PyTorch Neural Networks and Track Analysis for Top Quark Tagging

Genevieve Hayes[1,2,3] Colin Gay[2,3] and Alison Lister[2,3]

[1] Department of Engineering Physics, Queen's University, Kingston, Canada
[2] Department of Physics, University of British Columbia, Vancouver, Canada
[3] European Organization for Nuclear Research (CERN), Meyrin, Switzerland

Abstract. The identification of top quarks is motivated by their high mass and strong coupling to the Higgs mechanism. Boosted top quarks also allow for improved measurements of the Standard Model in the high momentum tails of event feature distributions. Neural networks have been proven as an effective method for distinguishing top quarks from Quantum ChromoDynamic (QCD) events using jet constituent features from the ATLAS and CMS calorimeters. In this project Deep Neural Networks (DNN's) and Long Short-Term Memory (LSTM) networks were built in PyTorch to compare their performances to previously tested Keras models. After applying similar preprocessing and optimization techniques, the performance of the PyTorch models was found to be highly comparable to the Keras models. Track features from the inner tracker offer promising new information to improve the performance of top tagging neural networks by utilizing information typically used in b-jet identification. Track features were analyzed and incorporated in processing scripts used to prepare the data for input to neural networks. It was found that keeping 100 p_T ordered tracks with p_T greater than 1 GeV could retain relevant information for jet classification while minimizing noise and computing time.

Keywords: Top Quark Tagging · Deep Neural Networks · LSTM Neural Networks · Track Analysis · ATLAS

1 Introduction

Top quarks are the heaviest fundamental particle that has been discovered, weighing almost as much as an atom of gold. Since top quarks do not hadronized before they decay, they are also the only quarks whose spin and charge quantum numbers can be measured directly. Because top quarks are abundantly produced at the LHC, a clear understanding of top signals is imperative to build an accurate model of the physics occurring within the detector. The most common decay chain of the top quark is presented in the Feynman diagram in Figure 1. In this project the hadronic decay channel of the W boson is of interest.

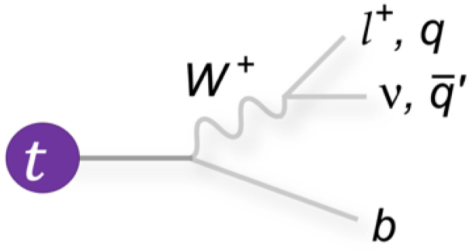

Fig. 1: Top quark decay into W boson and b quark.

Historically, standard jet algorithms have been used by ATLAS and CMS to seperate top quarks from quantum-chromodynamic (QCD) background arising from light quarks and gluons. These typically relied on the identification of three separate jets. At high momentum the decay products become so collimated it is more effective to use a single larger radius jet collecting all decay products of the top, so called boosted top jets. The first algorithms to identify boosted top jets relied on the computation of high-level quantities derived from physics knowledge. More recently, various types of neural networks, mostly built using the Keras platform, have been shown to successfully classify jets based on calorimeter information.

PyTorch is a deep learning framework that offers a promising alternative to Keras due to its increased flexibility, short training durations and debugging capabilities. PyTorch has also been shown to work well for high performance models and large datasets that require fast execution which is well suited to top tagging [4]. The first part of this project was the development of Deep Neural Networks (DNN's) and Long-Short Term Memory (LSTM's) networks in PyTorch for top tagging. The inputs to the models consisted of 4-vector kinematic information which was separated into training, validation and testing datasets. Each dataset was preprocessed to remove nonrelevant parameters and to improve training efficiency [8].

The second part of this project looked at improving upon these methods, considering complementary information from the inner tracker in parallel with the calorimeter features to improve the performance of the top tagging neural networks. The inner tracker of the ATLAS and CMS detectors measures the direction, momentum and charge of electrically-charged particles produced from pp collisions in the LHC, consisting mostly of charged pions and leptons [1]. Most frequently each of the pair-produced top quarks ($t\bar{t}$) decay into a W boson and a bottom (b) quark via $t \rightarrow bW$ followed by both W bosons decaying into a quark-antiquark ($q\bar{q}$) pair. The detection of the b-hadron with the top jet, using its measurable displacement from the primary vertex when it decays is hypothesized to improve neural network based top jet identification. For this purpose, pre-existing python scripts used to interpret and transform Delphes

simulated jets have been updated to incorporate track features for suitable input to the neural networks.

2 Introduction to Neural Networks

Neural networks are computer models that learn algorithms without being explicitly programmed, loosely based on the structure of the brain. They are versatile models that can learn from high volumes of complex data, making them a great option for top tagging. Another advantage of using this technique is that high level features do not need to be calculated, instead the network determines discriminants by observing low level features of the detectors.

2.1 Deep Neural Networks

A DNN is a neural network with more than one hidden layer, where each layer is comprised of nodes. In standard DNN's the connections between nodes consist of linear weights and biases which are optimized during training. Mathematically, the connections between nodes can be written as

$$y_j = \Theta(w_j x j + b) \tag{1}$$

where Θ is the activation function of the layer such as ReLu, sigmoid and softmax functions which adjust the inputs to the nodes. Back-propagation is used to compute the gradient descent with respect to weights, compare it to the desired outputs and then tune the model to minimize the loss function [7].

2.2 LSTM Networks

LSTM neural networks are Recurrent Neural Networks (RNN) with extended memory capabilities. RNN's work very well when analyzing sequential data due to an internal memory which allows the network to utilize information about previous entries to influence the weighting of the current output. In the case of an LSTM, this controls how the network reads, writes and deletes information. LSTM's solve problems such as exploding and vanishing gradients which can occur when the gradients either become very large (explodes) or very small (vanishes) when back-propagating through time [9].

2.3 Performance Evaluation

A common way of accessing the performance of neural networks is to use Receiver Operating Characteristic (ROC) curves which illustrate the background rejection as a function of signal efficiency. The background rejection and top tagging efficiency are defined as follows:

$$\text{Background Rejection} = \frac{1}{f_{pr}} \tag{2}$$

$$\text{Top Tagging Efficiency} = t_{pr} \tag{3}$$

where fpr is the false positive rate and tpr is the true positive rate. The Area Under the Curve (AUC) is then equal to the probability that the neural network will rank a randomly chosen signal event higher than a randomly chosen background event.

3 PyTorch Neural Networks with Jet Constituent Inputs

3.1 Dataset

The dataset used for training and evaluating the neural networks is the "Top Tagging Reference Dataset" provided by G. Kasieczka [2]. Kasieczka's dataset consists of 1.2M training events, 400k validation events and 400k test events simulated using Pythia8 [10] at centre-of-mass $\sqrt{s} = 14$ TeV. Each of the sets contain equal numbers of top quark signal and mixed quark-gluon background jets with each jet constituent defined by the 4-vector (px, py, pz, E). The detector response was incorporated into the features using the Delphes suite [6] with the ATLAS detector card. Jets are clustered using the anti-kT algorithm with R = 0.8. Only jets within the p_T range of 550 to 650 GeV were included. Multiple interaction and pile-up were excluded. A complete description of the dataset can be found in Reference [2].

It was previously found that an input of 30 jet constituents saturated Keras DNN performance, therefore only 30 jet constituents with the largest p_T were input into the PyTorch neural networks.

3.2 Neural Network Architecture

The DNN and LSTM models were both built in PyTorch. Preprocessing was applied to the 4-vectors of the Kasieczka dataset and the input layer of the networks consists of a sequence of (p_T, η, ϕ) describing the constituents within each jet. The five-step procedure consists of transformation, scaling, translation, rotation and flipping. The rationale behind this preprocessing is to prevent the DNN from learning the symmetries of space-time so it can more efficiently learn relevant discriminating features between signal and background [8]. It was found to improve the network performance significantly and reduce the computation time required for training and testing [8].

The first networks built were single layer feed-forward neural networks with large batch sizes and few epochs. These models evolved into DNN's by adding more dense layers. The best performing DNN consisted of and input layer with 90 nodes and 4 hidden layers with 300, 102, 12 and 6 nodes per layer before the final binary output layer. ReLu activation functions were applied throughout the hidden layers and the network was trained with the ADAM optimizer for a maximum of 20 epochs with a batch size of 96 and a learning rate of 0.001 to

minimize the cross-entropy loss. A batch size of 96 and a learning rate of 0.001 were used with the softmax activation function applied at the output layer.

Motivated by previously demonstrated improvements in background rejection in top tagging, a number of LSTM neural networks were also built in PyTorch. The addition of dense layers to the LSTM showed the greatest improvement in AUC. The best performing LSTM consists of an LSTM layer with 128 nodes followed by 2 fully-connected layers with 64 and 32 nodes respectively and a binary output layer. As with the DNN, the cross-entropy loss function was minimized using the ADAM optimizer with ReLu activation functions for the hidden layers and Softmax on the output later. A batch size of 96 was used with a learning rate of 0.001 for up to 20 epochs.

Parameters such as the learning rate, batch size, number of nodes per hidden layer and number of epochs were varied to explore their effect on the efficiency, accuracy and AUC. While overtraining was apparent in both the DNN and LSTM loss curves, neither L1 nor L2 regularization improved the network performances. L1 regularization still shows promise of further minimizing the loss function with more training epochs.

It should be noted that while the obvious parameters in successful Keras DNN structures were matched, there are default settings that are likely to differ between the Keras and PyTorch models.

3.3 Neural Network Performance Comparison

The validation data was used throughout training to ensure that the model was improving and not overtraining between epochs, but the overall PyTorch performance was tested after all epochs were completed on an independent test dataset. The PyTorch LSTM, PyTorch DNN and Pearkes' Keras DNN performances were tested separately on the test data.

The ROC curves of the best performing PyTorch DNN, Keras DNN and PyTorch LSTM are presented in Figure 2 [8][2].

Many more architectures have yet to be explored which could include changes in the optimizer, tuning of the batch size and the number of neurons. An increase in the background rejection of both the PyTorch and Keras LSTM's by a factor of 2 at high signal efficiencies in compared to the DNN's is apparent. The performance of the neural networks developed using PyTorch is highly comparable to those developed using Keras.

4 Incorporating Tracks

Jet constituent information only contains kinematic information, not information from particle-level charge or distance of closest approach to the primary interaction. Tracks from the inner tracking detector may be able to fill this gap by providing information regarding the displaced secondary vertices from decays of b-hadrons for better top quark identification. For this reason, track information from the detection of charged particles was analyzed and prepared for neural network training.

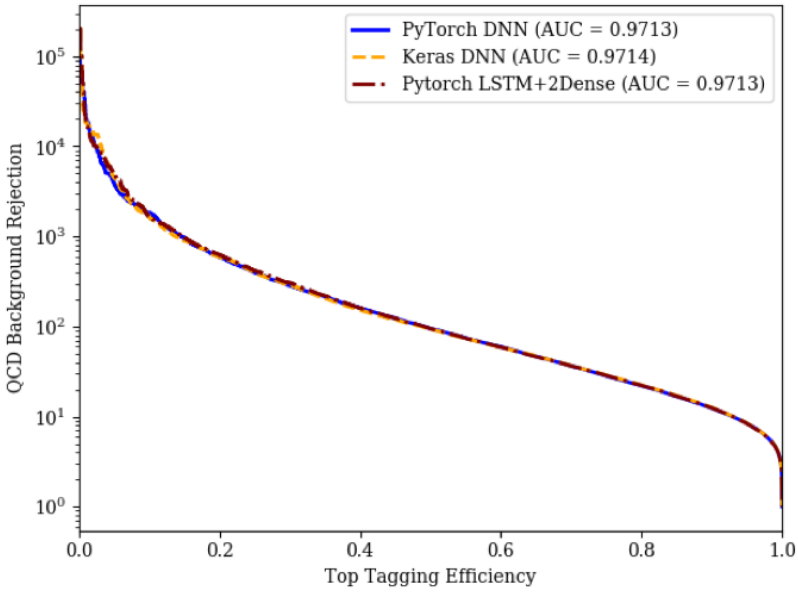

Fig. 2: ROC curves of the best performing PyTorch DNN, Keras DNN and PyTorch LSTM. Each model is presented with its respective Area Under the Curves (AUC).

4.1 Dataset

The dataset used to combine jet and track information is the same used for the development of jet constituent based Keras DNN's [8]. Sequential Standard Model Z' boson signal events were simulated using Pythia8 [10] at centre-of-mass $\sqrt{s}$= 13 TeV with pole masses ranging in 32 steps from 1400 to 6360 GeV. The detector response was incorporated into the features using the Delphes suite [6] with the CMS detector card. Hard QCD dijet processes are incorporated by using mixed quark-gluon background events with traverse momentum between 470 and 2790 GeV. Inelastic and non-diffractive soft jets are also generated. A complete description of the dataset can be found in Reference [8].

4.2 Technical Implementation

Event simulation in the Delphes framework generates root files which require classification, merging, subsampling and preprocessing to become inputs into PyTorch neural networks.

Each root file contains features from a large number of simulated events, all stored in TTree structure. The TTree contains branches such as tracks, jets and jet constituents and leaves which consist of the features corresponding to each of these branches. The structure can be viewed using TBrowser in root.

The branches are loaded into process_events_wtracks.py at the bottom of the script in the function process_events_wtracks(file_name, directory_name) as the

variable "t". "branchJet" is the pointer to the jet branch and "branchTrack" is the pointer to the track branch.

To ensure that the location of the first track feature is at the same location for each z_jet numpy array, cutting and zero padding is applied to the jet constituents. The distributions for the number of jet constituents in each jet is presented in Figure 3 for both signal and background jets.

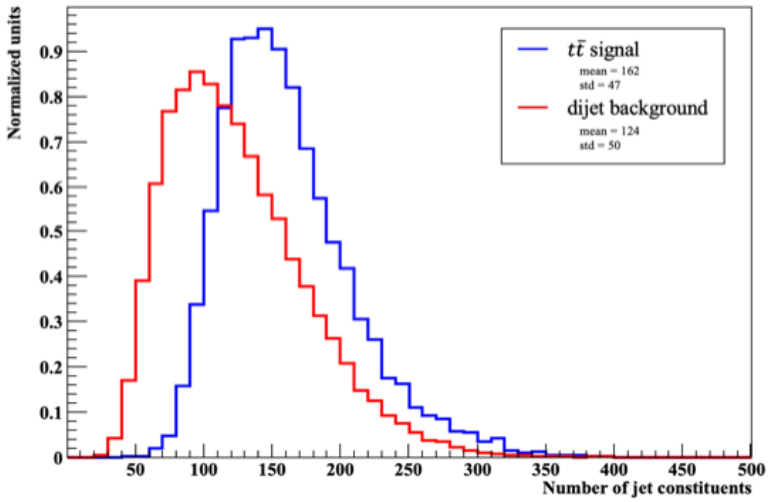

Fig. 3: Normalized distribution of the number of jet constituents within each signal jet (blue) and background jet (red).

4.3 Properties of Tracks Within Jets

Over 10000 events, the maximum number of jet constituents per jet was found to be 421. However, as illustrated in Figure 3, the vast majority of the jets have less than 300 jet constituents. For this reason, a maximum of 300 jet constituents is saved for each jet resulting in 1200 jet constituent features (p_T, η, ϕ and Reduction flag for each jet constituent). Jets that contain fewer than 300 jet constituents are zero padded up to 1200 features.

Within the function get_associated_tracks(), tracks that are within a dR of 1.0 of a jet are matched to that jet, where dR is defined as

$$dR = \sqrt{d\phi^2 + d\eta^2} \tag{4}$$

and $d\phi$ and $d\eta$ are the difference between the ϕ and η of the jet and the track respectively. The transverse momentum (p_T), azimuthal angle (ϕ), pseudorapidity (η), transverse impact parameter (d_0) and longitudinal impact parameter (d_Z) for each track are saved.

The tracks are then sorted in order of descending p_T, so that the 5 track features corresponding to the track with the highest p_T are the first 5 elements of y_jet_300 and so on. Normalized distributions of the number of tracks within

signal and background events are presented in Figure 4 without any track p_T cut and with track p_T cuts at 1 GeV, 2 GeV and 10 GeV.

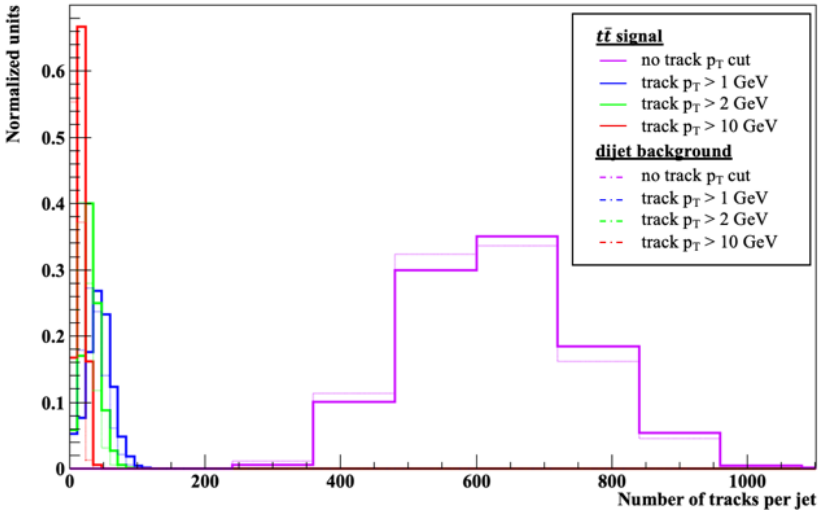

Fig. 4: Normalized distribution of the number of tracks per jet for signal events (solid line) and background events (dashed line) without a track p_T cut (violet) and with track p_T cuts at 1 GeV (blue), 2 GeV (green) and 10 GeV (red).

On average, about 80% of the tracks within a jet have a p_T less than 1 GeV and over 90% have a p_T less than 10 GeV. Since low p_T tracks carry little relevant information for jet classification, a 1 GeV p_T cut on the tracks was made to minimize irrelevant inputs and reduce computing resources. The distributions for the fraction of track p_T carried by the highest-p_T track, the sum of the highest 3 p_T tracks and the sum of the highest 10 p_T tracks for signal and background jets are presented in Figure 5.

On average, the 10 highest p_T tracks appear to carry over 50% of the p_T of the jet. The average p_T fraction is defined as

$$\frac{\sum_i^n \text{track}_i \; p_T}{\text{jet} \; p_T} \tag{5}$$

where track p_T is ordered from highest to lowest and is plotted from n=1 to n=100 in Figure 6.

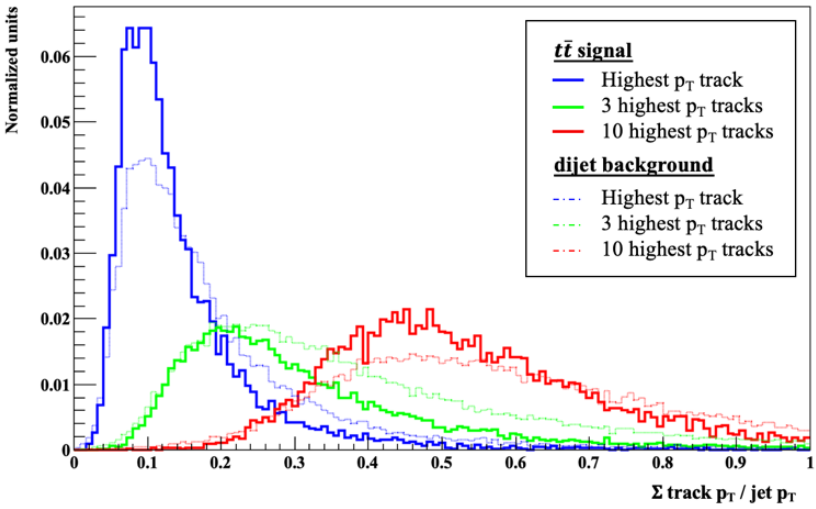

Fig. 5: Distribution of $\sum_i^n$ track$_i$ p_T / jet p_T carried by the highest-p_T track (n=1, red), the sum of the highest 3 p_T tracks (n=3, green) and the sum of the highest 10 p_T tracks (n=10, blue). Signal jets and background jets are presented with solid and dashed lines, respectively.

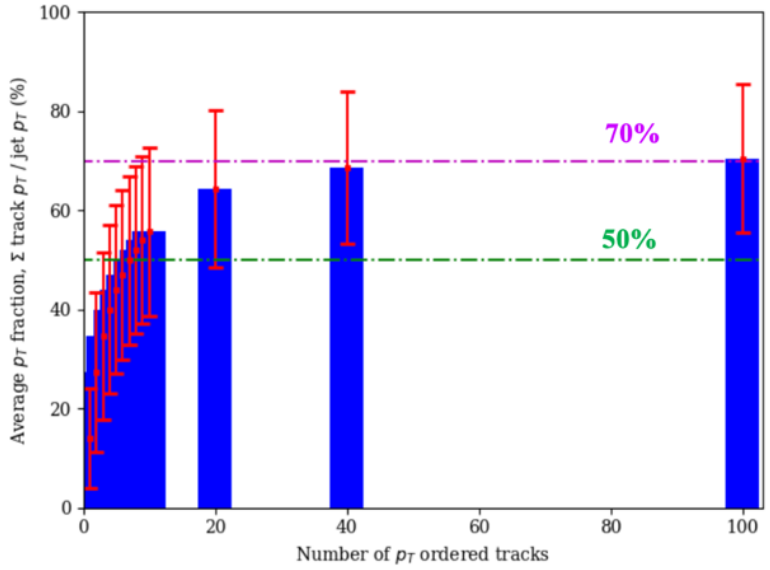

Fig. 6: Average p_T fraction, $\sum_i^n$ track$_i$ p_T / jet p_T as a function of n including the highest p_T track (n=1) up to the sum of the 100 highest p_T tracks (n=100). Standard deviations of the distributions are presented as error bars.

An average of 70% of the jet p_T is contained in the highest 100 p_T tracks. This is expected to be enough tracks to contain the relevant information about

the jet, therefore only using the highest 100 p_T tracks is hypothesized to be sufficient for neural network training and testing.

Within the function create_vector_delphes(), the numpy array z_jet is built for each jet by concatenating 20 jet features (x_jet), the 4-vectors from 300 jet constituents (y_jet_300) and 5 track features from the highest 100 p_T ordered tracks (y_track_sorted_100). The z_jet for each jet is output in a numpy file. Another root file is also generated which contains a number of histograms illustrating key features of the jets, jet constituents and tracks.

5 Next Steps

Now that the top and jet matched numpy arrays have been created, accompanied by diagnostic histograms for the track analysis, the next steps are as follows:

1. Merge and weight numpy arrays of the same type.
2. Extract flat p_T distributions from signal and background jets.
3. Preprocess tracks to match jet constituent translation, rotation and scaling.

While there is no obvious benefit of applying the rotation and scaling on tracks as is done for the clusters, the network performance both with and without preprocessing should be studied. In particular the hope is that the additional information contained in the tracks would be more focused on reconstructing displaced vertices than on the image-like properties of the calorimeter. Once all of this preprocessing has been completed, one possibility for incorporating the tracks is to input the track features to a neural network separate from the jet constituent neural network. An illustration of this workflow is presented in Figure 7.

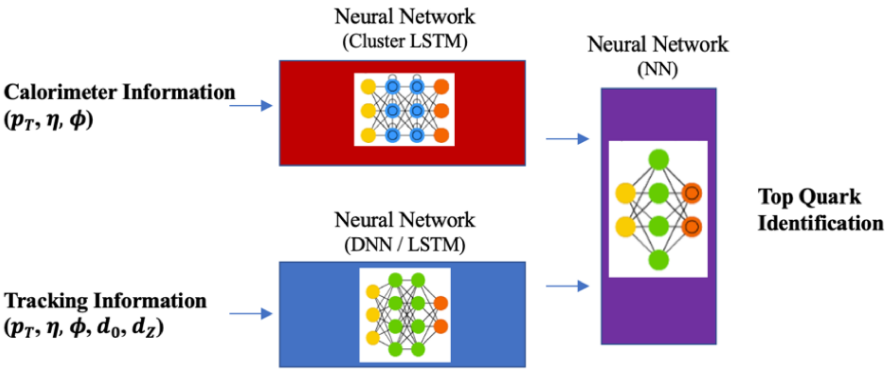

Fig. 7: Workflow for the parallel incorporation of calorimeter information and tracking information for top quark tagging.

Using a Cluster LSTM shows promising results for the calorimeter information neural network [5]. For the tracking information, a variety of neural network structures may prove to be successful, but a DNN or LSTM is a good place to start to benchmark the effect of adding tracking features [3].

6 Conclusion

Jet constituent based neural networks have been demonstrated as effective methods for distinguishing top quark jets from background jets. DNN's and LSTM networks were built in PyTorch for the purpose of differentiating top quark jets from QCD background jets. The inputs to the models consisted of preprocessed kinematic information which was separated into training, validation and testing datasets. Both networks demonstrated performance highly comparable to previously developed Keras models. Notably, the LSTM performance marked a two-fold increase in background rejection at high signal efficiency compared to the DNN performance.

Tracking features were included into preprocessing scripts motivated by the displacement of the top quark's b-jet from the primary vertex. It was found that adding 100 tracks with transverse momentum above 1 GeV could provide sufficient information to be incorporated into a neural network to contribute towards top quark tagging.

7 Acknowledgements

I would like to express my gratitude to the Institute of Particle Physics (IPP), the Natural Sciences and Engineering Research Council of Canada (NSERC), the University of British Columbia (UBC), and the European Organization for Nuclear Research (CERN) for sponsoring this research project. I would also like to thank my amazing supervisor, Alison Lister for her continual support and guidance, no matter the time zone. Finally, a very special thank you to Robin Newhouse and Jannicke Pearkes for all of their help, patience and mentorship throughout this project.

References

1. Aad, G., et al.: Performance of *b*-Jet Identification in the ATLAS Experiment. JINST **11**(04), P04008 (2016). https://doi.org/10.1088/1748-0221/11/04/P04008
2. Butter, A., et al.: The Machine Learning Landscape of Top Taggers. SciPost Phys. **7**, 014 (2019). https://doi.org/10.21468/SciPostPhys.7.1.014
3. Di Bello, F.A.: Optimisation of the ATLAS *b*-tagging algorithms for the 2017-2018 LHC data-taking. PoS **EPS-HEP2017**, 733 (2017). https://doi.org/10.22323/1.314.0733
4. Edureka: Keras vs tensorflow vs pytorch — deep learning frameworks (2018), `https://blog.exxactcorp.com/tensorflow-vs-pytorch-vs-keras-for-nlp/`
5. Egan, S., Fedorko, W., Lister, A., Pearkes, J., Gay, C.: Long Short-Term Memory (LSTM) networks with jet constituents for boosted top tagging at the LHC. https://arxiv.org/abs/1711.09059 (2017)
6. J. de Favereau et al.: Delphes 3 collaboration. Cornell University (2014). https://doi.org/10.1007/JHEP02(2014)057, `https://arxiv.org/abs/1307.6346v3`

7. Jain, A.K., Jianchang Mao, Mohiuddin, K.M.: Artificial neural networks: a tutorial. Computer **29**(3), 31–44 (March 1996). https://doi.org/10.1109/2.485891
8. Pearkes, J., Fedorko, W., Lister, A., Gay, C.: Jet Constituents for Deep Neural Network Based Top Quark Tagging. https://arxiv.org/abs/1704.02124 (2017)
9. Sundermeyer, M., Ney, H., Schlüter, R.: From feedforward to recurrent lstm neural networks for language modeling. IEEE/ACM Trans. Audio, Speech and Lang. Proc. **23**(3), 517–529 (Mar 2015). https://doi.org/10.1109/TASLP.2015.2400218, `https://doi.org/10.1109/TASLP.2015.2400218`
10. T. Sjöostrand et al.: An introduction to pythia 8.2. Cornell University (2015). https://doi.org/10.1016/j.cpc.2015.01.024, `https://arxiv.org/abs/1410.3012`