Classification of Higgs Boson Tau-Tau decays using GPU accelerated Neural Networks

Mohit Shridhar
Stanford University
mohits@stanford.edu, mohit@u.nus.edu

Abstract

In particle physics, Higgs Boson to tau-tau decay signals are notoriously difficult to identify due to the presence of severe background noise generated by other decaying particles. Our approach uses neural networks to classify events as signals or background noise.

1 Introduction

The discovery of the infamous Higgs Boson in 2012 was one of the most significant scientific milestones of the past century. The particles existence was predicted nearly 50 years ago, but verifying it experimentally became a monumental challenge. The search brought together more than 36 countries and cost nearly 13.25 billion dollars.

The existence of the particle spawned a number of other challenges including the study of how it decays into other particles (an essential characterization in particle physics). Physicists are specifically interested in the tau-tau decays of Higgs Bosons. But unfortunately, these events are deeply buried in severe background noise. The objective of this study is to apply supervised learning techniques to carryout simple binary classification: tau-tau decay vs. noise. Primarily, we will focus on using neural networks to methodically classify input measurements and benchmark their performance against other learning models.

Prior work in the field mainly revolves around the use of gradient boosting [2] or random forests. P. Sadowski et al. demonstrated that deep neural networks are well suited for improving the discovery significance of tau-tau decays. This classification problem was posed as a Kaggle challenge [1] in 2014.

2 Dataset

As the decay phenomenon is still a relatively new area of research, we didn't use any real data from particle accelerators. But instead, we used simulated datasets generated by CERN through the ATLAS experiments. These simulations were based on our current understandings of the Standard Model and empirical observations from various experiments.

The input vector consists of 30 features, which represent conditional parameters and measurements taken from various instruments. Typically the raw data from accelerators is heavily skewed, i.e. the occurrence of signal events (tau-tau decay) is vastly overshadowed by background events. The simulated dataset has a higher ratio of signal to background events in order to ease the process of training models with limited samples. So to compensate for this, the dataset provides a relative weighting factor for each datum that specifies the significance

of the event. Formally, given a training dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1, w_1), ..., (\mathbf{x}_n, y_n, w_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^{30}$ is a vector of input features, $y_i \in \{s, b\}$ is the label: signal or background, and $w_i \in \mathbb{R}^+$ is a weighting factor. The weights w_i also have a physical significance [1] :

$$\sum_{i \in \mathcal{S}} w_{i,s} = N_s \quad (1) \qquad \sum_{i \in \mathcal{B}} w_{i,b} = N_b \quad (2)$$

Where N_s and N_b are the expected occurrences of the each event. These values were computed by taking the conditional densities $p(\mathbf{x}_i)$ of the input features and dividing it by the instrumental densities $q(x_i)$:

$$w_i \approx \begin{cases} p_s(\mathbf{x}_i)/q_s(x_i), & \text{if } y_i = s, \\ p_b(\mathbf{x}_i)/q_b(x_i), & \text{if } y_i = b. \end{cases} \quad (3)$$

The performance of the system was quantified using a metric provided by the Kaggle challenge called the AMS objective function [1]:

$$\text{AMS}_c = \sqrt{2\left((\mathbf{s} + \mathbf{b} + \mathbf{b}_{reg}) \ln\left(1 + \frac{\mathbf{s}}{\mathbf{b} + \mathbf{b}_{reg}}\right) - \mathbf{s}\right)} \quad (4)$$

Where $\mathbf{s}$ and $\mathbf{b}$ are the expected number of signal events and background events respectively, selected by the learning model. Generally, higher AMS scores correspond to better classifications. The AMS score of the winning Kaggle submission was 3.85060 in 2014.

3 Methods

Our approach to the classification problem involves the use of dropout neural networks to statistically model the data. Due to the complex relationship between the input and output of the system, neural networks were deemed to be the appropriate learning model to tackle this problem. The fact that the input is heavily biased with background events makes the model prone to over-fitting issues [1]. Dropout neural networks have been proven to reduce over-fitting problems faced by standard neural networks [3]. Consider a standard feed-forward neural network with perceptron nodes:

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}) \end{aligned} \quad (5)$$

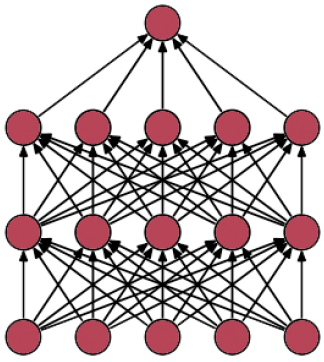

Figure 1: Standard Feed-Forward Network

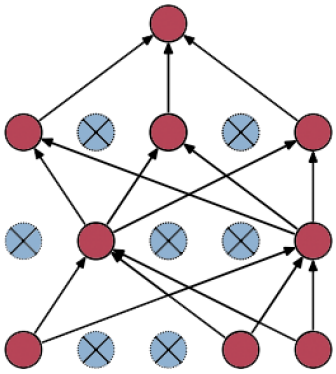

Figure 2: Dropout Network

where $l \in \{1, ..., L\}$ is a hidden layer among L layers, $\mathbf{y}^l$ is the output from layer l, $\mathbf{w}_i^{(l)}$ are the weights, and $\mathbf{b}_i^{(l)}$ are the biases. The nodes of a dropout neural network have a distinct probability of being dropped during training:

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p), \\ \tilde{\mathbf{y}}^l &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)} \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \tilde{\mathbf{y}}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}) \end{aligned} \tag{6}$$

where $r_j^{(l)}$ is a Bernoulli random variable with probability p of being 1. During every epoch, we are essentially sampling a sub-network from the larger deep network. In a way, the dropout technique is performing stochastic regularization to reduce over-training, i.e. the weights are being scaled using p as a factor: $\mathbf{w}_i^{(l)} \rightarrow p\mathbf{w}_i^{(l)}$. When using the model to predict labels using real data, we set $p = 1.0$ to include all the nodes. N. Srivastava et al. showed that this process results in a huge performance improvement across various neural architectures without using hyper-parameters tailored specifically for the architecture.

4 Experiments & Results

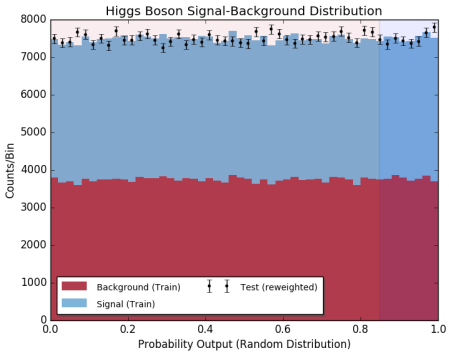

Figure 3: Random Distribution

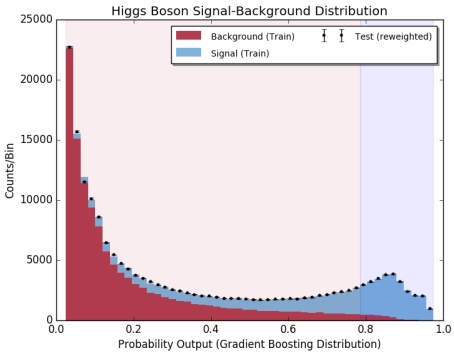

Figure 4: Gradient Boosting

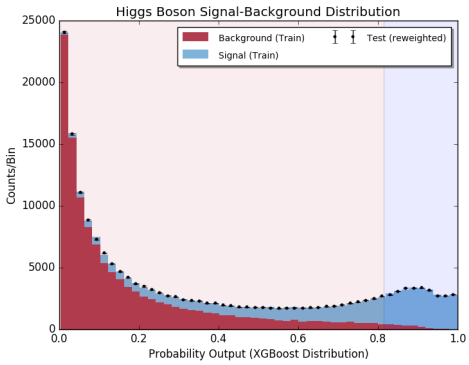

Figure 5: XGBoost

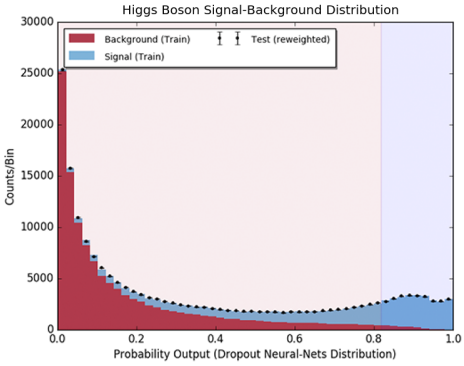

Figure 6: Dropout Neural Networks

We implemented a bag of dropout neural networks using Google's TensorFlow deep-learning library. The performance of the proposed model was compared with other popular learning models used

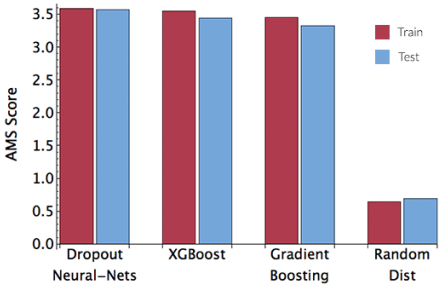

Figure 7: Accuracy Benchmark

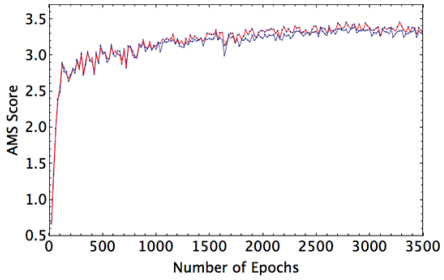

Figure 8: Over-Training

by the Kaggle challange participants: XGBoost & Gradient Boosting. We decided to train an ensemble of neural networks instead of just a single complex network to further reduce over-fitting [4].

From the input vector of 30 features, 4 were removed via manual-selection to prevent over-fitting. The input vector was normalized to have a mean: 0 & stddev: 1. The final learning model was an ensemble of 15 dropout neural networks with 3 hidden layers consisting of 450 neurons each: 26x450x450x450x2. The outputs were two softmax neurons, which individually predicted the probability of a sample being a signal or an event. An input was classified as a signal if the output of the softmax signal neuron was greater than 0.5575. Results from each network (from the bag) were simply amalgamated by averaging the output probabilities. The model was trained with 2-fold stratified cross-validation with random shuffling to prevent overfitting. Additionally, the cost function was set to reduce the mean of the cross-entropy loss. Each neuron in the hidden layer had a 0.5 probability of being dropped.

Figure 3 shows the output probability distribution of a model that randomly classifies an event with an even probability of 0.5. Such a purely random classifier results in an AMS score of 0.6 on average. The output of the other three models show a distinct segregation between the signal and background events, which correspond to the accuracy of the process. In each instance, if the output probability of the event being a signal was greater than 0.8, the event was classified as a signal. Figure 7 shows that our bag of dropout neural networks perform marginally better than XGBoost and Gradient Boosting ($\sim$ 0.1 AMS higher). The final score of the network ensemble averaged-out to $\sim$ 3.56 AMS; the winning Kaggle submission ($13,000 prize) had a score of 3.85. Our score roughly corresponds to a LB score (L2-norm) of 90% accuracy. The model was also validated on a separate dataset of sample size $550,000$.

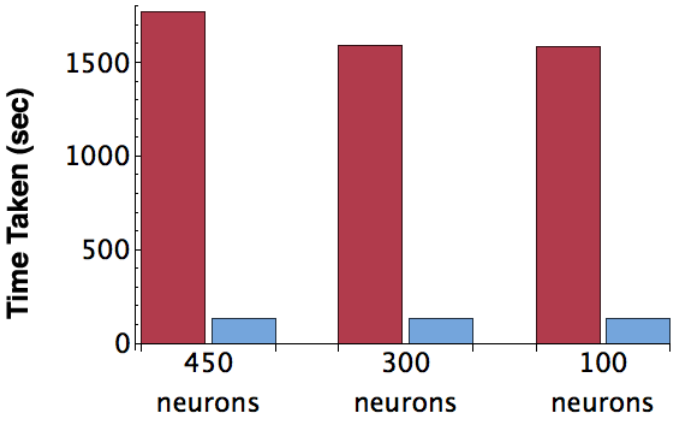

Figure 9: CPU vs GPU: Time taken to complete 100 epochs

In terms of scalability, our dropout neural networks definitely under-performs in relation to XGBoost and Gradient Boosting. The XGBoost model took about 4 minutes to train on a quad-core CPU. Whereas a single neural network took nearly 10 hours to achieve similar performance. The training dataset was composed of $350,000$ samples, but the ATLAS experiment generates petabytes of data, which could be computationally very expensive to train neural networks with.

GPU acceleration was used to speed-up the training process. We setup our dropout neural networks on an Amazon EC2 instance with 4 GPUs each containing $1,536$ CUDA cores and 4GB of memory. As shown in Figure 9, the GPU cluster was nearly 14 times faster than a standard quad-core CPU.

5 Conclusion

Our proposed solution to the Higgs Boson classification problem had an accuracy of $\sim 90\%$. Although, the testing dataset was considerably small compared to the amount of actual data produced by CERN, so it is still possible that the model was over-trained. In terms of improvements, we could consider boosting the dropout neural networks [5] instead of simply averaging results from the bag. Another neglected aspect was feature engineering. Understanding the physical significance of the input measurements could help identify the features that are over-training the model. Furthermore, PCA can be used to reduce the dimensionality of the input vector to improve performance and reduce the time taken to train the model.

6 References

[1] ATLAS collaboration (2014). *Dataset from the ATLAS Higgs Boson Machine Learning Challenge* 2014. CERN Open Data Portal. DOI: 10.7483/OPENDATA.ATLAS.ZBP2.M5T8.

[2] Jerome Friedman. Greedy Function Approximation: A Gradient Boosting Machine, 1999.

[3] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 2014, pages 1929-1958.

[4] Ury Naftaly, Nathan Intrator and David Horn. Optimal ensemble averaging of neural networks. *Comput. Neural Syst.* 1997, pages 283-296.

[5] Holger Schwenk, Yoshua Bengio. Boosting Neural Networks. *Comput. Neural Syst.* 2000.

[6] Rupesh Kumar Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, Jrgen Schmidhuber. Compete to Compute. *NIPS* 2013.

[7] Peter Sadowski, Pierre Baldi, Daniel Whiteson. Searching for Higgs Boson Decay Modes with Deep Learning. *Advances in Neural Information Processing Systems* 2014, pages 2393-2401.