Training a neural net to identify top quark jets

KDSMIL001 — September 2022

Abstract

We train a neural net to identify jets as originating from top quarks as opposed to other quarks; a process known as top tagging. High-level variables describing the jets are used as opposed to constituent data as constituent requires considerably more computing power. The model's performance is then assessed according to some commonly used statistics and investigated for different p_{T} cuts.

1 Introduction

In proton-proton collisions at the Large Hadron Collider (LHC) at CERN, top quarks (along with their anti-particle counterpart) are produced about once every few seconds. Due to their proportionally large mass ($\sim 173\,\mathrm{GeV}$) their decay time is too short for them to be observed directly, so the next best way to determine if a top quark was produced is to look at the hadronic jets produced by the interaction and determine if they were a result of a top quark or some other, less interesting quark. This process is called top-tagging.

One way to perform this classification between signal (top quark jet) and background (any other jet) is to use a neural net. Neural nets are well suited to this task as they are able to handle the large volumes of Monte Carlo truth data that are available to us and can easily be constructed to provide an output from 0 to 1, which we can take to be the neural net's classification of signal (closer to 1) and background (closer to 0). We used data from `https://cds.cern.ch/record/2825328` [2] to train and test a deep neural net in top-tagging using "high-level" quantities describing the jets. This report will focus on the creation of that neural net, its performance according to the confusion matrix, and some investigation into where the neural net performs best in relation to the kinematics of the jets being tagged.

2 The data

The data we used to train and test the neural net comes from the dataset used in [2]. It contains an equal number of background and signal jets. Due to computation power restrictions, we were unable to use their train dataset, so we split the test dataset into $\frac{2}{3}$ for training and $\frac{1}{3}$ for testing.

The data is split up into two types: constituent and high-level. The constituent data is simply the p_{T}, η, φ, and energy E of the particles making up each jet. This is a lot of information and would ultimately be the best data to use as it holds all the available information about a jet, which neural nets are very good at sifting through to find the information relevant to the task at hand. The only issue is that it takes a lot of RAM to load all the data into, as well as requiring a lot of computation power (or time) to really get to a meaningful result.

To try cut down on computation time and resources, we used the high-level data. These are 15 variables calculated for each jet, from the constituent data, which have been identified by [3] and [4] as summarising the data in a way that lends itself to top-tagging. The 15 variables are:

- Energy Correlation Ratios: ECF_1, ECF_2, ECF_3, C_2, D_2
- N-subjettiness: τ_1, τ_2, τ_3, τ_4

- Splitting Measures: $\sqrt{d_{12}}$, $\sqrt{d_{23}}$
- Centre of Mass Observable: $\text{Thrust}_{\text{MAJ}}$
- Q_W, L_2, and L_3

Some preprocessing was needed to prepare the data for the training. If we were to use two input variables with wildly different scales, say energy on the GeV scale and distance on the nanometre scale, then changing an energy parameter even slightly will have such a large impact on the output of the neural net, and thus the loss function, that changing a distance parameter will practically have no effect. To avoid this, we did some basic preprocessing which involved simply subtracting the mean for a specific quantity from the value for all the jets, then dividing by the standard deviation. This centers the data around 0 and gives it a standard deviation of 1. Doing this ensures that our neural net will not be trained to weight a specific input more simply because it's on a larger scale than another input.

Each jet has a label stating whether it is signal or background, as well as weights used in training to re-weight the background so that its p_{T} distribution is the same as that of the signal. This is done since the production of the background events requires an unphysical p_{T} spectrum, so they could either be re-weighted according to the spectrum observed at the LHC, or they could be weighted to be identical to the signal distribution. The latter was chosen in order to avoid the neural net falsely tagging a signal jet as background simply because the training dataset didn't have any signal jets at that p_{T} due to them not being generated.

The 15 high-level quantities, the labels, and the weights are all that we gave to the neural net for training.

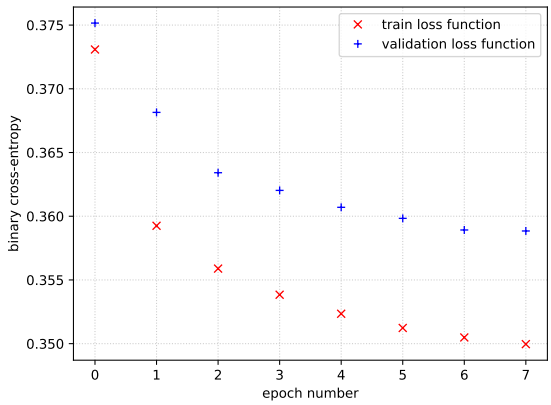
3 Creation and training of the neural net

The neural net was created using Python and Keras [5], which is an interface for using TensorFlow. We used an input layer with 15 inputs; one for each of the high-level quantities. We then used 3 hidden layers each with 20 neurons using the Rectified Linear Unit activation function. Finally, an output layer of 1 neuron using the sigmoid activation function was used to give us a prediction of signal (closer to 1) or background (closer to 0). The model was compiled with the argument `optimizer='adam'`, which has a learning rate of 0.001. We used a batch size of 100, with our training set being made up of around 1.6 million jets. 8 epochs were used. A validation dataset of 5% of the training dataset was used.

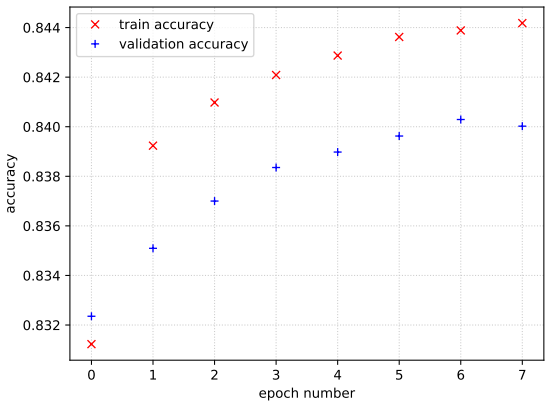
This configuration was chosen fairly arbitrarily, aside from the input and output layers, with a focus on simplicity and short run times (around 10 minutes of training). We chose the binary cross-entropy for our loss function as it is both simple and well-suited to a (this might shock you) binary classification task such as this one.

We can look at the evolution of the model with two parameters—the loss function and the accuracy—evaluated at each epoch in figure 3.1 for both the training and validation datasets. In this case, the accuracy is defined as the proportion of correct predictions when calling everything with prediction greater than 0.5 a signal, and background otherwise. This is a fairly rudimentary statistic and will be discussed in later sections.

The neural net finished training with a loss function value of 0.3500 and an accuracy of 0.8442.



(a) Progression of the loss function over the training of the neural net, shown for both the training dataset as well as the validation dataset. The loss function used was binary cross-entropy.



(b) Progression of the accuracy of the neural net over its training, evaluated on both the training dataset and the validation dataset. Accuracy here is defined simply as the proportion of correct predictions.

Figure 3.1

4 Predictions and neural net performance

With the neural net trained, we were then able to use it to predict the classification of jets in our testing set. This dataset contained around 8 million jets. All predictions greater than 0.5 were considered signal and all less than or equal to 0.5 were considered background. A first step towards iterating on the accuracy metric calculated earlier is to look at the confusion matrix, which tells us how many true positives (TP), false negatives (FN), false positives (FP), and true negatives (TN) were predicted. In our case we are considering signal to be "positive" and background "negative".

		Predicted Classification	
		Signal	Background
Actual Classification	Signal	367092	42453
	Background	80269	329945

Table 4.1: Confusion matrix for the predictions made by our neural net. Calculated using `sklearn.metrics.confusion_matrix` with signal (1) considered positive and background (0) considered negative. Going row by row, each entry is the number of true positives, false negatives, false positives, and true negatives [1]

Table 4.1 shows the confusion matrix for our neural net, with an overall accuracy of 0.85030. From the values shown, we can calculate some useful quantities that describe the predictive power of the neural net. The first two are simply the true positive and true negative rates. They are found with $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$ and $\text{TNR} = \frac{\text{TN}}{\text{TN+FP}}$. These are the proportions of true signal or background jets that were correctly identified as such, but this time the two classes are not mixed up so we can compare the neural net's ability to predict signal and background.

Next we can consider the positive and negative predictive values. These are defined as $\text{PPV} = \frac{\text{TP}}{\text{TP+FP}}$ and $\text{NPV} = \frac{\text{TN}}{\text{TN+FN}}$ and can be thought of as the proportion of, say, positive predictions that the neural net makes which are actually true positives. These statistics are subtly different

to the TPR and TNR as they tell us how often a certain prediction is correct, as opposed to how often a certain true value is predicted correctly. The PPV is often called the precision, while the TPR can be thought of as an accuracy.

We find these values to be

$$\text{TPR} = 0.89634, \quad \text{TNR} = 0.80432, \quad \text{PPV} = 0.82057, \text{ and } \text{NPV} = 0.88600.$$

These tell us a few things. Firstly, our neural net is more likely (around 9%) to correctly identify a signal event as signal than it is to identify a background event as background. We also see that when our neural net predicts a jet to be background, it is about 6% more likely to be correct than when it predicts a jet to be signal.

These two statements seem to be at odds with each other but they are in fact saying similar things. The higher TPR tells us that a true signal jet is less likely to be labelled as background than a background jet is to be labelled as signal. Similarly, the higher NPV tells us that if a jet is labelled as background, it is less likely that the jet is actually signal than a jet labelled signal is to actually be background. In other words, our neural net yields a more pure background sample than it does for signal.

For a visual representation of the distribution of signal and background events, we can plot a histogram of true signal and background events according to their prediction from the neural net. Figure 4.1 shows this.

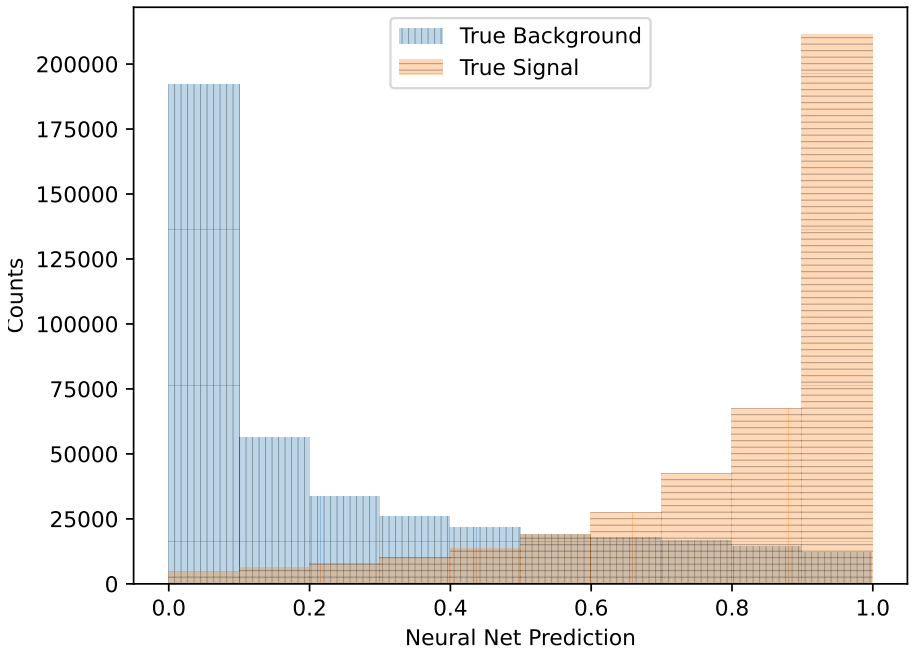
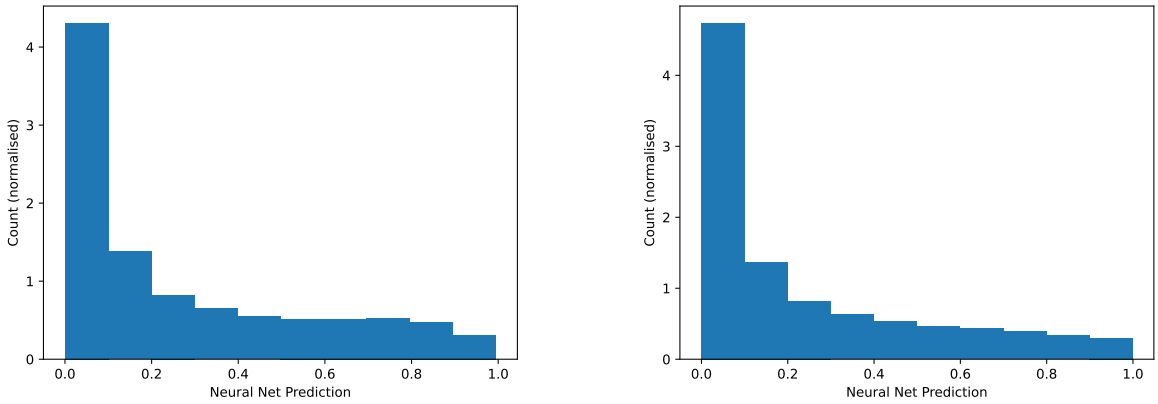


Figure 4.1: Histograms of true signal and background events, binned according to the prediction made by the neural net. Note that counts are not weighted according to the training weights as there are an equal number of signal and background events already and the weighting was only needed for dealing with the p_{T} distributions.

We can see qualitatively from figure 4.1 that there are more true background events predicted as signal than signal events predicted as background, which is exactly what the TPR, TNR, PPV, and NPV told us before. The plot otherwise looks fairly symmetrical and gives us a good indication that there isn't anything wildly wrong with our neural net. From this plot, we could decide on a first approximation to a best cut to make on the prediction from the neural net, to decide whether something is a signal or background. Changing the cut from 0.5 to something like 0.8 would serve to remove much of the background jets from our signal sample, making it far more pure. This would, however, lead to many signal events being classified as background, so our overall accuracy would go down. Similarly, cutting at 0.2 would make a more pure background sample. So there are trade-offs to be made when deciding on the cut that are influenced by the intended use of the neural net.

4.1 Performance in different kinematic regions

Another way that we can evaluate the performance of the neural net is to look at how it performs with respect to, say, the p_{T} of the jets it classifies.



(a) Histogram of neural net predictions of background events in the high p_{T} region.

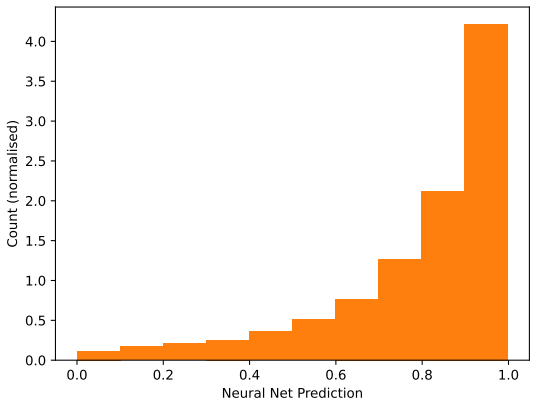
(b) Histogram of neural net predictions of background events in the low p_{T} region.

Figure 4.2

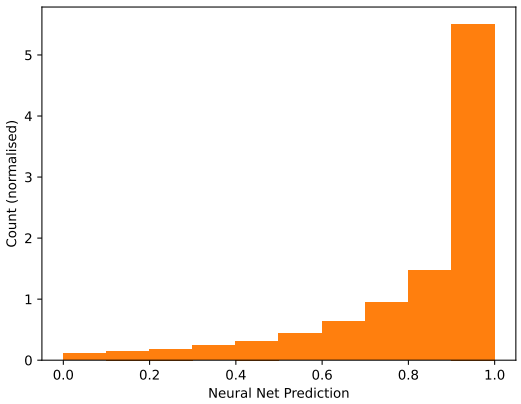
Figures 4.2 and 4.3 show histograms for the true background and signal jets, binned according to their neural net prediction. We have divided the two groups again by their p_{T} where low p_{T} is simply below the midpoint and high p_{T} is above that. The histograms have been normalised for easier comparison as there are more low p_{T} events than high.

We can see from the histograms that our neural net seems to perform a bit better at low p_{T} as the histograms for both background and signal are more sharply peaked than their high p_{T} counterparts. For a more quantitative view of this, we found the PPV, NPV, and overall accuracy of each p_{T} region:

	Low p_{T}	High p_{T}
PPV	0.79325	0.90893
NPV	0.90757	0.72474
ACC	0.84928	0.85484



(a) Histogram of neural net predictions of signal events in the high p_{T} region.



(b) Histogram of neural net predictions of signal events in the low p_{T} region.

Figure 4.3

We can see that the accuracy doesn't actually change appreciably, perhaps even favouring high p_{T} to some degree. What is interesting is that for low p_{T} the neural net performs in a similar fashion to how it performs on the whole, regarding the PPV and NPV, but for high p_{T} it seems to flip. This would mean that at high p_{T} the neural net is actually better at producing a pure signal sample than background.

This is a curious result as there doesn't seem to be any reason for the neural net to perform differently in different kinematic regions, but investigating the nature and cause of this phenomenon is outside the scope of this report.

5 Conclusion

We were able to train a neural net to perform top-tagging on simulated hadronic jets. An overall accuracy of 0.85030 was found when predicting test data that the neural net hadn't seen before in training. A higher accuracy could easily be achieved by simply running the neural net for longer or using more training data, but for the purposes of this report that accuracy is reasonable.

Looking at quantities calculated from the confusion matrix we found that the neural net is naturally more suited to producing a pure background sample than a pure signal sample. This would be useful if we wanted to make sure no signal events were accidentally labelled as background but we didn't specifically care about the purity of the signal sample. We also found that the neural net's behaviour switched when looking at jets with a p_{T} higher than the midpoint of our data, which would lend itself to a more pure signal sample.

References

[1] *Confusion matrix*. In: *Wikipedia*. Page Version ID: 1107701525. Aug. 31, 2022. URL: https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=1107701525 (visited on 10/02/2022).

[2] *Constituent-Based Top-Quark Tagging with the ATLAS Detector*. Tech. rep. Geneva: CERN, 2022. URL: https://cds.cern.ch/record/2825328.

[3] *Identification of hadronically-decaying top quarks using UFO jets with ATLAS in Run 2*. Tech. rep. Geneva: CERN, 2021. URL: https://cds.cern.ch/record/2776782.

[4] *Identification of Hadronically-Decaying W Bosons and Top Quarks Using High-Level Features as Input to Boosted Decision Trees and Deep Neural Networks in ATLAS at* $\sqrt{s}$ *= 13 TeV*. Tech. rep. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2259646.

[5] *Keras: the Python deep learning API*. URL: https://keras.io/ (visited on 09/29/2022).