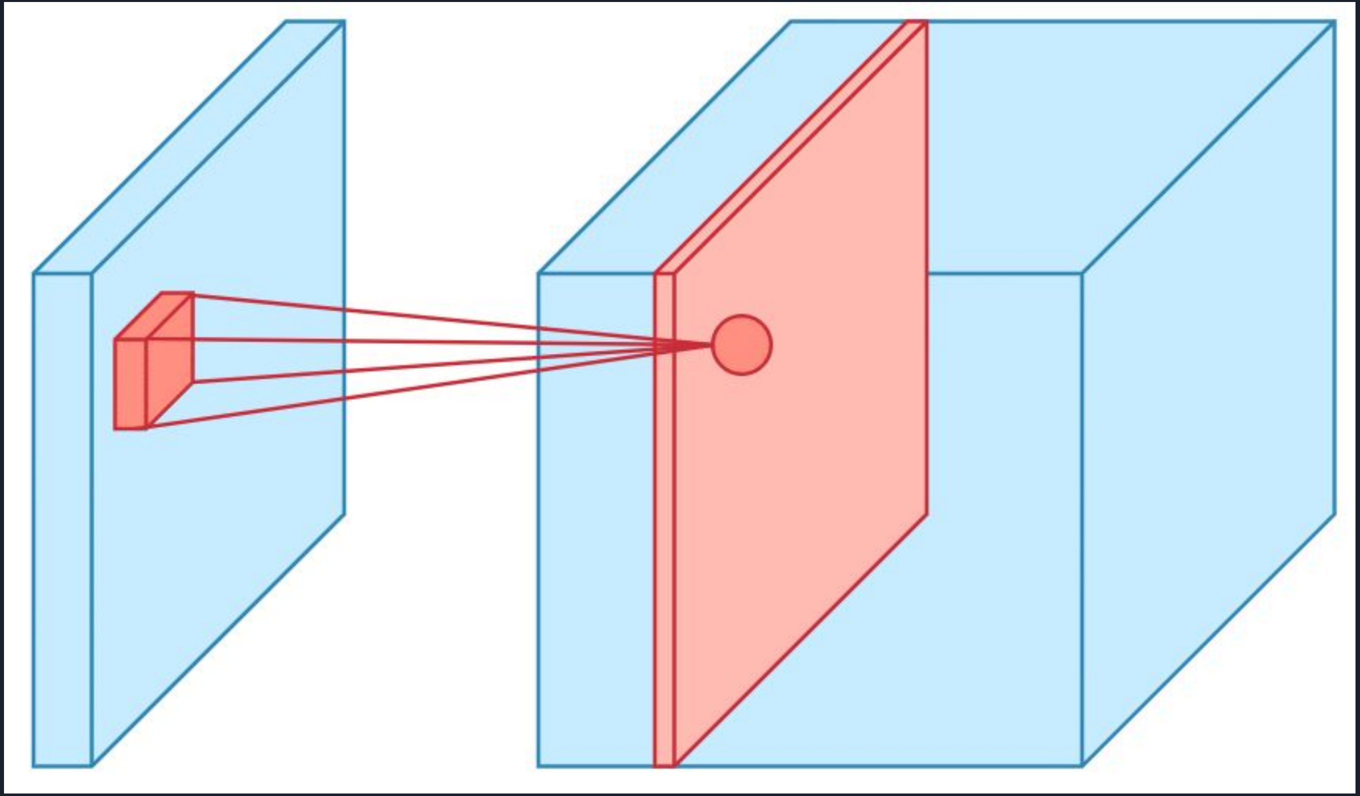


Figure taken from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolutional Neural Network (CNN)

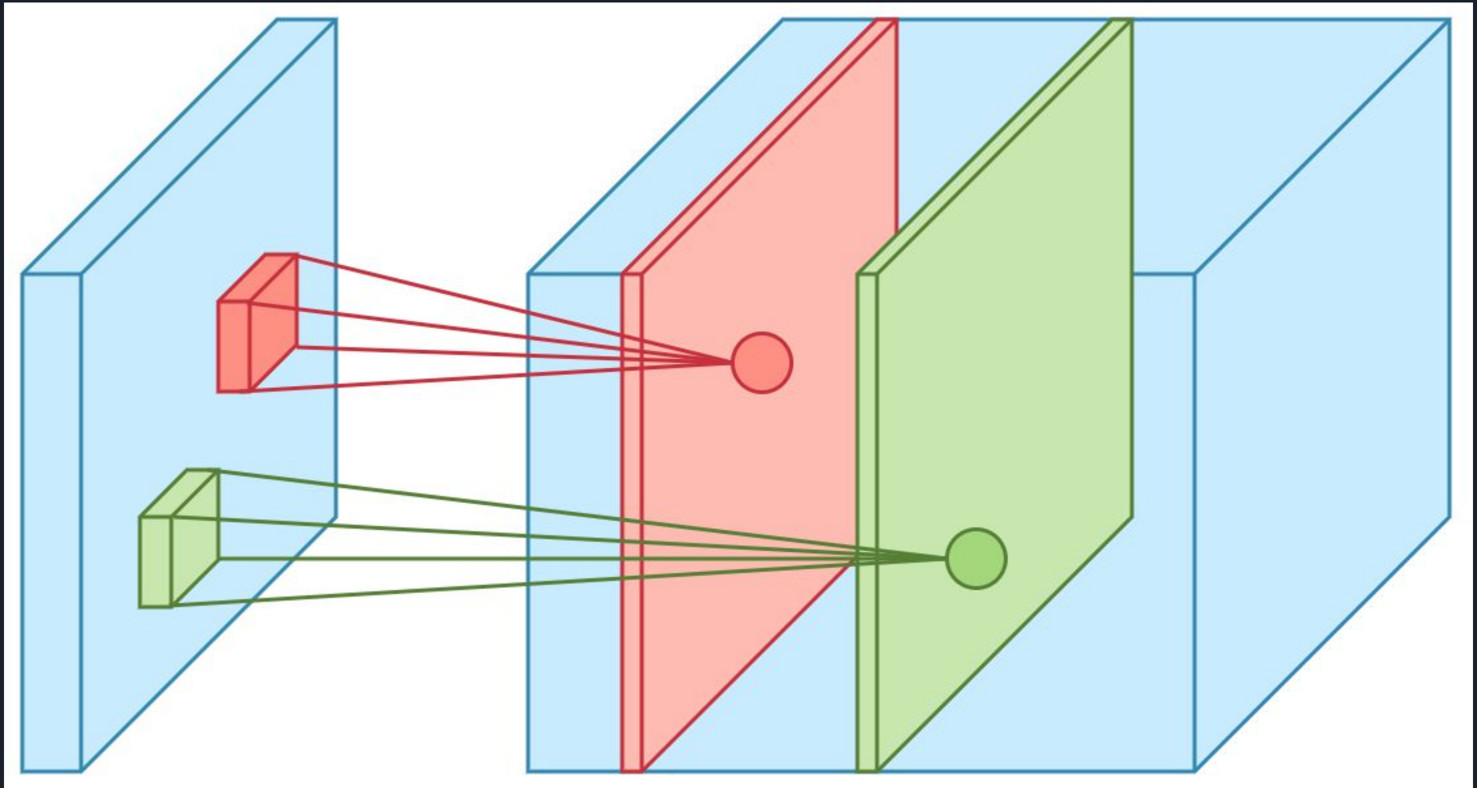


Figure taken from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolutional Neural Network (CNN)

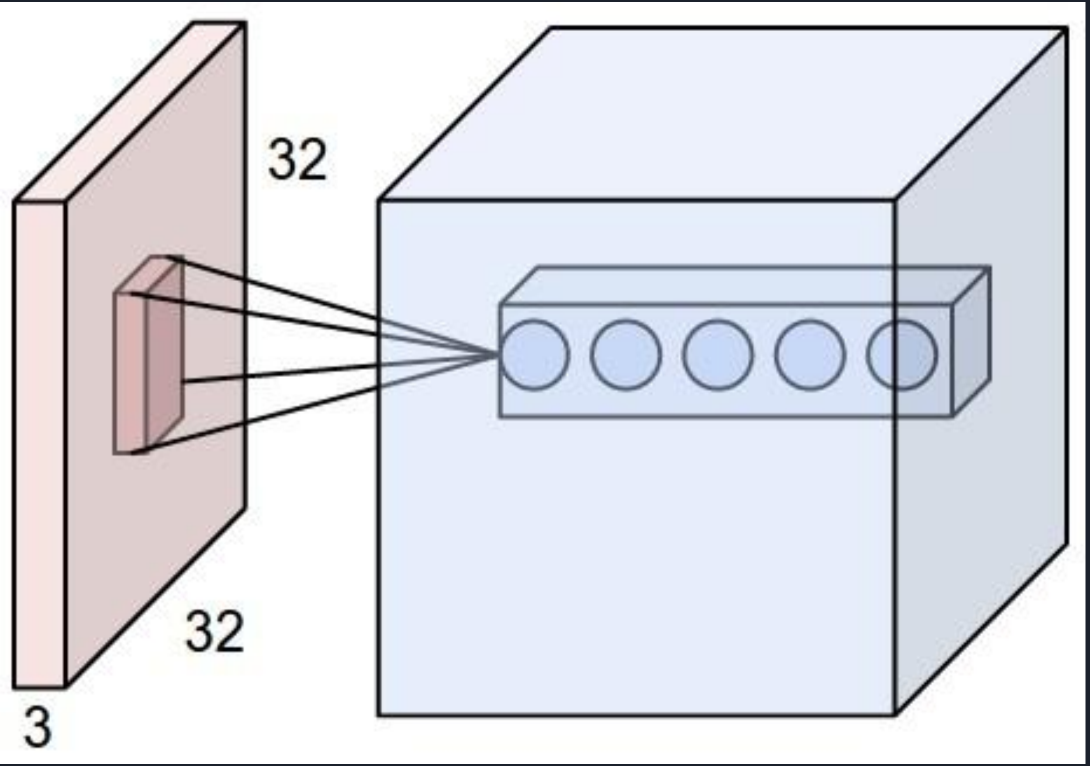
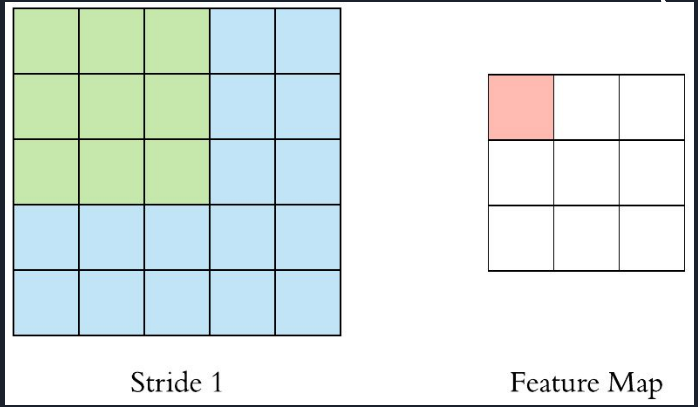
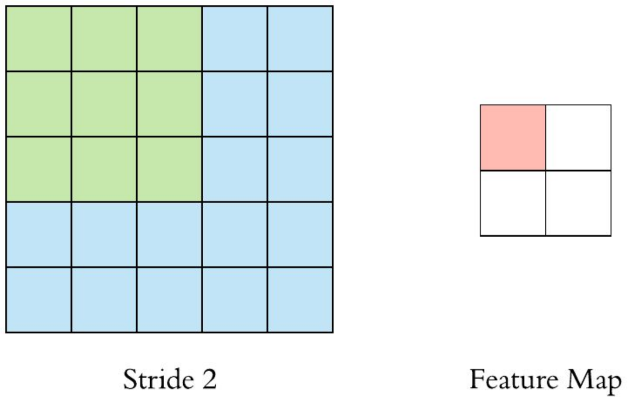
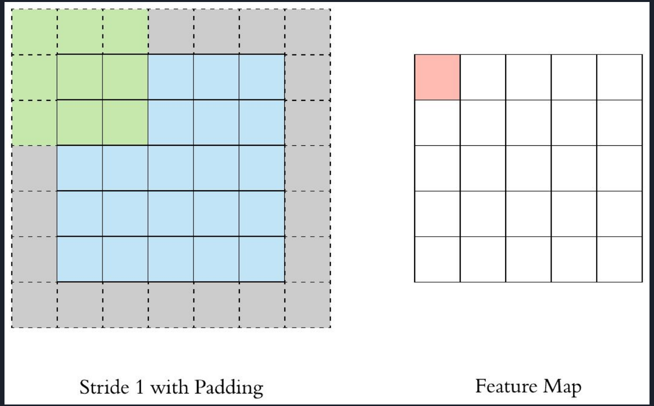


Figure taken from http://cs231n.github.io/convolutional-networks/

Convolutional Neural Network (CNN)







Figures taken from ttps://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolutional Neural Network (CNN)

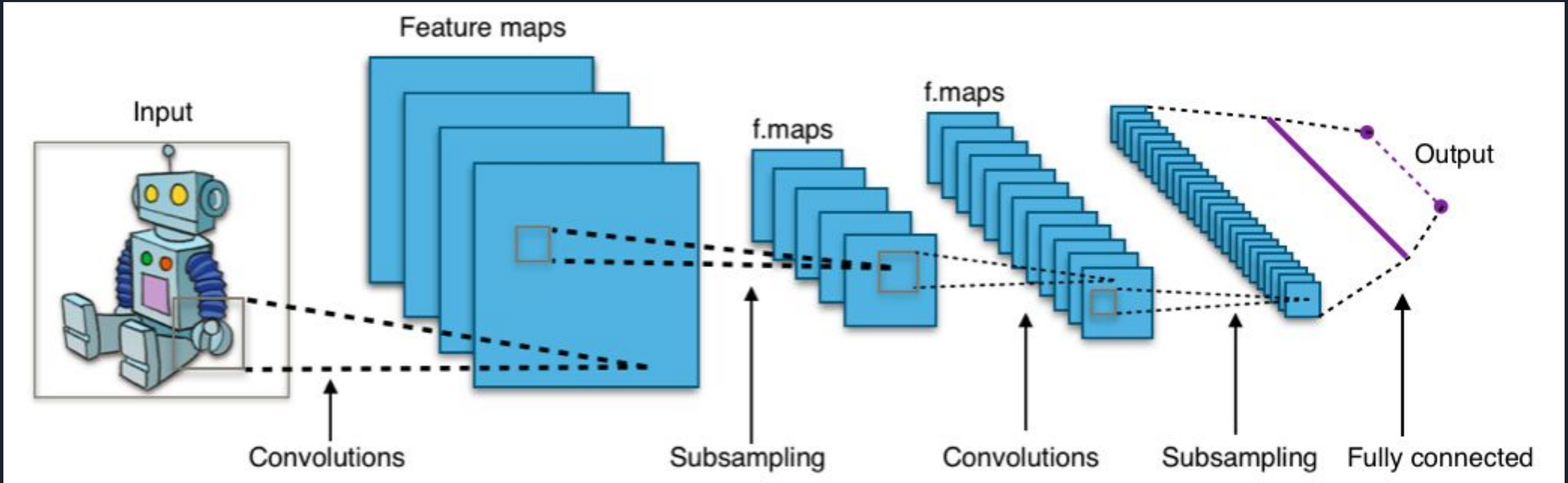


Figure taken from https://en.wikipedia.org/wiki/Convolutional_neural_network

Convolutional Neural Network (CNN)

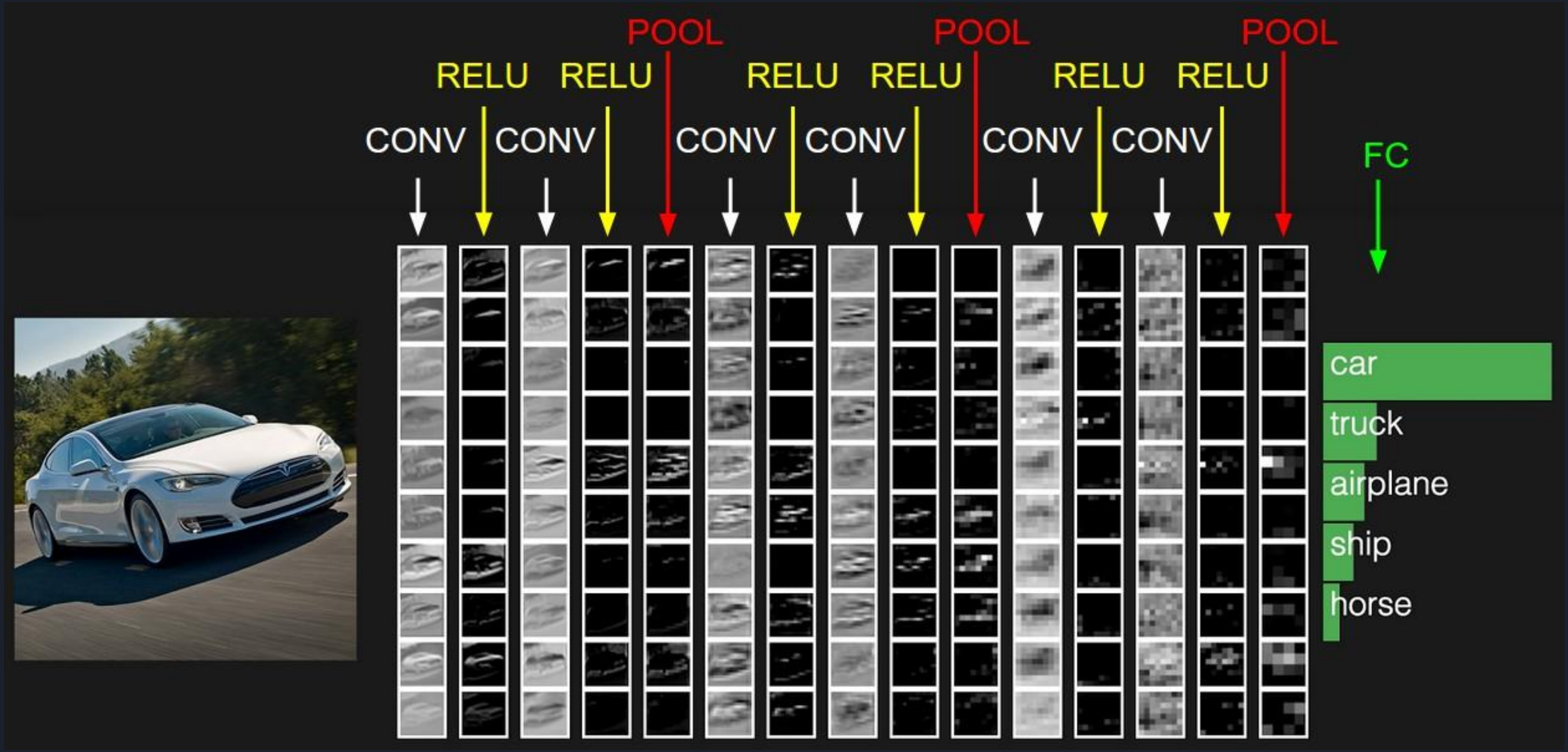


Figure taken from http://cs231n.github.io/convolutional-networks/

Common Architectures and Loss Functions

Common Network Architectures: VGG

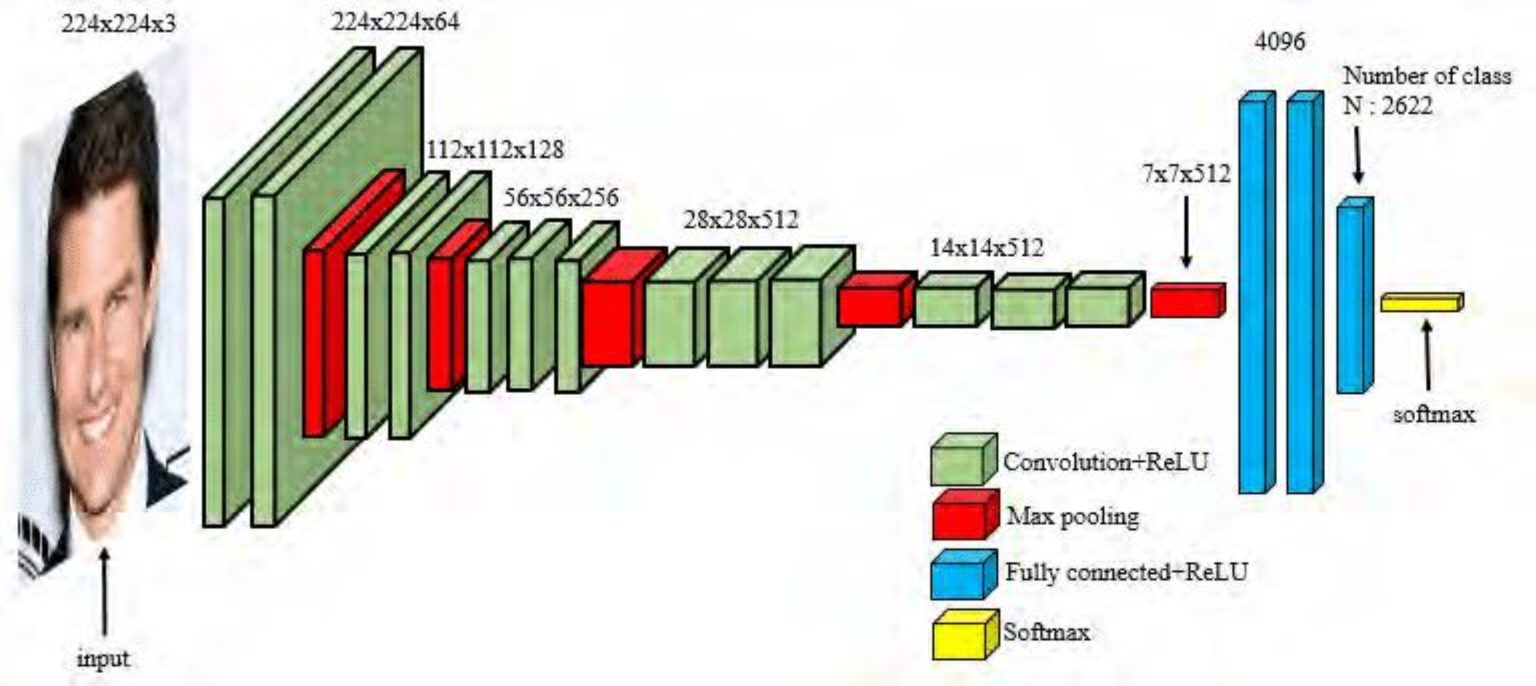


Figure taken from https://www.lri.fr/~gcharpia/deeppractice/chap_2.html

Common Network Architectures: ResNet

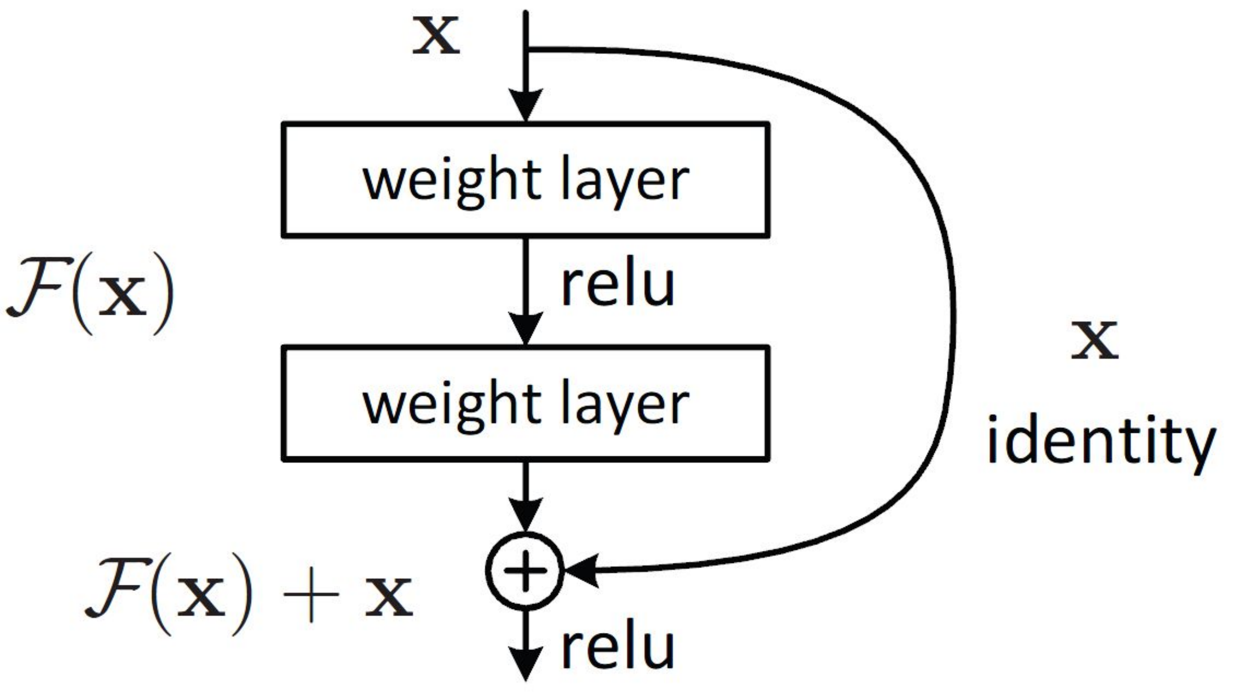


Figure taken from He et al. (2016), "Deep Residual Learning for Image Recognition"

Common Network Architectures: DenseNet

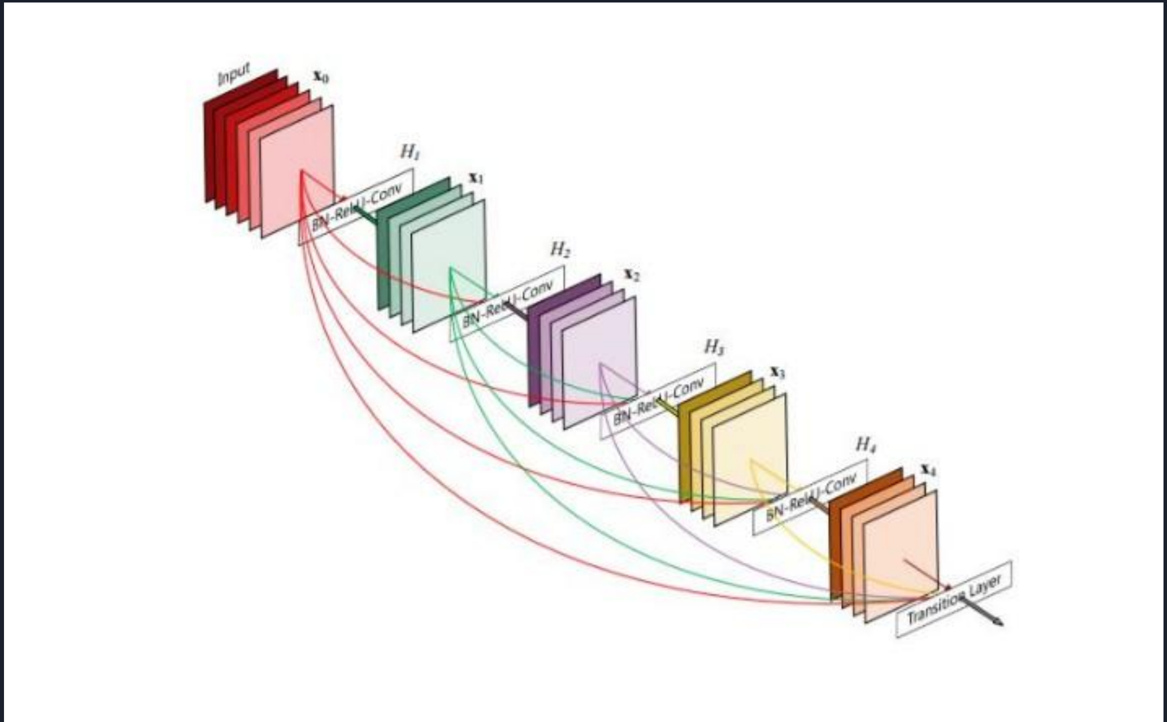


Figure taken from Huang et al. (2016), “Densely Connected Convolutional Networks”

Common Network Architectures: U-Net

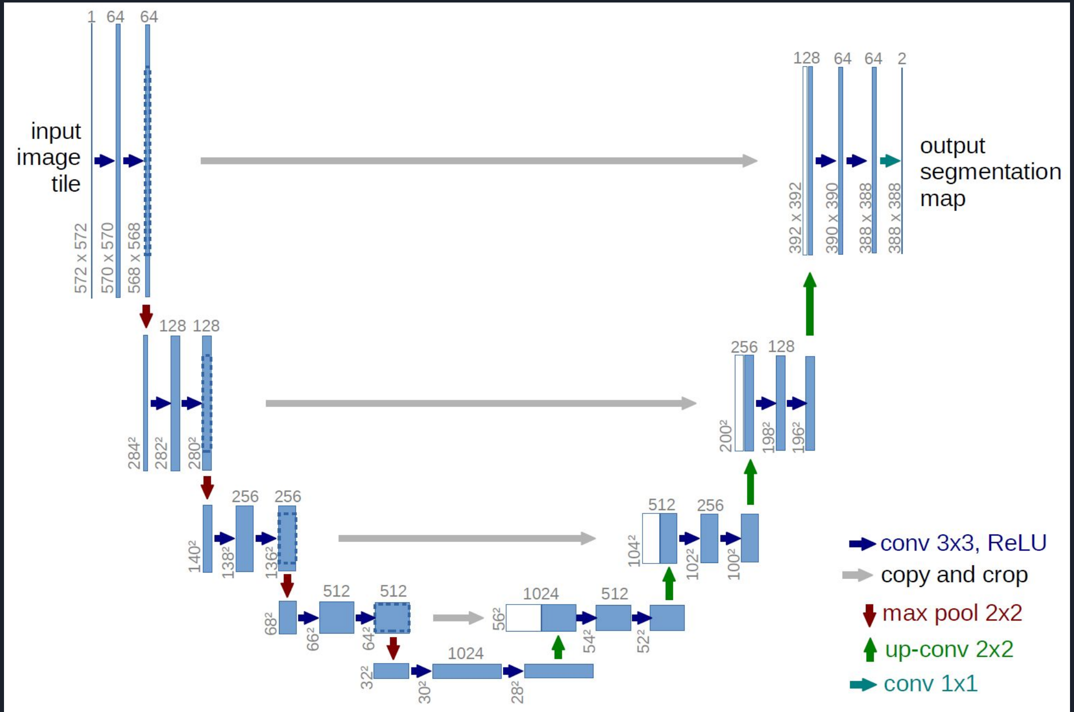


Figure taken from Renneberger et al. (2015), "U-Net: Convolutional for Biomedical Image Segmentation"

Common Network Architectures: Xception Network

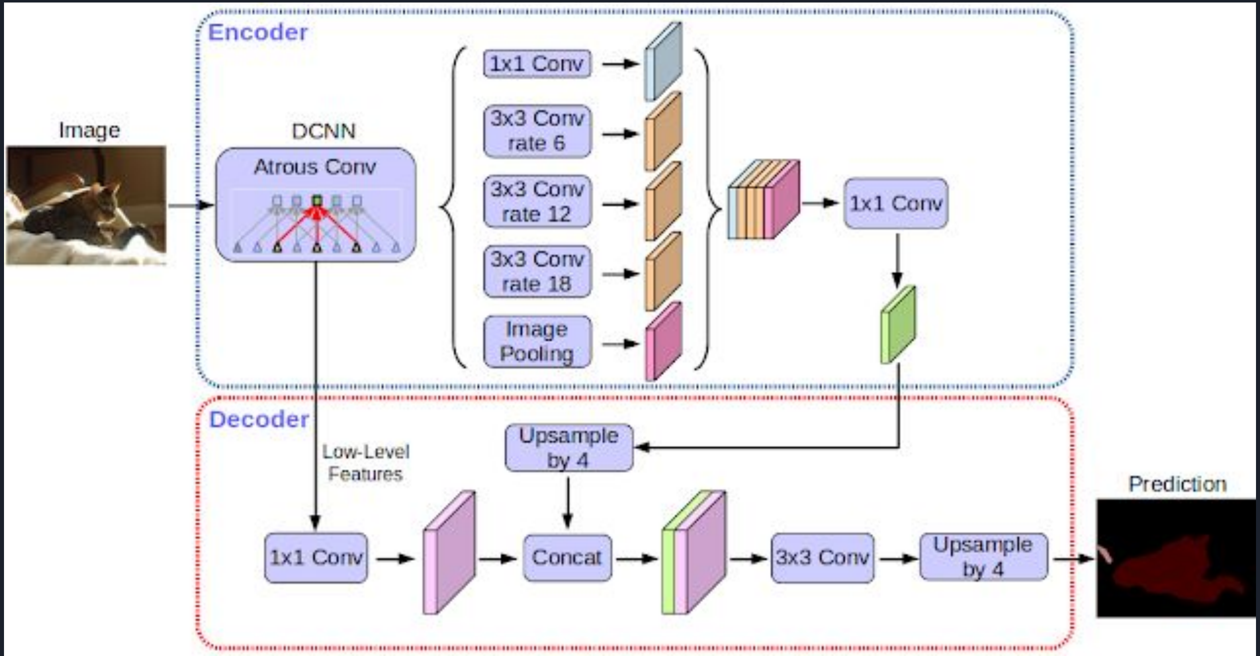


Figure taken from Chen et al. (2018), “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”

Loss Functions

- Cross Entropy Loss
- Hinge Loss
- Mean Absolute Error
- Mean Square Error
- Sörensen-Dice Loss
- …
- problem dependent choice

Applications in Particle Physics

Applications in Particle Physics

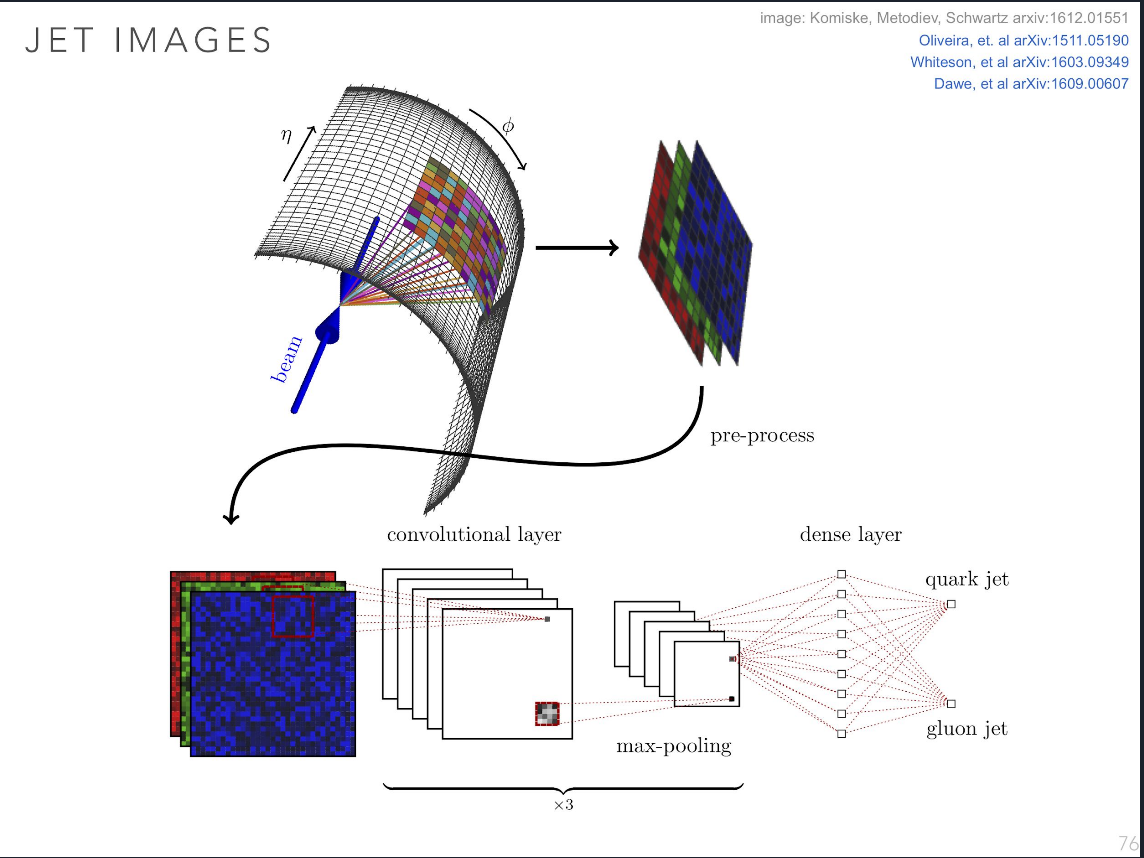


Figure taken from https://figshare.com/articles/NIPS_2016_Keynote_Machine_Learning_Likelihood_Free_Inference_in_Particle_Physics/4291565/1

Applications in Particle Physics

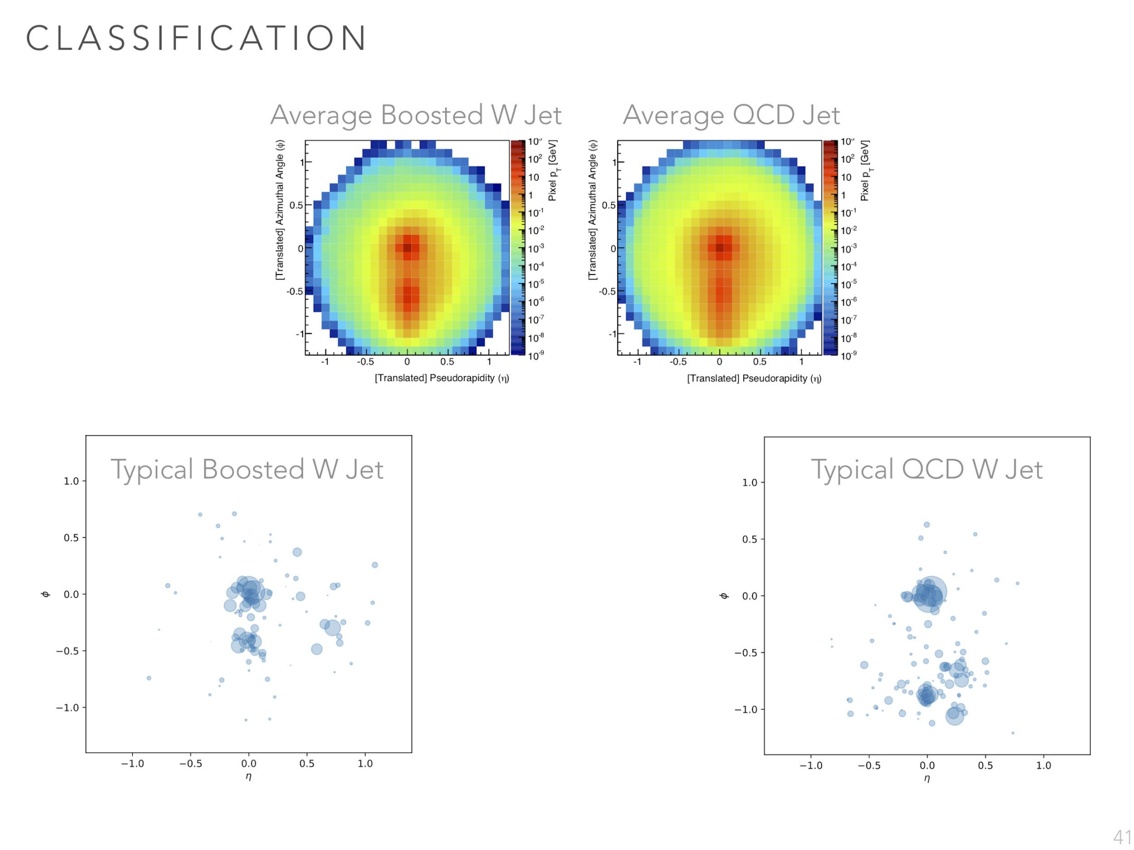


Figure taken from http://helper.ipam.ucla.edu/publications/dlt2018/dlt2018_14649.pdf

Applications in Particle Physics

JETS AS A GRAPH

Using message passing neural networks over a fully connected graph on the particles

- Two approaches for adjacency matrix for edges
  - **import** physics knowledge by using metric of jet algorithms $d^{\alpha}_{ii'} = \min(p^{2\alpha}_{ti}, p^{2\alpha}_{ti'}) \frac{\Delta R^2_{ii'}}{R^2}$
  - learn adjacency matrix and **export** new jet algorithm

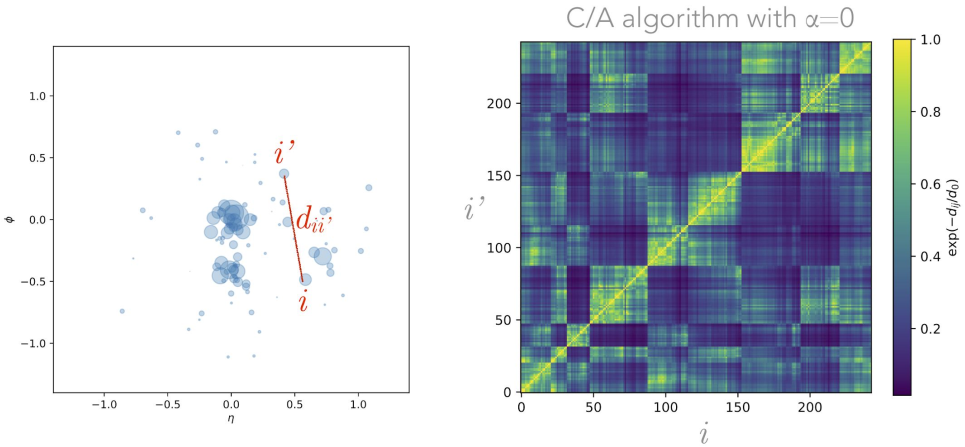


Figure taken from http://helper.ipam.ucla.edu/publications/dlt2018/dlt2018_14649.pdf

Applications in Particle Physics

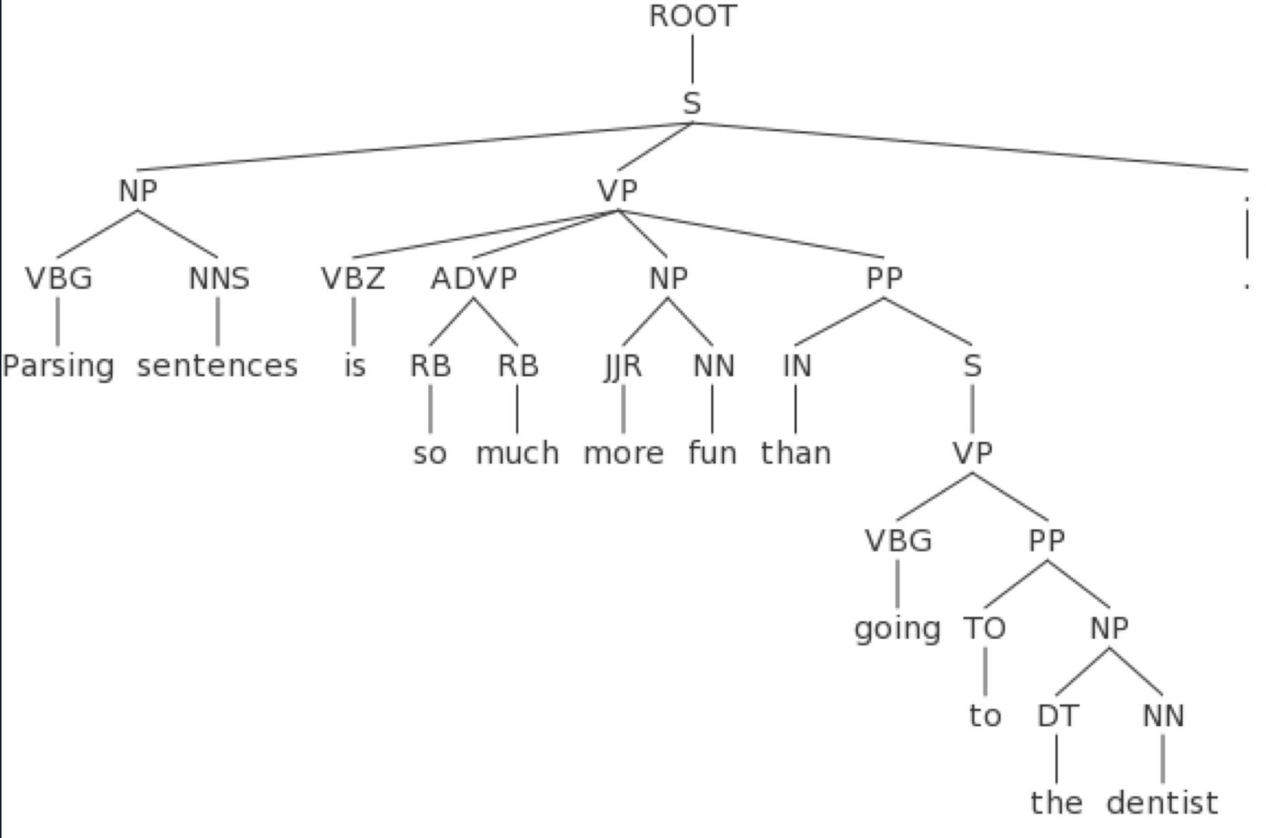


Figure taken from http://helper.ipam.ucla.edu/publications/dlt2018/dlt2018_14649.pdf

Applications in Particle Physics

- inspired by natural language processing: words follow a syntactic structure organized in a parse tree
- jet classification:
  - sentence ⇒ jet
  - words ⇒ 4-momenta
  - syntactic structure ⇒ structure dictated by QCD
  - parse tree ⇒ clustering history of a sequential recombination jet algorithm

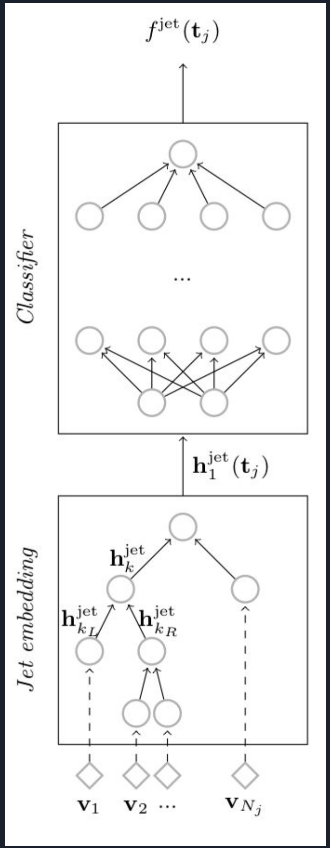


Figure taken from Louppe et al. (), “QCD-Aware Recursive Neural Networks for Jet Physics”

Applications in Particle Physics

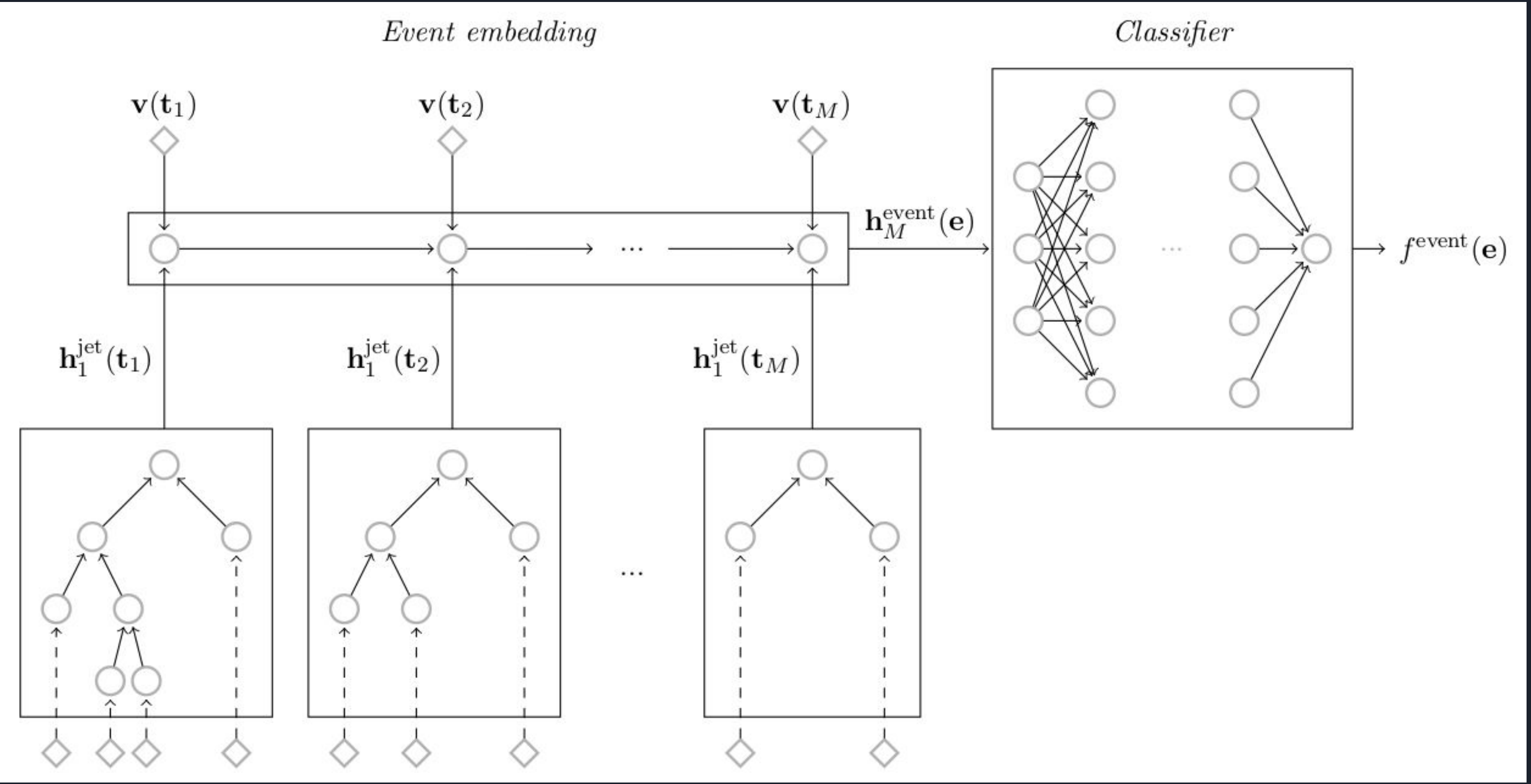


Figure taken from Louppe et al. (2017), “QCD-Aware Recursive Neural Networks for Jet Physics”

Tools for Deep Learning

Tools for Deep Learning

- mainly used Python libraries (March 2019): PyTorch, TensorFlow and Theano (out-dated)
- PyTorch and TensorFlow have
  - easy GPU implementation
  - automated gradient computation for various operations
  - building blocks of commonly used models
  - online available implementations of many (trained) models
  - tools for visualization
  - lot's of online tutorials and documentation
- Differences:
  - dynamic (PyTorch) vs. static (TensorFlow) graph definition
  - different ways of parallelization
  - PyTorch offers better development and debugging experience
- PyTorch or TensorFlow is a question of taste

Questions?

Break!

Part II:
Unsupervised Deep Learning Models

Contents

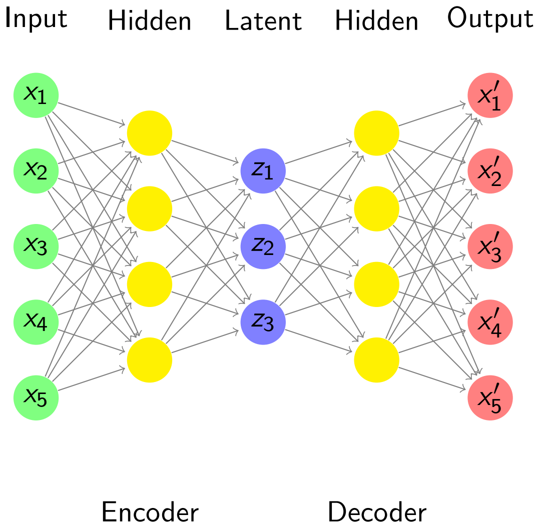
Section 1

Autoencoder (AE)

Autoencoder

- unsupervised learning models
- attempt to copy input to the output through some hidden layer
- consists of two parts: encoder and decoder
- dimension of hidden layer usually smaller than dimension of input data $\Rightarrow$ AE forced to capture the most salient features of the data
- learn useful properties of high dimensional data

Autoencoder



$$E = \left\| x'^{(i)} - x^{(i)} \right\|^2$$

Section 2

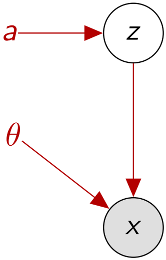
Variational Autoencoder (VAE)

Variational Autoencoder (VAE)

- VAE = generative latent variable models + neural networks to find optimal parameters [1]
- what you need to know about neural networks:
  1. ability to approximate arbitrary functions
  2. learn parameters using gradient descent
  3. convolutional neural networks (CNNs) learn convolution filters

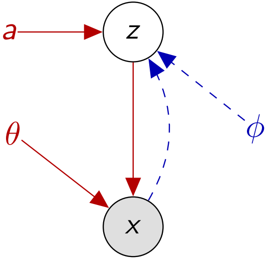
Variational Autoencoder (VAE)

- VAE = generative latent variable models + neural networks to find optimal parameters [1]
- generative latent variable models data generation:
  1. draw latent variable z (not observed) from prior distribution $z \sim p_a(z)$
  2. generate data x (observed) from conditional distribution $x \sim p_\theta(x \mid z)$

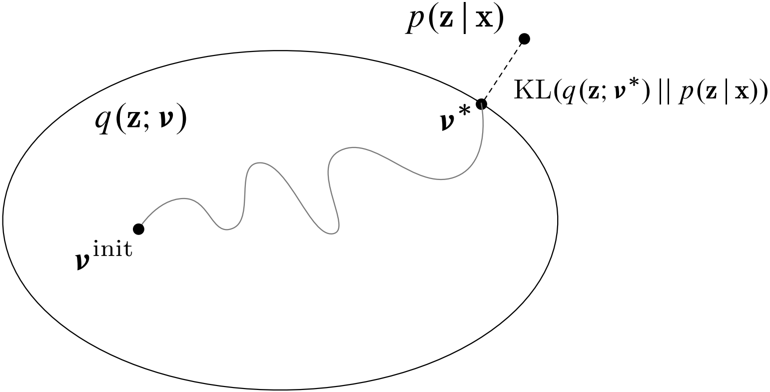


VAE

- interesting quantity: posterior distribution of latent variables given the data: $p_\theta(z \mid x)$
- problem: true posterior is intractable
- solution: introduce an approximate posterior $q_\phi(z \mid x)$



Variational Inference



$\nu \equiv \phi$

Figure taken from David Blei, for more on VI see e.g. Blei et al. (2017) [2]

VAE Objective

- objective I: find parameters ϕ that minimize difference between true and approximate posterior

$$\min_{\phi} \mathrm{KL}(q_\phi(z|x) \| p_\theta(z|x))$$

- objective II: find parameters θ that maximize the data log-likelihood

$$\max_{\theta} \log p_\theta(x)$$

VAE Objective

- use

$$\log p_\theta(x) \quad = \quad \mathcal{L}(p, q; x) \quad + \quad \mathrm{KL}(q_\phi(z|x) \| p_\theta(z|x))$$

VAE Objective

- use

$$\underbrace{\boxed{\log p_\theta(x)}}_{} = \mathcal{L}(p, q; x) + \underbrace{\boxed{\mathrm{KL}(q_\phi(z|x) \| p_\theta(z|x))}}_{\geq 0}$$

$$\overset{\max_\theta}{\downarrow} \qquad\qquad \overset{\min_\phi}{\downarrow}$$

VAE Objective

- use

$$\underbrace{\overbrace{\boxed{\log p_\theta(x)}}^{\max_\theta}}_{} = \underbrace{\overbrace{\boxed{\mathcal{L}(p, q; x)}}^{\max_{\theta,\phi}}}_{\text{lower bound to the log-likelihood called ELBO}} + \underbrace{\overbrace{\boxed{\mathbf{KL}(q_\phi(z|x)\|p_\theta(z|x))}}^{\min_\phi}}_{\geq 0}$$

VAE Objective

- new objective: maximize the lower bound (ELBO)

$$\max_{\theta,\phi} \mathcal{L}(p, q; x)$$

with

$$\mathcal{L}(p, q; x) = \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x) \| p_a(z))$$

VAE Objective

- new objective: maximize the lower bound (ELBO)

$$\max_{\theta,\phi} \mathcal{L}(p, q; x)$$

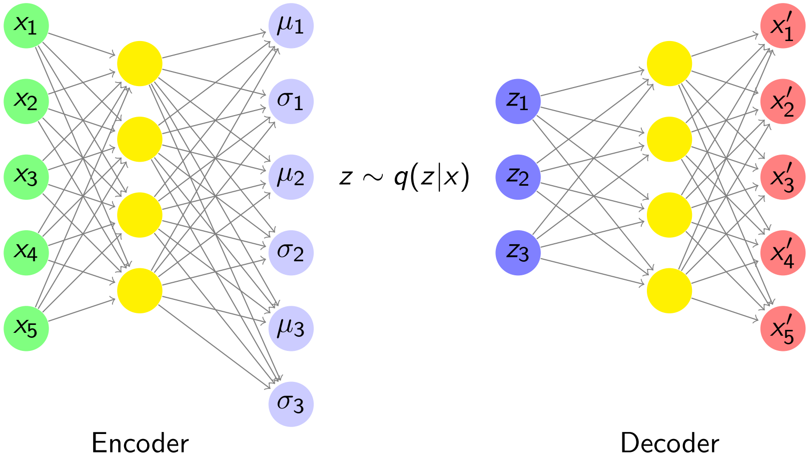
with

$$\mathcal{L}(p, q; x) = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \mathrm{KL}(q_\phi(z|x) \| p_a(z))$$

- approximate $q_\phi(z|x)$ and $p_\theta(x|z)$ with neural networks (encoder and decoder)
- find optimal network parameters θ and ϕ by minimizing the loss function

$$loss = -\mathcal{L}(p, q; x)$$

VAE Network Structure



$$q(z|x;\phi) = \mathcal{N}(\mu(x), \sigma(x))$$

$$p(x|z;\theta) = \mathcal{N}(\mu(z), \sigma(z))$$

VAE Gradient Descent

- minimize loss using gradient descent

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta loss(\theta_n)$$
$$\phi_{n+1} = \phi_n - \eta \nabla_\phi loss(\phi_n)$$

with

$$\nabla_{\phi,\theta} loss = -\nabla_{\phi,\theta} \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x|z)\right] + \nabla_{\phi,\theta} \mathrm{KL}(q_\phi(z|x) \| p(z))$$

- second term usually no problem

VAE Gradient Descent

- first term with respect to θ also no problem

$$\nabla_\theta \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x|z)\right] = \mathbb{E}_{z \sim q_\phi(z|x)} \left[\nabla_\theta \log p_\theta(x|z)\right]$$

$\Rightarrow$ use e.g. Monte Carlo sampling

- problem: expectation depends on ϕ
  $\Rightarrow$ gradient w.r.t. ϕ not easily computable

VAE Gradient Descent

- solution: reparameterization trick

$$z \sim q_\phi(z \,|\, x) \to z = h_\phi(\varepsilon, x) \quad \text{with} \quad \varepsilon \sim p(\varepsilon)$$

- example: reparametrization for Gaussian distribution

$$z \sim \mathcal{N}(\mu, \sigma) \to z = \mu + \sigma \cdot \varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, 1)$$

- after reparametrization trick expectation and gradient can be exchanged

$$\begin{aligned} &\nabla_\phi \mathbb{E}_{\varepsilon \sim p(\varepsilon)} \left[\log p_\theta(x|z = h_\phi(\varepsilon, x))\right] \\ &= \mathbb{E}_{\varepsilon \sim p(\varepsilon)} \left[\nabla_\phi \log p_\theta(x|z = h_\phi(\varepsilon, x))\right] \end{aligned}$$

$\Rightarrow$ all gradients can now be computed easily e.g. using Monte Carlo sampling

VAE Loss Function

- assume data to be generated from Gaussian distribution

$$p_\theta(x|z) \sim \mathcal{N}(x|f_\theta(z), 2^{-1}\mathbb{1})$$

$\Rightarrow$ expectation term becomes a mean squared error (MSE) between data and 'reconstructed' data

$$-\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \to \frac{1}{T}\sum_{t=1}^{T} \|x_t - f_\theta(z)_t\|^2 = \mathrm{MSE}(x, x')$$

- the KL-divergence acts as a regularizer on the approximate posterior
- standard VAE loss

$$loss = \mathrm{MSE}(x, x') + \mathrm{KL}\big(q_\phi(z\,|\,x)||p_a(z)\big)$$

Section 3

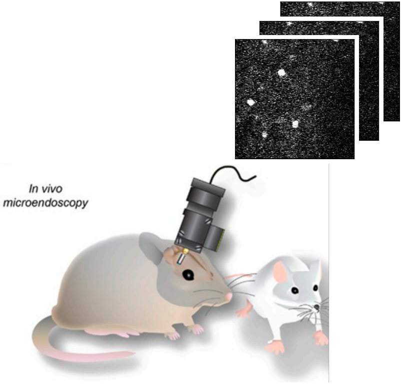
Example: The LeMoNADe Model

Example: The LeMoNADe Model

Kirschbaum et al. (2019),
"LeMoNADe: Learned Motif and Neuronal Assembly Detection in calcium imaging videos" [3]

What is Calcium Imaging?

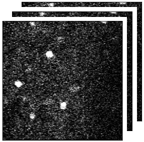
- microscopy technique
- observe activity of large cell populations on single-cell-level
- fluorescent Ca^{2+} tracer
  $\Rightarrow$ active cells light up
- widely applicable: in vitro and in vivo
- data = sequence of images
  = calcium imaging video



What are Neuronal Assemblies?

- neuronal assemblies = motifs = subsets of neurons firing in a spatio-temporal pattern
- could be crucial building blocks in neuronal information processing
- existence still debated
- cannot be detected manually

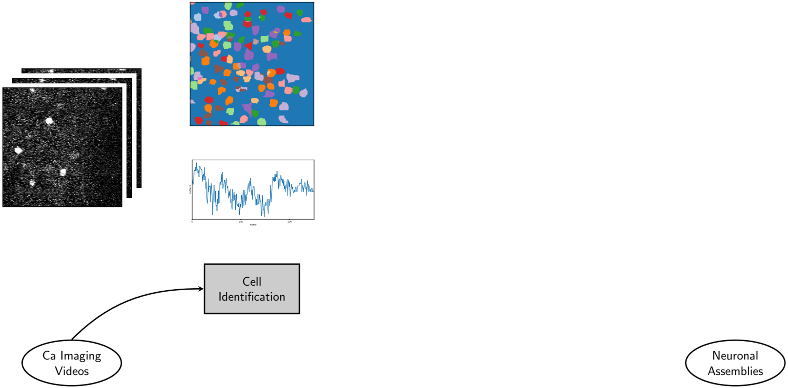
Detection of Neuronal Assemblies in Ca Imaging Videos



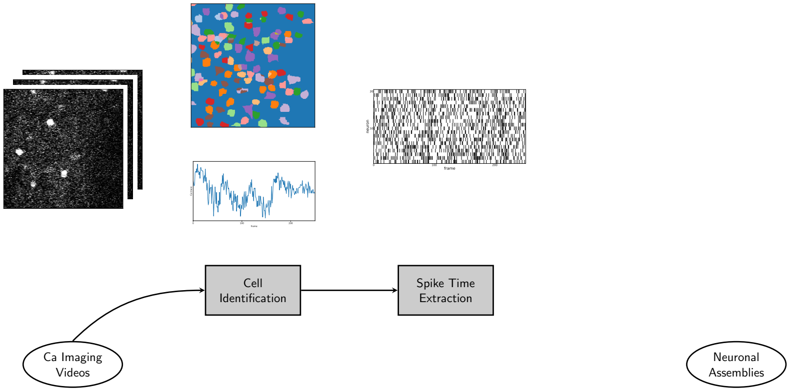
Ca Imaging Videos

Neuronal Assemblies

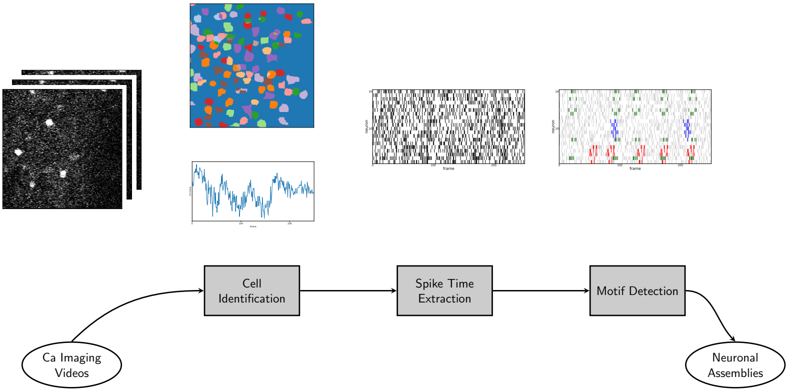
Detection of Neuronal Assemblies in Ca Imaging Videos



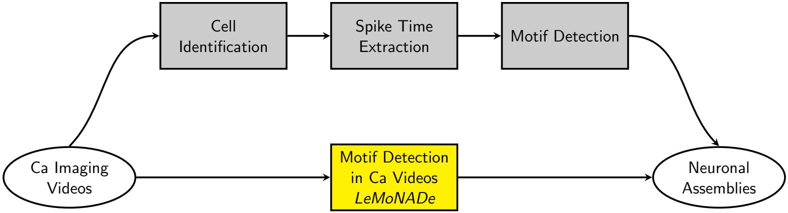
Detection of Neuronal Assemblies in Ca Imaging Videos



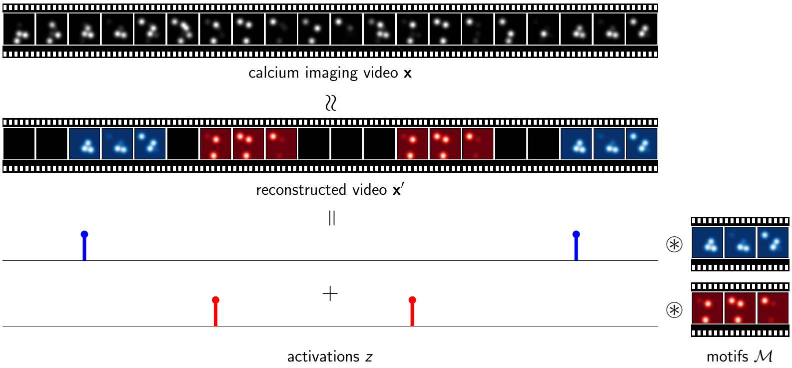
Detection of Neuronal Assemblies in Ca Imaging Videos



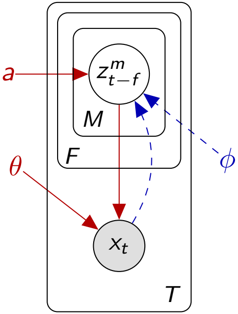
Detection of Neuronal Assemblies in Ca Imaging Videos



Idea


calcium imaging video $\mathbf{x}$
$\wr\wr$
reconstructed video $\mathbf{x}'$
$||$
$\circledast$
$+$
$\circledast$
activations z
motifs $\mathcal{M}$

LeMoNADe Model



Generative Model

$$\mathbf{z} \sim \prod_{t=1}^{T} \prod_{m=1}^{M} \mathrm{Ber}\left(z_t^m \mid a\right)$$

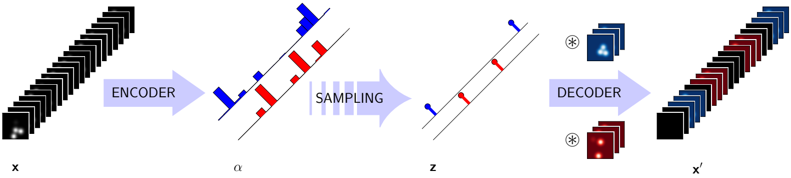
$$\mathbf{x} \mid \mathbf{z}, \theta \sim \mathcal{N}\left(\mathbf{x} \mid f_\theta(\mathbf{z}), 2^{-1}\mathbb{1}\right)$$

Recognition Model

$$\mathbf{z} \mid \mathbf{x}, \phi \sim \prod_{t=1}^{T} \prod_{m=1}^{M} \mathrm{Ber}\left(z_t^m \mid \alpha_t^m(\mathbf{x}; \phi)\right)$$

T = number of frames, M = number of motifs, F = length of each motif

LeMoNADe VAE



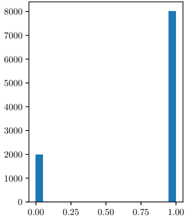
- latent space = activations of motifs in time
- motifs are either on or off
  $\Rightarrow$ Bernoulli-prior on z and Bernoulli distributions for $q(z \mid x)$
- data additiv mixture of motifs plus noise
  $\Rightarrow$ only one deconvolution layer in decoder
  $\Rightarrow$ decoding filters = motifs

LeMoNADe Reparametrization Trick

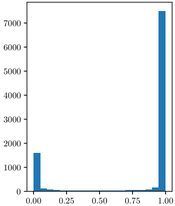
- $q_\phi(z \mid x)$ product of Bernoulli distributions
- Bernoullis = discrete $\Rightarrow$ no differentiable reparametrization trick available
- use BinConcrete continuous relaxation of Bernoulli instead

$$z \mid x \sim \text{Ber}(\alpha(x)) \rightarrow \tilde{z} \mid x \sim \text{BinConcrete}(\tilde{\alpha}(x), \lambda)$$

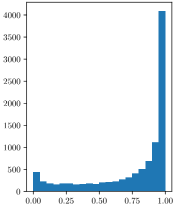
with $\tilde{\alpha} = \alpha/(1-\alpha)$



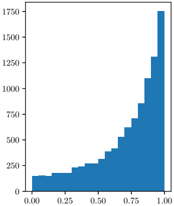
$\text{Ber}(\alpha = 0.8)$



$\text{BinConcrete}(\tilde{\alpha} = 4, \lambda = 0.1)$



$\text{BinConcrete}(\tilde{\alpha} = 4, \lambda = 0.6)$

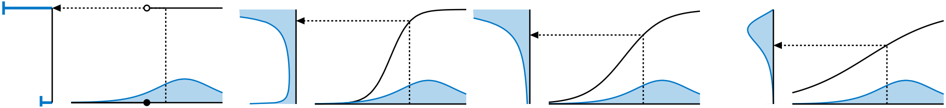


$\text{BinConcrete}(\tilde{\alpha} = 4, \lambda = 1.0)$

LeMoNADe Reparametrization Trick

- Gumbel-softmax reparametrization trick [4, 5]

$$U \sim \mathrm{Uni}(0, 1)$$
$$y = \log(\tilde{\alpha}) + \log(U) - \log(1 - U)$$
$$\tilde{z} = \sigma(y/\lambda) = \frac{1}{1 + \exp(-y/\lambda)}$$



(a) $\lambda = 0$ (b) $\lambda = 1/2$ (c) $\lambda = 1$ (d) $\lambda = 2$

Figure taken from Maddison et al. (2016) [4]

LeMoNADe Loss Function

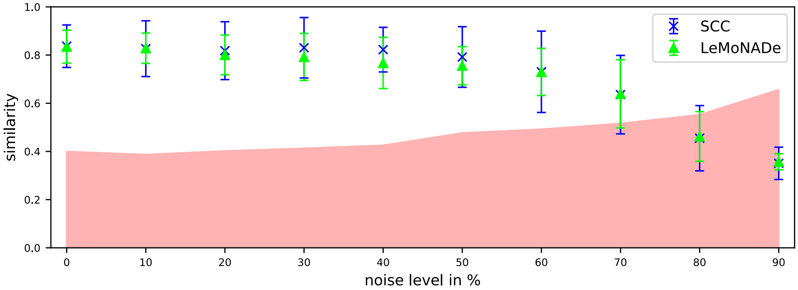
$$\mathit{loss} = \mathrm{MSE}(x, x') + \beta_{\mathrm{KL}} \cdot \mathrm{KL}\big(q_{\tilde{\alpha},\lambda_1}(y \mid x) || p_{\tilde{a},\lambda_2}(y)\big)$$

- $p_{\tilde{a},\lambda_2}(y)$ = relaxed and reparameterized prior $p_a(z)$
- $q_{\tilde{\alpha},\lambda_1}(y \mid x)$ = relaxed and reparameterized approximate posterior $q_\phi(z \mid x)$
- β_{KL} controls regularization strength [6]

Results on Synthetic Data

- 200 datasets
- 10 different noise levels from 0% up to 90% spurious spikes
- 3 motifs in each dataset
- cosine similarity computed between found and ground truth motifs, $1 =$ identical, $0 =$ orthogonal
- bootstrap test to determine 5%-significance threshold of similarity measure

Results on Synthetic Data

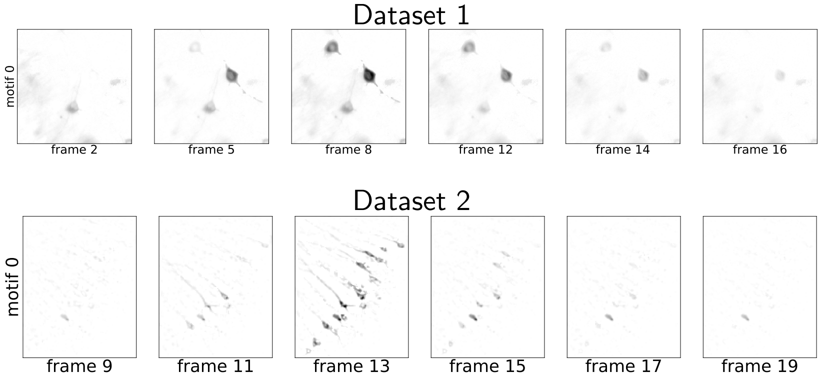


SCC = Sparse Convolutional Coding, Peter et al. (2017) [7], operating on spike matrix

red area = below BS 5% significance threshold

Results on Real Data

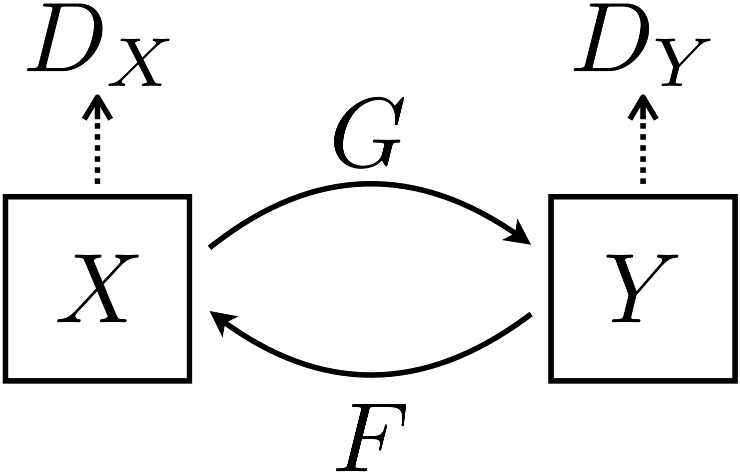
- two real datasets from hippocampal slice cultures
- in both cases one pattern identified



Section 4

CycleGAN

CycleGAN



CycleGAN

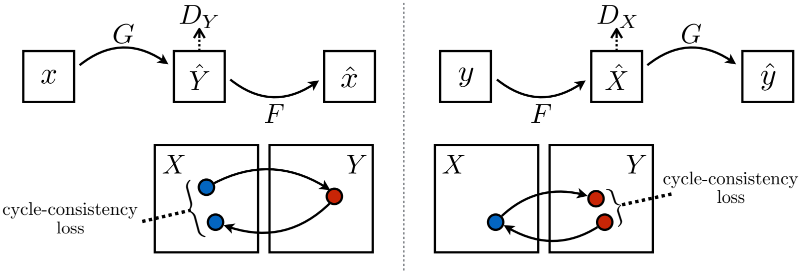


Figure taken from [8]

CycleGAN

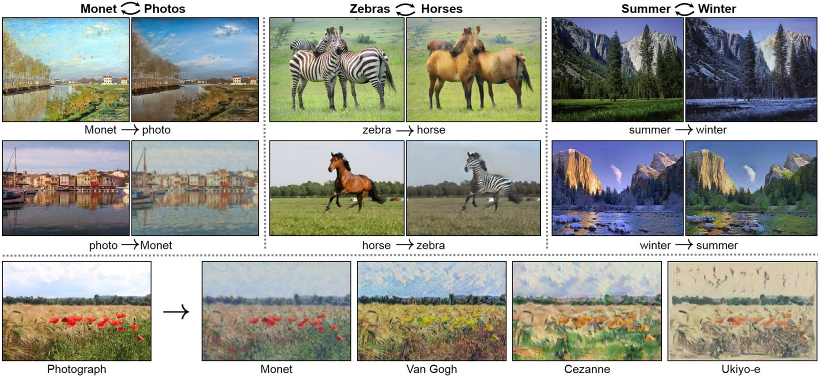


Figure taken from [8]

CycleGAN

Figure taken from https://junyanz.github.io/CycleGAN/

References

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[2] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, 2017.

[3] E. Kirschbaum, M. Haußmann, S. Wolf, H. Sonntag, J. Schneider, S. Elzoheiry, O. Kann, D. Durstewitz, and F. A. Hamprecht, "Lemonade: Learned motif and neuronal assembly detection in calcium imaging videos," in *ICLR*, 2019.

[4] C. Maddison, A. Mnih, and Y. Whye Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *ICLR*, 2016.

[5] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *ICLR*, 2017.

[6] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[7] S. Peter, E. Kirschbaum, M. Both, L. Campbell, B. Harvey, C. Heins, D. Durstewitz, F. Diego, and F. A. Hamprecht, "Sparse convolutional coding for neuronal assembly detection," in *NIPS*, 2017.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Questions?