CHAPTER 7

Results

Building upon the preparations described in the previous chapters, this chapter presents the four-top-quark signal classification results using Deep Neural Networks. In the first and second sections, both the construction and the obtained signal discrimination of Feedforward and Recurrent Neural Networks will be discussed. Section 7.4 presents a study on the training time of Neural Networks on both CPU and GPU. The last section is concerned with the reconstruction of two hadronically decaying tops; to improve the rejection of hard to distinguishing backgrounds in the four-top-quark analysis.
The statements made about Neural Networks in the following sections are not necessarily negotiable to other applications. The influence of hyperparameters and other ingredients of Neural Networks depends on the application and size of the dataset used.

7.1 Signal Classification using Feedforward Neural Networks

The strategy chosen to obtain an FNN that discriminates the $t\bar{t}t\bar{t}$ signal well from the background processes seeks to optimize the structure of the Neural Network and tune its most important hyperparameter. The correlation between different parameters is taken into consideration by investigating several parameters at the same time. To limit the number of Neural Networks that need to be trained to a reasonable amount, only parameters with presumably high correlations are varied simultaneously. Furthermore, unless mentioned otherwise, all performance measures obtained are based on a *binary classification* approach. In binary classification, only two classes are considered a signal and a background class. The AUC of the current classification is calculated at each Epoch on the testing set. The FNNs are trained until five consecutiveness Epochs did not improve the AUC, but at least 80 Epochs. The in the following section quoted AUC is always the mean of the two Neural Networks trained (Γ_{total}) unless stated otherwise. As described in Chapter 6, the 18 input features are first transformed to normal distributions and then reweighted before they enter the input layer.

Architecture

The first choice that has to be made is the number of parameters, also called the *size*, and the number layers of the Neural Network, referred to as the *depth*. As previously discussed, the architecture has to be complex enough to model the problem and not too complex to avoid overtraining.
The number of parameters needed for the $t\bar{t}t\bar{t}$ application is investigated by using fully connected layers and can be calculated according to Equation **??**. The parameters are distributed to the layers such that the number of neurons for progressive layers declines. Additionally, the minimum number of neurons in a hidden layer is defined to be one-quarter times the number of neurons in the input layer ($18 \cdot 0.25 = 4.5 \approx 5$), following the approach described in [???]. Since the dataset is small for a deep learning application, the number of hidden layers is kept small.
Neural Networks considered had 200 to 70000 tunable parameters and are evenly spread over the three

orders of magnitudes studied. For each investigated size, 10 Neural Networks with 1 to 10 hidden layers were trained. The only exceptions are Neural Networks with less than 1000 parameters, where the number of neurons would have fallen below 5. For Neural Networks with these sizes, only a maximum depth of 5 was considered. All other ingredients are caped constant and are summarized in Appendix **??**.

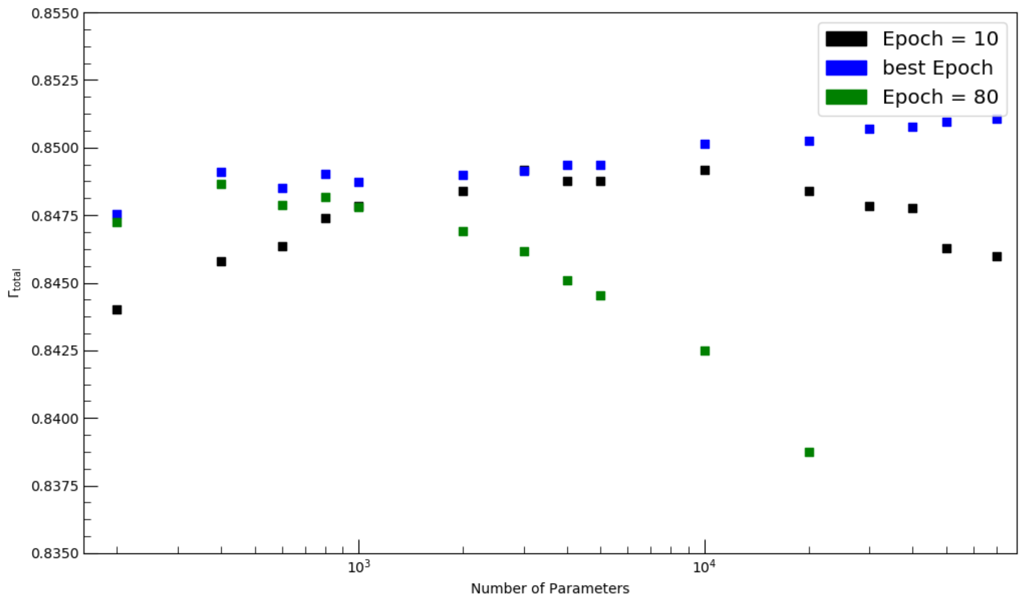


Figure 7.1: The averaged AUCs of FNNs with different number of parameters where the training is stop after 10, 80 and at the best-performing Epoch. The average includes the AUCs of the 4 best-performing Neural Networks of different depth.

Figure 7.1 shows the result of the number of parameter search where the Γ_{total} is the average of the 4 best-performing Neural Networks differing in the layer configuration. For all sizes, the standard deviation of the average is smaller than 0.001. For the blue squares, the Epoch with the highest AUC for each Neural Network was selected, whereas, for the black and green, the AUC at Epoch 10 and 80 is quoted. As can be seen, the Γ_{total} for the best Epoch increases, with small fluctuations, for higher numbers of parameters. For Neural Networks with more than 10000 parameters, the gained performance per additional trainable parameter is minor, indicating that the number of parameters is sufficient to obtain a reasonable separation between signal and background events. In the case of Neural Networks with few parameters, every weight needs to be fine-tuned to obtain the best possible performance, which can take several Epochs. On the other hand, for Neural Networks with many parameters, not all information available to a neuron is important. The learning process to identify which of the information is key can also slow down the training. These behaviors are reflected in the distribution of the training results evaluated at Epoch 10. The Neural Networks evaluated at Epoch 80 show a strong decrease in the Γ_{total} (obtained on the test sets) with the number of parameters, indicating overtraining. The tendency of large Neural Networks to have high overtraining is especially apparent in Figure 7.2 for high Epochs. However, when the AUC is calculated at the optimal Epoch, this dependency is significantly attenuated.

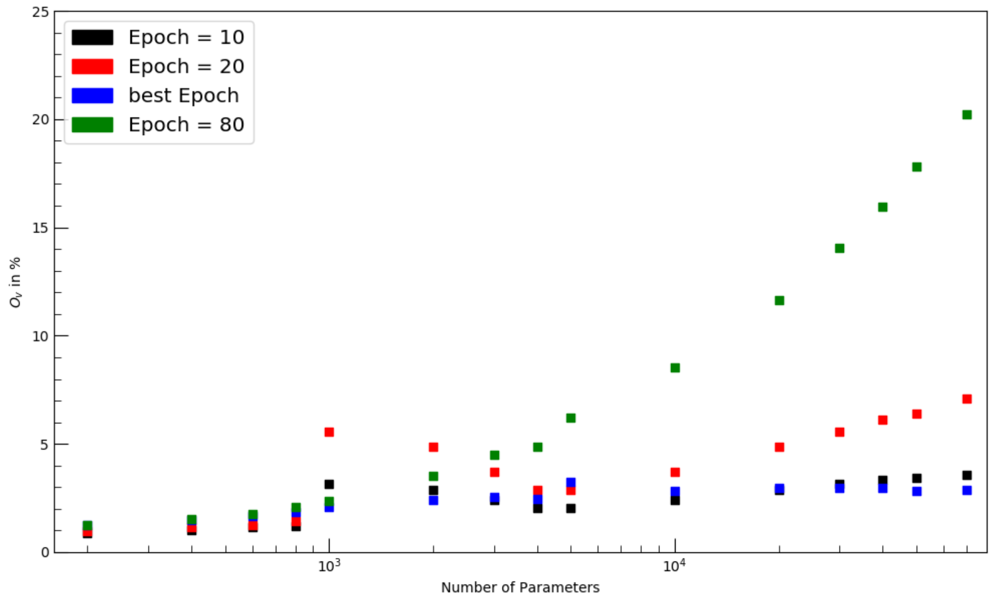


Figure 7.2: The average overtraining for different FNNs configurations. The training is stopped after 10, 20, 80 and at the best-performing Epoch. The average includes the overtrainings of the 4 best-performing Neural Networks of different depth.

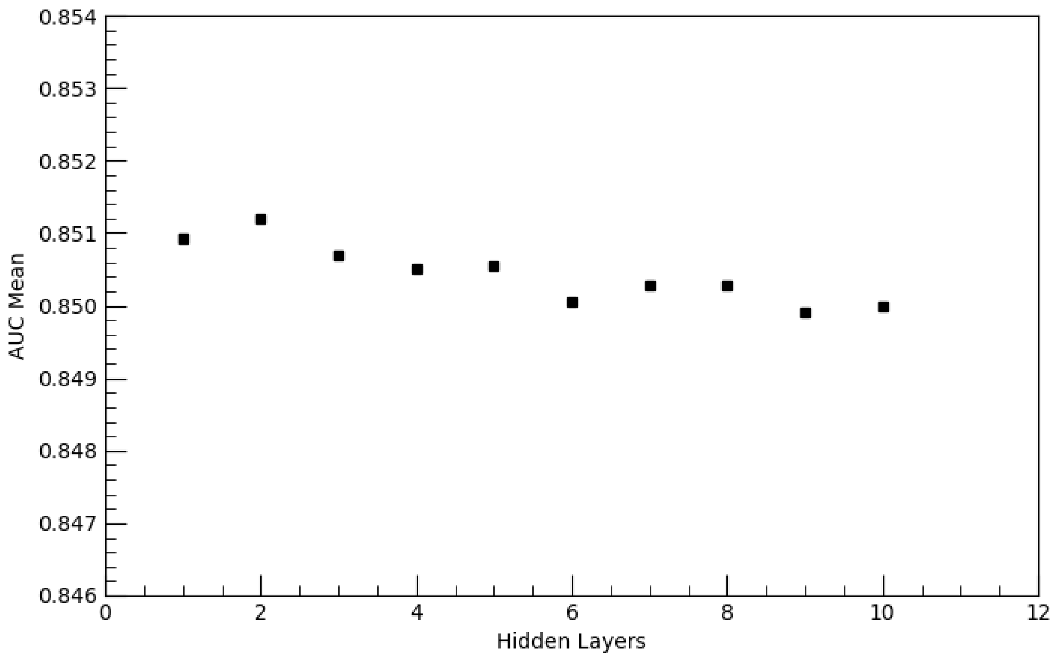


Figure 7.3: The average observed AUC for FNNs with different numbers of hidden layers. The average includes the AUCs of the 4 best-performing Neural Networks of different size.

The FNNs showing the highest AUC are models with low numbers of hidden layers, especially those with two layers. In Figure 7.3, the average AUC of the 4 best-performing Neural Networks with different sizes is shown. The overtraining was observed to be independent of the number of hidden layers.
The overall performance gained by varying the number of parameters and layers is small compared to the performance gained by hyperparameter tuning. This is because the selected optimization algorithm Adam is very stable in its final performance in combination with a reasonable initial learning rate. A cross-check using mini-batch Stochastic Gradient Descent (SGD) showed that for this algorithm, the performance does depend more on the number of parameters.
Summarizing the obtained result, it can be stated that a Neural Network for hyperparameter tuning should have at least 10000 parameters to ensure high flexibility and below 30000 to avoid high overtraining. For the studies on learning rate, activation function, and weight initialization, a Neural Network with 20000 parameters and 2 hidden layers is selected. Its overtraining is shown in Figure 7.4 together with the overtraining of other Neural Networks in the region of interest.

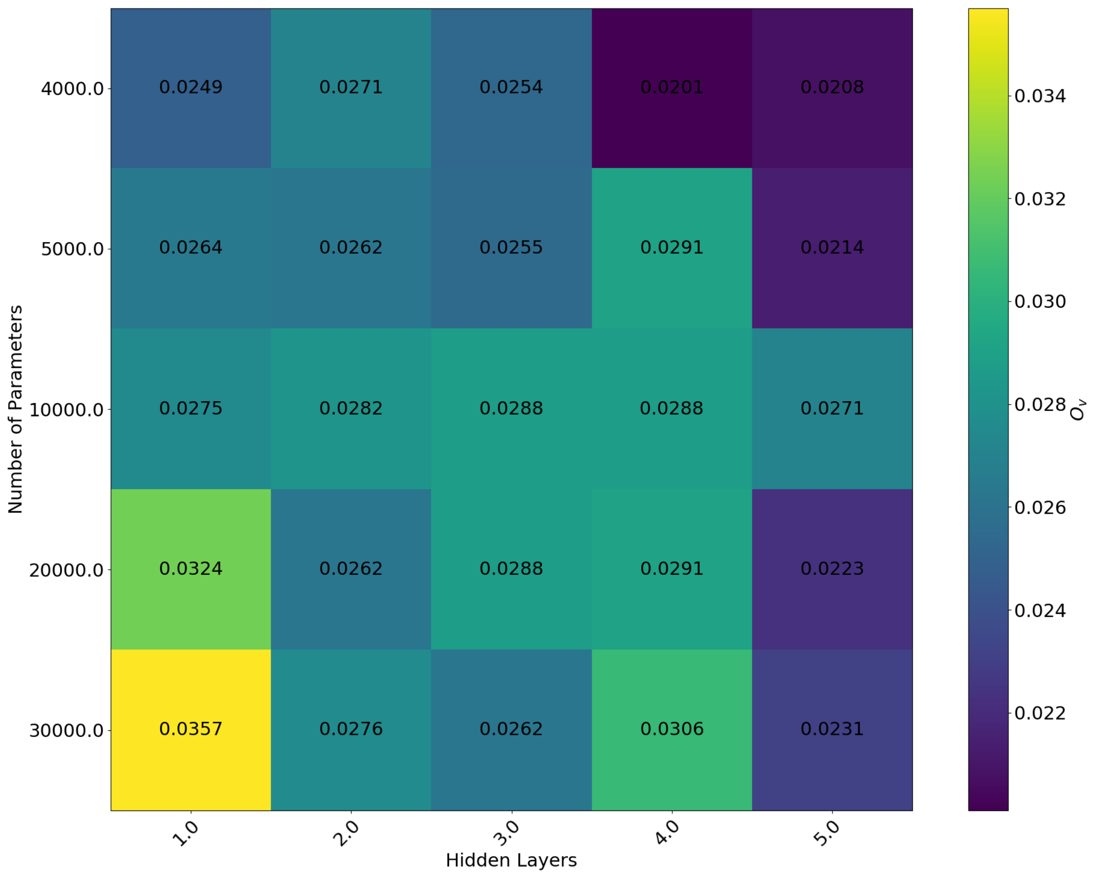


Figure 7.4: The overtraining for FNNs in the region were the observed AUCs are close to the obtained best-performance.

Weight Initialization and Activation Functions

The Neural Networks architecture selected in the last sub-section is now utilized to study the influence of activation functions and weight initialization on the AUC. As discussed in Chapter **??**, the different weight initializations are designed to prevent the output of the activation functions from vanishing or exploding during the forward pass. Therefore, a high correlation between the two can be expected.
The effect of activation functions is studied by setting the activation function of all neurons in all hidden layer to either ReLU, leaky ReLU, ELU ($\alpha = 1$), or SELU were three different slope constants for leaky ReLU are considered ($\alpha \in [0.1, 0.3, 0.5]$). For each activation function, the different weight initialization Glorot, He, and Lecun with normal and uniform distributions for all weights are tested. The activation function of the output layer remains to be the Sigmoid function. The biases are initialized to zero. All other parameters are kept constant and are summarized in Appendix **??**.

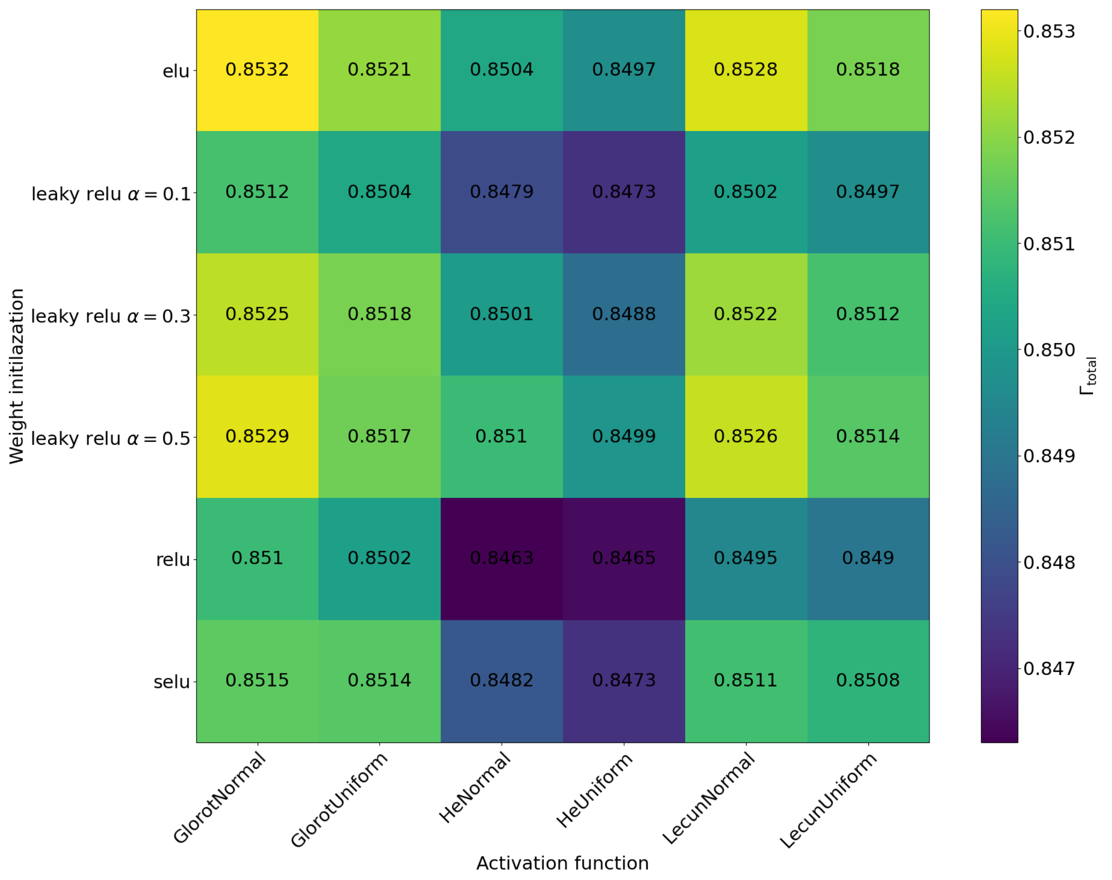
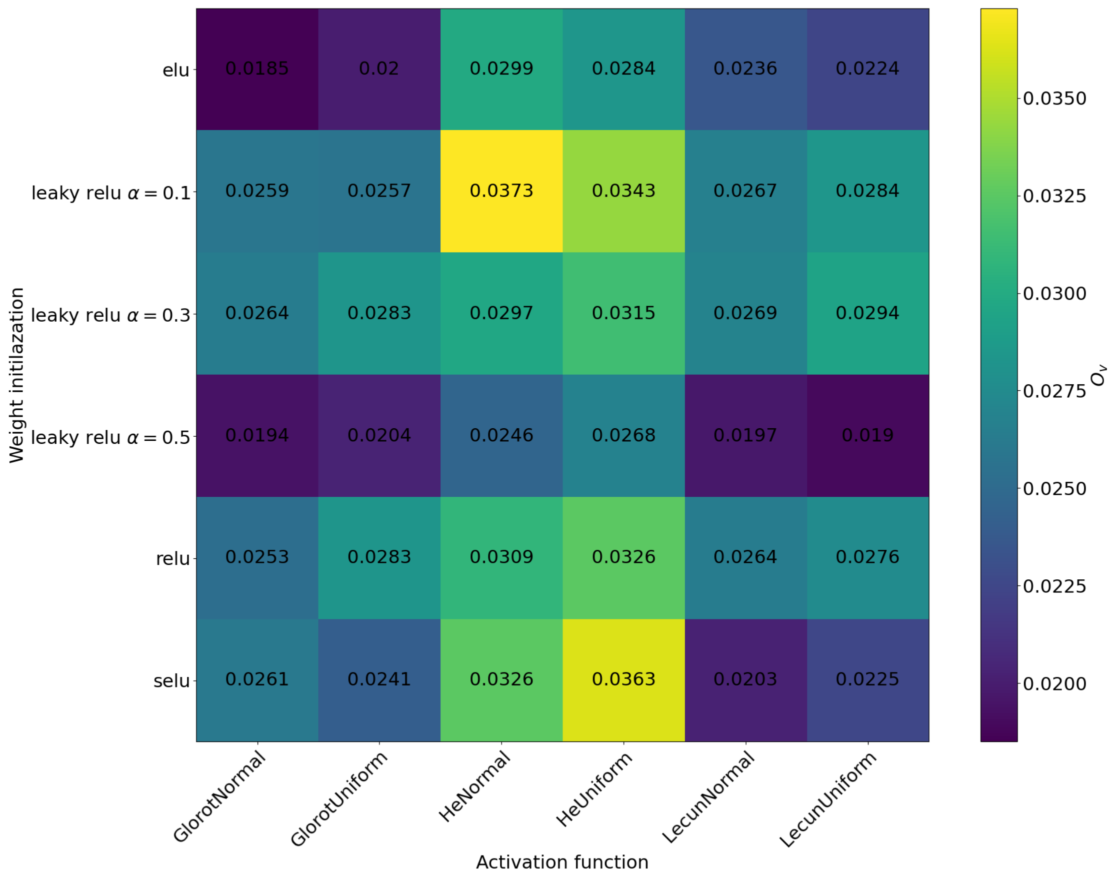


Figure 7.5: Obtained AUCs for different choices of activation functions and weight initialization.

Figure 7.5 shows the obtained AUCs for all combinations while Figure 7.6 shows the corresponding overtraining values. The best performance is reached when ELU is combined with the Glorot normal distribution weight initialization, which is the weight initialization that works best for all considered activation functions. The difference between normally and uniformly distributed weight initializations is minor but normally distributed initializations result in slightly higher AUC.
Activation functions with output values for negative net inputs different from zero outperform ReLU in all considered cases, indicating that a training in which some weights are never updated is not beneficial for this application. In fact, the performance increases with the slope constant α for leaky ReLU. As is the case for ELU, a smooth transition between the output values for negative and positive net input results in the best Γ_{total} for all weight initializations. The SELU activation function, specifically designed for FNNs,

does not lead to an improved result compared to ELU.
The overtraining for all considered activation functions and weight initializations remains below 5%. Counter-intuitively, the Neural Networks with the highest AUC also have the smallest overtraining and the lowest Γ_{total} on the training dataset. Therefore, the additional performance gained comes from the reduction of overtraining by the activation functions and weight initializations.



	GlorotNormal	GlorotUniform	HeNormal	HeUniform	LecunNormal	LecunUniform
elu	0.0185	0.02	0.0299	0.0284	0.0236	0.0224
leaky relu $\alpha = 0.1$	0.0259	0.0257	0.0373	0.0343	0.0267	0.0284
leaky relu $\alpha = 0.3$	0.0264	0.0283	0.0297	0.0315	0.0269	0.0294
leaky relu $\alpha = 0.5$	0.0194	0.0204	0.0246	0.0268	0.0197	0.019
relu	0.0253	0.0283	0.0309	0.0326	0.0264	0.0276
selu	0.0261	0.0241	0.0326	0.0363	0.0203	0.0225

Figure 7.6: The overtraining for different choices of activation functions and weight initialization.

Learning Rate and Optimization Algorithms

Building upon the results obtained in the first two studies, the performance for different optimization algorithms is investigated. The most impactful hyperparameters of the optimization algorithm are the learning rate, determining the step size of every weight update, and the batch size, the number of events used during one iteration.
Three different optimization algorithms are considered, namely SGD, Adam, and Rmsprop, in this study. For each optimization algorithm, distinct learning rates are investigated while also varying the batch size. The different batch sizes are selected as powers of 2 and reach from $2^8 = 256$ to $2^{14} = 16384$. All other ingredients are kept constant and are summarized in Appendix **??**.

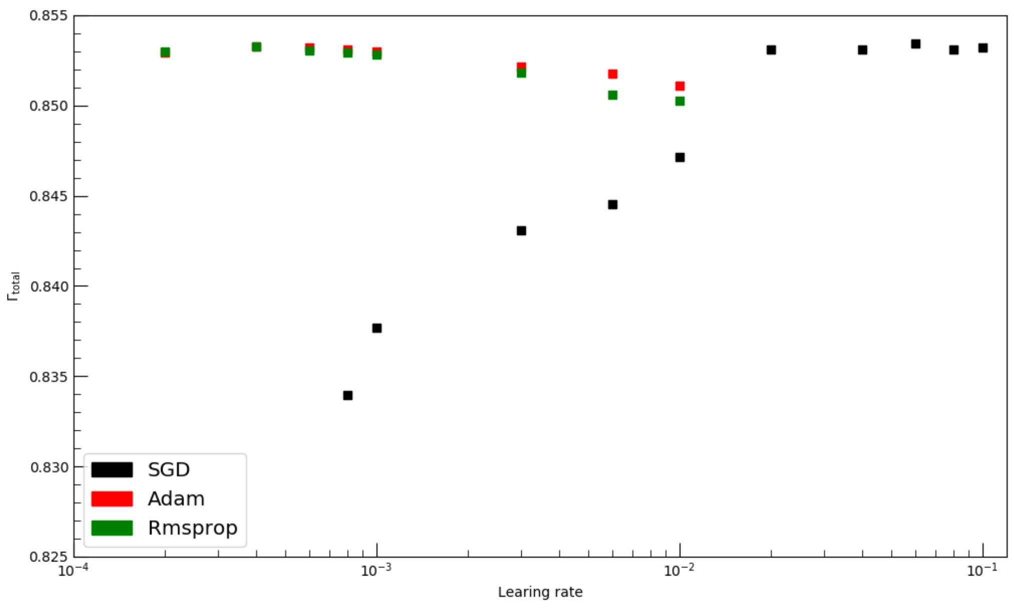


Figure 7.7: The obtained average AUC of FNNs with different learning rates and three distinct optimization algorithms. The average includes the AUCs of the 3 best-performing Neural Networks with different batchsizes.

Figure 7.7 shows the observed Γ_{total} values for different learning rates and optimization algorithms while averaging over the 3 best-performing batch sizes for each learning rate. All three investigated optimization algorithms show a similar peak performance. The improvements gained by fine-tuning the learning rate within one order of magnitude are minor. For Adam and Rmsprop, even learning rates one order of magnitude higher than the optimal observed learning rate result in only slightly worth AUCs while SGD is more sensitive. The reason being, Adam and Rmsprop calculate individually optimized learning rates for different parameters (weights and biases) based on the initial learning. At the same time, SGD always applies the same learning rate for all parameters. This internal optimization is also reflected in the fact that Neural Networks trained with SGD tend to need more Epochs until the best AUC is reached, compared to Neural Networks trained with Rmsprop or Adam. All considered models have a similar overtraining for all considered learning rates, shown in Figure 7.8. Which optimization algorithm is used for training does not influence the overtraining. The Neural Networks with less than 1% overtraining are the models that have poor AUC, indicating that the training for these models did not converge.

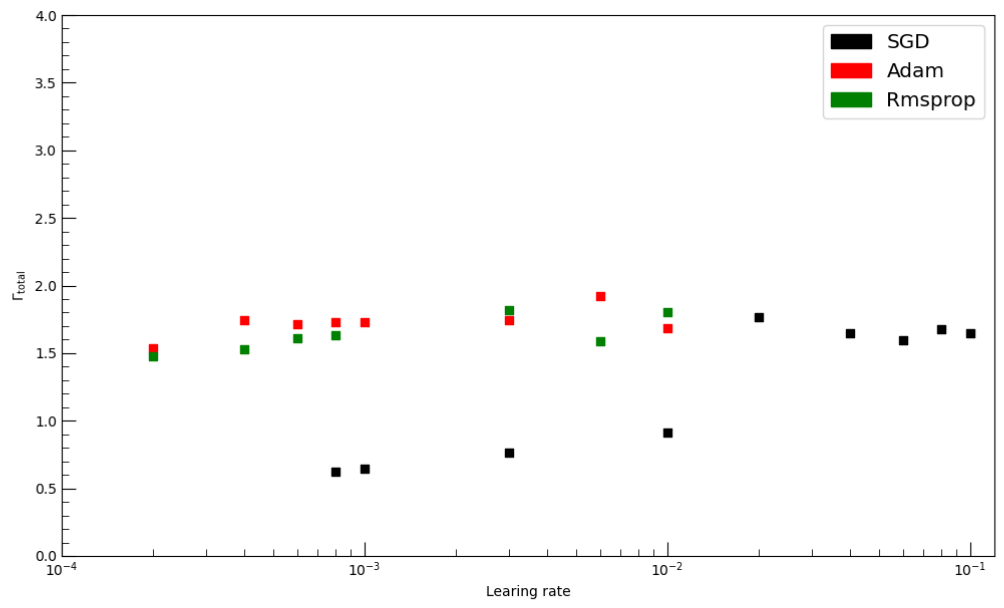


Figure 7.8: The overtraining of FNNs for different learning rates and three distinct optimization algorithms.

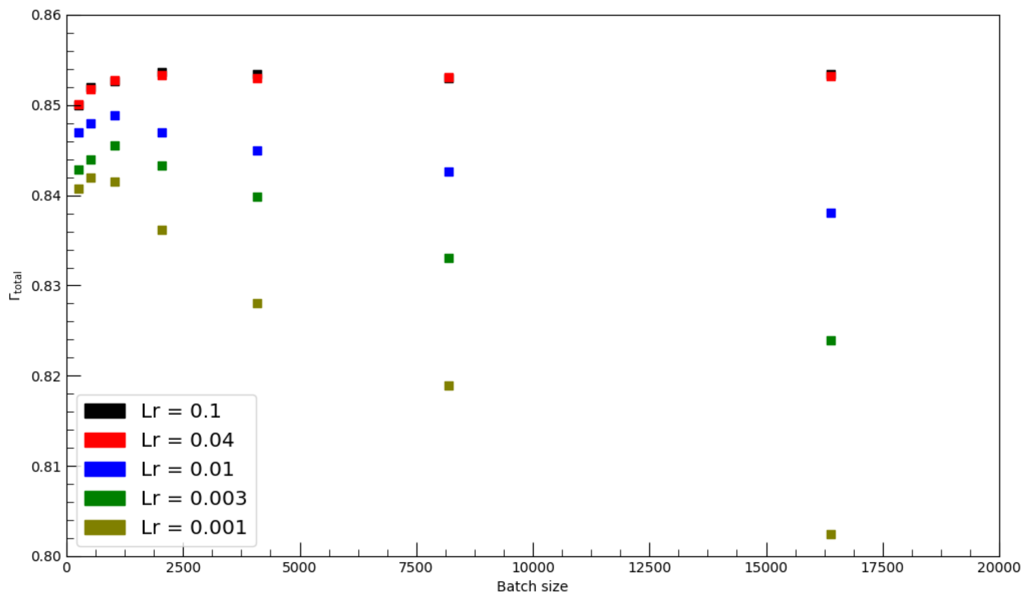


Figure 7.9: The AUC as a function of the batch size and the learning rate. The Neural Network were trained using SGD.

To study the dependence of Γ_{total} on the batch size, Figure 7.9 shows exclusively the results obtained by using SGD as the optimization algorithm. The AUC for carefully chosen learning rates (black and red) is almost independent of the batch size. On the other hand, for small learning rates, the batch size has a significant influence. For Neural Networks with small learning rates and large batch size i.e. few iterations per Epoch, the maximum considered number of 120 Epochs is reached. Thus, the training of those Neural Networks has not converged and results in a non-optimal weight configuration that decreases the obtained AUC. The drop in peak performance for small batch sizes (< 2000) can be attributed to the fact that a small batch may not contain events from all backgrounds. The weight updates calculated based on this batch may not be accurate enough to reach the optimal weight configuration during training. The best obtained Neural Network uses Adam with a learning rate of 0.003 and a batch size of 2048. The maximum Γ_{total} reached is 0.853, with an overtraining of 0.016 on the testing set.

Polynomial Learning Rate Decay

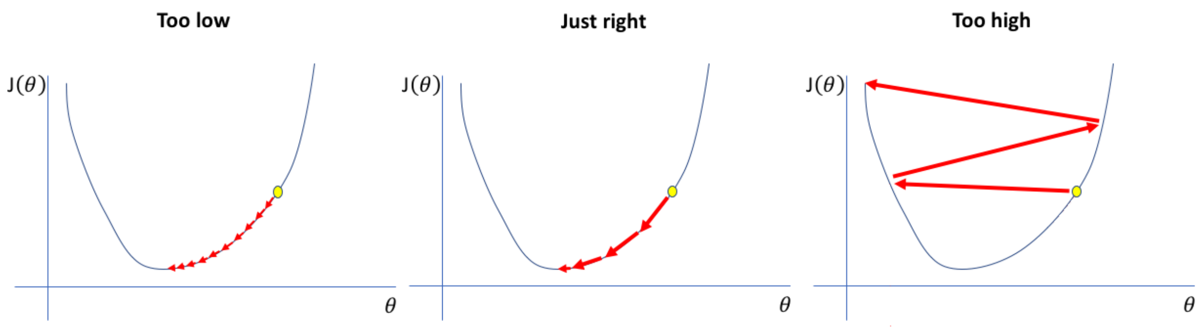


Figure 7.10: Training for three different learning rates at a minimum of the Loss function. Illustrating the advantage of polynomial learning rate decay.

Two scenarios frequently encounter during the training of Neural Networks are shown in Figure 7.10. In the first case, the learning rate is too low to reach the minimum of the Loss functions $J(\theta)$ during the training. On the other hand, if the learning rate is too large, the weight update overshoots the best performance point, resulting in a decreased AUC for large learning rates. Both cases can be avoided when using polynomial learning rate decay. In polynomial learning rate decay, the initial learning rate Lr_{int} is adjusted for each Epoch according to

$$Lr(E) = Lr_{\text{int}} \cdot (1 - \frac{E}{E_{\text{tot}}})^n \tag{7.1}$$

where $Lr(E)$ is the learning rate at Epoch E, and n is the used polynomial power. For this technique, the total number of Epochs E_{total} has to be fixed. Using polynomial learning rate decay results in more smooth coverage of the training towards the minimum.
For the studies on polynomial learning rate decay in this thesis, SGD was selected as the optimization algorithm to investigate the gained improvements independently of the parameter by parameter optimization used by Adam and Rmsprop. The initial learning is varied between 0.02 and 0.3, building upon the knowledge gained in the previous sub-section. Considered polynomials have the power of 1 to 3. The other parameters of the Neural Network are kept constant and are summarized in Appendix **??**.

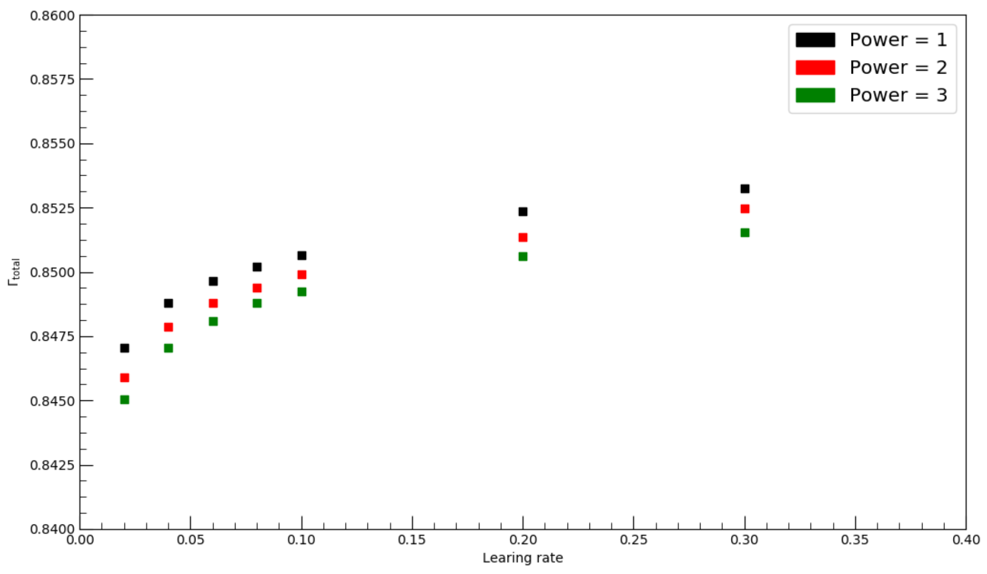


Figure 7.11: Obtained AUCs for FNNs trained using polynomial learning rate decay.

Figure 7.11 shows the results of the polynomial learning rate decay. Within the considered hyper-parameter ranges, no improvement of the peak performance can be observed compared to training with fixed learning rates. Learning rate decays with polynomials of power 1 perform best for all investigated learning rate. The reason for this is not apparent and needs further investigations in the future. One drawback of polynomial learning rate decay is that Neural Networks, on average, have to be trained long until the Epoch with the best performance is reached. However, for all learning rates considered except for $Lr_{\text{int}} = 0.02$ the Epoch having the best performance is more than 5 Epochs away from $E_{\text{tot}} = 80$. The overall stability of the best AUC against different learning rates is only improved slightly (cf. Figure 7.7).

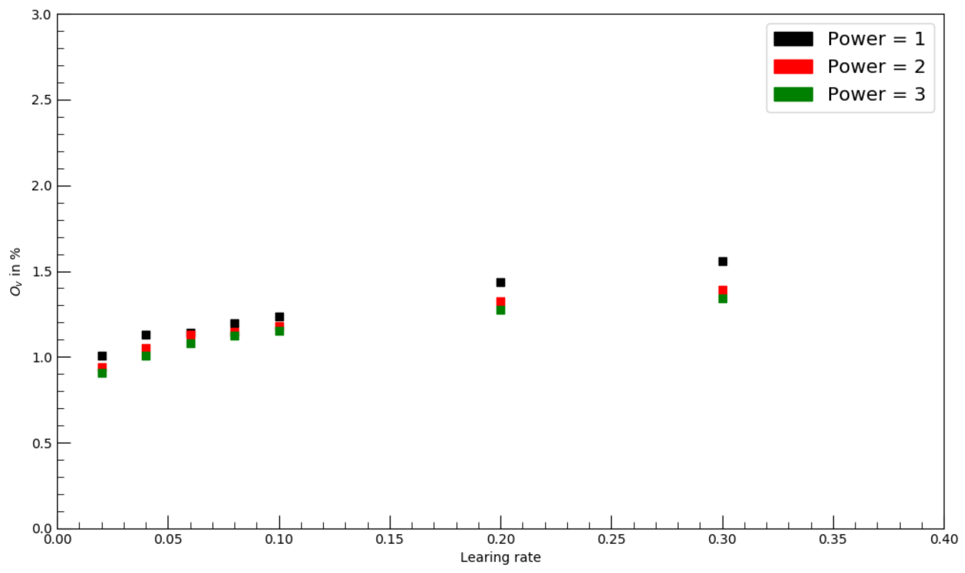
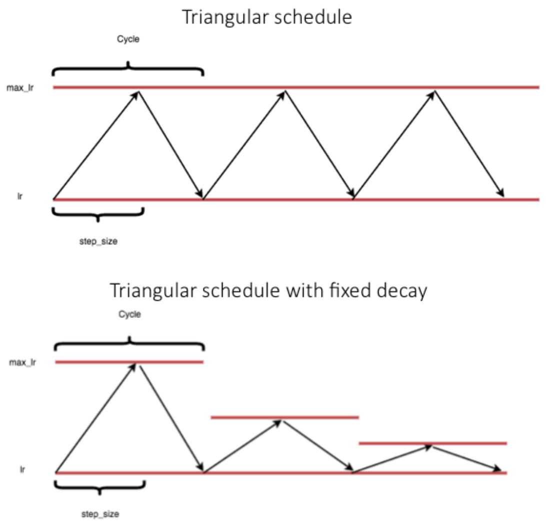


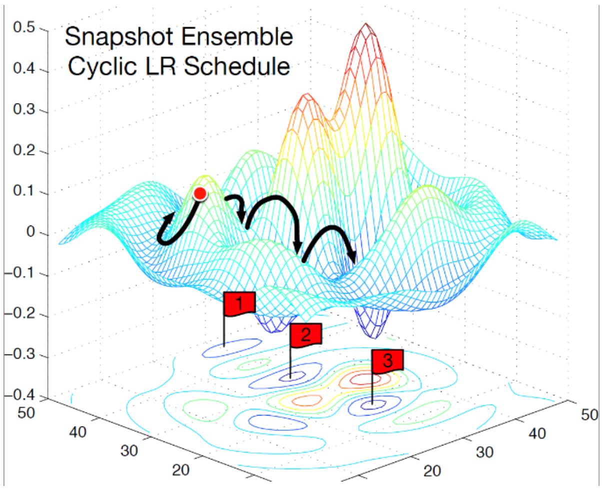
Figure 7.12: The overtraining for FNNs trained using polynomial learning rate decay.

The overtraining of the Neural Networks trained with polynomial learning rate decay, shown in Figure 7.12, decreases with the learning rate and is independent of the power of the polynomial used. Similar to the overtraining observed for SGD using a fixed learning rate, all trained models have an overtraining below 1.5‰.

Cyclic Learning Rate Decay



(a)



(b)

Figure 7.13: a) The learning rate schedule for cyclic learning rate decay with and without additional decay of the maximum learning rate. b) The topology of a Loss function for which cyclic learning rate decay can improve the performance of a neural Network.

Another attempt to improve the AUC by varying the learning rate during training is cyclic learning rate decay. The functionality is illustrated in Figure 7.13; during one cycle, the learning rate is linearly increased from a predefined minimum learning rate to a predefined maximum learning rate and after that linearly decreased until the minimum learning rate is reached. The number of batches needed to move from the maximum to the minimum learning rate or the other way around is a tunable hyperparameter called *stepsize*. Moreover, the maximum learning rate can be decreased after every cycle. This learning rate schedule is designed to prevent the training from getting stuck in local minima or on saddle points. The carried out cyclic learning rate decay study considers 3 different parameters. The maximum learning rates considered are of the magnitude 10^{-1}, while the minimum learning rates are of the magnitude 10^{-2}. Simultaneously 5 different stepsizes are considered, which are chosen close to the suggested $2-8\cdot$ iterations per Epoch ($\sim 160-640$) in [???]. Each combination was trained with a static maximum learning rate and a maximum learning rate that decreased to its half value after each cyclic. All other parameters are the same parameters used for the previous learning rate study.

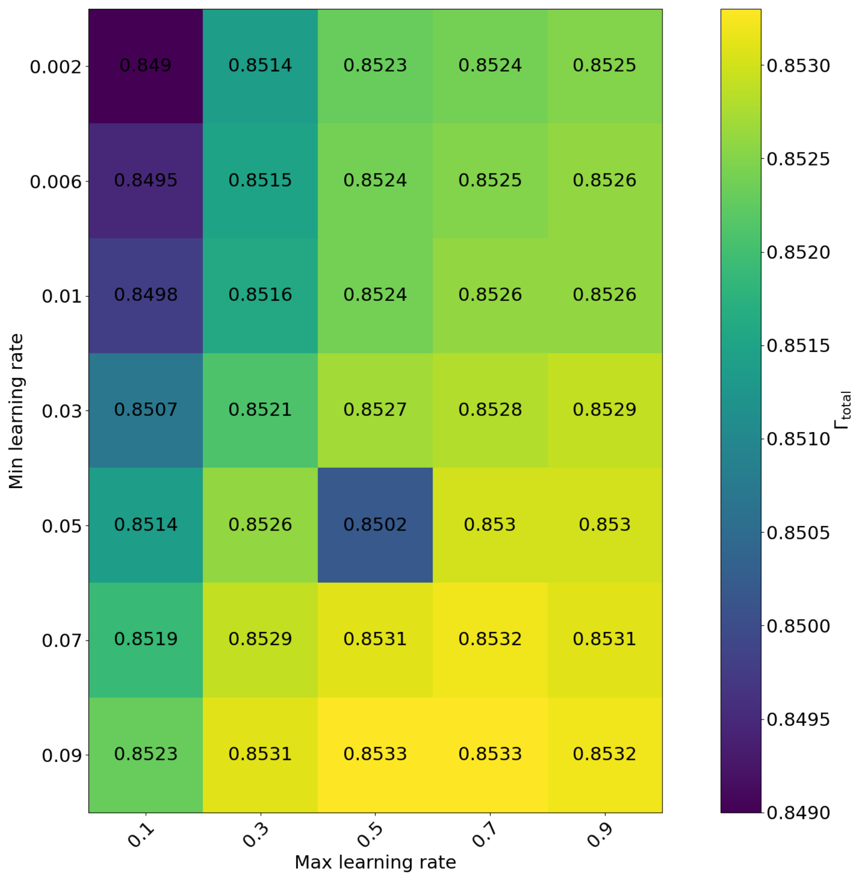


Figure 7.14: The dependence between the maximum and the minimum learning rate of FNNs trained with cyclic learning rate decay

The heatmap in Figure 7.14 shows the results obtained for the different combinations of minimum and maximum learning rate where the quoted Γ_{total} values are the average AUC over all stepsizes for both the fixed and the decaying maximum learning rate training. Varying the minimum learning rate improves the performance of the Neural Networks more than varying the maximum learning rate. Therefore, no second minimum that results in a better performance was found. The drop in performance obtained for models trained with a maximum learning rate of 0.1 is most likely not an indication for a local minimum but rather due to the fact that the training did not converge during 120 Epochs. For minimum learning smaller than 0.3, the same problem occurred. Similar to polynomial learning rate decay Neural Networks trained with cyclic learning rate decay need more Epochs to coverage compared to Neural Networks trained with a fixed learning rate.
The comparison between cyclic learning rate decay with fixed maximum learning rate and with additional decay of the maximum learning rate is shown in Figure 7.15 where only the 5 best-performing Neural Networks were considered for each learning rate. No significant difference between the two learning schedules (red and black squares) is observable.

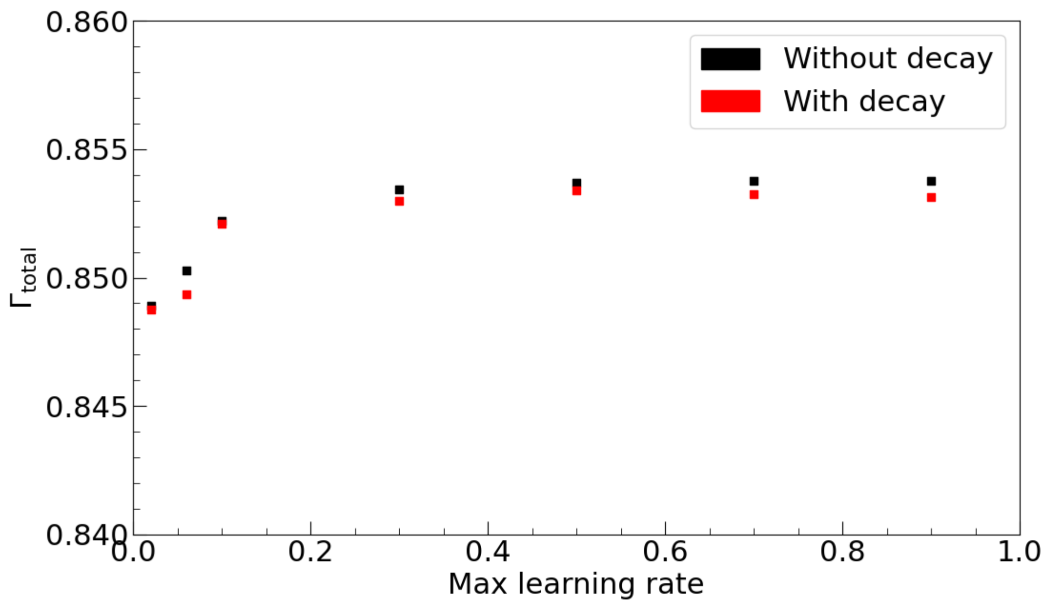


Figure 7.15: The observed AUCs for different maximum learning rate using cyclic decay with and without additional decay.

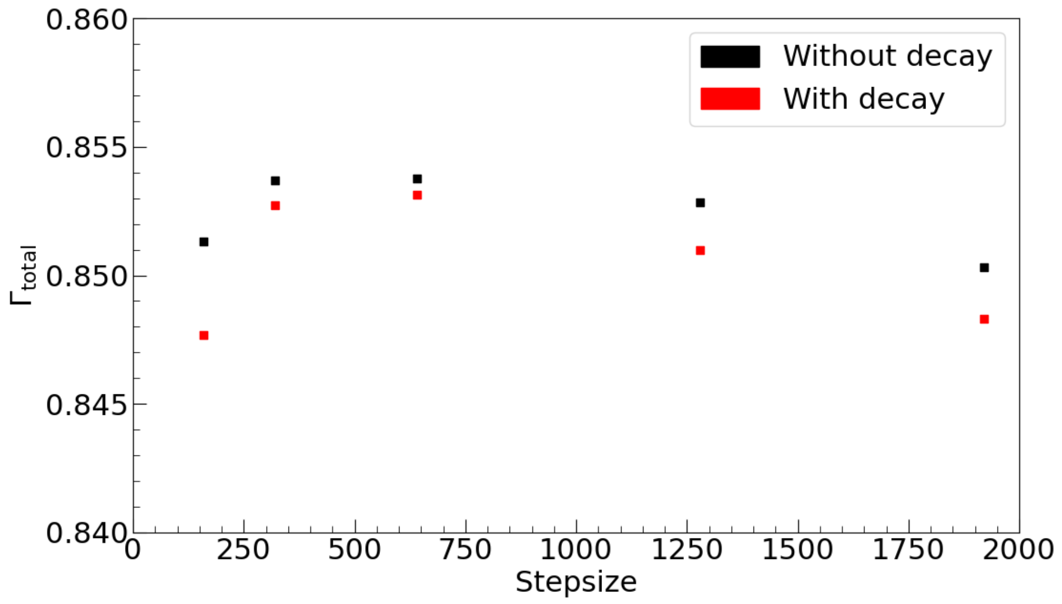


Figure 7.16: The dependence of the AUC on the stepsize chosen for the cyclic learning rate decay.

Figure 7.16 shows obtained AUCs for the diverse stepsizes and different types of learning schedules. Stepsizes in the recommended region between (160-640) result in the best AUCs while stepsizes below or above tend to decrease performance. The difference in performance between Neural Networks trained with the two distinctive learning rate schedules is small for stepsizes in the recommended region. Outside of the region, training without additional decay of the maximum learning rate works better.
The Neural Network with the best performance uses a constant maximum learning rate of 0.9, a minimum learning rate of 0.03, and a stepsize of 640. The obtained Γ_{total} of 0.854 with an overtraining of 1.7‰.

Regularization

The study of regularization aims to increase the AUC obtained on the testing dataset by decreasing the overtraining. The overtraining of the obtained models with under 2% is already below the point of exceptions (5 %), and thus it is unlikely to obtain a higher AUC.
The two methods applied are the dropout govern by the probability p for a neuron to be ignored during training and the Ridge regression govern by α the scaling parameter of the additional Loss function term. The investigated values for α reach from 0.001 (small regularization) to 0.1 (strong regularization) while the probability of Dropout is either 20%, 40%, or 60%. The Dropout is only applied to the neurons in the hidden layer, giving the Neural Network the possibility of recovering performance during the training. To investigate the correlation between the learning rate and the different regularization methods, 5 different fixed learning rates between 0.0005 and 0.01 are investigated. The selected optimization algorithm is Adam since this optimization algorithm was used for the best-performing Neural Network in the fixed learning rate study.

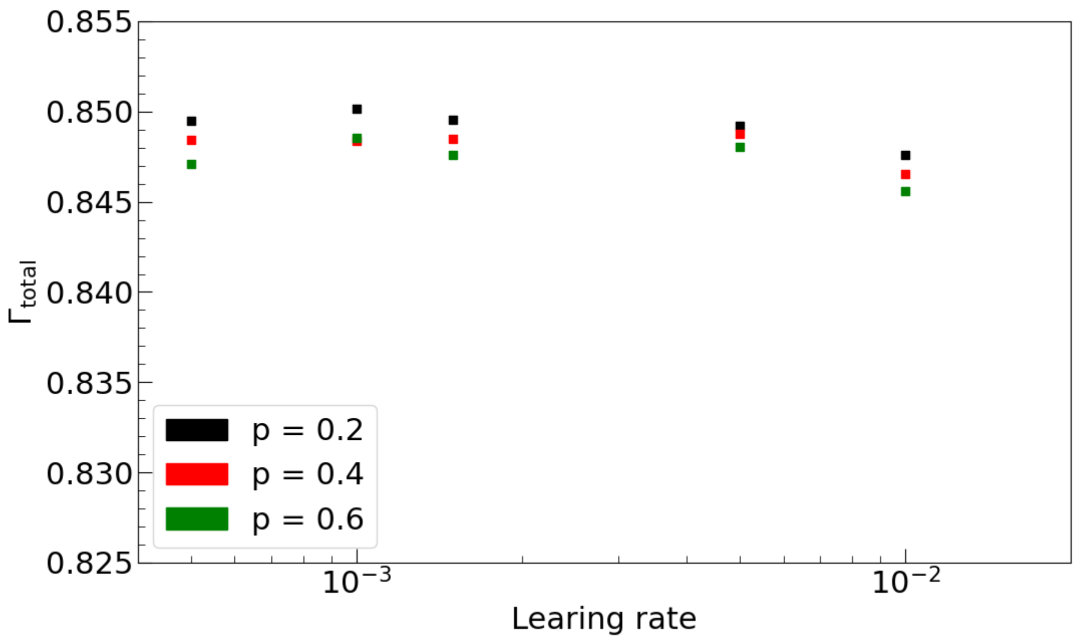
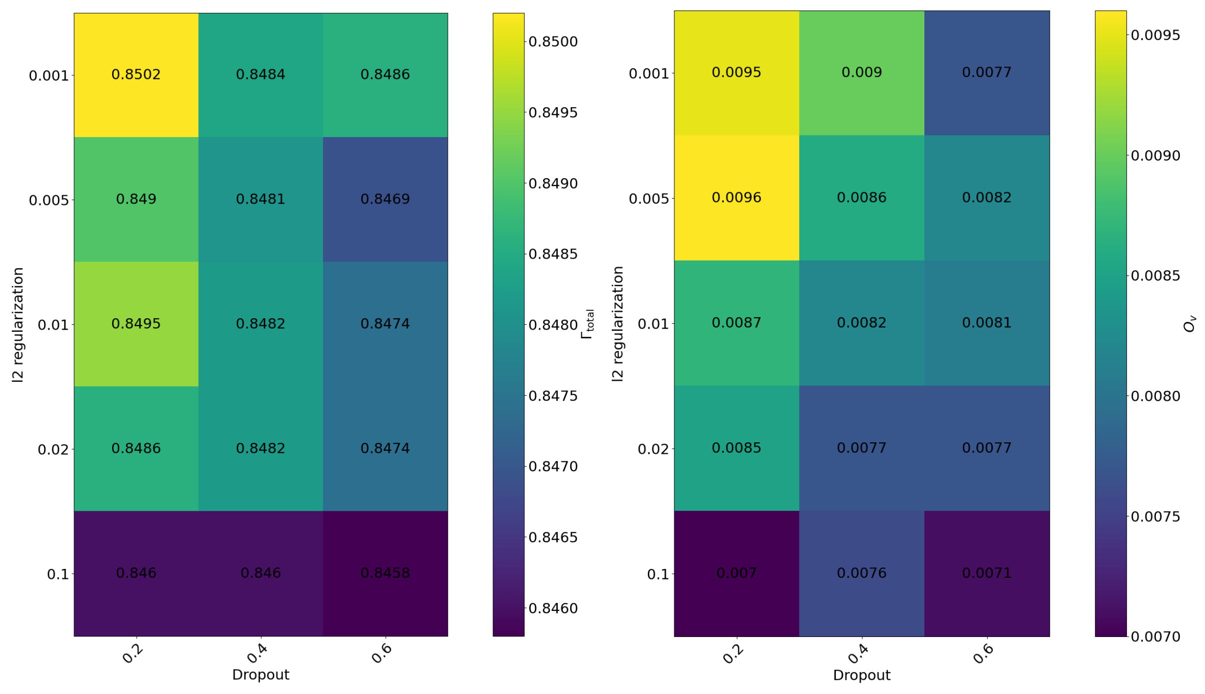


Figure 7.17: Obtained AUCs for different learning rates and distinct choices of p the Dropout probability.

Figure 7.17 shows the obtained AUCs for the different learning rates and Dropout values, where the α was fixed to 0.001. The peak performance has decreased by 0.003 compared to the results obtained without any regularization (cf. Figure 7.7). Neural Networks with a low probability of Dropout outperform those with a higher probability of Dropout as is to be expected for a regularization method. This behavior is also apparent in Figure 7.18, where the learning rate is fixed to 0.001. The performances decrease continuously from the top left corner, where only little regularization is applied to the bottom right corner, where a strong regularization is applied. The overtraining of the trained Neural Networks, shown in Figure **??**, decreases with the amount of regularization applied as intended.
In the investigated parameter range, a strong Ridge regression forcing the weight to be close to zero is a stronger restriction than dropping out neurons in the first hidden layer with a high probability.

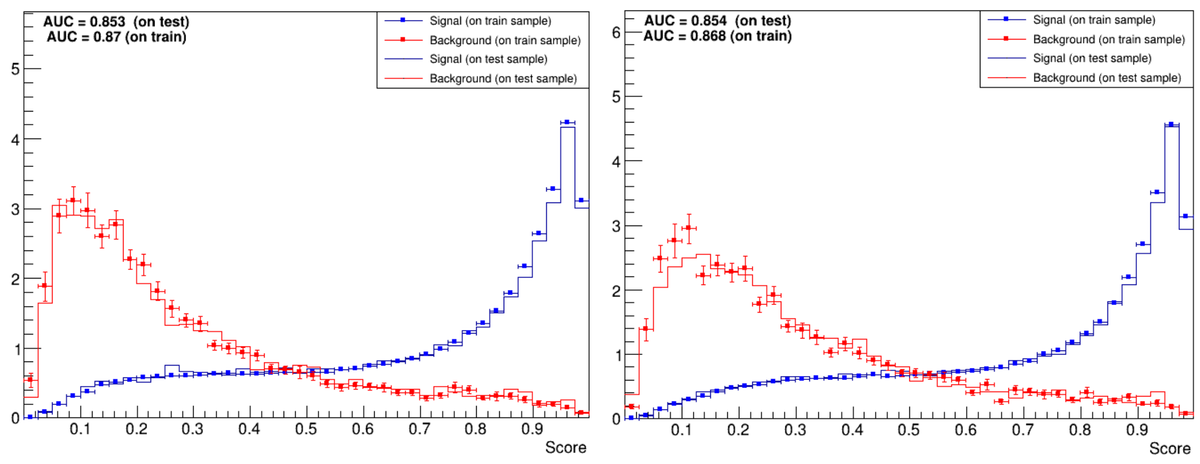


(a)

(b)

Figure 7.18: The dependence of the AUC and the overtraining for different Dropout probabilities and l2 (Ridge) regularization.

Achived Seperation

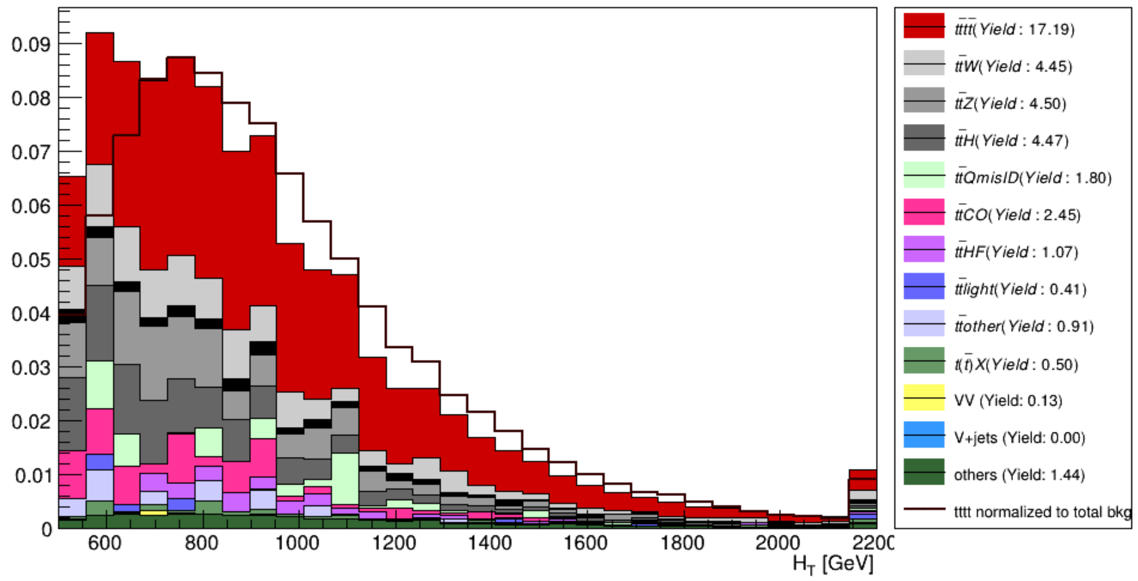


(a)

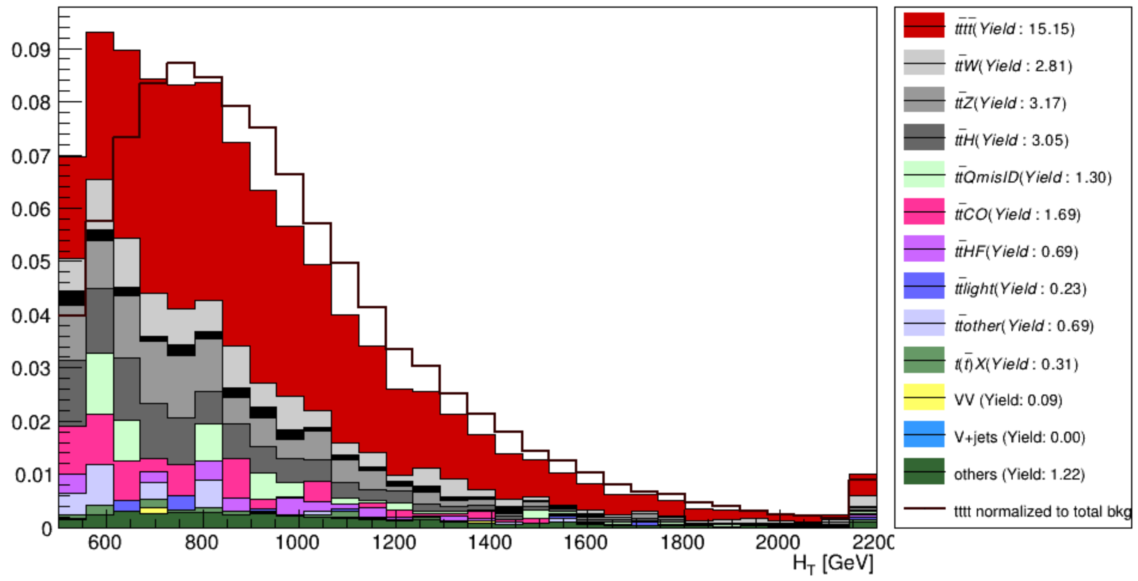
(b)

Figure 7.19: Observed Neural Network scores for the Neural Network trained with the background dataset containing even event numbers (a) and on the background dataset constraining odd event numbers (b).

The Neural Network obtained in the last sub-section is the best-performing model obtained for the $t\bar{t}t\bar{t}$ application. The score for the training on the dataset containing the background events with even event numbers is shown in Figure 7.19(a), whereas the score for the training on the dataset containing the background events with odd event numbers is shown in Figure 7.19(b). Both classifications assign most signal events to scores above 0.7, and background events scores below 0.5. This indicates that the Neural Network is better at identifying signal events than at identifying background events, which is expected since the background dataset contains many different processes. The distance in each bin between the dots and the solid lines is related to the overtraining of the model. Since the Neural Network used has a tiny overtraining, most dots and lines are very close.



(a)



(b)

Figure 7.20: The H_T distributions after applying cut to the FNN scores obtained for the training on the background dataset containing even event numbers (a) and the background dataset constraining odd event numbers (b).

To investigate which backgrounds are particularly hard to separate from the $t\bar{t}t\bar{t}$ a cut on the scores is optimized based on the signal efficiency. For both trainings is the optimal cut at 0.7 which increases the signal efficiency from 1.8 in the signal region to 3.0. The H_T distributions for the two cases are shown in Figure 7.20. The dominant background after allying the cut are still $t\bar{t}W$, $t\bar{t}Z$ and $t\bar{t}H$. The model trained

on the backgrounds with even event numbers is better in rejecting $t\bar{t}$CO while the model trained on the backgrounds with odd event numbers is better in rejecting $t\bar{t}$others. However, the difference is minor and therefore most likely a coincidence. The rejection of all other backgrounds is approximately equally good for both setups when normalized to the signal yield in the investigated region. The signal yield is reduced by a factor of 2 while most other backgrounds are reduced by a factor of ~ 25. The background yield of the dataset "others" mainly containing $t\bar{t}t$ is the background that is hardest to distinguish from $t\bar{t}t\bar{t}$ and is only reduced by a factor of 3. A similar result was obtained by the BDT used in the ATLAS official $t\bar{t}t\bar{t}$ analysis at $\sqrt{s} = 13\,\text{TeV}$ which used the same datasets. In fact, for the high BDT region a significant $t\bar{t}t$ contamination was observed. This can be especially problematic because the $t\bar{t}t\bar{t}$ cross section is only known at leading order $\sigma_{t\bar{t}t} = 1.9\,\text{fb}$ (at $\sqrt{s} = 14\,\text{TeV}$) [???]. If the $\sigma_{t\bar{t}t}$ at NLO is higher than the LO cross section $t\bar{t}t$ could become the dominant background for $t\bar{t}t\bar{t}$.
The bootstrapping results for the selected Neural Network performed on the validation set is shown in Figure 7.21 (on the next page). The seed quoted on the x-axis this the random seed (times 365) used for resampling the validation set. The average of all 50 AUCs obtained is shown in red. The obtained average Γ_{total} is 0.852 ± 0.005, where the error is the standard deviation of the obtained distribution. The error is larger than the performance gained by fine-tuning the learning rate within one order of magnitude. The AUC achieved on the testing set is within the error observed by bootstrapping.

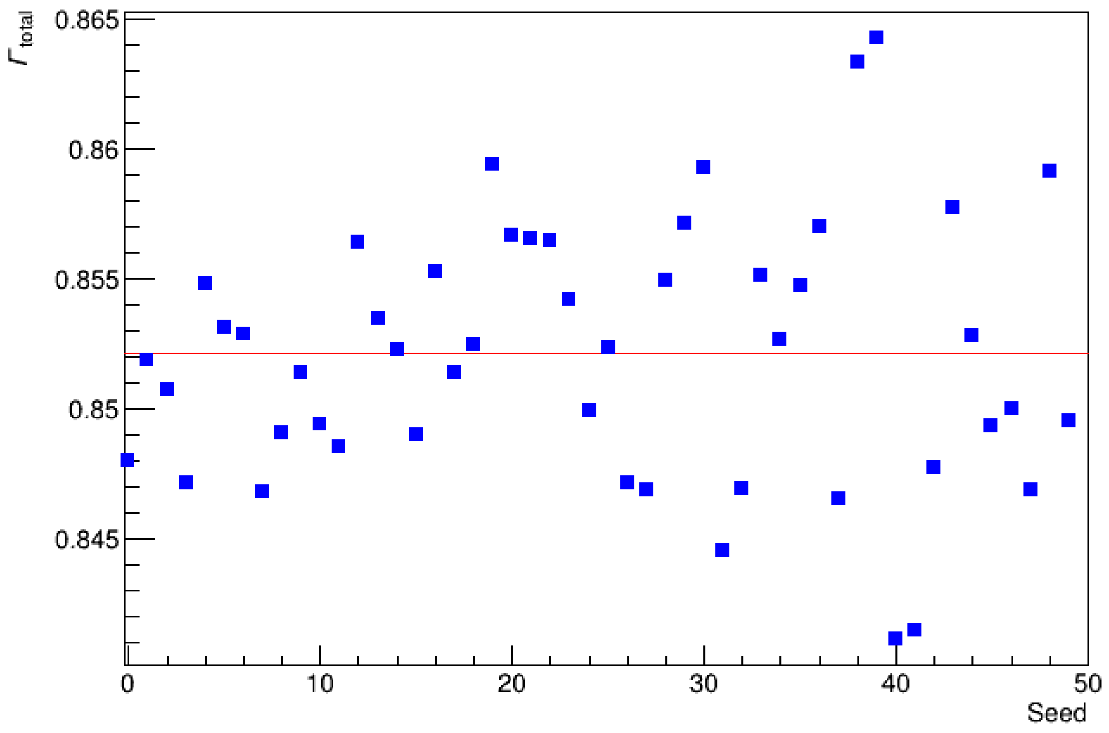


Figure 7.21: Observed AUCs for the bootstrapping on the validation dataset.

7.2 Multi Classification using Feedforward Neural Networks

The Feedforward Neural Networks trained in the last section had one neuron in the output layer and classified signal events against background events. The Neural Networks considered in this section all have several neurons in the output layer to individually treat different background processes. The outputs of the Neural Network are M probabilities, where M is the number of classes considered. Each probability is the likelihood of the given event to belong to class c. The AUCs quoted in the following sub-section are calculated in the same way that AUCs were calculated for the binary classification approach (single against all combined backgrounds).
All Neural Networks use the same setup summarized in Appendix **??**. The number of parameters is increased to approximately 300000 to account for the increased complexity of the problems considered. For convenience, V is redefined excessively for this section to be either a W, a Z, or a Higgs boson.

Fourteen Classes

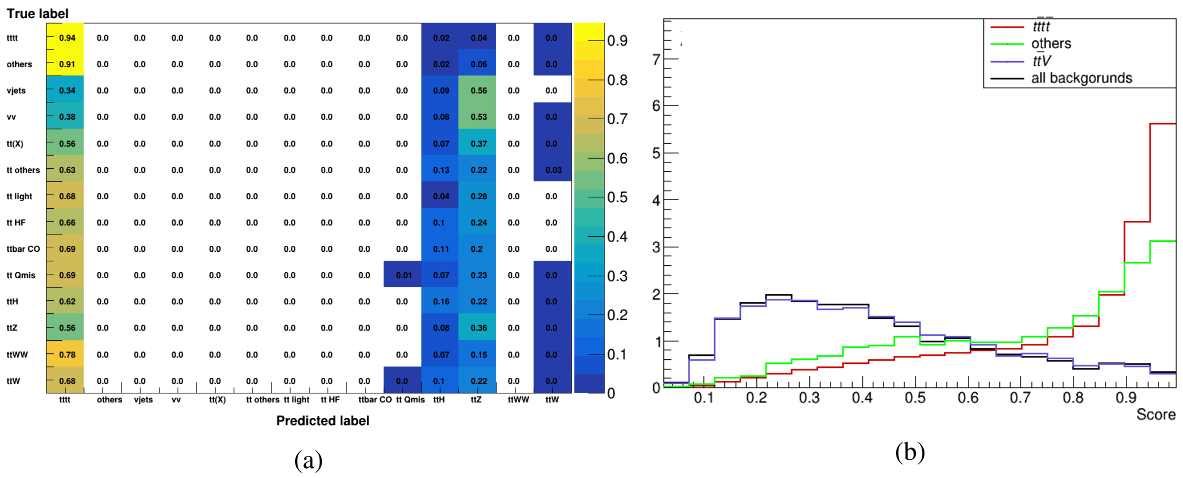


Figure 7.22: The confusion matrix and the Neural Network score for the multi classification with 14 classes.

The first multi classifier approach treats every background process as an individual class. Presenting the Neural Network with the opportunity to optimize the selection for each background. The best obtained Γ_{total} is 0.852 and did not increase the performance compared to the binary classification used previously. Figure 7.22(b) shows the score for the most important backgrounds. The similarity between the $t\bar{t}t\bar{t}$ (red) and the $t\bar{t}$ (blue) observed in Section 7.1 is apparent since the highest fraction of event for both processes end up in highest bins. The dominant backgrounds $t\bar{t}V$ (blue) are most commonly assigned a score below 0.5 but also have considerable contributions up to scores of 0.8. All other backgrounds (black) have a similar distribution to $t\bar{t}V$.
In Figure 7.22(a) the confusion matrix of the obtained classification is shown. The y-axis gives the information about the true event class. The predicated labels on the x-axis are determined by assigning an event to the class for which it has the highest probability. The numbers given in the different cells correspond to fractions of total events in the same row or column. A perfect classification would only have entries of 1 on the diagonal. The trained classifier has almost exclusively entries for three classes, which correspond to the processes with the highest yields. The $t\bar{t}t\bar{t}$ yield was renormalized to the combined total background yield as discussed in section **??**. Therefore, it is beneficial for the Neural Network to focus on the correct identification of $t\bar{t}t\bar{t}$ events rather than reject background events.

Class Weights

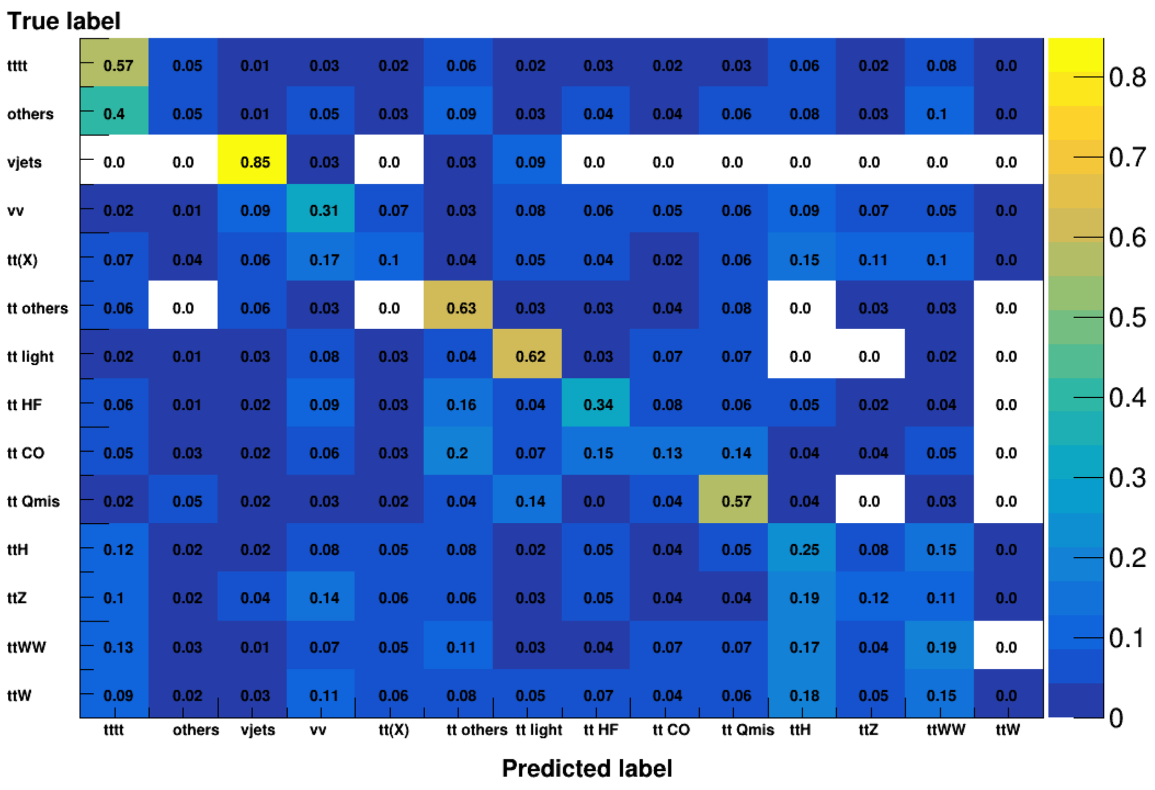


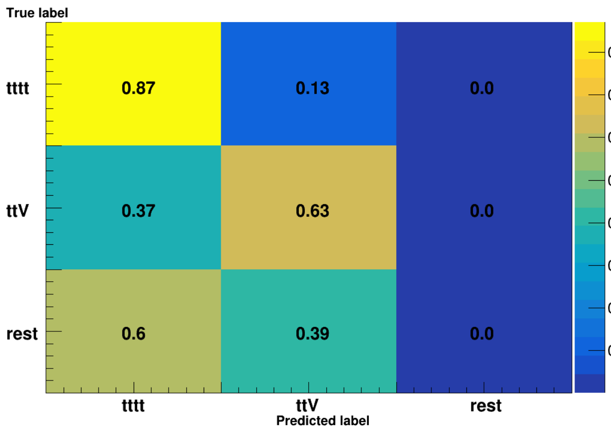
Figure 7.23: The confusion matrix for the multi classification with 14 classes where an additional class weight was applied during the training of the Classifier.

A way to force the neural network to pay more “attention” to minor backgrounds is to scale all processes to the same yield using class weights. This reweighting will decrease the AUC of the obtained classification. However insights about which backgrounds are hardest to distinguish for $t\bar{t}t\bar{t}$ can be gained. Thus providing a possibility to validate the observations about $t\bar{t}$ (others). The class weights w_c are calculated according to

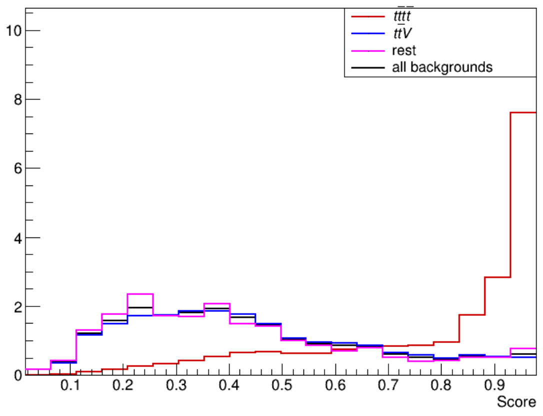
$$w_c = \frac{N_{\text{total}}}{M \cdot N_c} \tag{7.2}$$

where N_{total} is sum of all yields and N_c is the yield of class c.
The confusion matrix obtained by applying class weights is shown in Figure 7.23. As intended, most classes have their biggest fraction of events on the diagonal. The separation between different classes is limited by the choice of the input variables that were selected to discriminate signal from all backgrounds and not to discriminate between different backgrounds individually. In agreement with the previous studies, the dataset “others” is the background with the highest fraction of events predicated to signal events (0.4). The other most signal like backgrounds for this classification are $t\bar{t}H$ and $t\bar{t}WW$.

Three Classes



(a)



(b)

Figure 7.24: The confusion matrix and the Neural Network score for the multi classification with 3 classes.

In this approach, only three different classes are considered, namely a signal class, a class containing the most dominant backgrounds $t\bar{t}V$, and a class for all other backgrounds called "rest". No additional class reweighting is applied. This study aims to improve the AUC compared to the result obtained in the binary approach. The additional class of $t\bar{t}V$ is meant to help further reduce the dominant background. The results of this approach is shown in Figure 7.24. The events belonging to the "rest" class are identified as a signal or $t\bar{t}V$ event. This is because this class has a low combined yield of ~ 86 whereas $t\bar{t}V$ has a yield of 151 and $t\bar{t}t\bar{t}$ a yield of 237, during training. The identification of $t\bar{t}V$ has significantly improved compared to the previous multi classifier approaches. On the other hand, the identification of $t\bar{t}t\bar{t}$ has decreased from 0.94 to 0.87.
The observed AUC for the classification with three classes is 0.854. Compared to the binary classification with a fixed learning rate, the gained improvement is negligible.

7.3 Signal Classification using Recurrent Neural Networks

Compared to Feedforward Neural Networks, Recurrent Neural Networks are more complicated to train and more computationally expensive. A single neuron of the used LSTM structure has 5 activation functions. They all have specific functionalities that control the different cell states.
The strategy chosen to study RNNs is restricted to the most impactful parameters, namely the number of parameters, the number of hidden layers, and the learning rate. The optimization algorithm selected is SGD and will be used in combination with polynomial learning rate decay to ensure that the training convergence if a minimum in the parameter space is found. The maximum number of Epochs is restricted to 80, while the AUC is computed at every Epoch. The mean and standard deviation needed to perform the input feature transformation is obtained based on all objects in the considered feature. To be more concrete, to transform the leading electron's pseudorapidity to a normal distribution, the *eta* of all electrons is considered during the calculation of mean and standard deviation. Only the 7 highest objects for each particle type and event are considered to avoid using variables with significant mismodeling. Since several of the $7 \cdot 13 = 84$ input features contain many zeros (due to zero padding), Lasso regression must be applied. The scaling constant α_1 is fixed to 0.002.

Architecture

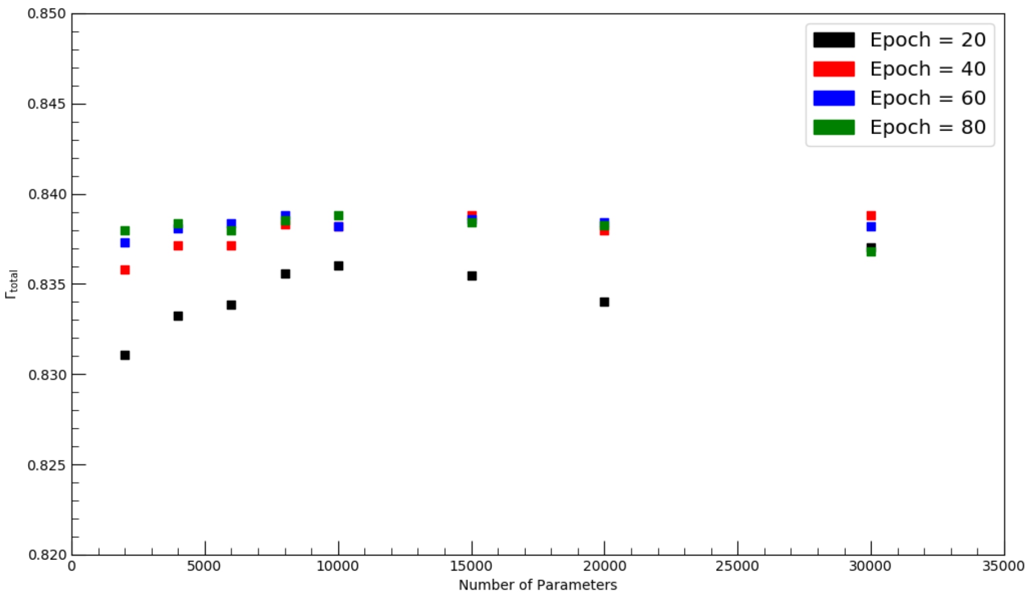


Figure 7.25: The average overtraining for different RNNs configurations. The training is stopped after 20, 40, 60 and after 80 Epochs. The average includes the overtrainings of the 4 best-performing Neural Networks of different depth.

As for the FNNs, the number of parameters and the number of layers is investigated first. Based on the knowledge gained in the first architecture study, the range for the number of parameters considered is between 2000 and 30000. The number of hidden layers tested is restricted to a maximum of 4 hidden layers.
Figure 7.25 shows the AUCs of Neural Networks with varying number of trainable parameters obtained at 20 (black), 40 (red), 60 (blue), and 80 (green) Epochs. Every quoted Γ_{total} is average over the 4 different hidden layer configurations considered. The performance for 40, 60, and 80 Epochs is similar for the most studied sizes, while the training at 20 Epochs has not converged yet. The Neural Networks with

30000 parameters evaluated at Epoch 80 show a decrease in performance, which hints to the fact that these models started to overtrain.
Almost all models have an overtraining below 3%, implying that the applied regularization is sufficient. The overtraining is independent of the number of parameters and hidden layers in the Neural Networks, which indicates that all trainings converged.
The Neural Networks with 1 or 2 hidden layers outperformed Neural Networks with 3 or 4 hidden layers in all considered cases. The peak performance of $\Gamma_{\text{total}} = 0.838$ was reached by a Neural Network with 15000 parameters and two hidden layers.

Learning Rate and Batch size

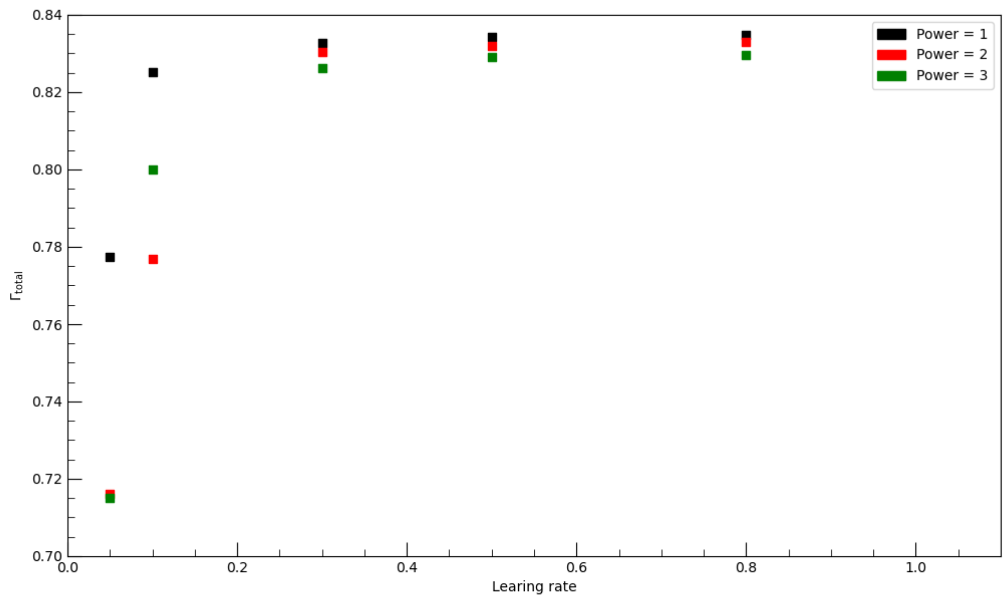


Figure 7.26: The obtained average AUC of RNNs with different learning rates. The average includes the AUCs of the 2 best-performing Neural Networks with different batchsizes.

The Neural Network with the best performance in the previous study is selected to investigate the AUC's dependence on the learning rate and the batch size. The considered batch size reaches from 10^{11} to 10^{14}, which was observed to be a high-performance region for FNNs. Following the same logic, learning rates between 0.05 and 0.8 with decay polynomials of power 1 to 3 are selected.
Figure 7.26 shows the average AUC obtained using the different learning rates and decay polynomials. The average includes the results for two best performing considered batch sizes (2048 and 4096). Similar to the observed results for FNNs, the performance of RNNs is stable within one magnitude around the learning rate that performs the best. Outside of the stable region, it decreases the performance of RNNs rapidly. Similarly has the power of the chosen polynomial a minor influence on the performance inside the stable region, whereas outside of the region, it can have a significant influence. One reason for this could be the more complex and thus more fragile training procedures for RNNs compared to FNNs. On the other hand, the fact that the learning rate decay of order 1 worked the best implies that the Neural Networks might not have converged. The Epoch at which the highest AUC is reached is close to the maximum 80 Epochs but more than 5 Epoch away. Therefore, a combination of both effects is the most likely explanation for the rapid drop.

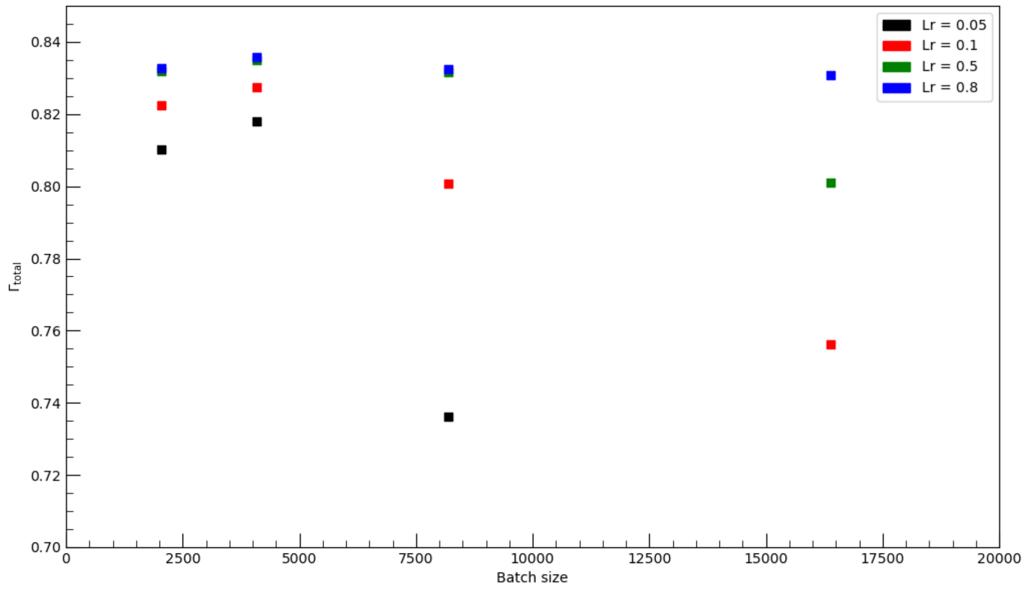


Figure 7.27: The AUC as a function of the batch size and the learning rate.

The dependence of the AUC on the batch size is shown in Figure 7.27. A similar dependence as for the FNNs can be observed. For learning rates close to the optimal obtained value, the batch size has a minor influence on the performance of the Neural Networks. For small learning rates, is the dependence larger. The decrease in performance when choosing a large batch size, combined with a small learning rate, is larger than the observed decrease for FNNs. This validates the assumption that the more complex training of RNNs is more sensitive to the choice of hyperparameters.
The best performing RNN has a significantly smaller AUC (0.842) than the best performance observed using FNNs (0.854). Therefore, the initial idea to present the Neural Network with the possibility of using the full information available did not improve Γ_{total}. The RNN failed in combining the input features in a way that discriminates the signal from the background more than the features used for FNNs, which are constructed using physical knowledge.
The overtraining of all considered Neural Networks is below 3%. Plots showing the dependence of overtraining can be found in Appendix **??**.

Achieved Separation

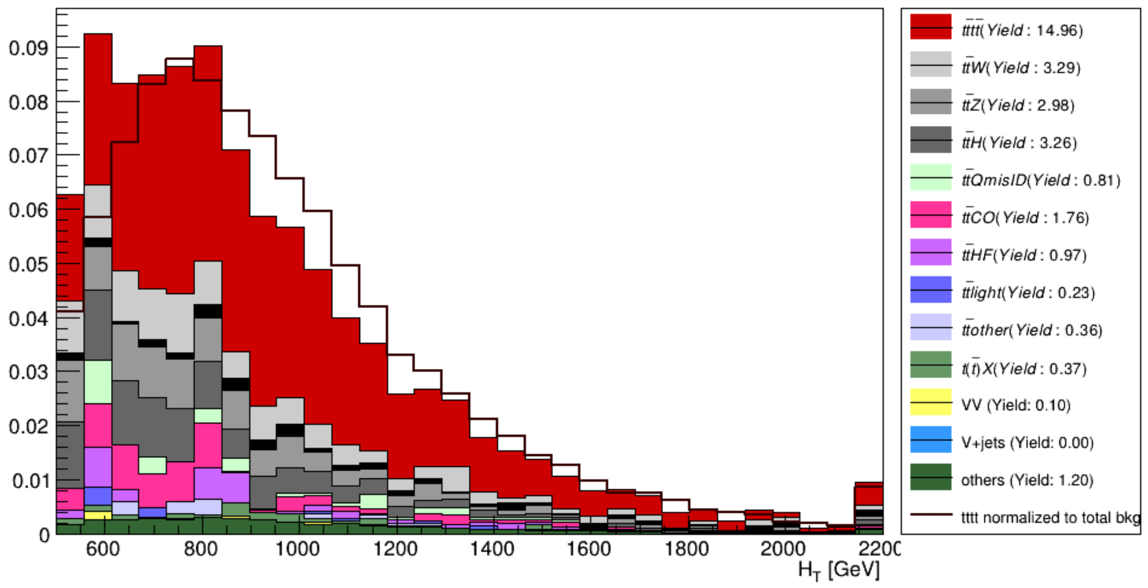


Figure 7.28: The H_T distribution after applying a cut to the RNN score obtained for the training on the background dataset containing even event numbers.

Proceeding in the same way that was chosen to analyze the results for the FNNs, a cut on the Neural Network score is optimized, which is used to selected likely to be signal events, the resulting H_T distribution for the training on the backgrounds that have even event number is shown in Figure 7.28. The H_T for the second training looks similar and can be found in Appendix **??** together with the score distributions.
The optimized cut is 0.85 and improves the signal efficiency to 2.7 compared to the Wsignal efficiency obtained in the full signal region (1.8). The dominant backgrounds are $t\bar{t}W$, $t\bar{t}Z$ and $t\bar{t}H$. The processes in the dataset others show the smallest reduction in yield besides the $t\bar{t}t\bar{t}$ signal. In compression to the H_T distribution (Figure 7.29) for FNNs non of the backgrounds can be identified to be the problematic background that decreases the performance of the RNN.
The AUCs obtained by bootstrapping are shown in Figure 7.29. The mean indicated by the red line is $\Gamma = 0.838$ with an error of 0.006. The combined AUC observed on the test datasets (0.842) is within the obtained error. Therefore, a bias introduced through hyperparameter tuning can be excluded.

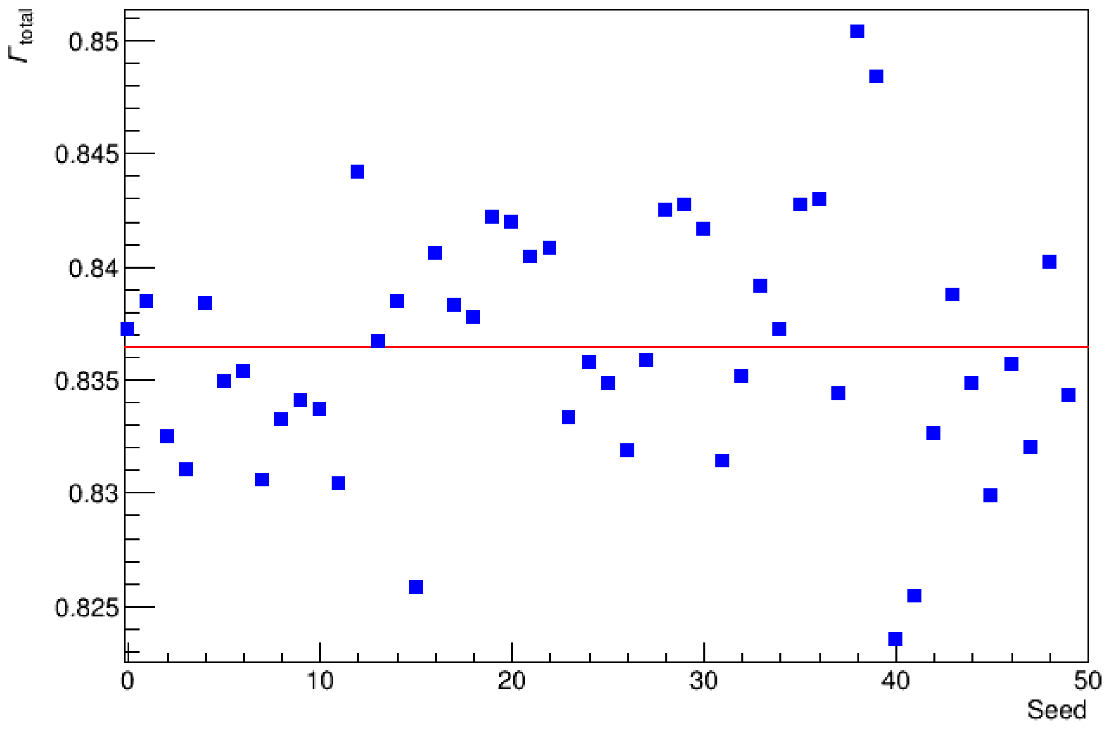


Figure 7.29: Observed AUCs for the bootstrapping on the validation dataset.

7.4 Performance Evaluation of CPUs and GPUs

All Neural Networks used in the previous studies were trained on central processing units (CPUs). However, in deep learning applications, it is prevalent to use graphics processing units (GPUs). Most operations performed during a Neural Network training are simple matrix multiplication of weight matrices and feature matrices. CPUs are designed to perform a few complicated tasks and struggle with multiple threads at the same time. On the other hand, GPUs are designed to perform many simple tasks at the same time, making them the optimal choice for deep learning applications.
The following study investigates the training time of Feedforward Neural Networks and Recurrent Neural Networks using CPUs and GPUs. The same datasets that were used in the previous Neural Network studies are utilized in this study. The for training used devices are intel Xeon e5-2620 v3 CPUs and Geforce gtx 1080 GPUs.
Neural Networks with trainable parameters between 2000 and 70000 parameters are considered. The optimization algorithm used is SGD with batch sizes between 2^{10} and 2^{16}. All other parameters are fixed to the best-performing Neural Networks of each type, as summarized in Appendix **??**. The AUC evaluation at each Epoch is turned off since the needed calculations can take several seconds. The quoted train time per Epoch is the average over 120 Epochs.

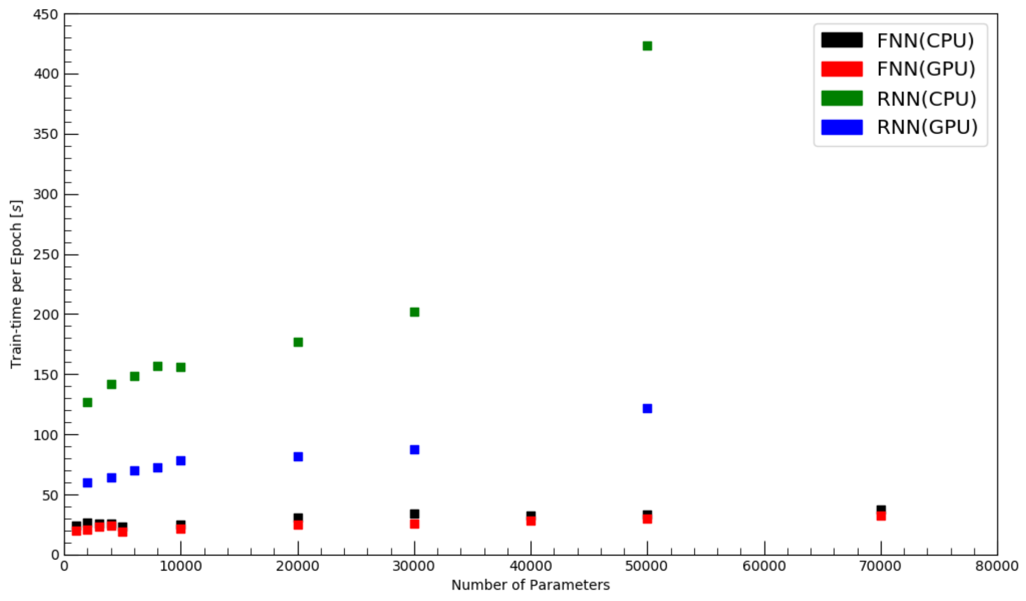


Figure 7.30: Training times per Epoch for FNNs and RNNs of different sizes trained on CPUs and GPUs.

Figure 7.30 shows training time for the Neural Networks of different sizes where the batch size was fixed to 8192. The RNNs train significantly longer than the FNNs due to their more complicated neuron structure and training algorithm. GPUs outperform CPUs for all considered Neural Networks. The gain in performance is especially high for Neural Networks with many tunable parameters. For the largest RNNs decreases the training time per Epoch by a factor of 4 when using GPUs compared to CPUs.
The results for the different batch sizes are shown in Figure 7.31, where a Neural Network with 50000 parameters was used. Again GPUs outperform CPUs over the full parameter space selected. The decrease in training time is small for small batch size and increase with large batch sizes. The reason for this is that only the calculations within one iteration are performed simultaneously. Since the number of performed calculations scales with the batch size also the performance gain does. The saturation of performance gain for Neural Networks with batch size larger than $\sim$ 16000 most likely has two sources. The first one

being that the maximum number of calculations at the same time is reached for the GPUs. The second one being that the observed standard deviation of the training times per Epoch was significantly higher for the batches of 2^{15} and 2^{16} indicating that the cluster note started to lag, potentially influencing the obtained result.
Conclusively it can be stated that even for the small dataset and Neural Networks considered the usage of GPU is highly advisable.

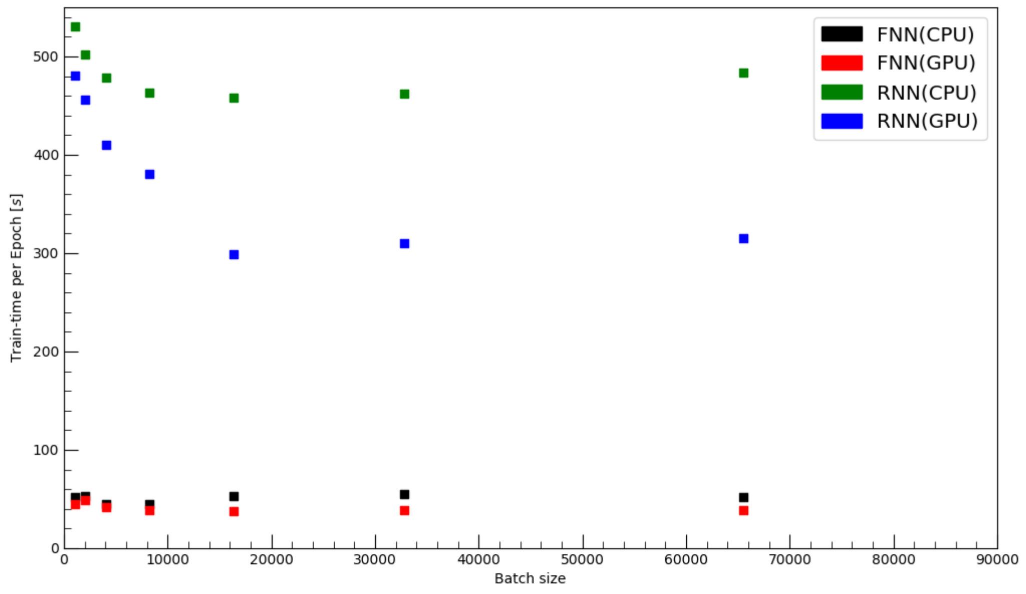


Figure 7.31: Training times per Epoch for FNNs and RNNs trained with different batch sizes and on CPUs and GPUs.

7.5 Two Hadronic Top Reconstruction for three Top Quark Production

The $t\bar{r}r$ contamination in the high score regions of the FNN, the RNN, and the BDT used in the official $t\bar{t}t\bar{t}$ motivate the search for a feature that discriminates the two processes well. Since both processes are very similar in their kinematics, the chosen approach aims at the final state particle composition. In the same sign two lepton channel, decay two of the four W bosons originating from the four tops leptonically while the other two decay hadronically. On the other hand, a $t\bar{t}t$ event produced without any associated particles has one hadronically decaying W boson in the final state. As discussed in Section **??** such a production is not possible at leading order. The clear separation between $t\bar{t}t\bar{t}$ and $t\bar{t}t$ is the for distorted by the particles produced in association with the three top quarks. Incorrect reconstruction of jets or leptons and charge misidentification can further perturb a possible reconstruction.
The datasets used for the reconstruction were introduced in **??**. The signal region is redefinition to match the conditions of the two hadronic top reconstruction. Events with 3 or more leptons are rejected; only events with two leptons of the same charge are considered. The number of required jets is raised from at least 6 jets to at least 8 jets. All other signal region conditions stay the same.

Reconstruction principle

The reconstruction presented in the following tries to identify $t\bar{t}t$ events by finial states that have one hadronic top and $t\bar{t}t\bar{t}$ events by finial states that have two hadronic top. Priority is given to the reconstruction of two hadronic tops, which are reconstructed from 6 jet combinations, 3 jets for each top. At least one of the two top candidates has to contain a b-jet. The assignment of the three jets to a W boson and a b-jet depends on the number of b-tagged jets contained within the top. If only one of the jets is b-tagged, this jet is assigned to the b-jet, and the two other jets are said to be the decaying produces of the W boson. If two jets are b-tagged, the two jet combination with the lowest discriminator is selected, but at least one b-tagged jet has to be assigned to the b-jet. If non or all jets are b-tagged, the combination with the lowest discriminator is selected.
The discriminator chosen for the reconstruction is based on χ^2 and contains one term for each of the four reconstructed objects (2 tops quarks and 2 W bosons).

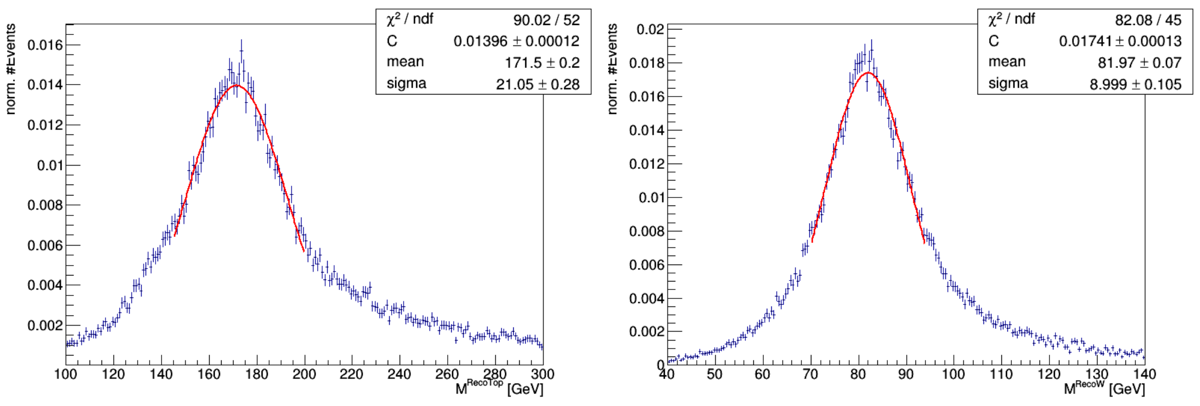
$$\chi^2_{2\text{had}} = \chi^2_{t_1} + \chi^2_{t_2} + \chi^2_{W_1} + \chi^2_{W_2} \tag{7.3}$$

The individual χ^2 terms are defined as

$$\chi^2_X = \frac{(m_{3/2j} - m_X)^2}{\sigma^2_X}; \quad X \in t, W \tag{7.4}$$

where $m_{3/2j}$ is the mass of the 3 or 2 jet combination. The mass and the width of the top quark and the W boson are obtained from the dataset, which will be discussed in the next sub-section.
The reconstruction of two hadronic top quarks is definite to be successful if $\chi^2_{2\text{had}}$ does not exceed a critical value χ^2_{c1}. The exact value of χ^2_{c1} is left as a tunable parameter of the reconstruction. If $\chi^2_{2\text{had}}$ is smaller than the χ^2_{c1}, the reconstruction of one hadronic top quark is carried out. The reconstruction of one hadronic top quark proceeds in the same way as the two hadronic top quark reconstruction using the discriminator $\chi^2_{1\text{had}} = \chi^2_{t_1} + \chi^2_{W_1}$. If $\chi^2_{1\text{had}}$ surpasses a second critical value χ^2_{c2}, the event is considered to have no hadronic top.

Calibration

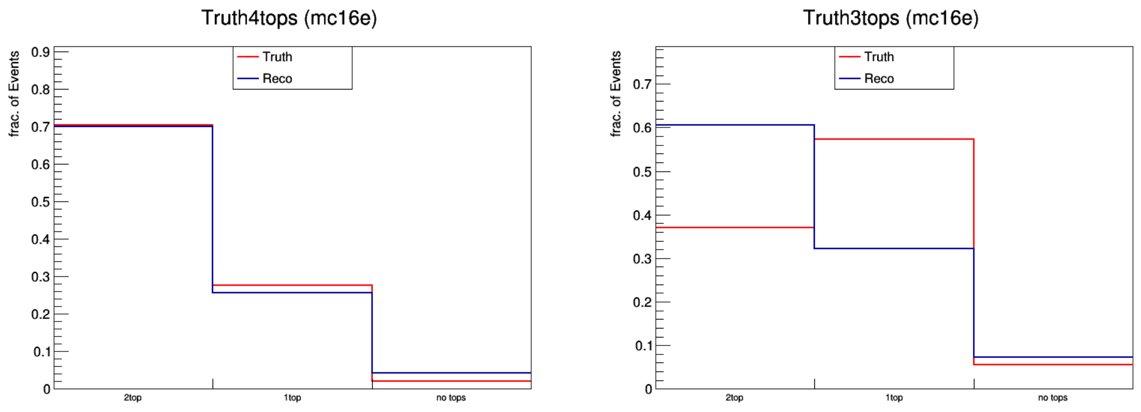


(a) (b)

Figure 7.32: Mass distributions of the truth matched reconstruction top quarks and W bosons.

The calibration of the top quark and W boson mass and width is performed by matching the reconstruction candidate to the *truth particles*. The kinematic and final state particles of the truth particles are known from the truth information in the used datasets. Only hadronically decaying truth tops or W bosons with final state particles that can be measured by the ATLAS detector are considered. The matching is performed based on the δR between the reconstruction candidates and the truth tops in the same event. The closest reconstruction candidate is assigned to the truth top. The matching of W bosons is performed in the same way and independently of the matching of tops.
Only matched pairs that have a $\delta R(\text{reco}, \text{truth})$ smaller than the critical value are considered for the final mass fit. Figure 7.32 shows the result of the calibration with a critical value of $\delta R(\text{reco}, \text{truth}) < 0.4$. The mass and the width of the particles is observed as the mean and standard deviation of the gaussian fit. The obtained masses and width are $m_t = 172$, $m_W = 82$, $\sigma_t = 21$, and $\sigma = 9$.
To study the dependence of the observed masses and widths, different critical values between 0.1 and 0.8 were considered. The mean of the distribution is approximately independent of the critical value. The standard deviation varied between 22.6 and 16.9 for the top quark and between 7.7 and 9.6 for the W boson. The critical value of $\delta R(\text{reco}, \text{truth}) < 0.4$ was selected as the point at which both weight distributions showed a approximately symmetric peak.
In order to have a comparison to the results discussed in the next sub-section, a *best-possible reconstruction* is defined. This reconstruction also uses truth information and, therefore, can not be used for measured data. All pre-requirement concerning the number of b-jets in the reconstructed particles are the same as for the χ^2 based reconstruction. The best-possible reconstruction first selects all top candidates with a mass within 3σ around m_t and contain a W candidate within 3σ around m_W. If there are more truth tops than reconstruction tops, the mass windows are broadened to 4σ. If the number of truth tops still excesses the number of reconstruction tops, all candidates kept. After this pre-selection is finished, the matching based on $\delta R(\text{reco}, \text{truth})$ as described previously is performed. The candidates closest to the truth tops in the same event are said to be the reconstructed tops.

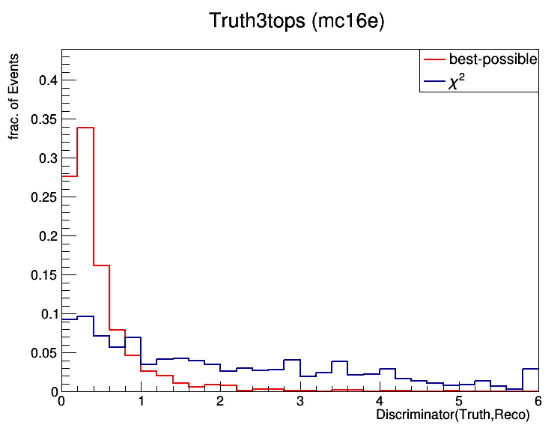
Reconstruction Results



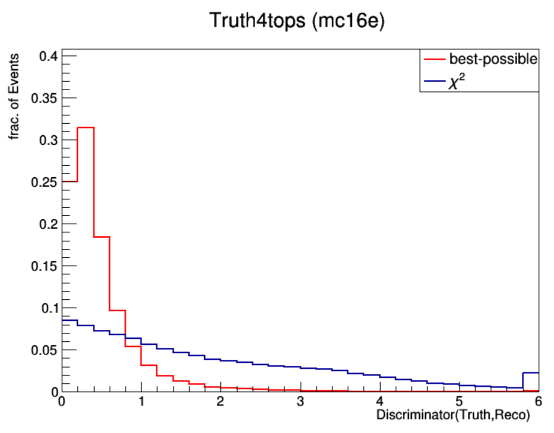
(a) $t\bar{t}t\bar{t}$ events (b) $t\bar{t}$ events

Figure 7.33: Observed reconstruction where the critical values were tuned such that the fraction of events in all three channels fit the true $t\bar{t}t\bar{t}$ distribution.

Figure 7.33 shows the number of identified hadronic top quarks for the $t\bar{t}t\bar{t}$ and the $t\bar{t}t$ processes for the χ^2 reconstruction (blue) and the true distribution (red). The critical values $\chi^2_{c1} = 5$ and $\chi^2_{c1} = 10$ are tuned such that the obtained results match the true $t\bar{t}t\bar{t}$ distribution. This results in a poor performance on the $t\bar{t}t$ dataset. In fact only 64% of the events $t\bar{t}t\bar{t}$ and 43% of the $t\bar{t}t$ events are categorized correctly. Most $t\bar{t}t$ one hadronic top events are reconstructed as two hadronic events and most non hadronic top events are reconstructed as one hadronic top events. The most likely explanation for this behavior is that the dominant production channel of $t\bar{t}t$ is associated production with a W boson. In fact, the original branching fraction of 0.72 [] is increased due to the signal region cuts to 0.99. In the case where the additional W boson decays hadronically a hadronic top decay can be easy faked. The additional W boson is also the reason why on truth level not all $t\bar{t}t$ events in the signal region have one hadronic top. The signal region requirement of two same sign leptons can be achieved for a two hadronic $t\bar{t}t$ event when the additional W boson decays leptonically.
The poor performance (only 64% of all events a categorized correctly) for the $t\bar{t}t\bar{t}$ is not apparent. The comparison of δR(reco, truth) between the χ^2 based reconstruction and the best-possible reconstruction, shown in Figure 7.34, shows that in most cases the incorrect jet combinations were selected. Furthermore, the kinematic distribution of the reconstructed objects differ in some cases from the truth and best-possible reconstruction distributions, shown in Figure 7.35.
A possible improvement of the χ^2 based reconstruction can be the introduction of an additional term proportional to the MV2c10 scores for the jets assumed to be the b-jet of the top decay. Another way to improve the performance is to reconstruct the leptonically decaying W bosons additionally. The difficulty in the reconstruction of multiple leptonically decaying W bosons is the allocation of E_T^{mis} to multiple neutrinos, which was the reason why this approach was not considered for this study.

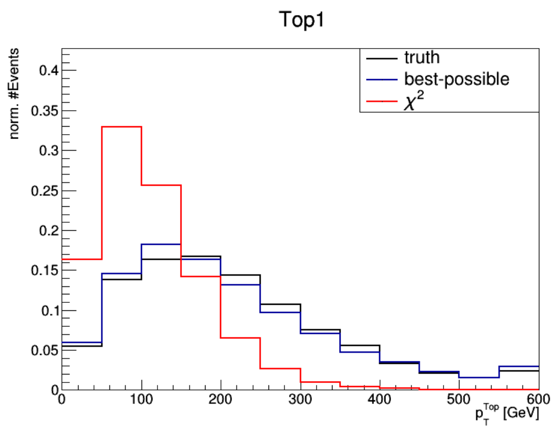


(a)

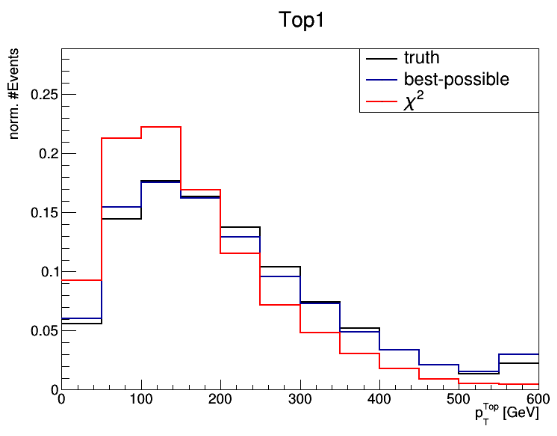


(b)

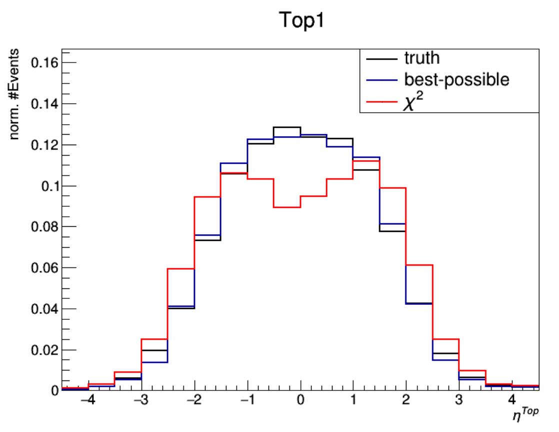
Figure 7.34: δR between the reconstructed top quarks and the truth top quarks for the $t\bar{t}t\bar{t}$ process (a) and the $t\bar{t}$ process.



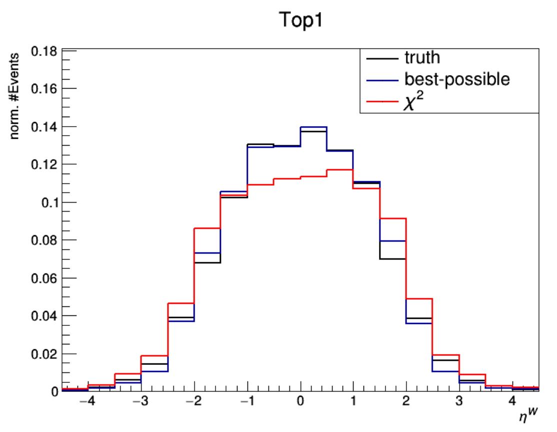
(a) Top p_T for the one hadronic top category



(b) Top p_T for the two hadronic top category



(c) Top η for the one hadronic top category



(d) W boson η for the one hadronic top category

Figure 7.35: Kinematic distributions for the truth and reconstructed top quarks and W bosons.