Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Lukas Fabrizio Klassen

born in Essen (Germany)

2022

Where are the HERWIG gluons?
Understanding generator dependencies in quark-gluon tagging using a bayesian dynamic graph convolutional neural network

This Bachelor Thesis has been carried out by Lukas Fabrizio Klassen at the
Institute for Theoretical Physics in Heidelberg
under the supervision of
Prof. Tilman Plehn

Abstract

PYTHIA and HERWIG are among the most popular Monte Carlo generators for quark and gluon jets. Recent studies have found that quark-gluon classifiers perform significantly worse on data generated by HERWIG compared to data generated by PYTHIA. This work introduces a bayesian classifier, Bayesian ParticleNet (BPN), which assigns both a prediction and an uncertainty for each jet. It is shown that the performance of the BPN is comparable to the performance of it's underlying architecture ParticleNet, a cutting-edge Dynamic Graph Convolutional Neural Network. The network outputs, together with a set of physically motivated observables, are used to systematically study the irregularities in the input data that cause the performance difference. It is found that HERWIG trained networks have difficulties in tagging gluons, which can be traced to a lack of gluon-rich regions in the observable space. Both prediction and uncertainty are shown to be strongly correlated to the particle multiplicity in the jet. It is further argued that a network's performance is sensitive to the seperability of the multiplicity distribution in the dataset.

Zusammenfassung

PYTHIA und HERWIG zählen zu den prominentesten Monte Carlo Simulatoren für Quark- und Gluon-Jets. Vergangene Studien zeigen dass Quark-Gluon-Klassifikatoren deutlich schlechtere Resultate liefern wenn sie auf HERWIG Jets getestet werden verglichen mit PYTHIA Jets. Diese Arbeit führt einen bayesischen Klassifikator, Bayesian ParticleNet (BPN), ein, welcher jedem Jet neben einer Klassifikationsaussage auch eine Unschärfe zuweist. Diese Arbeit zeigt, dass die Resultate des BPN vergleichbar sind mit den Resultaten des zugrunde liegenden ParticleNet, einem leistungsstarkten Dynamic Graph Convlutional Neural Network. Die Ausgabe des Netzwerkes wird mit physikalisch motivierten Observablen verknüpft um die relevanten Unterschiede in den Daten zu untersuchen. Dabei wird festgestellt, dass auf HERWIG trainierte Netzwerke Schwierigkeiten haben, Gluonen korrekt zu klassifizieren. Dies wird zurückgeführt auf einen Mangel an Phasenraum-Regionen mit hoher Gluonendichte. Es wird außerdem gezeigt, dass die Ausgabe des Netzwerkes sensibel gegenüber der Anzahl von einzelnen Teilchen im Jet ist. Es wird weiter argumentiert, dass die Tagging-Leistung sensibel gegenüber der Separierbarkeit der Anzahl-Verteilung ist.

Contents

1 Introduction

Particle jets are an omnipresent observation during proton-proton collisions at the Large Hadron Collider (LHC). Long considered an experimental annoyance, they have turned into a powerful tool in the search for new physics. The term "jet" generally refers to a collimated spray of particles. As jets are generally initiated by strong force interactions, they are described by quantum chromodynamics (QCD).
One of the essential questions to answer when analyzing a jet is the particle that initiated it, as jets initiated by different particles show different characteristics. A prominent example are jets initiated by hadronically decaying top jets, which show a distinct two-prong structure. These characteristics give rise to jet-tagging, the art of classifying a jet by the particle of its origin based on physical observations in the experiment.
One of the most essential jet-tagging tasks is the differentiation between quark and gluon initiated jets. Many searches for physics beyond the standard model rely on the observation of quark jets. Successfully filtering gluon jets has the potential to drastically enhance these searches, making quark-gluon discrimination a topic studied in numerous papers for the past decade. Quark-gluon discrimination is also one of the more challenging jet-tagging tasks. Unlike hadronically decaying heavy particles, jets initiated by quarks and gluons generally show no clear prong structure. The differences between them are more subtle. Numerous studies have introduced and studied QCD-motivated intermediate high-level observables [1–5] and their discriminative power in quark-gluon tagging. More recently, machine learning methods, particularily deep neural networks (DNNs), have been used to discriminate quarks from gluons, matching or exceeding intermediate observables [6, 7].
The majority of these studies rely on Monte-Carlo (MC) simulated data instead of experimental results. The MC simulators most frequently used in quark-gluon simulations are HERWIG [8] and PYTHIA [9]. It has been found that in terms of tractable observables, quarks and gluons are more distinguishable in PYTHIA then in HERWIG [4, 10, 11]. When using image-based convolutional neural networks (CNNs), the differences in the generated jets have led to a decreased performance of networks trained on HERWIG data compared to similar networks trained on PYTHIA [6]. Testing HERWIG jets on PYTHIA trained networks and vice versa has shown that the network's performance is correlated with the seperability of the testing data, not the training data, as shown in Fig. 1.
Systematic uncertainties in the understanding of the data is a major restriction on the use of

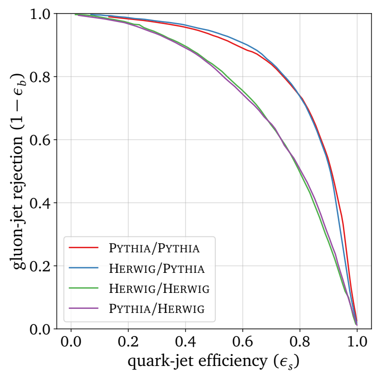


Figure 1: ROC curves for the PYTHIA- and HERWIG-trained CNNs applied to 200 GeV samples generated with both of the generators. Data taken from [6].

machine learning models in LHC physics. This work adresses these uncertainties by trying to untangle the PYTHIA-HERWIG-conundrum in quark-gluon tagging. Using a bayesian dynamic graph convolutional neural network (BDGCNN) that ouputs both a predictive score and a predictive uncertainty, a combination of the network outputs and tractable physical observables is used to systematically search for characteristics in the jet data that cause the performance difference.

Sec. 2 presents both the observables and the benchmark quark-gluon datasets used in this work. Sec. 3 outlines ParticleNet, a dynamic graph convolutional neural network for jet-tagging that was introduced in [12] and is used as the underlying architecture of a bayesian ParticleNet (BPN), which is introduced in Sec. 4. In Sec. 5, the results of the BPN are discussed with respect to both it's performance and it's classification output. The results are summarized in Sec. 6. In a final outlook several suggestions are made on how the knowledge of data uncertainty could be used to build more robust quark-gluon taggers.

2 quark-gluon observables and datasets

Despite the vast application of quark-gluon tagging, there is no strict experimental or theoretical definition of what either one of these jet types precisely is. A comprehensive study of alternative definitions is given in [2]. The definition widely used in jet-tagging is that quark or gluon jets refer to the parton which is produced in the hard process at leading order in perturbation theory and initiates the parton shower [3]. Gluon and quark jets are thus generated from $q\bar{q} \rightarrow Z(\rightarrow \nu\bar{\nu}) + g$ and $qg \rightarrow Z(\rightarrow \nu\bar{\nu}) + (uds)$ processes which occur during proton-proton collisions.

2.1 Benchmark datasets

The datasets used in this work are part of the `EnergyFlow` package [13–15]. The generators used are Pythia 8.226 and Herwig 7.1.4. For each generator, the package contains two datasets with 2M total jets each, one dataset containing only up-, down-, and strange-quarks (*uds*), the other also containing bottom- and charm-quarks (*udsbc*). With a number of sophisticated methods for tagging heavy bottom- and charm-quarks already in place, most quark-gluon tagging focuses on light (*uds*) quarks. This analysis also exclusively features the (*uds*) datasets. The simulations use default tuning and shower parameters and are carried out at a center-of-mass energy of 14 TeV. Hadronization and multi-parton interactions are turned on and no detector simulation is performed. The jets are defined in FastJet 3.3.0 [16] via an anti-k_T-algorithm [17]. The radius limit is set to $R = 0.4$. For each event the leading jet is kept if $500\,\text{GeV} < p_{\text{T,jet}} < 550\,\text{GeV}$ and $\eta_\text{jet} < 1.7$. For a jet with n_C consituents, the jet is defined via

$$x_i = \left\{(p_\text{T}, \eta, \phi, \text{PID})_k\right\} \quad \text{with} \quad k = 1, \ldots, n_\text{C} \tag{1}$$

with the transverse momentum p_T, pseudorapidity η, azimuthal angle ϕ and the particle ID (PID). The particle ID follows the *Monte Carlo Particle Numbering Scheme* of the Particle Data Group (PDG) [18]. The integer values assigned via the PID scheme are highly irregular and are thus not considered to be a good input for classifiation algorithms. For any PID-related observable in the training of DNNs in this work, the PID value is transformed to an experimentally more realistic ID. Based on their PDG assigned PID value, particles are indicated as being one of only five particles types (electron $e^\pm$, muon $\mu^\pm$, charged hadron $h^\pm$, neutral hadron h^0, and photon γ) as well as having the charge q. These categories are based on particle flow reconstruction algorithms at ATLAS and CMS, where $h^\pm = \pi^\pm/K^\pm/p/\bar{p}$ and $h^0 = K_L/n/\bar{n}$. The IDs are included as boolean features (1 if the particle belonging to this category is *true* and 0 otherwise). The original PDG IDs as well as the boolean representation for all particle types generated by Pythia and Herwig are shown in Tab. 1. The jets contain different numbers of constituents and are thus zero-padded to form arrays of similar length. Each data set consists of equal parts quarks and gluons and is contained in 20 files with 100k jets each. The recommended data split for train/test/validation is 1.6M/200k/200k, which is used throughout this work if not indicated otherwise.

2.2 Tractable observables

There are two central QCD-motivated features that allow for the discrimination between quarks and gluons. The observables used in this work are based on these features and have shown good performance in distinguishing quarks and gluons on both MC generated and real world data.

- In leading order, the ratio of the particle multiplicity n_pf of a gluon jet vs. a quark jet is proportional to the ratio in color charges (QCD charges) of gluons ($C_\text{A} = 3$) and quarks

particle		PDG-ID	q	is($e^{\pm}$)	is($\mu^{\pm}$)	is($h^{\pm}$)	is(h^0)	is(γ)
zero-padding	—	—	0	0	0	0	0	0
photon	γ	22	0	0	0	0	0	1
pion	π^+	+211	+1	0	0	1	0	0
anti-pion	π^-	−211	−1	0	0	1	0	0
kaon	K^+	+321	+1	0	0	1	0	0
anti-kaon	K^-	−321	−1	0	0	1	0	0
long-lived kaon	K_L	+130	0	0	0	0	1	0
neutron	n	+2112	0	0	0	0	1	0
anti-neutron	$\bar{n}$	−2112	0	0	0	0	1	0
proton	p	+2212	+1	0	0	1	0	0
anti-proton	$\bar{p}$	−2212	−1	0	0	1	0	0
electron	e^-	+11	−1	1	0	0	0	0
positron	e^+	−11	+1	1	0	0	0	0
muon	μ^-	+13	−1	0	1	0	0	0
anti-muon	μ^+	−13	+1	0	1	0	0	0

Table 1: Left: particles generated in proton-proton collisions in HERWIG and PYTHIA
Middle: Particle IDs according to *Monte Carlo Particle Numbering Scheme* of the Particle Data Group (PDG) [18]
Right: For quark-gluon tagging with particle-ID information, the PIDs are included in an experimentally realistic way by using only five particle types (electron, muon, charged hadron, neutral hadron, and photon), as well as the electric charge [12].

($C_\text{F} = 4/3$) [3]:

$$\frac{\langle n_\text{pf,gluon}\rangle}{\langle n_\text{pf,quark}\rangle} = \frac{C_A}{C_F} = \frac{9}{4} \tag{2}$$

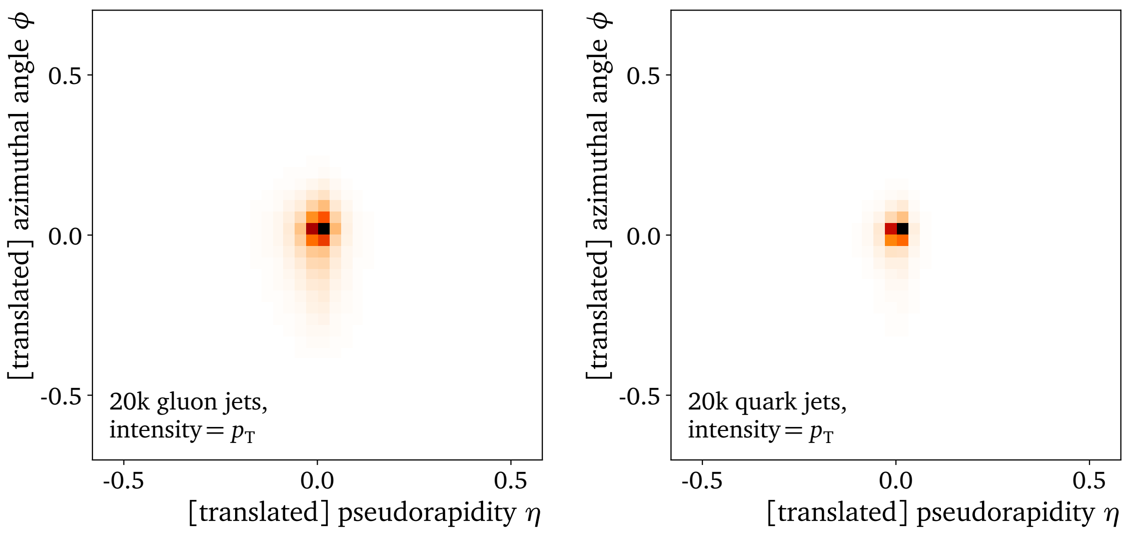
The larger color charge of gluons also leads to a wider linear radial moment (girth) w_pf of the jet, which can be visualized by looking at averaged quark and gluon jets in the (η, ϕ)-plane, see Fig. 2.

- In next-to-leading order, the splitting functions for quarks and gluons differ in the soft limit. This is measured via the generalized two-point correlation function [19] $C_1^{(\beta)}$, where quark-gluon discrimination has shown to work best for small positive β values ($\beta \simeq 0.2$). Due to the larger fragmentation of quarks, individual constituents in quark jets on average carry a larger part of the jet energy. This can be measured in the $\beta = 0$ limit of $C_1^{(\beta)}$ via $p_\text{T}D := 1 - C_1^{(0)}$ [20].

These four observables can be calculated algebraically using the raw data of the particle flow constituents which make up the jet:

$$\begin{aligned} n_\text{pf} &= \sum_{i_\text{pf}} 1 & w_\text{pf} &= \frac{\sum_{i_\text{pf}} p_{\text{T},i} \Delta R_{i,\text{jet}}}{\sum_{i_\text{pf}} p_{\text{T},i}} \\ C_{0.2} &= \frac{\sum_{i_\text{pf},j_\text{pf}} p_{\text{T},i} p_{\text{T},j} \left(\Delta R_{ij}\right)^{0.2}}{\left(\sum_{i_\text{pf}} p_{\text{T},i}\right)^2} & p_\text{T}D &= \frac{\sqrt{\sum_{i_\text{pf}} p_{\text{T},i}^2}}{\sum_{i_\text{pf}} p_{\text{T},i}}. \end{aligned} \tag{3}$$

n_pf simply counts the constituents in a jet. The girth w_pf adds the angular separation $\Delta R_i = \sqrt{\eta_i^2 + \phi_i^2}$ for each constituent and weights them with their respective transversal momentum $p_{\text{T},i}$ before normalizing with the total transversal jet momentum $p_{\text{T,jet}}$. $C_{0.2}$ takes the relative angular



(a) Averaged gluon-jet images (b) Averaged quark-jet images

Figure 2: quark and gluon jet images in the (η, ϕ)-plane, averaged over 20k individual images after preprocessing. The jet images are cropped to a 40×40 grid, covering $\eta = -0.566, \ldots, 0.566$ and $\phi = -0.684, \ldots, 0.684$. The color intensity corresponds to the binned p_T-magnitude. Due to the color charge ratio of 9/4, gluon jets (*left*) tend to have more constituents and a broader radiation pattern (girth) than quark jets (*right*).

separation $\Delta R_{ij} = \sqrt{(\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2}$ to the power of β and weights it with the product of the respective transversal momenta $p_{T,i} p_{T,j}$ before normalizing again with the squared total angular momentum. $p_T D$ sums over the squares of the transversal momenta, a value that is larger for a high variance in the $p_{T,i}$ spectrum, before again normalizing via summing the momenta of all constituents.
An additional discriminant for the girth is defined via ΔR_5, which is the mean angular seperation of the 5 jet constituents with the largest angular separation ΔR. An additional observable that identifies the larger fragmentation of quarks is the highest fraction of transversal momentum p_T contained in a single constituent, x_{max} [21]. The distributions of these observables for the 1.6M training jets from the benchmarking dataset are shown in Fig. 3. During training of the DNN's in this work, jets are typically cropped at 100 constituents in order to maintain data efficiency. Thus the distributions in Fig. 3 are also cropped at 100 constituents. The discriminative power of each observable can be estimated via the receiver operating characteristic (ROC) curve, which shows the inverse background mistag rate $(\epsilon_b)^{-1}$ as a function of the signal efficiency ϵ_s. The ROC curves and as well as the area under the curve (AUC) for all observables are shown in Fig. 4. The highest AUCs are found for the number of constituents n_{pf}, which aligns with the findings in [3]. For all observables the AUC is higher for PYTHIA then for HERWIG. This confirms prior results on the reduced seperability of HERWIG jets.

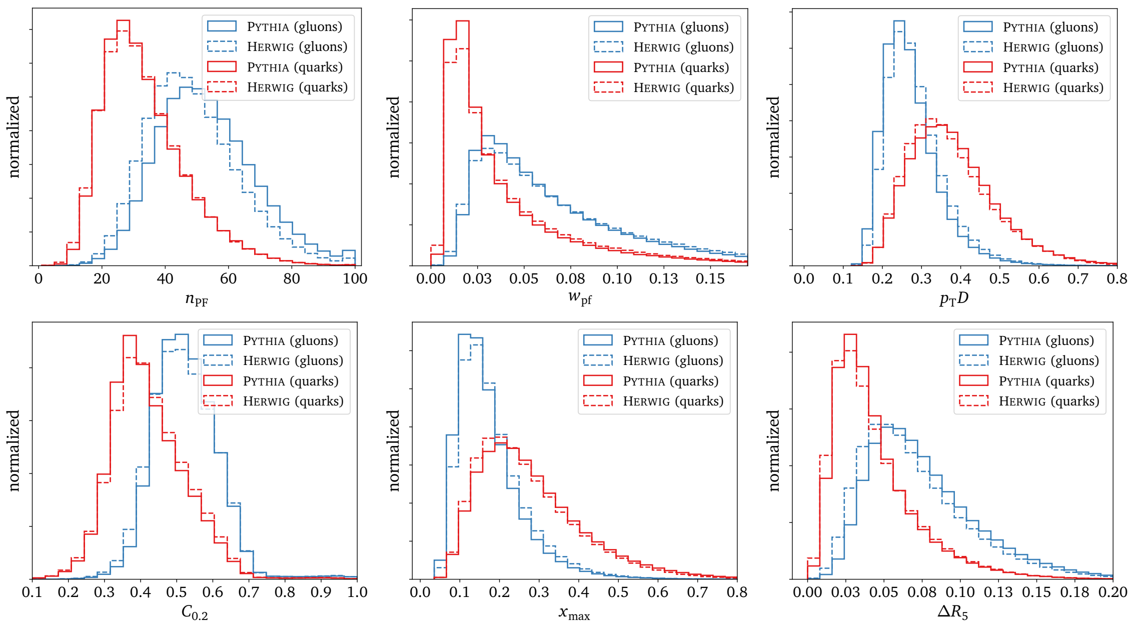


Figure 3: Observables for 1.6M training jets cut at 100 constituents for HERWIG and PYTHIA generated jets. The physically motivated positive correlations between n_{pf}, w_{pf} and ΔR_5 are evident, as well as the negative correlation between $p_{\mathrm{T}}D$ and $C_{0.2}$. Overall, the gluon distributions are shifted stronger between the generators then the quark distributions.

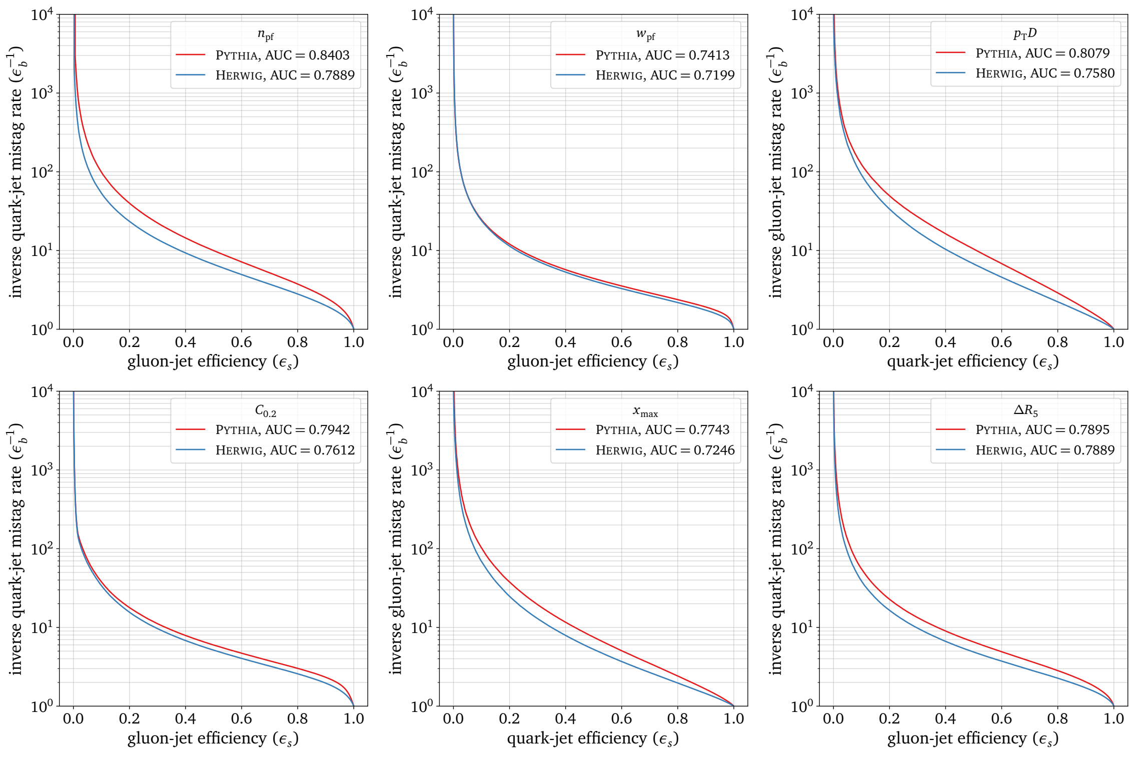


Figure 4: ROC curves and AUC's for 1.6M PYTHIA and 1.6M HERWIG jets computed via particle observables. For all observables the PYTHIA jets have a higher AUC then their HERWIG counterpart. As expected the n_{pf} observable generates the hightest AUC, indicating that in an uncorrelated analysis this observable distinguishes best between quarks and gluons.

3 ParticleNet: Jet Tagging via Particle Clouds

Using a bayesian classfier to analyze the systematic differences between the MC simulations requires a high performing backbone network. Strong performance metrics on common jet tagging benchmarking tasks are an indicator that the network does a good job at learning discriminative features in the data. Only once it is established that the network can learn such features can possible performance drops on alternative datasets be attributed to systematic differences in the training data.
The network chosen for this project is a light version of ParticleNet, a network that Qu and Gouskos introduce in [12]. In it's original publication it achieves state-of-the-art performance on differenct benchmark classification tasks, namely Top vs. QCD and quark vs. gluon discrimination. While recent transformer networks [22] have exceeded ParticleNet's performance, the large GPU time needed during training deemed transformer networks impractical for the aims of this resilience analysis. ParticleNet can be realized with limited computational resources and represents a good trade-off between performance and computational cost. It's uniqueness lies in the representation of the input jets: They are treated as permutation invariant particle clouds, which allows for the construction of a dynamically changing graph in the feature space as well as the use of edge convolution for efficient training and feature learning. Unlike jet images, this data structure inherits no a priori physical information. This should allow the network to learn without constraints the features which best discriminate quark- and gluon-initiated jets. The following subsections *particle clouds* and *network architecture* exclusively feature ideas, implementations and results from [12] with additional citations. While the orginal ParticleNet is implemented in TensorFlow/Keras, in subsection *PyTorch implementation and results* the network is reimplemented in `PyTorch` in order to simplify the process of adding features of a bayesian DNN. Subsection *Training ParticleNet-Lite* describes the differences in training between the original implementation and this work, highlighting in detail which aspects of the training do not overlap. The results from the original paper are treated as a benchmark to which the `PyTorch` implementation can be compared. In subsection *PyTorch implementation and results* it is shown that the re-implementation shows a comparable performance to the original version. This result renders the re-implementation suitable as a foundation for a new network, Bayesian ParticleNet (BPN).

3.1 Particle Clouds

All algorithms that aim to classify particle jets with respect to their particle of origin need as an input a specific representation of the input jet data. The way in which a jet is being represented has a great impact on it's suitability for specific deep learning models and their performance. A popular representation of jets is in the form of jet images: The energy of the jet constituents is mapped in the (η, ϕ)-plane before being binned to form the pixels of the jet image. With time, several pre-processing steps were introduced [23], boosting the performance of image based networks. Representing jets as images allows for the use of Convolutional Neural Networks (CNN's), which were popularized in the field of image recognition following the release and success of AlexNet [24] in 2012. CNN's have since been adapted and applied to a number of jet classification problems, including top tagging [23, 25, 26] and quark-gluon tagging [6, 7]. Despite the broad application of CNN's in particle physics, jet images as the underlying data structure have two key fallacies: First, they are computationally inefficient. In order to not miss any jet constituents while still providing a reasonable resolution in the area of interest, most particle tagging CNN's typically take input images with $\mathcal{O}(1000)$ pixels. At the same time, most jets across different classification problems and datasets contain only $\mathcal{O}(10)$ to $\mathcal{O}(100)$ constituents. This leaves a majority of the pixels vacant, thus passing a lot of empty values through the CNN's dense layers. The second downside lies in the information loss due

to the binning. Features such as particle ID, which are non-additive quantities of the individual constituents, cannot be considered.
It seems to be a more intuitive idea to just treat the jet simply as a set of constituent particles. Such an approach allows for calculation of any feature on constituent and jet level, including features that take into account the particle's ID. A choice has been to sort the particles in ordered lists used for recurrent neural networks (RNNs) [27–29] as well as binary trees used for recursive neural networks (RecNNs) [30,31]. Both RNNs and RecNNs are not indifferent to the order in which the particles of each jet are fed into the network. Thus it has to be assumed that the way the particles are sorted in the 1D lists or the binary trees affects the training in one way or the other. This poses the problem that, physically speaking, there is no inherent order to the particles that make up a jet. Sorting them by their transversal momenta p_T is the most popular option across the cited works. However this choice, while being practical, remains arbitrary.
[13] and [12] propose to regard jets as *particle clouds* - analogous to point clouds in computer vision. Mathematically speaking, a particle cloud representation is simply an *unordered* set containing the particles that make up a jet. For point clouds in computer vision, the points are generally considered to be correlated. Thus a network that takes point clouds as it's input is able to learn the internal structure of the higher dimensional object - jets in this case. The key to ParticleNet is that it makes use of both it's CNN like structure as well as the particle cloud representation of it's input data, which allows the use of a large set of features and enables the learning of small- and large-scale structures in the jet.

3.2 Network architecture

The success of CNNs can be largely attributed to two key reasons: First, the number of learnable parameters is greatly reduced by using a single kernel that is shared accross the entire image. The kernel's weights can use all locations in the image, allowing for effective learning. Secondly, convolutional layers can be stacked to form a deep network. Throughout these layers, features are learned in a hierarchichal approach [32]. Shallow layers learn local features while deeper layers learn more global features. Unlike images however, particle clouds do not feature the same grid-like structure on which a local kernel can easily be defined. Instead the very characteristic of a particle cloud is the irregular spacing of the unordered consituents. [33] proposes the edge convolution (EdgeConv), an operation that just like a classic convolution operation featues a kernel-like object which sees a large number of input features and which is also stackable. The point cloud is first represented as a graph where the vertices are the points themselves and the edges are the connections between each point and it's k neares neighbouring points. The way that for a regular convolution operation a local patch consists of the surrounding pixels, EdgeConv defines a local patch for each point as the set of the k nearest points. Let x_i denote a point in the point cloud and let $\boldsymbol{x}_i \in \mathbb{R}^F$ be the feature vector of x_i lying in the F-dimensional feature space. The features in $\boldsymbol{x}_i$ can depend on the coordinates of x_i but are not restricted to coordinate-dependent features (e.g. in the case of particle-ID related features). Let now $\{i_1, \ldots, i_k\}$ denote the indices of the k nearest neigbouring points of x_i. We then define the EdgeConv operation as

$$\boldsymbol{x}_i' = \mathop{\square}_{j=1}^{k} \boldsymbol{h}_\Theta\left(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}\right) \tag{4}$$

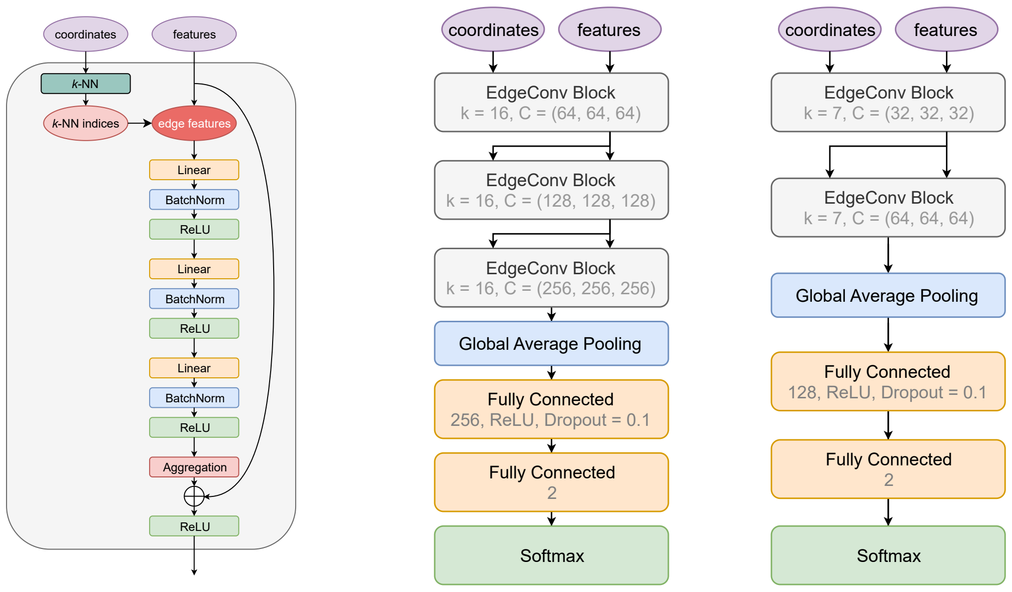
where $\square$ is a symmetric aggregation operation, i.e. max, sum or mean, and $\boldsymbol{h}_\Theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ is the so called *edge function*. The form of $\boldsymbol{h}_\Theta$ can be arbitrary, the function is parametrized by learnable parameters Θ and the same edge function is shared across all edges. $\boldsymbol{h}_\Theta$ is the equivalent to the kernel K in the image based convolution operation as it's parameters see the features of all points in the point cloud. $\square$ being invariant under permutation of it's input

feature vectors makes the EdgeConv operation permutationally symmetric, which is desireable due to the unordered nature of the input feature vectors. The EdgeConv operation also maps each feature vector $\boldsymbol{x}_i$ to a new feature vector $\boldsymbol{x}'_i$, meaning that it can be understood as a mapping of one point cloud to another. Thus is is possible to stack several EdgeConv operations, analogous to the multiple convolutional layers in an image-based CNN. After performing a single EdgeConv operation on a point cloud of points x_i, the original coordinates x_i can be used for determining the k nearest points before the updated feature vectors $\boldsymbol{x}'_i$ are used to determine a new generation of feature vectors $\boldsymbol{x}''_i$. However it has been shown to boost performance [33] if instead the k nearest neighbours are found with respect to the updated features $\boldsymbol{x}'_i$. This way the graph which is used to determine the k nearest neighbours is updated with each EdgeConv operation, which allows for the network to dynamically learn the proximity of points in the feature space. The dynamically changing graph results in the Dynamic Graph Convolutional Neural Network (DGCNN) [33].
ParticleNet takes the DGCNN idea and adapts it to best suit the problem of jet tagging. Fig. 5 shows a single EdgeConv-block in ParticleNet. For each particle it first finds the k nearest neighbouring particles based on the *coordinate* input to the EdgeConv block. For each particle and it's k nearest neighbours it then takes the corresponding *feature* input. ParticleNet uses a special form of the edge function

$$\boldsymbol{h}_\Theta\left(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}\right) = \overline{\boldsymbol{h}}_{\boldsymbol{\Theta}}\left(\boldsymbol{x}_i, \boldsymbol{x}_{i_j} - \boldsymbol{x}_i\right), \tag{5}$$

so instead of considering the actual features of the central particle and it's neighbouring particles, the difference between each of the k neighbouring particle's features and the central particle's features is calculated, resulting in the so-called *edge features*. The term *edge features* is appropriate as these features do not correspond to the neigbouring particles (vertices) themselves, but rather to the connection between them (edges). The k edge feature vectors are then forwarded into the edge function, which is implemented as a three-layer multi-level percepton (MLP). Each layer consists of a linear transformation, a batch normalization [34] and a ReLU layer [35]. After all k edge feature vectors are passed through the edge function, they are aggregated using the mean $\frac{1}{k}\sum$, which has shown to perform better then alternative aggregation functions such as max or sum. Inspired by ResNet [36], the k original feature vectors are passed down via a shortcut and are added to the aggregated output of the edge function. The combined feature vector is then again passed through a ReLU layer. The hyperparameters characterizing a single EdgeConv block are the number k of nearest neighbours considered for each point and the number of channels $C = (C_1, C_2, C_3)$ which determine the number of units in each linear transformation layer.

[12] introduces two alternative architectures, ParticleNet and ParticleNet-Lite, which are shown in Fig. 5b. ParticleNet-Lite is a smaller version of ParticleNet, which reduces the number of operations per input jet by an order of magnitude. ParticleNet (ParticleNet-Lite) consists of three (two) EdgeConv blocks, each using $k = 16$ ($k = 7$) nearest neigbours and the number of channels C being $(64, 64, 64)$, $(128, 128, 128)$ and $(256, 256, 256)$ ($(32, 32, 32)$ and $(64, 64, 64)$). The first EdgeConv-block takes as the coordinate input the (η, ϕ)-information from the raw data and as feature input the kinematic and optionally the ID-related features listed in Tab. 2 and Tab. 3. For both inputs the following EdgeConv blocks take the output of their preceeding block respecitively, adapting the idea of a dynamically changing graph. Afterwards the features of all particles in the particle cloud are aggregated via a global average pooling, a fully connected layer with 256 (128) channels, another ReLU layer and a dropout layer [37] with a drop probability of 0.1, which prevents overfitting. A last fully connected layer with 2 channels follows before a softmax function generates the output in the interval $[0, 1]$ used for the binary classification task.



(a) EdgeConv block structure. (b) ParticleNet (*left*) and ParticleNet-Lite (*right*) architectures.

Figure 5: *left*: A single EdgeConv block. Based on the *coordinate* input the k nearest neighbours for each particle are determined. Using the *feature* input, the edge features for each neighbouring particle are calculated and passed through a 3-layer MLP which acts as the edge function. Each MLP layer contains a linear transformation, a batch normalization and a ReLU activation. After the MLP, the features for all k particles are aggregated and the input features are added via a shortcut connection. Each EdgeConv block is characterized by the number of channels $C = (C_1, C_2, C_3)$ in the linear transformations as well as the number k of considered neighbours for each particle.
right: Two alternative architectures, ParticleNet (*left*) which features three EdgeConv blocks and ParticleNet-Lite (*right*) with two EdgeConv blocks. Both feature additional fully connected layers, an additional ReLu layer, a dropout layer and a softmax function generating the restricted output.
Taken from Ref. [12].

3.3 Training ParticleNet-Lite

Both architectures were implemented in this project, however the analysis relies entirely on ParticleNet-Lite. For the number of runs needed and the additional complexity introduced through the bayesian version, the computational cost of training the original ParticleNet exceeded the available resources.
The original implementation of ParticleNet and ParticleNet-Lite is based on TensorFlow/Keras and can be found on the authors GitHub repository. In order to simplify the process of adding bayesian characteristics, the network is first re-implemented using `PyTorch`. The training of the `PyTorch` implemented ParticleNet-Lite (PN-Lite) closely resembles the training described in [12]. This section highlights all similarities and differences in the training, which are also summarized in Tab. 4. In both implementations, the `AdamW` optimizer [38] with a weight decay of 0.0001 is used, minimizing the cross entropy loss

$$\mathcal{L}_{\text{BCE}} = \frac{1}{M} \sum_{m=1}^{M} -y_m \log f(x_m) - (1 - y_m) \log(1 - f(x_m)), \tag{6}$$

variable	definition
$\Delta\eta$	difference in pseudorapidity between the particle and the jet axis
$\Delta\phi$	difference in azimuthal angle between the particle and the jet axis
$\log p_T$	logarithm of the particle's transverse momentum p_T
$\log E$	logarithm of the particle's energy E
$\log \frac{p_T}{p_{T,\text{jet}}}$	logarithm of the particle's p_T relative to the jet $p_{T,\text{jet}}$
$\log \frac{E}{E_\text{jet}}$	logarithm of the particle's energy E relative to the jet energy E_jet
ΔR	angular separation between the particle and the jet axis ($\sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$)

Table 2: Kinematic input variables used in the quark-gluon tagging task [12].

variable	definition
q	electric charge of the particle (determined from the particle ID)
`isElectron`	if the particle is a electron $e^\pm$
`isMuon`	if the particle is a muon $\mu^\pm$
`isChargedHadron`	if the particle is a charged hadron $h^\pm = \pi^\pm/K^\pm/p/\bar{p}$
`isNeutralHadron`	if the particle is a neutral hadron $h^0 = K_L/n/\bar{n}$
`isPhoton`	if the particle is a photon γ

Table 3: Particle-ID input variables used in the quark-gluon tagging task [12].

where $f(x_m) \in [0,1]$ is the network's classification output for a single jet x_m and $y(x_m) \in 0,1$ is the truth label of the jet (0 for gluons, 1 for quarks). The epochs and their corresponding learning rates are slightly altered from the description in the original paper and instead follow the implementation as given in the authors repository:

$$\text{LR} = \begin{cases} 10^{-3} & \text{for epoch } \leq 10 \\ 10^{-4} & \text{for } 10 < \text{ epoch } \leq 20 \\ 10^{-5} & \text{for epoch } > 20 \end{cases} \tag{7}$$

The network is being trained for 30 epochs (24 in [12]), using a batch size of 128 (1024 in [12]). The smaller batch size is used due to memory constraints of the GPUs. Following [12] the network parameters are saved after each epoch and the model snapshot that has the highest accuracy on the validation data is used for the final evaluation.

For the initial EdgeConv block the network's *features* input takes a set of kinematic features shown in Tab. 2. Additionally, the *feature* input can contain the particle-ID related features shown in Tab. 3. As described in Sec. 2, the particle-ID related boolean features represent an experimentally realistic setup in which the exact ID of a particle may not be known but it can still be associated with a broader group of particles. In [12], ParticleNet and ParticleNet-Lite are trained both with and without particle-ID features, the results are given in Tab. 5. For quark-gluon tagging, [12] uses the full PYTHIA generated dataset described in Sec. 2 and the suggested train/validation/test split of 1.6M/200k/200k jets. For each jet only the 100 constituents with the largest transveral momenta p_T are being considered. The `PyTorch` implementation is also trained with the identical data split and p_T cutoff on PYTHIA as well as on HERWIG. In addition, it is trained and validated with a train/validation split of 400k/50k jets. This restricted training, which uses only 25% of the jets available, is carried out on both PYTHIA and HERWIG jets, in each case the training is repeated 10 times. Repeating the training allows quantitative statements on the uncertainties in the performance metrics.

training parameter	PN-Lite (TensorFlow/Keras)	PN-Lite (PyTorch)
batch size	1024	128
number of constituents	100, with highest p_T	100, with highest p_T
signal-to-background ratio	1.0	1.0
number of epochs	24	30
training/validation/testing	1.6M/200k/200k	1.6M/200k/200k, 400k/50k/50k

Table 4: Default ParticleNet-Lite (PN-Lite) training specifics for the original TensorFlow/Keras implementation (*left*) and the re-implemented PyTorch version (*right*).

network architecture	PID	accuracy	AUC	$\epsilon_b^{-1}(\epsilon_s=0.5)$	$\epsilon_b^{-1}(\epsilon_s=0.3)$
ParticleNet-Lite	✗	0.826	0.8993	32.8	84.6
ParticleNet	✗	0.828	0.9014	33.7	85.4
ParticleNet-Lite	✓	0.835	0.9079	37.1	94.5
ParticleNet	✓	0.840	0.9116	39.8 ± 0.2	98.6 ± 1.3

Table 5: Benchmark performance of the TensorFlow/Keras implementation of ParticleNet and ParticleNet-Lite on the quark-gluon tagging dataset (Pythia/Pythia), with and without PIDs [12].

3.4 PyTorch implementation: results

Tab. 6 shows the testing results for the PyTorch implementation trained/tested on 1.6M/200k jets. Looking at the results for the Pythia trained network and comparing it to the PN-Lite (with PID) results from Tab. 5, we see that while the original implementation has a slight edge in all performance metrics considered, the PyTorch implementation does have almost identical statistics. The training of DNN's is inherently statistical and determining precisely which aspects of the training or network implementation caused the slight difference in performance is all but easy. There is the possibility that choosing a batch size smaller by the order of one

network architecture	train/test	PID	accuracy	AUC	$\epsilon_b^{-1}(\epsilon_s=0.5)$	$\epsilon_b^{-1}(\epsilon_s=0.3)$
PN-Lite	Pythia/Pythia	✓	0.831	0.9049	36.3	91.0
PN-Lite	Herwig/Herwig	✓	0.761	0.8348	15.5	48.5

Table 6: Performance of the PyTorch implementation of PN-Lite trained/tested on 1.6M/200k Pythia and Herwig jets (different combinations) using the setting outlined in Sec. 3

magnitude might have affected the training, however a number of recent studies [39,40] have shown that for DNN's in general there is no clear correspondence between a larger batch size and a model's performance. It is worth noting that the PyTorch implementation of PN-Lite outperforms the larger ParticleNet trained without particle-ID related features. It also outperforms alternative quark-gluon discriminators available at the time that [12] was published, namely the Particle Flow Network (PFN) [13] and the Particle-CNN [41], as shown in [12].
Looking at both the receiver operating characteristic curves (ROC curves) shown in Sec. 3.4 as well as the performance metrics in Tab. 6, there is a sharp performance drop for the network trained/tested on 100% of all Herwig jets. For the networks trained on 25% of the datasets, Sec. 3.4 shows the ROC curves including errorbars which result from the multiple training

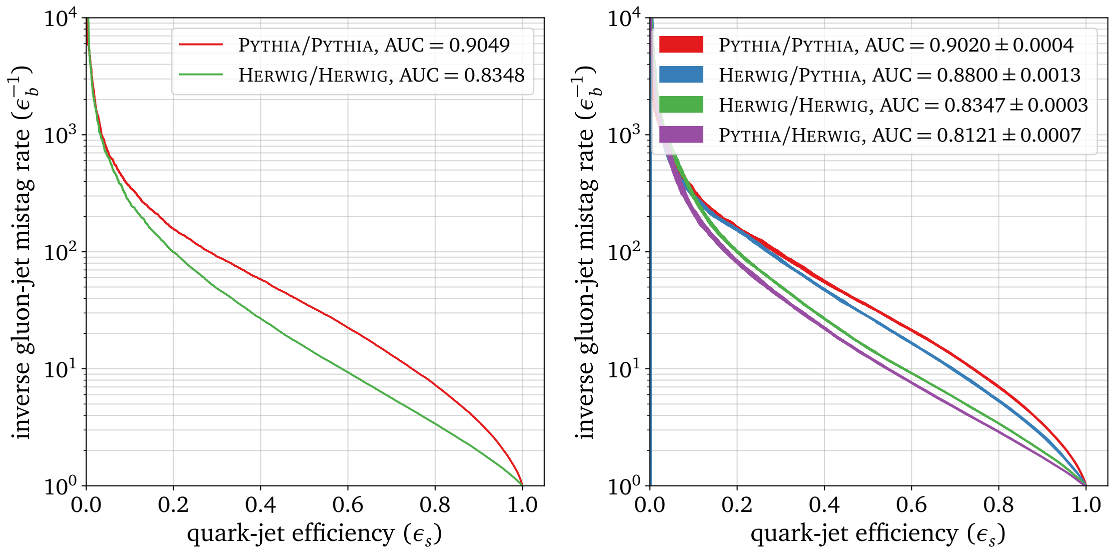


Figure 6: `PyTorch` implementation of PN-Lite trained/tested on 1.6M/200k (*left*) and 400k/50k (*right*) Pythia and Herwig jets (different combinations) using the training outlined in Sec. 3. The full statistics are shown in Tab. 6 and Tab. 7. The width of the curves on the right corresponds to the errorbars aquired through training iteration.

iterations. Comparing Tab. 6 and Tab. 7, is can be observed that for the train/test combinations Pythia/Pythia and Herwig/Herwig there is a small drop in performance across all metrics, which can be attributed to the smaller amount of training data. We additionally ob-

network architecture	train/test	PID	accuracy	AUC	$\epsilon_b^{-1}(\epsilon_s{=}0.5)$	$\epsilon_b^{-1}(\epsilon_s{=}0.3)$
PN-Lite	Pythia/Pythia	✓	0.829 ± 0.001	0.9020 ± 0.0004	34.7 ± 0.5	95.8 ± 2.7
PN-Lite	Herwig/Pythia	✓	0.796 ± 0.003	0.8800 ± 0.0013	28.1 ± 0.2	84.9 ± 2.3
PN-Lite	Herwig/Herwig	✓	0.762 ± 0.001	0.8347 ± 0.0003	15.1 ± 0.2	50.0 ± 0.8
PN-Lite	Pythia/Herwig	✓	0.718 ± 0.003	0.8121 ± 0.0007	12.7 ± 0.2	41.3 ± 1.1

Table 7: Performance of the `PyTorch` implementation of PN-Lite trained/tested on 400k/50k Pythia and Herwig jets (different combinations) using the training outlined in Sec. 3

serve that a network trained on Herwig performs significantly better when tested on Pythia jets then when tested on Herwig jets. While the combination Pythia/Pythia still yields the best results, the performance of the network seems to be stronger correlated to the jets used in testing then to the jets used in training. This is cemented by the performance of a Pythia trained network tested on Herwig, which performs even worse then the Herwig/Herwig scenario. Fig. 7 shows the predicted means and standard deviations on the 50k test jets for all four combinations of training/testing on Pythia and Herwig. The peculiarities in these distributions, such as the vacancy around $\mu_{\text{pred}} \to 0$ for Herwig trained networks, are reproduced by the bayesian classifier and will be discussed in detail in Sec. 4 and Sec. 5.

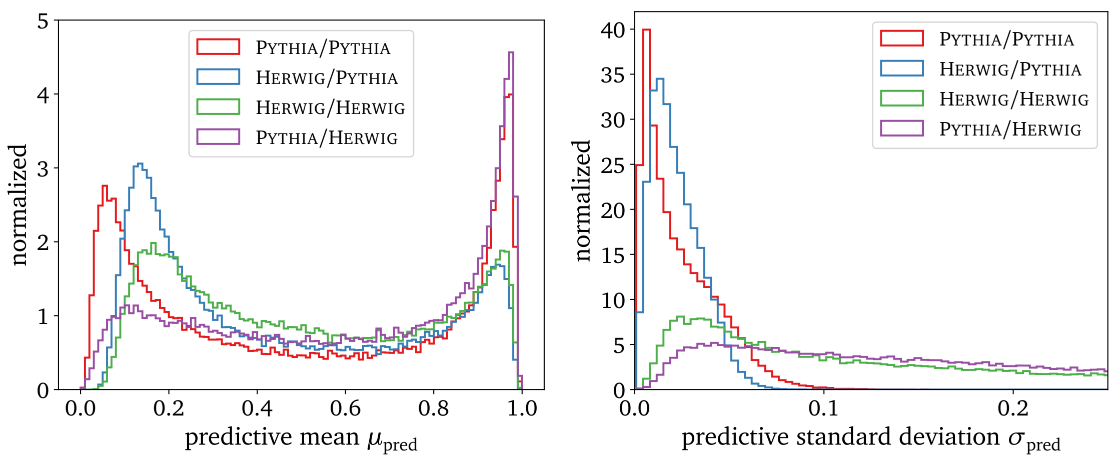


Figure 7: Predictive means and standard deviations from PN-Lite trained/tested on 400k/50k Pythia and Herwig jets (different combinations) using the training outlined in Sec. 3

4 Predictions plus uncertaincies: Bayesian ParticleNet (BPN)

Bayesian neural networks (BNNs) do not assign a classification output based on a fixed set of weights. Instead, they draw each weight from a learned distribution, yielding a probabilistic network output for which the uncertaincy can be quantified. [42] lists three possible sources of uncertainty in the particle physics context:

- finite but perfectly labeled training samples, coreresponding to a statistical uncertainty in the classification output, i.e. due to finite MC statistics;
- systematic inconsistencies in the training data and their labels, corresponding to systematic uncertainty in the classfication output;
- differences between the training and the test samples.

In the context of deep learning [43], two types of uncertainty are considered: (i) epistemic uncertainty, which describes the lack of statistics and can be reduced by adding more data/-training iterations and (ii) aleatoric uncertainty resulting from noise in the data which cannot fully be eliminated. [42] relates these uncertainties to the points listed above: epistemic uncertainty corresponds to statistical, aleatoric uncertainty to systematic unceratinties in the classification output. The latter is either related to irregularites in the input jets or is related to systematic differences between train and test data. Given the extent of the quark-gluon dataset described in Sec. 2 for both train and test data, it is most probable that uncertainties in the classification output are neither statistical, nor can they be attributed to systematic differences between train and test data. Therefore, observed uncertainties likely correspond to irregular input data. In the context of quark-gluon tagging, large uncertainties can point towards latent space regions in which quarks and gluons are hard to keep apart. As BNNs assign unceRainty jet-by-jet, they provide a powerful analysis tool: A jet with a large uncertainty is likely located in a latent space region where there is a large overlap between quark and gluon induced jets. This is the key motivation for adding bayesian features to the quark-gluon tagger introduced in Sec. 3. Bayesian networks have been implemented on different occasions in the context of particle physics [42, 44, 45], the following subsection *Bayesian neural networks* follows their descriptions of the mathematical principles behind bayesian deep learning. Subsection *Training BPN-Lite* outlines the training specifics of the network used in this work.

4.1 Bayesian neural networks

Given a training dataset $\mathcal{D} = \{(x_n, y_n)_{n=1}^N\}$, the weights ω of a BNN are not fixed but instead follow the posterior distribution $p(\omega \mid \mathcal{D})$, which according to Bayes' theorem can be obtrained via

$$p(\omega \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \omega) p(\omega)}{p(\mathcal{D})} \tag{8}$$

with $p(\mathcal{D} \mid \omega) = \prod_{n=1}^N p\left(y_n \mid x_n, \omega\right)$ being the likelihood of observing the training data $\mathcal{D}$ for a given network with parameters ω. $p(\omega)$ is the prior distribution which can be choosen freely and is in most applications a normal distribution with $\mu_\mathrm{p}, \sigma_\mathrm{p}$ being hyperparamters in the training. In any classification task, the goal is to predict the likelihood $p(c \mid x)$ of a test event x to be member of the class c. Following Eq. (8), $p(c \mid x)$ can be obtained via:

$$p(c \mid x) = \int d\omega p(\omega \mid \mathcal{D}) p(c \mid x, \omega) \tag{9}$$

with $p(c \mid x, \omega)$ being the network output. The integral can be interpreted as the sum over an infinitely large number of different weight configurations which are being weighted with their posterior probability $p(\omega \mid \mathcal{D})$. The posterior $p(\omega \mid \mathcal{D})$ cannot be directly accessed via a neural network, given the large number of parameters that need to be considered. A solution is given by variational interference [46], where the posterior is approximated by a distribution $q_\Theta(\omega)$ parametrized by learnable parameters Θ. This way Eq. (9) can be approximated as

$$p(c \mid x) \approx \int d\omega \, q_\theta(\omega) p(c \mid x, \omega), \tag{10}$$

which is solvable via Monte Carlo integration. In order to obtain the parameters Θ of $q_\Theta(\omega)$, the Killback-Leibler-divergence (KL-divergence)

$$\mathrm{KL}[q_\Theta(\omega), p(\omega \mid D)] = \int d\omega \, q(\omega) \log \frac{q_\Theta(\omega)}{p(\omega \mid D)} \tag{11}$$

is minimized. The KL-divergence is a measure of similarity between two distributions. It is 0 if the distributions are identical and positive otherwise. By minimizing the KL-divergence, q_Θ approximates $p(\omega \mid D)$ with increasing accuracy. Using Eq. (9), Eq. (11) can be rewritten as

$$\begin{aligned} \mathrm{KL}\left[q_\theta(\omega), p(\omega \mid \mathcal{D})\right] &= \int \mathrm{d}\omega q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega \mid \mathcal{D})} \\ &= \int \mathrm{d}\omega q_\theta(\omega) \log \frac{q_\theta(\omega) p(\mathcal{D})}{p(\omega) p(\mathcal{D} \mid \omega)} \\ &= \mathrm{KL}\left[q_\theta(\omega), p(\omega)\right] + \log p(\mathcal{D}) - \int \mathrm{d}\omega q_\theta(\omega) \log p(\mathcal{D} \mid \omega) \end{aligned} \tag{12}$$

The second term is called Bayesian bias term and as it does not depend on the parameters Θ, it can be omitted in the loss function. The first term depends on the chosen prior but not on the training data and can thus be regarded as a regularization term that prevents the model from overfitting. The third term includes the negative log-likelihood, resembling the cross entropy loss, weighted with q_Θ and integrated over the weight-space. Omitting the second term the loss function for a BNN is given by

$$\mathcal{L}_{\mathrm{BNN}} := \mathrm{KL}\left[q_\theta(\omega), p(\omega)\right] - \int \mathrm{d}\omega q_\theta(\omega) \log p(\mathcal{D} \mid \omega) \tag{13}$$

While the first term aims to minimize the difference between the chosen prior distribution for each weight and the tractable distribution $q_\Theta(\omega)$, the second term includes the log-likelihood which sums over all points in the training set,

$$\log p(\mathcal{D} \mid \omega) = \sum_{n=1}^{N} \log p\left(y_n \mid x_n, \omega\right). \tag{14}$$

Minimizing the negative log-likelihood is equivalent to finding the weights ω which maximize the probability of a point x_i being mapped to the correct label y_i. The integral over the log-likelihood can be approximated via sampling from the posterior distribution S times,

$$\int \mathrm{d}\omega q_\theta(\omega) \log p(\mathcal{D} \mid \omega) \approx \frac{1}{S} \sum_{s=1}^{S} \log p\left(\mathcal{D} \mid \omega_s\right) = \frac{1}{S} \sum_{s=1}^{S} \sum_{n=1}^{N} \log p\left(y_n \mid x_n, \omega_s\right) \tag{15}$$

where $\omega_s \sim q_\Theta(\omega)$ is the weight drawn for each sampling. In a realistic training setting, the sampling is not done on the entire dataset but instead on batches of size M, yielding a final BNN loss function

$$\mathcal{L}_{\mathrm{BNN}} \approx \mathrm{KL}\left[q_\theta(\omega), p(\omega)\right] - \frac{1}{S} \frac{N}{M} \sum_{s=1}^{S} \sum_{m=1}^{M} \log p\left(y_m \mid x_m, \omega_s\right). \tag{16}$$

Once the parameters Θ have been optimized, the prediction can be obtained via Eq. (9)

$$p(c \mid x) = \int d\omega q(\omega) p(c \mid x; \omega) \approx \frac{1}{N} \sum_{s=1}^{S} p\left(c \mid x, \omega_s\right) =: \mu_{\text{pred}} \tag{17}$$

with $\omega_s \sim q_\Theta(\omega)$. This predictive mean can be regarded as avering over an ensemble of networks with different weight configurations which are drawn from the learned distributions. The predictive standard deviation is accordingly defined via

$$\sigma_{\text{pred}}^2 := \frac{1}{S} \sum_{s=1}^{S} \left[p\left(y^* \mid x^*, \omega_s\right) - \mu_{\text{pred}} \right]^2 . \tag{18}$$

Predictive mean and predictive standard deviation form the combined output of a BNN. Most classification models map their unbounded output to a closed interval [0,1] using a final SoftMax layer, which in the special case of a binary classification problem is a sigmoid layer. [42] shows how such a sigmoid layer alters the BNN output distribution from it's original Gaussian shape and transform the uncorrelated outputs $\left(\mu_{\text{pred}}^{(\text{unconstr})}, \sigma_{\text{pred}}^{(\text{unconstr})}\right)$ into a parabolic correlation

$$\sigma_{\text{pred}} \approx \mu_{\text{pred}} \left[1 - \mu_{\text{pred}} \right] \sigma_{\text{pred}}^{(\text{unconstr})} \quad \text{with} \quad \mu_{\text{pred}} \in [0,1] \tag{19}$$

[42] argues that a poorly trained network will not output a parabolic correlation between μ_{pred} and σ_{pred}, meaning that this correlation can be treated as an evaluation criteria on whether the network trains properly.

4.2 Training BPN-Lite

When training BPN-Lite, both the prior and the weight distribution for each weigh is assumed to be Gaussian, with the mean and width of the prior $\left(\mu_p, \sigma_p\right)$ being fixed hyperparameters which are usually set to 0 and 1. Meanwhile the mean and width of the distribution from which each weight is sampled are learnable parameters $\left(\mu_q, \sigma_q\right)$. This way the KL-divergence Eq. (11) becomes

$$\begin{aligned} \text{KL}\left[q_{\mu,\sigma}(\omega), p_{\mu,\sigma}(\omega)\right] &= \frac{\sigma_q^2 - \sigma_p^2 + \left(\mu_q - \mu_p\right)^2}{2\sigma_p^2} + \log \frac{\sigma_p}{\sigma_q} \\ &= \frac{1}{2} \left\{ 2 \log \frac{\sigma_p}{\sigma_q} + \left(\frac{\sigma_q}{\sigma_p} \right)^2 + \left(\frac{\mu_q - \mu_p}{\sigma_p} \right)^2 - 1 \right\} . \end{aligned} \tag{20}$$

Setting $S = 1$, as it is usually done in order to minimize computational cost, and inserting 0 and 1 for $\left(\mu_{\text{p}}, \sigma_{\text{p}}\right)$ yields the complete loss function

$$\begin{aligned} \mathcal{L}_{\text{BPN}} &:= \frac{1}{N} \text{KL}\left[q_{\mu,\sigma}(\omega), p_{\mu,\sigma}(\omega)\right] - \frac{1}{M} \sum_{m=1}^{M} \log p\left(y_m \mid x_m, \omega\right) \\ &= \frac{1}{N} \sum_{i \in \{\omega\}} \frac{1}{2} \left\{ \mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1 \right\} \underbrace{- \frac{1}{M} \sum_{m=1}^{M} \log p\left(y_m \mid x_m, \omega\right) .}_{\text{(binary) cross-entropy loss}} \end{aligned} \tag{21}$$

The last term in Eq. (21) is still dependent on the weights ω. These weights can be reparameterized to make the term dependent on the learnable parameters $\left(\mu_q, \sigma_q\right)$ via

$$\omega \sim \mathcal{N}\left(\mu_q, \sigma_q^2\right) \implies \omega = \mu_q + \epsilon \sigma_q, \quad \epsilon \sim \mathcal{N}(0,1). \tag{22}$$

For each training iteration and weight the parameter ϵ is drawn from the normal distribution, after which the parameters $\left(\mu_q, \sigma_q\right)$ for each weight are minimized with respect to Eq. (21). The PyTorch implementation from Sec. 3 is transformed by replacing the linear layers in the EdgeConv blocks as well as the fully connected layers by bayesian linear and 2D-convolution layers taken from [47]. In these layers the weights ω are replaced by the Gaussian parameters $\left(\mu_q, \log\sigma_q^2\right)$ from which ω is obtained by sampling ϵ. The choice of using $\log\sigma_q^2$ as a parameter for minimization has been shown to stabilize the training [45]. The parameters in the batch-normalization remain regular learnable weights ω. The training statistics and architectures for a bayesian version of ParticleNet and ParticleNet-Lite are shown in Tab. 8.

The most notable difference is the number of epochs. [45] shows that BNN's can take significantly longer to converge, which is partially correlated to the larger number of learnable parameters. Thus training for more epochs guarantees a converging training. Again we only use the lite architecture for further testing due to computational constrains. The network is trained on both PYTHIA and HERWIG data seperately.

hyper-parameter	BPN	BPN-Lite
number of EdgeConv blocks	3	2
number of nearest neighbors k	16	7
number of channels C for	$(64,64,64)$	$(32,32,32)$
each EdgeConv block	$(128,128,128)$	$(64,64,64)$
	$(256,256,256)$	
channel-wise pooling	average	average
fully-connected layer	256 and ReLU	128 and ReLU
dropout probability	0.1	0.1
batch size	128	256
number of constituents	100, with highest p_T	100, with highest p_T
Bayesian bias terms?	no	no
re-sampling during validation?	no	no
training/validation/testing	400k/50k/50k	400k/50k/50k
signal-to-background ratio	1.0	1.0
number of epochs	100	100
prior mean $\mu_p = 0$	0	0
prior width σ_p	1	1
re-sampling for testing	50x	50x

Table 8: Default Bayesian ParticleNet (BPN) and BPN-Lite architectures [12]. Only the weights in the linear and 2D-convulutional layers are described by Gaussian distributions. The bias terms as well as the parameters in the batch-normalization layers are regular learnable parameters. The mean and width of the prior are given by $\mu_p = 0$ and $\sigma_p = 1$. The number of Monte Carlo samples used to calculate μ_{pred} and σ_{pred} is $S = 50$ (see Eqs. (17) and (18)).

4.3 BPN-Lite: results

The ROC-curves for BPN-Lite trained on PYTHIA and HERWIG are shown in Fig. 8, including cross-testing. The corresponding performance metrics are listed in Tab. 9. Across all train/test combinations and all performance metrics, the results of BPN-Lite are similar to the ones obtained for the deterministic ParticleNet-Lite (see Tab. 7). The distribution of the predicted mean μ_{pred} shown in Fig. 9 also resembles the distribution in Fig. 7.

The distribution of the prediction uncertainty σ_{pred} differs for the bayesian version, most notably for the networks trained on HERWIG. This difference can be attributed to the different

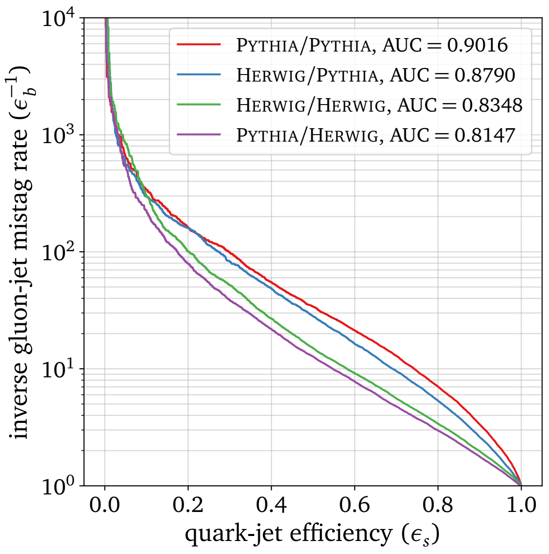


Figure 8: BPN-Lite trained/tested on 400k/50k PYTHIA and HERWIG jets (different combinations) using the default settings in Tab. 8.

network architecture	train/test	PID	accuracy	AUC	$\epsilon_b^{-1}(\epsilon_s=0.5)$	$\epsilon_b^{-1}(\epsilon_s=0.3)$
BPN-Lite	PYTHIA/PYTHIA	✓	0.820	0.9016	34.0	99.2
BPN-Lite	HERWIG/PYTHIA	✓	0.798	0.8790	28.2	82.5
BPN-Lite	HERWIG/HERWIG	✓	0.761	0.8348	15.0	52.1
BPN-Lite	PYTHIA/HERWIG	✓	0.689	0.8147	12.8	38.5

Table 9: Performance of the `PyTorch` implementation of BPN-Lite trained/tested on 400k/50k PYTHIA and HERWIG jets (different combinations) using the default settings in Tab. 8.

ways in which the uncertainties were obtained. For the deterministic version they are obtained

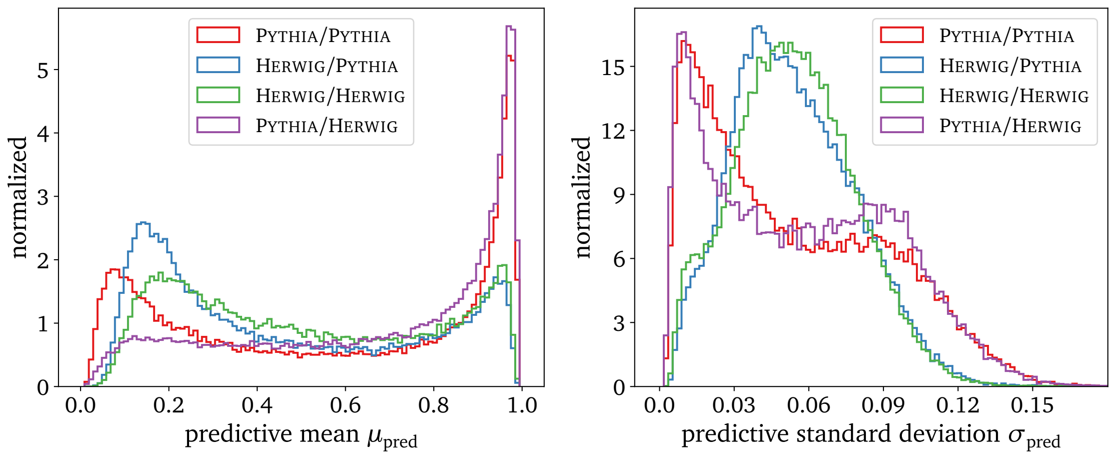


Figure 9: Predictive means and standard deviations from BPN-Lite trained/tested on 400k/50k PYTHIA and HERWIG jets (different combinations) using the default settings in Tab. 8.

by reiterating the training process multiple times. Thus the uncertainties shown in Fig. 7 are directly related to the statistical nature of the training algorithms involved. For the BPN however, the uncertainties correspond to systematic noise in the input data and can provide further insights. Fig. 10 shows the correlation bewteen μ_{pred} and σ_{pred}. The scattered points are transformed into a density map using Gaussian kernel estimation. For all train/test combinations

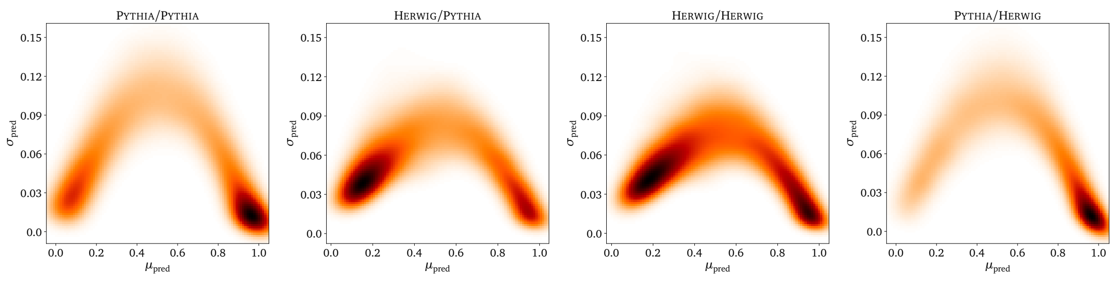


Figure 10: Correlation between μ_{pred} and σ_{pred} represented via a Gaussian kernel estimation. As expected, the curves roughly show a parabolic relation with deviations for certain combinations.

the distributions roughly resemble a parabola, as described in Eq. (19). Thus we can assume that the training ob the BPN is generally stable for both HERWIG and PYTHIA training data.

5 The PYTHIA-HERWIG conundrum

The differences in performance and in the distributions of μ_{pred} and σ_{pred} are rooted in differences in the training and testing data. From a human perspective it is not possible to know which high-level features in the data are learned by the network. Tractable physical observables and their correlation to the network output are a limited yet insightful way to make sense of the difference in behaviour between PYTHIA and HERWIG.
The results from the BPN shown in Fig. 8 and Fig. 9 allow for three key observations:

- The performance of the network is strongly dependent on the testing data and weakly dependent on the training data. Both networks tested on PYTHIA perform significantly better then the networks tested on HERWIG across all performance metrics.
- The μ_{pred} distribution for both HERWIG-trained networks is asymmetric, the area close to $\mu_{\text{pred}} \to 0$ is left vacant. It seems as if there were no HERWIG gluons.
- The σ_{pred} distribution for both HERWIG-trained networks has no clear peak at $\sigma_{\text{pred}} \to 0$. There is only a limited number of jets that a HERWIG-trained network classifies with confidence. Meanwhile PYTHIA trained networks also have a fair share of uncertain predictions with a small yet identifiable peak at $\sigma_{\text{pred}} \approx 0.09$.

The first observation has already been made in [6], where it is argued that the insensitivity to the training data confirms the robustness of DNN's in quark-gluon classification. However the observations in the predictive output, despite not having a large effect on the performance, are largely training dependent. This questions the robustness derived from the mere performance. The goal of this section is to understand above observations in terms of physical observables and give a more nuanced answer to the question of resilience in quark-gluon tagging.

5.1 Correlations and performance

It is a natural assumption that the classification output of the BPN corresponds to discriminative observables. A simple yet effective tool to determine which tractable observables correspond best with the features learned by the network is the Pearson's correlation coefficient (PCC)

$$\text{PCC} = \frac{\textbf{cov}(\mu_{\text{pred}}, x_i)}{\textbf{std}(\mu_{\text{pred}})\textbf{std}(x_i)} \tag{23}$$

between the predictive mean μ_{pred} and the tractable observable x_i. The PCC's for the observables introduced in Sec. 2 are shown in Tab. 10. While there is a significant correlation for all

train/test	n_{pf}	w_{pf}	$p_T D$	C_{02}	x_{max}	ΔR_5
PYTHIA/PYTHIA	−0.81	−0.46	0.70	−0.67	0.62	−0.64
HERWIG/PYTHIA	−0.82	−0.53	0.78	−0.77	0.69	−0.65
HERWIG/HERWIG	−0.83	−0.55	0.75	−0.76	0.65	−0.66
PYTHIA/HERWIG	−0.79	−0.47	0.61	−0.62	0.52	−0.66

Table 10: Correlation coefficients of the predictive mean μ_{pred} with tractable observables for 50k PYTHIA and HERWIG test jets.

observables, the correlation is the strongest for n_{pf}. The correlation is visualized in Fig. 11, where the n_{pf}-distribution is shown for different bins in μ_{pred}. As one would expect, jets which the network considers to be gluon-like have a large number of constituents while quark-like jets fall towards the lower end of the spectrum. The similarities in the bin structure for networks trained on the same generator shows that feature learning is strongly tied to n_{pf}, an

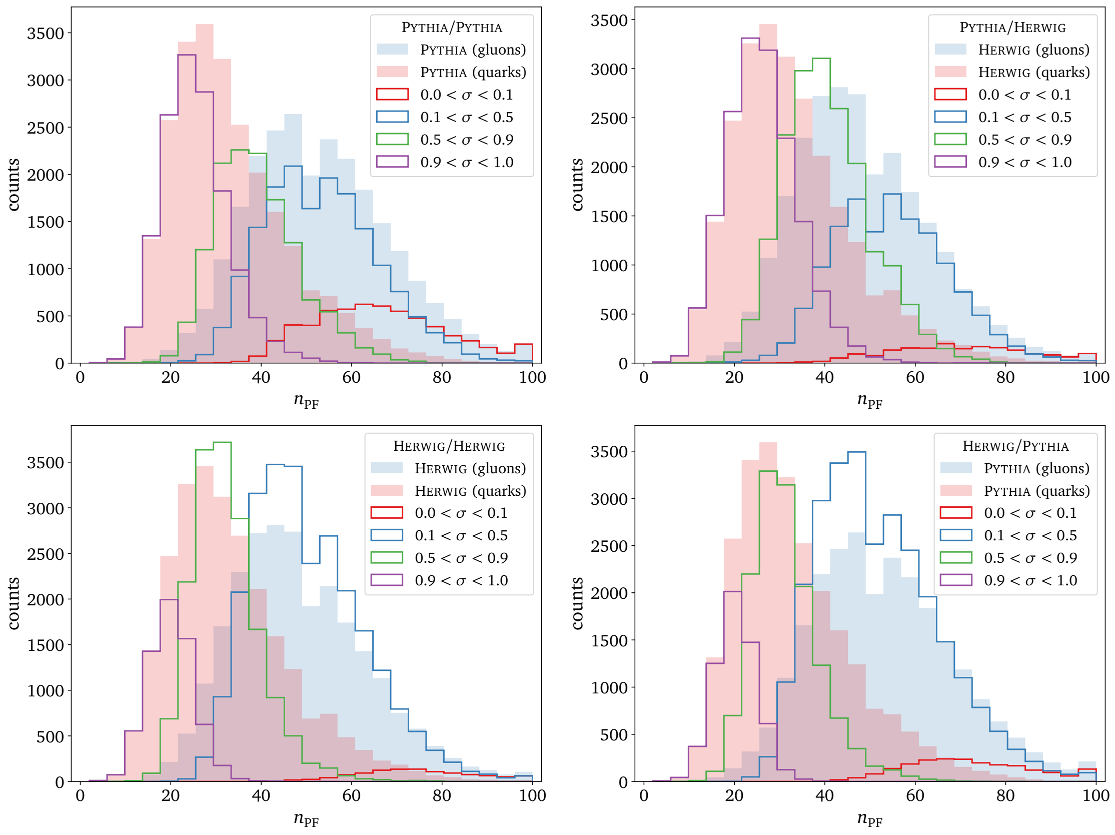


Figure 11: n_{PF} distributions in different μ bins for 50k Pythia and Herwig quarks and gluons. The filled histograms show all quarks and gluons in the testing data. The colored graphs show the distribution for μ_{pred}-bins. The μ_{pred} values are obtained with the train/test combination indicated in the figure legend. As expected, low predictive scores correspond with large n_{pf}. The bin structures are almost identical for networks trained on the same data, indicating that the network learning is strongly tied to the observable.

observation that is also made for other observables that correlate with μ_{pred}.
The predictive mean being strongly correlated to the number of particle flow objects gives a partial explaination for the poor performance of a Pythia-trained network when tested on Herwig. The n_{pf} distribution for the training data (Fig. 3) clearly shows that while the quark distributions look alike for both generators, the gluon distribution for Herwig is shifted towards smaller multiplicity values compared to Pythia. Thus a significant number of Herwig gluons, based on n_{pf}, look quark-like to a network that has learned the n_{pf} distribution of Pythia-generated jets. This leads to a high gluon-jet mistag rate ϵ_b for the Pythia/Herwig-case, which is reflected in the performance as seen in Tab. 9. Herwig trained networks learn a more dense distribution, thus Pythia jets tend to be located more towards the edges of distributions learned through Herwig. This allows a Herwig trained networks to correctly classify Pythia jets at a high rate.
The performance of tractable discriminants on experimental data of light quark and gluon jets tends to fall in between the performance achieved on Herwig and Pythia [11]. Given the correlation between tractable discriminants and classification output it can be assumed that experimental data would also perform in between Pythia and Herwig when applied to a deep neural network. Judging from the BPN results, training a DNN on data that is more or less discriminative than the data on which it is being tested does have a negative effect on the

performance. This of course is a motivation to simulate data that resembles the experimental reality as precisely as possible. However the performance difference is small compared to the difference induced by more or less discriminative test data.

5.2 Understanding the μ_{pred} distributions: Where are the HERWIG gluons?

Minimizing the cross entropy loss (Eq. (6)) will push classification scores towards the edges of the unit intervall $[0,1]$. The vacancy at $\mu_{\text{pred}} \to 0$ for HERWIG trained networks is a strong indication that for HERWIG jets there is no phase space region in which gluons can clearly be identified by the network. This hypothesis can qualitatively be strengthened by estimating the relative gluon density in the phase space region in which a PYTHIA trained network locates gluons. Estimating the density of quarks and gluons in a $\mathcal{O}(100)$-dimensional latent space is computationally difficult. The problem can be simplified by instead searching in a low-dimensional observable space, using the same tractable observables as above. Refs. [1, 2] systematically study the discriminative power in a two-dimensional observable space, also combining multiple oservables to form more sophisticated classification schemes. The approach shown here relies on pairwise comparison in a two-dimensional observable space. In the general case, a DNN learns high level correlations that cannot be understood in physical terms. Thus marking regions with gluon-like jets in terms of physical observables is only an approximation of the regions learned by the network.
In a PYTHIA/PYTHIA-scenario, the 10% test jets with the lowest classification score (most gluon-like) are located in a a plane of two observables. Their location of greatest density is (rather crudely) defined as a box in the observable plane. The edges of the box are determined by calculating the mean and standard deviation for the set of gluon like jets $(\mu_{x_i}, \sigma_{x_i})$ for each observable x_i. The edge of the box in the observable x_i is then simply given by the interval $[\mu_{x_i} - \sigma_{x_i}, \mu_{x_i} + \sigma_{x_i}]$. Marking the region via PYTHIA/PYTHIA test jets, the boundaries are then applied to both the PYTHIA and HERWIG training data. The relative gluon density is then quantified by calculating the ratio of gluons to quarks in this region. This analysis is repeated for all pairs of the observables listed in Sec. 2. The resulting quark-gluon ratios for the PYTHIA training jets, as well as the percentage of training jets in the found regions, are shown in Tab. 11.

interval	observable	w_{pf}	$p_T D$	C_{02}	x_{max}	ΔR_5
$[52, 81]$	n_{pf}	6.85 (18%)	6.66 (19%)	6.08 (21%)	6.69 (17%)	5.95 (22%)
$[0.032, 0.11]$	w_{pf}		4.99 (22%)	3.75 (34%)	4.22 (25%)	4.04 (29%)
$[0.19, 0.27]$	p_T			4.80 (25%)	4.59 (28%)	5.70 (20%)
$[0.48, 0.64]$	C_{02}				4.33 (26%)	4.34 (28%)
$[0.081, 0.175]$	x_{max}					5.29 (21%)
$[0.060, 0.138]$	ΔR_5					

Table 11: gluon/quark-ratio for PYTHIA training jets. Regions are defined via lowest 10% PYTHIA test jets.

Unsurprisingly, regions in observable space with a high particle multiplicity combined with an additional observable show the largest gluon/quark-ratios. Similar results were found in [3], where combining multiplicity and girth (n_{pf} and w_{pf}) yielded the best discrimination. Two regions with the highest g/q-ratio are visualized in Fig. 12. Taking the same box edges but transferring them to the HERWIG generated training data yields the ratios in Tab. 12. In the HERWIG data, the g/q-ratios are significantly lower across all observables, the difference is most pronounced for correlations with the multiplicity n_{pf}. The two key takeaways from this analysis are that (i) a HERWIG trained network cannot allocate gluons, even in higher

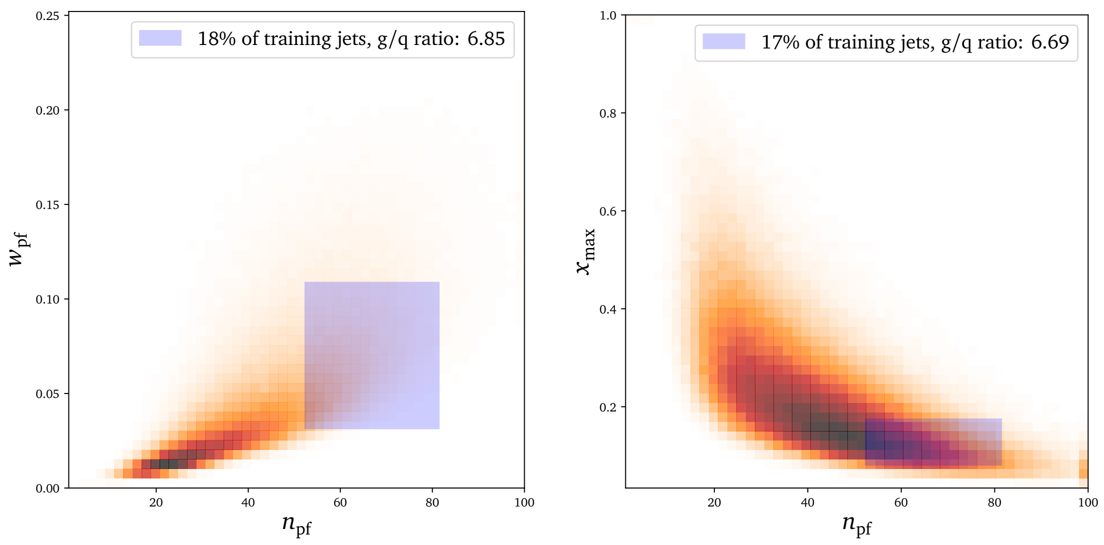


Figure 12: 400k PYTHIA training jets in two-dimensional observable planes. The highlighted box marks the area determined through low classification scores μ_{pred} and shows the quark/gluon-ratio as well as the percentage of jets in the region.

interval	observable	w_{pf}	$p_T D$	C_{02}	x_{max}	ΔR_5
$[52, 81]$	n_{pf}	5.28 (13%)	5.36 (15%)	4.94 (16%)	5.30 (14%)	4.82 (18%)
$[0.032, 0.11]$	w_{pf}		4.19 (19%)	3.56 (33%)	3.62 (23%)	3.45 (24%)
$[0.19, 0.27]$	p_T			4.07 (23%)	3.88 (26%)	4.60 (17%)
$[0.48, 0.64]$	C_{02}				3.27 (17%)	3.70 (25%)
$[0.081, 0.175]$	x_{max}					4.29 (18%)
$[0.060, 0.138]$	ΔR_5					

Table 12: gluon/quark-ratio for HERWIG training jets. Regions are defined via lowest 10% PYTHIA test jets.

dimensional observable spaces, and that (ii) a network's ability to identify gluons is closely linked to the seperability in the n_{pf} distribution, which is significantly reduced in HERWIG.
The close relation to the particle multiplicity and girth can also be approached by systematically constraining the data that the network is given to n_{pf}- and w_{pf}-related features. The BPN is retrained with similar settings as described in Sec. 4 except for one key difference: The *feature* input of the first EdgeConv block is given none of the p_T- and PID-related features in Tab. 2 and Tab. 3. Instead, the same coordinates (η, ϕ) from the raw data which are fed into the *coordinate* input are also fed into the *feature* input. The information in the combined inputs is thus constrained to the number of particles and their position in the (η, ϕ)-plane and does not include transversal momentum or the particle-ID of each constituent. The resulting ROC curves and output distributions, shown in Fig. 13, reproduce most of the features observed with a BPN trained on p_T- and PID-related features. These include the order of performance for the different train/test combinations, the gluon vacancy for HERWIG trained networks as well as the two-peak structure in the σ_{pred} distribution for PYTHIA trained networks. These shared features are another indicator for a strong correspondence between seperability of the n_{pf} distribution and the network performance.

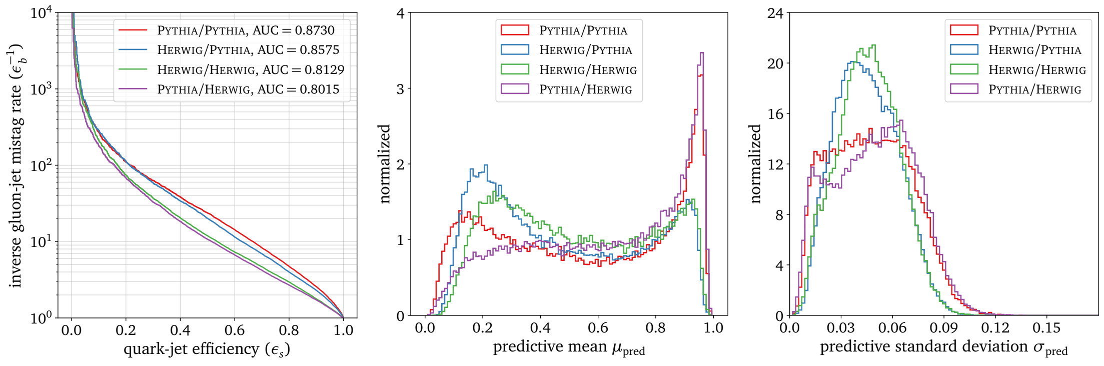


Figure 13: ROC curves, predictive means and standard deviations from BPN-Lite trained/tested on 400k/50k PYTHIA and HERWIG jets (different combinations) without p_T- or PID-related features.

5.3 Understanding the σ_{pred} distributions, peak by peak

Apart from giving insights on the quality of the network training, the parabolic curves in Fig. 10 give a first hint regarding the non-trivial peak structure observed in Fig. 9. Most pronounced is the low density in the low μ_{pred}, low σ_{pred} region for HERWIG trained networks. The parabolas seem to be tilted. This asymmetry can be visualized by showing σ_{pred} for gluons and quarks seperately (Fig. 14). The gluon distributions for HERWIG trainded networks asymptically ap-

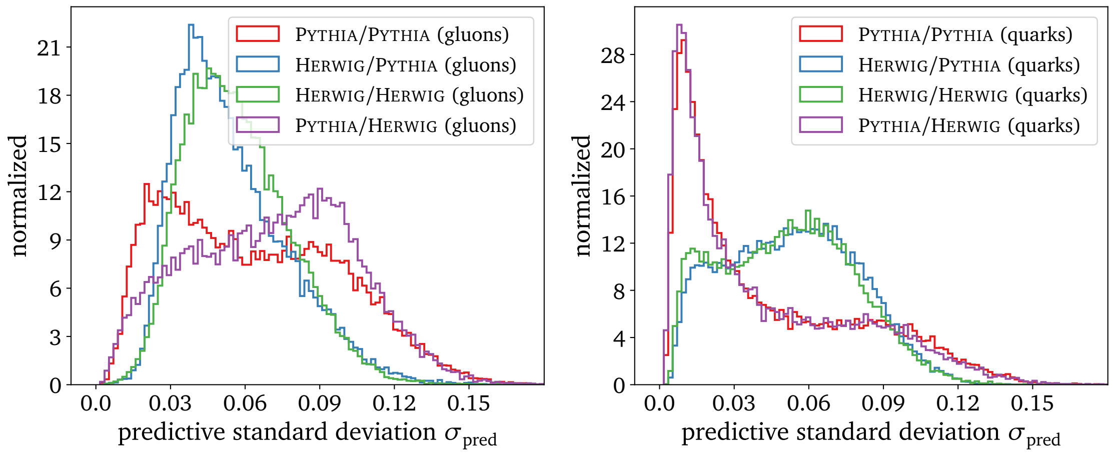


Figure 14: Predictive means and standard deviations from BPN-Lite trained/tested on 400k/50k PYTHIA and HERWIG jets (different combinations) using the default settings in Tab. 8. Seperate distributions for gluons (*left*) and quarks (*right*)

.

proach 0 at low μ_{pred}. This behavior resembles the observations for μ_{pred} in Sec. 5.3. Taylor expanding the parabolic relationship Eq. (19) around $\mu_{\text{pred}} = 0$ gives a linear relationship between classification score and uncertainty. This relationship of course is not without error, if the parabola was perfect then the σ analysis would be redundant. However the strong similaritiy between the two distributions indicates that the same irregularities in the HERWIG data that cause the vacancy at $\mu_{\text{pred}} \to 0$ cause it in the σ_{pred}-distribution.
For PYTHIA trained networks, the sharp peak at low σ_{pred} is quark dominated, although for PYTHIA/PYTHIA there is also a limited number of low σ_{pred} gluons. For the HERWIG trained networks, there is also a small peak of low σ_{pred} quarks. This shows that the quark-gluon asymmetry is not exclusively a HERWIG problem, as PYTHIA quarks are classified with high

confidence at a much larger rate then PYTHIA gluons. It seems as if this conclusion could already have been derived from the different magnitudes in the gluon and quark peaks in the μ_{pred} distribution. However repeating the training multiple times has shown that these magnitudes differ strongly for each training iteration, while the general peak structure in the σ_{pred} distribution remains stable.

Again the distributions can be understood in terms of physical observables. Analogously to

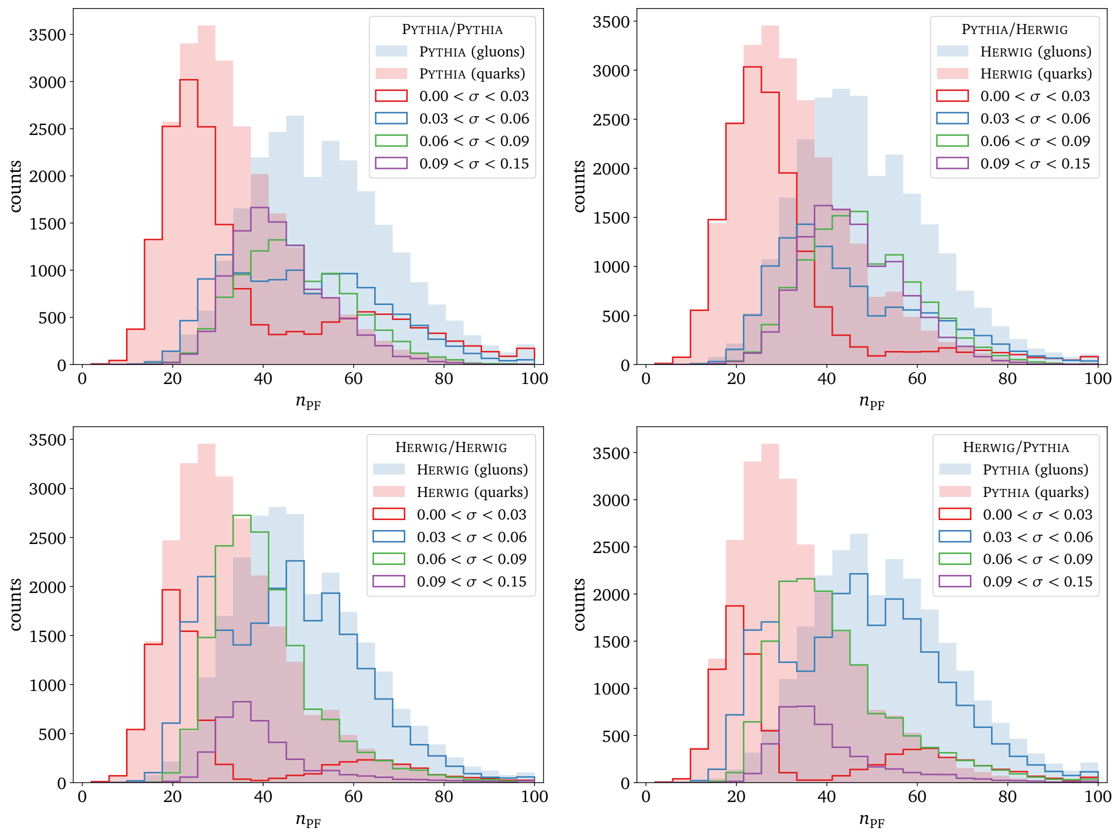


Figure 15: n_{pf} distributions in different σ bins for 50k PYTHIA and HERWIG test jets. Jets with low predicted uncertainty lie towards the edges of the distribution, whereas jets with high uncertainties are located in the overlap region between quarks and gluons.

Fig. 11, Fig. 15 shows the multiplicity distribution n_{pf} in bins of σ_{pred}. Jets with $\sigma_{\text{pred}} < 0.03$ lie almost exclusively on the edges of the n_{pf} spectrum, whereas the larger σ_{pred} bins fill the overlay region between quarks and gluons. The second peak at $\sigma_{\text{pred}} \approx 0.09$ in the PYTHIA trained networks are located at $n_{\text{pf}} \approx 40$, where quarks and gluons are hard to tell apart based on countable observables. The effect of larger σ_{pred} being distributed in overlay regions shows for all observables that are considered, but is strongest for observables which strongly correlate with μ_{pred}, namely n_{pf} and $p_T D$.

6 Conclusion and outlook

In the pursuit of understanding systematic uncertainties in HERWIG and PYTHIA jets, this work introduced a bayesian dynamic graph convolutional neural network (BDGCNN). In a first step, ParticleNet, a point-cloud based DGCNN, was reimplemented in `PyTorch` and trained to meet the benchmark set in [12]. In a second step, the network was bayesified using a modified loss function and by replacing standard weights ω_i by learnable distribution parameters (μ_i, σ_i). This network was again trained and tested to meet the benchmark set by the deterministic ParticleNet. The summarized results in Tab. 13 show that despite a slight decrease in overall performance, the BPN's statistics get close to the original deterministic implementation. The

network architecture	PID	accuracy	AUC	$\epsilon_b^{-1}(\epsilon_s = 0.5)$	$\epsilon_b^{-1}(\epsilon_s = 0.3)$
PN-Lite (TENSORFLOW/KERAS)	✓	0.835	0.9079	37.1	94.5
PN-Lite (`PyTorch`)	✓	0.831	0.9049	36.3	91.0
BPN-Lite	✓	0.820	0.9016	34.0	99.2

Table 13: Performance comparison of ParticleNet-Lite trained/tested on PYTHIA/PYTHIA. The values in the top row are taken from [12].

network outputs $(\mu_{\text{pred}}, \sigma_{\text{pred}})$ were then used to strategically analyze characteristics of the underlying datasets. In this process, three key finds were made:

- The network performance is connected to the test data, not the train data. The network predictions are strongly correlated to a set of physical observables, namely n_{pf}, which are more discriminative in PYTHIA then in HERWIG. During cross-tresting, a PYTHIA trained network mistags jets that based on their characteristics are clearly identifyable in PYTHIA but lie in shared quark-gluon regions in HERWIG. A HERWIG trained network learns a more dense distribution and can thus confidently assign PYTHIA jets as quarks and gluons as they lie closer towards the edges of HERWIG observable distributions. Training on data that is more or less discriminative than the test data does have a negative effect on the performance, but this effect is small compared to the performance drop caused by the irregularities in the test jets.
- The classification output of HERWIG trained networks is asymmetric, they mistag gluons at a much higher rate then quarks and classify them with low confidence. A qualitative analysis in two-dimensional observable planes shows that PYTHIA regions with a large gluon to quark ratio have no equivalent in HERWIG. This underlying uncertainty limits the network's ability to learn more high level features for gluon identification, leaving the area of $\mu_{\text{pred}} \to 0$ vacant. The largest g/q-ratios correlate with a large number of jet constituents, indicating that the shift of the gluon distribution towards smaller n_{pf} values in HERWIG might be the leading cause for the performance gap.
- Apart from a peak at $\sigma_{\text{pred}} \to 0$, which is quark dominated, PYTHIA trained networks assign a high uncertainty to a large share of quark and gluon jets.
  The difficulites at correctly identifying gluons also applies to PYTHIA, but with less severity. The additional peaks for PYTHIA trained networks correspond to overlap regions in the observable distributions, whereas low uncertainties lie at the edges of observable distributions. I.e. the peak at $\sigma_{\text{pred}} \approx 0.09$ for PYTHIA trained networks roughly corresponds to both quark and gluon jets which are located at $n_{\text{pf}} \approx 40$.

If now ParticleNet was chosen to represent the mutlitude of DNN classifiers in quark-gluon tagging, could their tagging be considered resilient with respect to the MC generator used? Based on this study, the performance will drop if the network is trained on data that is either

more or less discriminative then the test data. However this performance drop is still dominated by irregularites in the test data. In particular, a DNNs ability to correctly identify gluons is highly sensitive to small changes in the gluon phase space in the testing data. The BPN confirms that this sensitivity can be partially understood in a physical way by means of intermediate observables. The strongest evidence is found with respect to the particle multiplicity n_{pf}, where the HERWIG generated distribution is shifted towards smaller values compared to PYTHIA. The origin of the n_{pf} distribution's discriminative prowess, the fraction of color factors C_A/C_F, is also the leading cause for a number of other observable distributions. In addition, n_{pf} by definition correlates strongly with other observables. Therefore there is a good chance that the multiplicity distribution is just an indicator pointing towards the problematic regions in a high dimensional phase space that are difficult to access through tractable observables.
If a small phase space shift prevents a cutting-edge network from finding gluons, the question arises whether there are methods in the realm of machine learning that improve on existing methods by specifically targeting the uncertainties in the gluon space. There are a couple of promising methods of which two are being presented here:

- **Reweighting:** Reweighting [48,49] is a method that attempts to morph the phase space of gluons and quarks generated by PYTHIA into the phase space of HERWIG or vice versa. A classifier is trained to distinguish HERWIG generated jets from PYTHIA generated jets. Via the classification score $f(x_i)$ the factor that morphs the two probability densities $p_{\text{PYTHIA}}(x_i)$ and $p_{\text{HERWIG}}(x_i)$ can be estimated:

$$w(x_i) = \frac{p_{\text{PY}}(x_i)}{p_{\text{HE}}(x_i)} \approx \frac{f(x_i)}{1-f(x_i)}. \tag{24}$$

  The learned weights can then be applied to the datasets before another classifier tries to distinguish quark from gluon jets.
- **Contrastive learning:** Unsupervised contrastive learning algorithms learn to map low-level data to optimized observables and have been succesfully implemented for Top-QCD tagging [50]. Contrastive learning could potentially learn observables with greater discriminative power, even for HERWIG generated jets.

Both of these methods can be tested in the future for their ability to close the gap between HERWIG and PYTHIA.

7 Appendix

7.1 Observable distributions in μ_{pred} and σ_{pred} bins

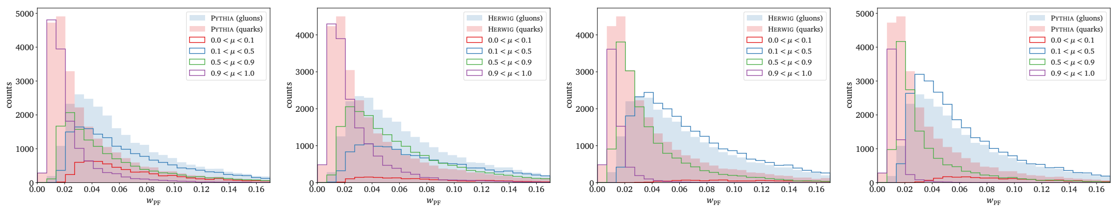


Figure 16: w_{pf} distributions in different μ bins for 50k Pythia and Herwig quarks and gluons

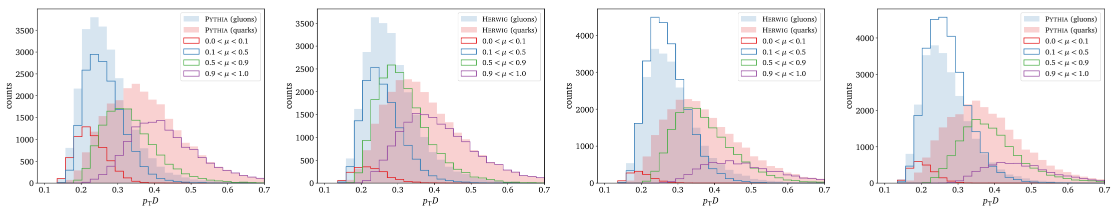


Figure 17: $p_{\text{T}}D$ distributions in different μ bins for 50k Pythia and Herwig quarks and gluons

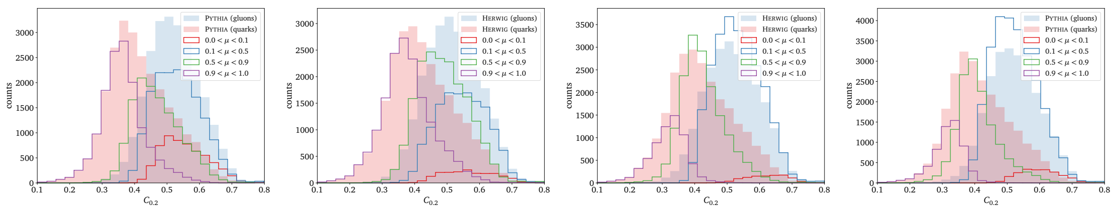


Figure 18: $C_{0.2}$ distributions in different μ bins for 50k Pythia and Herwig quarks and gluons

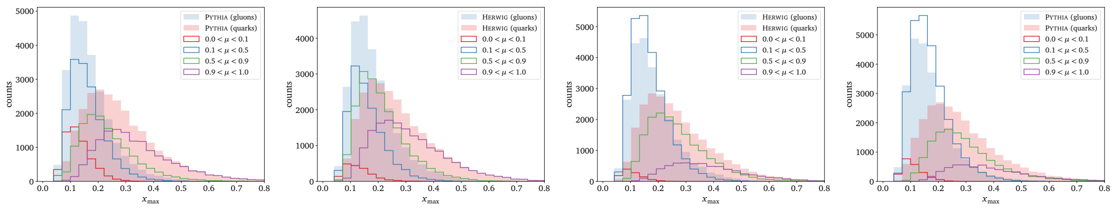


Figure 19: x_{max} distributions in different μ bins for 50k Pythia and Herwig quarks and gluons

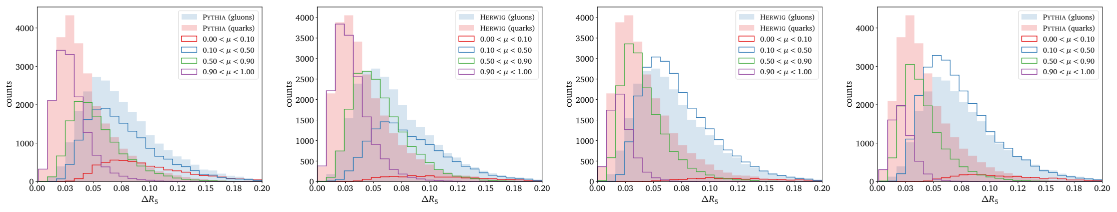


Figure 20: ΔR_5 distributions in different μ bins for 50k Pythia and Herwig quarks and gluons

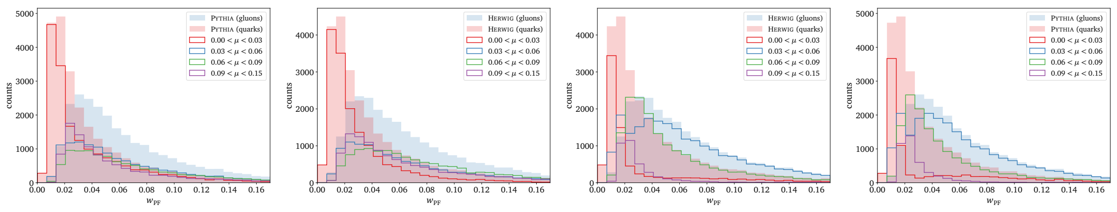


Figure 21: w_{pf} distributions in different σ bins for 50k Pythia and Herwig quarks and gluons

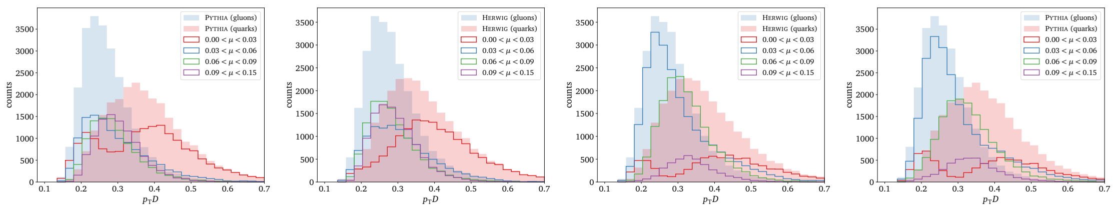


Figure 22: $p_{\text{T}}D$ distributions in different σ bins for 50k Pythia and Herwig quarks and gluons

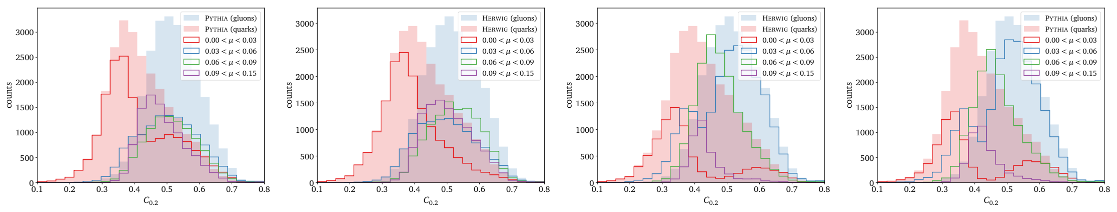


Figure 23: $C_{0.2}$ distributions in different σ bins for 50k Pythia and Herwig quarks and gluons

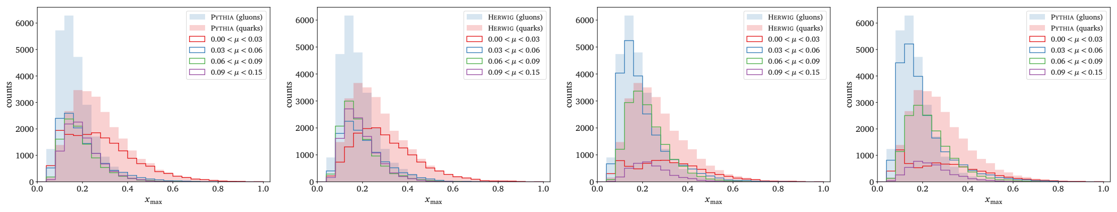


Figure 24: x_{max} distributions in different σ bins for 50k Pythia and Herwig quarks and gluons

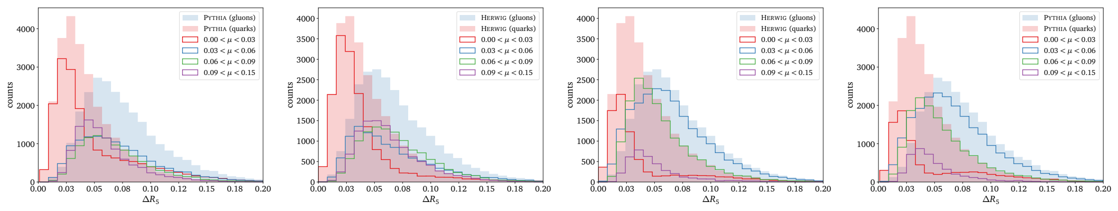


Figure 25: ΔR_5 distributions in different σ bins for 50k PYTHIA and HERWIG quarks and gluons

8 Acknowledgement

Large parts of this thesis are based on the in-process documentation *Resilience of Quark-Gluon Tagging: Where are the HERWIG gluons?* [51] which is co-authored by Anja Butter, Barry M. Dillon, myself, Tilman Plehn and Lorenz Vogel.

Some words of gratitude: I thank Luigi Favaro and Theo Heimel for providing countless helpful insights into the peculiarities of scientific computing and for their undying patience in doing so. I thank Huilin Qu for his continuous efforts to make cutting-edge machine learning algorithms accessible for particle physicists like myself. I thank Tilman Plehn for supervising this project, for asking the right questions, for connecting me with the right people and for moving us forward as a team. A special thanks goes out to Barry Dillon for not only helping me to find my project and for the many to-do lists, but also for helping my understanding of high energy physics grow throughout the last couple of months. A second special thanks goes out to Lorenz Vogel for co-authoring a majority of the code used in this project, for the know-how in bayesifying ParticleNet and for providing the modified SciPost-Layout in which this thesis is written. A final thanks also to Prof. Jan Pawlowski for dedicating time and effort by being the second examiner of this thesis.

References

[1] A. J. Larkoski, J. Thaler and W. J. Waalewijn, *Gaining (Mutual) Information about Quark/Gluon Discrimination*, JHEP **11**, 129 (2014), doi:10.1007/jhep11(2014)129, arXiv:1408.3122.

[2] P. Gras, S. Höche, D. Kar, A. Larkoski, L. Lönnblad, S. Plätzer, A. Siódmok, P. Skands, G. Soyez and J. Thaler, *Systematics of quark/gluon tagging*, JHEP **07**, 091 (2017), doi:10.1007/JHEP07(2017)091, arXiv:1704.03878.

[3] J. Gallicchio and M. D. Schwartz, *Quark and Gluon Tagging at the LHC*, Phys. Rev. Lett. **107**, 172001 (2011), doi:10.1103/PhysRevLett.107.172001, arXiv:1106.3076.

[4] J. Gallicchio and M. D. Schwartz, *Quark and Gluon Jet Substructure*, JHEP **04**, 090 (2013), doi:10.1007/JHEP04(2013)090, arXiv:1211.7038.

[5] C. Frye, A. J. Larkoski, J. Thaler and K. Zhou, *Casimir meets poisson: Improved quark/gluon discrimination with counting observables*, JHEP **09**, 083 (2017), doi:10.1007/JHEP09(2017)083, arXiv:1704.06266.

[6] P. T. Komiske, E. M. Metodiev and M. D. Schwartz, *Deep learning in color: towards automated quark/gluon jet discrimination*, JHEP **01**, 110 (2017), doi:10.1007/JHEP01(2017)110, arXiv:1612.01551.

[7] *Quark versus Gluon Jet Tagging Using Jet Images with the ATLAS Detector*, Tech. rep., CERN, Geneva, All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-017 (2017).

[8] J. Bellm, S. Gieseke, D. Grellscheid, P. Kirchgaeßer, F. Loshaj, G. Nail, A. Papaefstathiou, S. Plätzer, R. Podskubka, M. Rauch, C. Reuschle, P. Richardson *et al.*, *Herwig 7.1 Release Note* (2017), arXiv:1705.06919.

[9] T. Sjöstrand, S. Mrenna and P. Z. Skands, *PYTHIA 6.4 Physics and Manual*, JHEP **05**, 026 (2006), doi:10.1088/1126-6708/2006/05/026, arXiv:hep-ph/0603175.

[10] S. Badger, J. Bendavid, V. Ciulli, A. Denner, R. Frederix, M. Grazzini, J. Huston, M. Schönherr, K. Tackmann, J. Thaler, C. Williams, J. R. Andersen *et al.*, *Les houches 2015: Physics at tev colliders standard model working group report*, doi:10.48550/ARXIV.1605.04692 (2016).

[11] G. Aad, , B. Abbott, J. Abdallah, S. A. Khalek, O. Abdinov, R. Aben, B. Abi, M. Abolins, O. S. AbouZeid, H. Abramowicz, H. Abreu *et al.*, *Light-quark and gluon jet discrimination in $$pp$$ p p collisions at $$\sqrt{s}=7\mathrm {\ TeV}$$ s = 7 TeV with the ATLAS detector*, The European Physical Journal C **74** (2014), doi:10.1140/epjc/s10052-014-3023-z, https://doi.org/10.1140%2Fepjc%2Fs10052-014-3023-z.

[12] H. Qu and L. Gouskos, *ParticleNet: Jet Tagging via Particle Clouds*, Phys. Rev. D **101**, 056019 (2020), doi:10.1103/PhysRevD.101.056019, arXiv:1902.08570.

[13] P. T. Komiske, E. M. Metodiev and J. Thaler, *Energy Flow Networks: Deep Sets for Particle Jets*, JHEP **01**, 121 (2019), doi:10.1007/JHEP01(2019)121, arXiv:1810.05165.

[14] P. Komiske, E. Metodiev and J. Thaler, *Pythia8 Quark and Gluon Jets for Energy Flow*, Zenodo (2019), doi:10.5281/zenodo.3164691.

[15] A. Pathak, P. Komiske, E. Metodiev and M. Schwartz, *Herwig7.1 Quark and Gluon Jets*, Zenodo (2019), doi:10.5281/zenodo.3066475.

[16] M. Cacciari, G. P. Salam and G. Soyez, *FastJet user manual*, Eur. Phys. J. C **72**, 1896 (2012), doi:10.1140/epjc/s10052-012-1896-2, arXiv:1111.6097.

[17] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, JHEP **04**, 063 (2008), doi:10.1088/1126-6708/2008/04/063, arXiv:0802.1189.

[18] Particle Data Group, *Review of Particle Physics*, Prog. Theor. Exp. Phys. **2020**, 083C01 (2020), doi:10.1093/ptep/ptaa104.

[19] A. J. Larkoski, G. P. Salam and J. Thaler, *Energy Correlation Functions for Jet Substructure*, JHEP **06**, 108 (2013), doi:10.1007/JHEP06(2013)108, arXiv:1305.0007.

[20] CMS Collaboration, *Performance of quark/gluon discrimination in 8 TeV pp data*, CMS Physics Analysis Summary (2013), http://cds.cern.ch/record/1599732.

[21] H. Kucuk, *Measurement of the inclusive-jet cross-section in proton-proton collisions and study of Quark-Gluon Jet discrimination with the ATLAS experiment at the LHC*, Ph.D. thesis, University College London (2016).

[22] H. Qu, C. Li and S. Qian, *Particle transformer for jet tagging*, doi:10.48550/ARXIV.2202.03772 (2022).

[23] S. Macaluso and D. Shih, *Pulling Out All the Tops with Computer Vision and Deep Learning*, JHEP **10**, 121 (2018), doi:10.1007/JHEP10(2018)121, arXiv:1803.00107.

[24] A. Krizhevsky, I. Sutskever and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, In F. Pereira, C. Burges, L. Bottou and K. Weinberger, eds., *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc. (2012).

[25] G. Kasieczka, T. Plehn, M. Russell and T. Schell, *Deep-learning Top Taggers or The End of QCD?*, JHEP **05**, 006 (2017), doi:10.1007/JHEP05(2017)006, arXiv:1701.08784.

[26] S. Choi, S. J. Lee and M. Perelstein, *Infrared safety of a neural-net top tagging algorithm*, Journal of High Energy Physics **2019** (2019), doi:10.1007/jhep02(2019)132, https://doi.org/10.1007%2Fjhep02%282019%29132.

[27] D. Guest, J. Collado, P. Baldi, S.-C. Hsu, G. Urban and D. Whiteson, *Jet Flavor Classification in High-Energy Physics with Deep Neural Networks*, Phys. Rev. **D94**, 112002 (2016), doi:10.1103/PhysRevD.94.112002, arXiv:1607.08633.

[28] K. Fraser and M. D. Schwartz, *Jet charge and machine learning*, Journal of High Energy Physics **2018** (2018), doi:10.1007/jhep10(2018)093, https://doi.org/10.1007%2Fjhep10%282018%29093.

[29] *Identification of Jets Containing b-Hadrons with Recurrent Neural Networks at the ATLAS Experiment*, Tech. rep., CERN, Geneva, All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-003 (2017).

[30] G. Louppe, K. Cho, C. Becot and K. Cranmer, *QCD-aware recursive neural networks for jet physics*, Journal of High Energy Physics **2019** (2019), doi:10.1007/jhep01(2019)057, https://doi.org/10.1007%2Fjhep01%282019%29057.

[31] T. Cheng, *Recursive neural networks in quark/gluon tagging*, Computing and Software for Big Science **2** (2018), doi:10.1007/s41781-018-0007-y, https://doi.org/10.1007%2Fs41781-018-0007-y.

[32] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, In D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, eds., *Computer Vision – ECCV 2014*, pp. 818–833. Springer International Publishing, Cham, ISBN 978-3-319-10590-1 (2014).

[33] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein and J. M. Solomon, *Dynamic graph cnn for learning on point clouds* **38** (2019), doi:10.1145/3326362, https://doi.org/10.1145/3326362.

[34] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, In F. Bach and D. Blei, eds., *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *Proceedings of Machine Learning Research*, pp. 448–456. PMLR, Lille, France (2015).

[35] X. Glorot, A. Bordes and Y. Bengio, *Deep sparse rectifier neural networks*, In G. Gordon, D. Dunson and M. Dudík, eds., *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15 of *Proceedings of Machine Learning Research*, pp. 315–323. PMLR, Fort Lauderdale, FL, USA (2011).

[36] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, doi:10.1109/CVPR.2016.90 (2016).

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15**, 1929 (2014), http://jmlr.org/papers/v15/srivastava14a.html.

[38] I. Loshchilov and F. Hutter, *Decoupled Weight Decay Regularization* (2017), arXiv:1711.05101.

[39] S. L. Smith, P.-J. Kindermans, C. Ying and Q. V. Le, *Don't decay the learning rate, increase the batch size*, doi:10.48550/ARXIV.1711.00489 (2017).

[40] E. Hoffer, I. Hubara and D. Soudry, *Train longer, generalize better: closing the generalization gap in large batch training of neural networks* (2017), doi:10.48550/ARXIV.1705.08741, https://arxiv.org/abs/1705.08741.

[41] *Boosted jet identification using particle candidates and deep neural networks* (2017), https://cds.cern.ch/record/2295725.

[42] S. Bollweg, M. Haussmann, G. Kasieczka, M. Luchmann, T. Plehn and J. Thompson, *Deep-learning jets with uncertainties and more*, SciPost Phys. **8**, 006 (2020), doi:10.21468/SciPostPhys.8.1.006, https://scipost.org/10.21468/SciPostPhys.8.1.006.

[43] A. Kendall and Y. Gal, *What uncertainties do we need in bayesian deep learning for computer vision?*, doi:10.48550/ARXIV.1703.04977 (2017).

[44] Y. Xu, W. Xu, Y. Meng, K. Zhu and W. Xu, *Applying bayesian neural networks to event reconstruction in reactor neutrino experiments*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **592**, 451 (2008), doi:10.1016/j.nima.2008.04.006, https://doi.org/10.1016%2Fj.nima.2008.04.006.

[45] M. Luchmann, *Uncertainties with Bayesian neural networks in particle physics*, Master thesis, Ruprecht-Karls-Universität Heidelberg (2019), https://www.thphys.uni-heidelberg.de/~plehn/includes/theses/luchmann_m.pdf.

[46] D. M. Blei, A. Kucukelbir and J. D. McAuliffe, *Variational inference: A review for statisticians*, Journal of the American Statistical Association **112**, 859 (2017), doi:10.1080/01621459.2017.1285773, https://doi.org/10.1080%2F01621459.2017.1285773.

[47] K. Shridhar, F. Laumann and M. Liwicki, *A comprehensive guide to bayesian convolutional neural network with variational inference*, arXiv preprint arXiv:1901.02731 (2019).

[48] B. Nachman and J. Thaler, *Neural resampler for Monte Carlo reweighting with preserved uncertainties*, Phys. Rev. D **102**, 076004 (2020), doi:10.1103/PhysRevD.102.076004, arXiv:2007.11586.

[49] A. Andreassen and B. Nachman, *Neural Networks for Full Phase-space Reweighting and Parameter Tuning*, Phys. Rev. D **101**, 091901 (2020), doi:10.1103/PhysRevD.101.091901, arXiv:1907.08209.

[50] B. M. Dillon, G. Kasieczka, H. Olischlager, T. Plehn, P. Sorrenson and L. Vogel, *Symmetries, Safety, and Self-Supervision* (2021), arXiv:2108.04253.

[51] A. Butter, B. M. Dillon, L. F. Klassen, T. Plehn and L. Vogel, *Resilience of quark-gluon tagging: Where are the herwig gluons?* (2022).

Erklärung

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

Heidelberg, den 26.09.2022, [illegible]