Department of Physics and Astronomy
University of Heidelberg

Bachelor Thesis in Physics
submitted by

Maximilian Hammermann

born in Worms (Germany)

2022

Identification of inelastic interactions of light antinuclei in the Transition Radiation Detector in ALICE using Graph Neural Networks

This Bachelor Thesis has been carried out by Maximilian Hammermann at the

Physikalisches Institut in Heidelberg

under the supervision of

Prof. Dr. Klaus Reygers.

Abstract

This bachelor thesis deals with the identification of inelastic interactions of light antinuclei within the ALICE detector volume using neural networks operating on graph-structured data. For this purpose, data from a Monte Carlo simulation of p–Pb collisions at $\sqrt{s_{\mathrm{NN}}} = 5.02\,\mathrm{TeV}$ is used, in which primary tracks of light antinuclei such as antideuterons ($\overline{\mathrm{d}}$), antitritons ($\overline{\mathrm{t}}$) and antihelium-3 ($^{3}\overline{\mathrm{He}}$) are injected. The starting point for the graph creation consisted of so-called TRD tracklets that were constructed from a full detector simulation within the Transition Radiation Detector (TRD).

In this bachelor thesis, the creation of a data basis for an appropriate neural network is described, which includes a preselection of relevant data from the baseline Monte Carlo and the corresponding reconstructed information. Different approaches for graph constructions are tested in the context of first neural network architectures, which are optimized in further steps. After the final training of two network models, their performance was evaluated while comparing the classification results for different antinuclei and particles with different transverse momentum p_T. The performances of the neural networks were further compared with that of two simple cut-based approaches and a simple random forest, that all relied on handcrafted features which were constructed from the TRD tracklet information as well. It was shown that the neural networks surpass the performance of these simple models and were especially able to produce a clearly higher signal purity at equal signal efficiency. At the end of this thesis, a first outlook towards the potential deployment of the achieved results in a future physics analysis is provided.

Zusammenfassung

Diese Bachelorarbeit befasst sich mit der Identifikation unelastischer Wechselwirkungen leichter Antikerne innerhalb des ALICE Detektorvolumens mithilfe von neuronalen Netzwerken, die auf Daten mit einer Graphstruktur operieren. Hierzu wurden Daten aus einer Monte Carlo Simulation von p–Pb-Kollisionen bei $\sqrt{s_{\mathrm{NN}}} = 5.02\,\mathrm{TeV}$ verwendet, in die Primärspuren leichter Antikerne wie Antideuteronen ($\overline{\mathrm{d}}$), Antitritonen ($\overline{\mathrm{t}}$) und Antihelium-3 ($^{3}\overline{\mathrm{He}}$) eingebettet sind. Die Grundlage für die entsprechenden Graphkonstruktionen bildeten sogenannte TRD-Tracklets, die aus einer vollen Detektorsimulation innerhalb des Übergangsstrahlungsdetektors (TRD) konstruiert wurden.

In dieser Bachelorarbeit wird die Erstellung einer Datengrundlage für ein entsprechendes neuronales Netzwerk beschrieben, was eine Vorselektion relevanter Daten aus zugrundeliegenden Monte Carlo bzw. rekonstruierten Informationen beinhaltet. Es werden verschiedene Methoden von Graphkonstruktionen im Rahmen erster Netzwerkarchitekturen getestet, die in weiteren Schritten optimiert werden. Nach dem abschließenden Training zweier Netzwerkmodelle wurde deren Performance für unterschiedliche Antikerne und Teilchen mit unterschiedlichem transversalen Impuls p_T untersucht. Außerdem wurde die Performance der neuronalen Netze mit derjenigen zweier schnittbasierter Ansätze und einem einfachen Random Forest verglichen. Diese Modelle operierten allesamt auf handgefertigten Features aus den gleichen TRD-Tracklet-Informationen. Dabei zeigte sich, dass es mithilfe der neuronalen Netze möglich ist, die Performance dieser einfachen Modelle zu übertreffen und insbesondere eine deutlich höhere Signalreinheit bei gleicher Signaleffizienz zu gewinnen. Abschließend wird ein erster Ausblick auf den potentiellen Einsatz der erzielten Resultate im Kontext künftiger Physikanalysen gegeben.

Contents

1 Introduction and motivation

In the current understanding of galaxy formation, every galaxy is assumed to be embedded into a halo of dark matter [1]. The measurement of cosmic ray antinuclei such as antideuterons ($\overline{\mathrm{d}}$) and especially antihelium-3 ($^{3}\overline{\mathrm{He}}$) at energies of $E \lesssim 1\,\mathrm{GeV/nucleon}$ [2] can serve as a probe for the annihilation or decay of weakly interacting dark matter into standard model particles [3] within the mentioned halo. This is because standard model particles in primary decay channels result in the production of antinuclei such as antideuterons or antihelium-3, which propagate through galaxies as secondary particles over long distances. As a consequence, an increased particle flux of these antinuclei at lower energies compared to the expected signal from astrophysical background effects is hoped to be measured in spectrometer experiments in the lower-earth orbit. Two of these experiments are for example the ballon-born GAPS experiment (General Antiparticle Spectrometer) [4] or the AMS-02 experiment (Alpha Magnetic Spectrometer 2) on the International Space Station [5].

For the evaluation of these experiments, it is of critical importance to understand the occurrence of background effects from astrophysical processes, e.g. the production of antinuclei within the interstellar medium. Even though these background effects are expected to be rather small, when lower energy antinuclei are considered [6, 7], it is necessary to take these effects into account. Therefore, extensive efforts have been made in the past to theoretically and experimentally study production mechanisms of light (anti)nuclei, e.g. by the coalescence of (anti)protons and (anti)neutrons which are close to each other in phase space or by statistical hadronization models [8].

Moreover, it must be well understood, how antinuclei traverse the interstellar medium, e.g. whether and how often they annihilate with cosmic hadrons and how these effects might influence the results of the aforementioned experiments. Therefore, quantities like inelastic cross-sections of $\overline{\mathrm{d}}$ and $^{3}\overline{\mathrm{He}}$ were studied for different momentum ranges at the ALICE experiment at CERN (Conseil Européen pour la Recherche Nucléaire) under the use of different experimental proceedings and involving different parts of the detector systems [7, 9].

A Large Ion Collider Experiment (ALICE) is one of four big experiments currently operating at the Large Hadron Collider (LHC) and was designed to study properties of the quark-gluon plasma (QGP). The QGP is a state of deconfined quarks and gluons, that is believed to have been present in the early universe until about $10\,\mu\mathrm{s}$ after the big bang [10]. Properties of the QGP are predicted from the reconstruction of relativistic heavy-ion collisions, that are studied with ALICE.

However, the experimental environment has also shown its capabilities in other physics questions. Since antinuclei are copiously created in high energy proton–proton and heavy-ion collisions, they can serve as a source of these particles and the experimental setup of ALICE can provide particle identification (PID) and tracking capabilities in a high-multiplicity environment. Additionally, the various detector subsystems like the Time Projection Chamber (TPC) and especially the Transition Radiation Detector (TRD) provide different material budgets with which antinuclei can interact after having been produced in a particle collision.

Furthermore, the detectors represent a controllable experimental surrounding, that is studied in detail with Monte Carlo simulations. Within the simulations, the ground truth about the data-generating processes is known. By a subsequent comparison of the simulation results with the ones from real data taking, additional knowledge about the underlying physics can be obtained.

During the investigation of additional methods to identify inelastic interactions of light antinuclei within the ALICE detector volume, local measurements of the particles in the TRD, that are processed in the form of so-called TRD tracklets, have been shown to be useful within previous works [11]. In particular, the topology of these tracklets in the detector is supposed to give hints about the underlying physical processes. Interpreting this kind of information, that is in addition largely heterogeneous, is one example of a use case for modern deep learning techniques operating on graph-structured data. This is the case since TRD tracklets that are reconstructed nearby inside the detector are often related and even causally linked by underlying physical processes. Consequently, the locality and adjacency information of tracklets in the modular layer structure of the TRD can form a physically justifiable basis for a graph construction. The framework of the so-called graph neural networks (GNNs) then allows for combining the intrinsic geometric situation with the fact, that an expressive high-level feature selection under use of TRD tracklets providing a description of the underlying process is a priori not really intuitive. Furthermore, neural networks have already shown their capabilities within a large set of applications at other high energy physics (HEP) experiments [12]. In this thesis, it will be investigated whether GNNs are in fact applicable to the introduced physics problem and whether positive results can be gained from the considerations in general, that could be used within future physics analyses on this or similar problems.

In order to approach this, the thesis is structured as follows: In chapter 2, an introduction to the components of the ALICE experiment is given, which were mostly relevant for this work. Chapter 3 introduces general concepts of deep learning and explains the underlying principles of graph neural networks that were important within the scope of this thesis. In chapter 4, the information extraction from the used data sample is explained which serves the baseline for respective graph constructions and the first preliminary experiments on a suitable neural network architecture in chapter 5. Chapter 6 explains further modifications of the network architecture and presents the results of the final trainings. In chapter 7, the performance of the trained GNNs is compared to that of three other classification approaches, which relied on cut-based analyses and the deployment of a random forest algorithm with simple handcrafted features. In chapter 8, first steps towards the potential deployment of a GNN in further analyses are discussed by describing one possibility to determine a physically motivated working point. Chapter 9 provides a conclusion of the thesis with a final discussion of possible improvements and remaining research questions for the future.

2 The ALICE experiment

The main research objective of the ALICE experiment is the investigation of high energy p–p-, p–Pb- and Pb–Pb-collisions. Especially in heavy-ion collisions, a very high particle multiplicity is created which is a big challenge to cope during detector construction regarding position and energy resolution for particle identification and tracking tasks [13].

During the Long Shutdown 2 at the LHC, upgrades have been performed to improve the experimental setup for LHC Run 3. However, the used data within this thesis relies on the ALICE detector configuration as it was used during LHC Run 2 (cf. chapter 4). Therefore, a schematic drawing of the experiment during Run 2 is given in figure 1 including an illustration of the ALICE coordinate system.

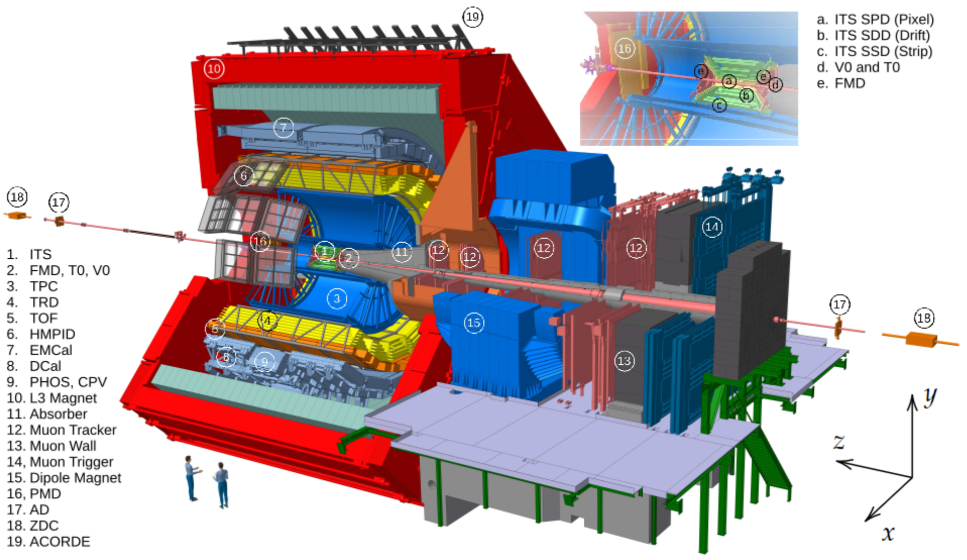


Figure 1: Schematic overview of the ALICE apparatus during LHC Run 2. The ALICE coordinate system is shown in the lower right corner with the z-axis pointing along the beam direction and the origin defined to be coinciding with the nominal interaction point. The respective parts of the central barrel and the muon arm are labeled with numbers on the left-hand side. Particular reference should be made to the Inner Tracking System (1), the Time Projection Chamber (3) and the Transition Radiation Detector (4). Figure adapted from [14].

The ALICE experiment is divided into the muon arm and the central barrel, but only the latter is of relevance in the scope of this thesis. Within the central barrel, multiple detector systems are arranged in a layer structure each fulfilling different tasks within the above-mentioned investigations. The barrel is surrounded by a solenoid magnet, which provides a magnetic field of $B = 0.5\,\mathrm{T}$ along the z-direction of the experiment.

In the following, the detector systems of ALICE, which were the most relevant for the physics of this thesis, will be introduced. This includes an explanation of the Inner Tracking System (ITS) and the Time Projection Chamber (TPC) which were mostly important because of the tracking tasks they fulfill. Moreover, the Transition Radiation Detector (TRD) is introduced in detail. Finally, a brief summary of the construction of TRD tracklets is given as well as a short overview of the material budget within the detectors in the ALICE central barrel.

2.1 Inner Tracking System (ITS)

The Inner Tracking system (ITS) is the innermost detector system of ALICE. During LHC Run 2, it consisted of six layers of silicon detectors covering together a radial range from 3.9 cm to 43 cm and a pseudorapidity range of $|\eta| < 0.9$ [15]. The first two layers of the ITS were made of silicon pixel detectors, which are surrounded by two layers of silicon drift detectors and finally two layers of silicon strip detectors. The main functions of the ITS are a high-resolution reconstruction of the primary vertex and the tracking respectively identification of low-momentum particles, which can not reach the Time Projection Chamber (TPC) [15]. Additionally, it can be used to improve the momentum and angular resolution of the TPC for particles with higher momenta.

Over the ALICE upgrade in the Long Shutdown 2, the ITS was replaced by a new system built out of seven cylindrical layers of silicon pixel detectors. It was possible to enlarge the active area to about $10\,\mathrm{m}^2$, shift it closer to the beam line due to a size reduction of the ALICE beam pipe and reduce its material budget significantly to a level of about $X/X_0 = 0.3\%$ per detector layer [16].

2.2 Time Projection Chamber (TPC)

The main purposes of the Time Projection Chamber (TPC) are the identification of charged particles, the associated track reconstruction and momentum determination. Its active area ranges from an inner radius of approximately 85 cm to an outer radius of 250 cm and covers the full azimuthal range and a pseudorapidity interval of $|\eta| < 0.9$ [15]. It consists of a cylinder with a volume of about $90\,\mathrm{m}^3$ and is filled with a gas mixture that is made of Ne, CO_2 and N_2 [15]. When charged particles traverse the TPC, they ionize the gas mixture. The so produced ionization electrons drift to the end plates of the TPC due to the application of an internal electric field. During LHC Run 2, these ionization electrons had been detected under the use of 72 multiwire proportional chambers (MWPC) with cathode pad readouts. Over the Long Shutdown 2, the MWPCs got replaced by so-called gas electron multipliers (GEMs) together with a new readout electronics setup. This allows to execute continuous readouts at collision rates of up to 50 kHz in Pb–Pb-collisions during LHC Run 3 [17].

In order to perform particle identification, the specific energy loss of a particle $\mathrm{d}E/\mathrm{d}x$ is measured together with the particle momentum and its charge. The expected specific energy loss per unit length due to ionization processes can then be described by parameterizations based on the Bethe-Bloch-formula, which are for example further described in [18]. By a comparison of expected energy loss curves with the actual measured energy loss as a function of momentum, a hypothesis for the true particle species can be set up.

Examples of expected energy loss curves are shown in figure 2 together with actual measurements from the TPC. It can be seen that antihelium-3 ($^{3}\overline{\mathrm{He}}$) can be clearly separated from lower-mass particles even at high momenta, which is a consequence of the fact, that the nucleus is doubly negatively charged. Furthermore, the discrimination of antideuterons ($\overline{\mathrm{d}}$) and antitritons ($\bar{t}$) from lower-mass particles can also be performed with a high purity in a wide momentum range.

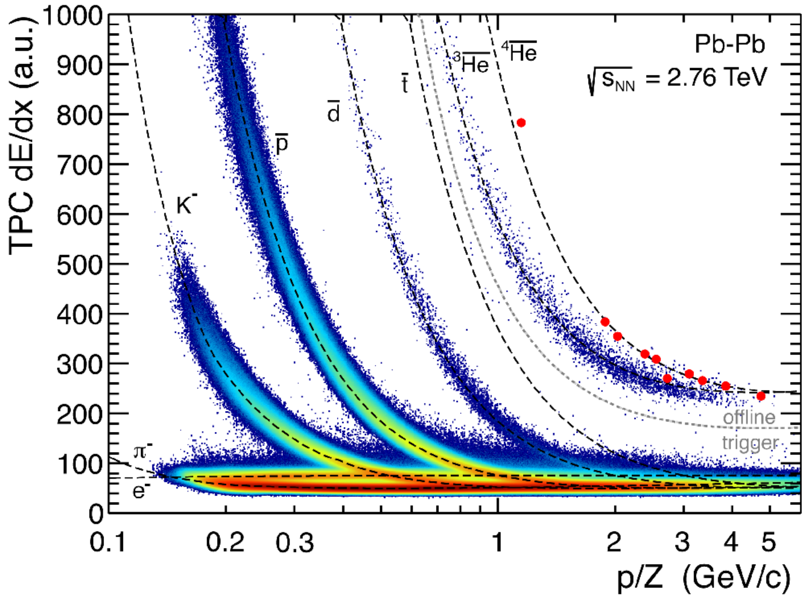


Figure 2: Energy loss of negatively charged particles in the TPC as a function of particle momentum/charge (p/Z). The black dashed lines indicate the expected energy loss for the respective particles according to the Bethe-Bloch-formula. Figure adapted from [18].

2.3 Transition Radiation Detector (TRD)

The Transition Radiation Detector (TRD) at ALICE is an ensemble of 522 individual detector chambers, that are arranged in 18 super modules around the beam line (labeled with numbers from 0 to 17). Each super module itself consists of five stacks along the beam direction (labeled from 0 to 4) and six layers aligned in the radial dimension (labeled from 0 to 5 and ranging from positions of 2.90 m to 3.68 m)[1]. With this, a full azimuthal and a pseudorapidity coverage of $|\eta| < 0.84$ is achieved [13]. The TRD has on the one hand been constructed to discriminate electrons and hadrons (e.g. pions) in high multiplicity events, but it is on the other hand of importance in the tracking of charged particles, which is the mainly relevant aspect during this thesis.

The operating principle of one TRD chamber is based on the one of a multiwire proportional chamber (MWPC) with a drift region inside a Xe-CO$_2$-gas atmosphere. A radiator is installed in front of the chamber and a pad readout is mounted at the back. This is shown in a schematic overview in figure 3. When charged particles traverse the so-called drift region of the MWPC, they ionize the detector gas in the corresponding volume. The so produced ionization electrons travel along the drift lines indicated in figure 3 and induce an ionization cascade in the detector gas [19] that is used to perform a d$E/$dx-measurement of the incoming

[1]In super modules 13 to 15, no TRD chambers were installed in stack 2 to reduce the material budget in front of the PHOS detector, which can be seen in figure 1.

particles. This allows to directly discriminate different particle species based on their specific energy loss.

The radiator itself is constructed in a sandwich structure of different foams and fibers, which are described in detail in [13]. Incoming highly relativistic particles with a Lorentz factor of $\gamma \gtrsim 1000$ pass the radiator and emit Transition Radiation (TR) when they cross the boundaries of media with different dielectrical constants. The produced TR photons in the X-ray region then enter the drift region of the MWPC (cf. figure 3) and ionize gas particles with a high probability such that an additional voltage signal in the readout chamber for these high-γ particles is measured. This allows further discrimination of hadrons and electrons, as their huge mass difference leads to different orders of magnitude in the Lorentz factor γ at similar momenta.

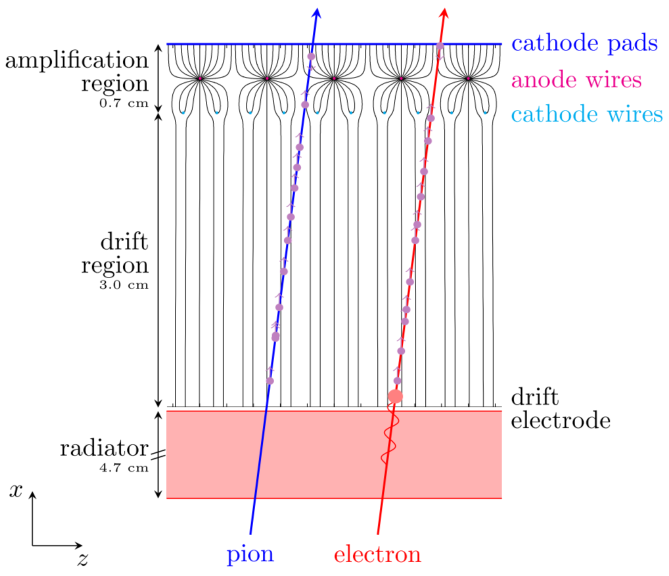


Figure 3: Schematic cross-section of a TRD chamber. In the lower left corner, a corresponding local coordinate system of the chamber is shown, which results from a rotation of the global ALICE coordinate system. The length of the radiator is not drawn to scale with respect to the other components. The energy deposition in the chamber due to TR photons of a highly relativistic electron is indicated by a red dot. Figure taken from [13].

2.4 Construction of tracklets and calibration

As pointed out before, the TRD offers the capability for charged particle tracking. This task is usually performed by the use of Kalman filtering such that the finding and the fitting of the respective tracks are performed at the same time [13]. To apply a Kalman filter-algorithm within a TRD tracking, the positional information of a particle traversing a single TRD chamber is usually represented by a local track segment, which is called tracklet [11]. The construction of these tracklets is based on an individual calibration technique, that was inter alia newly developed to provide the datasets that are used within this thesis [20].

Each tracklet consists of a three-dimensional offset point and a three-dimensional direction information. They are constructed by performing a time-binned readout of the pad voltage signal in each TRD chamber with analog digital converters (ADCs) using a time bin width of $\Delta t = 100\,\text{ns}$. Typically, the ADC readout provides signals in 24 time bins [13]. This information can afterwards be used to perform a straight-line fit to the time-binned signals of all pad positions within one TRD chamber using a singular value decomposition [20] after certain corrections, such as for the $\mathbf{E} \times \mathbf{B}$-effect on the motion of the ionization electrons in the drift region [13], were applied to the respective pad positions.

2.5 Material budget in the central barrel

All the aforementioned parts of the experiment have different material budgets, which is of importance when considering the occurrence of inelastic interactions between antinuclei and hadronic components of each detector. To illustrate this, the cumulative material budget in the ALICE detectors in the central barrel is shown in figure 4 looking at straight primary tracks.

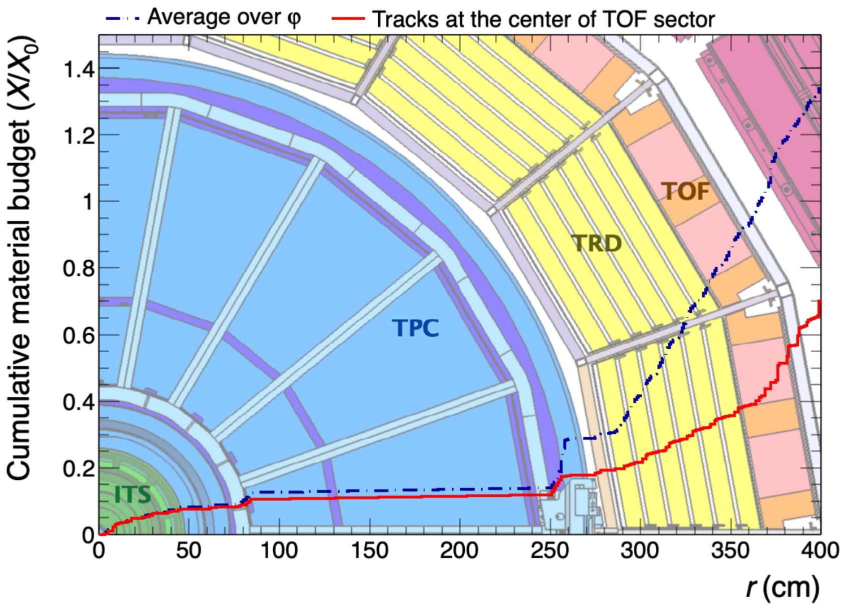


Figure 4: Cumulative material budget in the detectors of the ALICE central barrel as a function of radial distance for straight primary tracks. The blue curve represents an average taken over the full azimuth for particles that were emitted perpendicularly to the beam line. Figure taken from [9].

It can be seen, that the cumulative material budget rises considerably within in the TRD, which is the reason for an increased interaction probability of antinuclei. This is mainly due to the fact that the radiator ($X/X_0 = 0.69\,\%$), the pad planes ($X/X_0 = 0.77\,\%$) and the readout electronics ($X/X_0 = 1.18\,\%$) contribute significantly to the cumulative amount of radiation lengths, which is specifically for one TRD chamber given by $X/X_0 = 2.85\,\%$ for particles with normal incidence [13].

3 Deep learning and graph neural networks

In this chapter, an introduction to relevant concepts concerning graph neural networks (GNNs) will be given. Therefore, the basics of deep learning architectures such as fully-connected networks (FCNs) and convolutional neural networks (CNNs) will be discussed. After that, graph neural networks will be introduced as an example from the field of geometric deep learning, which is a generic term for deep learning techniques that do not operate on Euclidean domains [21].

First, it will be shown how graph neural networks can be generally introduced using the so-called graph network (GN) framework with a special focus on the so-called message-passing neural networks (MPNNs). Furthermore, it will be motivated how the process of convolution on ordered vectors in CNNs can be generalized to unordered sets in which relationships between different parts of the dataset are considered. This is known as the process of graph convolution. As a final step, the working principle of so-called graph attention networks (GATs) will be explained, which was an important concept regarding the GNN construction that was performed within this thesis.

3.1 Deep learning terminology, fully-connected and convolutional neural networks

Neural networks are nowadays very popular, since they can learn very complex nonlinear functions and especially produce high-level information output from sets of low-level features. This is beneficial, because classical machine learning algorithms like random forests (RFs) or boosted decision trees (BDTs) rely on handcrafted features that represent the underlying processes of data occurrence satisfactorily [22]. Features like this can be hard to construct, when the data-generating processes get very complicated or the provided notion of features is rather abstract [23]. However, the application of neural networks often results in models that are difficult or nearly impossible to interpret in terms of their decision process, which is a downside in the scope of a physics analysis.

3.1.1 Fully-connected networks

Fully-connected networks[2] (FCNs) are known as the simplest architecture, that is used in deep learning, but can also serve as a powerful baseline. They consist of multiple layers of neurons and each neuron of a certain layer is connected to all the neurons in the subsequent layer, such that a maximally dense network structure is created. A neuron is the smallest processing unit within a neural network and computes one-dimensional outputs $x' \in \mathbb{R}$ by weighting entries of a multidimensional input $\mathbf{X} \in \mathbb{R}^d$ [24]. Fully-connected layers were applied in this thesis to provide the final classification output of the constructed graph neural network (cf. chapter 6).

[2]A fully-connected network is in many applications also often called multilayer perceptron (MLP).

During one specific layer l, a feature input vector $\mathbf{X}_l \in \mathbb{R}^d$ is linearly transformed by a learnable weight matrix $W_l \in \mathbb{R}^{d' \times d}$ and added with a learnable bias term $\mathbf{b}_l \in \mathbb{R}^{d'}$ [12]. It must be noted that every neuron connection between two layers is assigned an independent weight such that there is no weight-sharing between adjacent neurons.

Afterwards, a nonlinear activation function φ, e.g. the often used rectified linear unit (ReLU), which is for some $x \in \mathbb{R}$ defined as

$$\mathrm{ReLU}(x) = \max(x, 0), \tag{1}$$

is applied elementwise to the output. This provides a nonlinear transformation of the original input $\mathbf{X}_l$ in the layer l to the input vector $\mathbf{X}_{l+1}$ of the layer $(l+1)$,

$$\mathbf{X}_{l+1} = \varphi(W_l \mathbf{X}_l + \mathbf{b}_l). \tag{2}$$

After a certain number of repetitions, a final output layer is constructed. The form of this layer depends on the use case of the network. In classification tasks, which is the kind of problem, that was relevant within this thesis, it typically contains C output neurons, where C denotes the number of different classes in the baseline classification problem. Like this, a score vector $\mathbf{S}_i \in \mathbb{R}^C$ can be computed for every data instance i, the components of which can be used to derive a class prediction $\hat{Y}_i$ of the model according to the frequently used decision rule

$$\hat{Y}_i = \underset{k \in \{0,\dots,C-1\}}{\mathrm{argmax}} \; (\mathbf{S}_i)_k. \tag{3}$$

The score vector $\mathbf{S}_i$ is in practice often normalized by applying a softmax function, such that the final model outputs can be interpreted as posterior probabilities. The softmax function [12] for a vector $\mathbf{x} \in \mathbb{R}^d$ is defined componentwise as

$$(\mathrm{Softmax}(\mathbf{x}))_j = \frac{\exp(x_j)}{\sum_{m=1}^{d} \exp(x_m)}. \tag{4}$$

In the context of this thesis, a binary classification task is considered: For a respective light antinucleus, it should be determined whether it underwent an inelastic interaction process in a predefined detector volume or not. Further details, how the respective class definitions were implemented, are explained in chapter 4.

In order to quantify the quality of the model prediction in a differentiable way, a loss function $\mathcal{L}$ is introduced. The specific form of the loss function depends on the application and for the case of classification tasks, the so-called cross-entropy loss is often used [25]. It is calculated from the known true class identity Y_i^* of an instance i, the derived class prediction $\hat{Y}_i$ and the normalized score vector $\tilde{\mathbf{S}}_i = \mathrm{Softmax}(\mathbf{S}_i)$, that is computed from the model output, with

$$\mathcal{L}(Y_i^*; \hat{Y}_i, \tilde{\mathbf{S}}_i) = -\sum_{k=0}^{C-1} \mathbb{1}[Y_i^* = \hat{Y}_i] \log\big((\tilde{\mathbf{S}}_i)_k\big), \tag{5}$$

where $\mathbb{1}$ is an indicator function, that is non-vanishing, only if $Y_i^* = \hat{Y}_i$.

The basic task is to find a set of parameters for the model such that the value of the loss function $\mathcal{L}$ gets minimized for a given training set. This is performed using gradient-based optimizers, e.g. gradient descent or ADAM [24], that are updating the parameters of the model using information about the gradients of the loss $\mathcal{L}$ in parameter space. The step width in parameter space is controlled via an adjustable hyperparameter, the so-called learning rate τ. The training algorithm that is used for efficient gradient management while making use of the layer structure of the neural network is called backpropagation [25]. During backpropagation, the model is evaluated while performing a so-called forward pass, where the respective gradients of the loss with respect to the outputs of each layer are computed and stored in memory. After that, the gradients of $\mathcal{L}$ with respect to its parameters are derived by taking the gradients of the loss during the forward pass and a backward application of the chain rule [12].

The above-mentioned gradients are typically updated using consecutive evaluations on random subsets of the training set, which are called batches. One reason for this strategy is that processing the complete training set at once could lead to difficulties, when the training set is too large to fit into memory. This might especially be the case, when the training is performed on GPUs to take advantage of their highly-parallelized computation infrastructure. The training is then performed in epochs, during which the complete training set gets processed once.

3.1.2 Convolutional neural networks

Convolutional neural networks (CNNs) represent a type of network that has led to revolutions in image classification and segmentation problems [25, 26]. This success resulted from the fact, that convolutional processes on pixel-based feature inputs can reveal underlying patterns in images, which purposefully incorporate feature properties in the neighborhood of certain pixels. To illustrate this, a convolutional process on a two-dimensional feature input is shown in figure 5.

A kernel function, which can be seen as a window that runs over the input, outputs a different representation of the features compared to the original one. The size of this representation depends on the definition of the kernel window, e.g. regarding the kernel size, the kernel stride or the definition of a feature padding [12]. Whereas the kernel stride is a term for the step width of the kernel function as it runs over the input, the feature padding describes the behavior of the kernel at the edges of the feature inputs. Since there might be multiple patterns of interest in the data, one convolutional layer in a CNN typically applies multiple kernels that are all convolved with the input data and provides therefore multiple so-called output channels [25].

Additional to a potential size reduction of the original features, the network structure associated with the convolutional process results in a much sparser connectivity between the neurons in subsequent layers than in a fully connected network [27]. This is because only entries in some neighborhood of a certain point in the original feature input contribute to one specific element in the output, which is a priori not the case in a fully-connected network. The notion of neighborhoods and locality is defined by the specific kernel functions that are used for the convolution.

Furthermore, the parameters that are used in each kernel function, are jointly learned based on all feature outputs that are computed, as each kernel window is slidden over the total amount of input features. This ensures that patterns in the input can be filtered in a way that is not affected by spatial translations.

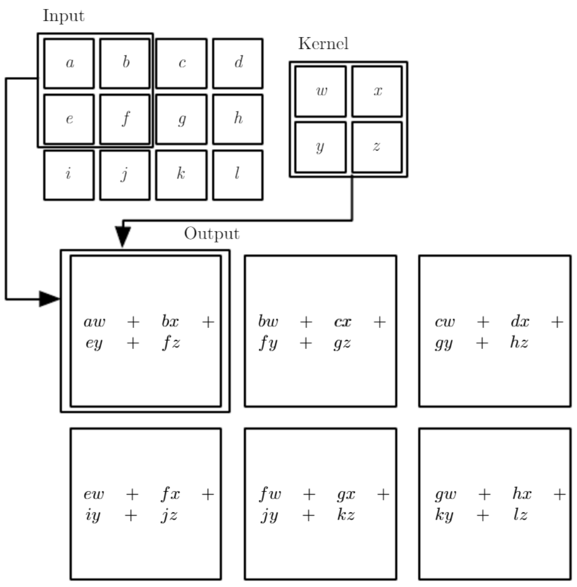


Figure 5: Process of convolution for a two-dimensional feature input. The defined kernel window runs over the feature input and produces an output of reduced size in this setting. The learnable weights of the kernel are shared over the whole feature input. Figure taken from [24].

3.2 Framework for graph neural networks

The working principle of the previous methods were based on the fact that some ordering of the underlying feature inputs was possible in a canonical way, e.g. by representing pixels within vectors or matrices, such that a natural sense of neighborhoods and relations between different elements existed. However, a way to achieve this kind of ordering might not always be very obvious, especially when the input information can not easily be represented by tabular data. This problem is addressed by the so-called graph network (GN) framework that was first presented in [27]. It builds the environment for the construction of general neural networks that work on unordered sets with some bias on the relation between the set elements. In the context of this thesis, only a subclass of these networks, the so-called message-passing networks, were considered. The principle of these networks will be explained in more detail in section 3.3. The specific application of this approach to the given physics problem is further explained in chapter 5.

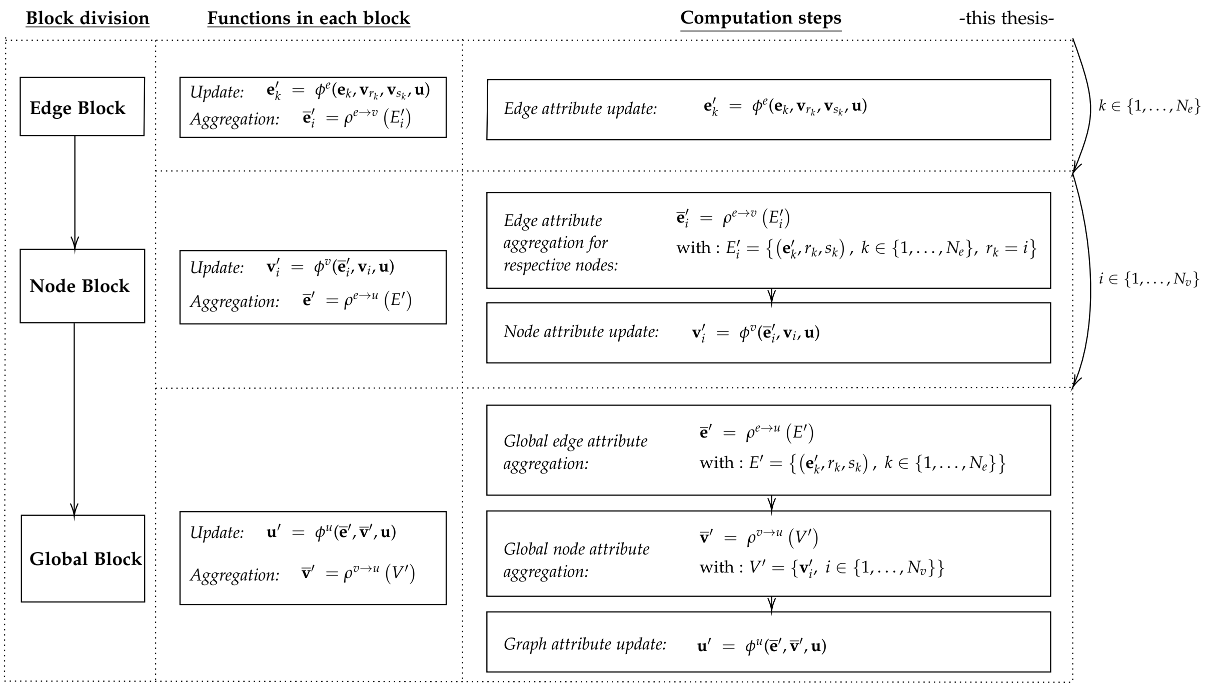
The basic idea of the graph network framework is, that unordered sets can be assigned a relational structure by representing them as a graph with certain edges and nodes. Due to the edges, a natural representation of relationships between different graph nodes can be obtained. The feature information on which the neural networks operate are then assigned to the different elements in the graph, e.g. the nodes or the edges.

Abstractly, a graph $\mathcal{G}$ can be represented as a 3-tuple, $\mathcal{G} = (\mathbf{u}, V, E)$, where $\mathbf{u}$ denotes a vector of attributes of the whole graph. V is the set of nodes associated with node feature vectors $\mathbf{v}_i$ such that $V = \{\mathbf{v}_i, i \in \{1, \dots, N_v\}\}$ and E is the set of edges associated with edge feature vectors $\mathbf{e}_k$ such that $E = \{(\mathbf{e}_k, r_k, s_k), k \in \{1, \dots, N_e\}\}$. N_v and N_e are the respective numbers of nodes and edges in a graph and r_k and s_k are the indices of the receiver/sender node, which are connected by a given edge k.

The graph network itself consists of three building blocks that increase in their hierarchical position: The edge block, the node block and the global block. Each of these blocks contains one update function ϕ, that computes updates of either edge, node or graph attributes. Furthermore, every block has one aggregation function ρ, that is used in the computation steps within certain updates. In this general setting, the specific form of these functions does not need to be specified. Only the aggregation functions ρ need to be constructed such that their output is invariant under permutations of the respective inputs. This reflects the requirement that the GN framework is robust against the occurrence of graph isomorphies. Broadly speaking, two graphs are isomorphic, if there exists a bijective map between them to interchange nodes such that the edge and the label structure of the nodes in the graph is preserved [28]. This fundamental symmetry is comparable to invariance under translations in a convolutional neural network, that has been mentioned before. The dependencies of each graph network function and their specific deployment within an operational step of a GN are illustrated in figure 6.

Initially, the features of all edges $\mathbf{e}_k$ are updated with the update function ϕ^e using the node features $\mathbf{v}_{r_k}$ and $\mathbf{v}_{s_k}$, i.e. the nodes that are connected by the respective edge, and the global graph attributes $\mathbf{u}$. In the next step, the features of each node get updated to $\mathbf{v}'_i$ via the node update function ϕ^v under use of the former node features $\mathbf{v}_i$, the global graph features $\mathbf{u}$ and the aggregated updated features of those edges that have the respective node as a receiver. The aggregation is performed using the aggregation function $\rho^{e \rightarrow v}$ of the edge block. In the last step, an update of the graph-level features is performed. Therefore, the beforehand updated node and edge features are globally aggregated using the aggregation functions $\rho^{e \rightarrow u}$ and $\rho^{v \rightarrow u}$. Subsequently, updated graph-level features $\mathbf{u}'$ are computed using the graph update function ϕ^u.

This very general setting has the advantage that a huge class of networks can be embedded into this framework. However, the application of the presented ideas to specific problems, especially regarding the network implementation in detail, is not straightforward. Therefore, multiple subclasses of graph networks have been developed with further constraints on their operational steps.



Block division
Functions in each block
Computation steps
-this thesis-
Edge Block
Node Block
Global Block
Update: $\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$
Aggregation: $\bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i)$
Update: $\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$
Aggregation: $\bar{\mathbf{e}}' = \rho^{e\to u}(E')$
Update: $\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$
Aggregation: $\bar{\mathbf{v}}' = \rho^{v\to u}(V')$
Edge attribute update: $\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$
Edge attribute aggregation for respective nodes: $\bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i)$ with : $E'_i = \{(\mathbf{e}'_k, r_k, s_k), k \in \{1,\ldots,N_e\}, r_k = i\}$
Node attribute update: $\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$
Global edge attribute aggregation: $\bar{\mathbf{e}}' = \rho^{e\to u}(E')$ with : $E' = \{(\mathbf{e}'_k, r_k, s_k), k \in \{1,\ldots,N_e\}\}$
Global node attribute aggregation: $\bar{\mathbf{v}}' = \rho^{v\to u}(V')$ with : $V' = \{\mathbf{v}'_i, i \in \{1,\ldots,N_v\}\}$
Graph attribute update: $\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$
$k \in \{1,\ldots,N_e\}$
$i \in \{1,\ldots,N_v\}$

Figure 6: Operational steps in a general graph network. The curved arrows on the right-hand side indicate loops over all respective edges/nodes. The used notation is explained in the text.

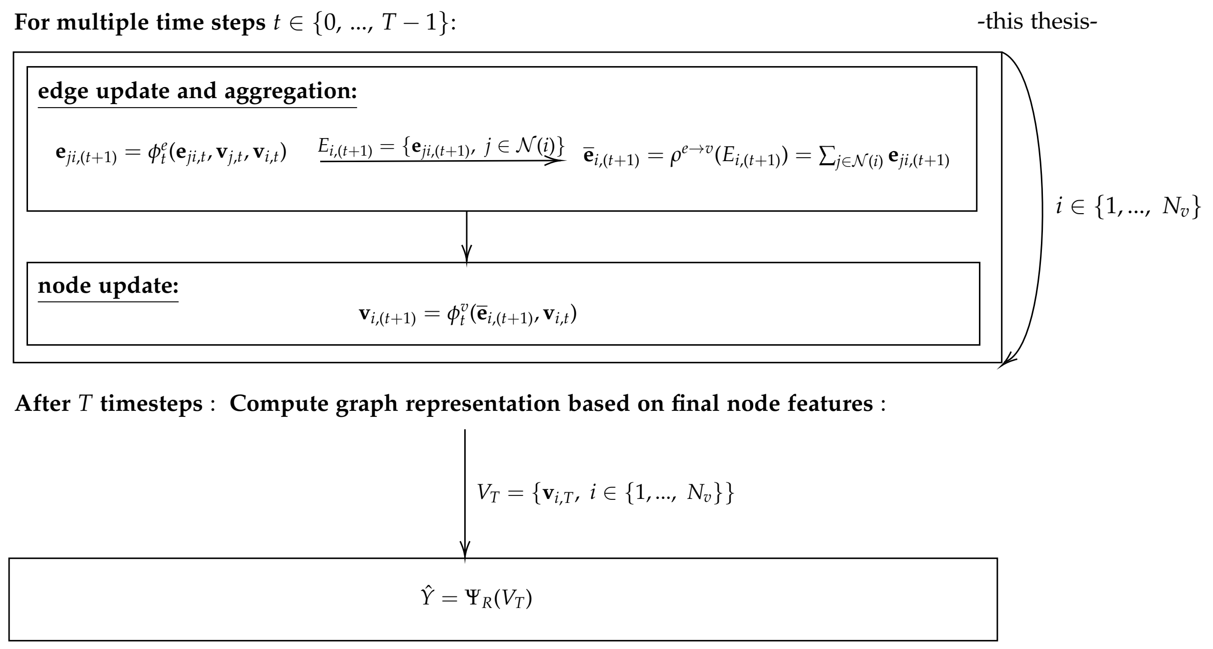
3.3 Message-passing neural networks

During this thesis, a subclass of graph networks was considered, that has e.g. raised attention in quantum chemistry due to successfully predicting the properties of certain molecules [29]. This type of implementation is called message-passing neural network (MPNN) and will be further explained for the specific setting of supervised graph predictions, which was the baseline problem considered within this thesis. There exist further use cases of MPNNs, e.g. the inference on properties of single nodes in a graph or the prediction of edge features. Further references, especially regarding the application of MPNNs in HEP, are for example summarized in [23, 12].

The underlying idea of a MPNN is that nodes exchange messages with their nearest neighbors repeatedly along their edges for a given number of time steps [29]. The output information, that results from feature propagation during a certain time step t, serves as an input for the computation of updated node features belonging to the subsequent time step $(t+1)$. After T time steps, the node features are passed to a permutation-invariant readout function, that constructs a graph representation based on the final node features. This process is illustrated in detail in figure 7.

It should be noted that the notation has changed compared to the general introduction on graph networks, because the awareness of different time steps is necessary for understanding the working principle of MPNNs.

One time step is defined by a complete update procedure of the features of all nodes in a graph. For reasons of better understanding, the different nodes are here viewed separately such that one specific node i is considered here. However, the respective processes can also



For multiple time steps $t \in \{0, ..., T-1\}$:
-this thesis-
edge update and aggregation:
$\mathbf{e}_{ji,(t+1)} = \phi_t^e(\mathbf{e}_{ji,t}, \mathbf{v}_{j,t}, \mathbf{v}_{i,t})$
$E_{i,(t+1)} = \{\mathbf{e}_{ji,(t+1)},\ j \in \mathcal{N}(i)\}$
$\bar{\mathbf{e}}_{i,(t+1)} = \rho^{e \to v}(E_{i,(t+1)}) = \sum_{j \in \mathcal{N}(i)} \mathbf{e}_{ji,(t+1)}$
$i \in \{1, ..., N_v\}$
node update:
$\mathbf{v}_{i,(t+1)} = \phi_t^v(\bar{\mathbf{e}}_{i,(t+1)}, \mathbf{v}_{i,t})$
After T timesteps : Compute graph representation based on final node features :
$V_T = \{\mathbf{v}_{i,T},\ i \in \{1, ..., N_v\}\}$
$\hat{Y} = \Psi_R(V_T)$

Figure 7: Functional principle of a message-passing neural network to perform graph-level predictions. The curved arrow on the right-hand side indicates a loop over all nodes in a graph. The first box represents the actual message-passing part and is performed for T time steps. Afterwards, a graph prediction $\hat{Y}$ is computed from the final node features using a readout function Ψ_R. The used notation is explained in the text.

be parallelized for all nodes in one graph and especially be expressed in terms of matrix operations.

In a certain time step t, the features of all edges from an arbitrary node j, while having the node i as a receiver, will be updated to $\mathbf{e}_{ji,(t+1)}$, using the edge update function ϕ_t^e. As an input, the former edge features $\mathbf{e}_{ji,t}$ and the current features of the nodes, that are linked by the respective edge, are taken. The function ϕ_t^e is a differentiable function and typically implemented as a simple neural network or a linear transformation (cf. section 3.4). After that, the respective edge features for the node i are aggregated to $\bar{\mathbf{e}}_{i,(t+1)}$ using the edge aggregation function $\rho^{e \to v}$, which is typically implemented as an elementwise summation, as it is indicated in figure 7. Therefore, the index set of all neighboring nodes for a given node i is denoted as $\mathcal{N}(i)$. Specifically, j is an element of $\mathcal{N}(i)$, if there exists an edge in the graph, that points from node j to node i. Finally, the features of the node i can be updated using the former node features $\mathbf{v}_{i,t}$ of time step t and the aggregation of the edge features $\bar{\mathbf{e}}_{i,(t+1)}$ as an input for the node update function ϕ_t^v. The latter is typically also implemented as a simple neural network or as an elementwise application of a nonlinear activation function [23].

So far, only node representations have been updated. To obtain a graph embedding from the local node information, a readout function Ψ_R is applied to the final node representations after message-passing. This involves a so-called pooling mechanism, that e.g. computes the elementwise average of all node features, which is also often called global average pooling. Finally, the obtained graph representation must be transformed to an output of the desired shape, which is for example a score vector with two entries in a binary classification task and typically done under the use of an FCN.

3.4 Graph convolution

Within the framework of a MPNN, it is possible to define a sense of convolution on graphs. There exist multiple approaches to do this and all of them generalize convolutions that are used in CNNs (cf. section 3.1.2), as one can also clearly represent grids of features as graphs. One problem to cope in this context is a rigorous definition of locality on a graph. Further information about this and the theoretical foundations of the different approaches implementing graph convolutional layers can for example be found in [30].

One ansatz that is frequently used to implement convolutions on graphs and that was also deployed during first investigations on candidate network structures during this thesis is the so-called GCN-layer, which was first proposed in [31]. The update of the matrix of node representations $V^{(l+1)}$ in layer $(l+1)$ can be directly computed from the previous ones $V^{(l)}$ using stacked matrix operations

$$V^{(l+1)} = \sigma(\hat{A} V^{(l)} W^{(l)}). \tag{6}$$

$W^{(l)}$ is a layer-specific trainable weight matrix, σ a nonlinear activation function (e.g. the ReLU function) and $\hat{A}$ denotes a normalized version of the adjacency matrix A for a given graph, in which self-connections are added. This means that every node feeds back its feature information to itself during message-passing. The so applied normalization process is called neighborhood normalization. The adjacency matrix A of a graph itself is defined by

$$A_{ij} = \begin{cases} 1, \text{ if } \exists \text{ edge from node } i \text{ to node } j \\ 0, \text{ else} \end{cases} \tag{7}$$

and $\hat{A}$ is then computed as

$$\hat{A} = D^{-1/2} A D^{-1/2} \tag{8}$$

with the so-called degree matrix $D = \sum_j A_{ij}$. The inverse square root of D is meant element-wise.

It has already turned out that neighborhood normalization might cause problems during graph-level predictions, as it might have a smearing effect on particularities within local graph structures [32]. Therefore, other layers have been introduced, that omit neighborhood normalization and apply different weight matrices to self- and the neighboring connections of a given node. The impact of these effects for the classification task in this thesis will be studied in detail in chapter 5. Further information on this sense of convolution is for example presented in [28] in the more general setting of so-called higher-order GNNs.

3.5 Graph attention networks

A further possibility to improve the performance of a general graph convolution is to identify which of the neighboring nodes are most important for a given node during their aggregation whilst message-passing. This is addressed by a so-called graph attention network (GAT), which has been proposed first in [33] by a team of researchers working at DeepMind. There also exist slightly different approaches which permute the computation steps that are used in the here presented GAT [34]. However, the idea of attention mechanisms is not new to deep learning and they have already shown their usefulness in a variety of problems, e.g. natural language processing [35].

The baseline of the computations from section 3.4 remains the same with the difference that for each so-called query node i, each of the neighboring nodes j is now assigned a different normalized weight α_{ij} during node aggregation within the update procedure. The node feature updates $\mathbf{v}'_i \in \mathbb{R}^{d'}$ of former node features $\mathbf{v}_i \in \mathbb{R}^d$ in one step of message-passing are computed by

$$\mathbf{v}'_i = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \alpha_{ij} W \mathbf{v}_j\right), \tag{9}$$

where $W \in \mathbb{R}^{d'\times d}$ denotes a learnable weight matrix and σ is again a nonlinear activation function. The weight coefficients α_{ij} are computed with

$$\alpha_{ij} = (\text{Softmax}(\tilde{\mathbf{e}}_i))_j \tag{10}$$

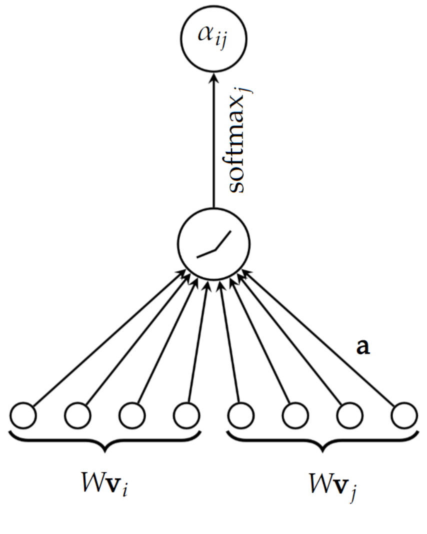
with a vector of so-called attention coefficients $\tilde{\mathbf{e}}_i \in \mathbb{R}^{(\#\mathcal{N}(i)+1)}$ of the query node i. These attention coefficients originate from an attention function that is implemented as a single-layer neural network. The structure of this network including the final softmax normalization is illustrated in figure 8(a).

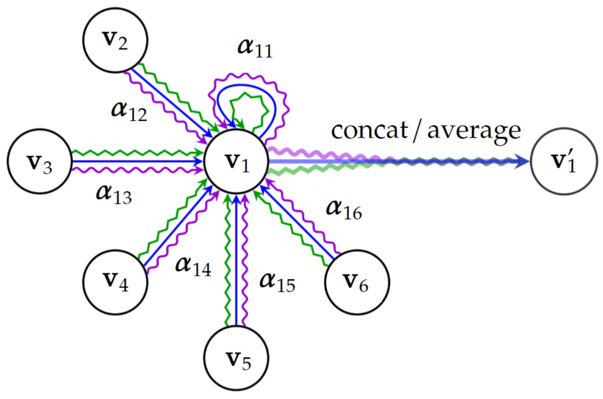
Initially, the feature vectors $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^d$ with $j \in \{1, \dots, N_v\}$ are jointly transformed by the above introduced learnable weight matrix W and then concatenated to a vector of dimension $n = 2d'$. Via an inner product with a learnable weight vector $\mathbf{a} \in \mathbb{R}^{2d'}$, a real number is obtained, which gets passed through a so-called leaky-ReLU activation

$$\text{LeakyReLU}(x;\beta) = \max(x, \beta x) \tag{11}$$

with the slope parameter $\beta = 0.2$. Subsequently, the coefficients are normalized using the relation shown in equation (10).

An additional aspect to cope with is the fact that the learning of attention coefficients might suffer from instabilities, that are in practice not desirable. This is addressed by so-called K-multihead attention during which the attention coefficient computation is performed K times in parallel with different weight matrices $W^{(k)}$ ($k \in \{1, \dots, K\}$) for the initial transformation within the network (cf. figure 8(a)). The resulting node update can be obtained by a concatenation of the results calculated with equation (9) for a complete vector of attention weights





$\mathbf{v}_2$
$\boldsymbol{\alpha}_{11}$
$\boldsymbol{\alpha}_{12}$
concat / average
$\mathbf{v}_3$
$\mathbf{v}_1$
$\mathbf{v}'_1$
$\boldsymbol{\alpha}_{13}$
$\boldsymbol{\alpha}_{16}$
$\mathbf{v}_4$
$\boldsymbol{\alpha}_{14}$
$\boldsymbol{\alpha}_{15}$
$\mathbf{v}_6$
$\mathbf{v}_5$
(b)

Figure 8: Illustration of graph attention networks. Figure (a) shows the structure of the attention network that is used to compute the normalized node weights α_{ij}. Figure (b) illustrates the update of node features in a graph for a specific node with node features $\mathbf{v}_1$ using multihead attention with $K = 3$ heads, which is indicated using arrows of different colors and forms. Further information about this is provided in the text. Adapted from [33].

$\boldsymbol{\alpha}_{ij} \in \mathbb{R}^K$ and the different weight matrices $W^{(k)}$ or by averaging over the different results before applying the final activation σ in equation (9). The first leads to an extended dimension of the updated node feature vector, while the averaging process is dimension-preserving. The process of multihead attention for a specific node and its nearest neighbors in a graph is illustrated in figure 8(b).

4 Data preselection

In this chapter, the construction of the dataset, that was used to generate an input for a graph neural network, will be explained. This was done using the *ROOT Framework* [36] and program code originating from the TRD Self Tracking Repository in [37]. The latter is written in the C++ programming language.

The simulation that built the baseline for this thesis consisted of 257.200 Monte Carlo events of p–Pb-collisions at $\sqrt{s_{\mathrm{NN}}} = 5.02\,\mathrm{TeV}$ using the detector configurations of ALICE during the LHC16q run period. In these events, light antinuclei like antideuterons ($\overline{\mathrm{d}}$), antitritons ($\overline{\mathrm{t}}$) and antihelium-3 ($^{3}\overline{\mathrm{He}}$) were injected as primary tracks to study antinuclei annihilation using the TRD as an absorber and tracker of the resulting fragments. The interaction of the antinuclei with the detector material was simulated using the GEANT4-toolkit [38]. For purposes of this thesis, the raw data were provided in the form of *ROOT*-files that contained information about the constructed TRD tracklets, a TPC track reconstruction and the true MC information of the underlying simulation.

At the beginning, the data set was investigated with a three-dimensional TRD event display. As an example, one complete p–Pb event from the dataset is shown in figure 9.

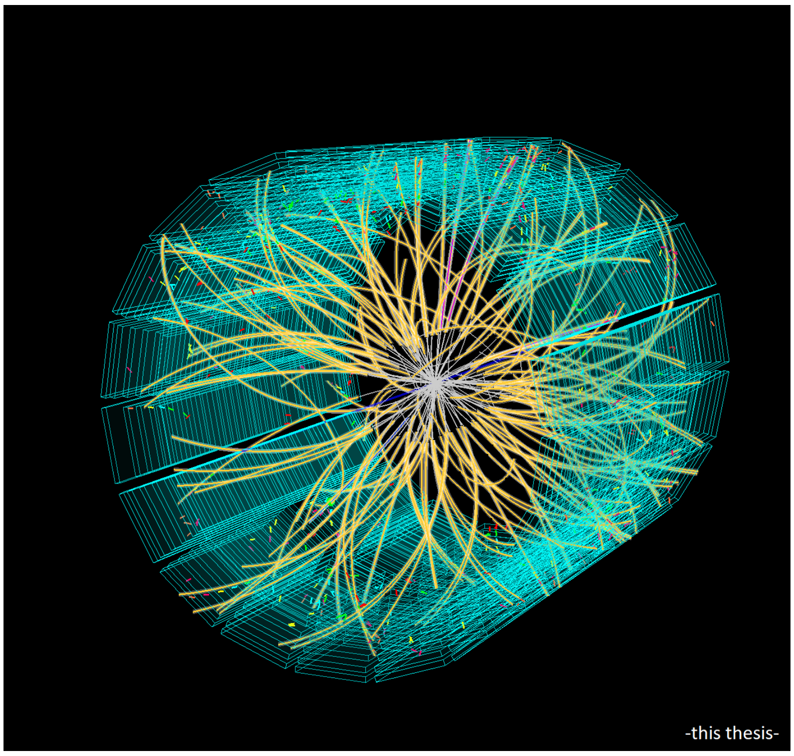


Figure 9: Three-dimensional display of a full p–Pb MC event. Different chambers of the TRD are displayed in turquoise, the reconstructed TPC tracks are shown in orange and are propagated to the outer TRD radius. The coloring scheme of the TRD tracklets reflects the TRD layer to which they belong.

The inspection of the dataset involved a filtering of the above-mentioned antinuclei using the true particle identity from the MC information and an investigation of their behavior in the detector volume with the additional graphical feedback. It was found that a lot of antinuclei undergo inelastic interaction processes at the end of the TPC volume or within the TRD chambers, during which daughter particles like light mesons (e.g. pions or kaons) or low-mass baryons and antibaryons (e.g. (anti)protons and (anti)neutrons) are multiply created in a particle shower. Furthermore, some antinuclei just passed the TRD volume without any interaction or caused the knock out of low-momentum electrons at different positions along their track.

Additionally, the resulting tracklet structures around the MC tracks of antinuclei were inspected, as they were supposed to form the baseline of a graph construction by serving as nodes of a graph with geometric features based on the tracklet offset and its direction. During this investigation, it was found that a connection of these tracklets resulting from inelastic interaction processes could not be provided by the application of an existing tracking algorithm, since there were typically too few tracklets reconstructed next to an interaction vertex to achieve a high tracking efficiency for the daughter particles and a tracking-based edge construction.

For the further proceeding, it was important to develop a connection between the reconstructed information and the underlying true processes that were provided by the MC information. As the considered antinuclei are typically reconstructed in the TPC due to their high specific energy loss and the fact, that the data only contained antinuclei in form of primary tracks, a matching between TPC and MC tracks of antinuclei was implemented. This matching used a parameter based comparison of the TPC and MC track helices. All the considered tracks were therefore represented using the AliHelix-class of AliROOT [39]. Antinuclei, for which no matching TPC track was found, were not considered in the following.

After these first investigations, the matched TPC tracks of the antinuclei were assigned binary labels based on whether the underlying particles undergo an inelastic interaction process in a defined target volume. This target volume was selected to cover a radial range from 270 cm to 345.5 cm which corresponds to a region in the outer TPC until a radius in layer 3 of the TRD. Referring to this binary labeling, particles that are considered to perform an inelastic interaction process within the target volume will be denoted as signal, whereas the remaining candidates that pass the volume without any interaction are called background particles.

In the following, the criteria for an assignment of antinuclei to the signal and the background class are explained: Every antinucleus of either signal or background class needs to have an MC momentum of $p > 0.3\,\mathrm{GeV}/c$ and its pseudorapidity η needs to fulfill $|\eta| < 0.84$. Particles that had a daughter creation vertex at radii below 270 cm, were assigned neither to the signal nor to the background class. A particle is considered a signal event if there exists a daughter creation vertex in the target volume, where at least two daughter particles with an MC momentum of $p > 0.04\,\mathrm{GeV}/c$ are created. Particles, that did not fulfill these criteria and did not get rejected because of a daughter creation vertex at $r < 270\,\mathrm{cm}$, were assigned to the background class.

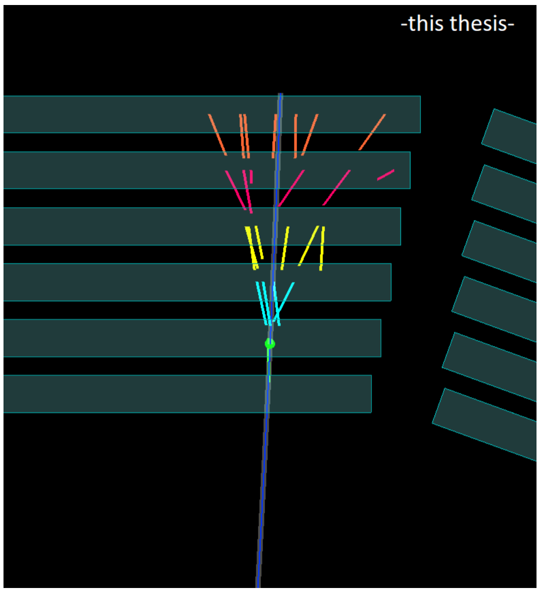
After the assignment of a binary label to the selected TPC tracks, a selection of the TRD tracklets was performed, which were considered to belong to the physical processes associated with a certain TPC track. This selection consisted of two steps: First, the tracklets were prefiltered based on whether they were located in the respective sector or in one of the adjacent sectors, where the propagated TPC track hit the TRD. Tracklets, that were reconstructed in other than these three sectors, were rejected. For the remaining tracklets, their three-dimensional distance of closest approach (DCA) of their offset point to the propagated TPC track was calculated using an iterative algorithm. If a tracklet had a DCA $< 35\,\text{cm}$, it was assigned to the respective TPC track. The so created tracklet structures were visualized in the TRD event display for signal and background particles. Some examples of this visualization process are illustrated in figure 10.

It can be seen that this preselection sometimes contains noise tracklets or even tracklets, that possibly originate from a track of another charged particle, which is located close to the propagated TPC track. However, neural networks are very often able to filter important information from a set of low-level data including a reasonable amount of noise. Consequently, no further selection cuts were applied to the surrounding tracklets, which could have additionally implied a loss of important information.

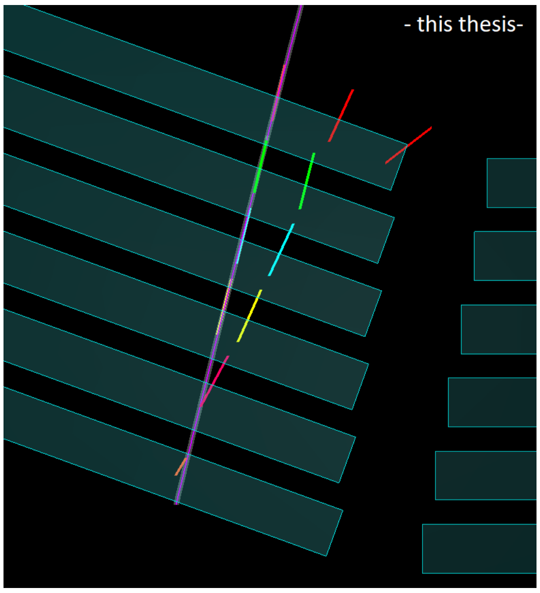
In the end, the final information on the TPC tracks of signal and background particles was stored in form of one *ROOT*-tree each. These trees were nested such that each TPC track contained the assigned TRD tracklets as subobjects. Before the local information of each tracklet was saved to the corresponding tree, the coordinates of its offset and direction vector were transformed to a local coordinate system of the sector, where the respective TPC track hit the detector, via a respective rotation around the z-axis of the experiment (cf. figure 1). This made it possible to reduce the spread of positional features for the different sectors of the TRD in advance and can be seen as an additional step of feature preprocessing.

The stored tracklet information then consisted of the localized offset and direction vector and was further replenished by the three-dimensional distance of closest approach of the tracklet offset to the respective TPC track. The information of each TPC track was made of the reconstructed Lorentz vector, the number of assigned TRD tracklets and a unique identifier of each track, that was built of the event number and the TPC track number within the given files. Furthermore, it contained the true particle identity originating from the MC information, geometric information of the track (e.g. its pseudorapidity η) and the signal tree also contained local information of the respective points, that were considered to be inelastic interaction vertices.

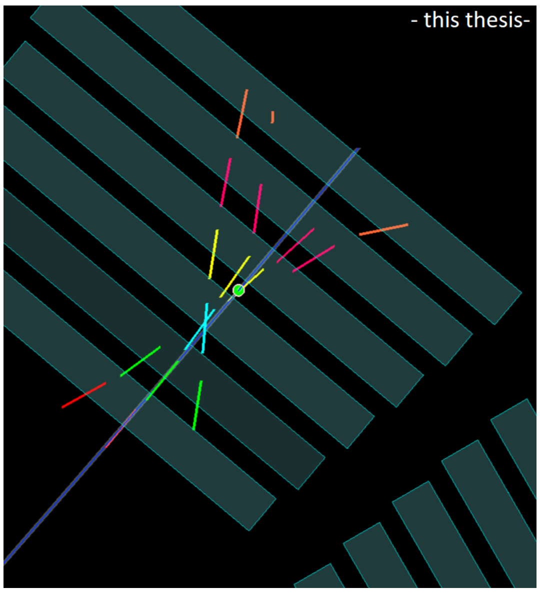
Before the final input for the GNN was created, a further cut on the produced datasets was applied taking into account the number n of assigned tracklets for each TPC track. Throughout the investigations in chapters 5 to 8, only TPC tracks were considered, to which at least seven tracklets were assigned. The purpose of this step was to reject particles, for which no adequate detector signal was present to be considered during classification. The assignment of less than seven tracklets can e.g. happen, when a particle does not pass the active volume of the TRD at all, when certain TRD chambers were not fully operational during the respective runtime or when the vicinity of the TPC track only contains the tracklets of a traversing particle itself. The impact of this preliminary selection process on the amount of statistics for the two classes



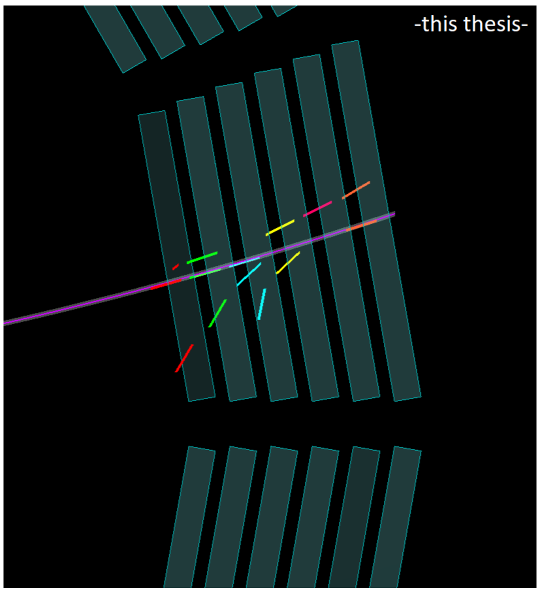
(a) Example of a signal event: An antideuteron nucleus, that serves as a nuclear interaction candidate, produces a shower of outgoing tracklets in the second TRD layer.



(b) Example of a background event: An antitriton nucleus, that passes the TRD with tracklets of an additional charged particle track crossing the propagated TPC track.



(c) Example of a signal event: An antihelium-3 nucleus, that serves as a nuclear interaction candidate. Tracklets of an additional charged particle track overlay tracklets of an in- and outgoing particle shower.



(d) Examle of a background event: An antitriton nucleus that passes the TRD with additional adjacent tracklets that are closely aligned.

Figure 10: Illustration of tracklet assignments to given TPC tracks in the *xy*-plane of the TRD. TPC tracks from the signal class are shown in blue, background tracks are depicted in violet and are propagated to the end of the TRD. The TRD tracklets are colored with respect to the TRD layer where they originate from. Red tracklets belong to layer 0, whereas orange tracklets belong to layer 5. For signal events, the determined interaction vertex is shown as a green dot.

is illustrated in table 1. It can be seen, that this cut significantly reduces the abundance of background particles in the dataset, but only shrinks the signal class to about 86.54% of its original size. Moreover, only about 7.83% of the so excluded instances were signal particles.

	no. of entries	no. of entries with $n > 6$	fraction
background	901 629	371 004	0.4115
signal	335 086	289 998	0.8654

Table 1: Impact of the cut on the number of assigned TRD tracklets on the abundance of the signal and background class within the dataset.

A further illustration of this preselection process is given in figure 11, where the signal and background distributions of the number of assigned tracklets to the TPC tracks are shown. It can be seen that the background distribution is peaked at $n = 6$ assigned tracklets which can be explained by the conjecture that a lot of preselected background events only contain the tracklets due to the traversing particle itself.

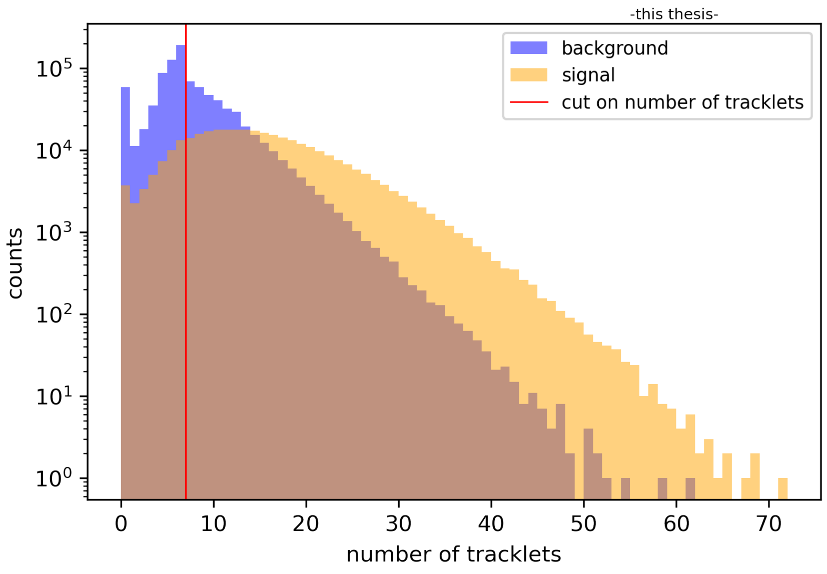


Figure 11: Distribution of the number of assigned tracklets for the TPC tracks of signal and background class. The bin width of the histogram is one. The red line illustrates the cut, that was applied to extract data for further investigations within the following chapters.

Additionally, figure 12 depicts the radial distribution of the points which were considered as inelastic interaction vertices of the signal particles, after the cut on the number of tracklets was applied.

It can be seen that there are peaks in the distribution that can be assigned to the different layers of the TRD. The reason for this is the high interaction probability of antinuclei within the TRD due to its material budget, which was discussed in chapter 2. Additionally, a lot of interactions take place in the outer area of the TPC. The two clear peaks in this radial range might be present because of the high material budget in the outer field cage and the outer

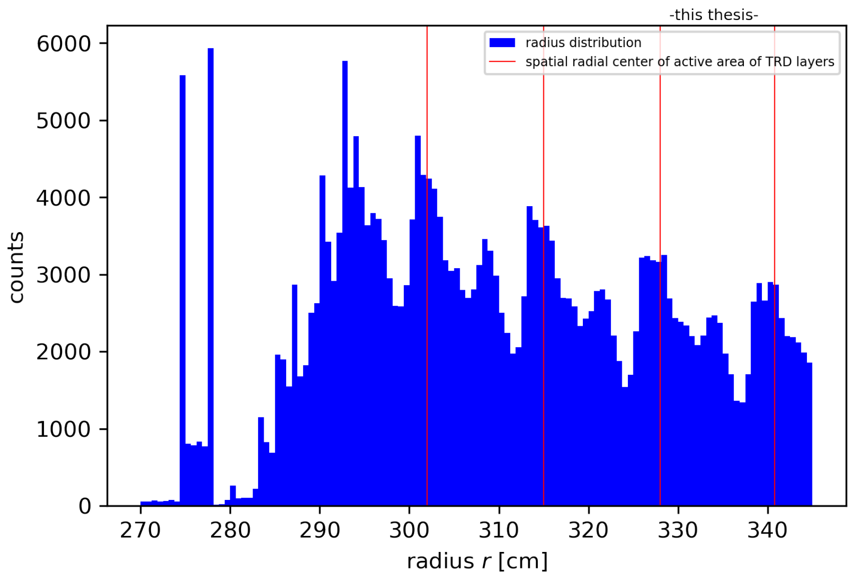


Figure 12: Radial distribution of space points, which were considered to be inelastic interaction vertices. The marked red lines show the spatial radial center of the active area for the respective TRD layers.

containment vessel [40], which limit the CO_2-gap of the TPC, but further studies are required. Moreover, a lot of inelastic interactions take place in front of layer 0 of the TRD.

Further plots, in which the distributions of the respective reconstructed transverse momentum p_T^{TPC} of the extracted signal and background particles are shown with a separation into different antinuclei species, can be found in the appendix in section A.1. These plots also illustrate the signal and background class balance for different species of antinuclei.

5 Pilot experiments on the network architecture

Neural networks have nowadays gained so much in popularity, because there exist powerful libraries which allow for a convenient and fast implementation of deep learning models. In the context of this thesis, the neural network library *PyTorch* [41] and the extension for graph neural networks which is called *PyTorch Geometric* [42] were used. Both libraries allow for training and execution of the respective architectures on graphics cards using the CUDA toolkit [43]. The implementation of the network architectures and their evaluation was performed using the *Python*-programming language. All the subsequent steps were executed on a DELL XPS 15 9500 laptop with an Intel Core i7-10750H and 32 GB DDR4-RAM. Furthermore, an NVIDIA GTX 1650 Ti with 4 GB GDDR5-RAM was used for running the above-mentioned CUDA extensions.

5.1 Graph construction from the preselected data

After the data preselection, that was described in chapter 4, graphs were built, that served as an input for a graph neural network. Therefore, the constructed *ROOT* trees of the signal and background class were embedded into a Python environment using the software package *uproot* [44].

First, graphs were built from the tracklet structures that were assigned to each TPC track. Each tracklet served as the basis for one node of a graph, while holding a seven dimensional feature information. This consists of the three-dimensional localized offset and directional information of the tracklet and its distance of closest approach to the respective TPC track.

For the edge creation, two different strategies were decided to be investigated during first network trainings: The first approach was to construct fully-connected graphs to insert as little prior knowledge on the connectivity in the graph as possible and let the GNN learn the importance of each node during message-passing via an attention network (cf. section 3.5). Therefore, every node is a priori connected to all the other nodes in a graph, which results in a maximally dense graph connectivity.

In the second approach, k-nearest neighbor graphs (k-NN graphs) [45] were constructed using preliminary biases on the positional information of the respective tracklet offsets. Therefore, the pairwise Euclidean distances between the offset points of the underlying tracklets were calculated. Afterwards, a specific node i was connected by an edge to another node j, if the corresponding tracklet offset of node j had a distance to the one of node i, which was under the k smallest ones over the complete graph. Consequently, each node spreads feature information to its positional k nearest neighbors within one step of message-passing and the resulting graph structure always contains self-connections. Like this, a sense of locality on the graphs was introduced, which was related to the tracklet geometry in position space. Furthermore, this results in a graph connectivity that is typically sparser than in fully-connected graphs.

Within the further proceeding, each graph was assigned global attributes such as the transverse momentum p_T^{TPC} of the respective TPC track, the event and track number, which made an identification possible within the framework of the raw data in the *ROOT* trees, and the associated true particle type, that originated from the MC information. This information allowed to conduct additional studies considering physical properties of the underlying particles. Furthermore, each graph received a binary label based on the fact, whether the underlying antinucleus belonged to the signal or the background class.

The representation of the graph data was performed by embedding them into single compressed files using the array representation library *NumPy* [46]. In these files, all the needed information for the graph construction, such as the node features, the adjacency matrix and global graph attributes were included. The advantage of this approach was that it allowed to embed subsets of graphs into *PyTorch Geometric* later on without having to reperform all previous computations and it further offered a rather simple way to deal with the large heterogeneity of the different tracklet structures.

In the next step, a *PyTorch Geometric*-dataset was constructed. This involved the implementation of a *PyTorch Geometric Dataset*-class, which was needed to represent the graphs within the framework correctly. It was decided to build a dataset, that was not embedded into RAM, but could be stored permanently on a hard disk instead. During file creation within a *PyTorch*-format, the previously created compressed *NumPy*-files were used as an input.

Subsequently, the final *PyTorch Geometric*-dataset got processed, during which the respective node features were normalized. Feature normalization or standardization is an often performed procedure of preprocessing during gradient-based training of neural networks. For the subsequent training and testing steps, an appropriate subset of the dataset was passed to a dataloader of *PyTorch Geometric* in order to create graph batches and automatically perform batch-shuffling during the training process.

5.2 First training steps

One challenge, that is omnipresent during the optimization of neural networks, is that their performance critically relies on the used architecture and the choice of good network hyperparameters. Furthermore, the training strategy itself, e.g. the choice of the optimizer and its configuration (e.g. learning rate), can have a huge impact on the quality of the network output after training. Since network training can become very expensive regarding computational resources and also in terms of time, some restrictions need to be applied on the tests beforehands. This also includes the fact that the first trainings are normally not conducted on the complete amount of available data and that the network training is typically not performed until final convergence. The network structure, that was investigated during these first considerations ("pilot experiments"), is illustrated in figure 13 and builds the framework for additional optimization steps that are explained in detail in chapter 6.

In this network, the graph input gets passed through a predefined number of message-passing layers with subsequent ReLU activation functions. After the message-passing steps, a global average pooling is applied for every graph. This means that a graph representation

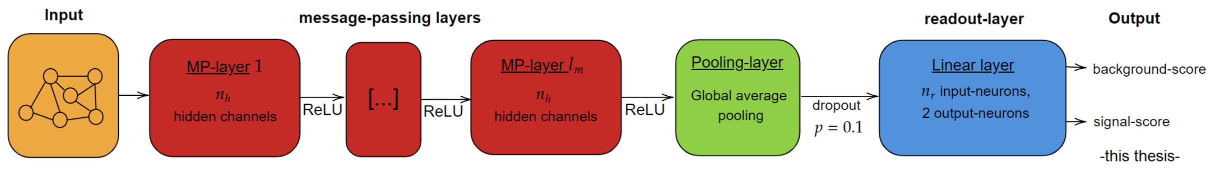


Figure 13: Starting point for a GNN architecture during the pilot experiments. The boxes in different colors stand for different parts of the network that largely belong together. The red boxes represent message-passing layers, whereas the green box stands for a pooling layer that computes a graph representation from the average of node representations within each graph. A final linear layer, which is displayed in blue, produces output scores for each graph referring to the defined signal and background class.

is obtained from an elementwise average over the feature representations of all nodes in the graph. Afterwards, a dropout is applied to the output of the pooling layer, which is a common technique in training neural networks to provide a regularization effect and therefore prevent overfitting. This is done by randomly setting outputs of a given layer to zero during training with a given dropout probability p. Afterwards, a single linear layer computes signal and background scores for each graph from which a class prediction can be derived by applying the decision rule that is shown in equation (3).

The main focus during this investigation was to find out how the application of different numbers and types of message-passing layers have an influence on the classification results. Therefore, four different layer types were selected for first experiments: Two layers (*GCNConv*, *GraphConv*) implement a graph convolution. The *GCNConv*-layer performs convolution in exactly the same way that was introduced in section 3.4. The *GraphConv*-layer slightly differs from this implementation as no neighborhood normalization as in *GCNConv* is performed and that different learnable weight matrices are applied to the original and neighboring nodes in a graph. The other two layers (*GATConv*, *GATv2Conv*) allow for building a graph attention network (cf. section 3.5) and differ in the way the attention coefficients are calculated and incorporated in the respective node updates. Further details about this can be found in [34].

All these layers are already implemented in *PyTorch Geometric* and were used with different numbers n_h of so-called hidden channels, which refers to the size of the feature representation that each node is assigned between multiple message-passing layers. The number of message-passing layers was varied between two and four and for the number of hidden channels, the values 64, 128 and 256 were tried out. Since the global average pooling preserves the dimension of the computed feature vectors, as the average is performed over the different nodes in a graph, the number of input neurons n_r of the linear layer was equal to the number of hidden channels n_h.

The training was performed using datasets of fully-connected graphs and k-NN graphs. Early tests within the above shown architecture were conducted with different numbers of nearest neighbors. A number of $k = 5$ nearest neighbors showed the most promising results, such that this value was used for systematic studies in the following. Furthermore, a rapid performance decrease was observed, when lower values for k were used. This can be reasoned by the fact that the resulting graphs contained increasingly more isolated nodes, which has a considerable impact on the process of message-passing.

As processing the complete datasets with all instances during these preliminary tests would have been infeasible in points of time consumption, three subsets of 20.000 graphs were sampled from all instances and used during the first trainings. Each dataset of 20.000 graphs was divided into 14.000 training graphs, 3.000 validation graphs and 3.000 test graphs. The training graphs were grouped into batches of size $N = 200$. The training was performed using the cross-entropy as a loss function and the ADAM-optimizer with a constant learning rate of $\tau = 0.005$ for 100 epochs. The model states after all epochs were logged by storing the respective model parameters. Furthermore, the model accuracies on the training and validation set were calculated during each epoch to monitor the training progress and further judge, how well the model generalizes to previously unseen data. Additionally, this strategy was applied to identify signs of overfitting, which would imply an increasing training accuracy, while the validation accuracy would simultaneously decrease over multiple epochs. However, no clear signs of overfitting were encountered throughout all pilot experiments, which was concluded from plots of the training and validation accuracies of the considered models over the training epochs.

After 100 epochs, the best model during training time was selected based jointly on the documented training and validation accuracies. This final model was then evaluated on the test set and the respective area under curve (AUC) was calculated. The AUC denotes the area under the so-called Receiver Operating Characteristic (ROC), for which the signal efficiency ε_s (also known as true positive rate) of the model is evaluated as a function of its background efficiency ε_b (also known as false positive rate). It can be seen as a measure that summarizes the prediction power of the model independent of the specific classification threshold for the final class assignment.

This procedure of training and testing was repeated for the three sampled subsets of the dataset for all mentioned network types, numbers of message-passing layers and hidden channels, respectively. The corresponding results were afterwards averaged for a certain network-/dataset configuration and the standard deviation of the obtained three values were calculated. The results of the investigation during the use of three message-passing layers of the four respective types with different numbers of hidden channels, while considering 5-NN graphs, can exemplarily be seen in tables 2 and 3. In this setting, the best overall model performances during the chosen optimization procedure were encountered, which was concluded based on the mean accuracies calculated on the respective test sets and the associated mean of the AUCs.

The full results for other numbers of message-passing layers and especially for the dataset with fully-connected graphs are shown in the appendix in section A.2. As it turned out, the performance on the dataset with fully-connected graphs was in almost all configurations behind the results, that were achieved during the consideration of NN graphs and larger standard deviations considering the final test accuracies and AUCs were encountered. This was the case for the convolutional and the attention-based methods, even though the con-

	GCNConv			GraphConv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.778 ± 0.003	0.774 ± 0.003	0.775 ± 0.002	0.852 ± 0.001	0.853 ± 0.001	0.852 ± 0.001
validation accuracy	0.773 ± 0.002	0.766 ± 0.002	0.765 ± 0.003	0.848 ± 0.001	0.848 ± 0.002	0.847 ± 0.002
test accuracy	0.768 ± 0.003	0.764 ± 0.003	0.767 ± 0.003	0.850 ± 0.002	0.851 ± 0.001	0.849 ± 0.002
AUC	0.856 ± 0.002	0.852 ± 0.002	0.854 ± 0.002	0.914 ± 0.001	0.915 ± 0.001	0.913 ± 0.001

Table 2: Results of pilot experiments with 5-NN graphs for three message-passing layers using two graph convolutional methods.

	GATConv			GATV2Conv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.851 ± 0.001	0.852 ± 0.001	0.850 ± 0.001	0.851 ± 0.001	0.850 ± 0.001	0.850 ± 0.001
validation accuracy	0.846 ± 0.002	0.847 ± 0.002	0.848 ± 0.002	0.846 ± 0.001	0.844 ± 0.001	0.844 ± 0.001
test accuracy	0.845 ± 0.002	0.851 ± 0.002	0.849 ± 0.002	0.844 ± 0.002	0.843 ± 0.001	0.845 ± 0.002
AUC	0.909 ± 0.002	0.914 ± 0.001	0.912 ± 0.001	0.908 ± 0.002	0.907 ± 0.001	0.910 ± 0.002

Table 3: Results of pilot experiments with 5-NN graphs for three message-passing layers using two different methods implementing a graph attention network.

volutional methods were performing significantly worse than the attention-based ones. This shows that a preliminary bias on the graph connectivity, which is further motivated by the underlying tracklet geometry, can help to improve the classification performance of the networks compared to a relatively unbiased proceeding.

Additionally, the naive approach of just passing the node features on graphs with full connectivity to a neural network and let an attention mechanism extract relevant information, did not work satisfactorily compared to the approach with NN graphs, when taking the above network structures and optimization steps as a baseline. Consequently, the dataset of fully-connected graphs was not used anymore while taking the above network architecture through further optimization steps.

When comparing the graph convolutional methods using NN graphs, it can be seen that the use of the *GraphConv*-layer within the network architecture delivers much better results than the *GCNConv*-layer. This might be due to the fact that the *GCNConv*-implementation loses important graph-structural information while performing neighborhood normalization, which has already been pointed out in recent publications, e.g. in [32]. The best overall performance is seen with the *GraphConv*-network with 128 hidden channels, even though the difference compared to 64 and 256 hidden channels is not assessed significant considering the standard deviations of the results.

For the attention-based solutions, both methods essentially achieve similar results during training, that are on average better than a normal graph convolution with *GCNConv* and only slightly worse than using *GraphConv*-layers. However, the *GATConv*-architecture shows a little better generalization to previously unseen data than the model using *GATV2Conv*-layers and the best results are obtained with 128 hidden channels as well. The performance of the models of identical type (same type and number of message-passing layers) when using different numbers of hidden channels, are not too dissimilar, but expose a little larger differences compared to the tests with *GraphConv*-layers.

6 Training of the final models

Due to the performance tests in the first training steps within chapter 5, two models were chosen to be carried through further optimization steps, before applying them to the complete dataset of 5-NN graphs. A detailed documentation of the respective results for the following optimization steps, like it was given in chapter 5, is not presented here. The two models, that were investigated further, were the chosen *GraphConv*- and the *GATConv*-model with three message-passing layers and 128 hidden channels each.

6.1 Additional optimization of the final models

In the first step of further optimization, the single linear layer, which was put at the end of both models, got replaced by a three layer fully-connected network with ReLU activations. This step was taken to enable the model to learn a more complex readout function, that computes the final signal and background scores. The details on the respective configuration of the number of input and output neurons are illustrated in figure 14.

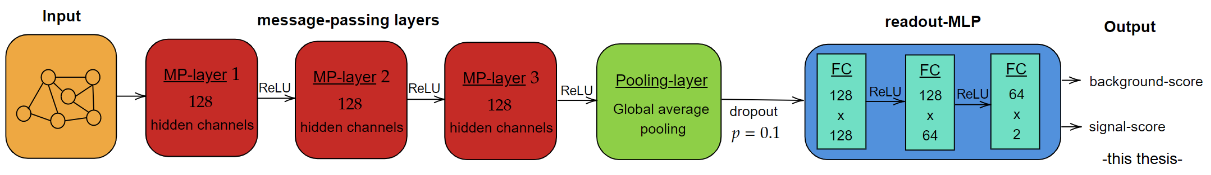


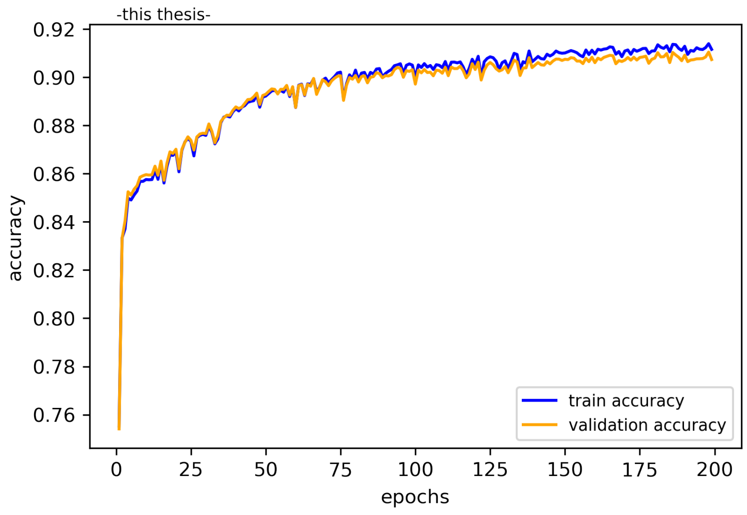
Figure 14: Final network configuration after additional optimization steps. It consists of three message-passing layers with 128 hidden channels each and a complete readout-MLP, that is built of three fully-connected layers using ReLU activations.

Additionally, the message-passing layers in the *GATConv*-model were enhanced to perform K-multihead attention with $K = 3$ heads, during which the feature dimension of $n_h = 128$ was preserved by applying the final layer activation after averaging the aggregated feature output. This step was done to stabilize the process of learning attention coefficients during message-passing. It was additionally observed to deliver visual improvements during training, as huge fluctuations within the registered train and validation accuracies became less abundant. Both modifications of the *GATConv*-model and the extension of the readout network for the *GraphConv*-model were identified to result in an additional performance improvement during similar testing strategies, as they were performed in chapter 5.

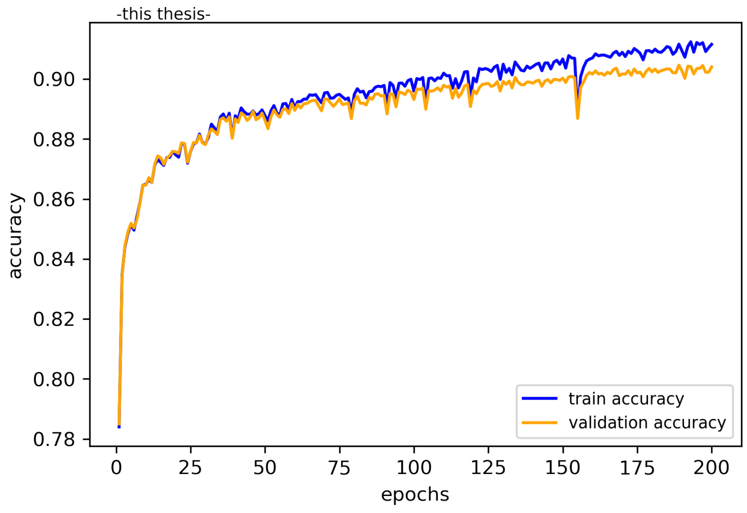
6.2 Training procedure and training results

After the final decision on the network structure, both models were trained using the complete dataset of 661.002 graph instances. During this training, the graphs were not separated due to their particle species and no separation of the data due to the transverse momentum p_T^{TPC} of the respective TPC tracks was performed either. Additional investigations on the effects of further data separation and the reasoning of the used final training procedure are presented at the end of this section.

The training was performed for 200 epochs, which took about 54.83 hours for the *GraphConv*-model and about 46.45 hours for the *GATConv*-model. For the optimization process of each model, a schedule on the learning rate of the ADAM-optimizer was applied to improve its convergence. It was decided to use a learning rate reduction every 30 epochs to 75% of the previous one starting in epoch 30. The initial learning rate τ was chosen to be $\tau = 0.005$. The complete dataset was split such that 70% of all instances were used during training, 15% for validation and 15% for testing. The training instances were presented in batches of size $N = 1000$ and the model accuracies on the training and validation set were calculated in every epoch to monitor the training progress. A plot of the resulting training and validation accuracies for both GNN models during the final trainings can be seen in figure 15.



(a) Results for the *GraphConv*-model.



(b) Results for the *GATConv*-model.

Figure 15: Trends of the training and validation accuracies during the final training process of both GNN models. Figure (a) shows the accuracy trends for the *GraphConv*-model and figure (b) for the *GATConv*-model.

After every epoch, the model states were saved by storing the respective model parameters. This allowed for selecting a model after training, which has shown very good performance in terms of training and validation accuracies and additionally prevents from choosing a potentially bad model due to noise in the resulting accuracies at the end of the respective training period.

The decision to terminate training of both models after 200 epochs was taken based on the convergence behavior of both models. It can be seen in figure 15(a), that the training and validation accuracy of the *GraphConv*-model did not change significantly any more during the last 30 epochs of training. Thus, the occurrence of additional improvements that considerably influence the model quality was considered rather unlikely.

For the *GATConv*-model, it can be seen in figure 15(b) that the training accuracy of the model was still increasing a little on average, but this was not significantly the case for validation accuracy for about 30 epochs. It was therefore concluded that substantial improvements on previously unseen data are not very likely, even when the training accuracy continues to rise further. However, the difference between the model performance on the training and the validation set at the end of training was not interpreted as a sign of overfitting, since the further increase in training accuracy did not lead to a decreasing validation accuracy.

After the termination of training, both models were evaluated using the graph instances in the beforehand separated test set. Additionally, the AUC and the so-called average precision score of the two different models were calculated. While the AUC considers the signal efficiency ε_s of the model as a function of its background efficiency ε_b in the ROC curve, the average precision score is based on the evaluation of the model purity ε_p as a function of its signal efficiency ε_s in the so-called precision-recall curve[3]. Like this, it can be visualized, how well the model can separate signal and background due to classifying a particle as signal, while taking into account that only a certain fraction ε_s of signal particles is classified correctly. In ideal cases, a classifier returns a very pure sample at a high signal efficiency, but in practice, a suitable trade-off between these two quantities must be applied. The average precision score can be seen as a single quantitative measure, how well a model can correctly identify all signal instances without classifying too many background particles wrongly.

An overview of the respective model accuracies on the training, validation and test set is given in table 4, where also the AUCs and the average precisions of both classifiers are included. Plots of the respective ROC and precision-recall curves are shown in chapter 7, when the performance of the GNNs is compared against additional simple classification approaches.

As it can be seen, both GNNs achieve satisfactory results. They provide high accuracies during training and are apparently both able to generalize to previously unseen data considering their performances on the test set. However, the *GraphConv*-GNN delivers a slightly better overall model performance and shows superior results on the test set compared to the *GATConv*-model. A detailed analysis of the model performances while looking at the different types of antinuclei and investigations of the model behavior in different ranges of transverse

[3]In machine learning jargon, the terms precision for the purity and recall for the signal efficiency are commonly used.

Model	***GraphConv*-GNN**	***GATConv*-GNN**
training accuracy	0.9139	0.9121
validation accuracy	0.9104	0.9046
test accuracy	0.9120	0.9037
AUC	0.970	0.964
average precision	0.964	0.957

Table 4: Overview of the model performance of both GNNs after final training.

momentum is provided in chapter 7.

During additional tests, the creation of particle-specific models was considered. Therefore, the used dataset was separated for the different species of antinuclei and similar trainings as before have been carried out. As it turned out, training these individual models from scratch did not result in an improvement of the model accuracies compared to deploying the general models on particle-specific data.

Moreover, it was tried to use pretrained models that were obtained with the unseparated dataset within different epochs and continue training under use of parts of the respective training/validation set, where the instances were separated according to their particle species. These investigations have also never led to reaching the individual performance of the final overall models on particle-separated data.

One reason on the side of physics for this could be that the topology of inelastic interaction processes of different antinuclei is not too dissimilar and that the individual neural network performance benefits from the respective samples across different antinuclei species. However, these differences can also result from the fact that further adjustments of the models for different particle species are needed, as the amount of available statistics gets reduced to about one third of the original size. In any case, approaches like this require further studies in the future, e.g. with more statistics from additional simulations.

6.3 Systematics of wrongly classified instances

To make further trainings in additional use cases of the GNNs more accurate, especially when using the models for physics analyses, it is of interest to identify problems in the classification process by a systematic study of misclassified instances. It is therefore interesting to search for similarities in the respective instances, which provides the chance of future improvements, e.g. in the data preselection.

To perform these investigations, the individual graphs, that were classified wrongly by the final models, were written out and matched with the underlying information that was derived from the raw data files.

An important point was that the radial distribution of the inelastic interaction vertices of the false negatives using both final GNNs is peaked at higher radii and was especially not similar to the original radial distribution of interaction points, that is shown in figure 12. This can be seen in figure 16.

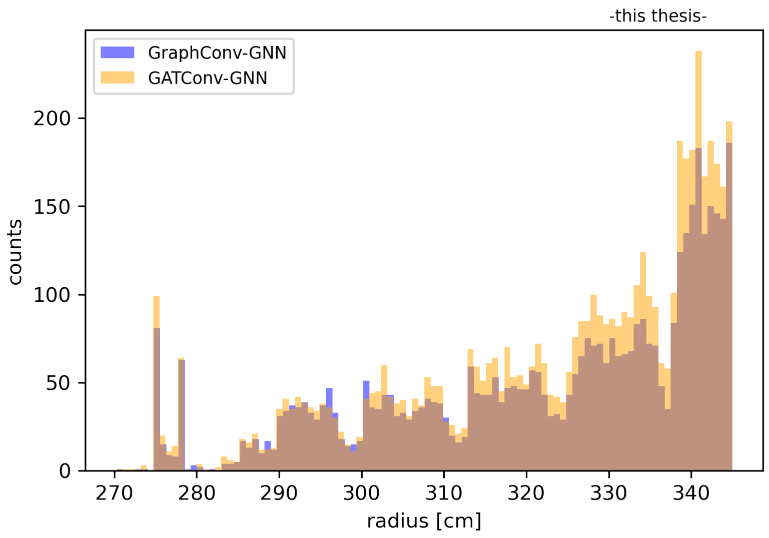


Figure 16: Radial distribution of interaction vertices of signal particles that are classified as background. The plot includes the results for both final GNNs.

One reason for this dissimilarity could be the fact that interacting particles in layers 4 and 5 of the TRD are also assigned to the background class as a result of the definition of the target volume. Consequently, the topology of nuclear interactions in the backstage area of the target volume might be confused with background events during the classification procedure. This could be further supported by the fact that inelastic interactions at higher radii are in general less abundant in the dataset (cf. figure 12) and are therefore seen less during training. However, interpretations like these must be treated with caution, as the exact decision process of the GNN is not transparent for the user. Consequently, additional investigations of these effects require further empirical studies, for example with further data samples or with a customized data preselection that uses a definition of a reduced target volume.

7 Comparison of graph neural networks against other classification approaches

The previous chapters have shown that it is indeed possible to apply GNNs to the given classification problem and achieve reasonable results. However, it is of interest, how GNNs behave in comparison to other simple classification approaches and find out, if they deliver a similar or even better performance. Within this chapter, two different approaches are presented: The first one relies on classical cut-based analysis on handcrafted features, as it is widely used in HEP. The second one is based on a random forest classifier that is built using the features of the cut-based models and additional positional features of the underlying tracklets. Both approaches were used with data, for which no preliminary separation of different particles respectively p_T-bins was performed during model creation, since this was also not done during the training of the neural networks.

7.1 Cut-based models

7.1.1 Creation of cut-based models

One problem that occurs when trying to apply a cut-based analysis is that the given setting makes it difficult to come up with handcrafted features such that they can be assigned a comprehensive physical meaning together with the applied cuts. This is the case since the positional information of the tracklets around a TPC track is rather abstract and it is therefore difficult to develop an intuition, which strategies might deliver good results.

In this thesis, two approaches to the cut-based model creation were investigated: The first one was based on applying a cut on the number of assigned tracklets n for each TPC track. As it can be seen in figure 11, the different distributions of the number of tracklets for the signal and the background class are expected to provide some separational power. This can also be assigned a physical meaning since antinuclei that perform inelastic interactions, typically produce multiple outgoing charged particles which are expected to leave signs of presence in the TRD in the form of additional reconstructed tracklets.

The cut search was performed while trying to obtain a maximum accuracy of the resulting model. This approach was selected because this was also the one chosen during neural network training, where the accuracies of the respective models were monitored to judge the performance during the model creation. However, further use of these models in the scope of a physics analysis might require to change the cut selection, such that e.g. statistical errors of the inferred physical quantities get minimized (cf. chapter 8).

In this model, a particle was identified as signal, if the number of tracklets n exceeded a certain threshold number n_{cut}. As it turned out, a cut on the number of tracklets with $n_{\text{cut}} = 13$ performed best in the given setting.

In a second approach, the mean distance $\overline{d}$ of all assigned tracklets from a given TPC track was considered. This was motivated by the fact that $\overline{d}$ is expected to be higher if a particle creates certain daughter particles that typically propagate away from the original TPC track.

The distribution of the respective mean distances for signal and background particles is illustrated in figure 17. Just as done before, an optimal cut $\overline{d}_{\text{cut}}$ was searched that provided a maximal model accuracy when applying the decision rule that every particle with $\overline{d} > \overline{d}_{\text{cut}}$ is classified as signal. The search was performed on a grid of spacing 0.1 cm from mean distances of 0 cm to 30 cm. The optimal cut value d_{cut} was found to be $d_{\text{cut}} = 15.6\,\text{cm}$, which is indicated by a red line in figure 17.

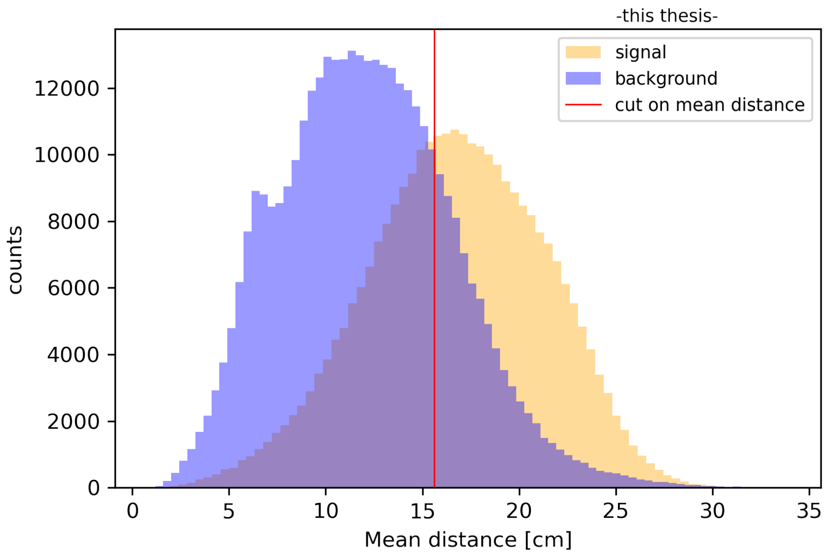


Figure 17: Distribution of the mean distance of all assigned tracklets from the TPC track for signal and background particles. The red line illustrates the cut that was found to produce the highest accuracy in the respective cut-based model.

The accuracies of the two individual cut-based models are shown in table 5. Additionally, the resulting efficiency and purity of each model is included.

Model	**cut on number of tracklets n**	**cut on mean tracklet distances $\overline{d}$**
accuracy	0.7113	0.7027
efficiency	0.592	0.600
purity	0.703	0.684

Table 5: Overview of the general performance of the two created cut-based models operating on the number of assigned tracklets and the mean distance of all assigned tracklets of the given TPC track.

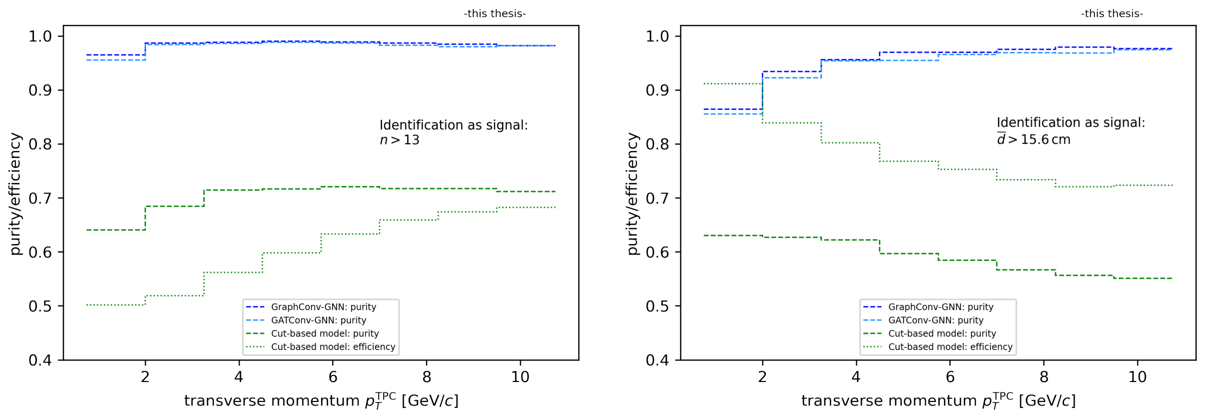
It can already be seen that the performance is not really satisfactory, as especially the efficiency of both models is rather low and the model accuracies do not exceed values of roughly 71%. One reason for this is possibly the occurrence of noise tracklets and the fact that the tracklet preselection, as it is performed here, is not suitable for the application of these simple cut-based models. The overall performance might improve with a different approach for the tracklet preselection or if the complete dataset was divided into subsets with additional constraints (e.g. with respect to the transverse momentum of the particle) and cut search was performed individually on the different subsets.

During further investigations, it was tried to combine both above used features and derive joint cuts to further improve the results. As it turned out, both features could not be combined meaningfully into a joined model. This is because the found optimal cuts by using the approach of obtaining a maximum accuracy delivered the same cut value on the number of surrounding tracklets as above and implied to overthrow the dependence on the mean distance $\overline{d}$.

Of course, there might be more expressive features, which could result in a better performance in the framework of cut-based analyses, as they could provide more separational power. However, it might then be problematic to interpret the respective features and assign them a reasonable physical meaning together with the derived cuts.

7.1.2 Comparison of cut-based models with the GNNs

For an additional comparison between the cut-based models and the GNN approaches, the resulting efficiencies and purities of both cut-based models were calculated by applying them separately to particles with different reconstructed transverse momentum p_T^{TPC}. Therefore, a division into eight p_T^{TPC}-bins of the same width between 0.75 GeV/c and 10.75 GeV/c was used. In the next step, the efficiencies of both GNNs were fixed to the efficiency of each cut-based model within the specific p_T^{TPC}-bin and the corresponding purities of the GNNs were calculated. The results are visualized in figure 18 together with the respective purities and efficiencies of the cut-based models.



(a) (b)

Figure 18: Comparison of the GNN purities to the purity of both cut-based models in eight different p_T^{TPC}-bins at equal efficiency. The efficiencies, for which the purities of the different models are shown, are depicted by a green dotted line and correspond to these of each cut-based model within the specific p_T^{TPC}-bin. Figure (a) shows the results for the cut-based model using the number n of surrounding tracklets and figure (b) for applying a cut on the mean tracklet distances $\overline{d}$.

Further plots, which were derived analogously with additional particle separation and the subsequent evaluation of the model purities in different p_T^{TPC}-bins at equal efficiency levels, can be found in the appendix in section A.3.

As depicted in figure 18, the GNNs provide purities that are a lot higher compared to the respective purities of the cut-based models and that are also nearly constant at the considered efficiency levels. Some exceptions thereof can be seen at lower p_T within the cut-based model on the mean tracklet distance $\overline{d}$. This is due to the fact that the corresponding efficiencies It should be noted that the purity of the cut-based model using the number of assigned tracklets does not change very much for different p_T with the exception that the purity becomes a little lower within the two bins of $p_T^{\mathrm{TPC}} < 3.25\,\mathrm{GeV}/c$. During the use of the cut-based model on the mean distance, a decreasing purity is observed as well as a monotonically decreasing efficiency over subsequent momentum bins for increasing p_T^{TPC} with a small exception in the bin with the highest p_T^{TPC}-values. One reason for this can be the different kinematic conditions during the interactions at different p_T^{TPC}, as the resulting tracks of daughter particles may have smaller curvature at higher p_T^{TPC} and therefore on average a smaller distance from the original TPC track.

Based on these results and especially under consideration of the respective model performances, when searching for cuts showing a good performance over the whole dataset, the approach of deploying cut-based models for this classification problem was not further extended. In summary, it was shown that the constructed GNNs clearly outperform these simple cut-based models.

7.2 Random forest

7.2.1 Description of feature selection and training

A further approach to benchmark the GNN was to use multivariate analysis methods that operate on additional features compared to the cut-based models. In this thesis, it was decided to use a random forest classifier, like it is implemented in the machine learning library *scikit learn* [47]. Decision trees are an important method in classical machine learning and are also widely used in HEP [12]. This is mainly because they are quite simple and fast to train within many machine learning libraries and their working principle is more intuitive compared to deep learning methods. They especially become powerful when multiple trees are used in an ensemble of predictors, such that weaknesses of individual classifiers can be compensated. Further enhancements of ensemble methods like boosting techniques, which are for example further explained in [12], were not considered within this thesis.

During the training of a random forest, an ensemble of decision trees is successively built, until the desired number of estimators is reached. For building individual trees, subsets of the complete dataset are considered and these instances are assigned leaves in the tree. The exact tree-structure is then derived from the given training set, as a split of a node in the tree is performed after determining an optimal cut according to a predefined split criterion [25].

Splits are typically executed, until a certain tree only has pure leaves (only instances of one class remain in the leave) or another termination criterion is fulfilled. In both cases, a leaf in the tree is created containing the remaining instances. When the forest is used for prediction, an instance is assigned to a leaf in the tree using the derived cuts during training. Afterwards, a probability estimate for the test instance belonging to a certain class is calculated by taking the number of certain class members in the assigned leaf and dividing them by the total number of instances in the leaf. The class prediction of a tree for a test instance is then typically derived by taking the class of the majority of training instances in the leaf to which the test instance was assigned. Finally, a forest prediction can be derived by a majority vote of all trees in the ensemble [25].

In this comparison, a simple random forest was trained that operated on nine different features: The distances of the seven nearest tracklet offsets to the TPC track, the number of assigned tracklets and the mean distance of all tracklet offsets from the respective TPC track. Like this, the underlying geometry of the tracklet configurations could be represented by a feature vector in a simple way. In the future, further studies should be done, whether different feature constructions could be applied to improve the following results and therefore represent the underlying tracklet configurations better within the setting of tabular data.

Before the performance of a trained random forest was compared to the those of the two different GNNs, a hyperparameter optimization was performed using a grid search approach. This means that all possible combinations of hyperparameters that can be constructed from initial options for each hyperparameter are tested. Within this thesis, only the number of estimators and the maximum tree depth were optimized by means of 5-fold cross validation on the complete dataset [25]. The maximum tree depth is part of a termination criterion for tree building and determines the maximum number of performed splits, before a final leaf is created. The result of the hyperparameter search is illustrated in table 6, where all tested hyperparameters are listed together with the ones, that resulted in the best mean test accuracy calculated from each of the five different splits during cross validation.

hyperparameter	**tested parameters**	**optimal hyperparameter**
number of estimators	50, 100, 200, 300	200
maximum tree depth	2, 3, 5, 8, 10, 12, 15, 18, 20, 25	15

Table 6: Results of the hyperparameter search for the random forest, that was compared against the two different GNNs.

After the hyperparameter search, the random forest was trained on the complete dataset of 661.002 instances. This was done by using 70% of the total amount of data for training and 30% for testing.

During training, a feature importance computation was performed in order to find out, which of the used features were the most relevant during the model construction. This was done by using a method that relies on feature permutation. Therefore, the features are randomly shuffled individually among the instances, such that the reference between a certain instance and its underlying true feature is destroyed. Depending on the fact, how important the feature for the resulting model is, this will cause the resulting accuracy on the model with permuted features to drop. By performing this procedure repeatedly, a mean accuracy decrease can be obtained due to randomly shuffling individual features, which can be seen as a measure of the importance of a certain feature in the given model. The corresponding results for the random forest are shown in figure 19.

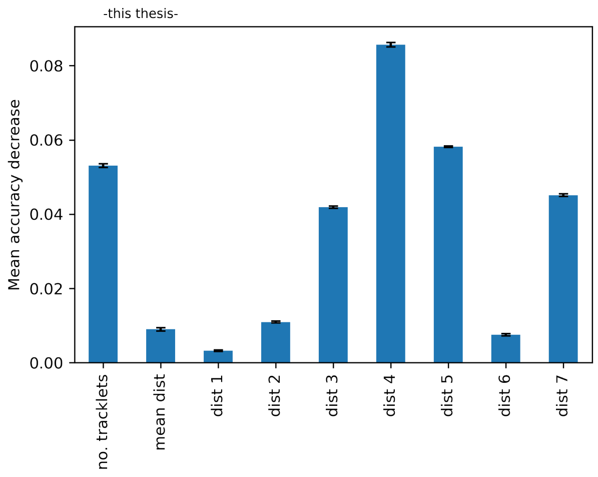


Figure 19: Mean accuracy decreases in the random forest model due to the individual permutation of features among the instances. The features that were used here and which are abbreviated on the horizontal axis are explained in the main text. The black dots on top of the bars show the standard deviation of the results, as the shuffling of each feature was performed ten times.

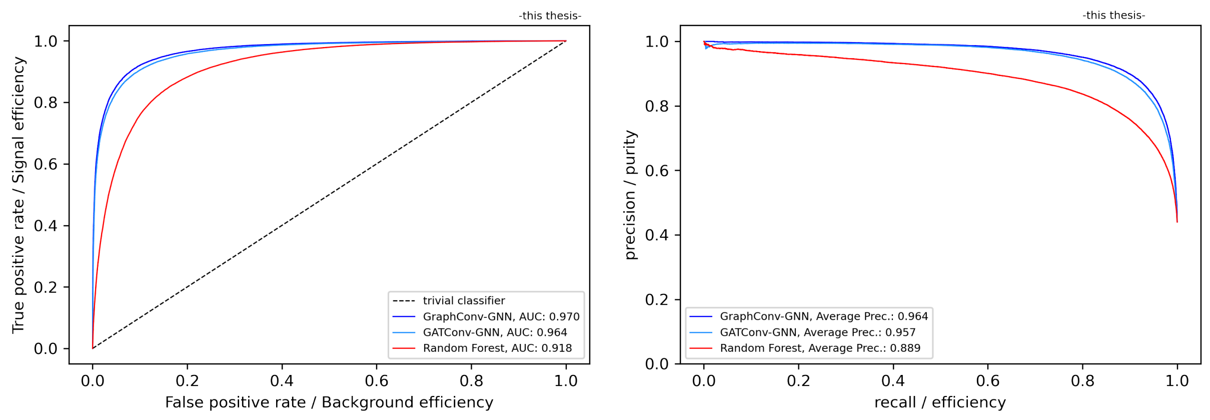
As it can be seen, the performance of the random forest mainly relies on the distances of tracklets, which are not in the immediate vicinity of a given TPC track. Additionally, the number of preselected tracklets is also a rather important feature. However, this importance ranking is not necessarily related to the individual separational power of each feature, which can be demonstrated by the example of the mean tracklet distance and the number of assigned tracklets (cf. section 7.1).

The training of the 200 estimators took about 4 minutes and 19 seconds, which illustrates the previously mentioned superiority of the random forest over the GNNs in points of the need for computation resources and training time. However, it does not reach the overall performance of the GNN in points of the resulting accuracies, which can be seen when comparing the tables 4 and 7.

performance measure	**value**
training accuracy	0.8885
test accuracy	0.8445
AUC	0.918
average precision	0.889

Table 7: Overview of the random forest performance after training.

The differences in performance can additionally be visualized by comparing the ROC and precision-recall curves of the random forest with the ones of both GNNs. This is illustrated in figure 20.



(a) ROC curves. (b) Precision-recall curves.

Figure 20: Comparison of the ROC and precision-recall curves of both GNNs with the ones of the trained random forest. The black dashed line in figure (a), which is titled with "trivial classifier" corresponds to a classification, that would assign the labels for all given instances randomly with equal probability.

It is visible, that the ROC curve of both GNNs is clearly above the one of the random forest and that the AUC of the GNNs is also noticeably higher. When looking at the precision-recall curves of the different classifiers, it can be seen that the ones of the GNNs are almost everywhere above the one of the random forest, which is also supported by the significantly higher average precision value. Furthermore, it is noticeable that the precision-recall curves of the GNNs remain much more constant for a wide range of efficiencies and that the GNNs provide a much higher purity at high efficiencies of $\varepsilon_s > 0.8$. The latter implies that the neural networks can significantly better provide a high classification purity, even when the efficiency of the classifier is chosen to be rather large.

7.2.2 Overall performance comparison for different antinuclei and transverse momenta

As it was explained before, the GNNs were trained on the complete dataset of 5-NN graphs without separation due to different kinds of antinuclei or the reconstructed transverse momentum. As a consequence, no comparable subdivisions of the dataset were performed during the training of the random forest either. Nevertheless, the individual performance of the classifiers on instances with different physical properties is an important point for a future physics analysis. This section therefore presents a performance comparison between the GNNs and the random forest based on a consideration of individual ROC curves. They were calculated during the application of the trained classifiers to respective parts of their test sets and the partitioning was performed according to the antinuclei species and the reconstructed transverse momentum of the given instances. To summarize the results, the AUCs of these individual

ROC curves are provided in the form of tables here, whereas plots of the ROC curves are provided in the appendix in section A.4. The appendix also contains plots of ROC curves, which were created for an application of the classifiers to particle-individual subsets with additional separation in p_T^{TPC}-bins. Within this subsection, a comparison has been performed using a separation of either the particle species (cf. table 8) or the reconstructed transverse momentum p_T^{TPC} (cf. table 9). The latter was performed by choosing a division of all instances by eight p_T^{TPC}-bins of equal width between 0.75 GeV/c and 10.75 GeV/c.

	$^3\overline{\mathrm{He}}$	$\overline{\mathrm{d}}$	$\overline{\mathrm{t}}$
GraphConv-GNN	0.971	0.965	0.971
GATConv-GNN	0.965	0.962	0.966
Random Forest	0.909	0.921	0.921

Table 8: AUCs of different classifiers during application on parts of the test set separated for different antinuclei.

	$0.75 - 2.0$ GeV/c	$2.0 - 3.25$ GeV/c	$3.25 - 4.5$ GeV/c	$4.5 - 5.75$ GeV/c
GraphConv-GNN	0.946	0.966	0.970	0.973
GATConv-GNN	0.940	0.960	0.967	0.967
Random Forest	0.851	0.907	0.925	0.926

	$5.75 - 7.0$ GeV/c	$7.0 - 8.25$ GeV/c	$8.25 - 9.5$ GeV/c	$9.5 - 10.75$ GeV/c
GraphConv-GNN	0.973	0.973	0.974	0.972
GATConv-GNN	0.969	0.969	0.970	0.964
Random Forest	0.927	0.930	0.930	0.936

Table 9: AUCs of different classifiers during application on parts of the test set separated for the reconstructed transverse momentum p_T^{TPC}.

When looking at the results for different antinuclei, it can be seen that both GNN models always provide significantly better AUCs than the random forest. The AUC for both GNNs considering $\overline{\mathrm{d}}$ is observed to be a little lower than for $\overline{\mathrm{t}}$ and $^3\overline{\mathrm{He}}$, but these differences are not assessed as significant. The random forest shows a little lower performance for $^3\overline{\mathrm{He}}$ and the results for the AUC for $\overline{\mathrm{d}}$ and $\overline{\mathrm{t}}$ are identical.

When comparing the results, in which the instances are partitioned according to the reconstructed transverse momentum p_T^{TPC}, all classifiers seem to perform worse for particles in the lowest p_T^{TPC}-bin. However, this decrease in performance compared to the other p_T^{TPC}-bins seems much lower for both GNNs than for the random forest and the AUC of the GNNs in this p_T^{TPC}-bin is still reasonably high with values of above 0.94. When comparing the results across all other p_T^{TPC}-bins, the performance of the GNNs and the random forest seems to be rather constant. These arguments also apply when the performance is evaluated for individual particles in certain transverse momentum bins (cf. appendix section A.4.3), where the lowest performance of all classifiers was observed for the lowest p_T^{TPC}-bin of $^3\overline{\mathrm{He}}$-particles. However, the AUC in this setting indicated a reasonable classification and was additionally much higher compared to the result of the random forest.

7.2.3 Purity comparison with the GNNs at different transverse momentum

As an additional step, the performance of the different classifiers was compared by considering the output purities at a fixed efficiency level across different transverse momenta. Like in the previous subsection, the test instances of the models were grouped into eight p_T^{TPC}-bins of the same width between $0.75\,\mathrm{GeV}/c$ and $10.75\,\mathrm{GeV}/c$. Afterwards, the purity of the random forest and both GNN models were calculated in the different p_T^{TPC}-bins while fixing the efficiencies in all bins to the predefined values of 70%, 80% and 90%. The results of this are illustrated in figure 21.

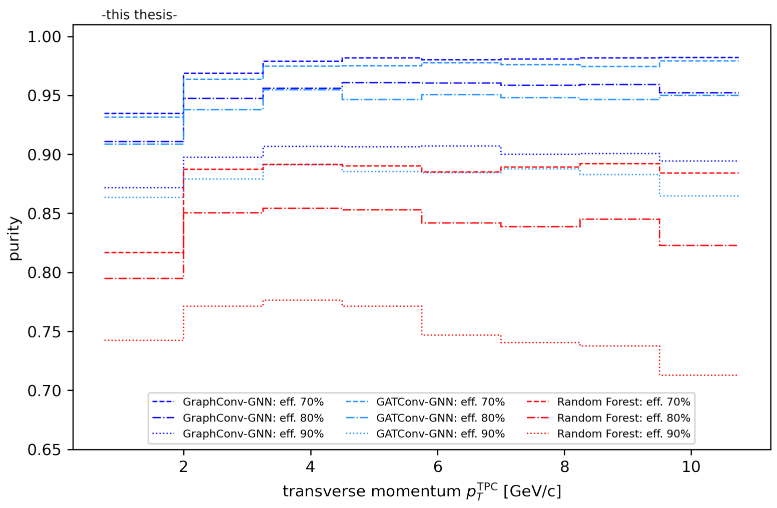
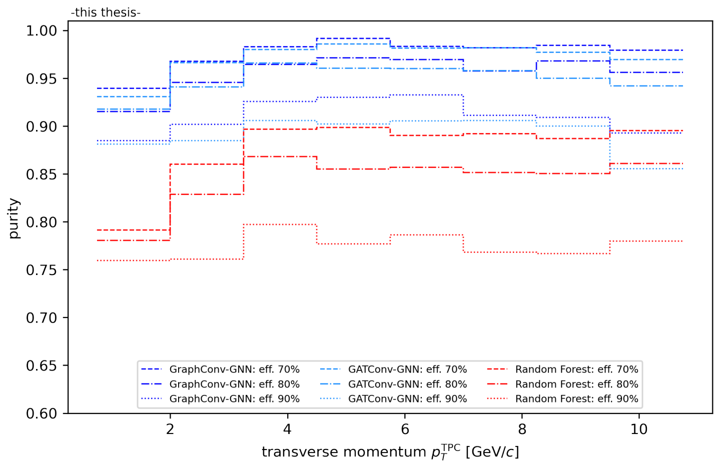


Figure 21: Comparison of the GNN purities to the purity of the random forest model in eight different p_T^{TPC}-bins at predefined efficiency levels of 70%, 80% and 90%.

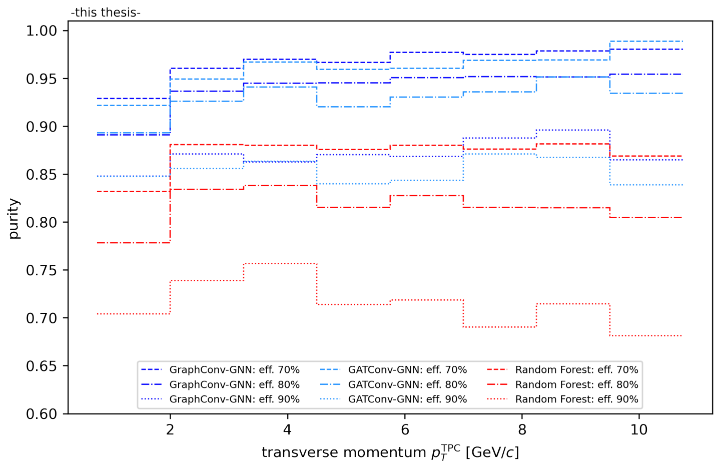
As it turns out, the purities of both GNNs are significantly higher than the ones of the random forest, when comparing the results at equal predefined efficiency levels. The distance between the purity curves grows with an increased predefined efficiency level. With a few exceptions, the purities of the *GraphConv*-GNN are a little higher than for the *GATConv*-GNN and this difference seems to increase with a growing efficiency level. Additionally, the purity of the random forest decreases significantly at efficiencies of 70% and 80% in the bin with the lowest p_T^{TPC}-values, which is an effect that is not that pronounced when considering both GNNs. This effect also matches the observation of decreased classifier performances in the bin with the lowest p_T^{TPC} in subsection 7.2.2.

Further plots, in which the above comparison is shown with additional subdivision taking into account different antinuclei species, are provided in figure 22.

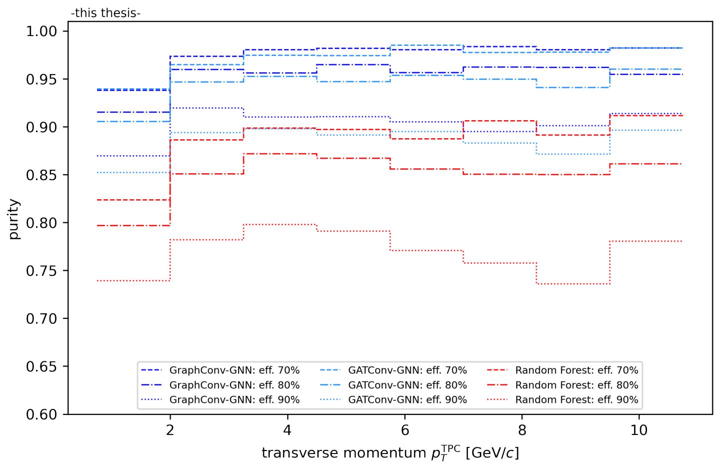
Essentially, the above-mentioned points also apply, when the results are considered with further antinuclei separation. Though, it was seen that the purities of both GNNs are more stable across different p_T^{TPC}-ranges for $^3\overline{\mathrm{He}}$- and $\bar{\mathrm{t}}$-nuclei than for antideuterons, which is particularly noticeable at an efficiency level of 90%.
In the end, it can be said that the GNNs were shown to perform overall better than the trained random forest. As it has already been pointed out, there might exist possibilities to improve



(a) Results for $^3\overline{\mathrm{He}}$.



(b) Results for $\overline{\mathrm{d}}$.



(c) Results for $\overline{\mathrm{t}}$.

Figure 22: Comparison of the GNN purities to the purity of the random forest model in eight different p_T^{TPC}-bins at predefined efficiency levels of 70%, 80% and 90% with additional separation of different antinuclei species.

the feature selection of the random forest and therefore gain performance by constructing a better underlying model.

However, it was also shown that the GNNs provide a rather constant performance over wide ranges of different transverse momentum and also for different antinuclei species under consideration of individual AUCs on data subsets, which is an important point for their application in the scope of future physics analyses.

8 Working point determination for a GNN

In this chapter, a first outlook towards the application of the GNNs in a physics analysis is provided. As it was motivated in the introduction, the identification of inelastic interactions of antinuclei in the TRD is of relevance during the determination of their cross-sections with hadronic components inside a predefined detector volume.

Within the investigations using the GNNs, a subset of overall N_0 candidates was considered, which were all required to have at least seven tracklets assigned to their underlying TPC track. This sample of N_0 candidates consists of N_s true signal and N_b true background instances. Consequently, one can assign particles with seven or more assigned tracklets an interaction probability p_{int} with

$$p_{\text{int}} = \frac{N_s}{N_0}, \tag{12}$$

which can be used to calculate the true interaction cross-section referring to the respective particle sample.

During a real physics application, the true class of a given antinucleus is not known and needs to be predicted by the trained model. Specifically, the consideration of a new sample of size N_0 with seven or more tracklets requires deriving an estimate $\tilde{N}_s$ for the number of true signal events from the final output of the classifier. It can be shown (cf. appendix section A.5), that the total number of true signal events can be estimated by

$$\tilde{N}_s = \frac{n_+ - \varepsilon_b N_0}{\varepsilon_s - \varepsilon_b}, \tag{13}$$

where n_+ is the total number of instances that are classified as signal. ε_s and ε_b denote the respective signal and background efficiency with which the GNN is used.

A working point of the classifier in a physics analysis, which is defined by a corresponding pair $(\varepsilon_b, \varepsilon_s)$ of background and signal efficiency (i.e. a point on a ROC curve), should be chosen such that statistical fluctuations of the inferred quantity $\tilde{N}_s$ become minimal[4]. A complete derivation, which takes into account fluctuations of $\tilde{N}_s$ due to the number of positively classified instances n_+ and also due to the signal and background efficiencies ε_s and ε_b, as they were determined on a test set of finite size, is sketched in the appendix in section A.5. In the following consideration, fluctuations of the efficiencies were neglected and only the variations of n_+ were taken into account.

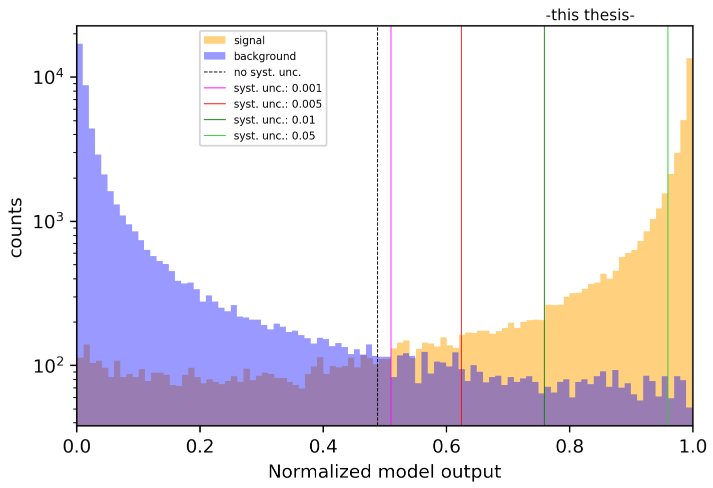
Under this assumption, the relative statistical uncertainty of $\tilde{N}_s$ can be calculated with

$$\frac{\Delta\tilde{N}_s}{\tilde{N}_s} = \frac{\sqrt{\varepsilon_s(1-\varepsilon_s)(N_0\varepsilon_b - n_+) + \varepsilon_b(1-\varepsilon_b)(n_+ - N_0\varepsilon_s)}}{\sqrt{\varepsilon_b - \varepsilon_s}\,|N_0\varepsilon_b - n_+|}, \tag{14}$$

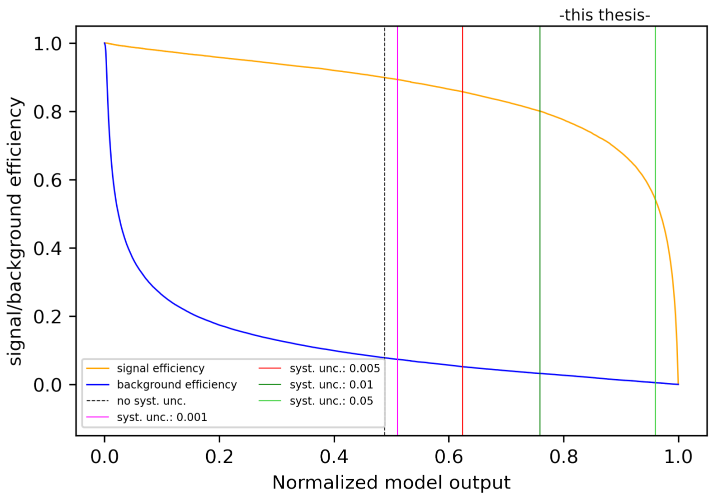
which was determined using linear error propagation and the sketch of the derivation, that is shown in the appendix in section A.5.

[4]In the following considerations, the effect of the preselection on the number of tracklets, which was performed at the end of chapter 4, was not taken into account and a sample of fixed size N_0 with seven or more tracklets was considered.

This expression only contains the number of positive classified instances n_+, the respective background and signal efficiencies ε_s and ε_b which are related within the ROC curve and the total number of events N_0 in the respective sample. For the following purposes, the test set was not used with additional separation criteria, e.g. a partition due to different antinuclei species.



(a) Working points within the distributions of the normalized model output scores for signal and background instances.



(b) Signal and background efficiencies as a function of normalized model output scores.

Figure 23: Illustration of the results of the working point determination for the *GraphConv*-model. In both figures, the blacked dashed line shows the optimal cut on the model score, if only statistical uncertainties were taken into account. Colored vertical lines illustrate the optimal working points when additional systematic fluctuations with a certain magnitude of the signal and background efficiencies are assumed. Further information about this is provided in the text.

An iteration over all pairs of efficiencies $(\varepsilon_b, \varepsilon_s)$ in the ROC curve of the test set then yields an optimal working point, which also corresponds to applying a cut on the normalized network output score. This is illustrated in figure 23(a), where the normalized signal score of the trained *GraphConv*-model for all test instances is shown in a histogram separated for signal and background.

The normalization was carried out by applying a softmax function on the final score vector of the network. The optimal threshold, that corresponded to the above-mentioned pair of signal and background efficiency $(\varepsilon_b, \varepsilon_s)$, is drawn with a black dashed line.

Further colored lines in figure 23(a) indicate where the optimal threshold would lie, if additional systematic uncertainties, e.g. errors in the beforehand class assignment of the TPC tracks, for the signal and background efficiencies were incorporated within equation (14). As no estimations of systematic uncertainties were performed within this thesis, optimal thresholds were exemplarily calculated for relative uncertainties on the obtained signal and background efficiencies of 0.1%, 0.5%, 1% and 5%. The goal of this was to visualize how the working point of the classifier changes in order to obtain a minimal overall error on the inferred quantity $\tilde{N}_s$. It can be seen that the optimal threshold gets systematically shifted towards higher values of the model output. This is a desired behavior, as additional uncertainties in the respective efficiencies need to be compensated by stronger requirements regarding a low background efficiency and therefore achieve a higher purity of the output. An additional illustration of this is given in figure 23(b), where the model signal and background efficiencies are shown as a function of the above described normalized signal score.

9 Summary and outlook

In this bachelor thesis, an approach for the detection of inelastic interactions of light antinuclei in the Transition Radiation Detector of ALICE using graph neural networks was developed. Therefore, a motivation for the physical relevance of this research topic was provided and an overview of the underlying theory of graph neural networks was given at the beginning of this thesis. This involved an introduction to relevant concepts of deep learning and an explanation of graph networks in general with further details about message-passing neural networks, graph convolutional networks and graph attention networks.

Afterwards, the data preselection from the files of the underlying MC simulation respectively particle reconstruction was described. Therefore, the TPC tracks of three different antinuclei ($\overline{\mathrm{d}}$, $\overline{\mathrm{t}}$ and $^{3}\overline{\mathrm{He}}$) were considered and a dataset to perform a binary classification was derived. TRD tracklets which were assigned to the respective TPC tracks using a DCA-approach provided the baseline for a graph construction, during which the tracklets gave rise to nodes in the graphs with corresponding positional features. The edge construction was performed by defining fully-connected graphs and k-NN graphs, mainly with $k = 5$, in those ensembles, where at least seven tracklets were assigned to the corresponding TPC track. The connectivity in the NN graphs was derived from positional biases on the TRD tracklet positions, for which the pairwise relative distance between the tracklet offsets was taken into account.

In the next step, these two sets of graphs were used for testing first network architectures with different methods of message-passing and network hyperparameters. It was found that providing a set of NN graphs as an input delivered better results than with fully-connected graphs within the framework of the tested network architectures. Therefore, only NN graphs were considered during further analyses. Furthermore, these pilot experiments have also provided two configurations of GNNs and hyperparameters that were carried through additional optimization steps. The first approach relied on a variant of a graph convolutional network and the second one made use of a graph attention network.

During the final trainings, all constructed graphs were used at once and no differentiation according to the antinuclei species or the transverse momentum of certain particles was applied. Both final networks achieved overall test accuracies of clearly above 90% and an overall AUC of above 0.96. Furthermore, the trained classifiers showed overall similar performance when their ROC curves were calculated on subsets of the test set that were separated for different kinds of antinuclei and due to the reconstructed transverse momentum p_T of the particles. Some exceptions to this were found in the lowest p_T-bins, where the performance of the GNNs dropped a little during evaluations with and without particle separation. However, the AUC of both GNNs in the low p_T-bins never decreased too critically and the overall performance on these data subsets remained satisfactory.

In a further step of this thesis, both trained networks were compared to classification approaches relying on cut-based analysis and a simple random forest. Compared to these simple approaches, the GNNs have shown a superior performance and especially provided a significantly higher output purity for all tested predefined signal efficiencies across the complete range of transverse momentum p_T.

Finally, a method for a working point determination of the GNNs was described, that relied on the minimization of statistical fluctuations of the number of predicted inelastic interaction events within the used test set. It was also briefly shown how the results would change if systematic uncertainties of different size on the considered signal and background efficiencies were incorporated in the minimization procedure.

For the future, it needs systematic studies whether the respective data preselection can be improved to further gain performance of the networks, e.g. on the side of the tracklet selection around a given TPC track. Additionally, extra optimization of the graph construction could be performed by trying different solutions for the determination of graph connectivities, e.g. by using dynamic nearest neighbor graph constructions within the training process taking into account node feature updates during the process of message-passing [23]. Moreover, strategies for the GNN improvement from the physics side could be investigated, e.g. by incorporating properties of the respective TPC track into the GNN as a conditional input. So far, these properties are available, but have not been used as additional features during the GNN construction. Besides, further network optimization could be carried out, which might lead to an extra improvement of the network performances.

As Monte Carlo simulations are typically not able to represent the experimental environment perfectly, it needs to be investigated, whether the developed methods behave well on real data and how exactly the transition between these kinds of data could be executed. During this step, it is of interest how the application of GNNs might improve the mostly relevant inference on the overall interaction probability (cf. chapter 8) compared to other classification approaches.

It should be mentioned that the training and optimization of the GNNs is a costly procedure, especially in terms of computation resources and training time. As it could be seen during the pilot experiments, the performance of the GNNs hugely relies on their configuration and the choosing of reasonable architectures is typically based on trial and error. The same argument applies to finding reasonable graph constructions, if the application does not provide a practicable canonical way for implementing this. Additionally, the interpretation of these models in terms of their decision process is much harder compared to methods of classical machine learning, for example of ensemble predictors like random forests.

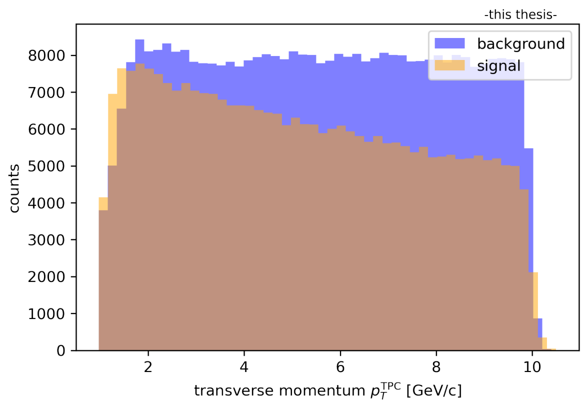
However, it should also be emphasized that the here presented application of GNNs allowed for operating on low-level feature information from the experiment, which was enriched by a relational structure and delivered very good results. Furthermore, graph neural networks provided the advantage, that they can elegantly cope with data that exposes a heterogeneous structure, which is often the case within many applications in HEP. This includes that once a reasonable way of data representation in terms of graphs is found, not much explicit feature engineering is required anymore. This can be an advantage, if the data generating processes are not accessible by well established physical quantities serving as high-level and especially interpretable feature inputs. Therefore, graph neural networks have been shown to provide a considerable method to identify inelastic interaction processes of light antinuclei in the TRD and the here developed strategy might also be applicable to similar physical problems, which is a further objective that should be investigated in the future.

A Appendix

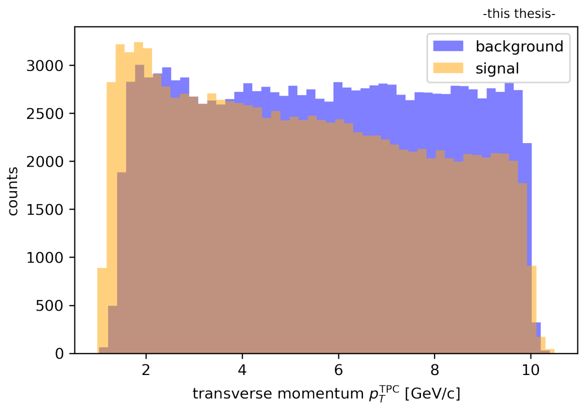
In this part of the thesis, additional material that was referenced in the main text is provided.

A.1 p_T^{TPC}-distributions of preselected antinuclei

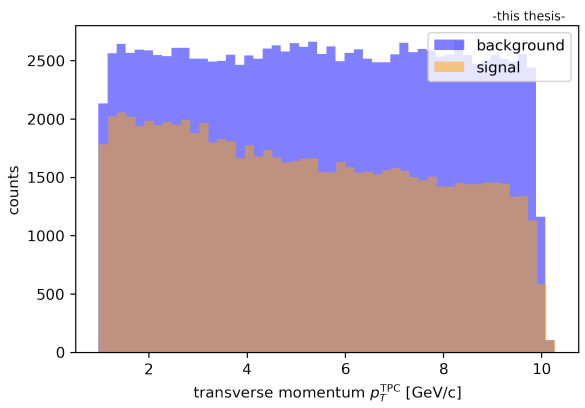
In this part of the appendix, plots of p_T^{TPC}-distributions are shown for different antinuclei.



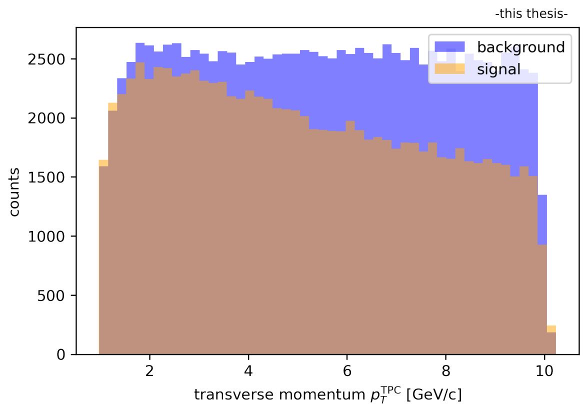
(a) Results for all antinuclei without further separation.



(b) Results for $^3\overline{\mathrm{He}}$.



(c) Results for $\overline{\mathrm{d}}$.



(d) Results for $\overline{\mathrm{t}}$.

Figure 24: p_T^{TPC}-distributions of different antinuclei.

A.2 Additional results from pilot experiments

In this part of the appendix the remaining results of the pilot experiments on the GNN architecture that were described in chapter 5, are shown. This especially includes the full results for the investigations on fully-connected graphs and the ones using 5-NN graphs with two and four message-passing layers.

	GCNConv			GraphConv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.770 ± 0.002	0.780 ± 0.003	0.773 ± 0.003	0.851 ± 0.001	0.851 ± 0.001	0.852 ± 0.001
validation accuracy	0.756 ± 0.003	0.769 ± 0.002	0.768 ± 0.003	0.845 ± 0.001	0.846 ± 0.002	0.844 ± 0.002
test accuracy	0.758 ± 0.003	0.766 ± 0.003	0.763 ± 0.002	0.847 ± 0.002	0.847 ± 0.002	0.845 ± 0.002
AUC	0.850 ± 0.002	0.857 ± 0.002	0.852 ± 0.002	0.913 ± 0.002	0.914 ± 0.002	0.912 ± 0.002

Table 10: Results of pilot experiments with 5-NN graphs for two message-passing layers using two graph convolutional methods.

	GATConv			GATV2Conv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.849 ± 0.002	0.851 ± 0.002	0.848 ± 0.002	0.848 ± 0.001	0.849 ± 0.002	0.847 ± 0.001
validation accuracy	0.842 ± 0.002	0.844 ± 0.002	0.843 ± 0.002	0.842 ± 0.002	0.843 ± 0.001	0.842 ± 0.002
test accuracy	0.843 ± 0.002	0.846 ± 0.001	0.845 ± 0.002	0.845 ± 0.002	0.842 ± 0.002	0.844 ± 0.002
AUC	0.910 ± 0.002	0.913 ± 0.001	0.912 ± 0.001	0.912 ± 0.002	0.906 ± 0.002	0.910 ± 0.002

Table 11: Results of pilot experiments with 5-NN graphs for two message-passing layers using two different methods implementing a graph attention network.

	GCNConv			GraphConv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.773 ± 0.003	0.774 ± 0.003	0.779 ± 0.002	0.847 ± 0.001	0.849 ± 0.001	0.849 ± 0.001
validation accuracy	0.760 ± 0.002	0.764 ± 0.003	0.760 ± 0.003	0.844 ± 0.001	0.843 ± 0.002	0.844 ± 0.002
test accuracy	0.767 ± 0.003	0.768 ± 0.002	0.765 ± 0.003	0.843 ± 0.002	0.845 ± 0.001	0.847 ± 0.001
AUC	0.857 ± 0.002	0.860 ± 0.002	0.855 ± 0.002	0.908 ± 0.002	0.911 ± 0.001	0.913 ± 0.001

Table 12: Results of pilot experiments with 5-NN graphs for four message-passing layers using two graph convolutional methods.

	GATConv			GATV2Conv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.849 ± 0.002	0.848 ± 0.002	0.847 ± 0.002	0.842 ± 0.001	0.842 ± 0.001	0.843 ± 0.001
validation accuracy	0.844 ± 0.002	0.843 ± 0.002	0.845 ± 0.002	0.836 ± 0.001	0.838 ± 0.002	0.839 ± 0.001
test accuracy	0.843 ± 0.002	0.843 ± 0.001	0.845 ± 0.002	0.837 ± 0.002	0.840 ± 0.001	0.839 ± 0.002
AUC	0.910 ± 0.001	0.907 ± 0.001	0.908 ± 0.001	0.905 ± 0.001	0.908 ± 0.001	0.906 ± 0.002

Table 13: Results of pilot experiments with 5-NN graphs for four message-passing layers using two different methods implementing a graph attention network.

	GCNConv			GraphConv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.738 ± 0.004	0.740 ± 0.004	0.746 ± 0.004	0.777 ± 0.004	0.773 ± 0.003	0.767 ± 0.003
validation accuracy	0.734 ± 0.004	0.735 ± 0.004	0.738 ± 0.004	0.764 ± 0.004	0.762 ± 0.004	0.761 ± 0.004
test accuracy	0.733 ± 0.004	0.732 ± 0.004	0.741 ± 0.004	0.765 ± 0.004	0.767 ± 0.004	0.758 ± 0.004
AUC	0.789 ± 0.004	0.787 ± 0.003	0.793 ± 0.003	0.845 ± 0.003	0.853 ± 0.003	0.837 ± 0.003

Table 14: Results of pilot experiments with fully-connected graphs for two message-passing layers using two graph convolutional methods.

	GATConv			GATV2Conv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.830 ± 0.002	0.827 ± 0.003	0.829 ± 0.002	0.835 ± 0.003	0.834 ± 0.003	0.832 ± 0.003
validation accuracy	0.831 ± 0.003	0.825 ± 0.003	0.828 ± 0.003	0.833 ± 0.002	0.835 ± 0.003	0.834 ± 0.003
test accuracy	0.827 ± 0.003	0.828 ± 0.003	0.824 ± 0.002	0.832 ± 0.003	0.833 ± 0.003	0.829 ± 0.003
AUC	0.901 ± 0.002	0.902 ± 0.002	0.898 ± 0.002	0.907 ± 0.002	0.907 ± 0.002	0.904 ± 0.002

Table 15: Results of pilot experiments with fully-connected graphs for two message-passing layers using two different methods implementing a graph attention network.

	GCNConv			GraphConv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.721 ± 0.004	0.723 ± 0.004	0.721 ± 0.004	0.771 ± 0.004	0.758 ± 0.004	0.757 ± 0.004
validation accuracy	0.725 ± 0.003	0.724 ± 0.003	0.718 ± 0.004	0.755 ± 0.004	0.759 ± 0.004	0.751 ± 0.003
test accuracy	0.721 ± 0.004	0.720 ± 0.004	0.716 ± 0.003	0.752 ± 0.004	0.757 ± 0.003	0.754 ± 0.004
AUC	0.773 ± 0.004	0.771 ± 0.003	0.768 ± 0.003	0.816 ± 0.004	0.824 ± 0.004	0.791 ± 0.004

Table 16: Results of pilot experiments with fully-connected graphs for three message-passing layers using two graph convolutional methods.

	GATConv			GATV2Conv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.834 ± 0.003	0.830 ± 0.004	0.829 ± 0.004	0.837 ± 0.003	0.838 ± 0.003	0.836 ± 0.003
validation accuracy	0.829 ± 0.004	0.831 ± 0.003	0.827 ± 0.003	0.833 ± 0.002	0.830 ± 0.002	0.828 ± 0.003
test accuracy	0.827 ± 0.003	0.829 ± 0.002	0.828 ± 0.002	0.832 ± 0.003	0.831 ± 0.003	0.830 ± 0.003
AUC	0.901 ± 0.002	0.904 ± 0.002	0.903 ± 0.002	0.909 ± 0.002	0.908 ± 0.003	0.906 ± 0.003

Table 17: Results of pilot experiments with fully-connected graphs for three message-passing layers using two different methods implementing a graph attention network.

	GCNConv			GraphConv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.718 ± 0.004	0.724 ± 0.004	0.719 ± 0.004	0.764 ± 0.003	0.762 ± 0.004	0.758 ± 0.004
validation accuracy	0.708 ± 0.003	0.711 ± 0.003	0.712 ± 0.003	0.753 ± 0.004	0.756 ± 0.004	0.759 ± 0.003
test accuracy	0.712 ± 0.004	0.713 ± 0.004	0.715 ± 0.004	0.762 ± 0.004	0.755 ± 0.004	0.754 ± 0.003
AUC	0.745 ± 0.004	0.743 ± 0.004	0.751 ± 0.004	0.805 ± 0.004	0.782 ± 0.003	0.775 ± 0.004

Table 18: Results of pilot experiments with fully-connected graphs for four message-passing layers using two graph convolutional methods.

	GATConv			GATV2Conv		
number of hidden channels	**64**	**128**	**256**	**64**	**128**	**256**
training accuracy	0.831 ± 0.004	0.833 ± 0.004	0.827 ± 0.003	0.834 ± 0.003	0.832 ± 0.003	0.834 ± 0.002
validation accuracy	0.827 ± 0.002	0.830 ± 0.003	0.824 ± 0.002	0.829 ± 0.004	0.828 ± 0.003	0.827 ± 0.003
test accuracy	0.831 ± 0.002	0.829 ± 0.002	0.826 ± 0.003	0.829 ± 0.004	0.830 ± 0.003	0.829 ± 0.003
AUC	0.903 ± 0.002	0.904 ± 0.003	0.901 ± 0.003	0.907 ± 0.002	0.906 ± 0.002	0.906 ± 0.002

Table 19: Results of pilot experiments with fully-connected graphs for four message-passing layers using two different methods implementing a graph attention network.

A.3 Comparison of cut-based models and GNNs for different antinuclei

This part of the appendix shows further plots on the performance comparison of the two different GNNs with the constructed cut-based models. The strategy for creating the plots below and the construction of the respective models were described in section 7.1.

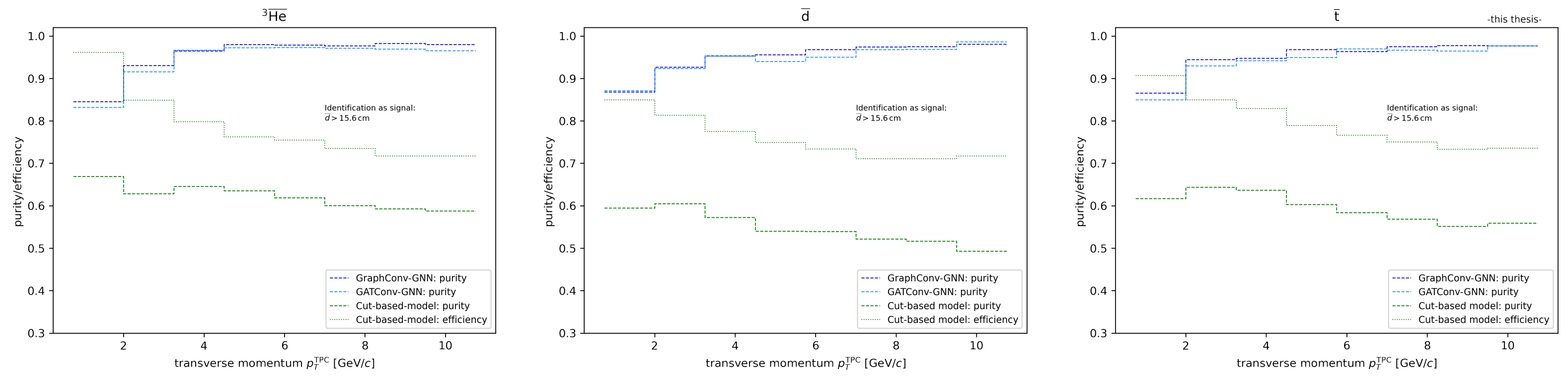


Figure 25: Comparison of the GNN purity to the purity of the cut-based model on the mean tracklet distance in different p_T^{TPC}-bins at equal efficiency for different species of antinuclei. The species of the respective antinucleus can be seen in the title of each plot. The binwise efficiencies for which the resulting purities are shown are depicted as a dotted green line.

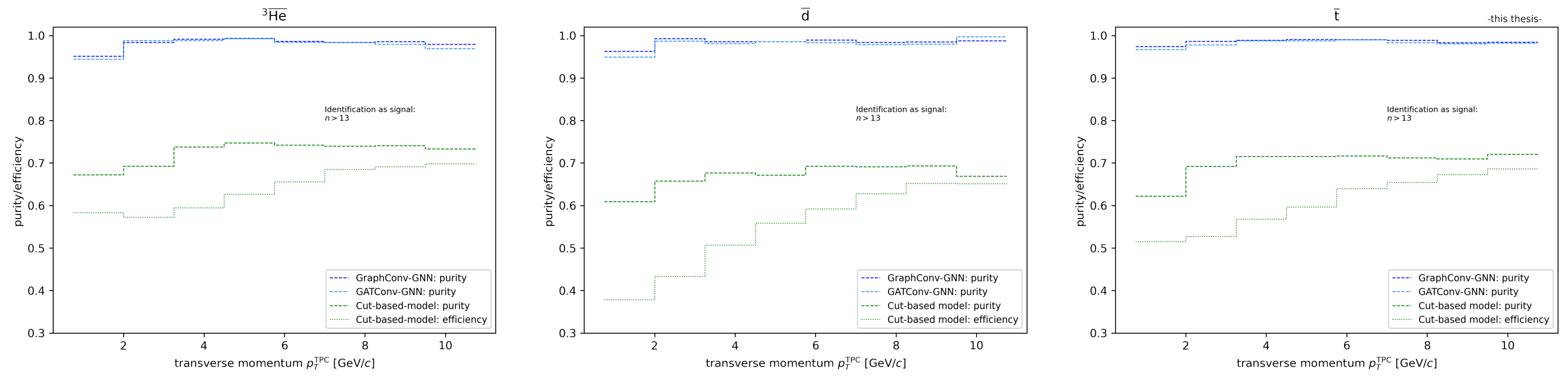
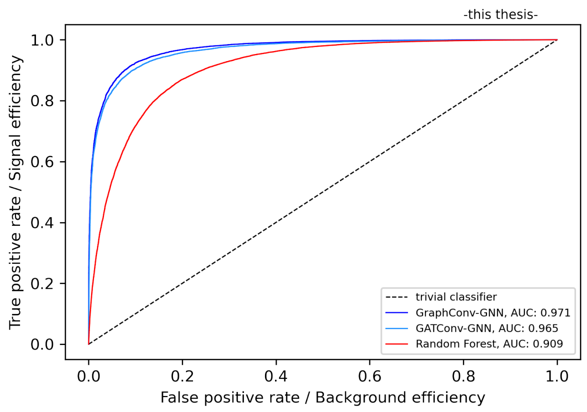


Figure 26: Comparison of the GNN purity to the purity of the cut-based model on the number of assigned tracklets in different p_T^{TPC}-bins at equal efficiency for different species of antinuclei. The species of the respective antinucleus can be seen in the title of each plot. The binwise efficiencies for which the corresponding purities are shown are depicted as a green dotted line.

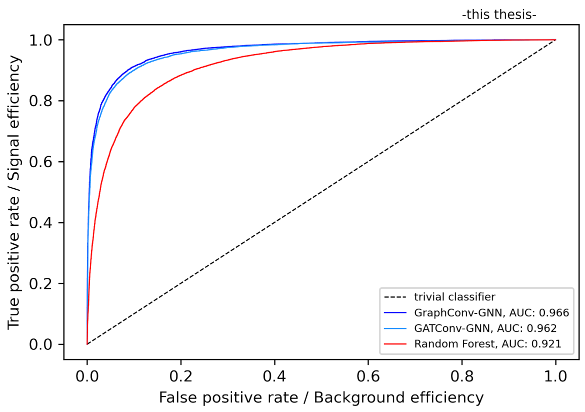
A.4 Comparison of random forest and GNNs

In this part of the appendix, additional plots are provided on the performance comparison between the trained GNNs and the random forest.

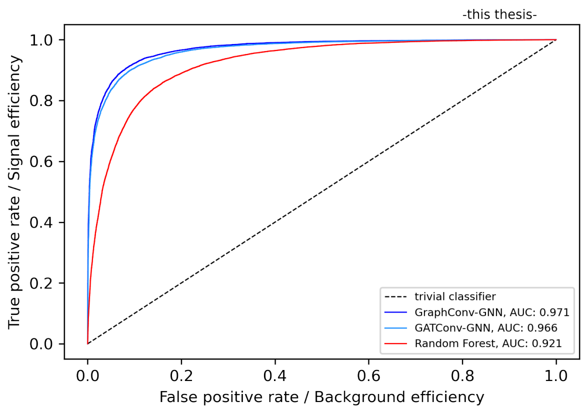
A.4.1 ROC curves of random forest and GNNs during application to different antinuclei



(a) Results for $^3\overline{\mathrm{He}}$.



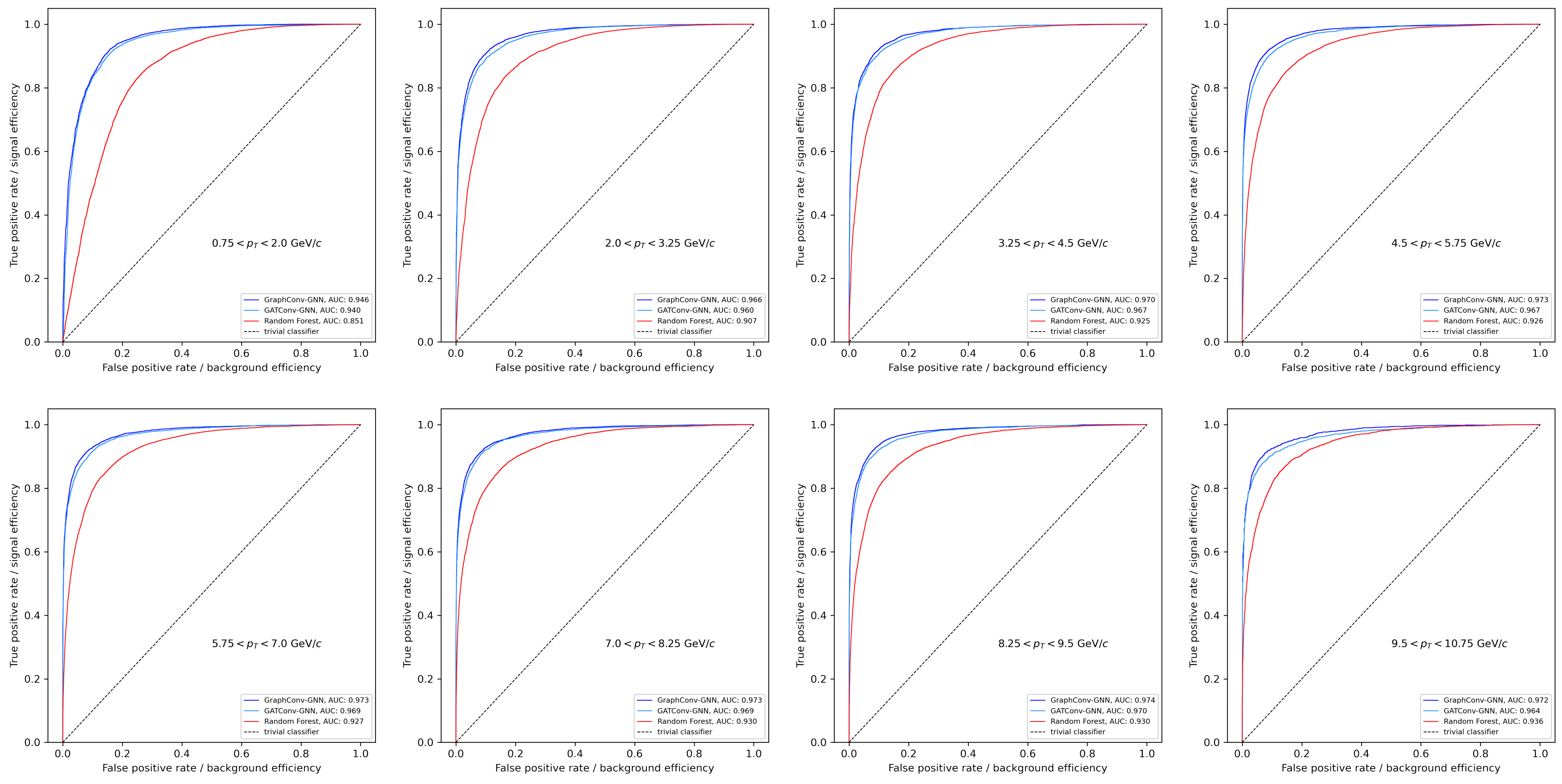
(b) Results for $\overline{\mathrm{d}}$.



(c) Results for $\overline{\mathrm{t}}$.

Figure 27: ROC curves for the separate application of the random forest and the GNNs to instances of different antinuclei species. The black dotted line in all plots, which is titled with "trivial classifier", corresponds to a classifier that would assign the labels for all given instances randomly with equal probability.

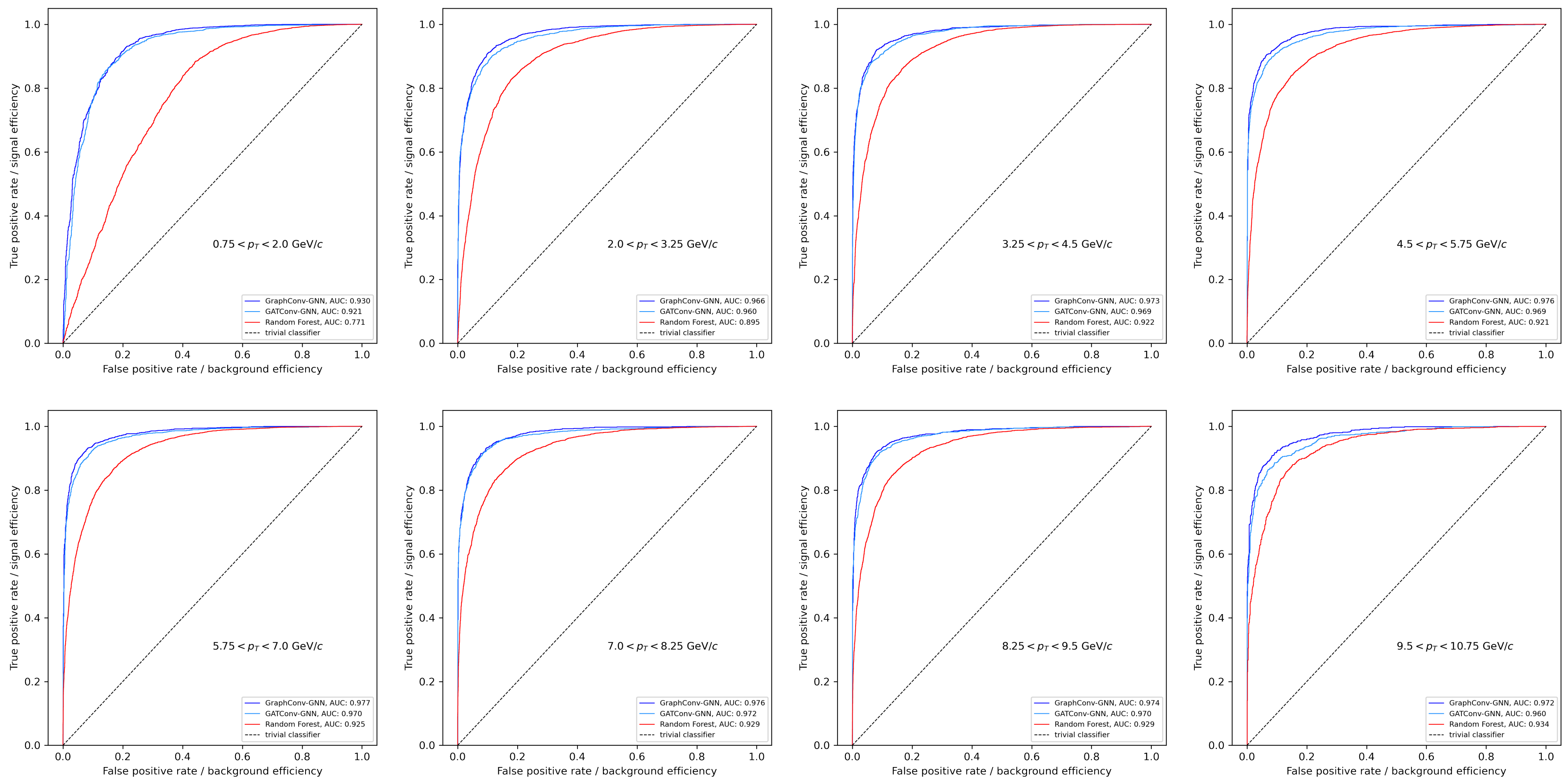
A.4.2 ROC curves of random forest and GNNs during application to particles of different transverse momentum



-this thesis-

Figure 28: ROC curves for the separate application of the random forest and the GNNs to instances of different reconstructed transverse momentum p_T^{TPC}. The black dotted line in all figures, which is titled with "trivial classifier", corresponds to a classifier that would assign the labels for all given instances randomly with equal probability.

A.4.3 ROC curves of random forest and GNNs during application to different antinuclei of different transverse momentum



-this thesis-

Figure 29: ROC curves for the application of the random forest and the GNNs to $^{3}\overline{\mathrm{He}}$-nuclei of different transverse momentum p_T^{TPC}. The black dotted line in all plots, which is titled with "trivial classifier", corresponds to a classifier that would assign the labels for all given instances randomly with equal probability.

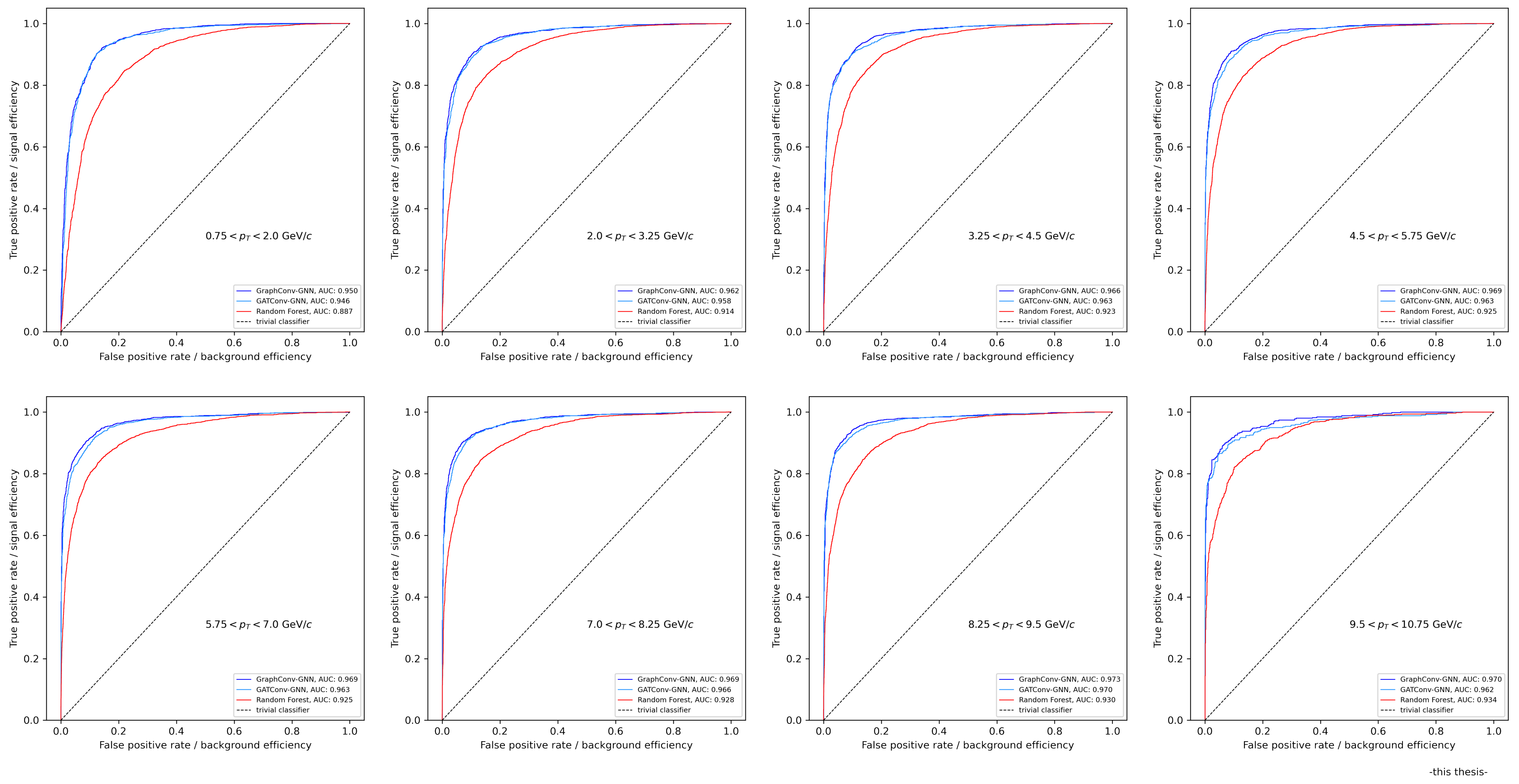


Figure 30: ROC curves for the application of the random forest and the GNNs to $\overline{\mathrm{d}}$-nuclei of different transverse momentum p_T^{TPC}. The black dotted line in all plots, which is titled with "trivial classifier", corresponds to a classifier that would assign the labels for all given instances randomly with equal probability.

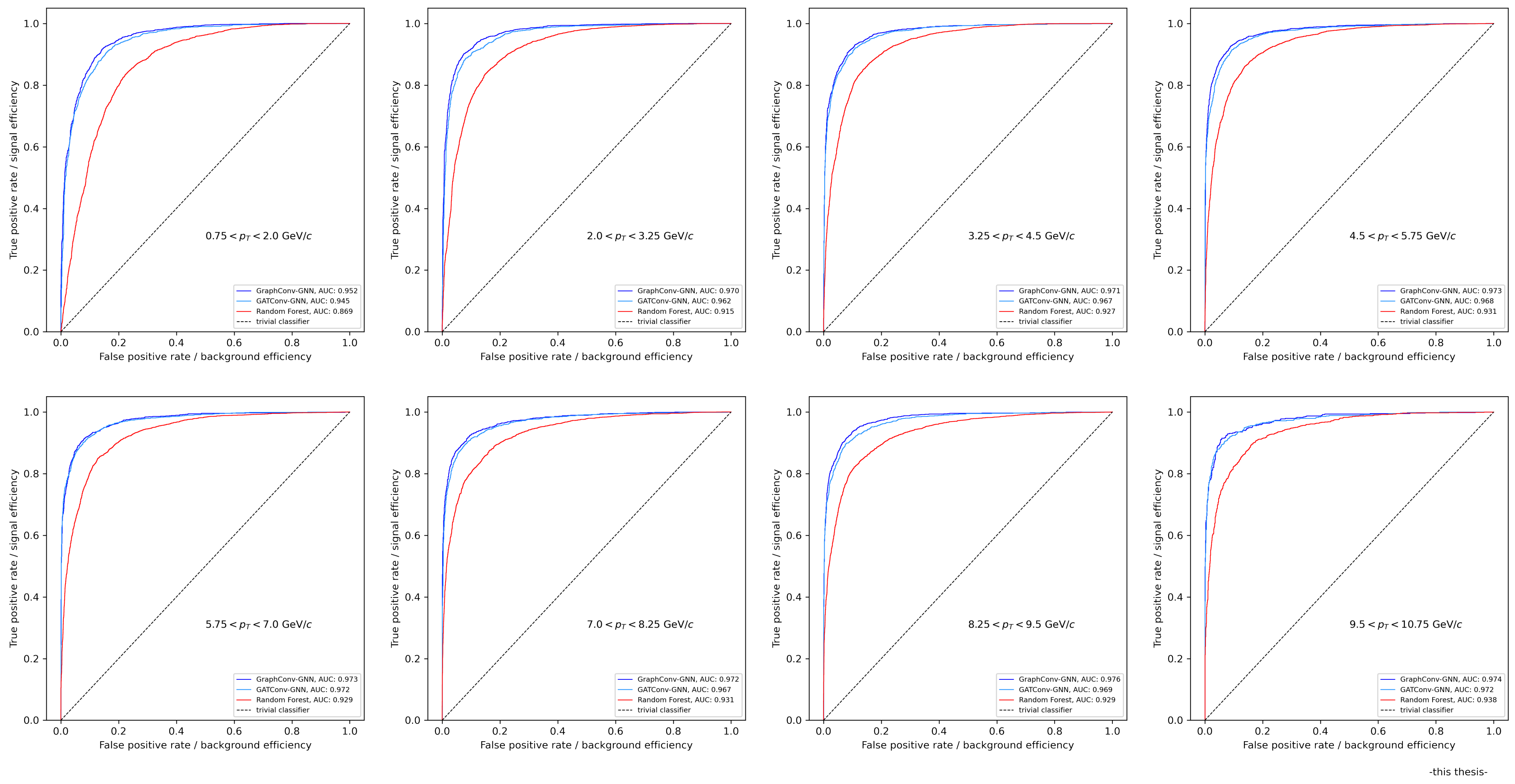


Figure 31: ROC curves for the application of the random forest and the GNNs to $\bar{t}$-nuclei of different transverse momentum p_T^{TPC}. The black dotted line in all plots, which is titled with "trivial classifier", corresponds to a classifier that would assign the labels for all given instances randomly with equal probability.

A.5 Additional calculations for the working point determination

In this section of the appendix, additional calculations for the working point determination that was shown in chapter 8 of this thesis are provided. Since a binary classification task is considered, the total number of candidates N_0 in a sample that is presented to the GNN can be written as

$$N_0 = N_s + N_b = n_+ + n_-, \tag{15}$$

where N_s denotes the number of true signal instances and N_b the number of true background instances. Furthermore, n_+ is the number of instances, that were classified as signal and n_- the number of instances classified as background. The signal and background efficiency ε_s and ε_b are defined with

$$\varepsilon_s = \frac{n_s}{N_s} \tag{16}$$

$$\varepsilon_b = \frac{n_b}{N_b}, \tag{17}$$

in which n_s denotes the number of signal events, that are classified correctly and n_b the number of background events that are classified as signal. As a consequence, n_+ can be expressed by

$$n_+ = n_s + n_b = \varepsilon_s N_s + \varepsilon_b N_b, \tag{18}$$

which implies equation (13) under use of equation (15).

As it was sketched in section 8, the search for an optimal working point of the GNN is based on the minimization of the statistical error on the inferred number $\tilde{N}_s$ of true signal events. Applying linear error propagation to equation (13) yields

$$\Delta\tilde{N}_s = \sqrt{\sigma_{n_+}^2 \left(\frac{\partial \tilde{N}_s}{\partial n_+}\right)^2 + \sigma_{\varepsilon_s}^2 \left(\frac{\partial \tilde{N}_s}{\partial \varepsilon_s}\right)^2 + \sigma_{\varepsilon_b}^2 \left(\frac{\partial \tilde{N}_s}{\partial \varepsilon_b}\right)^2}, \tag{19}$$

where the variance of n_+ is given by

$$\sigma_{n_+}^2 = \sigma_{n_s}^2 + \sigma_{n_b}^2. \tag{20}$$

n_s and n_b follow binomial distributions regarding the fact, whether a true signal or background candidate is classified to be signal or background. Consequently, the variances in equation (20) are given by

$$\sigma_{n_s}^2 = N_s \varepsilon_s (1 - \varepsilon_s) \tag{21}$$

$$\sigma_{n_b}^2 = (N_0 - N_s)\varepsilon_b (1 - \varepsilon_b). \tag{22}$$

Because of equation (16), the variance $\sigma^2_{\varepsilon_s}$ can be derived using error propagation with

$$\sigma^2_{\varepsilon_s} = \frac{\sigma^2_{n_s}}{N_s^2} = \frac{\varepsilon_s(1-\varepsilon_s)}{N_s} \tag{23}$$

and analogously for $\sigma^2_{\varepsilon_b}$ using equation (17).

After calculating the derivatives in equation (19), dividing by $\tilde{N}_s$ and inserting equation (13) for N_s, an expression for the relative statistical error of $\tilde{N}_s$ can be obtained, which also takes into account statistical fluctuations in the efficiencies ε_s and ε_b. Systematic uncertainties of the signal and background efficiencies on the respective particle sample can be incorporated in equation (19) by adding them in quadrature and weighting them with respective derivatives.

Bibliography

[1] Risa H. Wechsler and Jeremy L. Tinker. "The connection between galaxies and their dark matter halos". *Annual Review of Astronomy and Astrophysics*, 56(1):435–487, September 2018. URL: https://doi.org/10.1146%2Fannurev-astro-081817-051756, doi:10.1146/annurev-astro-081817-051756.

[2] Alejandro Ibarra and Sebastian Wild. "Prospects of antideuteron detection from dark matter annihilations or decays at AMS-02 and GAPS". *Journal of Cosmology and Astroparticle Physics*, 2013(02):021–021, February 2013. URL: https://doi.org/10.1088%2F1475-7516%2F2013%2F02%2F021, doi:10.1088/1475-7516/2013/02/021.

[3] K. Perez and P. von Doetinchem *et al.*. "Cosmic-ray antinuclei as messengers of new physics: Status and outlook for the new decade". *Journal of Cosmology and Astroparticle Physics*, 2020(08):035–035, August 2020. doi:10.1088/1475-7516/2020/08/035.

[4] Masayoshi Kozai. "The GAPS experiment – a search for cosmic-ray antinuclei from dark matter". *Journal of Physics: Conference Series*, 1468(1):012049, February 2020. doi:10.1088/1742-6596/1468/1/012049.

[5] AMS Collaboration and Rui Pereira. "Astrophysics with the AMS-02 experiment". *arXiv*, 2007. URL: https://arxiv.org/abs/0710.0984, doi:10.48550/ARXIV.0710.0984.

[6] Fiorenza Donato, Nicolao Fornengo, and Pierre Salati. "Antideuterons as a signature of supersymmetric dark matter". *Physical Review D*, 62(4), July 2000. URL: https://doi.org/10.1103%2Fphysrevd.62.043003, doi:10.1103/physrevd.62.043003.

[7] ALICE Collaboration. "First measurement of the absorption of $^{3}\overline{\mathrm{He}}$ nuclei in matter and impact on their propagation in the galaxy", 2022. URL: https://arxiv.org/abs/2202.01549, doi:10.48550/ARXIV.2202.01549.

[8] Sebastian Hornung. "Production of (anti-)^{3}He and (anti-)^{3}H in p–Pb collisions at $\sqrt{s_{\mathrm{NN}}} =$ 5.02 TeV measured with ALICE at the LHC". *Dissertation, University of Heidelberg*, 2020.

[9] The ALICE Collaboration. "Measurement of the Low-Energy Antideuteron Inelastic Cross Section". *Physical Review Letters*, 125(16), October 2020. URL: https://doi.org/10.1103%2Fphysrevlett.125.162001, doi:10.1103/physrevlett.125.162001.

[10] Roman Pasechnik and Michal Šumbera. "Phenomenological Review on Quark–Gluon Plasma: Concepts vs. Observations". *Universe*, 3(1):7, January 2017. URL: https://doi.org/10.3390%2Funiverse3010007, doi:10.3390/universe3010007.

[11] Sven Hoppner. "Implementation of a Stand Alone Tracking Algorithm for the Transition Radiation Detector in ALICE using a Kalman Filter Approach". *Bachelor Thesis, University of Heidelberg*, 2020.

[12] Particle Data Group. "Review on Machine Learning", November 2021, access on 24.05.2022. URL: https://pdg.lbl.gov/2021/reviews/rpp2021-rev-machine-learning.pdf.

[13] The ALICE Collaboration. "The ALICE Transition Radiation Detector: Construction, operation, and performance". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 881:88–127, February 2018. URL: https://doi.org/10.1016%2Fj.nima.2017.09.028, doi:10.1016/j.nima.2017.09.028.

[14] A. Tauro. "ALICE Schematics as during RUN2", 2017, access on 14.07.2022. URL: https://cds.cern.ch/record/2263642.

[15] The ALICE Collaboration. "The ALICE experiment at the CERN LHC". *Journal of Instrumentation*, 3(08):S08002–S08002, August 2008. doi:10.1088/1748-0221/3/08/s08002.

[16] "More details on the ALICE ITS", access on 09.05.2022. URL: https://alice.cern/node/5531.

[17] J. Adolfsson *et al.* The upgrade of the ALICE TPC with GEMs and continuous readout. *Journal of Instrumentation*, 16(03):P03022, March 2021. URL: https://doi.org/10.1088%2F1748-0221%2F16%2F03%2Fp03022, doi:10.1088/1748-0221/16/03/p03022.

[18] The ALICE Collaboration. "Performance of the ALICE experiment at the CERN LHC". *International Journal of Modern Physics A*, 29(24):1430044, 2014. arXiv:https://doi.org/10.1142/S0217751X14300440, doi:10.1142/S0217751X14300440.

[19] Herrmann Kolanoski and Norbert Wermes. "Teilchendetektoren - Grundlagen und Anwendungen". *Springer Spektrum Berlin, Heidelberg*, 2016. URL: https://doi.org/10.1007/978-3-662-45350-6.

[20] Martin Kroesen. Private communication.

[21] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges", 2021. URL: https://arxiv.org/abs/2104.13478, doi:10.48550/ARXIV.2104.13478.

[22] Robert Tibshirani and Trevor Hastie *et al.* "The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition". *Springer New York, NY*, 2009. URL: https://doi.org/10.1007/978-0-387-84858-7.

[23] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. "Graph neural networks in Particle Physics". *Machine Learning: Science and Technology*, 2(2):021001, January 2021. URL: https://doi.org/10.1088%2F2632-2153%2Fabbf9a, doi:10.1088/2632-2153/abbf9a.

[24] Ian Goodfellow *et al.* "Deep Learning", 2016. URL: http://www.deeplearningbook.org.

[25] Gareth James *et al.* "An Introduction to Statistical Learning". *Springer New York, NY*, 2013. URL: `https://doi.org/10.1007/978-1-4614-7138-7`.

[26] Yann LeCun *et al.* "Gradient based learning applied to document recognition", November 1998. URL: `http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf`.

[27] Peter W. Battaglia *et al.* "Relational inductive biases, deep learning, and graph networks", 2018. URL: `https://arxiv.org/abs/1806.01261`, `doi:10.48550/ARXIV.1806.01261`.

[28] Christopher Morris and Martin Ritzert *et al.* "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks", 2018. URL: `https://arxiv.org/abs/1810.02244`, `doi:10.48550/ARXIV.1810.02244`.

[29] Justin Gilmer, Samuel S. Schoenholz, and Patrick F. Riley *et al.* "Neural Message Passing for Quantum Chemistry", 2017. URL: `https://arxiv.org/abs/1704.01212`, `doi:10.48550/ARXIV.1704.01212`.

[30] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering", 2016. URL: `https://arxiv.org/abs/1606.09375`, `doi:10.48550/ARXIV.1606.09375`.

[31] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks", 2016. URL: `https://arxiv.org/abs/1609.02907`, `doi:10.48550/ARXIV.1609.02907`.

[32] Keyulu Xu and Weihua Hu *et al.* "How Powerful are Graph Neural Networks?", 2018. URL: `https://arxiv.org/abs/1810.00826`, `doi:10.48550/ARXIV.1810.00826`.

[33] Petar Veličković and Guillem Cucurull *et al.* "Graph Attention Networks", 2017. URL: `https://arxiv.org/abs/1710.10903`, `doi:10.48550/ARXIV.1710.10903`.

[34] Shaked Brody, Uri Alon, and Eran Yahav. "How Attentive are Graph Attention Networks?", 2021. URL: `https://arxiv.org/abs/2105.14491`, `doi:10.48550/ARXIV.2105.14491`.

[35] Sneha Chaudhari and Varun Mithal *et al.* "An Attentive Survey of Attention Models", 2019. URL: `https://arxiv.org/abs/1904.02874`, `doi:10.48550/ARXIV.1904.02874`.

[36] Fons Rademakers and Philippe Canal *et al.* "root-project/root: v6.20/04", April 2020. `doi:10.5281/zenodo.3895855`.

[37] Alexander Schmah. "TRD-self-tracking repository". URL: `https://github.com/aschmah/TRD-self-tracking`.

[38] S. Agostinelli *et al.* "GEANT4 — a simulation toolkit". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003. URL: `https://www.sciencedirect.com/science/article/pii/S0168900203013688`, `doi:https://doi.org/10.1016/S0168-9002(03)01368-8`.

[39] "AliRoot Core: AliHelix Class Reference", access on 10.06.2022. URL: `http://alidoc.cern.ch/AliRoot/master/class_ali_helix.html`.

[40] Jens Wiechula. "Commissioning and Calibration of the ALICE-TPC". *Dissertation, Goethe-University of Frankfurt*, 2008.

[41] Adam Paszke and Sam Gross *et al.* "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[42] Matthias Fey and Jan Eric Lenssen. "Fast Graph Representation Learning with PyTorch Geometric", 2019. URL: `https://arxiv.org/abs/1903.02428`, `doi:10.48550/ARXIV.1903.02428`.

[43] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. "cuda", release: 10.02.1989, 2020. URL: `https://developer.nvidia.com/cuda-toolkit`.

[44] Jim Pivarski and Pratyush Das *et al.* "scikit-hep/uproot3: 3.14.4", February 2021. `doi:10.5281/zenodo.4537826`.

[45] F.F. Yao D. Eppstein, M.S. Paterson. "On Nearest-Neighbor Graphs", 1997. URL: `https://doi.org/10.1007/PL00009293`.

[46] Charles R. Harris and K. Jarrod Millman *et al.* "Array programming with NumPy". *Nature*, 585(7825):357–362, September 2020. `doi:10.1038/s41586-020-2649-2`.

[47] F. Pedregosa and G. Varoquaux *et al.* "scikit-learn: Machine learning in Python". *Journal of Machine Learning Research*, 12:2825–2830, 2011.

List of acronyms

ADC	Analog Digital Converter
ALICE	A Large Ion Collider Experiment
AMS 02	Alpha Magnetic Spectrometer 02
AUC	Area under Curve
BDT	Boosted Decision Tree
CERN	European Center for Nuclear Research
CNN	Convolutional Neural Network
DCA	Distance of Closest Approach
FCN	Fully-connected network
GAPS	General Antiparticle Spectrometer
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GEM	Gas Electron Multiplier
GN	Graph Network
GNN	Graph Neural Network
HEP	High Energy Physics
ITS	Inner Tracking System
LHC	Large Hadron Collider
MC	Monte Carlo
ML	Machine Learning
MLP	Multilayer perceptron
MPNN	Message Passing Neural Network
MWPC	Multiwire Proportional Chamber
NN-graph	Nearest Neighbor-graph
PID	Particle Identification
QGP	Quark-Gluon Plasma
ReLU	Rectified Linear Unit
RF	Random Forest
ROC	Receiver Operating Characteristic
TPC	Time Projection Chamber
TR	Transition Radiation
TRD	Transition Radiation Detector

List of Figures

List of Tables

Acknowledgement

First, I want to thank Prof. Dr. Klaus Reygers for giving me the opportunity to write my bachelor thesis in the ALICE group and especially having the chance to work on such an interesting topic. I have really enjoyed the constructive discussions with you and I am thankful for you always providing support, when problems of any kind occurred.

I also want to thank Prof. Dr. Silvia Masciocchi for agreeing to be the second referee on my thesis.

Furthermore, I want to thank M. Sc. Martin Krösen for his guidance throughout this thesis and especially the constructive input on the sides of physics and machine learning. Thank you very much for having always been available, even in the evenings and on weekends and the time you have spent on proofreading my thesis.

I also want to thank Dr. Alexander Schmah, who provided the data baseline for this thesis and who was always open and very patient in answering my questions.

Many thanks also go to B. Sc. Sven Hoppner, who helped me a lot at the beginning to get acquainted with C++ and the TRD Self Tracking-code.

Last but not least, I want to thank my parents and my girlfriend Lara, who have always supported me during the last three years of my studies and especially in the last months. Without you, this would not have been possible!

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 04.08.2022,

[illegible]

Maximilian Hammermann