Neural Network Classification of Top Quark Production at the Large Hadron Collider (LHC)

Introduction

Background

The Standard Model (SM) of particle physics developed in the 1970s has become well established as our best understanding of the basic building blocks of the universe. All matter in the universe is made of elementary particles called fundamental particles, and these occur in two basic types called quark and leptons – each group consisting of six particles which are related in pairs (or 'generations'). For the quarks, one of the generations (pairs) consists of the 'top quark' and the 'bottom quark', the former being the subject of our discussion below.

The fundamental particles are governed by four fundamental forces: the strong force, the weak force, the electromagnetic force, and the gravitational force. These are a result of the exchange of force-carrier particles, which belong to a broader group called bosons. Matter particles transfer discrete amounts of energy by exchanging bosons. Each fundamental force has a corresponding boson. The weak force, responsible for the radioactive decay of atoms, is carried by the W and Z bosons.

Even though the SM is the best model for subatomic particles physicists currently have, there are still a lot of potential for fine tuning and addressing many unanswered questions. The data collected by the high energy physics (HEP) experiments at the LHC has the potential to address some these questions. The advent of machine learning (ML) and more specifically deep learning has allowed scientists to handle higher dimensional data and solve more complex problems than what was previously possible. ML methods are used in multiple fields of physics including but not limited to HEP, astrophysics, anomaly detection, simulations etc. In experimental HEP at the LHC specifically, ML methods are being used to extract small signals from the much more common background events.

Objectives

One such signal of interest is the production of top-quark-antiquark pairs along with the Z boson referred to as the $t\bar{t}Z$ event. This event is a rare process in the framework of the standard model which makes it particularly interesting to study because it affects the measurements of other important SM processes like $t\bar{t}$ production in association with the Higgs boson. The top quark due to its large mass and coupling to the Higgs Boson plays a special role in electroweak physics within the standard model. Finally, it also serves as an irreducible background to other rare processes and searches for Beyond the Standard Model (BSM) phenomena. All these reasons make it an important event to distinguish from background events.

Proton-proton collisions at the Atlas experiment at the LHC are more likely to produce a pair of top quarks which is ideal for studying our signal. A $t\bar{t}Z$ event candidate at the LHC includes of course the production of a top quark, an anti-top quark and a Z boson. One top decays hadronically (transforms into protons, neutrons or other particles made of quarks) and the other decays leptonically (transforms into electrons, muons or taus). The Z boson decays into two muons. $t\bar{t}Z$ events can be selected based on the expected detector signatures at ATLAS, but some other background events like the WZ (production of W and Z bosons) can give similar detector signatures and can be orders of magnitude higher than the process of interest.

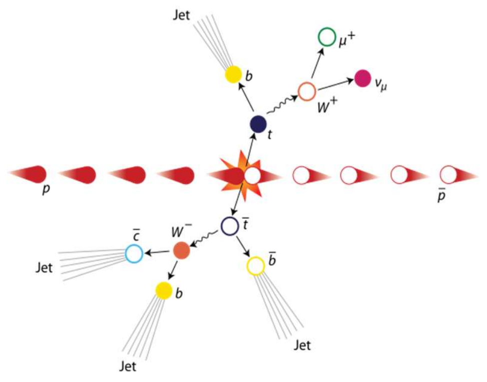


Figure 1: Top Quark Production

Our objective is to train a neural network classifier to distinguish between the signal and background events. The dataset being used is a simulation of the $t\bar{t}Z$ (signal) and WZ (background) events using the Monte Carlo method. Features include number of reconstructed objects, momenta and energies of particles, direction of particles and the missing transverse momentum. We will also examine the effect of different features and optimize the neural network architecture to improve classification performance.

Methods

We have used two machine learning algorithms, namely Decision Trees (DT) and Neural Networks (NN), for classification of signal and background events. The original dataset for signal and background events has 741538 rows (samples), 18 feature columns and 1 label column which has binary values 1 and 0 - 1 signifying a signal and 0 signifying a background.

Algorithms and Metrics Used

In the pursuit of the best performing classifier, we decided to first train a Decision Tree to establish a baseline model. A decision tree splits the data into groups based on the value of a feature (see figure). At each node, the DT decides which feature best separates the data and this process continues until either each node is pure (contains only a single class, 1 or 0) or a stopping criterion is met (for example, maximum depth of tree). The decision about splitting the data into groups is made by minimizing different criterion such as Gini index, entropy etc. at each node, which also happens to be one of the hyperparameters (predefined settings which are tuned to maximize performance of ML algorithms) for DTs. For example, the Gini index is calculated as:

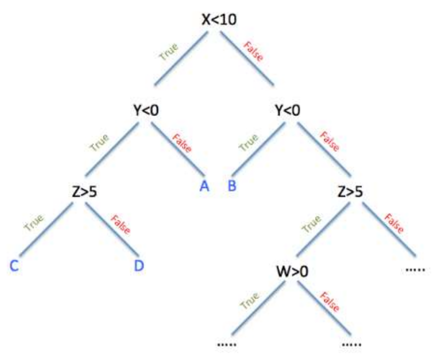


Figure 2: Decision Trees Example

$$Gini = 1 - \Sigma \mathrm{P(i)}^2$$

where the sum goes from 1 to n where n is the number of classes and P(i) represents the ratio of positive observations (1s) to total observations at each node.

One of the advantages of such an ML model is interpretability. We can easily use DTs to determine feature importances for different features. As we will see below, this also allows us to select a subset of the data and features without compromising on the performance of the model to reduce computation time.

The second algorithm used is a feed forward neural network (NN). This is simpler than counterparts like recurrent neural networks (RNNs) or convolutional neural network (CNN), but is relevant for our problem given the nature of our features. With more computational power, RNNs could be used to further improve performance.

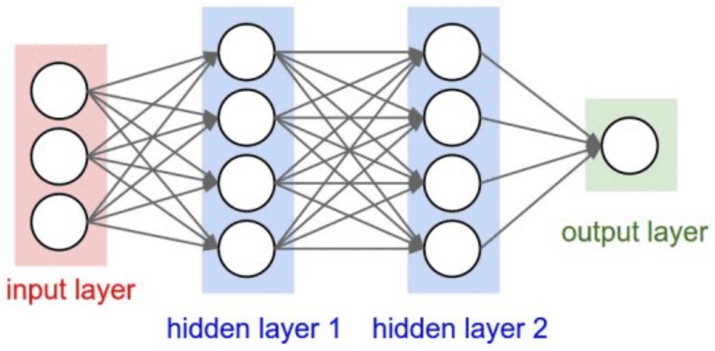


Figure 3: Structure of a simple neural network

A NN consists of three types of layers: the input layer (our features), hidden layers and the output layer. Each layer is made up of units called neurons and the neurons in each layer are connected to all neurons in the previous layers by weights (fully connected network). A non-linear activation function is applied to the weighted sum of inputs at each layer, which allows the network to learn complex relationships in the data. ReLU (Rectified Linear Unit) and tanh are some activation functions which were experimented with for obtaining the best model.

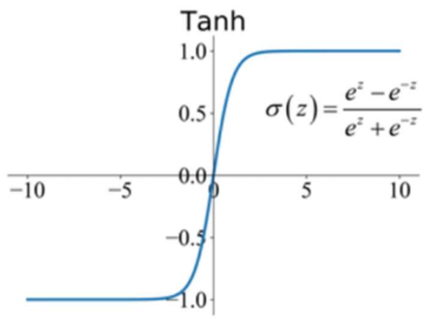


Figure 4: The tanh activation function

These weights are learned during the training process using techniques like back propagation where the predicted value from the network is compared with the true value, and an error is calculated using a loss function. This error is sent back through the network one layer at a time and weights are updated according to their contribution to the error. The best performing model is determined by minimizing the loss function.

The binary cross entropy loss is used which is the most common loss function for binary classification. It quantifies the difference between predicted probability distribution and binary label for each instance. The sigmoid activation function is used for the output layer which outputs a result between 0 and 1. This represents the probability of the result being in the positive class (signal).

We use the AUC metric (Area under the Receiver Operating Characteristic Curve) as our classification metric which is a standard metric in machine learning with higher AUC values indicating higher classification accuracy across a range of thresholds. Baldi et al. (2014) suggest that physicists might be interested in other metrics such as signal efficiency at some fixed background rejection but AUC is insightful as it is directly connected to accuracy and provides a more comprehensive picture. A random or dummy classifier will result in an AUC value of 0.5, and a perfect classifier will give an AUC = 1. Any mention of 'score' below indicates the AUC score.

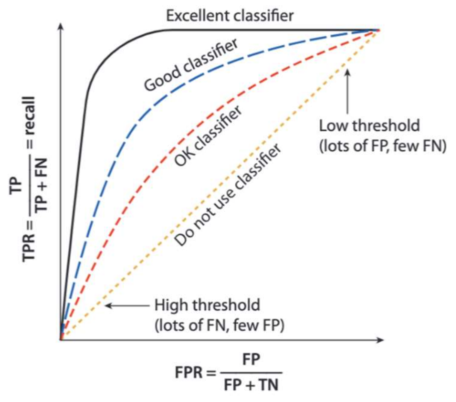

Excellent classifier
Good classifier
OK classifier
Do not use classifier
Low threshold (lots of FP, few FN)
High threshold (lots of FN, few FP)
$TPR = \frac{TP}{TP + FN} = recall$
$FPR = \frac{FP}{FP + TN}$

Figure 5: ROC (Receiving Operating Characteristic) Curve

Results

Training and Testing Strategies

The number of leptons for all observations in our dataset is equal to 3 and hence was removed from the feature set, leaving 17 features. Feature engineering was experimented with but showed no improvement in scores on train and test sets. Guest et al. (2018) argue in their paper that feature engineering by applying physics knowledge is often suboptimal specially in problems involving event selection using neural networks. This was also confirmed after multiple iterations of training DT and NN models.

Outliers were removed using z-score calculations (threshold = 1.8) not only to improve performance but also to reduce computation time. Distribution of features after removing outliers is shown below. This left us with 357071 rows (final dataset) which were used for splitting into train and test sets with an 80-20 split. The training set was used for 3-fold cross validation and hyperparameter optimization, while the test set was used to evaluate the best hyperparameters on unseen data. Finally, for both models, cross validation was performed on the entire final dataset to get the most accurate result. Cross validation ensured that results were significant both during hyperparameter optimization and fitting on the final dataset.

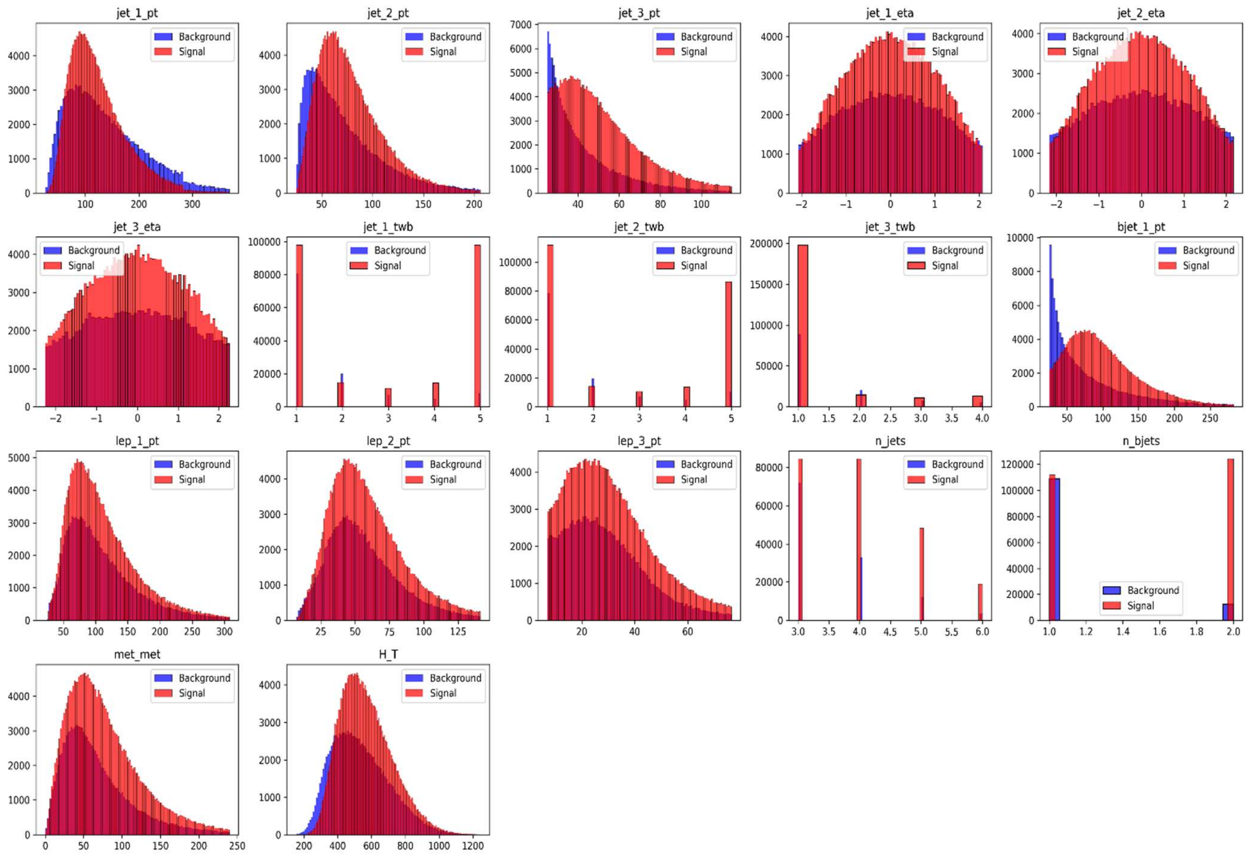


Figure 6: Distribution of Features after outlier removal

The DT allowed us to establish a threshold for a smaller subset (30% in our case) of training data by monitoring train training and validation scores over a range of training sizes using a learning curve. Validation scores stopped improving after 30% of the data for a simple decision tree with no max depth specified. This reduced computation time for hyperparameter optimization for both the DT and NN. The smaller subset was obtained using stratified sampling which maintained the class distribution of the final dataset and ensured that optimal performance was maintained.

Final Results

The 10 best features from the decision tree model fit on the training set were extracted and ranked. The decision tree with best hyperparameters was trained on two sets of data: The final dataset and the top 10 features extracted from the 30% subset of the final dataset. The latter was done to examine the effect of feature removal on the model, but there was no difference worth mentioning.

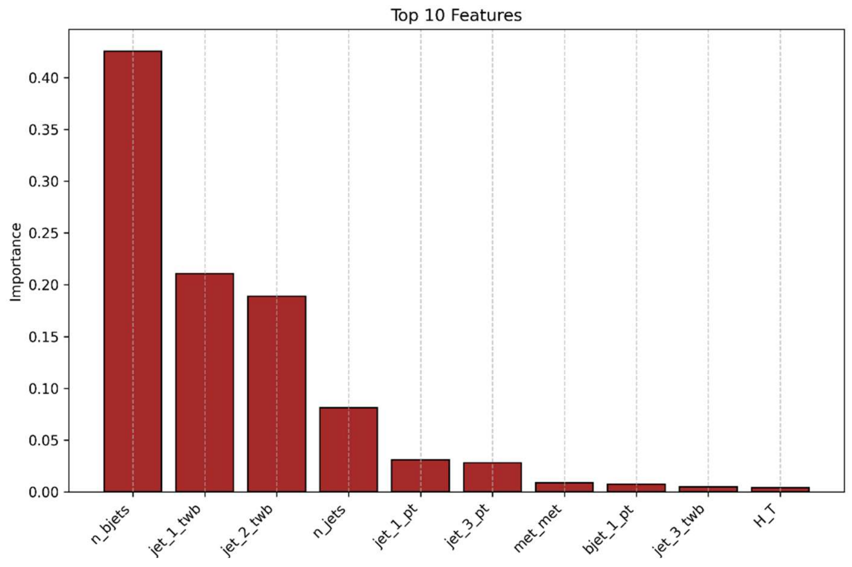
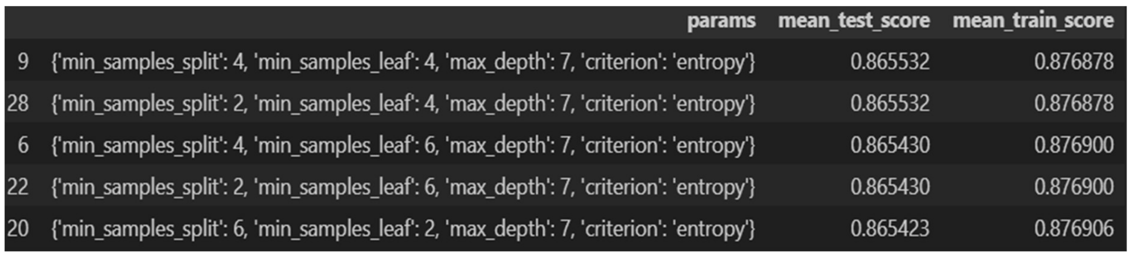


Figure 7: Top 10 Features from decision tree model

The hyperparameters, mean train score and mean test score of the 5 best performing DT models after cross validation on the subset data are given below. We used the top performing model to fit the model on the final dataset. After cross validation, the AUC score for the decision tree was with best hyperparameters was 0.87, which was a significant improvement from the random classifier (AUC = 0.5). The confusion matrix and ROC curve for the DT are also given.



	params	mean_test_score	mean_train_score
9	{'min_samples_split': 4, 'min_samples_leaf': 4, 'max_depth': 7, 'criterion': 'entropy'}	0.865532	0.876878
28	{'min_samples_split': 2, 'min_samples_leaf': 4, 'max_depth': 7, 'criterion': 'entropy'}	0.865532	0.876878
6	{'min_samples_split': 4, 'min_samples_leaf': 6, 'max_depth': 7, 'criterion': 'entropy'}	0.865430	0.876900
22	{'min_samples_split': 2, 'min_samples_leaf': 6, 'max_depth': 7, 'criterion': 'entropy'}	0.865430	0.876900
20	{'min_samples_split': 6, 'min_samples_leaf': 2, 'max_depth': 7, 'criterion': 'entropy'}	0.865423	0.876906

Table 1: Top 5 Best performing models after RandomSearchCV

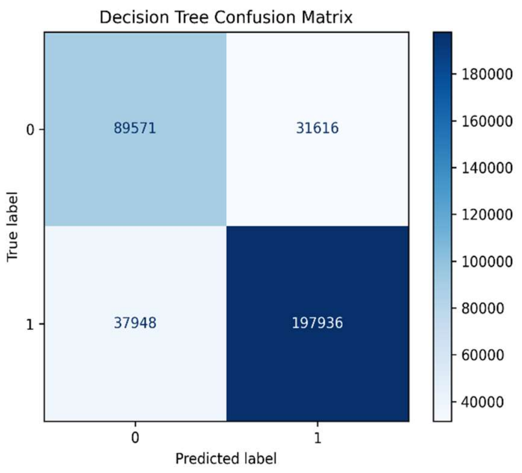


Figure 8: Decision Tree Confusion Matrix

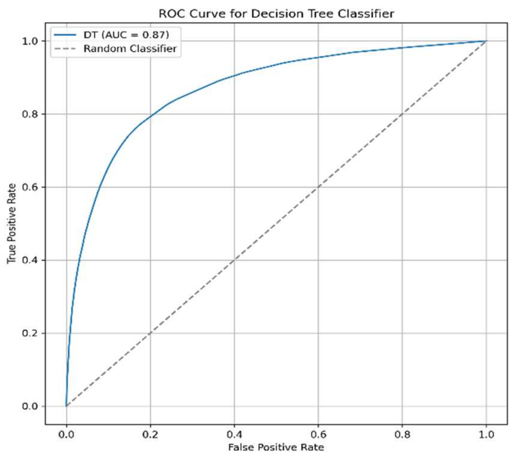


Figure 9: Decision Tree ROC Curve

The batch size and epochs (number of times the model goes back and forth) for the NN was predetermined to be 128 and 50 respectively keeping in mind the restricted computational resources. After hyperparameter optimization, the following network architecture was chosen to produce the best results: 2 hidden layers both with 450 neurons each, the first layer having activation 'relu' and the second having 'tanh'. Multiple optimizers were also tested and even though there wasn't a large difference between different optimizers, the Adam optimizer with a learning rate of 0.001 worked the best.

The best performing NN on the 30% subset with top 10 features gave an AUC of 0.89 and after cross validation on the final dataset, the mean AUC score was 0.88. The background rejection vs signal efficiency [(1-False Positive Rate) vs True Positive Rate] is commonly used to visualize results of ML models in such questions of signal detection (Baldi et al., 2014) and is shown below for both the Decision Tree and the Neural Network.

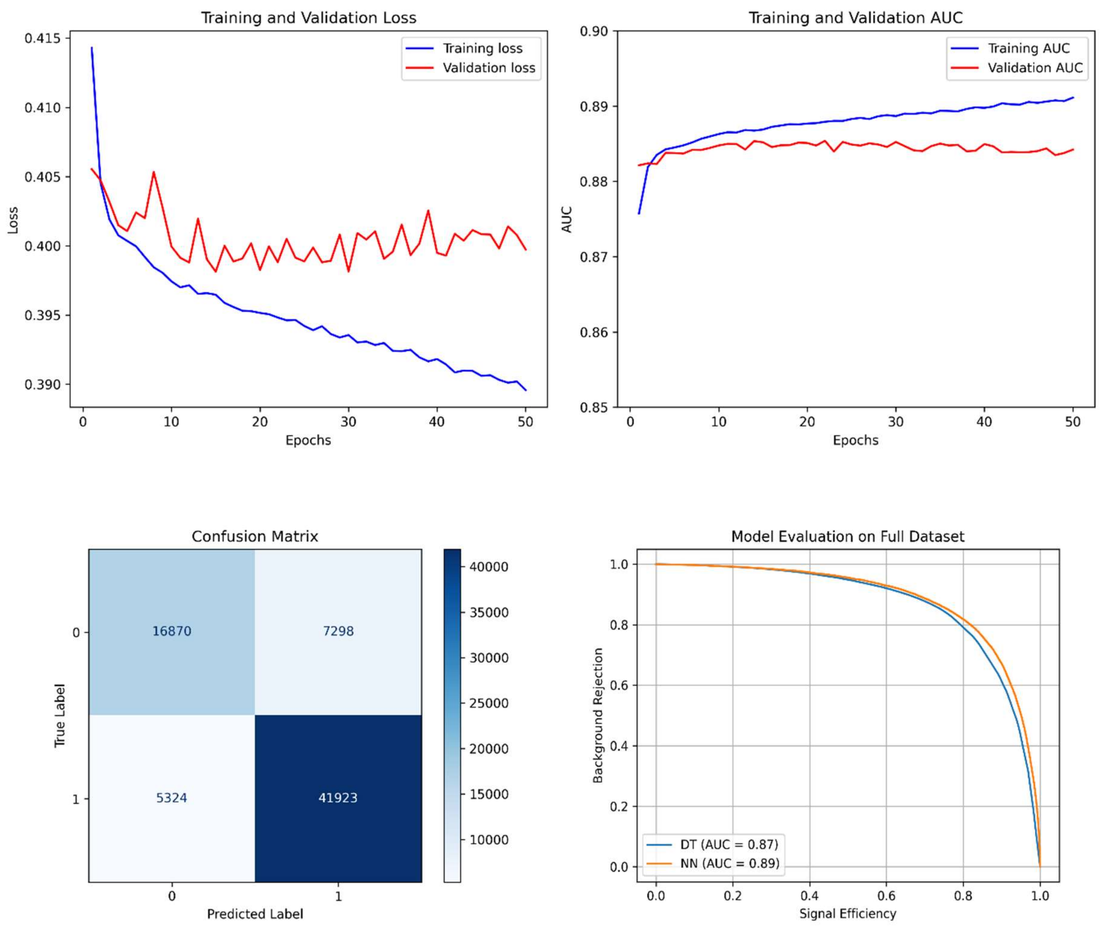


Figure 10: (a) Training and Validation Loss and AUC for best performing NN on subset data (b) Confusion Matix for NN on subset data (c) Model Evaluation on Full Dataset for DT and NN

Discussion and Summary

Both the best performing decision tree and the neural network on the final dataset resulted in AUC values close to 0.9. This can be interpreted as the model being very good at ranking signals more highly than background events. The confusion matrices do show that there are a considerable number of false positives and false negatives, but there is no indication of one of those metrics dominating over the other. That, along with the large number of true positives and negatives and the high AUC score is a good indicator that the model is distinguishing well between the two classes. One surprising result was the similarity in AUC scores for the decision tree and neural network. In theory the NN is supposed to be more complex and capture better relationships in the data than the DT. However, in this case, the feature relationships appear to be simple enough to be captured by the DT.

After multiple iterations of training the DT, the feature importance results clearly indicate that n_bjets (the number of bjets observed, where a bjet is a collimated spray of particles originated from the hadronization of a bottom quark) is the most important feature compared to others. However, there is no indication that using the top subset of features improve performance, for both NNs and DTs.

As discussed in the methods and results above, we successfully trained a neural network classifier on the simulated data to distinguish $t\bar{t}Z$ events from background WZ events. The model can reliably be used for the detection of these events and further study of the top-quark-antiquark pair. This can further be applied to fine tuning and extension of the Standard Model of Particle Physics.

One of biggest limitations encountered was restrictions on computing power and access to real event data. While simulated data does allow more instances for signal data, real event data might be used to capture more complex relationships between low level detector features using neural networks with more complicated architecture. Having more computational power will allow the testing of a bigger hyperparameter space for the NN and even test RNNs for our dataset.

The background rejection vs signal efficiency plot given above is one of the ways physicists can practically use the results and interpret the model. As suggested in the methods section above, the signal efficiency at a fixed background rejection can be used to focus on further improving the signal efficiency without compromising the ability to reject background events. It can further be used to better model background events and optimize analysis techniques both of which can improve sensitivity of event measurements.

References

1. https://home.cern/science/physics/standard-model
2. https://cernbox.cern.ch/s/lVR2Z8dzaxhtSJZ
3. Baldi, P., Sadowski, P., & Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nat Communications*, *5*. https://doi.org/10.1038/ncomms5308
4. Guest, D., Krammer, K., & Whiteson, D. (2018). Deep learning and its applications to LHC physics. *Annual review of Nuclear and Particle Science*. *68*. 161-181. https://doi.org/10.1146/annurev-nucl-101917-021019
5. Bourilkov, D. (2019). Machine and Deep learning applications in particle physics. *International Journal of Modern Physics A*. *34*(35). https://doi.org/10.1142/S0217751X19300199