Using Jet Images and Deep Neural Networks to Identify Particles in High Luminosity Collisions

Jovan Nelson

May 4, 2017

Searches for new physics at the Large Hadron Collider often require identifying highly boosted hadronic decay products of heavy particles. When deep neural networks are applied to pixelated images of jets with and without internal structure, they can out-perform traditional observables used to classify jet sources. In order to understand how these techniques would perform in realistic experimental conditions, we applied them to simulated proton collision events with 0, 35, 140, and 200 additional proton-proton interactions. In this study we have shown that it is possible for deep neural networks applied to jet images containing extra radiation from additional interactions to distinguish significant substructure features and maintain classification performance.

1 Introduction

We can attempt to recreate the microscopic early universe by accelerating particles to the highest possible energies and colliding them. The key to understanding particle collisions (and hence the dynamics of the universe) is measuring the properties of particles emanating from these collisions. As some particles interact with detector material, they form dense showers of particles, called jets. Most jets are the result of the fragmentation of standard model (SM) particles such as quarks and gluons. At collision energies like those at the Large Hadron Collider (LHC) [1], production of new heavy particles can result in jets with interesting substructure properties. A complication in these searches is distinguishing between jets of interest and jets from background processes. Developing novel tools to classify these jets has become a focused pursuit of particle physicists over the last several years. A unique approach to accomplishing this task is to take a "picture" of these jets and then use computer vision and neural network techniques to distinguish between different types of jets. Based on a study performed at Stanford Linear Accelerator (SLAC) [11, 16], it has been shown that jet image techniques show promising performance. Yet, as the LHC moves toward its upcoming high intensity phase, more studies are needed to understand if these tools are robust against increasingly dense environments with many simultaneous collisions.

2 Conventional Jet Substructure Tagging

Jets are formed from clusters of hadrons and other particles that are produced by the fragmentation of quarks and gluons. These showers are what we use to infer the presence of quarks and gluons that have color charge in collision decay products. Particles with color charge are prevented from existing individually because of Quantum Chromodynamics (QCD) confinement. Jets carry the mass and energy of these particles, thus if we reconstruct and identify them, we can see what happens at the heart of a high energy collision. A common particle clustering algorithm used in these studies to create jets is called the anti-k_T algorithm [2] and produces jets that have cone-like shapes with a characteristic radius, similar to those shown in Fig. 1.

Many models of physics beyond the SM predict new particles with very large masses (on the order of $\sim$ 100 GeV to $\sim$ 1 TeV). Some examples are

supersymmetric or vector-like top quark partners [3, 4] that could stabilize the mass of the Higgs boson [5, 6], which is responsible for the mass of the fundamental particles, e.g. electrons. Another particle predicted by several models is heavy electroweak boson called the W', which is used in this study. Heavy new physics particles often decay into lighter SM particles (on the order of $\sim$ 1 GeV to $\sim$ 100 GeV) and therefore often carry away large momenta. These SM particles in turn may decay into light quarks (up, down, strange, charm, bottom), these quarks become highly Lorentz boosted. When this occurs, jets become very collimated and begin to overlap. Jets from boosted particles typically have high transverse momenta (p_T) and are best reconstructed using a large clustering radius that envelopes multiple small-radius jets. Figure 1 gives an example of how jets from increasingly boosted particles merge. The two leftmost pictures represent low p_T particles that decay while the two rightmost pictures are roughly 10 to 50 times that p_T. At the LHC, which probes the energy frontier, distinguishing between jets due to heavy particle production and high momentum light quark jets from multijet backgrounds is a real challenge. Improving our ability to analyze the internal particle content of these boosted jets is crucial for unlocking new discoveries in particle physics.

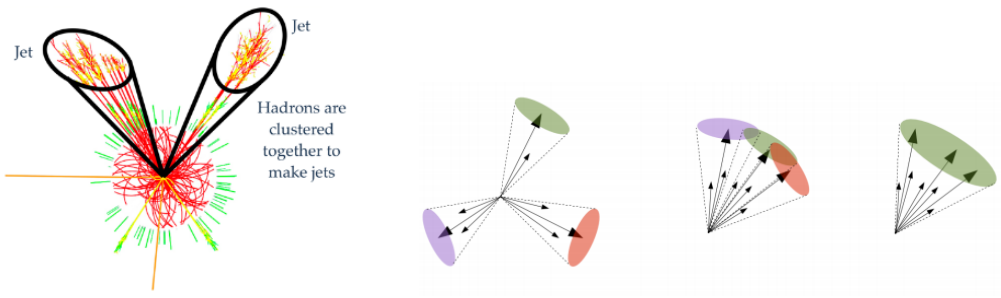


Figure 1: Examples of jet reconstruction with a typical cone shape (left). As the momentum of a decaying particle rises, jets from the decay products become more colinear (right).

To overcome the difficulties posed by the merging of jets, substructure identification techniques have been developed to differentiate between QCD background jets and jets from decays of boosted bosons or top quarks. This can be done with "grooming" techniques (such as soft drop [7], pruning [8], or trimming [9]) that remove low momentum and wide-angle particles from a jet

to better isolate the high momentum components. This gives a more accurate measurement of the invariant mass of the jet constituents (m_{jet}). The clusters of energy inside a jet can be characterized with observables derived from the jet constituents (tracks, and topological clusters, or "subjets"), such as the jet shape variable "N-subjettiness" [10]. This variable, labeled τ_N, quantifies the consistency of a jet with having N subjets. These conventional features are commonly used to tag jets from different sources. For example, jet from a W boson decaying into two quarks would be expected to have a mass near 80 GeV and two subjets. Although these methods have proven to be effective, computer vision and machine learning techniques could give much greater discriminatory power, in particular at the high intensity LHC.

In the studies presented here, W bosons are used as an example of a jet with internal structure. Jets from boosted hadronically decaying W bosons are created by simulating a theoretical heavy electroweak boson, W', that decays to a W boson and a Z boson. In this simulation the W boson decays hadronically, $W \to qq'$, and the Z boson decays to neutrinos, $Z \to \nu\bar{\nu}$. This produces a very clean event topology with one large-radius jet from the W boson decay and missing energy from the neutrinos. Background jets from light quarks are taken from QCD multijet simulations.

3 Jet Images and Computer Vision

Josh Cogan et al. at SLAC proposed a novel approach to jet classification through computer vision inspired techniques [11]. They create a visual representation of a jet by displaying its energy in a grid of calorimeter towers with spacing $\Delta\eta \times \Delta\phi = 0.1 \times 0.1$, spanning [-2.5, 2.5] in η (the polar angle with respect to the beam axis) and [0,2] in ϕ (azithmuthal angle around the beam). The transverse energy deposited by the jet's constituents in this grid of towers are stored as a pixelized intensity image, or "jet-image". Jet-images contain useful properties: they can be described by a fixed set of values, they do not compress information into set of derived variables, such as m_{jet}, and similarities between jets can be easily computed using standard linear algebra computations. With this visualization, a parallel can be drawn between jet physics and the facial recognition techniques that are strongly supported and developed by industries.

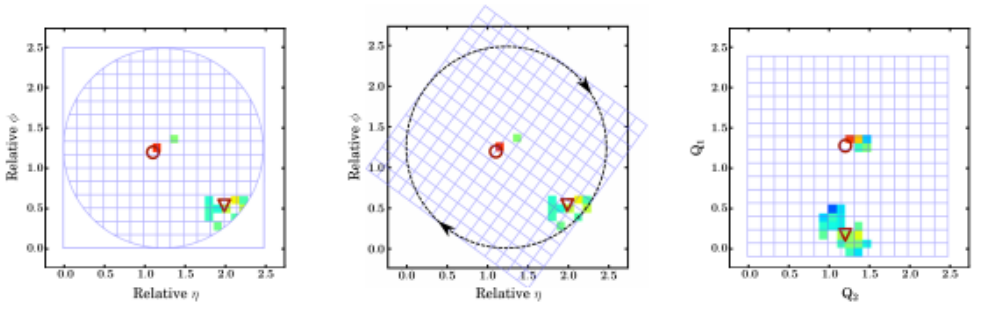
In order to classify faces in an image, it is possible to learn features from the pixel distributions in example images of the type of face that needs to

be recognized. Yet various arbitrary differences between example images, such as lighting or angle, can act as "noise" that obscures the important features of the image. These can be mitigated by preprocessing the images to extract the most important features. This significantly improves a network's performance for classifying faces. Using this analogue Cogan's group applies a series of preprocessing steps to extract the most important features from the pixel intensity distribution of a jet.

Noise reduction is done by applying a grooming algorithm to reduce the number of soft particles in the jet. Point-of-interest finding means locating the the positions of the leading areas of transverse energy deposition. Alignment makes the relative location of the primary feature in the grid always the same, through three actions: rotation to remove the symmetric nature of decay angle in the $\eta - \phi$ by putting the second leading subjet at $\pm\pi/2$, translation centers the leading energy deposit of jet in the same pixel, and reflection flips the image so that its maximal transverse energy always appears on the right side of the images. Examples of these alignment actions are shown in Figs. 2 and 3.

When constructing discriminants found in facial recognition Cogan et. al. used Fisher's Linear Discriminant (FLD) [12]. The FLD uses information from variations within the same type of image, and is therefore not strongly influenced by variations that do not distinguish between image types. Using a training set of preprocessed jet-images from two classes (signal and background), FLD produces a discriminant, called a "Fisher-jet", which has the same dimensionality as the jet-images. Jet-images are then projected onto the Fisher-jet to identify features of the two classes with positive and negative values in the projection.

Their case study differentiates between jets produced by boosted hadronically decaying W bosons (the signal class) and jets produced by light quarks (the background class). Their samples were produced by the Monte Carlo event generators MADGRAPH [13] and HERWIG++ [14] with proton-proton collisions at 8 TeV. Generated events are hadronized using the PYTHIA8 program [15]. Their jet-images were made from 25×25 calorimeter grids and they performed the preprocessing steps described previously. Figure 4 shows the "Fisher-jet" discriminant. In Fig. 5 the performance of the Fisher-Jet discriminant is shown: it performs better than a ratio of N-subjettiness variables at correctly identifying the W boson jets by a small but noticeable margin [11]. This demonstrates the potential of this method and the power of computer vision.



(a) Jet-image prior to rotation

(b) Rotated pixel grid

(c) Jet-image after projection onto rotated grid, before translation

Figure 2: Example of three image preprocessing steps. [11]

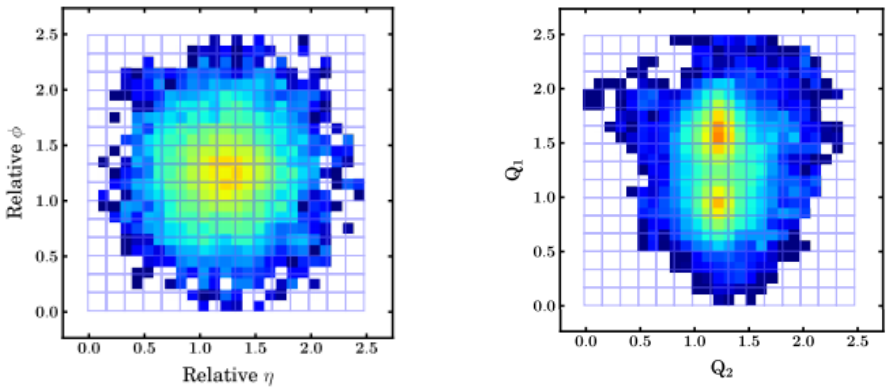


Figure 3: An example jet-image before (left) and after (right) processing steps. [11]

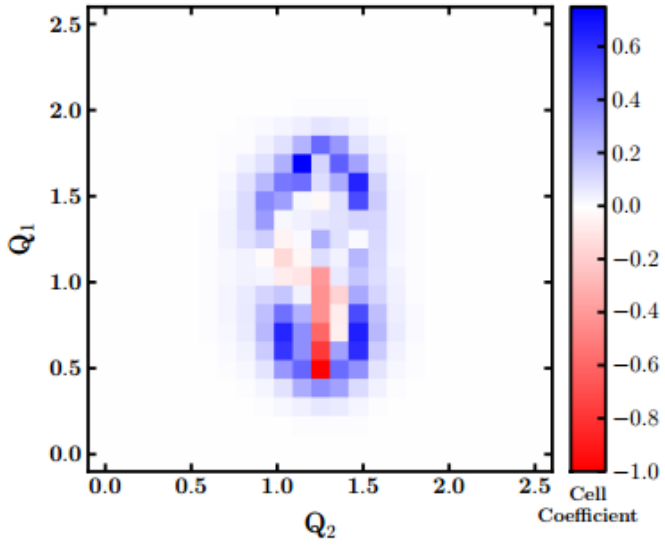


Figure 4: Example of a "Fisher-jet" from the FLD. [11]

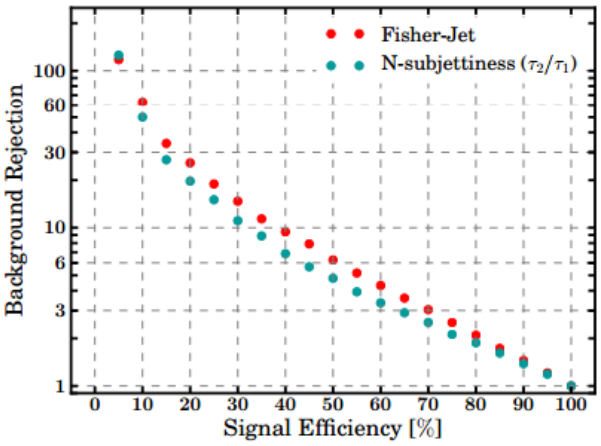


Figure 5: Performance of the FLD compared to τ_2/τ_1 in identifying W boson jets and rejecting mulitjet background jets. [11]

4 Extension of Jet Images to Deep Networks

Luke de Oliveira et al., associated with Stanford University and SLAC, have extended the jet-image technique by applying deep neural networks (DNN) rather than Fisher discriminants [16]. They used DNNs to classify the same type of boosted hadronically decaying W jets apart from a multijet background. These neural networks significantly outperform conventional classifiers such as N-subjettiness. Their jet-image framework is the starting point for the studies to be described in this thesis, so more details are given in later sections. It is first important to describe neural networks in general and the network architectures of interest for this analysis.

4.1 Deep neural networks

A neural betwork, or more precisely an artificial neural network (ANN), is a computational model based on studies of animal brains [17]. Dr. Robert Hecht-Nielsen, a pioneer in the development of neural networks, describes an ANN as a "...computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs." [18]. They are constructed using basic artificial neurons that carry and modify weights and biases as observations are injected into the system (Fig. 6).

It can be difficult for users to understand how the network is performing and why they work so well, but the impressive performance of ANNs in computer vision, speech recognition, and in natural language processing is undeniable [19]. Key components to an ANN framework are layers and activation functions. Layers refer to a set of artificial neurons which can either be input layers, output layers, or hidden layers (layers between the input and the output layer). At each layer, there are activation functions applied to inputs. These functions are usually non-linear (such as a logistic function) and thus are essential to 'learning' robust features about the data.

Machine learning algorithms such as ANNs can be categorized as supervised or unsupervised learning. In supervised learning methods the inputs have labels that identify the true class (signal or background) before the network is trained, whereas in unsupervised learning methods the labels are not known prior to learning. Here we will use supervised DNNs where "deep" signifies that the network has any hidden layers in between the input and output layers. This allows for more complex learning of the data. Of course

more complexity means more computation time and creates risks of overfitting, where the generality of decision boundaries are lost and the results of the network are too specific to the exact learning sample. The types of layers used and the depth of the network must be examined when constructing a deep network to achieve optimal performance that is still generalizable to other datasets. Two DNN architectures have been implemented for jet-image studies: fully connected layer ("MaxOut") networks and convolutional networks.

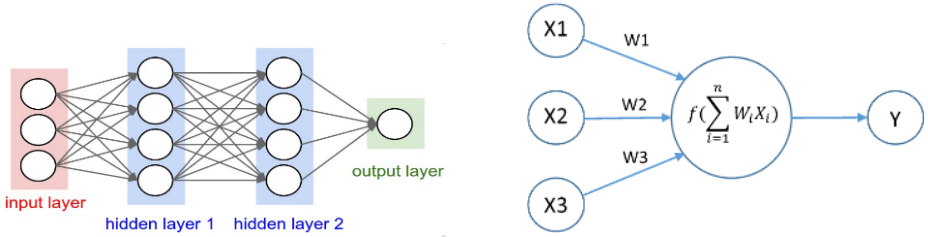


Figure 6: Diagrams of ANN architectures, with layers in which an activation function acts on inputs to create an ouput.

4.2 MaxOut (Fully Connected Layer) Networks

A fully connected machine learning layer takes all available variables, or "features", as inputs and every neuron is connected to every neuron in the previous layer. At each layer an activation function can be applied to create non-linear decision boundaries that detect the features of the input. A MaxOut [20] architecture uses fully connected layers with the MaxOut function, which is essentially a piecewise linear approximation of a convex function. The MaxOut function takes an input vector $\vec{x}$ and computes an output vector z using k linear weights w and offset terms b:

$$\vec{z}_{j\in[1,k]} = \sum_i \vec{x}_i w_{ij} + b_j, \tag{1}$$

with final output taken as $\max_{j\in[2,k]}(\vec{z}_j)$. A Rectified Linear Unit (ReLu), a piecewise function that is linear above zero and zero otherwise, can be considered a special case of a MaxOut activation function with $k = 2$. The Maxout is highly flexible since it can incorporate any number of piecewise linear functions.Maxout was orginally developed to maximize the effects of

"Dropout". Dropout is a function which reduces overfitting to training data by randomly turning off a percentage of neurons in a given layer. Dropout and Maxout pair nicely to reduce overfitting and to optimize learning [20]. The DNN implemented with MaxOut is structured as follows [16]:

- 1st layer (Input Layer): MaxOut fully connected layer with 256 neurons, random initialization of weights, k=5 , and a dropout rate of 30%.
- 2nd Layer: Maxout fully connected layer with 128 neurons, random initialization of weights, k=5, and a dropout rate of 20%.
- 3rd Layer: ReLu fully connected layer with 64 neurons, and a dropout rate of 20%.
- 4th Layer: ReLu fully connected layer with 25 neurons, and 30% dropout rate.
- 5th Layer (Output Layer): Sigmoid fully connected layer with 1 neuron.

An example of the network output can be seen in Fig. 7, showing the correlation of the network output to pixel activation in the inputs. This image shows a strong correlation in the location of the secondary energy deposit, expected to be signficantly more pronounced in W boson jets than in background jets.

4.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) [21] applies convolution filters, or kernels, with sets of weights w that operates on small $n \times n$ windows of an image array. This filter is moved through the input image in $n \times n$ steps and creates a convolved feature map with outputs $z = sigmoid(\vec{x}w)$. The feature map indicates local dependencies in the input image. There are three components that determine the size of the feature map and the performance of the CNN: depth, "stride", and zero padding. Depth refers to the number of filters used to for the convolution operation, which can be thought of as stacking 2D matrices. Stride is number of pixels by which the window moves as it convolves the image, and zero padding adds zeros on the borders of the image, allowing kernels to be applied on border segments

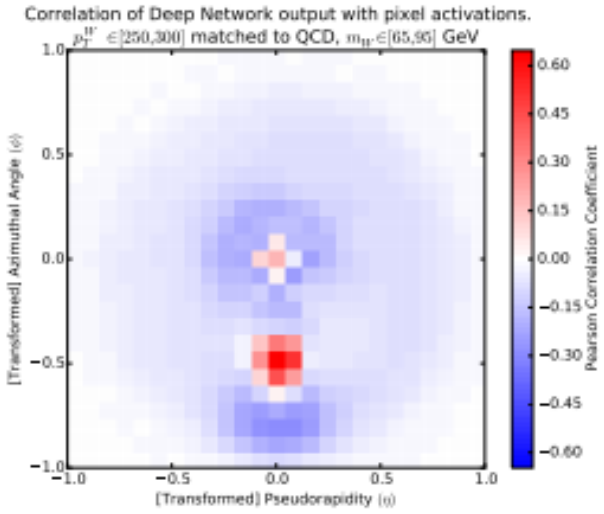


Figure 7: Example of the type of information contained in the MaxOut network. [16]

as well as central segments. After the convolution step, a ReLu activation function is applied to introduce non-linearity and create a "Rectified" feature map, which is a transformation of the orginal feature map. Then a smaller filter, called "MaxPooling", takes non-overlapping windows of the rectified feature map as inputs to reduce the number of dimensions in each feature map, while retaining the important information by taking the largest element of the rectified feature map. The convolution layer, ReLu activation layer, and Max Pooling layer work together as convolutional unit, shown in Fig. 8.

To learn the jet-images with a CNN, Oliveira et al. used three convolutional units with window sizes 11×11, 3×3, and 3×3. Each window has a depth of 32, stride of 1×1 (i.e., the window moves up-down and left-right one pixel at a time), and zero padding is used. The window for Max Pooling for each convolutional layer is 2×2, 3×3, and 3×3 respectively. All convolutional layers are regularized with the L2 weight matrix normalization. Since the ReLu can yield arbitually large outputs, Local Response Normalization is used to rescale the outputs of the three convolutional units. These are then followed by two fully connected layers to reconnect the output of the convolutional layers. The outline of their network is as followes [20]:

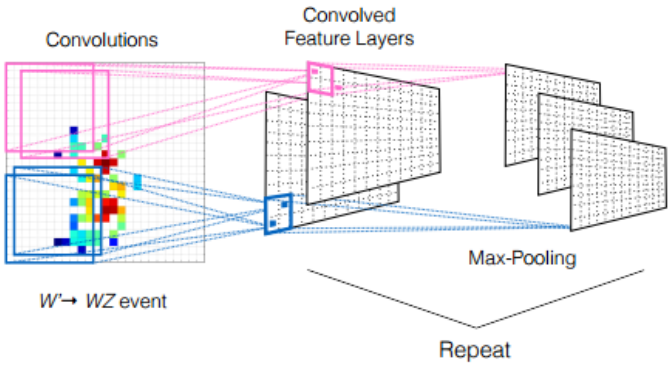


Figure 8: Convolutional Filters [16]

- 1. [Convolution → ReLu → MaxPooling] * 3 units
- 2. Output of 1 → Local Response Normalization
- 3. Output of 2 → [20% Dropout → fully connected → ReLu]
- 4. Output of 3 → fully connected layer and 10% Dropout
- 5. Output of 4 → Sigmoid fully connected layer

Figure 9 shows Receiver Operating Characteristic curves produced by Oliveira et al., demonstrating the gains possible by using deep networks to discriminate between heavy particle jets and multijet background jets compared to a traditional variable τ_2/τ_1. A combination of computer vision networks and neural networks can allow for new perspectives in studying jet substructures.

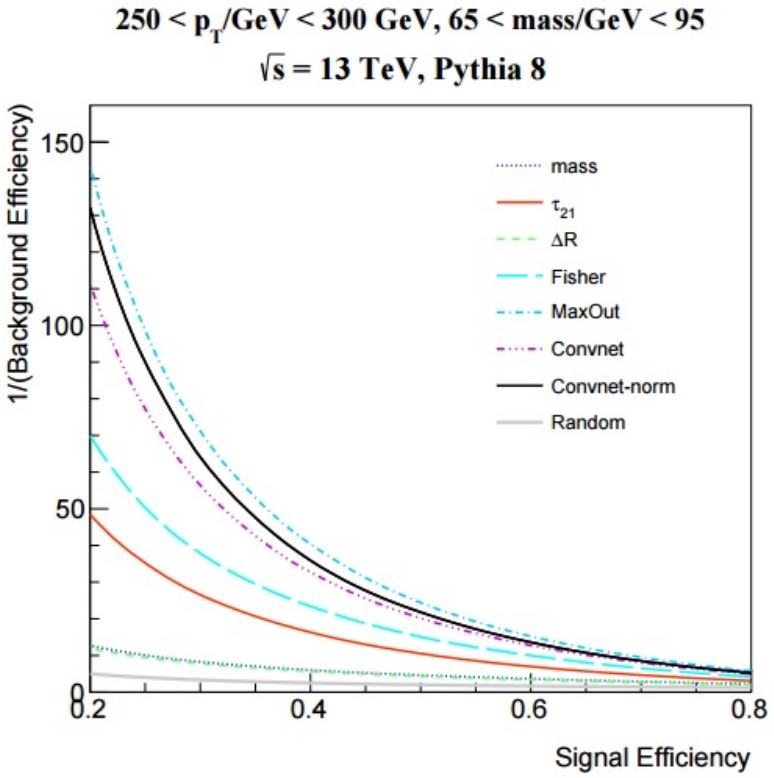


Figure 9: ROC curve of $W' \to WZ$ vs QCD. [16]

5 Jet Images at the High Luminosity LHC

The LHC is expected to generated approximately 50 petabytes of data in 2017 [22]. This is an enormous volume of data to analyze, so particle physics employ as many advanced computing tools at their disposal as possible. Machine learning tools provide robust analysis options and can give insight not seen with traditional algorithms. Boosted Decision Trees, ANNs, and Support Vector Machines have already been employed in high energy physics analysis because of their promising performance [23, 24, 25]. The development and the improvement of these tools helps identify physics objects and look for interesting physical phenomena.

As the LHC moves into its next phase with higher luminosity than ever before, there will be an increase in the number of simultaneous proton-proton

collisions, called "pileup". The number of average collisions per crossing of the LHC proton beams will increase from the current value of approximately 35 – 50 interaction up to 200! This increases the combinatorial complexity and rate of mis-reconstructed charged particle trajectories, and adds extra energy to jets from particles that did not originate in the same collision. These extra particles can distort the measurements of jets from boosted particles, such as the mass or subjet directions, decreasing the effectiveness of traditional search variables. For deep learning jet-images techniques to be used realistically in the LHC environment the response to the presence of extra soft particles originating from pileup is critical to study.

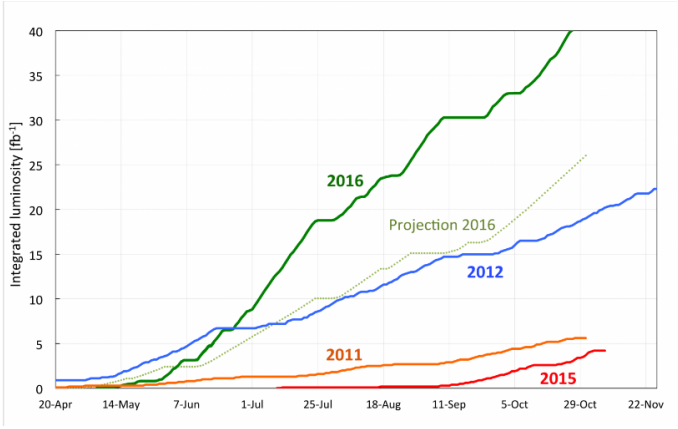


Figure 10: The LHC in 2016 has surpassed expectations and trampled past runs for integrated luminosity.

5.1 Simulations and Jet Image Creation

We have conducted studies on the performance of the jet imaging DNNs when a high number of pileup interactions have been incorporated into the simulation of the collisions. Boosted W signal and QCD background simulations were produced using Pythia 8.170 at $\sqrt{s} = 13$ TeV. Information about the generated particles in the $W' \to WZ$ and multijet decays are stored in the Les Houches Event format [26], a standard format that can be used by many programs such as Delphes. Delphes [27], a fast multipurpose detector simulator, is used to simulate various levels of pileup and cluster jets from calorimeter energy deposits. In Delphes we defined a calorimeter grid rang-

ing from [-2.4276, 2.4276] in η and $[-\pi, \pi]$ in ϕ. Each cell or tower of this calorimeter has a size $\eta \times \phi = .0714 \times .0714$. The baseline performance of the deep networks is tested by running Delphes with zero pileup interactions.

Other pileup configurations (35, 140, and 200 interactions) are also produced to test the effects of pileup on the performance of the classifier. Jets are clustered in Delphes using the anti-k_T algorithm with a characteristic radius of 0.8 (guiding the choice of the grid size for the calorimeter). The effects of pileup are mitigated in the Delphes simulation by applying some established tools for pileup subtraction: charged particles not originating from the same collision as the jet are subtracted because they can be tracked to a vertex, and the effect of neutral pileup particles is reduced by subtracting energy based on the opening area of the jet [28, 29]. Various grooming algorithms such as soft drop are applied on the jets and the N-subjettiness variables are computed.

After producing jets in Delphes the following selection criteria are applied: $200 < p_T < 300$ and $65 < m_{jet} < 95$. If there is more than one jet in the event passing these criteria, the highest p_T jet is analyzed. Since the calorimeter grid has edges at $\eta = \pm 2.4276$, jets which are too close to these edges for a 25×25 grid of calorimeter cells to be created around their center are rejected. Calorimeter tower cells with transverse energy deposits from jet constituents are stored. The angular distance in $\Delta\eta$ and $\Delta\phi$ between the the two leading subjets is calculated from two types of subjets: the subjets identified by the soft drop algorithm, and the primary and secondary axes of the jet identified using the PCA technique, where a covariance maxtix is created from all jet constituents describing their variation from the mean in η and ϕ. If the soft drop algorithm identifies two subjets, the angle between them is used to rotate the jet images. Otherwise the rotation angle is calculated from axes identified with the PCA technique. The ratio of N-subjettiness variables $\tau_{21} = \tau_2/\tau_1$ is also stored for comparison against the deep network results.

5.2 Image Processing

Python 3.4.8 with the modules Keras 2.0.3 [30], Theano 0.9.0 [31], Scikit-Learn 0.18.1 [32], and Scikit-Image 0.13.0 [33] were used for pre-processing and to create DNN architectures. Keras is a deep learning library that provides simpler user access for the Theano or Tensorflow machine learning programs, making it straightforward to create different types of deep networks. Scikit-Learn is machine learning library which we use to do validate

network performace and visualize the results of training a deep network, such as ROC curves. Scikit-Image is an image processing library which we use to perform the rotations and parity inversion of the jets.

Following the framework developed by Oliveria et. al, images are processed according to the steps described in section 2. The jet images are binned using the calorimeter towers. No normalization the pixel intensity is preformed because can distort information contained within the jet-images as the mass represented by the energy in the pixels would not be conserved. The images are rotated through the angle described previously so that the secondary subjet is located at $-\pi/2$.

Example average jet-images for the W boson jets and background jets at each pileup level are shown in Figs. 11 – 14, before and after the rotation procedure. The difference between the images as pileup increases is clear: with higher pileup there are more low-momentum energy deposits in the calorimeter. Visually, one can see that the two subjets become less distinct with higher pileup and that the intensity in pixel far from the hard clusters. In the W boson jets the second subjet is well separated from the primary subjet since there are two unique quarks producing subjets. In the background jets the second subjet is less distinct and blends into the soft energy deposits at high pileup because the main souce of subjets in these jets is random splitting of gluons into two quarks. Therefore on average the secondary subjet of the QCD background is not as prominent as the secondary subjet of the boosted W boson.

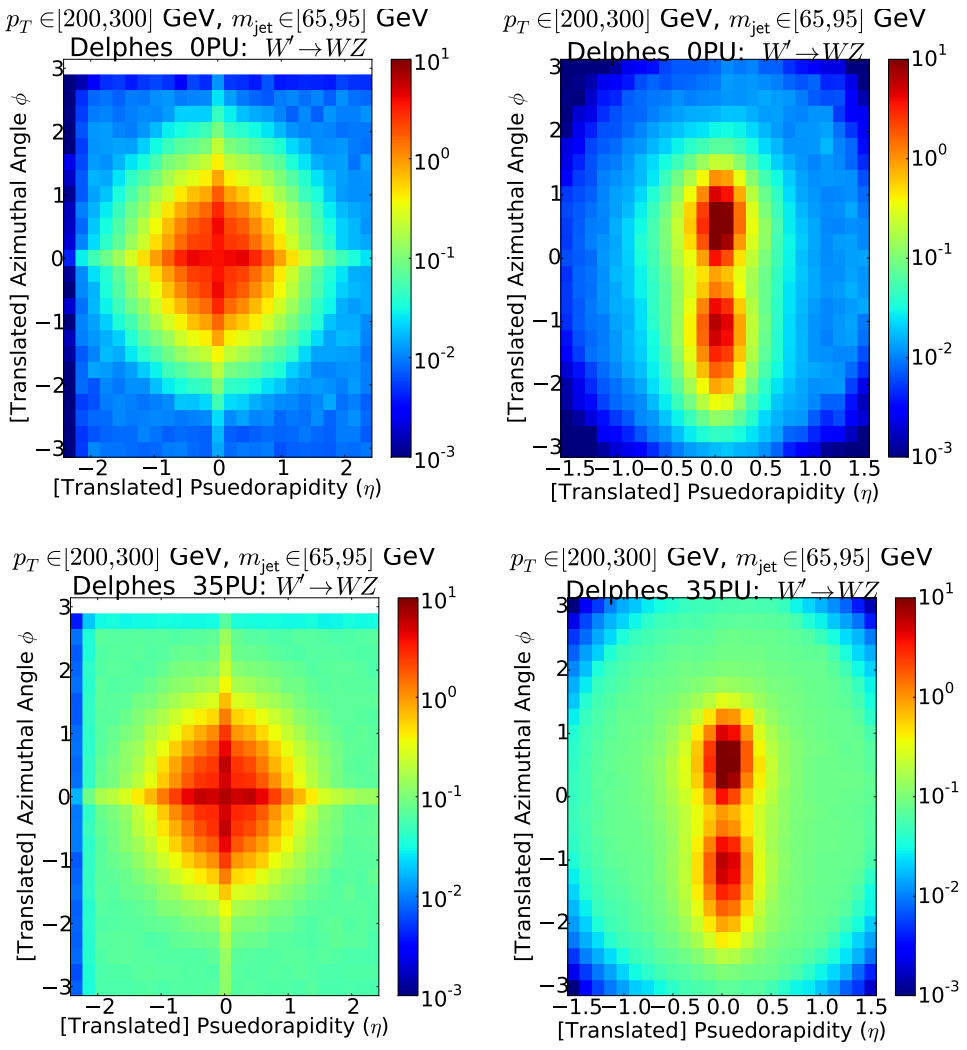


Figure 11: Average jet-image of signal jets with 0 (top) and 35 (bottom) pileup interactions, before (left) and after (right) rotation.

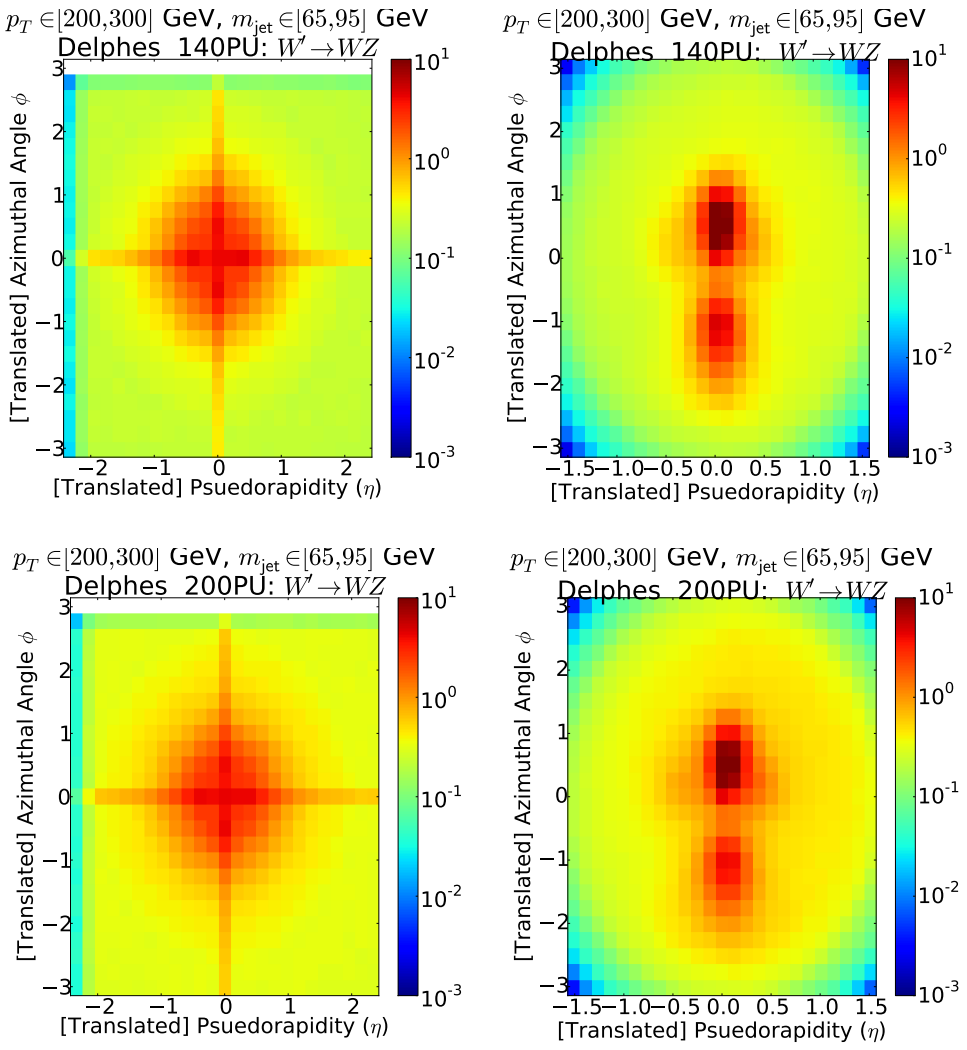


Figure 12: Average jet-image of signal jets with 140 (top) and 200 (bottom) pileup interactions, before (left) and after (right) rotation.

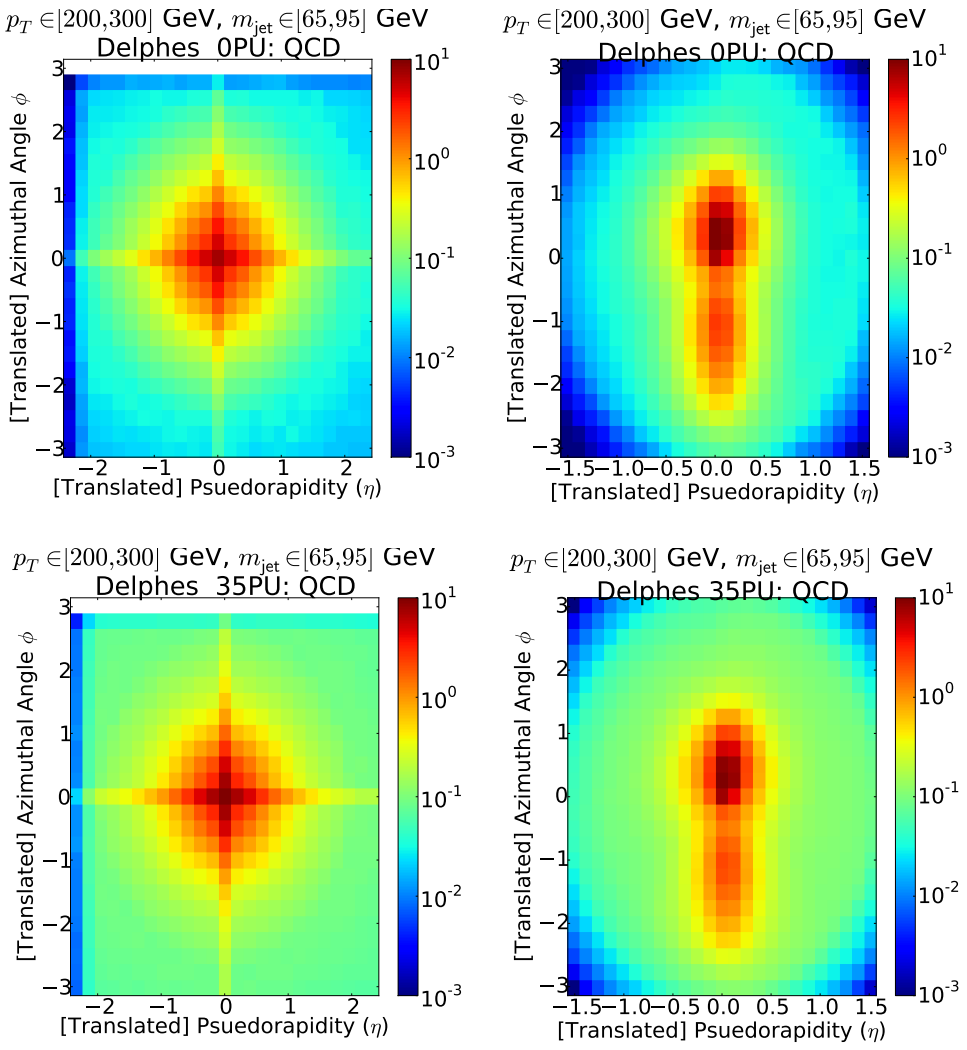


Figure 13: Average jet-image of background jets with 0 (top) and 35 (bottom) pileup interactions, before (left) and after (right) rotation.

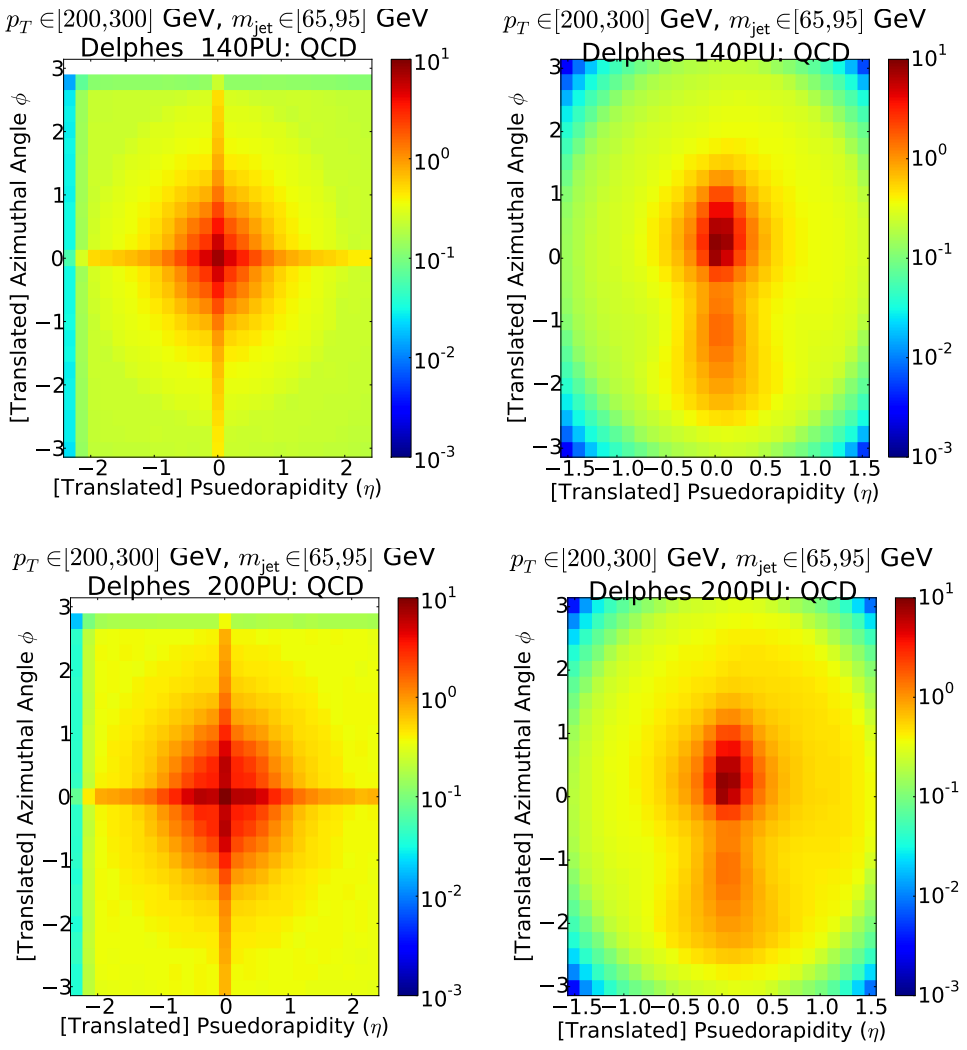


Figure 14: Average jet-image of background jets with 140 (top) and 200 (bottom) pileup interactions, before (left) and after (right) rotation.

5.3 DNN Framework

Due to software availability we focused on building and training a MaxOut network which was shown in Fig. 9 to perform slightly better than a CNN. To estimate the impact of pileup on the MaxOut network, we trained the network using 3-fold cross-validation on 180,000 jet-images from the 0-pileup samples. Inputs are weighted so that the p_T distribution of signal jets matches the distribution of background jets. Cross-validation is performed by training the network and evaluating its performance in a set number (k) of "folds", or iterations: in each iteration a fraction $1/k$ of the inputs are saved for testing and the remaining inputs are used for training. At each fold the training inputs are shuffled and up to 50 iterations of training are performed, stopping when the area under the ROC no longer increases. In this way the entire sample is used to evaluate the network in small independent segments, reducing the dependence of the network's performance on the exact input sample. The cross-validation sample contains 22% signal jets and 88% background jets.

The network trained on the 0-pileup samples is then also tested on 210,000 jet-images with 35 pileup (17% of which are signal jets), 220,000 jet-images with 140 pileup (12% of which are signal jets), and 190,000 jet-images with 200 pileup (11% of which are signal jets). Of the three sets of network outputs from the cross-validation, we chose the most conservative to use in these evaluations. We also trained three different maxout networks on jet-images with 35, 140, and 200 pileup. These networks were tested during the cross-valdiation procedure on input jet-images with the same number of pileup as was used to train them. In Fig. 15, the distributions for the MaxOut Networks trained and tested with the sample pileup are shown. The network exhibits noticeable jumps around .5-.6 for 0 and 35 pileup and around .4-.5 for 140 and 200 pileup. At this middle junction, the MaxOut Network has trouble in distinguishing between signal and background. This is also seen in the Oliveria et al. paper [16].

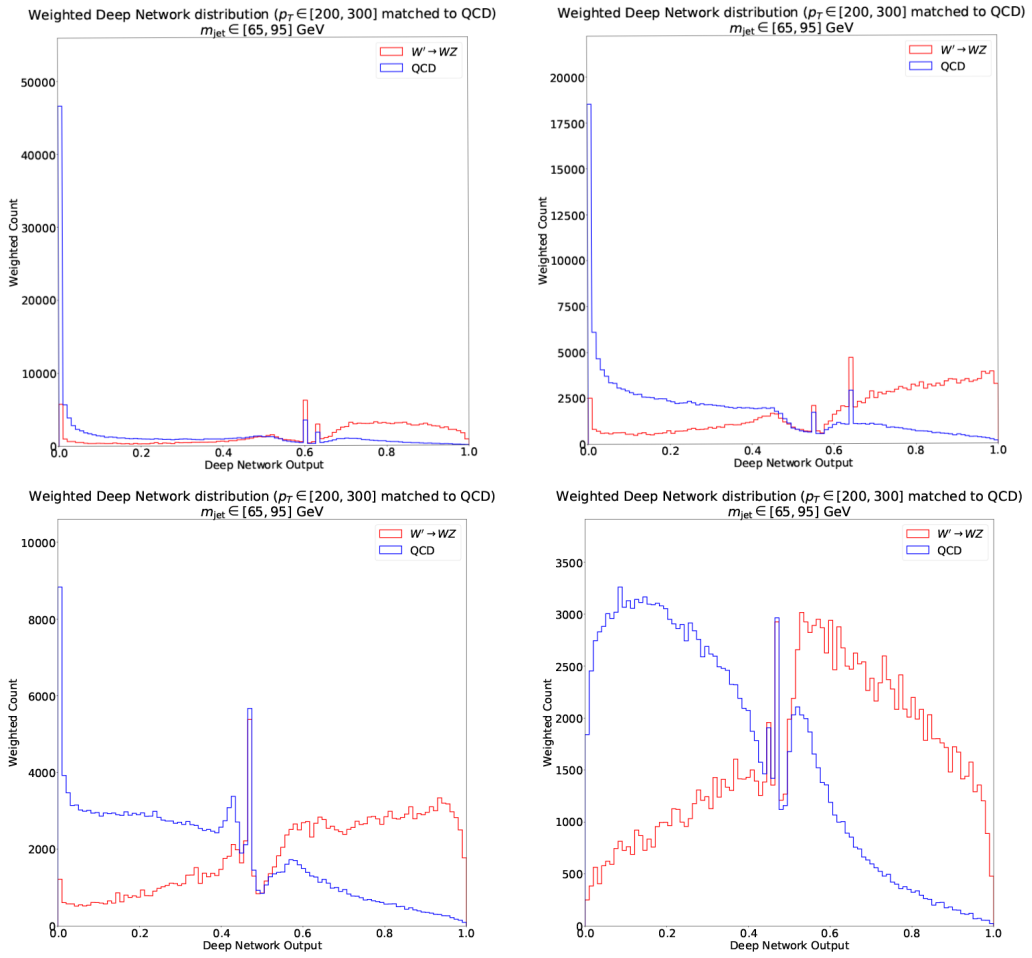


Figure 15: The network outputs are compared to signals of the same pileup. Top-left is 0 pileup, top-right is 35 pileup, bottom-left is 140 pileup, and bottom-right is 200 pileup.

As pileup increases, the extra radiation in the jets causes signal and background jet-images to look more similar because there is enough extra energy deposits to make the images "hazy." The boosted W signal has a much "brighter" secondary subjet than the background jet. When the network learns the features of the 0 pileup images it must have a strong focus on the secondary subjet which is weak in the QCD case.Fig. 16 shows that the Max-Out network has a strong linear correlation for the subjets, and arguable more so for the secondary subjet. It is also important to notice the anti-correlation around the subjets, especially in-between them. The anti-correlation of the network output with activity at the fringes of the jets decreases with higher pileup, thus more pixels loose their significance towards learning the features of the jets because these pixels are typically filled with radiation from pileup. Higher detector interactions are also causing the correlation of the secondary subjet for 140-pileup and 200-pileup to become more spread out. This wider area of correlation means the resolution of the secondary subjet is decreasing. The network is considering the energy outside of the subjet as important as well.

Further evidence for the importance of the subjets is shown with correlation plots of a Maxout network only trained on 0-pileup. The difference is that the correlation of all pixels decreases significantly with more pileup, though the network continues to favor the secondary subjet. At higher pileups, this second leading jet becomes more hidden by soft radiation and the network is unable to distinguish signal from background as clearly.

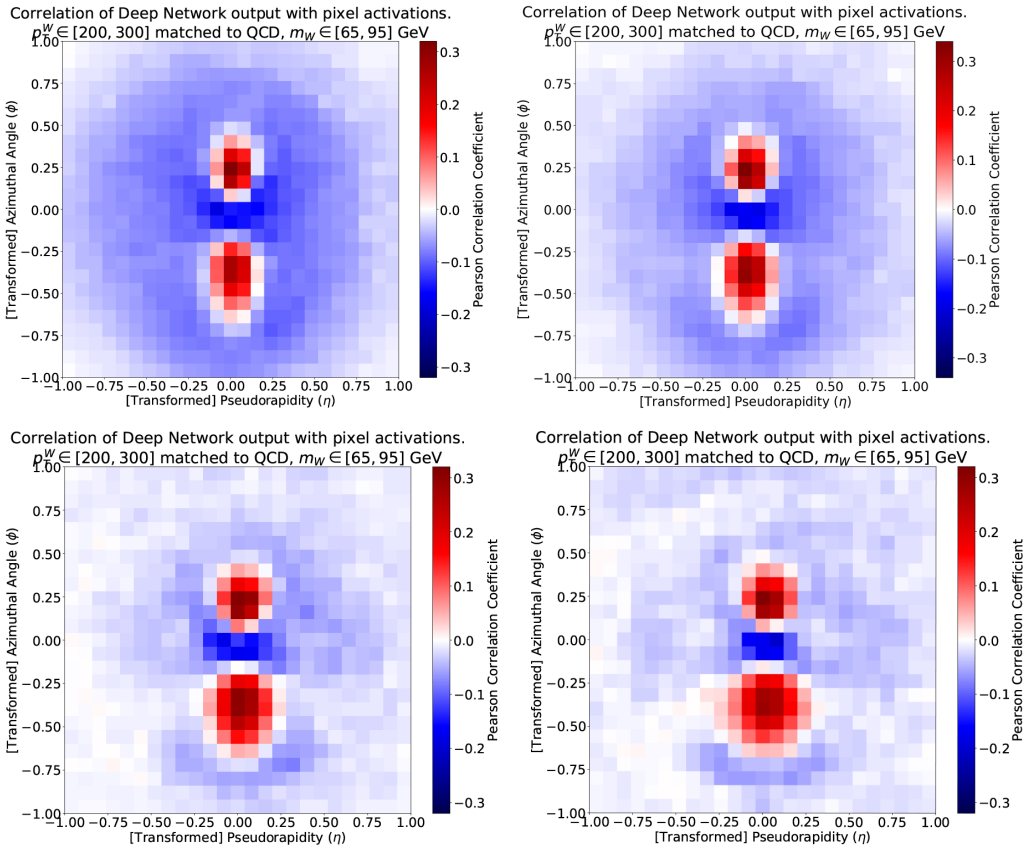


Figure 16: Correlation between with the intensity of each pixel to the Max-Out network output.The network outputs are compared to signal and background of the same pileup. Top-left is 0 pileup, top-right is 35 pileup, bottom-left is 140 pileup, and bottom-right is 200 pileup.

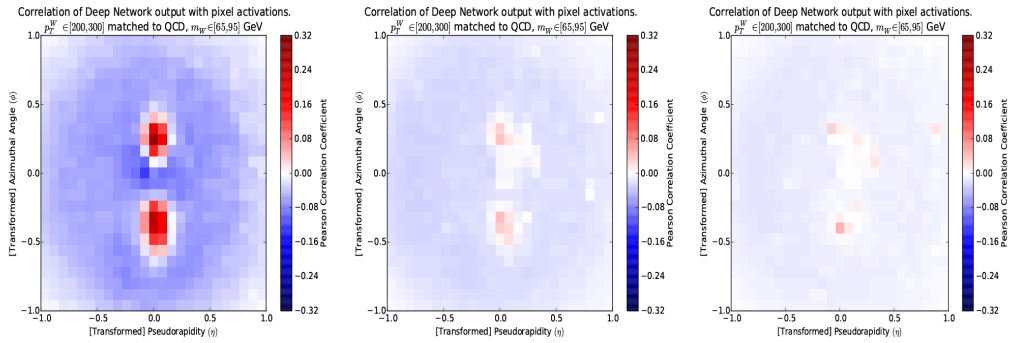


Figure 17: Correlation between with the intensity of each pixel to the MaxOut network output. This MaxOut network was only trained on 0 pileup and then tested with 35, 140, and 200 pileup (in that order from left to right).

5.4 Signal and Background Distributions

In Fig. 18, ROC curves show the signal effciency versus background rejection of the MaxOut networks. Using the network trained with 0-pileup jet-images ato predict the idenity of jets in the higher pileup samples shows very poor performance. However, when training networks with high pileup jet-images, performance is recovered to the extent that the network trained on 140 pileup jet-images nearly reproduces the performance at current LHC conditions with 35 pileup. Even in the extreme conditions of 200 pileup the network is able to isolate important features of the jets and provide meaningful discrimination.

Signal Effciency	0 pileup	35 pileup	140 pileup	200 pileup
25%	2.1%	2.6%	2.7%	3.0%
50%	6.8%	8.3%	9.5%	10%
75%	19%	23%	27%	30%

Table 1: Background effciencies from the ROC cruve describing MaxOut networks trained and tested with the same pileup

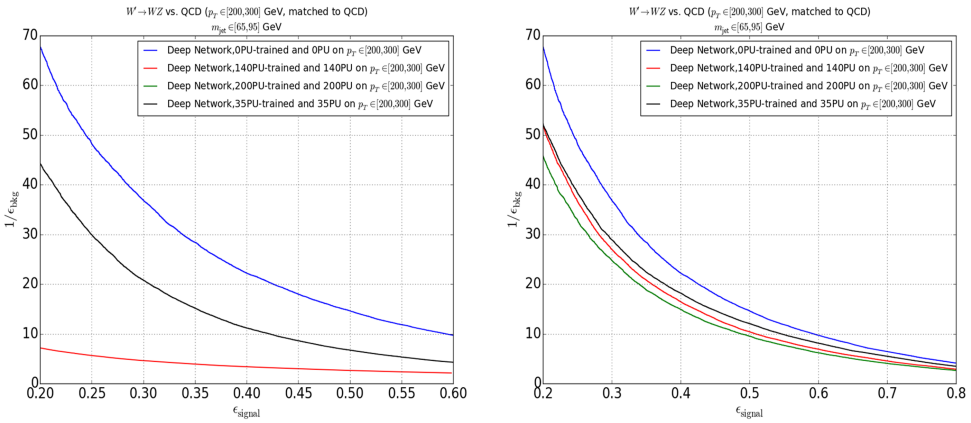


Figure 18: ROC curve comparing the 0-pileup trained MaxOut networks applied on jet-images with 0, 35, 140, and 200 pileup interactions (left). However, 200 pileup does not show on the graph because the network is unable to reject background. ROC curves comparing MaxOut networks trained and tested with the same pileup (Right)

6 Conclusion

The combination of visual learning and deep learning is an interesting approach to understanding the physics of jets. It offers new possible perspectives that may not be obvious with current algorithms focused on traditional physics variables. Thus our examination of its performance on simulated data representing more realistic detector conditions is important. We can conclude from this initial study that any algorithms or networks used for jet identification in high pileup conditions will need to be specially trained for the pileup conditions, and will likely need training updates if the average pileup changes over time, or performance will be lost. We see that an increase in pileup diminishes the network's ability to differentiate signal from background, however dedicated training shows that the effects of pileup can mitigated in situ by the network to large extent. One of the goals of perforamce for the High Luminosity LHC is replication of current performace levels and we see that this achieved with the deep network trainings, especially for the case of 140 pileup. To expand this study in the future, we would like to increase the magnitude of training samples, investigate diverse network architectures, and quantify performance of the network compared

to other variables or other methods of pileup mitigation.

References

[1] Lyndon Evans and Philip Bryant. LHC Machine. *JINST*, 3:S08001, 2008.

[2] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_{t} jet clustering algorithm. *JHEP*, 04:063, 2008.

[3] Martin Schmaltz and David Tucker-Smith. Little Higgs review. *Ann. Rev. Nucl. Part. Sci.*, 55:229, 2005.

[4] Andrea De Simone, Oleksii Matsedonskyi, Riccardo Rattazzi, and Andrea Wulzer. A First Top Partner Hunter's Guide. *JHEP*, 1304:004, 2013.

[5] Serguei Chatrchyan et al. Observation of a new boson with mass near 125 GeV in pp collisions at $\sqrt{s} = 7$ and 8 TeV. *JHEP*, 06:081, 2013.

[6] Serguei Chatrchyan et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett. B*, 716:30–61, 2012.

[7] Andrew J. Larkoski, Simone Marzani, Gregory Soyez, and Jesse Thaler. Soft Drop. *JHEP*, 05:146, 2014.

[8] S.D. Ellis, C.K. Vermilion, and J.R. Walsh. Techniques for improved heavy particle searches with jet substructure. *Phys. Rev. D*, 80, 2009.

[9] David Krohn, Jesse Thaler, and Lian-Tao Wang. Jet Trimming. *JHEP*, 02:084, 2010.

[10] Jesse Thaler and Ken Van Tilburg. Maximizing Boosted Top Identification by Minimizing N-subjettiness. *JHEP*, 02:093, 2012.

[11] Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwartzman. Jet-Images: Computer Vision Inspired Techniques for Jet Tagging. *JHEP*, 02:118, 2015.

[12] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[13] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *JHEP*, 07:079, 2014.

[14] M. Bahr et al. Herwig++ Physics and Manual. *Eur. Phys. J.*, C58:639–707, 2008.

[15] Torbjörn Sjöstrand, Stephen Mrenna, and Pẽter Skands. A brief introduction to pythia 8.1. *Comput. Phys. Comm.*, 178, 2008.

[16] Luke de Oliveira, Michael Kagan, Lester Mackey, Benjamin Nachman, and Ariel Schwartzman. Jet-images — deep learning edition. *JHEP*, 07:069, 2016.

[17] J.A. Leonard M.A. Kramer. Diagnosis using backpropagation neural networks—analysis and criticism. *Computers and Chemical Engineering*, 14:1323–1338, 1990.

[18] Shelly Ehrman. Dr. Robert Hecht-Nielsen Joins KUITY Corp. Scientific and Technology Advisory Board. http://www.marketwired.com/press-release/dr-robert-hecht-nielsen-joins-kuity-corp-scientific-and-technology-advisory-board-1645011.htm, 2017.

[19] Michael Nielson. Neural networksand deep learning. `http://neuralnetworksanddeeplearning.com/about.html`, 2017.

[20] I. J. Goodfellow, D. Warde-Farley, A. Mirza, M. andCourville, and Y. Bengio. Maxout Networks. *arXiv e-prints*, abs/1302.4389, February 2013.

[21] An Intuitive Explanation of Convolutional Neural Networks. `https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`, 2016.

[22] WLCG Collaboration. Worldwide lhc computing grid. `http://wlcg.web.cern.ch/`.

[23] Y. Coadou. Boosted decision trees. `https://indico.cern.ch/event/472305/contributions/1982360/attachments/1224979/1792797/ESIPAP_MVA160208-BDT.pdf`.

[24] Andreas Hoecker, Peter Speckmayer, Joerg Stelzer, Jan Therhaag, Eckhard von Toerne, and Helge Voss. TMVA: Toolkit for Multivariate Data Analysis. *PoS*, ACAT:040, 2007.

[25] Mehmet Özgür Sahin, Dirk Krücker, and Isabell-Alissandra Melzer-Pellmann. Performance and optimization of support vector machines in high-energy physics classification problems. *Nucl. Instrum. Meth.*, A838:137–146, 2016.

[26] Johan Alwall et al. A Standard format for Les Houches event files. *Comput. Phys. Commun.*, 176:300–304, 2007.

[27] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *JHEP*, 02:057, 2014.

[28] Matteo Cacciari and Gavin P. Salam. Pileup subtraction using jet areas. *Phys. Lett.*, B659:119, 2008.

[29] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The Catchment Area of Jets. *JHEP*, 04:005, 2008.

[30] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[31] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[33] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.