Department of Physics and Astronomy
Heidelberg University

Bachelor thesis in Physics
submitted by

Christopher Lüken-Winkels

born in Trier (Germany)

2020

Applying Capsule Networks on boosted Higgs-tagging

This Bachelor thesis has been carried out by

Christopher Lüken-Winkels

at the

Institute for Theoretical Physics

at

Heidelberg University

under the supervision of

Prof. Tilman Plehn

Applying Capsule Networks on boosted Higgs-tagging

Christopher Lüken-Winkels

Abstract

In this thesis we use Capsule Networks to distinguish boosted Higgs-bosons decaying into bottom pairs against QCD-background in calorimeter images. We show that these networks are able to reach the performance of, so far established, Convolutional Neural Networks. By additionally representing their outputs as vectors, Capsule Networks offer a better representation of the input in their predictions. We present an analysis of the latent space the vectors are embedded in, to show their ability to extract event and subjet information and use it to enhance the Capsule Networks to even outperform the Convolutional Neural Networks.

Zusammenfassung

In dieser Arbeit verwenden wir Kapselnetzwerke, um Zerfälle schneller Higgs-Bosonen in bottom-Paare von QCD-Untergrund in Kalorimeter Bildern zu unterscheiden. Wir zeigen, dass diese Netzwerke vergleichbare Ergebinsse zu bisher etablierten Convolutional Neural Networks erzielen. Da sie ihr Klassifikationsergebnis zusätzlich als Vektor darstellen, ist daraus mehr Information über die klassifizierten Daten zu gewinnen. Wir stellen eine Analyse des Raumes, indem diese Vektoren eingebettet sind, vor, um ihr Potetial Event- und Subjetinformationen gleichzeitig darzustellen zu zeigen und mit deren Hilfe eine Verbesserung der Kapselnetzwerke möglich ist, womit die Convoltional Neural Networks übertroffen werden können.

Contents

Chapter 1

Introduction

As in many fields of research, neural networks are also applied in analysing events, measured at the Large Hadron Collider (LHC). Their capability to separate special decays form background events in calorimeter images is already demonstrated [1–3]. An application of Convolutional Neural Networks (CNNs) on classifying $H \longrightarrow b\bar{b}$ against QCD background is shown in [3], producing a higher detection significance in contrast to standard two-prong-tagging methods. The proposed architecture consists of two streams: one analyzing the double-b-tagged jet image and the other one analyzing the global event structure. Since most standard Convolutional Neural Networks lack the possibility of propagating spacial information of the input through the full network [4], the classification is less likely to extract useful information from the global event properties. As the performance studies of [3] showed, the networks accuracy is mostly dependent of the double b-tagged jet image. In Capsule Networks (CapsNets) the Dense part of the CNN is replaced by Capsule Layers, which use the geometrical information by construction which enables them to classify full calorimeter images in a single stream network [1]. First, we compare the performance of the two approaches. Our second point of interest are the relations between the input and the output capsules and whether the events are organized in a physically meaningful way in the output latent space. Taggers that are not based on machine learning [5] mostly focus on the subjet structure and thus loose the possibility to use the full event level information. Capsule Networks offer a great opportunity to combine both event and subjet information in their classification outcome.

Chapter 2

Neural Networks

In the recent years, neural networks became a powerful tool for data-analysis by systematically finding and exploiting general patterns in large sets of data. In this chapter we introduce the tools used in our application.

2.1 Layers

A network consists of a collection of neurons. Each neuron returns an output given its input and parameters, with the parameters being separated into weights and biases. Additionally, an activation function is applied that introduces a non-linearity to the neuron. This is important to describe complex dependencies using a set of neurons. To do so they are organized in layers, with the input propagating form layer to layer to calculate an output. Fully connected layers are the most basic of such layer structures.

2.1.1 Fully Connected Dense Layers

Looking at two layers of this type, with one providing the input for the other, each input neuron x_i is connected with each output neuron y_j. Connected in this manner means that there is a weight w_{ji} which determines the influence of input neuron i to output neuron j. The output of such a layer can be described as the vector $\boldsymbol{y}$:

$$\mathbf{y} = \phi\left(W\boldsymbol{x} + \boldsymbol{b}\right) , \tag{2.1}$$

where the components of $\boldsymbol{y}$, $\boldsymbol{x}$ and $\boldsymbol{b}$ are y_j, x_i and b_j respectively and W is the matrix consisting of the entries w_{ji}. The activation function is applied component-wise. An example is presented in Figure 2.1, with black and red lines representing weights and biases and nodes representing the activation of each neuron. Networks constructed from an input layer, one hidden layer and an output layer of this type are proven to be able to approximate any Borel measurable function from one finite dimensional space to another, if a sufficient amount of parameters are available and a suitable activation function is used [6]. Yet to retrieve a more efficient approximation, other layers such as Convolutional and Capsule Layers are required.

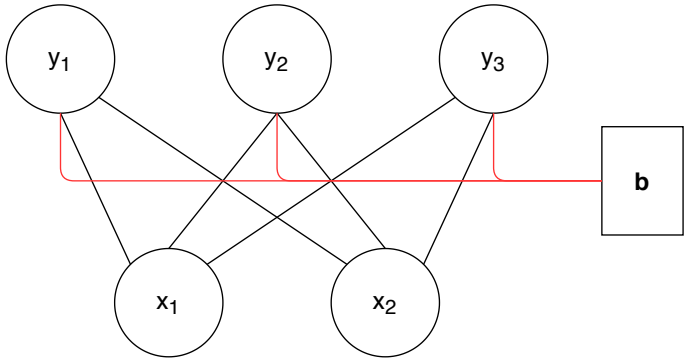


Figure 2.1 Example for fully connected Dense Layer with inputs x_i, outputs y_i and bias-vector $\boldsymbol{b}$.

2.1.2 Convolutional Layers

Convolutional Layers consist of a set of kernels, which contain weights that are applied to neighboring input nodes. Thereby, each kernel constructs a so called channel from the input and uses the same weights for all input regions. The scheme of the two dimensional convolutional operation is shown in Figure 2.2.

This yields the possibility to exploit correlations in the input, which is important in, for example, pattern recognition and image analysis. Additionally, the number of weights is drastically reduced in contrast to a fully connected layer. For comparison: A fully connected version of the example shown in Figure 2.2 with 12 input and 6 output neurons, would consist of 72 weights while it takes just 4 parameters using the convolution. To construct Convolutional Layers that are suitable for specific tasks, there are some parameters that determine the properties of the networks operation: kernel size, stride and padding.

The kernel size describes shape of the kernel, used for the convolutional operation. If, for example, a rather fine structure in the input is to be observed the kernel should be rather small and vice versa.

To preserve spacial dimensions in convolutions with bigger kernel sizes and not reduce it as in Figure 2.2 or to fulfill boundary conditions, padding is used. By attaching additional nodes to the edge of the input, padding increases its size. The activations of the newly constructed nodes are typically set to zero, but to include, for example, periodic boundary conditions the, values of the other end of the input can be used.

Stride is the property that determines the step width in which the kernel moves along the input data for the different output nodes of one channel. If it is set to a number bigger than one along any dimension the size of the channels is reduced, which is often combined

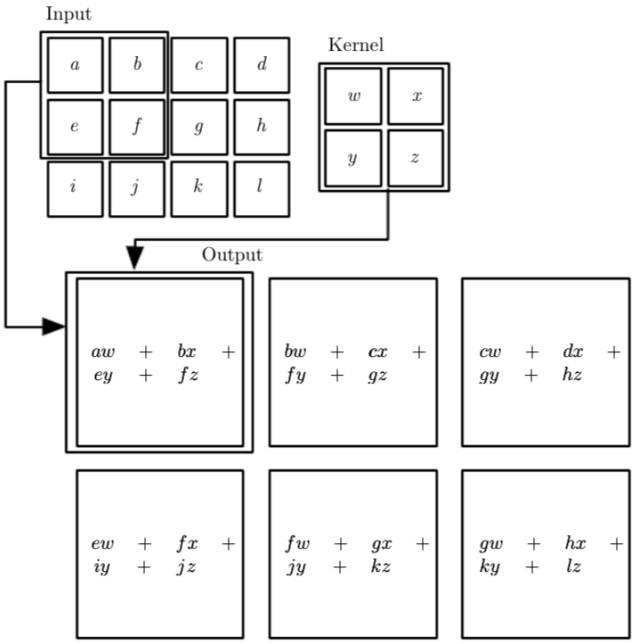


Figure 2.2 Scheme of a two dimensional convolutional operation. Figure taken from [7, p.330].

with an increase of channels to keep the number of neurons.

2.1.3 Pooling Layers

A Pooling Layer is a common substitute for stride being set to a greater value than one, to reduce the size along the spacial dimension. It takes a given number of input neurons in form of a kernel and applies a simple operation without parameters to calculate an output. In most cases max pooling or average pooling is used which return either the maximum or the average of all neurons in the kernel region.

2.2 Activation Functions

Without activation functions, neural networks would only be able to model linear functions. The non-linearity introduced by the activation functions are an important part of the capability of networks to model such a wide range of functions.
A rectified linear unit (ReLU) is often used for this purpose. It is defined by:

$$\mathrm{ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{2.2}$$

If the outcome of a network is desired to be a probability distribution, as in most classification problems, a common choice for the activation is the softmax function:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}\,, \tag{2.3}$$

where x_i is a component of the N-dimensional input vector $\boldsymbol{x}$.

The activation function referred to as squashing, which is most commonly used in CapsNets [8] (see Section 2.5) is defined for vectors, which represent neurons for this network type. The length of the vector is pushed towards zero for short and slightly lower than one for long vectors:

$$\boldsymbol{s}(\boldsymbol{x}) = \frac{\|\boldsymbol{x}\|^2}{1+\|\boldsymbol{x}\|^2}\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\,, \tag{2.4}$$

where $\boldsymbol{s}$ is the output vector of the activation function.
Besides these three, many more activation functions are used but are not needed our studies. A collection can be found in [9].

2.3 Losses

To determine the quality of the outcome of a neural network, loss functions are used. They can be designed to measure its performance with respect to any desired property. If the information is given, a direct comparison between desired outcome and network prediction is possible. For classification purposes where the outcome of the network is a probability distribution of the input corresponding to a set of classes, cross-entropy loss is the most common choice. It is defined by:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i\,, \tag{2.5}$$

where $\boldsymbol{y}$ is the probability distribution the network is to reconstruct, referred to as the target vector. $\hat{\boldsymbol{y}}$ is the networks prediction and C the dimension of the vectors and thereby number of classes.

For Capsule Networks this is replaced by the margin loss. It is applied separately to a set of vectors, $\boldsymbol{y}^{(i)}$ where each vectors length represents the membership of an input to a certain class. The target is denoted by $\hat{y}^{(i)}$. It has value of one if class i is present and zero otherwise. The loss for one of the vectors is then defined by:

$$L\left(\hat{y}^{(i)}, \boldsymbol{y}^{(i)}\right) = \hat{y}^{(i)} \max\left(0, m_+ - \left\|\boldsymbol{y}^{(i)}\right\|\right)^2 + \lambda\left(1-\hat{y}^{(i)}\right)\max\left(0, \left\|\boldsymbol{y}^{(i)}\right\| - m_-\right)^2\,, \tag{2.6}$$

with $m_{+/-}$ being the target length for the vectors if the input belongs/doesn't belong to its class and λ weighting the influence of instance of different classes. The complete loss is then derived by summing over all classes. Analogously to [1] we choose $m_{+/-}$ to be 0.9/ 0.1, and λ as 1.

To measure the distance between a predicted probability distribution and a target distribution the Kullback–Leibler Divergence (KLD) can be used [10]. For discrete distributions it is defined by:

$$\mathrm{D_{KL}}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) , \tag{2.7}$$

where P and Q are the distributions, defined on the domain $\mathcal{X}$. When used on network outputs P represents the target and Q the prediction.

2.4 Training and Optimizers

Having the loss for the desired property at hand the goal is to minimize this quantity, which means to choose the optimal parameters. This task is automatized by calculating the gradient of the loss for a given input and target w.r.t. the networks parameters (the weights and biases) and feeding this to an optimizer which updates the weights. This is referred to as training. Ideally, the loss function is minimized and the network learns the desired connection between in- and output. As it is highly optimized and commonly used we use the optimizer Adam [11]. If the dataset the network is training on is too small or the network trains for too long it can memorize this set rather than generalize the connection of input and output, which is called overfitting. To prevent this, regularization methods are applied, for example Dropout [12]. When using dropout, the activations of a randomly chosen subset of the network is set to zero and are not optimized in one training iteration. This way subsets of the network can be trained and if the complete network is tested without dropout the outcome is comparable to the average of all these subsets.

2.5 Capsule Networks

Capsule Networks were introduced in [8] and are based on the idea that the networks neurons are grouped into vectors also referred to as capsules. Each capsule represents a certain abstract feature, with its length denoting the probability of this feature being present in the input and the entries representing its properties. Layers of this type consist of a set of these vectors. The implementations used for this thesis are based on [1] and [13].

2.5.1 Primarycaps

A main advantage of the Capsule Layer is the retention of the positional information of the represented feature. As we use the Capsule Layers as subsequent layer of convolutional ones a special reshaping routine is needed, which is done by the Primarycap layer. The positional information of the input in a convolutional network is contained in the pixel positions of each channel. To construct a capsule, neurons along the channel axis of the Convolutional Layer are used. If the number of channels is larger than the desired capsule dimension, multiple capsules per channel are constructed. If a Convolutional Layer of shape X×Y×C is the input to a Primarycap with desired capsules of dimension D, the resulting number of capsules N is given by $\mathrm{X}\times\mathrm{Y}\times\frac{\mathrm{C}}{\mathrm{D}}$, where the capsule dimension has to be chosen as a denominator of the channel number.

2.5.2 Capsule Layers

To calculate the output vectors from input capsules a new layer is defined. Each input vector has a contribution to every output capsule, defined by different weights for each component. For N D-dimensional input capsules $\boldsymbol{x}^{(j)}$ and N′ D′-dimensional output capsules $\boldsymbol{y}^{(j')}$, N×N′ new vectors $\boldsymbol{u}^{(j,j')}$ are calculated using the weights $W^{(j,j')}$. The components of the intermediate vectors $\boldsymbol{u}^{(j,j')}$ are calculated by:

$$u_{i'}^{(j,j')} = \sum_{i=1}^{D} w_{ii'}^{(j,j')} x_i^{(j)} \,, \tag{2.8}$$

with $u_{i'}^{(j,j')}$, $x_i^{(j)}$ and $w_{ii'}^{(j,j')}$ denoting the components of $\boldsymbol{u}^{(j,j')}$, the input capsule $\boldsymbol{x}^{(j)}$ and the D×D′ matrix $W^{(j,j')}$ respectively. As in [1] the arrangement of the indices has no meaning besides offering clarity in the notation.

These vectors are combined using another set of weights $b^{(j,j')}$ which are normalized using the softmax function (2.3) to obtain the weights $c^{(j,j')}$:

$$\boldsymbol{v}^{(j')} = \sum_{j=1}^{N} c^{(j,j')} \boldsymbol{u}^{(j,j')} \,. \tag{2.9}$$

By using the squashing function (2.4) on these vectors the output capsules $\boldsymbol{y}^{(j')}$ are constructed. The layer calculations for $N = 2$, $D = 3$, $N' = 3$ and $D' = 2$ is shown in Figure 2.3.

The weights $b^{(j,j')}$ are calculated by so called routing which iteratively collimates the output vector with one of its inputs. It replaces the activation function to use the vector properties. The weights $b^{(j,j')}$ are modified according to the scalar products of the corresponding vector $\boldsymbol{u}^{(j,j')}$ with the output vector $\boldsymbol{y}^{(j')}$:

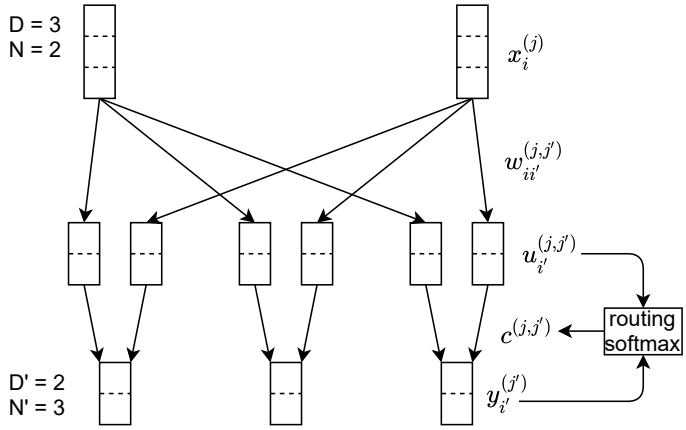


Figure 2.3 Capsule Layer procedure for a layer with two three-dimensional to a layer with three two-dimensional vectors.

$$b^{(j,j')} \leftarrow b^{(j,j')} + \boldsymbol{u}^{(j,j')}\boldsymbol{y}^{(j')} \,. \tag{2.10}$$

Lastly, the softmax function is applied on these weights and the new set of squashed vectors $\boldsymbol{y}^{(j')}$ is calculated. This redefinition can be done multiple times. Iterating this process three times improved the performance the best in former analysis [8] which is why it is also used in our network architectures.

Chapter 3

Particle and Jet Physics

3.1 The Standard Model

The standard model (SM) is a theory describing a set of elementary particles together with their interactions. The models predictions are in good agreement with experiments on a large energy scale and thus presents us with the best tested theory in this field of research. It proposes the existence of mainly two classes of particles: fermions and bosons with half-integer and integer spins respectively. The fermions are further divided into quarks and leptons namely up, down, strange, charm, bottom and top and the electron, muon and tau with a neutrino being assinged to each of the three leptons. Additionally an anti-particle is assigned to each fermion. All fermions except the neutrinos posses a mass in the standard model. Further information is presented in Figure 3.1.
Interactions of these particles are described by the exchange of bosons. The force of two ferminos acting upon each other is described by three types of forces in the SM. The strong force couples to all quarks through gluons, the electromagnetic force acts on to all particles that carry electric charge and is mediated via photons and the weak force effects all fermions and is transported by the $W^{+/-}$ and Z bosons. The Higgs boson explains the origin of mass and thus only couples to the massive particles. However the SM cannot describe all fundamental aspects of physics as the gravitational force is not included. Additionally differences from experimental observations such as the oscillation of massive neutrinos or particles, explaining dark matter. To include such effects new models with new particles beyond the standard model (BSM) are developed. The search for these particles is the main task of todays colliders.

3.2 Colliders and Jet Production

The biggest of these colliders is the Large Hadron Collider (LHC). By accelerating and colliding two proton beams it reaches a center of mass system (CMS) energy of $\sqrt{s} = 13\text{TeV}$. During the collision new particles (final state particles) can form if enough energy is provided. These decay further as they travel towards the calorimeters where the products can be observed. If the final state particles are strongly boosted, their decay products will enter the detector collimated and result in a high number of counts in a small detector region. These are referred to as jets. The goal of analysing the detector data is to reconstruct the final state particle with significant precision to prove its existence or to measure its production rate. Focusing on the jet structure has been shown to be the most

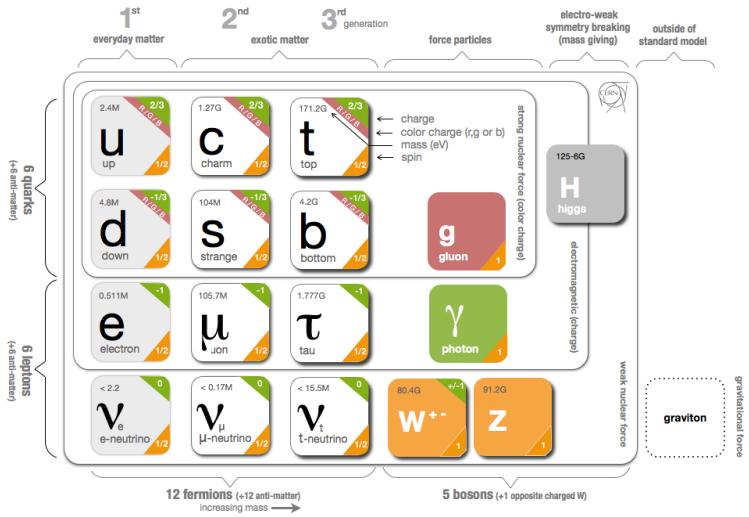


Figure 3.1 Particles of the standard model. Image taken from [14].

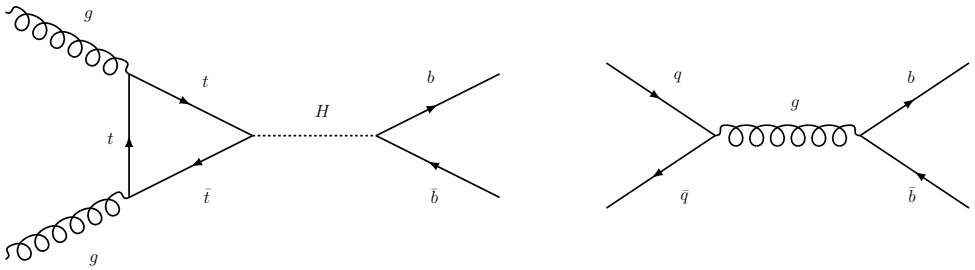


Figure 3.2 Example Feynman diagramms of bottom-pair production via Higgs boson (left) and quark anti-qurak annihilation (right).

effective way of reconstruction. The Higgs boson was the last fundamental particle to be newly detected at the LHC by the ATLAS and the CMS collaboration [15, 16].

3.3 $H \longrightarrow b\bar{b}$

As the Higgs boson is coupling to proposed dark matter particles in many theories, it is an interesting particle to investigate for the search for BSM particles. Yet the detection of the Higgs via its decay channel into a bottom pair with a high branching ratio ($\mathcal{B}(H \to b\bar{b}) \approx 58\%$ [17]) is a challenging task due to a strong QCD-background. Examples of the two production channels are shown in Figure 3.2. The events of interest are the, ones where a strongly boosted bottom pair is produced. In the detector, the resulting jets show two subjets (one for each bottom quark), referred to as Higgs candidate jets. The seperation of boosted Higgs decays against the signal-like QCD-background was approached by the CMS collaboration [18] by analyzing the subjet structure and Joshua Li et al. [3] using Convolutional Neural Networks on both, event and subjet information. During our analysis we want to study the performance of Capsule Networks on the task of discrimination.

Chapter 4

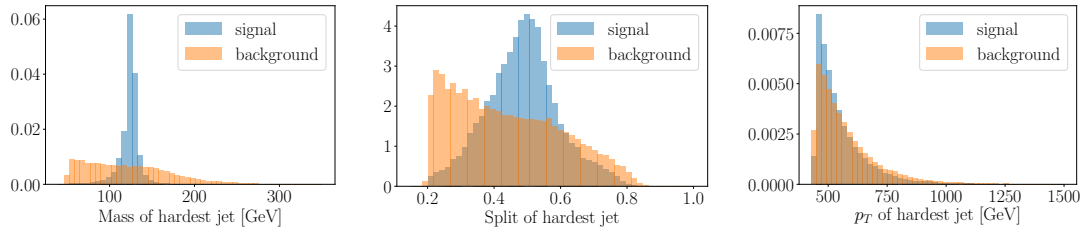
Application of Capsule Networks on $H \longrightarrow b\bar{b}$

4.1 Dataset

For a good comparison the same data as in [3] is used. It includes pp collisions with $\sqrt{s} = 13$TeV, simulated using MadGraph5_aMC@NLO 2.6.2 [19] and Pythia 8.226 [20]. The background consist of two, three and four jet events and signal events contain a Higgs-jet with either one or two other jets. To filter the events that contribute to boosted Higgs jets as they are described in 3.3, the jets in the data were clustered using Fastjet [21, 22] and the anti-k_T algorithm [23] with $R = 0.8$. To extract subjet information $R = 0.2$ was used. Only data with the hardest jet having a $p_T > 450$GeV and fulfilling a double b tag are used. Having applied these cuts, the hardest jet is always the two prong Higgs candidate. For more details concerning the event generation, please consider [3]. The dataset was split into a training, test and validation set with about 58000, 19000 and 19000 instances respectively that are statistically independent. For performance evaluations different parts of the dataset where used for the three categories.

To allow a distinction between signal and background, the network has to learn the input features that deviate strongly between signal and background. The most obvious difference is the mass distribution of the hardest jet in the event (see Figure 4.1a). While the signal data has its distribution centered around the Higgs mass, the background distribution is wider. However this feature is not directly included in the data presented to the network and can only be learned indirectly. As the Higgs candidate jet always contains a two prong structure, we can define the split of its two subjets by measuring their distance in η and ϕ. The distribution of this split in the hardest jet varies strongly between signal and background data and is directly obtainable from the input data, as it is a geometrical feature (see Figure 4.1b). Other distributions such as the transverse momentum or the positions of the events jets are almost identical for both inputs.
In addition, correlations of different observables can differ from signal to background. Even though only a small difference is visible in p_T itself (Figure 4.1c), its connection with the earlier mentioned jet split yields is a clear distinction 4.2.

For the background data there is no strong correlation visible in contrast to the signal. Subjets of strongly boosted Higgs decays are more collimated, which is well visible in the data. Especially in the high p_T regime a smaller jet radius can be seen. This is an interesting connection to be observed and exploited by the network, since a correlation of an event and subjet information has to be learned.



(a) **(b)** **(c)**

Figure 4.1 Normalized distributions of mass (a), subjet split (b) and p_T (c) of hardest jet of 50,000 signal and background instances.

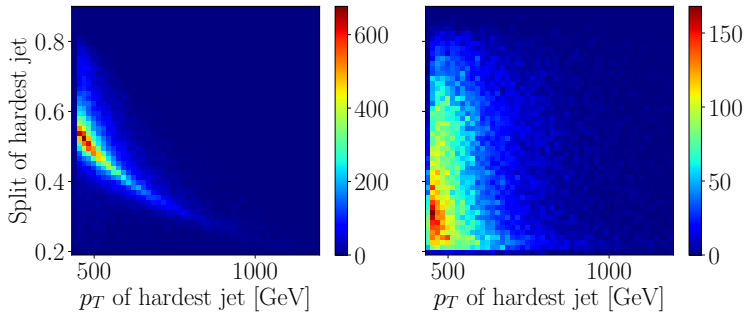


Figure 4.2 Correlation of p_T and subjet split of the hardest jet for signal (left) and background (right).

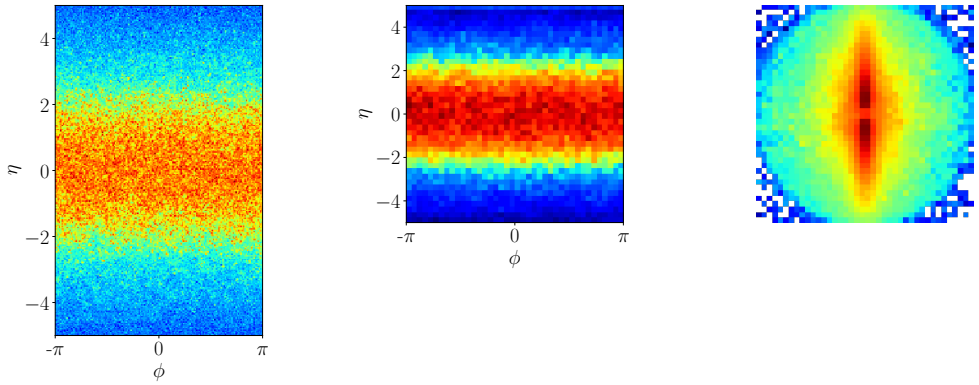


Figure 4.3 Average of 5000 images of the input types. Left: high resolution event, middle: low resolution event, right: jet image.

The data is presented to the networks in form of calorimeter images. To study the CapsNets output, we train it on three different input types: A low resolution global event image with a resolution of 40×40 pixels, the corresponding image of the hardest jet with the same resolution and a high resolution event image with 160×250 pixels. The event images represent the positions according to the azimutal angle ϕ with its full range from $-\pi$ to π and the pseudo rapidity η with the range $|\eta| < 5$. The jet-clustering to extract the jet images is done using FASTJET. Each image consists of four channels, containing the information about the p_T of all, only the charged and only the neutral particles and the charge deposited in each pixel. An average image of the first of these channels can be see in Figure 4.3.

4.2 Architectures

The benchmarking of the Capsule Networks performance is done with respect to the CNN architecture of [3]. For the first part of the evaluation we reconstruct the CNNs architecture and build a Capsule Network, with the Dense Layers being replaced by Capsule Layers. The input consists of two of the input images, the low resolution event image and the jet image. The images are propagated through distinct streams in a set of convolutional blocks, before being flattened and concatenated to be further processed in Dense Layers, with the final output of two neurons, indicating the probability of the input being signal or background. In construction of the CapsNet we substitute the flattening with a PrimaryCap and the Dense Layers with Capsule Layers. The lengths of the two output vectors are then used for classification with their difference denoting the classification outcome. The downsampling in the convolutional blocks is done by Pooling Layers and the event stream includes padding along the azimutal direction to incorporate the periodic boundary condition. For all Convolutional and Dense Layers we apply ReLU, except for the last Dense Layer, where the softmax activation is used. The Capsule Layers are normalized by the squashing activation. The used CNN has about 2.8 million trainable parameters while the Capsule Network has about 450 thousand weights and biases, with the Capsule Layers being modeled after default networks that where used for early analyses. To ensure that the disadvantage in the number of trainable parameters does not prevent the CapsNet from reaching the CNNs performance, we introduce the architecture referred to as Capsule big with larger Capsule Layers resulting in about the same amount of parameters as the CNN. The structures of the networks can be seen in Figure 4.4.

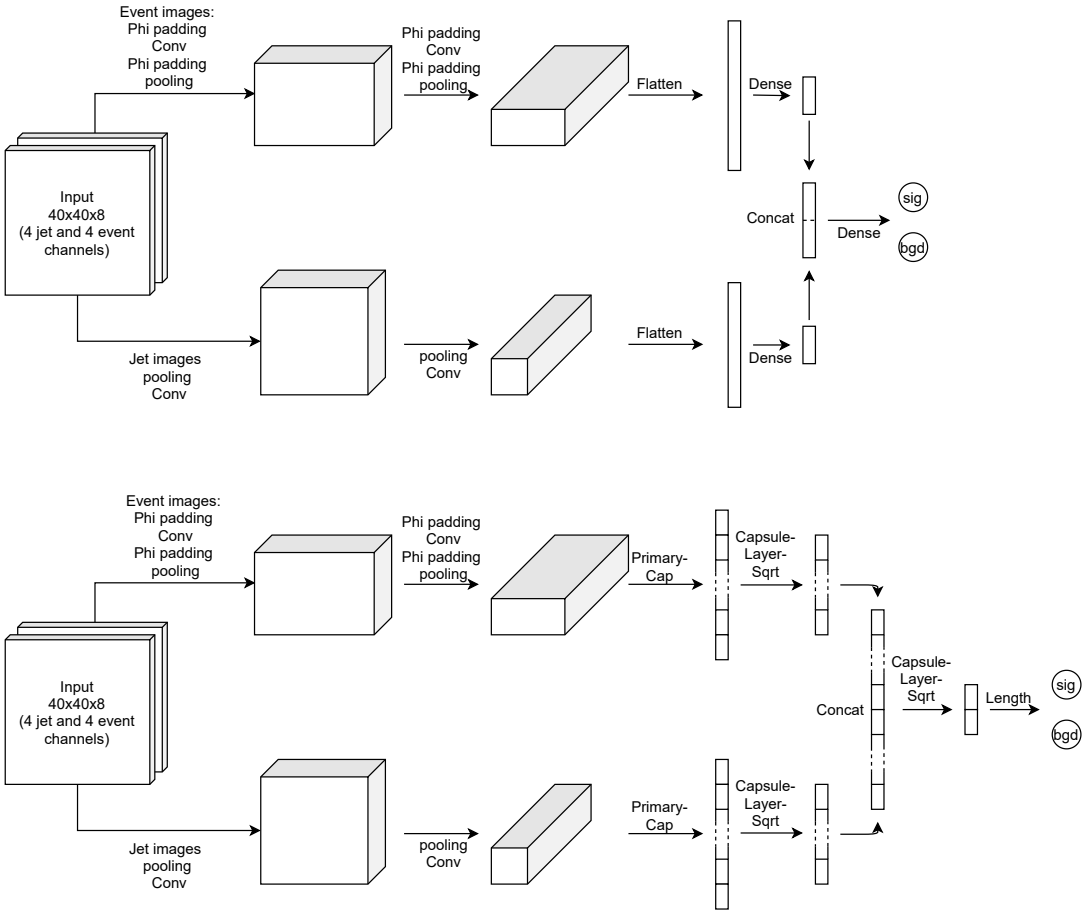


Figure 4.4 Architectures of the CNN (upper) and CapsNet (lower) two stream networks. If multiple labels are attached to an arrow, the operations are applied consecutively.

For the application on high resolution event images only a single stream network is required. We build the models as for the event stream using PhiPadding layers and pooling. Additionally Dropout is used as regularization with a droprate of 0.25. For evaluation we use a CNN of this kind with about 1.6 million and a CapsNet of about 450 thousand parameters. The output capsules are eight-dimensional. All implementations are done using Keras [24] with Tensorflow [25] backend.

4.3 Performance Evaluation

To measure the performance we use the ROC-curve and significance plots. Both use the ratio of correctly classified signal events to the their total number (ϵ_S) and the number of background events being falsely classified over the total number of background events (ϵ_B). Both are calculated for different thresholds for the hard classification on the smooth

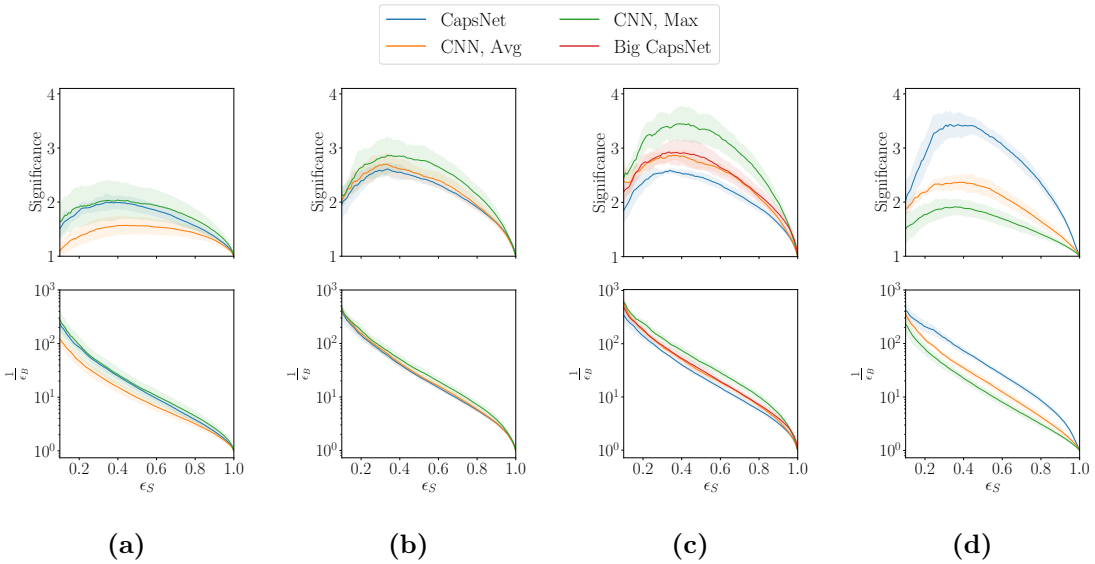


Figure 4.5 ROC and significance curves for trainings on different inputs for CNNs with average and max pooling and CapsNets: (a): low resolution event, (b): jet image, (c): jet and event image, (d): high resolution event. The upper plots show the significance and the lower ones the ROC curves. Five trainings per plot, strong lines denote the mean, colored areas show the standard deviation.

output of the networks. For the ROC-curve for each of these values $\frac{1}{\epsilon_B}$ is plotted against ϵ_S. The significance is calculated by:

$$\text{Significance}(\epsilon_S) = \frac{\epsilon_S}{\sqrt{\epsilon_B(\epsilon_S)}} \tag{4.1}$$

The quality of the network is measured by the maximum of the significance curve. To improve reliability we train each model on five randomly shuffled training sets. The results can be seen in Figure 4.5.

Both network types perform almost equally well on the low resolution event and jet image input, with the CNNs doing slightly better on the latter. For the two stream architectures the CNN outperforms the CapsNets regardless of its parameters. For the high resolution input the opposite is the case: The Capsule Network seems to be more capable to extract meaningful information from the full event image. However the best CapsNets and CNNs do perform equally well within the precision of the evaluation as we see in the direct comparison of the two in Figure 4.6.

4.4 Criteria of Classification

As the CapsNets perform well we want to examine its criteria of classification by looking at correlations of its classification output with the observables we determined to be important

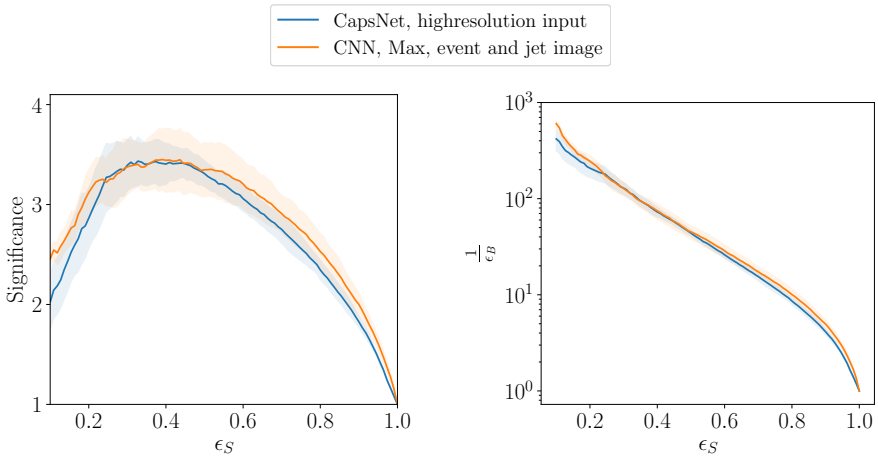


Figure 4.6 Comparison of best setups of CNN and CapsNet. Left: Significance curves, Right: ROC curves

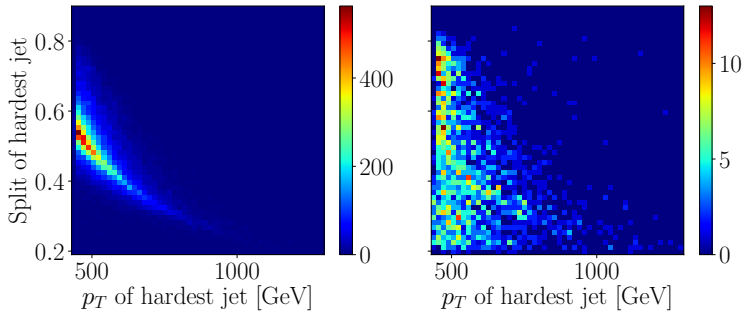


Figure 4.7 Correlation of p_T and subjet split of the hardest jet for signal data classified as signal (left) and signal data classified as background (right).

for the task. The connection to the split of the subjets of the hardest jet is a clear criterion for the classification as the correlation is well visible in Figure 4.2. To check if the network also learned the correlation of the transverse momentum with its subjet-split we look at signal data being correctly and falsely classified by a network that was trained on the high resolution images. The less "signal-like" events should diverge from the curve which is visible Figure 4.2. If no correlation is learned both correlation plots should resemble the general shape of the signal data with only showing differences along the axis of the subjet-split. Yet in Figure 4.7 we see that the falsely classified signal events resemble the correlation plot of the background data. It is thus clearly visible that the connection of these observables is learned. However the data representation of this kind is also given by the standard CNN-based classifiers and doesn't need the capsule architecture. Their special property is the representation of the classification as vectors in their latent space which we want to analyse further.

4.5 Data Organization by the Capsule Network

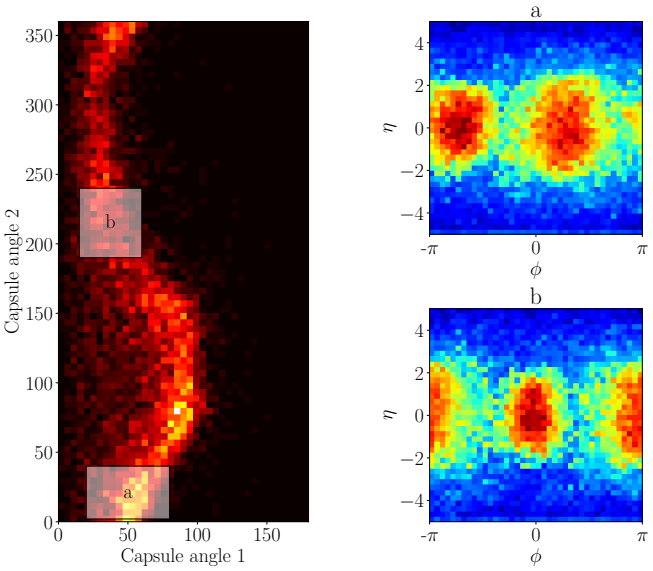
The discriminative power of the networks is clearly visible from the performance evaluation. Yet to confidently use them on real data in physics research it is crucial to have an understanding of how the networks operate. By representing their classification output as vectors Capsule Networks provide a more detailed description of the features they are sensitive to. We want to analyse the additional information the CapsNets encode in contrast to CNNs at different input types. To do so, we look at the angular distributions in the latent space of the output capsules.

To visualize their output, we train Networks with two or three dimensional output capsules and plot histograms of the classification of many input instances in the latent space. For two dimensions the representation is straight forward on a flat two dimensional plot. For three dimensions we only look at the angular distribution of the output latent space, with the angles being constructed using standard spherical coordinates. The angular positions of the classifications are then directly shown in a flat coordinate system. This representation of the sphere leads to a strong distortion but general shapes such as closed curves can still be well seen. To understand the criteria of organization we look at the referring input or its observables of certain latent space regions. To make the differnent plots easily distinguishable, the average input images will be presented in logarithmic scale in a more colorful colormap and the latent space is shown as standard two dimensional histogram with darker colors. Observables and are represented by scatterplots.

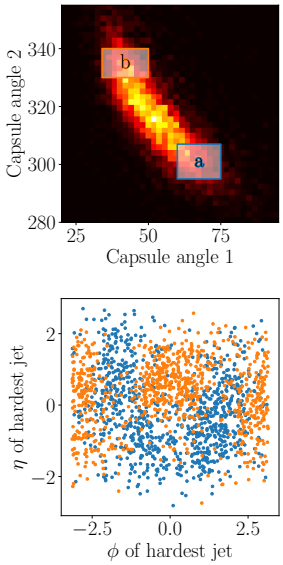
Because the CapNets are not motivated by any loss to distribute the input in any direction besides the radial one, we want to understand the information encoded in the outputs angular direction. The input types as they are used in Section 4.3 provide different information to the network and thus result in a variation of the latent space representation. We want to study these in order to better understand the networks learning process.

4.5.1 Two Stream Network

To examine the discrimination of the two stream Network we separately look at each stream, starting with the event stream. While the signal is spread out far in the latent space the background capsule is more collimated. The three dimensional representation in a closed curve is already indicating the distribution according to an observable with cyclic boundary conditions. Looking at the differences in the input of different regions, the main criterion of organization is the ϕ-coordinate of the hardest jet (Figure 4.8a). The background capsule on the other hand is learning a combination of both: the η and ϕ position. It is used for a differentiation as it is seen in the different correlations of the coordinates in different regions (Figure 4.8b). This leads to the conclusion that the network is able to



(a) Output latent space of signal capsule (left) with referring average input images (right)



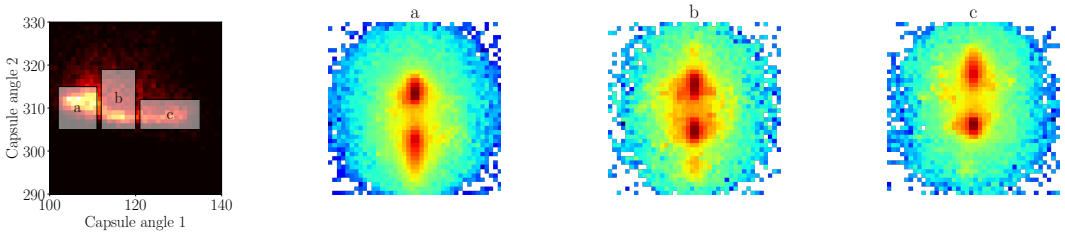
(b) Output latent space of background capsule (top) with referring observable scatterplot (bottom)

Figure 4.8 Differences in input data in different regions of the output latent spaces of (a) the signal and (b) the background capsule for three dimensional networks, trained on low resolution event images.

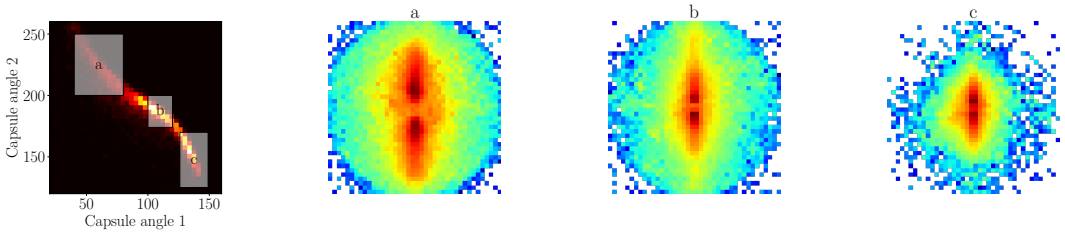
locate the hardest jet to focus on regions in the input that contain the crucial information, namely the hardest jet.

If the network is only presented with this information in its second stream, other features have to be learned. The populated region in the latent space is smaller than before. For the events, geometrical information was used to distribute the data in the angular direction of the output latent space. The differencies in the input images are more subtle in the jet images, which offers less criteria to organize by. For the background a clear distinction by the split of the subjets is visible as it can be seen in Figure 4.9b. The signal capsule orders the jets according to the disbalance of the p_T of the subjets. This can occur due to the one of the bottom particles being more boosted in the laboratory frame. As before the geometrical information is used in for the classification in the angluar direction.

If the Capsule Network is presented with both inputs in its full architecture, the interesting question is whether the stronger varying geometrical information of the event or the, classification wise more important, jet information is used for the angular distribution. Looking at Figure 4.10 we see big differences in the average jet images for different regions. The average event images only show slight changes. The background capsule organizes the data by the subjet split as before. The signal capsule on the other hand learns a



(a) Signal capsule



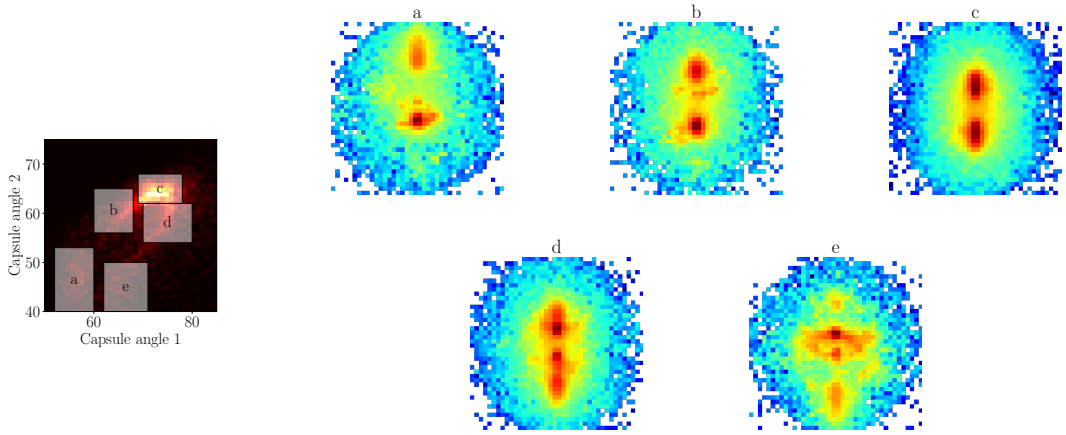
(b) Background capsule

Figure 4.9 Differences in input data in different regions of the output latent spaces of (a) the signal and (b) the background capsule for three dimensional networks, trained on jet images.

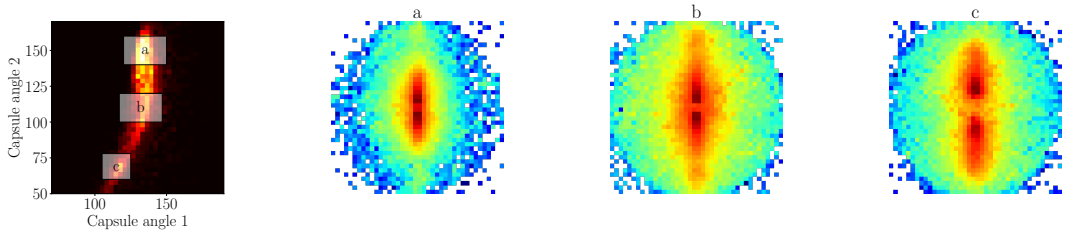
more detailed structure of the hardest jet differentiating, not just by the imbalance of the transverse momentum of the subjet, but also by its direction. As it appears to be an information not being highly interesting for the classification task itself it organized in the angular direction of the capsule. Even though the event data isn't organized itself, its seems to help to gain information on the subjet structure which we can see in the better distinction of the input by the signal capsule. As the subjets are the most sensible information for the classification, this might be the reason for the improved classification, seen in the performance evaluation.

4.5.2 One Stream Network

The one stream Capsule Network provides a much better classification performance than the two stream architecture. Its difference in the performance gives rise to the a useful connection between event and subjet information. Looking at the representation of the high resolution input, the network again, as for the low resolution images, learns features of the position of the hardest jet. While the signal capsule is more sensitive to the ϕ-coordinate of the Higgs candidate jet, the background, as on the low resolution input, learns a correlation of its ϕ and η position. A two dimensional example is shown in Figure 4.11 in the appendix. This shows the capability of the network to locate the hardest jet as this is the region with the most useful information. This way the sparsity of the input images can be circumvented.



(a) Signal capsule

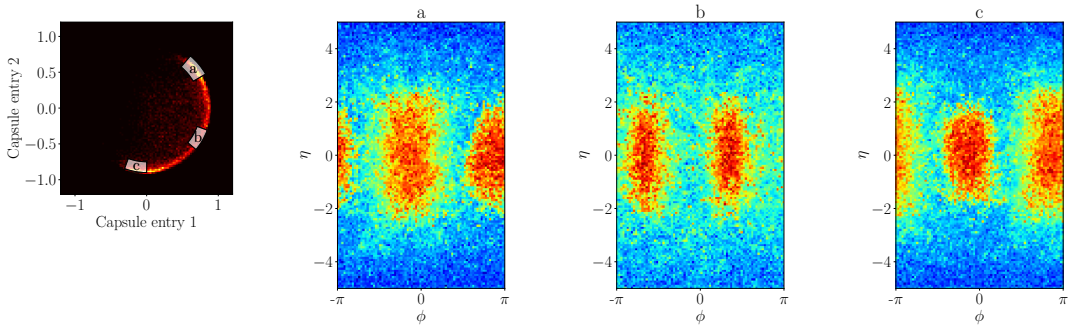


(b) Background capsule

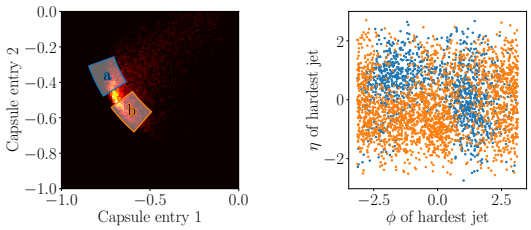
Figure 4.10 Differences in input jet images in different regions of the output latent spaces of (a) the signal and (b) the background capsule for three dimensional networks, trained on and event and jet images.

However being a symmetry of the physical process, the azimuthal angle is no reasonable feature to learn for the classification besides its role in finding the jets. To take the necessity to learn this feature, we rotate all images such that all hardest jets of the events share the same ϕ coordinate. This way the networks signal capsule has the possibility to focus on features with more discriminative power. The background capsules capability to obtain the pseudo rapidity from the image is further enhanced by this preprocessing, producing a sharp resolution for this coordinate (see Figure 4.12b). Being the most dominant geometric feature of the input we would expect the same for the signal. However only a slight organization with respect to η of the hardest jet can be seen. The orientation of the subjet is clearly different in distinct latent space regions. The more refined data allows for a better representation of the input in the latent space.

As a last manipulation of the input, we additionally shift all η-coordinates in such a way that the hardest jet is located in the center of the input images. The last geometrical features extractable from the input are now the position of the remaining jets and the structure of the hardest one. As before a focus on the most informative region is expected and clearly visible in Figure 4.13). The structure of the hardest jet is the main criterion to



(a) Signal capsule

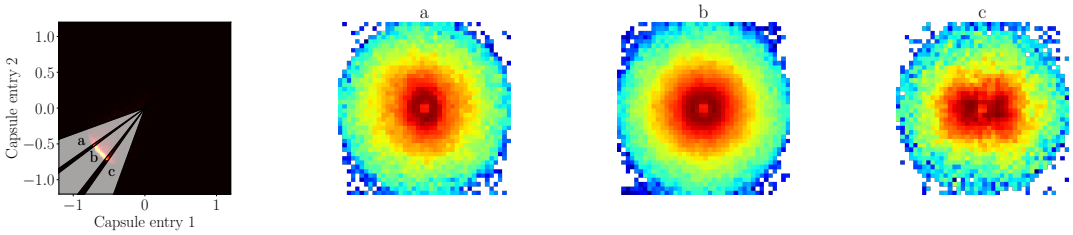


(b) Background capsule

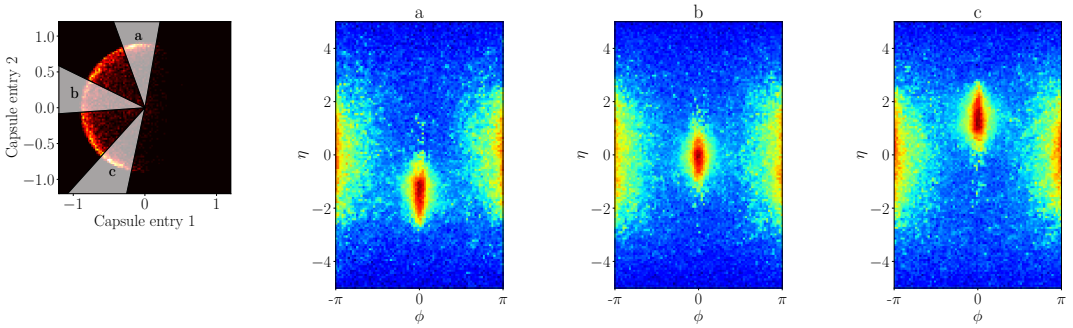
Figure 4.11 Differences in input data in different regions of the output latent spaces of (a) the signal and (b) the background capsule for two dimensional network, trained on high resolution event images. Average input images are shown for signal and correlations of ϕ and η position of the hardest jet are visible for the background.

order the data in the latent space with the background capsule again learning the split of the subjets (see Figure 4.13b). The signal capsule only learns a more subtle differentiation in the orientation of its jets (see Figure 4.13a). The even stronger focus on the hardest jet again shows the focus of the CapsNet on the most informative region. As it was visible for the two stream network, we expect the better distinction of the events according to the subjet structure to be accompanied with an improvement in performance.

To verify whether this is again the case, we compare the performance to the CapsNet that was trained on the original images as it was used for the performance evaluation. Networks with the same architectures and hyperparameters are again trained five times on the different input types. The results can be seen in Figure 4.14 and show an improvement due to the centering of the ϕ coordinate of the hardest jet outperforming the former best CNN and CapsNet models. The second modification seems to have no influence within the precision of the evaluation.

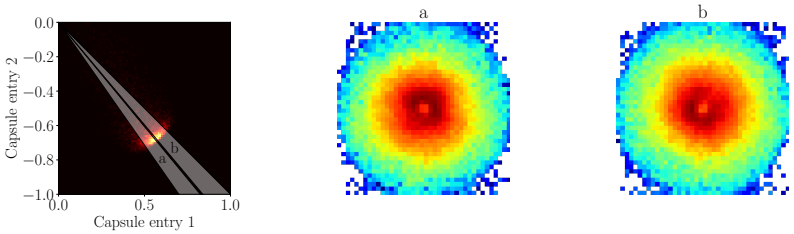


(a) Signal capsule

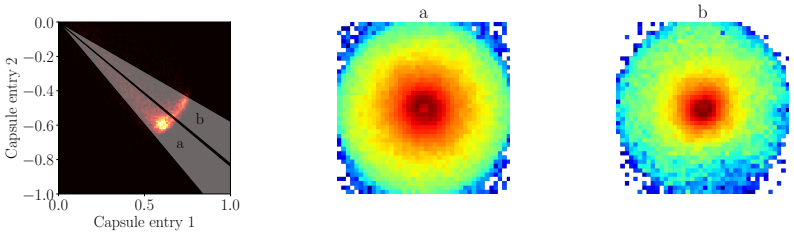


(b) Background capsule

Figure 4.12 Differences in input data in different regions of the output latent spaces of (a) the signal and (b) the background capsule for two dimensional network, trained on high resolution event images with a centered ϕ-position of the hardest jet. Zoomed jet images are shown for signal, full event images for background



(a) Signal capsule



(b) Background capsule

Figure 4.13 Differences in zoomed jet images of the input data in different regions of the output latent spaces of (a) the signal and (b) the background capsule for two dimensional network, trained on high resolution event images with a centered ϕ and η-position of the hardest jet.

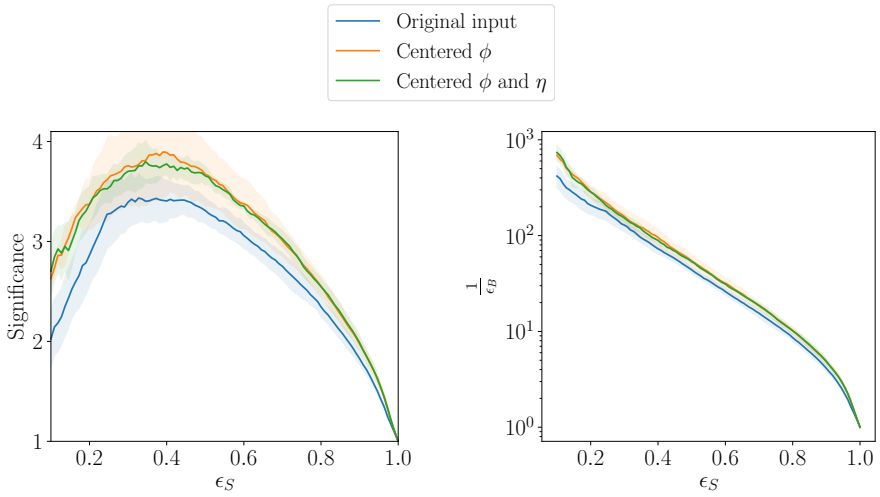
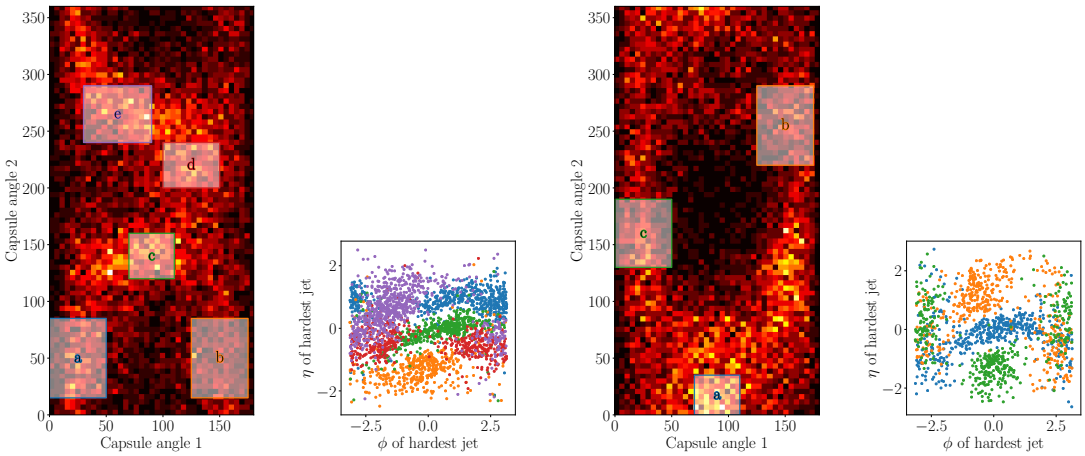


Figure 4.14 Comparison of CapsNets with different input types. Left: Significance curves, Right: ROC curves

4.5.3 Populating the Latent Space

To better understand the criteria for the angular distribution in the latent space we want to force the network to populate a larger part of it. To achieve this, we introduce an additional loss. We want the network to use as much of the latent space as possible and thus define the target of its angular distribution to be a uniform distribution. We define a differentiable function which constructs a histogram for the given data, approximating the steps by sigmoid functions. The distributions of the azimuthal and the polar angle of the latent space representiation of a batch, are fed seperatiely to the loss function. Using the Kullback–Leibler divergence of the constructed histogram with a uniform distribution as target, a new loss is calculated and added to the loss for the classification task, with a weight regulating its influence. The differentiability of the histogram function is crucial to the learning process since otherwise no optimization of this task would take place. In two dimensions the performance completely collapsed if the additional loss was applied. For three dimensions the classification accuracy also decreases a lot (about 67% in contrast to about 75% for other three dimensional models). Yet some classification is learned and the consequences of the further spread in the latent space can be observed. The task of populating a larger region of the latent space worked out well as the range of both angles is fully used. Looking at the criteria of the split, a clear correlation with the hardest jets position is visible. As one can see in Figure 4.5.3 both, ϕ and η of the hardest jet, are organized by the network in a more precise way as before. Thus the new loss reached its goal to encode more refined information in the angular classification but with the huge payoff in the performance it has to be refined further to be put to reasonable use.



(a) Signal capsule **(b)** Background capsule

Figure 4.15 Differences in input data in different regions of the output latent spaces of (a) the signal and (b) the background capsule for three dimensional network with additional loss, trained on high resolution event images.

Chapter 5

Conclusions and Outlook

In this thesis we analyzed the application of Capsule Networks in distinguishing the boosted Higgs decay into a bottom pairs against QCD-background, also producing boosted bottom pairs. First, we compared the performance to a CNN based approach. While the CNN performed better in a two stream architecture given low resolution event and jet images, the CapNets outperformed it, when the networks were presented only with high resolution event images. Yet in comparison of the best models within all categories, the CNNs and the CapsNets could distinguish signal and background events equally well (see Section 4.3).
For the second part, we focused on the representation of the input in the output latent space of the classification capsules. Experiments to motivate the network to use the full latent space in Section 4.5.3 showed good results in this manner, but also resulted in a strong decrease in the performance. As it was probed in this thesis, it is thus no viable option to include more information in the latent space. Yet, also without the additional loss, the Capsule Networks encoded interesting information in its output capsules. The representation of both, event and subjet level information is clearly visible. The radius of classification is most sensitive to subjet information (Section 4.5) and the angular distribution in the latent space corresponds to the position of the Higgs candidate jet in the event, if this varies in the input images. As the networks represent the jets position in the classification angle, it represents the input region it focuses on and gives a better insight on its classification criteria. It does not just represent the geometrical feature of the input that varies strongly, but the most interesting one for the classification as we could see on the analyses of the two stream network in Section 4.5.1. Consecutively stripping the information the network encoded in its latent space in Section 4.5.2 and presenting it with the information it needed to learn before, the focus of the network on the hardest jet got more precise. This also lead to better models, outperforming the former best Capsule Network and the best CNN. Showing an even better performance than the CNNs, giving better insights in its classification process and being able to use and represent event and subjet information simultaneously, Capsule Networks are great tools to better distinguish Higgs decays against QCD-background.

As the performance of the networks is connected to its ability to organize the data in the latent space, it would be interesting to investigate Capsule Networks with higher dimensional output capsules, as these show a better performance. For an analysis as in this thesis, their dimensionality could be reduced by applying a Principle Component Analysis (PCA) [26]. Additionally, to study whether more detailed information, than the

one we can perceive in our analysis, is encoded in the output latent space, one could use a well trained Capsule Network as encoder for an Autoencoder [27]. The decoder is then trained to reconstruct the input images from the CapsNets output. This yields the possibility to fully extract the information, encoded by the network regardless of the dimensioanlity of the model.

References

[1] Sascha Diefenbacher et al. "CapsNets continuing the convolutional quest". In: *SciPost Physics* 8.2 (2020) (cit. on pp. 1, 7, 8).

[2] Gregor Kasieczka et al. "The Machine Learning landscape of top taggers". In: *SciPost Physics* 7.1 (2019) (cit. on p. 1).

[3] Joshua Lin et al. "Boosting $H \to b\bar{b}$ with machine learning". In: *Journal of High Energy Physics* 2018.10 (2018), p. 101 (cit. on pp. 1, 12, 13, 15).

[4] Geoffrey Hinton, Sara Sabour, and Nicholas Frosst. "Matrix capsules with EM routing". In: 2018 (cit. on p. 1).

[5] Jonathan M. Butterworth et al. "Jet Substructure as a New Higgs-Search Channel at the Large Hadron Collider". In: *Physical Review Letters* 100.24 (2008) (cit. on p. 1).

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366 (cit. on p. 3).

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on p. 5).

[8] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules". In: *CoRR* abs/1710.09829 (2017). arXiv: `1710.09829` (cit. on pp. 6, 7, 9).

[9] Chigozie Nwankpa et al. "Activation Functions: Comparison of trends in Practice and Research for Deep Learning". In: *CoRR* abs/1811.03378 (2018). arXiv: `1811.03378` (cit. on p. 6).

[10] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press, 2002 (cit. on p. 7).

[11] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: `1412.6980 [cs.LG]` (cit. on p. 7).

[12] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958 (cit. on p. 7).

[13] Xifeng Guo. *A Keras implementation of CapsNet in NIPS2017 paper "Dynamic Routing Between Capsules"*. 2017 (cit. on p. 7).

[14] *CERN Document Server*. `https://cds.cern.ch/record/1473657`. Accessed: 2020-02-25 (cit. on p. 12).

[15] S. Chatrchyan et al. “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 30–61 (cit. on p. 12).

[16] The Atlas Collaboration. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: 2012 (cit. on p. 12).

[17] The LHC Higgs Cross Section Working Group et al. *Handbook of LHC Higgs Cross Sections: 3. Higgs Properties*. 2013. arXiv: `1307.1347 [hep-ph]` (cit. on p. 12).

[18] A. M. Sirunyan et al. “Inclusive Search for a Highly Boosted Higgs Boson Decaying to a Bottom Quark-Antiquark Pair”. In: *Physical Review Letters* 120.7 (2018) (cit. on p. 12).

[19] J. Alwall et al. “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations”. In: *Journal of High Energy Physics* 2014.7 (2014) (cit. on p. 13).

[20] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. “A brief introduction to PYTHIA 8.1”. In: *Computer Physics Communications* 178.11 (2008), pp. 852–867 (cit. on p. 13).

[21] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. “FastJet user manual”. In: *The European Physical Journal C* 72.3 (2012) (cit. on p. 13).

[22] Matteo Cacciari and Gavin P. Salam. *Dispelling the N^3 myth for the Kt jet-finder*. 2005. arXiv: `hep-ph/0512210 [hep-ph]` (cit. on p. 13).

[23] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. “The anti-ktjet clustering algorithm”. In: *Journal of High Energy Physics* 2008.04 (2008), pp. 063–063 (cit. on p. 13).

[24] François Chollet et al. *Keras*. `https://keras.io`. 2015 (cit. on p. 16).

[25] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015 (cit. on p. 16).

[26] Karl Pearson F.R.S. “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. eprint: `https://doi.org/10.1080/14786440109462720` (cit. on p. 27).

[27] Cheng-Yuan Liou et al. “Autoencoder for words”. In: *Neurocomputing* 139 (2014), pp. 84–96 (cit. on p. 28).

List of Figures

Acknowledgements

First of all I want to thank Prof. Tilman Plehn for the opportunity to combine physics and machine learning on a new interesting topic and his help in understanding their connection. A special thanks goes to Michel Luchman, providing advice whenever it was needed on either computational or physics questions and for proof reading this thesis. For always being open for discussions on computational problems I want to thank Lukas Blecher. Finally, I want to thank the whole LHC Physics and New Particles Group for a welcoming atmosphere, making the time here very enjoyable.

Declaration of Authorship

I hereby certify that this thesis has been composed by me and is based on my own work, unless stated otherwise.

Heidelberg, 02.03.2020

Christopher Lüken-Winkels