Department of Physics and Astronomy
University of Heidelberg

Bachelor Thesis in Physics
submitted by

Nicholas Kiefer

born in Bad Soden (Germany)

2018

Quarks and Gluons

Discriminating Jets With Deep Neural Networks

This Bachelor Thesis has been carried out by Nicholas Kiefer at the
Institute for Theoretical Physics in Heidelberg
under the supervision of
Prof. Dr. Tilman Plehn

Abstract

In the last few years, it has been shown that neural networks can be applied to particle physics to improve discrimination between various collider signatures and their backgrounds. We present an optimized, physics-inspired neural network for quark and gluon jet discrimination. On a pure sample, our Deep Neural Network outperforms a conventional Boosted Decision Tree and has comparable performance to an image-based Convolutional Network. We also show the performance of our network in simulated invisible Higgs + monojet events, where a lower discrimination power is achieved, partly because both signal and background events can contain quark jets and gluon jets.

Zusammenfassung

In den letzten Jahren wurde gezeigt, dass Neuronale Netzwerke in der Teilchenphysik die Diskriminierung von Detektorsignalen und deren Untergrund verbessern können. Wir präsentieren ein physik-motiviertes Deep Neural Network für quark-gluon jet Klassifikation. Das Netzwerk zeigt eine bessere Diskriminierung als Boosted Decision Trees und eine vergleichbare mit bild - basierten Convolutional Netzwerken. Wir wenden weiterhin das Netzwerk auf simulierte unsichtbare Zerfälle von Higgs bosonen zusammen mit einem Monojet an. Die Diskriminierung ist hier schlechter, auch weil Signal und Untergrund jeweils Quark und Gluon Jets enthalten.

Contents

1 Introduction

The Standard Model has enjoyed outstanding success since the prediction of the Higgs boson [1–3] and the confirmation of its existence by the ATLAS collaboration [4] and the CMS collaboration [5]. The Standard Model is self-consistent up to a very high energy, namely the Planck energy. This success is dampened by the fact that the Standard Model does not include gravity, and therefore still has to be joined up with General Relativity. Moreover, there are several phenomena which current physics cannot explain. Since 2012 (the confirmation of the existence of the Higgs boson), there has been more and more concentrated effort to look for physics beyond the Standard Model (BSM). Some of those phenomena which do not get described correctly by the Standard Model include neutrino oscillations [6] and the baryon asymmetry.

An exciting field is the search for *Dark Matter* (DM), which could potentially explain multiple discrepancies between theory and measured reality. There have been a multitude of proposals for the detection of dark matter at particle accelerators (for a review see e.g. [7]). Because it is still unclear what properties the new matter must have the predictions differ very much. The Higgs sector is here of importance because some BSM theories would allow the Higgs boson to decay to DM invisibly.

The last years have shown that looking for new physics has become increasingly difficult. The lack of observation of dark matter at particle accelerators means that experimentalists need to go to higher energy, while also increasing the precision and efficiency of the analysis. In the last 20 years, a new field in Science has emerged that reflects the jump in computer performance and the size of data to process. This data science especially had a breakthrough with the rise in the performance of machine learning algorithms. Mainly, Deep Learning has achieved new heights of performances with the invention of efficient algorithms and new structures. The yearly competition of *imgnet* has shown the efficiency of convolutional networks for classifying objects in pictures [8], and we will continue to expect an even better performance.

It has been shown in the last few years, that machine learning and especially neural networks might also have a useful application in particle physics. One drawback of deep learning is for instance, that networks require large amounts of training data. Huge amounts of data have always been generated at colliders, which therefore begs the question of whether machine learning can be applied in the analysis.

There are various phenomenological studies using neural networks to show that the multivariate analysis can be outperformed. This has been shown for quark and gluon jet discrimination and top tagging though many other fields will be explored in the future. The studies are using different variations of neural networks like Convolutional Neural Networks (CNNs) or densely connected networks. CNNs outperform high-level variables, but if enough high-level information is included the improvement in performance is saturated [9]. Boosted Decision Trees are among the methods that are currently used, so there is a baseline to compare performance to. These consist of simple classification algorithms that make use of previously constructed input. Boosted means here that for performance gain

multiple such trees are trained, so that the final decision can be a mean of the output of the trees or that best performing models can be chosen at each instance of training.
In this work we want to concentrate on using neural networks, specifically a physics-inspired deep neural network, to look at the discrimination power on quark and gluon jets, Higgs + monojet and Z + monojet events at proton - proton collisions with a beam energy of $\sqrt{s} = 13$ TeV. We will start with an explanation of the physics involved, as well as the technical side of neural networks. We then describe the sample generation procedure, and present our results. We will then compare our model to a previous work of Schwartz et al. [10]. We will then apply our quark-gluon tagger to a benchmark BSM scenario, invisible Higgs decays in association with monojets. Finally we will summarize our results and give an outlook to future prospects.

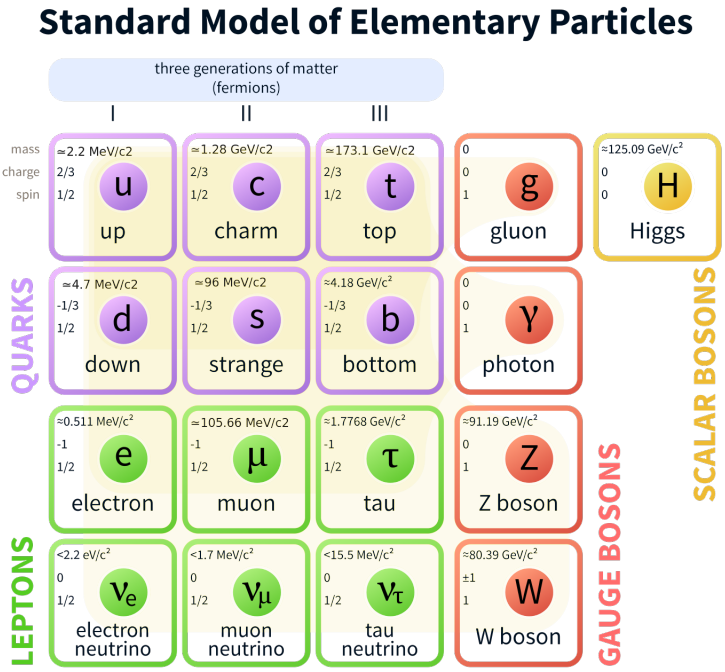

Figure 1: The Standard Model. Taken from [11].

2 Background

2.1 Standard Model

2.1.1 Particles

The Standard Model consists of elementary particles, which are sorted into families, where members have similar traits. All particles are assumed to be infinitely small in size and can have zero mass depending on the particle. Fig. 1 shows a table of every particle with additional information.

The particles can be divided into fermions and bosons, which consequently get assigned a half-integer spin and integer spin. The Fermions can be further divided into three generations of quarks and leptons. The leptons are the electron, the muon and the tau, and each lepton has a neutrino counterpart. The Standard Model assumes the mass of the neutrinos to be zero.

The quarks consist of two quarks per generation, historically named up, down, charm, strange, top and bottom. The top quark mass is too high for the top quark to exist long enough to form bound states.

2.1.2 Interactions

The fundamental particles interact with each other through three different fundamental forces, mediated by the bosons. The bosons consist of five particles, and mediate the forces in the following way. The massless photon γ, which has charge zero, is the mediating

particle of the electromagnetic force; $W^{\pm}$- and Z^0-bosons mediate the weak force, and gluons the strong force.
Not every particle experiences every force. The weak force and the electromagnetic force act on all fermions and the strong force acts only on quarks and gluons. The strong attraction of quarks leads to the formation of hadrons, which can be further divided into baryons, which consist of three quarks, and mesons, which consist of a quark and antiquark pair. Exotic hadrons with five quarks have already been observed at LHCb [12]. These are called pentaquarks.
The strong force leads to a phenomenon called 'color confinement', which prohibits the existence of stable particles with non-zero color charge. This confinement is a result of the gluons, as mediator of the strong force, having color charge themselves, unlike photons which have electric charge zero.

2.1.3 Higgs Boson

The Higgs boson is a scalar boson with spin zero. It is a consequence of the Higgs-mechanism [13]. The Higgs mechanism explains the non-zero mass of the gauge bosons W^+, W^- and Z^0 which mediate the weak force. The Higgs boson is of considerable interest in the search for new physics, because the Higgs can potentially couple to any singlet gauge field ϕ. This would imply a decay of the Higgs to $\phi\phi$ (provided low enough mass m_ϕ), therefore looking for invisible decays of the Higgs is important for BSM. A decay to $\phi\phi$ then means, that it is not detectable anymore, if there are no additional interactions involving ϕ. We call that an invisible decay.

2.2 Particle Colliders

In particle colliders the idea is to use the high energy collision of particles to (temporarily) produce new and other particles. The high energy is needed to overcome the electrostatic repulsion and to provide the energy for the masses of particles. There are different accelerators depending on the accelerated particles and targets.
The LHC is the biggest particle accelerator to this day. It is a proton-proton collider having a beam energy of up to $\sqrt{s} = 13$ TeV. Multiple experiments are running with beams supplied by the LHC. Every experiments measures the electric charge of the particles as well as the energy stored in the calorimeters. Additional information is gained by tracking the way of the particles through the detector. A running experiment produces a lot of data in a very short time. When two proton beams collide there is not only one collision happening but several at the same time. The subsequent fragments will also interact with each other. It is therefore necessary to have tools to separate the events of interest (Signal) from the ones that only appear to be signal-like (Background). A higher instantaneous luminosity of the particle collider will also lead to more particles produced (*pile-up*), so there is need for good discrimination algorithms besides higher precision of the measuring devices.
There are two more phenomena leading to even more particles to distinguish. *Initial State*

Figure 2: Example Feynman diagrams for signal (left) and background (right) of Higgs boson production.

Radiation and *Final State Radiation* can produce particles outside of the hard-scattering processes happening in the primary collision.

2.2.1 Quark and Gluon Jets

Jets are a common occurrence in particle colliders. They are a result of the strong force and color confinement. When a quark pair gets separated the potential energy between those will rise. When the force pulling the quarks apart is high enough the connection will break and with the set-free energy a new pair of particles will be produced. This follows out of confinement because a single quark cannot exist on its own because every quark has a color charge. Only 'white' particles i.e. particles with a net color charge of zero can be stable. The newly created pair of quarks will also drift apart. This process repeats and leads to a multitude of particles flying in the same direction. The cluster of particles is called a jet. In particle trackers they have a characteristic look.
As described there is in theory an initial particle (it does not necessarily have to be a pair) that induces a jet. We distinguish between quark and gluon jets in this work, referencing the initializing particle in the name. This is a theoretically well-defined name, although in practice it can be unclear what that means [14]. In particle colliders only final-state products are measured not initial-state particles. Additionally, it is possible for the initial particles to interact in ways that make it harder to define what the initiating particle of the jet is.
Jets are a natural way of discriminating background and signal. In many processes the signal is either combined with a quark jet while the background comes with a gluon jet or the other way around. Fig. 2 shows the leading order Feynman diagram for the single Higgs production in association with a jet. Gluon fusion is the production channel with the highest cross section. The top loop is only one possibility, though because the Higgs boson couples to the mass, it is the most likely process happening. A gluon is produced, which will generate a jet. The background Feynman diagram in Fig. 2 consists of a Z boson together with a quark jet.
Although the source of the jets are two different particles, the constituents and shape of the jet can be quite similar. Therefore methods and variables have been constructed for the discrimination of a quark jet and a gluon jet. We will later show and explain these variables.
One main difference between quark and gluon jets is in their radiation pattern. The probabilities to emit a quark from a gluon or a gluon from a quark are linked to the color

factors. When calculating to leading order the ratio of the average number of particles in a jet will be $\sim\frac{C_A}{C_F}$ [15], where $C_F = 4/3$ is the color-factor for a gluon emission from a quark, and $C_A = 3$ the color-factor associated with a gluon emission from a gluon. We therefore expect on average $9/4$ times more constituents in a gluon jet.

2.3 Machine Learning

2.3.1 Boosted Decision Trees

Boosted Decision Trees (BDTs) are multivariate classifiers. They are an improvement over Decision Trees extending the machine-learned cut flow to multiple trees. This is done to avoid overfitting. A Decision Tree is made up of several branches, where binary decisions are made. The following subsets are divided further till the final subsets theoretically only contain signal events or background events. The input to an BDT are often specifically constructed variables. We will introduce these variables for quark and gluon jets later.
This classification technique is widely used in particle physics as they are reliable and make choices on reconstructible parameters. In this work we want to focus on a different sector of Machine Learning, though we will use performances from BDTs for comparisons.

2.3.2 Deep Neural Networks

In the last few years Deep Learning and especially neural networks have shown remarkable performances on a variety of tasks. The basic idea is very simple though the complete architecture can be very complex. Neural networks are a category of Machine Learning and can be trained either supervised or unsupervised. We will focus on supervised learning, where we have access to truth labels on the data.
The simplest neural net one can build contains one layer. This layer can hold multiple neurons but to make it even less complex, one can start with one neuron. This neuron has an input in the form of some real number x and an output, once again in the form of some real number y. The linear transformation on the input by one neuron can be written as:

$$y(x) = \sigma(wx + b). \tag{1}$$

This neuron has a weight w and a bias b, which are called parameters. These are adjusted in the training of the neural network. σ is a placeholder for a so-called activation function. These are often non-linear, and we will explain them later.
Making the architecture more complex means making the network deeper (adding more layers) or broader (adding more neurons to layers). The first layer receives the input, and is therefore called the input layer. The succeeding layers are called hidden, they only receive input from the previous layer. The last layer is not hidden and called the output layer.

2.3.3 Dense Layer

The most basic layer is the Dense Layer. A neuron in a dense layer is connected to (receives input from) every neuron in the previous layer. Every connection has a weight and every neuron has a bias. The output is also called activation. The activation a_j^l of neuron j in layer l is therefore [16]:

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) = \sigma(z_j^l). \tag{2}$$

The sum is over every neuron in the previous layer $l-1$. The output z_j^l before the application of the activation function σ will be important later.

2.3.4 Backpropagation

The Backpropagation algorithm is the heart of the learning feature of neural networks. It makes use of the *loss function* to compute updates for the weights in each layer. We will use the gradient descent algorithm. The basic guiding principle is the gradient of this loss function with respect to each weight. Updating the weights in the direction of the negative gradient is supposed to find a (local) minimum and therefore minimize the loss overall. The update done is further weighted with a factor η called the *learning rate*. The procedure is the following:

$$v_t = \eta \nabla_a C_t \tag{3}$$

$$a_{t+1} = a_t - v_t. \tag{4}$$

The computation takes the following form: the network gets some input x and produces some output y, from which the loss function can be calculated. The error is then back-propagated by calculating the gradient of the loss function with respect to the individual weights. This is multiplied with a learning rate and then the weights get updated by subtracting this error. After this procedure the network is ready to receive new input, to further minimize the error.

2.3.5 Loss Function

There are several choices for loss functions (also called cost functions). The obvious and most easiest to understand is the mean-squared-error:

$$C(w,b) = \frac{1}{2n}\sum_x |y(x) - a|^2 \tag{5}$$

This function simply measures the squared error C from the predicted output $a = \sigma(wx+b)$ of the network to the actual result y. The size of the training sample set is n. The loss in this case can be dominated by outliers. There are more sophisticated loss functions, we

will implement the cross entropy:

$$C = -\frac{1}{n}\sum(y \ln a - (1-y)\ln(1-a)) \tag{6}$$

This loss function comes from information theory and usually results in better performance, avoiding the so-called *learning slow down* [16]. Unlike the mean-squared-error the cross entropy is always high when the error is high and small when the error is small, therefore exhibiting some features during training that leads to a faster convergence.

2.3.6 Optimizers

Besides the parameters of the network, there are also hyperparameters, which is what one would change when optimizing a network. The parameters themselves will get updated through the backpropagation. The learning rate is one such parameter. The most well know optimization method is the stochastic gradient descent (SGD). After setting the learning rate the algorithm will randomly sample the data and update the parameters based on the loss. The SGD can be extended by various methods aimed to use more information than just one gradient, and several methods were created, like Adam [17], Adagrad [18], Adadelta [19], RMSProp [20]. I will briefly explain the Adam optimizer.
The learning rate η is not a fixed hyperparameter anymore, but rather defined per parameter. The parameter θ at step t for $t > 1$ can be computed with the following algorithm:

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \tag{7}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{8}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{9}$$

$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)} \tag{10}$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)} \tag{11}$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{(\sqrt{\hat{v}_t} + \epsilon)}, \tag{12}$$

where β_1, β_2 and ϵ are hyperparameters, that weren't changed during this work based on recommandations in Ref. [17]. β_1, β_2 are generally close to 1, and ϵ a small parameter close to 0.
The Adam optimizer includes methods to further improve on the gradient. An exponentially decaying average of the gradient (Eq. (8)), and an exponentially decaying average of the squared gradient (Eq. (9)) is kept, and bias corrected in Eq. (10) and (11). Adam, like SGD, gives better results for a network if *batch training* is used, instead of taking the whole training data set and updating the parameters based on the mean error of all samples, batch training only uses a randomized batch for each update. This, on the one hand, introduces statistical fluctuation into the updates which can help avoid getting stuck in a local minimum, on the other hand, updates are done more often leading to smaller steps and better control with the hyperparameter of the batch size, which can be tuned.

2.3.7 Activation

There are several choices at each layer of the network for the activation function σ. A breakthrough in the performance of neural networks was achieved when the *Rectified Linear Unit* (short: *ReLU*) function was applied to hidden layers [21]. It has the following definition:

$$ReLU(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} = max(0, x), \tag{13}$$

and has several advantages over common *tanh* or *sigmoid* functions. The gradient is easy and fast to compute, there is no learning slow down because of vanishing gradients and the input is not projected unto (0,1). The last advantage can be a disadvantage if the *ReLU* function is the activation function for the output layer, so there is a different function applied. The *softmax* function has the benefit of being a probability distribution. It is defined as follows, with the output z (see Eq. (2)) of the neuron j in the layer l:

$$softmax(z_j^l) = \frac{e^{z_j^l}}{\sum_k e^{z_k^l}}. \tag{14}$$

The output of the network can therefore be seen as a probability. The most useful cut for a two class classification then needs to be determined by some metric, for example on a ROC curve.

2.3.8 Regularization

A common problem in the training of neural networks is *overtraining*. The backpropagation algorithm is only concerned with minimizing errors. The network can consist of over 100,000 parameters meaning that with sufficient time and parameters nearly any data can be fitted correctly. But rather than an accuracy of 100% we want the network to understand the fundamental difference between the classes we are trying to predict. So instead of letting the network pick up on things that are potentially only a statistical deviation which can happen for a small sample set, we want the network to gain some generalization power. *Overfitting* means that the performance on the training data gets better, while the performance on the testing data stays the same or gets worse. With *Regularization* one can avoid overfitting the network by introducing penalties for updates most commonly associated with overfitting. This often means dependence on a singular weight. There are two regularization methods applied in this work.

Dropout is a method appplied during training. With a certain rate α random neurons in one layer will get deactivated, so that the network will get trained without those connections. The performance will be worse during training, because of a smaller architecture with missing neurons, but a higher generalization is achieved in the final result. This is a result of the bias-invariance trade-off where one needs to find an optimal point between generalization and fitting.

L2-Regularization introduces an additional term in the loss function:

$$C_{L2} = C + \frac{\lambda}{2n} \sum_w w^2, \tag{15}$$

where λ is the strength of the regularization, a hyperparameter that needs to chosen. The squared sum of the weights in the cost will lead to smaller weights getting favoured by the optimizer. This is once again motivated by the notion that large weights typically lead to a drop in performance and should be avoided.

2.3.9 Weight initialization

Once weights and biases are chosen it is clear how the networks gets updated parameters during training. This algorithm does not include the initial state the network is in. A good way of initializing the weights is obviously choosing the weights out of a normal distribution with mean zero. The initial state of the network is very important, the complex structure of parameter space can lead to convergence problems if the parameters are not initialized properly. The parameters in one layer will get drawn out of a normal distribution with different standard deviations, there are two that are important for this work. *Glorot-normal* (also called Xavier-normal) is defined by

$$stddev = \sqrt{\frac{2}{(in + out)}} \tag{16}$$

and *He-normal* is defined by

$$stddev = \sqrt{\frac{2}{in}} \tag{17}$$

where *in* is the number of input connections to the layer and *out* the number of output connections. These were proposed in Ref. [22] and [8] respectively.

2.3.10 LoLa

LoLa is a neural network developed by Kasieczka et al. in Ref. [23]. It is a feed-forward network with two custom layers called Lorentz Layer and Combination Layer. Originally developed for tagging top jets over QCD background there is reason to believe that other particle jet events can also be tagged with it. This network is trained on the output of a usual collider simulator, meaning that it is fed with four-momentum of detected particles. After the transformation done in the *CoLa* and *LoLa* there are three dense layers, the first two are ReLU-activated and the output layer softmax-activated. The layers generally have a decreasing amount of neurons, the last layer only has two, giving out a probability for signal and background. The softmax-activation means that the output sums up to one.
The *CoLa* is a **Co**mbination **La**yer which aims to improve performance by adding linear combinations of the input vectors instead of relying on the network to do that itself. The input 4-momenta $k_{\mu,i}$ get multiplied with a matrix $C_{i,j}$. This matrix has trainable elements with the additional feature to include an identity matrix. With the identity

matrix included an untouched copy of the 4-vectors is included in the output of CoLa. The general form of $C_{i,j}$ is

$$C_{i,j} = \begin{pmatrix} 1 & 0 & \dots & 0 & C_{1,N+2} & \dots & C_{1,M} \\ 0 & 1 & & \vdots & C_{2,N+2} & \dots & C_{2,M} \\ \vdots & \vdots & \ddots & 0 & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & C_{N,N+2} & \dots & C_{N,M} \end{pmatrix}. \tag{18}$$

when the input is N 4-vectors and $M - N$ linear combinations should be included. M is a hyperparameter that needs to be tuned. These 4-vectors (now $\tilde{k}_j$) then get transformed in the **Lo**rentz **La**yer (LoLa):

$$\tilde{k}_j \to \hat{k}_j = \begin{pmatrix} m^2(k_j) \\ p_T(k_j) \\ w_{jm}^{(E)} E(k_m) \\ w_{jm}^{(d)} d_{jm}^2 \end{pmatrix}. \tag{19}$$

This should only be a loss-less rotation in observable space aimed to make the important information for the network easy to learn. A full description can be found in Ref. [23]. The LoLa makes use of the Minkowski metric, appearing in the first and last entry. This Layer was edited for this work to adapt the network for quark and gluon jets.

3 Setup and Analysis

3.1 Simulation of Events

We simulate our particle collision with *Pythia* v.8215 [24]. This is a high energy physics event simulator, which gives us full control over the generated samples. Along with *Sherpa*, it is used in particle phenomenology to work with collider physics data without some of the drawbacks in real-world applications. It is very important to note that these simulators work with perturbation approximation of the complete theory and therefore only produce results accurate to a certain extent. Additionally, it has been shown that using Sherpa or Pythia can lead to different results that DNNs are sensitive to [10, 25]. The general procedure of the simulators is the following. Matrix elements for the production of particles out of the initial ones are calculated. Light particles are radiated; heavier particles are decayed. Subsequent QCD radiation is emitted. This is called showering. Finally, the hadronization to hadrons and their decays is computed.
Both event generators use Monte-Carlo-methods by repeatedly random sampling the phase-space, in order to obtain numerical results. Because we are dealing with a large number of samples that leads to a sufficient approximation of real jet events.
The generated events are then processed in *Delphes* [26]. This is a fast-detector simulator which applies first real-world based cuts and requirements on our samples. Delphes offers the possibility to simulate explicit particle experiments (i.e. ATLAS or CMS). We require the detector-level particles to have $|\eta| < 2.5$ and $p_T \geq 1$ GeV. This is due to spatial and precision constraints in real-world experiments.
Delphes does not automatically identify jets. This is done using *FastJet* [27] v3.1.3, a program developed to allow different jet finding algorithms. We use a jet cone radius of $R = 0.4$ and cluster the particles into anti-k_T jets [28]. The anti-k_T jet finding algorithm is infrared and collinear safe and is resilient against pile-up for high energies at the LHC [28].
The samples we are producing and using are pure in the sense that every other process is excluded. We do not have any unrelated processes in our samples which would require us to further apply cuts.

3.1.1 Quark and Gluon Jets

For our pure quark and gluon jet samples we use Pythia and simulate dijet events and keep the subprocesses $gg/q\bar{q} \to q\bar{q}$ and $qg \to qg$ switched on for quark jets; for gluon jets we keep the subprocess $gg/g\bar{g} \to gg$ switched on. The process $qg \to qg$ is not taken into account here, because this would then not be pure quark-gluon jets. We require the summed transverse momentum of the constituents (written as $p_{T,j}$) to be between 200 and 220 GeV.
We produce 700000 events of quark and gluon jets and split them up for further use. The data shown to the network during training is the training data as well as the validation data, the networks parameters and hyperparameters are updated according to predictions on these. The testing data is fed to the network after the optimization is done to calculate

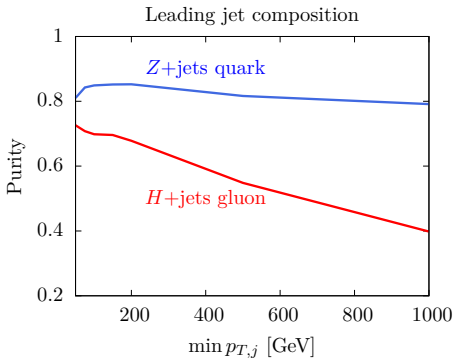

Figure 3: Jet composition of H+j as function of $p_{T,j}$

the ROC on an statistically independent sample. The samples Training, Validating and Testing have size 60%, 20% and 20%.

3.1.2 Monojets

For our samples of invisible Higgs + jet decays with according Z + jet background we use Sherpa. The Higgs is kept as stable particle in H+jets, and the Z is decayed to neutrinos. The resulting events are either quark or gluon jets. The events tagged as signal are usually gluon jets, in Fig. 3 we show the jet composition of H+jet and Z+jet as function of $p_{T,j}$. The events tagged as background are quark jets as well as gluon jets, the background is more pure than the signal as seen in Fig. 3.
We produce multiple samples for training, validating and testing, the samples contain jets with different $p_{T,j}$. We chose the slices 200-250 GeV, 300-350 GeV, 400-450 GeV, 500-550 GeV to sample the behaviour of the network. Each of the slices contain ~ 500,000 events which are again split up into 60% training, 20% validating and 20% testing.

3.2 Performance of LoLa

DeepTopLoLa was originally written for boosted hadronic top quark decay vs. QCD background discrimination. To start off, we trained the network with default parameters to confirm its integrity. We use Keras [29] with a Theano [30] backend on a GPU cluster with GeForce GTX Titan GPUs.
Fig. 4 in the attachment shows an example ROC curve. The data is from a training with default parameters. A ROC curve or *Receiver-Operating-Characteristic* curve will be our method to show a network's performance. Plotted is 1/false positive rate for a given true positive rate. There is an intuitive explanation for these plots. The networks output is the probability it is sure that the input was signal-like. A true positive is therefore a signal event that gets identified as such. A false positive is a background event that gets falsely labelled as signal from the network. The higher $1/(false\,positive\,rate)$ is, for a given true positive rate, the better is the network. The worst a network can do is random guessing. Then the false positive rate is as high a the true positive rate. In a ROC curve that is a diagonal line between (0,0) and (1,1). If a curve is even lower than that, we can simply

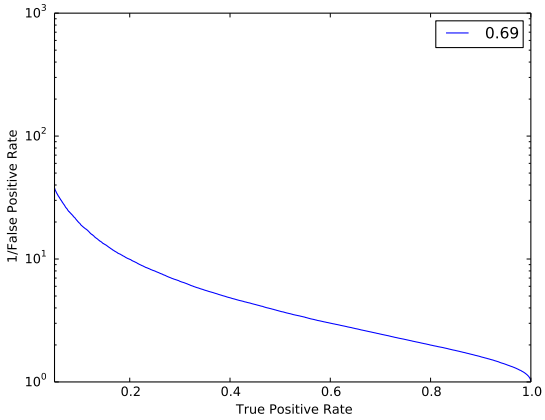

Figure 4: ROC curve for default setup on top vs QCD discrimination

negate the output and get a performance above the diagonal line.
From the ROC curve we can infer another metric for comparisons of different networks. The area under the curve (shortened to AUC) will give us a quick way of summarizing the networks performance. The worst case of random guessing has an AUC of 0.5, while the theoretical limit is an AUC of 1. In Fig. 4 we see an example ROC curve with an AUC of 0.69. The training and testing was done on top vs. QCD samples. This is obviously a different task to quark-gluon jet discrimination, but it illustrates that a optimization of a network can lead to very different results (a typical top tagger has an AUC upwards of 0.90).
The architecture of the model can be summarized in terms of the categories presented in Sec. 2.3. The LoLa layer takes as input 30 4-momenta, sorted by p_T. After that two dense layers with ReLU-activation functions follow, they have 100 and 50 units each; the last dense layer has two units and is softmax-activated. The optimization is done with SGD with a learning rate of 0.005. The loss gets computed as mean squared error.
For final performances it is obviously of interest to get a high true positive rate with the lowest false positive rate possible. The point of the best performance is not trivial to see in ROC curves, so we will later also show *significance improvement characteristic* (SIC) curves, where $Signal/\sqrt{Background}$ is plotted.
To specialize LoLa and to help the network learn faster we consider several variables

designed for quark and gluon jet discrimination [31]. They are defined below:

$$n_{PF} = \sum_{i_{PF}} 1 \tag{20}$$

$$w_{PF} = \frac{\sum_{i_{PF}} p_{T,i} \Delta R_{i,jet}}{\sum_{i_{\mathrm{PF}}} p_{T,i}} \tag{21}$$

$$p_T D = \frac{\sqrt{\sum_{i_{\mathrm{PF}}} p_{T,i}^2}}{\sum_{i_{\mathrm{PF}}} p_{T,i}} \tag{22}$$

$$C = \frac{\sum_{i_{PF}, j_{PF}} E_{T,i} E_{T,j} (\Delta R_{ij})^{0.2}}{(\sum_{i_{PF}} E_{T,i})^2} \tag{23}$$

$$x_{max} = \frac{max(p_{T,i})}{\sum_{i_{\mathrm{PF}}} p_i} \tag{24}$$

$$\sum_{i_{\mathrm{PF}}} p_i \cdot 0.95 = \sum_{i_{\mathrm{PF}}}^{N_{95}} p_i, \tag{25}$$

where the sum is over the jet constituents, which are particle flow (PF) objects. These include charged particles, neutral hadrons and photons. We define:

$$p_{jet} = \sum_{i_{\mathrm{PF}}} p_i \tag{26}$$

$$\eta_i = \log\left(\frac{E_i + p_{i,z}}{E_i - p_{i,z}}\right) \tag{27}$$

$$\Phi_i = \arctan2(p_{i,x}, p_{i,y}) \tag{28}$$

$$\Delta\Phi_{ij} = |\Phi_i - \Phi_j| \tag{29}$$

$$\widetilde{\Delta\Phi}_{ij} = \begin{cases} \Delta\Phi_{ij} & \text{if } \Delta\Phi_{ij} \leq \pi \\ 2\pi - \Delta\Phi_{ij} & \text{if } \Delta\Phi_{ij} > \pi \end{cases} \tag{30}$$

$$\Delta R_{ij} = \sqrt{(\eta_i - \eta_j)^2 + \widetilde{\Delta\Phi}_{ij}^2}. \tag{31}$$

Some remarks to them:

As we have already described quark and gluon jets differ in their number of constituents. It is therefore useful to include this number. We expect the number of constituents to be higher on average in gluon jets.

w_{PF} weighs the distance from the jet axis with the transverse momentum. Particles with a large deviation from the jet axis are therefore weighted less when they are soft. w_{PF} contains information about the spread around the jet axis. We expect gluon jets to 'spread' more than quark jets.

$p_T D$ is a measure for how many outliers in terms of transverse momentum are in the jet. For equal distribution among the constituents it tends to one, for an asymmetric distribution it tends to zero. We expect quark jets to have a higher $p_T D$ because the particles radiate less.

C is a two point correlation function. It correlates the transverse energies E_T of the

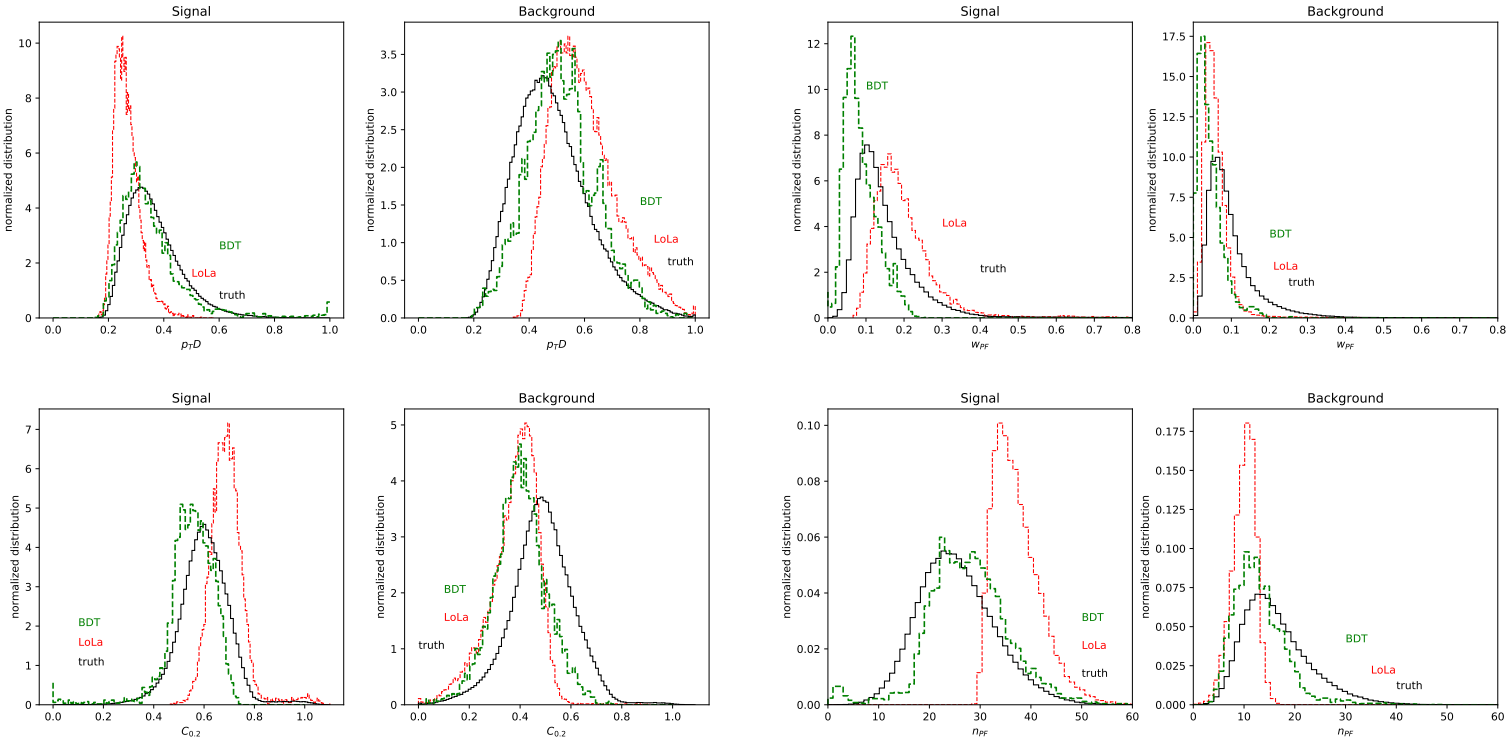

Figure 5: Histograms of the quark-gluon variables (part 1). Truth is computed with every event, LoLa and BDT is computed only with the 30% most signal- (or background-) like events. These are the correctly-identified events where LoLa and the BDT outputs the highest probability.

constituents while also taking their distance into account. As we already expect the gluon jets to spread more, we therefore also expect the energies of the constituents to be less aligned with the jet axis.

x_{max} is the highest share of p_T of one constituent in the jet. This variable is motivated by a variable with the same name in Ref [10] that works on images, where x_{max} would correspond to the highest fraction of total p_T contained in a single pixel. We expect quark jets to have a higher x_{max} based on the fact that gluon jets radiate more.

N_{95} is the minimum number of constituents required to get 95% of the whole jet p_T. The 95 is chosen like in Ref. [10], although there it was defined per pixel, This could, like in [32], be adjusted to a lower number, this fine-tuning was not done.

We computed the variables for the quark-gluon samples mentioned in Section 3.1.1 and binned them to make histograms. They are shown in Fig. 5 and 6, labelled as truth (black). We see a clear separation in most cases, though there is some overlap.

It is therefore of interest to see if and how the variables are correlated. We show a 2d histogram plot for the variables in Fig. 7 and 8.

Like expected, we see some correlation between the variables. The variable N_{95} is clearly correlated to n_{PF}, we can already see in their one-dimensional histograms that they only differ by a constant. A correlation between $p_T D$ and x_{max} can be seen, although $p_T D$ takes more into account than just the highest fraction (it is sensible to any outlier). Whenever

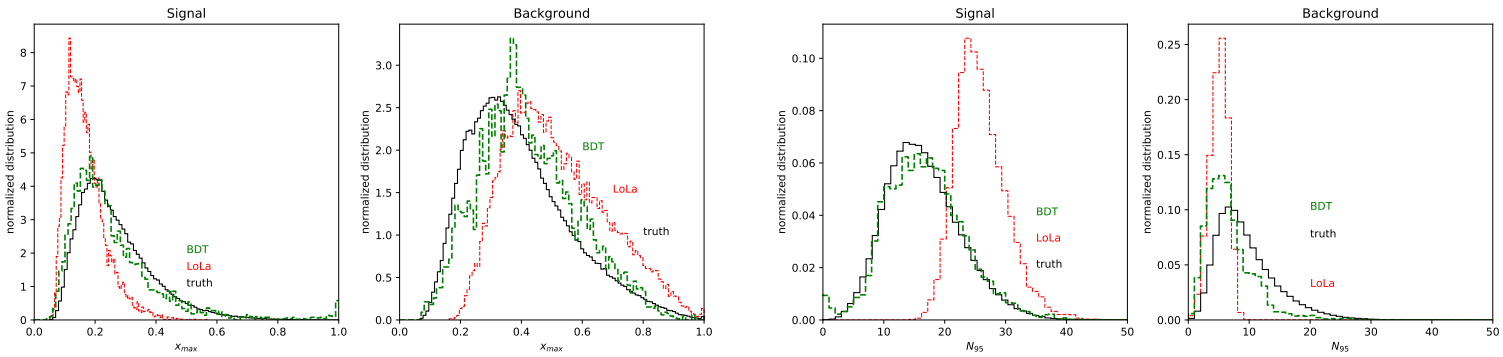

Figure 6: Histograms of the quark-gluon variables (part 2). Truth is computed with every event, LoLa and BDT is computed only with the 30% most signal- (or background-) like events. These are the correctly-identified events where LoLa and the BDT outputs the highest probability.

we see low correlation it is likely that the variables contain different information about the separation of quark and gluon jets. We see little correlation between $p_T D$ and n_{PF}, n_{PF} and x_{max}, and C and x_{max}.

The correlations imply that for a higher efficiency in terms of computing power and complexity it is possible to exclude certain variables, though this work will not focus on that.

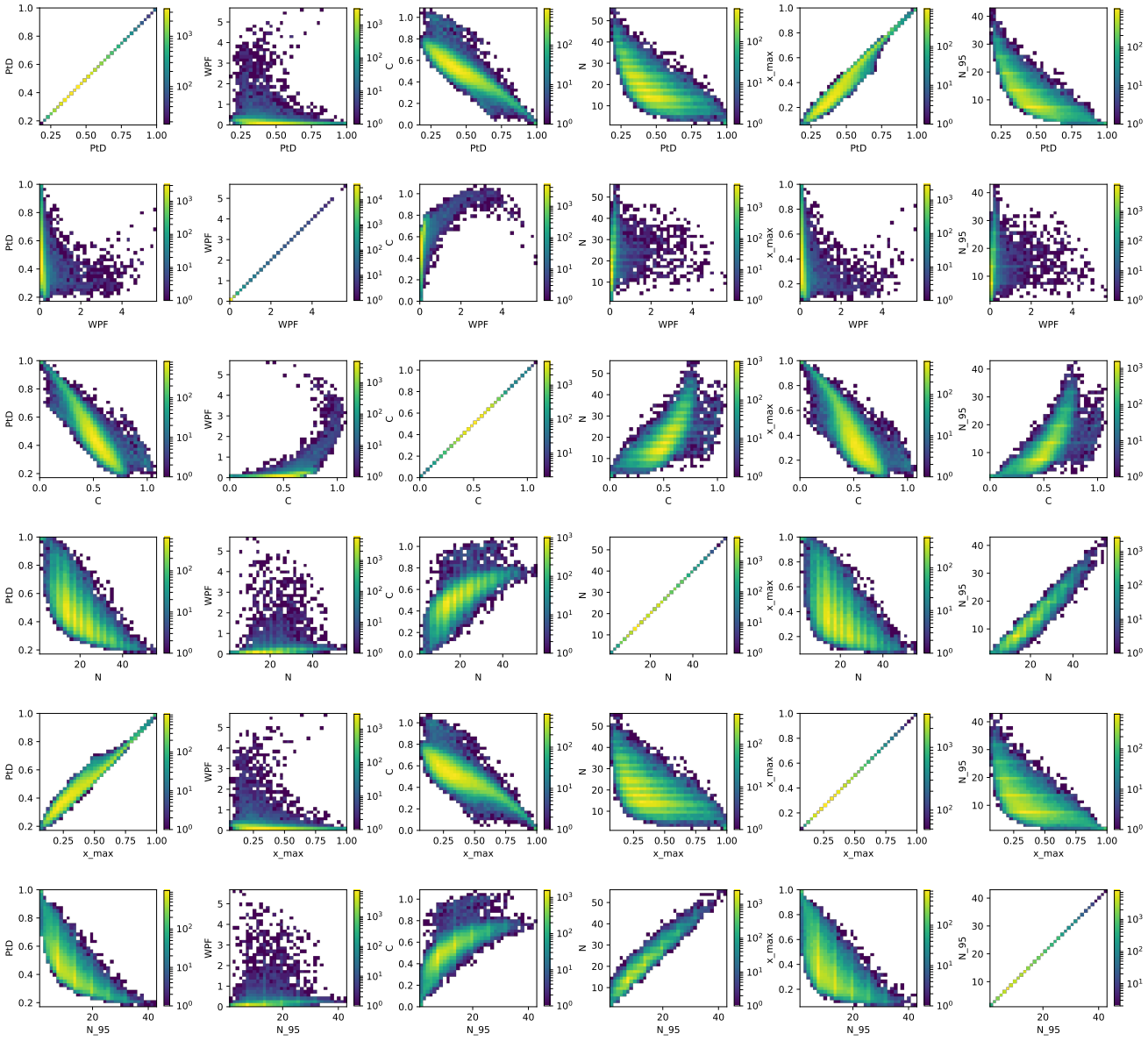

Figure 7: 2d histograms for the variables for quark jets.

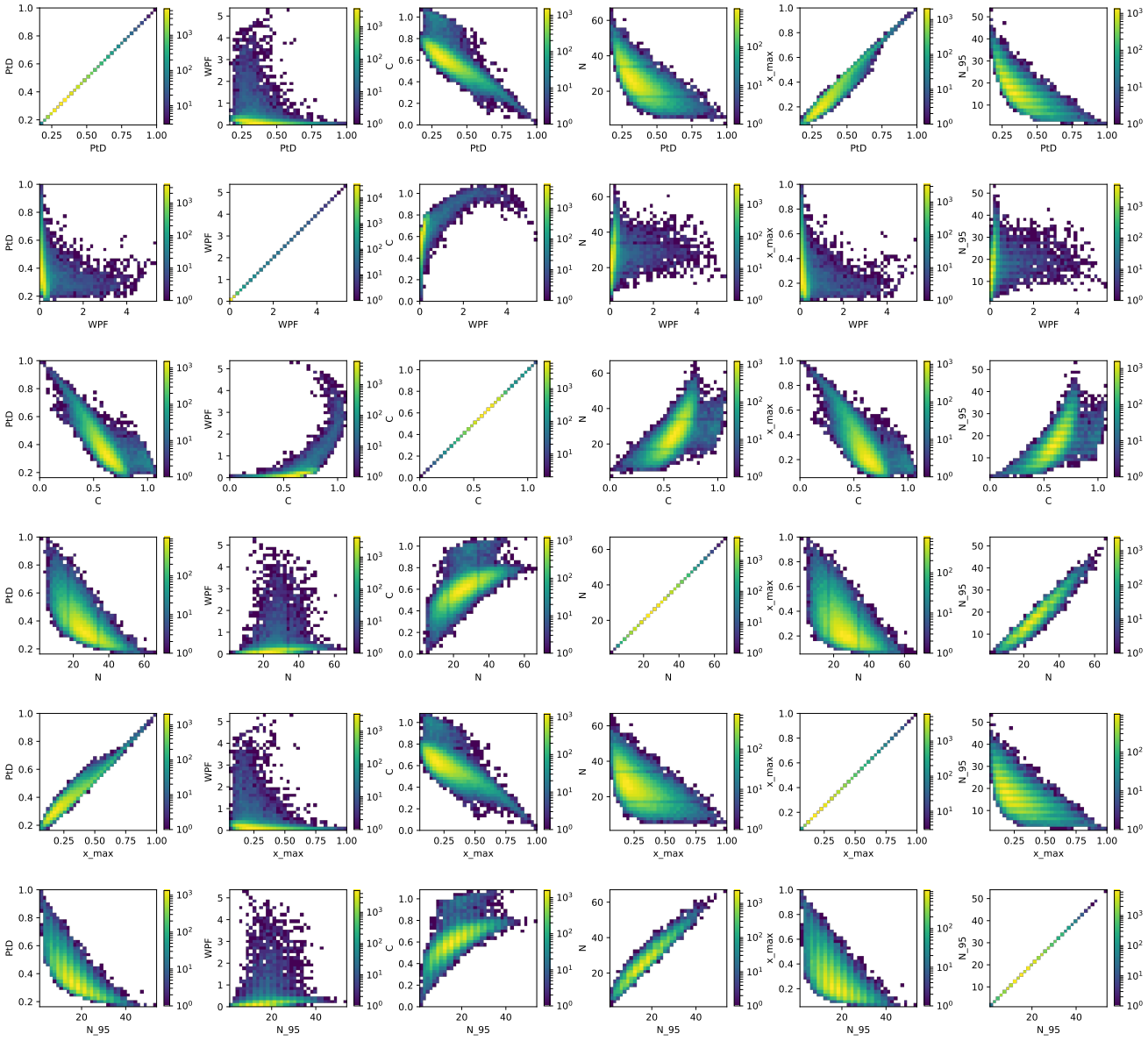

Figure 8: 2d histograms for the variables for gluon jets.

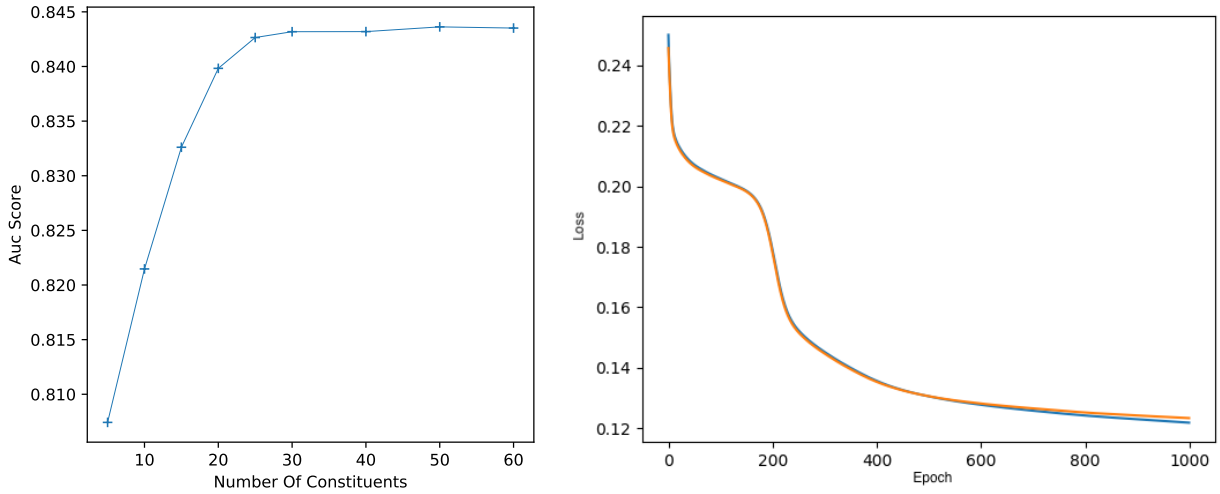

Figure 9: Left: AUC as function of number of constituents. The connection between the points is just for visibility and does not correspond to actual data. Right: A loss curve with stagnation during training, likely caused by non-optimal intialization functions for the weights.

3.2.1 Maximizing Performance

We present several ways we examined to maximize the performance of LoLa not just in terms of the metric AUC but also concerning time of training and training sample size. An instantaneous improvement in convergence can be made by switching the optimizer from SGD to Adam. As described, Adam will introduce new hyperparameters and can lead to quicker convergence. A problem we encountered early on is a stagnation of loss with subsequent normal behaviour. Fig. 9 (right) shows an example loss. The source to this problem, as it only occurs during the initial stages of the training, can be the weight initializing functions. Choosing correct functions is of importance for convergence time as seen in Fig. 9 (right). Setting the functions to 'glorot-normal' or, as in Ref [10], to 'He-normal' improves convergence. We see slightly better performance with 'He-normal' weight initialization.

In contrast to top decay with QCD background the number of constituents is higher and seems to be important for LoLa to discriminate quark / gluon jets. An initial sample had to be revised to include all jet constituents. In the samples themselves the constituents are ordered with decreasing p_T, so including more constituents automatically means including softer ones. We show the performance of the network measured in AUC in Fig. 9 (left) as function of the number of constituents that the network sees during training.

If we only consider the five particles with highest p_T the area under the curve is already over 0.80. The most information therefore seems to be in those particles. For the best performance we need to include 25 or more constituents. We can see that at roughly $n = 25$ the signal (truth) in Fig. 6 has its maximum.

The number of training samples used gives a look inside the networks training procedure.

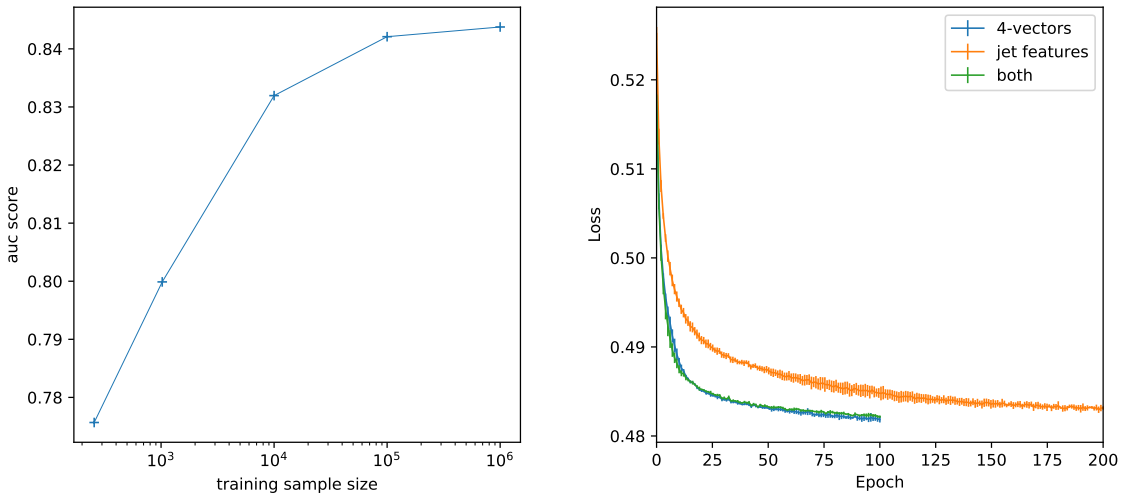

Figure 10: Left: The AUC as function of the training sample size. The connection between the points is just for visibility and does not correspond to actual data. Right: The mean loss function with error, stopping at 100 epochs for training with default LoLa parameters and stopping at 200 epochs for the quark-gluon variables on their own.

By limiting the number in training, one can check if a sufficient amount is used in full training. This is also part of an optimization, the training time scales of order $\mathcal{O}(n)$ with the training sample size n. In Fig. 10 we show the AUC as function of n. It can be seen that a higher n would result in minimal performance gain. The logarithmic scaling of the axis further implies that training would take exponentially more long. It is therefore not useful to increase the size of the training sample beyond the range we consider.

Although we included specific variables for the jet discrimination, LoLa is a very general network for particle physics discrimination appliances. We checked if an inclusion of the variables improves performance. An unchanged result would mean that LoLa is able to construct its own methods, maybe even reconstructing parts of the variables. A closer look at what networks specifically pay attention to has been done in Ref. [33].

We chose three settings of the network to compare the performance. For the first, no quark-gluon variables were enabled and the network only worked with the transformed 4-vectors in LoLa. For the second, the quark-gluon variables are included, and for the last, only the quark-gluon variables were switched on. The network was trained five times with each setting, to give a meaningful result with an error. Training was stopped after 100 epochs, only for the last setting training was continued until epoch 200. Fig. 10 shows the mean loss function, with standard deviation as error. The choice of 100 epochs was more than enough, only the last setting needed more time to converge to a minimum.

The performances each have AUC scores which can be averaged. Table 1 shows the AUC for each setting.

Some experimentation was made on how deep the network is. This included adding a

LoLa + qg variables	0.84358 ± 0.00005
qg variables	0.8437 ± 0.0001
LoLa	0.84369 ± 0.00008

Table 1: Comparison of Performance with variables switched off/on

fourth and fifth dense layer, again with ReLU activation, though because no improvement could be observed, these layers were removed again to keep the computation time low.
The maximal performance on quark-gluon samples processed in Delphes reaches an AUC of 0.84. The ROC curve is shown in Fig. 11, with the label Delphes. To see what events are ideal for the network to classify, we take the 30% that are correctly identified events that have the highest prediction output of the network and plot again the histograms of the variables. The same is done for the BDT. They are shown in Fig. 5 and 6 labelled LoLa and BDT. We see that the BDT is closer to the truth. That gives some information about the output probability of LoLa. While its performance is slightly better, it is further away from the truth in the histograms. LoLa therefore has more events that are classified correctly but have a low output probability. We can see that no event with n between 20 and 30 is in the 30% best identified events. Comparing the signal to the background histograms we always see a clear separation between them. That means in terms of the variables the events are very different; only these events produce a high output probability in LoLa.
The final model has the following architecture. CoLa is the first layer, followed by LoLa with the additional quark-gluon variables. After LoLa there are two dense layers with 100 and 50 units, both have a ReLU activation function. A dropout with rate 0.2 is applied after the first dense layer and with rate 0.1 after the second dense layer. An L2 regularization with strength 0.0005 is applied to both dense layers. The weights are initialized with He normal functions. The final dense layer has two units and is softmax activated. The learning rate is 0.0001 and we train with a batch size of 128.

3.2.2 Switching to Constituent Level

The variables defined in Eqs. (20 - 25) are working at jet-level. This means that for one event or jet each variable only has one value. This value is assigned to the jet. LoLa works on a constituent-level, taking 4-vectors as input to the network. Constituent-level means that for each constituent in the jet a value is defined. To bring the jet-level variables to the constituent-level they were edited to be defined per constituent. This includes taking out the denominator in most cases and removing sums over the jet constituents. Some variables are can not be defined in this way (e.g. the number of constituents would be just one for every constituent), so they were not included. The stripped down versions of the variables are:

$$w_{PF,i} = p_{T,i}\Delta R_{i,jet} \tag{32}$$

$$(p_T D)_i = p_{T,i}^2 \tag{33}$$

The variables N_{PF}, x_{max}, N_{95} are not included anymore, because they cannot be defined on a constituent-level. For C, three different versions are implemented. Instead of summing over the index j we can also maximize or minimize over it. This is motivated by the treatment for the fourth entry in Ref. [23] (Equation 6). They can be written like:

$$C_{sum} = \sum_{j_{PF}} E_{T,i}E_{T,j}(\Delta R_{ij})^{0.2} \tag{34}$$

$$C_{min} = \min_{j_{PF}} E_{T,i}E_{T,j}(\Delta R_{ij})^{0.2} \tag{35}$$

$$C_{max} = \max_{j_{PF}} E_{T,i}E_{T,j}(\Delta R_{ij})^{0.2} \tag{36}$$

In this new form the variables are appended to the output-vector of LoLa as defined in Eq. (19). A thorough analysis of whether each variation is necessary to include is not done, because computation time is low we just include every variation. The performance in AUC does not change, in fact, because of a faster implementation, the computation time for one epoch goes down. This means that we have a faster implementation while getting the same performance.

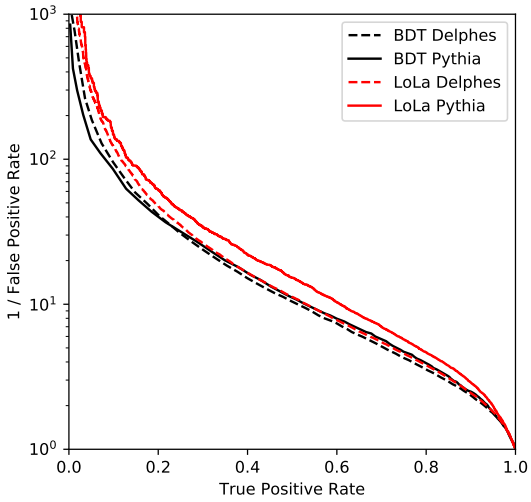

Figure 11: A BDT and LoLa on samples either processed in Delphes or taken directly from Pythia.

3.2.3 Impact of detector simulation

As already noted, the output of Monte-Carlo event simulators seems to differ, especially impacting the performance of networks on quark-gluon jet discrimination. This is also true for information degradation in detector simulator like Delphes. Fig. 11 shows the performance on samples after processing by Delphes and taken directly from Pythia. We see a drop in AUC and it is clear that the BDT and LoLa perform better when having more precise particle-level information that is unaffected by detector smearing. The simulated detector effects therefore make a difference. The drop in performance is bigger for the LoLa network, atleast in the range of 0.2 to 0.8 true positive rate. LoLa seems to gain more discrimination power when bypassing Delphes in the preprocessing steps. Real-world data would obviously be already smeared-out when coming out of a detector, so this gain in discrimination is only theoretical, though one has to note that the fast-detector simulation that Delphes does might not be a perfect recreation of real detector effects.

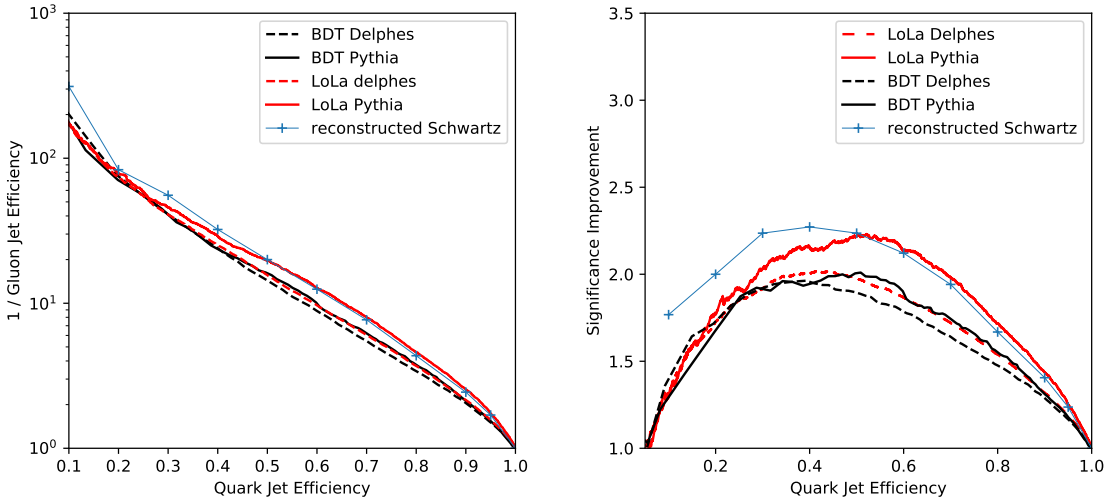

Figure 12: Left: ROC curve of LoLa, on Delphes processed samples, final state samples from Pythia and a reconstructed ROC curve from Ref. [10]. Right: SIC curves of the classifiers with a SIC curve from the reconstructed ROC curve of Ref. [10].

3.3 Comparing to a CNN

A different approach to quark-gluon discrimination has been proposed in Ref. [10]. The output of a detector gets interpreted as a picture, making the use of Convolutional Networks possible. The images fed to the network are in the (η, ϕ) space, while pixel intensities correspond to the transverse momentum that get measured by the calorimeters. Using Fig. 5 in Ref. [10] we can check if our network has a reasonable performance. In Fig. 12 (left) we show the ROC curve for LoLa and for the CNN. The curve for the CNN was reconstructed using EasyNData [34]. The performance is very similar, the CNN seems to be better at a low quark jet efficiency and slightly worse at a high quark jet efficiency, crossing over at 0.5 quark jet efficiency. We have to note though, that the reconstruction with EasyNData has an error that is not pictured here. Especially in the low positive rate the reconstruction is rather inaccurate, as one can see for a Quark Jet Efficiency of 0.2. Nevertheless, the comparison shows that LoLa performs reasonably well on quark-gluon jet samples.

A SIC curve is a useful way to show the signal over background discrimination power of a network. Here, the signal efficiency divided by the square root of the background efficiency is plotted as function of the signal efficiency. The curve has the same behaviour as the Gaussian significance $S/\sqrt{B}$. This curve has a maximum, at which point the best ratio for discrimination is achieved. In Fig. 12 (right) we show the SIC curve for the BDT, LoLa and the CNN. We see that the reconstruction probably has a large error, especially for low quark jet efficiency. Maximum classification power is achieved at ~0.4 quark jet efficiency for the CNN, while Lola peaks at ~0.55. It is visible that LoLa performs better for a quark jet efficiency of 0.5 and higher.

X	200 - 250	300 - 350	400 - 450	500 - 550	quark-gluon
200 - 250	0.794	0.664	0.645	0.634	0.769
300 - 350	0.678	0.781	0.645	0.635	0.744
400 - 450	0.590	0.641	0.772	0.701	0.694
500 - 550	0.655	0.664	0.718	0.734	0.674
quark-gluon	0.686	0.662	0.636	0.634	0.844

Table 2: Performance on monojet samples. The network is trained on the sample corresponding to the row and tested on sample corresponding to the column.

3.4 Tagging Monojets

Identifying the invisible decay of the Higgs boson together with a monojet is a very useful application of quark-gluon tagging. One naturally expects a drop in performance, because the sample is impure, meaning that it contains quark jets labelled as signal and vice versa. It is clear from Fig. 13 (left) that the learning takes longer and is not as efficient as before. The slow learning would imply that a higher learning rate leads to the same result in fewer epochs, but we do not observe this behaviour. Instead, the loss then converges to a higher point. Therefore, a low learning rate is still used, but the number of epochs for training is increased. Fig. 13 (right) shows an experimental training with a scheduled learning rate drop at 400 epochs and 800 epochs. The final gain in performance is only minimal, the learning rate drop mostly helps in stabilizing the model during training.
An equal behaviour is observed with an increased batch size, where the loss converges fast towards a minimum that is above the minimum priorly achieved.
The final model for monojet discrimination has the same architecture as the model for quark-gluon discrimination, though the training is continued up to 1000 epochs. The training time therefore is much longer, around 24 hours.
To study the performance on monojet samples, we train the network on all samples with different $p_{T,j}$ separately, as well as on the pure quark-gluon jet samples and test all instances on all samples, so that the generalization power of each instance is recorded. Table 2 shows the performance in terms of AUC. The indicator in each column and row mentions the range of the $p_{T,j}$ of each sample in GeV. The model is trained on the slice indicated by the row and tested on the slice indicated by the column.
We observe, that the best performance of 0.794 is obtained on the samples with the lowest $p_{T,j}$. As expected the network does the best when tested in the range it was trained on, though the performance clearly decays with higher $p_{T,j}$, in the range of 500 - 550 GeV $p_{T,j}$ the AUC is only 0.734.
The network trained on pure quark-gluon samples (which are in the range of 200 - 220 GeV $p_{T,j}$), also has its best performance on the sample with 200 - 250 GeV $p_{T,j}$, besides the already observed performance of 0.844 on pure quark-gluon jets.
It is important to note that no exact statistical errors can be given. To be able to give an error, the network would have to be trained multiple times in a row, which was not possible due to time constraints. A rough error estimation can be given, because the AUC

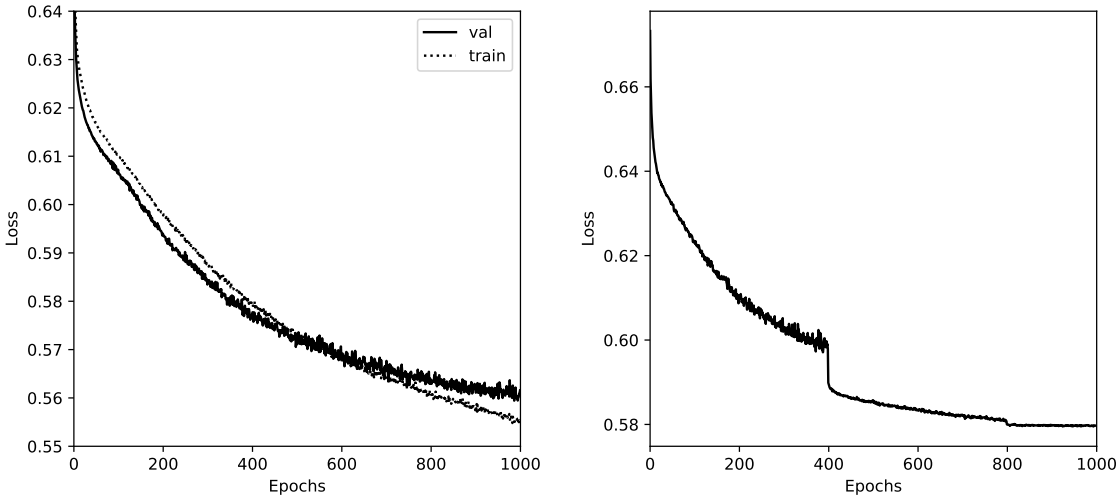

Figure 13: Left: Default loss curve on 200 $p_{T,j}$ monojet samples. Right: Validation loss curve on 300 $p_{T,j}$ monojet samples with a scheduler for learning rate.

was calculated in each epoch, though only on the validation sample. Extrapolating to the test sample, every AUC given in table 2 has an uncertainty of about 0.002. This error only comes from the forced update to the weights of the network, so decreasing the change by lowering the learning rate would lead to a smaller error here. Training the network multiple times in a row would allow the computation of a true statistical error, when the optimizer would converge to a different minimum. Our network is stable enough though, that the given rough error is a good estimate of the real error.

4 Conclusion

In this thesis we have studied a physics-inspired DNN applied to quark-gluon jet discrimination on pure samples and further on monojet samples. We edited the output of a custom layer to specifically include quark-gluon variables, designed to make the discrimination and learning process easier. The quark-gluon jet samples were produced with Pythia and we described the processes and cuts applied to simulate real events.
The performance, measured in AUC, is shown to be proportional to the number of training samples used. We show that the number of events used in the final performance is enough to saturate performance, while still allowing a reasonable computing time.
Additionally, we show that performance is dependent on the number of constituents included in the jet. The shown histograms indicate that we are able to include all constituents in the final performance because computation time is not too high. We saw the performance already reaching its maximum before including every constituent.
We studied whether the newly included variables have a positive effect on performance. They do not, but we see an equal performance when only training on the quark-gluon variables. This could mean, that a maximum of information is captured in both the quark-gluon variables as well as the 4-vectors computed in LoLa.
The impact of a detector on the discrimination power of the network was studied with Delphes and compared to the performance of a BDT. LoLa outperforms a conventional BDT in both cases. The gain on Delphes processed samples is minimal but noticeable, while the performance on final state events is clearly better.
We compare the performance of LoLa on quark-gluon jet samples to a Convolutional Neural Network that works with images in the $\phi - \eta$ plane and see a similar, if not better in some areas, performance. Especially for a high signal efficiency our DNN shows a better discrimination.
On samples processed in Delphes within a range of 200 - 220 GeV of $p_{T,j}$ we can reach a performance of 0.84 AUC. The training takes about two to three hours.

The performance of the network was tested on monojet samples, specifically produced to simulate invisible Higgs decays with invisible Z decays as background. The invisible Higgs decay is to be understood as new physics. While clearly expecting some loss, because both signal and background contain quark and gluon jets, the best performance reaches 0.794, which is only slightly worse than on pure quark-gluon samples. We studied the performance on multiple samples with different $p_{T,j}$, seeing that a lower $p_{T,j}$ leads to a better discrimination. The generalisation power was tested as well, by choosing the training sample to have a different $p_{T,j}$ than the testing sample, we saw that the discrimination power largely carries over.
The network was also trained on pure quark-gluon jet samples and then tested on the monojet samples, to gain an insight of how much the network can generalize the distinction between quark and gluon jets. The AUC is lower than for training on monojet samples, though it is still in an acceptable range.

LoLa is better than a conventional BDT for discriminating quark and gluon jets. We can especially improve performance working with final state events. A good separation can be achieved on invisible Higgs decays vs. according background. The currently most used production channel for the Higgs boson is through weak boson fusion. This channel has a very clean signature, in contrast to the channel investigated here. WBF will therefore continue to be the most relevant production channel, although it has the downside that is has a small cross section. Our approach with a DNN shows the potential of quark-gluon jet discrimination to the less studied Higgs+jet channel.

References

[1] F. Englert and R. Brout, *Broken symmetry and the mass of gauge vector mesons*, Phys. Rev. Lett. 13 (1964) 321.

[2] P. W. Higgs, *Broken symmetries, massless particles and gauge fields*, Phys. Rev. Lett. 13 (1964) 321.

[3] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble, *Global conservation laws and massless particles*, Phys. Rev. Lett. 13 (1964) 585.

[4] The ATLAS Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, arXiv:1207.7214 [hep-ex]

[5] The CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, arXiv:1207.7235 [hep-ex]

[6] The Super-Kamiokande Collaboration, Y. Fukuda et al, *Evidence for oscillation of atmospheric neutrinos*, arXiv:hep-ex/9807003

[7] Giorgio Arcadi et al., *The Waning of the WIMP? A Review of Models, Searches, and Constraints*, arXiv:1703.07364 [hep-ph]

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*, arXiv:1502.01852 [cs.CV]

[9] Liam Moore, Karl Nordström, Sreedevi Varma, Malcolm Fairbairn, *Reports of My Demise Are Greatly Exaggerated: N-subjettiness Taggers Take On Jet Images*, arXiv:1807.04769 [hep-ph]

[10] Patrick T. Komiske, Eric M. Metodiev, and Matthew D. Schwartz, *Deep learning in color: towards automated quark/gluon jet discrimination*, arXiv:1612.01551 [hep-ph]

[11] Wikimedia Commons, Link (visited on 07/28/2018)

[12] LHCb collaboration, *Observation of $J/\Psi p$ resonances consistent with pentaquark states in $\Lambda_b^0 \to J/\Psi K^- p$ decays*, arXiv:1507.03414 [hep-ex]

[13] Sally Dawson, Christoph Englert, Tilman Plehn, *Higgs Physics: It ain't over till it's over*, arXiv:1808.01324 [hep-ph]

[14] Philippe Gras, Stefan Höche, Deepak Kar, Andrew Larkoski, Leif Lönnblad, Simon Plätzer, Andrzej Siódmok, Peter Skands, Gregory Soyez, Jesse Thaler, *Systematics of quark/gluon tagging*, arXiv:1704.03878 [hep-ph]

[15] M. Tanabashi et al. (Particle Data Group), *2018 Review of Particle Physics* Phys. Rev. D 98, 030001 (2018)

[16] Michael Nielsen, *Neural Networks and Deep Learning*, online book (visited on 07/28/2018)

[17] Diederik P. Kingma, Jimmy Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs.LG]

[18] John Duchi, Elad Hazan, Yoram Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, 12(Jul):2121-2159, 2011

[19] Matthew D. Zeiler, *ADADELTA: An Adaptive Learning Rate Method*, arXiv:1212.5701 [cs.LG]

[20] Geoff Hinton, Neural Networks for Machine Learning, online lecture

[21] Xavier Glorot, Antoine Bordes, Yoshua Bengio, *Deep Sparse Rectifier Neural Networks*, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pmlr-v15-glorot11a

[22] Xavier Glorot, Yoshua Bengio, *Understanding the difficulty of training deep feedforward neural networks*, pmlr-v9-glorot10a

[23] Anja Butter, Gregor Kasieczka, Tilman Plehn, Michael Russell, *Deep-learned Top Tagging with a Lorentz Layer*, arXiv:1707.08966 [hep-ph]

[24] Torbjörn Sjöstrand et al., *An Introduction to PYTHIA 8.2*, arXiv:1410.3012 [hep-ph]

[25] James Barnard, Edmund Noel Dawe, Matthew J. Dolan, Nina Rajcic, *Parton Shower Uncertainties in Jet Substructure Analyses with Deep Neural Networks*, arXiv:1609.00607 [hep-ph]

[26] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, M. Selvaggi, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, arXiv:1307.6346 [hep-ex]

[27] Matteo Cacciari, Gavin P. Salam, Gregory Soyez, *FastJet user manual*, arXiv:1111.6097 [hep-ph]

[28] Matteo Cacciari, Gavin P. Salam, Gregory Soyez, *The anti-k_t jet clustering algorithm*, arXiv:0802.1189 [hep-ph]

[29] Chollet, François and others, *Keras*, https://keras.io

[30] Theano Development Team, *Theano: A Python framework for fast computation of mathematical expressions*, arXiv:1605.02688 [cs.SC]

[31] Anke Biekötter, Fabian Keilbach, Rhea Moutafis, Tilman Plehn, Jennifer M. Thompson, *Tagging Jets in Invisible Higgs Searches*, arXiv:1712.03973 [hep-ph]

[32] J. Pumplin, *How to tell quark jets from gluon jets*, Phys. Rev. D 44 (1991) 2025

[33] Wojciech Samek, Thomas Wiegand, Klaus-Robert Müller, *Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models*, arXiv:1708.08296 [cs.AI]

[34] P. Uwer, *EasyNData*, link (visited on 11/08/2018)

Acknowledgements

I would like to thank Prof. Tilman Plehn for giving me the opportunity to work in his group and get an insight into actual research. I want to acknowledge Michael Russell for his guidance in this project and proof-reading this thesis. I also want to thank Theo Heimel, Michel Luchmann and Jennifer Thompson for helpful discussions and shared laughs. Finally i want to thank the whole group for the kindness and nice working atmosphere.

Erklärung

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