technische universität
dortmund

Arbeit zur Erlangung des akademischen Grades
Master of Science

Study of convolutional neural networks for strange-tagging based on jet images from calorimeters

Nils Julius Abicht
geboren in Hamburg

2020

Lehrstuhl für Experimentelle Physik IV
Fakultät Physik
Technische Universität Dortmund

Erstgutachter:	Priv.-Doz. Dr. Johannes Erdmann
Zweitgutachter:	Prof. Dr. Johannes Albrecht
Abgabedatum:	11. September 2020

Abstract

A method for the identification of jets initiated by prompt strange quarks would complement the set of already existing jet-tagging algorithms. In this thesis, an algorithm for the discrimination between strange jets and down jets is presented. It uses convolutional neural networks to classify jet images built from energy depositions in calorimeters. In an additional step, the algorithm is combined with previous work, where track properties were used as input for long short-term memory recurrent neural networks. Further studies indicate that the jet-image based classification mostly relies on the in the hadronic calorimeter deposited energy fraction, which is for strange jets on average higher than for down jets. Discrimination power based on the track properties is mostly tied to tracks that originate from secondary vertices, which occur more often in strange jets. While the jet-image based algorithm achieves a performance similar to a simple benchmark algorithm, the combined algorithm appears to learn additional discrimination power from correlations between the jet images and the tracking information. For signal efficiencies of 30% and 70%, background efficiencies of respectively 18% and 53% are achieved, compared to respectively 22% and 57% achieved by the benchmark algorithm.

Kurzfassung

Eine Methode zur Identifizierung von Jets, die aus prompten Strange-Quarks entstehen, würde bereits existierende Jet-Tagging-Algorithmen vervollständigen. In dieser Arbeit wird ein Algorithmus zur Unterscheidung von Strange- und Down-Jets präsentiert. Mit Convolutional Neural Networks werden Jet-Images klassifiziert, die aus Energiedepositionen in Kalorimetern erstellt werden. Zusätzlich wird der Algorithmus mit einer vorherigen Arbeit erweitert, in der Spureigenschaften als Input für Long Short-Term Memory Recurrent Neural Networks benutzt werden. Weitere Studien lassen vermuten, dass die Unterscheidung der Jet-Images hauptsächlich auf dem für Strange-Jets verglichen mit Down-Jets im Durchschnitt höheren Anteil von in dem hadronischen Kalorimeter deponierter Energie basiert. Die Unterscheidungskraft basiert außerdem stark auf den Eigenschaften der Spuren, die von in Strange-Jets häufigeren sekundären Vertices stammen. Auch wenn der auf den Jet-Images basierende Algorithmus ähnliche Ergebnisse wie ein einfacher Benchmark-Algorithmus erziehlt, scheint der kombinierte Algorithmus zusätliche Unterscheidungskraft aus Korrelationen zwischen den Jet-Images und den Spureigenschaften zu lernen. Für Signaleffizienzen von 30% und 70% werden Untergrundeffizienzen von 18% beziehungsweise 53% erziehlt, verglichen zu Untergrundeffizienzen von 22% beziehungsweise 57%, die mit dem Benchmark-Algorithmus erziehlt werden.

Contents

1 Introduction

High-energy particle collisions at colliders such as the Large Hadron Collider (LHC) [1] are used to test and study the Standard Model of particle physics (SM). Around the interaction points, multi-purpose detectors such as the detectors at the ATLAS [2] or the CMS experiment [3] are built. Since quarks and gluons produced in the collisions cannot exist in free form, they radiate other quarks and gluons with which they form bound states (hadrons). The initial particle and particles produced in this hadronization typically have similar four-vectors. The collection of these particles together with the energy depositions they leave in the calorimeters of a detector are called a jet. Achieving the best possible determination of the particle that initiated the hadronization and thereby the jet (jet-tagging) is a fundamental task for many analyses that measure properties of, or test the SM.

With different initiating particles of a jet giving rise to different jet properties, several jet-tagging algorithms are already in use. Top quarks produce large-radius jets with a distinctive substructure. Jets from bottom, and to some extend charm quarks can be identified by their tendency to have displaced track vertices and jets from gluons have characteristic features which can be exploited. An accurate algorithm for the identification of jets from strange quarks (s-tagging) would complete the set of jet-tagging algorithms, possibly opening up access to applications, such as the direct measurement of the $t \to W^+ s$ [4] and $H \to s\bar{s}$ decays [5] as well as enabling a multitude of searches for physics beyond the SM.

However, the jets produced by strange quarks (s-jets) are difficult to distinguish from jets that arise from the light first-generation quarks (u- and d-jets). A possible s-tagging algorithm could rely on the momentum-weighted fraction of hadrons containing strange quarks, which is larger for s-jets in comparison to u- and d-jets. Some of these hadrons have rather long lifetimes, thus they usually deposit most of their energy in the hadronic calorimeter of a detector. Neutral pions on the other side, which are more frequent in u- and d-jets, decay almost instantly, mostly into photon pairs, depositing their energy in the electromagnetic calorimeter. Therefore, s-jets tend to deposit more of their energy in the hadronic calorimeter.

It is the aim of this thesis to find differences between the energy depositions that s-jets and d-jets leave in the calorimeters. For this purpose, convolutional

neural networks (CNNs) are used for the classification of jet images built from these energy depositions. Training and testing of these CNNs is performed on Monte Carlo (MC) simulations with a detector layout inspired by the CMS detector.

As a second step, the results are combined with previous work on the use of tracking information for s-tagging [6], focused on the reconstruction of secondary vertices, which occur more often in s-jets. Concatenations of the long short-term memory neural networks (LSTMs), that are used to analyze the tracking information, and the jet-image CNNs are studied.

This thesis is structured as follows: The next section gives a short overview of the SM and properties of s-jets in a multi-purpose detector. Section 3 covers the event generation and detector simulation. Section 4 describes how the jet images are built from the MC samples. Section 5 gives an overview of how neural networks work and in Section 6 methods for discriminating between s- and d-jets are studied and their performance is evaluated. Conclusions of this thesis are given in the final section.

2 Strange jets in the Standard Model and their behavior in multi-purpose detectors

2.1 Overview of the Standard Model of particle physics

The Standard Model of particle physics (SM) is a quantum field theory that describes all known elementary particles and the strong, the weak as well as the electromagnetic interactions between them. Only the gravitational force is not included. For many decades now, it has been the most accurate description of the experimental data collected at high energy particle experiments such as the LHC.

In the SM, the elementary particles with half-integer spin are called fermions. Six quark flavors and six lepton flavors of the elementary fermions exist, which can be subdivided into respectively three generations of up-type quarks with electric charge $q = +2/3\, e$ (up, charm and top quarks), down-type quarks with $q = -1/3\, e$ (down, strange and bottom quarks), charged leptons with $q = -1\, e$ (electrons, muons and tau leptons), uncharged leptons (electron, muon and tau neutrinos), as well as a corresponding antiparticle for each of them with the same mass but opposite charge quantum numbers. Except for the massless neutrinos, the generations are ordered based on mass with the heaviest particle in the third generations.

In addition to that, there are elementary particles with integer spin, also called bosons. The Higgs boson has spin $s = 0$ and can be interpreted as an excitation of the Higgs field, a field that can break the symmetry in interactions and thereby explains why particles have mass. The other elementary bosons have spin $s = 1$ and are mediators of the fundamental interactions in the SM. The massless photon (γ) mediates the electromagnetic interaction, which only couples to electrically charged particles. Massless gluons (g) mediate the strong interaction, which couples to quarks as well as other gluons. Unlike the other interactions, the strong interaction does not continuously decrease in strength with larger distances. Therefore, the potential energy between two quarks or gluons gets larger, the further they are apart, up to a point where it is energetically more favorable, that additional quarks or gluons are produced from the vacuum. Finally, the W and the Z bosons, which carry mass, mediate the weak interaction. They couple to all SM fermions. The

coupling strength of the interactions, and therefore the interaction rates are not constant, but depend on the energy scale. An overview of the SM particles is given in Figure 2.1.

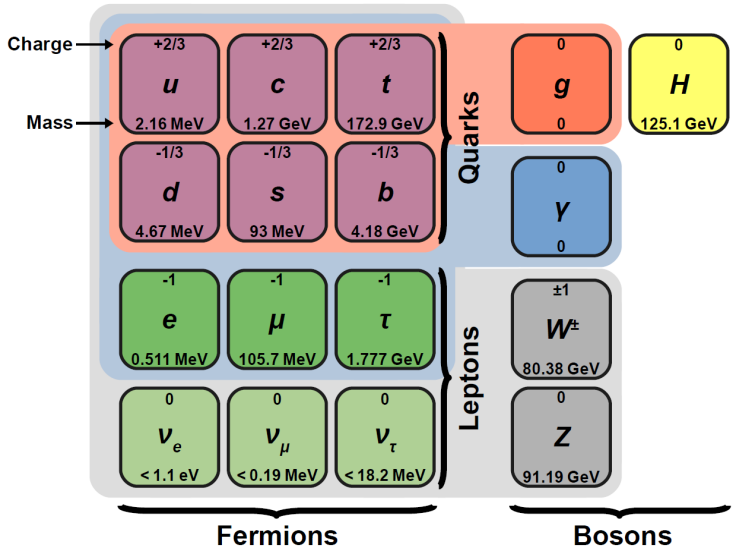


Figure 2.1: Schematic overview of the elementary particles in the SM, their masses, their electric charge and their interaction with the fundamental SM forces.

2.2 Properties of strange and down jets in a multi-purpose detector

In high-energetic collisions of for example protons at the interaction points of particle experiments such as the LHC, elementary particles, such as strange or down quarks can be produced. As described in Section 2.1, quarks can never exist alone. Hence, additional gluons and quarks are produced until the energy of the particles falls below a certain threshold. The quarks form bound states, which are composite particles that consist of either a quark and an antiquark (mesons) or three quarks or three antiquarks (baryons). This process is called hadronization. The particles that stem from the initial particle usually have similar four-vectors and form a particle jet.

Large detectors are used for recording the particles that are produced in the

hadronization process. Multi-purpose detectors, such as the CMS or the ATLAS detector are suitable for a variety of measurements. Typical components for such detectors include an inner tracking detector (ID), an electromagnetic calorimeter (ECAL) and a hadronic calorimeter (HCAL), which are arranged cylindrically around the beam axis. The azimuthal angle ϕ and the pseudorapidity η are used as spatial coordinates. The latter is derived from the angle to the beam axis θ, according to $\eta = -\ln(\tan\theta)$. In contrast to θ, distances in η are invariant under Lorentz-transformations along the beam axis. Distances in the η-ϕ plane can be defined as $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$, where $\Delta\eta$ and $\Delta\phi$ are distances in η and ϕ, respectively.

In the ID, which usually has a radius of about one meter order of magnitude, tracks of charged particles are reconstructed, usually with high resolution. From the tracks, the position of the interaction point, also called the primary vertex (PV) can be reconstructed. The high resolution also opens up the possibility of secondary vertex (SV) measurements. While most tracks are associated to the PV, a SV occurs, when a particle travels a small distance from the PV and than decays while still inside the ID. Typically, the ID is within a magnetic field, which also enables momentum measurements for charged particles through the curvature of their tracks. In the calorimeters, electrons, positrons, photons and hadrons interact with the detector material and usually deposit their energy in the calorimeter cells until they are stopped. Electrons, positrons and photons usually deposit most of their energy in the ECAL, while hadrons usually deposit most of their energy in the HCAL. Algorithms can be used, to find clusters of energy depositions in the calorimeters, which are then labeled as jets. Depending on which particle initiated the jet, properties of the calorimeter signature of the jet as well as associated tracks may differ.

Methods for the discrimination between s-jets and jets that are initiated by top, bottom or charm quarks as well as by gluons are already in place. This thesis focuses on the mostly unexplored problem of discriminating s-jets from d-jets. The methods used in this thesis are expected to be equally applicable to the discrimination between s- and u-jets, since u-jets mostly have the same properties as d-jets. The hadronization and showering of both strange and down quarks mainly produces the lightest possible hadrons, the charged and neutral pions, $\pi^{\pm}$ and π^0. The only differences between s- and d-jets arise from the preservation of quark flavor during hadronization. A hard scattering strange quark is far more likely to hadronize into a hadron containing a strange quark (strange hadron), most likely a kaon. Kaons are mesons that consist of a strange quark and an up or down antiquark, or their respective antiparticles. Therefore, they are either electrically neutral (K_L^0, K_S^0) or have an electrical charge of $\pm 1\ e$ ($K^{\pm}$). The higher the energy fraction that the strange hadrons carry in a jet, the more that can lead to a different detector

signature. Since the strange hadrons that stem directly from a prompt strange quark usually carry a higher energy fraction than possible strange hadrons in d-jets, this can be used to separate s- from d-jets.

One difference becomes apparent in the calorimeters. The charged pions are detector stable, which means that their average decay length is much larger than usual detector dimensions. They therefore usually reach the HCAL, where they deposit most of their energy. However, on average, about one third of the pions in a jet are neutral pions. A neutral pion almost instantly decays into two photons, which deposit most of their energy in the ECAL. The four types of kaons on the other hand, which are characteristic for s-jets, mostly deposit their energy in the HCAL. Three of the four kaon types, the K^+, the K^- and the K_L^0, all having about the same relevance for s-jets, are detector stable. Only the K_S^0 as well as most of the in comparison more exotic strange baryons have an average decay length of a few centimeters, therefore, mostly decay inside the ID. Some of the decay products can still deposit a significant amount of energy in the HCAL, though. Looking at the K_S^0, it decays with a branching ratio of 31% into two neutral pions, in turn decaying into four photons. 69% of the K_S^0 still decay into two detector stable charged pions. So, in total, compared to d-jets, s-jets tend to deposit a larger fraction of their energy in the HCAL as opposed to the ECAL.

Another possible handle is the higher occurrence of SVs in s-jets. These are due to decays of the already mentioned K_S^0 and strange baryons such as the Λ baryon inside the ID. Most particles produced in the fragmentation of down quarks are either detector stable or instantly decay at the PV. Therefore, the properties of tracks that can be used to reconstruct a SV are of particular interest for discriminating between s- and d-jets.

3 Monte Carlo Generation

The discrimination between s-jets and d-jets is investigated on Monte Carlo samples for the processes $pp \to s\bar{s}$ and $pp \to d\bar{d}$. MadGraph5_aMC@NLO [7] at leading order in α_S with the NNPDF2.3LO PDF set [8] at $\sqrt{s} = 13\,\text{TeV}$ is used to simulate the hard scattering process of 1950000 events for each of the two processes. The subsequent parton showering and hadronisation are simulated with Pythia8 [9]. To simulate the detector response, Delphes version 3.4.1 [10] is used with a modified version of the CMS card. Jets are clustered from calorimeter towers with the anti-k_t algorithm [11] and a distance parameter of $R = 0.5$.

The Delphes-CMS simulation provides a sufficient but simplistic example of a general-purpose detector. Of special importance for the creation of jet images is of course the calorimeter simulation. There, a few simplifications are made by Delphes, none of which should have a major impact, though.

- Photons, electrons and positrons deposit all of their energy in the ECAL, most hadrons deposit all their energy in the HCAL.
- Particles are exclusively decayed with Pythia, but only particles with average decay lengths $c\tau < 1000\,\text{mm}$ can decay; Delphes does not simulate in-flight particle decays. Therefore, particles, such as the charged pion ($c\tau \approx 7.8\,\text{m}$), the charged kaon ($c\tau \approx 3.7\,\text{m}$) or the K_L^0 ($c\tau \approx 15.3\,\text{m}$) always reach the calorimeters. The K_S^0 and the Λ baryons that are not decayed with Pythia reach the calorimeters as well. In contrast to all other hadrons, they deposit 30% of their energy in the HCAL and 70% in the HCAL, which is a simplistic representation of the energy depositions they would leave in the calorimeters if a proper decay was simulated.
- One ECAL layer and one HCAL layer are simulated. Each particle, that reaches the calorimeters hits exactly one cell in each layer. Muons and neutrinos are assumed not to interact with the calorimeters.

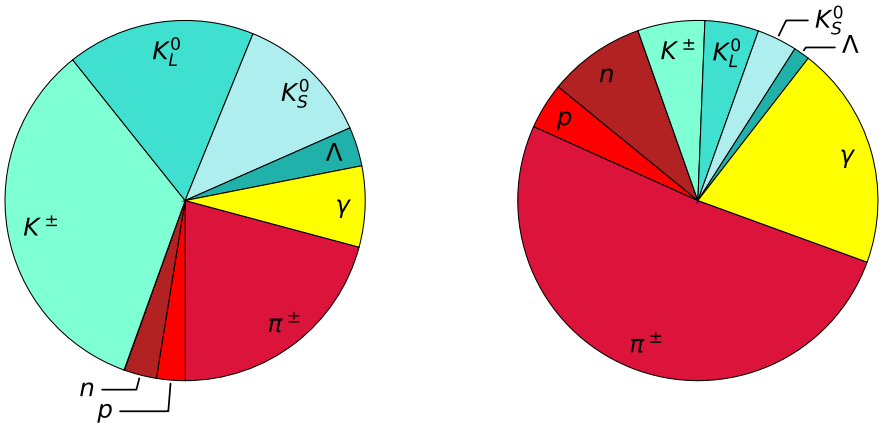
In addition to that, the sizes and the edges of the calorimeter cells are slightly modified for this thesis' relevant pseudorapidity of up to $|\eta| < 3$. This is done to enable an easy one-to-one representation of cells as pixels in the jet images. The ECAL cells are mostly left unchanged, only a small irregularity at $\eta = 1.5$ is smoothed out.

The size of every cell in $\eta \times \phi$ is 0.0174×0.0174 rad. By default, the HCAL cells are a bit more unsteady with sizes of about 0.087×0.087 rad for $|\eta| < 1.65$, doubled at $|\eta| > 1.65$. This is adjusted to sizes of exactly 0.087×0.087 rad across the whole η range, with every cell edge exactly aligning with the edge of respectively 5×5 ECAL cells.

4 Jet images, tracking information and preprocessing

4.1 Jet selection

The jets in the simulated samples must be geometrically matched to quarks or antiquarks from the hard scattering process within a radius of $\Delta R = 0.5$. The according truth flavor is assigned to the matched jets. Further requirements on the jets are a transverse impuls $p_{\mathrm{T}} > 20\,\mathrm{GeV}$ and an absolute pseudorapidity $|\eta| < 2.5$. In addition, only the jet with the highest p_{T} in every event is kept. From a total of over 3.6 Million jets, this preselection results in just less than 800 000 s- and d-jets each. For understanding possible differences between the two types of jets, as discussed in Section 2.2, it is interesting to study which type of truth-level particle has the highest p_{T} in a jet. This particle is likely to stem directly from the hadronization of the prompt strange quark and is called the leading particle. Considered here are stable and detector stable particles, as well as K_S^0 and Λ baryons, as they are of particular importance for the discrimination between s- and d-jets. The leading particles of the simulated s- and d-jets are shown in the pie charts in Figure 4.1. About half of the leading particles in s-jets are $K^{\pm}$ or K_L^0, five times as many as in d-jets. Photons, which mainly stem from neutral pion decays, are the leading particle in about every fifth d-jet, roughly three times more often than in s-jets. This difference helps to discriminate between these jet types, as it allows to identify the s-jets based on their typically higher energy fractions in the HCAL, as described in Section 2.2. This is further illustrated in Section 6.2.6, where CNNs are trained to distinguish between s-jets with different types of leading particles. The higher ratio of K_S^0 and Λ baryons in s-jets also enables recognition of s-jets based on the SVs in the ID.



s-jets *d*-jets

Figure 4.1: Detector stable particles as well as K_S^0 and Λ baryons with the highest p_T in *s*-and *d*-jets at truth-level. Shades of blue correspond to particles containing strange quarks, red to composite particles without strange quarks and yellow to photons, which mostly come from neutral pion decays and deposit their energy in the ECAL.

4.2 Construction and preprocessing of jet images

For every jet, the energy depositions in the calorimeters within $\Delta R = 0.5$ around the jet axis are used for creating a jet image. For that, the depositions in the ECAL and the HCAL are handled separately. Since longitudinal segmentation of the calorimeters into layers is not simulated, this leaves one ECAL- and one HCAL-image for every jet. The ECAL image has a size of 65×65 pixels, where every pixel corresponds to a cell in the ECAL. The HCAL images have the same number of pixels. With the edges of the HCAL cells being exactly five times larger than those of the ECAL cells, every HCAL cell covers 5×5 pixels in the image. The intensity corresponding to an HCAL cell is equally distributed over all of the 25 pixels. Both images cover the same 1.131×1.131 rad space in the η-ϕ plane. For the scope of this thesis, the combination of the ECAL image and the HCAL with a shape of $2 \times 65 \times 65$ is referred to as a jet image. These jet images are very sparse, with an average of about six cells with energy depositions per jet image, for *s*- as well as *d*-jet images. The distributions of the number of non-empty ECAL and HCAL cells per jet image are shown in Figure 4.2.

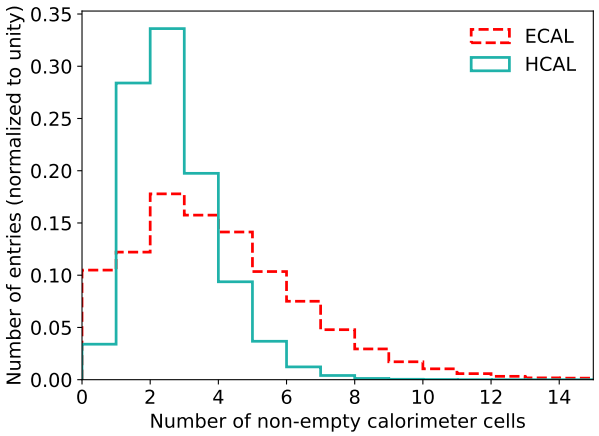


Figure 4.2: Distributions of the number of ECAL as well as HCAL cells with energy depositions per s- or d-jet in the simulated samples.

The intensities $I_{\text{cell},i}$ of the calorimeter cells, which are used to build the jet images, are calculated as

$$I_{\text{cell},i} = \frac{E_{\text{cell},i}}{\cosh \eta_{\text{cell},i} \cdot p_{\text{T, Jet}}}, \tag{4.1}$$

where $E_{\text{cell},i}$ is the energy deposition in the i^{th} cell, $\eta_{\text{cell},i}$ is the pseudorapidity of that cell and $p_{\text{T, Jet}}$ is the pseudorapidity of the jet. The intensity can be interpreted as the fraction of transverse energy deposited in the cell, compared to the total jet-p_{T}.

The calorimeter cells have discrete η and ϕ position, which can lead to unwanted effects and artifacts during the further preprocessing. For example, two energy depositions in two adjacent ECAL cells should explicitly correspond to the intensity of two adjacent pixels in the ECAL image. However, after a rotation, the positions of both cells might correspond to just one pixel, which would not be the intended calorimeter signature. To avoid this, instead of the η and ϕ values of each cell center, a random position inside the cell is used. This way, the positions of the energy depositions are independent from the positions of the calorimeter-cell grid and a generally valid calorimeter signature is achieved.

Before the intensities are calculated according to Equation (4.1), three additional preprocessing steps are applied. These are supposed to facilitate the training of neural networks by making the substructure and patterns in the jet images easier to extract and are inspired by previous work in which CNNs where used on jet images [12, 13].

First, the center of each jet image is adjusted to match the center of intensity instead of matching the jet axis. The coordinates of the center η_{center} and ϕ_{center} are calculated as

$$\eta_{\text{center}} = \sum_i \frac{I_{\text{cell},i} \cdot \eta_{\text{cell},i}}{I_{\text{cell},i}}, \qquad \phi_{\text{center}} = \sum_i \frac{I_{\text{cell},i} \cdot \phi_{\text{cell},i}}{I_{\text{cell},i}}, \tag{4.2}$$

where i enumerates all cells of the jet. The corresponding HCAL cell is set to be the pixel in the center of the jet image. Figures 4.3(a,b) and 4.4(a,b) show the averages of 40 000 s-jet ECAL and HCAL images before and after this preprocessing step. The corresponding d-jet images, which are very similar, are shown in Appendix A.1. The centering has a very small effect and is barely visible in the average images. This is the case, because the deposited energy in most cases is already centered around the jet axis. The intensity in the central 15×15 pixels in the average ECAL image as well as the average HCAL image is increased by about 1% by this preprocessing step.

In the second preprocessing step, the image is rotated, so that the principal axis of every jet's intensities are aligned vertically. The principal axis is defined as the axis $\vec{n}$ in

$$T_{\text{max}} = \max_{|\vec{n}|=1} \frac{\sum_i |I_{\text{cell},i} \vec{r}_i \vec{n}|}{\sum_i |I_{\text{cell},i} \vec{r}_i|} \tag{4.3}$$

with the jet's cell intensities $I_{\text{cell},i}$ and their position in the η-ϕ plane with respect to the center of intensity $\vec{r}_i$. The axis $\vec{n}$ is calculated iteratively in increments of 4°. Around the axis that maximizes T_{max}, another eight axes in increments of 1° are considered. The axis that achieves the highest T_{max} value is chosen as principal axis and the positions of the energy depositions are rotated in η and ϕ accordingly. There are two possible ways for rotating the images to achieve a vertical principal axis. However, regardless of which one is applied, the end result stays the same because of the third preprocessing step. The average ECAL and HCAL images after the rotation are shown in Figure 4.3(c) and 4.4(c). The intensity in the left and in the right side of the image is reduced, while the intensity along the vertical central axis is increased.

In the third preprocessing step, the depositions are flipped along the central axes in both η and ϕ in such a way, that the upper right quadrant has the highest sum of intensity. The result is shown if the Figures 4.3(d) and 4.4(d). After the flip, about 29% of the total energy deposited in the calorimeters is in the upper right quadrant of the average jet images. About 28% of the energy are in the upper left quadrant and just less than 23% and 21% are in the lower quadrants. If there is any jet substructure encoded in the energy depositions, they should be made easier

to learn, since any secondary or tertiary energy *clusters* are always placed into the same positions in the images, namely above or to the upper right of the main cluster respectively.

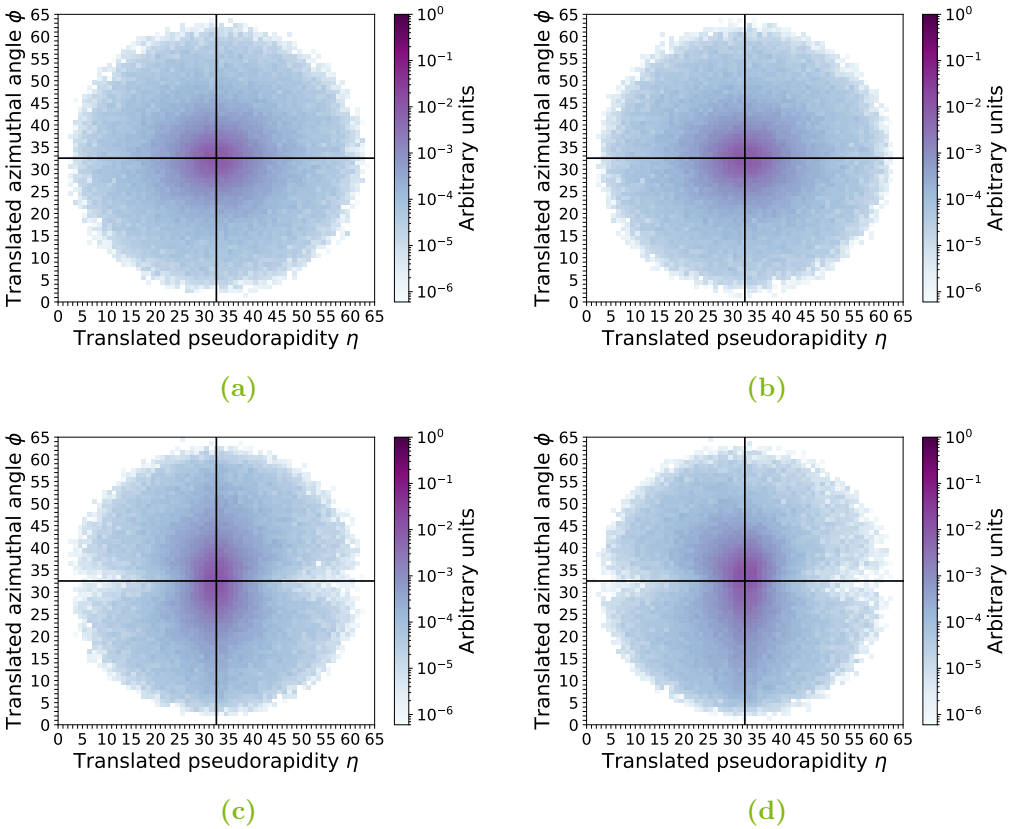


Figure 4.3: Average of 40 000 ECAL *s*-jet images at different stages of the preprocessing. In figure (a) with no preprocessing, in figure (b) centered on the center of intensity, in figure (c) rotated to a vertical principal axis and in figure (d) flipped to have the highest sum of intensity in the upper right quadrant.

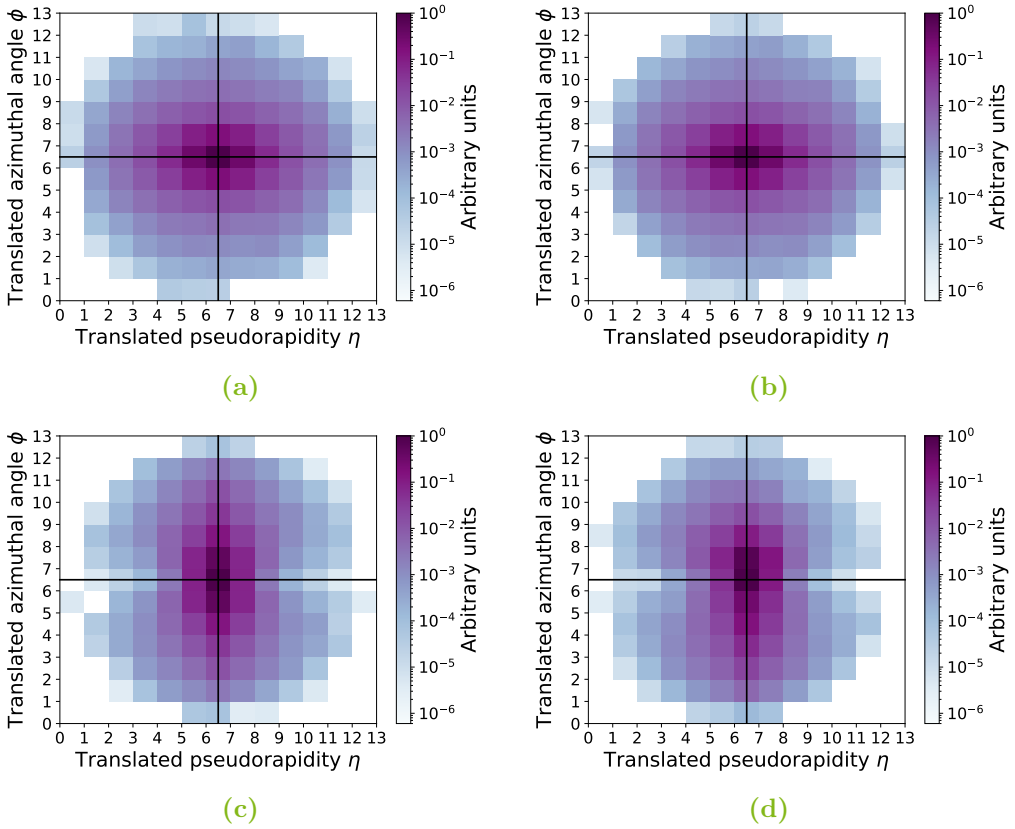


Figure 4.4: Average of 40 000 HCAL s-jet images at different stages of the preprocessing. In figure (a) with no preprocessing, in figure (b) centered on the center of intensity, in figure (c) rotated to a vertical principal axis and in figure (d) flipped to have the highest sum of intensity in the upper right quadrant.

4.3 Preprocessing of tracking information

In addition to the jet images, properties of the associated tracks of the jets are used as well. Tracks are associated to a jet, if they are within $\Delta R = 0.5$ around the jet axis. Figure 4.5 shows the distribution of the number of tracks per s-jet and per d-jet. Both distributions are very similar. On average, the jets have between four and five tracks, A total of twelve properties are defined, analogously to Ref. [6]:

1. the track p_{T},
2. the track η,
3. the signed difference between the track η and the jet η, $\Delta\eta$,
4. the signed difference between the track ϕ and the jet ϕ, $\Delta\phi$,
5. the ratio of the track p_{T} and the jet p_{T}, x,
6. the track momentum perpendicular to the jet axis, $p_{\mathrm{T}}^{\mathrm{rel}}$,
7. the track momentum in the direction of the jet axis, p_z^{rel},
8. the transverse impact parameter of the track, d_0,
9. the longitudinal impact parameter of the track, d_z,
10. the distance in x-direction, Δx, between the PV and either the associated SV (if the track is associated to a SV) or the position of the track's first hit in the ID (otherwise),
11. Δy, defined analogously to Δx,
12. Δz, defined analogously to Δx.

Tracks are associated to a SV if they share the same point of origin within a transverse distance $4\,\mathrm{mm} < R < 450\,\mathrm{mm}$ around the beam axis. The ID layers from which the positions of the tracks' first hits in the ID are calculated are defined to be at radii $R = 50\,\mathrm{mm}$, $R = 90\,\mathrm{mm}$, $R = 120\,\mathrm{mm}$, $R = 300\,\mathrm{mm}$, $R = 370\,\mathrm{mm}$, $R = 440\,\mathrm{mm}$ and $R = 510\,\mathrm{mm}$. An idealized but reasonable 100% tracking efficiency is assumed. Tracks from particles that are produced at $R > 510\,\mathrm{mm}$ are not considered. The track properties describe track angles, both with regard to the beam axis as well as the jet axis, and track momenta, in general and with regard to the jet p_{T}. The origin of the tracks is described with impact parameters as well as the coordinates of an associated SV, if applicable, or else, the first hit in the ID. Some of the track properties are deliberately redundant, so that they can easier be learned by neural networks. To facilitate the learning of correlations between tracking information and jet images, two additional variables with values between 0 and 1 are added for

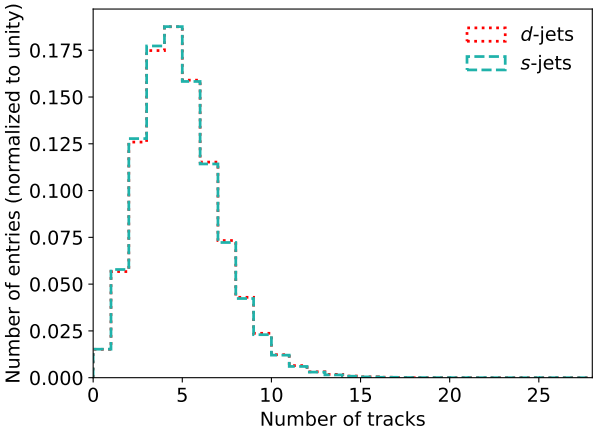


Figure 4.5: Distribution of the number of tracks per s- and d-jets in the simulated samples.

every track, that correspond to the translated η-ϕ position of the track in the jet image after the preprocessing. As described in Section 2.2, tracks from SVs are especially relevant for the classification of s- and d-jets. Therefore, the tracks are ordered based on their transverse distance $R = \sqrt{(\Delta x)^2 + (\Delta y)^2}$ of their SV or first hit in the ID from the PV. Tracks with the same R are ordered based on their p_{T}. This way, tracks that share the same SV occupy adjacent positions in the input for LSTMs, which helps with learning from those SVs.

4.4 Reweighting in jet-η and jet-p_{T}

The discrimination between s- and d-jets has to be independent of the jet-η and jet-p_{T}. Possible discrimination power in these variables is not inherent to the jet flavor but rather due to the specific production process of the samples. Even though the differences are very small, s-jets are on average slightly more central and have a higher jet-p_{T} compared to d-jets. In order to keep the neural networks from exploiting these correlations of jet-η and jet-p_{T} to the jet flavor, training weights w_{train} are applied to every jet. To obtain reasonable weights, the η-ϕ space is divided into several regions, with the goal to use specific weights for every region. Generally, smaller regions lead to better results as they minimize differences between the s-jet and d-jet distributions inside that region. However, the regions still need to be large enough to ensure sufficient statistics in every region. Since the exact region boundaries only have a marginal effect on the result and differences in the

distributions are small even without the reweighting, the regions are divided up with a rather heuristic algorithm: First, 25 η-slices are defined with every slice containing the same number of d-jets. Then, p_T-slices are defined, that form rectangular regions together with the η-slices. The size of every p_T-slice is chosen in such a way that every region in a slice contains a minimum of at least 1000 d-jets, which leaves a tolerable statistical uncertainty of about 3% for these regions.

One possible approach is, to choose weights that make the distributions of both s- and d-jets as flat as possible in the η-ϕ space. Therefore, jets in sparsely populated regions get heavier weights. One problem with this approach is, that the very rare jets with the highest p_T get weights, more than two orders of magnitude higher than jets in heavily populated regions. This is not desirable, as it leaves high emphasis on a small number of jets, leading to statistical instability. Instead, weights that adjust the d-jet distribution to match the s-jet distribution are chosen. For every s-jet, w_train is equal to one. The weight applied to each down jet in a certain region is calculated from the ratio of s-and d-jets in that region:

$$w_\mathrm{train} = \frac{\text{number of } s\text{-jets in region}}{\text{number of } d\text{-jets in region}} \tag{4.4}$$

The regions and their corresponding weights for d-jets are shown in Figure 4.6(d). The Figures 4.6(a) and 4.6(b) show the effect of the reweighting. The vast majority of the jets are within $20\,\mathrm{GeV} < \text{jet-}p_\mathrm{T} < 40\,\mathrm{GeV}$. This is also the energy region where the absolute difference between s- and d-jets is the largest. The reweighting reduces those differences by a significant amount. Therefore, machine learning algorithms should not be influenced by jet-η and jet-p_T after the reweighting.

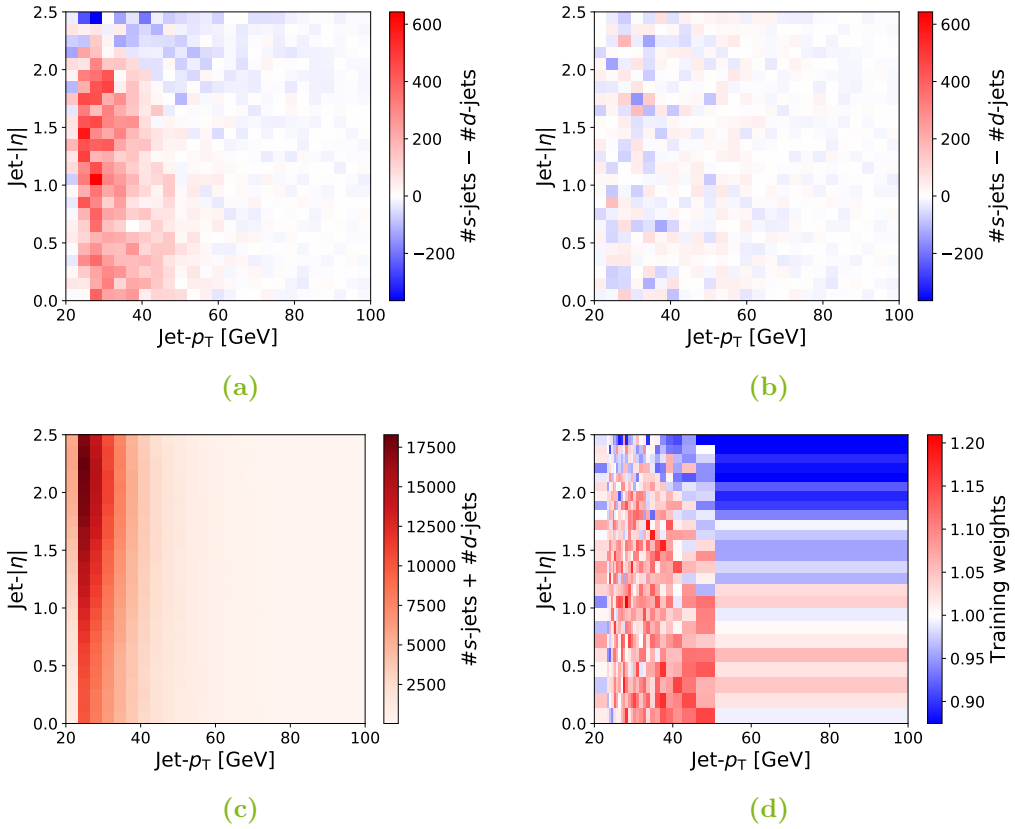


Figure 4.6: Total bin-difference in the η-p_T space between s- and d-jets before (top left) and after (top right) reweighting. Total bin content of s- and d-jets (bottom left) and η-ϕ regions (bottom right) for reference.

5 Application of artificial neural networks for classification of jet images and tracks

5.1 Introduction to neural networks

For complex classification problems such as s-tagging, artificial neural networks (ANNs) can possibly achieve better performances than more traditional methods. A variety of features of one particular jet could for example be fed as a vector $\vec{x}$ into the input layer of a feed-forward neural network (FFNN), where they would then be passed through an arbitrary number of hidden layers, each consisting of any desired number of so called nodes. Each node takes the output of all nodes from the previous layer as input. Nodes store a vector of weights $\vec{w}$, one weight for every input value x_i, which is used together with a bias b to perform an affine transformation $y = \vec{w}^\top \cdot \vec{x} + b$. From this, the output of the node is then calculated with a nonlinear activation function $\sigma(y)$ and passed to every node of the next layer. Without this nonlinearity, a FFNN would just perform a linear transformation of the input vector.

This type of layer is called a fully connected layer. The output layer of a FFNN usually consists of just one node in the case of a binary classification, or one node per class in the case of a multinomial classification. For binary classifications, a cut on the output of the single node gives a class prediction, for example, either "s-jet" or "d-jet". When distinguishing between more than two types of jets, each node of the output layer corresponds to one class. The jets are predicted to belong to the class of the node with the highest output value. Often, fully connected layers are used as the last layers of a more advanced ANN, as described in the following sections.

The weights and biases of all the nodes first have to get optimized by training the ANN (the model) with example inputs that are labeled with the class they belong to. This training usually requires a large number of training samples, which can be generated with MC simulations. During training, each time the classification of an arbitrary number of samples (batch size) is done, the performance of the model is evaluated based on a chosen loss function. This function quantifies with a loss value (the loss), how well the predictions of the model match the true labels. With

a so-called backpropagation algorithm, the gradient of the loss with respect to the weights is approximated and in order to minimize the loss, the weights are updated according to an optimizer algorithm. In order to iteratively decrease the loss of the model, the full training sample is used multiple times, each iteration being called an epoch. This training process is monitored with an independent validation sample, on which the loss is calculated after each epoch. Generally, the model is fully trained when the loss on this sample does not significantly decrease anymore. After the training process, the model can then be used to classify data with unknown labels.

The behavior of a FFNN is heavily dependent on the number of its hidden layers as well as the number of nodes in each layer, which are hyperparameters of the network. Other hyperparameters are the employed activation functions, the number of epochs and the batch size used for training, the employed optimizer algorithm, which has some more parameters itself, the initialization of the model weights as well as regularization. Regularization methods can be necessary to keep the model from overtraining, which is the problem, that a model might adjust its weights too specifically to the features of the training sample, instead of learning the underlying pattern. Then, the model does not generalize well and performs worse on unknown samples. One method to prevent this, is to randomly ignore nodes in every training batch, which is called dropout. The probability for each node to be ignored is called the dropout rate. Applying dropout forces a model to not rely too heavily on specific nodes and it can help with generalization.

5.2 Convolutional neural networks and jet images

It would be possible to use every pixel of a jet image as an input for a FFNN. However, this would be computing intensive and by breaking down the images into one-dimensional vectors, not take advantage of the underlying spacial structures. This is why CNNs are usually used for the classification of images. They take the jet images represented as a three-dimensional matrix, in this thesis with a shape of $2 \times 65 \times 65$, as input. Convolutional layers employ so called filters, weight matrices of usually rather small size, which are consecutively applied in a grid to the whole input. For every point of the grid, a filter multiplies each of its weights with the corresponding value of the input matrix, as illustrated in Figure 5.1. As described in Section 5.1, an activation function is applied to the sum of these multiplications. For each filter, all of those outputs for every point of the grid are passed to the next layer. The output of the last convolutional layer is flattened and passed to fully connected layers, eventually giving a prediction for the jet image.

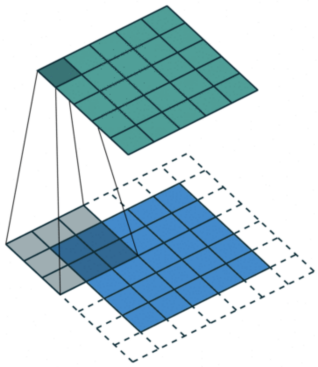


Figure 5.1: Abstract visualization of of how a convolutional filter (gray) converts an input matrix (blue) into an output matrix (green) [14].

Each filter can possibly learn distinctive features and patterns. The first layer's filters are prone to learning patterns of smaller size, their scope being limited by their filter size. Filters in subsequent layers can pick up on more sophisticated and more abstract patterns. Taking in the output of previous layers, their weights are also affected by a larger area of the input images.

The number of convolutional layers in a CNN, the number of filters per layer as well as their filter size all are important hyperparameters of the network, that need to be chosen according on the task at hand. As described in Section 4.2, the jet images are very sparse and therefore do not contain a lot of information. From this, one can already conduct, that CNNs do not necessarily need a large number of trainable weights, to learn all features in the jet images. On the other hand, filters in the first layer should be large enough, to simultaneously cover multiple non-empty pixels, in order to learn underlying patterns.

5.3 Recurrent neural networks and tracking information

Recurrent Neural Networks (RNNs) are another type of ANNs. RNNs take a sequence of feature vectors as input, in this case, the features of the track sequence belonging to a jet. The feature vector of each track in this sequence is called a time step. In this thesis, a subclass of RNNs, Long Short-Term Memory networks (LSTMs) are used. An LSTM layer consists of a module, depicted in Figure 5.2. After a random initialization for the first time step, the module iteratively takes the next time step $\vec{x}_t$ as well as the output vector of the previous time step $\vec{h}_{t-1}$ as inputs. These inputs are concatenated and further processed through the module.

The module also has a hidden cell state $\vec{C}$, which is passed and modified through all the time-step iterations of the module. The cell state is a vector, which stores relevant information from previous time steps. The size of the cell state, also called the number of units, which is is equal to the size of the output vector, is a hyperparameter. Each time step can remove and add information to the cell state. This is controlled by the interaction between the inputs and the so called gates of the module. An LSTM module has three gates: The forget gate, the input gate and the output gate. Gates consist of one or two fully connected layers, each layer being fed with the concatenation of the module inputs $[\vec{x}_t, \vec{h}_{t-1}]$. The number of nodes and therefore the size of the output of each layer is equal to the number of the cell state's units. The weights of these gate layers are optimized during training.

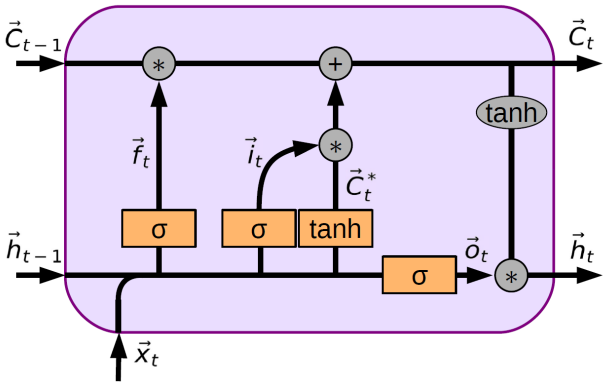


Figure 5.2: Schematic overview of the fully connected layers (orange), the information flow (black lines and arrows) and the arithmetic operations (gray) in an LSTM module.

The forget gate has one layer, which nodes use the sigmoid function as activation function. Therefore, its output $\vec{f}_t$ is a vector with a number between zero and one for every unit in the cell state. By element-wise multiplication of $\vec{f}_t$ with the cell state $\vec{C}_{t-1}$, information is removed from the cell state, especially from units that are multiplied with a number close to zero.

The input gate consists of two layers. One uses the hyperbolic tangent as activation function. Its output vector $\vec{C}_t^*$ has entries between -1 and 1. It is going to be used to update the cell state through element-wise addition and can be thought of as the *prototype* for an updated cell state. First though, $\vec{C}_t^*$ is element-wise multiplied with values between 0 and 1 from the output $\vec{i}_t$ of the gate's other layer, which uses the sigmoid function as activation function. This multiplication decides,

how much the cell state is updated. In summary, the cell state is updated with each time step, according to

$$\vec{C}_t = \vec{f}_t * \vec{C}_{t-1} + \vec{i}_t * \vec{C}_t^*, \tag{5.1}$$

where $*$ is the operator for element-wise multiplication. Finally, the output gate, which is one layer with the sigmoid function as activation function, decides, how much of the cell state is made visible as output of the module. For this, the cell state is first pushed through the hyperbolic tangent and then element-wise multiplied with the gate's output vector $\vec{o}_t$, according to:

$$\vec{h}_t = \vec{o}_t * \tanh\left(\vec{C}_t\right) \tag{5.2}$$

The module-output vectors of each time step can be used to build a new sequence as input for the next LSTM layer. Or, the module output of only the last time step can be used as input for fully connected layers, in order to predict the flavor corresponding to the tracks.

6 Discrimination between strange and down jets

In this section, methods to distinguish between s- and d-jets based on the calorimeter and tracking information are explored. First, a simple benchmark method is presented, then, different CNN architectures are tested and finally, combinations with LSTM layers that use tracking information are studied. To evaluate the performance of a method, the accuracy (ACC), which is the rate of correctly classified jets out of all jets, and the so called receiver operating characteristic (ROC) curve is considered. A ROC curve plots the signal efficiency ϵ_S, which is the rate of correctly classified s-jets, against the background efficiency ϵ_B, which is the rate of incorrectly classified d-jets, for different cut thresholds on the output of a classifier. A random classification of samples would always yield an ϵ_S equal to ϵ_B for every cut threshold. An additional performance measure is the area under the curve (AUC), which is 0.5 for a random classification and 1.0 for perfect separation. The AUC is equal to the probability, that a random signal sample (s-jet) is ranked higher than a random background sample (d-jet).

6.1 Using a simple cut on the ECAL fraction

For a simple benchmark method, the fraction of energy deposited by a jet in the ECAL divided by its total energy deposition $f_{\mathrm{ECAL}} = E_{\mathrm{ECAL}} \,/\, E_{\mathrm{total}}$ is used. As discussed in Section 2.2, this variable is the main discriminating feature between s- and d-jets' energy depositions in the calorimeters. Figure 6.1 shows the f_{ECAL} distribution for 750 000 s-jets and 750 000 d-jets. The best cut on the distribution achieves an accuracy of 0.567 and an AUC of 0.591. The corresponding ROC curve is shown in Figure 6.14. With the simplistic detector simulation, about 13% of the s-jets (9% of the d-jets) deposit all their energy in the HCAL and about 2% of the s-jets (5% of the d-jets) in the ECAL. Therefore, their f_{ECAL} is exactly equal to zero or one, respectively. Since these jets cannot be separated with a cut on f_{ECAL}, this method can only achieve signal efficiencies larger than 13% and smaller than 98% and background efficiencies larger than 9% and smaller than 95%, due to the simplistic detector simulation.

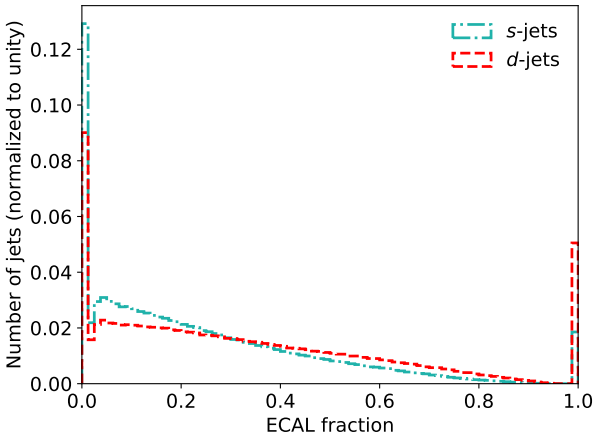


Figure 6.1: Distribution of the ECAL fraction for s- and d-jets.

6.2 Using CNNs on the jet images

In the following, the use of CNNs for the classification of the jet images is explored. The neural networks are implemented with Keras [15] using TensorFlow [16] as back-end. As described in Section 5.2, the network's performance depends on a variety of hyperparameters. However, the differences in both accuracy and AUC are not significant, when varying the most important hyperparameters in a reasonably large range. Since the training process of the networks is also computing intensive with training times between one and two days for most of the tested network architectures on a training sample of 300 000 jet images and even considerably longer for certain architectures, a multi-dimensional grid search to optimize the hyperparameters independently of each other is not feasible. Instead, one manually optimized CNN is used as a basis in order to vary selected hyperparameters sequentially to find the optimal performance. An optimized CNN with the best hyperparameters found with this hyperparameter optimization is trained and tested in Section 6.2.8.

The architecture of the CNN used as a basis for optimizing the hyperparameters is referred to as the "standard configuration" and is visualized in Figure 6.2. First, the jet images of size $2 \times 65 \times 65$ are passed through three convolutional layers, each followed by a max-pooling layer. The first layer performs a three-dimensional convolution with four filters of size $2 \times 16 \times 16$. Three-dimensional filters are chosen in order to learn correlations between the ECAL and the HCAL image. The other convolutional layers are two-dimensional and have four 8×8 or 4×4 filters, respectively. In this thesis, filters are always quadratic and the filter size is referred

to by their edge length, in this case 16, 8 and 4. The first layer uses a stride-size of two, the others of one. Each pooling layer has a filter size of two. In addition, a dropout rate of 0.2 is applied to each convolutional layer. The output of the final pooling layer is flattened and concatenated with the jet-η and jet-p_{T}, which makes it possible for the network to learn correlations between the jet images and the jet-η and jet-p_{T}. Finally, the network consists of two fully connected layers with 128 and 64 nodes and the output layer with two nodes, which use the sigmoid function [1] as activation function, in order to classify the jets into s- and d-jets. For this binary classification, just one output node should have been used. However, using two nodes instead should have the same result. In the following, when the classifier output is mentioned, the output that corresponds to s-jets is meant. All other convolutional and fully connected layers use the ReLU function [2] as activation function. During the training, the binary crossentropy is used to calculate the loss and Adam [17] with a learning rate of 0.001 is used as optimizer. The neural network is trained over 30 epochs with a batch size of 5000 jet images.

Throughout the following studies, different numbers of jets are used for the training process. The training process is monitored with an validation set of 100 000 jets and after the training process, the model with the best validation accuracy is chosen. These 100 000 jets are exclusively used for the monitoring and another 400 000 jets are used for an unbiased evaluation of the accuracy and the AUC achieved with this model. This large size ensures that small performance changes are not dominated by statistical fluctuations. An independent test sample with 400 000 jets is used to test the optimized CNN. The training, evaluation and test samples all consist of an equal number of s- and d-jets. The jet-image pixel intensities and the other input features (jet-η, jet-p_{T} and, where applicable, each of the twelve track features) are scaled with Scikit-Learn's [18] StandardScaler, which transforms each pixel or feature by subtracting the median of all pixels or the feature and scaling the variance to one.

6.2.1 Dependence on training statistic

In order to find a sufficiently large training statistic at which the performance of the CNN saturates, the standard-configuration CNN is trained with 120 000, 300 000, 500 000 as well as 800 000 jet images. With a larger training-sample size, networks with larger capacity are expected to perform better. Hence, an additional CNN, for which the effective capacity is increased by using six instead of four

[1] The sigmoid function $\sigma(y) = \frac{1}{1+\exp(-y)}$ has a range from 0 to 1 and is usually used as activation function for a single output node for a binary classification.

[2] The ReLU function $\sigma(y) = \max(0,\ y)$ has a non-vanishing gradient for domain values $y > 0$.

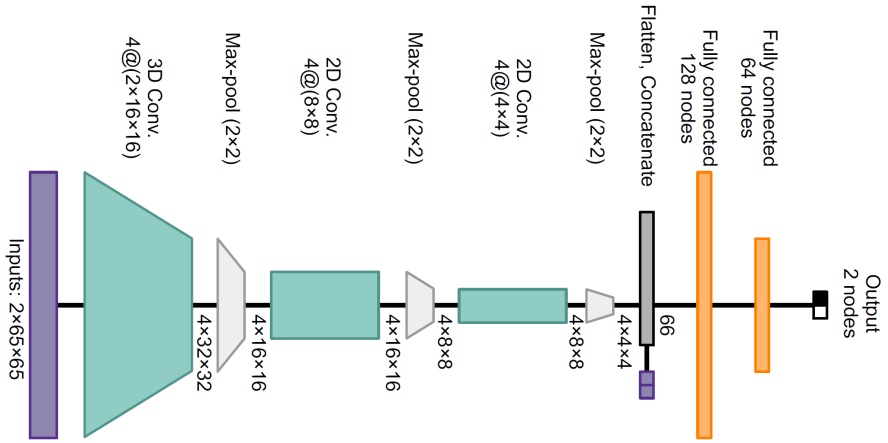


Figure 6.2: Convolutional neural network architecture that is used for evaluating the influence of hyperparameters on the discrimination strength between down- and strange-quark jet images. The images are of size $2 \times 65 \times 65$ and cover an area of 1.131×1.131 rad in the η-ϕ plane. The output of the convolutional layers is concatenated with the jet-η and p_{T}.

filters and removing the regularization with dropout, is trained and evaluated as well.

For this network, the accuracy and the AUC, as shown in Figure 6.3, improve considerably by about 0.008 when increasing the training sample from 120 000 to 300 000 jet images. However, using 500 000 or 800 000 instead of 300 000 jet images for the training process yields only marginal improvement. The accuracy improves by less than 0.001 and the AUC by about 0.002. The standard-configuration network achieves similar AUCs. It also achieves a significantly better AUC when using 300 000 instead of 120 000 jet images in the training process. The accuracy increases with training-sample sizes of 300 000, 500 000 and 800 000 but is roughly 0.005 worse than the accuracy achieved by the network with higher capacity. Since training time is a relevant factor, the training-sample size of 300 000 is used for the following studies.

Due to a mistake, the validation sample for the hyperparameter optimization first consisted of 200 000 s-jet images but only of about 180 000 instead of 200 000 d-jet images. On this sample, the standard-configuration network had achieved higher accuracies, comparable to those of the network with the increased capacity. Therefore, the standard-configuration network was still used in the further hyperparameter optimization. While the performances were reevaluated with a balanced validation sample and the training of the optimized architecture and subsequent studies was

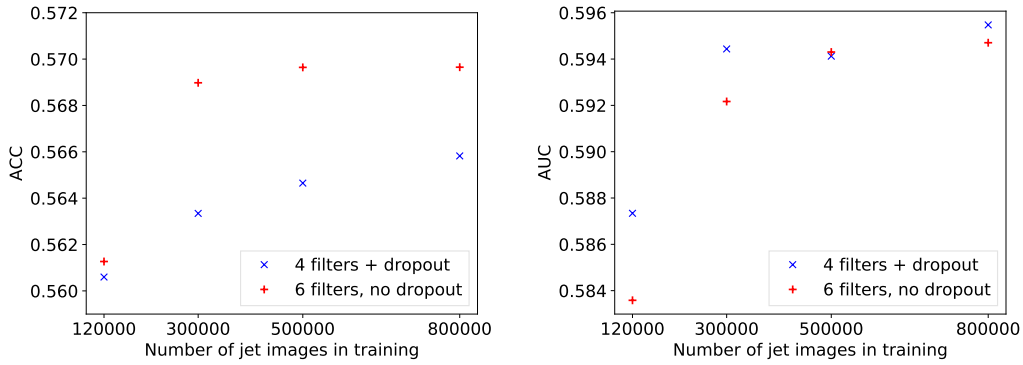


Figure 6.3: Classification accuracy and AUC of s- versus d-jets of two networks with different capacities as a function of the number of jet images in training on an independent sample of 400 000 jet images.

redone, time did not allow to redo the full hyperparameter optimization.

6.2.2 Number of convolutional layers

Next, the performance of networks with up to four convolutional layers is evaluated. Additionally, a network with a stride-size of one instead of two in the first convolutional layer and networks with a pooling sizes of three instead of two are also evaluated. The accuracy and the AUC achieved by those networks is shown in Figure 6.4. The network with one convolutional layer achieves the second lowest AUC and a moderate accuracy compared to the other architectures. This subpar performance might be due to the fact that one single layer cannot learn more abstract features from the jet images. The network with four convolutional layers on the other side might be too complex and has by far the lowest accuracy.

The networks with two or three convolutional layers all achieve similar performances with accuracies between 0.565 and 0.570 and AUCs between 0.590 and 0.594. Only the network with a stride-size of one in the first layer has a lower AUC of 0.589. This can be explained through the sparsity of the jet images. Translating a filter by just one pixel in a jet image has a high chance to still cover the exact same energy depositions as before, so that no new information can be learned and the training of meaningful filter weights is possibly hindered.

For the optimized architecture, three convolutional layers with pooling sizes of

two are chosen, even though the architecture with two convolutional layers has a better accuracy and the network with a pooling size of three in the first layer has a better AUC. These differences of about 0.001 are very minor though and the corresponding ROC curves in Figure 6.5 show little difference. However, the configuration with three convolutional layers and a pooling size of two has the highest capacity and the largest output size. This should synergize better with the optimized architecture, which has, as described in Section 6.2.4, more nodes in the fully connected layers.

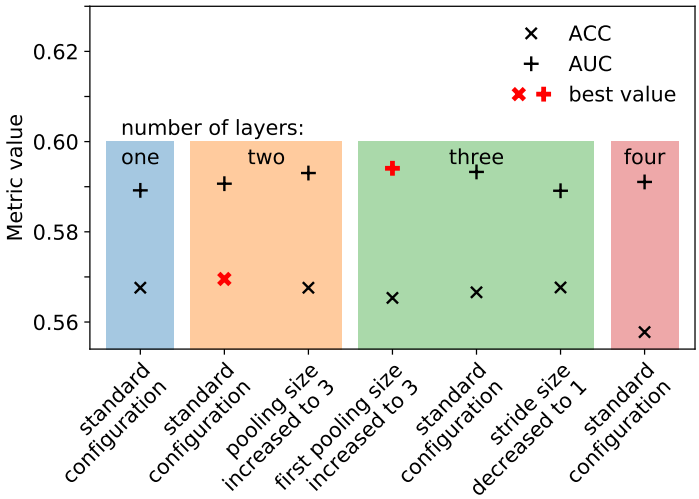


Figure 6.4: Accuracy and AUC achieved by CNNs with different numbers of convolutional layers, pooling sizes and stride-sizes on an independent validation sample of 400 000 jet images.

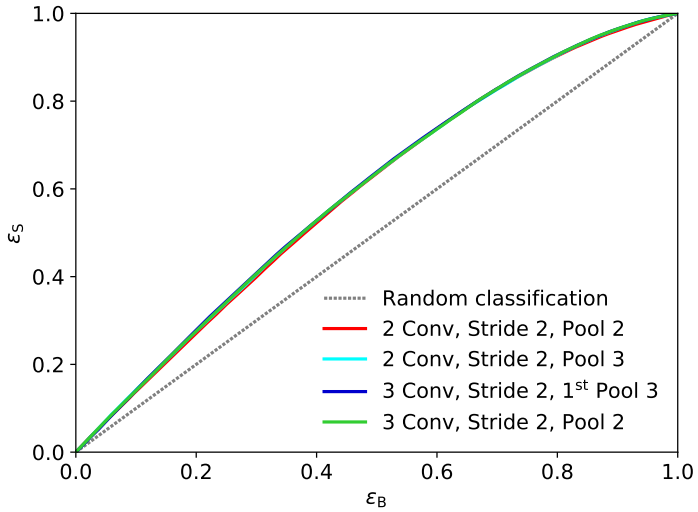


Figure 6.5: ROC curves for the four networks with the best accuracies and AUCs during the optimization of the number of convolutional layers, evaluated on the validation sample.

6.2.3 Number and size of filters

The standard-configuration network is used as basis to evaluate the performance of filter sizes in the first convolutional layer of 4, 8 or 16, with the filter size in every subsequent convolutional layer being halved, down to a minimum size of 2. In the following, these filter sizes are referred to by the filter size in the first convolutional layer. For every filter size, networks with different numbers of filters are trained. The results are shown in Figure 6.6. The networks with filter sizes of four achieve AUCs up to 0.005 worse than those of the other networks. Smaller filters have a higher chance of covering less energy depositions in a jet image at once. For example, in a jet image where all energy depositions are more than four pixels apart, filters of size four can never cover more than one energy deposition at once. This makes it harder to learn specific shapes and structures from the images.

When comparing networks with filter sizes of 8 and 16, they achieve similar good performances with filter sizes of four, eight and twelve. Only the network with four filters of size 16 achieves an accuracy that is about 0.005 worse than the networks' with eight or twelve filters accuracy. The network with twelve filters of size 16 achieves the best accuracy and AUC, which are both about 0.001 better than the next highest accuracies and AUCs. Such minor differences can very well be due

to statistical fluctuations and training the same network a second time can yield slightly different results. Nevertheless, as was shown in Figures 4.3 and 4.4, a large part of the energy depositions are close together in the center of the jet images. When a filter of size 16 scans a jet image, it is likely to simultaneously cover most or even all of the depositions at some point. This might facilitate the learning of substructure and therefore be a very slight advantage over filter sizes of 8. That the scan covers all depositions at some point and only subsets of the depositions when scanning the edges of the jets might also explain the subpar performance with four filters, since these might not suffice to account for both of those scenarios well enough.

The networks with 16 or 64 filters achieve worse performance than the networks with 8 or 12 filters. The network with 16 filters has by far the worst performance. A reason for the inferior performance could be the larger output size of the convolutional layers with more filters, which is, analogously to the network architecture shown in Figure 6.2, equal to $4 \times 4 \times$ number of filters. The rather small fully connected layers might not synergize well with this larger output size. Therefore, variations of the networks with 16 and with 64 filters, that use two fully connected layers with 512 and 256 nodes instead of 128 and 64 nodes are trained as well. The performance of the network with 16 filters is better than with the smaller number of nodes but still inferior to the network with 12 filters. The network with 64 filters performs worse with the higher number of nodes.

Since the best accuracy as well as the best AUC is achieved by the network with 12 filters of size 16, these hyperparameters are chosen for the optimized network. On the erroneous validation sample that was initially used during the hyperparamter optimization, the network with 4 filters of size 16 had performed considerably better. This is why the standard configuration was not updated to 12 filters and networks with 4 filters are still used for the further hyperparameter optimization.

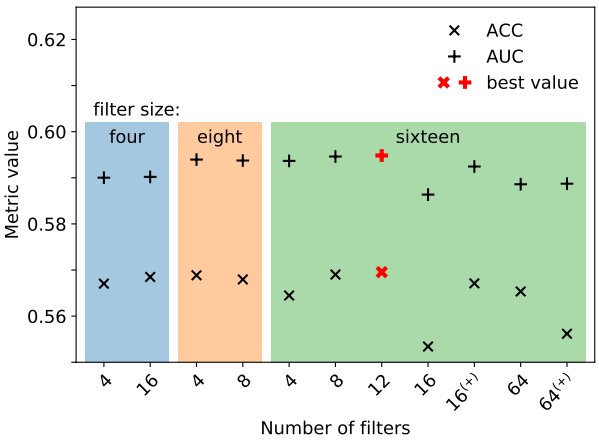


Figure 6.6: Accuracy and AUC achieved by CNNs with different numbers of filters and filter sizes on an independent validation sample of 400 000 jet images. Numbers of filters marked with $^{(+)}$ refer to networks with 512 and 256 nodes instead of 128 and 64 nodes in the fully connected layers.

6.2.4 Number and size of dense layers

In the following, standard-configuration networks with varying numbers of fully connected layers and nodes in these layers are trained and evaluated. For networks with more than one fully connected layer, the number of nodes is halved in every subsequent layer. These networks are referred to by their number of nodes in the first layer. The results are shown in Figure 6.7. Generally, the differences in accuracy and AUC when varying the number of fully connected layers or nodes are even smaller than with variations of the other hyperparameters optimized so far. Depending on the number of layers, networks with 128, 256, 512 and 1024 nodes are trained. The networks with one or two fully connected layers achieve the best accuracy as well as the best AUC with 512 nodes. These networks with lower as well as higher numbers of nodes achieve a worse performance. For the network with three fully connected layers, the best performance is achieved with only 256 nodes, which makes sense since networks with more layers can reach the same capacity with a smaller number of nodes. The differences between the best accuracies and AUCs achieved with one, two or three fully connected layers are smaller than 0.001 but the network with two fully connected layers is slightly better than the others.

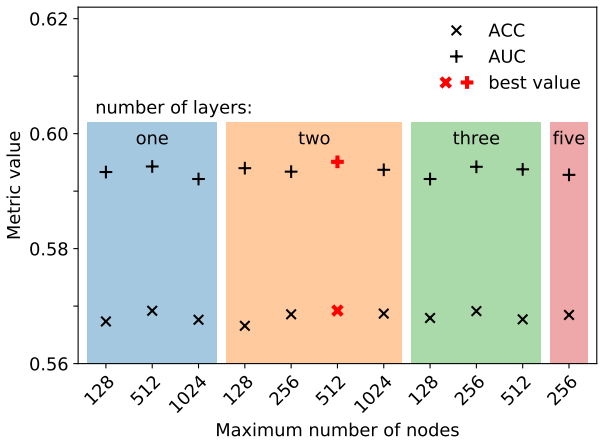


Figure 6.7: Accuracy and AUC achieved by CNNs with different numbers of dense layers and nodes on an independent validation sample of 400 000 jet images.

A network with five fully connected layers and 256 nodes is trained as well. While the achieved accuracy is very close to the peak accuracies of the networks with fewer layers, the AUC is more than 0.002 below the AUC achieved by the network with two fully connected layers and 512 nodes. Differences of 0.002 or larger are usually not due to fluctuations. Therefore, this decrease indicates that the output of the convolutional layers probably does not require a larger capacity in the fully connected layers. For the optimized network, two fully connected layers with 512 and 256 nodes are chosen. The standard-configuration network used in the next section as a basis for further hyperparameter optimization is also updated and uses this configuration instead of the 128 and 64 nodes as used before.

6.2.5 Other hyperparameters

As described in the beginning of this section, the standard-configuration network uses a dropout rate of 0.2 on every convolutional layer, a batch size of 5000, the ReLU function as activation function for the convolutional and fully connected layers and the Adam optimizer with a learning rate of 0.001. In the following, networks in which these hyperparameters are varied are evaluated. The results are shown in Figure 6.8, where the accuracy and AUC achieved by the standard-configuration network is marked with the red, dotted lines. The networks with no dropout or a dropout rate of 0.3 achieve AUCs that are about 0.002 or 0.003 worse than for the standard configuration, respectively. The network with the dropout rate of 0.1 achieves a slightly better accuracy but also a slightly worse

AUC. Since dropout rates of 0.1 and 0.2 achieve a comparable performance, the higher regularization with the dropout rate of 0.2 is chosen for the optimized network.

The smaller batch sizes during the training process slightly increase the accuracy and achieve about the same AUC, compared to a batch size of 5000. Increasing the batch size to 10 000 decreases the AUC by about 0.002. Since the batch size of 3000 has the best performance, this size is chosen for the optimized network.

The network that uses the leaky ReLU function [3] with a slope of 0.01 as activation function achieves a slightly better accuracy and AUC, compared to the standard-configuration network. With a slope of 0.2, the accuracy is even better but the AUC is slightly worse than with the standard-configuration network. With a slope of 0.3, both accuracy and AUC are about 0.001 better than with the standard configuration. Even though this is a small performance difference, the leaky ReLU function with a slope of 0.3 is chosen for the optimized network.

Additionally, networks that use different optimizers with default configuration are evaluated. Neither the network with the RMSprop [19], the AdaDelta [20] or the AdaGrad [21] optimizer achieve a significantly better accuracy and all achieve a slightly worse AUC compared to the Adam optimizer. Additionally, networks that use the Adam optimizer with larger as well as smaller learning rates are evaluated as well. The network with the larger learning rate of 0.005 achieves about the same accuracy but a very slightly worse AUC, when compared to the standard-configuration network. The network with the smaller learning rate of 0.0001 achieves a significantly worse accuracy, about 0.008 below the standard-configuration accuracy. Since it achieves the best performance, the Adam optimizer with a learning rate of 0.001 is kept for the optimized network.

Finally, a variation of the standard-network architecture with two additional convolutional layers is trained. These layers are parallel to the three convolutional layers of the standard architecture and also take the jet images as input. The additional layers each have 12 filters of size four. After both layers, a max-pooling layer is used with a pooling size of two. The output of the second layer is concatenated with the output of the standard-architecture convolutional layers and, as before, with the jet-η and the jet-p_{T} and then fed into the fully connected layers. With the smaller filters, the network might be able to learn additional patterns in the jet images, to which the bigger filters might not be sensitive to. The performance of this network is

[3] The leaky ReLU function $\sigma(y) = \begin{cases} \alpha y, & \text{if } y < 0 \\ y, & \text{otherwise} \end{cases}$ also has a non-vanishing gradient for $y < 0$ and therefore, nodes are less likely to get stuck at large negative values.

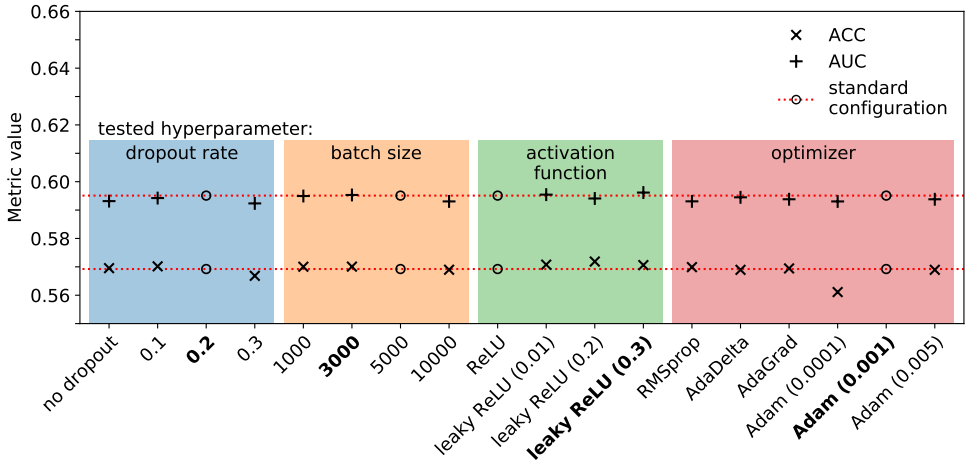


Figure 6.8: Accuracy and AUC for networks with different dropout rates, batch sizes, activation functions and optimizers on an independent validation sample of 400 000 jet images. The hyperparameters chosen for the optimized network architecture are printed in bold.

shown in Figure 6.12. The achieved accuracy is slightly worse, about 0.002, than the accuracy of the standard-architecture network. The AUC is considerably worse, by about 0.007. Hence, the additional layers with the small filters do not add relevant information and no additional layers are used in the optimized network.

6.2.6 Influence of the leading particle on CNN classification

The leading particle in a jet, as described in Section 4.1, can have a significant effect on the calorimeter signature. In the following, the effect that the leading particle of an s-jet has on the predictions of the CNNs is studied. For this purpose, a CNN is trained for the classification of jet images into five classes. The five classes are defined as

1. d-jets with any type of leading particle,
2. s-jets with a K_L^0 as leading particle (16.9% of the s-jets),
3. s-jets with a K_S^0 or a Λ baryon as leading particle (15.7% of the s-jets),
4. s-jets with a $K^\pm$ as leading particle (33.8% of the s-jets),
5. and s-jets with any other leading particle
   (charged pions, photons, protons or neutrons, 33.6% of the s-jets).

For the training of the network, 120 000 jet images are used and the network is evaluated on another 500 000 jet images. Both samples contain an equal number of jet images from every of the five classes. The network was trained during an earlier stage of the thesis with a slightly different network architecture as well as jet selection and preprocessing for the jet images in both samples. These differences should not have any significant effect on the classification. For time reasons and since it is not necessary to achieve the best possible performance for these studies, the training of the network was not redone.

The network has five nodes in the output layer with the softmax function [4] as activation function, corresponding to the five classes. The output distributions of these nodes on the validation sample, separately for every class, is shown in Figures 6.9(a), 6.10(a,c) and 6.11(a,c). The network has moderate discrimination power between s-jets with a leading K_L^0 or $K^\pm$ (long-lived kaons) and the other jets. The discrimination power between s-jets with a leading K_L^0 and s-jets with a leading $K^\pm$, as well as the discrimination power between the other classes is very low. Figure 6.9(b,c,d) shows the output distributions that correspond to the d-jet classification, exemplary for true d-jets, true s-jets with a leading $K^\pm$ and true s-jets with a leading K_S^0 or Λ baryon, each for different ECAL fractions. On average, higher output values are assigned to jets with high ECAL fractions, whereas lower output values are assigned to jets with low ECAL fractions. This indicates that the discrimination power of the network is largely, but not exclusively, based on the ECAL fraction of the jets and that s-jets are mainly identified based on the presence of long-lived kaons that carry a high energy fraction of the jet.

Figure 6.10(b,d) shows classifier output distributions each corresponding to one of the classes of s-jets with a leading long-lived kaon, for a long-lived kaon class (b) and a class of jets that does not have a leading long-lived kaon (d). Long-lived kaons usually deposit most of their energy in the HCAL. Therefore, jets that deposit a small fraction or none of their energy in the HCAL are assigned small output values close to zero and can be separated quite well from s-jets with a leading long-lived kaon.

Figure 6.11(b) shows the classifier output distribution corresponding to s-jets with a leading K_S^0 or Λ baryon for jets of this class. Roughly two thirds of these leading particles decay into detector stable hadrons that deposit their energy in the HCAL. Most of the other leading-particle decays result in at least some energy depositions in the ECAL. While jets with low or medium ECAL fraction receive output values

[4] The output of the softmax function $\sigma(y_i) = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$ of the node corresponding to class i, where j enumerates all nodes, represents the probability that a jet belongs to the class i.

between 0.1 and 0.3, the jets with high ECAL fractions receive significantly higher output values. A similar slope can be seen in Figure 6.11(d) for the *other* s-jets. This slope might correspond to the jets with a leading photon.

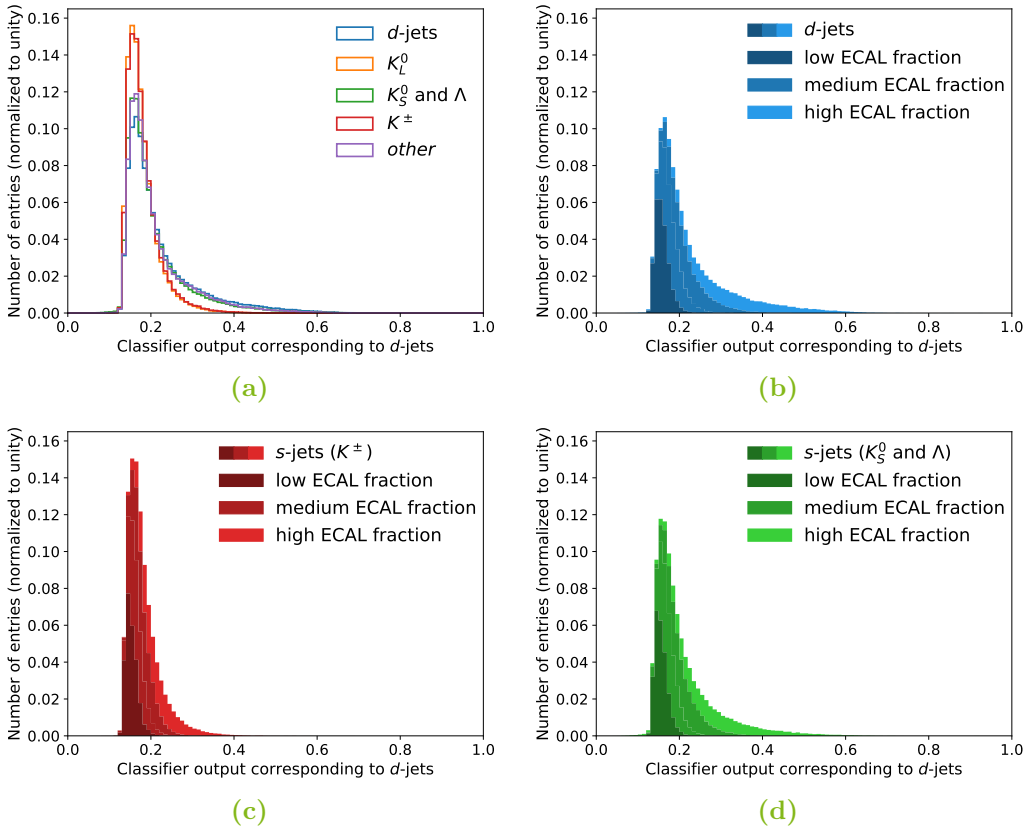


Figure 6.9: Distribution of classifier output corresponding to d-jets, (a) for all five classes and (b,c,d) respectively for d-jets, s-jets with leading $K^\pm$ and s-jets with leading K_S^0 or Λ baryon. Figures (b), (c) and (d) show the distributions for jets with different ECAL fractions, stacked on top of each other. Dark shades correspond to 25% of the jets in the respective class that have the lowest ECAL fraction, light shades to 25% of the jets with the highest ECAL fraction.

Since the discrimination power between d-jets and s-jets that do not have a leading long-lived kaon is extremely low, a network is trained for binary classification between s- and d-jets on a training sample in which only s-jets that have a leading long-lived kaon are used. This is done in order to test if this emphasis on these more relevant s-jets in the training process can increase the performance of the network on the validation sample, which also contains the other types of s-jets. The network architecture is the standard-configuration architecture that was also used

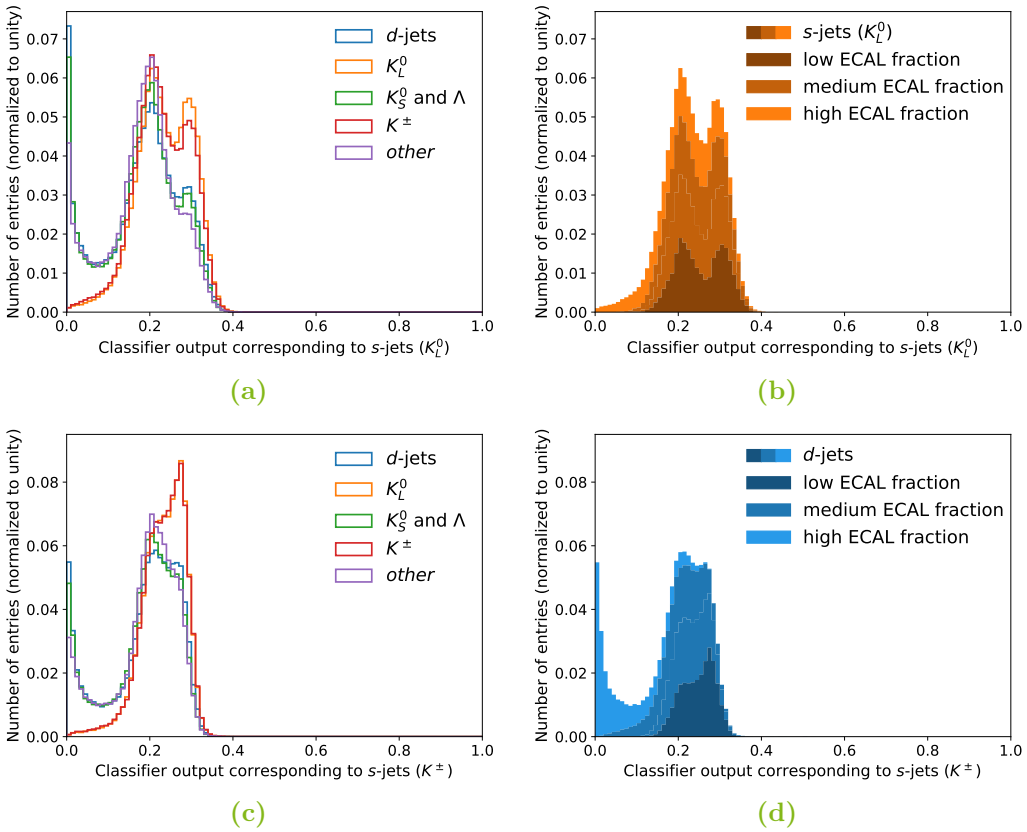


Figure 6.10: Distribution of classifier output corresponding (a,b) to s-jets with leading K_L^0 and (c,d) to s-jets with leading $K^\pm$, (a,c) for all five classes and (b,d) respectively for s-jets with leading K_L^0 and d-jets. Figures (b) and (d) show the distributions for jets with different ECAL fractions, stacked on top of each other. Dark shades correspond to 25% of the jets in the respective class that have the lowest ECAL fraction, light shades to 25% of the jets with the highest ECAL fraction.

in the previous section. For the training process, 150 000 d-jet images as well as about 100 000 jet images from s-jets with a leading $K^\pm$ and about 50 000 jet images from s-jets with a leading K_L^0 are used. The network performance is evaluated on the sample that was also used to evaluate the networks in the previous sections. The results are shown in Figure 6.12. When compared to the standard-architecture network that was trained on all types of s-jets, the performance is significantly worse with the achieved accuracy being about 0.004 lower and the AUC being about 0.008 lower. This implies that jet images from all types of s-jets can contain relevant information for the discrimination between s- and d-jets.

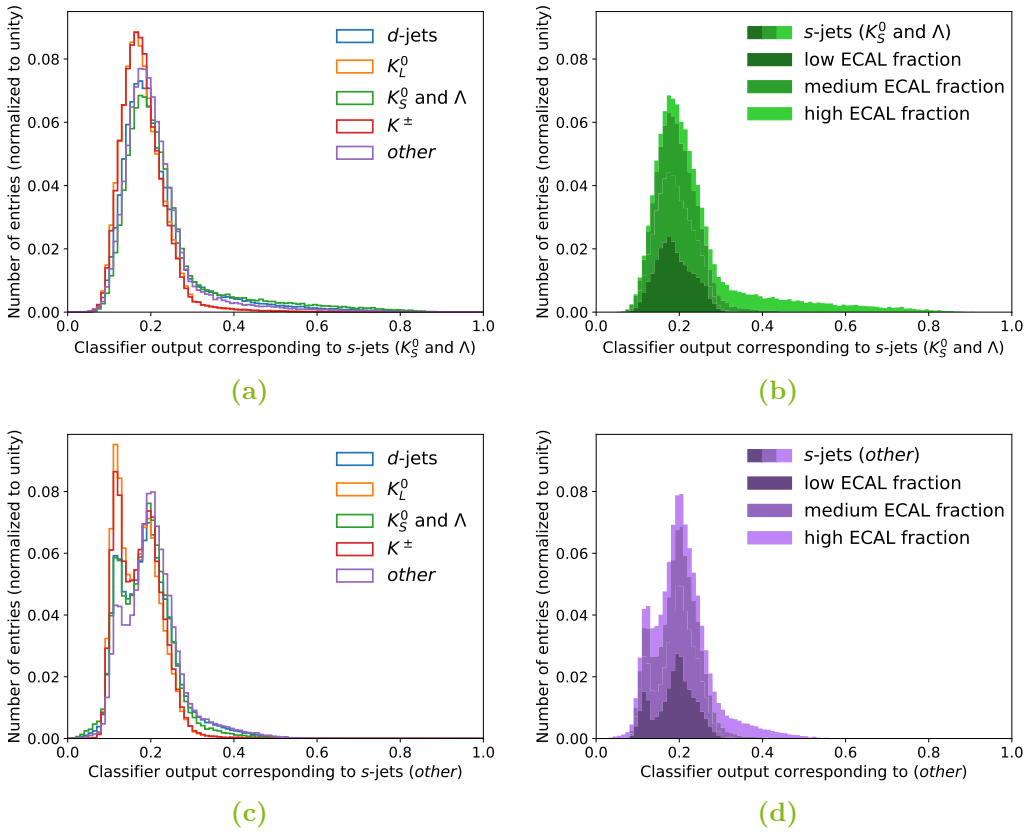


Figure 6.11: Distribution of classifier output corresponding (a,b) to *s*-jets with leading K^0_S or Λ baryon and (c,d) to *s*-jets from the class *other*, (a,c) for all five classes and (b,d) respectively for *s*-jets with leading K^0_S or Λ baryon and *s*-jets from the class *other*. Figures (b) and (d) show the distributions for jets with different ECAL fractions, stacked on top of each other. Dark shades correspond to 25% of the jets in the respective class that have the lowest ECAL fraction, light shades to 25% of the jets with the highest ECAL fraction.

6.2.7 Pretrained models

Pretrained models are neural networks with predefined architecture and weights that were already trained. Several pretrained models for image recognition were made publicly available and are integrated into Keras. The advantage of these models is, that they were trained on a large number of different images and that they have versatile architectures that can work well in different image recognition scenarios. Because of this versatility, they can be used as a basis in order to recognize general patterns in the jet images, even though they were not actually designed for jet-image

classification. The first convolutional layers of CNNs usually learn basic patterns, which can often be used for any classification task. Subsequent layers usually learn the more specific patterns. The output of the convolutional layers of a pretrained model can be used as input for fully connected layers, where only the weights of the fully connected layers are trained. Alternatively, the weights of one or more of the last convolutional layers can be retrained as well, thereby possibly adjusting the weights according to the specific patterns in the jet images.

The VGG16 [22] pretrained model, which was trained on roughly 1.2 Million images belonging to 1000 different classes, consists of thirteen convolutional layers, two fully connected layers and an output layer as well as five max-pooling layers in between the convolutional layers. The convolutional layers have between 64 and 512 filters of size 3×3. The two fully connected layers have 4096 nodes. Instead of these layers, three layers with respectively 256, 128 and 64 nodes are used, which is more suitable for the classification of s- and d-jet images, as discussed in Section 6.2.4. Only the weights in the fully connected layers are updated during training, whereas for the convolutional layers, the pretrained weights are kept. Another model is trained, where the weights of the last two convolutional layers are trained as well. The results are shown if Figure 6.12.

The network in which only the weights in the dense layers are trained achieves an accuracy and an AUC between 0.001 and 0.002 worse than with the standard-configuration network. This only slightly worse performance shows that small filters can achieve good performances as well. However, a much larger number of convolutional layers is used. The network in which the last two convolutional layers are retrained achieves a significantly worse performances with the accuracy about 0.010 and the AUC 0.014 worse than with the standard-configuration network. Hence, the pretrained weights of the last convolutional layers, which were trained with a large number of general images not related to jet images, are better than the weights trained with jet images. The training of the weights of the fully connected layers is enough to enables the network to classify the jet images.

Another pretrained model, the ResNet152V2 [23], is a model with 155 convolutional layers. A network with the ResNet152V2 layers instead of the VGG16 layers is trained and evaluated as well but achieves an accuracy of only 0.524. With the high training time of about a week required for training the fully connected layers for the VGG16 pretrained model and the even longer training times when retraining the last convolutional layers or when using the ResNet152V2 layers, further optimizations are not feasible.

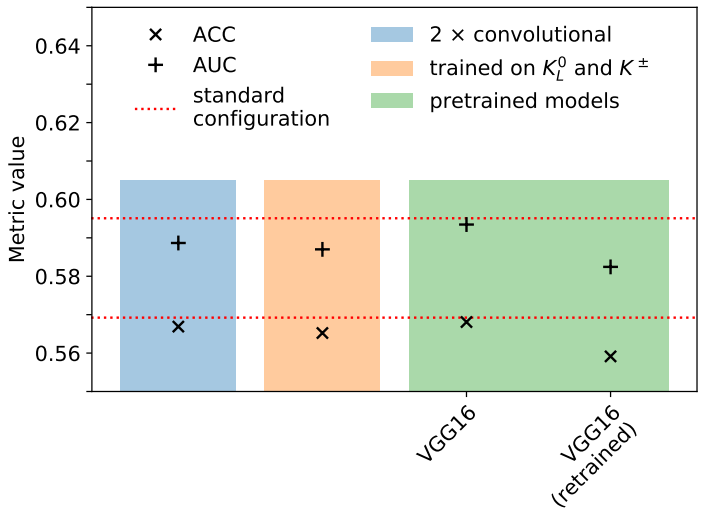


Figure 6.12: Accuracy and AUC achieved on an independent evaluation sample of 400 000 jet images by a network with an additional, parallel line of convolutional layers, a network that was only trained on s-jets with a leading K_L^0 or $K^\pm$ and d-jets as well as networks that use pretrained models.

6.2.8 Evaluation of optimized architecture

The hyperparameters for the optimized network are summarized in Table 6.1. 125 000 additional s-jet images and 125 000 additional d-jet images that were not used before are added to the previous training sample for a total of 550 000 jet images. The other samples are the same as those used in the previous sections. The network is trained over 50 epochs instead of 30 epochs, after which the model with the best accuracy on the 100 000 jet images that are used to monitor the training process is chosen.

The distribution of the classifier output, evaluated on the training sample as well as the test sample, is shown for the s-jet images and d-jet images in Figure 6.13. The normalized number of entries in the bins of the test sample divided by the normalized number of entries in the bins of the training sample is shown as well for bins with a relevant bin content. For bins with very little or no bin content, this ratio can take on extreme values due to statistical fluctuations. The difference between training and test sample for the twelve largest bins is less than five percent in every bin and less than two percent for the majority of them. Therefore, the distributions are in good agreement and show now indications of overtraining.

On the test sample, which contains 200 000 s-jet images and 200 000 d-jet images, the optimized network achieves an accuracy of 0.569 and an AUC of 0.596. When compared to the benchmark algorithm, this is a slight improvement of 0.002 in accuracy and 0.005 in AUC. The ROC curve for this network is shown in Figure 6.14. The performance is especially better at higher efficiencies. The signal and background efficiencies for different working points are shown in Table 6.2.

Table 6.1: Optimized CNN hyperparameters for the classification of s- and d-jets.

Hyperparameter	Value
Number of convolutional layers	3
Number of filters per layer	12
Filter size	$(2 \times 16 \times 16)$, (8×8), (4×4)
Stride-size	2, 1, 1
Number of max-pooling layers	3
Pooling size	(2×2)
Dropout rate for convolutional layers	0.2
Number of fully connected layers	2
Number of nodes per layer	512, 256
Activation function	leaky ReLU, slope = 0.3
Optimizer	Adam, learning rate = 0.001
Batch size	3000

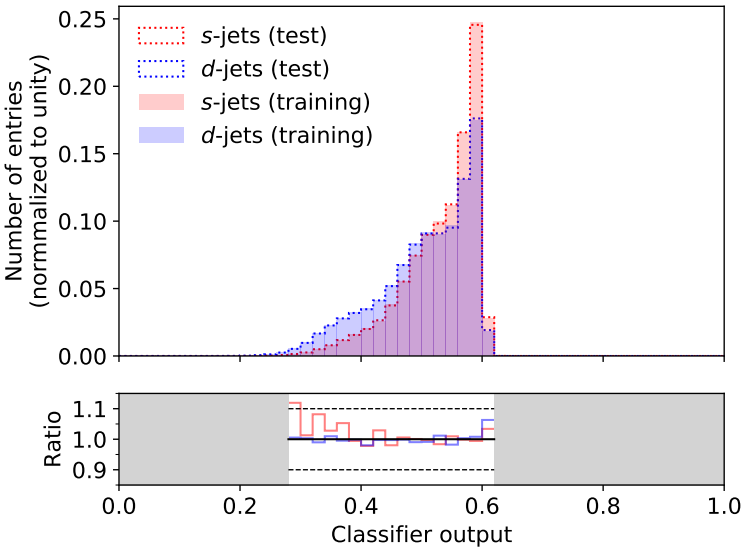


Figure 6.13: Distribution of the classifier output of the optimized CNN for *s*- and *d*-jets evaluated on the training and on the test sample. The size of the training bins divided by the size of the test bins is shown underneath the bins with relevant bin content.

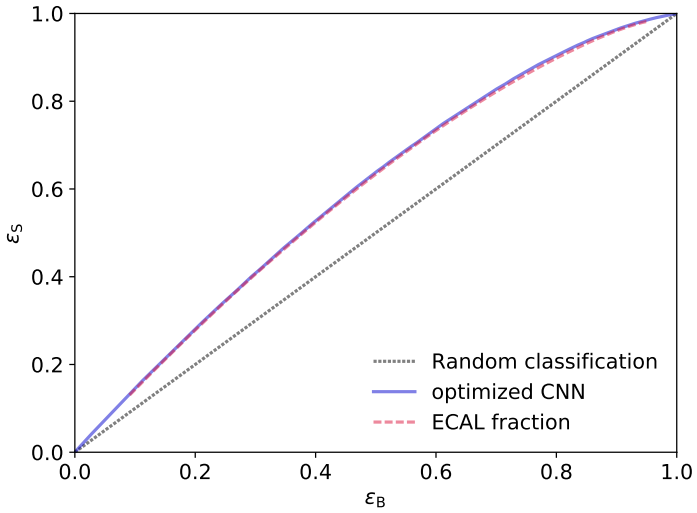


Figure 6.14: ROC curve for the optimized CNN evaluated on the test sample as well as ROC curve for the ECAL-fraction based benchmark algorithm.

6.3 Using a combination of CNNs on jet images and LSTMs on tracking information

In the following, LSTM layers that take the tracking information of the jets, described in Section 4.3, as input are added to the optimized CNN architecture. The LSTM output is concatenated with the output of the convolutional layers as well as the jet-η and jet-p_{T} and is used as input for the fully connected layers. Since the tracking information is added, fully connected layers with a larger capacity might perform better. Hence, three fully connected layers instead of two are used with 512, 256 and 128 nodes, respectively. A network with one LSTM layer with 50 units as well as a network with three LSTM layers with respectively 50, 25 and 12 units is trained and evaluated on the same samples that were used for the previous networks. The network with one LSTM layer achieves and accuracy of 0.583 and an AUC of 0.618, whereas the network with three LSTM layers achieves a higher accuracy of 0.588 and an AUC of 0.624. In the following, the better network with three LSTM layers is used.

6.3.1 Correlation between jet images and tracking information

Using a network that uses jet images as well as tracking information, as opposed to two separate networks, has the advantage that correlations between tracks and energy depositions in the calorimeters can be learned, especially from their respective positions in the η-ϕ space. In order to test if the network learns these correlations, it is trained and evaluated on modified samples, where the non-empty pixels in the jet images are randomly displaced. Therefore, the pixel positions do not properly match the track positions. If the network does not learn the correlations anyway, it should achieve similar performances with the regular samples and the samples with displaced jet images. However, a decrease in performance would indicate that the network can learn the correlations. The displacement is applied to the jet images by drawing random uniform values η_{rand} and ϕ_{rand} that satisfy the condition $\sqrt{\eta_{\mathrm{rand}}^2 + \phi_{\mathrm{rand}}^2} \leq 0.3$. These values are added to the coordinates of all energy depositions of a jet image $\eta_{\mathrm{cell},i}$ and $\phi_{\mathrm{cell},i}$ according to

$$\eta'_{\mathrm{cell},i} = \eta_{\mathrm{cell},i} + \eta_{\mathrm{rand}}, \qquad \phi'_{\mathrm{cell},i} = \phi_{\mathrm{cell},i} + \phi_{\mathrm{rand}}. \tag{6.1}$$

The ROC curves achieved by the network trained and evaluated with the displaced jet images as well as the regular jet images are shown in Figure 6.15. For d-jet efficiencies of about 20%, the s-jet efficiency is about 10% better with the regular jet images than with the displaced ones. This indicates, that discrimination power is learned from the correlation between jet images and tracking information.

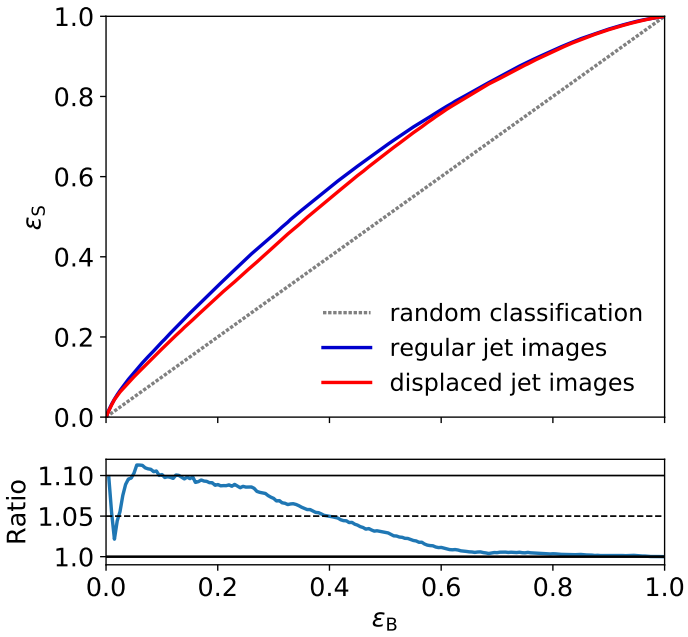


Figure 6.15: ROC curves for the combined network with three LSTM layers trained and evaluated on the training and validation sample with regular jet images as well as the training and validation sample with displaced jet images. The signal efficiency achieved with the regular jet images divided by signal efficiency achieved with the displaced jet images is also shown.

As an additional test, networks with one convolutional layer instead of three are trained and evaluated on the regular samples and the samples with displaced jet images. As mentioned in Section 5.2, each subsequent convolutional layers tends to learn more abstract features than the previous layer. Therefore, using just one convolutional layer might increase the amount of discrimination power that the network learns from the correlation, since the output of the convolutional layers is not as abstract.

The ROC curves achieved by networks with filter sizes of 4, 8, 16 or 32 are shown in Figure 6.17. For small efficiencies, the ratio between the ROC curves can take on extreme values due to statistical fluctuations. Differences between the ROC curves achieved with the regular jet images and the displaced jet images are small for the networks with larger filters. For the network with a filter size of 16, at the maximum of the difference, the s-jet efficiency with the regular jet images is about 3% better than with the displaced jet images. For the network with a filter size of 32, the

ROC curves are very similar. These filters cover an area of about 0.56×0.56 rad in the η-ϕ space. The smaller performance differences with larger filter sizes might be explained with the displacement being less relevant for larger filters. For the network with a filter size of 4, the s-jet efficiency achieved with the regular jet images is at maximum about 5% better than with the displaced jet images. For the network with with a filter size of 8, the ROC-curve differences are similar to the differences for the network with three convolutional layers, except at higher efficiencies, where the difference with one convolutional layer is slightly larger. In summary, these studies indicate, that the combined network learns discrimination power from the correlations between jet images and tracks. Also, there are no indications that networks with fewer convolutional layers learn significantly more correlations between the jet images and the tracking information.

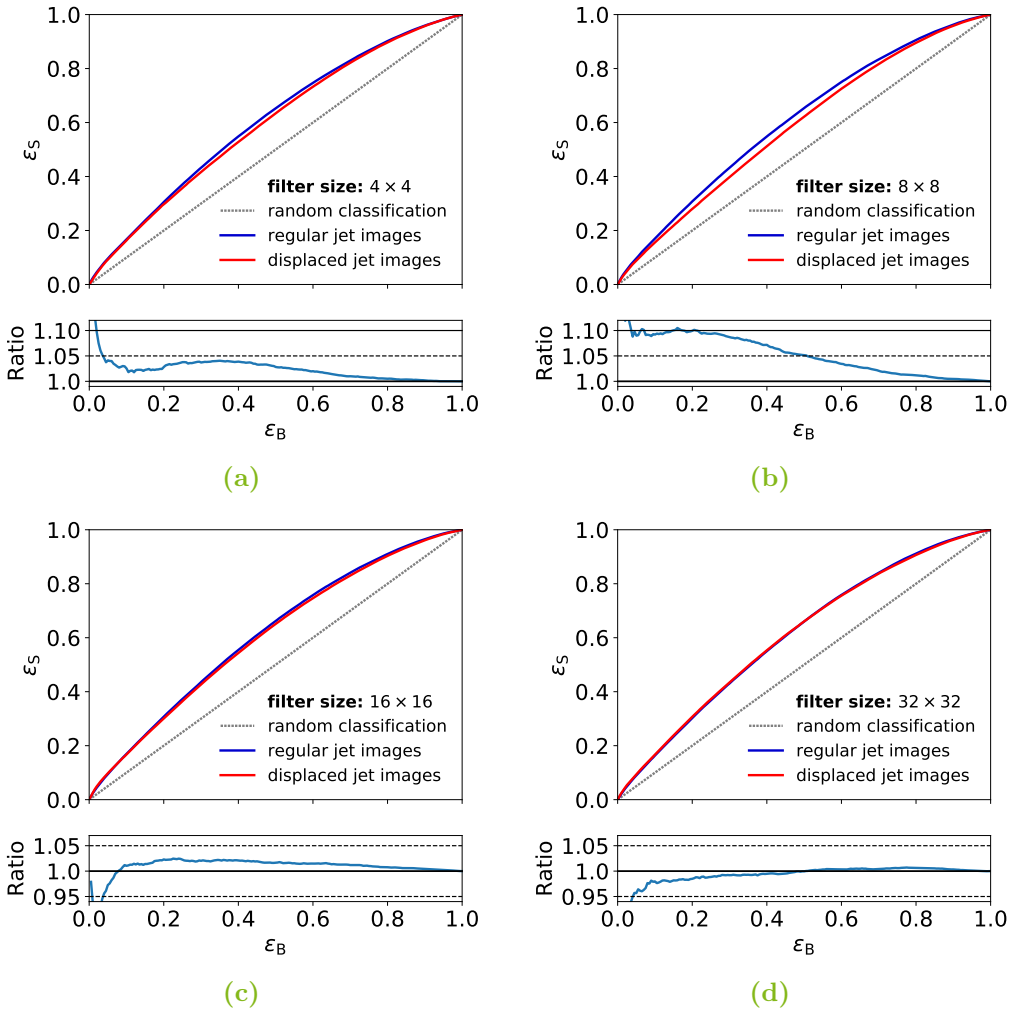


Figure 6.16: ROC curves for combined networks with one convolutional layer and different filter sizes, each trained and evaluated on the training and validation sample with regular jet images as well as the training and validation sample with displaced jet images. The signal efficiency achieved with the regular jet images divided by signal efficiency achieved with the displaced jet images is also shown.

6.3.2 Evaluation of combined neural network

Analogously to the optimized CNN in Section 6.2.8, the combined network with three LSTM layers is trained over 50 epochs on the enlarged training sample with 550 000 jets. The output distribution of the network when evaluated on the training

sample and the test sample, separately for s- and d-jets, is shown in Figure 6.17. The distributions are in good agreement and therefore show no indication of overtraining. On the test sample, the combined network achieves an accuracy of 0.588 and an AUC of 0.624. The ROC curve is shown in Figure 6.18 and signal and background efficiencies for different working points are shown in Table 6.2. Additionally, the filter weights of the convolutional layers after the training process are visualized in Appendix A.2.

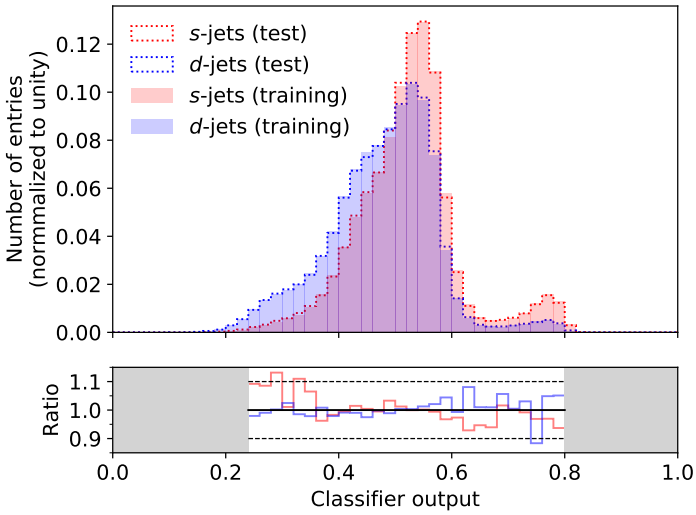


Figure 6.17: Distribution of the classifier output of the combined neural network for s- and d-jets evaluated on the training and on the test sample. The size of the training bins divided by the size of the test bins is shown underneath the bins with relevant bin content.

Table 6.2: Background efficiencies achieved on the test sample by the optimized CNN, the combined neural network and the benchmark algorithm for given signal efficiencies.

ϵ_S	ϵ_B (CNN)	ϵ_B (CNN + LSTM)	ϵ_B (benchmark)
15%	10.3%	7.5%	10.5%
30%	21.5%	17.9%	21.6%
50%	37.7%	33.8%	37.9%
70%	56.2%	52.6%	56.5%
90%	79.4%	77.7%	80.3%

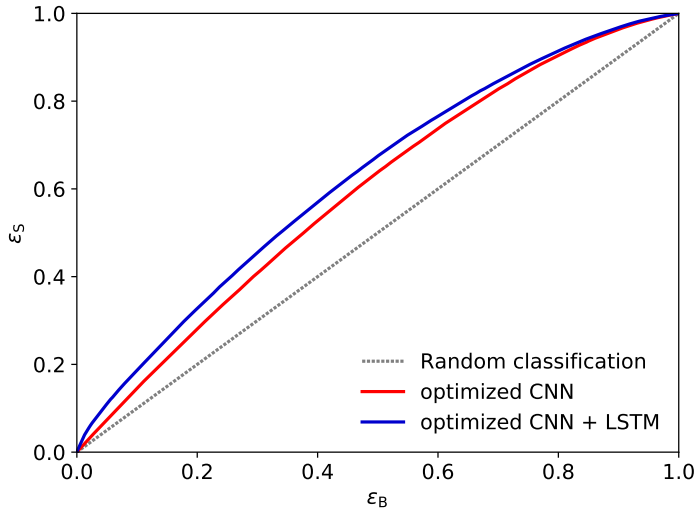
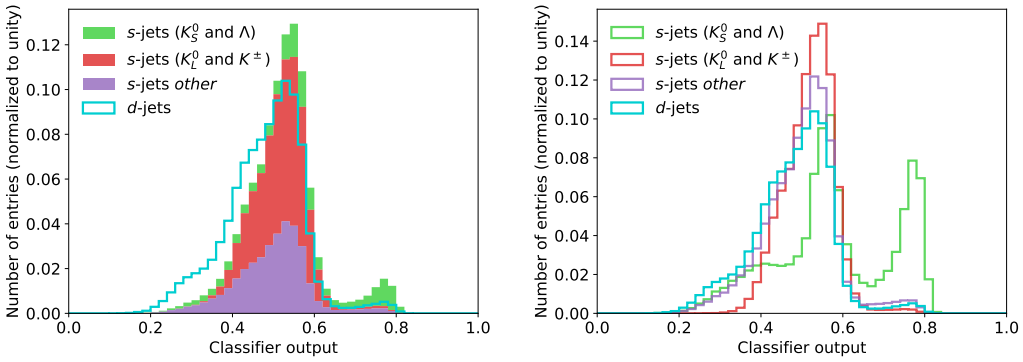


Figure 6.18: ROC curves for the combined neural network and the optimized CNN evaluated on the test sample.

Figure 6.19 shows the output distributions evaluated on the test sample separately for d-jets, s-jets with a leading long-lived kaon, s-jets with a leading K_S^0 or Λ baryon and other s-jets. On the left side, the distributions of the different types of s-jets are stacked on top of each other, while the distributions are shown separately and each normalized to unity on the right side. In contrast to the output of the optimized CNN presented in Section 6.2.8, the output of the combined network has a two-peak structure. All types of jets significantly contribute to the first peak at output values of around 0.5. On average, higher output values are assigned to s-jets with a leading long-lived kaon. The second peak at output values of around 0.8 is much smaller and mostly consists of s-jets with a leading K_S^0 or Λ baryon. This indicates that the network learns discrimination power from hadron decays where both tracks of the decay products originate from a SV. While the combined network has discrimination power between d-jets and s-jets with leading strange hadrons, the output distribution for s-jets with other leading particles is very similar to the output distribution for d-jets.



(a) (b)

Figure 6.19: Distribution of the classifier output of the combined neural network evaluated on the test sample for s-jets with a leading K_L^0 or leading $K^\pm$, s-jets with a leading K_S^0 or Λ baryon, s-jets with other leading particles and d-jets. In figure (a), the distributions for the different types of s-jets are stacked on top of each other. Figure (b) shows all distributions separately, each normalized to unity.

7 Conclusions

In this thesis, convolutional neural networks (CNNs) were used for the discrimination between Monte-Carlo simulated jets initiated by strange quarks (s-jets) and jets initiated by down quarks (d-jets). The training was performed on jet images built from energy depositions in calorimeters. For the detector simulation, a multi-purpose detector layout inspired by the CMS detector was used. The network architecture was optimized with regard to the achieved accuracy as well as the area under the ROC curve. It was found that network architectures with small numbers of filters and large filter sizes achieve good performances. For the optimized CNN, three convolutional layers with twelve filters each as well as two fully connected layers with respectively 512 and 256 nodes were used.

Investigations of the network output indicate that the discrimination with CNNs primarily relies on the higher momentum weighted fraction of long-lived kaons in s-jets. These usually deposit most of their energy in the hadronic calorimeters (HCAL), whereas neutral pions, which are more common in d-jets, usually deposit most of their energy in the electromagnetic calorimeters (ECAL).

As a benchmark algorithm, a simple cut on the fraction of deposited energy in the ECAL was used. For s-jet efficiencies of 30% and 70%, the optimized CNN achieves d-jet efficiencies of 21.5% and 56.2%, which is only slightly better than the d-jet efficiencies of respectively 21.6% and 56.5% achieved by the benchmark algorithm. In an additional step, the classification based on jet images was combined with previous work [6], where long short-term memory (LSTM) recurrent neural networks were used in order to discriminate between s- and d-jets based on tracking information. A combined network was trained that uses jet images as well as tracking information. In agreement with previous studies, it was shown that in contrast to the CNN, this network learns a significant amount of additional discrimination power from s-jets that contain short-lived strange hadrons which mostly decay inside the ID. Further studies indicate that there are correlations between energy depositions in the jet images and the tracking information that can be exploited by the network. With the combined network, d-jet efficiencies of 17.9% and 52.6% for s-jet efficiencies of respectively 30% and 70% are achieved.

The presented algorithm shows that neural networks that use calorimeter and tracking information from multi-purpose detectors can be used for for the identification of s-jets, complementing already existing algorithms for jet tagging. This could give access to applications such as the measurement of the $t \to W^+ s$ and $H \to s\bar{s}$ decays and searches for physics beyond the Standard Model of particle physics, where strange-tagging methods help to identify processes with strange quarks in the final states.

A Appendix

A.1 Average d-jet images

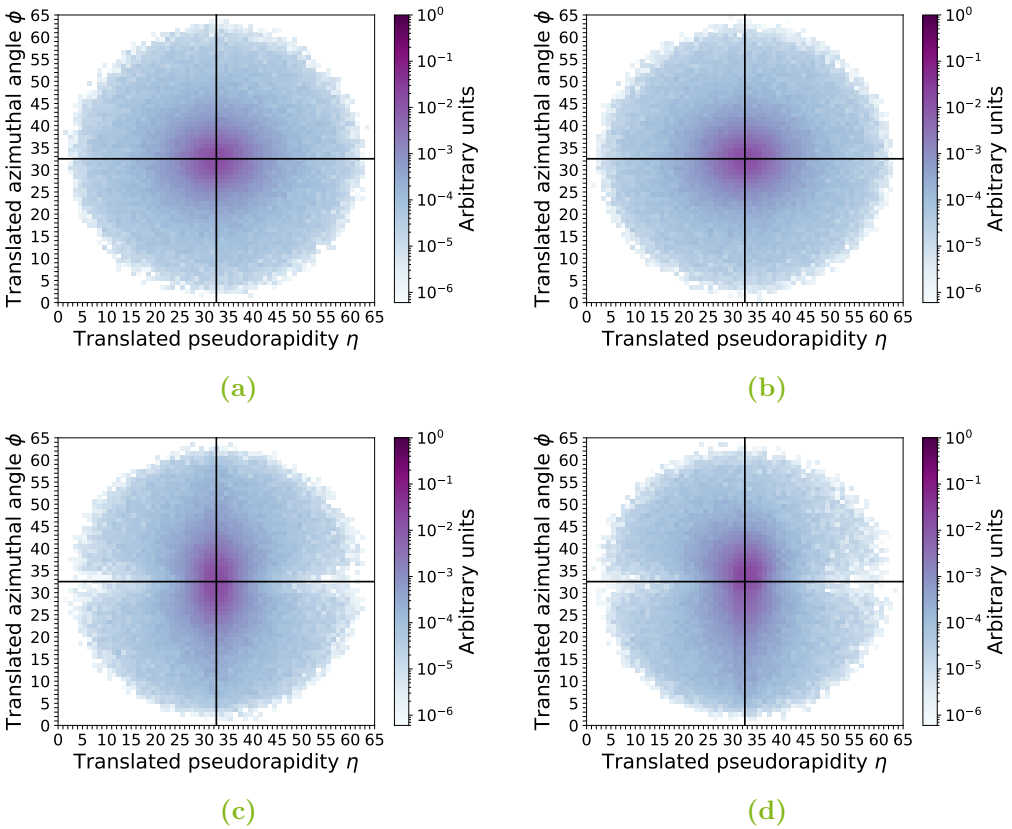


Figure A.1: Average of 40 000 ECAL d-jet images at different stages of the preprocessing. In figure (a) with no preprocessing, in figure (b) centered on the center of intensity, in figure (c) rotated to have a vertical principal axis and in figure (d) flipped to have the highest sum of intensity in the upper right quadrant.

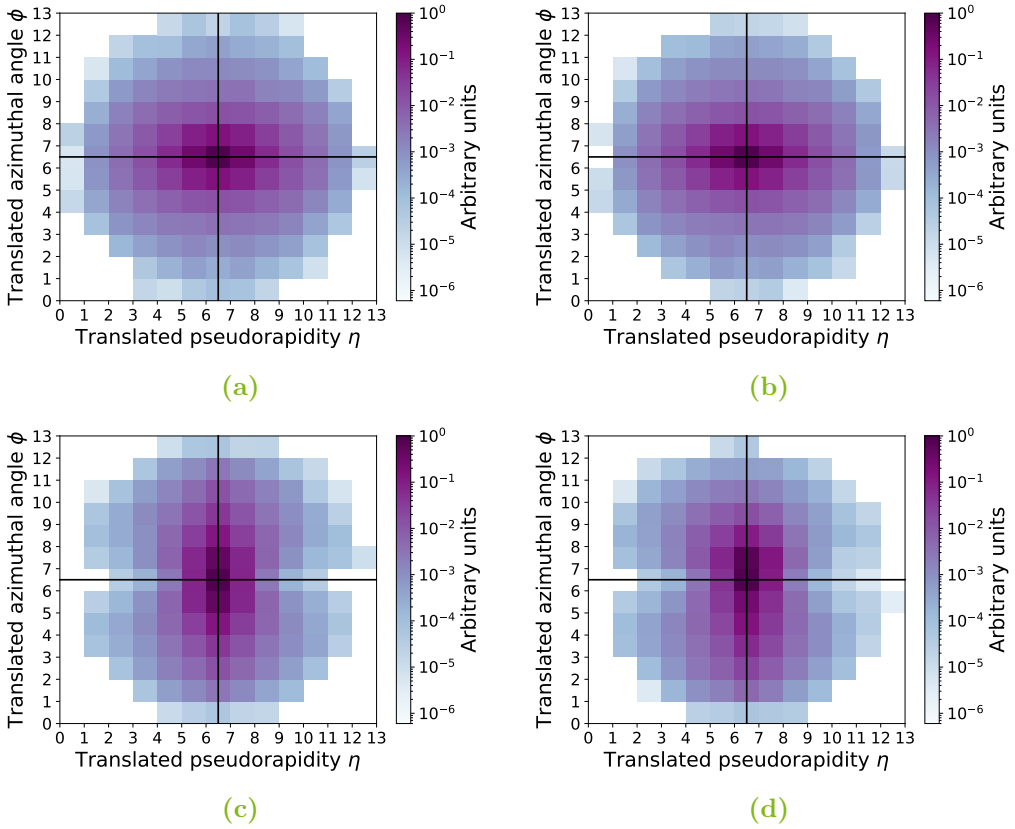


Figure A.2: Average of 40 000 HCAL d-jet images at different stages of the preprocessing. In figure (a) with no preprocessing, in figure (b) centered on the center of intensity, in figure (c) rotated to have a vertical principal axis and in figure (d) flipped to have the highest sum of intensity in the upper right quadrant.

A.2 Visualization of convolutional filters

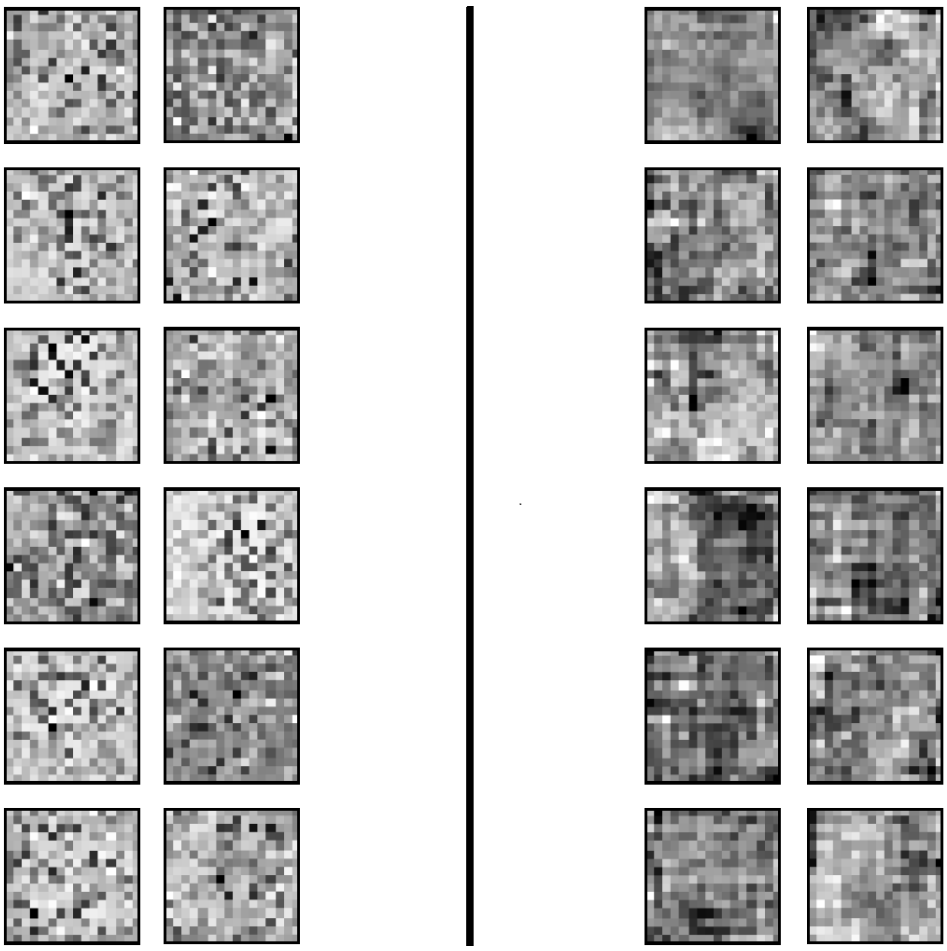


Figure A.3: Two-dimensional visualization of the filter weights of the first convolutional layer in the combined neural network. The filter weights that correspond to the ECAL are shown on the left, the filter weights that correspond to the HCAL on the right.

Figure A.4: Visualization of the filter weights of the second convolutional layer in the combined neural network.

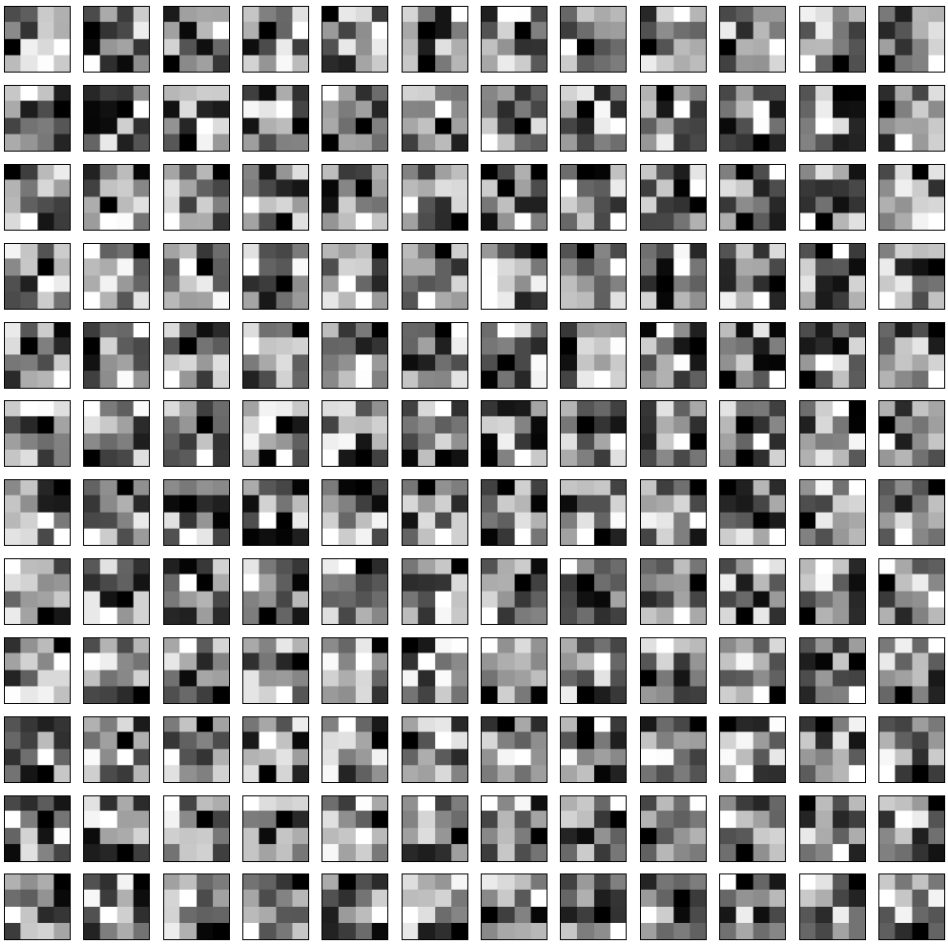


Figure A.5: Visualization of the filter weights of the third convolutional layer in the combined neural network.

Bibliography

[1] L. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008) S08001.

[2] ATLAS Collaboration,
The ATLAS Experiment at the CERN Large Hadron Collider,
JINST **3** (2008) S08003.

[3] CMS Collaboration, *The CMS Experiment at the CERN LHC*,
JINST **3** (2008) S08004.

[4] A. Ali, F. Barreiro, and T. Lagouri,
Prospects of measuring the CKM matrix element $|V_{ts}|$ at the LHC,
Phys. Lett. B **693** (2010) 44, arXiv: 1005.4647.

[5] J. Duarte-Campderros, G. Perez, M. Schlaffer, and A. Soffer, *Probing the Higgs–strange-quark coupling at e^+e^- colliders using light-jet flavor tagging*,
Phys. Rev. D **101** (2020) 115005, arXiv: 1811.09636.

[6] J. Erdmann, *A tagger for strange jets based on tracking information using long short-term memory*, JINST **15** (2020) P01021, arXiv: 1907.07505.

[7] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, and T. Stelzer,
MadGraph 5 : Going Beyond, JHEP **06** (2011) 128, arXiv: 1106.0522.

[8] R. D. Ball et al., *Parton distributions with LHC data*,
Nucl. Phys. B **867** (2013) 244, arXiv: 1207.1303.

[9] T. Sjöstrand et al., *An introduction to PYTHIA 8.2*,
Comput. Phys. Commun. **191** (2015) 159, arXiv: 1410.3012.

[10] J. de Favereau et al., *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, JHEP **02** (2014) 057, arXiv: 1307.6346.

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

[12] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman,
Jet-Images – Deep Learning Edition, JHEP **07** (2016) 069,
arXiv: 1511.05190.

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

[14] I. Shafkat, *Intuitively Understanding Convolutions for Deep Learning*, https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1, 2018.

[15] F. Chollet et al., *Keras*, https://keras.io, 2015.

[16] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous distributed systems.*, https://www.tensorflow.org, 2015.

[17] D. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, (2014), arXiv: 1412.6980.

[18] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, J. Mach. Learn. Res. **12** (2011) 2825, arXiv: 1201.0490.

[19] G. Hinton, N. Srivastava, and K. Swersky, *Lecture 6e–rmsprop: Divide the gradient by a running average of its recent magnitude*, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012.

[20] M. D. Zeiler, *ADADELTA: An Adaptive Learning Rate Method*, CoRR (2012), arXiv: 1212.5701.

[21] J. Duchi, E. Hazan, and Y. Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, J. Mach. Learn. Res. **12** (2011) 2121.

[22] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, (2014), arXiv: 1409.1556.

[23] K. He, X. Zhang, S. Ren, and J. Sun, *Identity Mappings in Deep Residual Networks*, CoRR (2016), arXiv: 1603.05027.

Acknowledgements

I am very grateful to Prof. Kröninger for giving me the opportunity to work in his group, which was a superb experience through and through. I would also like to thank Priv.-Doz. Johannes Erdmann for being a great supervisor and Prof. Johannes Albrecht for being the second referee of this thesis. I was especially impressed by Dr. Olaf Nackenhorst's will to improve my unscientific scribbling, which helped a lot, thanks! Special thanks also to all the others that read parts of my thesis, Björn, Cornelius, Felix, Flo, Kevin and Sonja. I am also thankful to Andrea and Markus (while he was still with us) for being the heart and soul of E4 and to my lovely 137-office colleagues, especially to Froch for making sure that the administrative side of my thesis ran harmoniously. I would also like to express my gratitude to anyone I might have forgotten, my aunt in Wuppertal and Annu, for trying to explain English comma rules to me.