Machine learning for pTW unfolding

Kirill Perminov, Moscow Institute of Physics and Technology, Russian Federation

Supervisor: Simone Amoroso

September 4, 2019

Contents

1 Introduction

W boson is an essential particle in the Standard Model. Precise measurement of the mass of the W boson allows to check the consistency of the SM. One of possible ways to measure m_W is to reconstruct the spectrum of the transverse momentum of the lepton in the following reaction: $W \to l\nu$. As long as the W boson can also have the non-zero transverse momentum, the estimation for its spectrum is also required. As a result, the reconstruction of the p_T^W spectrum is essential in the m_W measurement. Nevertheless, the distribution of reconstructed values differs from the distribution of true values (fig. 1). So, we come up with the idea to build some estimator that gives the probability density function of the p_T^W for each event, not just the estimated value of the p_T^W.

Unfolding is the process that allows to reconstruct the distribution of a specific physical quantity by using the observed distribution and knowledge about the measuring device. For each event, it is possible to determine the probability density function of the measured value of p_T^W provided that we know the true value of p_T^W. Indeed, Monte Carlo simulation allows to conduct many experiments with known true values and, consequently, to obtain the transfer matrix (in the case of binned variables) of the detector. If there is much data, all we need is to inverse the transfer matrix. However, the result of ordinary inversion is usually bad, so the process should be regularized somehow.

In this paper, machine learning methods are applied. The neural network is used as a classifier (for individual events) that processes measured information and predicts probabilities to fall into a given bin. Machine learning requires some events with known target values (train data). However, if the distribution of target values among the train data is much different from the distribution among real events, then the neural network gives biased results. The iterative unfolding is a solution for this issue.

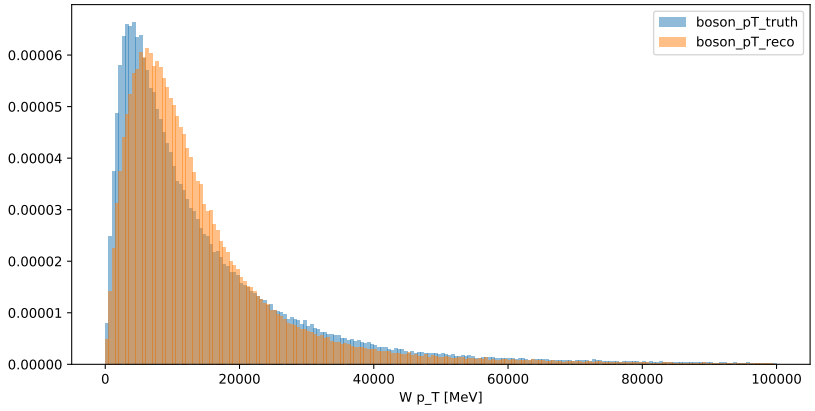

Figure 1: Distributions of true (`boson_pT_truth`) and reconstructed (`boson_pT_reco`) values of the p_T^W for the Monte Carlo simulation

2 Machine learning

2.1 Classification

Classification is one of the problems in machine learning. There are much information that we measure in every event. There is also the output variable (the bin of the true p_T^W) that is known only for some events (train data generated by Monte Carlo simulation). The purpose is to estimate probabilities (in our case) of different output values for the rest of events (test data).

The first step is to create a model. This is a function that contains a set of free parameters. This function takes measured variables for one event as an argument. The output of the function is the sequence of probabilities.

The second step is to fit the free parameters. The train data is essential at this step. Since the output of the function depends on the free parameters, the purpose is to make this output closer to the real output values that we know for the train data. It is important to set a proper loss function that determines how far the predictions are from the real values. In this paper, the categorical cross entropy is used as a loss function:

$$H(\mathbf{p}, \mathbf{q}) = -\sum_{e=1}^{N_{\text{evt}}} \sum_{i=1}^{N_{\text{bin}}} p_i^e \log q_i^e \ ,$$

where:

- N_{evt} is the number of train events
- N_{bin} is the number of possible output values. In our case, it is the number of bins
- p_i^e is the true probability of the i^{th} value for the e^{th} event
- q_i^e is the predicted probability of the i^{th} value for the e^{th} event

The third step is to make predictions for the test data. The free parameters are set by using the train data. So, the quality of the model for the test data is expected to be worse than for the train data. However, the quality for the test data is an important metric of the model. That is why we use Monte Carlo events for the test data as well as for the train data.

2.2 Artificial neural networks

Artificial neural network is a special type of models with the large number of parameters that are organized in an understandable structure (fig. 2). It consists of layers of neurons. Each neuron calculates the linear combination of a free constant and outputs of neurons from the previous layer. After that this linear combination is transformed by a non-linear activation function. The result is the output of the neuron. The first layer consists of the input variables. The last layer gives predictions of the model. All other layers are called hidden. In this paper we use only one hidden layer.

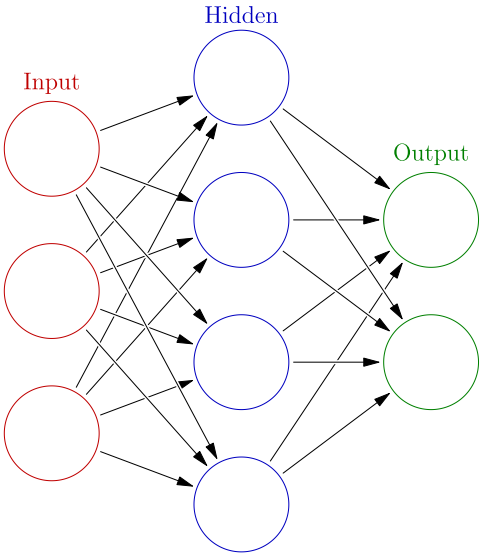

Figure 2: An artificial neural network with one hidden layer [3]

Coefficients in mentioned linear combinations should be fitted while training but the number of layers, number of neurons in each layer and activation functions should be set during the building of the model. The number of neurons in the input layer equals the number of input variables, N_{inp}. The number of neurons in the output layer equals N_{bin}, the number of bins of the true p_T^W. The number of neurons in the hidden layer equal to κN_{bin}^2 where $\kappa = 8$. Such number is inspired by the response matrix in the case of binned input. It should be reminded that in a traditional unfolding with one input variable the general idea is to inverse the response matrix that consists of N_{bin}^2 values. So, the neural network is expected to learn this matrix somehow. ReLU is used as an activation function for the hidden layer:

$$\text{ReLU}(x) = \max(0, x)$$

The output layer must consist of neurons that produce numbers from the interval $(0, 1)$. Moreover, the sum of these numbers must equal 1. Softmax fulfills the requirements:

$$\text{softmax}_i(x_1, ..., x_{N_{\text{bin}}}) = \frac{e^{x_i}}{\sum_{k=1}^{N_{\text{bin}}} e^{x_k}}$$

Gradient descent allows to find approximately optimal values of the parameters of the neural network. The general idea is to pass the train data through the neural network and calculate gradients by using the backpropagation algorithm. One epoch is when all train events are passed through the network. Several dozens of epochs are usually necessary to fit the model.

3 Unfolding

3.1 Predicting the distribution

Predicted probabilities of different bins of the true p_T^W for the test data are used to estimate the p_T^W distribution. Namely, we calculate the sum of probabilities for all events within each bin. As a result, these N_{bin} values can be considered as heights of the histogram for the p_T^W. If the classifier is perfect, such estimation of the p_T^W distribution is expected to be precise.

However, this estimation is biased. If some bin is more probable in the train data, then the classifier overstates predictions of this bin for the test data. In this way, the predicted distribution is affected by the distribution of the train data. The cause of such effect consists in Bayes' theorem. Let x be measured data about some test event. Then the conditional probability density function of the p_T^W can be calculated by using the following formula:

$$p\left(p_T^W|x\right) = \frac{p\left(x|p_T^W\right)\ p\left(p_T^W\right)}{p(x)}$$

The function $p\left(x|p_T^W\right)$ can be estimated due to the train data. Unfortunately, the prior distribution $p(p_T^W)$ can not be estimated correctly. This is why the predictions are biased if the train data and the test data have different p_T^W distributions.

3.2 Iterative algorithm

Despite the fact that the predictions are far from the real values, the predicted distribution is slightly better than the distribution of the train events. So, the predictions can be used to improve the train data. Namely, we set weights for the train events that make the distribution of the train data equal the predicted distribution.

Since the train data is changed, it is reasonable to train the model again. The new predictions are going to be more accurate than the previous ones. Moreover, these new predictions can be used to change the p_T^W distribution of the train data again.

As a result, we come up with the idea of the following iterative algorithm:

- Train the model
- Make predictions
- Change the p_T^W distribution ("prior") of the train data by changing weights of the train events
- Repeat

However, now the predicted p_T^W distribution of the test data has a huge impact on the training, especially for late iterations. The problem is that small uncertainties for the test data lead to big uncertainties for a distribution predicted on an iteration with a large serial number. These uncertainties need to be at least estimated.

3.3 Bootstrap

Bootstrap technique is used to estimate statistical uncertainties of the unfolding. The general idea is to create many replicas of the test data that are slightly different. As a result, the predicted distributions are also different. These differences allow to estimate the standard deviation of the result.

The number of events occurred over a fixed period of time has Poisson distribution:

$$P(k \text{ events}) = e^{-\lambda} \frac{\lambda^k}{k!}$$

Here λ is the mean number of events. It is known that the sum of independent Poisson random variables also has Poisson distribution. Moreover, as always, the mean of the sum is equal to the sum of the means. In this way, we set weights for every test event. These weights have Poisson distribution with $\lambda = 1$. Since the weights are random, replicas differ from each other. Trained models are different starting from the second iteration because the p_T^W distribution of the test data influences the training process for the second and subsequent iterations.

4 Results

4.1 The original initial prior

First of all, it is worth checking that predictions of the model are adequate if the distribution of train events matches the distribution of test events. Indeed, if the model can't be trained even in this case, then it is necessary to change the architecture of the neural network. Otherwise, the model is appropriate.

Let's split randomly Monte Carlo events into three sets: train, validation and test. The train set is used to fit the parameters of the neural network. The test set allows to compare the unfolded (predicted) distribution with the real one. The validation set is used to track two metrics during the training. The first metric is the loss function, namely, categorical cross entropy. The second metric is accuracy:

$$\text{Accuracy} = \frac{\text{The number of events classified correctly}}{\text{The number of all events}}$$

In this way, the validation set and the test set are not used during the training. However, they are useful because predictions for these sets can be compared with the true values.

After random separation, all three sets have approximately the same p_T^W distribution. This is why the first iteration is expected to give the best result and the least uncertainty. Indeed, iterations are like a random walk since the distribution of train events has been already similar to the distribution of test events. Values of the loss function and accuracy for different epochs can be seen on fig. 3. The p_T^W distribution of the test set and the unfolded distribution are compared on fig. 4. Results show that the model is suitable for the further testing with different prior distributions.

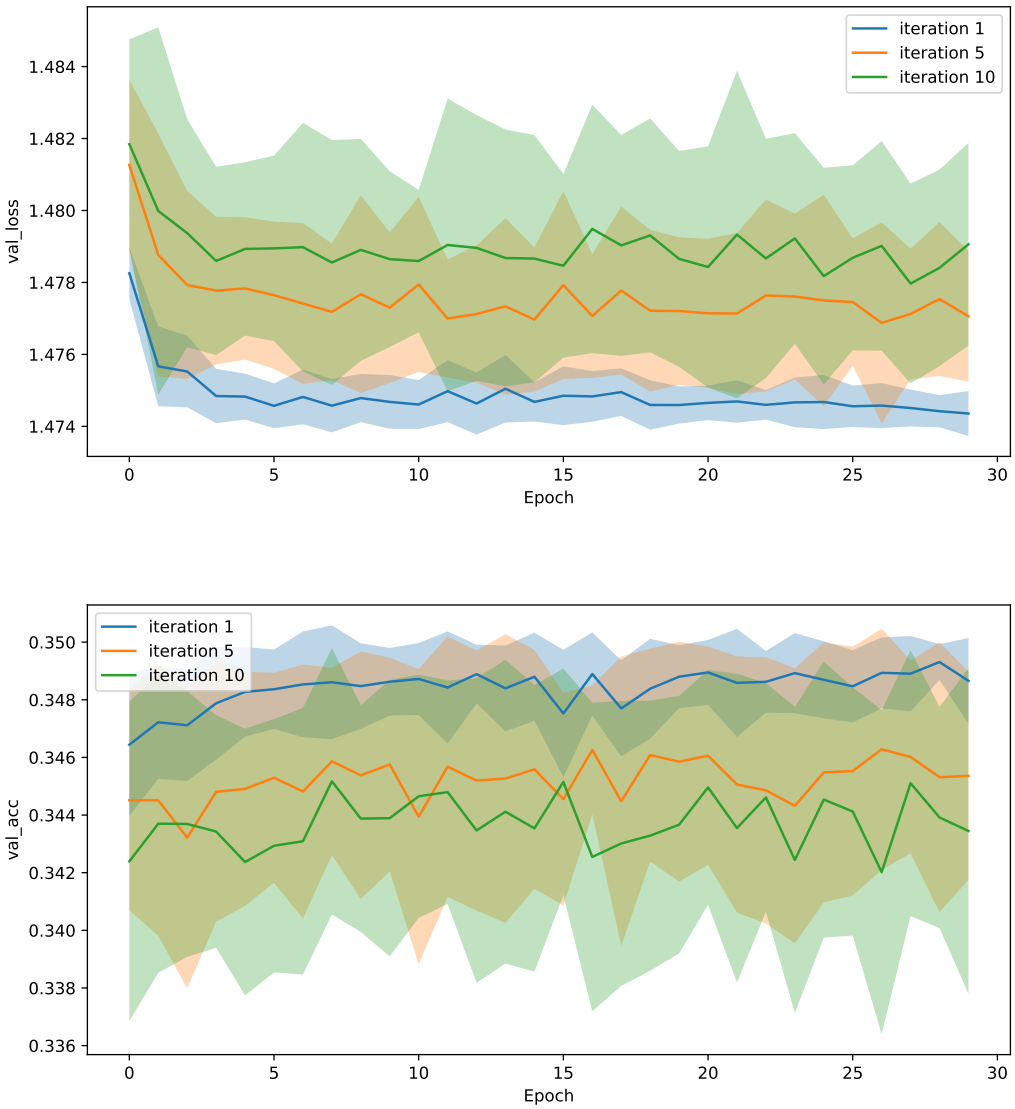

Figure 3: Top: The loss function for the validation data during the training. Bottom: Accuracy for the validation data during the training. Standard deviations were calculated by using the bootstrap technique. The original initial prior. Input variable: `boson_pT_reco`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

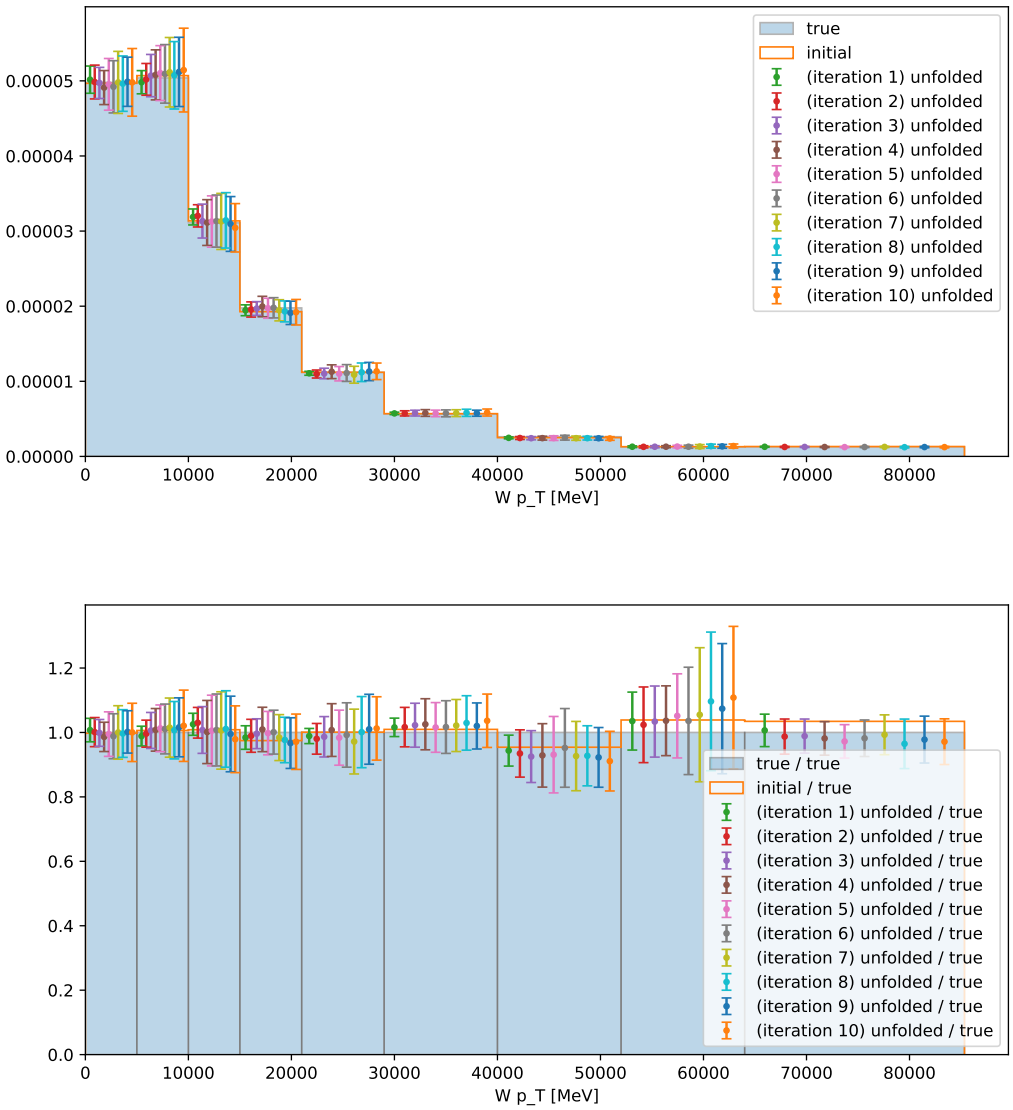

Figure 4: Top: the true p_T^W distribution of the test set, the initial distribution of the train set and results of the unfolding for 10 iterations. Bottom: the ratio $\frac{\text{unfolded distribution}}{\text{true distribution}}$. The rightest bin is the overflow bin. The original initial prior. Input variable: `boson_pT_reco`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

4.2 The shifted initial prior

The results of the unfolding with the original initial prior distribution are unreasonably optimistic. In the real life, we do not know the p_T^W distribution as long as our ultimate purpose is to measure it. However, the p_T^W distribution among train events can be slightly changed. Then the results of the unfolding will be worse but metrics will be more realistic. Let $p_0(p_T^W)$ be the original probability density function of the p_T^W among train events. It can be shifted to the right by 5%:

$$p_1(p_T^W) = \frac{p_0(p_T^W/1.05)}{\int_0^{+\infty} dx\, p_0(x/1.05)} = \frac{p_0(p_T^W/1.05)}{1.05}$$

In practice, we build the histogram of $1.05p_T^W$ and use it as the initial prior distribution. It means that the appropriate weights of the train events need to be set before the first iteration. The results are shown on fig. 5, 6. Predictions are worse than for the original initial prior distribution. However, now the differences between iterations are visible. Nevertheless, the initial distribution is still very similar to the distribution among test events.

4.3 The flat initial prior

For the next test, it is worth trying to make as few assumptions about the true p_T^W distribution as possible. We set the weights of the train events to make the initial distribution flat. Besides, we should make the probability of the overflow bin equal to a nonzero value. Otherwise test events that fall into the overflow bin would be processed incorrectly. For this reason, the initial probability of the overflow bin is set into 0.01. The results are shown on fig. 7, 8.

4.4 1D vs 2D unfolding

An opportunity to process multiple input variables is an advantage of artificial neural networks. Until now, we have considered 1D-unfolding: only one input variable, namely, the reconstructed p_T^W that is named `boson_pT_reco`. However, as shown on fig. 7, 8, this information is not enough to obtain a precise estimation of the true p_T^W distribution. Among measured variables there is the hadronic recoil, or `sumEt`, that is approximately equal to the true p_T^W. It can be used during the training and making the predictions. The results are shown on fig. 9, 10.

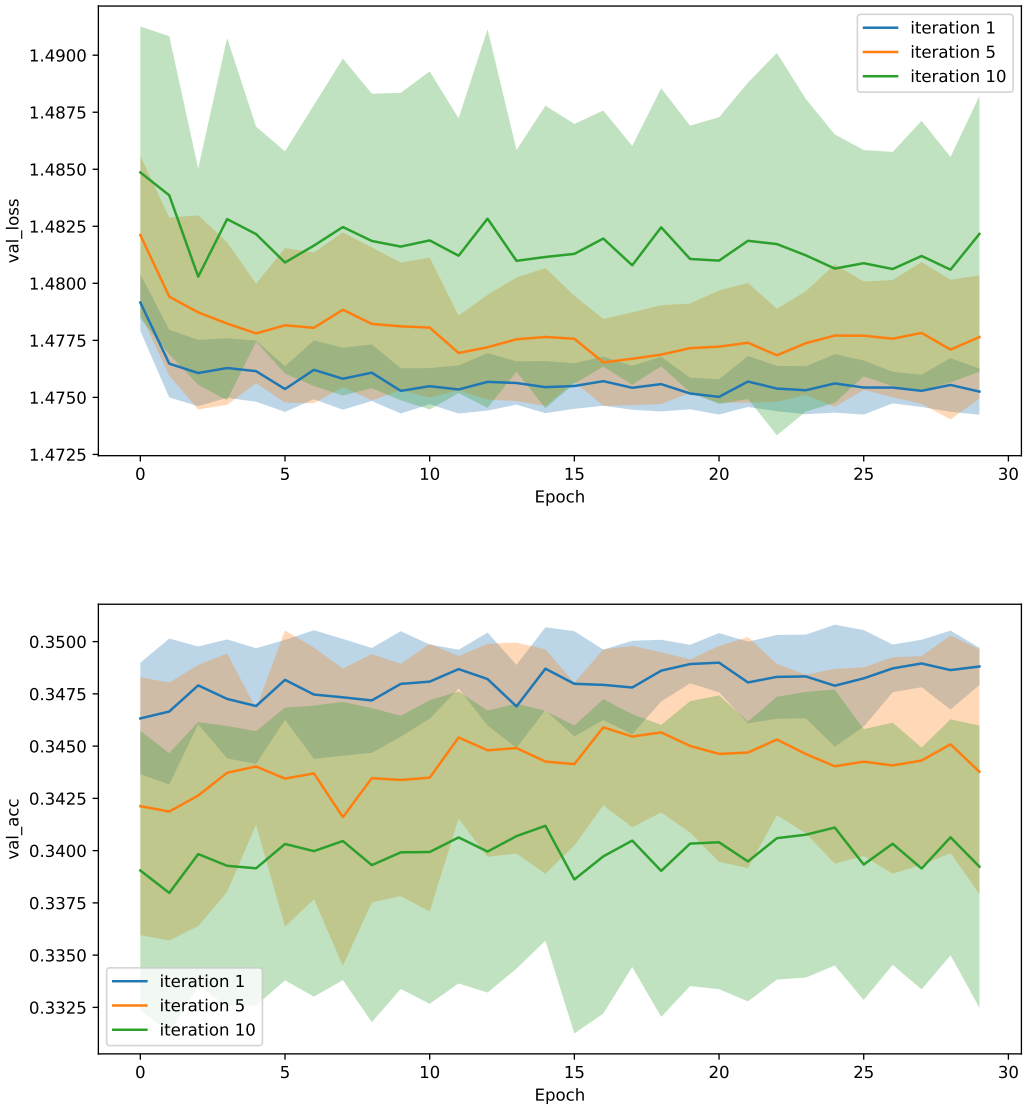

Figure 5: Top: The loss function for the validation data during the training. Bottom: Accuracy for the validation data during the training. The prior is shifted by 5%. Input variable: `boson_pT_reco`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

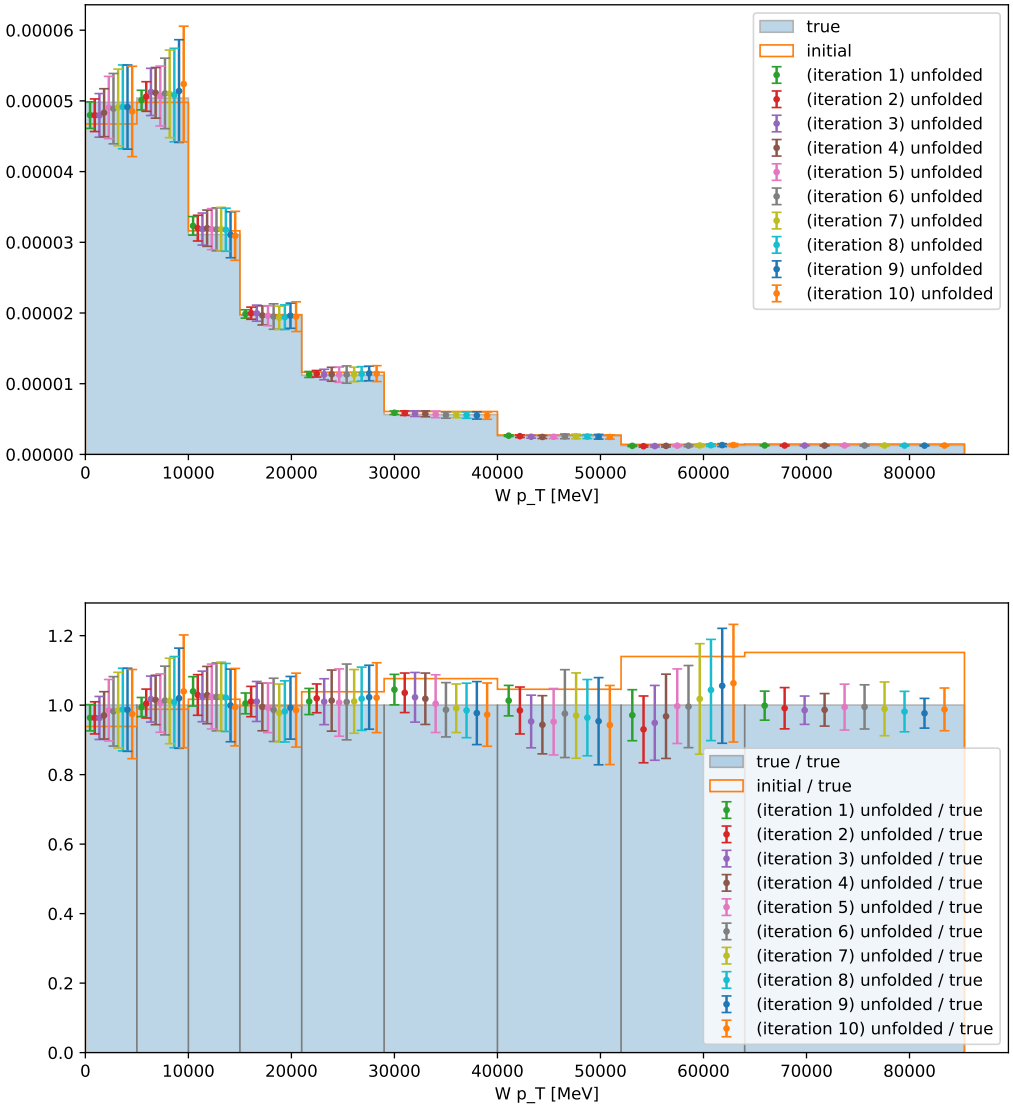

Figure 6: Top: the true p_T^W distribution of the test set, the initial distribution of the train set and results of the unfolding for 10 iterations. Bottom: the ratio $\frac{\text{unfolded distribution}}{\text{true distribution}}$. The rightest bin is the overflow bin. The prior is shifted by 5%. Input variable: `boson_pT_reco`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

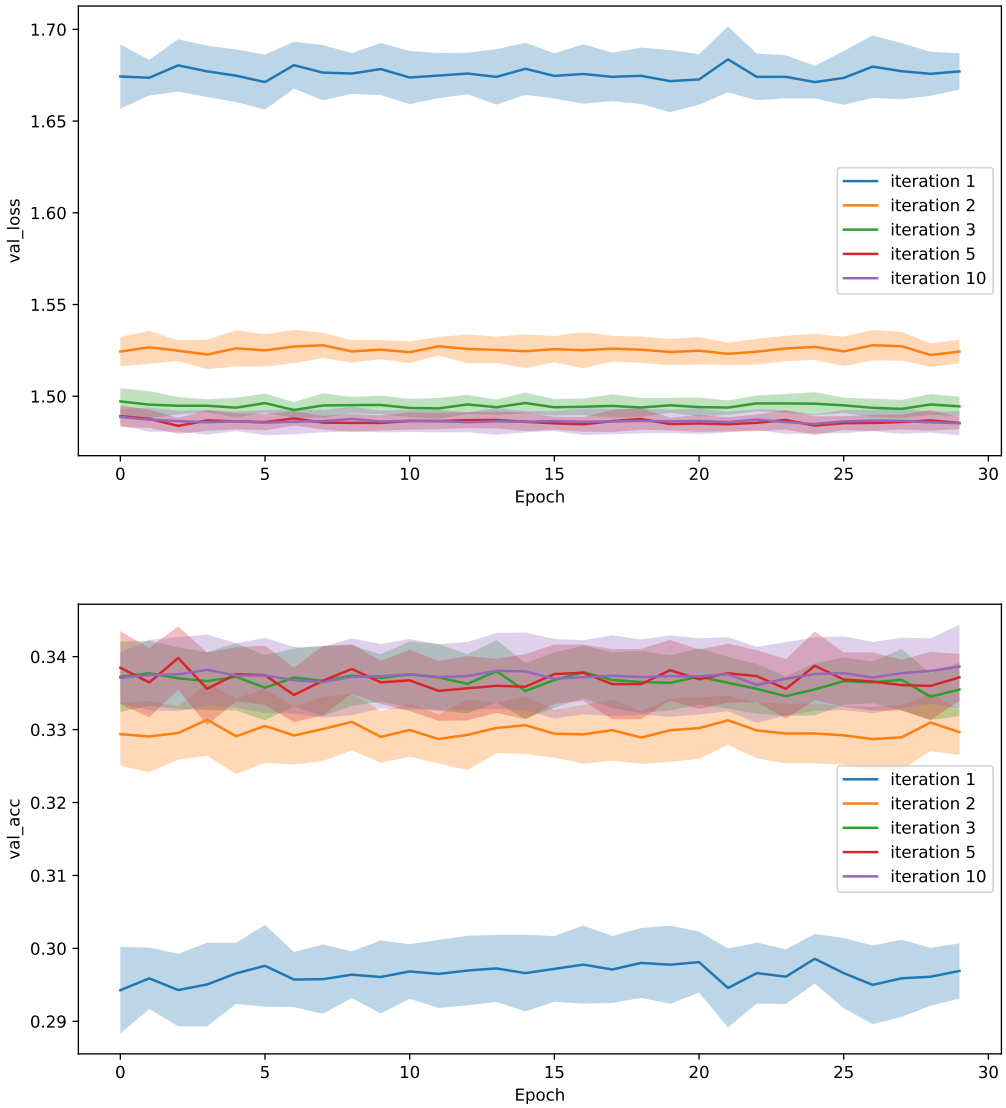

Figure 7: Top: The loss function for the validation data during the training. Bottom: Accuracy for the validation data during the training. The flat initial prior. Input variable: `boson_pT_reco`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

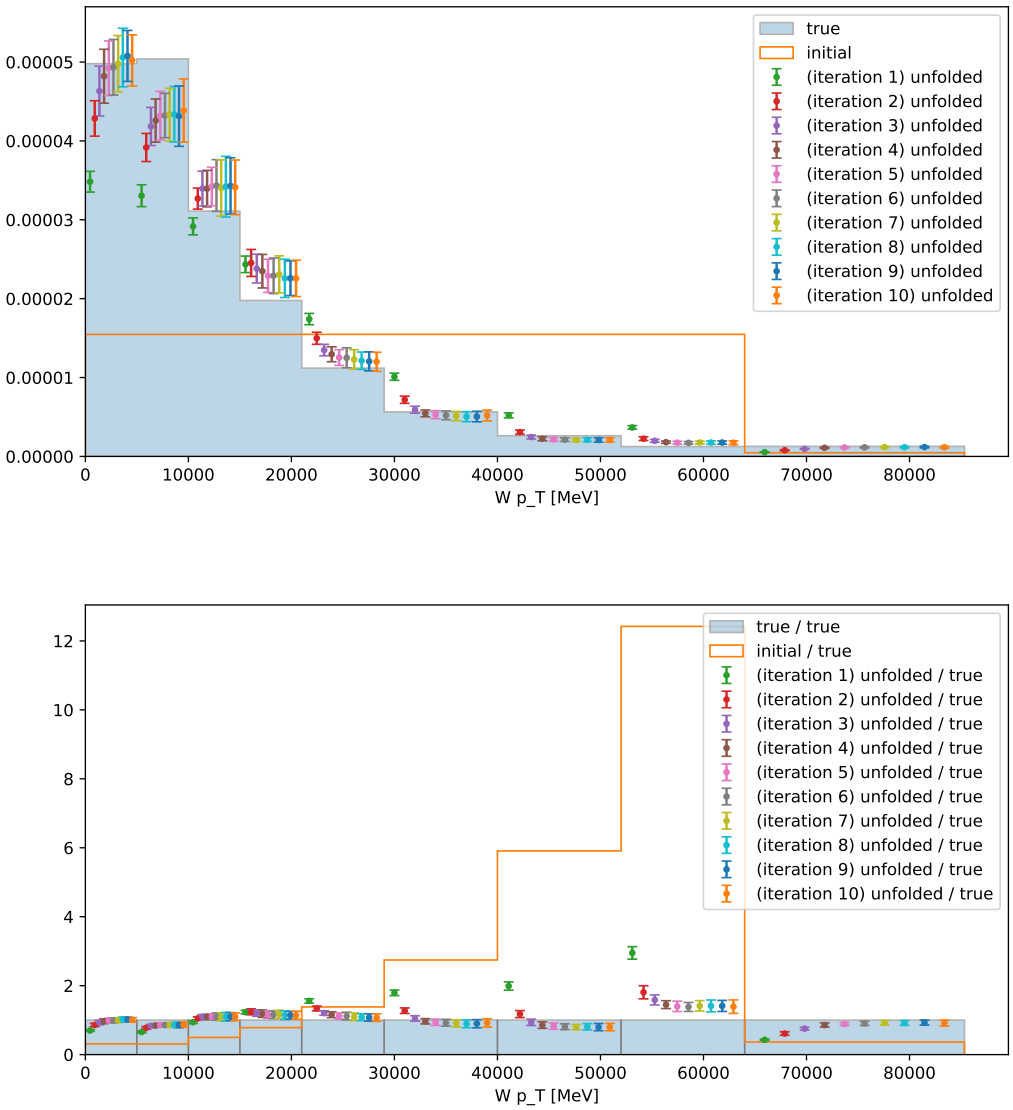

Figure 8: Top: the true p_T^W distribution of the test set, the initial distribution of the train set and results of the unfolding for 10 iterations. Bottom: the ratio $\frac{\text{unfolded distribution}}{\text{true distribution}}$. The rightest bin is the overflow bin. The flat initial prior. Input variable: `boson_pT_reco`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

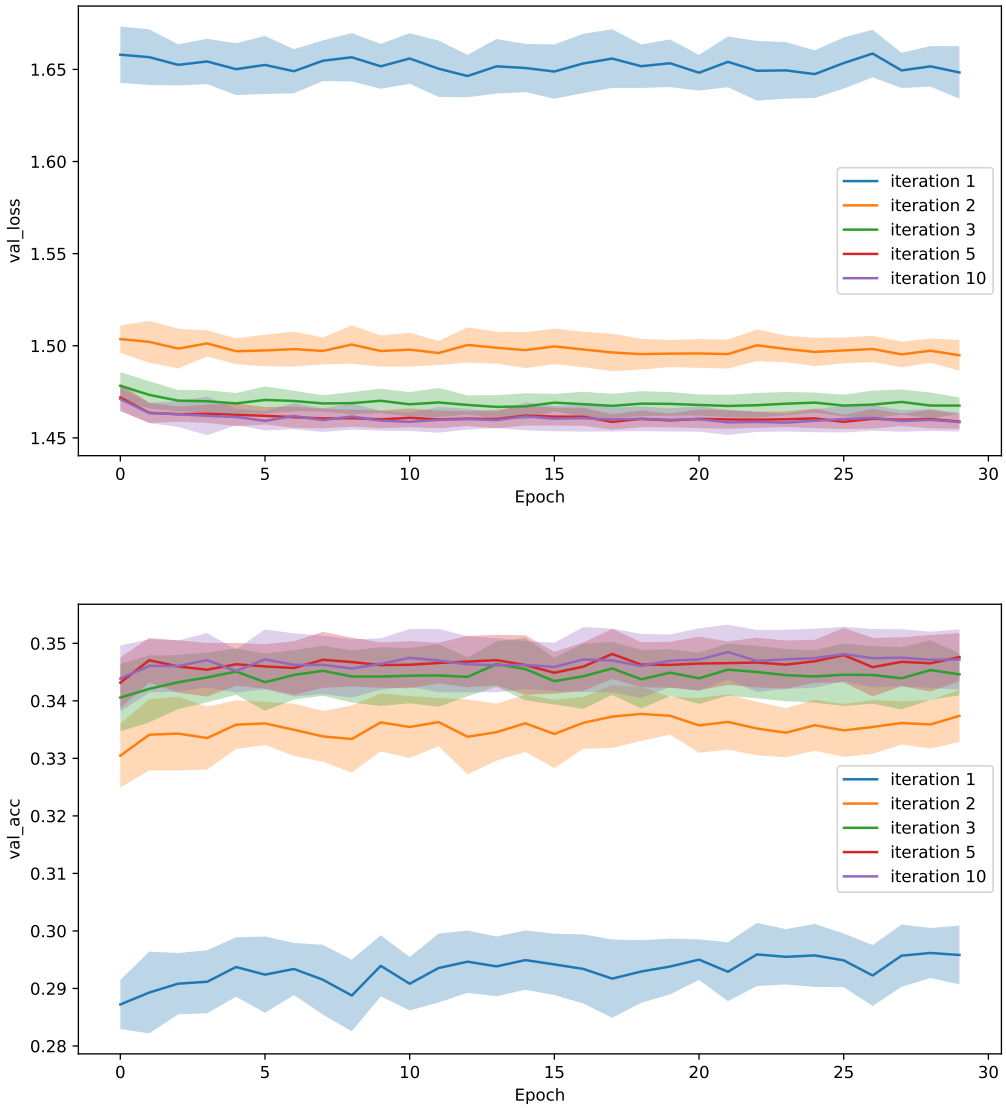

Figure 9: Top: The loss function for the validation data during the training. Bottom: Accuracy for the validation data during the training. The flat initial prior. Input variables: `boson_pT_reco`, `sumEt`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

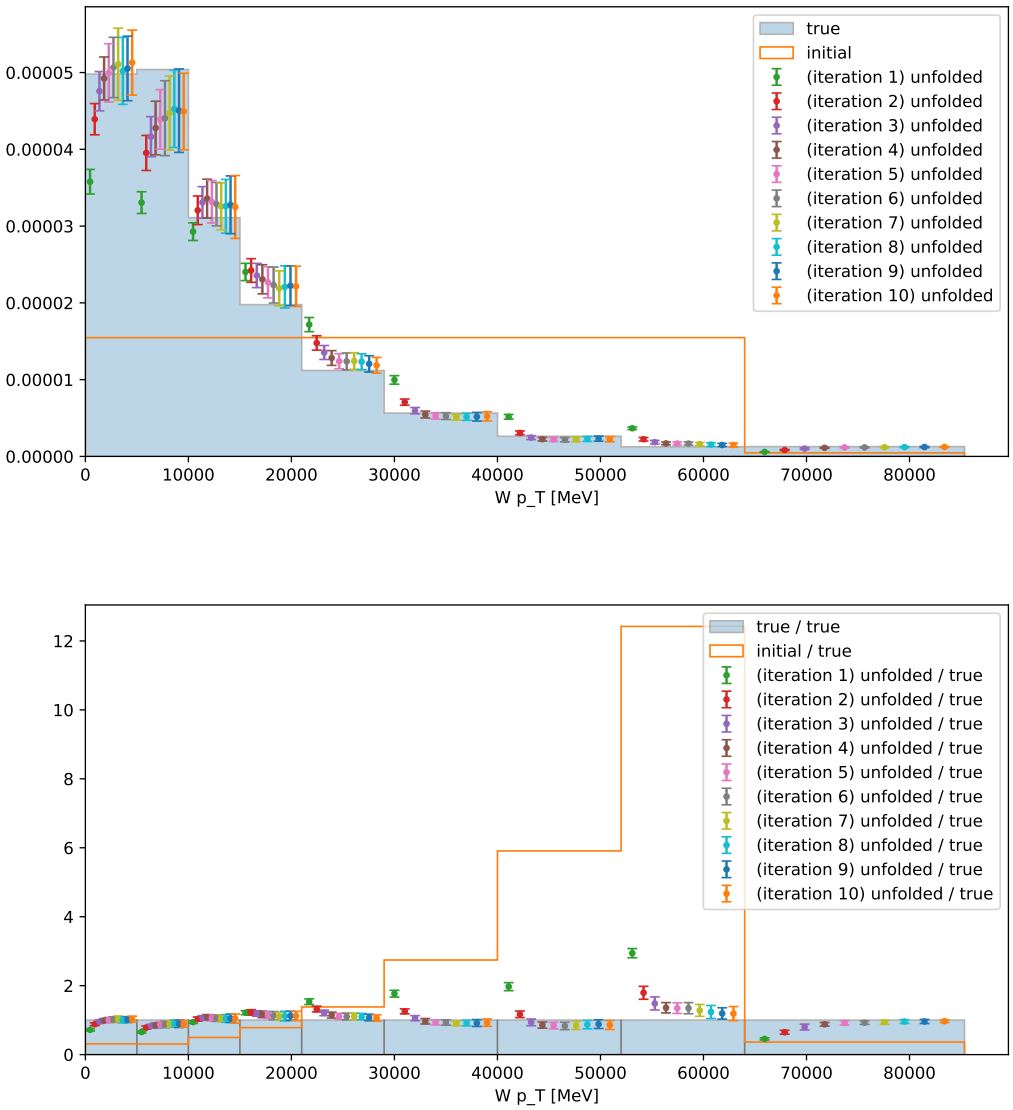

Figure 10: Top: the true p_T^W distribution of the test set, the initial distribution of the train set and results of the unfolding for 10 iterations. Bottom: the ratio $\frac{\text{unfolded distribution}}{\text{true distribution}}$. The rightest bin is the overflow bin. The flat initial prior. Input variables: `boson_pT_reco`, `sumEt`; the train set size: 250000 events; the validation set size: 83333 events; the test set size: 83333 events; the number of bootstrap replicas: 20.

5 Summary

Machine learning is useful for the iterative unfolding. The application of this method to the ATLAS measurement of the p_T^W works properly. Statistical uncertainties can be estimated through bootstrap replicas. The results can be improved by adding new input variables.

Further study will consist in optimization of the network architecture, finding the optimal binning and the optimal set of input variables. Besides, systematic uncertainties should be included into analysis.

Acknowledgements

I would like to express my appreciation to my supervisor, Dr. Simone Amoroso, for his guidance and assistance. His patience and leniency made all the process comfortable and pleasant for me. Besides, the research would not be possible without Alexander Glazov who proposed the architecture and assisted me with his valuable advice.

Besides, I am grateful to the DESY ATLAS group, summer students and the DESY Summer Student Program organizers for their kind words, delicious ice cream and my good memories.

References

[1] ATLAS Collaboration. *Measurement of the W-boson mass in proton-proton collisions at 7 TeV with the ATLAS detector*, arXiv:1701.07240 [hep-ex]

[2] A. Glazov. *Machine learning as an instrument for data unfolding*, arXiv:1712.01814 [physics.data-an]

[3] Wikipedia contributors. *Artificial neural network*, Wikipedia, The Free Encyclopedia

[4] G. D'Agostini, *Improved iterative Bayesian unfolding*, arXiv:1010.0632 [physics.data-an]

[5] ATLAS Collaboration, *Prospects for the measurement of the W-boson transverse momentum with a low pileup data sample at $\sqrt{s}$=13 TeV with the ATLAS detector*, ATL-PHYS-PUB-2017-021