4-top Event Discovery using Bi-directional LSTM Models

Cyril de Kock - s1047180
Radboud University, Nijmegen, The Netherlands
cyril.dekock@student.ru.nl

ABSTRACT

The search for 4-top events could reveal new physics beyond the Standard Model. This paper describes an approach to discriminate 4-top events from a host of different background processes using both particle- and event-level features, as well as an approach to distinguish all the different processes. A neural network configuration is proposed to extract the maximum amount of information from the features of each event and this network is subsequently applied to both the binary and multiclass problem.

1 INTRODUCTION

The production of events with four of the heaviest quark are called 4-top events. These events could point to undiscovered physics beyond the Standard Model. For this reason, detecting and measuring these events is an important topic in Particle Physics. This paper describes an attempt to discriminate 4-top events from a host of background processes. Each event decays into a set of particles which are measured and form the features that describe each event. There are 52 different variables in the data that can be used as machine learning features these variables will be referred to in this report by their group denomination as specified in the assignment. This means that all energy transverse components of the 4-vectors will be referred to as *E*. First, section 2 will go over the properties of the dataset and explain any transformations applied. Then, section 3 will detail the models used as well as any fine-tuning techniques applied. Lastly, section 4 will present and examine the results of the different models followed by a short conclusion on the results of the project.

2 DATA

The data consists of 200, 000 events which specify the process which generated the event, the particles detected and a few other features. 100, 000 of these events are background and the other half consists of 4-top events. Normally 4-top events appears orders of magnitude less frequently than most other events but due to the generated nature of the dataset the distribution of events deviates from the one that would naturally occur. The 100, 000 background events are divided into the ttbar, ttbarHiggs, ttbarW and ttbarZ processes with each 25, 000 datapoints. This distribution implies the dataset is balanced when it comes to the binary classification into foreground and background but imbalanced in the case of the multiclass problem, which will have to be accounted for in model training.

Each event in the data has the same set of attributes. The process that generated the data, the *MET* and *METphi* which are properties of the missing transverse energy vector of the event alongside with ten particles and their properties as well as some simulation related fields. The particles are described by five different fields, the kind of particle as well as their charge(*obj*), their energy(*E*), the transverse component of the momentum(*pt*), theta(*eta*) and phi angle(*phi*). Note that not every event has the same amount of information, most events do not produce ten particles and thus will contain some empty values as part of the data.

2.1 Preprocessing

For the binary classification into background and foreground the labels are encoded as 1 representing the 4-top events and 0 representing the four classes of background events. For the multiclass classification where all the different processes need to be identified separately the labels are encoded using a 1-hot encoding scheme where the label for each event is a vector of five elements with a one and four zeroes. Simply encoding each category as a integer would also work but this has the downside of potentially inducing an ordering into the data where in reality there is none.

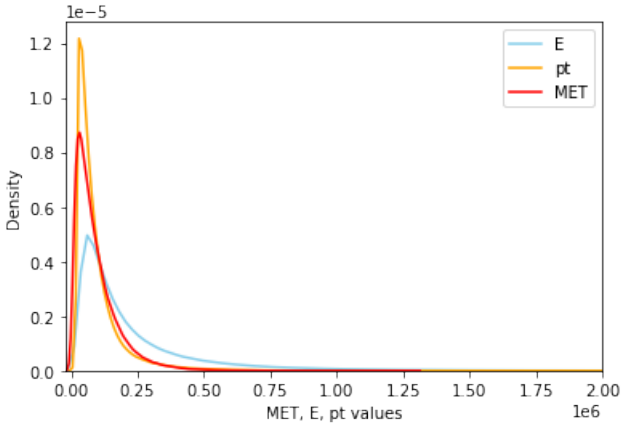

Figure 1: Distributions of E, pt and MET variables

As a first step the distributions of all variables were plotted to get insight into the shape of the data. One can observe from 1 that the *E*, *pt* and *MET* variables all heavily skew left but have an extremely long tail towards the right. Since the variables are only expressed as positive values they can be log-transformed to a Gaussian-like shape as can be seen in 2. Since the log is a monotonic transformation this should lead to a more stable optimisation process while not affecting the optima of the distribution.

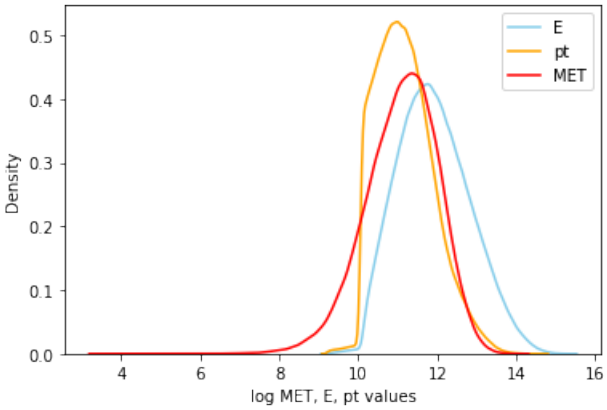

Figure 2: Log-distributions of E, pt and MET variables

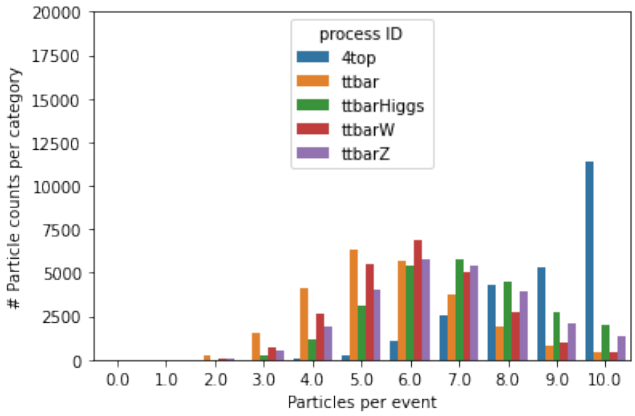

Figure 3: Particle count distribution split by process

All the columns with numerical values (MET, METphi, E, pt, eta and phi) are standardised to ensure that differences in scale between features do not adversely affect the loss function. This is done by removing the mean and scaling to unit variance by calculating the standard score of each sample using:

$$z = \frac{x - \mu}{\sigma}$$

With μ and σ as the mean and standard deviation of each respective feature and x the value of the sample being scaled.

Since the size of the available information changes for each event, any missing particles are filled up by zero-padding. This is necessary as most machine learning models need the input to be of a consistent size. Additionally, the fact that information is missing for particular events can be valuable information in itself. Figure 3 shows that the amount of particles varies heavily depending on the process that generated the event. Neural Networks generally are expected to learn these implicit relations from the data but the count of particles per event could be a valuable feature for other machine learning models. Any missing particles are thus replaced by a zero value. This transformation is applied to the data as a whole since any columns without missing values will remain unaffected by the process. Note that the standardisation is always done after the log-transform and that the zero-padding is done after the standardisation to avoid introducing artificial artifacts in the data distribution.

The data is split into a training and a validation set. The validation set contains 10%($n = 20000$) of the events in the data. The split is done in such a way that the distribution of the labels in the training data matches that of the validation data. While training the Neural Networks a fraction of 0.05($n = 9000$) of the training data is used to monitor the validation loss while training.

3 APPROACH

Two different models were experimented with for the binary classification. A simple Neural Network using categorical embeddings [1] to represent the object variable and a bi-directional LSTM model using the same embeddings but a different structure for the remaining data as described in 3.1. The simple model is augmented with a particle count feature that counts the number of particles in the event. Normally Neural Networks implicitly extract such features in their hidden layers but since figure 3 revealed the feature to have a lot of discriminative power it is worth trying to explicitly pass this information.

3.1 Data Formatting

Before feeding the data to the LSTM model the 4-vectors of the particles are formatted by putting the *E, pt, eta & phi* into a list. All ten of the 4 vectors are then stacked upon each other resulting in a $(10, 4)$ matrix for each event. The simple Neural Network is simply fed the values in the 4-vectors as one input each. The *obj* variables are kept separate as these will be embedded separately. To generate the embeddings we do have to encode the *obj* as categorical variables whereby each object is encoded as a number (e.g. $j \rightarrow 6$). These variables are then fed as a separate input to the model and fed into an embedding layer which learns the feature representation during model training. The particle charge information (+ or −) that accompanies the e and m particles is encoded with the respective particle. Ergo, $e-$ and $e+$ are encoded as two separate numbers and have individual embeddings generated during training. An attempt was made to encode the charge information separate from the particle and to use it as a stand-alone feature but this had no impact on performance and was thus dropped. The *MET* and *METphi* features are passed as a separate input to the LSTM model resulting in a total of three input layers.

3.2 Models

The categorical embeddings and 4-vectors described in the previous section are concatenated into a $(10, 104)$ matrix (ten sequences with a length of one hundred and four) and then fed into two subsequent Bi-directional LSTM layers, the *MET* and *METphi* input is forwarded to a separate dense layer before being concatenated to output of the LSTM layers. This concatenation of all features is then passed on to another dense layer before reaching the output layer. All the layers except for those that produce the model output use the Relu activation function [4]. Since Relu is a asymmetric activation function and thus a He-Normal weight initialisation scheme [2] is used for the layers. The output layer only has a single neuron with a Sigmoid activation function. Figure 4 provides a visualisation of the architecture used in the binary classification problem.

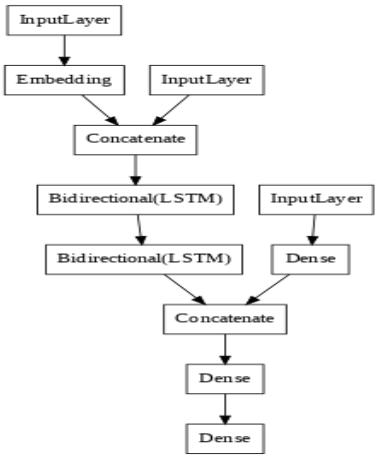

Figure 4: Model architecture for binary classification task

The multiclass classification is tackled using a reworked version of the Neural Network used for the binary task. The output layer is replaced by a Softmax layer with five neurons, one for each of the background classes. The rest of the network remains unchanged. The multiclass classification is aided by using a dictionary of class weights which are passed to the Neural Network at training time and adjust the loss function in favor of the underrepresented classes, the background classes in this instance. Both Neural Networks were trained with Adam [3] as the optimizer of choice due to it's adaptive learning rate and the quick convergence it enables. An early stopping scheme was implemented to prevent further model training if the loss did not improve for five subsequent iterations. Initially a learning-rate scheduler was used to try and find appropriate points to lower the learning rate as validation loss hit a plateau but this was found to be of little value given Adam's native adaptive learning rate property.

3.3 Multiclass Probabilities

One way of potentially augmenting the performance of the binary classifier is to use the information from the multiclass data. There are two ways this could be done, the first would be to calculate sample weights that quantify how much each individual sample should contribute to the loss function. These can be calculated based on the prior probabilities of the classes as well as set manually. Another way would be to use the output from a multi-class classifier to augment the binary predictions. This can be accomplished by simply combining the predictions for the different background classes and transforming the multiclass output into a binary, but also by using the multiclass predictions as input for the binary classifier. This paper will focus on the latter option by co-optimizing both the binary and multiclass predictions. A double-output model is created by taking the design of the multiclass Neural Network feeding the output from the multiclass softmax layer to another dense layer that is in turn passed to the output layer doing the binary classification.

3.4 Prior

In the previous section the class probabilities of the five different processes' were used to influence the training procedure. However, they can also be used as a prior. Combined with the output of the softmax function which can be interpreted as a probability distribution over the possible labels for each datapoint. We can then multiply this output by a prior over the possible labels to output posterior probabilities for each class. The equation below describes Bayes rule which can be used to calculate the posterior distribution:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Where Y stands for the possible labels and X the data. In practice it is impossible to calculate the marginal probability $P(X)$ in this particular case since we would have to integrate over all possible sets of model parameters. However, since $P(X)$ is nothing but a normalising constant that ensures the posterior integrates to one, we can drop it from the equation. It can subsequently be rewritten as:

$$P(Y|X) \propto P(X|Y)P(Y)$$

This allows easy calculation of the posterior but has the disadvantage of it no longer being interpretable as a probability.

4 RESULTS & ANALYSIS

In this section the algorithms discussed in section 3 will evaluated and compared. The model performance in the case of the binary classification is measured using simple accuracy. The multiclass classification was evaluated by monitoring precision, recall and the F1-score per class. Summary statistics such as micro-averages of the aforementioned metrics could be used but do not give a complete picture due to the imbalance in performance between the different classes.

Model	Accuracy
Simple NN	0.863
Simple NN + Particle count	0.867
LSTM	0.870
LSTM + Multiclass Probabilities	0.871

Table 1: Binary classification results

Table 1 displays the results of the different algorithms designed to tackle the binary classification problem. As the table suggests, there is not a large difference in performance between the individual methods. The more complicated LSTM model performs slightly better but it is not clear whether this is due to the structure of the model or simply due to the increase in parameters.

The multiclass problem is tackled by taking the best performing model from the binary classification and adapting it to the multiclass case. Thus, in this particular case the LSTM model is adapted by the changes suggested in section 3.2. Reversing the multi-output model and adding the output of the binary classification as information to the multiclass procedure did not produce the similar increase in performance. The model predicts a distribution over the five different labels due to its Softmax activation function. To get the actual prediction the *argmax* of the vector is taken. The distribution in the vector can also be multiplied by a prior over the classes to get a posterior distribution over the classes as described in 3.4. The performance of these two approaches are compared in table 2. With the metric 'prec' standing for 'precision' and ttbarH as shorthand for ttbarHiggs for formatting reasons.

Model	Metric	4-top	ttbar	ttbarH	ttbarW	ttbarZ
LSTM	Prec	0.90	0.48	0.28	0.35	0.29
	Recall	0.81	0.58	0.35	0.44	0.19
	F1	0.85	0.53	0.31	0.39	0.23
LSTM+Prior	Prec	0.81	0.48	0.35	0.36	0.34
	Recall	0.94	0.58	0.21	0.43	0.12
	F1	0.87	0.53	0.26	0.39	0.18

Table 2: Multiclass classification results

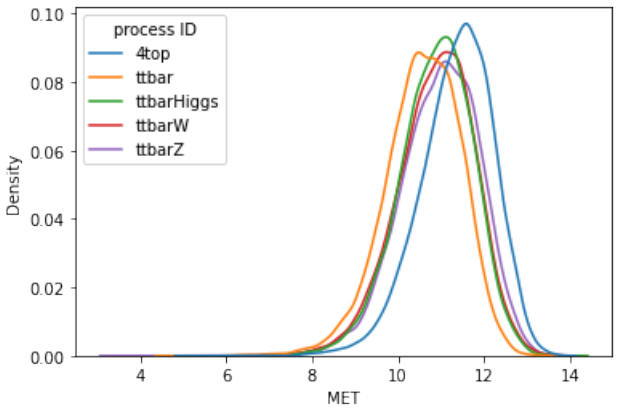

Figure 5: Log distribution for MET variable split by process

As shown by table 2 it is very hard to accurately distinguish the five classes. Specifically the background classes all suffer from both low precision on recall. Figure 5 illustrates the cause of this, the distribution of the *MET* variable is quite between the 4-top foreground and most of the background processes. However, the differences between the variable distributions for the background processes are far less significant. Only the ttbar process slightly deviates. The same observation can be made for the *E*, *pt* and *eta* as illustrated by figure 7 attached in appendix A. Note that these plots were produced by sampling 25000 4-top datapoints to ensure comparible densities. This is further shown by figure 6 which shows how predicted labels align with the gold set. It can be seen the ttbar processes are often confused. The application of the prior provides small improvements in precision for all the background processes but does this at the expense of of recall, due to the fact the prior devalues the background classes due to their lower frequency. The inverse of the prior could be used by taking the x^{-1} for each x in the prior and would produce the inverse result, which would be increased recall for the background classes but a lower precision and the reverse for the 4-top events. Which result is preferred depends on which aspect of the classification process has priority.

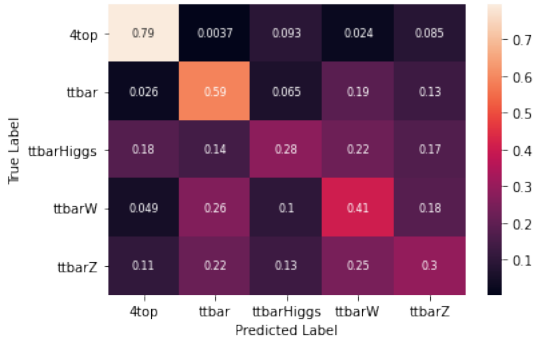

Figure 6: Confusion Matrix for the multiclass classification with use of prior

5 CONCLUSION

In conclusion, it can be stated that the binary classification which aims to distinguish 4-top events from the background performed well overall by being able to correctly classify 87% of cases. However, when trying to distinguish the different kinds of background processes from the 4-top events and each other the algorithm runs into trouble. It is hypothesised that this is due to the inherently hard nature of the problem caused by very similar feature distributions. To solve this, either more sophisticated separation techniques or a dataset that provides a clearer distinction between the classes would be required. The exact models and code for reproduction can be found on github.

REFERENCES

[1] Cheng Guo and Felix Berkhahn. 2016. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737* (2016).

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[3] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[4] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).

A APPENDIX

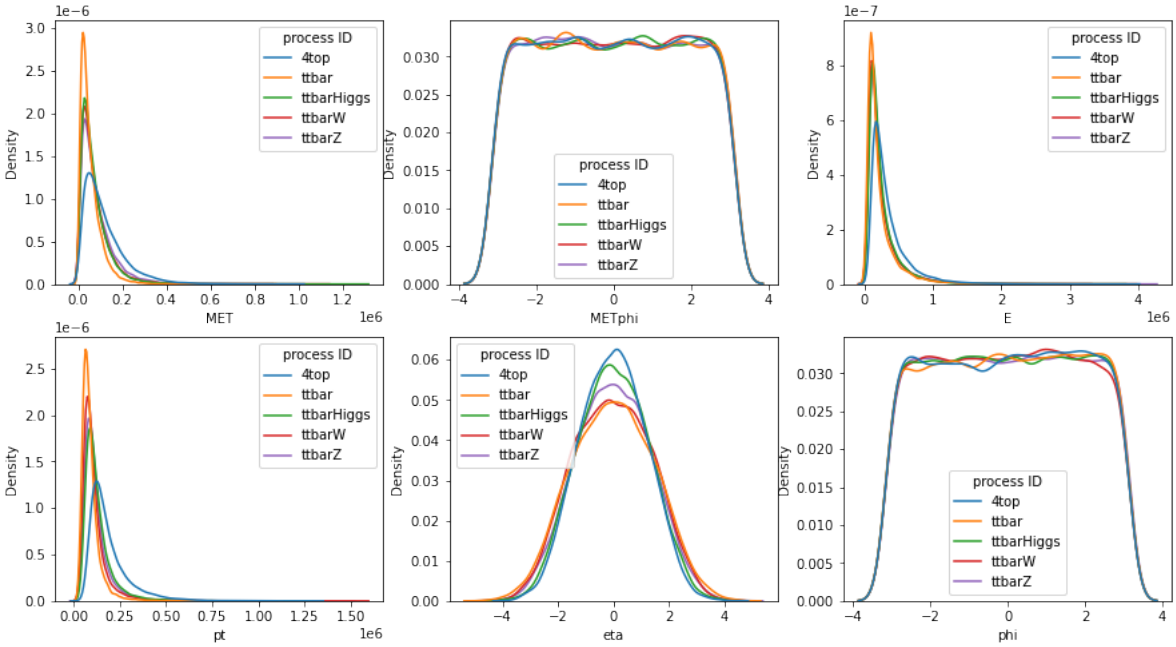

Figure 7: Distributions of all non-object variables per process ID in the multiclass instance.