UNIVERSITY OF Southampton

COMP6208: Advanced Machine Learning

Classification for the Masses

William Thomas
Electronics and Computer Science
wlt1g16@ecs.soton.ac.uk

Jamie Outram
Electronics and Computer Science
jpo1g16@ecs.soton.ac.uk

Shabeer Rauf
Electronics and Computer Science
smr3g16@ecs.soton.ac.uk

Yanislav Donchev
Electronics and Computer Science
ydd1g16@ecs.soton.ac.uk

Antoine Poullet
Electronics and Computer Science
ap6u16@ecs.soton.ac.uk

1 Introduction

The ATLAS experiment at the LHC was looking for evidence of the Higgs Boson, the particle which gives mass to other particles in the standard model of physics. The data for the challenge were simulated events modelling the expected background and signal events. The challenge aims to produce a classifier which can effectively distinguish between signal and background events in a way which maximises the statistical significance of observing those events.

We applied many approaches to this challenge, including XGBoost, random forests and Deep Neural Networks (DNNs). We produced DNN with a private set score of 3.764, which would have placed 7th in the original competition and is, to the best of our knowledge, the best for a single model - all of the higher scores for which we could find the method used ensemble methods. We believe our success was due to: a novel loss function, softAMS; hyperparameter tuning; testing a variety of methods; and writing utilities to allow quick development and testing.

Kaggle's scoring used a prediction of the statistical significance of the events given the classifier's accuracy, called Approximate Median Significance (AMS). It is given by equation 1, where s is the sum of the weights of the correctly labelled signal events, b is the sum of the mislabelled background events and b_{reg} is a regularisation term set to 10.0.

$$\text{AMS} = \sqrt{2\left((s+b+b_{reg})\ln\left(1+\frac{s}{b+b_{reg}}\right)-s\right)} \tag{1}$$

The submission is an ordered list of predictions based on confidence that they are signal events and the scoring function creates a curve by iterating over each possible threshold to find the optimal AMS score [1]. The winner of the original challenge was an ensemble of 70 deep neural networks which achieved a score of 3.805 on the private data-set, whilst XGBoost was able to achieve a score of 3.761 with strong feature engineering but far fewer resources [2].

2 Imputation methods

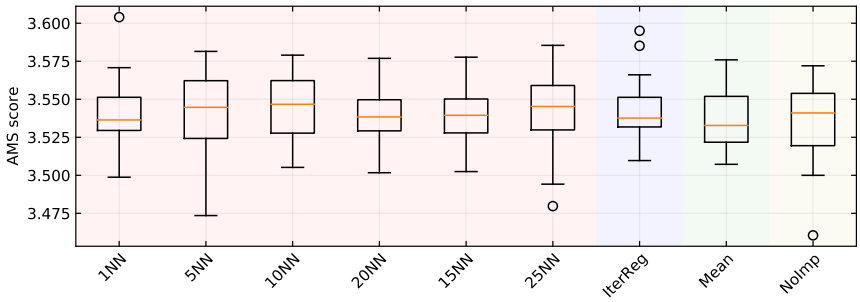

Figure 1: XGBoost 1 (See Figure 2) over 30 seeded training runs showing AMS with kNN (xNN where x = k), Iterative (IterReg) and Mean imputation methods

.

Most of the missing data in the provided data-sets were a result of the physical jets, given by *PRI_jet_num*, not occurring during the event. There were between 0 and 3 jets per event, with fewer jets resulting in more missing data. Due to the physical significance of the missing data and the

high percentage of missing values, they were not imputed. This left only the estimated Higgs boson mass for imputation, with the small effect shown in figure 1.

3 Model Comparison

Several simple models were tested in order to provide insight into potential solutions. The AMS scores of these models are shown in figure 2. Logistic regression produced a relatively low score as it is not suited to non-linear problems. The naive Bayes classifier also produced a low AMS because it operates under the false assumption that the features are independent. Decision trees provided a high accuracy on the training data but failed to generalise. Ensemble decision tree methods, such as random forests and boosted trees, alleviate overfitting. These methods have higher AMS scores and are candidates for fine-tuning. Multi-layer perceptrons (MLPs) of different depths were also tested, with the deeper networks scoring nearly as well as XGBoost and random forest.

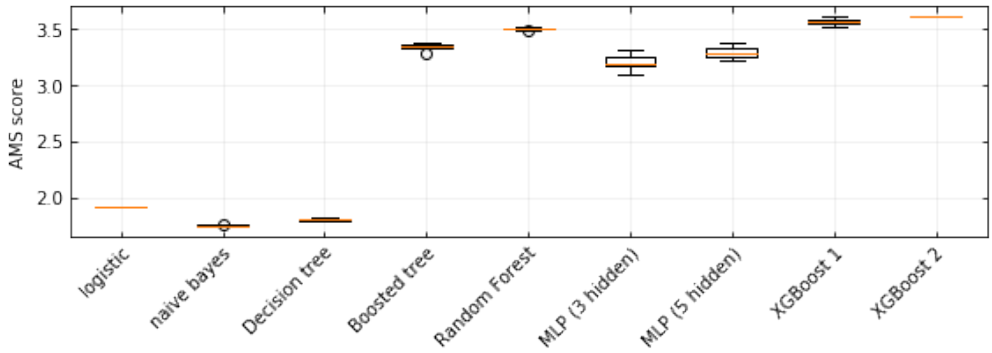

Figure 2: Comparison of AMS scores for different machine learning models.

We tried splitting the data by jet number and training a model for each split; however, at best, these only approached the performance of a single DNN, with AMS scores of 3.70 at best after softAMS training.

3.1 XGBoost

XGBoost is an implementation of gradient boosted decision trees, newly introduced at the time of the Higgs boson challenge. Though it did not win, XGBoost received high praise for its simplicity, going toe to toe with computationally expensive ensemble neural-net and decision-tree solutions.

The AMS of our best XGBoost models are shown in Figure 2. Both use nearest neighbour imputation on the training data-set, and all C.A.K.E, phi and engineered phi features were excluded as they cause the model to over-fit easily, leading to a much lower AMS. The difference between the two models is in their hyperparameter tuning (See Appendix Tables 3 and 4). The use of AMS as an alternative objective function was explored but did not yield good results.

3.2 Deep Neural Networks

Whilst DNNs have provided ground-breaking advancements in areas of ML such as Computer Vision and Natural Language Processing (NLP), their superiority over traditional models when it comes to tabular data is still subject to debate. The winner of the Higgs Boson competition was a

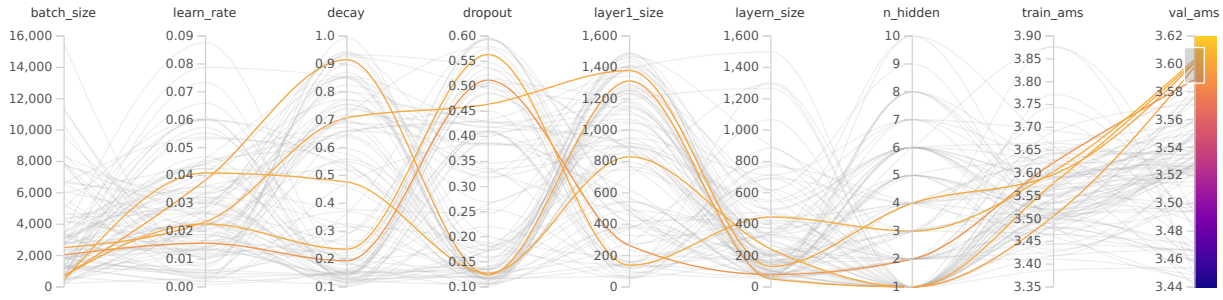

Figure 3: The initial hyperparameter search results. Each line represents one combination and leads to the respective training and validation AMS scores on the right. A total of 149 networks were tested (gray lines) and the best performing ones ($val\_ams \geq 3.59$) are highlighted.

DNN but a submission involving an implementation of Gradient Boosted trees managed to perform comparably while using fewer resources.

A paper submitted to ICLR 2020 claims to have found an architecture that could consistently outperform more traditional models with minimal trade-offs [3]. This new model uses the recently developed entmax transformation to relax decision trees and make them differentiable to apply end-to-end optimisation [4]. It also builds on the advancements made by CatBoost which achieved state-of-the-art performances on numerous tabular data-sets by applying gradient boosting on oblivious decision trees [5]. Results of running this network on the dataset can be found in the appendix.

4 Hyperparameter Tuning

According to [6], a model's performance is highly sensitive to the learning rate and layer size, and moderately sensitive to the network depth, regularisation weight and batch size. The hyperparameters chosen for optimisation are shown in figure 3. The learning rate is multiplied by the *decay* hyperparameter every 10 epochs; *layer1_size* and *layern_size* define the width of the first and last hidden layers respectively; the widths of intermediate layers are a linear interpolation of these two numbers. Each hidden layer had dropout, batch normalisation, and a ReLU activaiton, apart from the last layer which had only a sigmoid activation. BCE loss with an Adam optimiser were used. The hyperparameter search space was too large for exhaustive search, so we used Gaussian process regression to optimise them for the final validation AMS score [7].

This initial search showed that the best models use batch sizes between 250 and 2500; learning rates between 0.015 and 0.04; have no more than 4 hidden layers; and the last hidden layer is no bigger than 400 units. We then ran two more hyperparameter tuning sessions (see Figures 6 and 8 in the Appendix) with narrower sampling distributions. The best performing network got a public score of 3.65 using the C.A.K.E. and phi differences features. It used the structure in Figure 9 with *block1_size=1*, *block2_size=1*, *l1_size=251*, *lm_size=206*, *ln_size=92*. Full hyperparameters information can be found in Tables 1 and 2 in the Appendix.

5 SoftAMS

$$\frac{1}{\text{softAMS}} = (s + b + b_{reg}) \ln\left(1 + \frac{s}{b + b_{reg}}\right) - s \approx \frac{2}{\text{AMS}^2} \tag{2}$$

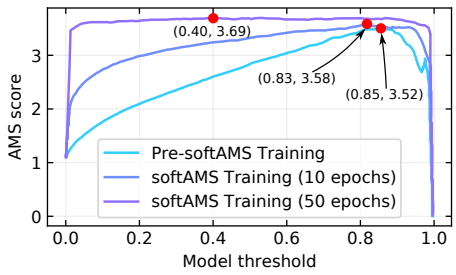

Figure 4: The effect of AMS training on the AMS curve. Cross entropy training tends to produce curves which have a steady upward slope followed by a sharp cliff, whereas AMS training flattens these and results in steep cliffs at each end.

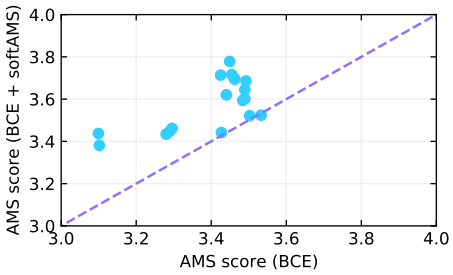

Figure 5: Public data AMS before and after softAMS training for a series of DNNs with widths of 250-750 and 3-8 hidden layers. All networks were trained with BCE loss for 75 epochs before softAMS for 150 epochs.

The AMS is more a function of sensitivity than accuracy and cannot be optimised directly because it involves many step changes. We developed a relaxation which fixes these; rather than using a threshold, model confidence is used to create a weighted sum of event weights. The result is equation 2, where $s = \mathbf{y_s} \cdot \mathbf{w_s}$ and $b = \mathbf{y_b} \cdot \mathbf{w_s}$ for a set of predictions $\mathbf{y}$ and normalised event weights $\mathbf{w}$. Figure 4 shows how softAMS changes the ramp like curve of BCE loss trained networks into a plateau with a higher maximum. SoftAMS approximates the average AMS across the curve and therefore it will flatten the curve, which is a desirable trait because it improves generalisation by not relying on the precise location of a peak. It simultaneously tries to maximise the sensitivity of the classifier.

The softAMS loss could not train a randomly initialised network well, converging on an AMS of about 1; however, when a pre-trained network was used, the results were far better. The previously trained DNNs with C.A.K.E. features overfit with softAMS, so a simpler architecture search without using C.A.K.E. and phi features was performed, testing architectures with 3-8 hidden layers each with widths of 250, 500 or 750. They had dropouts of 0.5 and batchnorm on each hidden layer. They were trained with Adam using BCE and softAMS for 75 and 150 epochs with learning rates of $1e{-}3$ and $1e{-}5$ respectively. BCE converged within 50 epochs so the improvements were not the result of longer training. Figure 5 shows the effect of softAMS on the model scores, with almost all showing improvement. The best performing network trained this way had 3 hidden layers, each 750 wide. Networks with 3 to 5 hidden layers, each 500-750 wide, also had consistently high AMS scores. The best model scored 3.753 and 3.764 on the public and private data respectively. The score on the training data is 4.22, indicating overfitting.

6 Conclusion

We introduced a novel loss function, softAMS, which performed well. This, combined with hyperparameter tuning, would score 7th in the *Higgs Boson Machine Learning Challenge*. The models that scored better than ours used ensembling. We believe that, had we had more time, fine-tuning of the softAMS model's hyperparameters or an ensemble of neural networks trained with softAMS would reduce overfitting and be even more competitive.

References

[1] "Learning to discover: the higgs boson machine learning challenge." https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf.

[2] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[3] S. Popov, S. Morozov, and A. Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," in *International Conference on Learning Representations*, 2020.

[4] B. Peters, V. Niculae, and A. F. Martins, "Sparse sequence-to-sequence models," in *Proc. ACL*, 2019.

[5] A. V. Dorogush, A. Gulin, G. Gusev, N. Kazeev, L. Ostroumova, and A. Vorobev, "Fighting biases with dynamic boosting," *ArXiv*, vol. abs/1706.09516, 2017.

[6] J. Tobin, S. Karayev, and P. Abbeel, "Troubleshooting deep neural networks - a field guide to fixing your model." http://josh-tobin.com/assets/pdf/troubleshooting-deep-neural-networks-01-19.pdf.

[7] P. Sayak, "Bayesian hyperparameter optimization - a primer." https://www.wandb.com/articles/bayesian-hyperparameter-optimization-a-primer, 2020.

Appendices

A Network Architectures

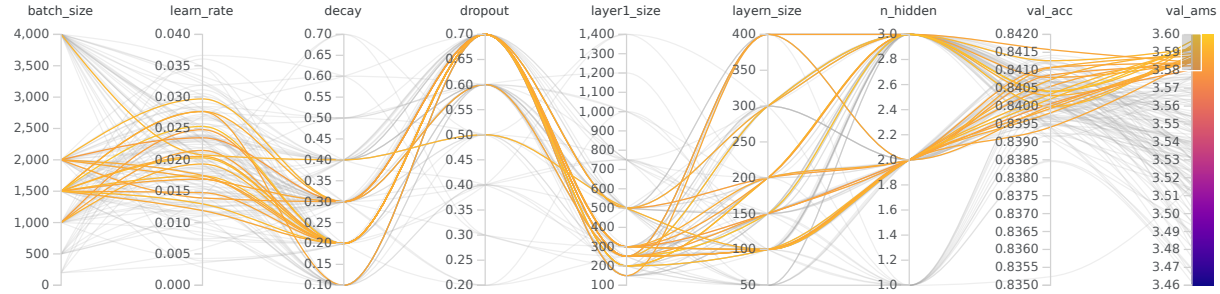

Figure 6: The refined search for hyperparameters. Each line represents one combination and leads to the respective training and validation AMS scores on the right. A total of 152 networks were tested (gray lines) and the best performing ones ($val\_ams \geq 3.59$) are highlighted.

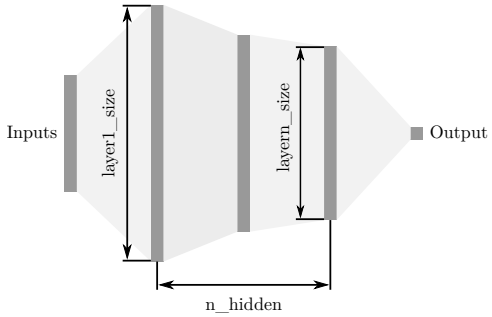

Figure 7: The network architecture used in the initial search (Figure 3) and refined search (Figure 6).

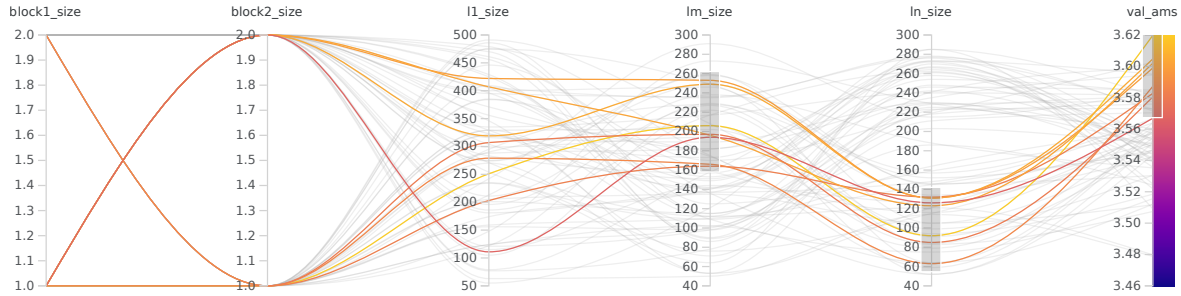

Figure 8: The final search for architecture. Each line represents one combination and leads to the respective training and validation AMS scores on the right. A total of 79 networks were tested (gray lines) and the best performing ones ($val\_ams \geq 3.59$) are highlighted.

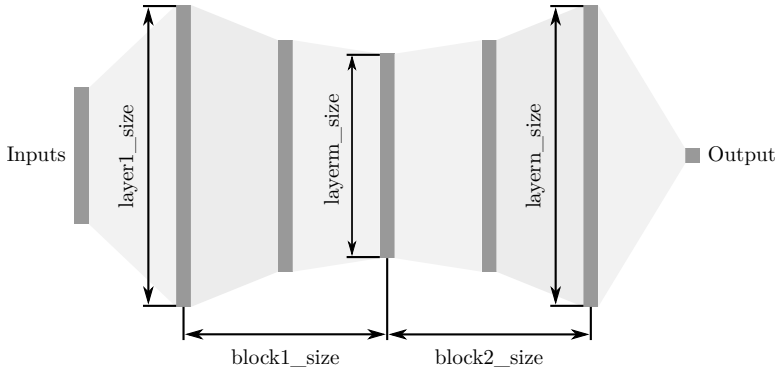

Figure 9: The network architecture used in the final architecture search (Figure 8).

B Hyperparameters

Table 1: Best hyperparameters achieving AMS of 3.65 with the architecture in Figure 9 and with optimisation results in Figure 8.

Hyperparameter	**Value**
batch_size	1500
block1_size	1
block2_size	1
l1_size	251
lm_size	206
ln_size	92
learn_rate	0.02
decay	0.2
dropout	0.7

Table 2: Best hyperparameters achieving AMS of 3.58 with the architecture in Figure 7 and with optimisation results in Figure 6.

Hyperparameter	**Value**
batch_size	1500
n_hidden	3
layer1_size	250
layern_size	150
learn_rate	0.02533
decay	0.2
dropout	0.7

Table 3: XGBoost Model 1 hyperparameters achieving maximum AMS of 3.604

Hyperparameter	**Value**
max_depth	11
min_child_weight	7
gamma	2.00
eta	0.05
subsample	0.7
colsample_bytree	0.8

Table 4: XGBoost Model 2 hyperparameters achieving AMS of 3.603

Hyperparameter	**Value**
max_depth	11
min_child_weight	7
gamma	2.25
eta	0.05
subsample	1.0
colsample_bytree	1.0

C NODE network results

These are the validation scores of the NODE architecture on the Higgs Boson dataset.

It is important to note that these results were attained without optimising different hyperparameters like the optimisation function and imputation, meaning that there is still room for improvement.

Table 5: Validation scores of running the NODE network with different layer sizes

Architecture (# of layers-1 ; total # of trees ; tree depth)	**AMS score**
2-1024-6	3.48
2-1024-8	3.50
2-2048-6	3.58
4-1024-6	3.54
8-1024-6	3.58