Using a Convolutional Neural Network for Tagging of Jets Produced During Controlled Proton-Proton Collisions

Shameer Ahmad

10/01/2022

Physics and Astronomy Research Laboratories
University College London
London, United Kingdom

Abstract

Identifying the type of jet produced in a hadron collider is difficult without computational aid. The images are often very similar to each other. Thus, Convolutional Nueral Networks (CNNs) can be trained to classify jet images. For the CNN mentioned in this notebook, the best test loss and accuracy was 0.4814 and 0.7968, respectfully. The areas under the curves of the ROC and PR curves were 0.865 and 0.845, respectfully. The F1-score was 0.796.

Contents

1 Introduction

The Larger Hadron Collider (LHC) is the largest particle accelerator and collider in the world. The LHC produces 2 beams of protons and sends them in opposite directions around a circular collider (radius of 4km). It then, using strong magnets, collides the beams at 8 different particle detectors. This collisions are called "Proton-Proton" collisions. The collision will create an "explosion" of different particles. Partons are produced which are quarks and gluons which hadronize into a stream of hadrons. This stream is in the shape of a narrow cone and is called a jet. The proton-proton collision also produced W-bosons and H-bosons.

These are heavy particles and so will decay into lighter particles. The W-boson decays into a quark and antiquark (up or down), therefore hadronization occurs and 2 jets are produced . The H-boson decays into a bottom-quark and bottom-antiquark, therefore hadronization occurs and 2 jets are produced. Another one of the particles produced by the collision is a top-quark. This is also a heavy particle. It first decays into the W-boson and bottom-quark. The W-boson is also a heavy particle and therefore decays into quark, antiquark (up or down) and a bottom-quark. So, three types of quarks are produced and therefore three jets.

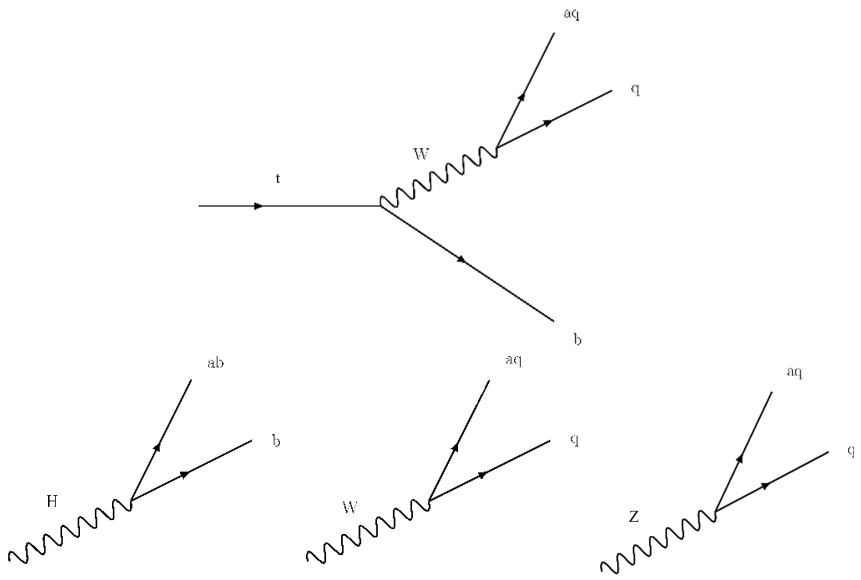

Figure 1: Top-quark, Z-boson, H-boson and W-boson decay Feynman diagrams. These decay into quarks which will produce a stream of hadrons called jets.

The multiple jets produced by the top-quark, W-boson and H-boson are collectively called fat jets. Therefore, jets are (mainly) of 3 types; a QCD jet (jets formed directly from partons), W jet (jets formed from W-decay) and Top-Quark Jets (jets formed from Top-decay). QCD stands for Quantum-Chromo Dynamics and it is the theory that describes quarks and their hadronization. There are also other fundamental particles produced by the proton-proton collision. The fundamental particles that can't produce jets are: e, $\mu, \tau, \gamma, \upsilon_{e,\mu,\tau}$. The fundamental particles that can are: u, d, s, c b, g, t, W, Z, H [1]. See [1] for names of symbols.

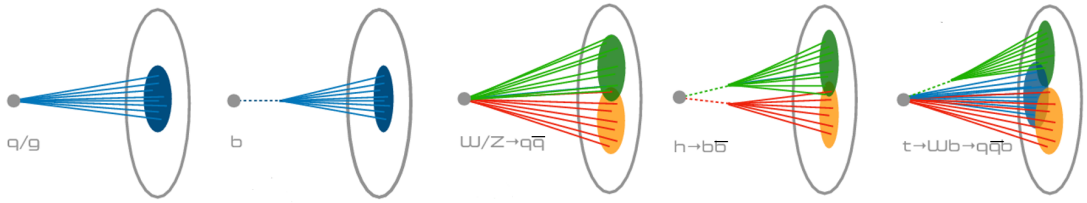

Figure 2: [2] From left to right, illustration of jet produced directly from partons, jet produced from bottom-quark, fat jet produced from W/Z-boson, fat jet produced from H-boson and fat jet produced from top-quark.

These energy of these jets are deposited in calorimeters within a detector at the LHC. After some processing, these energy deposits are displayed as an image that can display the jets produced. An example of such an image is displayed below. We can use machine learning techniques to distinguish between the types of jet. The most common technique is using "convolutional neural networks" (CNNs) by inputting a jet image and training the network to identify the type of jet. The identification of jets is called jet tagging [3]. A CNN is supervised type of deep learning algorithm which can classify an input image by taking small aspects/objects in the image and differentiate it from other aspects [3]. In this context, our CNN will be a tagger. Therefore, other classifications can be distinguished and images identified by their classification.

For this project, a CNN was designed to distinguish between W-boson jets (also called the "signal") and the QCD jets (also called the "background"). Therefore, this problem is a binary classification problem. The number of W-boson jets and QCD-jets, for this project, is 10,000 each. Figure 3 shows an image of a W-jet and a QCD-jet from the data used. The brightest pixels are (likley) the jets since jet have the highest energy due to their particle count. It is clear that the signal image has more than one detection. There are 2 brightest pixels thus 2 jets present although there is interference from other detections (non-jet products of the proton-proton collision). The background image clearly has one jet with interference from other detections. These images are more clear than others and will therefore be one of the easier images to identify for the CNN.

The jets are detected in terms of their transverse momentum (p_T) and, thus, each jet image is of a different p_T. There are billions of collisions happening in a single second in the LHC, therefore, there are billions of jets produced and detected [1]. This data is terabytes in size. Therefore, a sample of the data is used to train and test the CNN. The dataset is in a specific range of p_T.

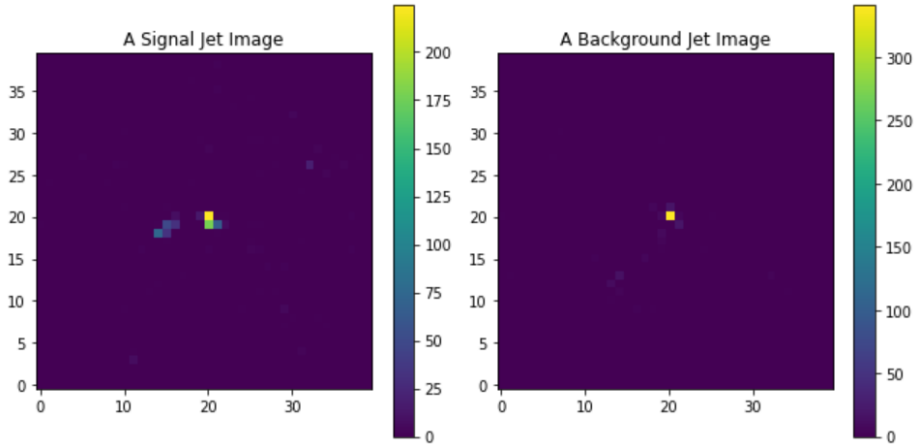

Figure 3: A W-jet (signal) and QCD-jet (background) image. the x-axis is the Pseudorapidity (η) and the y-axis is the azimuthal angle (ϕ) [3]. The intensity of the pixels / the colour-bar is the pixel transverse momentum (pixel p_T) (i.e. intensity).

2 Methodology

The image data were in the form of a .npz file. The images were separated into W-jets and QCD-jets and the labels were made. For QCD-jets, the labels were assigned to be 1 and for W-jets, 0. This represents the probability of being a background (QCD) jet; the background jets have 100% probability of being a background jet and signal jets have 0% probability of being a background jet. Thus, the CNN will be designed such that the output will be the probability of the input image being a background jet.

Image data had a shape of (20000, 40, 40, 1). This means that there are 20000, 40 pixels by 40 pixels images with 1 colour channel. The labels are 20,000 1s and 0s in a 1D column array corresponding to the images. Thereafter, the training and test data was split where the training data was 80% and the test data was 20% of the dataset. Furthermore, adding noise to the data made the performance worse due to the nature of the images. The jets are not large in the image and therefore a random distribution of noise throughout the 40x40 pixel will barely affect the "jet area" of the image.

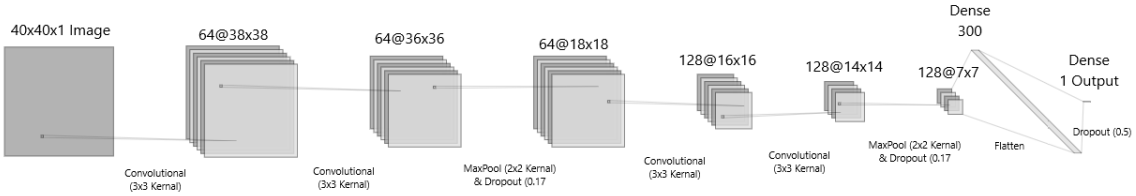

Figure 4: The CNN used to reduce an image of a jet to identify the class.

There a myriad ways to create a CNN for binary classification of images. However, the most common layers are the 'Conv2D' layer followed by the 'MaxPool2D' layer. They are convolutional and max-pooling layers, respectfully. The input image, as mentioned before, is 40x40 pixels with 1 colour channel. The first 2 layers of the CNN used are convolutional layers of 64 feature maps with a 3x3 kernel. These reduced image size to 38x38 then 36x36. Thereafter, max-pool layer was used with 2x2 reduction factor and a dropout layer with a rate of 0.17. Dropout layers were used throughout the network to reduce effects of overfitting.

To extract more features, 2 more convolutional layers with 128 feature maps and a 3x3 kernel are added. Another max-pool layer (2x2 reduction factor) with a dropout layer (0.17 rate) is used. This reduced the image size to 7x7. The last layer is flattened into a 1-dimensional layer and then a dense layer of 300 neurons with a dropout (0.5 rate) are applied. The final layer is a dense layer of 1 neuron. The output is be the probability of

the input image being a background jet.

The 2 activation functions used were ‘relu‘ and ‘sigmoid‘. ’ReLU’ was used with every ’dense’ and ’convolutional’ layer. ReLU stands for ”Rectified Linear Unit” activation function. After much trial and error, ReLu gave the best results for the non-output layers. It returns 0 for any negative inputs but returns the value back if positive. ’Sigmoid’ is applied to the last layer because it converts the last layer to output a probability distribution since it converts the inputs to a number between 0 and 1.

When compiling the model, the loss function used was binary crossentropy. Mean-Squared Error (MSE) was initially used but there were some disadvantages. It was sensitive to the network architecture. For instance, changes in dropout lead to a constant accuracy and loss. MSE is more appropriate for Gaussian random variables while binary crossentropy is more appropriate for binary classification [4].

The equation for binary crossentropy is [4]:

$$\mathrm{BC} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \tag{1}$$

where $\hat{y}_i$ is the label, y_i is the output probability of the CNN and N is the number of input images.

The equation for MSE is [5]:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2}$$

The RMSprop optimizer stands for Root Mean Squared Propagation and was found to perform better than the ’Adam’ optimizer (Adaptive Moment Optimization algorithms). Also, the learning rate was increased from the default 0.001 to 0.0018 for better performance. The optimizer configuration was determined through trial and error. The model was trained with a batch-size of 200 and for 30 epochs.

3 Results, Analysis and Discussion

The training accuracy of the data is increasing while the testing accuracy is "stalling". With more epochs, the training accuracy will increase beyond 0.95. The loss decreases for both the training and test data but stalls for the test data. However, the loss still remains high in contrast to when the MSE loss functions were used. This is due to crossentropy losses being more sensitive to misclassifications than MSE losses [6]. For this reason, every single extreme misclassification will exponentially increase the loss compared to the epoch before. This is why, below, losses will suddenly increase drastically. The lowest training and testing loss achieved was 0.3856 and 0.4814, respectfully. The highest training and testing accuracy accuracy was 0.8494 and 0.7968, respectfully. The plots are shown below:

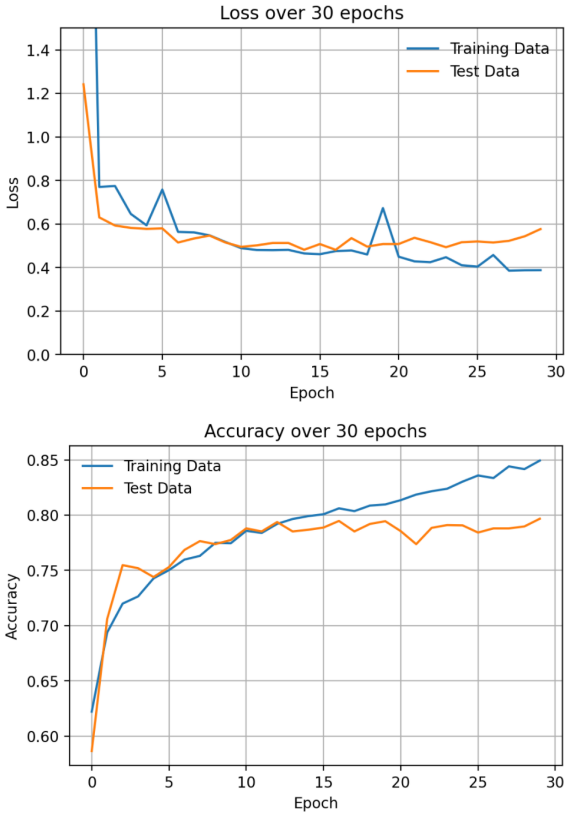

Figure 5: The loss and accuracy against epochs for the CNN.

The loss and accuracy of the training data is decreasing with every epoch and is would get better if continued. However, this is not the case with the test data due to overfitting. Different configurations and frequency of dropout was applied to the CNN yet the data kept

overfitting or underfitting (if too much dropout). Furthermore, adding noise to the data made the performance worse due to the nature of the images. The jets are not large in the image and therefore a random distribution of noise throughout the 40x40 pixel will barely affect the "jet area" of the image. Therefore, it is most likely that the minute dataset is the issue. The total dataset is only 20,000 images. This is small for CNN trying to differentiate a small number of pixels (which represent the jets) compared to a large image dataset like MNIST (where there are 60,000 images and the images have more pixels). A much larger dataset is needed.

Another way to evaluate the performance of a CNN is the Receiver Operating Characteristics (ROC) curve. It is a plot of the true positive rate (TPR- how often the model gets the right classification of the positive (in this case QCD jet) class) against the false positive rate (FPR- how often the model gets the wrong classification of the positive class). The ROC curve plots TPR against FPR for different thresholds. The threshold is the probability at or beyond which the result will be classified as positive. The rejection-efficiency curve was also plotted which is similar to the ROC. Rejection is 1/FPR and efficiency is just TPR.

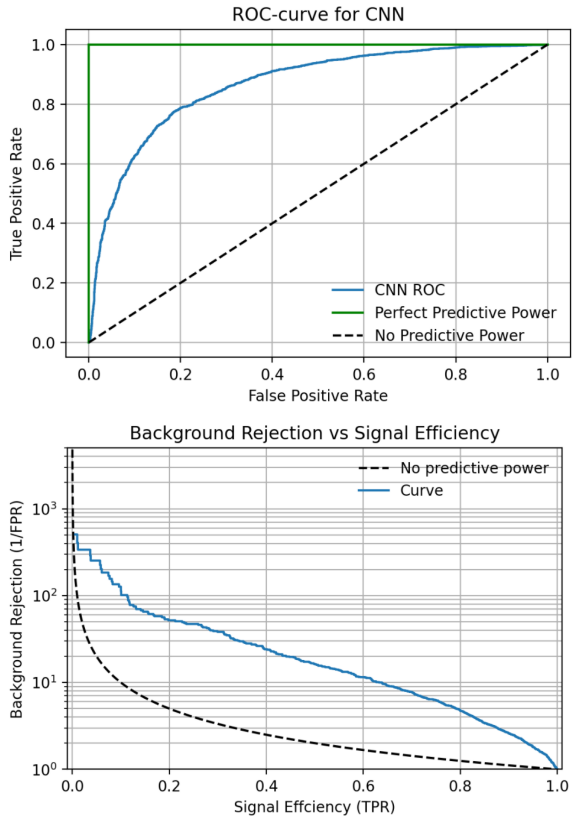

Figure 6: ROC and Rejection-Efficiency. The greater the area, the better the performance.

From this, it is clear that if the tagger (CNN) efficiency is 80% (ie the TPR is 0.8), then the tagger will get 1/5 classifications incorrect (ie the FPR is 0.2). However, if we increase the efficiency to 90%, then the tagger will get about 2/5 misclassification and a lower background rejection. Therefore, we can see that increasing the efficiency by about 10% will double the number of misclassifications. Thus, there is a compromise and a balance is needed. Perhaps a tagger efficiency of 0.6 is ideal. For both plots, the curve is closer to the perfect curve than the straight line. This shows good predictive power which is reinforced by an AUC (area under the curve) of 0.865. However, it is not anything exceptional. This is mainly due to, again, the lack of training data. A much larger data set is needed.

A Precision-Recall curve is plotted to further assess performance. Precision is the rate at which the CNN gets the positive result correctly when considering all the positive predictions. Recall is simply the TPR.

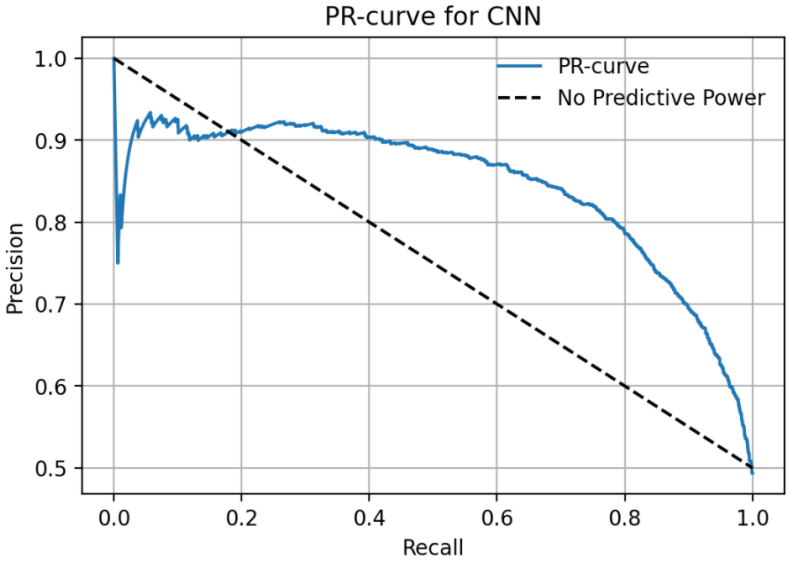

Figure 7: ROC and Rejection-Efficiency curves for the model.

The curve appears to have a high area. This shows good predictive power which is reinforced by an AUC of 0.845. However, it is not anything exceptional. This is mainly due to, again, the lack of training data. A much larger data set is needed. Also, there is a sharp drop at the beginning of the curve. This is due to drops in correct positive classification (ie rise is false positives) since precision decreases [3].

The F1-score (also called F-score) is the harmonic mean of the precision and recall and, therefore, is another performance metric [7]. The F1-score for this model is 0.796. Considering a perfect score is 1, this is a decent score. However, it's not exceptional for the reasons mentioned above.

4 Conclusion

The performance of this CNN was not ideal. The best test loss and accuracy was 0.4814 and 0.7968, respectfully. The areas under the curves of the ROC and PR curves were 0.865 and 0.845, respectfully. The F1-score was 0.796. To improve the performance, we must increase the number of images used to train the CNN.

Furthermore, we could randomly rotate the images for variability. Also, we could train for longer but this would use more computing power. Lastly, noise could be added to the center of the images; where the jet and other detections are. Thus, the main improvement should be variation in the images which can be achieved through adding more, adding noise to or rotating images [3].

5 Bibliography

References

[1] M. Strassler, "Jets: The Manifestation of Quarks and Gluons," Of Particular Significance, Oct. 20, 2011. https://profmattstrassler.com/articles-and-posts/particle-physics-basics/the-known-apparently-elementary-particles/jets-the-manifestation-of-quarks-and-gluons/ (accessed Jan. 2, 2022).

[2] E. A. Moreno et al., "Interaction networks for the identification of boosted H→bb¯ decays," Physical Review D, vol. 102, no. 1, Jul. 2020, doi: 10.1103/physrevd.102.012010.

[3] S. Macaluso, "Pulling Out All the Tops with Computer Vision and Deep Learning", NHETC, Dept. of Physics and Astronomy Rutgers, The State University of NJ Piscataway, NJ 08854 USA.

[4] "How to choose cross-entropy loss function in Keras?," knowledge Transfer, May 22, 2021. https://androidkt.com/choose-cross-entropy-loss-function-in-keras/ (accessed Jan. 10, 2022).

[5] C. Versloot, "About loss and loss functions," MachineCurve, Oct. 04, 2019. https://www.machinecurve.com/index.php/2019/10/04/about-loss-and-loss-functions/ (accessed Jan. 12, 2022).

[6] G. Seif, "Understanding the 3 most common loss functions for Machine Learning Regression," Medium, Oct. 03, 2021. https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3.

[7] J. Brownlee, "How to Use ROC Curves and Precision-Recall Curves for Classification in Python," Machine Learning Mastery, Aug. 30, 2018. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/ (accessed Jan. 17, 2022).