Exploring Deep Learning Paradigms in Astroparticle and High Energy Physics:

From Gamma Ray Source Classification to Anomaly Detection at the LHC

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Thorben Finke, M.Sc.
aus
Erkelenz

Berichter: Prof. Dr. Michael Krämer
Prof. Dr. Martin Erdmann

Tag der mündlichen Prüfung: 22.09.2023

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek verfügbar.
Exploring Deep Learning Paradigms in Astroparticle and High Energy Physics:
From Gamma Ray Source Classification to Anomaly Detection at the LHC

by

Thorben FINKE

Supervisors

Prof. Dr. Michael Krämer
Prof. Dr. Martin Erdmann

This thesis was realized at the

Institute for Theoretical Particle Physics and Cosmology,

RWTH AACHEN UNIVERSITY

Aachen, June 2023
Abstract

Exploring Deep Learning Paradigms in Astroparticle and High Energy Physics:
From Gamma Ray Source Classification to Anomaly Detection at the LHC
by Thorben Finke

The ever-increasing amount of data collected and stored from physics experiments opens up opportunities for new analysis methods. Deep learning has proven to be able to extract complex patterns from huge amounts of data. In this thesis, we explore different deep learning paradigms that can be applied directly to the data.

First, we explore supervised classification of $\gamma$-ray sources detected by the Fermi Large Area Telescope. Some of the sources provided by the Fermi collaboration are already categorized to belong to certain astrophysical source classes. We apply neural networks to extract properties from these labeled sources that are relevant for the categorization of so far unclassified sources. We show that these networks trained on the flux measurements are competitive with previous methods using features designed by experts. Additionally, we incorporate uncertainties into our classification using a Bayesian neural network. Using a toy model, we demonstrate the advantages over deterministic networks. We provide source type candidates for unclassified sources and for blazars of uncertain type. These candidates can be used for population studies to better understand the $\gamma$-ray background. Understanding this background is relevant to search for any $\gamma$-ray signal, e.g. from dark matter annihilation.

We move from supervised to weakly supervised and unsupervised methods as we turn to high energy particle physics, where labels according to signal or background are not available in the collected data. Weakly supervised and unsupervised methods are directly applicable to such data. We demonstrate a large potential increase in sensitivity for searches with control and signal regions using the classification without labels method. As an example, we use the ATLAS mono-jet search and a dark matter signature as a potential new physics signal.

Unsupervised learning can be used to estimate the probability density in many dimensions. We use an autoregressive transformer setup to learn the density of QCD and top jets. Being able to extract the density allows for solving tasks such as classification, reweighting, or data generation. We test the resulting estimate by sampling from it, and trying to distinguish these samples from the data. The resulting poor separation demonstrates the quality of the density estimate.
Zusammenfassung

Erforschung von Deep-Learning-Paradigmen in der Astroteilchen- und Hochenergiephysik

Von der Klassifizierung von Gammastrahlenquellen bis zur Entdeckung von Anomalien am LHC

von Thorben FINKE


Declaration of Authorship

I, Thorben Jörg Finke, declare this thesis and the work presented in it are my own and have been generated by me as the result of my own original research. Hiermit erkläre ich an Eides statt / I do solemnly swear that:

1. This work was done wholly or mainly while in candidature for the doctoral degree at this faculty and university;

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated;

3. Where I have consulted the published work of others or myself, this is always clearly attributed;

4. Where I have quoted from the work of others or myself, the source is always given. This thesis is entirely my own work, with the exception of such quotations;

5. I have acknowledged all major sources of assistance;

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

7. The research carried out during this thesis and described in this work has led to several scientific publications, listed on the following page.

Date

Signature
List of Publications

This thesis is based on some of the research that I carried out over the course of my doctoral studies at the Institute for Theoretical Particle Physics and Cosmology of RWTH Aachen University. This research has led to several publications. In the following, I list each publication and specify my contributions.

1. I contributed to research published in SciPostPhys.10.2.046 [1] together with Elias Bernreuther, Felix Kahlhoefer, Michael Krämer and Alexander Mück. I contributed the implementation and validation of the used networks and the data preprocessing. I contributed to the discussion of results and took part in the writing process.

2. Section 4 is based on my contribution to research published in MNRAS 507 (2021) 3 [2] together with Michael Krämer and Silvia Manconi. I contributed key ideas to the application of neural networks. I implemented and validated the networks. I contributed substantially to the discussion of methods and results and supported the writing of substantial parts of the paper.

3. I contributed to research published in JHEP 06 (2021) 161 [3] together with Michael Krämer, Alessandro Morandini, Alexander Mück and Ivan Oleksiyuk. My contribution consisted in providing key ideas and co-supervising Ivan Oleksiyuk’s Bachelor’s thesis, which led to this publication. I contributed substantially to the discussion of results and writing of the paper.

4. Section 5 is based on my contribution to research published in JCAP 04 (2022) 04, 023 [4] together with Anja Butter, Felicitas Keil, Michael Krämer and Silvia Manconi. I contributed the implementation and validation of the networks, including the Bayesian setup. I contributed substantially to the discussion of methods and results and wrote substantial parts of the paper.

5. Section 6 is based on my contribution to research published in JHEP 08 (2022) 015 [5] together with Michael Krämer, Maximilian Lipp and Alexander Mück. I contributed the implementation and validation of the DGCNN. I co-supervised Maximilian Lipp during his Master’s thesis, which led to this publication. I performed the training and evaluation of the DGCNN for mutual validation of the results. I contributed to the simulation and validation of Monte Carlo data. I contributed substantially to the discussion of results and wrote substantial parts of the paper.

6. I contributed to research submitted for publication to SciPost [6] together with Thorsten Buss, Barry Dillon, Michael Krämer, Alessandro Morandini, Alexander Mück, Ivan Oleksiyuk and Tilman Plehn. I contributed substantially to the discussion of methods and results in the $k$-means section. I validated implementation and results obtained with density estimators on energy flow polynomials and contributed key ideas to the data preprocessing. I contributed to the writing of the paper.
7. Section 7 is based on my contribution to research submitted to JHEP [7] together with Michael Krämer, Alexander Mück and Jan Tönshoff. I contributed the adaption of the physics data to the transformer architecture. I performed the validation of the transformer implementation and implemented the sampling method. I contributed substantially to the discussion of the method and results and wrote substantial parts of the paper.
## Contents

Abstract i  
Zusammenfassung iii  
Declaration of Authorship v  
List of Publications vi  

### I. Introduction 1

### II. The toolbox 5

1. Deep learning, a shallow example 8  
   1.1. Activation, optimization and regularization 10  

2. Architectures 13  
   2.1. Recurrent layers 13  
   2.2. Autoencoders 14  
   2.3. Dynamic graph convolutional neural networks 15  
   2.4. Transformer-encoders 17  

3. Performance measures 21  

### III. Supervised learning for γ-ray classification 25

4. Classification of *Fermi*-LAT sources with deep learning using energy and time spectra 29  
   4.1. The data: AGN and PSR in the 4FGL-DR2 catalog 29  
   4.2. Classification methods 30  
      4.2.1. Fully connected neural network 30  
      4.2.2. Recurrent neural network 31  
   4.3. Results 32  
      4.3.1. Performance on labeled sources of the 4FGL-DR2 catalog 32  
      4.3.2. Cross-match between 3FGL and 4FGL-DR2 catalogs 37  
      4.3.3. Predictions for unclassified sources of the 4FGL-DR2 catalog 39  
   4.4. Using an autoencoder for feature selection 42  
   4.5. Summary 44  

5. Classification of *Fermi*-LAT blazars with Bayesian neural networks 46  
   5.1. The data: blazar subclasses in the 4FGL-DR2 catalog 46
VI. Appendix 117

A. CWoLa with different region sizes 118
B. Significance improvement including systematic uncertainties 119
C. Density estimation for anomaly detection in many dimensions 120

Acronyms 122
Part I.

Introduction
Deep Learning (DL), as a specific type of Machine Learning (ML), is currently revolutionizing many fields of research. Especially since OpenAI released its chatbot ChatGPT [8] to the public in November 2022, it is impossible to avoid discussions about the impact of neural networks and artificial intelligence. However, the model behind this recent breakthrough, namely the transformer model, is already six years old and was published by a team of Google researchers in 2017 [9]. The beginning of the recent excitement about DL in a wide range of research areas goes back even further to 2012. In that year, a deep Convolutional Neural Network (CNN) outperformed all previous attempts in an image recognition task [10]. Moreover, DL in general as an approach to mimic the human brain has a very long history, starting with McCulloch and Pitts in 1943 [11]. The open source availability of DL libraries such as TENSORFLOW [12] and PYTORCH [13] is driving an ever-accelerating development in research areas far beyond data and computer science.

ML refers to algorithms that can improve at a given task with experience, i.e. more data. Common tasks are regression or classification, for which the algorithm has to learn to fit functions such that a given input produces the desired output or class label. The field of ML is divided into different paradigms. Supervised learning is distinguished from unsupervised learning. Supervision in this context refers to whether or not target values are provided during training. The boundary between supervised and unsupervised methods is fuzzy. The region in between the two is often referred to as semi- or weakly supervised. There are other paradigms, such as reinforcement learning, but they are beyond the scope of this thesis.

Within this thesis we do not cover the question if large language models like GPT (Generative Pre-trained Transformer) [14] are actually intelligent, or what this even means. We are interested in how we can make use of the ability of DL to analyze huge amounts of data and find patterns to advance our understanding of fundamental physics. A review on the interplay of the physical sciences and ML, including various applications, is given in reference [15]. Albeit it is from 2019, and thus to some extent outdated given the rapid development of new methods, it nicely illustrates the broad applicability and the interest of the various communities in ML.

Classification is generally a prime example for supervised learning, where a model is trained to separate different kinds of data instances into classes, based on known instance-class pairs used for training. We will employ supervised methods for the classification of astrophysical $\gamma$-ray sources that are part of the Fermi Large Area Telescope (Fermi-LAT) source catalog. These sources can be categorized according to the objects that emit the radiation. The most common sources are Active Galactic Nuclei (AGNs) and PulSaRs (PSRs), which can be further subclassified. The Fermi-LAT as part of the Fermi satellite has been launched into orbit in 2008. It detects an increasing number of new $\gamma$-ray sources that need to be categorized. The unambiguous classification of these objects is costly, since multiwavelength observations are required. This definite classification cannot keep up with the rate of new sources that are detected by Fermi-LAT. Thus, a ML based, first classification can provide interesting candidates and support analyses that require the $\gamma$-ray source
population. ML has already been applied since the first version of the Fermi-LAT catalog with the goal to provide candidates for sources that are likely to belong to some class [16].

By now, the Fermi-LAT collaboration has released the third data release of its fourth catalog with twelve years of data taking and more than six thousand detected γ-ray point sources [17]. As the amount of data increases, there is more and more interest in automated classification schemes and finding candidates of so far unknown source classes [18, 19]. With the larger amount of data, the transition from mostly tree based ML algorithms to neural networks starts to show promising results [20–22].

Compared to the γ-ray source data of Fermi-LAT, High Energy particle Physics (HEP) provides enormous amounts of data. The Large Hadron Collider (LHC) is not only at the energy frontier, but also at the frontier of data production. At the CMS experiment alone, proton bunches cross with a frequency of 40 MHz, producing up to 1 billion proton-proton interactions per second [23]. Each of these produces some (sparse) signature in the more than 100 million detector readout channels. The amount of produced data is around one petabyte per second. Storing all of this collision data is neither feasible nor desirable. Most of it is thrown away during the trigger process, which is tuned to keep events that might be interesting for follow-up analyses. These interesting events are collected in order to test our current understanding of the fundamental interactions of matter, i.e. the Standard Model of particle physics (SM), and discover potential Beyond the Standard Model (BSM) signatures.

ML has played a role in the analysis of this data for a long time, often under the name of multivariate analysis. A famous example is the discovery of the Higgs boson in 2012 [24, 25], where ML was used for example to identify b-jets. While the application of ML in physics already has a long history, there has been a large increase of interest since 2012. An iNSPIRE HEP search request provided by reference [26] shows an exponential increase of ML and DL papers, reaching more than 2500 papers in 2022 [27]. Moreover, there is a community effort to collect ML related papers within the field of HEP in the living review of machine learning for particle physics [28].

Since the discovery of the Higgs boson, many BSM theories have been proposed and tested. So far, no compelling evidence for any specific BSM model has been found. Therefore, there is a need to boost signal-model agnostic searches at the LHC. DL provides the tools to handle the huge amounts of data and additionally the very specific needs for new physics searches.

In this thesis, we focus on jets, which are one of the dominant signatures at the LHC. A jet consists of a collimated spray of particles. Such a spray of particles is produced, when a color-charged parton from the hard interaction showers, i.e. radiates additional quarks and gluons, which finally hadronize. The resulting particles can then decay or be stable on time scales sufficient to travel to the detector where they can be measured. Depending on the initial parton, the signatures in the detec-
tor can vary. One application for which DL has already shown superior performance is the supervised tagging of jet flavor, i.e. determining the initial parton given the detector signature [29]. However, such a tagger has to be trained on simulated data, because no labeled data is available at the LHC.

The amount of data and the unprecedented precision to which physicists are able to simulate this data provide a marvelous playground for DL. Also, the fields of weakly supervised and unsupervised methods benefit from the large amount of data available. These paradigms are applicable directly on experimental data, even without labels. Hence, they open new avenues for analyzing LHC data and potential for discoveries of BSM physics. We investigate the application of the weakly supervised Classification Without Labels (CWoLa) [30] method. This method relies only on the separation of the data into a control and a signal region, and it is thus directly applicable to experimental data. This separation is often available, and we will use the recent ATLAS mono-jet search [31] as an example.

In addition, we use unsupervised learning to perform density estimation in the high-dimensional space of jet constituents. Having a reliable density estimator, the densities can be used for classification according to the likelihood ratio, the reweighting of simulated events, or the data driven generation of events [32]. Additionally, the density can in principle be used for anomaly detection [33]. However, the curse of dimensionality makes density estimation in many dimensions difficult, because in high-dimensional spaces, all datasets are sparse. We tackle this difficulty in an autoregressive setup, similar to reference [32], but using the transformer architecture introduced in reference [9].

In addition to this general introduction, more detailed introductions to the respective topics are given at the beginning of each part. This thesis is structured as follows: In part II, we briefly introduce DL and provide the tools that are used in the other parts of this thesis. We then descent the ladder of supervision, starting with supervised classification of $\gamma$-ray sources in part III. We investigate the use of fully connected and recurrent neural networks using the photon flux measurements as input in section 4. In section 5, we include uncertainties to the classification by using Bayesian neural networks. Weak supervision and finally unsupervised learning applied to HEP jet data are the topics of part IV. We adapt the CWoLa method to boost the sensitivity of a search with signal and control region in section 6. In section 7, we build an analogy between jets and natural language and apply a transformer for unsupervised density estimation. We finally give our conclusions and closing thoughts in part V.
Part II.
The toolbox
In this part, we want to give a brief introduction to DL as a subfield of ML and introduce the concepts and methods we use throughout this work. Parts of this introduction are based on reference [34] and we highly recommend this reference for a more in depth introduction into the broad field of DL. For a more fundamental and theoretical introduction to ML and basics of DL, we refer the reader to reference [35]. A hands-on introduction into the basics of ML and DL and their implementation with TENSORFLOW [12] can be found in reference [36]. Moreover, for an introduction to DL with focus on recent applications in physics research, we recommend reference [37].

ML refers to algorithms that improve on a given task with experience, i.e. the algorithm becomes better the more data is provided to it. Basic ML algorithms, such as logistic regression or decision trees, rely heavily on the representation of data to be chosen according to the task at hand. The key idea behind DL is to let the algorithm learn the representation best suited for the task itself. DL methods do so by learning a hierarchy of representations, building complex representations from simpler ones, until a good representation of the data for the task at hand is found. It is called deep, because many of these hierarchical feature extraction steps can be stacked. Another interpretation is that DL is a sequential execution of computations and deep referring to a long sequence. The usual example is the decomposition of images by first learning edges, then corners and contours, continuing with object parts and finally full objects. The example of images is a nice visualization of the concept, making it easy for humans to understand, but the same holds for any other data format.

In the realm of physics in general and the field of HEP in particular, experts have spend decades to develop good features. The notion of good however depends on the particular task and thus varies. The description of a collimated spray of particle as a jet, motivated by the common origin of these particles (e.g. a showering parton) is one example. There are many jet algorithms to separate a full event into individual jets, each with its own benefits and drawbacks, see e.g. references [38,39]. Moreover, to describe properties of a jet, physicists have developed further representations, describing substructure by a smaller list of features compared to the full list of constituent four momenta. We refer to reference [40] for a more in depth review of jet substructure. Famous examples are the N-subjettiness [41] giving a measure how well a jet is described by N subjets, or energy flow polynomials [42], which can be shown to form a complete infrared safe basis for jet observables. Lots of theoretical and experimental expert knowledge goes into the construction of these features. However, it has been shown that DL can outperform these human crafted features, starting from the low-level information of a jet’s constituents, e.g. in the task of tagging boosted top jets [29].

The way a DL algorithm processes data, in terms of the sequence of operations it applies, is referred to as the model architecture. The no free lunch theorem states that no one optimization algorithm is better than any other if their performance is
averaged across all possible problems [43]. However, we are usually not interested to find the one optimization algorithm for every possible task there is. We are concerned with a tiny subset of these tasks, e.g. the classification of jets. For such a niche of tasks, the inductive bias introduced by a certain model architecture can make a large difference in performance, see e.g. the comparison of top taggers in reference [29].

In the following, we briefly introduce the main concepts behind DL in section 1. We present the different architectures that we employ within this thesis in section 2. Finally, we introduce relevant performance measures in section 3.
1. Deep learning, a shallow example

A DL model is a parameterized function acting on some input. As such, it can be written as \( f_\Theta(x) \), where \( \Theta \) denotes the adjustable parameters of the function, and \( x \) denotes the input(s) to the model. The procedure of training consists in the optimization of the parameters such that the function maps inputs close to some target \( t \). In all applications within this thesis, the training can be formulated as minimization of some cost or loss function. Usually, as the number of parameters is large, this optimization cannot be performed analytically. Instead an iterative approach based on the gradients of the loss function with respect to the model parameters called gradient descent is used.

We briefly introduce this procedure and necessary ingredients using the simple example of a fully connected network. A sketch of such a network is shown in figure 1.1. A simple Neural Network (NN) consists of nodes (gray circles) that are collected in layers. Nodes within one layer are not interconnected. The connections between layers are realized via trainable weights, i.e. \( w_i \in \Theta \). Layers for which all nodes are connected to all nodes in the preceding layer are called fully connected or dense. A model constructed from this kind of layer by adding several layers of arbitrary sizes is referred to as Dense Neural Network (DNN) or fully connected network in this thesis, but is also often called a Multi Layer Perceptron (MLP).

Each node in the model is associated to some calculation. While the input nodes only pass the data into subsequent layers, the hidden and output nodes perform weighted sums of their inputs according to the learned parameters. In the example architecture of figure 1.1, the values of the hidden node vector are obtained via matrix multiplication of the input vector with the weight matrix \( w_1 \in \mathbb{R}^{3 \times 5} \). Stacking such linear transformations results itself in a linear transformation. To gain expressiveness, a nonlinearity referred to as activation function is introduced. The calculation performed by a node can thus be written as a function of the outputs of the previous layer \( x' \) and the weight connections \( w \) as

\[
n(x', w) = \sigma(x'^T w + b) ,
\]

where \( b \in \Theta \) is a trainable parameter called bias, which we neglected in the discussion before, and \( \sigma \) is the activation function. Stacking layers, including potentially many nodes, increases the number of trainable parameters the model has and thus what functions can be approximated by \( f_\Theta \). For fully connected layers and certain nonlinearities, e.g. the Rectified Linear Unit (ReLU), a network with one single hidden layer can be shown to be able to approximate any function to arbitrary precision, in the limit of infinitely many nodes \([44, 45]\). While this universal approximation theorem holds theoretically, it is in practice easier to train deeper networks with smaller layers, achieving better performance than using a single hidden layer with many nodes. This is in line with the hierarchic feature extraction mentioned in the introduction to part II, which is only possible when stacking multiple layers.

Finally, to optimize the parameters of the network, we need some optimization
goal. Usually, the goal is the minimization of a cost or loss function over a training dataset. This function depends strongly on the task and is commented on for each task individually in this thesis. Here, we briefly mention two common loss functions: Mean Squared Error (MSE) and Cross Entropy (CE). MSE is mainly applied to regression tasks and CE to classification. For an output of the network $y$ and the given target values $t$, they are given for a single data instance by

$$CE(t, y) = \sum_c t_c \ln(y_c)$$

(1.2)

$$MSE(t, y) = \frac{\sum_i (t_i - y_i)^2}{\sum_i 1}.$$  

(1.3)

Applying a network to classification, it is common to set the number of output nodes to the number of classes in the data. The target vector for single class targets is then given by a one-hot encoding, i.e. $t$ is zero in all except one dimension, the dimension corresponding to the target class.

To summarize, a NN can be seen as a parameterized function $f_\Theta$ that is optimized to map some input $x$ to desired target values $t$ in order to minimize some loss function. This is the simple, but fundamental idea behind DL. The way to perform this optimization and what kind of nonlinearities are commonly used is discussed in the following.
1. Deep learning, a shallow example

1.1. Activation, optimization and regularization

As discussed above, the activation function at each layer is necessary for the network to be able to learn nonlinear functions. The most common activation function is the ReLU function, due to its simplicity. ReLU returns the identity for positive inputs and zero else, see equation 1.4. This function and its gradients are simple to calculate, as the gradients are zero for negative and one for positive values. One problem that can arise for this activation is the occurrence of dead weights, i.e. weights that are never updated, because the gradient is always zero. Slight modifications to ReLU are leaky ReLU (equation 1.5), which introduces a (small) constant slope to negative inputs, and Parameterized ReLU (PReLU), which introduces this slope for negative values as trainable parameter, see equation 1.6. We found that ReLU generally performs well. However, in special applications there are significant performance improvements for activation functions with gradients for negative inputs. One such case is the use of jet images as input. These are defined such that the input is always positive [46], leading to potentially many dead nodes when using plain ReLU. We saw in the application of AutoEncoders (AEs) on jet images, the benefit of using PReLU [3], in line with reference [47]. Another application where the choice of activation has had significant impact is discussed in section 6.

An additional activation function that is frequently used is softmax, see equation 1.7. In contrast to the ReLU variants, softmax is not applied to individual nodes, but takes into account all nodes of a layer. The output of this function is positive and normalized to sum one and is therefore particularly interesting when thinking about probabilities. The softmax activation is thus often used in the output layer of classification networks to allow for a probabilistic interpretation. Every output node then corresponds to the probability of the input to belong to the class associated to that node.

The functional forms of the mentioned activation functions are

\[
\text{ReLU} = \max(0, x) \tag{1.4}
\]

\[
\text{leakyReLU} = \begin{cases} 
ax, & \text{if } x \leq 0 , \text{ a fixed} \\
x, & \text{else}
\end{cases} \tag{1.5}
\]

\[
\text{PReLU} = \begin{cases} 
\alpha x, & \text{if } x \leq 0 , \alpha \text{ trainable} \\
x, & \text{else}
\end{cases} \tag{1.6}
\]

\[
\text{softmax} = \frac{\exp(x_i)}{\sum_i \exp(x_i)} . \tag{1.7}
\]

There is a plethora of additional activation functions, see e.g. reference [48] for a recent review. However, in this work, we mainly use the ones introduced here.

Having all the ingredients for our model, we need to introduce a procedure to optimize the weights in order to minimize the loss. As already mentioned, the potentially huge number of parameters and large number of layers prohibit the analytic
1.1. Activation, optimization and regularization

solution of the loss minimization. Moreover, finding the global minimum of the loss with respect to some training data might not be the ultimate goal. If the model has sufficient capacity, the global minimum of the loss will be a perfect fit to the training data, which is most likely not generalizing well to new data. This phenomenon of remembering the training set is called overfitting.

The way DL models are typically trained is called gradient descent. The weights are updated following the gradient of the loss function with respect to the model parameters. Usually one update is calculated not on the full training data, but on a small fraction of it. This small chunk of data is called a batch or mini batch. The gradient calculated over a single batch estimates the gradient of the full dataset. However, due to comparatively small batch sizes, this estimate introduces stochasticity into the updates, which has been shown to improve generalization [49]. Usually, one needs to find a balance between small batches that generalize well, but require many updates for convergence and long runtimes, and larger batches, which result in more stable gradient updates and allow speedup through parallel data processing.

While the basic principle behind the gradient descent method is simple, there are several different optimizers, i.e. different ways to use the gradient to update the weights. The most basic algorithm is called Stochastic Gradient Descent (SGD). Within this algorithm the weight update is described by

$$\Theta(t + 1) = \Theta(t) - \eta \nabla_\Theta L,$$  \hspace{1cm} (1.8)

where $t$ is the iteration step and $\eta$ is called the learning rate, determining the step size for a single update. The loss is denoted as $L$. The notion stochastic arises from randomly sampled mini batches. Another optimizer that is based on SGD, but adapts the learning rate for individual parameters is Adam [50]. The Adam optimizer adapts the learning rate for individual parameters by using moving averages of the first and second moments of the gradients. The moving averages are updated according to

$$m_n(t + 1) = \beta_n m_n(t) + (1 - \beta_n)(\nabla_\Theta L)^n,$$  \hspace{1cm} (1.9)

where $m_n$ denotes the $n$-th moment. The hyperparameters $\beta_n$ modify the importance of the current gradient within the update of the moving mean. The update rule for the weights is then given by

$$\Theta(t + 1) = \Theta(t) - \eta \frac{m_1(t)}{\sqrt{m_2(t)} + \epsilon},$$  \hspace{1cm} (1.10)

with a global learning rate $\eta$ and a small constant $\epsilon$ to prevent division by zero. The default values for these parameters suggested in reference [50] are $\eta = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. They are generally a good starting point for further optimization. There are plenty more methods that we do not discuss here. Some of them also take higher order derivatives into account. We refer the reader to references [34,51] for more details and a comparison.
We mentioned already the problem of overfitting and bad generalization for accumulating the gradient over the full training set. Using mini batches reduces how prone a model is to overfitting, as the stochasticity introduces some regularization. However, if the model has enough capacity, it can still overfit the training data. To prevent this from happening one introduces additional regularization to reduce the capacity of a model and to favor certain solutions over others. One way to perform this regularization is to add a term to the loss function. Common regularizations are $l_1$ and $l_2$ regularization. These regularizations add the term $\lambda \sum_i |\Theta_i|^p$ to the loss function with $p = 1, 2$ for $l_1, l_2$ regularization, respectively. The strength of the regularization can be controlled with the hyperparameter $\lambda$. The model is thus pushed towards small weights. While $l_2$ regularization favors overall smaller weights, $l_1$ regularization can be used to create sparse models, i.e. pushing weights to zero.

Another regularization method that we use in this work is dropout [52]. Dropout regularization randomly sets a certain fraction of nodes to zero during training. This random selection changes for every batch. The network is thus forced to learn redundant information to be able to compensate these missing connections. It can be seen as an effective way of learning an ensemble of models in one training run, instead of training individual networks. The fraction of weights that are set to zero is an additional hyperparameter. Moreover, dropout can be applied not only to all layers, but also just to a subset of layers, or with different fractions for each layer. To reduce the number of hyperparameters introduced by using dropout, one often uses the same fraction for all layers.

To conclude this section, we want to highlight that the advances of DL applications in physics are only possible because there are free open source libraries providing the necessary ingredients described above. Additionally, they provide the necessary hardware optimization for the straightforward application of these methods on Graphical Processing Units (GPUs), giving significant speedup. The libraries used in this work are TENSORFLOW [12] and PYTORCH [13]. In particular, they implement the backpropagation algorithm, short backprop [53]. This algorithm allows for the efficient calculation of gradients that are needed for updating the model parameters.

We have discussed how a DNN, such as the one displayed in figure 1.1, is trained and what considerations go into setting up and training a model. A DNN already allows for some variation in how to connect layers, e.g. in section 4 we use multiple input layers and connect these inputs after some individual processing. In the following section, we introduce various additional architectures and building blocks that are used throughout this thesis.
Figure 2.1: Unrolling of a recurrent layer without outputs. The function $f$ updates
the cell state $h$ taking into account the cell self itself and the current input. The
scheme is inspired by figure 10.2 (page 366) of reference [34].

2. Architectures

So far, we have only discussed fully connected layers. There are other ways of con-
necting nodes, ordering parameters in layers and repeating blocks of layers. Some of
them are discussed in this section. The detailed architectures are discussed in the re-
spective sections, where they are applied. This includes the way layers are combined
to a full model and how hyperparameters introduced in this section are set.

2.1. Recurrent layers

We start with recurrent layers [53], which have been designed to work with sequential
data. Typically, they are introduced in the field of time series data, but the sequence
does not need to be in a timely context. Nevertheless, the notion of a time step
for a step forward in the sequence has become established and is also used here.
Recurrent layers scale well to different sequence lengths, due to their weight sharing.
This is similar to convolutional layers [54] that are well known for image processing.
Recurrent layers apply a recurrent cell to each time step of the input sequence. The
cell consist of a cell state $h$ and a function $f$ processing new inputs sequentially,
taking into account the cell state. The exact design of how the input is processed
and the cell state is updated differs between various recurrent layer implementations,
e.g. Long Short Term Memory (LSTM) [55] or Gated Recurrent Unit (GRU) [56],
to name the most prominent ones. When a data instance is given as input to the
recurrent layer, the cell receives the first time step of the sequence, updates its
cell state and continues with the subsequent time step, using the same $f$ for the
processing step as before. The cell state is what makes recurrent layers different to
one dimensional convolutions with kernel size one, as the output of the current step
becomes a function not only of this step in the input, but all previous steps as well.

The most simple cell design consists of a fully connected network processing the
input and using the output of that network as cell state. The flow of information is
shown schematically in figure 2.1. The output of such a recurrent layer can either
be a sequence of same length as the input or a single vector. Sequence to sequence
modeling is obtained by forwarding the cell state $h^t$ at every time step $t$. In contrast,
2. Architectures

Figure 2.2: Schematic view of an undercomplete AE: the input \(i\) is encoded to some reduced latent representation \(z\) and then decoded to the output \(o\). Encoder and decoder are usually composed of NNs.

the single output vector is generated by using only the final cell state after the last time step. Returning the full sequence allows to stack several recurrent layers to form a Recurrent Neural Network (RNN). Finally, the output of the last recurrent layer in a RNN can be the input to a fully connected network that performs e.g. classification.

RNNs have been applied in physics for example to supernova classification \([57]\), denoising of gravitational waves \([58]\), jet classification \([59]\), jet density estimation \([32]\), or track reconstruction at the LHC \([60]\). In section 4, we employ a RNN with the simple cell design described above, making use of returned sequences to stack recurrent layers and connecting the final hidden state of the last recurrent layer to a fully connected classifier.

2.2. Autoencoders

An AutoEncoder (AE) \([61]\) is designed to compress and decompress data. The basic architecture is depicted in figure 2.2. It consists of an encoder and a decoder which are commonly implemented as NNs. The encoder processes the input and projects it to a latent space. The decoder reconstructs the initial input from this hidden representation. During training, the weights of both sub-networks are trained at the same time to minimize the difference between input and output. To prevent the network from learning the identity mapping, the architecture needs to be constrained. A common choice is to make the latent dimension smaller than the input, which is referred to as an undercomplete AE. Alternatively, by introducing an additional sparsity constraint (\(l_1\) regularization), the latent dimension can be chosen to be larger. Such an AE is then denoted as overcomplete. Within this thesis, we consider undercomplete AEs, for which the latent space works as an information bottleneck. The choice of the latent dimension is problem dependent. One usually aims for
the intrinsic dimension of the input data, such that the encoding can be maximally efficient.

While most applications construct the AE to be symmetric around the bottleneck dimension, i.e. the decoder often has as many layers and mirrors the dimensionality of the encoder in these layers, this does not need to be the case. If the decoder is a linear mapping and the measure of similarity between input and reconstruction is the MSE, the AE latent space spans the same space as Principle Component Analysis (PCA). An AE with a nonlinear decoder can thus be seen as a generalization of PCA to nonlinear correlations.

The AE is trained in an unsupervised way, as only the input and no additional labels are needed. It is thus a common tool when the data and the task at hand do not provide labels. Typical applications of AE in physics are feature extraction [62,63], denoising [64,65] or anomaly detection [47,66]. By enforcing a particular distribution of the data in latent space, e.g. a normal distribution, the decoder can be used to generate data [67]. These Variational AutoEncoders (VAEs) have been used in HEP, e.g. in references [68–70] for anomaly detection or in reference [71] for jet simulation. We use the AE architecture for unsupervised feature extraction in section 4.

2.3. Dynamic graph convolutional neural networks

We present the Dynamic Graph Convolutional Neural Network (DGCNN) introduced in reference [72] in this section. We apply it in the context of particle physics as classifier for jets. The adaption of the DGCNN to jet classification has been performed under the name ParticleNet in reference [73].

The main building blocks of these architectures are edge convolutions (EdgeConv) operating on point clouds. A point cloud is a collection of points that do not follow any particular ordering. Thus, the architecture is designed such that it is invariant under permutations of the input. Each point is characterized by some features. In the original application to computer vision, these features are the point coordinates in three dimensional, Euclidean space. Switching to jet physics, the points correspond to individual jet constituents, and the features correspond to measured particle properties. These properties are the position in the detector, but also derived quantities from their energy and momentum, see the corresponding sections 6 and 7 for more details on the used features.

The first step in the EdgeConv consists of generating a $k$ Nearest Neighbors (KNN) graph from the points and their coordinates. Note that the coordinates used for the graph construction can differ from the particle features. In particular, they can be a subset of the features. The graph convolution is then similar to the usual convolution on images, except that the local patch seen by the filter is defined over these KNN neighborhoods. The convolution is performed over every point in the point cloud. It is parameterized by a fully connected neural network $h_{\Theta}$, taking as input the features of the central point and one of its neighbors. For each neighbor, this edge function $h_{\Theta}$ constructs so called edge features. We end up with $k$ such edge feature vectors that
2. Architectures

Figure 2.3: Sketch of the EdgeConv operation for one central node $x_i$. Left shows the connections of this central node to its nearest neighbors. The edge function $h_{\Theta}(x_i, x_j)$ is applied to construct the edge features $e_{i,j}$ which are aggregated to give the updated version of node $i$, $x'_i$. This figure is an adaptation of figure 2 in reference [72].

are aggregated by a permutation invariant function $\Omega$ to get updated node features $x'$ for the next layer. The process can be expressed by

$$x'_i = \Omega^k_{j=1} h_{\Theta}(x_i, x_j). \quad (2.1)$$

We sketch this operation in figure 2.3. Note that the function $h_{\Theta}$ does not need to have the same input and output dimension, thus the dimension of $x'$ and $x$ can be different. The aggregation function $\Omega$ needs to be invariant under permutations of its input to maintain permutational invariance of the EdgeConv. We adopt the choice of reference [73] to set $\Omega$ to be the mean. The same kernel function $h_{\Theta}$ is used for all points $x_i$ of the cloud and its neighbors $x_j$. It is thus the analogue to the convolutional kernel in a CNN. In reference [72], several alternatives to directly using $x_i$ and $x_j$ as input to the edge function $h_{\Theta}$ are proposed. We adopt the modification of the inputs to $x_i$ and $x_i - x_j$. This choice incorporates local structure by the second argument while keeping the global structure by using the features of the central point $x_i$.

Since the edge function $h_{\Theta}$ is implemented as a fully connected network, one can in general adapt the number of layers and the number of nodes within each layer. However, we fix the number of layers to three, as in references [72, 73], and fix the number of nodes to be the same for all three layers. The number of nodes is then varied only between different EdgeConv blocks. Batch normalization [74] is applied after every fully connected layer.

The EdgeConv operation updates every point in the point cloud and produces another cloud as output, in which the number of features describing a point is potentially different. Therefore, one can stack EdgeConv blocks. The dynamic part of the DGCNN enters, when the next EdgeConv generates a new KNN graph using the output of the previous layer as coordinates. The DGCNN can learn to put mean-
2.4. Transformer-encoders

The transformer model has been introduced in reference [9] for machine translation. It has replaced convolutional and recurrent models as state of the art for Natural Language Processing (NLP). The key ingredient is the use of an attention mechanism [76]. The proposed architecture consists of an encoder-decoder structure. However, we only need the encoder part within this work and thus focus on the introduction of the corresponding components.

The transformer is designed to work on natural language. As such, the inputs are discrete. To allow the network to learn similarities between words, the discrete space is embedded into a continuous one. We use a simple lookup table as embedding. It consists of a matrix that has as many rows as there are words (or more generically categories) and a given number of columns. The number of columns accounts for the embedding dimension. When an input is given to the embedding, the row corresponding to the given word or category is selected and passed to the subsequent part of the transformer. The values within the lookup table are updated just as the other weights of the transformer model, using gradient descent.

Within the original transformer, the embedding is passed to a positional encoding. Without this positional encoding, the transformer does not have an explicit notion of proximity of two entries in a sentence. In our application, we do not need this positional encoding and the embedding is directly fed to the transformer-encoder block. We show the processing steps within one transformer-encoder layer in figure 2.4. A
2. Architectures

Figure 2.4: Sketch of the transformer-encoder layer together with the embedding. The sketch is adapted from figure 1 in reference [9], removing the transformer-decoder structure and moving the layer normalization before the multi-head attention and the feedforward (fully connected) network.

transformer-encoder block is constructed by repeating this layer structure multiple times.

As a first step, a layer normalization (LayerNorm) [77] is applied. LayerNorm calculates the mean \( \mu \) and variance \( \sigma^2 \) of the activation \( a_i \) over all nodes of the current layer. The normalized activation is given by

\[
a'_i = \frac{a_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \times \gamma + \beta ,
\]

where \( \gamma \) and \( \beta \) are trainable parameters and \( \epsilon \) is a small number (10\(^{-5}\)) added for numerical stability. It is important to note, that the same \( \mu \) and \( \sigma \) are used for all nodes of the layer for one instance. The normalization parameters change however for different data instances. This is in contrast to batch normalization [74], which applies different \( \mu \) and \( \sigma \) to every node, but the same for different data instances. In the original setup of reference [9], the LayerNorm is applied after the multi-head attention and also after the fully connected layer (see below). We choose to apply it before both these layers, because it has been shown that this simplifies training in reference [78].

The normalized activations are passed to a multi-head attention layer. Attention mechanisms use a query and key-value pairs. Queries and keys are used to calculate weights, which are in turn used to calculate the weighted sum of the values. There are different ways to calculate these attention weights. In reference [9], the authors
2.4. Transformer-encoders

introduce the scaled dot-product attention

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V, \quad (2.3)$$

where $Q \in \mathbb{R}^{n \times d_k}, K \in \mathbb{R}^{n \times d_k}, V \in \mathbb{R}^{n \times d_v}$ denote the query, key and value matrices of a sequence with length $n$, respectively. The scaling factor $d_k$ corresponds to the dimension of keys and queries. Values can have a different dimension $d_v$. The scaling is performed since the variance of the softmax function arguments becomes larger for higher dimensions. The scaling then prevents ending up in regions with too small gradients, arising from the flat behavior of the softmax function at large absolute values.

In their setup, the authors of reference [9] apply multi-head attention, which can be written as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O,$$

with

$$\text{head}_i = \text{Attention}(QW^Q_i, KW^K_i, VW^V_i). \quad (2.4)$$

In this equation, $h$ refers to the number of heads. The matrices denoted by $W^{Q/K/V}$ are trainable projections, mapping the keys, queries and values to different representations per head. The trainable output projection $W_O$ projects the concatenation to the desired output dimension, which is set to be the dimension of the input in reference [9]. This allows the model to pay attention to different correlations in the input. The output of the individual heads is concatenated. If $d_m$ is the original dimension of keys, queries and values, the projection matrices reduce the dimension of each head to $d_k = d_m/h$, i.e. $W^Q_i, W^K_i \in \mathbb{R}^{d_m \times d_k}, W^V_i \in \mathbb{R}^{d_m \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_m}$. Projecting the individual heads to the reduced dimension makes the computational costs similar to that of single-head attention with the full model dimension.

The setup is applicable to variable lengths in the input, i.e. the transformer can work with sentences of varying length. During training, it is more efficient to pad all sequences within a batch to the same length. Ignoring these padded values is particularly easy for the attention mechanism. The attention weight assigned from each value to any of the padded values is set to zero. In practice, the entries of the matrix $QK^T \in \mathbb{R}^{n \times n}$ corresponding to padded values are set to negative infinity using a padding mask. The weight is then set to zero by applying the softmax function, resulting in a normalized output for valid points in the sequence. In addition to the padding mask, any value of the attention matrix can be set to zero in the same way. We will use this property to introduce a causal mask in section 7. This mask sets the attention between any word in the sentence to all following words to zero. In the original setup, this causal mask is used in the decoder during training. It disables the translating transformer from seeing the full translated sentence. Thus, it enforces an autoregressive prediction of the next word of the translation, i.e. the next word can only depend on previous words of the translated sentence. As mentioned above, the transformer-encoder layer does not have an explicit notion of position in
3. Performance measures

<table>
<thead>
<tr>
<th></th>
<th>True value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>True positives</td>
</tr>
<tr>
<td>Negative</td>
<td>False negatives</td>
</tr>
</tbody>
</table>

Table 3.1: Confusion matrix for binary classification. The entries can either contain the absolute number of instances falling into any of the four categories or be normalized to sum one per row, column or overall.

the sequence. Nevertheless, the causal mask allows the network to learn positions implicitly, since information on the number of seen entries is available. This results in the transformer-encoder with causal mask learning positions if they are relevant [79].

Since we only use the encoder structure of the full transformer architecture, we only need to consider self-attention. In self-attention, the key, query and value matrices are the same. In particular, it holds that $d_k = d_v = d_m/h$.

The output of the multi-head attention is combined with its input to preserve the original information. Since the architecture is designed such that the dimension of the values before and after attention is $d_m$, they are combined simply by addition, denoted by the Add layer in figure 2.4.

The sum of the attention output and the initial input is fed through an additional LayerNorm. The normalized output is given to a fully connected network with two layers, which is applied to every element of the sequence. The first layer has some hidden dimension $d_h$ and the second layer projects back to the model dimension $d_m$. Again, a residual connection is implemented by adding the input of the LayerNorm to the output of the fully connected network. This final output of the transformer-encoder layer can then be given to another transformer-encoder layer. Alternatively, it can be used as representation for an additional network performing some task. In the original transformer setup, the transformer-encoder layer is repeated several times before the representation is given to the decoder.

In section 7, we build an analogy between particles forming jets according to some ordering and words making up sentences according to grammar. We apply another fully connected layer element wise to the transformer-encoder output, to map to the dimension of our particle dictionary. Together with the causal mask introduced above, we can then build a model that predicts the next step of a sequence given all previous elements. We employ this setup for density estimation.
3. Performance measures

The performance evaluation of a NN is strongly dependent on the task. Within this thesis we mainly focus on binary classification tasks. Calling one of the two classes positives, performance measures can be denoted by the number of false positives (FP), true positives (TP), false negatives (FN) and true negatives (TN). Their definition from predicted and true label can be read off the confusion matrix in Table 3.1. Normalizing the confusion matrix such that all entries sum to one gives the corresponding rates, i.e. TPR, FPR, TNR, FNR.

A first metric that is often used is the accuracy. It is given by \((\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})\), i.e. the number of correctly classified instances divided by the total number of instances. Having unbalanced datasets, the accuracy is dominated by the class that includes the majority of instances. As an example, we consider a dataset consisting to 90% of one class. A classifier that always predicts this class obtains an accuracy of 90%, even though it does not provide any useful information. Thus, such data needs different metrics. We can define precision \(p = \text{TPR}/(\text{TPR} + \text{FPR})\) and recall \(r = \text{TPR}/(\text{TPR} + \text{FNR})\). Precision shows the fraction of positive predictions that is correct and recall the fraction of positive instances that is correctly classified. The F1 score is the harmonic mean between precision and recall

\[
F1 = \frac{2pr}{p + r}.
\]  

(3.1)

It lies between zero and one, with one being the optimal score, and is often used instead of the accuracy in imbalanced datasets. While the accuracy does not depend on which class we call positive, the F1 score does. In imbalanced tasks, it is common to call the minority class positive, enhancing the sensitivity to performance in that class.

So far, we have introduced performance metrics that use hard class assignments. The prediction of the classifier sorts each instance to one of the two classes. In practice, the classifiers that we use predict some score which can be interpreted as a probability to belong to the positive or negative class. The assignment to one class can be generically set by adjusting the threshold. One commonly used class assignment for classification is using the class with highest probability. This boils down to a threshold at 0.5 for binary classification. However, we can adapt this threshold, trading precision for recall and vice versa. To obtain threshold independent metrics, we can scan through all possible thresholds and draw the Receiver Operating Characteristic (ROC) curve. This curve shows the TPR as a function of the FPR. The Area Under the ROC Curve (AUC) is bound between zero and one. An example ROC curve is shown in the left panel of figure 3.1. A perfect classifier results in an AUC of one, a random classifier in an AUC of 0.5 and a classifier that is perfectly inverting the truth gives an AUC of zero. The AUC can thus be used as a threshold independent score. We also show the inverse ROC curve, i.e. the ROC curve when switching the notion of positive and negative. The two curves give the same AUC, because the separation of the two classes remains the same. However, the regions
3. Performance measures

![Example ROC curves](image)

Figure 3.1: Example ROC curves in the representation used to calculate the AUC (left) and the representation typically used in jet classification (right). The gray dashed line indicates a random classification, i.e. no separation power within the classifier score. The shaded area in the left plot is stated as the AUC of the black curve. The red curves correspond to evaluating the ROC curve for inverted labels, i.e. we swap signal and background. The same score distribution is used for the two panels and both ROC curves.

contributing to the AUC change.

In physics applications, we usually associate the positive class with some signal and the negative class with the corresponding background. We replace the notion of FPR with background efficiency $\epsilon_B$ and TPR with signal efficiency $\epsilon_S$. Moreover, we use an alternative representation of the ROC curve if we expect to have a large background and little signal. We present the background rejection $1/\epsilon_B$ as a function of $\epsilon_S$ as ROC curve, using a logarithmic scale for the background rejection. We show this representation of the previous example in the right panel of figure 3.1. Common scores in benchmarking jet classifiers are obtained for background rejections at fixed signal efficiencies or signal efficiencies at fixed background rejections. In this representation, the difference between the original ROC curve and the inverted becomes more prominent. Even though the AUC as summary measure is the same, the performance at relevant thresholds (high background rejection) is very different. If the signal contains instances that are perfectly distinguishable from the background this shows up as infinite background rejection at finite signal efficiency (black curve). Swapping the labels, we obtain background events that are perfectly separable. This leads to a steep (vertical) drop of background rejection at signal efficiency of one (red curve). The evaluation of the full ROC curve and a look at the score distribution is thus often much more informative than just the AUC.

Finally, one can generally evaluate the networks according to the loss function they are trained to minimize. This is particularly true if the loss has some likelihood interpretation. However, the best loss achievable is often very problem dependent and a given quantitative improvement in the loss is often difficult to interpret. For
a given task, the loss function is usually used for hyperparameter optimization.

We use the python package scikit-LEARN [80] as an efficient tool to calculate the metrics presented in this section, in particular the ROC curves and the F1 score.
Part III.

Supervised learning for $\gamma$-ray classification
Supervised learning for $\gamma$-ray classification

The key idea of applying DL methods to low-level observables close to the actual measurement can be applied to many fields in physics. As a first example, we will employ supervised classifiers to classify astrophysical $\gamma$-ray sources. While experiments like the Fermi $\gamma$-ray space telescope detect an ever increasing number of $\gamma$-ray point sources, the definite categorization of those cannot keep up. Providing a fast, DL based, first classification can thus be beneficial for population studies that in turn are for example relevant for better understanding of the $\gamma$-ray background or the blazar sequence [81]. This can help constrain $\gamma$ ray signatures, e.g. from Dark Matter (DM) annihilation [82].

The Fermi $\gamma$-ray space telescope with its primary instrument, the Large Area Telescope (LAT), detects $\gamma$-rays in the energy range from a few MeV to one TeV [83]. The Fermi-LAT observes the full sky. More than five thousand point-like sources have been discovered within our galaxy and beyond. The data is collected in source catalogs, which are made publicly available. At the time of this project, the 4th Fermi-LAT $\gamma$-ray catalog, data release 2 (4FGL-DR2) [84] based on ten years of data taking was the latest version.\footnote{By the time of writing the 4FGL-DR3 catalog has been released, including twelve years of data taking and 6659 detected sources [17].} It includes 5788 detected objects. The same analysis method introduced in the first release of the fourth catalog [85] is used.

The numbers of sources for the various different object classes are shown in figure III.1.

Most objects (3508) in the 4FGL-DR2 catalog are AGNs. These objects consist of a supermassive black hole at the center of a galaxy. They emit jets of radiation through the accretion of matter, covering a broad range of frequencies, reaching into $\gamma$-ray energies. Most of these AGNs (98\%) are blazars, i.e. their emitted jet points in the direction of our line of sight. Those can be further subdivided into Flat Spectrum Radio Quasars (FSRQs) and BL Lacerae objects (BLLs), named after the first object detected of this class. They differ in their spectral properties [86]. About one third of blazers is not yet further subclassified, i.e. they are denoted as Blazars Classified as Uncertain type (BCUs). We refer the reader to reference [87] for an overview of AGNs.

The second largest class of classified sources (259) consists of PSRs, which can be further subdivided into MilliSecond Pulsars (MSPs) and YouNG pulsars (YNPs). PSRs are rotating neutron stars that are highly magnetized and surrounded by a plasma magnetosphere. MSP are believed to be old pulsars, which went through the accretion of a companion object, reducing their rotational period [88].

AGNs and PSRs differ in their variability. While PSRs are non-variable on long time scales, AGNs show varying patterns in time scales of months. Additionally, fluxes of AGNs as a function of the energy (energy spectra) generally follow a softer power law compared to PSRs. PSRs also have additional curvature in their spectra, on top of the harder power law [89].

Finally, there is a small fraction of objects that are neither AGNs nor PSRs. We
Supervised learning for $\gamma$-ray classification

Figure III.1: Visualization of source counts per class in the 4FGL-DR2 catalog. The violet solid boxes correspond to the labels provided by the catalog. The dashed boxes show the classification tasks considered in this work. The classification of the cyan boxes on the right is presented in section 4. The subclassification of BCUs shown in orange and yellow is part of section 5.

Sources of the 4FGL-DR2 catalog can be identified as or associated to a certain object class. Identification depends on correlated timing information at various wavelengths. Association is performed by observing positional coincidence with a source at other wavelengths. About one third of sources in the Fermi-LAT catalog (1794) is neither associated nor identified as belonging to a specific class of objects. While the catalog grows, this fraction remains similar [90]. In the following, we refer to these unidentified and unassociated sources combined as UNClassified (UNC) sources. Looking at the spectral properties of these sources and their positions shows that they should be composed of various classes [85].

Datasets in astronomy are rapidly growing both in size and in complexity, because of improved detector sensitivity and longer data taking. ML in general and DL methods in particular thus become more and more interesting for analyzing these datasets. Within the Fermi-LAT catalogs, ML has been applied to provide source type candidates for previously unclassified sources, both for existing classes like AGN and PSR [91–93] and new classes like DM subhalos [18, 19]. Moreover, the classification of BCUs with neural networks has been performed in references [20–22]. Prior to our studies, providing candidate classes for the UNC sources has been based mainly on handcrafted features extracted from photon fluxes as function of energy or time.
Supervised learning for $\gamma$-ray classification

The performance then strongly depends on the choice of features. In reference [92], a method for automatic feature selection was employed. This improved the classification performance over previous feature selections by experts in the field. In HEP, classification tasks have seen the superior performance of DL methods directly accessing low-level information, e.g. constituent level information when classifying jets [29]. For the classification of Fermi-LAT sources, this corresponds to using flux measurements as function of energy or time directly as inputs to our classifier instead of hand crafted features. We refer to them as energy spectra and time spectra in the following. A first application for the classification of BCUs using a NN on low-level information is demonstrated in reference [22].

It is our goal to evaluate how well DL methods perform on the task of $\gamma$-ray source classification in this part. We use NNs to predict the class of UNC sources, as well as BCUs, and provide source candidates for the different classes. The final, conclusive classification of a source requires multiwavelengths observation and potentially extensive optical spectroscopy, see e.g. references [94,95]. This procedure is time consuming and costly. In particular, since the number of sources detected by Fermi-LAT increases faster than follow-up observations can deliver classifications. For example, the fraction of BCUs rose from $\sim$13\% in the first Fermi-LAT catalog [96] to 40\% of the total AGN population in the 4FGL-DR2 catalog [84].

We first investigate the classification of UNC sources into AGNs and PSRs, and the subsequent classification of PSRs into MSPs and YNGs using a fully connected and a recurrent neural network in section 4. This path is highlighted in cyan in figure III.1.

The expensive follow-up studies needed for a final source classification make a reliable estimate of the classification uncertainty desirable. One way to incorporate a notion of uncertainty to the prediction of a classifier is the use of Bayesian Neural Networks (BNNs) [97, 98]. The accessible uncertainty estimate of a BNN gives a handle on the interpretation of the network performance. BNNs have been employed in astroparticle physics for example in the context of supernova classification in reference [99] and the Galactic center excess in reference [100]. More broadly they have also been used in HEP for jet classification [101], jet calibration [102] and event generation [103, 104]. We apply a BNN to the classification of BCUs in section 5, which is depicted in yellow and orange in figure III.1. We perform an extensive evaluation of the uncertainty estimate and compare it to an ensemble estimate using multiple DNNs. Finally, we provide candidate sources for FSRQs and BLLs out of the BCUs from the 4FGL-DR2 catalog.

The classification results of both tasks can be used directly for follow-up studies and to guide follow-up observations. On the one hand, interesting sources for follow-up observations can be the candidates provided to verify the classification. On the other hand, the unambiguous categorization of sources that cannot be classified with the applied methods provides the most information for future classifiers. Moreover, these sources may belong to yet unknown classes [18, 19].
4. Classification of *Fermi*-LAT sources with deep learning using energy and time spectra

In this section, we look at the cyan path in figure III.1, i.e. the binary classification of UNC sources into AGNs or PSRs and the subsequent classification of PSRs into YNGs and MSPs. The content and results of this section closely follow reference [2]. We discuss the dataset and our preprocessing in section 4.1. We introduce the classifier architectures used in section 4.2. Results are presented in section 4.3. Additionally, we introduce an unsupervised method for feature selection in section 4.4. We finish with a summary of our findings in section 4.5.

4.1. The data: AGN and PSR in the 4FGL-DR2 catalog

In this section, we use AGNs and PSRs included in the 4FGL-DR2 catalog [84]. Within this catalog each source is described by 74 features. These features contain the position and measured \( \gamma \)-ray flux values, as well as handcrafted features constructed from the measurements.

The flux is measured in seven energy bands, ranging from 50 MeV to 300 GeV (50-100 MeV, 100-300 MeV, 300-1000 MeV, 1-3 GeV, 3-10 GeV, 10-30 GeV, 30-300 GeV). The catalog also contains the integrated flux in ten yearly bins. In addition to the flux values, the catalog contains the significance of the flux coming from a point source rather than the diffuse background for each energy band as well as for each yearly bin. This significance is given by the test statistic \( TS = 2 \log(\mathcal{L}/\mathcal{L}_0) \), where \( \mathcal{L} \) and \( \mathcal{L}_0 \) are the likelihood functions with and without the corresponding source included in the flux model, respectively. Prior to the second data release, the 4FGL catalog contained the flux history in two month intervals. These are no longer provided as yearly bins were shown to contain most information on the variability [85].

When using the time and energy spectra, we thus have \( 7 \times 2 \) energy band features (flux and significance for each energy band) plus \( 10 \times 2 \) time features per source (flux and significance for each yearly bin).

In the PSR class, we select those sources from the catalog that are identified as or associated with pulsars. In the catalog, identified sources are reported with upper case letters in the feature CLASS1. Associated sources are referred to by lower case letters. Thus, we select sources that fulfill \( CLASS1 \in \{ \text{PSR, psr} \} \). The subclassification of PSRs into MSPs and YNGs is taken from the publicly available list of LAT-detected \( \gamma \)-ray pulsars [105], after cross-matching this list to the PSRs from the 4FGL-DR2 catalog.

AGNs have multiple subclasses and we select all of these under the label AGN. Again, we include identified as well as associated sources. Overall, we select those sources for which \( CLASS1 \in \{ \text{FSRQ, fsrq, BLL, bll, BCU, bcu, CSS, css, RDG, rdg, NLSY1, nlsy1, agn, ssrq, sey} \} \). We refer the reader to references [84,85] for more information on the definitions of AGN subclasses. Finally, the UNC sources
are reported with an empty CLASS1 feature.

The resulting numbers of sources falling into the individual classes are shown in figure III.1. Within this section we consider the rightmost branch following the UNC sources and their subsequent classification. The classification of BCUs is discussed in section 5.

Not all features in the catalog are available for all sources. When working with handcrafted features, this needs to be taken into account. One way to do so is to remove those sources for which necessary features are not available, thus reducing the training statistics. We further describe a procedure for dealing with missing values in section 4.4. The time and energy spectra however are available for all sources. Therefore, we do not discard any of the classified objects and the numbers of figure III.1 correspond to the amount of sources at hand for training and validation of our networks.

We make use of the 3FGL catalog [90] to cross-match our classification performance between catalogs. In the 3FGL catalog there are 1745 AGNs and 167 PSRs, compared to 3508 and 259 respective sources in the 4FGL-DR2 catalog. Of the 1010 unclassified sources in the 3FGL catalog 294 are classified as AGN (258) or PSR (36) in the newer 4FGL-DR2 catalog. The data format in the 3FGL is slightly different. Only five energy bands are provided. Additionally, the flux history is given in 48 monthly bins, as opposed to the yearly bins, and no significance is stated. When working with 3FGL sources, we describe each source by \(5 \times 2\) features for the energy bands and 48 features for the time spectra.

We preprocess the time and energy spectra by taking their logarithm and standardizing them. Standardizing here means subtracting the mean and dividing by the standard deviation for each feature, i.e. each bin of the energy and time spectra individually.

4.2. Classification methods

We employ two kinds of deep neural networks in this section. A baseline fully connected neural network to which we refer as DNN and a RNN. The networks presented in this section are implemented using TENSORFLOW 2.1.0 [12] with the built-in version of KERAS [106]. We introduce the setup of the DNN in section 4.2.1 and the RNN in section 4.2.2.

4.2.1. Fully connected neural network

A DNN corresponds to the basic architecture described in section 1. Here, we construct the architecture such that it takes energy and timing information as separate inputs. Both inputs are flattened, i.e. the flux information and the significances are stacked into a vector for both timing and energy input. The two inputs are separately fed through two fully connected networks with ReLU activation. The outputs
of these two networks are concatenated and fed to a final fully connected classification network with two output nodes, corresponding to the two classes. We apply \texttt{softmax} activation to the final layer to obtain a probability interpretation of a source belonging to either one of the two classes.

The architecture is flexible in the sense of number of layers and nodes, but also allows for adding more features in a third branch or replacing one of the branches for the spectra. We make use of the flexibility to evaluate the influence of varying inputs.

We perform hyperparameter optimization, tuning the number of layers per branch as well as the nodes within and the number of layers and nodes after merging the branches. Since the dataset is comparably small, we use ten-fold cross-validation for this optimization. The data is divided into ten folds such that the class proportions are the same in all sets, using the \texttt{StratifiedKFold} function of \textsc{scikit-learn} \cite{scikit-learn}. We then train a network on nine folds and leave one for validation. This is repeated ten times, such that each fold is taken for validation once. That way, we get a prediction for each source in the training set from a network that was not trained on the given source. We do this for all networks in a random parameter scan and choose the parameters from the best performing model in terms of the F1 score (equation 3.1). In the calculation of the F1 score, we set PSR to be the positive class, as they comprise the minority of sources.

Additionally, we tune the parameters controlling the training, i.e. the initial learning rate, the number of epochs trained and the batch size. We fix the optimizer to be the Adam optimizer \cite{Kingma2014}, as we found it to give better results than plain SGD.

Our search suggests to use one layer with 16 nodes for each of the input branches. After concatenation, the input to the final classification network is thus of dimension 32. The classification network consists of one hidden layer with four nodes and the output layer. We apply \texttt{ReLU} activation in all layers except the last, where \texttt{softmax} is applied. For the training procedure, we find good results when training with a learning rate of 0.01 and a batch size of 64 for 50 epochs.

### 4.2.2. Recurrent neural network

RNNs are designed to work on sequential data. Thus, they are perfectly suited to be applied to the energy and time spectra of gamma ray sources. Their working principles are explained in section 2.1.

As for the DNN, we treat time and energy spectra in two separate branches. Using the same hyperparameter optimization procedure as for the DNN, we find best performance for three recurrent layers with eight nodes each for both branches. Concatenating the two branches gives a 16 feature input to a final fully connected part of the network, which has the same structure as for the DNN, i.e. one hidden layer with four nodes and a final output layer with two nodes. Using a bidirectional RNN, i.e. feeding the sequence forwards and backwards, does not show improvements. Also, using LSTM layers or GRUs does not change performance. Thus, we stay with the basic RNN implementation, corresponding to the \texttt{SimpleRNN} layer in \textsc{Tensorflow}. 

31
Within the recurrent layers \texttt{tanh} activation is used, the hidden fully connected layer applies \texttt{ReLU} activation. The classification output is obtained using \texttt{softmax} activation.

The training is performed using the Adam optimizer [50] with an initial learning rate of 0.001 and a batch size of 32 for 50 epochs.

The key difference between RNN and DNN is the processing of the input. The RNN processes the input strictly sequential, while the DNN processes the full spectra at once.

4.3. Results

As proposed in reference [91], we perform sequential binary classification. First, we separate between AGNs and PSRs, the two main categories in the 4FGL-DR2 catalog. Second, we subdivide PSRs into MSPs and YNGs. To estimate our performance on the UNC sources, we test on labeled data within the 4FGL-DR2 catalog. Moreover, we perform a cross-match between the 3FGL and the 4FGL-DR2 catalogs. We train on labeled sources of the older catalog and test on sources that had not been classified in the 3FGL catalog, but have obtained class associations in the 4FGL-DR2 catalog. The cross-match allows to quantify effects from the selection bias, i.e. the bias that brighter sources are collected and classified more easily and thus UNC are dominated by fainter sources.

In a binary classification task, where the output of our network consists of two nodes which sum to one, the usual assignment to a class is done using a threshold of 0.5. That means that a data instance is assigned to a given class if the corresponding node has a value larger than 0.5, i.e. is the larger of the two outputs. As described in section 3, one can vary this threshold to obtain the ROC curve. Moreover, the best threshold might not be at 0.5, depending on the data as well as the relevant metric to optimize. Whenever we state a metric that assumes a certain classification threshold, e.g. accuracy and F1 score, we set the threshold such that it maximizes the accuracy in the training set. Thus, this threshold varies for different training runs and data selections.

We start by evaluating our NNs on the labeled sources of the 4FGL-DR2 in section 4.3.1. We perform a cross-match between the 3FGL and the 4FGL-DR2 catalogs in section 4.3.2 to evaluate the classifier behavior on newly classified sources. Finally, we apply our classifiers to UNC sources in section 4.3.3.

4.3.1. Performance on labeled sources of the 4FGL-DR2 catalog

We split the sources into a training and a test set, by selecting 30% of the data for testing. The split is performed such that the class proportions in test and training set are the same. The results stated in this section correspond to mean and standard deviation of ten runs. For each run, the network weights are newly initialized and the train-test-split is performed anew. Therefore, the uncertainties reported in this
4.3. Results

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Accuracy [%]</th>
<th>F1 score [%]</th>
<th>AUC [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>97.59 ± 0.35</td>
<td>81.73 ± 3.43</td>
<td>97.99 ± 0.75</td>
</tr>
<tr>
<td>VAR</td>
<td>97.90 ± 0.44</td>
<td>84.09 ± 3.27</td>
<td>98.34 ± 0.69</td>
</tr>
<tr>
<td>BAND</td>
<td>97.70 ± 0.32</td>
<td>82.30 ± 2.49</td>
<td>98.23 ± 0.52</td>
</tr>
<tr>
<td>HIST</td>
<td>95.45 ± 0.34</td>
<td>64.03 ± 3.92</td>
<td>92.88 ± 1.09</td>
</tr>
<tr>
<td>RNN</td>
<td>97.31 ± 0.36</td>
<td>79.27 ± 2.82</td>
<td>97.40 ± 0.89</td>
</tr>
<tr>
<td>VAR</td>
<td>97.49 ± 0.30</td>
<td>80.76 ± 2.40</td>
<td>97.75 ± 0.81</td>
</tr>
<tr>
<td>BAND</td>
<td>97.51 ± 0.28</td>
<td>80.80 ± 2.17</td>
<td>96.83 ± 1.07</td>
</tr>
<tr>
<td>HIST</td>
<td>94.43 ± 0.72</td>
<td>50.83 ± 8.42</td>
<td>89.26 ± 2.77</td>
</tr>
</tbody>
</table>

Table 4.1: Classification results for AGN vs. PSR classification for DNN and RNN. The architectures with VAR use the variability index instead of the time spectra. BAND refers to using only the energy spectra and HIST to using only the time spectra.

section take into account the variability of the training and test sets, as well as the variability of the network due to its initialization. We refer to section 5 for a more thorough interpretation of uncertainties.

As described in section 4.2, we can easily swap any of the spectra with some of the handcrafted features. We investigate how the networks perform, when using the variability index instead of the time spectra (denoted as VAR in table 4.1), when not using any timing information at all, i.e. only using the energy bands (BAND) and when only using the time spectra (HIST). The variability index summarizes the source variability inherent in the time spectra to a single number.

We show relevant performance measures for DNN and RNN with these different inputs in table 4.1. Again, we set PSR to be the positive class in the calculation of the F1 score according to equation 3.1. First of all, we notice a strong decrease in performance when only using the time spectra (HIST). Moreover, when removing the timing information and only training on the energy spectra, the performance and stability increases slightly, compared to the benchmark using both energy and time spectra. Best performance is achieved when replacing the time spectra by the variability index (VAR).

This can be explained by considering the smallness of the dataset. For little data, a classifier is susceptible to noise in the inputs. Noise here refers to features that do not contain relevant information for the classification task at hand. The timing adds only little information to the task that can be captured by the variability index. Using the full time spectra however adds 20 features, carrying little information per feature. This degrades the performance. With sufficient data, the networks should learn to extract the relevant features themselves and ignore noisy inputs. The benchmark performance is thus expected to increase with additional data in upcoming catalogs.

Furthermore, we see similar trends in the RNN and the DNN. The overall performance of the DNN is slightly better than the RNN. A possible explanation is that
4. Classification of Fermi-LAT sources with DL using energy and time spectra

<table>
<thead>
<tr>
<th></th>
<th>Network</th>
<th>True AGN</th>
<th>TruePSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted AGN</td>
<td>DNN</td>
<td>$1042.1 \pm 3.7$</td>
<td>$16.4 \pm 5.4$</td>
</tr>
<tr>
<td></td>
<td>RNN</td>
<td>$1042.3 \pm 4.3$</td>
<td>$19.6 \pm 3.3$</td>
</tr>
<tr>
<td>Predicted PSR</td>
<td>DNN</td>
<td>$10.9 \pm 3.7$</td>
<td>$61.6 \pm 5.4$</td>
</tr>
<tr>
<td></td>
<td>RNN</td>
<td>$10.7 \pm 4.3$</td>
<td>$58.4 \pm 3.3$</td>
</tr>
</tbody>
</table>

Table 4.2: Confusion matrix for AGN vs. PSR classification with DNN and RNN. Central values correspond to the mean over ten train-test splits and network initializations, stated uncertainties give the standard deviation of these ten runs.

the RNN can only process the inputs sequentially, which makes extraction of certain correlations more difficult.

Another way to compare the networks is to look at the confusion matrix. The confusion matrices for the two benchmark architectures are shown in table 4.2. While the numbers are very similar for DNN and RNN, this does not imply that the classified sources need to be the same. This is particularly true, as the sources in the test set vary between runs and do not need to be the same for DNN and RNN at all. A more detailed discussion of the overlap in DNN and RNN decision making is part of section 4.3.3. Dividing the numbers in the confusion matrix by the sum of the columns gives us intra-class accuracies. Both DNN and RNN classify almost all ($\sim 99\%$) of the AGNs correctly. For PSRs this rate is only at $78\%$ and $74\%$ for DNN and RNN, respectively. While the accuracy for PSRs of the RNN is smaller, it fluctuates less.

The authors of reference [92] state a higher overall accuracy of $99.19 \pm 0.16\%$, using a random forest classifier combined with an automatic feature selection of the handcrafted features within the catalog. They compare different methods and worse performance is achieved for example for logistic regression ($94.72 \pm 0.3\%$) and support vector machines ($95.77 \pm 0.31$) combined with feature selection. They need to neglect some sources, since the catalog does not have the corresponding features. Alternatively, they fill these missing values e.g. by the mean of those sources in the catalog which have a value assigned to that feature (see section 4.4).

A comparison beyond a single threshold of DNN and RNN in the standard setup, i.e. with both energy and time spectra, can be made by investigating the ROC curves (see section 3) in the left panel of figure 4.1. We see that they agree within their fluctuations, which are given by the worst and best performance of the ten runs. However, the DNN performs slightly better.

As one can see in figure III.1 and table 4.2, we deal with a very unbalanced dataset. Such an imbalance can lead to bad performance if the network focuses only on the more populated class. We can see that this is to some extent the case in our application from the different intra-class accuracies. This easily happens, as
4.3. Results

Figure 4.1: ROC curves for AGN vs. PSR classification (left) and MSP vs. YNG classification (right). Presented are the mean ROC curves of the DNN as solid black line and the RNN as dashed red line. The shaded area corresponds to the envelope of performances obtained in different runs.

the gradients to better classify the majority class are obtained more often during training. In other words, the overall loss is reduced more when performing better on the majority class. Being better on the majority compensates for somewhat worse performance on the minority class. Standard ways to deal with imbalances are weighting, downsampling or data augmentation.

Weighting refers to a class dependent factor in the loss. This factor is usually chosen such that the contribution of the minority class to the loss becomes the same. That means, the minority class is weighted by \( w_{\text{min}} = \frac{N_{\text{maj}}}{N_{\text{min}}} \), where \( N_{\text{maj}} \) and \( N_{\text{min}} \) give the number of instances in the majority and minority class respectively. For the classification of AGN vs. PSR, this would result in a weight factor of \( \frac{3508}{259} \approx 13 \). One interpretation of this weight factor is an increased learning rate for the minority class. The factor of \( \sim 13 \) would thus increase the gradient step by an order of magnitude for PSRs, which can lead to large oscillations around loss minima and instabilities during training.

In the alternative approach of downsampling, the majority class is reduced to a similar size as the minority class by simply choosing a subset of instances. This approach can be used, when sufficient training statistics is available. Because of the already limited amount of data available in our case, we want to keep as much of it as possible. Therefore, downsampling is not an option for us.

The last alternative mentioned here is data augmentation and refers to creating additional data points of the minority class. There are different ways of doing so. The easiest is plain repetition, i.e. one uses each instance or a subset of the minority class as often as necessary to balance the two classes. In our case, this would again correspond to the factor of \( \sim 13 \). The advantage over the weight factor is that within a batch one has more variation in the minority class and not just a constant factor in front of the gradient update. Moreover, one can combine the repetition of instances with adding noise. As we deal with measurements that have
some uncertainty, each instance we observe might just as well be observed slightly differently, according to some measurement uncertainty. This uncertainty should not change the class label and thus an instance with some small noise added can contain additional information. A more advanced method is called Synthetic Minority Oversampling TEchnique (SMOTE) [107]. SMOTE can be summarized by a simple formula: $S_{\text{aug}} = S_1 + r(S_2 - S_1)$. The augmented instance $S_{\text{aug}}$ is calculated to be at some random distance $r \in (0, 1)$ between two randomly chosen instances of the minority class $S_1$ and $S_2$. To get an augmented source, we draw two PSR instances randomly and $r$ from a uniform distribution in the interval $(0, 1)$.

We find that all of these methods hardly affect the performance. They do change the performance metrics obtained at the generic threshold of 0.5. However, since we adapt the threshold according to the performance on the training set, this is not a relevant threshold for us. The performance metrics at the determined threshold do not change. Moreover, the ROC curves, containing information of all thresholds, do not differ significantly using these augmentation methods. We thus decide not to use data augmentation in our standard setup.

Next, we investigate the further subclassification of PSRs into YNGs and MSPs. The dataset for this classification task is substantially smaller. We have only 230 labeled sources. Luckily, these are more balanced than AGNs and PSRs. Namely, we have 127 YNG and 103 MSP sources. We use 161 sources for training and the remaining for testing, again performing multiple train-test splits and network initializations. We found that reducing the batch size reduces training stability. With so little data and the batch size as before, we need more epochs to obtain sufficient gradient updates. Therefore, we increase the number of epochs to 150 (compared to the 50 used before).

The ROC curves for DNN and RNN using time and energy spectra with significance information on this task are shown in the right panel of figure 4.1. The overall performance is worse than AGN vs. PSR classification, as one could expect already from the smaller dataset and the closer relation between the two source classes. Also, the fluctuations are larger and the curves are less smooth, due to the limited statistics available for testing. The limited statistics is also visible in the jump at zero, where $\epsilon_{\text{MSP}} \approx 0.6$ as soon as $\epsilon_{\text{YNG}} > 0$, i.e. $\sim 60\%$ of MSPs are included as soon as a threshold including any YNG is chosen. For more statistics, this should become a smoother transition. Nevertheless, the DNN shows some advantage, especially in the stability of its performance. The uncertainty band is smaller over most of the $\epsilon_{\text{YNG}}$ range.

We show performance metrics for different architectures respectively different features in table 4.3. In contrast to the AGN vs. PSR classification, we consider the Galactic latitude (GLAT) as additional discriminating feature. This feature is also directly measured and not inferred from the spectra. It was shown to have discriminating power for MSP vs. YNG classification in reference [91]. We refer to training with time and energy spectra plus the Galactic latitude by GLAT in table 4.3. Additionally, we again remove the time spectra information and add the Galactic latitude,
4.3. Results

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Accuracy [%]</th>
<th>F1 score [%]</th>
<th>AUC [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>83.91 ± 3.69</td>
<td>84.47 ± 3.50</td>
<td>90.25 ± 2.65</td>
</tr>
<tr>
<td>GLAT</td>
<td>87.25 ± 3.82</td>
<td>88.00 ± 3.66</td>
<td>93.34 ± 2.09</td>
</tr>
<tr>
<td>BAND GLAT</td>
<td>87.10 ± 3.86</td>
<td>87.82 ± 3.63</td>
<td>92.84 ± 3.41</td>
</tr>
<tr>
<td>RNN</td>
<td>82.03 ± 5.15</td>
<td>82.69 ± 4.50</td>
<td>90.03 ± 4.11</td>
</tr>
<tr>
<td>GLAT</td>
<td>86.52 ± 2.83</td>
<td>87.46 ± 2.75</td>
<td>93.04 ± 2.32</td>
</tr>
<tr>
<td>BAND GLAT</td>
<td>85.51 ± 3.37</td>
<td>85.72 ± 3.90</td>
<td>92.61 ± 2.83</td>
</tr>
</tbody>
</table>

Table 4.3: Same as table 4.1, but for MSP vs. YNG classification. The architectures with GLAT use the Galactic latitude as additional feature, while BAND GLAT refers to removing the time spectra and adding the Galactic latitude.

to which we refer as BAND GLAT.

One can see a significant performance increase for both DNN and RNN when adding the Galactic latitude. For the DNN, the accuracy increases from almost 84% to slightly above 87%. Removing the time spectra does not show notable variation in performance, considering the overall large fluctuations. The evaluation on such small sets is notoriously difficult. Large fluctuations are expected, since testing only considers 69 sources. The wrong classification of one source already makes a difference of 1.4% in accuracy. As the train-test split is repeated ten times, there are splits in which more or less hard to distinguish sources end up in the test set.

The performance stated in reference [92], using feature selection and boosted logistic regression, is at an accuracy of 89.61 ± 2.34%. Again, they compare different methods and the random forest classifier that performs best on classifying AGNs against PSRs has an accuracy of 86.30 ± 2.14%. The worst performance is obtained using logistic regression (without boosting) with an accuracy of 83.56 ± 1.82. They again only state the accuracy and no alternative performance measures.

4.3.2. Cross-match between 3FGL and 4FGL-DR2 catalogs

In this part, we investigate how well our approach works between different catalog generations. The longer the data taking period of the Fermi-LAT, the more statistics is available, also for fainter sources. Moreover, the same is true for other experiments, allowing for association or identification of previously unclassified sources. Comparing the 3FGL and the 4FGL-DR2 catalog, a total of 294 UNC sources of the 3FGL are now labeled as AGN or PSR. The 294 sources split into 258 AGNs and 36 PSRs, i.e. the fraction of AGNs in this set of newly classified sources is at 87.8% compared to the 93.1% within the 4FGL-DR2 catalog. These newly classified sources comprise our test set in the cross-match.

This cross-match test allows for an evaluation of the effect the sample-selection bias can have. This effect, also referred to as Malmquist bias or covariate shift in astronomy, describes the fact that brighter sources are detected first [108]. The
4. Classification of Fermi-LAT sources with DL using energy and time spectra

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Measure</th>
<th>70% Training set</th>
<th>100% Training set</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>Acc [%]</td>
<td>94.63 ± 0.42</td>
<td>94.42 ± 0.44</td>
</tr>
<tr>
<td></td>
<td>AUC [%]</td>
<td>92.69 ± 2.31</td>
<td>92.96 ± 1.48</td>
</tr>
<tr>
<td>RNN</td>
<td>Acc [%]</td>
<td>94.05 ± 0.67</td>
<td>94.73 ± 0.46</td>
</tr>
<tr>
<td></td>
<td>AUC [%]</td>
<td>94.84 ± 1.76</td>
<td>95.13 ± 0.77</td>
</tr>
</tbody>
</table>

Table 4.4: Classification results for AGN vs. PSR classification for DNN and RNN for the cross-match between the 3FGL and 4FGL-DR2 catalogs. We show the results for training on 70% and 100% of the labeled sources within the 3FGL for training and testing on the newly labeled sources in the 4FGL-DR2 catalog.

The overall performance is lower compared to the results obtained for the 4FGL-DR2 catalog in table 4.1, i.e. an accuracy of more than 97%. First, the training data is a factor of ~0.73 smaller. We saw however that this might be negligible for this amount of data. Second, the fraction of PSRs in the cross-match test set is slightly
4.3. Results

<table>
<thead>
<tr>
<th>Architecture</th>
<th>PSR (acc [%])</th>
<th>AGN (acc [%])</th>
<th>YNG (acc [%])</th>
<th>MSP (acc [%])</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>78</td>
<td>1050</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>(92.75 ± 3.25)</td>
<td>(98.91 ± 0.32)</td>
<td>(93.70 ± 5.20)</td>
<td>(86.14 ± 4.79)</td>
</tr>
<tr>
<td>DNN</td>
<td>20</td>
<td>22</td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td>GLAT</td>
<td>(92.19 ± 5.26)</td>
<td>(89.49 ± 4.49)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNN</td>
<td>21</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAND GLAT</td>
<td>(91.45 ± 5.66)</td>
<td>(90.17 ± 4.93)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNN</td>
<td>57</td>
<td>1076</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(94.14 ± 2.64)</td>
<td>(98.86 ± 0.27)</td>
<td>(97.94 ± 2.73)</td>
<td>(–)</td>
</tr>
<tr>
<td>RNN</td>
<td>2</td>
<td>12</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>GLAT</td>
<td>(97.07 ± 2.79)</td>
<td>(95.80 ± 5.81)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNN</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAND GLAT</td>
<td>(97.68 ± 4.61)</td>
<td>(93.55 ± 6.71)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5: Classified UNC sources using time and energy spectra (DNN, RNN), adding the Galactic latitude (GLAT) and using only the energy spectrum in combination with the Galactic latitude (BAND GLAT). The accuracies are estimated applying the threshold of 0.9 to the labeled test sets within the 4FGL-DR2 catalog. They thus might not correspond to the true accuracy within the UNC sources.

higher. Since the networks are in general better in identifying AGNs, due to the larger amount of available sources in the training set, this translates into lower accuracy. However, this effect is not sufficient to explain the gap in performance. So, we are left with a discrepancy that might be explained by the selection bias. We confirm the effect, when testing the networks trained on 70% on the remaining 30% of the 3FGL. Here, we find an accuracy of 96.99 ± 0.50% for the DNN, ~2% higher than on the cross-matched sources.

Our results in terms of the overall accuracy are again in line with the findings in reference [92]. They also do not observe a particularly strong performance increase when training on more sources. They achieve an accuracy of 94.1 ± 1.04% and 95.6 ± 0.77% using the boosted logistic regression on 70% and 100% of the 3FGL sources, respectively.

4.3.3. Predictions for unclassified sources of the 4FGL-DR2 catalog

The previous sections provide validation of our method to reliably predict the class of a source, given the time and energy information. In this section, we finally apply the method to UNC sources of the 4FGL-DR2 catalog to provide candidates for further analyses. For that purpose, we perform first a classification of UNC sources into AGNs and PSRs. We then further classify the PSR candidates into MSPs and YNGs. For those sources that we provide as candidates, we want to be rather certain. This
4. Classification of Fermi-LAT sources with DL using energy and time spectra

Figure 4.2: Galactic latitude distribution for AGNs (left) and PSRs (right) of the 4FGL-DR2 catalog. The distribution of labeled as well as our predicted candidate sources is shown. On the left, we also show the distribution for UNC sources (dotted).

holds especially since some of the UNC sources might be neither AGNs nor PSRs, but some other type of object. Thus, we call a UNC source a PSR candidate if the average predicted PSR probability of the ten networks trained in section 4.3.1 is above 0.9. The same threshold is applied to the AGN probability. While this does not ensure that provided candidates might be of another class, it should reduce the risk.

We show the numbers of candidates obtained for the different source classes by our NNs in table 4.5. For the first classification, we use the standard setup, i.e. DNN and RNN with time and energy spectra including significances as input. For the PSR subclassification, we additionally present the numbers for using the Galactic latitude as supplementary input (GLAT) and for using only the energy information together with the Galactic latitude (BAND GLAT). We see that adding the Galactic latitude helps mainly to increase the number of MSPs with high probabilities. Additionally, we show an accuracy that is estimated by applying the 0.9 threshold test set of the labeled sources within the 4FGL-DR2 catalog. The uncertainty of this estimate is again given by ten training runs with different initialization and splitting of test and training set.

The DNN leaves 551 out of 1679 UNC sources unclassified, i.e. does not assign them to AGNs or PSRs with a score higher than 0.9. The left panel of figure 4.2 shows that most of these remaining unclassified sources lie within the Galactic plane. Within that plane precise measurements are difficult, due to the strong Galactic background emissions. More than half of the 551 sources have low point source significances of 5 to 10. Additionally, they are marked in the catalog with some analysis flags. These flags are assigned if the systematic uncertainties have a strong dependence on the modeling of the diffuse $\gamma$-ray background. Alternatively, the source to background ratio is low or there can be source confusion. For more details regarding the analysis flags, see reference [85]. Similar results are obtained by the RNN.
4.3. Results

We further validate that our source candidates have reasonable feature distributions in figure 4.3. We consider the logarithm of the variability index and the square root of the curvature significance for the log parabola spectral fit (LP SignCurv). The variability index (also used in section 4.3.1) summarizes the flux variation over time in a single number. The LP SignCurv indicates in how far the energy spectrum is a curved spectrum, rather than a pure power law. Those features are known to have discriminating power for the classification of AGNs against PSRs, as PSRs have low variability and show significant curvature, while AGNs do not exhibit the curvature and are more variable. The figure shows AGN and PSR candidates from the DNN as points and the corresponding classified sources of the 4FGL-DR2 catalog as contours from a kernel density estimate. Even though the networks are not trained on the features, but the energy and time spectra, the source candidates follow the distributions of those features nicely. This shows that the networks indeed find relevant information directly from the spectra that is also interpretable. Note that the selected features fulfill the DNN threshold of 0.9. Thus, it is expected to have strong separation in at least some features. The features in figure 4.3 are specifically chosen to have strong separation of the two classes. Nevertheless, even for the selected features there is some overlap in the selected sources, hinting towards more complex correlations learned.

The subclassification of PSRs into YNGs and MSPs shows, once more, the difficulties of the small dataset. For the RNN in the standard setup, there are no MSP candidates. The separation of the two classes is not strong enough to pass the threshold of 0.9. The RNN already finds fewer PSR candidates, increasing the difficulty of limited test statistics. Nevertheless, the classifications obtained with DNN and RNN
4. Classification of Fermi-LAT sources with DL using energy and time spectra

are consistent. Approximately 91% of sources identified by the DNN as AGN or PSR are equally classified by the RNN. This percentage is limited by the fewer PSR candidates found by the RNN and can only reach up to 98%. While this is true for the chosen threshold of 0.9, we verified that the RNN also finds all DNN candidates when applying a threshold of 0.5 to its predictions. Setting the RNN threshold to 0.8 (0.7) increases the fraction of DNN candidates that are also found by the RNN to 97.7% (99.2%).

We find that out of the 21 MSPs classified by the DNN standard setup, 16 are also included in the candidates obtained by including the Galactic latitude. In total, the three DNNs with different inputs find 29 MSP candidates. For the RNN, the eight MSP candidates found using only the energy spectrum and the Galactic latitude are fully contained in the twelve candidates of the RNN using the time spectrum in addition (RNN GLAT). Moreover, eleven of those are also included in the 29 candidates from the DNN classifiers. The YNG candidates found by the RNN are all contained in the DNN candidates.

Last, we want to compare our candidate list to those in the literature. Out of the twelve MSP candidates found by the RNN, seven are also found by reference [91]. They used the 3FGL catalog and the sources that we find in addition to them are reported with low significance, i.e. close to the detection threshold, in that catalog. Only three of our twelve sources are also found in the list of 20 MSP candidates provided by reference [92]. They use the random forest classifier and apply a softer threshold of 0.5 for the AGN vs. PSR classification and a harder threshold of 0.98 for the YNG vs. MSP classification. To get a final answer on the association of the candidates with MSPs, YNGs or even a completely different class, multiwavelengths observations of these sources are required. A full list of our predictions using DNN and RNN on UNC sources of the 4FGL-DR2 catalog is publicly available at reference [109].

4.4. Using an autoencoder for feature selection

We discussed in section 4.3.2 that the classification performance can be affected by the sample selection bias inherent to astrophysical data. In this section, we investigate a method of feature extraction that can take into account all sources, including the unlabeled ones. Constructing features that are meaningful for fainter sources as well can reduce the bias. Feature selection also helps in reducing noisy input, which can help to prevent overfitting in ML methods, see e.g. reference [92].

In reference [92], feature selection is performed using a random forest classifier and recursive feature elimination. This procedure consists of training a classifier on all available features and obtaining an importance score for each. The least important feature for classification is eliminated and the procedure repeated until the performance of the classifier no longer improves significantly (used in reference [92])

\(^{2}\)For lower thresholds, more sources are found and the maximum value is no longer at 98%.
4.4. Using an autoencoder for feature selection

or a desired number of features is found. This selection method however relies on labeled data for training the classifier and evaluating the feature importance. Also, the selected features depend on the task at hand and might not be describing sources well in general.

Here, we approach the selection differently. We do not select features according to improve some classification, but we choose those features that encode most of the information contained in all source features. One way to do so is the use of an AE. This approach has for example been chosen by the authors of reference [63] in the context of light curve analysis for supernova classification. An AE is a NN architecture that compresses (encodes) inputs into a latent space and inflates (decodes) this latent space back to the input space. It is then trained to minimize the difference between the input and the corresponding decoding. The latent space is chosen smaller than the input space, thus making compression of information necessary. A more detailed introduction of AEs is given in section 2.2.

We use a fully connected AE with three hidden layers and PReLU activation. We choose a five dimensional latent space, i.e. the second hidden layer has five nodes. The latent dimension is chosen to be comparable to reference [92], where the automated feature selection algorithm results in five features. We train the AE on the features included in the 4FGL-DR2 catalog. A first preselection of these features is necessary, as not all features are given for all sources. We take the approach followed by reference [92] and remove those features which are not available for more than five percent of the sources. Then, we remove all sources for which more than five features are missing. Finally, all remaining missing values are replaced by the mean of the missing feature over all other sources. We end up with 40 features before feature reduction. The final preprocessing step consists of standardization, i.e. subtracting the mean and dividing by the standard deviation for each feature respectively.

We consider the task of AGN vs. PSR classification and use 70% of the labeled sources that are not removed in the previous step as training set and the remaining 30% as test set. We train the AE on the test set to extract relevant features. Training on the test set would correspond to training on UNC sources in the final task of classifying those. A fully connected classifier with one hidden layer of 40 nodes and ReLU activation is trained on the extracted features of the training set, i.e. the latent representation of that set in the AE. We perform splitting of the data and training of both networks (AE and classifier) ten times. We find an accuracy of 97.96 ± 0.26% and a F1 score of 84.42 ± 2.04%. Applying the classifier directly on the 40 features gives 97.91 ± 0.48% accuracy and a F1 score of 82.52 ± 3.92%. While the overall performance is similar, the procedure with reduced numbers of features is much more stable.

The performance still remains below the 99.19% accuracy stated in reference [92]. However, we do not select features according to best classification and might thus reduce the sample (and feature) selection bias. The further investigation of the AE’s capabilities for feature selection, in particular with regard to the selection bias, are left for future work.
4. Classification of Fermi-LAT sources with DL using energy and time spectra

4.5. Summary

In this section, we have introduced a new approach for the classification of unidentified and unassociated sources of the 4FGL-DR2 catalog. We have classified these sources first into AGN or PSR and then further subclassified the PSRs into MSPs and YNGs. For this task we employed DNNs and RNNs directly to the energy and time spectra available for all sources within the catalog. We have seen that even for this comparably small dataset NNs are competitive with handcrafted features, which need further feature reduction methods to perform well. While the main purpose is the classification of UNC sources, we have validated our setup in two ways. First, we evaluated the performance on labeled sources within the catalog, reaching performances competitive with the methods tested in reference [92]. Second, we cross-matched the older 3FGL catalog and applied our method to previously unclassified sources that are now labeled in the 4FGL-DR2 catalog. The reduction in performance for this cross-matched dataset can be explained only partly by different compositions of the test set. The remaining difference can be associated to the sample selection bias, i.e. the tendency in astrophysical data to observe brighter sources earlier and with more significance compared to fainter ones.

We have identified 1050 (1076) AGN and 78 (57) PSR candidates out of 1679 UNC sources, by applying a threshold of 0.9 to our DNN (RNN) classifier output. The strict threshold is chosen to prevent labeling of UNC sources that correspond to a different class than AGN or PSR. The accuracy for these classifications is estimated close to 99% for AGNs and close to 93% for PSRs. This accuracy estimate is based on labeled sources in the 4FGL-DR2 catalog and does not take the sample selection bias into account. Similar results are obtained with the RNN. The further subdivision into MSPs and YNGs using the DNN leaves us with 22 YNGs and 21 MSPs. For this task, the RNN finds a reduced set of only five YNGs and no MSPs. We showed that the DNN predictions are consistent when changing the inputs by including the Galactic latitude and excluding the time spectra. The RNN predicts twelve MSPs when adding the Galactic latitude, which are consistent with the DNN predictions. We showed that there is overlap of our candidates with previous studies.

Overall, we found good agreement between RNN and DNN performance, with slight benefits for the DNN. Since the DNN is simpler and also faster, we consider this architecture as our benchmark for subsequent work, e.g. in section 5.

As the sources have been categorized by our networks only based on γ-ray properties, these labels should not be taken as exact. Rather, they provide information on what sources should be investigated further by multiwavelengths observations for final identification. The strategy of which sources to look at can be twofold. First, some of the provided candidates can be examined to confirm or reject the classification. Second, if the classification is reliable, those sources that remain unclassified should be analyzed, as their labeling can contribute the most to better classifications. Since our networks are flexible, one can include multiwavelengths information in the future. The task then becomes to combine various measurements of all sources.
4.5. Summary

We further employed a method for feature selection based on AEs. Performance wise the results between the two approaches studied in this section are similar with the AE approach giving smaller fluctuations over various runs. The selection of features using the AE has the advantage that the AE can be trained on unlabeled sources. This has the potential to reduce the selection bias, as features relevant to describe both bright and fainter sources can be constructed.

So far, in $\gamma$-ray classification the notion of uncertainties in the assignment of source candidates is missing. We introduce uncertainties estimated using BNNs and test their reliability in section 5.
5. Classification of Fermi-LAT blazars with Bayesian neural networks

In section 4, we performed a two stage, binary classification of UNC sources into AGNs and PSRs, and then of PSRs into MSPs and YNGs. In this section, we change to the subclassification of AGNs into BLLs and FSRQs and include a reliable uncertainty to our classification. Before, we estimated the uncertainty by training multiple DNNs. An alternative way to obtain an uncertainty from a NN is to make it Bayesian, i.e. replacing the fixed weights by trainable distributions. We apply a BNN to the subclassification of AGNs with the goal of classifying BCUs in this section. This section and the results therein follow closely reference [3].

We describe the relevant details and differences of the data used in this section compared to section 4 in section 5.1. We introduce the classifiers we apply, in particular the BNN, in section 5.2. The formal correspondence between BNNs and DNNs and how this correspondence turns out in practice within a toy model are part of section 5.3. In section 5.4, we discuss the effects of data augmentation on the uncertainty estimate using the same toy model. Finally, we present our results on the classification of BCUs in section 5.5 and summarize our findings in section 5.6.

5.1. The data: blazar subclasses in the 4FGL-DR2 catalog

We use data from the 4th Fermi-LAT γ-ray catalog, data release 2 (4FGL-DR2) [84]. The general properties are discussed in section 4.1. Here, we go into more detail on the AGN subclasses, in particular blazars.

Within the 4FGL-DR2 catalog we find almost 3500 blazars that are subdivided into ∼1300 BL Lacerae objects (BLLs), ∼750 Flat Spectrum Radio Quasars (FSRQs) and ∼1400 Blazars Classified as Uncertain type (BCUs). The exact numbers are shown in figure III.1. Our goal is the classification of these BCUs as FSRQ or BLL. The unified classification scheme of AGNs [87] states that all AGNs correspond to nuclei of galaxies with jet and tori that emit thermal as well as non-thermal emission. Blazars are those AGNs, for which the jet points towards our line of sight. The differences between subclasses are of observational type. For the BLL and FSRQ classification, these differences are based in their emission lines and continuum emission. Since the differences are observational, the classification might not always correspond to distinctive physical differences [81]. The observational features of BLLs are weak emission lines with potential absorption features. FSRQs on the other hand have strong and broad optical emission lines [87]. Observations thus suggest strong external radiation fields as well as dust tori. The observed class populations need not be representative of the full distribution in the universe, but might be driven by observational bias, see e.g. reference [87] for a more thorough discussion.

Strong distinguishing features for BLLs and FSRQs are the mean spectral power law index $\Gamma_\gamma$ and the luminosity $L_\gamma$. For FSRQs the following holds in general:
5.1. The data: blazar subclasses in the 4FGL-DR2 catalog

Figure 5.1: Energy spectra for BLLs and FSRQs of the 4FGL-DR2 catalog. The lines connect the median in each energy band. The shaded regions connect upper and lower quartiles.

\[
\Gamma_\gamma \gtrsim 2.2 \text{ and } L_\gamma \gtrsim 10^{39} \text{ W. For BLLs the opposite inequalities mostly hold, i.e. } \Gamma_\gamma \lesssim 2.2 \text{ and } L_\gamma \lesssim 10^{39} \text{ W. While these inequalities provide some separation of the classes, there is overlap between the two classes in these properties. Just as the properties of UNC sources suggest a composition of AGNs and PSRs (see section 4), the properties of BCUs suggest a composite of BLLs and FSRQs.}
\]

As in section 4, we want to use features for the classification that are as close to the measurement as possible. We found in section 4.3.1 that the classification performance hardly depends on the information contained in the time spectra. While the task of AGN vs. PSR classification is different, we expect this to hold also here. Therefore, we only use the energy spectra, i.e. the flux measured as a function of the energy, in this section. That these low-level features are well suited for classification of BCUs has also been shown in reference [22]. We show a summary of the data in figure 5.1. The figure shows the median flux for each of the seven energy bins together with upper and lower quartiles. The lines connecting different energy bins are shown to guide the eye. Only measurements integrated over the energy bands are given in the catalog. One can see the different slopes, i.e. the difference in the power law index, between the two classes. Moreover, the FSRQs have larger variations in the last bin, while the BLLs vary more strongly in the first bin. The shown spectra are not directly fed into the NNs. Instead, we subtract the mean and divide by the standard deviation over the training set for each energy band, after taking the \(\log_{10}\) as in figure 5.1.
5. Classification of Fermi-LAT blazars with BNNs

5.2. Classification methods

Building on our findings in section 4, we employ NNs for classification. We have seen that these are prone to overfitting, in particular since we have only limited data available. We briefly discussed the effects of data augmentation in section 4.3.1, and we revisit this discussion once more in section 5.4. The RNN described in section 4.2.2 did not show any significant advantage over the DNN. Thus, we stick to the simpler architecture (the DNN), as our benchmark. To estimate stability and uncertainty of the DNN, we previously trained it several times and looked at its variation. In this section, we compare this approach to the uncertainties obtained from a BNN. We first introduce the hyperparameters chosen for the DNN in section 5.2.1. Then, a more in depth explanation of BNNs in general and the particular architecture used in this section is given in section 5.2.2. The models in this section are implemented in Tensorflow [12] version 2.4.1 and the built-in version of Keras [106].

5.2.1. Fully connected neural network

The network used here is a simplified version of the one described in section 4.2.1. As we are only interested in the energy spectra, we do not need several branches. Thus, our network takes the standardized flux of each energy bin as its seven inputs, i.e. we also omit the significances compared to section 4. The input is processed by one hidden layer with 16 nodes and ReLU activation, and the final output layer, again applying softmax activation. The final layer has two nodes, each giving the probability of an input to belong to one of the classes (BLL or FSRQ). The network is trained using the Adam optimizer [50] with an initial learning rate of $10^{-3}$. We add a $L_2$-regularization term to the CE loss. Usually, this regularization enters with a hyperparameter that balances regularization and classification loss. However, this parameter is fixed in our setting, since we want to evaluate a correspondence between DNN and BNN (see section 5.3). We use a batch size of 32 and train the network for 250 epochs. The hyperparameters stated here were found to perform best in a random hyperparameter search, tuning the number of layers, their nodes, the batch size, the number of training epochs, the initial learning rate as well as the optimizer (tested were Adam and SGD). We used ten-fold cross-validation for the hyperparameter optimization, just as in section 4.2.1. The performance is robust against small changes, however one needs to be careful in giving the DNN too much expressiveness, i.e. making it too deep or too wide. This leads to overfitting, which we cannot compensate by adding regularization beyond the $L_2$-regularization already applied, because we do not want to spoil the correspondence to BNNs.

5.2.2. Bayesian neural network

For deterministic NNs, the weights are fixed after training and do not change when performing inference multiple times. BNNs replace these fixed weight values by trainable distributions. Thus, the weights are drawn according to their respective distribution for every inference. Performing inference of the same input several times
propagates this variation of the weights to the output and generates a distribution of output values. This distribution then allows us to estimate the final prediction, e.g. by taking the mean, as well as an uncertainty, e.g. by taking the standard deviation of the obtained distribution. During training, the BNN tries to learn the model posterior \( p(w|D) \), i.e. the probability distribution of the weights \( w \) given the training data \( D \). A deterministic NN can be seen as a maximum likelihood estimator of this distribution. The true posterior cannot be learned directly, since it is not tractable for the many parameters, or it is not suitable to evaluate exact distributions for every update step. Instead, the posterior is approximated by some parameterized distribution \( q_\theta(w) \) with trainable parameters \( \theta \). The BNN is then trained to minimize the difference between this approximate posterior and the actual posterior. One measure of this difference is the Kullback-Leibler (KL) divergence \([110]\) given by

\[
\text{KL}(q_\theta(w)||p(w|D)) = \int dw \ q_\theta(w) \log \frac{q_\theta(w)}{p(w|D)} = \int dw \ q_\theta(w) \log \frac{q_\theta(w)}{p(D|w)p(w)} + \text{const} = \text{KL}(q_\theta(w)||p(w)) - \int dw \ q_\theta(w) \log(p(D|w)) + \text{const} . \quad (5.1)
\]

The first row just gives the definition of the KL divergence. In the second row, we replace the posterior by applying Bayes theorem, the additional constant is \( \log(p(D)) \). This likelihood of the data cannot be calculated. Since it is independent of the network parameters \( \theta \), it is irrelevant for the optimization. To get to the third row, we rewrite the log by pulling out \( \log(p(D|w)) \) with a minus sign. One can immediately see that the KL divergence between our parameterized weight distribution \( q_\theta(w) \) and the prior \( p(w) \) is left. The second term is an expectation value of the log-likelihood of the data given the weight distribution. This corresponds to the likelihood loss used in the training of deterministic NNs. In particular, it boils down to the MSE for regression assuming Gaussian distributed errors on the regression targets or the CE for classification, see e.g. pages 140-142 and pages 205-207 of reference [35]. The first term can thus be seen as an additional regularization arising from the prior in the Bayesian setting. Averaging equation 5.1 over a training set of size \( N \) results in

\[
\frac{1}{N} \text{KL}(q_\theta(w)||p(w|D)) = \frac{1}{N} \text{KL}(q_\theta(w)||p(w)) - \frac{1}{N} \sum_{i=1}^{N} \log(p(x_i|w))) \\
\approx \frac{1}{N} \text{KL}(q_\theta(w)||p(w)) - \frac{1}{M} \sum_{i=1}^{M} \log(p(x_i|w)) . \quad (5.2)
\]

As first step, we replace the integral in equation 5.1 by an expectation of samples. Second, we apply the approximation of this sample average by the average over a small batch of size \( M \) rather then the full dataset. This approximation is done for any training of NNs that makes use of batch updates. It is important to note that the first, regularizing term in equation 5.2 keeps the \( 1/N \) scaling. Thus, the effect of
5. Classification of Fermi-LAT blazars with BNNs

the prior is reduced with increasing dataset size, as one would expect.

So far we have not made assumptions on the approximate posterior and the prior. Usually they are both taken to be multivariate, diagonal normal distributions with the posterior parameters being trainable. The KL divergence of two diagonal Gaussians can be calculated for each weight independently and the total KL divergence is then given by the sum over all weights’ contributions

$$KL(q_\theta(w)||p(w)) = \sum_i \log \frac{\sigma_{p,i}}{\sigma_{q,i}} + \frac{\sigma_{q,i}^2}{2\sigma_{p,i}^2} + \frac{(\mu_{p,i} - \mu_{q,i})^2}{2\sigma_{p,i}^2} - \frac{1}{2},$$  (5.3)

where $\mu_{p/q,i}$ ($\sigma_{p/q,i}$) are the prior and posterior means (standard deviations) of weight $w_i$, respectively. This can be further simplified if we set the prior mean $\mu_{p,i} = 0$ and neglect all those terms that do not depend on our posterior parameterization. Setting the prior mean to zero does not restrict the model, as any shift can be compensated by the bias term, which is taken to have width zero, i.e. it is set to be deterministic. We end up with the simplified KL divergence

$$KL_{\text{simp}}(q_\theta(w)||p(w)) = \sum_i \frac{\mu_{q,i}^2}{2\sigma_{p,i}^2} + \frac{\sigma_{q,i}^2}{2\sigma_{p,i}^2} - \log \sigma_{q,i}. \quad (5.4)$$

This is the term we apply together with the one over $N$ factor as regularization to the BNN, i.e. the first term of equation 5.2.

We set the architecture to be the same as for the DNN, i.e. one hidden layer with 16 nodes and ReLU activation followed by the two dimensional output with softmax activation. Also the training parameters are the same, we train using the Adam optimizer with an initial learning rate of $10^{-3}$ for 250 epochs with a batch size of 32. For the implementation of the BNN, we use the implementation of probabilistic layers within Tensorflow-Probability 0.12.1 [111]. We train these layers using the Bayes by backprop algorithm introduced in reference [112]. To be able to take the gradients of the loss with respect to the parameters of the posterior estimate, the reparameterization trick is applied. This results in sampling of a random weight as $w_i = \mu_{q,i} + \sigma_{q,i} \cdot \epsilon_i$. Here, $\mu_{q,i}$ and $\sigma_{q,i}$ are the trainable mean and standard deviation of the parameterized posterior $q_\theta(w)$, and $\epsilon$ is drawn from the standard normal distribution $\mathcal{N}(0, 1)$. The weights $w_i$ then follow a normal distribution $\mathcal{N}(\mu_{q,i}, \sigma_{q,i})$, as desired, but the parameters are available for the calculation of the gradient. During training the noise added to the weights would ideally be independent for each weight and also each data instance, to not introduce correlations in the gradient estimate. Correlations in the gradient estimate can slow down training and hinder convergence. However, this is not feasible, as it would require as many copies of the weights as there are samples in a single batch. We use the flipout estimator [113], which efficiently decorrelates the gradients for samples within one batch update. The flipout estimator samples pseudo-independent weights by randomly switching signs.
of $\epsilon_i$ for each sample. This includes some computational overhead, but leads to a more accurate estimate of the gradients. In particular, the scaling of the variance of the gradient with $1/\sqrt{B}$, where $B$ is the batch size, as expected from independent $\epsilon$ is recovered. We refer the reader to reference [114] for a practitioner’s review of BNNs and alternative implementations of Bayesian inference with NNs.

5.3. Correspondence between fully connected and Bayesian neural networks

The BNN gives us an ensemble of networks to estimate an uncertainty by sampling several times from the weight distributions. An alternative to obtain an ensemble of networks is to simply train several independent DNNs. The latter can be called the frequentist approach, as opposed to the Bayesian approach, in terms of different views on statistics. In this subsection, we intend to compare the two approaches from a theoretical as well as an application point of view.

A single DNN does not provide a distribution of weights. However, we can interpret its weights as a point estimate of the mean value of the BNN. This allows us to identify the first term in equation 5.4 with the usual $L_2$-regularization

$$L_2 = \lambda \sum_i w_i^2 = \sum_i \frac{\mu_{q,i}^2}{2\sigma_{p,i}^2}.$$ (5.5)

The two are equivalent once we set $\lambda = \frac{1}{2\sigma_p^2}$. We can omit the index $i$, as we only consider having the same prior for all weights, i.e. $\sigma_{p,i} = \sigma_p$. Additionally, the regularization in the BNN is divided by the number of training samples. We thus train our DNN with $L_2$-regularization, fixing $\lambda = \frac{1}{2N\sigma_p^2}$. Regularizing the DNN helps preventing overtraining and supports variation in predictions obtained from different DNN initializations for a given data point.

While some part of the regularization is now equivalent, the BNN is regularized by additional terms in equation 5.4, taking into account the prior and posterior width. As these additional terms follow directly from the Bayesian ansatz in equation 5.1, they take into account the limited size of the dataset. Moreover, the distribution of weights in several trained DNNs is not expected to be Gaussian. Training multiple independent DNNs should thus be closer to the actual posterior distribution. Nevertheless, the BNN can construct highly non-Gaussian distributions in the predictions from combining many Gaussians together with the nonlinearities in the activation functions.

We test the behavior and correspondence of the BNN and the DNN on a well-controlled dataset before their application to Fermi-LAT data. To do so, we sample two classes from two multivariate Gaussian distributions. We choose the Gaussians to be seven dimensional, in order to stay close to the setup applied to the physics
data later on. The two classes are very simple. For the first class, the mean values decrease linearly from 1 to -1 over the seven dimensions. The standard deviations are independent and decrease from 1.1 to 0.9. The second class inverts the order, i.e. mean values increase from -1 to 1 and the standard deviations increase from 0.9 to 1.1. We use the architectures as described in section 5.2.

The training set size is fixed to 1000 samples per class, i.e. a similar order of magnitude as in the AGN subclassification, however exactly balanced here. We then train the BNN once and sample the weights 100 times to obtain 100 predictions for each data point. To compare to the distribution we get from DNN training, we train 100 DNNs in two different ways each. First, we use the same training set as for the BNN and train 100 different initializations of the DNN. Second, we not only reinitialize the DNN, but also resample the training set. This is only possible in our toy example, because we have the exact distributions. We thus end up with three different ways to estimate a prediction and its uncertainty on an independently sampled test set.

We show the results in figure 5.2. We do not want to focus on the performance itself, which is rather good given the simple task, but the correlations between mean predictions and standard deviations of the predictions for the BNN and the two DNN setups. We take the DNN with independent training data as the benchmark,
as it should have the most reliable predictions and uncertainties, incorporating both network convergence and limited amount of data. The first row shows the correlations of the mean predictions $\mu$ of the BNN and the DNN trained with a fixed training set with the benchmark. One can see strong linear correlations for both panels. However, the spread around the diagonal is smaller for the BNN. The second row shows correlations of the estimated uncertainties $\sigma$, calculated as the standard deviation of predictions. Again, there is strong correlation for both BNN and DNN with fixed training set with our benchmark. While the BNN stays close to the diagonal at higher uncertainties and is above it for lower uncertainties, the DNN is below over the full range. Thus, the DNNs trained on a fixed training set significantly underestimates the uncertainty. The DNN ensemble trained that way cannot capture the additional uncertainty arising from a limited dataset. It can only capture the uncertainty due to different convergence of individual networks. This is improved for the BNN as it directly incorporates an uncertainty related to the size of the dataset. In reference [101], a study on BNNs for top jet classification finds similar results for independent training sets of the DNN. The authors additionally weaken the formal corresponds by varying the $l_2$ regularization strength $\lambda$ and adding dropout to the DNN, finding better correspondence for slightly reduced $\lambda$ and some dropout.

Our results show that simply training a DNN multiple times is not sufficient to estimate reliable uncertainties, as long as the uncertainty from small dataset size is relevant. The comparison to a DNN trained on independent datasets shows that the BNN is indeed able to capture the full uncertainty. For $\gamma$-ray source classification, we cannot simply generate more data following the correct distributions. We are thus relying on a small dataset of around 2000 sources in blazar subclassification, for which the uncertainty arising from this limited data is non negligible, as we have seen.

5.4. Data augmentation

The dataset we are looking at is not only small, with the 2000 labeled blazars, but also imbalanced. Close to 63% of the sources are BLLs. The imbalance is not as strong as for AGN vs. PSR classification in section 4, however still significant. Imbalances can lead to biases, i.e. a better performance on the majority class at the cost of poorer performance on the minority class. We already briefly discussed the ways to treat such an imbalance in section 4.3.1. We found no improved performance by any of the augmentation methods we used for the task of AGN vs. PSR classification. Here, we want to additionally consider the implications data augmentation has on the estimated uncertainties. Moreover, we stay in our well controlled toy model, allowing us to compare data augmentation with generating more data from the corresponding distribution.

We again consider the SMOTE algorithm, but in a slightly improved version compared to section 4. To remind the reader, within SMOTE new instances are generated as $S_{\text{aug}} = S_1 + r(S_2 - S_1)$, where $S_1$ and $S_2$ are random instances selected from within
5. Classification of Fermi-LAT blazars with BNNs

Figure 5.3: Comparison of mean prediction $\mu$ (top) and standard deviations $\sigma$ (bottom) for a BNN trained on 3000 instances per class with 2000 SMOTE augmented sources per class (left) and the BNN trained on 1000 instances per class (right). Both are compared to the predictions of a BNN trained on 3000 instances per class. The data corresponds to the toy model described in section 5.3.

The class that needs to be augmented and $r \in (0, 1)$ is drawn from a uniform distribution. Instead of picking $S_2$ arbitrarily from all instances in the same class as $S_1$, we select it randomly from a subgroup, containing the $k = 5$ nearest neighbors of $S_1$.

We choose three setups to compare the behavior of our BNN. First, we use the same setting as before in section 5.3 and train the BNN on 1000 samples per class. Second, we increase the number of samples per class to 3000 by applying the SMOTE algorithm to generate the additional instances. Third, we again train on 3000 instances per class, but sample them according to the known Gaussian distributions.

In figure 5.3, we show the same correlations as in figure 5.2, but for the three BNN setups. Here, we take the BNN with the larger training set of 3000 actual samples as benchmark. One can already see from the mean predictions in the top left panel that using SMOTE leads to an overconfident model. Many predictions are very close to zero or one and the spread for intermediate values is large. The baseline BNN trained on 1000 examples per class is much better aligned with the benchmark. Moreover, when looking at the standard deviations, we find that the SMOTE augmented BNN can no longer provide reliable uncertainties. The bulk of sources is in the lower left corner, below the diagonal, i.e. the SMOTE BNN underestimates the majority of uncertainties. For the remaining samples, the uncertainty scatters all over the place,
not giving a sound uncertainty estimate. On the other hand, we see a consistent reduction of uncertainty between the BNN trained on 1000 instances and the BNN with 3000 actual training instances.

We thus see that one needs to be careful with data augmentation when trying to estimate uncertainties. It is in general possible to estimate the uncertainty with augmented samples using a BNN. But, to do so it is necessary to incorporate the assumptions made for the augmentation of data into the prior of the BNN. While this is possible for augmentation using functions of individual samples, e.g. image rotation [114], it is not clear how to proceed for the SMOTE algorithm and our dataset.

5.5. Results

Given the results of section 5.4, we decide not to use data augmentation for the 4GL-DR2 data. Similar to section 4.3, we first discuss the evaluation of our networks on the labeled sources of the 4FGL-DR2 catalog in section 5.5.1. Further, we perform once more a cross-match between the 3FGL and 4FGL-DR2 catalog, but now for the task of blazar subclassification in section 5.5.2. Finally, we present our results on the classification of BCUs in section 5.5.3.

5.5.1. Performance on labeled sources of the 4FGL-DR2 catalog

Whenever dealing with Bayesian statistics, it is useful to understand the dependence of the results on the applied prior. We showed in equation 5.2 that the KL divergence between prior and approximate posterior is scaled by a factor one over training set size. While this naturally reduces the prior dependence for large datasets, we should carefully investigate it here. To evaluate the performance of our BNN, we perform ten-fold cross-validation and generate 500 predictions for each source. This has the drawback that we have to train 10 BNNs, but the benefit that ROC curves and performance measures shown in the following take into account the full statistics of labeled sources, i.e. all 2052 BLLs and FSRQs. This is different to what we did in section 4.3, where we took varying 30% of the full data for testing and performed the ten-fold cross-validation only for the hyperparameter search. We represent the classification of a single source by the mean and the standard deviation of the 500 predictions.

To investigate the prior dependence, we set the prior mean to zero and vary the prior width \( \sigma_p \). Setting the prior mean to zero does not restrict the model, as discussed in section 5.2.2. We show the ROC curves for varying priors in the left panel of figure 5.4. There are two curves for every prior width. The dashed curves show the mean of the 500 ROC curves we get from the 500 weight draws of the BNN. The solid curves show the ROC curve for taking the mean prediction. Those two are different since taking the mean prediction is effectively an ensemble method and it has been shown that ensembles often outperform individual classifiers, see e.g. the survey in reference [115]. This is especially true for weak classifiers, as individual classifiers may
be sensitive to different features, and combining them allows for different distinctions to be found. For a narrow prior width of 0.01, we see that the BNN does not learn anything and becomes just a random classifier. In fact, the BNN learns the class proportions and sets the output to this value for any given input. Increasing the prior by an order of magnitude to 0.1 already almost saturates the performance. For prior widths above 1, no significant change in the ROC curve is visible. This holds even at a prior width of $10^5$. The reason can be seen in equation 5.4. The two first terms are proportional to $1/\sigma_p^2$, generating a pull of mean and standard deviation of the posteriors to zero. The regularization becomes stronger than what can be gained in loss by performing a better classification if $\sigma_p$ is small. When pushing the prior width to very large values, the first two terms in equation 5.4 vanish. Without any regularization the BNN would set all $\sigma_q$ to zero, as the random fluctuations are reducing performance. The remaining negative logarithmic term prevents that from happening. The network thus learns to maximize performance, while maintaining a balancing variance in the weights.

The right panel of figure 5.4 shows the AUC and the average of the absolute value of the mean and of the standard deviation for the BNN posterior estimates. In addition to the step like performance increase in terms of AUC, we see a smoother transition regarding the averages of absolute mean and standard deviation. As the performance and the posterior width are varying only little beyond a prior width of one, we set our prior to be a standard normal distribution, i.e. we set $\mu_p = 0$ and $\sigma_p = 1$.

Now that the prior is fixed, we can examine the performance of our BNN on the labeled sources of the 4FGL-DR2 catalog. We show the resulting ROC curve in figure 5.5. The solid and dashed line are already shown in the left panel of figure 5.4. Additionally, we include the a subset of 100 different ROC curves from various weight
5.5. Results

Figure 5.5: Left: BNN ROC curves, the dashed line corresponds to the mean of the single draws and the solid line is the ROC curve of the mean BNN prediction. Right: Comparison of the calibration curves for BNN and DNN. The lighter blue lines in both plots correspond to 100 individual weight draws of the BNN.

draws of the BNN. These additional curves illustrate the difference between the solid and the dashed line. The dashed line is obtained by taking the mean of the ROC curves from different BNN realizations. To get the solid line, we instead first take the mean of all BNN predictions and then calculate the ROC curve from these mean predictions.

In addition to the ROC curve, we show the calibration curve in the right panel of figure 5.5. The points along the curve are calculated by first binning the FSRQ probability into ten bins such that each bin has the same number of sources in them. Then, for each bin the average predicted FSRQ probability gives the x-coordinate and the fraction of FSRQ sources in the bin the y-coordinate. Perfect calibration lies on the diagonal, i.e. the probability of having a FSRQ with a given predicted score is equal to that score. This highlights the relevance of well calibrated classifiers, as soon as we interpret the score as probability. We average the prediction of the DNN over 100 runs, where we independently initialize the weights and perform the division into ten folds. For the BNN we use the 500 predictions from different weight draws. We see that both BNN and DNN are well calibrated. The figure shows not only the calibration of the mean prediction, but 100 randomly chosen individual predictions of the BNN. These individual runs are also well calibrated, however with slightly wider spread around the ideal calibration.

So far we have not taken the uncertainty into account. We show how the standard deviation changes as a function of the mean prediction of the FSRQ probability ($\mu_{FSRQ}$) in the left panel of figure 5.6. We compare 100 BNN predictions to 100 DNN predictions. We reduce the number of predictions so we do not have to train...
5. Classification of Fermi-LAT blazars with BNNs

![Figure 5.6: Left: Correlation between the standard deviation of the FSRQ prediction and its mean for 100 BNN draws and 100 DNN runs. Right: 4FGL-DR2 sources in the plane spanned by the power law index best fit value and its uncertainty. The color coding shows the classification uncertainty of the BNN.](image)

5000 DNNs.\(^3\) We show individual γ-ray sources as scatter points and add a histogram showing the mean value for each bin. First, we notice that the distribution forms an arc. This is exactly what one would expect. First, for predictions very close to zero or one, the network should be very certain. Moreover, a mean value close to zero or close to one does not allow for large fluctuations, i.e. large standard deviations. Close to the boundaries the assumption to estimate the uncertainty by the standard deviation breaks down. An alternative would be for example using medians and quartiles. Nevertheless, the qualitative findings remain the same. The uncertainty obtained from 100 DNNs is much smaller than for the BNN. Even though we perform the split into ten folds independently for each DNN run, the training sets used for the different runs are not independent, but almost the same. As we have seen in section 5.3 this leads to underestimated uncertainties, as the limited dataset size is not (fully) taken into account. One can thus interpret the difference between the two arcs to be this additional uncertainty, assuming that BNN and DNN share the same variance in network convergence.

Particularly interesting are the sources that are at intermediate probabilities, but have small uncertainty. The fraction of FSRQs in the dataset is at about 0.37. If a source has the same probability under the distribution of FSRQ and BLL, i.e. \(p_{FSRQ}(x) = p_{BLL}(x)\), we expect it to obtain this score. This holds, because we do not correct for the imbalance during training. The feature of low uncertainties lies around this value. Sources with \(\mu_{FSRQ} \approx 0.37\) and small uncertainties thus correspond to sources which the network is certain that they are equally likely under the FSRQ and the BLL distribution.

Another interesting part are the few sources that have a very high score close to

\(^3\)Since we perform ten-fold cross-validation, we need to train ten networks for each prediction of the full dataset.
one, but nevertheless show a large standard deviation. To obtain this behavior, it is necessary that these sources switch the predicted class for only very few BNN realizations. E.g. having 100 predictions of which 99 give \( p_{\text{FSRQ}} = 1 \) and only one with \( p_{\text{FSRQ}} = 0 \), results in \( \mu_{\text{FSRQ}} = 0.99 \), but gives a standard deviation \( \sigma_{\text{FSRQ}} = 0.10 \). These sources should be especially interesting for follow-up observations.

The right panel of figure 5.6 shows the sources in the plane spanned by the uncertainty of the power law index and the power law index itself. The power law index is fitted to the energy spectra and assigned an uncertainty according to how well the fit describes the spectrum. The color coding represents the uncertainty of the BNN prediction. There are two trends that become visible in this plot. As discussed in section 5.1, the two classes are well separated by the power law index, roughly at a threshold of 2.2. The uncertainty clearly increases from both sides towards this value. Furthermore, there is an increasing uncertainty in the prediction with increasing uncertainty of the power law index. Large uncertainty in the power law index corresponds to fluctuating energy spectra, for which it is difficult to figure out which class they belong to. The point with highest uncertainty in the prediction is at a power law index close to 2.2, but at very small uncertainty of this fit. We looked at this spectrum in particular and found that it is out of the typical flux range, with higher fluxes in all energy bins compared to the other sources. This is an example for which the BNN learns to assign high uncertainty to data that is beyond (however close to) the domain of training data. This is not always the case and cannot be guaranteed. In general, uncertainties provided by a BNN are only reliable in regions of data where they are trained on, see e.g. references [116, 117] for a more detailed discussion.

5.5.2. Cross-match between 3FGL and 4FGL-DR2 catalogs

In this section, we evaluate how well our BNN works on the classification of BCUs of the 3FGL catalog that have obtained a class label in the newer 4FGL-DR2 catalog. As discussed in section 4.3.2 the labeled samples in astrophysical data are biased towards brighter sources. Thus, this test allows us to evaluate in how far performance on yet unclassified sources might degrade. We cross-match blazars in the 4FGL-DR2 catalog to the previous 3FGL catalog. The training set, i.e. those blazars that are classified with the same class in both catalogs, consists of 653 BLLs and 458 FSRQs. The test set, i.e. those sources that were BCUs in the 3FGL and are now classified as BLL or FSRQ, contains 647 BLLs and 283 FSRQs.

As already hinted in section 4.3.2, we do not use the 3FGL features in this section. Instead, we use the energy spectra of the selected sources as given in the 4FGL-DR2 catalog. We train the BNN on the full training set and then evaluate the performance on the test set. The architecture and training parameters are the same as discussed in section 5.2.

Using the mean prediction of the BNN and a classification threshold of 0.5, we find
5. Classification of Fermi-LAT blazars with BNNs

<table>
<thead>
<tr>
<th>Threshold (τ)</th>
<th>N&lt;sub&gt;Sources&lt;/sub&gt;</th>
<th>Acc [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>703</td>
<td>93.31</td>
</tr>
<tr>
<td>0.6</td>
<td>577</td>
<td>93.93</td>
</tr>
<tr>
<td>0.7</td>
<td>455</td>
<td>95.16</td>
</tr>
<tr>
<td><strong>0.8</strong></td>
<td>327</td>
<td><strong>96.33</strong></td>
</tr>
<tr>
<td>0.9</td>
<td>194</td>
<td>95.88</td>
</tr>
</tbody>
</table>

Table 5.1: Number of classified sources and the accuracy in the cross-match test set for different thresholds τ. In bold are the selected thresholds for what we refer to as loose (0.5) and tight (0.8) selection.

We used the 3FGL catalog to select the thresholds also for the BCU classification within the 4FGL-DR2 catalog, as the choice requires labels of the test set. It is not guaranteed that the accuracy is the same. Alternatively, we could have used the cross-validation procedure within the 4FGL-DR2 catalog (see section 5.5.1). However, this would not include the sample selection bias. Nevertheless, we show the confusion matrices for both, cross-match and 4FGL-DR2 sources, in table 5.2. The resulting accuracies for the upper row, i.e. the cross-match data are listed in table 5.1. For
5.5. Results

<table>
<thead>
<tr>
<th>Pred BLL</th>
<th>Pred FSRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>True BLL</td>
<td>261</td>
</tr>
<tr>
<td>True FSRQ</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pred BLL</th>
<th>Pred FSRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>True BLL</td>
<td>518</td>
</tr>
<tr>
<td>True FSRQ</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pred BLL</th>
<th>Pred FSRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>True BLL</td>
<td>861</td>
</tr>
<tr>
<td>True FSRQ</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pred BLL</th>
<th>Pred FSRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>True BLL</td>
<td>1142</td>
</tr>
<tr>
<td>True FSRQ</td>
<td>81</td>
</tr>
</tbody>
</table>

Table 5.2: Confusion matrix using a threshold of 0.8 (left) or 0.5 (right) for cross-matched sources (top row) and sources of the 4FGL-DR2 catalog (bottom row) using ten-fold cross-validation.

For the 4FGL-DR2 sources, the accuracies are 96.7% and 92.6% for the tight and the loose selection, respectively. They are thus in good agreement with the results we see for the cross-match dataset. The percentage of sources that are classified with the loose selection are 88.7% and 75.6% for the 4FGL-DR2 and the cross-match sources, respectively. The tight selection keeps 60.6% and 35.2% for the two datasets. While the accuracies are similar, the additional amount of training data and the absence of the sample selection bias allow the BNN to be more certain on a larger fraction of sources. We see in section 5.5.3 that the ratio of classified BCUs within the 4FGL-DR2 catalog is between the two scenarios compared here. Overall, we see that the accuracy estimate is rather stable and thus expect it to hold as well for the BCU selection performed in the next section.

5.5.3. Predictions for BCU in the 4FGL-DR2 catalog

We apply the two selection thresholds to the BCUs in the 4FGL-DR2 catalog after training on the full set of labeled sources. We obtain 756 BLLs and 363 FSRQs as well as 429 BLLs and 178 FSRQs for the loose and tight selection, respectively. The distribution of the power law index for labeled BLLs and FSRQs as well as BCUs are shown together with our candidates in figure 5.7. We see that the distributions for BLLs and FSRQs are centered around 2.0 and 2.5 for both the labeled sources and our candidates. The tight selection mostly removes source candidates that have an intermediate power law index. Looking back at the right panel of figure 5.6, this is well consistent, as the sources with intermediate power law index are assigned higher uncertainty.

In addition to creating a list of individual targets that are promising for follow-up observations, we can use the results of our classification to complement population studies. These studies try to estimate the collective properties of γ-ray blazars. The results can then be used to quantify different astrophysical contributions to the γ-ray background emission (see references [118–120]). A good understanding of this
5. Classification of Fermi-LAT blazars with BNNs

5.6. Summary

We have presented BNNs as an architecture to classify \( \gamma \)-ray sources with reliable uncertainties. The focus in this section is the classification of blazars into BLLs and FSRQs. Similar to section 4, we employed our networks directly on the measured flux spectra. Due to the small dataset size (\( \mathcal{O}(2000) \)), DNNs are prone to overtraining, resulting in poor generalization. BNNs show more robustness, as they have an intrinsic regularization from the prior.

Within our study of a toy example in section 5.3, we showed how an ensemble of DNNs underestimates the predictive uncertainty if trained on the same data. The BNN is more suitable to capture not only the uncertainty arising from model convergence, but also the uncertainty coming from limited training statistics. We find that the uncertainty estimate within the toy model follows closely the frequentist approach, for which we trained various DNNs on independent training sets. The same holds not only for the uncertainty, but also the mean prediction.

Furthermore, we have studied the effect of data augmentation. We compared

---

Figure 5.7: Distribution of power law index for candidates applying the loose selection (dashed) and the tight selection (filled) compared to labeled sources (solid) and BCUs (dotted).

The background is needed to constrain exotic \( \gamma \)-ray emissions, e.g. from DM. We predict a population of 60% (32%) of the 4FGL-DR2 \( \gamma \)-ray blazars to be BLLs (FSRQs). This takes into account both the already labeled and our candidate sources, using the loose selection. Future model fits could include our tight candidates to complement statistics, particularly in the low-flux regime. The application of our source labels to these potential follow-up investigations is left for future work. The list of our candidates is provided at reference [109].
5.6. Summary

Sampling more data from the true distributions to SMOTE augmented data. We found that not incorporating the assumptions going into the augmentation of data in the prior results in underestimated and unreliable uncertainties from the BNN. We therefore once more refrain from applying SMOTE data augmentation. Performing augmentation that is closer to sampling from the actual distributions should however still improve the results.

For the classification of labeled blazers in the 4FGL-DR2 catalog, we scrutinized the BNN performance and its dependence on the prior. We see that for a sufficient prior width (\(O(1)\)) the performance and the uncertainty estimate becomes independent of the prior. The multiple evaluations of a BNN can be seen as an ensembling method, showing better performance combining predictions compared to individual draws of the BNN weights.

We find that multiple DNNs trained on data that is not independent between runs indeed estimate significantly smaller uncertainties than the BNN. The assigned uncertainties allow for some interpretation of the BNN. As an example, we studied how the uncertainty in the source type prediction varies in the plane spanned by the fitted power law index and the uncertainty of this fit. Larger variations in the energy spectra and power law indices between the characteristics of BLLs and FSRQs lead to larger uncertainties.

A cross-match test in line with section 4.3.2 gives us an estimate on how the sample selection or Malmquist bias might influence the performance when classifying BCUs. We choose two different working points that combine the mean prediction and the estimated uncertainty. We estimate the accuracy of these working points both on the cross-match dataset and on the labeled sources of the 4FGL-DR2 catalog, giving similar results. We get an accuracy of \(~\sim\)96\% for the tight selection, which maximizes the accuracy. The loose selection still achieves \(~\sim\)93\% accuracy, maximizing the number of sources that are classified.

Finally, we perform the blazar subclassification of BCUs and find \(~\sim\)1100 and \(~\sim\)600 candidates for the loose and tight selection respectively. The candidates we find show similar distributions, e.g. in the power law index, compared to the already labeled sources. The inter class overlap is reduced for the tight selection. Moreover, recent work on the classification of BCUs finds a large overlap with the candidates that we provide, using a gradient boosting decision trees [121].

To conclude, we find that BNNs are particularly well suited for classification tasks for which a non-negligible contribution of uncertainty comes from the limited amount of data available. Their self-regularizing properties allow for training with reduced tendency of overfitting. Moreover, if reliable uncertainties are needed, widely used DNN ensembles tend to undershoot while BNNs have shown to accurately estimate those uncertainties.
Part IV.

Less than supervised learning for LHC data analysis
In part III, we applied supervised ML to astrophysical data. We now switch from γ-ray sources to jet signatures at the LHC, and at the same time from supervised to less than supervised methods. The term "less than supervised" is taken from reference [122] and refers to methods that can be applied directly to experimental data of the LHC, where labels are not available. This is in contrast to the previous sections, where the definite classification of γ-ray sources can be provided. Within this thesis, we focus on two tasks. First, we employ the CWoLa method [30] to boost the discovery potential of mono-jet searches in section 6. Second, we perform density estimation in the high-dimensional space of low-level jet features in section 7.

The ultimate goal for the application of DL to new physics searches would be to apply an algorithm to the full LHC data resulting in a set of events that are not described by the SM. This generic task is best summarized under the term anomaly detection. In the most general case, no clear categorization of background and signal is available within the data. The field of anomaly detection in HEP is developing fast, with many new ideas coming up on short time scales. Efforts to compare new methods have been made in terms of community challenges. The LHC Olympics 2020 [123] and the Dark Machines Anomaly Score Challenge [124] provide benchmark datasets. We refer to the anomaly detection section in the Living Review of ML in HEP [28] for an up to date list of publications. During the course of this thesis, we have worked on a variety of unsupervised methods, i.e. methods not requiring any labels, and their application to anomaly detection.

The goal of anomaly detection is to find data instances in a dataset that are different from the rest without a priori knowledge on how they differ. There is a distinction between outlier anomalies, i.e. anomalies that are outside of the space that is populated by normal data, and anomalies that occur as overdensities. Overdensities are so called group anomalies, as any single instance could be perfectly normal data, but the anomaly arises from their abundance. A physics example for these overdensities are resonances, e.g. a signal localized in some invariant mass distribution.

Methods for anomaly detection can be built from a variety of unsupervised methods. We investigated the use of AEs, k-means clustering and explicit density estimation for anomaly detection in reference [6]. We conducted a more in depth study of AEs, their potential and their limitations in reference [3].

The purpose of AEs in terms of their original task is the reduction of dimensionality with minimal loss of information, see section 2.2. When applying an AE to anomaly detection, the assumption made is that the reconstruction of normal data is better than of anomalous data. The reasoning behind this assumption is that the AE sees normal data more frequently during training and thus should perform better in its reconstruction. The reconstruction error thus becomes an approximation for the probability density. In reference [3], we demonstrated that, in addition to the frequency of occurrence, the complexity of the data plays a crucial role. Using an AE for anomaly detection is biased towards finding anomalies that have a higher intrinsic complexity and fails in finding simpler anomalies. Current studies on normalized AEs are a step towards overcoming this bias [125]. Normalized AEs are energy-
based models that make the probability interpretation of the reconstruction loss more immediate through proper normalization. Our study of AEs in HEP has been part of my master’s thesis [126]. It has been continued by Ivan Oleksiyuk in his bachelor’s thesis [127] and beyond, leading to the publication of reference [3].

The application of $k$-means clustering to anomaly detection followed from the observation that an AE with very limited reconstruction capability showed less a reduced complexity bias [70]. The main idea behind the application to anomaly detection boils down to $k$-nearest-neighbor density estimation [128, 129], using the cluster centers obtained by $k$-means clustering as representatives for the background data. A similar approach using $k$-medoids is used in reference [130]. The intensive study of this method, with various approaches to combine the results of the clustering algorithm with anomaly detection, has been part of Ivan Oleksiyuk’s master project [131].

These methods are designed to find out of distribution anomalies as they all have the underlying idea of tagging something that is not (or rarely) seen during training as anomalous. To find an overdensity, an estimate of the background density is necessary. This estimate can be compared to the actual densities in data to find potential signal, see e.g. references [5,132–134] and section 6.

It is impossible to extract a sample of pure signal or pure background events in experimental data. However, it is common to separate the data into regions that contain mainly background (control regions) and regions in which one expects some signal (signal regions). These regions are usually obtained by cuts in one dimensional distributions, e.g. the missing transverse energy of an event or some invariant mass. There is much more information within an event in addition to these cut observables. There have been various ML methods suggested to exploit this additional information in auxiliary features to enhance the sensitivity of searches that include control regions [30, 132–143]. Additionally, there has already been a first application to an experimental analysis [144].

In HEP, the true nature, signal or background, of an event is known only for simulated data. For the LHC, data is simulated with unprecedented accuracy, matching many kinematic distributions between simulation and experimental data. However, systematic differences in low-level particle information are difficult to estimate. Therefore, applying DL methods that are trained in a supervised manner on low-level, simulated data to experimental data carries the risk of unknown systematic uncertainties [145, 146]. Moreover, supervised methods are limited to the task they are trained on. A model trained to find a certain kind of signal in a given background cannot be used reliably to find another signal. While there are examples for which a classifier trained on a given signal is sensitive to other signals as well [147], this is not generically true. In reference [1], we showed that, even for the same signal model, DL classifier performance can degrade strongly when evaluated on model parameters different from the ones used during training. We consider the same signal model as in that reference in section 6, applying a weakly supervised framework to boost the sensitivity of an existing search.
Uncovering differences between signal and control region in features beyond those separating the two regions is the key idea behind the CWoLa method [30]. A classifier is trained to distinguish the two regions, given the auxiliary features. This classifier, if trained properly, is sensitive to any difference between the two regions, independent of any signal assumptions. While a supervised classifier trained on a particular signal sets the upper limit on what sensitivity can be achieved, CWoLa can be used directly on data and search for any signal. In HEP in general and at the LHC in particular, it is not clear what the potential signal might be. It is thus important to employ general, model agnostic methods.

We discuss how one can use CWoLa to boost the sensitivity of searches that already provide a signal and a corresponding control region in section 6. We apply the CWoLa method to the mono-jet search conducted by the ATLAS collaboration [31]. CMS has performed a similar search in reference [148], with a stronger focus on setting limits to specific models. We show that the sensitivity to an exemplary DM signature, namely semi-visible jets from a strongly interacting sector, can be drastically improved, even for imperfect control regions.

An extension to the CWoLa method for resonance searches is called Classifying Anomalies THrough Outer Density Estimation (CATHODE) [133]. This method uses sidebands around a resonance to train a conditional density estimator. The density is interpolated to the resonant (signal) region and used for sampling. The samples then comprise the control region in the CWoLa setup and data is used for the signal region. This comparison of samples to data within the signal region allows the use of features that are correlated with the selection criteria of signal and control region. The standard CWoLa method [137,138] does not allow that, as the auxiliary features are required to be equally distributed for background in the control and signal region. The study of CATHODE, starting from the density estimation and scrutinizing the classifier distinguishing samples and data has been part of Marie Hein’s master project [149] and is currently ongoing.

Going beyond weak supervision, we consider the unsupervised task of density estimation in section 7. Learning the probability density of LHC data can be a promising task, as it allows for improved analyses, generation of synthetic data or anomaly detection [26,32]. Typically, densities are only estimated in lower dimensional spaces of handcrafted features. We have seen that the direct use of low-level features as input to DL methods allows the extraction of correlations beyond high-level features for a given task. Still, the curse of dimensionality makes density estimation in high-dimensional spaces notoriously challenging. It becomes increasingly difficult to fill the phase space with increasing dimension, i.e. in high-dimensional spaces all datasets are sparse.

Therefore, density estimation in HEP has so far been focused on low dimensional data, e.g. in references [132–134]. For higher dimensional data the focus has been on the generation of data, see reference [150] for a review. To generate data it is not necessary to be able to calculate $p(x)$ for a given instance $x$.

A prominent example for models that are able to sample without a tractable prob-
ability density function are Generative Adversarial Networks (GANs) [151]. In a GAN setup, two networks, a generator and a discriminator, are trained as adversaries. With sufficient data and training time an equilibrium is reached at which the generator generates samples according to the data distribution \( p(x) \) and the discriminator is not able to distinguish samples from actual data. The high fidelity and flexibility in the setup of GANs make them a prominent architecture in HEP. Examples for application are event generation [152], fast detector simulation [153] or unfolding of detector effects [154], we refer the reader to the GAN section of reference [28] for a more exhaustive list. Due to the adversarial nature of this setup, it can however be difficult to stabilize the training convergence.

An alternative approach for data generation that additionally gives a tractable density is the application of normalizing flows [155]. Normalizing flows construct a bijective mapping, limiting their flexibility as input and output dimension are fixed and equal. Usually, the risk of manifold collapse [156, 157] complicates the application of normalizing flows to higher dimensional data. Manifold collapse occurs if data lives in a lower dimensional space than the features describing the data, e.g. a circle described by \( x \) and \( y \) coordinates in Euclidean space lives on a one dimensional submanifold. Normalizing flows can be correct in estimating the in-manifold density, but give the wrong overall normalization. Moreover, training can become unstable as the density estimator needs to learn a delta peak in some dimension. In references [158–161], normalizing flows are nevertheless successfully used for high-dimensional detector simulation. Noise in physics data prohibits an exact collapse of the manifold onto a submanifold and can thus allow for sufficiently accurate training.

We focus on yet another way to obtain a tractable probability density of high-dimensional data in section 7. We employ the transformer-encoder architecture introduced in reference [9], modified to perform autoregressive density estimation. A similar setup has been introduced for jet physics using a RNN in reference [32]. Our setup is closely related to Transformer for Density Estimation (TraDE) [162]. The transformer architecture is state-of-the-art in NLP [163]. The relevant parts for our application are described in section 2.4. Current applications in HEP modify this architecture to fit particle physics data, e.g. by incorporating certain symmetries, and apply it to a wide range of tasks such as jet tagging [164, 165], reconstruction [166, 167], learning jet representations [168] and data generation [169–171]. We follow another approach by modifying particle physics data to look more like natural language in line with reference [172]. More specifically, we consider jet data. Natural language consists of a discrete set of words that form a sentence according to some grammar. In the analogy to jet physics, the words become constituents, the full sentence becomes the jet, and the ordering of constituents imposes a grammar.

We evaluate our density estimate in two ways. First, by training two density estimators, one on top jets and one on QCD jets, we can evaluate an estimate of the likelihood ratio. Using this estimate as classification score, we can compare to the performance of a supervised classifier, which approximates the performance of the correct likelihood ratio. Within this comparison we find that the model indeed learns relevant differences in the densities of the two classes, albeit being far from
optimal. Second, we evaluate the quality of the density estimate by sampling from it and comparing these samples to data. Learning the correct density leads to samples that are indistinguishable from data. We evaluate the difference between samples and data using marginalized features, but also using a ParticleNet classifier. We find overall good agreement for our transformer samples.

This part is structured as follows. We apply the CWoLa method to the ATLAS mono-jet search in section 6. In section 7, we discuss the application of transformer to density estimation in the high-dimensional space of jet constituents.
6. Weakly supervised deep learning for mono-jet searches

In this section, we investigate how to use the CWoLa method introduced in reference [30] in searches that provide signal and control regions. As an example we exploit the most recent ATLAS mono-jet search [31]. We introduce semi-visible jets as signal, a signature arising from a strongly interacting dark sector. While the method itself is model agnostic, i.e. it can be employed for the search of any signal in the data and does not depend on us to provide a specific signal model. The example helps to introduce the method and quantify improvements.

We introduce the CWoLa method in detail in section 6.1. The search and its relevant SM backgrounds are discussed along with the exemplary signal in section 6.2. We show our results using a simplified setup with only the main background in section 6.3.1 and using a more realistic background composition in section 6.3.2. Finally, we summarize our findings in section 6.4. The content of this section builds on the master’s project of Maximilian Lipp [173] and closely follows the resulting publication in reference [5].

6.1. Classification without labels

A typical setup at HEP experiments defines a Signal Region (SR) and a Control Region (CR). The SR is constructed such as to maximize the fraction of signal

\[ f_{\text{SR}}^{\text{A}} = \frac{N_{\text{SR}}^{\text{A}}}{N_{\text{SR}}^{\text{B}}} \].

Here, we use the index \( A \) to refer to anomalous events (signal) and the index \( B \) for background. \( N_{\text{SR}}^{\text{A}} \) thus refers to the number of signal events in the SR. The maximization of \( f_{\text{SR}} \) requires some idea or knowledge about a potential signal for which this fraction is optimized. The CR is then constructed to estimate the properties of the background in the SR. It is therefore necessary that it contains no signal or at least much less than the SR. We denote the signal fraction in the CR as

\[ f_{\text{CR}}^{\text{A}} = \frac{N_{\text{CR}}^{\text{A}}}{N_{\text{CR}}^{\text{B}}} \]

and by construction it holds that \( f_{\text{CR}} < f_{\text{SR}} \).

This directly translates to the CWoLa method introduced in reference [30]. If data can be split into two sets that only differ in the fraction of signal, one can show that the ideal classifier between the two sets is also the ideal classifier for signal vs. background. The setting is visualized in figure 6.1. The left panel shows the usual supervised setting, in which signal and background comprise separated sets. The CWoLa setting is displayed in the right panel. There, instead of signal and background labels we have a signal region and a control region with different fractions of signal. The ideal classifier between SR and CR is given by the likelihood ratio, according to the Neyman-Pearson lemma [174]. The likelihood ratio can be written as

\[ L_{\text{SR}/\text{CR}} = \frac{p_{\text{SR}}}{p_{\text{CR}}} = \frac{f_{\text{SR}} p_{\text{S}} + (1 - f_{\text{SR}}) p_{\text{B}}}{f_{\text{CR}} p_{\text{S}} + (1 - f_{\text{CR}}) p_{\text{B}}} = \frac{f_{\text{SR}} L_{\text{S}/\text{B}} + (1 - f_{\text{SR}})}{f_{\text{CR}} L_{\text{S}/\text{B}} + (1 - f_{\text{CR}})} , \] (6.1)

where \( p_{\text{B}} \) and \( p_{\text{S}} \) refer to the probability densities of background and signal, respec-
6. Weakly supervised deep learning for mono-jet searches

![Diagram of supervised and CWoLa classifiers]

Figure 6.1: Visualization of the basic idea behind CWoLa. Left: The usual supervised setup. Right: Two mixed regions used for CWoLa training. The signal fraction in the SR is much higher than in the CR. This figure is adapted from figure 1 in reference [30].

In the last step, we wrote the likelihood ratio as a function of the likelihood ratio between signal and background $L_{S/B} = p_S / p_B$, i.e. the ideal test statistics for separating signal and background.

A key assumption of the CWoLa method becomes apparent in equation 6.1. We use the same probability densities for background (and signal) in the numerator and the denominator. This implies that the background samples in the signal and the control region follow the same distribution. This assumption is crucial within the CWoLa method. It directly implicates that we are not able to use for our classification the features that are used to separate CR and SR. Furthermore, the features we use for classification may not be correlated with the separation of SR and CR. We refer to these uncorrelated, additional features as CWoLa features in the following. Within equation 6.1, this becomes obvious if we write $p_S / p_B = p_S / p_B(x)$, where now $x$ are the CWoLa features used for classification. We investigate how well the CR needs to be constructed in order to fulfill this assumption sufficiently in section 6.3.2.

If we take the partial derivative of $L_{SR/CR}$ with respect to $L_{S/B}$, we get

$$\frac{\partial L_{SR/CR}}{\partial L_{S/B}} = \frac{(f_{SR} - f_{CR})}{(f_{CR} L_{S/B} + 1 - f_{CR})^2} > 0 . \tag{6.2}$$

The inequality follows from the construction of the regions such that $f_{SR} > f_{CR}$ is fulfilled. Thus, the two likelihood ratios are strictly monotonically related, resulting in the same decision surfaces for optimal classifiers on the two sets.

A more instructive, but less formal, way to understand why this approach should work is as follows: The background distribution in both the SR and CR is identical, meaning that a classifier cannot distinguish between the two regions based on the background alone. The same holds true for the signal. As a result, the only knowledge the classifier can learn is that there is more signal in the SR. The only way to assign a higher likelihood for the signal to fall into the SR is by learning the difference...
6.1. Classification without labels

between signal and background.

The statements above are true only for optimal classifiers. In practice, we do not
know how close a trained classifier is to being optimal. However, using simulated data
allows us to get an estimate by comparing a classifier trained in the CWoLa setup
to a supervised classifier trained on sufficient statistics. Two factors make training
the CWoLa classifier particularly difficult. First, we are interested in small signal
rates, because large amounts of signal are mostly excluded already. Second, we want
to use low-level features, to be as model agnostic as possible. We have seen that
the classification performance of a CWoLa classifier weakens with additional features
that do not contain information, due to the additional noise [149]. We investigate
how well the idealized scenario agrees with realistic applications and imperfections
in the CR in section 6.3.

At this point, we want to stress that this approach is model-agnostic after applying
the cuts to separate SR and CR. The CWoLa method generates sensitivity beyond
the cut features to any differences in the CWoLa features. In particular, there is no
further assumption on what the signal might look like in these features. While this
is true in general, the selection of CWoLa features does play a role.

As an example, CWoLa has been shown to improve sensitivity for bump-hunt
searches in references [137,138]. Bump hunts are prominent searches in HEP. They
assume that a potential signal is localized (resonant) in some feature, e.g. a recon-
structed mass. The construction of signal and control region is then straightforward.
The SR is placed around the resonance with some width related to how broad one
expects the signal to be distributed. The CR consists of the side bands, i.e. the
regions adjoining the SR. The regular bump hunt fits a function in the side bands
and interpolates into the SR, assuming a smooth background distribution. If there
is an additional signal, it shows up as excess beyond the interpolated background
expectation. The CWoLa method can boost the sensitivity, by selecting a subset of
the data with higher signal significance. To achieve this goal, we train a classifier
on SR vs. CR, as described above. We then apply a cut on the classifier to keep
only a fraction of most SR like data. We can extract the number of background
events that we expect in the SR from the classifier scores in the CR. Any excess
above that number can then be related to a potential signal. However, keep in mind
that the CWoLa features used to train the classifier need to be uncorrelated with
the resonant feature. Methods to decrease the need for uncorrelated features have
been proposed. In reference [140], the auxiliary features are first decorrelated from
the cut features and then used in a classifier. The authors of reference [133] combine
density estimation to interpolate the CWoLa features from the sidebands into the
SR. A similar approach is followed in reference [143]. While the features no longer
contain information about the difference between SR and CR, there are sculpting
effects reducing performance when scanning the SR over the resonant feature. This
scan is necessary as the position of the resonance is a priori unknown. A possible
solution by changing to an uncorrelated latent space is discussed in reference [134].

We do not follow the bump-hunt approach in this work. Instead, we demonstrate
the improvement of a mono-jet search using CWoLa. The details on how to apply this method to such a search are outlined in the following.

6.2. CWoLa for mono-jet searches

The search strategy outlined in the previous section 6.1 is not tied to any particular search. We briefly discussed the application to bump hunts. However, the method is applicable whenever a SR and a CR can be constructed. In this section, we follow the ATLAS mono-jet search in reference [31] as an example going beyond bump hunts. In this search, events are selected to the SR if they have large missing transverse energy $E_T^{\text{miss}} > 200$ GeV, and a leading anti-$k_T$ jet $^{[38]}$ with radius $R = 0.4$, transverse momentum $p_T^{\text{jet}} > 150$ GeV and pseudo-rapidity $|\eta^{\text{jet}}| < 2.4$. On top of that, the events may contain at most three additional jets with $p_T^{\text{jet}} > 30$ GeV and $|\eta^{\text{jet}}| < 2.8$. An additional cut is applied on the angular separation between the missing energy and any of the jets, i.e. $\Delta \phi(p_T^{\text{jet}}, p_T^{\text{miss}}) > 0.4$. If the event contains identified leptons, it is vetoed. In reference [31], the ATLAS collaboration analyzes LHC Run 2 data with an integrated luminosity of 139 fb$^{-1}$ in several inclusive and exclusive $E_T^{\text{miss}}$ regions. We only consider the inclusive region IM1, with $E_T^{\text{miss}} > 250$ GeV. This was the most inclusive region in the previous ATLAS mono-jet search analyzing 36.1 fb$^{-1}$ of data, see reference [175]. This SR includes more than one million recorded events. The standard analysis consists of cut and count, i.e. cuts are applied and events are counted and compared to a background estimate from CRs. Additional information, e.g. from the substructure of jets, is not taken into account. Thus, CWoLa can boost this search’s sensitivity by including such additional information. As we expect a small signal fraction and need to train a classifier, CWoLa benefits from high statistics. With increasing amount of data, the method will become more and more applicable for higher $E_T^{\text{miss}}$ cuts as well, which currently show higher sensitivity for the signal we consider $^{[176]}$.

In the following, we discuss the SM backgrounds expected in the SR in section 6.2.1. We introduce our example signal in section 6.2.2. Finally, we go into further details of the CWoLa setup for this particular search in section 6.2.3.

6.2.1. SM backgrounds

The main contribution to the SM background (61%) comes from Z+jet production, with the Z boson decaying into neutrinos, thus staying invisible to the detector. The second largest contribution (31%) is W+jet production, with the W boson decaying leptonically and the detector not identifying the lepton. Subleading backgrounds are semi-leptonic $t\bar{t}$ production (3.5%) and diboson production with one invisibly decaying boson (2%).

All these backgrounds have corresponding control regions in reference [31]. The CR for the Z+jet background consists of Z+jet events in which the Z boson decays into charged leptons, which are detected and allow the reconstruction of the boson.
For the W boson background the CR contains those events for which the lepton is identified. The cuts defining the CR are the same as for the SR, replacing the missing transverse energy by a cut on the transverse recoil momentum $p_{T}^{\text{recoil}}$. The recoil momentum is given by $p_{T}^{\text{recoil}} = p_{T}^{l} + p_{T}^{\text{miss}}$, where $p_{T}^{l}$ is the transverse momentum of the identified leptons, i.e. one lepton for the W boson and two leptons for the Z boson decay. The decision whether an event belongs to the SR or the CR thus depends only on the decay of the boson and does not affect the jet dynamics. We can therefore use any feature describing the jet as one of the CWoLa features discussed in section 6.1.

We do not have access to LHC data, so we use Monte Carlo simulation in our analysis. This allows us to perform evaluations that take into account the true event labels. We simulate events with a center of mass energy of 13 TeV using MadGraph 5 [177] for the hard process, followed by Pythia 8.2 [178] for the parton shower, Delphes 3 [179] for a fast detector simulation, and FastJet [180] for jet-clustering. We use the default settings for all tools, except for changing the detector card of Delphes to the ATLAS Card.

The leading QCD jets for the two main background processes are generated through Initial State Radiation (ISR). Thus, QCD dynamics determines their structure, independent of the vector boson and its decay. Since the jet structure is then the same for the W+jet and the Z+jet background, we simulate only Z+jet production. Also, we only simulate the subsequent decay of the Z boson into neutrinos and use the corresponding jets in the SR as well as in the CR, as the jet properties do not depend on the Z boson decay. In fact, we checked that jets in events in which the Z boson decays into muons are indistinguishable from jets in events where the Z boson decays into neutrinos, using the same classifier employed in the rest of this section.

We additionally simplify the simulation of the additional backgrounds. For the $t\bar{t}$ background we only simulate the semi-leptonic channel, i.e. one top decays hadronically into three jets and the other one decays into a bottom quark and a leptonically decaying W boson. We apply a cut on the W boson transverse momentum to fulfill $p_{T}^{W} > 250$ GeV. Leading jets of these events fall into CR or SR depending on whether the charged lepton from the W boson decay is detected or not. For diboson events we simulate WZ production with an invisibly decaying Z boson and the W boson decaying into jets. The $p_{T} > 250$ GeV cut is set on the transverse momentum of the Z boson.

While we simplify the simulation, we do not expect a more accurate description to significantly change the qualitative characteristics of the leading jets in these events. In particular, we discuss the effects of mismodeling the CR in section 6.3.2, and we expect effects from more accurate simulation to be smaller compared to our observations in that section. Other, subdominant backgrounds stated in reference [31] are ignored in this work for simplicity.
6. Weakly supervised deep learning for mono-jet searches

6.2.2. New physics example: strongly interacting dark matter

We introduce an example signal to show the discovery potential of the CWoLa method. While the method itself is model agnostic, its performance depends on the signal that is hidden in the data. The dependence enters, because the classifier is trained on whatever signal might be present in the data. How different this signal is from the background determines how well the classifier can separate it, and hence how much (or how little) signal is needed for the classifier to be able to identify it. Thus, the numbers presented in section 6.3 cannot be seen as general limits, but only in the context of the model we use. The sensitivity to modified jet dynamics motivates the use of a strongly interacting DM model as an example. We use the model introduced in reference [176], which we also used in an earlier study applying supervised DL in reference [1]. For a more in depth discussion of this model and its phenomenology, we refer the reader to reference [181].

The model includes two quark flavors $q_d$, charged under an additional $SU(3)$ gauge symmetry, in a dark sector. Moreover, the dark quarks are charged under an additional, broken $U(1)'$ gauge group, introducing a heavy $Z'$ mediator. The coupling of the dark quarks to the mediator is given by $e_d$. The two flavors have opposite charge with respect to the $U(1)'$. The dark sector couples to the SM via a coupling of strength $g_q$ between $Z'$ and the SM quarks. This coupling of the dark sector to the SM allows the pair production of dark quarks at the LHC. After production, these dark quarks shower in the dark sector similar to QCD. They form bound states at some confinement scale $\Lambda_d$. The model only takes into account pseudoscalars $\pi^0_d$ and $\pi^\pm_d$ (dark pions) and vector mesons $\rho^0_d$ and $\rho^\pm_d$ (dark rho mesons). The indicated charges refer to the mesons’ charges under the $U(1)'$ symmetry and not to SM charges. Other dark mesons and baryons are neglected, as they are assumed to be sufficiently heavy. The dark pions are stable. For the charged $\pi^\pm_d$ this follows as long as no lighter particles charged under the $U(1)'$ exist. An imposed dark G-parity renders the $\pi^0_d$ stable. Introducing this G-parity requires an even number of quark flavors. In contrast to the dark pions, the $\rho^0_d$ mixes with the $Z'$. It thus couples to the SM through this mixing. The decay of the charged dark rho mesons into SM quarks is strongly suppressed. Therefore, the phenomenology of this model is dominated by the coupling of the $\rho^0_d$ to SM quarks. Moreover, the model assumes $m_{\rho_d} < 2m_{\pi_d}$, forbidding the decay of $\rho^0_d$ into dark pions.

LHC phenomenology of this model is dominated by on-shell $Z'$ production. Di-jet resonant searches are sensitive to the decay back to SM particles and constrain the branching fraction of the $Z'$ mediator into SM quarks [182]. The decay into dark quarks is less constrained, see reference [176] for limits arising from recasting the previous mono-jet search from ATLAS [175].

Dark quarks from $Z'$ decays shower and form mesons within the dark sector. Of these mesons, the $\rho^0_d$ decays shower and form mesons within the dark sector. Of these mesons, the $\rho^0_d$ decays shower and form mesons within the dark sector. Of these mesons, the $\rho^0_d$ decays shower and form mesons within the dark sector.
6.2. CWoLa for mono-jet searches

$(c\tau_{\rho_d} \gtrsim 1\text{ mm})$ or semi-visible jets $(c\tau_{\rho_d} \lesssim 1\text{ mm})$. In this work, we only consider semi-visible jets. They are called semi-visible, because a fraction of the original dark jet ends up visible in the detector. The fraction of visible energy is given by the fraction of $\rho_0^d$ produced. Assuming a small mass splitting between dark pions and dark rho mesons, the relative abundances of the dark mesons can be calculated by counting degrees of freedom. This results in an expectation value of the invisible fraction of $r_{\text{inv}} = 0.75$.

The main signature of this DM model are thus di-jet events with missing energy that is aligned to one of the jets, because statistical fluctuations in the invisible fraction lead to one jet with more missing energy. Hence, the typical signature does not end up in the mono-jet SR. Nevertheless, mono-jet searches are sensitive to this model, as the high invisible fraction also leads to events in which one of the jets stays completely invisible. For such events missing energy and the remaining jet point into opposite directions, i.e. $\Delta\Phi(p_{\text{jet}}^T, p_{\text{miss}}^T) \approx \pi$. Because this model does not produce isolated leptons, there is no contribution to any of the CRs designed for the various SM backgrounds. Thus, the fraction of signal in the CR is $f_{\text{CR}} = 0$. The choice of this model is only exemplary and the mono-jet search might not be the most sensitive analysis. In reference [176], a dedicated search for semi-visible jets, proposed in references [183, 184], applied to this model is also discussed. Moreover, there have been dedicated searches for semi-visible jets conducted by CMS in reference [185] and more recently by ATLAS in reference [186].

In this section, we set the parameters of the model as follows. The heavy mediator mass is set to $m_{Z'} = 2\text{ TeV}$, the coupling between SM quarks and the $Z'$ mediator to $g_q = 0.1$, the dark sector coupling to $c_d = 0.4$, and the dark quark masses to $m_{q_d} = 0.5\text{ GeV}$. The dark confinement scale and the dark meson masses are set to $\Lambda_d = m_{\pi_d} = m_{\rho_d} = 5\text{ GeV}$. Additionally, we set the decay length of $\rho_d^0$ to zero, so that we only obtain semi-visible jets and no emerging jets. The choice of parameters is not so much intended to form a viable candidate model for DM, but has shown to be an interesting benchmark for DL, e.g. in reference [6]. For exclusion limits on the model parameters, including the regime of emerging jets, the reader is referred to reference [181]. Differences between the dark jets and SM jets arise from the different running of the gauge couplings, the absence of heavy quarks in the shower and substructures emerging from dark meson decays.

To simulate this signal, we use the same toolchain as discussed in section 6.2.1. We add the Universal FeynRules Output (UFO) [187, 188] implementation of the model to MADGRAPH [177] for the simulation of the dark quark pair production. The shower in the dark sector, including hadronization and decay of the dark mesons, is performed within PYTHIA [178] using the hidden valley module [189]. After the detector simulation with DELPHES [179], we apply the cuts according to the IM1 signal region of the ATLAS search.
6.2.3. Applying CWoLa to the mono-jet search

We add the CWoLa setup described in section 6.1 to the mono-jet search with background and signal as described in the previous subsections. The CWoLa SR and CR are chosen as the SR and CR of the ATLAS search. A more detailed discussion on the construction and measurement of control samples for each background is provided in reference [31].

We use low-level features of the leading fat jet in each event as CWoLa features. As discussed in section 6.2.1, the leading background jet is independent of the assignment of an event to SR or CR. Therefore, we can use any feature describing this jet. We obtain the leading fat jet by reclustering the events into anti-$k_T$ jets [38] with $R = 0.8$. We limit ourselves to using the 40 jet constituents with highest transverse momentum.

For classification we employ a DGCNN [72] based on the ParticleNet [73] architecture. This is one of the most powerful classifiers in HEP, according to a study on top jet classification in reference [29], and it has shown superior performance on semi-visible jets in reference [1].

The architecture is described in more detail in section 2.3. We use three EdgeConv blocks with 64, 128 and 256 features, respectively. Each block contains three convolutions and uses 16 nearest neighbors for the graph construction. During training dropout of 0.1 is applied. Within the hidden layers of the network we apply LeakyReLU with $\alpha = 0.1$. The original implementation in reference [73] uses ReLU activation. However, we find that training is more stable and the network is sensitive to smaller signal fractions when applying LeakyReLU.

We implement the network in TENSORFLOW [12] version 2.6.0 with the built-in version of KERAS [106]. The CE is minimized using the Adam optimizer [50] with the default settings. We reduce the learning rate by a factor of ten if the performance does not improve for eight consecutive epochs. Training is stopped if the loss does not decrease for another four epochs or if 75 epochs are reached. We do not use a separate test set, as we want to exploit the full dataset available. Alternatively, one could use a cross-validation setup and train several classifiers. However, we find in section 6.3 that overfitting is not an issue for the large dataset we use.

As features describing each constituent we use the same seven features already introduced in reference [73]. These are \{$\Delta \eta, \Delta \phi, \log(p_T), \log(p_T/p_T^{\text{jet}}), \log(E), \log(E/E^{\text{jet}}), \Delta R$\} with $\Delta \eta = \eta - \eta^{\text{jet}}, \Delta \phi = \phi - \phi^{\text{jet}}$ and $\Delta R = \sqrt{\Delta \eta^2 + \Delta \phi^2}; \eta, \phi, p_T$ and $E$ refer to the rapidity, the azimuthal angle, the transverse momentum (in GeV), and the energy (in GeV) of the constituent, respectively. The graph constructed in the first EdgeConv block takes into account only $\Delta \eta$ and $\Delta \phi$ for the nearest neighbor determination. All other layers calculate Euclidean distances in the full hidden feature space.

The output of the classifier $s \in [0, 1]$ gives the probability of the input to belong to the SR. For small $f^{\text{SR}}$, the distribution of this score is expected to peak close to 0.5,
6.3. Results

as the classifier cannot distinguish most of the data. We choose a threshold $t$ and select those events with $s > t$ as anomalous, i.e. most signal like. We set $t$ such that one permille of the CR passes the cut. This efficiency in the CR is denoted as $\epsilon_{\text{CR}}$ and with our threshold selection $\epsilon_{\text{CR}} = 0.001$. This choice is to some extent arbitrary.

In a search with available labels, one could plot the significance improvement and choose the threshold according to the maximum there. However, this is not possible in the CWoLa setting applied to actual data. We consider 0.001 to be a good choice, because it balances the increasing statistical uncertainty with fewer selected events and the reduced systematic uncertainties (assuming them to be a constant fraction of the background estimate). We elaborate further on this choice in appendix B.

As the CR contains only background, it holds that $\epsilon_{\text{CR}} = \epsilon_{\text{SR}}^B = \epsilon_{\text{SR}}^B$, where the lower index denotes background and the upper index the region. The number of events that the classifier selects in the SR is given by $n_{\text{SR}} = \epsilon_{\text{SR}}^N \Lambda_{\text{SR}} = \epsilon_{\text{SR}}^{\text{B}} \Lambda_{\text{SR}} + \epsilon_{\text{SR}}^S \Lambda_{\text{SR}}^A$. Without any anomalous events in the SR, i.e. the null hypothesis assuming only SM events, the expected number of SR events is thus $n_{\text{exp}} = \epsilon_{\text{SR}}^{\text{B}} \Lambda_{\text{SR}} + \epsilon_{\text{SR}}^{\text{CR}} \Lambda_{\text{SR}}$, since $N_A = 0$. If the actual number of events selected from the SR ($n_{\text{SR}}$) differs from the expected number by more than just statistical fluctuations, the null hypothesis can be rejected in the usual way.

Motivated by the order of magnitude of events in the SR IM1 of the ATLAS search, we choose $N_{\text{SR}} = N_{\text{CR}} = 10^6$ as our default. By applying the classifier threshold we set $n_{\text{CR}} = 1000$. This number is fixed by our threshold choice $t$, however $n_{\text{CR}}$ for the same $t$ would vary for independent datasets with an uncertainty of $\sqrt{n_{\text{CR}}}$.

Additionally, there is the usual Poisson error on the number of selected events in the SR. The uncertainty on $n_{\text{exp}}$ is then given by adding the two uncertainties in quadrature resulting in $\sqrt{(\sigma_{\text{CR}})^2 + (\sigma_{\text{SR}})^2} = \sqrt{2} \sigma_{\text{CR}}$. Since $n_{\text{exp}} = n_{\text{CR}} = 1000$, a $5\sigma$ statistical excess needs at least $5 \times \sqrt{2} \sigma_{\text{CR}} n_{\text{exp}} = 224$ additional events.

In general, one can change the threshold and maximize the SR excess. However, we avoid the discussion about the look-elsewhere effect by not following this idea here. We argue in the following sections that a discovery using the CWoLa method is not so much driven by statistical considerations but rather by an iterative improvement of understanding the quality of the CR.

6.3. Results

We first discuss a proof of principle in section 6.3.1. For that, we take into account only the most dominant background and a perfect CR, i.e. the background in CR and SR follows the same distribution. We then introduce the additional backgrounds and investigate the CWoLa method for imperfections in the CR in section 6.3.2.

6.3.1. Proof of principle

As a first step, we take the CWoLa assumption to be exactly true. We do so by using the same $Z+$jet simulation setup for SR and CR, see section 6.2.1. We additionally neglect the other backgrounds and only consider $Z+$jet. This accounts already for
6. Weakly supervised deep learning for mono-jet searches

<table>
<thead>
<tr>
<th>$f^\text{SR}$</th>
<th>$n^\text{SR}$</th>
<th>$n^\text{SR}_A$</th>
<th>$n^\text{SR}_B$</th>
<th>$(n^\text{SR} - n^\text{SR}<em>\text{exp})/\sqrt{2n^\text{SR}</em>\text{exp}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>1048</td>
<td>0</td>
<td>1048</td>
<td>1.07</td>
</tr>
<tr>
<td>0.2%</td>
<td>1065</td>
<td>47</td>
<td>1018</td>
<td>1.45</td>
</tr>
<tr>
<td>0.4%</td>
<td>1107</td>
<td>100</td>
<td>1007</td>
<td>2.39</td>
</tr>
<tr>
<td>0.5%</td>
<td>1175</td>
<td>184</td>
<td>991</td>
<td>3.91</td>
</tr>
<tr>
<td>0.6%</td>
<td>1306</td>
<td>247</td>
<td>1059</td>
<td>6.84</td>
</tr>
<tr>
<td>0.7%</td>
<td>1389</td>
<td>367</td>
<td>1022</td>
<td>8.70</td>
</tr>
<tr>
<td>0.8%</td>
<td>1500</td>
<td>419</td>
<td>1081</td>
<td>11.18</td>
</tr>
<tr>
<td>1%</td>
<td>1666</td>
<td>625</td>
<td>1041</td>
<td>14.89</td>
</tr>
<tr>
<td>2%</td>
<td>2357</td>
<td>1392</td>
<td>965</td>
<td>30.34</td>
</tr>
<tr>
<td>4%</td>
<td>4182</td>
<td>3269</td>
<td>913</td>
<td>71.15</td>
</tr>
</tbody>
</table>

Table 6.1: Number of events $n^\text{SR}$ selected from the SR for $N^\text{CR} = N^\text{SR} = 10^6$ for varying signal fractions $f^\text{SR}$. We add the number of selected anomalous and background events ($n^\text{SR}_A$ and $n^\text{SR}_B$), which are not known for experimental data and show an estimate of the statistical significance of a possible discovery.

92% of the backgrounds, because the jet dynamics in W+jet (31%) and Z+jet (61%) events should be the same, as we discussed in section 6.2.1. These idealized conditions are used to explore the principal properties of the method. The fraction of signal events in the CR is $f^\text{CR} = 0$. We use one million events in both CR and SR. For the SR we keep the fraction of anomalous events as a free parameter, such that the number of anomalous events can be written as $N^\text{A} = N^\text{SR}_A = f^\text{SR}N^\text{SR}$.

An important aspect to note is that the signal events can also contain ISR jets. Such an ISR jet can be the leading jet of the event and thus be selected for our CWoLa features. While it is part of an anomalous event, the jet itself is a pure QCD jet, indistinguishable from the background. In a supervised setup, one would match the final jet chosen from an event to an initial parton and remove those jets not originating from dark quarks from the data. This is for example done for top jets in the top tagging benchmark dataset used in reference [29]. However, this is not possible in our setup, as we want it to be directly applicable to experimental data, where such a matching is impossible. Thus, the fraction of anomalous jets as opposed to anomalous events is somewhat lower than $f^\text{SR}$.

We present the number of selected events from the SR for various $f^\text{SR}$ in table 6.1. The results presented take as score $s$ the mean of five classifiers. We do so to be less affected by fluctuations in the training that can occur in particular for small signal fractions. This introduces improvement by the ensembling effect already discussed in section 5. The first and probably one of the most important tests to perform is the null test. We need to ensure that we do not claim to find a signal if no signal is present in the data. As we can see from the first row of table 6.1, the CWoLa
method does not introduce a fake signal. The selected number of events is within the expected statistical fluctuations. An excess in this setup with the same background distribution in SR and CR can only occur when overfitting the data, since we use the same data for training and testing (see section 6.2.3). Thus, the dataset is sufficiently large to not observe overfitting with the ParticleNet classifier. For an imperfect CR or smaller amounts of data, the potentially systematically higher number of selected events in the SR needs to be estimated, e.g. by independent CRs, and included as uncertainty in the significance calculation.

Introducing sufficient signal, e.g. four percent, the CWoLa setup finds an excess with 71σ statistical significance beyond doubt. Even for \( f_{SR} = 0.01 \), which corresponds to \( N_{A}^{SR} = 10000 \), an excess of 666 events is found, corresponding to almost 15σ statistical significance. The ATLAS mono-jet search [31] provides a model independent upper limit on BSM events in IM1 of ∼40k at 95% confidence level. The sensitivity in this idealized setup thus reaches far beyond that of the original search. In table 6.1, we also present the composition of the selected SR events, demonstrating that the excesses are indeed due to signal selected in addition to the expected background. This separation is not possible in an experimental setting. One would investigate the selected jets to find the difference between SR and CR events. By construction, the method directly provides the background estimate from the selected events in the CR, to which one would then compare the selected events from the SR. Finding a pattern in the excess can then either be explained by a SM background that was not (sufficiently) taken into account in the CR or by some BSM signal. The follow-up analysis of selected events provides the properties of this potential signal that hint towards suitable new physics models.

The measured excess drops rapidly for signal fractions below 0.6%. At that point the classifier is no longer able to pick up the features distinguishing signal from background. The problem is that there are only 6000 signal events in a total of two million events. This goes beyond an imbalanced training on 6000 signal events and 1M background events in a supervised setup. Since every event by itself looks noisy, the classifier needs to identify patterns emerging from a collection of events. If now there are too few signal events for the signal patterns to stick out of the background noise, the classifier will no longer be able to find those. Reducing the dimensionality of the inputs can permit finding much smaller total numbers of signal, as e.g. shown in reference [133]. However, this selection introduces model dependence, since selected features can be better or worse to distinguish different signals. In the supervised case, one could help the classifier by weighting or augmenting the signal, see the discussion in section 5.4. This is not possible for our setup, because neither labels nor the signal fraction are available.

In figure 6.2, we show the distribution of the classifier score \( s \) for \( f_{SR} = 1\% \) (left) and \( f_{SR} = 0.5\% \) (right). We show three distributions. First, the CR, which consists only of background, i.e. Z+jet events. Second, the Z+jet background in the SR. Third, the signal in the SR. The first thing to notice is the strong peak at \( s = 0.5 \). This peak shows that for most of the data it is impossible to sort it into either SR or CR. There is a tail towards higher \( s \), i.e. higher probabilities for events in the SR,
6. Weakly supervised deep learning for mono-jet searches

Figure 6.2: Distribution of the classifier score $s$ for $f^{SR} = 1\%$ (left) and $f^{SR} = 0.5\%$ (right). The CR data, consisting only of Z+jet background, is shown in black. The SR curve is separated into a signal (blue) and background (red) contribution. The threshold $t$ applied to keep 1000 events of the CR is indicated by the gray dotted line.

but no tail to lower scores. A value considerably lower than 0.5 cannot be obtained, as there is no subclass of the background that is not also present in the SR. If there were scores significantly below 0.5, this would be a sign of overfitting. Each jet by itself is unique and only in one of the two regions. A model with sufficient capacity could learn these individual jet structures and their assignment to the regions. The Z+jet distributions in the two regions nicely agree, which we expect, as they follow the same distributions in their features. We indicate the threshold selected on the CR data. Beyond this threshold, the signal over background ratio in the SR is strongly enhanced. Summing the bins from the threshold to one results in the numbers presented in table 6.1.

The right panel of figure 6.2 shows the same distributions, but for $f^{SR} = 0.5\%$. At first the left edge looks much less steep. However, this is just due to the scaling of the x-axis, which goes beyond 0.8 for $f^{SR} = 1\%$ and only up to 0.55 in the right panel. The classifier is not able to pull the signal to higher scores as it would be punished by more CR background events that look signal like and would also get higher scores. The limited signal statistics does not allow the classifier to find signal specific features.

We want to know, how the performance depends, on the one hand, on the number of signal events, and on the other hand, on the fraction of signal events. We train on different dataset sizes, going from 250k events per region to 500k and finally to our default of 1M events. In table 6.2, we show the number of selected events in the three scenarios with a fixed number of anomalous events in the SR of $N_A = 10k$. We find that the number of selected anomalous events is quite stable, only reducing slightly, even though the signal fraction is reduced by a factor of four. The decreasing
6.3. Results

\[
N_n^\text{SR} = \frac{(n_n^\text{SR} - n_n^\text{SRexp})}{\sqrt{\sum 2n_n^\text{SRexp}}}
\]

<table>
<thead>
<tr>
<th>(N^\text{SR} )</th>
<th>(n_n^\text{SRexp} )</th>
<th>(n_n^\text{SR} )</th>
<th>(n_A^\text{SR} )</th>
<th>(n_B^\text{SR} )</th>
<th>( (n_n^\text{SR} - n_n^\text{SRexp})/\sqrt{2n_n^\text{SRexp}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>250k</td>
<td>250</td>
<td>916</td>
<td>686</td>
<td>230</td>
<td>29.78</td>
</tr>
<tr>
<td>500k</td>
<td>500</td>
<td>1113</td>
<td>660</td>
<td>453</td>
<td>19.38</td>
</tr>
<tr>
<td>1000k</td>
<td>1000</td>
<td>1666</td>
<td>625</td>
<td>1041</td>
<td>14.89</td>
</tr>
</tbody>
</table>

Table 6.2: Same as table 6.1, but fixing the number of anomalous events \(N_A^\text{SR} = 10k\) and varying the number of SR (and CR) events. We add the number of expected SR events \(n_n^\text{SRexp}\) for clarity.

<table>
<thead>
<tr>
<th>(N^\text{SR} )</th>
<th>(n_n^\text{SRexp} )</th>
<th>(n_n^\text{SR} )</th>
<th>(n_A^\text{SR} )</th>
<th>(n_B^\text{SR} )</th>
<th>( (n_n^\text{SR} - n_n^\text{SRexp})/\sqrt{2n_n^\text{SRexp}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>250k</td>
<td>250</td>
<td>362</td>
<td>126</td>
<td>236</td>
<td>5.01</td>
</tr>
<tr>
<td>500k</td>
<td>500</td>
<td>680</td>
<td>258</td>
<td>422</td>
<td>5.69</td>
</tr>
<tr>
<td>1000k</td>
<td>1000</td>
<td>1666</td>
<td>625</td>
<td>1041</td>
<td>14.89</td>
</tr>
</tbody>
</table>

Table 6.3: Same as table 6.2, but for a fixed fraction \(f_n^\text{SR} = 0.01\) of anomalous events.

significance is due to the larger number of expected events resulting from the fixed threshold at \(\epsilon^\text{CR} = 0.001\). Alternatively, we fix the fraction \(f_n^\text{SR} = 0.01\) and change the dataset size. The corresponding numbers are shown in table 6.3. We see that the performance improves with more data and thus a higher absolute number of signal events. This is very promising in light of the high-luminosity phase of the LHC.

These findings are underscored by the ROC curves for signal vs. background separation shown in figure 6.3. We show the median ROC curve together with the band between best and worst performance of the five ParticleNet runs. To evaluate the ROC curve, we test the classifier trained in the CWoLa setup on 100k signal and background jets. Note that the median ROC curve does not correspond to the ROC curve obtained by taking the mean prediction, see e.g. section 5. The left panel shows how the classifier trained in the CWoLa setting on different region sizes with constant \(N_A = 10k\) performs on the task of tagging jets from DM events against jets from \(Z++\text{jet}\) events. We see that the different curves lie on top of each other, i.e. the classifiers perform equally well. There is a minor decrease in performance for larger region sizes, as the signal has to be extracted from a larger amount of uninformative background noise. However, this effect is much smaller than the effect shown in the right panel. Here, the ROC curves for the same region sizes as before, but with constant \(f_n^\text{SR} = 1\%\) are shown. We first notice that the classification performance is increasing, as already seen in table 6.3. Moreover, the stability indicated by the colored bands improves.

We add the ROC curve from a supervised classifier for comparison. For the supervised setup, we train on 100k signal and 100k background jets. The solid supervised curve corresponds to using the same labels as for the CWoLa curves shown, i.e. there
are events labeled as signal, for which the leading jet does not originate from a dark quark. We thus see that the performance of the CWoLa classifier is actually close to the ideal classifier, approximated by the supervised setup. Nevertheless, there is room for improvement. In particular at the threshold we use ($\epsilon_B^{-1} = 10^3$), almost a factor of two in signal efficiency can still be gained. To be explicit, the CWoLa setting with 1M events and $f_{SR} = 1\%$ achieves $\epsilon_S(\epsilon_B = 0.001) = 0.054$ and the supervised classifier 0.092. This gap closes for increasing $f_{SR}$, e.g. for $f_{SR} = 4\%$ the CWoLa setup achieves already 0.077 signal efficiency at our working point.

The dashed, green ROC curve in figure 6.3 shows the performance on those semi-visible events, in which the leading fat jet is matched to an initial dark quark, i.e. the parton level dark quark falls within the jet radius of the selected fat jet. There is an improved performance in particular at intermediate signal efficiencies. This curve corresponds to the upper limit on tagging semi-visible jet events. It takes into account that some of the events are actually background considering only the information available for the classifier, i.e. the leading jet.

For the presented studies we use the same number of events in SR and CR. However, in practice the branching fractions define the amount of data collected in the two regions. For the Z boson decay, a smaller fraction decays into charged leptons and the CR would thus be smaller. The branching ratio into neutrinos is $\sim 20\%$ and into charged leptons $\sim 10\%$ [190]. Thus, we expect approximately twice as many events in the SR compared to the CR. We performed the same tests with an unbalanced dataset, with a CR half the size of the SR. Weighting the CR during training, we find no significant loss of discovery power, except for the increased statistical uncertainty from the threshold estimate. We show the ROC curves comparing the
6.3. Results

Table 6.4: Number of events selected from the different background classes in the SR and CR for different compositions of the CR. There are no anomalous events ($f_{SR} = 0$) and we set $r_{t\bar{t}}^{SR} = 3.5\%$ and $r_{VV}^{SR} = 2\%$.

<table>
<thead>
<tr>
<th>$r_{t\bar{t}}^{SR}$</th>
<th>$r_{VV}^{SR}$</th>
<th>$n_{Z+\text{jet}}^{SR}$</th>
<th>$n_{t\bar{t}}^{SR}$</th>
<th>$n_{VV}^{CR}$</th>
<th>$n_{Z+\text{jet}}^{CR}$</th>
<th>$n_{t\bar{t}}^{CR}$</th>
<th>$n_{VV}^{SR}$</th>
<th>$n_{SR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>0%</td>
<td>1000</td>
<td>0</td>
<td>1053</td>
<td>1127</td>
<td>2043</td>
<td>4223</td>
<td></td>
</tr>
<tr>
<td>1.75%</td>
<td>1.0%</td>
<td>541</td>
<td>247</td>
<td>212</td>
<td>581</td>
<td>557</td>
<td>420</td>
<td>1558</td>
</tr>
<tr>
<td>2.80%</td>
<td>1.6%</td>
<td>980</td>
<td>3</td>
<td>17</td>
<td>1065</td>
<td>6</td>
<td>17</td>
<td>1088</td>
</tr>
<tr>
<td>3.15%</td>
<td>1.8%</td>
<td>957</td>
<td>24</td>
<td>19</td>
<td>962</td>
<td>26</td>
<td>32</td>
<td>1020</td>
</tr>
<tr>
<td>3.50%</td>
<td>2.0%</td>
<td>823</td>
<td>160</td>
<td>17</td>
<td>793</td>
<td>175</td>
<td>21</td>
<td>989</td>
</tr>
<tr>
<td>5.00%</td>
<td>3.0%</td>
<td>966</td>
<td>18</td>
<td>16</td>
<td>960</td>
<td>22</td>
<td>3</td>
<td>985</td>
</tr>
<tr>
<td>3.50%</td>
<td>1.6%</td>
<td>903</td>
<td>89</td>
<td>8</td>
<td>821</td>
<td>93</td>
<td>26</td>
<td>940</td>
</tr>
<tr>
<td>2.80%</td>
<td>2.0%</td>
<td>983</td>
<td>5</td>
<td>12</td>
<td>996</td>
<td>7</td>
<td>12</td>
<td>1015</td>
</tr>
</tbody>
</table>

After demonstrating the general sensitivity of the CWoLa method in an idealized setting in section 6.3.1, we now scrutinize how a more realistic background and different compositions of SR and CR influence the method. We add the $t\bar{t}$ and diboson backgrounds discussed in section 6.2.1 according to their corresponding fractions to the SR, i.e. $r_{t\bar{t}}^{SR} = N_{t\bar{t}}^{SR}/N_{SR}^{SR} = 3.5\%$ and $r_{VV}^{SR} = N_{VV}^{SR}/N_{SR}^{SR} = 2\%$. We use $r$ as opposed to $f$ to highlight that here we are not dealing with signal but additional backgrounds. The lower index indicates the particular background and the upper index the corresponding region. We evaluate different compositions of the CR in the following.

6.3.2. Reality checks

Starting without our signal, i.e. $f_{SR} = 0$, we show the number of selected events per background category in the CR and SR for different compositions of the CR in table 6.4. If we ignore the additional backgrounds in the CR, i.e. setting $r_{t\bar{t}}^{CR} = r_{VV}^{CR} = 0$, the CWoLa method finds a large excess of events in the SR. The additional events are all coming from the missing backgrounds as the main background shows the expected number of events in the SR, as before e.g. in table 6.1. The classifier identifies jet substructures related to $t\bar{t}$ and diboson events. We showed in reference [1] that the ParticleNet classifier performs significantly better on the classification of top jets than on the identification of semi-visible jets. Here however, this seems not to be the case, as only $\sim 1100$ top jets are selected. This number needs to be compared to the $f_{SR} = 4\%$ row in table 6.1, where more than 3000 additional semi-visible jet events have been selected. The reason for the worse identification of $t\bar{t}$ events in this task is that we are not only looking at boosted hadronically decaying top jets, where all decay products of the top are contained in the leading fat jet. Instead, we...
have a comparably low $p_T^{\text{jet}}$ threshold of 150 GeV, i.e. top jets that are not highly boosted at all. The leading fat jet in $t\bar{t}$ events can thus be initiated by any of the individual partons in the event or combinations of those. For individual partons, no obvious distinction from the Z+jet ISR background is possible. The diboson background behaves slightly differently. Here, we have a 2% contribution to the SR, so we compare to the $f_{\text{SR}}^{f} = 2\%$ column in table 6.1. While $\sim 1400$ semi-visible jets were extracted for this signal fraction, more than 2000 diboson events are selected. The W boson in our diboson simulation is boosted, as the recoiling Z boson must have a $p_T > 250$ GeV. Thus, the decay products of the W boson are likely to end up in the selected fat jet and form a two-pronged substructure. This structure differs more significantly from the ISR background and can thus be tagged better than the semi-visible jet structures.

The naive example highlights that more selected events in the SR can have two causes: new physics or mismodeling of the SR background in the CR. Generally, input from theory and Monte Carlo is required to build proper CRs. The leading jets for the different backgrounds need to be measured from different event topologies and then combined to match expected rates in the SR. We thus ask the question of how well the CR needs to be modeled. For perfect modeling, i.e. $r^{\text{CR}}_{t\bar{t}} = 3.5\%$ and $r^{\text{CR}}_{VV} = 2\%$, we find no statistically significant deviation from the expectation in table 6.4. Moreover, for underestimating the additional backgrounds in the CR by 10%, i.e. $r^{\text{CR}}_{t\bar{t}} = 3.15\%$ and $r^{\text{CR}}_{VV} = 1.8\%$, we still find agreement with the expected number of selected events. Even for a relative mismodeling of 20% the deviations are not very significant. Finally, an overestimation of the additional backgrounds does not have strong effects on the number of events selected in the SR.

We conclude that the modeling of the absolute background numbers needs to be accurate at the percent level. This is also true for standard searches already. For example, the mono-jet search by ATLAS considered here states a 1.2% systematics dominated uncertainty [31]. However, the modeling does not need to be perfect. A relative understanding at the level of 10% is sufficient. Especially, an overestimation of small background components does not harm the CWoLa setup significantly.

Overall, we suggest an iterative, self-correcting setup. Finding an excess of events in the SR should trigger efforts to better understand the CR construction. The additional events from the SR provide hints towards what background might be mismodeled. The procedure is then repeated with a refined CR. Finally, either no excess is observed or one turns to new physics interpretations, which again can be guided by the investigation of the selected SR events.

We now reintroduce the semi-visible jet signal with a 1% contribution to the SR, keeping the additional contributions from $t\bar{t}$ and diboson events. The selected numbers of events and their composition are shown in table 6.5 for varying $t\bar{t}$ and diboson rates in the CR. First, the performance of the idealized scenario presented in section 6.3.1 is recovered when modeling the additional backgrounds correctly. Moreover, also slight mismodeling does not degrade performance drastically. We see that
6.3. Results

<table>
<thead>
<tr>
<th>$r_{\bar{t}t}^{CR}$</th>
<th>$r_{VV}^{CR}$</th>
<th>$n_{Z+\text{jet}}^{CR}$</th>
<th>$n_{\bar{t}t}^{CR}$</th>
<th>$n_{Z+\text{jet}}^{SR}$</th>
<th>$n_{\bar{t}t}^{SR}$</th>
<th>$n_{VV}^{SR}$</th>
<th>$n_{A}^{SR}$</th>
<th>$n_{SR}^{SR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>0%</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1089</td>
<td>1245</td>
<td>1826</td>
<td>223</td>
</tr>
<tr>
<td>2.80%</td>
<td>1.6%</td>
<td>876</td>
<td>108</td>
<td>16</td>
<td>838</td>
<td>134</td>
<td>37</td>
<td>456</td>
</tr>
<tr>
<td>3.50%</td>
<td>2.0%</td>
<td>963</td>
<td>23</td>
<td>14</td>
<td>996</td>
<td>25</td>
<td>32</td>
<td>633</td>
</tr>
<tr>
<td>5.00%</td>
<td>3.0%</td>
<td>971</td>
<td>12</td>
<td>17</td>
<td>1034</td>
<td>8</td>
<td>10</td>
<td>575</td>
</tr>
</tbody>
</table>

Table 6.5: Number of events selected by the classifier from the different classes in the SR and CR for different compositions of the CR. There is a fixed fraction of anomalous events ($f_{SR} = 0.01$) and we set $r_{\bar{t}t}^{SR} = 3.5\%$ and $r_{VV}^{SR} = 2\%$.

...
6. Weakly supervised deep learning for mono-jet searches

Figure 6.4: Distribution of classifier scores \( s \) in the SR for the Z+jet (red) background and the additional \( t\bar{t} \) (orange) and diboson (green) backgrounds in the SR. The four panels show the distribution for different \( r_{tt}^{CR} \) and \( r_{VV}^{CR} \). Top left: \( r_{tt}^{CR} = 3.5\% \) and \( r_{VV}^{CR} = 2\% \), matching the SR rates. Top right: \( r_{tt}^{CR} = 0 \) and \( r_{VV}^{CR} = 0 \), ignoring the additional backgrounds. Bottom left: \( r_{tt}^{CR} = 5\% \) and \( r_{VV}^{CR} = 3\% \), overestimating the additional backgrounds. Bottom right: \( r_{tt}^{CR} = 2.8\% \) and \( r_{VV}^{CR} = 1.6\% \), underestimating the additional backgrounds.

6.4. Summary

In this section, we have shown that the CWoLa method can significantly boost the sensitivity of already existing searches. We emphasize that this method works without any signal model specific assumptions if the signal contributes to the mono-jet SR. As the search has been conducted by ATLAS already, the data is available and an implementation of CWoLa should be straightforward.

We used Monte Carlo simulated data and a well-motivated new physics model, including a strongly interacting dark sector, to show that signs of new physics contributing less than 1% to inclusive signal regions could be discovered with high significance. The traditional search in reference [31] sets the model independent limit
6.4. Summary

to ~3% at 95% confidence level. Previous searches for the particular model that we have considered in this work using supervised [1] or unsupervised DL [6] have shown that this semi-visible jet signature is rather challenging to find. A comparison between semi-visible jet tagging and tagging of boosted hadronically decaying top jets is part of reference [1]. Signal models producing more distinct jet substructures could potentially be found with even smaller rates.

While the CWoLa method offers a useful discovery tool, its weakly supervised nature means that it typically yields inferior performance compared to fully supervised methods for a given task. We saw, however, that in the limit of sufficient data CWoLa asymptotically closes the gap to supervised performance. The method based purely on data is not able to provide general exclusion limits. It can only provide limits through the use of benchmarks based on Monte Carlo simulation. However, the use of Monte Carlo benchmarks can be challenging, because an estimate of the effects on the classifier when mixing Monte Carlo and data becomes necessary. Nevertheless, the CWoLa method has strengths, including its model-agnostic setup and direct applicability to data. By comparing signal and control regions, the method allows for an iterative understanding of the relationships between these regions, which can be valuable in data analysis. This is a key advantage over fully unsupervised methods such as AEs, as it is not obvious how to make statistically meaningful discoveries for those methods [191].

We have scrutinized the behavior of the CWoLa method with increasing amount of data. For a constant fraction of signal events, which one would expect for a given new physics model, the method benefits from more data. This is particularly good news in view of the upcoming high luminosity phase of the LHC, as traditional searches might already be systematics limited, e.g. the ATLAS search considered in this work. Moreover, we have investigated how well the CR has to depict the SR for the method to work. We find that an understanding of the SR backgrounds at the percent level is needed, however a relative deviation of ~10% in the estimate does not spoil the performance. This is especially true if the mismodeling takes the direction of overestimating backgrounds in the CR.

We conclude that BSM searches assisted by the CWoLa method and reanalyses of data that is already collected in signal and control regions hold great promise.

---

4Corresponding to the exclusion of 40k additional events with an estimated background of ~1.3M events.
7. Learning the language of jets with transformers

In this section, we turn to unsupervised learning and investigate density estimation in high-dimensional jet data. We do so by building an analogy between natural language and jets. This allows us to apply a transformer-encoder network for autoregressive density estimation. The content of this section and the presented results closely follow reference [7].

The data preprocessing and the way we employ the transformer architecture are described in section 7.1. We present the results on density estimation in section 7.2. The quality of the density estimate in terms of jets generated by sampling from the learned distribution is discussed in section 7.3. Finally, we summarize our findings in section 7.4.

7.1. Setup

In this section, we present our setup. We start by introducing the dataset and how we modify the data to look more like natural language in section 7.1.1. The architecture and training procedure used for density estimation as well as the sampling procedure are described in section 7.1.2.

7.1.1. Translating the top tagging benchmark dataset to natural language

We use the top tagging benchmark dataset to show and evaluate the performance of our method. The dataset is publicly available in reference [192]. It provides hadronically decaying top jets as signal. The background in this dataset consists of light QCD jets, i.e. jets originating from up, down, strange, charm quarks, and gluons.

From a physics perspective, the task of top tagging is motivated by the importance of the top quark in new physics searches. Since it has the strongest coupling to the SM Higgs boson, the top quark provides a portal to study the properties of the Higgs. Many BSM scenarios predict modified properties of the Higgs, which could be measured in deviations from the SM in interactions involving the top quark, see e.g. reference [193] for a review of exotic Higgs decays.

The tops included in the dataset decay hadronically, i.e. the top decay into a W boson and a b quark is followed by the W boson decaying into two quarks. This gives rise to a distinct, three pronged structure from the three quarks. The jets contained in this dataset are highly boosted and clustered with large jet radius, so that all the decay products of the top quark are contained in one so-called fat jet. Traditional taggers based on high-level observables take advantage of this jet substructure. They use for example subjettiness ratios [41], which indicate the extent to which a jet can be described by a given number of subjets. In addition, kinematic properties arising from the W boson mass and the top mass can be used in these taggers [194]. It has been shown in reference [29] that DL based taggers using low-level information
7.1. Setup

as input outperform those traditional taggers on high-level observables. Moreover, this dataset has been used not only for benchmarking supervised classification performance, but also unsupervised anomaly detection, see e.g. references [3, 47], and reference [66] for a similar dataset with slightly different cuts.

The dataset is simulated using **Pythia 8** [178] for proton-proton collisions at a center of mass energy of 14 TeV. The simulation is simplified by ignoring possible multiple interactions and pile-up. A fast detector simulation is performed with **DELPHES** [179] using the default ATLAS detector card. Jets are clustered with **FastJet** [180] using the anti-\(k_T\) algorithm [38] with a large jet radius of \(R = 0.8\). The radius is set to a large value to capture all decay products of the top in a single jet. To ensure that a top fat jet contains all decay products, the initial, parton level top as well as all parton level decay products must lie within the jet radius of the detector level top jets. Additional cuts are set on the transverse momentum of the jets and the pseudo rapidity such that each jet fulfills \(p_T^{\text{jet}} \in [550 \text{ GeV}, 650 \text{ GeV}]\) and \(|\eta^{\text{jet}}| < 2\). For each jet the four momenta of the 200 constituents with highest \(p_T\) are stored. The momenta are extracted from **DELPHES** using the energy-flow algorithm. The full dataset comprises 1 million top jets and 1 million light QCD jets. They are split into a training, validation and test set containing the same amount of QCD and top jets. The sets consist of 1.2M, 400k and 400k jets respectively.

For our setup, we restrict ourselves to the leading 50 constituents during training of the density estimator. During inference and sampling we consider up to 100 constituents. We represent a jet by a series of its constituents ordered by decreasing transverse momentum \(p_T\). The investigation of alternative orderings, e.g. taking into account some clustering history as suggested in references [32, 172], is left for future work. Each constituent \(i\) is represented by the tuple \((p_{T,i}, \Delta \eta_i, \Delta \phi_i)\), where \(\Delta \eta_i = \eta_i - \eta^{\text{jet}}\) and \(\Delta \phi_i = \phi_i - \phi^{\text{jet}}\). To get a dictionary of possible particles in analogy to words in natural language, we discretize by binning. The angular variables \(\Delta \eta\) and \(\Delta \phi\) are binned into 29 equidistant bins in the range \([-0.8, 0.8]\). The range is set according to the jet radius, since only few constituents are expected to be beyond this range in anti-\(k_T\) jets. This binning is similar to the binning done when representing jets as images, see e.g. reference [46]. The transverse momentum of the particles is divided into 39 bins. Those are taken equidistant in log-space to account for the steeply falling \(p_T\) distribution. We set the range \([p_{T_{\text{min}}}, p_{T_{\text{max}}}]\) such that 99.9% of the constituents in the QCD training set fulfill \(p_{T,i} > p_{T_{\text{min}}}\). The maximum is set to the maximum \(p_{T,i}\) in the same dataset. Finally, we add an overflow and an underflow bin for all features. Each constituent is then described by three integers denoting the respective bins \(p_T \in [0, 40], \Delta \eta \in [0, 30], \text{ and } \Delta \phi \in [0, 30]\). To allow the application of a density estimator trained on QCD jets to top jets and vice versa, we use the same binning for the discretization of both jet classes. That is, we do not set the \(p_T\) bin range differently for top jets, even though the \(p_{T,i}\) distribution is different.

We check whether this discretization has significant impact on the information contained within the jet description by comparing the classification performance of
7. Learning the language of jets with transformers

a supervised tagger on continuous and discrete data. We employ the ParticleNet classifier introduced in reference [73] as it has shown the best performance on the top tagging benchmark in reference [29]. We use the implementation of the network as well as the training procedure discussed in section 6. ParticleNet originally uses seven features for each constituent \(i\), namely \(\Delta \eta_i\), \(\Delta \phi_i\), \(\log(p_{T,i})\), \(\log(p_{T,i}/p_{jet}^T)\), \(\log(E_i)\), \(\log(E_{jet})\), and \(\Delta R\). The choice of constituent features we use for density estimation does not allow the reconstruction of features related to the jet and constituent energy in the laboratory frame, because the relative angular coordinates do not allow the reconstruction of the jet direction. We thus reduce the number of features used for classification by removing the two energy related features. The resulting ROC curves for ParticleNet on continuous (original) and the discrete data are shown in the left panel of figure 7.1. Their agreement shows that the discretization does not destroy relevant information for this task. Moreover, removing the energy information does not harm performance, as the ROC curves also agree well with the ParticleNet curves shown in reference [29]. Note that this test is not conclusive for every possible classification task. A direct comparison by applying a classifier between discrete and continuous data is however not possible. Classification of the discretized data against the original data is trivial by searching for the discrete values. A classifier trained on that tasks achieves perfect separation. In the future, we intent to go to larger datasets such as the JetClass dataset [195], which contains 100M jets of ten different classes. We plan on further evaluating the information loss of discretization on other classification tasks within that dataset.

We want our density estimator to be flexible with respect to the number of constituents, both during inference of the density and during sampling. We therefore introduce a stop token to our dictionary of particles. This is in analogy to an end of sentence token in NLP. When using 50 constituents during training, we add the
7.1. Setup

Including the stop token, we have $41 \times 31 \times 31 + 1 = 39402$ different particles in our dictionary. However, not all of them are actually realized in the training set. In the QCD jet training set, $\sim 35\%$ of the possible states are not populated. The ratio of populated particle states is even lower when looking at a particular position in the jet. This distribution is shown in figure 7.1 on the right for different training set sizes, i.e. different overall numbers of jets. At the peak only $\sim 35\%$ of the possible particles are present. Thus, when predicting a particle, the network needs to set the probability for many particles to zero. This is not possible, as the softmax activation used to obtain probabilities can only result in zero if the input becomes negative infinity (or underflows numerical precision). One could manually set the probability for all particles that do not occur in the training set to zero. However, in a larger dataset, these might be present and the density estimate should be able to account for that. This increase in the number of populated particle states with increasing dataset size is also displayed in the right panel of figure 7.1. Thus, instead of setting the probability of some particles to zero by hand, we use top-$k$ sampling, which we introduce in section 7.1.2.

7.1.2. Transformer-encoder for density estimation in many dimensions

The transformer architecture presented in reference [9] consists of an encoder-decoder structure. Here, we use only the encoder part, which is described in more detail in section 2.4, together with its building blocks. We train this architecture for autoregressive density estimation, i.e. to estimate the probability of constituent $i$ given all previous constituents $1$ to $i - 1$. The probability of a jet can then be written as

$$p(x) = p(x_1)p(x_2|x_1) \cdots p(x_n|x_1 \cdots x_{n-1}).$$

(7.1)

Each individual constituent in equation 7.1 is described by a tuple of integers giving the $p_T$, $\Delta\eta$ and $\Delta\phi$ bin as discussed in section 7.1.1. The discrete particles are mapped to a trainable, continuous space using an embedding layer as explained in section 2.4. We use separate embeddings for the three features into the embedding dimension, which we set to 256. These three embeddings are then added up to form one vector of dimension 256 for each constituent.

One might ask why we first discretize the data and then use an embedding to get back continuous values. We could just as well use the continuous values directly as input to predict the discrete particle states. However, once we sample, we sample from the discrete space and use the discrete samples in further steps to generate consecutive particles. Training on continuous values and then using only the discrete values during sampling might reduce sampling accuracy.

The embedding is fed into a block of eight layers with multi-head attention, as
implemented in PyTorch [13] as TransformerEncoderLayer. We set the hidden dimension $d_h$ for all eight layers to the same value as the embedding dimension $d_m$, i.e. 256 and use four heads for the multi-head attention. The attention mechanism allows the network to learn which correlations between constituents are particularly important. Multi-head attention allows taking into account different correlations in a single step, we refer to section 2.4 for more details. The output of the last TransformerEncoderLayer then consists of 256 features for each particle. A final fully connected layer maps these 256 dimensions to the size of our particle dictionary, i.e. 39402 (including the stop token). The application of softmax activation for each constituent allows the interpretation of the output as the probability of each constituent to be one of the 39402 particle states. The mapping from the three tuples to one of the particle states is chosen as

$$b_i = p_{r,i} + 41 \times \Delta \hat{\eta}_i + 41 \times 31 \times \Delta \hat{\phi}_i$$

(7.2)

and can be easily inverted to get from $b_i$ back to the tuple $(p_{r,i}, \Delta \eta_i, \Delta \phi_i)$. Within and after the last TransformerEncoderLayer we use LayerNorm [77] (see section 2.4) and Dropout of 0.1 (see section 1.1).

The last mapping from 256 to 39402 dimensions shows one of the difficulties encountered in our setup. This layer alone has $256 \times 39402 + 39402 \approx 10M$ parameters. The overall architecture has $\sim13.3M$ parameters. Therefore, we need to balance the resolution, i.e. the number of different particle states, with the number of parameters in the model.

We train the network for 50 epochs on the full training dataset, i.e. 600k jets, using the Adam optimizer [50] and a batch size of 100. The initial learning rate is set to $5 \times 10^{-4}$. We apply a cosine learning rate schedule that reduces the learning rate to $10^{-6}$ at the end of the training. The training parameters have not been optimized, but worked rather out of the box.

As mentioned in section 7.1.1, we use the 50 leading constituents for each jet during training. Jets with less constituents are zero padded, i.e. we add as many zeros as needed to have 50 constituents. We apply a padding mask to these additional constituents, setting the attention to these particles to zero. Moreover, we apply a causal mask, since we want the prediction of particle $i$ to depend only on preceding constituents. This allows the evaluation of the full jet in a single pass during training and inference of the density. We want to stress that we limit ourselves to 50 constituents just to investigate the extrapolation performance of our density estimation. There is no computational limitation on these 50 constituents and the transformer setup itself is flexible with respect to the number of constituents.

Since we want to include the first particle in the evaluation of the density and we want to be able to sample new jets from scratch, we add a zeroth particle to all jets. We add the same, arbitrary particle for all jets. That way, the network cannot obtain information from this start token and learns the marginalized distribution of the first actual particle. We calculate the loss as the CE between predicted and
actual next particle and sum this up over all particles, ignoring the padded ones. Since our architecture preserves the length of the input sequence, i.e. the number of constituents, and we added the start token, we obtain predictions for up to 51 constituents. However, we only need to predict the 50 constituents of the original jet. Thus, we remove the last prediction, which has no counterpart in the jet, in the calculation of the loss.

Once the network is trained, we can use it to predict the probability of a jet \( p(x) \) by applying equation 7.1, i.e. multiplying the probabilities \( p(x_i) \) that the network assigns to the true constituents of the jet. For numerical reasons, we consider the log-probability and therefore add the logarithms of these probabilities.

Additionally, we can sample from the learned distribution. While we were able to apply the causal mask for training and inference, this is not possible for sampling. To sample, we first feed a jet containing only the start token that was used during training. The network then predicts a probability distribution for the first constituent over the 39402 possible particles of our dictionary. We sample according to this prediction and append the resulting particle to the jet. The jet is then fed through the network again, now with the start token and the first constituent. The network now returns a prediction for the first and the second constituent. We use the prediction for the second constituent, sample a particle and add it to the jet. This procedure is repeated until either a maximum number of constituents or the stop token is sampled.

As discussed in section 7.1.1, many of the possible particles in our dictionary never occur. The softmax activation does not allow the network to push probabilities to exactly zero. This becomes problematic as we have many of these unlikely particles. Looking at the right panel of figure 7.1, we see that for any constituent position at least 20k particles should be zero. Sampling 100 constituents, we expect at least one unlikely constituent if the probability assigned to these rare particles is of the order of \( 5 \times 10^{-7} \). In fact, the median probability assigned to the first constituent is an order of magnitude lower at \( 5 \times 10^{-8} \). However, particles with probabilities below \( 2 \times 10^{-5} \) contribute 1% to the full probability. This is a known problem also in NLP [196], where after a certain beginning of a sentence only a very limited number of words (compared to all possible words) can follow. An established way of handling this problem is by truncating the probability distribution. There are different ways on choosing how to truncate. The easiest would be to simply set all probabilities below some threshold to zero. Choosing this threshold is however difficult, particularly in physics, where small probabilities can be true. A more dynamic approach is the method of top-\( k \) sampling. In top-\( k \) sampling, the next word (or in our case particle) is sampled only from the \( k \) highest probabilities. An alternative approach is called nucleus sampling. In this approach again, only the highest probabilities are considered. In contrast to top-\( k \), the number of possible predictions is variable. Truncation is performed by collecting the highest probabilities until their sum is higher than a given threshold. All probabilities that are lower than the last one added to overcome the threshold are set to zero. The reader is referred to reference [196] for a more in depth discussion of these different sampling methods.
7. Learning the language of jets with transformers

Figure 7.2: Left: Estimated jet probability distribution for top (blue) and QCD (black) jets. The estimate is shown for a transformer trained on QCD (solid) and a transformer trained on top jets (dashed). Right: Correlation between jet constituent multiplicity and the probability estimate. The contours contain 25%, 50% and 75% of the density estimated by a kernel density estimate of 50k test set jets. The dashed line corresponds to $n = \log(p) / \log(1/N_d)$, i.e. the scenario in which each particle in the particle dictionary obtains the same probability for all jet positions.

Because of its simplicity, we apply top-$k$ sampling. This leaves us with the task of setting $k$. To determine suitable values of $k$, we once more look at the right panel of figure 7.1. The value should be somewhere between 3k and 10k, as most constituent positions have a number of distinct particles in this range. We choose $k = 5000$, as we find that this value gives good suppression of tails in the jet mass distribution while maintaining sufficient variation present in jet data (see section 7.3). The comparison between sampling from the full distribution and using top-$k$ with different values of $k$ is part of section 7.3. Note that the 5000 possible particles given at any position vary depending on all previous constituents of the jet.

7.2. Density estimation

After training the transformer network, we can use it to assign a probability to any given jet. We train on the 50 leading constituents, but evaluate the probabilities on 100 constituents. The extrapolation properties to more constituents are discussed further in section 7.3. We show the distribution of log-probabilities assigned to top and QCD jets in the left panel of figure 7.2. The top jet distribution is shifted to smaller probabilities. However, this is mainly a result of the higher multiplicities in top jets, as higher multiplicity results in smaller probability. We show the correlation between the probability and the multiplicity in the right panel of figure 7.2. The decrease in probability is close to exponential in the multiplicity. This behavior is expected, as every additional constituent adds a factor in equation 7.1. For a uniform prediction over the particle dictionary, i.e. every possible particle has the same probability at any given position, we expect the correlation to be exactly exponential. To be more precise, the probability would be given by $p = (1/N_d)^n$, where $N_d = 39402$.
7.2. Density estimation

is the number of particles in the particle dictionary and \( n \) is the multiplicity of jet constituents. This uniform probability assignment is added in the plot to guide the eye. One can extract an effective \( \tilde{N}_d \) for the correlation by fitting the slope of the QCD and top data in the right panel of figure 7.2. The fit results in \( \tilde{N}_d^{\text{QCD}} = 338 \) and \( \tilde{N}_d^{\text{top}} = 488 \). Thus, the slope would correspond to uniform distributions in much fewer than the almost 40k different particle states. We showed that the number of different particles at each position is significantly higher compared to these fit values in the right panel of figure 7.1. This is a first demonstration of the transformer being able to extract more than just trivial information, e.g. the bare number of different particles at any position.

At the lower end of the log-probability distribution we see a small plateau (around \( \log(p) = -600 \)). This plateau arises from not including more constituents, as all jets that would have additional constituents end up in that region.

While the probabilities become smaller, the phase space, i.e. the number of possible jets, increases. Therefore, even the jets with smallest probabilities are indeed present when sampling from the network, see section 7.3.

In addition to the top and QCD probabilities obtained by training on QCD jets, we show the distributions resulting from training on top jets as dashed lines in the left panel of figure 7.2. One can see a shift of the top jets to higher, and of the QCD jets to lower probabilities. Still, the QCD jets are much more likely, as they have fewer constituents. We do not plot the multiplicity-probability correlation when training on top jets in the right panel. Nevertheless, we evaluate the slope and find effective numbers of distinct particles \( \tilde{N}_d \) changing to \( \tilde{N}_d^{\text{QCD}} = 445 \) and \( \tilde{N}_d^{\text{top}} = 428 \). The decrease of \( \tilde{N}_d^{\text{top}} \) and the increase of \( \tilde{N}_d^{\text{QCD}} \) compared to the transformer trained on QCD jets show that indeed characteristics that differ between the two sets are learned.

These results highlight a general problem with using density estimation for anomaly detection. By using the density as an anomaly score, we expect the data not in the training set will have low densities. In our setup, this approach is only able to find top jets from a QCD background, but not vice versa. This holds even when we decorrelate from the number of constituents. The reason for this is that the distribution of top constituents is broader, i.e. at each position in the jet more entries of our particle dictionary are viable candidates. Moreover, the high likelihood regions at central angular values overlap in top and QCD jets. Similar results are obtained with an AE [3]. An additional problem with using density estimation for anomaly detection is that the density depends on the preprocessing of the input. We have studied these effects in references [3,6] and they have been studied in more detail in reference [197]. In addition, we suffer from the high dimensionality. Even if the two jet classes are well separated in some dimension, this may not be accessible in the overall density, which is dominated by the many dimensions without separation. We discuss this problem in some more detail in appendix C.

While anomaly detection using the density turns out to be difficult, one can use
7. Learning the language of jets with transformers

![Diagram](image)

Figure 7.3: Left: Log-likelihood ratio $s$ for top and QCD jets. Right: Resulting ROC curve for top vs. QCD classification using $s$ (red) compared to the ParticleNet performance (black). Solid lines correspond to evaluation on the test set, dashed lines correspond to a subset of 200k jets per class taken from the training set.

The learned densities for supervised classification. Having the probabilities for each jet under the top and QCD distribution, we can evaluate the likelihood ratio. We calculate the classification score as $s = \log(p_t(x)) - \log(p_q(x))$, where the index $t$ ($q$) denotes training on top (QCD) jets. Correct density estimation would make this score the ideal classifier score [174], and $s$ should then be monotonically related to an ideal, supervised classifier. We show the distribution of $s$ for both jet classes and the ROC curve resulting from varying the classification threshold in $s$ in figure 7.3. Indeed, while the shifts seem small in figure 7.2, they are still orders of magnitude in probabilities and there is resulting separation power in the classifier designed this way. The performance is significantly lower than that of the ParticleNet classifier, although this setup is also fully supervised. It is fully supervised, because we need labels to perform separate training on top and QCD jets. Still, the two transformer networks are not trained on the classification task of finding differences, but on the much harder task of estimating the individual densities. That this is a much harder task can also be seen in the comparison of ANOMaly detection with Density Estimation (ANODE) [132] and CATHODE [133]. The first performs two density estimates, one for the control region and one for the signal region, and builds the likelihood ratio explicitly, while the latter learns only a background density and then trains a classifier to distinguish this estimate from data. This comparison is performed on much lower dimensional data (on four features), making the density estimate much simpler compared to the full jet constituent space we are considering. We thus expect the discrepancy between the two methods to increase with dimensionality, as the density estimation becomes more and more difficult. A method to adapt the density estimation for classification by directly incorporating the likelihood ratio in the loss function has been proposed in reference [198].

The dashed lines in figure 7.3 show the potential of likelihood evaluation using transformers. We get these $s$ distributions and the corresponding ROC curves when evaluating the transformer networks on a subset of the training set instead of an
7.2. Density estimation

Figure 7.4: Training and validation loss for transformer training on QCD jets (left) and top jets (right). The lighter lines correspond to means of a few hundred updates, the darker lines are moving averages of the lighter values taking into account a window size of 50 entries. For QCD training, we show also a reduced (Red.) architecture with 6 transformer-encoder layers and a hidden dimension of 128.

independent test set. Considering the log-likelihood ratio $s$, top jets are shifted to slightly larger and QCD jets to lower values, resulting in a much stronger separation power as indicated by the dashed ROC curve in the right panel of figure 7.3. Such a difference in performance between training data and test data shows a bad generalization of learned features. The most common sources for generalization gaps are either that the test data is drawn from another distribution than the training data or overfitting of the model. We know that the datasets are drawn from the same distribution, as they have been simulated with the same settings. Overfitting to the training data can usually be prevented by implementing an early stopping that terminates training if the performance of the network becomes worse on an independent validation set during training. We implement early stopping in the ParticleNet training, hence the performance on training and test set are the same for this classifier. During training of the transformer there is however no increase in validation loss, as shown in figure 7.4. Early stopping as implemented for the ParticleNet classifier would thus not change the transformer results. An alternative source for the generalization gap can be an overcapacity of the model, such that it is able to learn general features and additionally memorize some of the training data. This interpretation of the origin of the generalization gap is supported by the steps visible in the training loss in figure 7.4. They occur at the beginning of each epoch after some training. The loss drops significantly for the first few batches of the epoch. This can be a sign of an overparameterized network, as instances that are seen late in epoch $i$ might be memorized. They can be part of early batches of epoch $i + 1$, as the data is shuffled after every epoch, reducing the overall batch loss. The network has sufficient capacity to keep the relevant general information and memorize the training data, leading to no increase in the validation loss. We see the same patterns also for training on top jets in the right panel of figure 7.3.

We already apply dropout during training, however these training curves motivate
7. Learning the language of jets with transformers

![Figure 7.5](image)

Figure 7.5: Sampling time as a function of the maximum number of constituents sampled for QCD jets. We sample 10k jets using batches of 100 jets and state the average time per jet for sampling from the full distribution and using top-\(k\) sampling with \(k = 5000\).

the use of a smaller architecture or stronger regularization. We test a smaller setup and reduce the number of encoder blocks to six and the hidden dimensions of the model to \(d_m = d_h = 128\). The resulting training curves are also shown in figure 7.4 for training on QCD jets. We see indeed a small reduction of the generalization gap. Moreover, the final validation loss is slightly smaller compared to the larger architecture. This trend holds over multiple training runs. Nevertheless, we stick with the larger architecture, because we find that the jets generated with this network have a higher quality, see section 7.3. Overall, we draw the conclusion that our setup will highly benefit from additional data.

7.3. Jet generation

We have discussed the quantitative evaluation of jet probabilities in section 7.2. In this section, we sample from the learned density, i.e. we generate new jets. We use the same models as in section 7.2 for the density estimation. Thus, training is performed on up to 50 constituents, but we compare samples and data with up to 100 constituents. When comparing samples to data, we use the discretized data for a fair comparison.

Efforts in training NNs to generate reliable data usually go hand in hand with the promise of faster event generation compared to Monte Carlo event generators. Similar to the authors of reference [32], we do not focus on our model as a generative model, but are rather interested in the capability of its density estimation in this work. Nevertheless, we show the time needed for jet generation on a NVIDIA Tesla
7.3. Jet generation

V100-SXM2-16GB GPU in figure 7.5. To obtain a sampling time per jet, we sample 10k jets in batches of 100 and divide the final time by the number of jets. We fit a quadratic function to the measured sampling times to show the quadratic scaling with the maximal number of constituents per jet. This scaling will break down eventually, when the maximum number of constituents that are sampled becomes smaller than the maximum number set for sampling. The sampling time will then be constant, as we stop sampling when every jet in the batch is complete, i.e. the stop token is sampled. We find that top-k sampling with $k = 5000$ is slightly faster than full sampling. While the sorting of probabilities generates some overhead, it is then faster to sample from the 5k entries compared to the full dictionary. Sampling with our setup is slower than previous attempts using GANs [199, 200], by two to three orders of magnitude. This is due to the iterative sampling of one constituent after the other arising from the autoregressive approach of density estimation. The authors of reference [32] benchmark their autoregressive model only on a CPU, which makes it by far slower than using Monte Carlo. Nevertheless, the GAN approaches need the multiplicity as additional input, while our setup generates jets according to the multiplicity distribution by itself. Moreover, we did not perform any optimization for sampling speed. More efficient implementations of the sampling procedure and a study of the dependence of sampling speed on the batch size are ongoing.

7.3.1. QCD jets

We show the distributions of various features for samples and simulated data in figure 7.6. To go back from the integer bin indices to physical units, we use the central value of each bin introduced in section 7.1.1. The value for the zeroth (underflow) bin is calculated by subtracting one bin width from the lowest bin center. Equivalently, the value for the overflow bin is calculated by adding one bin width to the highest bin center. This can lead to sampled constituents with too large $p_T$, as the overflow bin is mapped to $p_T(\hat{p}_T = 40) = 722$ GeV, which is already higher than the maximum $p_T^{\text{jet}}$ cut.

Assuming massless constituents, we can calculate the four momentum for each particle in the reference frame of the jet as

$$p_i = \begin{pmatrix} e_i \\ p_{x,i} \\ p_{y,i} \\ p_{z,i} \end{pmatrix} = \begin{pmatrix} \cosh(\Delta\eta_i) p_T,i \\ \cos(\Delta\phi_i) p_T,i \\ \sin(\Delta\phi_i) p_T,i \\ \sinh(\Delta\eta_i) p_T,i \end{pmatrix}. \quad (7.3)$$

First of all, we find excellent agreement in the produced multiplicity up to 100 constituents. Since we stop sampling at a maximum of 100 constituents, the last bin contains also those jets which would have more constituents if we continued to sample. Sampling from the full distribution shows a small excess in the overflow bin of the multiplicity, i.e. there is a small number of jets with too many constituents. Moreover, it also samples a few jets with too few constituents, hardly visible at the
Figure 7.6: Comparison of QCD jets sampled from the learned probabilities of the transformer and QCD jets from simulation. Each histogram is built from 200k jets. We show samples using top-$k$ with $k = 5000$ (red) and using the full distribution (blue).

low end of the histogram. In particular, empty jets can occur, where the stop token is sampled as first particle. These are very rare (1 in 100k) and ignored in the following comparisons. The log($p_T$) distribution of the constituents is well reproduced independently of using top-$k$ sampling. Constituents are sampled slightly too central, as can be seen in the $\Delta\eta$ distribution. The $\Delta\phi$ distribution looks very similar, which is why we do not include it here. Differences between top-$k$ sampling and sampling from the full distribution are particularly visible in the tails of reconstructed observables. As an example, we show the jet mass $m_{\text{jet}}$. The jet mass is obtained by calculating the jet momentum as sum of the constituents’ four momenta from equation 7.3, and taking the square root of the Minkowski product. Sampling from the full distribution produces a few jets with much too high masses, up to 7272 GeV. We show an overflow bin in the histogram for better visualization of the comparison between top-$k$ samples and the data. The high values occur because too many high $p_T$ constituents are sampled if the probabilities are not truncated. The network learns the $p_T$ ordering during training together with typical steps in decreasing $p_T$, a high $p_T$ constituent starts a new chain of decreasing $p_T$. One could enforce the $p_T$ ordering at every sampling step to overcome this issue. However, we want to interfere as little as possible. Moreover, the high $p_T$ constituents sampled at later position are
7.3. Jet generation

Figure 7.7: ROC curves for sample vs. data classification using a ParticleNet classifier on QCD jets (left) and top jets (right). The curves are shown for samples using top-$k$ sampling with $k = 5000$ (red) and for sampling from the full particle dictionary (blue).

not the only possible errors occurring when sampling from all particles within the dictionary. Thus, top-$k$ sampling is the better solution.

Going beyond the evaluation of sample accuracy by one dimensional histograms, we perform a classifier test. We train a supervised ParticleNet classifier to distinguish samples from discretized data. The classifier test has been suggested for the evaluation generators for fast detector simulation in reference [158]. The setup is the same as in section 7.1.1. It has been shown that some generators that are able to reproduce individual features well fail the classifier test [158,161]. This is in particular true for GANs. While they are trained with a discriminator, which performs exactly this classifier test, they are stopped at some point where an equilibrium might or might not be reached. Training a new classifier can then be sensitive to other deviations than the discriminator used in the adversarial setup. In reference [158], GAN samples are perfectly separated from actual data, i.e. the classifier achieves an AUC of one.

We show the resulting ROC curves for training the ParticleNet classifier on 200k samples against 200k QCD jets from the dataset in figure 7.7. We use jets from the validation set for training the classifier, as they have not been seen during training of the transformer. While we plot the loss of this set in figure 7.4, we do not use that loss for any optimization. The ROC curves are evaluated on an independent set of 100k samples and 200k jets from the test dataset. We show three curves for top-$k$ and full sampling, respectively. Each curve corresponds to an independent initialization of the transformer together with generation of independent training and test samples as well as an independent training of the ParticleNet classifier. The spread in these
7. Learning the language of jets with transformers

![Figure 7.8: Jet mass distributions (left) and ROC curves for a ParticleNet classifier trained to distinguish samples from data (right) for different values of $k$ in top-$k$ sampling.](image)

Curves thus incorporate fluctuations from both network training and the sampling. First of all, the performance is very stable as hardly any spread is visible for both, top-$k$ and full sampling. When using top-$k$ sampling, we see that the fraction of samples that can be (almost) perfectly separated from data is reduced. For QCD jets and sampling from the full distribution, we find that 6% of the samples can be well separated from data (given by the sample efficiency at data rejection of $10^4$). This fraction drops to 1.5% for top-$k$ samples. A first indication for this improvement can be seen in the $m_{\text{jet}}$ distribution in figure 7.6. For the full sampling we have 1.1% of the sampled jets with higher mass than the highest QCD jet mass. Using top-$k$ sampling this is reduced to only 0.01%. The AUCs of these ROC curves are $\sim 0.62$ and $\sim 0.63$ for top-$k$ and full sampling, respectively. While the difference in sample quality is significant at small sample efficiencies, this is not reflected by the AUC. The reduction of the ROC curve to one number (the AUC) should thus always be interpreted with some care. This has also recently been confirmed by reference [201] in the context of evaluating generative networks. How well the samples agree with the data on a dataset not used during transformer training can be taken as a measure of how well the correct density is learned. The results presented in this section demonstrate the impressive quality of the probability estimate using the transformer.

While we do not study the value of $k$ in detail, we make sure that changing it by a factor of two does not strongly affect our findings. We show the jet mass distribution in the left panel of figure 7.8. Choosing $k = 10k$ already significantly suppresses the tail, however some jets with higher jet masses remain. Reducing $k$ below 5000, we observe an increasingly narrow distribution. The reduction by a factor of two does not harm the modeling of the tail. Using $k = 500$, we see a shift in the bulk of the distribution to smaller jet masses and an overall to narrow distribution. The distribution becomes narrower until there is only a single jet mass when using $k = 1$.
7.3. Jet generation

because this corresponds to always sampling the same jet.

We show ROC curves from a ParticleNet classifier trained to distinguish samples with varying $k$ from data in the right panel of figure 7.8. For readability, we only plot one curve per chosen $k$. The stability of these curves for individual $k$ is in line with figure 7.7, with a slightly wider spread for $k = 10k$. We see hardly any difference using 10k or 5k for the top-$k$ sampling. Reducing $k$ to 2.5k weakens the performance slightly at higher sample efficiencies. At small $\epsilon_{\text{Samples}}$, the quality of the other samples is recovered. A better performance at high sample efficiencies means separation of data from the domain of samples. Reducing $k$ too much thus leads to a lack in diversity in the sampled jets, as one would expect. Overall, the choice of $k = 5000$ delivers the best sample accuracy in terms of the classifier test and the visual inspection of the jet mass distribution.

7.3.2. Top jets

We show the distribution of features for top jet samples and data in figure 7.9. The selection of features is the same as for QCD jets in figure 7.6. The multiplicity distribution is slightly worse and shifted to higher multiplicities for both ways of sampling. The $\Delta \eta$ (and the $\Delta \phi$) distribution matches better. Compared to the
7. Learning the language of jets with transformers

![Graph](image)

Figure 7.10: Same as figure 7.7, but for the classification of top jets against samples from a transformer trained on top jets.

The feature distributions hint towards a better performance of the ParticleNet classifier when distinguishing top jets from our top samples compared to the QCD case. We show the corresponding ROC curves in figure 7.10. The observed spread between different runs is much larger and the separation of samples and data is overall better. Still, we see the benefit of applying top-$k$ sampling ($k = 5000$). The sample efficiency at data rejection of $10^4$ lies between 1.5\% and 4.4\% for top-$k$ sampling and between 5.9\% and 7.9\% for full sampling. The AUC of 0.66 to 0.74 for top-$k$ sampling is significantly lower than 0.83 to 0.86 for full sampling, which in turn means better sample accuracy. In contrast to the QCD case, top-$k$ sampling improves the samples for both low and high $\epsilon_S$. Thus, it reduces the amount of samples in regions not covered by data, while increasing the amount of samples in regions where they are underrepresented when sampling from the full distribution. The AUC is mainly sensitive to the second part, as we have seen for QCD jets.

The larger fluctuations of the classifier suggest that training on a larger dataset should be beneficial for top jets. This is not surprising as the three pronged substructure of top jets together with the higher multiplicities reveal much more correlation that need to be picked up by the transformer. Moreover, the multiplicity distribution...
in figure 7.9 shows that training on 50 constituents might not be sufficient as this lies within the peak of the distribution.

7.4. Summary

Transformers have proven to be an extraordinary powerful DL architecture, setting new standards in the field of NLP. They have shown to scale to large datasets with ever improving performance. In this section, we have demonstrated the potential of transformer networks for density estimation in HEP. For this purpose we adapt low-level jet information, i.e. angular and momentum information of the jet constituents, to fit into the realm of NLP. We do so by discretizing the phase space and creating a finite, but large, dictionary of possible particles. We impose a $p_T$ ordering as grammar for the transformer to learn. The use of low-level features allows to access important information that might get lost when relying on handcrafted, task specific features. It has however as drawback the curse of dimensionality.

We have checked that changing to a discrete space does not destroy significant information by applying the powerful ParticleNet classifier to distinguish boosted jets from hadronically decaying top quarks from light QCD jets. The separation power is the same on continuous and discrete data. We perform the probability estimate of QCD and top jets with an autoregressive approach. The transformer learns to predict the next constituent of a jet based on all previous constituents. The training is performed on a set of 600k jets. The resulting densities from training on top jets and QCD jets show that model independent anomaly detection using the density is difficult. We intent to further investigate this approach in high-dimensional spaces in the future. By constructing the likelihood ratio from two transformers, one trained on top jets and the other on QCD jets, we have demonstrated that the transformer is capable of learning properties specific to the underlying physics of QCD and top jets. We have seen that the transformer is operating below its capacity, as the likelihood ratio on training data is significantly more separating compared to independent test data. In the course of this observation, we find that the training curves of the transformer also show signs of a too small dataset. In particular, there are steps in the training loss at the beginning of each epoch after some amount of training has passed and the training loss continues to decrease after the validation loss has saturated. We thus conclude that more training data will still benefit the model training and consequently the quality of the density estimate.

The sampling capability of the transformer allows for an additional evaluation of the density estimate. To suppress the sampling of extremely unlikely particles, we adopt the top-$k$ sampling method from NLP. We show that in particular the sampling accuracy of reconstructed observables like the jet mass benefit from this method. A perfect probability estimate would result in samples that are indistinguishable from data. We thus train a ParticleNet classifier to separate samples from data. Our generated samples are also hard to distinguish beyond one dimensional features,
which is a property that is difficult to achieve with other models like GANs \cite{158,161}.

This work has shown the potential of transformers for density estimation in HEP. The ability to process large amounts of data makes them a prime candidate for applications on LHC data already present, and also future datasets with an eye on the high luminosity phase of the LHC.

We considered discrete data in analogy to words in NLP. In general, the architecture and approach for density estimation can be adapted for continuous data. One way to achieve this is by learning parameterized Gaussian mixtures instead of the probability distribution over a particle dictionary. This approach is followed in reference \cite{162}. Moreover, a first application in physics has been performed after our publication in reference \cite{203}. An alternative way to proceed is the optimization of our discretization. We saw that the broader $\Delta \eta$ and $\Delta \phi$ distributions of top jets are better reproduced than the narrow peak in QCD jets. One could optimize the binning to foster training, e.g. by having fewer particle states that are not populated by data. Additionally, the influence of the constituents’ ordering on the density estimation is worth exploring. In particular, it is interesting to investigate in how far a more physical ordering as applied in reference \cite{32} helps the transformer, or if the model is able to pick up relevant information all by itself. Furthermore, we did not perform any hyperparameter optimization, neither for the architecture nor for the training.

Finally, training the transformer on density estimation is an unsupervised task and can thus be performed directly on data. Pre-training on a vast amount of data and using such a pre-trained model in a refinement has shown incredible potential in NLP \cite{14} and might also be applicable in the field of HEP.
Part V.

Conclusion
Conclusion

The application of Deep Learning (DL) to various fields of physics has gained a lot of momentum in recent years. DL models are able to learn complex correlations within a large amount of high-dimensional data. This allows for the extraction of features that are relevant for a given task and has boosted sensitivity compared to features constructed by experts. Within this thesis we make use of this property by applying DL methods with varying levels of supervision to low-level features, close to the original measurement. We have applied supervised classification to \( \gamma \)-ray source data, where labeled data is available, albeit expensive. We have then explored methods directly applicable to data without labels, as is the case at the Large Hadron Collider (LHC).

Exploiting the idea that DL is able to extract relevant features for a task at hand, we apply classifiers to the energy and time spectra recorded in the 4th Fermi Large Area Telescope (Fermi-LAT) \( \gamma \)-ray catalog, data release 2 (4FGL-DR2) in part III. We investigate the behavior of a Dense Neural Network (DNN) and a Recurrent Neural Network (RNN) on the task of classifying the \( \gamma \)-ray source types in section 4. We classify unassociated and unidentified UNClassified (UNC) sources within that catalog into Active Galactic Nuclei (AGNs) and PulSaRs (PSRs). We additionally perform the subclassification of PSRs into MilliSecond Pulsars (MSPs) and YouNG pulsars (YNGs).

In a first step, we evaluate the performance of our networks on the labeled sources of the catalog. Our results are competitive with previous results using the handcrafted features of the catalog [91, 92]. The accuracy of our classification is slightly reduced when considering a cross-match between the 4FGL-DR2 catalog and the previous, third version. This cross-matching procedure allows us to evaluate a sample selection bias that is present in astrophysical data, known as Malmquist bias. It relates to the fact that brighter sources are observed earlier and with cleaner signatures, being more likely to be classified.

We propose an alternative approach to feature selection using an AutoEncoder (AE), similar to reference [63] in the context of light curve analysis. The AE is an unsupervised method and can thus be trained on unlabeled data, i.e. sources that have not been assigned to a category yet. Training the feature extraction on UNC sources can reduce the Malmquist bias, as features are selected according to represent those sources well.

Finally, we provide AGN and PSR as well as MSP and YNG candidates in reference [109]. We find overlap as well as explainable discrepancies with references [91, 92], arising from different classification procedures and the use of the older 3FGL catalog in those references.

We go beyond the mere classification by introducing a Bayesian Neural Network (BNN) for reliable uncertainty estimation in section 5. By employing a toy model, we demonstrate that an ensemble of DNNs underestimates uncertainties, as it does not incorporate the uncertainty arising from limited training data. The BNN on the other hand is able to include this uncertainty, due to the applied prior.
Conclusion

Moreover, the prior serves as an intrinsic regularization for the BNN, reducing the susceptibility to overfitting compared to the DNN. Using the same toy model, we find that data augmentation leads to significant overconfidence in the predictions if it is not considered in the prior.

Having established our setup with the toy model, we turn to the classification of Blazars Classified as Uncertain type (BCUs) in the 4FGL-DR2 catalog into BL Lacerae objects (BLLs) and Flat Spectrum Radio Quasars (FSRQs). Using the energy spectra of the sources as input, we again first evaluate the performance on the labeled sources of the 4FGL-DR2 catalog. We show that the obtained uncertainty allows for some interpretation of the network behavior. In particular, the uncertainty is large for intermediate power law indices and large fluctuations in the energy spectrum.

We use again a cross-match between catalogs to set a tight and a loose selection threshold, taking into account the estimated uncertainty. While the tight selection maximizes the accuracy at the cost of fewer classified sources, the loose selection maximizes the number of classified sources.

Finally, we apply these two thresholds to the classification of BCUs and select BLL and FSRQ candidates. We provide these candidates in reference [109]. This list of candidates can be used further in population studies that allow for better modeling of the cosmic $\gamma$-ray background. The better modeling in turn can further constrain additional signatures that might be contained in the sources of the Fermi-LAT catalog, beyond currently known source types. Detecting a particular type of such an additional source class, namely Dark Matter (DM) subhalos, using a BNN has been part of a follow-up study in reference [204]. The final classification of candidates relies on multiwavelengths analyses, which are costly. Our primary classification thus provides potentially interesting targets for further observations. Further investigations, potentially using model independent methods to discover sources that do not belong to any of the known classes, are a promising research direction. One such potentially promising avenue could be the use of BNN uncertainties for anomaly detection. Additionally, the application of Classification Without Labels (CWoLa) like methods can also be transferred to astrophysical data, as demonstrated in reference [205].

From the supervised classification of $\gamma$-ray sources, we turn to weak supervision and unsupervised learning in part IV. We apply the CWoLa method [30] to the latest ATLAS mono-jet search [31] in section 6. CWoLa allows to boost sensitivity in a signal model agnostic manner, whenever a control and a signal region are available. While signal model independence goes hand in hand with reduced performance on a specific signal compared to fully supervised methods, the benefit of covering a broad range of potential signals prevails. This is especially true at the LHC, where we have not yet found the well-motivated new physics that we might have expected. We want to stress that the CWoLa setup applied in section 6 nevertheless approaches the supervised performance. Moreover, the CWoLa method is applicable directly on data, where the correct classification of an event to signal or background is not available. This eliminates uncertainties from transferring a classifier from Monte Carlo simulations to actual data.
Conclusion

We show the sensitivity increase at the example of semi-visible jets arising from a strongly interacting dark sector [176]. Using simulated data, we find discovery potential below a 1% signal fraction in the signal region. This sensitivity reaches far beyond the ATLAS search, as they assign a 1.2%, systematics dominated uncertainty to the SM background in the signal region. While the original search is dominated by systematic uncertainties and simply taking more data will not increase the sensitivity much further in a straightforward way, CWoLa will highly benefit from much more data. This is particular good news in view of the high luminosity phase of the LHC, which will provide an additional factor of ten in integrated luminosity [206].

The way the CWoLa method is applied does not depend on the particular signal considered. Nevertheless, the performance is signal model dependent as the classifier will have different separation power depending on the signal at hand. Therefore, the CWoLa method can only be used as a discovery tool and does not provide general limits, in contrast to the ATLAS search. Since the signal we present as an example has been found to be rather challenging [1, 6], we expect similar or even better sensitivity to a broad range of other signals.

We investigate the sensitivity of the method to mismodeling of the background in the control region by ignoring, underestimating, or overestimating subdominant backgrounds in the control region. We show that an absolute understanding of the rare backgrounds at the percent level is required. However, a relative agreement at the 10% level is sufficient.

Overall, the CWoLa method shows the potential to significantly enhance the discovery potential of already conducted analyses. The method can be applied directly to data and provides a background estimate from the control region. In an iterative process, CWoLa can be used to improve the understanding and modeling of relevant backgrounds to reduce differences between control and signal region. Finally, one can either explain the signal region by eliminating every difference to the control region, or an unexplainable difference remains that provides a basis for an interpretation in terms of new models.

We used the ParticleNet classifier for our studies. We are currently investigating, how incorporating symmetries, e.g. Lorentz invariance, in the classifier architecture can increase the sensitivity to small signal fractions. In references [207] a Lorentz invariant graph convolutional neural network (LorentzNet) has been shown to outperform ParticleNet on small datasets. While our full dataset is not small, the amount of signal is. Therefore, we may benefit from additional inductive biases in the architecture. Moreover, the score distributions offer more discrimination power than just cutting at some threshold. It will be interesting to see how a statistical analysis of the score distribution as suggested in reference [208] may still improve the results.

Going beyond weakly supervised methods, we turn to the unsupervised task of density estimation in section 7. We employ a transformer-encoder network for autoregressive density estimation in the phase space of jet constituent momenta. For such high-dimensional spaces, density estimation is a notoriously difficult task due to the curse of dimensionality. Since transformers have shown excellent performance
in the realm of natural language, we choose to adapt jets to resemble words and sentences. The analogy between jets and constituents to sentences and words is built by discretizing the possible states of particles and introducing $p_T$ ordering as a simple grammar.

We apply the density estimator to boosted, hadronically decaying top jets and light QCD jets separately. The likelihood ratio constructed from the two transformers provides separation power. This shows that the transformer is able to find relevant structural differences in the density estimation.

To further evaluate the quality of our density estimate, we sample from the learned distributions. We suppress noise in the samples by using the top-$k$ sampling method, which only takes into account the $k$ highest probabilities. We see a good match between samples and data in one dimensional distributions. The top-$k$ sampling especially improves the tails of distributions like the jet mass. We use the same ParticleNet [73] classifier used in section 6 to differentiate samples from data, in order to evaluate sample quality beyond simple, one dimensional distributions. The bad separation power obtained by the classifier supports the high quality of our samples.

Overall, the transformer shows promising results in this first attempt. In the near future, the application to larger datasets should be investigated, as we see overfitting of the model. Moreover, other sampling methods, e.g. nucleus sampling [196], and more efficient implementations should be explored, as the current setup is not competitive in terms of sampling time to other generative models on constituent level [199, 200]. Nevertheless, the direct accessibility of the probability and the impressive results of the classifier test make the further investigation of this setup a path worth following. Of special interest is the application to anomaly detection. While we demonstrate that this is a difficult task in many dimensions, the authors of reference [33] find separation of top and QCD jets in both directions, using diffusion models.

To conclude, we have seen that DL provides a broad range of methods that can be applied to improve the way we analyze data in physics. The low barrier into the development of DL methods and architectures, due to efficient implementations provided by open source codes, allows the physics community to quickly adapt new ideas from computer science to our needs. These ideas are transformed into methods and tools that finally foster a deeper understanding of the data and thus of the underlying physics. In particular the use of weakly supervised and unsupervised methods, albeit difficult to develop and evaluate, presents itself as a promising way to discover what is currently beyond our understanding.
Acknowledgments

I would like to take this opportunity to express my gratitude and appreciation to those who have contributed to the completion of my PhD thesis.

First and foremost, I am especially grateful to my first supervisor, Prof. Dr. Michael Krämer, for giving me the opportunity to work on my thesis topics. His support and guidance have been instrumental in shaping my research and academic growth. Despite the challenges posed by the Covid-19 pandemic, Prof. Krämer consistently provided a fruitful scientific environment, fostering innovation and collaboration.

Second, I want to thank Prof. Dr. Martin Erdmann for agreeing to be my second supervisor. His lecture on deep learning in physics research introduced me to the broad topic I now also wrote this thesis about.

Furthermore, a special thanks is due to Dr. Alexander Mück. Throughout my entire PhD, as well as during my Master’s thesis, he has been a constant source of support and guidance. His open-door policy and willingness to offer assistance and insights have been invaluable throughout my PhD journey, including the process of writing this manuscript.

Besides Prof. Krämer and Dr. Mück, I am grateful for the opportunity to collaborate with a talented group of people. I would like to thank Elias Bernreuther, Thorsten Buss, Anja Butter, Barry Dillon, Marie Hein, Felix Kahlhoefer, Felicitas Keil, Maximilian Lipp, Silvia Manconi, Alessandro Morandini, Ivan Oleksiyuk, Tilman Plehn, and Jan Tönshoff. Working together on various topics has provided me with numerous opportunities for stimulating discussions and intellectual growth.

Since we were allowed back in the office, it has been a pleasure to share it with Magnus Cornelius Schaaf and Sven Yannick Klein. I hope we will finally order that pizza.

I would also like to acknowledge the efforts of Elias Bernreuther, Sven Yannick Klein, Alexander Mück, Kathrin Nippel and Friederike Sommer for their valuable assistance in proofreading this thesis and providing helpful suggestions to enhance its clarity and coherence.

Lastly, I want to express my appreciation to my parents for providing me with everything I needed to pursue my goals. I would also like to thank Rike for her constant support throughout the last nine years.

My work has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant 400140256 - GRK 2497: The physics of the heaviest particles at the Large Hadron Collider. I gratefully acknowledge the computing time granted by the NHR4CES Resource Allocation Board and provided on
the supercomputer CLAIX at RWTH Aachen University as part of the NHR4CES infrastructure. The calculations for this research were conducted with computing resources under the project rwth0934. I used DeepL Write for checking grammar and style of certain paragraphs when I was in doubt. Parts of the abstract were translated with DeepL Translate.
Part VI.

Appendix
A. CWoLa with different region sizes

We expect about half as many Z+jet events in the Control Region (CR) as in the Signal Region (SR), due to the lower branching fraction of the Z boson decaying into charged leptons. Training with a reduced CR corresponding to 500k events against the full 1M event SR with \( f_{SR} = 0.01 \) results in the Receiver Operating Characteristic (ROC) curve shown in the left panel of figure A.1. While there is some reduction in the signal tagging performance, this is mainly at intermediate signal efficiencies \( \epsilon_S \). At the threshold that we use, \( \epsilon_B = 0.001 \), the resulting signal efficiency is largely unaffected by the reduced CR.

We show the corresponding distribution of the classifier score \( s \) in the right panel of figure A.1. Note that the CR curve is reduced by a factor of two compared to the SR background. Correcting for this factor results again in almost perfect agreement of the score distributions of background in the SR and the CR. The threshold is set such that 500 events survive in the reduced CR. In the signal region 1609 events lie beyond this threshold. Out of the selected SR events, 604 correspond to anomalous events and the remaining 1005 to background, in line with \( n_{SR}^{exp} = 1000 \). This is similar to the results in table 6.1 for \( f_{SR} = 0.01 \), where 625 anomalous events are found on top of 1041 background events. The estimate of the statistical significance is however reduced to 11.12 compared to 14.89. This is a result of the higher uncertainty in the threshold estimation from the smaller CR.

Figure A.1: Comparison of classifier performance for training on a balanced dataset (black), i.e. SR and CR contain 1M events, and an inbalanced dataset in which the CR has 500k events and the SR 1M (purple). The balanced curve corresponds to the standard setup shown in section 6.3.
Another aspect one needs to keep in mind when talking about the significance improvement with CWoLa searches is systematic uncertainty. As discussed in section 6, the ATLAS search we consider is systematics limited, with a systematic uncertainty of ~1.2%. A typical way to look at the improvement a classifier provides over the cuts applied prior to the classifier is the significance improvement curve. Considering only the statistical significance, the significance improvement is given by \( SI = \frac{\epsilon_S}{\sqrt{\epsilon_B}} \). Including an additional relative systematic uncertainty \( \alpha \) from the original background estimate, the significance improvement is instead given by \( SI = \frac{\epsilon_S \sqrt{(1 + \alpha^2 B)}}{(\epsilon_B + \alpha^2 \epsilon_B B)} \). First, this reduces to the purely statistical significance improvement for \( \alpha = 0 \). Second, it introduces a direct dependence on the number of background events, resulting in larger significance improvement with increasing amount of data. This is true even if the classifier performance in terms of the ROC curve remains the same. We show the significance improvement as a function of \( \epsilon_S \) in figure B.1. We present the purely statistical improvement, which has a maximum of around two. Including some systematic uncertainty, which we assume to be independent on the classifier cut, the significance improvement increases. Already for \( \alpha = 0.008 \) the maximum significance improvement reaches 16. The maximum is close to 24 at \( \alpha = 0.012 \), corresponding to the systematic uncertainty stated in the ATLAS mono-jet search [31]. The improvement increases further with larger \( \alpha \).

In addition to the significance improvement curves, we show at which signal efficiencies we have background efficiencies of \( 10^{-2} \), \( 10^{-3} \) and \( 10^{-4} \). While our choice of \( \epsilon_B = 10^{-3} \) does not maximize the significance improvement, it is generally close to...
the maximum. Moreover, the variation by a factor of ten still shows large significance improvements. This is in line with our argument in section 6.2.3, considering the balance between increasing statistical uncertainty and reduced systematic uncertainty with stricter cuts.

C. Density estimation for anomaly detection in many dimensions

To discuss why the density is difficult to use as an anomaly score in high-dimensional data, we consider two distributions from which we can easily sample and compute the density. Let the background distribution be a multivariate, diagonal normal distribution with dimension $d$, all means at zero, and the identity in $\mathbb{R}^{d \times d}$ as covariance matrix, i.e. $p_B(x) = \mathcal{N}(0, I)$. We first consider a signal distribution for which we shift the mean of the first dimension to 2, i.e. $p_S(x) = \mathcal{N}((2, 0, \ldots, 0)^T, I)$.

We show in figure C.1 how the log-probability is distributed under the signal and background distribution for signal and background samples. From top left to bottom right, we increase the number of dimensions $d$ from one over five to 100 and finally 500. The top left shows how well the two samples can be separated using only the background density, because it only considers the dimension in which the signal is shifted. The signal is at much lower probabilities, and using $\log(p_B(x))$ as the anomaly score gives an AUC value of 0.85. The likelihood ratio performs better with an AUC of 0.92, because it also includes information about the signal distribution. As the dimension $d$ increases, the distributions become more and more similar. In 500 dimensions the AUC goes down to 0.55. Note that the score cannot go below 0.5, because all dimensions except the first are equally distributed, and the first dimension leads to some slightly lower probabilities of the signal. Finally, the information on the separation is still available in the likelihood ratio $s = \log(p_S(x)) - \log(p_B(x))$. This is also visible in the figure C.1 since there remains a (small) shift between the distributions of the same color. Using the likelihood ratio as score, the AUC remains at 0.92 also for 500 dimensions.

This example shows how added features that contain no information degrade the performance of the density as an anomaly score. In addition, increasing the number of dimensions can have more severe effects if the signal follows a narrower distribution compared to the background. To illustrate this, we show the same distributions as in figure C.1, but for $p_S(x) = \mathcal{N}((2, 0, \ldots, 0)^T, 0.9I)$. The added features are thus distributed slightly narrower for the signal compared to the background, but with the same mean. For this scenario, the AUC drops below 0.5 to 0.38 and 0.14 for $d = 100$ and $d = 500$, respectively. At the same time, the likelihood ratio becomes better, with an AUC increasing from 0.92 to 0.97. The separation using the likelihood ratio improves because each additional feature contains information.

---

5The mean vector is of dimension $d$, so there are $d - 1$ zeros, resulting in no zero at all for $d = 1$. 120
Figure C.1: Log-probability distribution for background (black) and signal (blue) samples under the background (solid) and the signal (dashed) distribution. From top left to bottom right the dimension $d$ of the multivariate normal distributions $p_{B/S}$ increases from one to five to 100 to 500.

Figure C.2: Same as figure C.1, but for a signal distribution $p_S$ with more narrow covariance of 0.9 compared to background with 1.
Acronyms

4FGL-DR2 4th *Fermi*-LAT γ-ray catalog, data release 2
4FGL-DR3 4th *Fermi*-LAT γ-ray catalog, data release 3
AE AutoEncoder
AGN Active Galactic Nucleus
ANODE ANOmaly detection with Density Estimation
AUC Area Under the ROC Curve
BCU Blazer Classified as Uncertain type
BDT Boosted Decision Tree
BLL BL Lacerae object
BNN Bayesian Neural Network
BSM Beyond the Standard Model
CATHODE Classifying Anomalies THrough Outer Density Estimation
CE Cross Entropy
CNN Convolutional Neural Network
CR Control Region
CWoLa Classification Without Labels
DGCNN Dynamic Graph Convolutional Neural Network
DL Deep Learning
DM Dark Matter
DNN Dense Neural Network
*Fermi*-LAT *Fermi* Large Area Telescope
FSRQ Flat Spectrum Radio Quasar
GAN Generative Adversarial Network
GPU Graphical Processing Unit
GRU Gated Recurrent Unit
HEP  High Energy particle Physics
ISR  Initial State Radiation
KL  Kullback-Leibler
KNN  k Nearest Neighbors
LHC  Large Hadron Collider
LSTM  Long Short Term Memory
ML  Machine Learning
MLP  Multi Layer Perceptron
MSE  Mean Squared Error
MSP  MilliSecond Pulsar
NLP  Natural Language Processing
NN  Neural Network
PCA  Principle Component Analysis
PReLU  Parameterized ReLU
PSR  PulSaR
QCD  Quantum Chromo Dynamics
ReLU  Rectified Linear Unit
RNN  Recurrent Neural Network
ROC  Receiver Operating Characteristic
SGD  Stochastic Gradient Descent
SM  Standard Model of particle physics
SMOTE  Synthetic Minority Oversampling TEchnique
SR  Signal Region
TraDE  Transformer for Density Estimation
UFO  Universal FeynRules Output
UNC  UNClassified
VAE  Variational AutoEncoder
YNG  YouNG pulsar
References


References


References


References


References


References


References


References


References


References


[174] J. Neyman and E. S. Pearson, *IX. On the problem of the most efficient tests of statistical hypotheses*, *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* **231** (1933), no. 694-706 289–337.


