Unfolding technique for the $t\bar{t}$ $e-\mu$ analysis via machine learning (deep neural network)

DESY Summer Student Programme, 2019

Luiza Adelina Ciucu
University of Geneva, Switzerland

Supervisors:
Thorsten Khul, Yichen Li

May 24, 2020

Abstract

The validity of the Standard Model is tested also through precision measurements of the top quark properties. Inferring from measured quantities what happens in reality in nature is called unfolding. Traditional unfolding methods via migration matrices have limitations. A new method of unfolding using machine learning via neural networks (NN) has been proposed in a paper. In this report this NN technique is applied for unfolding the jet energy transverse momentum in the context of the $t\bar{t}$ $e-\mu$ analysis in ATLAS. A deep NN architecture is optimised for this task, leading to an improved performance over the default architecture suggested in the paper.

UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES

Contents

1 Introduction

1.1 The Standard Model

The Standard Model (SM) is a theory that describes the elementary particle (fermions and bosons) and their interactions via three fundamental forces. The SM elementary particles are illustrated in Figure 1 and described in more detail below.

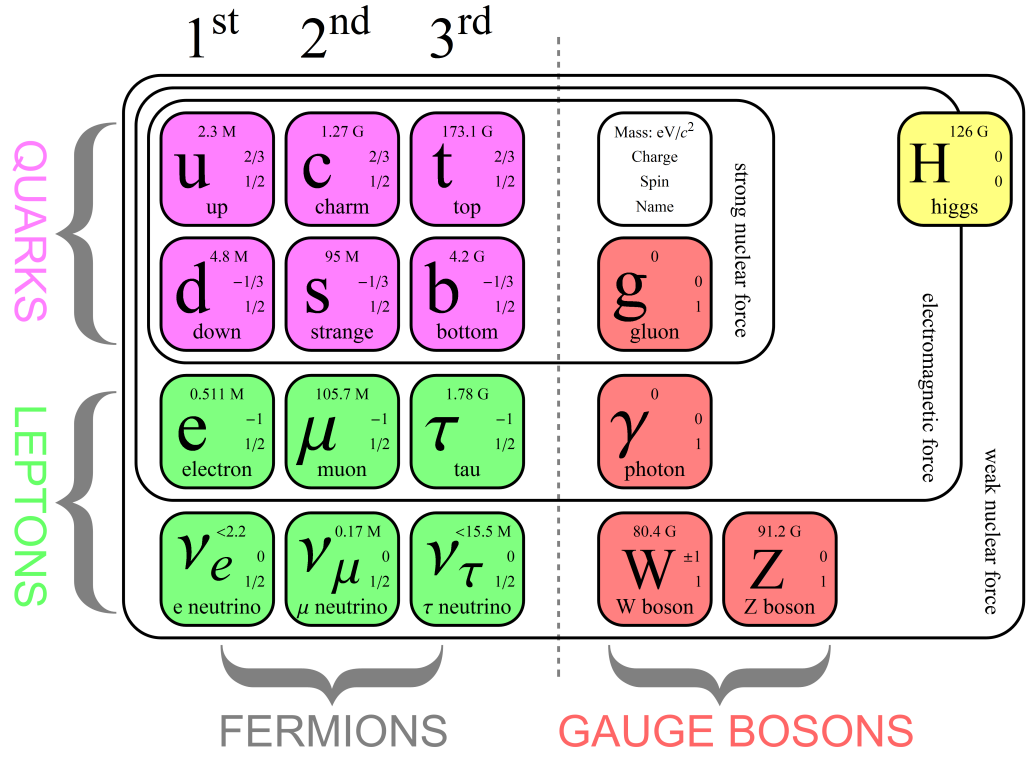

Figure 1: The particles of the Standard Model.

There are 12 fermions representing the constituents of matter. They have a semi-integer spin (1/2) and they can be divided into two categories: quarks and leptons. The quarks have fractional electric charges and form barions and mesons. They six quarks are up ($u^{+2/3}$), down ($d^{-1/3}$), charm ($c^{+2/3}$), strange ($s^{-1/3}$), top ($t^{+2/3}$), and bottom ($b^{-1/3}$). The three electrically-charged leptons have a negative charge: the electron (e^-), the muon (μ^-), and the tau lepton (τ^-). These have corresponding neutrally-charged leptons caled neutrinos (${\nu_e}^0$, ${\nu_\mu}^0$, ${\nu_\tau}^0$). These particles form matter. For each matter particle there is an anti-matter particle that has an opposite electric charge.

Fermions interact with each other via the echange of other elementary particles called bosons. Bosons are carriers of the elementary forces and have an integer spin (1). There are eight type of gluons (g) that carry the strong force. The photon (γ) is responsible for the electromagnetic force. The W^+, W^- and the Z bosons carry the weak force.

There is also another type of boson, a scalar elementary particle of spin zero (0), called the Higgs boson. The Higgs boson is the latest elementary particle discovered, in 2012, by

the ATLAS and CMS detectors at CERN. It is predicted by the mechanism that explains how the elementary particles acquire mass.

1.2 The top quark

The top quark (t) is a very interesting elementary particle to study, for several reasons. It is the heaviest particle from the SM, with a mass m_t=173.3 GeV/c^2. This implies the top quark has the highest Yukawa coupling constant. The top quark decays very quickly ($\tau = 10^{-25}$ s), before having the time to hadronize, meaning to form stable hadrons. Therefore it is the only quark that can be studied alone. It is observed indirectly through its decay products. The top quark was discovered in 1994 independently by the CDF and DZero experiments at FermiLab in proton-antiproton ($p\bar{p}$) collisions at $\sqrt{s} = 1.8$ TeV.

Measuring the top-quark properties is key to test the validity of the SM. The ATLAS and CMS experiments at CERN measure the properties of the top quark in great detail. If deviations are found in the top quark properties with respect to the SM predictions (*e.g.* different cross-sections), it implies the existence of new physics phenomena beyond the Standard Model (BSM). This would produce a revolution in particle physics, as the SM has not been contradicted by any experiment since its creation in the 1960s.

The top quarks are produced in pairs by strong interactions. The Feynman diagrams for the leading order are illustrated in Figure 2. At the LHC the quark-antiquark annihilation (left) represents 10% of the cases, while the gluon-gluon fusion (center and right) represent 90% of cases. The LHC has a very high luminosity and produces large quantities of top quarks. For this reason the LHC is considered to be also a top-quark factory.

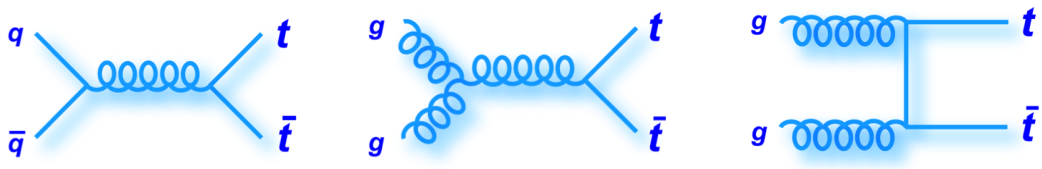

Figure 2: The Feynaman diagrams for the top-quark pair production. At the LHC the quark-antiquark annihilation (left) represents 10% of the cases, while the gluon-gluon fusion (center and right) represent 90% of cases.

The top quark decays in almost 100% of the time in a W and a bottom (b) quark. The W boson can decay either to a pair of a quark or an antiquark ($W \to q\bar{q}$), or to a pair of leptons ($W \to e\bar{\nu}_e$, $W \to \mu\bar{\nu}_\mu$, $W \to \tau\bar{\nu}_\tau$). The τ lepton decays further leaving as a visible signature in the detector an electron, a muon or two quarks. Since there are two top quarks, they decay to two W bosons and two b quarks. Quarks hadronise and appear in the detector as streams of collimated particles called jets.

Counting the number of charged leptons (electrons or muons), the top quark events can be grouped into three categories: di-lepton, lepton+jets (one lepton plus jets), and all-hadronic (only jets). The percentage of decay for each case, called branching ratio (BR) is presented in the left-hand side of Figure 3. In this report we study the subset of the

di-lepton with an electron or a muon. The $t\bar{t}$ $e-\mu$ decay is present in only 2% of the cases. It is advantageous because the background is very low comparing to others decay channels. The background comes in most of the cases from the Z boson decay plus other jets. The full Feynman diagram of production and decay of the $t\bar{t}$ $e-\mu$ is presented in the right-hand side of Figure 3. The latest ATLAS paper on the $t\bar{t}$ $e-\mu$ analysis is presented in [1].

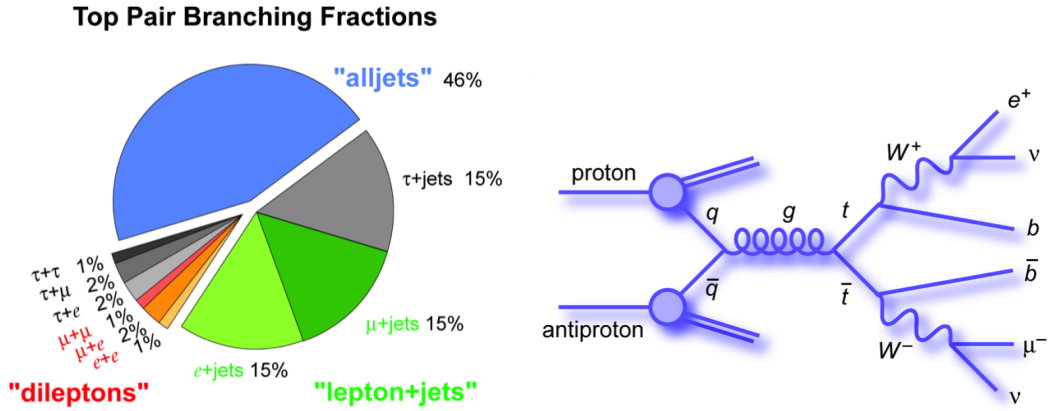

Figure 3: Left: the pie chart of the top-quark pair decay. Right: the Feynman diagram including production and decay of the $t\bar{t}$ $e-\mu$ process.

1.3 The experimental setup

The experimental setup used in this study is from the ATLAS experiment at CERN.

At the moment, the Large Hadron Collider (LHC), which is situated at CERN, is the most powerful proton-proton collider in the world, as illustrated in Figure 4. After the Tevatron and the Large Electron-Positron Collider (LEP) era, a new machine was needed for new discoveries in particle physics. The LHC was designed to achieve a center-of-mass energy $\sqrt{s} = 14\,TeV$. Two of the biggest goals of the LHC are to study the Standard Model and to test its validity.

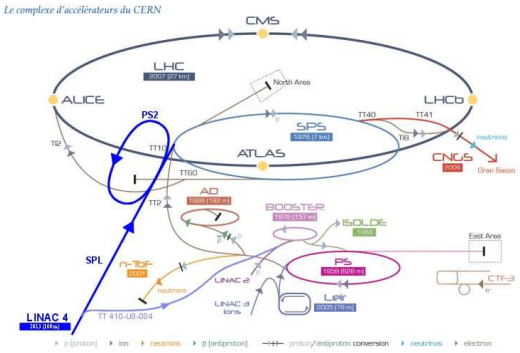

Figure 4: The LHC Accelerator Complex System.

This project is affiliated to the international collaboration of the A Toroidal LHC ApparatuS (ATLAS) [2]. ATLAS is the largest of the four LHC detectors. ATLAS is one of the two general-purpose particle physics detectors at the LHC, the other being CMS. The two detectors and collaborations perform a similar research program. A new discovery must be observed by both detectors to be believed as true. ATLAS and its subdetectors is illustrated in Figure 5.

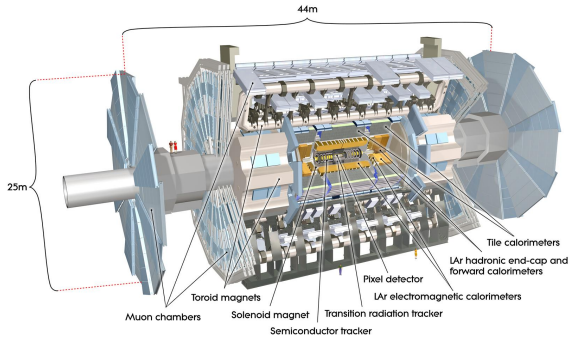

Figure 5: The ATLAS detector and its sub-detectors.

This report presents results obtained using 43076 simulated events of the $t\bar{t}$ $e-\mu$ process within the ATLAS experiment [3]. After an event selection, for each event there is the information available at the reconstructed stage (measured) stage and at the truth (generated) stage. The study focuses on the jet with the highest transverse momentum (p_{T}), also called the leading jet, in the interval from 0 to 500 GeV, with a constant bin width of 20 GeV, which results in 25 bins in total.

2 Unfolding methods

As introduced in Section 1, after the discovery of the Higgs boson at CERN, it becomes even more important to test the validity of the SM via precision measurements on SM particles. Properties of such particles are measured in the ATLAS detector. The measured quantities are close, but not identical to the true value of the quantities these particles had in reality in nature, due to detector smearing effects. Every measurement has a total uncertainty, formed from statistical and systematic uncertainties. It is therefore important to have a mechanism to reverse the detector smearing by inferring the true value of a quantitity given the measured value of the quantity. Such mechanism is called *unfolding* and is the topic of this project. The estimated true quantity from unfolding the reconstructed quantity is then compared with the theoretical prediction. If a difference is observed even after taking into account the experimental and theoretical uncertainties, a deviation from the SM would be observed. If confirmed at five standard deviations (σ), it would represent a revolution in particle physics. Therefore, unfolding methods are key to testing the SM and searching for new physics beyond the Standard Model (BSM). An introduction to unfolding methods for particle physics can be studied in [4].

There are two types of unfolding methods: a traditional method using a binned 2D hiso-

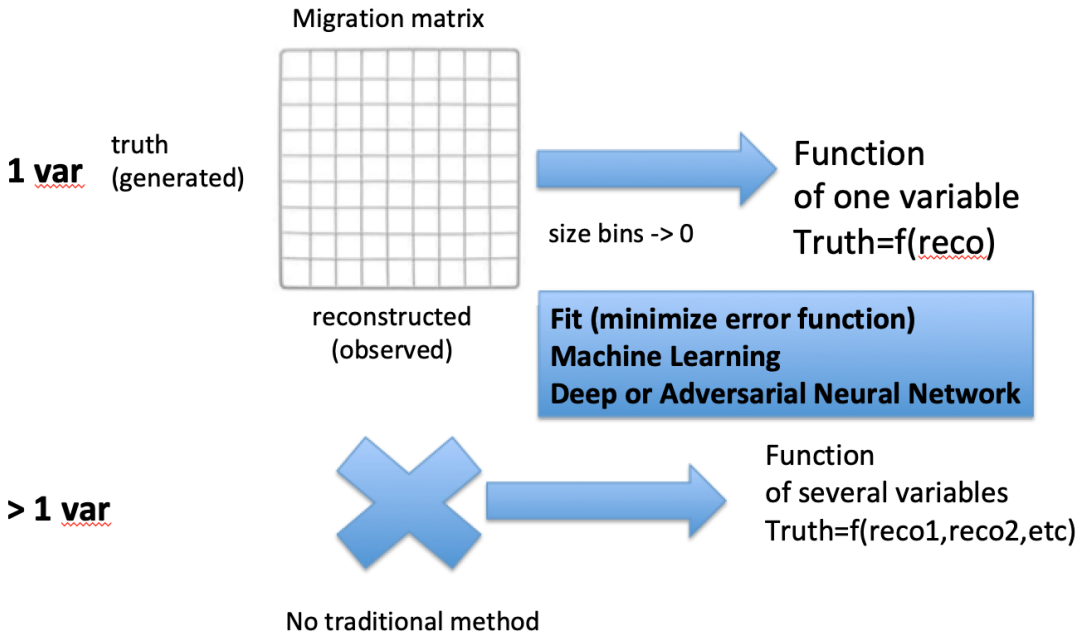

Figure 6: Comparison of the two unfolding method: traditional (migration matrix, one variable) and machine learning (continuous function, several variables).

gram representing a migration matrix for one variable, and machine learning predicting a continuous function for several variables, as illustrated in Figure 6.

2.1 The traditional method: the migration matrix

In the traditional method, only one quantity is used at a time. In simulated data, there are two versions for each quantity of the same event: a reconstructed quantity (or reco, which for measured data would be the observed quantity) and the generated quantity (that would be the truth, or the true value in nature). An interval and binning is chosen for this variable. A 2D histogram is filled, as a given event is in a particular bin in the reco on the horizontal and in a particular bin in the truth on the vertical. The histogram is called a migration matrix, as it describes how a quantity migrates from a particular bin in the reconstructed to another bin (usually nearby) for the truth. Ideally, this matrix is diagonal. In practice, there are non diagonal values that are non zero. Such a matrix is used in the traditional unfolding.

For the simulated dataset studied in this report, the migration matrix of the leading jet p_{T} bin value is presented in Figure 7, taken from [5]. There are 25 bins possible for the leading jet p_{T} values between 0 and 500 GeV, with 20 GeV bins. The number in a cell represents the probability (as a percent) of an event in the truth bin i to be reconstructed in a reco bin j. The numbers are normalised so that the sum of values on a horizonthal row is 100 (percent). The more diagonalizable matrix, the easier unfolding will be. Events are migrating from one truth bin to other bins for reco, and vice-versa.

Traditional unfolding is an iterative Bayesian process. After four iterations, the unfolded

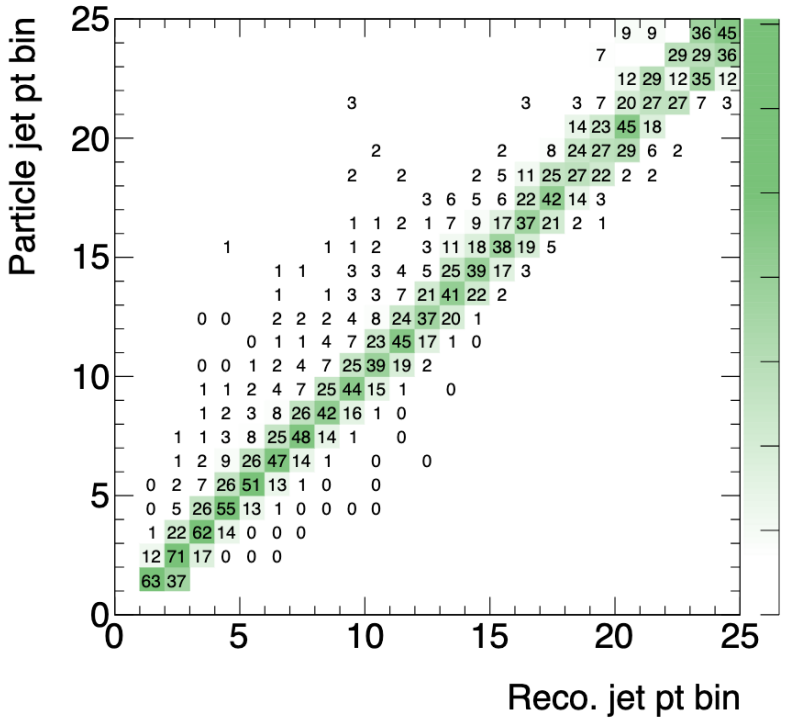

Figure 7: The leading particle jet p_{T} bin versus the leading reco jet p_{T} bin, for the same events of this study [5].

results are presented in Figure 21, taken from [5]. The left hand-side presents the jet distribution for truth and for the truth unfolded from the reco, while the right-hand side presents the coefficients matrix in a 2D histogram format.

2.2 The new method: machine learning

As the bin width is decreased continuously closed to infinitesimal values, the migration matrix becomes a function of one variable that maps each reco quantity to its truth quantity. This is equivalent to a function `truth=f(reco)`. But it still can have only one variable. Finding such a function is in general found via minimizing an error function over some parameter space, a process which is also called a *fit*. A fit with even several variables can be done successfully via machine learning, where a functions is learned of the form `truth=f(reco1, reco2, etc.)`.

A new approach of unfolding via machine learning has been proposed by A. Glazov at DESY Hamburg [6], together with a code example [7] of training a (deep) neural network on toy simulated data using TensorFlow [8] via [9] in `Python`, using numpy [10], as illustrated in Figure 9.

The goal of the study is to adapt this example to use real data coming from the jet p_{T} distribution from the $t\bar{t}$ $e-\mu$ analysis. We use this `.root` flat tree [3]. This file contains both the reconstructed and truth jets, already matched. Meaning the i^{th} event from reco, corresponds to the j^{th} event from truth. Each jet collection contains jets already the jets

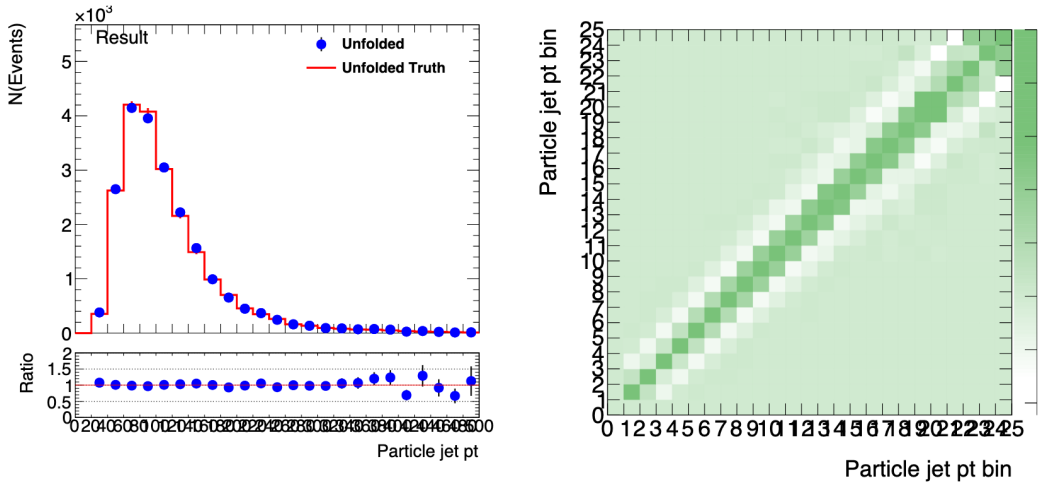

Figure 8: Traditional unfolding results for the leading jet p_{T} in a 1D and 2D histogram.

Figure 9: A modern (deep) neural network is typically trained in Keras with a TensorFlow backend, run in Python, using numpy.

ranked by p_{T}. We take index 0 to consider the leading reco jet and the leading truth jet.

Using this approach, the leading jet p_{T} unfolding is formulated in the example illustrated in Figure 10. For both reco and truth, p_{T} larger than 500 GeV are changed to 499.999 GeV, and thus moved to the last bin. This is equivalent to moving the overflow bin of a histogram to the last bin of a histogram. The truth and reco are thus ensured to have the same number of potential bins (25), given the 0-500 GeV range and the bin width of 20 GeV. The simplest case of only one variable is considered: the jet p_{T}, or more exactly, the jet p_{T} bin. Let's consider an example for the reconstructed jet (the input of unfolding) and the truth jet (the output of unfolding).

For a reconstructed jet, the ratio between the p_{T} and the bin width represents the input to the machine learning. For example, for a 57.8 GeV jet, the value is 2.89. The jet would fit in the third bin, or the bin of index 2 (index values start at 0).

For a truth jet, things are similar, but a bit more complex. Imagine the same jet has a different value for the truth, equal with 60.4 GeV. Dividing by the bin width, 3.02 is obtained. The truth jet falls in the fourth bin, or the bin of index 3. The goal of the unfolding via machine learning is to learn and then predict, or infer, this number 3. 3 is only one possible answer out of finate number of posibilities corresponding to the 25 bins. Each bin can be considered one category of potential output. Each jet belongs to only one category. This problem is called classification and can indeed be solved via machine learning. To note that in this method is not attempted to predict exactly the p_{T} of the

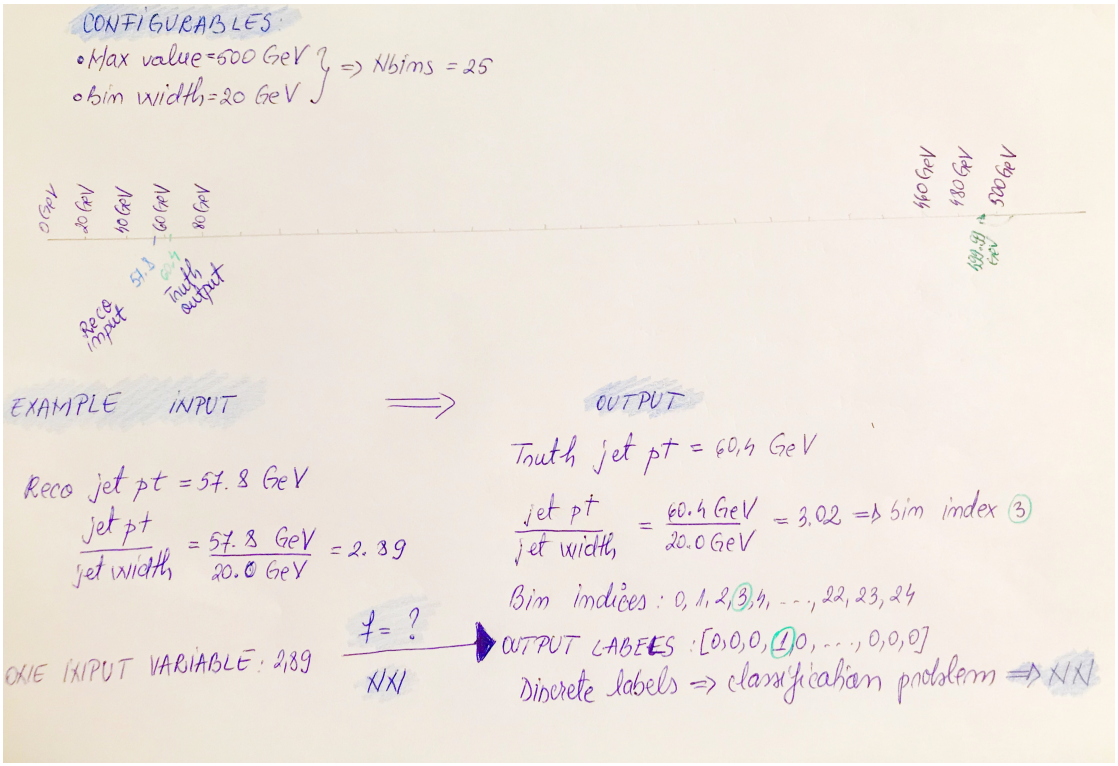

Figure 10: Exemple of problem posing for the leading jet p_{T} unfolding with the machine learning approach of [6] [7].

truth jet. There would be a continous interval of possible values. Such a problem is called regression. It is also possible to be solved via machine learning, but it is more complex.

In this project we choose to predict only the bin where the jet p_{T} falls into, or a multi-label classification or categorization problem. The categorisation is a common problem solved via modern Machine Learning methods. Possible methods could be Boosted Decision Trees (BDT) [14] [15] and Artificial Neural Networks (ANN or NN) [15].

Classification is usually binary, distinguishing between two possibilties: signal or background, good or bad, 1 or 0. It is therefore technically easier to formulate that for the input of 2.89, the output not as the bin index 3, but as a range of 25 values, each being 0 or 1, all of them being zero, except for the index 3, where the value is 1, as illustrated in Table 1. Finding such a function connecting this type of input to this type of output for all the jets in the event is done via a deep neural network (NN) training in this project. The NN will try to learn and predict for each of the 25 positions the values of 0 or 1. But in practice it will output for each position a real number with a value between 0.0 and 1.0. The NN will return the probability that for a given jet its p_{T} falls in any of these bins. The total probability for all bins must be 1.

The input and output layers of the NN are fixed by the problem formulation. The architecture of the middle layers can be optimised for the specific problem. The NN architecture suggested in the code example [7] is denoted in this report the old NN. Several architectures

Jet	Input	Output
Physics	2.89	3.02
Classification	2.89	3
Neural Network	2.89	[0,0,0,1,0,0,...,0,0,0]

Table 1: Example of jet reco (input) and truth (output) from physics to the NN.

and NN hyper-parameters have been optimised in this study by changing one parameter while keeping all the rest constant. The best of the many NNs tried is denoted in this report the new NN.

The following section 3 will describe the optimisation process and compare the two NNs in terms of the NN-specific figures of merits of accuracy and loss. After that, the section 4 will compare the two NNs in terms of the unfolding of the leading jet p_{T}.

3 Neural Network Training

3.1 Coding environment

The toy data code example [7] uses Jupiter Notebook. The ML commands can be run either from Jupiter Notebook or from a regular Python script from a laptop with Anaconda with Python2 or Python3. For this problem, one needs the ability to run `ROOT` in the same environment to read the `.root` file with the jet data from the analysis. ROOT can not be run from Anaconda, however. One option is to split the code into two separate files, each run in separate environments and communicating to each other via numpy arrays. This is not ideal, however, since optimising a NN requires many steps. One needs the ability to have all code in one environment and run it all at once.

Another option is to run the Jupiter Notebook in the server provided by ATLAS called SWAN [11]. This environment has access to both Keras and ROOT at the same time. Although all commands can be run at once, the server would often die while running due either to bad wifi connection, or due to CERN server being busy.

To run effectively, it is better to run from only one Python script, instead of from one Jupiter Notebook. The recommended solution is to run from CERN's lxplus machines. Right after the `ssh`, a `singularity` [12] ennviroment is set up via a `docker` container that contains the ML sofware. While `ROOT` is not contained, the `uproot` [13] library is included. `uproot` allows to read the `.root` files in numpy arrays, the data types that are used natively by the ML libraries such as `TensorFlow` inside `Python`. It is this setup that is found to be the most efficient and is used in this project.

3.2 k-fold

NN training requires that the data is split in two independent sets. Some is used to train the NN, while the other is used for testing (also called validation). In this project half of the data (the events with even index) is used in training, and the other half of the data (the events with odd index) is used in testing. Since the data is split in two, it is said to

have a k-fold=2. It is the simplest case and is what is used in this report.

Since there are 43076 events, there are 21538 events in training and also 21538 in testing. Each event has a collection of jets, from which the leading jet in p_{T} is used for both the reco (input) and truth (output).

Other possibilities involve having a k larger than 2, for example 5. In that case, the data is split into 5 independent parts. The training and testing contain 5 consecutive steps. One trains in 4 parts and tests in 1, then repeats with another 1 until the data is evaluated once in each event.

3.3 Neural network overview

In machine learning (ML) a solution for a problem is found without being explicitly programmed. The two main classes of ML are supervised and unsupervised. In this project supervised learning is used, as the output is known and is labeled. The machine learning method tries to learn what the labels are as a function of the input.

Several machine learning algorithms can be used to perform the classification task. In this project neural networks [15] are used. In general, to find a multidimensional highly non-linear function, it is efficient to train a NN. The NN algorithm is inspired from the brain structure and functions. The brain contains millions of neuron cells forming a network where electro-chemical impulses are being passed between them. An artificial neural network is formed by a number of interconnected artificial *neurons*, or *nodes* respects this architecture.

A neural network is formed by several layers, each containing several nodes. The number of nodes on the first (input) and last (output) layers are fixed by the problem to solve. They are 1 and 25, respectively, for the problem presented in this project. The layers in between are called hidden layers. Their number and the number of their nodes can be optimised for hte particular problem. If there is one hidden layer, the NN is said to be *shallow*. If there are more than one hidden layer, the NN is said to be *deep*. If each node is connected to all nodes of the layer before and after, the NN is said to be fully connected. A general structure of a fully connected deep NN (DNN) is presented in Figure 11.

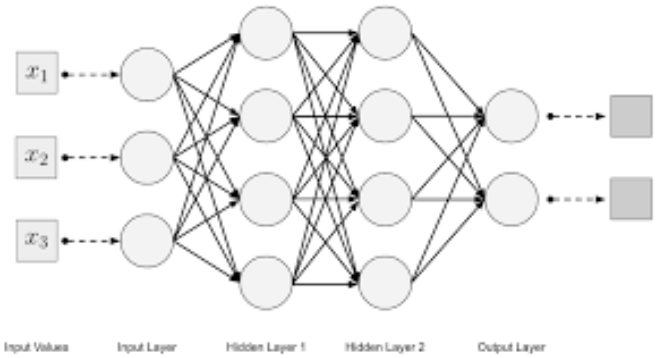

Figure 11: Diagram of a general architecture of a DNN. Credit image: O'Reilly. [16].

The *Universal Approximation Theorem* states that a neural network with one *hidden layer* can in principle approximate any N-dimensional function to an arbitrary degree of accuracy, given a sufficiently large (though finite) number of nodes. In practice however it is more suitable to use multiple hidden layers connected in series [15].

In a fully connected NN, each node takes a weighted linear combination of the outputs from nodes of the previous layer, adds its own bias values, and applies an *activation function*, then outputs the result to the nerons of the next layer, as illustrated in Figure 12. The activation function is choosen via optimisation for each neuron when the architecture of the NN is defined.

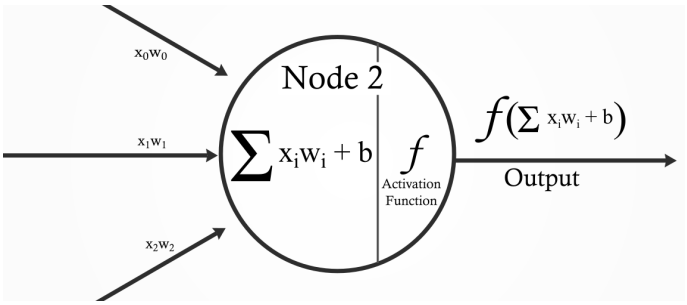

Figure 12: Diagram of a neuron or node in a NN, with its weight inputs, its bias and the output via the activation function. Credit image: The Fork [17].

Once the NN architecture is set the layers, nodes and activation functions, the total function of the NN is parametrized by all the weights for the connections between nodes plus the biases of each node. Training the NN means learning these weights so that the NN function can predict well the output when a new data not seen before is taken as input.

NN training is done learning on the training dataset and testing on the testing dataset. Running once over all the data from training and testing represents an *epoch*. During an epoch, events are analysed in groups called *batches*. The number of epochs to run on and the number of events in each batch can be optimised.

Let's take a look at how the NN training happens. At first the NN has random values for the weights and biases. For each of the events in the first batch, data comes in, and the NN predicts an output values, that are at first very different from the true desired output value. To evaluate how far away are the predicted output from the desired output, a *loss* function is defined that can be chosen from several formulas, but have the generic form of a sum over the absolute values of these differences. The goal of the NN training is to update the values of the weights and biases so that the loss function is minimized. After the first batch, the NN changes the weights via a back-propagation aglorithm that can be selected from several, and usually it is a form of gradiant descent. For each new batch, the weights and biases change, and become continously closer to the correct values, as the loss function becomes gradually smaller. When all the training events are used, the first epoch is finished. The NN function at this point is then applied to the new dataset from the testing dataset, which is not split in batches, and a loss function is also calculated. The

entire procedure repeats for the number of epochs chosen. At the end, the final weights and biases define the final NN that has been learned.

There is also another figure of merit of how well does a NN perform in training and testing. It is called *accuracy* and is related to the number of of true positives or false negatives. The larger the accuracy, the better.

For the training dataset, the loss and accuracy values always improve. But in the testing dataset they can start to degrade if we train for too many epochs. By degrading it means that the loss value starts to grow, and the accuracy value starts to decrease. That is called over-training and consists of memorizing the inputs, and thus not being able to predict correctly any more for new inputs.

3.4 Neural network optimisation

An optimisation of the NN training hyper-parameters described in the Section 3.3 is made by changing only one at a time, while keeping the other constant. The best choice is the one with largest accuracy and smallest loss values. The best of the studied NNs is denoted as the new NN. It is a deep NN. The NN from the code example is denoted as the old NN. It is a shallow NN. Figure 13 visualises the architectures and activation functions of the old NN (top) and the new NN (bottom).

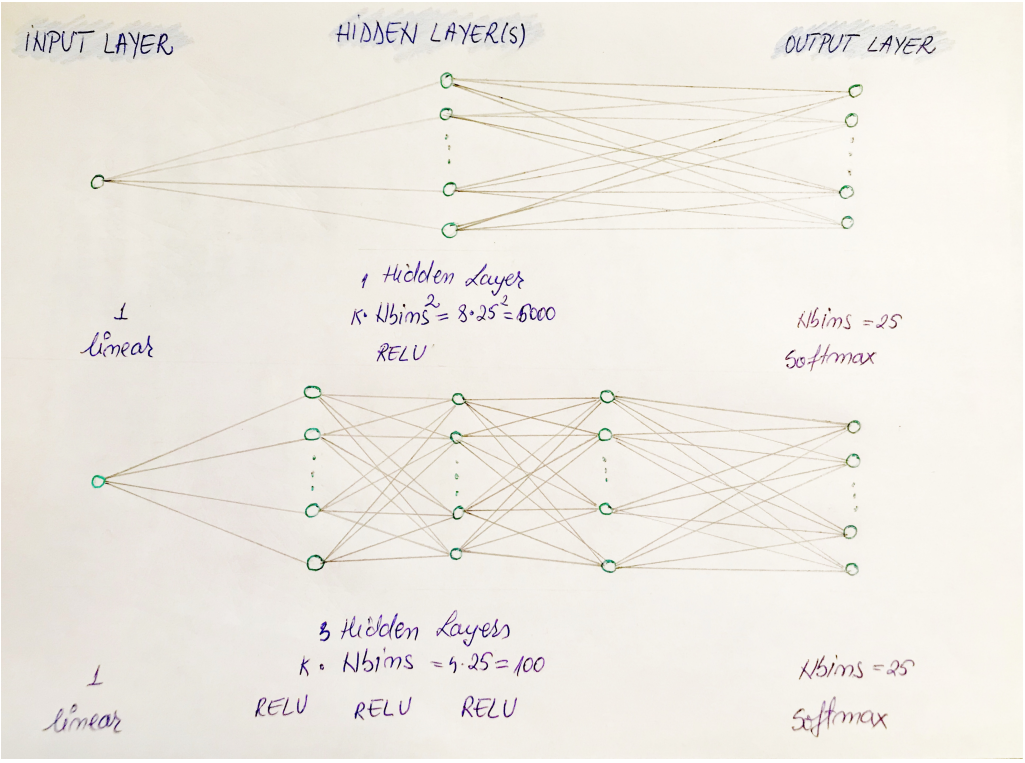

Figure 13: NN architectures and activation functions for the old and shallow NN (top) and the new and deep NN (bottom).

The input and output layers are the same, constrained by the physics problem to solve. The input layer has only one node and uses the *linear* activation function. It means the input data is passed out from the first neuron. The output layer has 25 output variables. It uses the *softmax* activation function [18], which ensures that the output of each node is a number between 0 and 1 and the sum of all these numbers is exactly 1. This ensures that the value of each node is interpreted as the probability that the jet falls in the bin index represented by that node.

The hidden layers have differences. The old NN has only one layer with very many nodes. The new NN has three layers, each with fewer nodes. Both NN use the same *ReLU* activation function [19]. ReLU is the recommended modern activation function to use as default when starting to study a new problem. The ReLu activation function is zero for negative values and the diagonal for positive values, as seen in Figure 14. In future developments of the project, more complex activation functions can be used.

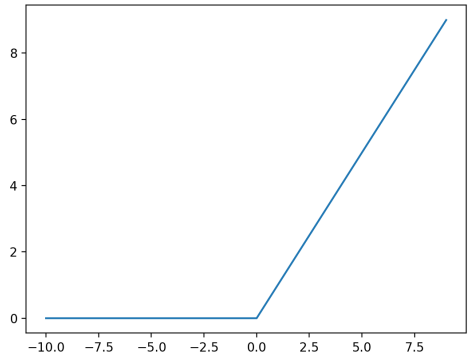

Figure 14: The ReLu activation function is zero for negative values and the diagonal for positive values [19].

The possible choices in hyper-parameters comparing the old and the new NN are summarised in Table 2.

Choice	**Old NN (toy data example)**	**New NN (my best choice)**
number of nodes per input layer	1	1
number of nodes per output layer	Nbins = 25	Nbins = 25
Number of hidden layers	1	3
k	8	4
Number of nodes per layer	$k \cdot Nbins^2 = 5000$	$k \cdot Nbins = 100$
activation function input	linear	linear
activation function hidden	ReLU	ReLU
activation function output	softmax	softmax
batch size	1000	200
number of epochs	150	150

Table 2: NN architecture and hyper-parameters comparing for the old NN, suggested in the toy data paper, and the new NN, from the optimisation of this study.

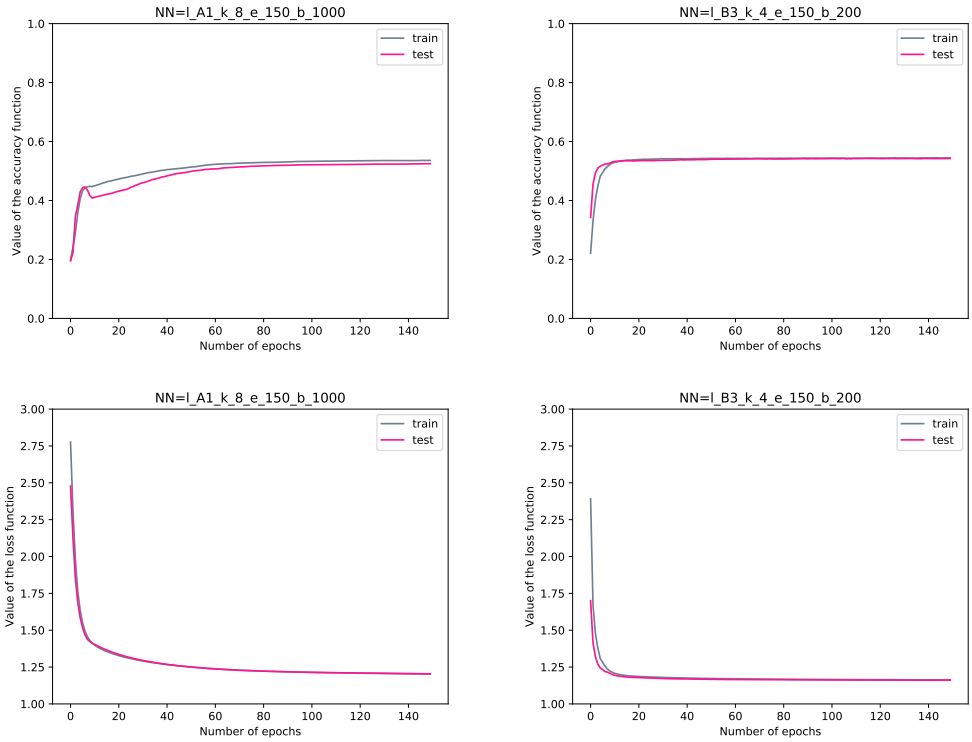

Figure 15: Overlay of train and test for the accuracy (top) and loss (bottom). Old NN (left) and new NN (right).

Figure 15 shows the overlay of train and test for the two figures of merit (accuracy and loss). The train and test curves are very similar. This confirms that the NNs are not overtrained. Figure 16 shows the overlay of the old and the new NNs for the two figures of merit. For the accuracy, the larger the better. For the loss, the smaller the better. The new NN is better than the old NN for both figures of merit.

4 Results

The shapes of the distributions of the truth and reco leading jet p_{T} observables, used as the NN output and input, are very similar, but not identical (especially in the first bins), as illustrated in Figure 17 for the training set (left) and for the testing set (right). From the jet p_{T} distribution, given the bin width, the distribution of the index of the jet p_{T} bin is built. The true output (generated, truth) vs input (reconstructed) is illustrated in Figure 18.

Once the NNs are trained, the output of the NNs is predicted. The bin with the largest probability is chosen as the predicted bin index. The distributions of the index of the jet p_{T} are overlaid in Figure 19 for the true output (blue) and the predicted unfolded output for the old NN (red) and the new NN (green). The new NN architecture is closer to the true output than the old NN. The old NN has some larger spikes and even some empty bins.

Another figure of merit that can be calculated is similar to the loss function. Instead of adding all values differences between the truth and the predicted truth in quadrature, the distribution of these differences is plotted. Both outputs are integers, as representing bin indices. Therefore the difference is also an integer. The greater the count at zero difference and the narrower the peak, the better. The entire training exercise is done also for other bin widths as well: 10, 20, 50, 100 GeV, as compared in Figure 20. In all cases, teh new NN performs better than the old NN. However, the smaller the bin width, the bigger the improvement of the new NN on top the old NN. To make finer predictions, it is ideal to have the bin width as small as possible. Then the new NN improves the most.

Finally, the 2D histogram migration matrices of the index of the jet p_{T} can be built, as illustrated in Figure 21. The closer to a diagonal matrix, the better. In all plots, the real true value is on the horizontal. In the top plots on the vertical there the reconstructed. This plot is used in the traditional unfolding method. In the middle plots, on the vertical there is the truth predicted by the old NN. It is noticed that some bins are empty. In the bottom plots on the vertical tehre is the truth predicted by the new NN. The migration matrix is closer to being a diagonal for the new NN than for the old NN, further supporting that the new NN training. is better than the old NN training.

5 Conclusions

In this report an unfolding study of the unfolding with a machine learning method for the leading jet p_{T} in the $t\bar{t}$ $e-\mu$ analysis is presented. The question to answer is given the reconstructed leading jet p_{T} from 0 to 500 GeV, with 20 GeVbins, in what bin falls the truth (generated) leading jet p_{T}. Since there are a finite (25) number of possible answers, it is a classification problem, which can be solved by training of and inferring from an artificial neural network. The NN training is done in TensorFlow via Keras in Python. A method and code example using toy data was followed, and adaptated for the ATLAS simulated data. The NN architectureis optimised and hyper-parameters are fine-tuned to maximize the accuracy values and minimize the loss values in the test sample. The chosen NN performs better than the NN choice suggested in the toy data example.

The project lasted for eight weeks. Given more time, several improvements or further developments are possible. Training one NN is only the first (zeroth) step of a ML-based unfolding method. The performance would improve more NNs are trained. Only one simulation file is used. But using more simulated data in training and testing contains more events and more leading jets, leading to a better training. Only one input variable is used, but several other variables can be added in the ML method, unlike in the migration matrix method. A possible extra variable can be the leading jet η. A further possible improvement is that instead of a k-fold=2 (using training and test each at 50% of the events), a larger k-fold to be used (*e.g.* 5).

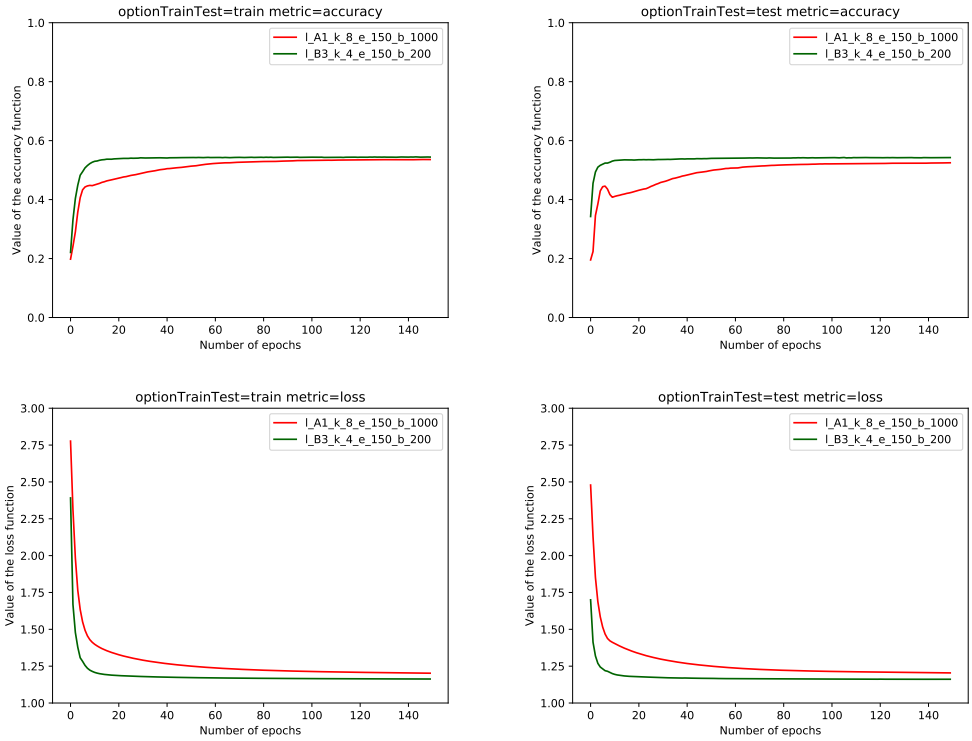

Figure 16: Overlay with old and new NN for the accuracy (top) and loss (bottom). Train (left) and test (right).

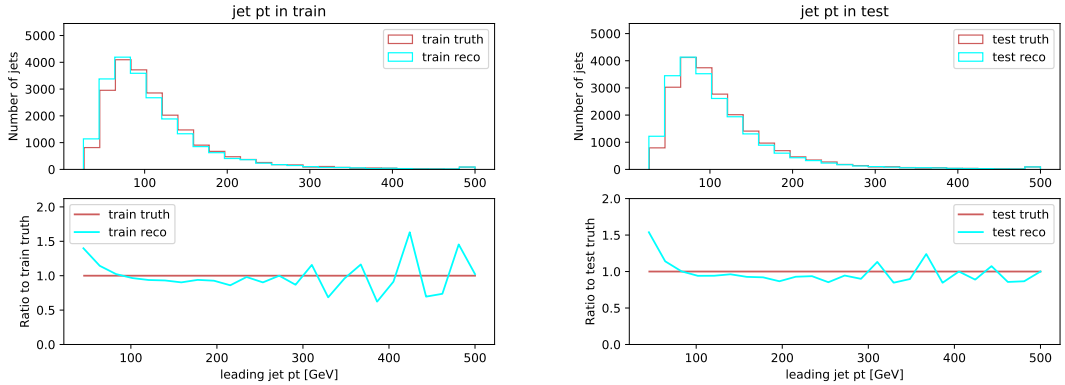

Figure 17: Overlay with ratio pad of the truth and reco leading jet p_{T}. Train (left) and test (right).

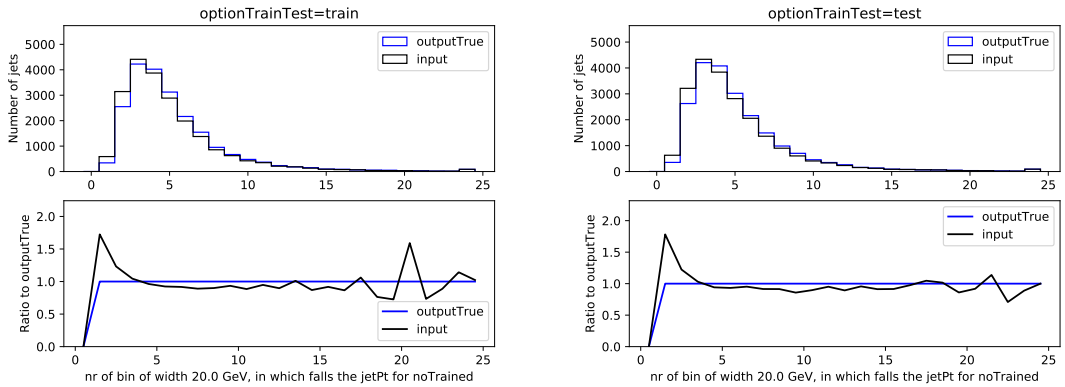

Figure 18: Overlay with ratio pad of the number of the bin of the truth and reco leading jet p_{T}. Train (left) and test (right).

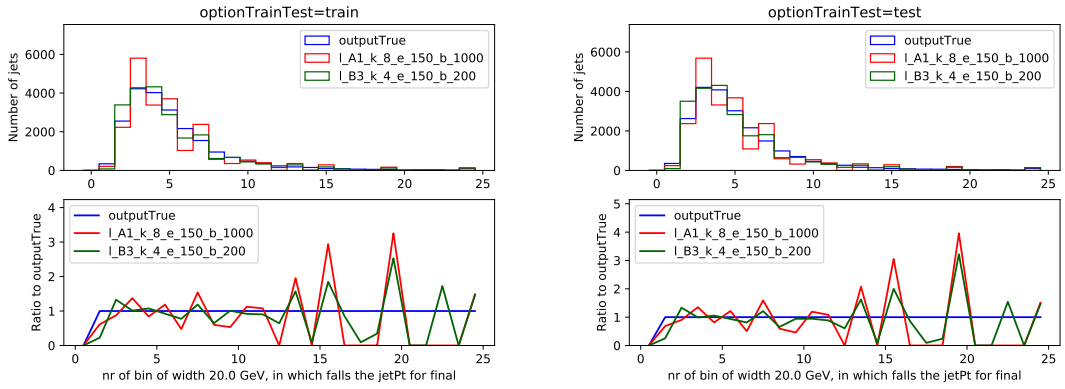

Figure 19: Overlay with ratio pad of the number of the bin of the truth (blue), the predicted truth by the old NN (red) and the predicted truth by the new NN (green). Train (left) and test (right).

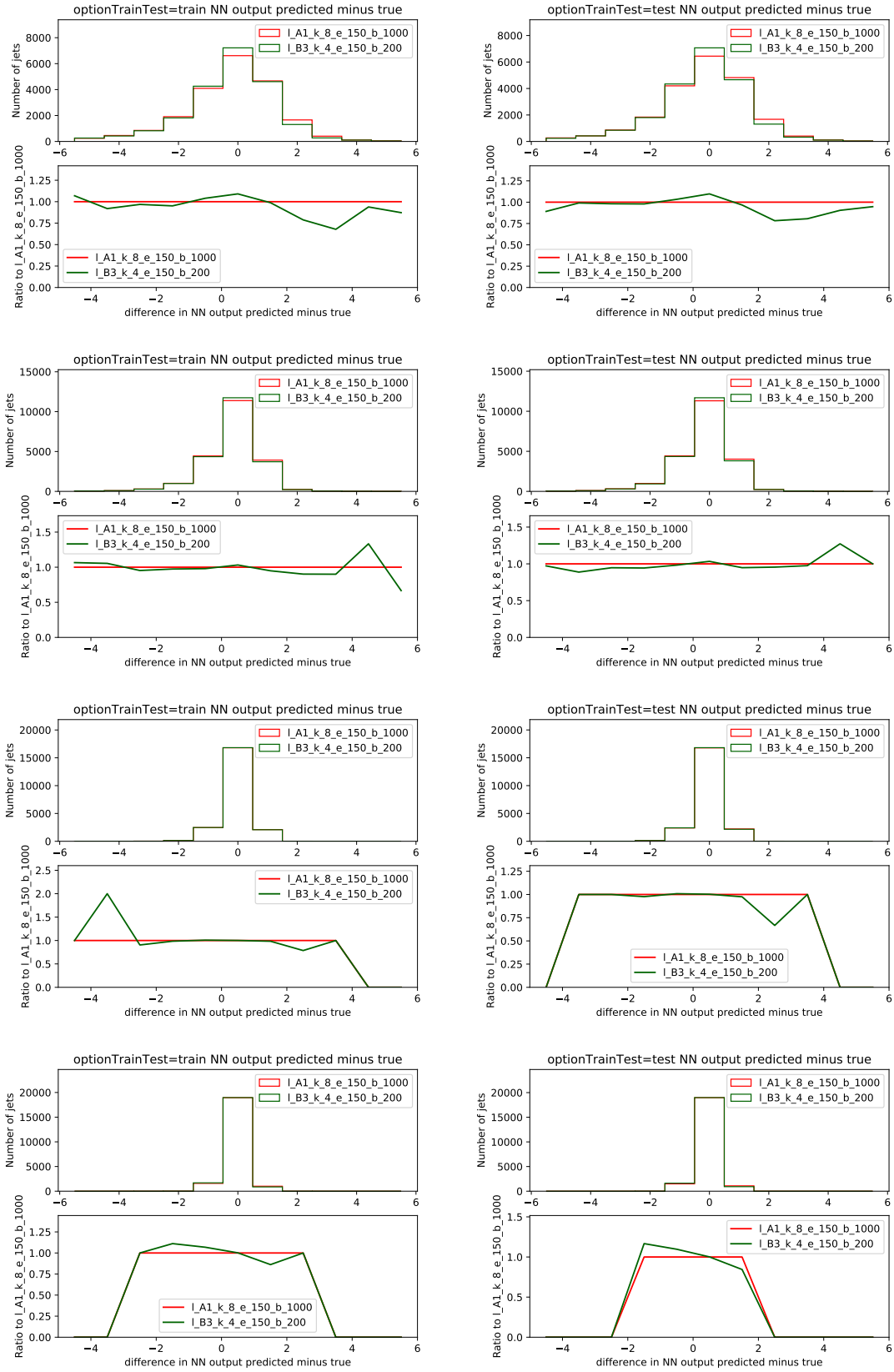

Figure 20: Overlay with ratio pad of the difference between the predicted bin index and the correct bin index, for the old NN (red) and the new NN (green). From top to bottom, the bin widths of 10, 20, 50, 100 GeV. Train (left) and test (right).

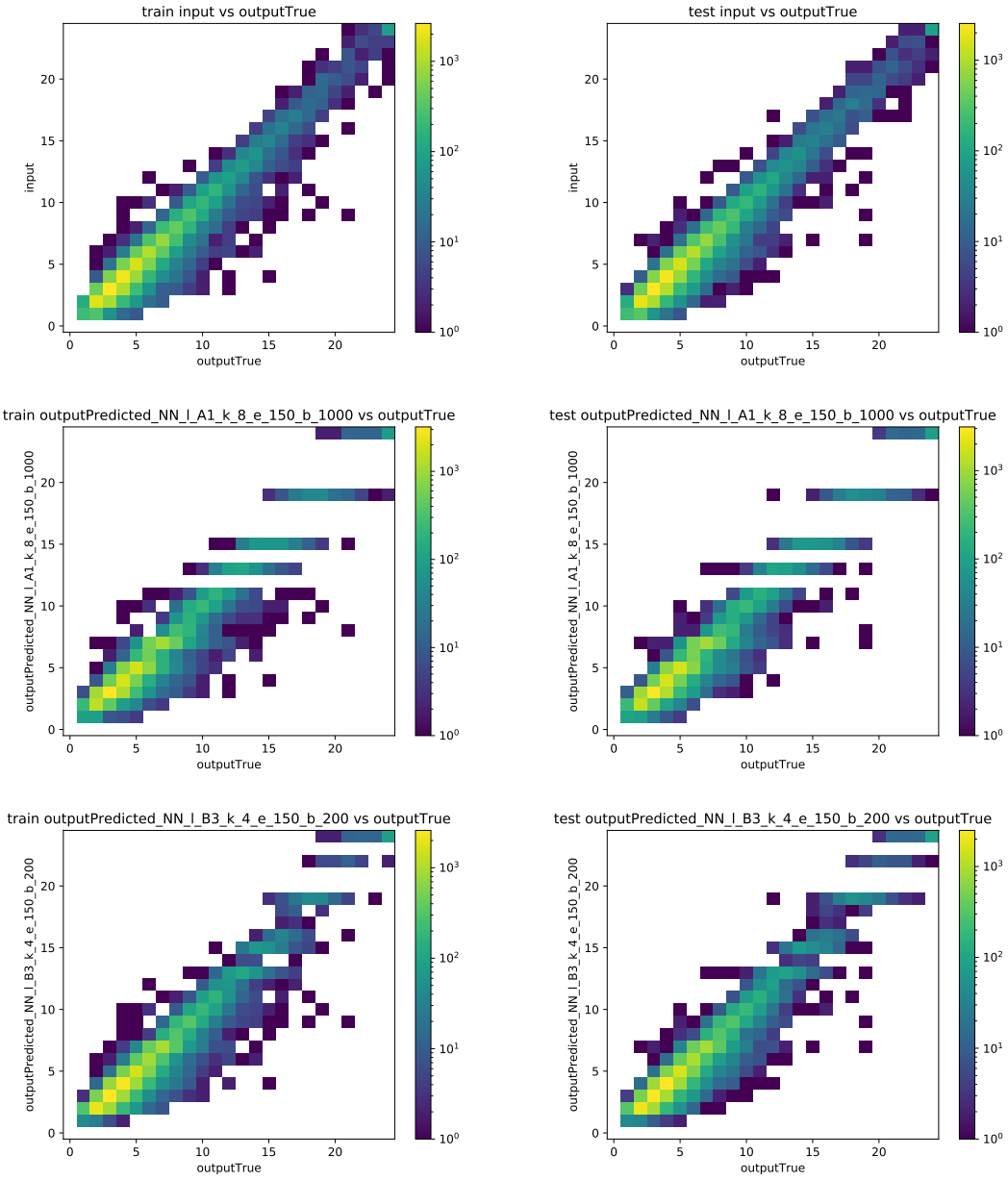

Figure 21: 2D histogram representing migration matrices of the bin indices of the jet p_{T}. In all plots on the horizontal there is the true output values. On the vertical from top to bottom there is the input (reco), the predicted output with the old NN and the predicted output with the new NN. Train (left) and test (right). Since new NN gives a migration matrix closer to diagonal and is better than the old NN.

Acknowledgements

I would like to express my gratitude to DESY for the opportunity to take part in the Summer School program at DESY Zeuthen for the summer of 2019. The particle physics lectures are very enriching. I am grateful to the ATLAS group at DESY for having hosted me for a eight week research project on machine learning unfolding in the $t\bar{t}$ $e-\mu$ analysis. I would like to thank my supervisors, prof. Thorsten Kuhl and postdoc Yichen Li, for introducing me to this exciting topic and their great supervision. I am also thankful for their help to the PhD students Matthieu Robin and Martin Habedank I feel also lucky to have had amazing colleagues from many countries at the summer school, with whom I had a lot of fun and will keep in contact in the future.

References

[1] *Measurement of jet activity produced in top-quark events with an electron, a muon and two b-tagged jets in the final state in pp collision at s=13 TeV with the ATLAS detector*, **arXiv**: 1610.09978v2, `https://arxiv.org/pdf/1610.09978.pdf`

[2] The ATLAS experiment, `https://atlas.cern`

[3] The $t\bar{t}$ $e-\mu$ ROOT file used `/afs/cern.ch/user/l/lciucu/public/data/MLUnfolding/user.yili.18448069._000001.output.sync.root`

[4] , Stefan Schmitt, Daniel Brizger, *Slides Unfolding in High Energy Physics at the Terascale statistics school 2014*, `http://www.desy.de/~sschmitt/talks/UnfoldStatSchool2014.pdf?fbclid=IwAR3CB3CwJOQchKfbZRIg6ktRf6DbI5KTD5cdtBbNHS1lKb-q3yrwagyua5w`.

[5] Yichen Li, *ZUnfold framework, an unfolding framework written at Zeuthen*, `http://www.desy.de/~liyichen/Unfolding.pdf`

[6] Alexander Glazov, *Machine learning as an instrument for data unfolding*, **arXiv**: 1712.01814v1, `http://inspirehep.net/record/1641082`

[7] Alexander Glazov, *Toy data example for machine learning as an instrument for data unfolding using TensorFlow via Keras in a Python Jupyter Notebook*, `https://github.com/aglazov/MLUnfold/blob/master/Unfold.ipynb`

[8] The TensorFlow machine learning library, `https://www.tensforflow.org`

[9] The keras machine learning library, `https://keras.io`

[10] The Numerical Python library, `https://www.numpy.org`

[11] The SWAN server at CERN, `https://swan003.cern.ch`

[12] The singularity command to run the ML environment,
`singularity exec '/cvmfs/unpacked.cern.ch/registry.hub.docker.com/atlasml/ml-base:latest'bash`

[13] The uproot package to read .root files in numpy arrays in Python, `https://github.com/scikit-hep/uproot`

[14] Joshua Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*, 2017

[15] Andrew Ng, *Machine Learning. Coursera online course*, URLhttps://www.coursera.org/learn/machinelearning

[16] Adam Gibson, Josh Patterson, Deep Learning, book published at O'Reilly, https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html

[17] TheFork.com, *Activation functions*, https://theffork.com/activation-functions-in-neural-networks/

[18] Uniqtech, *Understand the Softmax Function in Minutes*, article on Medium.com, https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

[19] Jason Brownlee, *A Gentle Introduction to the Rectified Linear Unit (ReLU)*, https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/