Tandem Project Report: Classification in particle physics using machine learning

Research · June 2020

DOI: 10.13140/RG.2.2.24097.02408

CITATIONS
0

READS
476

5 authors, including:

Gianluca Bianco
University of Bologna
5 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Johannes Albrecht
Technische Universität Dortmund
546 PUBLICATIONS **25,752** CITATIONS
SEE PROFILE

Co-funded by the Erasmus+ Programme of the European Union

TP-2 Report

Classification in particle physics using machine learning

Florian Mausolf[a], Gianluca Bianco[b], Olaf Nackenhorst[a], Johannes Albrecht[a], Matteo Franchini[b]

[a] *Technische Universität Dortmund, Germany*
[b] *Università di Bologna, Italia*

Abstract

The *iTHEPHY Tandem Project* is designed for students from Italy, Germany and France, where small groups of students are built and supervised by postdoctoral researchers and professors. In the context of particle physics, different experimental and theoretical topics are treated. Our work was to get an introduction to machine learning techniques and to understand a measurement by the ATLAS collaboration about the coupling of Higgs boson to tau leptons. With regard to that, we performed a machine learning classification on a subset from the ATLAS analysis, simulated using Monte Carlo methods at a centre-of-mass energy of $8\,\mathrm{TeV}$. We tried two complementary machine learning approaches: Deep Neural Networks and Boosted Decision Trees. The combination of both outputs was found to give the most significant separation of signal and background events.

Contents

Introduction

The goal of our tandem project is to get an introduction to machine learning in the context of particle physics. More precisely, our project work consists of three main parts: After getting started with the basics of machine learning and investigating a paper about Higgs boson decays from the ATLAS collaboration individually, a classification problem is worked out using a subset of the same analysis. The dataset contains simulated Higgs boson decays and the related background processes.

The Higgs boson is named after Peter Higgs, who predicted the existence of the Higgs Field and its quantum excitation, the Higgs boson itself, along with other physicists already in the 1960s. Throughout the Higgs mechanism, it is explained how elementary particles acquire mass. The existence of a Higgs boson was finally proved by the ATLAS experiment and the CMS experiment at the CERN Large Hadron Collider (LHC) in 2012 [1, 2]. Consequently, Peter Higgs and his collaborator Francois Englert were awarded the Nobel Prize in Physics in 2013 for their theoretical predictions.

After its discovery, the properties of the Higgs boson like its mass and spin-parity quantum numbers were studied. In the analysis of interest for our project, *Evidence for the Higgs-boson Yukawa coupling to tau leptons with the ATLAS detector* [3], is found evidence that a direct coupling of Higgs bosons to fermions exists. This is an important fact with respect to the mass generation of fermions in the Standard Model. Nevertheless, the decay of Higgs bosons into tau leptons is a very rare process at the LHC and therefore, a low signal contribution is buried under background events. This suggests the use of machine learning techniques to separate signal and background events in a multidimensional feature space.

The classification problem was set up as the *Kaggle Higgs Boson Machine Learning Challenge [4]* in order to promote collaboration between high energy physicists and data scientists. In 2014, around 2000 competitors tried to achieve the most significant classification and a price of $\$13\,000$ was awarded. The data set used for the challenge and for our project consists of Monte Carlo (MC) simulations from the ATLAS experiment and it is taken from the CERN Open Data Portal [5]. It is a subset of the ATLAS analysis mentioned above.

1. ATLAS Analysis: Evidence for $H \to \tau\tau$ decays

In this paragraph, the analysis from the ATLAS collaboration is explained in more details.

For the analysis, the full proton-proton dataset from the 2011 and 2012 data-taking periods is used. The data correspond to integrated luminosities of $4.5\,\mathrm{fb}^{-1}$ and $20.3\,\mathrm{fb}^{-1}$ at a centre-of-mass energy of $7\,\mathrm{TeV}$ and $8\,\mathrm{TeV}$, respectively.

Data are taken using the ATLAS detector [6], which is a multi-purpose detector composed by an inner tracking system (silicon pixel detector, silicon strip detector and a transition radiation tracker) surrounded by a superconducting solenoid, electromagnetic and hadronic calorimeters and a muon spectrometer. It nearly covers the entire solid angle around the point of collision. A three-level trigger system is used to select events.

At the LHC, Higgs bosons can be produced mainly in four different processes, namely gluon fusion (ggF), vector-boson fusion (VBF), associated production with a W or Z vector boson (VH) and the simultaneous production with a top-quark pair. The first three are dominant and considered in the analysis. Feynman diagrams of the leading order production modes are shown in figure 1.

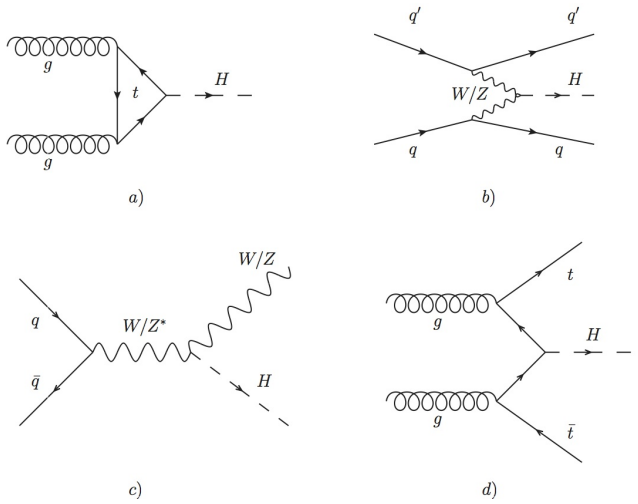


Figure 1: Main Higgs boson production processes at the LHC:
a) Gluon fusion b) Vector-boson fusion c) Associated production of a Higgs boson and a vector boson d) Simultaneous production with a top-quark pair [7].

The production modes lead to different final-state characteristics and thus, events are categorised in the following way:

- VBF category: this category is enriched in Higgs bosons produced via vector-boson fusion. Kinematically, these events are characterised by the presence of two jets with a large separation in pseudorapidity.

- Boosted category: this category aims to contain Higgs bosons produced via gluon fusion with additional jets from gluon radiation. High transverse momentum of the Higgs boson is required: $p_T > 100\,\mathrm{GeV}$.

Events are simulated using MC generators to model signal and background distributions. The ggF and VBF signal events are simulated using `POWHEG` [8, 9, 10, 11] interfaced to

PYTHIA8 [12], VH signal events are simulated with PYTHIA8. For ggF and VH processes, next-to-next-to-leading order (NNLO) accuracy in QCD and next-to-leading order (NLO) electroweak corrections are considered. The VBF production process is normalised to the cross section in NLO QCD accuracy and NLO electroweak corrections as well as an approximate NNLO QCD correction are applied. The main background, given by $Z/\gamma^* \to \tau\tau$ events, is simulated using data from $Z/\gamma^* \to \mu\mu$ events, where the muon event properties are replaced by simulated tau decay final-state particles. This approach aims to model the boson kinematics, hadronic activity and pile-up contributions very well.

For the event selection, single lepton, dilepton and di-τ_{had} triggers are used. According to the final-state particles, where each tau lepton decays either leptonically ($\tau \to l\,\nu\,\overline{\nu}$ with $l = e,\,\mu$) or hadronically ($\tau \to \mathrm{hadrons}\,\nu$), the two categories are divided into three channels, denoted by $\tau_{\mathrm{lep}}\tau_{\mathrm{lep}}$, $\tau_{\mathrm{lep}}\tau_{\mathrm{had}}$ and $\tau_{\mathrm{had}}\tau_{\mathrm{had}}$. Different selection criteria are required for the channels and they slightly differ for the two data-taking periods. For each channel, the tau decay candidates need to be isolated, to have opposite electric charge, and to pass certain p_T thresholds and other kinematic constraints.

For the background estimation, different data driven methods and simulations among the channels are used. Several control regions are used to improve the background estimation. The dominant background process, common to all background channels, comes from $Z \to \tau\tau$ decays. Other background processes are: fake τ, $Z \to ll$, $W + \mathrm{jets}$, top quarks, diboson background and other Higgs boson decays.

Due to the large amount of background processes and few signal contributions, a multivariate analysis based on Boosted Decision Trees (BDT) is applied. It is cross-checked using a cut based analysis. In the BDT analysis, a single discriminant is built for each event, which is in the range of -1 (most background-like) to 1 (most signal-like). For each category and channel, a BDT is trained separately using MC simulations. Due to different kinematics among the channels, different input variables are used and only the ones which lead to an improved discrimination are kept.

The BDT output distribution and discrimination between signal and background are affected by systematic uncertainties. They can be grouped into the following categories:

- Experimental uncertainties: luminosity, efficiencies for triggers, reconstruction and identification of particles, jet and tau energy scales and the energy resolution.
- Background modelling uncertainties.
- Theoretical uncertainties: cross sections, parton distribution functions, modelling of the underlying event, ...

The signal strength μ, defined as the ratio of the estimated signal yield in the measurement over the Standard Model expectation, is determined using a simultaneous maximum-likelihood fit on all categories. Information from the control region is used to stabilise the fit. Free and constrained nuisance parameters are involved.

Considering a Higgs-boson mass of $m_H = 125.36\,\mathrm{GeV}$, the final result for the signal strength is:

$$\mu = 1.43^{+0.27}_{-0.26}(\mathrm{stat.})^{+0.32}_{-0.25}(\mathrm{syst.}) \pm 0.09(\mathrm{theory\ syst.}). \quad (1)$$

The value is consistent with the Yukawa coupling strength predicted by the Standard Model.

In addition, a test statistic is constructed, according to a profile likelihood ratio, to measure the compatibility of the background-only hypothesis with observed data. Data

are compatible with the background-only hypothesis at the following p-value:

$$p_0 = 2.7 \times 10^{-6}, \tag{2}$$

corresponding to a deviation of $4.5\,\sigma$ from the background expectation, while $3.4\,\sigma$ are expected. The measurement provides evidence for the direct coupling of Higgs bosons to fermions. The final signal and background yield is shown in figure 2.

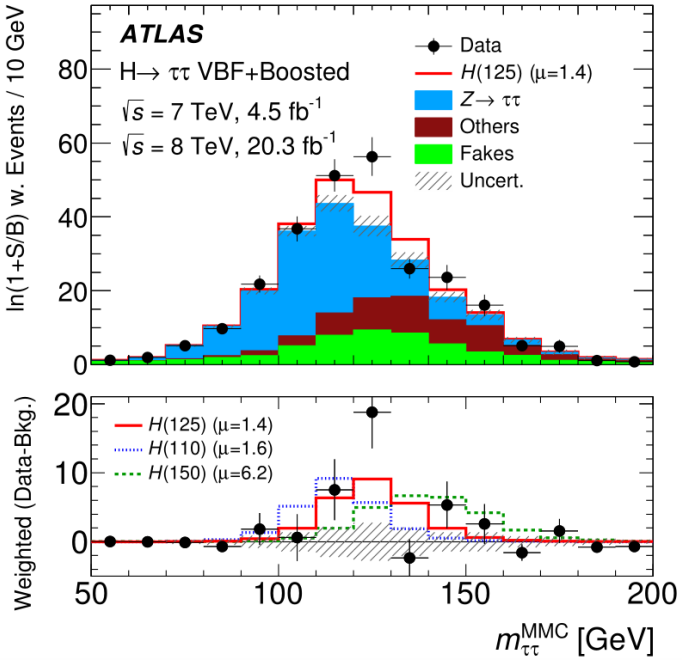


Figure 2: Signal and background contributions as a function of the invariant mass of the tau-lepton pair combined from all channels. The events are weighted by $\ln(1 + S/B)$, where S and B are the signal and background predictions from the BDT. The smaller bottom figure is a weighted histogram showing the signal yield in comparison to the Standard Model expectation.

2. Machine Learning Techniques

A formal and widely quoted definition of machine learning was provided by Tom Mitchell [13]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." That means in the particle physics context: Using machine learning algorithms, a computer can discriminate between signal and background events in high dimensional datasets. To do so, a supervised training instead of an explicit programming is needed.

For this project work, two different kind of classifiers are used: Deep Neural Networks (DNNs) and Boosted Decision Trees (BDT). These Models are explained in the following chapter.

2.1. Deep Neural Networks

Artificial neural networks are built similar to the human brain. They comprise several nodes that are connected and create an output value from a multidimensional input. If there are hidden layers in between, the network is called Deep Neural Network.

Therefore, each DNN consists of an input layer, a certain number of hidden layers, and an output layer. This last layer of the DNN consists of at least one node. A DNN with two hidden layers is shown schematically in figure 3.

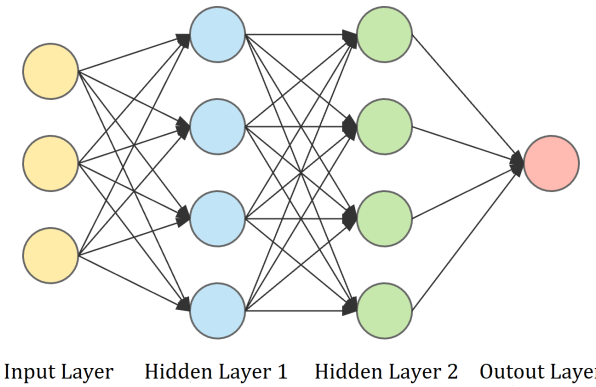


Figure 3: Basic structure of a DNN.

The network classifies $n-$dimensional input vectors $\vec{x}$, where each dimension x_i can be e.g. a physical observable. Then the input layer consists of exactly n nodes. Usually, the input is scaled to a certain range to improve the classification. Every node of the input layer is connected to every node of the first hidden layer by a weight. Each hidden layer is connected to the next one in the same way, until the last hidden layer is connected to the output layer. If the output layer consists of exactly one node, a single discriminating variable is calculated. The output value of each node is computed with an activation function. An example of a useful activation function is the rectified linear unit function $ReLU$:

$$\mathrm{ReLU}(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \tag{3}$$

For the output node, a useful activation function is the sigmoid function σ:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{4}$$

that has output values from zero to unity. In our case, an output value close to zero is interpreted as a background-like event, a value near to unity as a signal-like event.

In each layer, the calculation of the output $\vec{y}$ from the input $\vec{x}$ can be summarised as

$$\vec{y} = f(\mathbf{W}\vec{x} + \vec{b}), \tag{5}$$

where f is the activation function, $\mathbf{W}$ is the kernel, which is a matrix containing the weights and $\vec{b}$ is called bias, that adds a degree of freedom to each layer.

The weights of the network are optimised in a process called supervised training. The loss function is minimised in each epoch. The loss is a performance measure of the classification and gets a high value for wrong classified objects. A commonly used loss function is the binary cross entropy $H_p(q)$:

$$H_p(q) = -\sum_{i}^{N_{\text{events}}} \left[p_i \log q_i + (1 - p_i) \log(1 - q_i)\right] \tag{6}$$

Where $\vec{p} = (p_1, \ldots, p_{N_{\text{events}}})$ contains the true label of each simulated event, $\vec{q} = (q_1, \ldots, q_{N_{\text{events}}})$ is the current estimate of the DNN and N_{events} is the number of simulated events.

The weights are optimised using a gradient descent of the loss. This can be written as

$$w^{(i+1)} = w^{(i)} - h \cdot \nabla H_p(q, w^{(i)}), \tag{7}$$

$$b^{(i+1)} = b^{(i)} - h \cdot \nabla H_p(q, b^{(i)}), \tag{8}$$

where h is the learning rate, and $w^{(i)}$ and $b^{(i)}$ are the weight and the bias of the i-th training epoch. A very efficient way of gradient descent based optimisation is implemented in the Adam algorithm [14], which is commonly used.

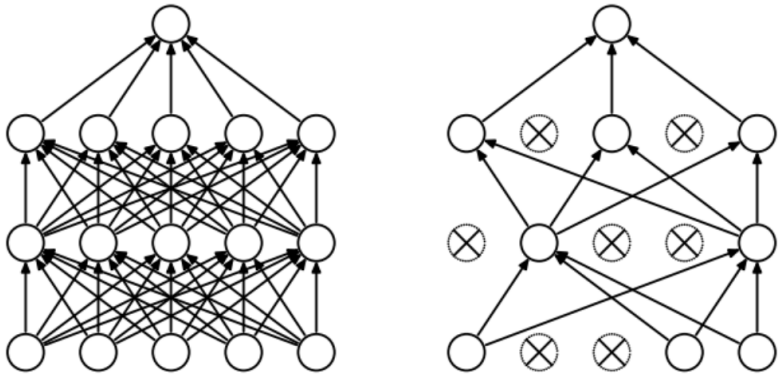
The weights are adapted after a batch of training examples is evaluated. The batch size is a hyperparameter and can vary from small values to the number of training events.

For a DNN with a high number of trainable parameters, it is important to avoid overtraining: a flexible classifier may conform too closely to the training samples and the same setting will not perform well on an independent test data sample. Some useful regularisation methods to avoid overtraining are:

- Dropout layers: In each training step, only a node is active with a probability of $1 - r$, where r denotes the dropout rate. This makes the neural network build strong subsets of neurons and shown in figure 4.
- Penalty terms: there is an additional loss for big weights by adding an extra term with the penalty-parameter λ. Two methods are common: L1-regularisation, with a penalty of $\lambda|w_i|$ and L2-regularisation, with $\frac{1}{2}\lambda|w_i|^2$.

2.2. Gradient Boosted Decision Trees

Gradient boosted decision trees are methods useful to combine many weak learners into a strong classifier. Here, the weak learners are decision trees. A decision tree splits



(a) Standard Neural Net (b) After applying dropout.

Figure 4: Illustration of the dropout method: in a training step, some neurons are disconnected [15].

data samples recursively according to cuts on features. The data is split based on a value of one of the input features at each node. Terminal nodes are called leaves. They represent a class label or probability.

BDT consider an additive model of M trees in the following form:

$$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x) \tag{9}$$

where h_m denotes the $m-$th decision tree and γ_m is the step length. The model is created iteratively in the following way:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{10}$$

where the h_m tries to minimise the loss function L via

$$h_m = \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x_i)), \tag{11}$$

where n is the number of training samples and y_i is the $i-$th label [16].

Gradient boosting attempts to solve the minimisation numerically. This method is similar to the one used for DNNs. The step length γ_m is chosen as:

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}\right) \tag{12}$$

In a model built like this, there are certain tuneable parameters. Some examples are:

- The number of trees M.
- The maximum depth of each tree, which is the number of nodes in tree.
- The number of leaves per tree.
- The learning rate, a scaling factor to the step length.
- The loss function L.

3. Classification Analysis

The main procedures that we followed to perform the classification are described in this chapter.

3.1. The Dataset

The dataset is a part of the analysis explained in section 1. The data are simulated through MC generators. For the Kaggle Higgs Boson Machine Learning Challenge, some simplifications were made on the dataset:

- Only the $\tau_{\text{lep}}\tau_{\text{had}}$ channel data are included.
- Only dominant background events are included.
- Events with negative weights are removed.
- Derived variables that may help to classify events are calculated from the primitive variables.
- Correction factors for the signal and background events have not been applied.

The dataset consists of 30 variables, from which 13 are derived from the other 17 primitive variables. The primitive variables are: transverse momentum p_T, pseudo-rapidity η and azimuthal angle ϕ of the lepton, the hadronically decaying tau, the missing transverse energy and the two leading jets as soon as available. The derived variables are physically meaningful variables calculated from the primitive ones by the ATLAS collaboration. Therefore, there are high correlations between the primitive and derived variables, which are shown in figure 5. In addition, there is a weight and a label for each event. The weight represents the cross-section of the processes, being usually higher for the background events. It corrects the fact that a similar number of signal and background events is provided for the training. The label consists of s or b values for signal or background events. The following subsets with normalised weights are used for the analysis:

- Training set: $250\,000$ events. Used for the training.
- Public leaderboard set: $100\,000$ events. Used as validation set.
- Private leaderboard set: $450\,000$ events. Used as test set.

3.2. Classification Procedures

Two different machine learning models are probed for the classification: BDT and DNNs. As in the Kaggle challenge, each classification result is evaluated using the approximate mean significance (AMS) on the validation set, in order to find the best model:

$$\text{AMS} = \sqrt{2\left((s+b+b_{reg})\ln\left(1+\frac{s}{b+b_{reg}}\right)-s\right)}, \tag{13}$$

where $s = \sum_i w_i^s$ is the sum over the true positive signal weights, $b = \sum_i w_i^b$ is the sum over the false positive weights and $b_{reg} = 10$ is a regularisation term that reduces the variance of the AMS.

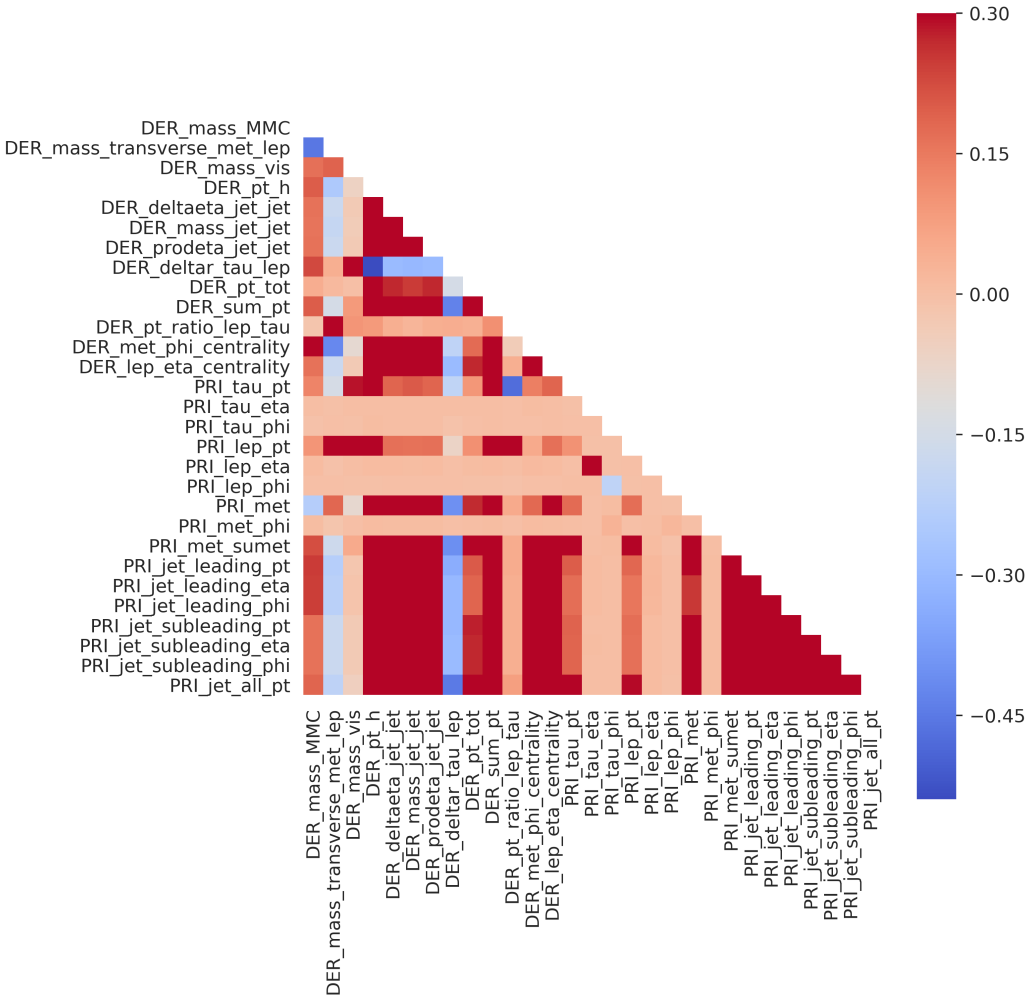


Figure 5: Correlation matrix: The linear correlations between each feature are shown as a heat map.

Different scaling algorithms are evaluated for the input data; the best classification is achieved using standard scaling. Each feature x is transformed according to

$$z = \frac{x - \mu}{s}, \tag{14}$$

where μ is the mean of the variable and s is the standard deviation. Moreover, the distribution of the derived variable for the invariant mass of the Higgs decay candidate is set to the mean of the distribution for each missing value. This improved the classification slightly.

Different DNN architectures are applied. They are implemented using `Keras` in the `TensorFlow` backend [17]. It seemed to be helpful to split the dataset according to the number of jets and then train a DNN for each subset separately:

- Events with 0 jets (100 000 events).
- Events with 1 jet (78 000 events).

- Events with ≥ 2 jets ($72\,000$ events).

Meaningless features about jets are then dropped for the subsets. The dropped features are 13 variables for the $0\ jets$-subset, eight variables for the $1\ jet$-subset and all variables are kept for the $\geq 2\ jets$-subset. The combination of the classifications processes, performed on each subset, reaches a higher validation set AMS than the one obtained using only one DNN for the whole dataset. A reason might be too low statistics in the training set, so that the DNN is not able to evaluate the different kinematics and missing values set to a dummy value accurately.

Another improvement of the classification is reached by adding variables formed by relative angles. All added features differ in the distributions for Signal and background. In particular, the new features are:

- The azimuthal angle between the lepton and the hardonically decayed tau lepton.
- The azimuthal angle between the lepton and the missing transverse energy.
- The azimuthal angle between the two jets.
- The difference of the pseudorapidities of the lepton and the hardonically decayed tau lepton.

Their distributions are shown in figure 6. In general, this correlation between variables can be found by a DNN on its own, but here, by adding these new features the AMS is slightly increased. Then a different number of primitive angular variables is dropped for each subset, if keeping the variable does not lead to an improved classification.

A grid search is performed is performed for each subset, which tries permutations of different hyperparameters. The DNN structures with the highest performance are summarised in table 1. All networks use binary cross entropy as loss function, ReLU as activa-

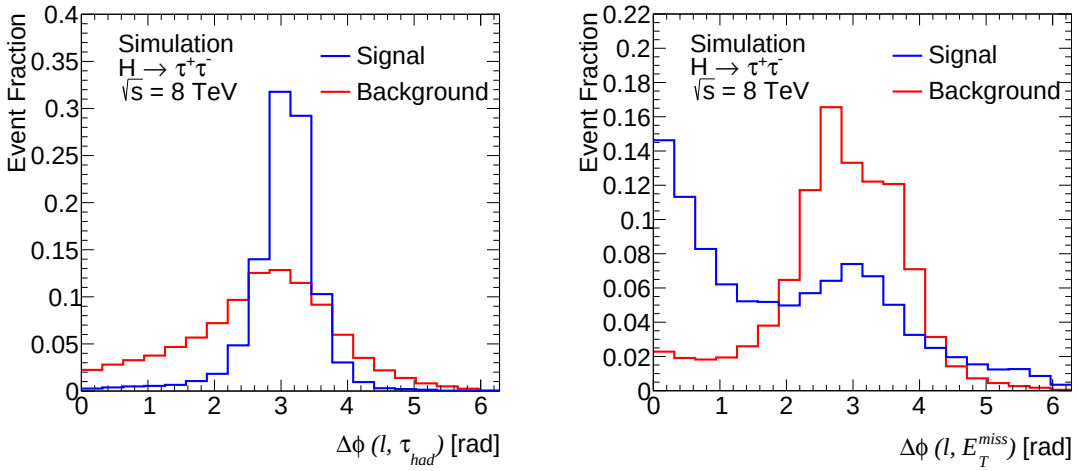
Table 1: Structure of the three DNNs used for our classification.

Subset	Hidden Layers	Nodes	Dropout rate	Regularisation
0 Jets	6	$32, 64, 128, 64, 32, 8$	0.1	L1, $\lambda = 0.0001$
1 Jet	5	$64, 64, 64, 32, 8$	0.1	L1, $\lambda = 0.0001$
2 Jets	6	$64, 128, 64, 64, 32, 8$	0.1	L2, $\lambda = 0.0001$

tion function for the hidden layers and a sigmoid output. The Adam algorithm is used for the optimisation. The *early stopping* method is used in order to avoid overtraining: each training process is stopped, when the accuracy of the validation set classification did not increase over the last ten training epochs and the best weights are restored. The accuracy A is defined as

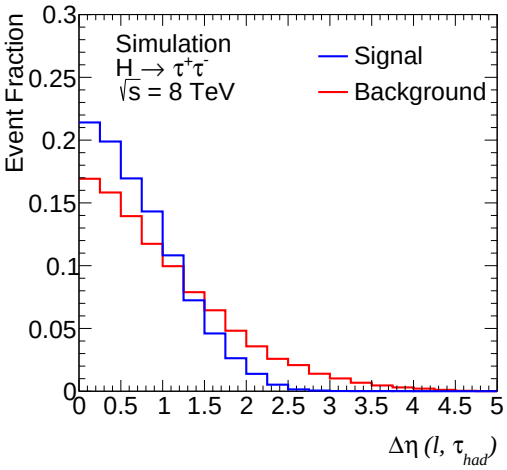
$$A = \frac{\mathrm{tp} + \mathrm{tn}}{\mathrm{tp} + \mathrm{tn} + \mathrm{fp} + \mathrm{fn}}, \tag{15}$$

where tp and tn denote the true positive and true negative classifications, while fp and fn denote the false positive and false negative classifications. The learning curves of the three DNNs are shown in figure 7. The networks reach accuracies in the range $0.82 - 0.85$, showing a clear separation between signal and background events for all subsets. Due to the increasing validation set accuracy in the training process, no hint for overtraining is found. After combining the classifications of the three DNNs, the AMS is calculated. The

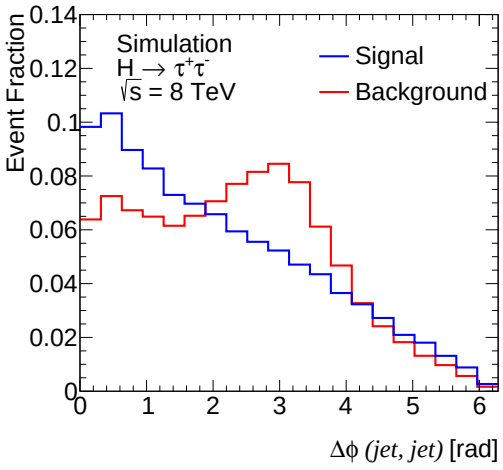


(a) $\Delta\phi(l, \tau_{\text{had}})$.

(b) $\Delta\phi(l, E_{\text{T}}^{\text{miss}})$.



(c) $\Delta\eta(l, \tau_{\text{had}})$



(d) $\Delta\phi(jet, jet)$.

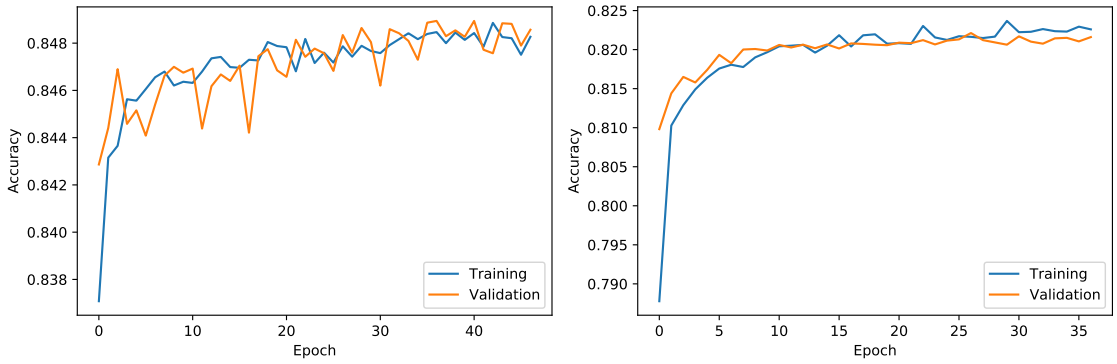
Figure 6: Distributions of the new variables, signal and background are shown separately. In (a), (b) and (c) only events without jets are considered.

highest significance of the validation set classification is 3.55, if events with a DNN output above a cut parameter of 0.83 are denoted as signal.

Complementary BDT-based models are probed in order to increase the AMS of the classification. The BDT classification is implemented using the `HistGradientBoostingClassifier` from `scikit-learn`. The classifier uses the algorithm explained in section 2.2, but first, every feature is histogrammed and a bin for missing values is provided in each feature. Therefore it is useful for the dataset, because many missing values are included. The algorithm is performed on the whole dataset, including the new features and standard scaling.

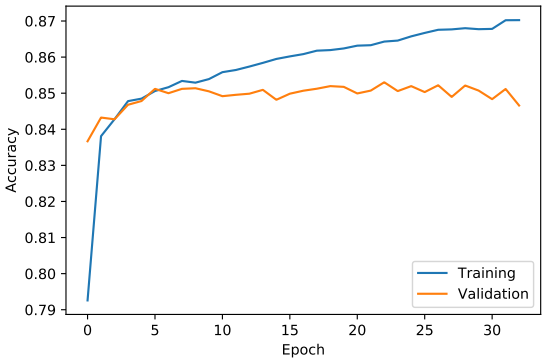
As in the previous part, the best combination of hyperparameters is found using a grid search. The following configuration is used:

- The number of trees: $M = 90$.
- The maximum depth of each tree: 50.
- L2 regularisation with $\lambda = 0.5$.



(a) 0 Jets.

(b) 1 Jet.



(c) 2 Jets.

Figure 7: Learning curves of the three DNNs. The accuracy is shown after each training epoch for the test and validation set.

- Learning rate: 0.1.
- Maximum number of leaves per tree: 50.
- Loss: binary cross entropy.
- Maximum number of bins for the histograms: 50.

The AMS from the BDT classification is 3.58 at a cut parameter of 0.83, which is slightly higher than the DNN classification.

3.3. Final Results

At the end, the two results from the different models are combined using a supervised logistic regression on the output distribution of the training set. This method builds an optimised combination of both outputs, which minimises the categorical cross entropy. The combination gets a higher AMS than the one of each model itself.

The AMS from each model and the combined score as a function of the cut parameter on the test set are shown in figure 8. The best AMS score from the combination is 3.65 at a cut parameter of 0.88.

The final shape of signal and background events for the test and training set is shown in figure 9. The discrimination between signal and background is really clear for both the

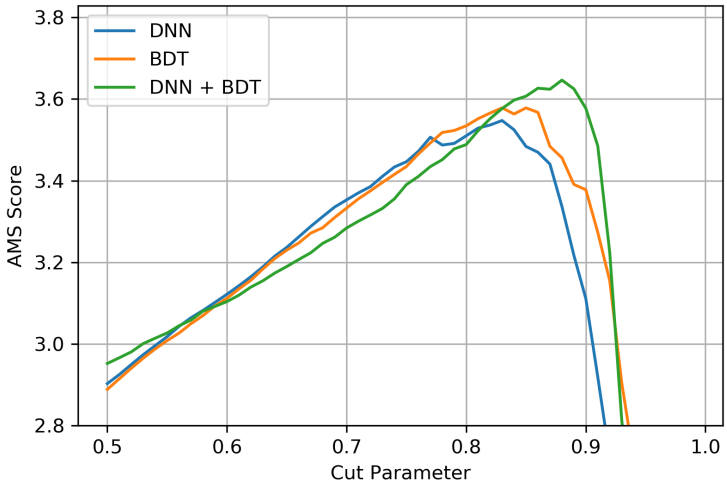


Figure 8: AMS of the test set classification as a function of the cut parameter for the DNN model, the BDT model and the combined model. The combination is observed to reach the highest AMS.

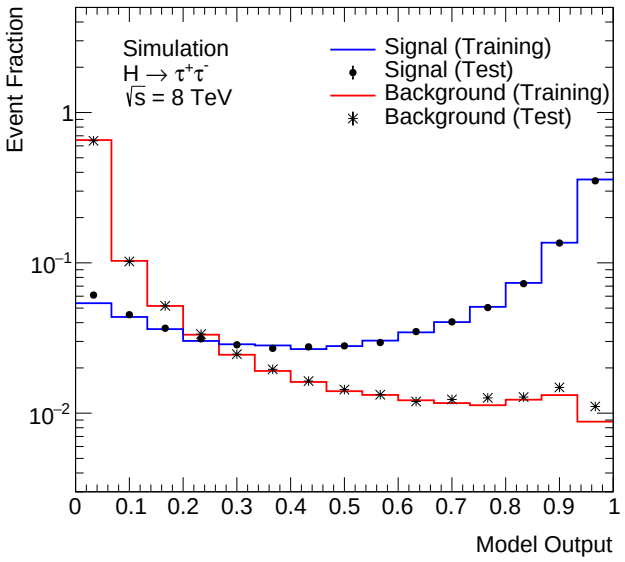


Figure 9: Model output distribution of the training test set. Signal and background are shown separately and normalised to unity. Small disagreements between both sets are found.

sets. The distributions of the two datasets are very similar, but there is still a significant offset in the four bins with the highest model output of the background distribution. More of the background events of the test set are assigned with a model output close to one than in the training set. This might be an effect of overtraining. More clearly, this is visible in the ratio of weighted events in figure 10: there is an excess of test over training set of a factor around 1.1 that cannot be explained by statistic fluctuations.

In order to investigate the reason of the small disagreement between test and training distributions, the plots of the normalised and weighted distribution are also given for the

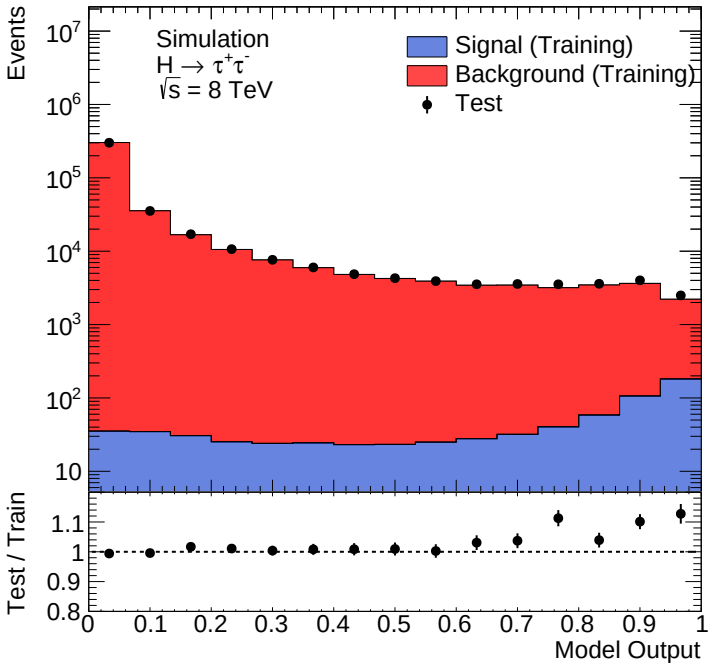


Figure 10: Weighted distribution of signal and background events of the training set. The weighted test set distribution is overlaid and their ratio is given below. Distributions of training and test set are similar, except for the last bins: the excess of test over training events is of a factor around 1.1.

DNN and for the BDT model separately in the appendix, see figures 11 and 12. The main disagreement between test and training set distribution arises from the BDT model, while both distributions of the DNN output are in good agreement. It shows that the BDT model is slightly overtrained on the training set, but it still gets a very accurate classification of the test set.

4. Conclusion

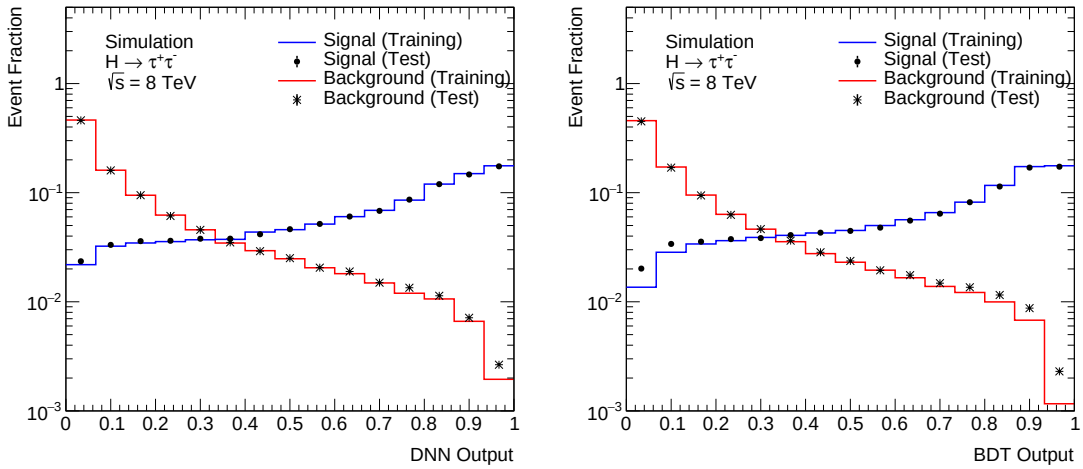
We worked on this project from December 2019 to April 2020. At the end, we want to summarise our experience in this way: it has been a very interesting experience that helped us to work on a project being part of a team. When we started, we were able to treat the problem from a theoretical point of view, which we had learned previously in other courses about statistical data analysis and particle physics. However, we only had very small practical experience with the implementation of machine learning algorithms, codes and particle physics analysis.

In this project we obtained complementary theoretical and practical knowledge to our former academic education. In particular, we got experience in implementing code in the context of particle physics. We also improved our Python programming skills significantly. Moreover, it was very motivating to work in a team with another student from a different country.

Another helpful point was to have the classification problem of the project available as a Kaggle challenge. This motivated us to get a better score and to compete with other people. We uploaded our results to see our rank in respect to the other participants. We scored at rank 445 of 1784. In addition, Kaggle provided lots of documentation material and a help forum, where common questions and topics are discussed.

Finally, we want to thank Dr. Olaf Nackenhorst for the organisation of our group in the Tandem Project, for all his help during the elaboration phase and for the opportunity to participate in this great experience. It has been a pleasure to join the frequent and interesting meetings where we summed up and discussed our progress and where open questions were answered. We definitely learned a lot in this project.

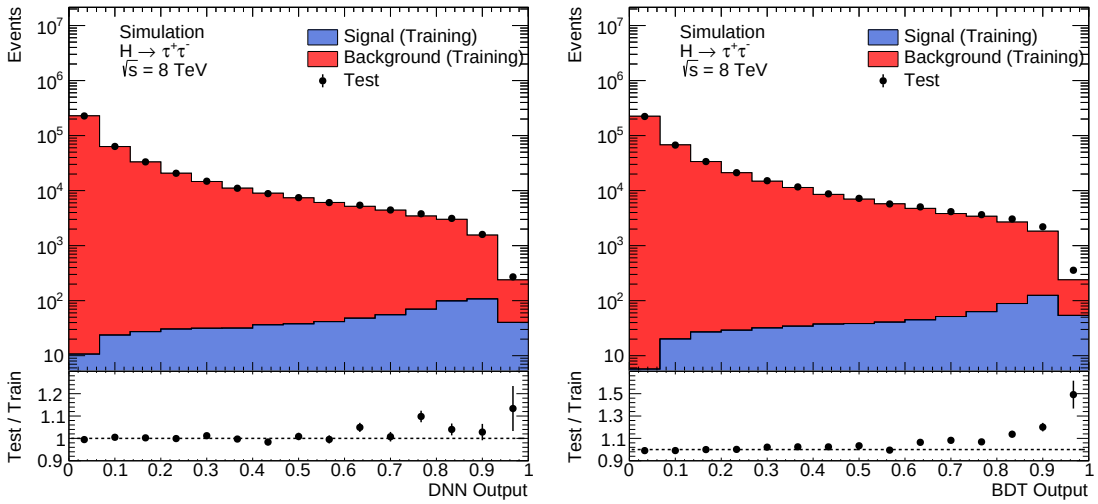
Appendix



(a) DNN Classification

(b) BDT Classification

Figure 11: Distribution of signal and background shape of the test and training set, separately for the DNN and the BDT model. The distributions of the test and training set from the DNN model are in good agreement, while a systematic bias is found for the BDT model.



(a) DNN output.

(b) BDT output.

Figure 12: Weighted distribution of signal and background events of the training set with respect to the model discriminant. The test set distribution is overlaid and the ratio of test over training events is given in figure 12a for the DNN and in figure 12b for the BDT classification. The training and test set distributions of the BDT model differ, there is an excess of test set contributions of $50\,\%$ in the last bin. For the DNN output, the statistics on the last bin are low: the agreement between test and training set is around one standard deviation.

References

[1] ATLAS Collaboration. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Phys. Lett. B* ***716*** *1*, 2012.

[2] CMS Collaboration. Observation of a new boson at a mass of 125gev with the cms experiment at the lhc. *Phys. Lett. B* ***716*** *30*, 2012.

[3] ATLAS Collaboration. Evidence for the higgs-boson yukawa coupling to tau leptons with the atlas detector. *JHEP* ***04*** *117*, 2015. doi: 10.1007/JHEP04(2015)117.

[4] Claire Adam-Bourdarios, Glen Cowan, Cécile Germain, Isabelle Guyon, Balázs Kégl, and David Rousseau. Learning to discover: the higgs boson machine learning challenge. *NIPS 2014 Workshop on High-energy Physics and Machine Learning*, 2014.

[5] ATLAS Collaboration. Dataset from the atlas higgs boson machine learning challenge 2014. *CERN Open Data Portal*, 2014. doi: 10.7483/OPENDATA.ATLAS.ZBP2.M5T8.

[6] ATLAS Collaboration. The atlas experiment at the cern large hadron collider. *JINST* ***3*** *S08003*, 2014.

[7] ETH Zürich. Higgs boson.

[8] P. Nason. A new method for combining nlo qcd with shower monte carlo algorithms. *JHEP* ***11*** *040*, 2004.

[9] Stefano Frixione, Paolo Nason, and Carlo Oleari. Matching nlo qcd computations with parton shower simulations: the powheg method. *JHEP* ***11*** *070*, 2007.

[10] C. Oleari S. Alioli, P. Nason and E. Re. A general framework for implementing nlo calculations in shower monte carlo programs: the powheg box. *JHEP* ***06*** *43*, 2010.

[11] P. Slavich E. Bagnaschi, G. Degrassi and A. Vicini. Higgs production via gluon fusion in the powheg approach in the sm and in the mssm. *JHEP* ***04*** *088*, 2012.

[12] S. Mrenna T. Sjöstrand and P.Z. Skands. A brief introduction to pythia 8.1. *Comput.Phys. Commun.* ***178*** *852*, 2008.

[13] Tom Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[15] Srivastava and Nitish et al. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] François Chollet et al. Keras. `https://keras.io`, 2015.