LIP-STUDENTS-20-06

Advanced data analysis methods for dark matter research associated with the quark top at the LHC

Miguel Caçador Peixoto[1,a]

[1] *Universidade do Minho*

Project supervisors: N. Castro, M. Romão

October 2020

Abstract. A Machine Learning model was trained on simulated data of proton-proton collisions at the Large Hadron Collider at CERN derived using Monte Carlo methods to be used as a binary classifier. The data contained background events from the standard model and events from a beyond the standard model signal related to dark matter research. It was concluded that the use of machine learning models in the search of new physics is quite relevant since, on the simulated data, the trained model managed to get a area under the ROC curve of 0.9947 in the test dataset using Monte Carlo Dropout techniques that showed to be useful in the model optimization.

Keywords: LHC, CERN, ATLAS, Machine Learning, Dark Matter

1 Introduction

Along the years the standard model (SM) has been scrutinized by the scientific community, checking if it can reliably describe the world around us and for now, it has never failed, withstanding the test of time. In this project the SM is tested once more using simulated data from the ATLAS detector, one of the four major experiments on the LHC at CERN. In the experiment two protons collide producing various outcomes, one of them could be a theoretical beyond the standard model (BSM) signal event represented in fig. 1.

The simulated data is derived from Monte Carlo methods for later being used to train a deep neural network (DNN) that will serve as a binary classifier to distinguish a background event from a signal event.

The idea is to study in simulation environment how relevant are the applications of a machine learning model in this context and check if they can be used on the search for new physics, aiming for later use in real world applications.

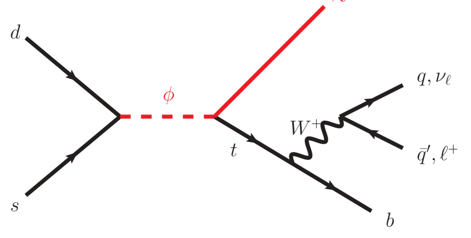


Figure 1. Non-resonant t-channel and s-channel production of a top-quark in association with a vector boson V which decays into two dark matter particles; resonant production of a coloured scalar Φ that decays into a dark matter particle and a top-quark; and single production of a vector-like T quark decaying into Zt ($\rightarrow$ v v b W). [1]

[a] e-mail: miguelpeixoto457@gmail.com

The signal in question is presented in Figure 1, it's a BSM signal since the Φ and χ, given in red, aren't predicted by the SM.

This signal fits in to the dark matter (DM) research program at the LHC thanks to χ being the DM candidate. Considering that if it exists, it would give rise to a top with great momentum and a large quantity of transverse energy missing, therefore making this particular signal a good candidate for the experiment due to the clear experimental signature of its existence.

2 Data Exploration

The data is divided on multiple files in csv form, each one corresponds to a sample containing 250-450k rows (events) and 465 different columns (features). For example, the features contained information about the missing transverse energy, the number of jets, the direction of the jets, etc.

A sample is a specific outcome of the two protons colliding, therefore separating the data into background samples (events/processes that are already well known from the standard model) and signal samples (the signals events from figure 1).

2.1 Data Pre-Processing

Data Cuts

The initial 465 different features included columns that where used for the data generation as well some non relevant columns that was removed.

Accordingly to the event in fig 1, the cuts are the following:

1. 0 leptons
2. At least 1 fatjet
3. At least 1 b-tag

The drop rates for every one of the cuts are as following:

1. Background Data:
   - 0 leptons drop ratio: 0.3438 %
   - At least 1 b-tag: 81.2528 %
   - At least 1 fatjet: 2.2760 %
   - Total drop rate: 81.3892 %
2. Signal Data:
   - 0 leptons drop ratio: 0.5268 %
   - At least 1 b-tag: 34.2679 %
   - At least 1 fatjet: 0.0074 %
   - Total drop rate: 34.4409 %

Extra Features

Beyond applying cuts to the data it was also added the 'Label' column which contained 0 if the event was background or an 1 if it was signal, the 'Name' column containing the corresponding name of the sample for each event and finally it was calculated the 'Weights' column containing the weight of every event. This last feature will be approached in greater detail below.

Sample and Class Weights

Each sample has a cross-section associated with it, that is the probability of a specific process has of occurring when the two protons collide. In the original data files the weights of the events inside every sample corresponded to the weight of the sample itself. However, for training purposes, what mattered is the probability of each event had of occurring, not the sample as a whole.

For this purpose it was calculated a new probability on a new feature called 'Weights'. It was calculated for each sample by dividing the cross-section of the sample by the number of events on each sample.

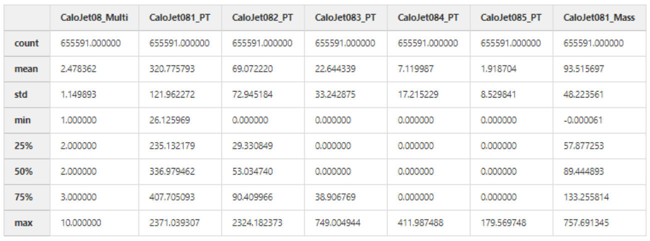
Apart from that there was much more background events compared to signal events (69% more background than signal), which meant that without correction the model would 'learn' more class A than class B for example, since the events in class A would pass through the model alot more times that the events in class B. This would result in the model having a low predictive accuracy for the infrequent class. So due to this problem a dictionary was created showing the relationship between both classes that will later be used when training the model to avoid this problem.

Training, Validation and Test Set

For the last step in data pre-processing it was necessary to split the data in 3 different datasets: a test, validation and a training set. Each one having 1/3 of the data to keep the statistical distribution of the features.

The training set will be used to train the model, the validation dataset will provide an unbiased evaluation of a model fit on the training dataset using the loss and some other metrics and finally the test set is used to provide an unbiased evaluation of how well the model generalizes to new data.

After the pre-processing the data totaled 72 different features to serve as input of our model.



	CaloJet08_Multi	CaloJet081_PT	CaloJet082_PT	CaloJet083_PT	CaloJet084_PT	CaloJet085_PT	CaloJet081_Mass
count	655591.000000	655591.000000	655591.000000	655591.000000	655591.000000	655591.000000	655591.000000
mean	2.478362	320.775793	69.072220	22.644339	7.119987	1.918704	93.515697
std	1.149893	121.962272	72.945184	33.242875	17.215229	8.529841	48.223561
min	1.000000	26.125969	0.000000	0.000000	0.000000	0.000000	-0.000061
25%	2.000000	235.132179	29.330849	0.000000	0.000000	0.000000	57.877253
50%	2.000000	336.979462	53.034740	0.000000	0.000000	0.000000	89.444893
75%	3.000000	407.705093	90.409966	38.906769	0.000000	0.000000	133.255814
max	10.000000	2371.039307	2324.182373	749.004944	411.987488	179.569748	757.691345

Figure 2. Tabular representation of some features after the pre-processing

3 Deep Neural Networks (DNN)

3.1 What are Artificial Neural Networks?

Artificial Neural Networks (ANNs) [2] are a class of algorithms that are inspired in the way a brain operates, recognizing relationships in the data without being explicitly programmed to. An ANN, shown in Figure 3, only contains 3 types of layers: input layer, a hidden layer and an output layer.

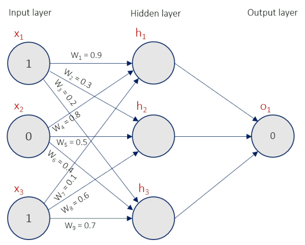


Figure 3. A Neural Network [3]

But how does a ANN learn?

As stated above an ANN tries to mimic the way a brain works, It uses **neurons** that are interlinked to each other through **weights** and **biases**. Those parameters are the heart of the neural network, and by changing them to specific numerical values it's possible to process any input and get a desired output.

Each neuron in a network transforms data using a series of computations: a neuron multiplies an initial value by some weight, sums results with other values coming into the same neuron, adjusts the resulting number by the neuron's bias, and then normalizes the output with an activation function (for example the S shaped sigmoid).

This process is repeated until it reaches the output layer that provides the predictions related to the classification task at hand.

A loss function is also specified, it's used to evaluate a candidate solution (i.e. a set of weights) and it does that by comparing the initial outputs with a provided correct answer.

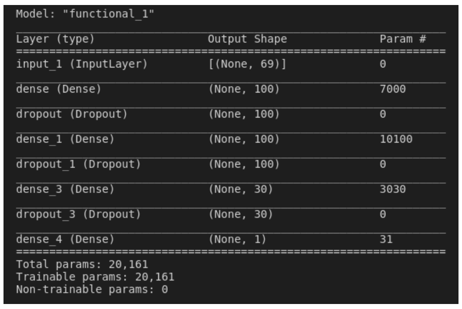
What the Neural Network effectively does when is training ("learning") is to manipulate the weights using gradient descent in order to find a local minimum of the loss function.

3.2 What are DNN's?

Deep Neural Networks or DNN are essentially Artificial Neuronal Networks with more than two non-output layers. DNN filter information through multiple hidden layers enabling them to comprehend deeper and more complex relationships in the information.

3.3 The Model

The model consists in 5 layers: 1 input layer with standardization, 3 dense layers with dropout and 1 output layer presented in Figure 4.



```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 69)]              0
_________________________________________________________________
dense (Dense)                (None, 100)               7000
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_3 (Dense)              (None, 30)                3030
_________________________________________________________________
dropout_3 (Dropout)          (None, 30)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 31
=================================================================
Total params: 20,161
Trainable params: 20,161
Non-trainable params: 0
```

Figure 4. Model Summary

The purpose of the standardization layer is, as the name suggests, to standardize the data. It's a scaling technique where the values of the different features are centered around the mean with a unit standard deviation. This means that the mean of the attributes becomes 0 and the standard deviation 1. This is done so that the model doesn't see many discrepancies on the data therefore giving the same importance to all features avoiding bias towards one in particular for instance.

3.3.1 Monte Carlo Dropout

The Dropout [4] method consists in temporally disabling neurons to improve the generalization of the network and prevent overfitting. Consequently, at each epoch each neuron has a probability of being disabled which means that the weights associate with it will equal 0.

On the standard Dropout method the disabling of the neurons would only occur during training, but on MC Dropout [5] method this would also happen when evaluating the model using the test or validation set: each event goes through the model multiple times making a distribution. This distribution is then averaged thus getting a more accurate prediction for each event than if used the traditional dropout technique.

On the final model, the probability used for each neuron on the 3 hidden layers being disabled was 10%.

3.3.2 Training Model Settings

For fitting the model it was used a maximum of 500 epochs with **early stopping** callback enabled with a patience of 30 monitoring the validation loss. In other words if the model trains for 30 consecutive epochs without seeing any improvement on the output of the loss function it will stop the training seeing that it most likely found an local minimum of the function.

The tensorboard and the Model Checkpoint callback were also used when fitting the model. The first enabled the logging of training metrics (e.g accuracy and AUC) and see their improvement throughout training. The second guarantees that when the training ended, only the best performing model iteration on the validation data would be kept as our final model.

4 Model Evaluation

4.1 No Dropout vs Dropout vs MC Dropout

The ROC score was calculated for the different scenarios using the same data. The results are displayed below:

	ROC Score	Improvement
Without Dropout	0.8990	-
W/Dropout	0.9934	10,5%
W/MC Dropout	0.9947	10,6%

Table 1.

Below it's possible to see how both models evolve over the epochs.

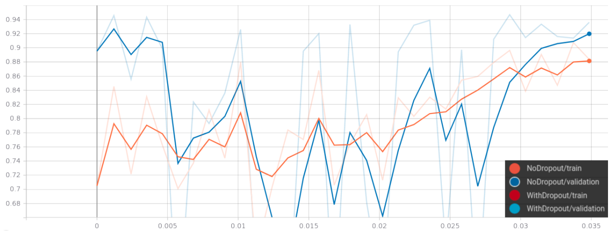


Figure 5. Evolution of the area under the ROC curve for the model without dropout.

The dropout performs better, 10,5% than the model with no dropout implementation. This was expected since theoretically a model with dropout is more generalized.

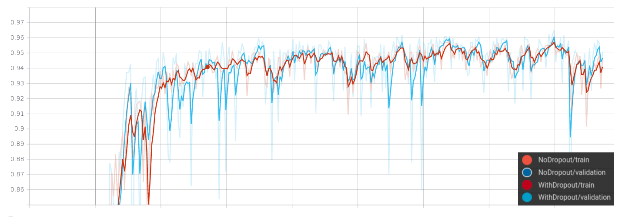


Figure 6. Evolution of the area under the ROC curve for the model with dropout implementation

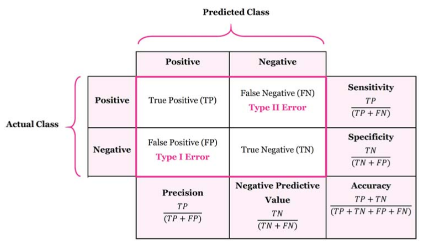
It's also evident that it needs more time to find the best weights (the early stopping callback has stopped the model without dropout implementation a lot sooner that the model with dropout implementation), this is likely due to non-ideal hyper-parameters and this problem may be solved with hyper-parameters optimization.

Furthermore it's possible to squeeze a little bit more performance using MC Dropout (0,1% improvement compared to Dropout) considering the model will output the mean of a probabilistic distribution.

4.2 Metrics

4.2.1 Confusion Matrix

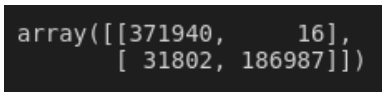
Confusion matrix, also known as an error matrix, is one of the most popular metrics that allows the visualization of the performance of an algorithm in table form.



		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

Figure 7. A Confusion Matrix Explained, as well some other metrics

On the final model, the following confusion matrix was obtained:



```
array([[371940,     16],
       [ 31802, 186987]])
```

Figure 8. The confusion matrix on the test data of the best Model

In the figure 8 the background and signal data are the positive and negative class respectively, as reported by fig 7.

This means that 371940 background events where correctly classified as background (TP) and 16 of them where wrongly classified as signal (FN). On the other hand 31802 signal events where wrongly classified as background (FP) and 186983 where correctly classified as signal (TN).

There is still room for improvement considering the more diagonal the confusion matrix is, the better the accuracy would be, thus having a better model.

The final model with MC Dropout Implementation got an accuracy of 95,0% using our test data and a final ROC score of 0,995

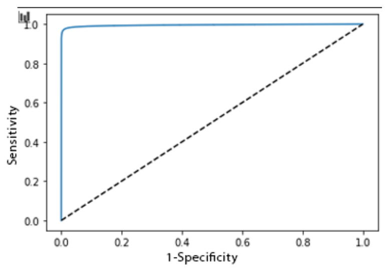


Figure 9. The ROC Curve of the final model

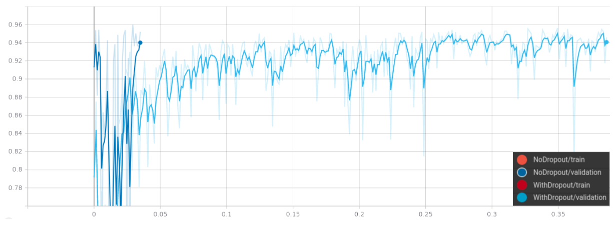


Figure 10. Model accuracy over epochs.

4.3 Standard deviation's

Lastly, one of the advantages of using simulated data is the possibility to check how well the model performs in each sample separately.

The model was evaluated for each sample on the fraction of the data that wasn't used for training nor validation. With that, the accuracy and the mean of the standard deviation (the model prediction vs the label) was calculated and presented in Table 2.

	Accuracy	Mean of the SD
BKG_ttbar	0.999962	5.14905×10^{-5}
BKG_Wjets	1.0	3.08148×10^{-5}
BKG_WW	0.999958	9.20867×10^{-5}
BKG_Zjets	1.0	1.17766×10^{-5}
BKG_ZW	0.999798	2.11444×10^{-4}
BKG_ZZ	0.999960	5.08952×10^{-5}
SIGNAL	0.866125	8.57968×10^{-1}
SIGNAL_15	0.865916	8.57967×10^{-1}

Table 2. Accuracy and the mean of the SD of the model for each sample.

It is also possible to plot the histograms of the stds of each sample (Fig. 11).

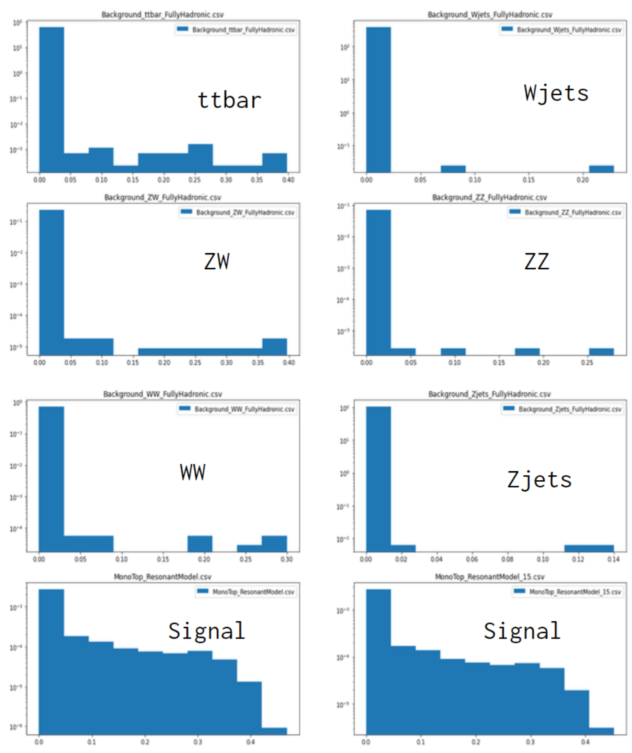


Figure 11. Logarithmic scale of the standard deviation distributions by sample

The Figure 11 shows the standard deviations of our model predictions in each sample. What this means is that the model tried to predict the events that weren't used for training contained in the samples. Using the predictions of the model, the distributions of the standard deviations for each sample where plotted, showing the confidence that the model has in classifying events from each sample.

Accordingly to Figure 11, it's possible to infer that there are some background samples more alike to the signal than others. For example the ttbar is more similar to the signal than the Zjets background due to higher stds. It's also possible to see big standard deviation's on the signal samples, this means that the model is less confident when classifying signal events since they appear to be similar to the background, making the job of the model harder.

5 Conclusion

The data was explored and pre-processed, removing the non relevant features and applying data cuts derived from the Feynman diagram in fig. 1. A model was then created and trained from the simulated data of the ATLAS detector.

The model had an MC Dropout implementation that improved the performance of the model by 10,6% on the ROC score compared to the initially traditional DNN.

In a simulated environment the model was capable of reasonably distinguish between a background event and a signal event having a good final ROC Curve (fig. 9). This is an indication that Machine Learning surely has a real world applications in the field of particle physics and it should serve as a tool on the finding of new physics.

Acknowledgements

I would like to thanks my supervisors Nuno Castro and Miguel Romão for answering all of my questions and doubts, for referencing books and resources as well of kindly providing some enlightenment sessions that shown to be really useful throughout the internship.

I've learned the basics of what it means to be a data scientist, what a neural network really is and how it works, how to handle data and pre-process it and also some python modules such tensorflow, keras, pandas and numpy. It was a wonderful experience.

References

[1] Collaboration, A. T. L. A. S., Barton, A. E., Bertram, I. A., Bouhova-Thacker, E. V., Fox, H., Henderson, R. C. W., ... Muenstermann, D. (2019). Search for large missing transverse momentum in association with one top-quark in proton-proton collisions at s= 13 TeV with the ATLAS detector. Journal of High Energy Physics, 2019(5). CMS, ATLAS

[2] Mehlig, B. (2019). Artificial neural networks. arXiv preprint arXiv:1901.05639.

[3] Imdadul Haque Milon, "Neural Network: A Complete Beginners Guide", https://medium.com/gadictos/neural-network-a-complete-beginners-guide-from-scratch-cf1fc9d5cd12, 2019

[4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[5] Gal, Y., Ghahramani, Z. (2016, June). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning (pp. 1050-1059).