b-tagging using Neural Network

Kaifu Lam

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Physics

University of Washington

June 1, 2017

Committee:

Prof. Shih-Chieh Hsu

Prof. Jeffrey Wilkes

Program Authorized to Offer Degree:

Physics

University of Washington

Abstract

b-tagging using Neural Network

Kaifu Lam

Chair of the Supervisory Committee:

Professor Shih-Chieh Hsu

Physics

The existence of heavy particles, such as Higgs bosons and top quarks, which have short lifetime, cannot be detected directly and is inferred by the existence of their decay products. The bottom quark, observed as a jet of particles (b-jet) in the detector, is a common decay product of heavy particles. Therefore, the identification of b-jets (b-tagging) in particle detectors is essential for studying the physical processes of these heavy particles. b-tagging algorithms reconstruct particle trajectories, formulate parameters from the observed data, and classify particle flavors based on these parameters. This paper demonstrates that better b-tagging performance can be achieved by using neural network models, as opposed to using b-tagging parameters alone, as classifiers.

Keywords: experimental particle physics, b-tagging, recurrent neural network

Acknowledgement

I'd like to thank Prof. Shih-Chieh Hsu, Dr. Samuel Meehan, Dr. Daniel Guest and Julian Collado for providing guidance on this project. I'd also like to thank Pak Kau Lim, Alex Hostiuc and Todd Olson for providing technical assistance, especially during the very beginning of this effort. Tuition assistance is partially provided by Boeing LTP program. Last but not least, I'd like to thank family and friends for their continued support for this endeavor.

Table of Contents

Introduction

The progress of modern experimental particle physics has been driven by new technologies in the past century. [1] Since CTR Wilson invented the cloud chamber in the early 20th century, scientists and engineers have developed many generations of tools to detect and classify new particles. The last and most recently observed particle is the Higgs boson. Theorists had proposed the existence of the Higgs field and its particle excitation, the Higgs boson, since 1960s, but it was not experimentally confirmed until July 2012, by both ATLAS and CMS experiments in CERN, during the first run (Run 1) of the Large Hadron Collider (LHC).

Detectors in CERN, such as ATLAS and CMS, are some of the most sophisticated instruments that humankinds have ever built. Two protons, accelerated by the LHC to nearly the speed of light with a combined center-of-mass energy of 8 TeV (during Run 1), travel in opposite directions and collide in the center of the detectors, resulting in showers of daughter particles. The detectors then record the trajectory and energy of these daughter particles, generating 60 million megabytes of collision data every second. [2] The useful data is then summarized, stored and analyzed. Electronic sensors, computer servers and data analysis algorithms are the backbone for contemporary experimental particle physics.

One of the main tasks of the detector is to identify particles and their flavors. Since heavy particles are short lived and they decay before reaching the detector, the existence of heavy particles is inferred by the detection of their lighter decayed products inside the detector. One of the common decay products of heavy particles, such as top quarks (t) and Higgs bosons (H), is the bottom quark (b). However, detecting b quarks, which are observed in the detector as showers of particles called b-jets, is not an easy task. [3] This is because distinguishing b-jets from the jets of other (lighter) quarks is difficult. Over the years, jet reconstruction algorithms have been developed in an attempt to classify jet flavor (e.g. bottom, charm, etc.). However, none of them are currently being utilized for b-tagging in major detectors at CERN.

The reconstructed jet, a mathematical object created by the jet reconstruction algorithms, is represented by a collection of observable variables. These variables, by themselves, are generally not good classifiers of jet flavors. However, a machine learning algorithm trained by a collection of these variables is shown to have good jet classifying power.

The rapid development of machine learning algorithms in the past few decades allows computers to perform such tasks as facial recognition and speech recognition. One class of machine learning algorithms is the neural network. A neural network is a mathematical model containing simple units, called neurons, which are densely connected to each other. The neurons are assembled in layers such that the input (e.g. jet parameters) is connected to the output (e.g. jet flavor). Given a large enough training dataset, and model fine-tuning, the neural network will be able to predict the jet flavor based on given jet parameters. Since 2014, significant progress has been made to develop different configurations of multi-layered neural network, called deep neural network (DNN). A subclass of DNN, called the recurrent neural network (RNN), can perform classification based on input data that is sequential and / or has variable lengths (e.g. speech or certain jet reconstruction variables). It was shown that RNN can be a good jet flavor classifier. [4]

The purpose of this paper is to verify previous effort in utilizing the RNN representation for b-jet identification. [4] This paper will first provide a high level summary of particle identification physics.

Then, the simulated pre-abstracted expert level parameters of proton-proton (pp) collisions are introduced, and the linear discrimination power of each parameter is evaluated. Finally, the classification performance of a parameter trained neural network classifier is evaluated. The algorithm implementation utilized a Gated Recurrent Unit layer; however, since the expert level input data is not sequential, the GRU network is run in "vanilla" mode, similar to a "classical" neural network.

Physics Motivations – Particle decay channels

The bottom quark is a common decay product of heavier particles like the Higgs bosons and top quarks. The ability to identify bottom quarks in the detectors will advance our understanding of these short-lived heavy particles, as well as their physical processes. It will also enable us to study new physics theory, such as SuperSymmetry. [5]

Higgs process

The standard model predicted the existence of Higgs field in 1960s. Figure 1 [6] shows the processes by which the quantum excitation of the Higgs field, the Higgs boson, can be created. In a particle accelerator, Higgs bosons can be created when two protons, which are made of quarks (q) and gluons (g), collide. Since the Higgs boson has a short mean lifetime of ~10^{-22}s, it decays into daughter particles before reaching any detectors. Consequently, the existence of a Higgs boson is inferred by the existence of decay products observed by the detector.

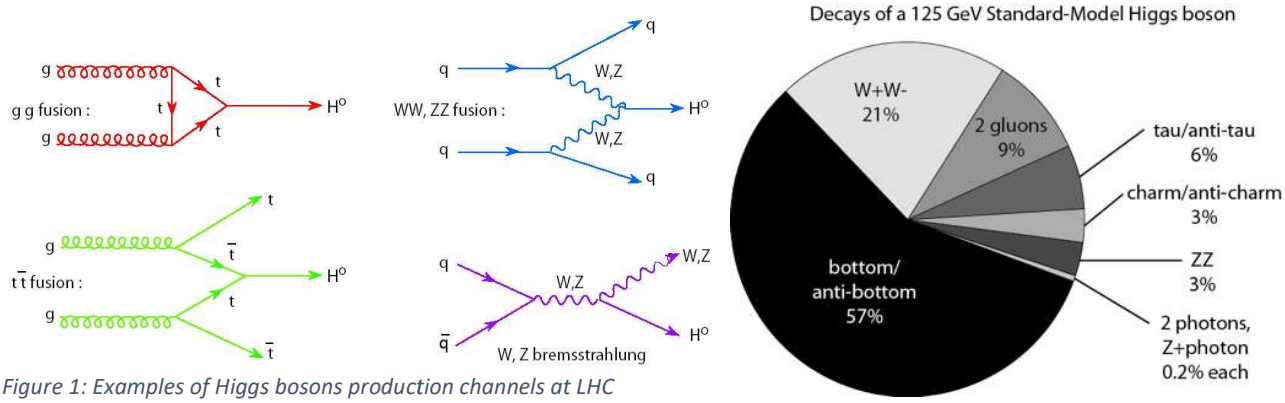


Figure 1: Examples of Higgs bosons production channels at LHC

Figure 2: 125 GeV Higgs Bosons decay modes and their relative frequencies of occurrence

As seen in Figure 2 [6], the Higgs boson decay into a bottom quark (b) and anti-bottom quark ($\bar{b}$) pair with a 57% probability. This is because the b quarks (4 GeV) are the heaviest quarks that the Higgs boson (125 GeV) can decay to; Top quarks (t) (172 GeV), the heaviest quark known so far, are too heavy to become the decay products of Higgs bosons. Note that due to Quantum Chromodynamics (QCD) confinement, the daughter quarks created from the Higgs decay will hadronize, and can only be observed in the detector as narrow cones of jets of composite particles (hadrons or mesons). A single quark cannot exist by itself.

Therefore, the identification of bottom jets (b-jets) in the detector, or b-tagging, is important for further studying the Higgs process.

Top process

The top quark (t) is a heavily researched subject. Its production channels at LHC are shown in Figure 3. It has a mass similar to that of a gold atom, and is the heaviest elementary particles known to date. The large mass of the top quark makes it couple strongly with the Higgs boson. Therefore, a better understanding of the top processes will enable us to study the Higgs processes. [7]

The t quark has a very short life time, about 5 x 10^{-25} sec. This allows t quarks to decay before hadronization. The result is that nearly all t quarks decay into W and b (t -> W + b). Therefore, the identification of b-jets is important to further our understanding of top physics.

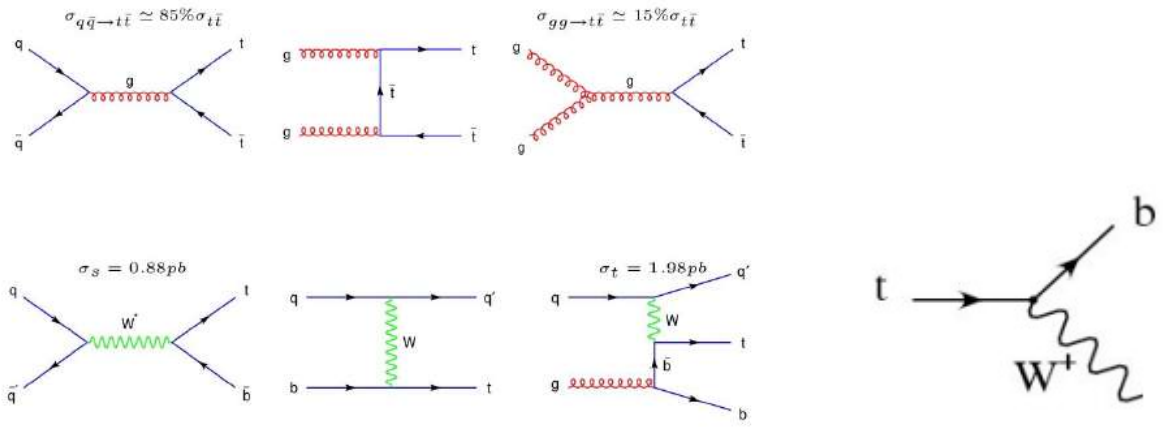


Figure 3: Top quark production channels and the most common decay channel at LHC

Decay lifetime and detector geometry

b quark has a favorable lifetime that allows the b hadron to decay inside the detector, creating a secondary vertex signature. This feature is used to distinguish jets of b hadrons from other jets. Light particles (e.g. hadrons containing up (u) or down (d) quarks) are stable and have long lifetime. They can travel through the tracking detectors without decay. On the other hand, heavy particles (e.g. hadrons of top quarks) decay quickly near the pp collision point (primary vertex) after their creation.

The b hadron, lighter than t but heavier than hadrons of u or d, has a lifetime of 1.5 picoseconds (1 ps = 10^{-12} sec) in the particle frame of reference. Since the b hadron is traveling near the speed of light, its lifetime is observed to be about 150 ps in the lab frame with a decay range of about 5 cm, which is within detection range of the inner detectors. The calculation summary is shown in Table 1.

General purpose detectors usually comprise of three main classes of detectors: trackers, calorimeters, and muon detectors. The configuration of the ATLAS detector is documented in Appendix A. [8]

Table 1: b hardonrs lifetime and decay distance sample calculation summary

Particle Frame	Lab Frame
Travel @ 0c Lifetime: 1.5 x 10^{-12} sec Distance: 0.5 mm	Travel @ 0.99995c, γ =100 Lifetime: 150 x 10^{-12} sec Distance: 5 cm

Flavor Tagging: How are b-jets identified from pp collision raw data?

Processing raw data into meaingful data is a complicated task. In general, the raw data collected by detecetors is first reconstructed into tracks (trajectories of daughter particles), and, subsequently, multiple tracks are reconstructed into a jet, the observable object originates from one quark. Parameters of these reconstructed tracks or jets are generated by b-tagging algorithms, such as Impact Parameter and Secondary Vertex. [9] The reconstructed jets are then classfied (flavor tagged).

Impact Parameter (IPxD)

The IPxD algorithm returns impact parameters for each track in a jet. Impact Parameter is the closest approach distance to the primary vertex of the back projection of a track. d0 is the impact parameter transverse to the beam line, while z0 is the impact parameter along the beam line. (Figure 4, [10]) These impact parameters are set to positive if the track intersects the jet axis in front of the primary vertex, and vice versa.

The impact parameter significance, i.e. $d0/\sigma_{d0}$and $z0/\sigma_{z0}$, where σ is the uncertainty of the impact parameter, is an important variable for b-tagging. In general, a larger impact parameter significance indicates a clear secondary vertex signal, which in turns represents the presence of a heavy jet.

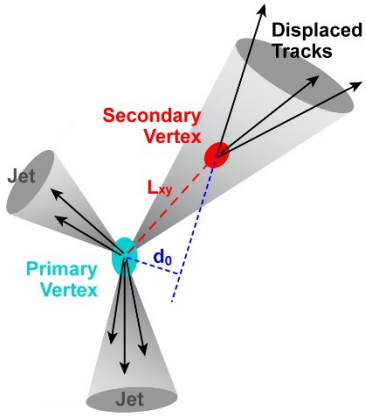


Figure 4: Diagram showing impact parameter d0 for each track

Secondary Vertex (SV)

The SV algorithm fits tracks with high impact parameters to reconstruct the secondary vertex. Tracks are removed from the vertex until the vertex fit $\chi^2 < 4.5$. Variables such as secondary vertex mass and energy fraction are results of SV.

Challenge and Opportunity

These existing algorithms generate track level parameters that do not yield good jet flavor classifying power. Physicists have to create hyperparameters (expert variables) based on experience to abstract discriminating features. In contrast, a well trained multivariate DNN can potentially learn from track level parameters without any manual abstraction, and produce good particle identification results.

The Dataset

The Monte Carlo simulated pp collision track parameter dataset is identical to the dataset previously utilized by Guest et el to evaluate DNN b-tagging performance. MADGRAPH v2.2.3 was utilized to generate parton level events. PYTHIA v6.428 was utilized to simulate parton shower and hadronization. DELPHES v3.2.0 fast simulation was utilized to simulate detector response. Ten million jets were analyzed in this study.

For each jet produced by a pp collosion event, the simulation generates 16 expert level observable variables and 26 track level observable variables. Each simulated jet is identfied as either a b-jet (signal) or not (background). This paper will only consider the 16 expert level variables, shown in Table 2.

Table 2: A list of expert level parameters considered in this study

Jet Variables	Algorithms	General Descriptions
1. Jet pT	N/A	Jet momentum transverse to the beam line
2. Jet eta	N/A	Pseudorapidity
Tracking Variables		
1. Track_2_d0_significance	IP3D	d0 significance of the 2nd highest pT track
2. Track_3_d0_significance	IP3D	d0 significance of the 3rd highest pT track
3. Track_2_z0_significance	IP3D	z0 significance of the 2nd highest pT track
4. Track_3_z0_significance	IP3D	z0 significance of the 3rd highest pT track
5. N_tracks_over_d0_threshold	IP3D	Count of tracks with d0 signif. > 1.8σ
6. Jet_prob	IP3D	Product of all track d0 significance in a jet
7. Jet_width_eta	IP3D	Width of track distribution in eta coordinates
8. Jet_width_phi	IP3D	Width of track distribution in phi coordinates
Vertex Variables		
1. Vertex_significance	SV	Reconstructed 3D vertex displacement divided by vertex error
2. N_secondary_vertices	SV	Count of reconstructed vertices
3. N_secondary_vertex_tracks	SV	Count of tracks associated
4. Delta_r_vertex	SV	Angular separation between vertex and jet
5. Vertex_mass	SV	Mass of all tracks associated to the vertex
6. Vertex_energy_fraction	SV	Fraction of jet energy related to the vertex

Methodology

Existing algorithms

The existing algorithms utilized expert variables separately to classify b-jets. The histogram distributions of expert level variables previously presented by Guest et el and that created by this study are presented in Figure 5 and Figure 6 respectively. The figures are expected to be and are indeed identical because both are generated from the identical dataset. The variables have the same distributions. The visual differences of the two plots is mainly due to plotting styles and histogram bin sizes. Figure 6 groups together charm jets (c-jets) and light jets into 'background' because this paper only considers 2-class classification (b jets vs. others).

Machine Learning – Neural Network

Traditional neural networks have been used for b-tagging at CERN in the past few decades. They serve as non-linear multivariate classifiers for jet flavor. With the recent advancement in Deep Neural Network (DNN) techniques, Guest et el demonstrated that DNN algorithms yield better jet flavor classification performance by extracting deeper features from jet parameters.

A deep neural network consists of many layers (therefore, "deep") of non-linear processing units. In this study, neural network performance in jet flavor classification is demonstrated using a Gated Recurrent Unit (GRU). However, since only expert level variables are included in this study, the GRU network is implemented in "vanilla" mode, much like a "classical" neural network. The implemented GRU network will become a true RNN when track level variables are included as model input in the future. The RNN and GRU architecture is provided in Appendix B.

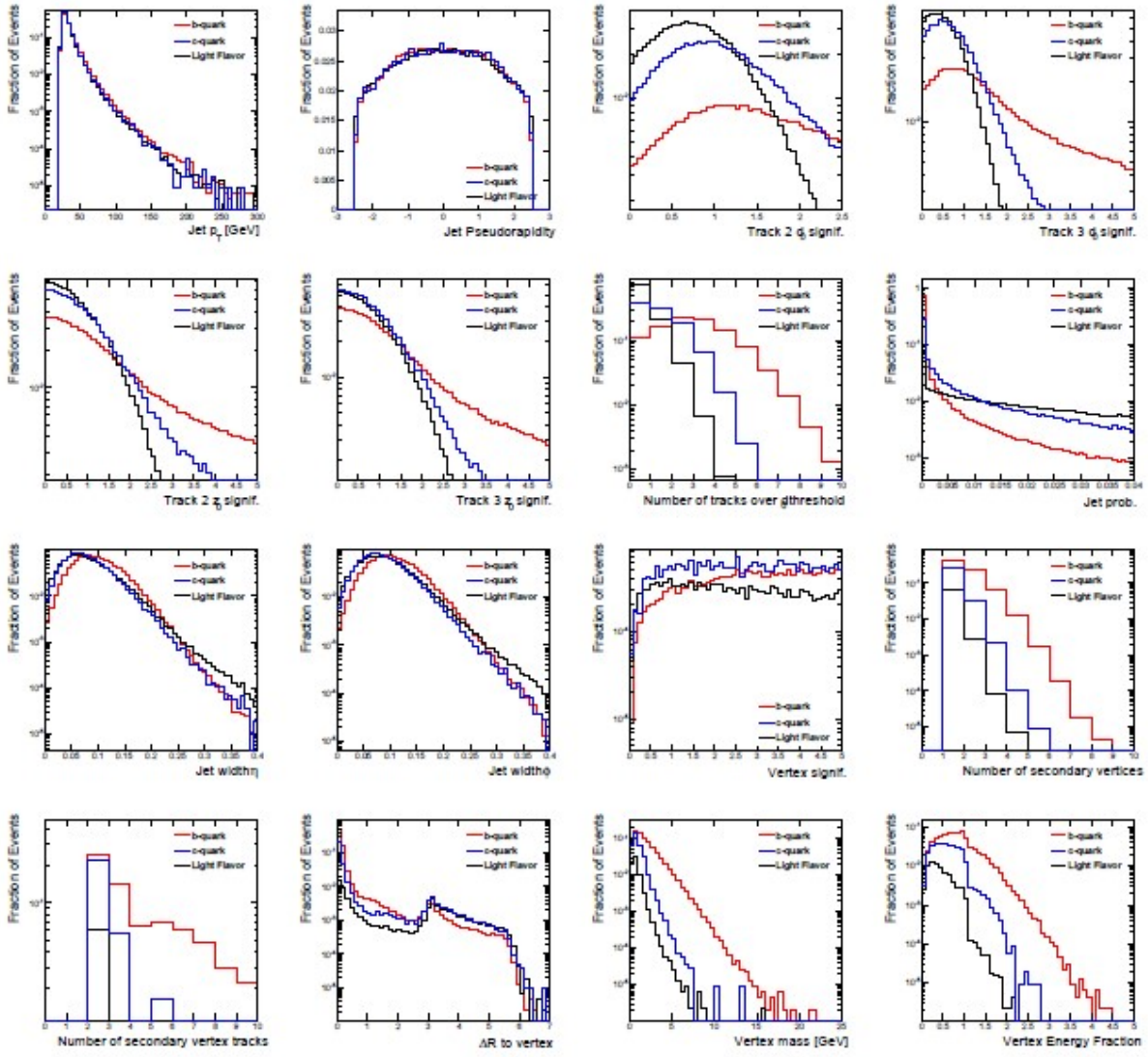


Figure 5: Histograms of expert level variables published by Guest et el.

Expert level variables verfication

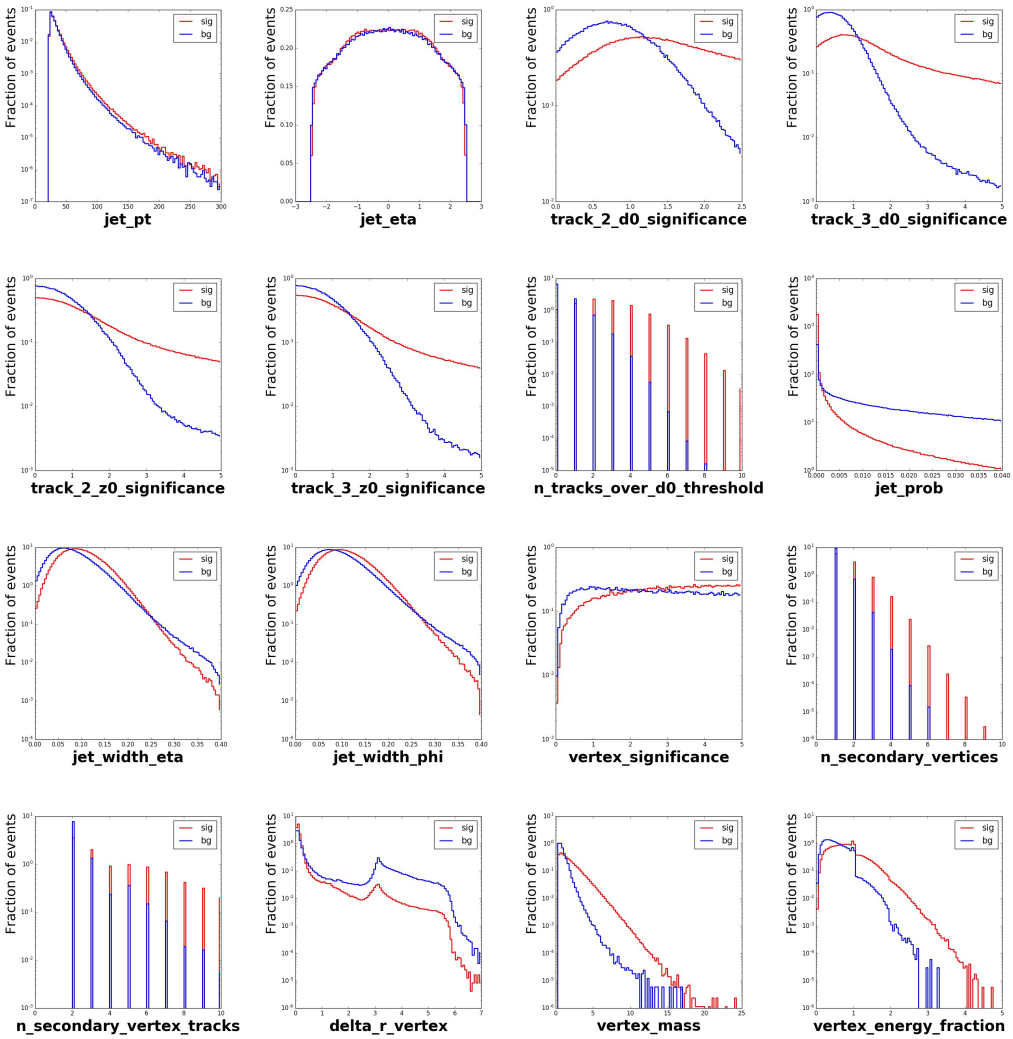


Figure 6: Histograms of expert level variables generated in this study

Results and Discussions

b-tagging performance

Signal / background classification performance can be visualized by plotting the Receiver Operating Charateristics (ROC) curve. The ROC curve is plotted in a 2D space: True Positive Rate vs. False Positive Rate. The confusion matrix [11], and the definition of True Positive Rate nad False Positive Rate is given in Figure 7.

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

$$True\ Positive\ Rate\ (TPR) = \frac{TP}{Actual\ Positives} = \frac{TP}{TP + F}$$

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{Actual\ \boldsymbol{Negatives}} = \frac{FP}{TN + F}$$

Figure 7: Confusion Matrix, and the definition of True Positive Rate and False Positive Rate

The single measurement used for classification performance is the Area Under Curve (AUC) of the ROC curve. The AUC value of 1 indicates that the classifier is expected to be able to distinguish signals from background every time, whereas the AUC of 0.5 indicates that the classifier is not any better than random guessing. AUC is calculated using the composite trapizpoidal rule for integration approximation. [12]

$$AUC = \int_a^b (x) \approx \frac{\Delta x}{2}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]$$

b-tagging performance of each expert level variable

The classification performance of each expert level variable used as a linear discriminator for b-jet signals is shown in Figure 8.

Out of the 16 variables, 'n_tracks_over_d0_threshold' performs the best as a b-tagging classifier, with a AUC = 0.87. This variable counts the number of tracks with d0 significance >1.8σ. 'jet_prob' (jet probability) also yields acceptable classfier performance with AUC = 0.83. This variable is a product of d0 significance of every track in the jet. A high jet probability value indicates that the track is likely to originate from a displaced vertex.

Other IPxD and SV variables have mediocre b-tagging performance. Jet variables 'jet pT' and 'jet eta' have AUC values of 0.53 and 0.50 respectively, and cannot be used alone as b-tagging classifiers.

ROC curves for expert level variables

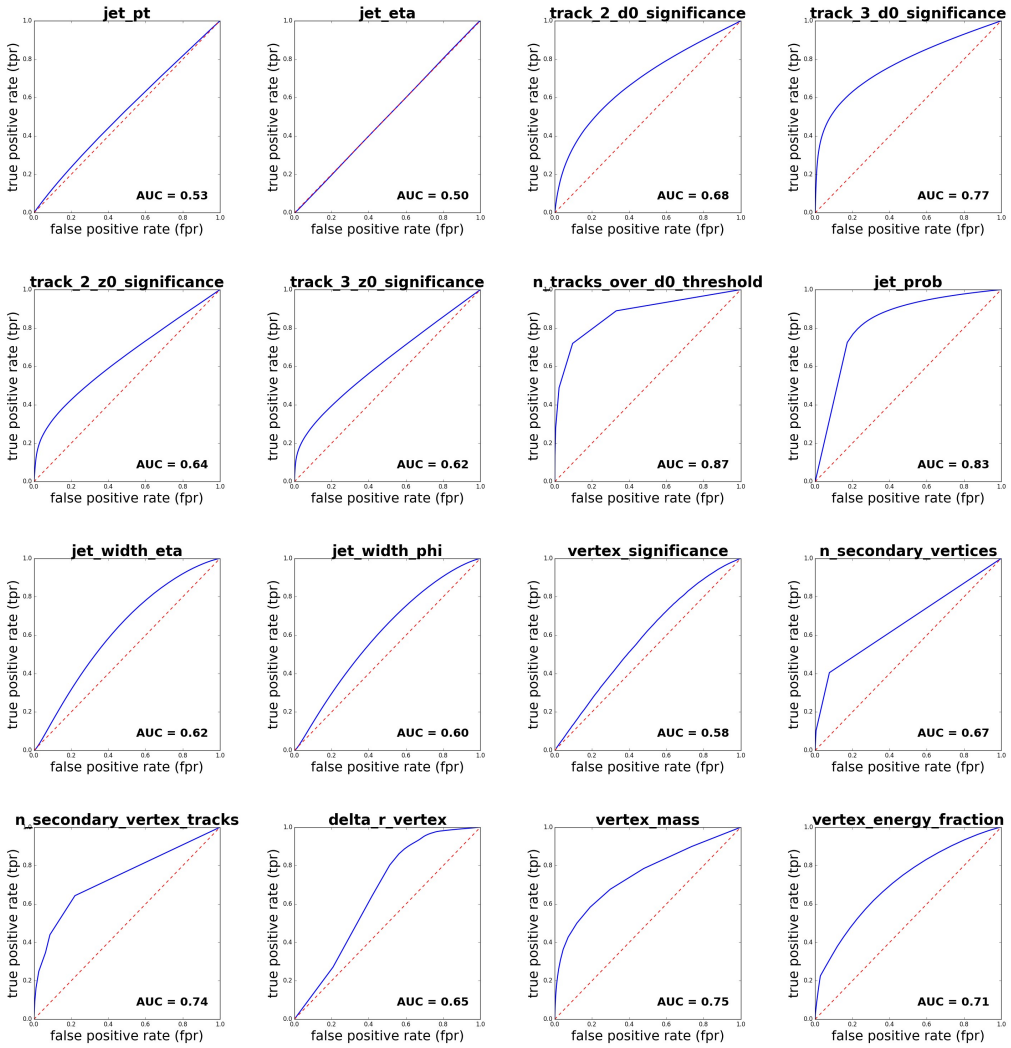


Figure 8: Classification power of each expert level parameter

Neural network performance

Figure 9 shows that the neural network utilized in this study has a better b-tagging performance (AUC = 0.90) than any single expert level parameter. This result demonstrates that utilizing more parameters helps improve classification performance in a high bias neural network environment.

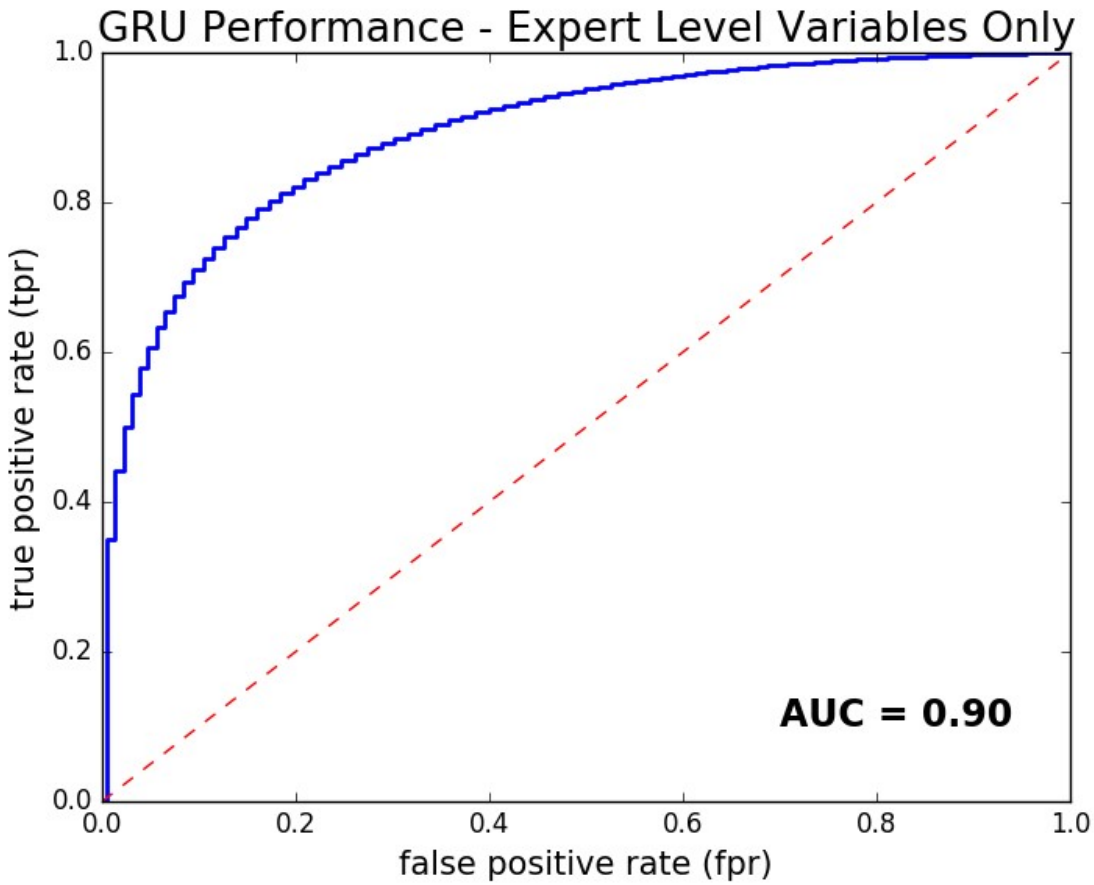


Figure 9: ROC curve for GRU network trained by expert level variables

Conclusion

This study has demonstrated that b-tagging parameters alone, with the exception of "number of tracks with d0 significance > 1.8σ" and "jet probability", generally cannot serve as b-tagging classifiers. A neural network utilizing expert level parameters is a better classifier than using b-tagging parameters alone.

Future Study

A neural network utilizing only expert level parameters is a good classifier, but its performance can be improved further. The following is a list of ideas to consider for the next phase of the study:

Observe the DNN performance using the following as training set:

- Track level input
- IPxD alone or SV alone
- Calorimeter data only
- Calorimeter and expert level data
- Calorimeter, expert level and track level data

Reference

1. Garutti, E. *The Physics of Particle Detectors*. 2012. Available from: http://www.desy.de/~garutti/LECTURES/ParticleDetectorSS12/L1_Introduction_HEPdetectors.pdf
2. The ATLAS Collaboration. *Detector & Technology*. Available from: https://atlas.cern/discover/detector
3. LHC@InternationalMasterClass. *Z-Path - The Higgs Boson*. Available from: http://atlas.physicsmasterclasses.org/en/zpath_hboson.htm
4. Guest, D., et al., *Jet flavor classification in high-energy physics with deep neural networks.* Physical Review D, 2016. **94**. Retrieved from: http://adsabs.harvard.edu/abs/2016PhRvD..94k2002G
5. CMS Collaboration *Search for supersymmetry using razor variables in events with b-tagged jets in pp collisions at sqrt(s) = 8 TeV*. ArXiv e-prints, 2015. **1502**. Available from: http://adsabs.harvard.edu/abs/2015arXiv150200300C
6. Silverman, D. *Higgs Production and Decay Channels*. 2012 [cited 2012 November]. Available from: http://sites.uci.edu/energyobserver/2012/11/26/higgs-production-and-decay-channels/
7. Clement, C. *Properties of the Top Quark*. in *SLAC Summer Institute on Particle Physics (SS104)*. 2004 of Conference.' Retrieved rom: URL
8. The ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*. 2008: Journal of Instrumentation. Retrieved from: http://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08003/meta
9. The ATLAS Collaboration, *Performance of b-jet identification in the ATLAS experiment.* Journal of Instrumentation, 2016. **11**: p. P04008. Retrieved from: http://adsabs.harvard.edu/abs/2016JInst..11P4008A
10. The D0 Collaboration, *Observation of Single Top Quark Production*. 2009. Retrieved from: https://www-d0.fnal.gov/Run2Physics/top/singletop_observation/
11. Raschka, S. *Confusion Matrix*. Available from: http://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/
12. scipy.org. *numpy.trapz*. Available from: https://docs.scipy.org/doc/numpy/reference/generated/numpy.trapz.html
13. Potamianos, K., *The upgraded Pixel detector and the commissioning of the Inner Detector tracking of the ATLAS experiment for Run-2 at the Large Hadron Collider*, The ATLAS collaboration, Editor. 2016. Retrieved from: https://arxiv.org/abs/1608.07850
14. Vogel, A. and T.A. Collaboration, *ATLAS Transition Radiation Tracker (TRT): Straw Tube Gaseous Detectors at High Rates.* CERN notes, 2013. Retrieved from: https://cds.cern.ch/record/1537991
15. Wilkens, H. and T.A.L. Collaboration, *The ATLAS Liquid Argon Calorimeter: an overview.* Journal of Physics, 2009. **Conference Series 160 (2009)**(012043). Retrieved from: http://iopscience.iop.org/article/10.1088/1742-6596/160/1/012043/pdf
16. Sliwa, K. *"ATLAS Overview and Main Results"*. ArXiv e-prints, 2013. **1305**. Available from: http://adsabs.harvard.edu/abs/2013arXiv1305.4551S
17. Ng, A., *Machine Learning - Lecture 11*. Coursera.org. Retrieved from: https://www.coursera.org/learn/machine-learning
18. Karpathy, A., *The Unreasonable Effectiveness of Recurrent Neural Network*. 2015. Retrieved from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
19. Chung, J., et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. ArXiv e-prints, 2014. **1412**. Available from: http://adsabs.harvard.edu/abs/2014arXiv1412.3555C
20. Manning, C. and R. Socher, *Natural Language Processing with Deep Learning CS224N/Ling284 - Lecture 8: Recurrent Neural Networks*. 2017. Retrieved from: http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture8.pdf
21. Manning, C. and R. Socher, *Natural Language Processing with Deep Laerning CS224N/Ling284 - Lecture 9: Fancy Recurrent Neural Networks for Machine Translation*. 2017. Retrieved from: http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture9.pdf

Appendix A: ATLAS detector configuration

b-jets are identified by analyzing the data produced by the detector, which depends on the detector configuration. The ATLAS experiment at CERN LHC has the following configuration [8]:

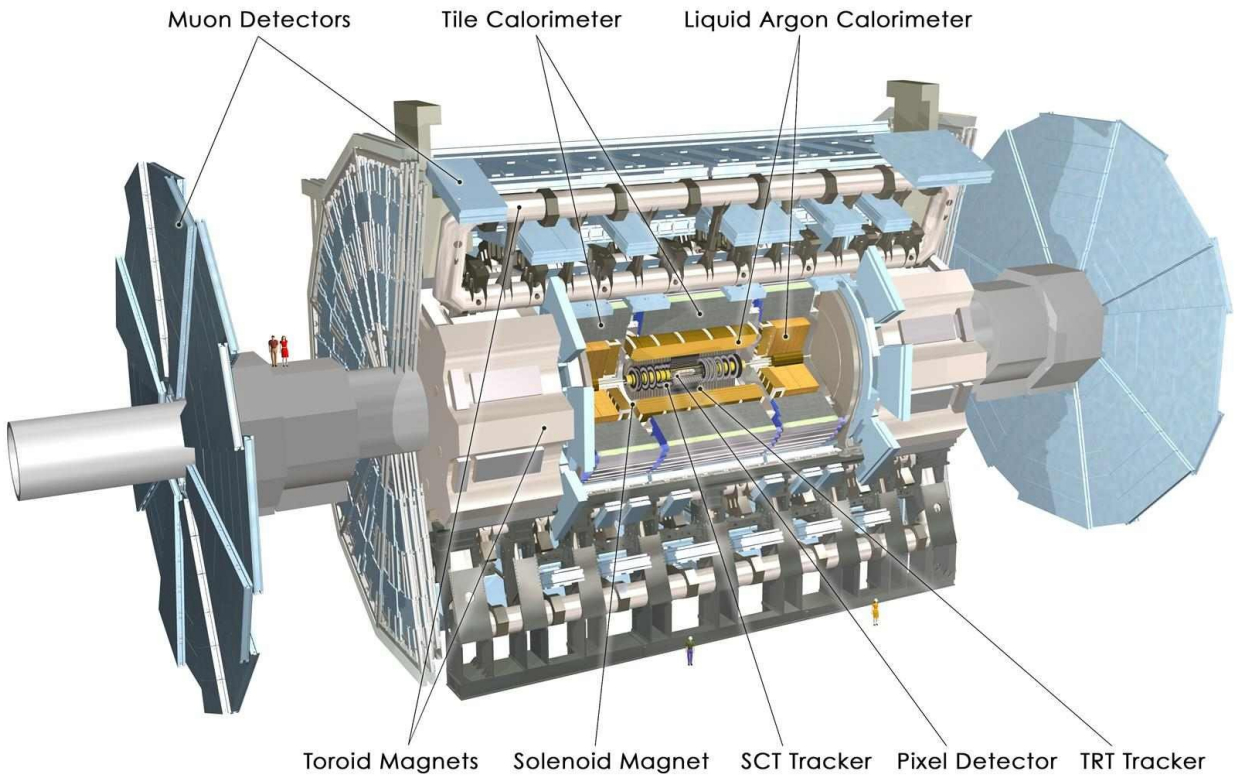


Figure A: ATLAS detector anatomy

The two main class of raw data collected are: 1) Tracks of electrically charged hadrons, detected by "trackers", and 2) energy of hadrons, detected by "calroimerters". The trackers are placed inside the Central Solenoid Magnet, which generates a uniform 2T magnetic field along the beamline (z-axis). The magnetic field bends charged particles for momentum measurement. The calorimeters are placed inside the Barrel Toroid and the End-Cap Toroid, which produce a uniform 4T magnetici field along the beamline. The outermost detector is the muon detector, which is not be covered in this paper.

Trackers

Trackers detect the trajectory of charged particle.There are three layers of trackers: Pixel Detector, Semiconductor Trackor and Transitional Radiation Tracker. Their distance from the beamline is shown in Figure B. [13]

The innermost layer (closest to the primary vertices, the pp collision points) of tracking detector is the silicon pixel detector. It has about 80 million pixels with a cylindrical area of $1.7m^2$ in a 3 cylindrical barrel configuration. It works similar to detectors inside the digital camera. When a charged particle passes through the detector, it liberates electrons in the semiconductor, producing a signal, which is then recorded.

The next layer is the semiconductor tracker. It has about 6 million channels with a cylindrical area of 6 million m^2 area in a 4 cylindrical barrel configuration.It is similar to the pixel detector, but with a lower resolution.

The third layer is the Transition Radiation Tracker (TRT). [14] It consists of 4-mm diameter drift tubes made of Kapton and reinforced carbon fiber. The tube consists of Xenon gas and, in the center, a gold plated tungsten wire. When a charged particle passes through the tube, it ionizes the Xe gas to create more electrons. Free electrons then drift towards the gold wire and create an amplified signal. The charged particle also create drift radiation when it passese through polymer fibers between the straws. The TRT can distinguish (classify) charged particles by analyzing the relative strength (count of drift electrons) of signal.

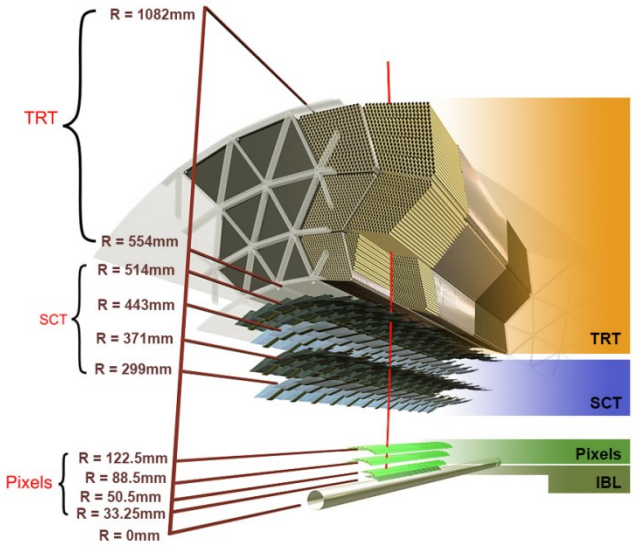


Figure B: Tracking detector range from beamline

Calorimeters

There are two main layers of calorimeters in ATLAS: The Liquid Argon (LAr) Calorimeter and the Tile Hadronic Calorimeter (TileCal). The two calorimeters together can absorb most known particles, except neutrinos and muons. [15]

The LAr Calorimeter has four main components: EM Barrel, EM end-cap, Hadronic end-cap and forward calorimeter. Energy of eletrons and photons are deposited in the EM calorimeters when they interact with the active medium of LAr. The energy of hadrons is deposited into the thick copper absorbers in the hadron calorimeters through nuclear interactions.

The TileCal is made of iron plates (passive material for particle absorbsion) and plastic sciintilators (active material, producing photons as charged particles pass through). The energy of photons produced in the plastic scintillators is proportional to the energy depostited by the particle. The photomultiplier tubes amplifies the photon signal and the digitized signal is recorded.

Trigger

ATLAS detects one billion pp collisions per second, generating 60 TB data per second. The trigger system down select to 100 events per second for storage. The down select criteria is complicated. Example events for data storage include high particle trasverse momentum or di muon events. [16]

For more information, refer to reference [8].

Appendix B: Machine Learning, deep neural network and GRU architecture

A neural network is a class of multivariate mathematical model that, upon satisfactory training and model tuning, can perform abstraction tasks given the input data. Similar to linear regression, the neural network is a supervised machine learning algorithm. It tries to minimize the error (variance in linear regression) while relating the input data to the output data during training. If the model is well trained, it will provide good output prediction given new input data.

Below is an example of a "classical" neural network [17, 18]:

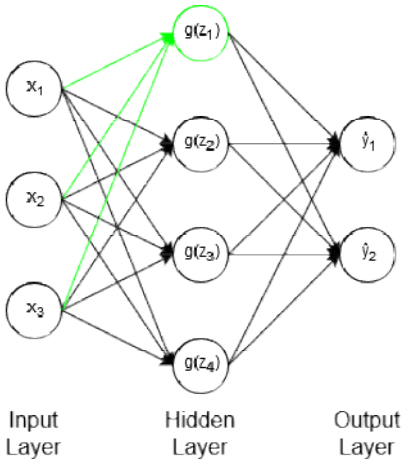


Figure C: "Classical" Neural Network configuration

The input layer can be considered as an input vector. In figure C, the input vector is a 3 dimensional vector [x1, x2, x3]. Each arrow is a vector transfer function to the next layer. In other words, the arrow can be thought of as multiplying the input vector to weightings, such that (using the green lines from Input Layer to Hidden Layer as an example):

$$z_1 = \theta^T x = \theta_{11}x_1 + \theta_{12}x_2 + \theta_{13}x_3$$

Then this result is multiplied to a non-linear activation function, the sigmoid function for example, inside the circles. Using the green circle as an example:

$$g(z_1) = sigmoid(z_1) = \frac{1}{1 + e^{-z_1}}$$

The output layer is the output vector estimated by the model, referred to as $\hat{y}$ or $h_\theta(x)$. In Figure C, it is a 2 dimensional vector [$\hat{y}1$, $\hat{y}2$], e.g. [Signal, Background]. The output vector [1,0] indicates a signal and [0,1] indicates a background. The estimated output vector by feeding the input vector through the

model can take on values between 0 and 1, and can be considered as the probability of signal or background of the given input vector.

This above process going through the network from left to right is called “forward propagation”.

Now that we have the expected output vector calculated, during training, we can compare this to our answer and get the prediction error of our expected output. We also want to update the weighting θ from right to left along the network, so that the model can become more accurate in predicting output in the next iteration. This process is called “back propagation” in supervised training.

The error of the model, or the ‘distance’ between the true outcome and the expected outcome, is measured by the cost function J. The cost function of the neural network utilized in this study is quite complicated and will not be shown here. In linear regression, where m is the total number of training examples, the cost function is the sum of error:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Optimization / Back Propagation

The goal for machine learning is to minimize the cost function by updating the model weight θ, such that the difference between the expected value and the true value is small. Stochastic Gradient Descend, one of the many optimization schemes, is shown below:

$$Repeat\ for\ each\ training\ example\ \{\theta = \theta - \alpha\frac{\partial}{\partial\theta}J(\theta)\}$$

Each of the model weight of the transfer function is updated throughout the network from right to left. This process is called “back propagation”.

Shown above is the simplest form of neural network in machine learning. Deep neural network loosely means that the neural network model has many layers, thus theoretically more capable to abstract features from the input dataset.

Recurrent neural network

The Recurrent Neural Network algorithm is a subclass of Deep Neural Network [18]. The input and output data can be in the form of sequences, which the classical neural network cannot handle. RNN can also work well for non-sequential data. Examples of recurrent neural network configurations are shown in Figure D. Each red box is an input vector, each green box is the hidden layer with RNN neurons (or units), and each blue box is an output vector. The “one to one” configuration is considered the “vanilla mode”, similar, but not identical, to the “classical” neural network in Figure C, tilted 90 degrees anti-clockwise.

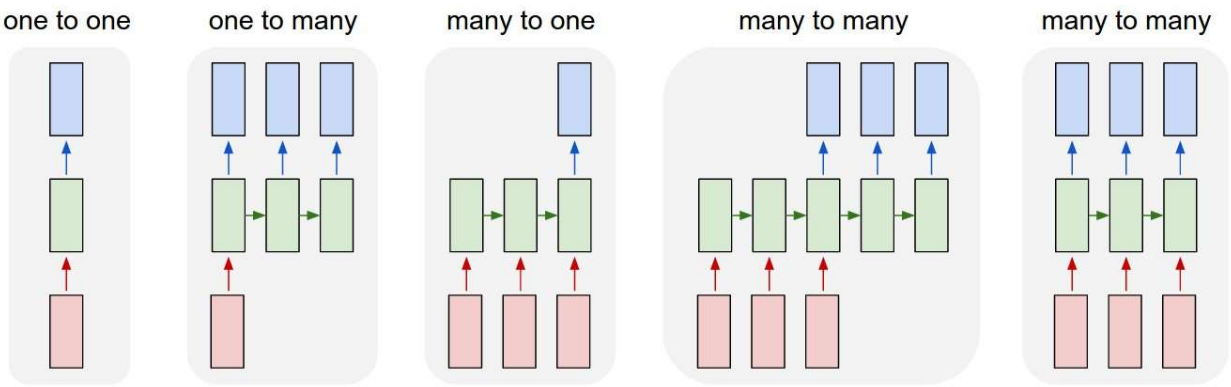


Figure D: Recurrent Neural Network configurations

Using the "many to many" configuration as an example, the general model for RNN is shown in Figure E. The function f is a sigmoid function (σ), and function g is tanh. W, V and U are weights, while b and c are bias units.

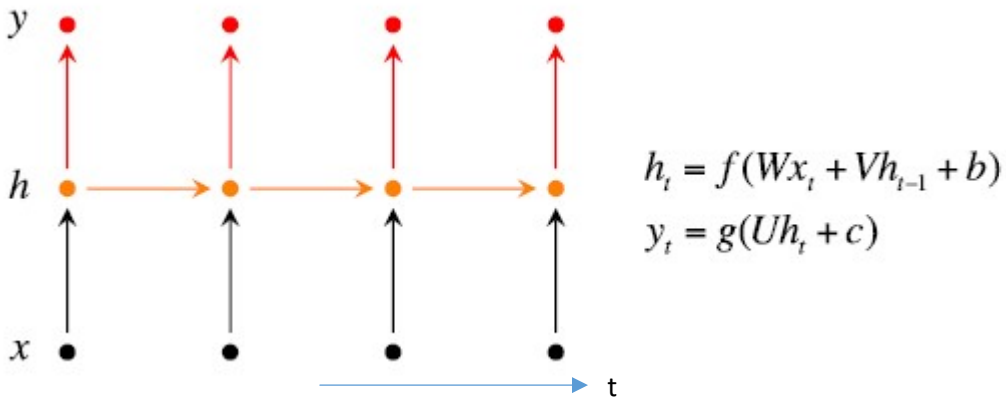
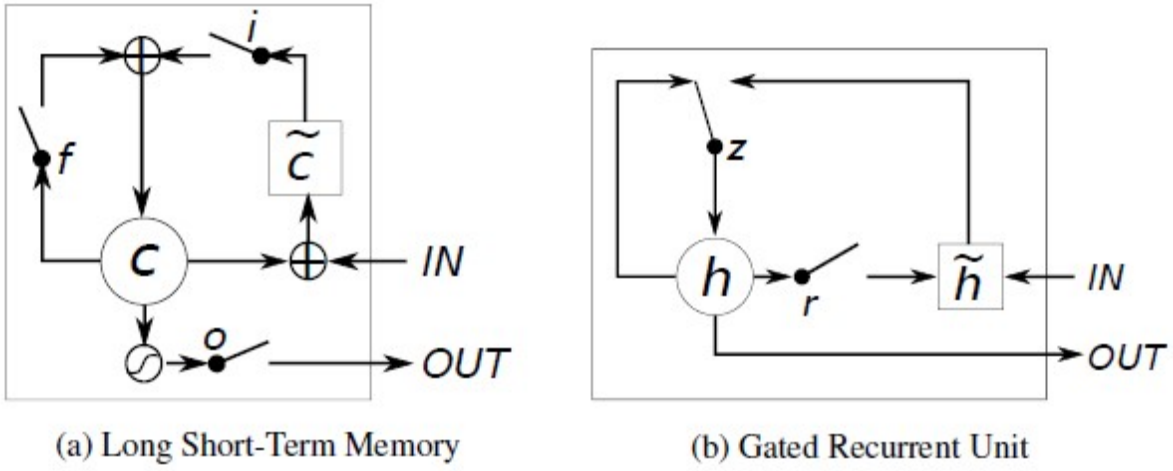


Figure E: "many to many" RNN configuration

The neurons in the hidden layer can be of various configurations. The Long Short Term Memory (LSTM) is probably the most popular RNN in recent years. The Gated Recurrent Unit (GRU) [19] used in this paper can be considered a simpler version of LSTM. The neural network utilized in this study implements a GRU in "vanilla" mode, shown as "One to One" in Figure D. When sequential input data is utilized, the neural network implemented in this study will become a true RNN in the "Many to One" configuration. A side by side comparison between LSTM and GRM is shown in Figure F.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure F: A figure showing configuration of a) LSTM and b) GRU

The GRU model is shown below. [20, 21]

Update gate:	$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$
Reset gate:	$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$
New memory content:	$\tilde{h}_t = \tanh(Wx_t + r_t \circ Uh_{t-1})$
Final memory content:	$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$

The LSTM model is shown below.

Input gate:	$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$
Forget gate:	$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$
Output:	$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$
New memory cell:	$\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1})$
Final memory cell:	$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$
Final hidden state:	$h_t = o_t \circ \tanh(c_t)$

Algorithm implementation

The neural network in this study is implemented with Keras using Theano backend. The input 3D tensor has this dimension: [Jets, Tracks in a Jet, Variables] (Figure G)

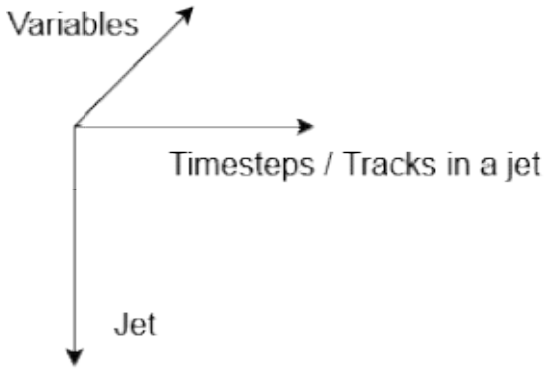


Figure G: RNN input tensor dimension

Note that since expert level data only has one set of variables per jet, the dimensional length of second dimension (Timesteps / Tracks in a Jet) is 1. Therefore, the GRU network implemented by this study runs in "vanilla" mode, similar to a "classical" neural network. However, the implemented network will become a true RNN when track level data is utilized as input.

The back propagation parameter update utilizes the Adam optimizer. The network is run using a batch size of 512, with 5 epochs.

Appendix C: Python scripts, dataset location and set up instructions

Refer to detail instructions in Google drive.
https://drive.google.com/file/d/0B3qwNGluXsHSYTVHa19NSjE3a1U/

Programming language: Python 2.7

Required packages: Numpy, h5py, matplotlib, theano, keras

Dataset location: Tev01 scratch space

.h5 file: /phys/groups/tev/scratch4/user3s/kaifulam/dguest/gjj-pheno/v1/Julian/gjj_Variables.hdf5

Script location: https://github.com/kaifulam/Gjj-pheno

To plot single jet histograms and ROC curves:

1) Run track_collect_high_rev2.py, which outputs 3 .csv files for histogram plotting: signal, background and bin.
2) In the same directory as the .csv files, run track_collect_high_plot.py to plot histograms
3) In the same directory as the .csv files , run track_high_ROC_plot_OneFig_Rev3.py to plot ROC curves

For machine learning:

1) Run ML_try2_high.py, which outputs 2 .csv files: true positive rate and false positive rate
2) Run ML_ROC_plot_Rev2.py, which output ROC curve for machine learning model