FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Large-Scale Data Analysis for Higgs Boson Mass Reconstruction in ttH Production
Student:	Bc. Petr Urban
Supervisor:	doc. Dr. André Sopczak
Study Programme:	Informatics
Study Branch:	System Programming
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2019/20

Instructions

1) Study the current IEAP's code for the Higgs boson analysis implemented on the CERN ATLAS experiment.
2) Develop and extend the existing Higgs boson analysis code to implement a mass reconstruction algorithm.
3) Test the code on existing simulated Higgs boson events and optimize the mass reconstruction based on a neural network to resolve the combinatorics of the reconstructed objects, and take into account the contribution from particles which escaped detection. Calibrate the mass reconstruction based on the known simulated mass.
4) Apply the developed algorithm to the actual recorded data, and determine the Higgs boson mass in the provided data set recorded by the ATLAS experiment, and determine the uncertainty on the measurement.
5) Implement the reconstructed Higgs boson mass as a new parameter in order to increase the Higgs boson detection significance (increase the ratio of signal over background events).

References

Will be provided by the supervisor.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 8, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Large-Scale Data Analysis for Higgs Boson Mass Reconstruction in ttH Production

Bc. Petr Urban

Department of Theoretical Computer Science
Supervisor: doc. Dr. André Sopczak

May 8, 2019

Acknowledgements

I would like to thank my supervisor doc. Dr. André Sopczak. He was willing to help me whenever I had a question or was stuck during my research. He also contributed by important remarks and recommended quality sources.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on May 8, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Urban, Petr. *Large-Scale Data Analysis for Higgs Boson Mass Reconstruction in ttH Production*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato práce se zabývá problémem rekonstrukce klidové hmotnosti částice známé jako Higgsův boson pomocí technik strojového učení – neuronových sítí. Je zaměřena na rozpadový kanál $2\ell SS+1\tau_{had}$. V první části je vysvětlen problém rekonstrukce klidové hmotnosti a popsány některé přístupy k jeho řešení. Další část popisuje fázi rekonstrukce klidové hmotnosti všech částic pomocí exaktních výpočtů a algoritmů. Poté jsou použita data získaná během rekonstrukce ostatních částic $t\bar{t}H$ systému. Na různě velkých datasetech, je natrénováno a otestováno několik neuronových sítí pro predikci/odhad klidové hmotnosti Higgsova bosonu na úrovni simulace. Nakonec jsou připraveny neuronové sítě pro odhad klidové hmotnosti na úrovni detektoru a pro rozlišení signálu a pozadí.

Klíčová slova ATLAS, CERN, Higgsův boson, klidová hmotnost, neuronové sítě

Abstract

This thesis deals with the problem of reconstruction of the invariant mass of the Higgs boson using machine learning techniques – neural networks. It focuses on the $2\ell SS + 1\tau_{had}$ decay channel. In the first part, the problem of mass reconstruction is explained and some used approaches to the problem are described. The next part describes the phase of reconstructing the invariant mass of all the particles using exact formulas and algorithms. Then, the data obtained during reconstruction of the other particles of the $t\bar{t}H$ system are used. Several neural networks are trained and tested on different datasets to predict/estimate the invariant mass of the Higgs boson on truth level. Finally, neural networks for estimating the invariant mass of the Higgs boson on detector level and distinguishing signal events from background are prepared.

Keywords ATLAS, CERN, Higgs boson, invariant mass, neural networks

Contents

List of Figures

List of Tables

Introduction

Apart from the basic elementary particles forming atoms, the Higgs boson is probably the best known particle by the public. It is a part of the Standard Model – a theory describing elementary particles and various forces between them that make up matter. And the Higgs boson is the particle that gives hope to many physicists to extend the model and make it more precise.

Although the existence of a new particle was predicted in 1960s it took more than 50 years to confirm the hypothesis, because the observation of the Higgs boson is complicated. One reason for this long waiting is that the production of the Higgs boson requires high energy proton-proton collisions and the Large Hadron Collider, where such collisions can be achieved, was put in the operation relatively recently (2008).

However, despite all these difficulties, in 2012 two experiments (ATLAS and CMS) at LHC recorded a new particle which was later identified as the Higgs boson. Since then many physicists have been focused on its research and believe that its mass is around 125 GeV/c^2.

The goal of this work is to train a neural network that will be able to reconstruct the invariant mass of the Higgs boson from the final products of its decay and distinguish signal events from background data in the so-called $t\bar{t}H$ production. That should enable physicists more detailed and precise studies of the collisions, which might help to get more information about the properties of the Higgs boson and the Standard Model.

In the first chapter of this thesis, information about the Higgs boson and ATLAS experiment is provided. The second chapter is focused on the theory of neural networks which will be used for developing the invariant mass reconstruction algorithms. The third chapter deals with related researches and studies about reconstructing the invariant mass of other particles or working with similar data. Finally, the last chapter presents the results – a neural network estimating the invariant mass of the Higgs boson and description of other models which will be later used for distinguishing the signal events from background data.

Chapter 1

Searching for the Higgs boson

The Higgs boson is an elementary particle of the standard model of particle physics. It is important, because it explains the mass characteristics of other particles.

Since its prediction in 1964 [21, 6], the Higgs boson had not been observed until 2012. Because of its short lifetime, it cannot be observed directly but it is possible to detect products (quarks and leptons) of its decay and reconstruct its characteristics from these products.

In July 2012, the ATLAS and CMS experiments at CERN's LHC announced the observation of a new particle with characteristics of the Higgs boson and confirmed the half century old theory[1].

1.1 Standard model

The Standard model of particle physics is the theory describing and classifying all known elementary particles. In addition, it describes the electromagnetic forces and strong and weak interactions between the particles (see figure 1.1).

The known elementary particles are:

quarks — three pairs of quarks (up-down, charm-strange and top-bottom),

leptons — three leptons (electron, muon, τ) with associated neutrinos,

bosons — force carriers responsible for particle interactions (W boson, Z boson, photon, gluon and Higgs boson).

[1] Next year (2013) scientists Englert and Higgs were awarded the Nobel prize in physics for "the theoretical discovery of a mechanism that contributes to our understanding of the origin of mass of subatomic particles, and which recently was confirmed through the discovery of the predicted fundamental particle, by the ATLAS and CMS experiments at CERN's Large Hadron Collider" [28, 6].

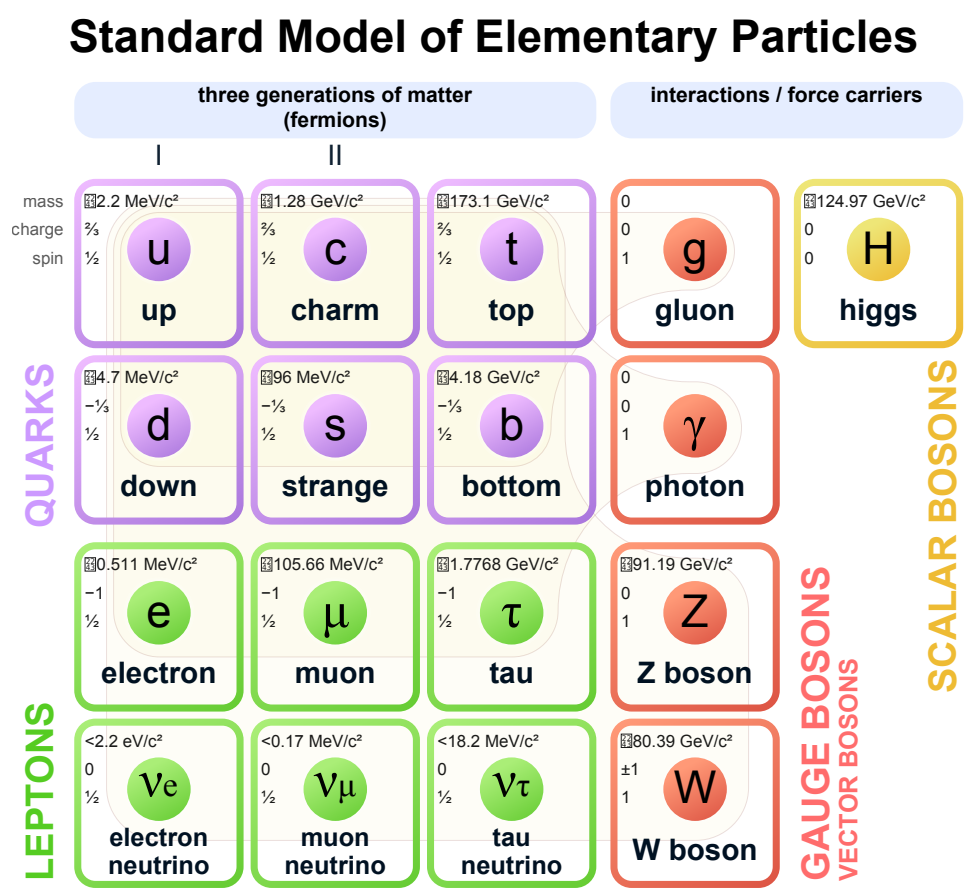


Figure 1.1: Standard model of particle physics.
Source: https://en.wikipedia.org/wiki/Standard_Model

1.2 CERN

The European Organization for Nuclear Research, established in 1954, located near Geneva is a research organisation mainly known for operating a network of particle accelerators, a particle decelerator and especially Large Hadron Collider (LHC).

Many detectors are part of this network, notably CMS and ATLAS detectors, where the Higgs boson was observed for the first time. You can see the scheme of the whole complex in figure 1.2.

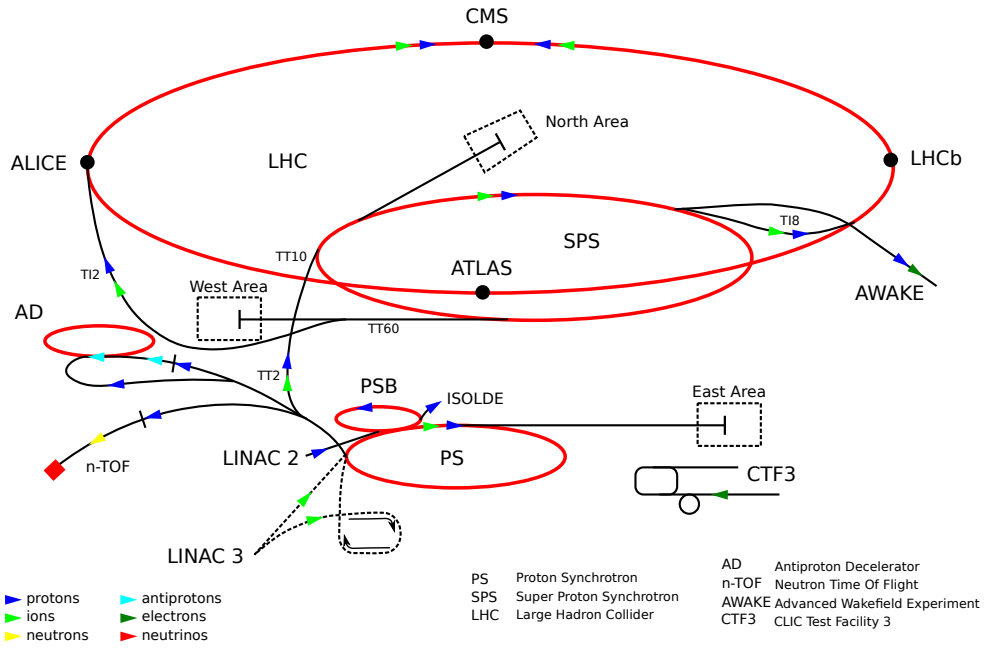


Figure 1.2: CERN complex.
Source: https://en.wikipedia.org/wiki/CERN

1.3 ATLAS Detector

The ATLAS detector [39] is a multipurpose particle detector build to detect proton collisions from the Large Hadron Collider (LHC).

1.3.1 Inner Detector

The inner detector is the first part of ATLAS to see the decay products of the collisions, so it is very compact and highly sensitive. It consists of three different systems of sensors all immersed in a magnetic field parallel to the beam axis. The Inner Detector measures the direction, momentum, and charge of electrically-charged particles produced in each proton-proton collision [14].

1.3.2 Calorimeter

Calorimeters measure the energy a particle loses as it passes through the detector. It is usually designed to stop entire or "absorb" most of the particles coming from a collision, forcing them to deposit all of their energy within the detector.

Electromagnetic calorimeters measure the energy of electrons and photons as they interact with matter. Hadronic calorimeters sample the energy of hadrons (particles that contain quarks, such as protons and neutrons) as they interact with atomic nuclei [13].

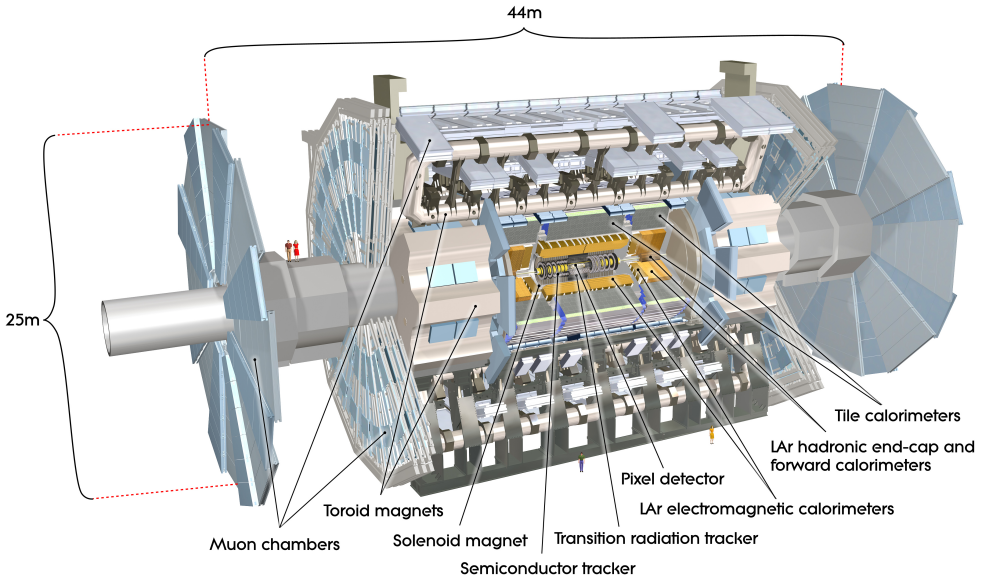


Figure 1.3: ATLAS scheme.
Source: https://cds.cern.ch/images/CERN-GE-0803012-01

1.3.3 Muon Spectrometer

Muons are particles that usually pass through the Inner Detector and Calorimeter undetected. The muon spectrometer identifies and measures the momenta of muons [16].

1.3.4 Magnets

The magnet system of ATLAS bends particles around the various layers of detector systems, making it easier to contain the tracks of particles. The main sections of the magnet system are: Central Solenoid Magnet, Barrel Toroid and End-cap Toroids [15].

1.4 Decay channels

Decays of the Higgs boson can happen in seven ways during $t\bar{t}H$ production. Those are represented by the final state of the system and are called decay channels. Each channel is characterized by its decay products – the number and flavour of charged leptons and number of hadronically decaying τ.

$1\ell + 2\tau_{had}$ one light lepton
two hadronically decaying τ lepton candidates,

$2\ell SS + 1\tau_{had}$	two same-charge light leptons one hadronically decaying τ lepton candidate,
$2\ell OS + 1\tau_{had}$	two opposite-charge light leptons one hadronically decaying τ lepton candidate,
$3\ell + 1\tau_{had}$	three light leptons one hadronically decaying τ lepton candidate,
$2\ell SS$	two same-charge leptons zero hadronically decaying τ lepton candidates,
3ℓ	three light leptons zero hadronically decaying τ lepton candidates,
4ℓ	four light leptons zero hadronically decaying τ lepton candidates.

It can be easily seen (figure 1.4 from [1]) that these channels are mutually exclusive – it cannot happen that one event covers two or more channels.

The study of these channels should bring more insight into the Higgs boson characteristics and lead to improvement of the Standard model of particle physics.

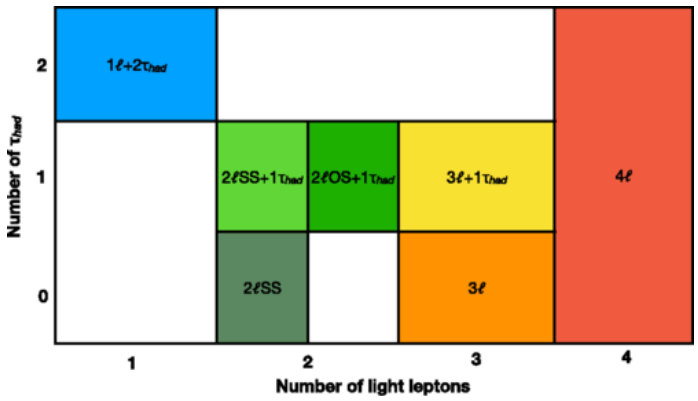


Figure 1.4: Decay channels.

Chapter 2

Artificial neural networks

In this chapter, the reader can find basic information about neural networks – their inspiration in biological world as well as formal definitions and maths that make them work. (activation function, loss functions or error backpropagation algorithm).

In the last section, various types of NN are presented.

2.1 Origin

Artificial neural networks (ANNs) are computational models inspired by biological brain and nervous system. Just like in case of the biological nervous system, the basic unit of ANN is called neuron.

Their similarity is shown in figure 2.1 and summarized in table 2.1. Each artificial neuron has n inputs. Each input has a weight which was assigned during learning phase of the network (or is random at the beginning). Weighted inputs are summed, passed into activation function which produces output value. This value becomes the input of selected neurons in next layer of the network or the output of the network in case of last layer.

Table 2.1: Comparison of a biological and an artificial neuron.

Artificial neuron	**Biological neuron**
Inputs	Dendrites
Outputs	Axons
Sum function	Neuron body
Activation function	Nucleus
Weights	Synapses strength

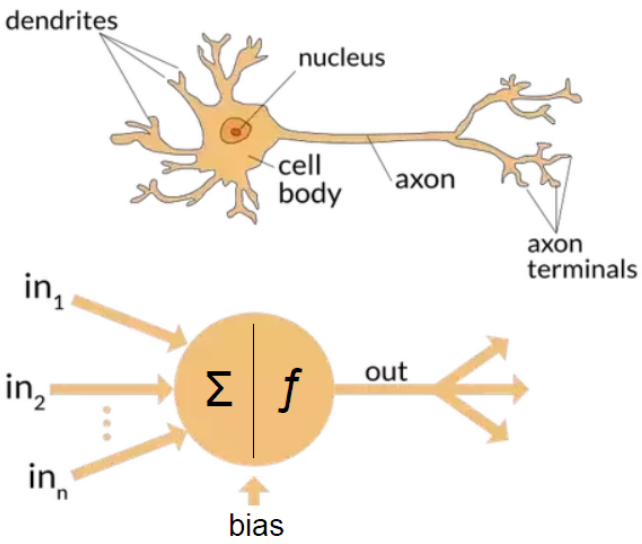


Figure 2.1: Comparison of biological and artificial neuron.
Source: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

Table 2.2: Comparison of a biological and an artificial network.

Artificial NN	Biological NN
specialized	able to learn different tasks
tree structure (except recurrent NN (see section 2.2.3))	complicated topology
different learning and evaluating phases	non-stop learning and evaluation
gradient descent (see algorithm 1) for adjusting weights	unknown

2.2 Formal definitions

In this section formal definitions of neuron, neural network and their parts can be found.

2.2.1 Neuron

Formally, neuron is a mathematical function with $n + 1$ inputs. Signal on the first input x_0 is usually assigned a value – bias (b). Remaining n inputs are actual inputs that depend on the outputs of neurons in previous layers.

Output of ith neuron can be computed:

$$y_i = \varphi\left(\sum_{j=1}^{n} w_{ij} * x_j + b\right) \tag{2.1}$$

where φ is the activation function. We denote its argument (term between parentheses) as ξ and call it potential of the neuron.

2.2.1.1 Activation function

Activation function defines the final output of the neuron. It takes the values computed by multiplying weights and inputs and maps them into the chosen range $((0 \dots 1), (0 \dots \infty), \dots)$. Follows a compendium (and a table 2.3 summarizing) of basic activation functions, more can be found at [18, 33].

Identity (linear) function is usually used in input layers of neural networks. It can either linearly transform the input or leave them unchanged (figure 2.2).

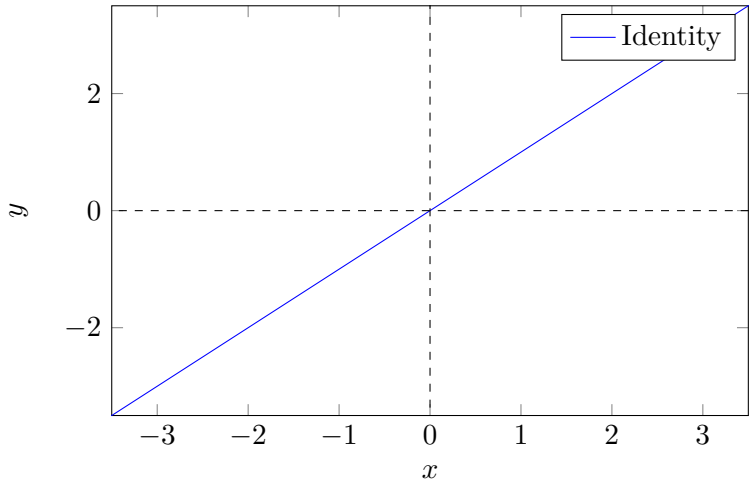


Figure 2.2: Identity/Linear activation function

Rectified Linear Unit is similar to linear function but allows only non-negative outputs. All values lower than zero become zero (figure 2.3).

Softplus is a smooth approximation to ReLU. It is differentiable at and around zero (figure 2.4).

Step function acquiring only two values ($0 and 1$) and its differentiable approximation – logistic sigmoid (figure 2.5).

2.2.2 Neural network

Formally, a neural network is a tuple (N, C, X, Y, w, t), where:

N is a set of neurons (non-empty),

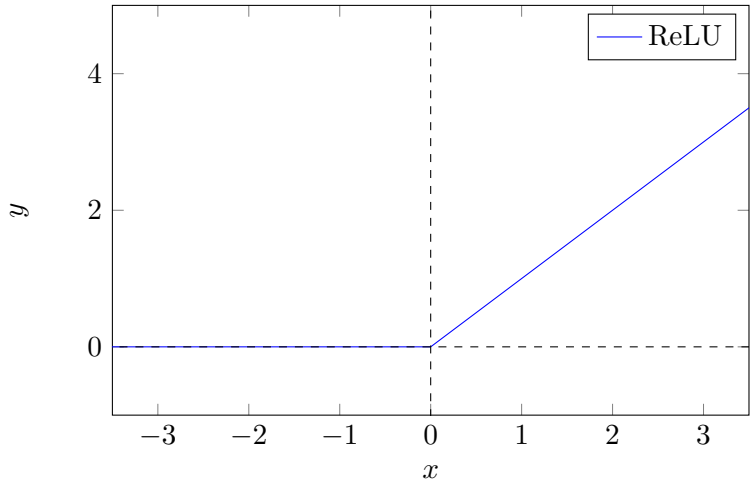


Figure 2.3: ReLU activation function

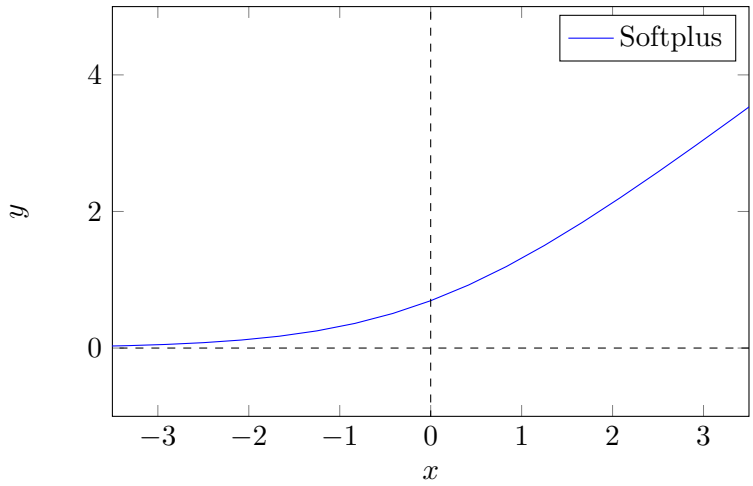


Figure 2.4: Softplus activation function

$E \subseteq N \times N$	is a set of oriented edges going between neurons (non-empty),
$X \subset N$	is a set of input neurons (non-empty),
$Y \subset N$	is a set of output neurons (non-empty),
$w \colon E \mapsto \mathbb{R}$	is a weight function,
$t \colon N \mapsto \mathbb{R}$	is a network bias function.

Usually, the set Y contains one node (one target variable), but in case of more than one target variable, the set can be arbitrarily large.

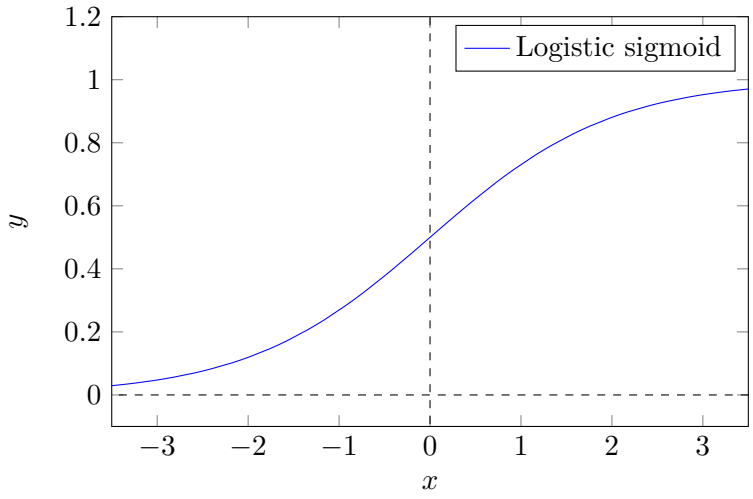


Figure 2.5: Sigmoid activation function

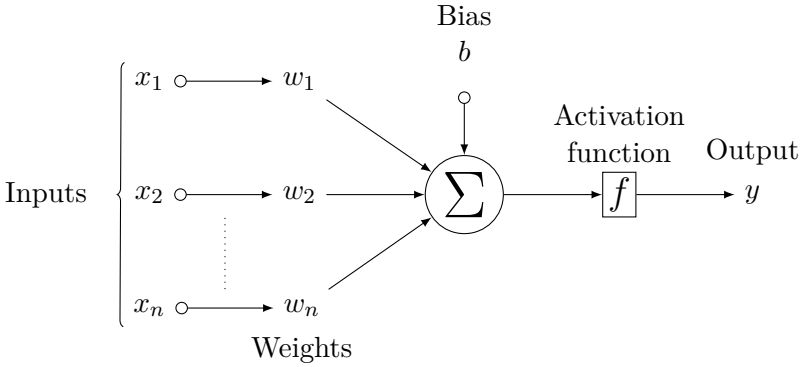


Figure 2.6: Neuron scheme.

2.2.2.1 Training

Training of neural network consist of a forward pass, when the neurons simply count their outputs according to equation

$$y(x_1, \ldots, x_n) = \varphi\Big(f(w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n)\Big). \tag{2.2}$$

Then the error is computed and the algorithm continues by doing the backward pass and weights in the whole network are adjusted.

The adjusting of weights comes when a batch of selected size is processed. The size of one batch affects memory used (larger batch – more memory) in the process, speed of training and accuracy of the gradient estimation – see sections 2.2.2.2 and 2.2.2.3.

The length of training is set in number of so called epochs. One epoch is

Table 2.3: Activation functions.

Name	Formula	Notes
Identity	$y = x$	No change to weighted inputs.
Rectifier (ReLU)	$y = \max(0, x)$	Zero (when $x \leq 0$) or linear.
Softplus	$y = \ln(1 + e^x)$	In some sources ([33]) also called SmoothReLU. This functions is smoother near $x = 0$.
Step function	$y = \begin{cases} 1 & if\ x \geq \theta \\ 0 & if\ x < \theta \end{cases}$	Output of neuron is 1 once a threshold θ is reached.
Sigmoid	$y = \frac{1}{1 + e^{-x}}$	Smoother (and differentiable) than step function.

one forward pass and one backward pass of all the input samples. It is easy to see, that with more epochs of training it takes longer time to finish, but on the other hand, it is possible to achieve better accuracy of the model.

2.2.2.2 Loss function

Once the output of the final neuron is known, two cases may occur:

1. the output (predicted value) matches the expected value (label) – no error,

2. the output (predicted value) differs from the expected value (label) – in that case, the error needs to be quantified.

Depending on whether the NN is solving regression or classification problem, different loss functions ($\boldsymbol{\mathcal{L}}$) are used.

In case of classification, one of the most used loss functions is Cross Entropy [20].

Equation 2.3 is generic formula for computing entropy for single sample in dataset with K classes, where $y^{(k)}$ is the real probability that the sample be-

longs to class k, $\hat{y}^{(k)}$ is the same probability computed by the neural network. ϵ is usually added to ensure the term in logarithm is not equal to zero.

$$\sum_{k}^{K} -y^{(k)} \log\left(\hat{y}^{(k)} + \epsilon\right) \tag{2.3}$$

For binary classification, we get the following equation (2.4).

$$-\Big(y \log(\hat{y} + \epsilon) + (1 - y) \log(1 - \hat{y} + \epsilon)\Big) \tag{2.4}$$

And for the entire dataset:

$$\boldsymbol{\mathcal{L}} = -\frac{1}{n} \sum_{i=1}^{n} \Big[y^{(i)} \log(\hat{y}^{(i)} + \epsilon) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)} + \epsilon)\Big] \tag{2.5}$$

In case of regression problems, where the desired output is a value from $\mathbb{R}$, mostly Mean Squared Error (MSE, equation 2.6) or Mean Squared Logarithmic Error (MSLE, equation 2.7) are used (y is the actual value, $\hat{y}$ is the predicted value, n is number of inputs).

MSLE is usually used when we do not want to penalize huge differences in the predicted and the actual values when both predicted and true values are huge numbers [20].

$$\boldsymbol{\mathcal{L}} = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 \tag{2.6}$$

$$\boldsymbol{\mathcal{L}} = \frac{1}{n} \sum_{i=1}^{n} \big(\log(y^{(i)} + 1) - \log(\hat{y}^{(i)} + 1)\big)^2 \tag{2.7}$$

Another possible function (especially when dealing with large values) is Mean Absolute Percentage Error (MAPE, equation 2.8).

$$\boldsymbol{\mathcal{L}} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{2.8}$$

For further information about other loss functions see [20, 5].

2.2.2.3 Backpropagation

Now, when we have metrics to compute the error, the neural network can learn from its mistakes (supervised learning). It will do so by adjusting weights by algorithm called error backpropagation, which used method called gradient descend [32].

Backpropagation algorithm traverses the graph representing the neural network backwards (layer by layer), computing loss gradient in each layer (see figure 2.7) and adjusting the weights of inputs for each neuron (see algorithm 1).

How much the weights are adjusted is influenced by hyper-parameter called learning rate. If the learning rate is low, the algorithm converges slowly, but the risk of missing any local minima is also lower. So, proper setting of this parameter is very important. However it is hard to do so analytically, so common practice is to try more values and select the best one.

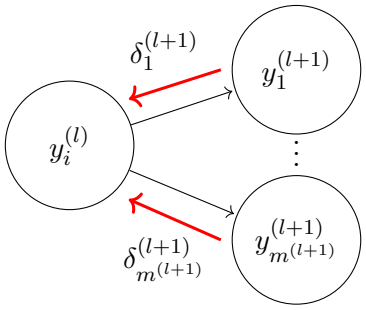


Figure 2.7: Backpropagation of errors through the network.

.

Algorithm 1: Error backpropagation algorithm for a layered neural network represented as computation graph $G = (V, E)$ [36].

1. For a sample (x_n, y_n^*), propagate the input x_n through the network to compute the outputs $(v_{i_1}, \ldots, v_{i_{|V|}})$ (in topological order).
   (a) Given a topological sort $V = (v_{i_1}, \ldots, v_{i_{|V|}})$, sequentially compute the layers' outputs, also denoted by v_{i_j}.
   (b) Then $y(x_n; w) = v_{i_{|V|}}$ is the network's output.
2. Compute the loss $\mathcal{L}_n := \mathcal{L}(v_{i_{|V|}}, y_n^*)$ and its gradient
$$\frac{\partial \mathcal{L}_n}{\partial v_{i_{|V|}}}. \tag{2.9}$$
3. For each $j = |V|, \ldots, 1$ compute
$$\frac{\partial \mathcal{L}_n}{\partial w_j} = \frac{\partial \mathcal{L}_n}{\partial v_{i_{|V|}}} \prod_{k=j+1}^{|V|} \frac{\partial v_{i_k}}{\partial v_{i_{k-1}}} \frac{\partial v_{i_j}}{\partial w_j}. \tag{2.10}$$
   where w_j refers to the weights in node i_j.

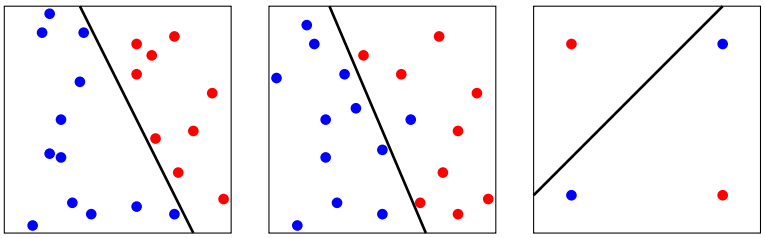


Figure 2.8: Perceptron possibilities, a) A linearly separable problem, b) A nonlinearly separable problem, c) The XOR problem.

2.2.3 Types of networks

In this section, the basic types of NN are described. Beginning with the simplest "NN" called perceptron, continuing with its evolution multilayer perceptron and ending with specialized more complex convolutional networks.

2.2.3.1 Perceptron

Perceptron is the simplest neural network consisting from one neuron only. Due to its simplicity, its possibilities are limited to linearly separable problems – it creates a hyperplane[2] separating the two group of samples.

In figure 2.8 are three examples. First one (a) can be solved by perceptron without error, in second (b) perceptron could find a solution with only one miss-labeled sample but in third example (c) which shows XOR (exclusive or) problem with only four possible samples, perceptron always fails in separating the samples although it is an easy function/logical operation[3].

2.2.3.2 Multilayer perceptron

To distinguish data that are not linearly separable, we need multilayer perceptron (MLP)[4]. It is fully connected (each neuron from every layer is connected to all neurons in following layer) and consists of three types of layers:

Input — Neurons in input layer for "reading" input. They have linear activation function – they only pass the input data to first hidden layer.

Hidden — At least one in each MLP, neurons have nonlinear activation function.

[2]Hyperplane is a subspace whose dimension is one less than that of its ambient space. In case of 2-dimensional space, the hyperplane has only 1 dimension – it is a line.

[3]XOR in boolean logic conjunctive form: $(\bar{x} + \bar{y}) \cdot (x + y)$

[4]MLP is not a single perceptron with multiple layers but network with many perceptrons/neurons organized in layers.

Output Output layer of at least one neuron.

In figure 2.9 you can see a multilayer perceptron with one hidden layer.

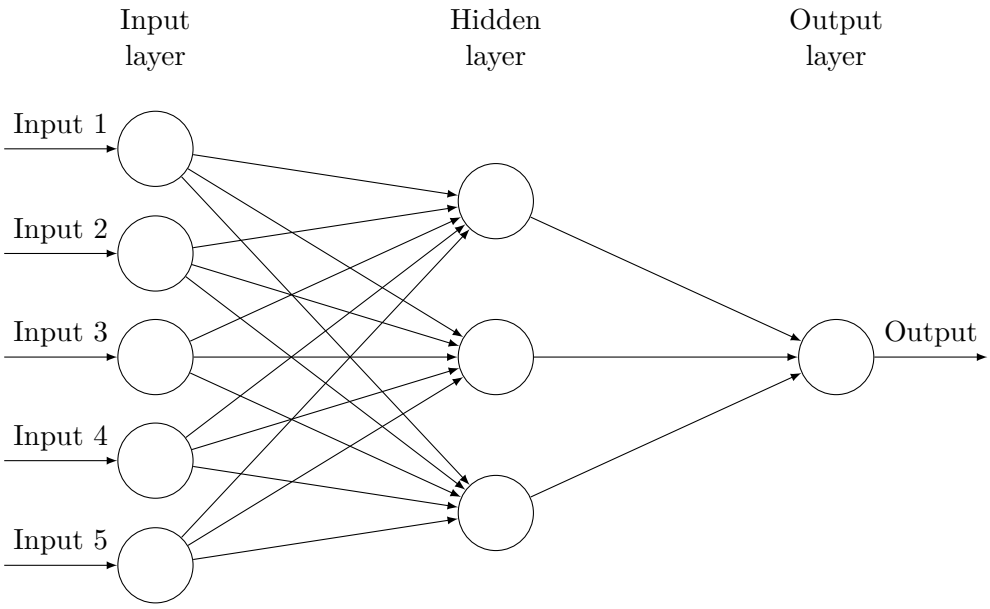


Figure 2.9: Multilayer perceptron.

2.2.3.3 Recurrent neural network

Recurrent neural network (RNN) differs from FNN by allowing loops – connection going to the same neuron or previous layers. It is mostly used for tasks, where context is important – text recognition (current word depends on the rest of the sentence). For more information about RNN see [37, 40].

2.2.3.4 Convolutional neural network

Convolutional neural networks (CNNs) are used for visual imagery (image recognition). They are similar to MLPs, but use more types of layers []:

convolutional Most important part (hence the name) of CNN consisting of filters – neurons with small receptive field covering only few pixels of the image. During the forward pass, the filter is convolved across ("scans") the whole image (previous layer). This way, feature maps are produced.
The idea behind this concept is, that neighbouring pixels are

related and should be processed together, whereas processing pixels from opposite parts of the image is wasting CPU time.

ReLU — Removes negative values (see section 2.2.1.1).

pooling — This layer splits 2-dimensional input into non-overlapping rectangles and chooses the maximum value (in case of max pooling). Also called downsampling. The idea here is, that the exact location (pixel) of a feature is less important than its rough location (left upper corner, middle, etc.).

fully connected — Processes the output in similar way as non-convolutional MLP and classifies the image.

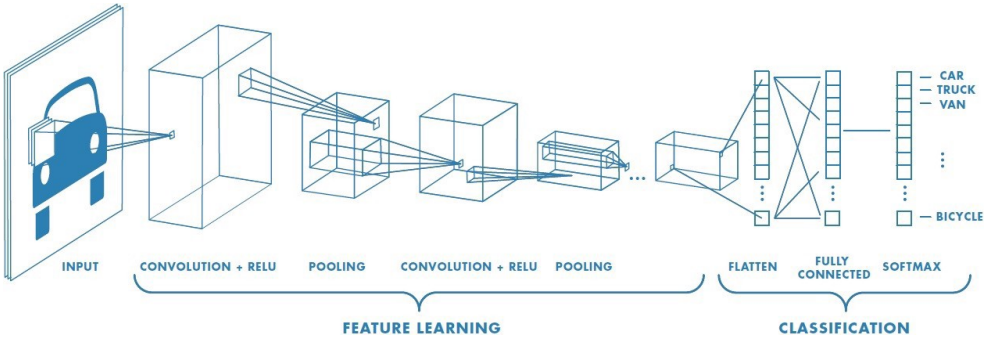


Figure 2.10: Convolutional NN – example.
Source: https://cdn-images-1.medium.com/max/2400/1*vkQ0hXDaQv57sALXAJquxA.jpeg

These layers (except the fully connected in the end) are usually chained (see figure 2.10). The number of repetitions can depend on the size of the original image the layer is created for and also on the details it should be able to recognize. Other important hyper-parameters are the size of the receptive field in convolutional layer,

For more information about CNNs see [22, 37].

2.2.3.5 Other types

The selection of described NN types is not taxative, but covers the main concepts which many other types of NN build on. You can get more information about other types of NN in literature mentioned earlier in this chapter or here: [37, 26].

Chapter 3

Related research

In this chapter, the reader can find information about invariant mass, different ways of its reconstruction both in terms of different particles and different methods.

3.1 Invariant mass

Particle physicists use the word "mass" to refer to the quantity (sometimes called "rest mass") which is proportional to the inertia of the particle when it is at rest – it is the part of the mass that is independent of the motion (speed, momentum) of the particle [17, 24].

It is measured in GeV/c^2 and calculated from the energy of the particle (E), its momentum ($\vec{p}$) and the speed of light (c) using the following formula:

$$m = \frac{1}{c^2}\sqrt{E^2 - p^2c^2} \tag{3.1}$$

With this equation, the invariant mass of arbitrary detectable particle can be easily computed – when the particle is detected, we know its momentum and energy and the speed of light is a constant.

But not all particles can be observed (which is the case of the Higgs boson), so their masses have to be reconstructed from products of their decay. These particles' vectors are first added together and then the equation 3.1 is applied to the new composite vector.

However, in the case of the Higgs boson, even some of the decay products (neutrinos) cannot be detected, so when the equation 3.1 is used, the computed invariant mass is lower than real invariant mass because of the missing (undetected) energy and to get a better estimation, more complex methods have to be used.

3.2 Other particles

As was pointed above, the problem of the invariant mass reconstruction is not limited only to the Higgs boson. It is a feature of every elementary particle, so it has been studied before on other particles.

In [4] the invariant mass of τ lepton pair is reconstructed using neural networks. Apart from using basic characteristics of decay products (p_T, η, ϕ, E) as the input for the neural network, they also use precomputed invariant masses of the final products and information about missing energy.

Conventional approach ([3, 25, 35]) is making use of the angular distance of the two particles coming from mutual parent particle:

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} \tag{3.2}$$

This is commonly used to tell, which of two or more particles of the same type comes from the same parent particle as the "sibling" particle[5] and which one is from other part of the decay system.

3.3 Other channels

Since the Higgs boson has multiple decay channels, its invariant mass reconstruction has already been studied ([2]) despite its existence has been confirmed for relatively short time.

In [19] one of the channel ($Hb\bar{b}$) is studied. In this case, the reconstruction of the invariant mass is easier, because b quarks are detectable, so it is not burdened by missing energy. Although this study deals with different channel and is using different machine learning technique (BDT), it still brings useful insight for studying the $2\ell SS + 1\tau_{had}$ channel.

3.4 $2\ell SS + 1\tau_{had}$ channel

The reserach of [35] focuses on the $H \to \tau\tau$ channel and studies the kinematics of the events – details of the decay process of the particles. Also, it deals with the composition of the missing energy (the ratio of contribution from the three neutrinos coming from the Higgs boson and the one coming from the antitop quark).

Apart from the missing energy, the correct assigning of leptons, quarks (non-b jets) and bottom quark (b jets) is an important problem to solve – more on this topic in section 4.4.

For better understanding, figure 3.1 shows a Feynman diagram of the $2\ell SS + 1\tau_{had}$ channel. Both top quarks are decaying into bottom quarks and

[5] Sibling particles are those sharing the same parent particle.

W bosons. In case of the hadronic top quark, it further decays into two quarks, in case of the leptonic top quark, it decays into a lepton-neutrino pair.

The other lepton of the same charge (requirement of the channel) comes from the Higgs boson ($H \rightarrow \tau\tau$). Since one τ must be hadronic (another channel requirement), it decays into two quarks. Remaining products of the $H \rightarrow \tau\tau$ system are three neutrinos.

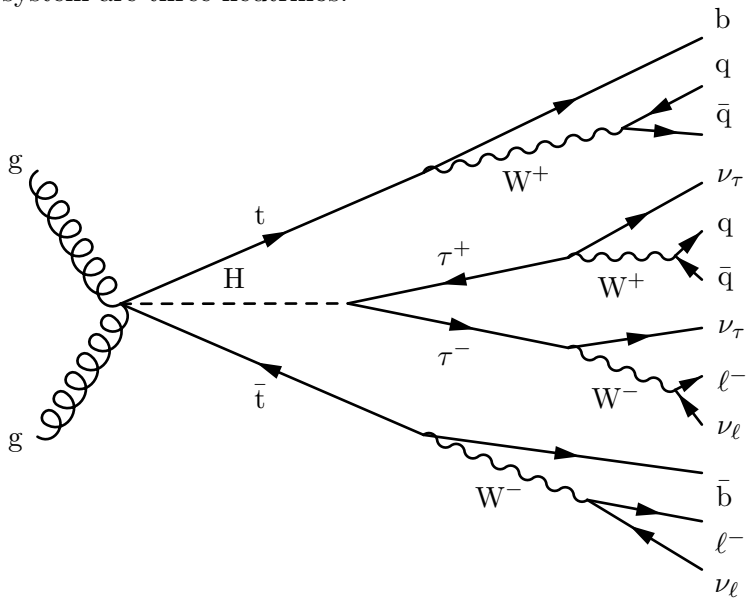


Figure 3.1: $t\bar{t}H$ system decay – $2\ell SS + 1\tau_{had}$ channel

3.5 Summary – feature selection

Based on the studies mentioned above [3, 25, 19], what seems to be the best approach is using precomputed invariant masses of final products (quarks and leptons) and do not add the detailed attributes (p_T, η, ϕ, E), because the invariant mass already contains this information[6].

Also, adding another feature – the angular distance of related particles (coming from same parent or sharing the same grandparent) can help improve the accuracy of the estimations.

Adding information about total energy missing in the system is also helpful.

Another good practice is precomputing not only the invariant masses of the final products, but also of the different parts of the decay system even in case missing energy.

[6]Invariant mass is computed using energy and momentum of the particle which can be obtained by transformations (see section 4.1.1.1) of p_T, η, ϕ, E.

Chapter 4

Implementation

This chapter describes the approaches used to analyse the datasets and in the end, the results are presented. Reader can also find pseudocodes of the most important algorithms here.

4.1 Software

Crucial software and its versions used:

ROOT	6.14.04
Python	3.6.4
Numpy	1.14.0
Pandas	0.22.0
TensorFlow	1.7.0
Keras	2.2.4

4.1.1 ROOT

ROOT is a data analysis framework developed by CERN [10], written mostly in C++. It is one of the most used software tools in particle physics. Some of its biggest advantages and the reasons for its firm position in the field are:

histograms Histograms are easy to create and work with as ROOT provides a large number of functions/methods to get the most of histograms [7].

root files In particle physics experiments, the detector often output large datafiles with hundreds of attributes/columns. But each scientist usually needs to work with only a small fraction of the columns.

If these data were saved in CSV or similar files, they would have to read the whole file and select only a small portion of columns. ROOT and its TTree class enables to load only specified columns efficiently [12].

intergation — As ROOT is written in C++, it can be used in C++ code, but it is also inegrated with other "data science languages" like R or Python (extension PyROOT) [8, 9].

In this work, ROOT (PyROOT) was only used in the first phase, when working with original datasets and selecting attributes/features which were later used for training the neural networks.

4.1.1.1 TLorentzVector

TLorentzVector is a four-vector class[7] used for description of momentum and energy of particles in a spherical coordinate system [11].

The primary method used to set these properties is $SetPxPyPzE$.

However, the detector output (and used dataset) does not contain these variables, but only transverse momentum (p_T), energy (E), azimuthal angle (ϕ) and pseudorapidity η, which is the spatial coordinate describing the angle (θ) of the particle relative to the beam axis (polar angle)[41].

To compute the cartesian momenta (p_x, p_y, p_z) from p_T, ϕ, η, these conversions need to be used:

- $p_x = p_T * \cos\phi$
- $p_y = p_T * \sin\phi$
- $p_z = p_T * \sinh\eta$

For convenience, the $TLorentzVector$ class contains also the method called $SetPtEtaPhiE$ and the conversions are done automatically while the object is being created. The function to compute the invariant mass is also part of this class (see equation 4.1).

$$m_{inv} = \sqrt{e * e - p_x * p_x - p_y * p_y - p_z * p_z} \tag{4.1}$$

4.1.2 Python

To ensure compatibility with packages listed below, Python version 3.6.4 was used.

Pandas, Numpy — Packages almost necessary for scientific computing with Python [29, 30].

[7] Class representing a vector consisting of four components.

Keras, TensorFlow — Keras is a popular high-level neural networks API. It is (among others) capable of running on top of TensorFlow – open-source software library developed by Google for dataflow programming and is used for machine learning applications – mostly neural networks [23, 38].

4.2 Data

The dataset contains 787 988 records – events. Each event describes one collision and subsequent decay of individual particles. For this description, there are 1338 attributes. However, most of them are not useful for analysing $2\ell SS + 1\tau_{had}$ decay channel (see 1.4).

4.3 Truth level

Truth level data are the easiest to work with. They are Monte Carlo samples generated by PYTHIA[8].

The decay structure is saved in several vectors with prefix `m_truth_`. What data are stored in the vectors used for analysis is explained in the following list:

`m_truth_pt` — Transverse momenta of the particles.

`m_truth_eta` — Pseudorapidity of the particles (see section 4.1.1.1)

`m_truth_phi` — Azimuthal angle of the particles.

`m_truth_e` — Energy of the particles.

`m_truth_pdgId` — Particle codes (see section 4.3.1).

`m_truth_barcode` — Unique IDs of the particles.

`m_truth_parents` — Barcodes of particles' parents.

`m_truth_children` — Vector of barcodes of particles' children.

4.3.1 Particle codes

To distinguish individual types of particles, the Particle Data Group (PDG[9]) particle codes are used [27, 31].

The particles that are important for the analysis have the following codes in the PDG standard:

[8] PYTHIA is a simulation program for high-energy physics events like elementary particle collisions [34].

[9] Particle Data Group is an international collaboration of particle physicists.

1	down quark,
2	up quark,
3	strange quark,
4	charm quark,
5	bottom quark,
6	top quark,
11	electron (lepton),
12	electron neutrino,
13	muon (lepton),
14	muon neutrino,
15	τ (lepton),
16	τ neutrino,
24	W boson,
25	Higgs boson,
111, 211	mesons (particles made up of two quarks),
− sign	all of the particles above have their antiparticles with the same PDG standart code with " − " sign.

Redrawing the Feynman diagram described in previous chapter (figure 3.1) and using these codes, we get:

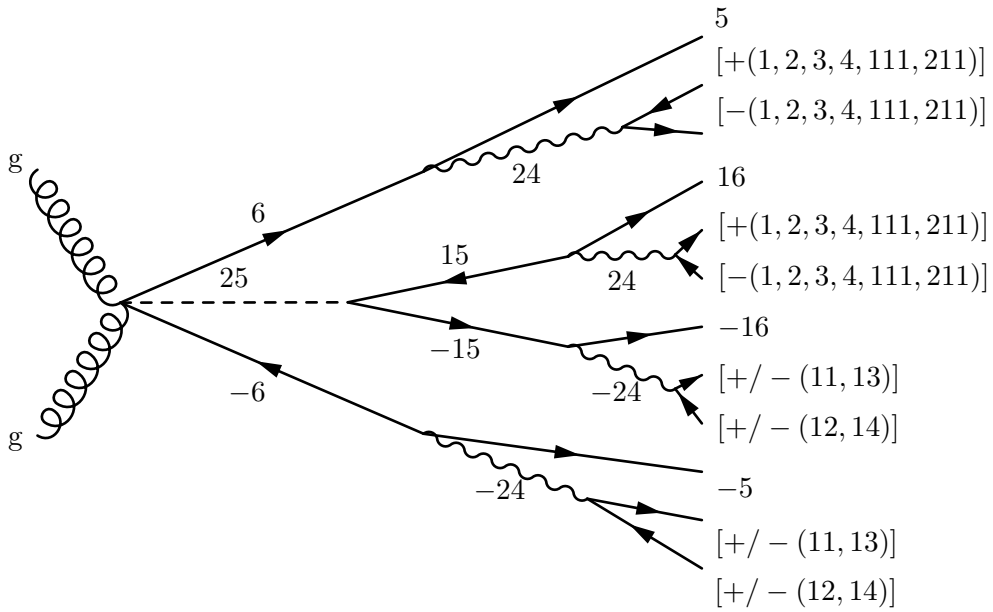


Figure 4.1: $t\bar{t}H$ system decay – $2\ell SS + 1\tau_{had}$ channel, with PDG codes.

4.3.2 Algorithm

The provided Monte Carlo data were stored in the root file in a tree-like way (each particle had its parent's barcode and non-leaf particles had list of their children's barcodes). To be able to work with the data, I created class `Node`, where I stored basic attributes of particles (`pt, eta, phi, e`) as well as their particle codes, its indexes in the original file and lists of their children.

Then I was able to find the particles I needed and build tree structures representing parts of the Feynman diagram. First, I go through the list of particles to find the root particle (top quark (6), antitop quark (-6) or Higgs boson (25), see 4.3.1 for details), which is saved into an object named *node*.

After finding the root particle, recursive method $find\_children$ (see algorithm 2) is called. As a result, a tree structure representing the particle life is saved in a tree with root in the *node* object.

With this tree structure, following tasks are much easier – computing the invariant mass of arbitrary node (particle). You can see the algorithm for this task here (3).

But before the invariant mass reconstruction algorithm is needed, we must make sure, that the event is part of $2\ell SS + 1\tau_{had}$ decay channel. To do so, method *check_channel* is used.

For the sake of simplicity, the code is not presented here but it can be found in source codes folder. It recursively traverses the tree of the Higgs boson decay and counts the number of quarks, leptons and neutrinos in its leaves.

Algorithm 2: Recursive method *find_children* for searching products of particle decay and building a tree structure representing the particle life.

Input : Vector $pdgID$, vector $barcode$, vector $children\_barcodes$, number of particles n
Output: Root of the decay tree – in object *node*

```
for child in children_barcodes[self.i] do
    for i from 0 to n − 1 do
        if child == barcode[i] then
            new_node := Node(pt[i], eta[i], phi[i], e[i], pdgID[i], i)
            new_node.find_children(pdgID, barcode, children_barcodes, n)
            self.children.append(new_node)
        end
    end
end
return self
```

Since one τ is supposed to decay hadronically and the other one leptonically, the needed numbers are:

- 1 lepton
- 2 quarks
- 1 lepton associated neutrino
- 2 Tau associated neutrinos

Similar procedure is applied to the top quark and anti-top quark branches.

top quark	Must decay into two quarks and one bottom quark.
anti-top quark	Must decay into one bottom quark, one lepton and one neutrino associated with the lepton.

Based on the selection criteria listed above and using the algorithms described, 13 673 events were selected for further analysis.

4.3.3 Dataset features

Based on the information summarized in chapter 3, these features were selected for each event:

`met_met, met_phi`	missing energy characteristics,
`r_w_jets`	distance between quarks coming from the W boson,

Algorithm 3: Method *reconstruct_mass* for computing invariant mass of the system from leaf nodes.

Input : Tree structure representing part of the Feynman diagram of the event ($self$)
Output: Invariant mass of the system

```
1  if self.children then
2  |  tlv := TLorentzVector()
3  |  flag := True
4  |  for child in self.children do
5  |  |  if flag then
6  |  |  |  tlv := child.reconstruct_mass() flag := False
7  |  |  end
8  |  |  else
9  |  |  |  tlv := tlv + child.reconstruct_mass()
10 |  |  end
11 |  end
12 |  return tlv
13 end
14 else
15 |  tlv := TLorentzVector()
16 |  tlv.SetPtEtaPhiE(self.pt, self.eta, self.phi, self.e)
17 |  return tlv
18 end
```

`r_w_b_jet`	distance between the W boson and the bottom quark,
`r_higgs_jets`	distance between quarks in the Higgs boson branch,
`top_q_1_mass`	invariant mass of a quark in the top quark branch,
`top_q_2_mass`	invariant mass of a quark in the top quark branch,
`top_b_mass`	invariant mass of the bottom quark in the top quark branch,
`top_w_mass`	invariant mass of the W boson in the top quark branch,
`top_mass`	invariant mass of top quark,
`higgs_q_1_mass`	invariant mass of a quark in the Higgs boson branch,
`higgs_q_2_mass`	invariant mass of a quark in the Higgs boson branch,
`higgs_q_comb_mass`	invariant mass of combined quarks in the Higgs boson branch,

`higgs_lep_mass`	invariant mass of lepton in the Higgs boson branch,
`higgs_mass_visible`	visible part of the invariant mass of the Higgs boson,
`tau_mass`	invariant mass of τ,
`antitau_mass`	invariant mass of anti-τ,
`tau_visible_mass`	visible part of the invariant mass of τ,
`antitau_visible_mass`	visible part of the invariant mass of anti-τ,
`antitop_lep_mass`	invariant mass of lepton in the antitop branch,
`antitop_b_mass`	invariant mass of b quark in the antitop branch,
`antitop_mass`	invariant mass of the antitop quark,
`antitop_mass_visible`	visible part of the invariant mass of the antitop quark,
`higgs_mass`	invariant mass of the Higgs boson – target variable.

However, all of these features will not be available when working with the detector level data. That is why some of these features have "visible" in their names. It means that the value was computed only from the particles that can be detected (excluding neutrinos). But the data are still useful, because they will serve for testing NN and tuning their hyper-parameters.

4.4 Detector level

In case of the detector level, creation of the dataset, which will be later used as input for the neural network, is more complicated. There are three main reason for this:

- The values obtained from the detector are inaccurate (measurement uncertainty).
- The dataset contains less information about the detected particles than in case of the truth values from simulator – the particle codes are unknown and only few groups of particles are distinguished
  - jets ("traces of quarks")
  - leptons
  - neutrinos (they pass the detector without interaction)
- The total detected energy of the system is lower than the real energy, because of the undetectable particles.

4.4.1 Algorithm

Since there are some conditions that the event must meet to be part of $2\ell SS + 1\tau_{had}$ channel, the algorithm (4) starts by checking these (2 b jets, 2 non-b jets, 2 same-charge leptons and 1 hadronic τ). After that, it can assign the non-b jets to the top quark branch, then checks both b jets to find the one which most probably (based on the the invariant mass of it and the other non-b jets) comes from the top quark.

The other b jet is then assigned to the antitop branch and both leptons are checked. Since there is missing energy caused by the neutrino, the algorithm does not select the lepton based on the invariant mass, but based on the distance between the lepton and the b jet (see equation 4.2). The other lepton is then assigned to the Higgs boson branch and all needed features can be computed and saved.

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} \tag{4.2}$$

Algorithm 4: Steps of the algorithm for extracting features on detector level dataset.

1. Divide the jets into two groups – b jets and non-b jets.
2. Check whether the needed conditions are met:
   (a) 2 b jets,
   (b) 2 non-b jets,
   (c) 2 same-charge leptons,
   (d) 1 hadronic τ.
3. Assign the non-b jets to the top quark branch.
4. Check the b jets (invariant mass after combining with non-b jets) and assign them to the top quark and the antitop quark branches.
5. Check the leptons (their distances from b jet) and assign them to the antitop quark and the Higgs boson branches.
6. Compute all needed features (see section 4.4.2).

4.4.2 Dataset features

The features names and description remains the same as in case of truth level. Most features remained, but those affected by the missing energy of undetectable neutrinos had to be omitted.

`met_met, met_phi`	missing energy characteristics,
`r_w_jets`	distance between quarks coming from the W boson,
`r_w_b_jet`	distance between the W boson and the bottom quark,
`top_q_1_mass`	invariant mass of a quark in the top quark branch,
`top_q_2_mass`	invariant mass of a quark in the top quark branch,
`top_b_mass`	invariant mass of the bottom quark in the top quark branch,
`top_w_mass`	invariant mass of the W boson in the top quark branch,
`top_mass`	invariant mass of the top quark,
`higgs_q_comb_mass`	invariant mass of combined quarks in the Higgs boson branch,
`higgs_lep_mass`	invariant mass of lepton in the Higgs boson branch,
`higgs_mass_visible`	visible part of the invariant mass of the Higgs boson,
`tau_visible_mass`	visible part of the invariant mass of τ,
`antitau_visible_mass`	visible part of the invariant mass of anti-τ,
`antitop_lep_mass`	invariant mass of lepton in the antitop branch,
`antitop_b_mass`	invariant mass of b quark in the antitop branch,
`antitop_mass_visible`	visible part of the invariant mass of the antitop quark,
`higgs_mass`	invariant mass of the Higgs boson – target variable.

4.5 Reconstruction of the invariant mass

In this section two approaches to the reconstruction of the invariant mass are described.

4.5.1 Exact formula

Using exact formula (see section 4.1.1.1) is how the label `higgs_mass` is obtained. This formula also serves for computing invariant masses of other particles and creating histograms of 13 673 samples that were selected from the original dataset.

4.5.1.1 Truth level

On truth level, the exact formula should be precise, because all the data (information about all particles) is available and there should be sharp peaks around the following values:

- 80.38 GeV/c^2 – W boson
- 174.98 GeV/c^2 – top quark
- 125.09 GeV/c^2 – Higgs boson

As it is clear from the histograms (see figures 4.2, 4.3, 4.4 and 4.5), the peaks around mentioned values are visible. This confirms successful reconstruction of the invariant mass of the particles.

In case of the Higgs boson (figure 4.5), the small left tail can be explained by various random photon emissions that may be causing the missing energy in the system. The missing right tail is also the reason why this plot does not have a Gaussian curve fit.

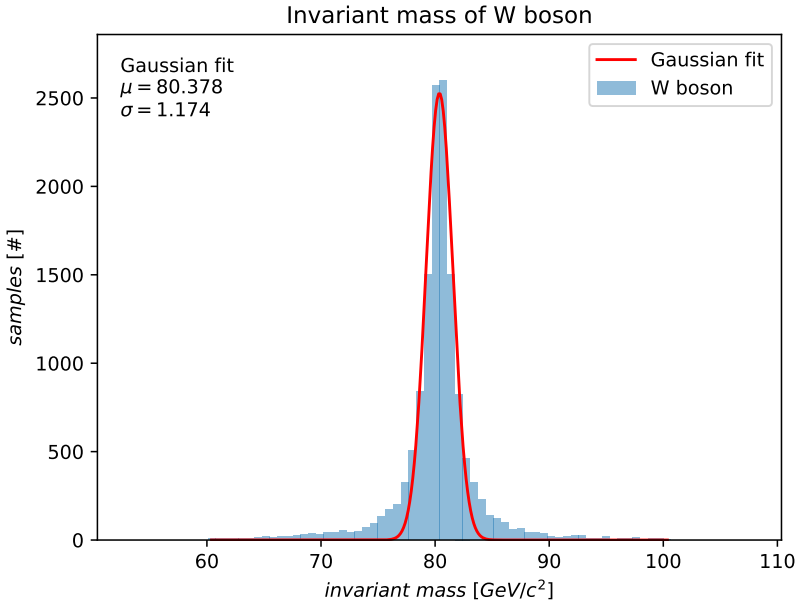


Figure 4.2: Reconstructed invariant mass of the W boson (from the top quark) – truth level.

4.5.1.2 Detector level

Unlike in the case of the truth level, the exact formula cannot give precise results (peaks around expected values), because of the missing energy. That

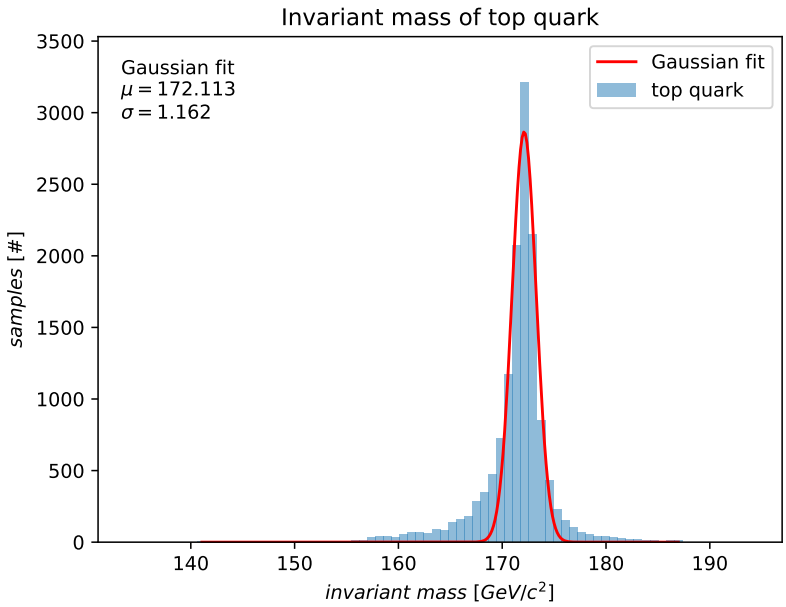


Figure 4.3: Reconstructed invariant mass of the top quark – truth level.

is why the neural networks (or other machine learning algorithms) need to be used.

In preparation for the detector level testing, two neural networks were created. The neural network created during the research on the truth level will be used with minor modifications of the code. The second prepared neural network is a classifier that will be trained to distinguish the signal events from background data. Skeleteons of both neural networks' codes can be found on the enclosed CD.

4.5.2 Neural networks

In this section, the process of training neural network and tuning its hyper-parameters is described. In first phase, the neural networks work only with truth level data to estimate the invariant mass of the Higgs boson and reproduce its histogram presented in section 4.5.1.1.

4.5.2.1 Truth level – results

First, different neural networks topologies (varying in number of neurons in each layer, number of layers and activation and loss functions) were tested on full truth dataset (working with all features listed in section 4.3.3). Also different hyper-parameters (loss function, learning rate, batch size, number of epochs) were tuned using cross validation.

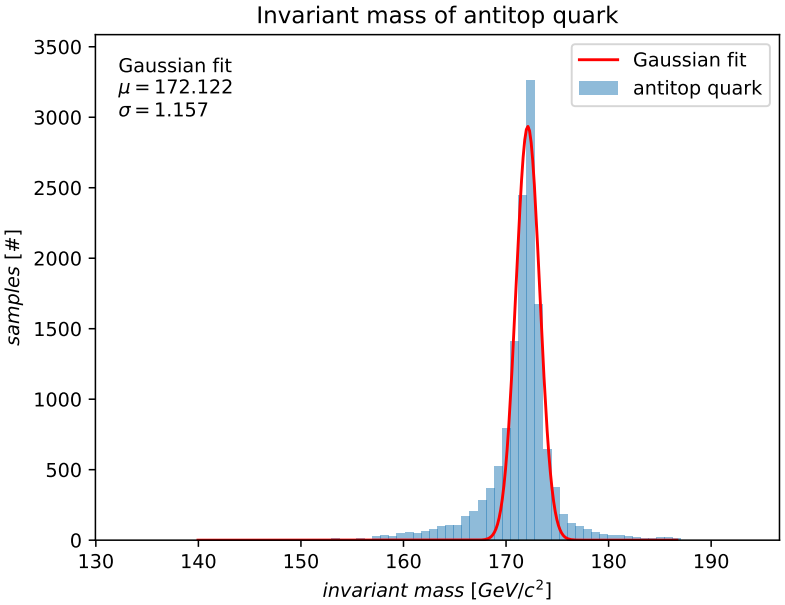


Figure 4.4: Reconstructed invariant mass of the antitop quark – truth level.

Detailed results can be find on enclosed CD, only few interesting results will be shown here:

As can be seen in figure 4.6, the convergence is relatively fast (5-10 epochs), but the result (percentage error) depends on the splitting the dataset to train and validation parts.

Next figure 4.7 shows how bigger batch size influenced the convergence. After changing the batch size from 2000 to 5000 (other parameters were left unchanged), it took almost two to three times more epochs to converge (compare with figure 4.6).

As a result, the best topology and parameters (in terms of percentage error on the reconstructed invariant mass of the Higgs boson and training time) were chosen and were subject to further analysis with usage of different datasets[10] (see table 4.1).

In figure (4.8), the cross validation of the best network is shown. The mean loss on training sample was around 1.5 %, in case of validation sample, the mean loss was slightly higher (with mean around 1.75 %). No overfitting was recorded.

[10]While working with the different datasets, also some other topologies and hyper-parameters were tested. Although they occasionally provided better results, they were not significantly better than the chosen network, so they were not tested thoroughly.

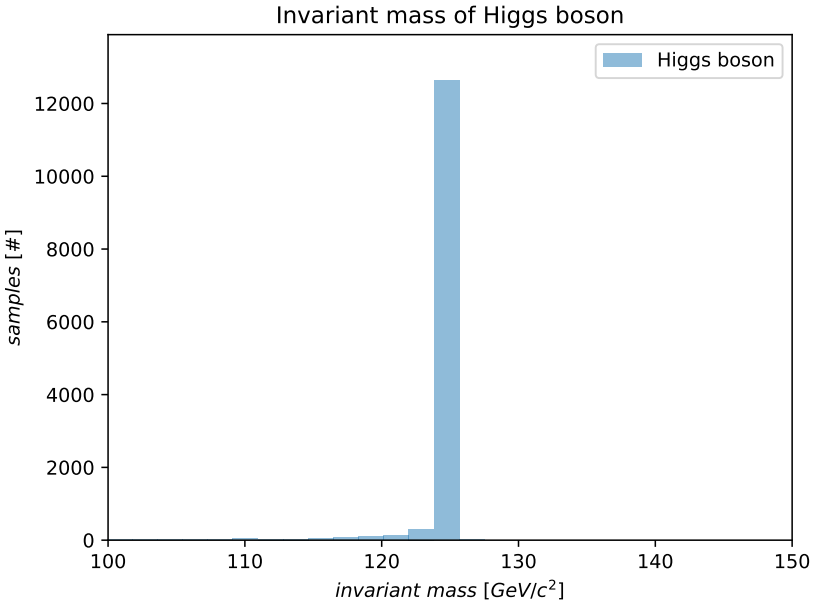
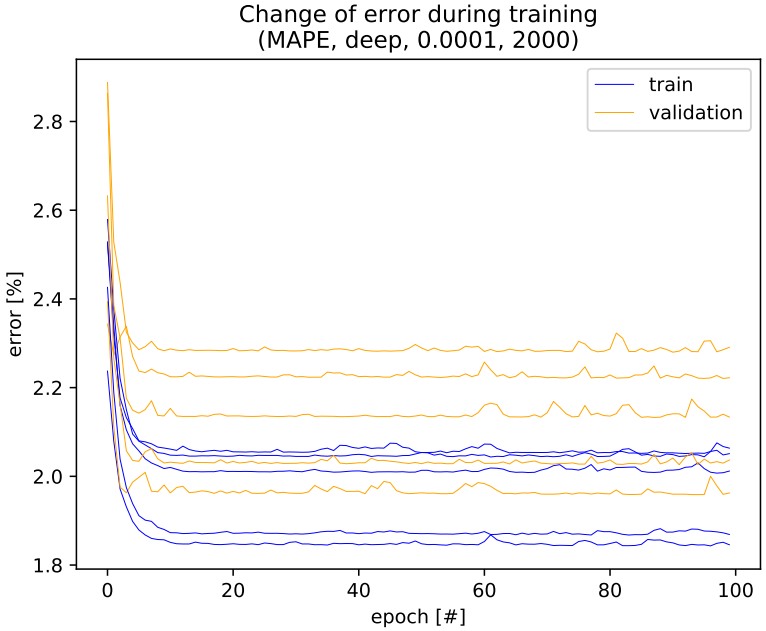


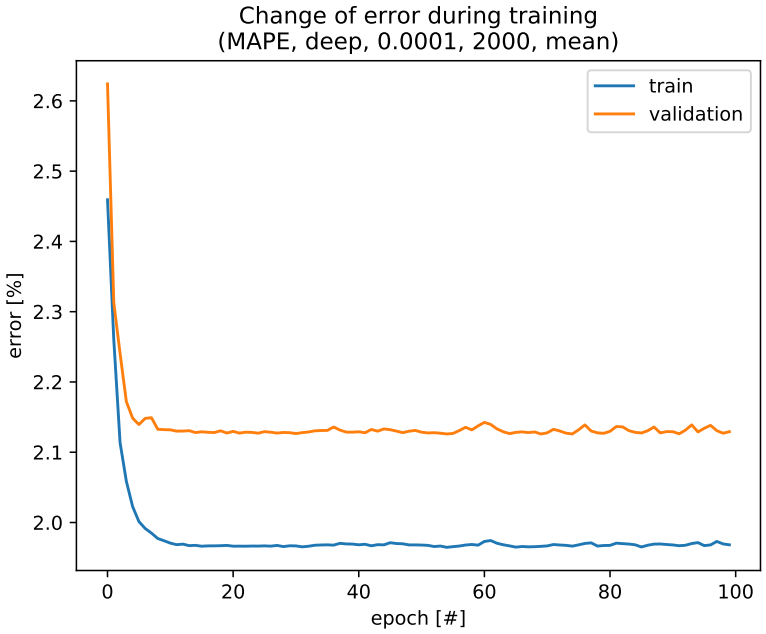
Figure 4.5: Reconstructed invariant mass of the Higgs boson – truth level.

Table 4.1: Selected hyper-parameters.

Parameter	Value
learning rate	0.01
batch size	2000
number of epochs	100
activation function	ReLU
loss function	MAPE
network topology	two hidden layers with 2*`dimension_of_data` and `dimension_of_data` neurons
train/test split	70:30

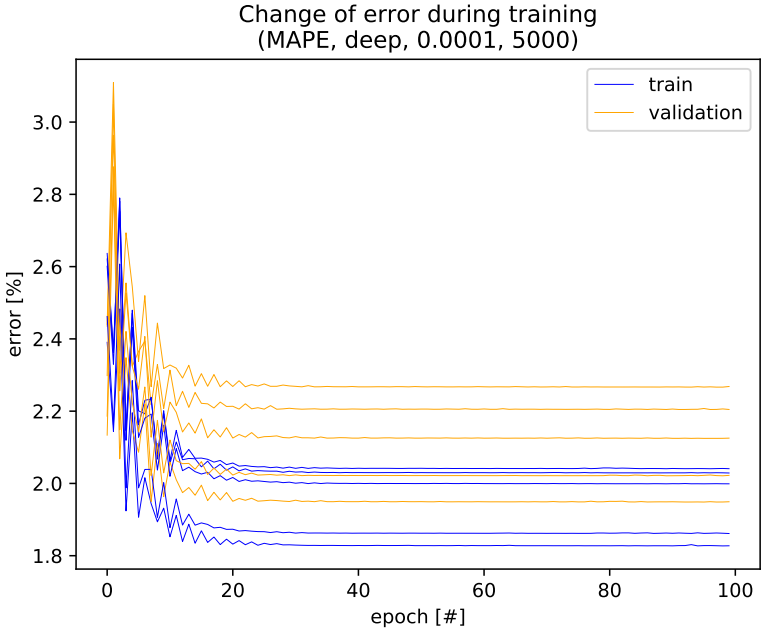


(a) Figure shows the changes of mean percentage error during the training phase of the neural network on 5 different validation splits.

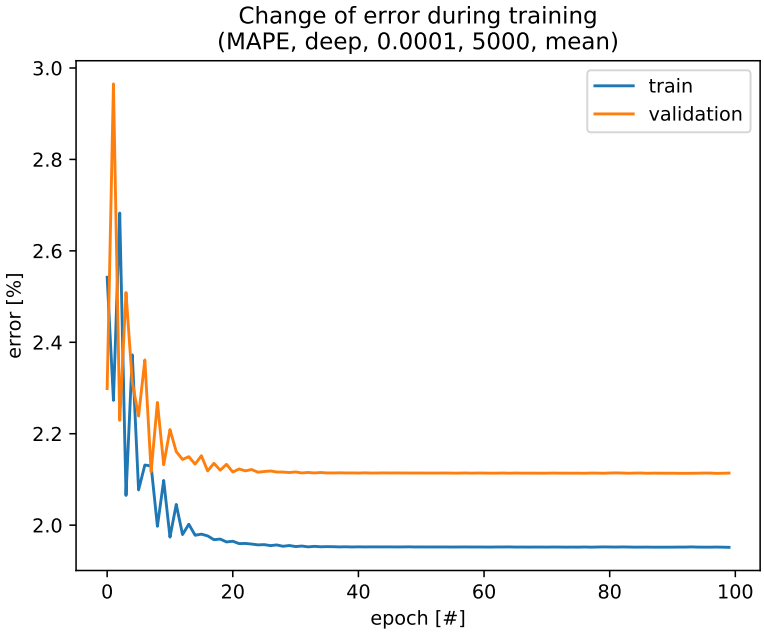


(b) Average over the results of the 5 splits from figure 4.6a.

Figure 4.6: Dependence of mean percentage error on different splits of the training dataset.

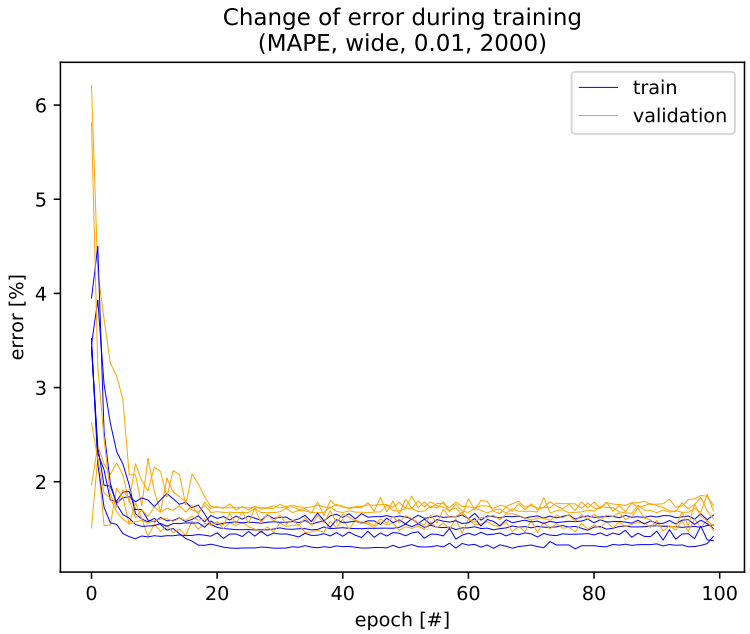


(a) Figure shows the changes of mean percentage error during the training phase of the neural network on 5 different validation splits.

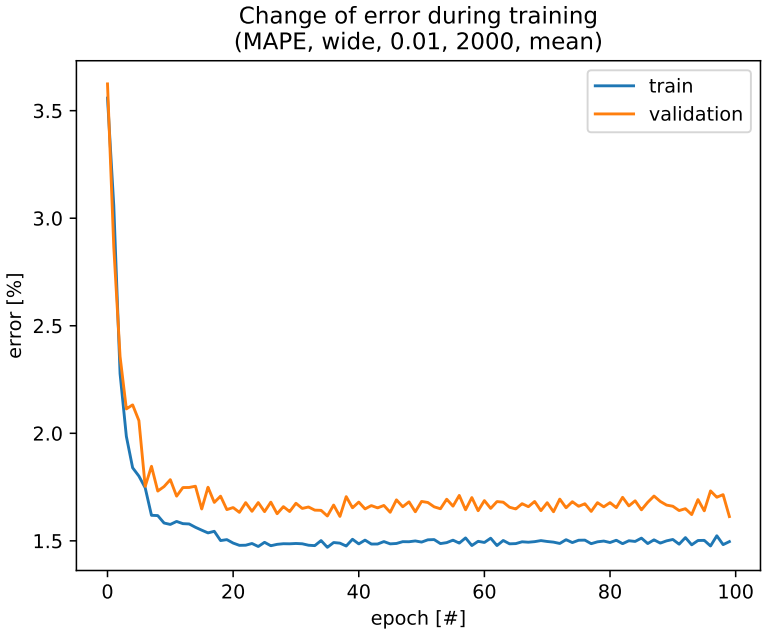


(b) Average over the results of the 5 splits from figure 4.7a.

Figure 4.7: Slower convergence with larger batchsize.



(a) Figure shows the changes of mean percentage error during the training phase of the neural network on 5 different validation splits.



(b) Average over the results of the 5 splits from figure 4.8a.

Figure 4.8: The best configuration from different networks and hyper-parameters settings.

After having chosen the network topology and hyper-parameters, a series of tests on six neural networks was performed. These test are described here and summarized in table 4.2 in the end of this section.

The first network was trained on the whole dataset containing full information about all three branches of the Feynman diagram of the Higgs boson decay (23 features).

Since it had the most (all) information about the event, it predicted the closest estimations with mean absolute percentage error around 2.4 %.

In the figure 4.10a, there is a visible peak around value 124 GeV/c^2 which is close to the expected mass of 125.09 GeV/c^2. It does not copy the distribution of the values computed by the exact method – it has a tail on the right side (unlike the histogram of exact method). This confirms that the network is not overfitted and also has not learned to predict one value.

Figure 4.10b, shows the distribution of errors. It is, as expected, centered around 0, with high peak.

Figure 4.9 visualizes the importance of individual features. The model is trained, tested once (to get benchmark results), but after that, one by one, each feature column is noised (multiplied by a random number from normal distribution ($\mu = 1, \sigma = 0.1$)) and the change of the target (predicted) value is measured. Edges of the rectangles represent first and third quartile, whiskers represent minimum and maximum of all of the data. Results are sorted from the most important feature (bottom) to the least important one (top).

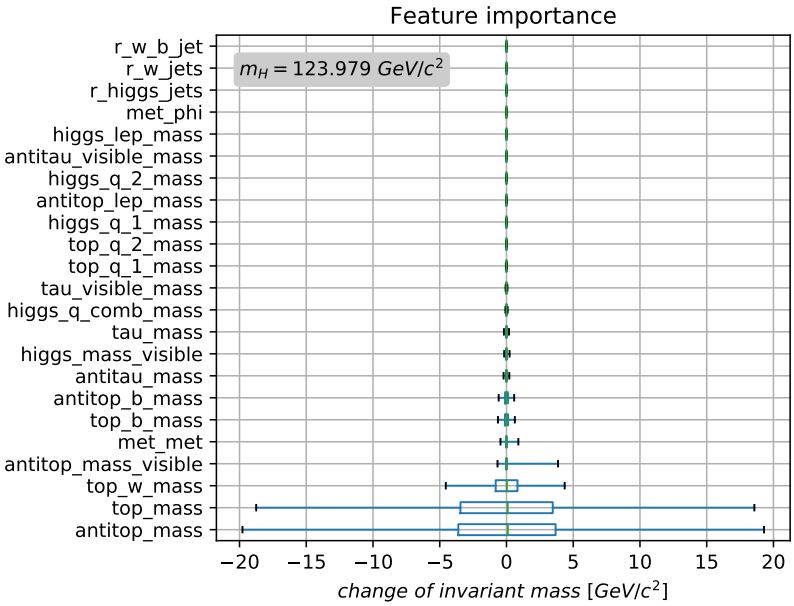
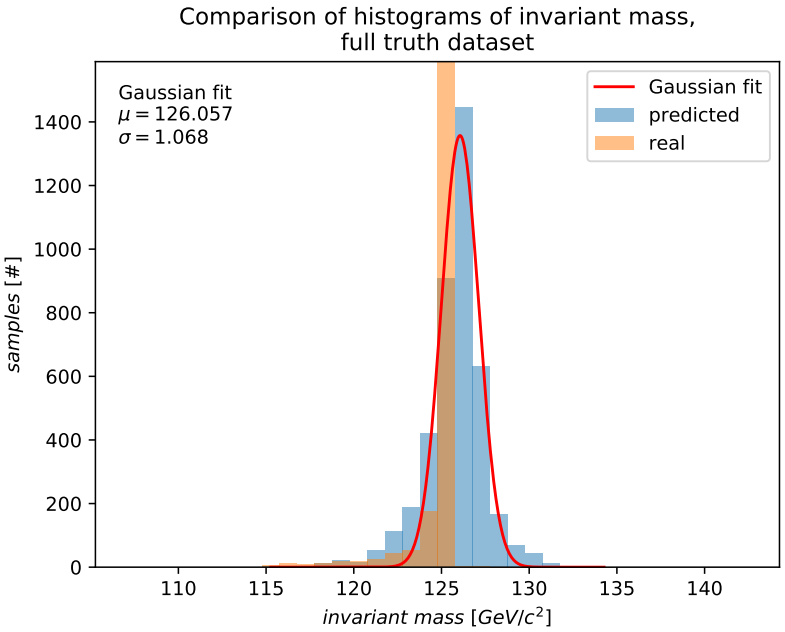
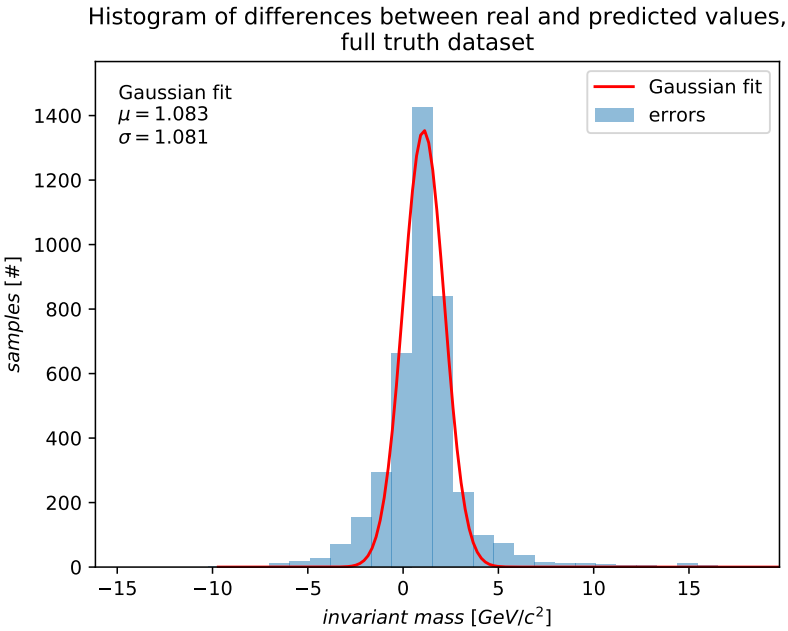


Figure 4.9: Feature importance – full truth dataset



(a) Comparison of the values predicted by the neural network (blue) and the values computed by exact method (orange).



(b) Errors – differences between values computed by exact method and values predicted by the neural network.

Figure 4.10: Invariant mass of the Higgs boson – full truth dataset.

The second network was trained on the visible part of the truth dataset containing information about all three branches of the Feynman diagram of the Higgs boson decay. Unlike the first network, this one does not have access to information about neutrinos (which cannot be detected), so it cannot compute the masses of some particles and it has to operate only with the "visible" part of the mass (20 features).

This model still had enough information (though it lost the most important feature "antitop_mass" and had to work with its visible part) to provide good estimations with mean absolute percentage error around 2.5 %.

In the figure 4.12a, there is still a visible peak around value 124 GeV/c^2, but the deviation is higher. This matches the expectations – the network lost the most important feature and had less information to work with.

Figure 4.12b shows the distribution of errors and figure 4.11 shows features sorted by their importance.

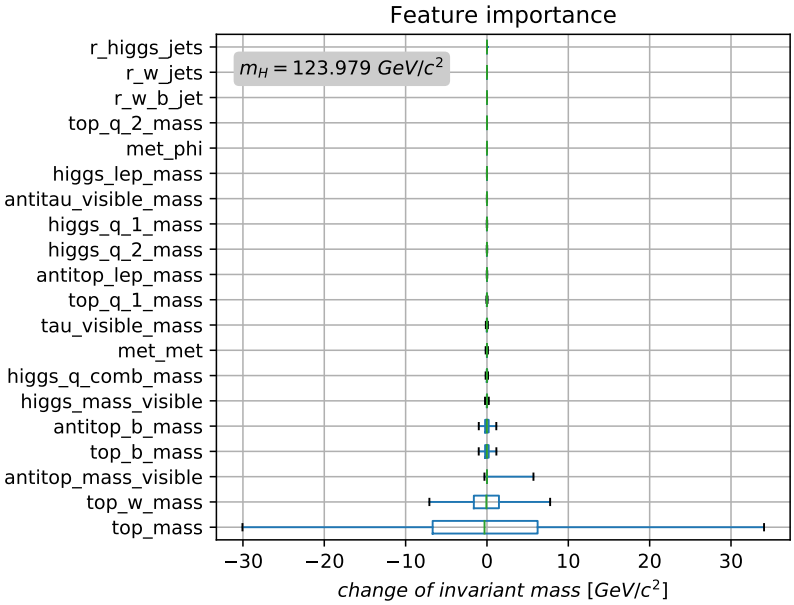
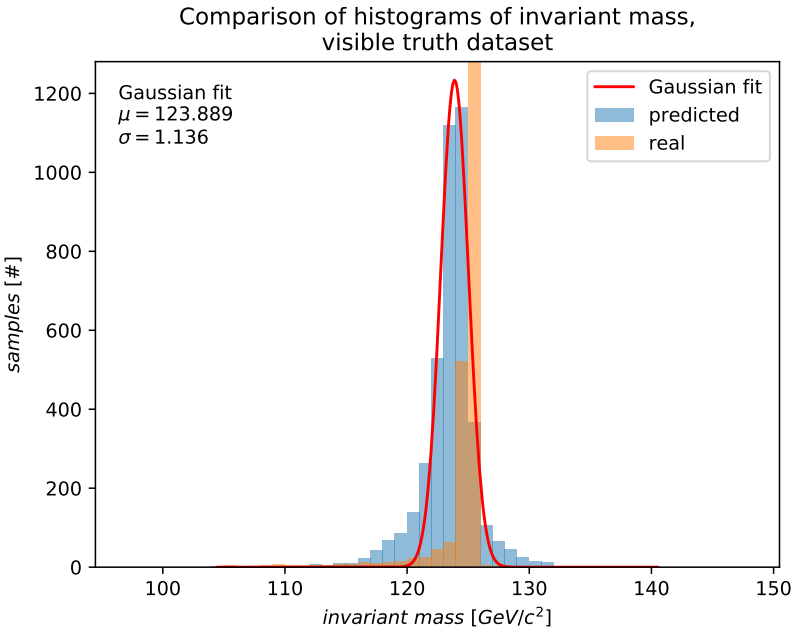
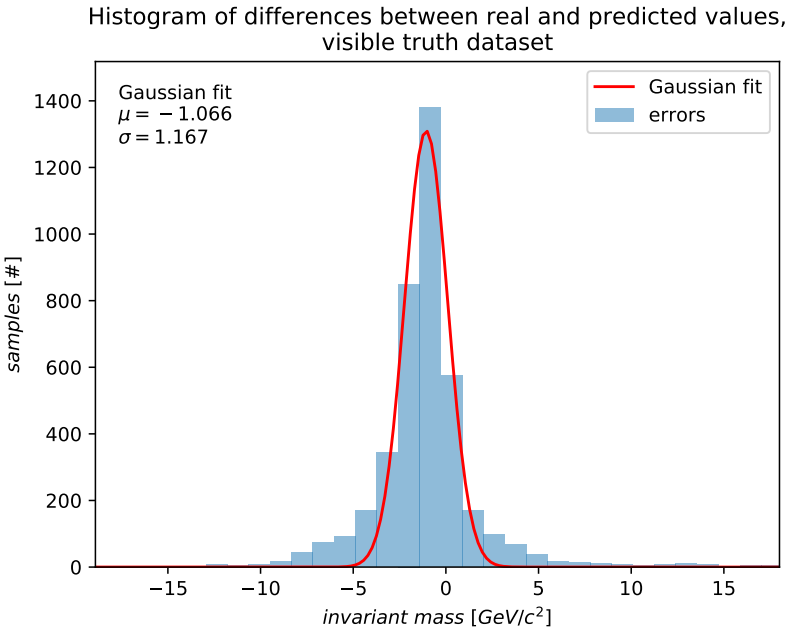


Figure 4.11: Feature importance – visible part of the truth dataset.



(a) Comparison of the values predicted by the neural network (blue) and the values computed by exact method (orange).



(b) Errors – differences between values computed by exact method and values predicted by the neural network.

Figure 4.12: Invariant mass of the Higgs boson – visible part of the truth dataset.

The third network was trained on the detector available part of the truth dataset containing information about all three branches of the Feynman diagram of the Higgs boson decay. Unlike the second network, this one does not have access to information about jets coming from the Higgs boson, because they are already included in τ information in the output of the detector (although technically, they can be detected), so again, the network has less information than in the previous case (17 features).

This model lost only three less important features, so the estimations did not get much worse – mean absolute percentage error was around 2.6 %.

In the figure 4.14a, there is still a visible peak around value 124 GeV/c^2, but the deviation is again slightly higher than in the case of the full dataset but lower than in the case of the previous dataset with more information. This may mean that one of the features that were in the previous dataset does not contribute to better results but makes them worse.

Figure 4.14b shows the distribution of errors and figure 4.13 shows features sorted by their importance.

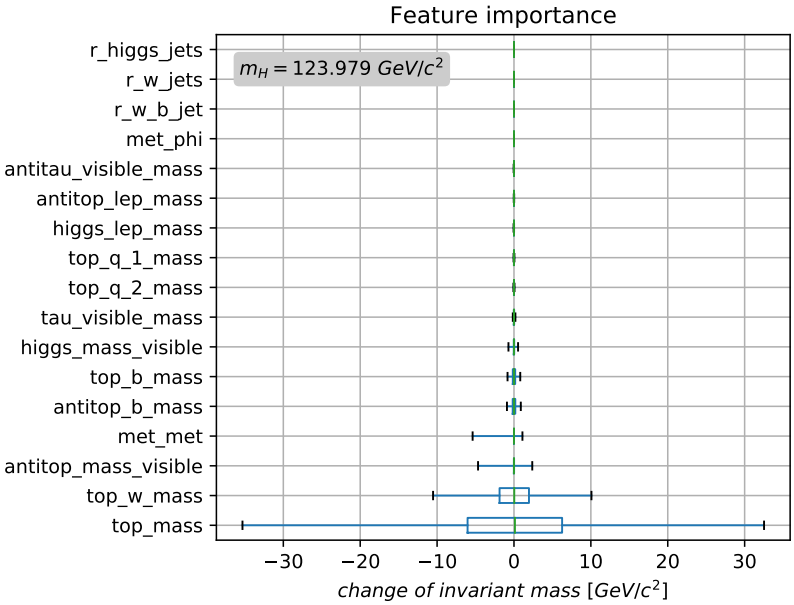
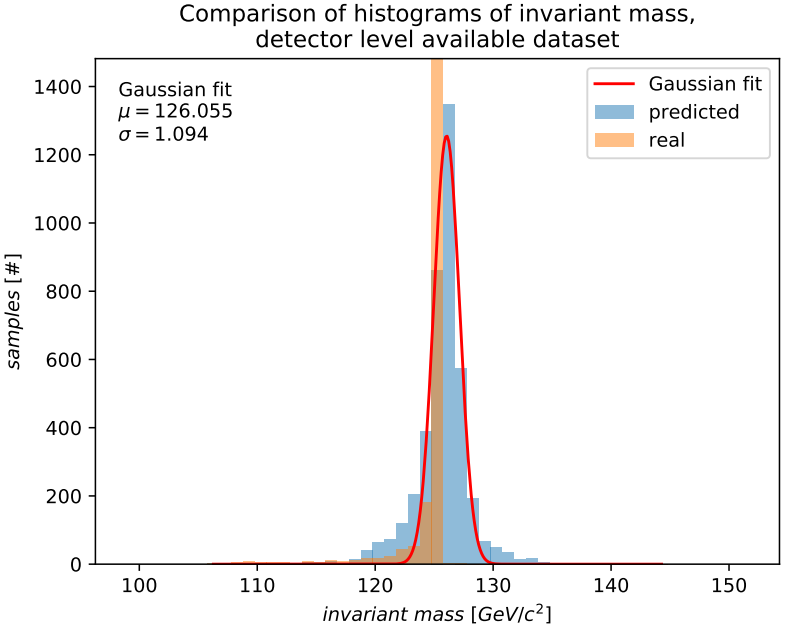
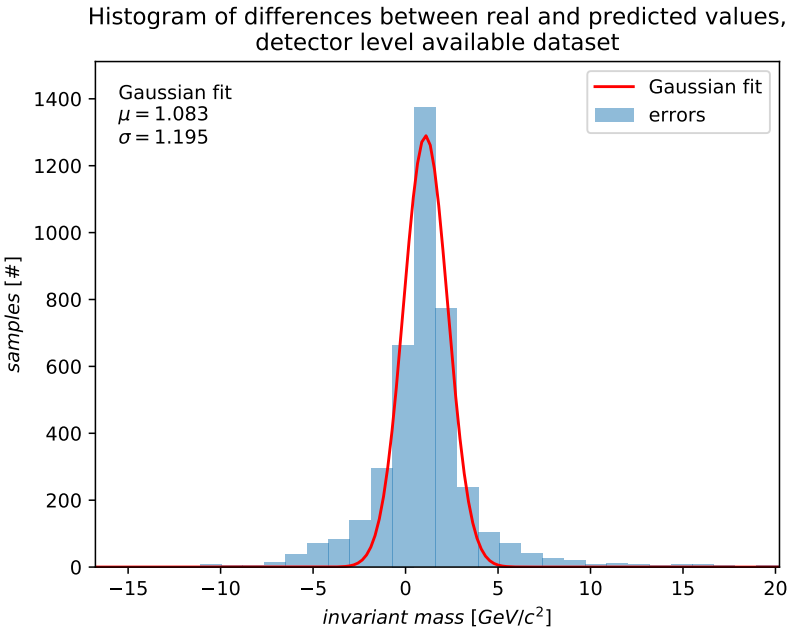


Figure 4.13: Feature importance – detector available part of the truth dataset.



(a) Comparison of the values predicted by the neural network (blue) and the values computed by exact method (orange).



(b) Errors – differences between values computed by exact method and values predicted by the neural network.

Figure 4.14: Invariant mass of the Higgs boson – detector available part of the truth dataset.

The fourth network was trained on the dataset containing full information about only the Higgs boson branch of the Feynman diagram of the Higgs boson decay (12 features).

Although this model lacks almost half of the features, the estimations of the invariant mass of the Higgs boson remained relatively precise – mean absolute percentage error was around 3.3 % (1 % worse than the first network working with the full dataset).

In the figure 4.16a, the peak is still centered around the expected value, however, the deviation has doubled.

Figure 4.16b shows the distribution of errors and figure 4.15 shows features sorted by their importance.

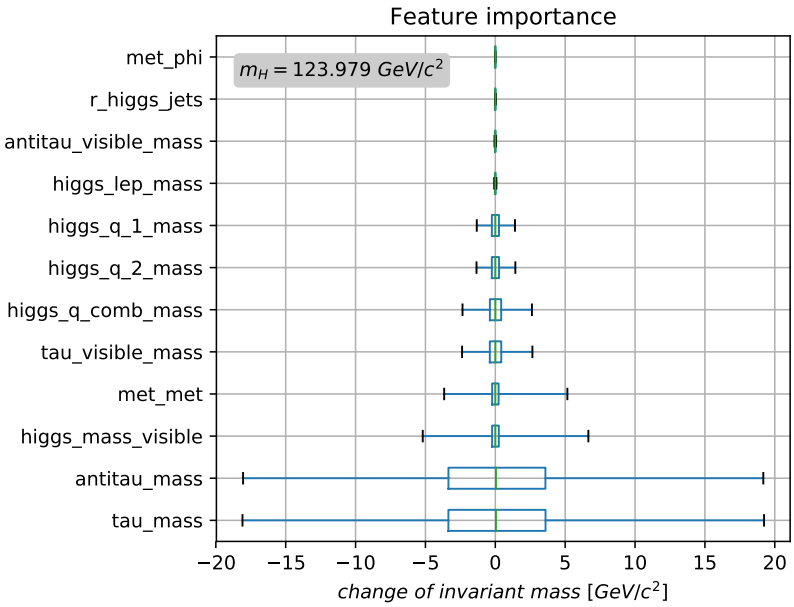
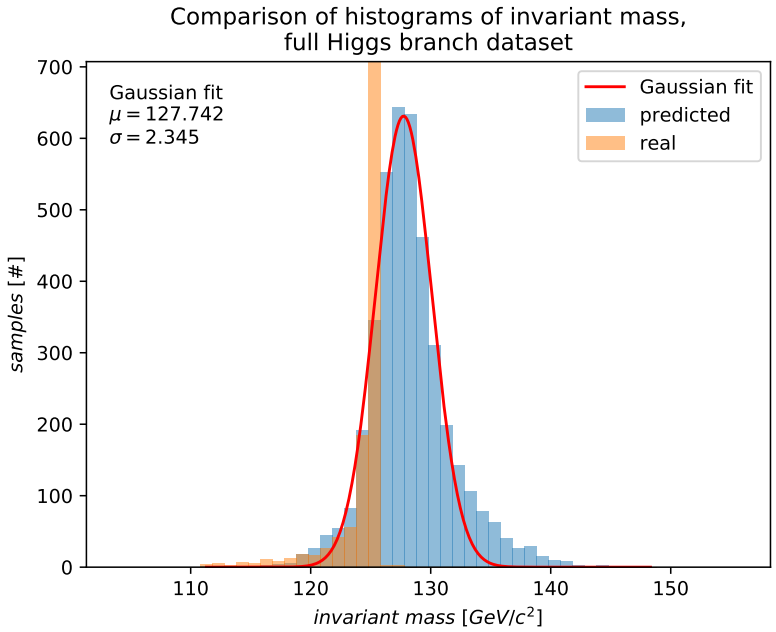
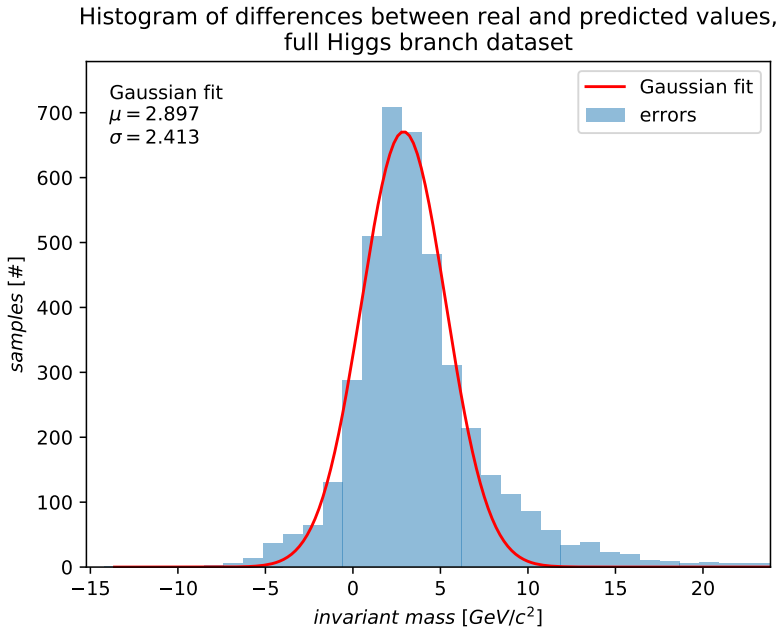


Figure 4.15: Feature importance – full truth dataset (Higgs branch).



(a) Comparison of the values predicted by the neural network (blue) and the values computed by exact method (orange).



(b) Errors – differences between values computed by exact method and values predicted by the neural network.

Figure 4.16: Invariant mass of the Higgs boson – full truth dataset (Higgs branch).

The fifth network was trained on the visible part of the truth dataset containing information about only the Higgs boson branch of the Feynman diagram of the Higgs boson decay (10 features).

In case of this model, we can observe the first significant drop in the accuracy of estimations – mean absolute percentage error was around 10.2 %.

In the figure 4.18a, the peak is significantly lower and shifted to lower values (120 GeV/c^2) and the deviation is approximately five times higher than in the previous case.

Also the distribution of errors in the figure 4.18b is much wider. Figure 4.17 shows features sorted by their importance.

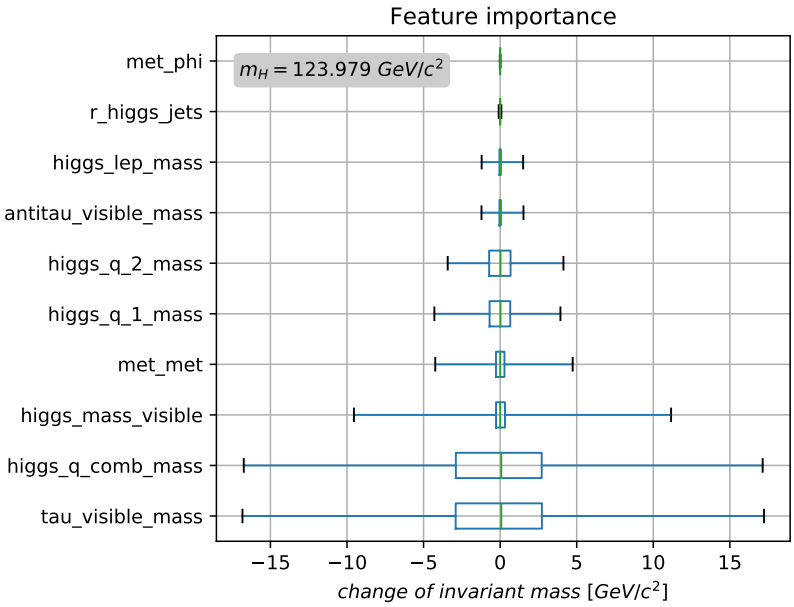
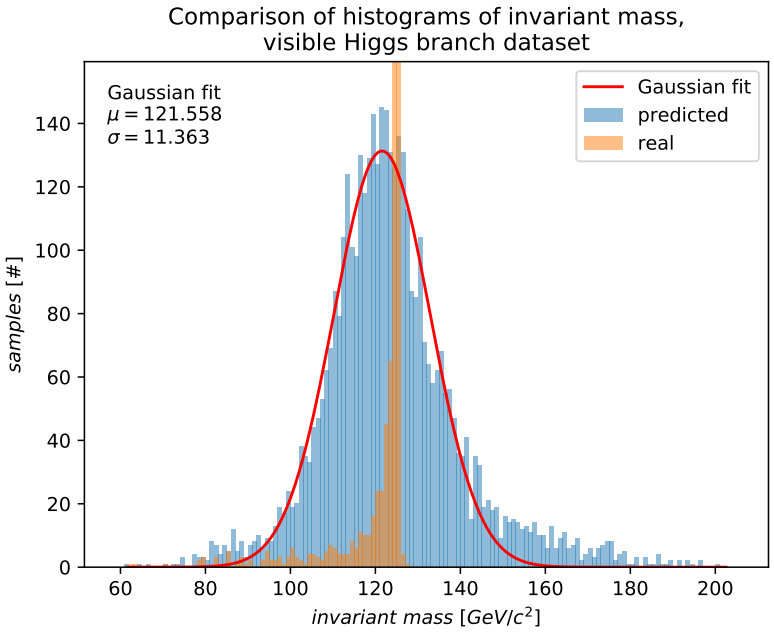
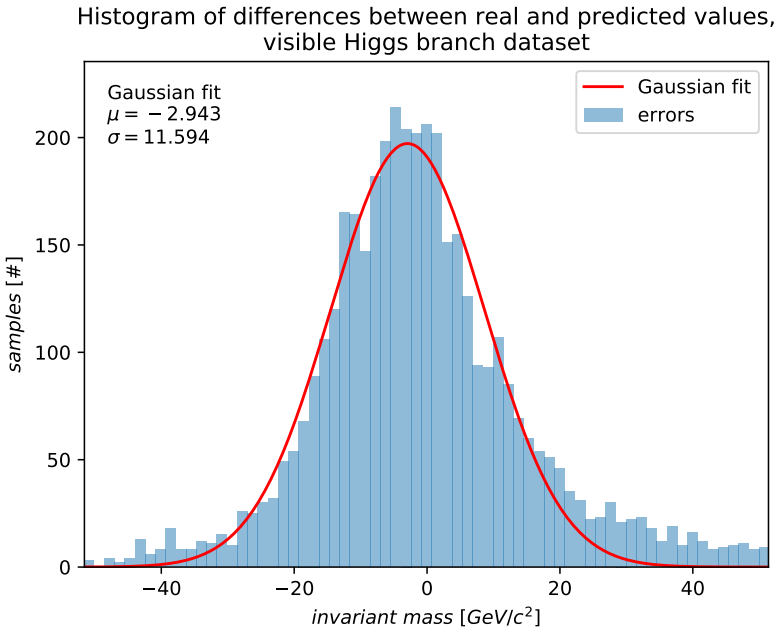


Figure 4.17: Feature importance – visible part of the truth dataset (Higgs branch).



(a) Comparison of the values predicted by the neural network (blue) and the values computed by exact method (orange).



(b) Errors – differences between values computed by exact method and values predicted by the neural network.

Figure 4.18: Invariant mass of the Higgs boson – visible part of the truth dataset (Higgs branch).

The sixth network was trained on the detector available part of the truth dataset containing information about only the Higgs boson branch of the Feynman diagram of the Higgs boson decay (7 features).

As expected this model with the least features available provided the worst results – mean absolute percentage error was around 21.0 %.

In the figure 4.20a, the peak is 10 GeV/c^2 (around $110 - 115$ GeV/c^2) off and the deviation is approximately five times higher than in the previous case.

Also the distribution of errors in the figure 4.20b is even wider that in the previous case (with some cases reaching error of over 100 GeV/c^2). Figure 4.19 shows features sorted by their importance.

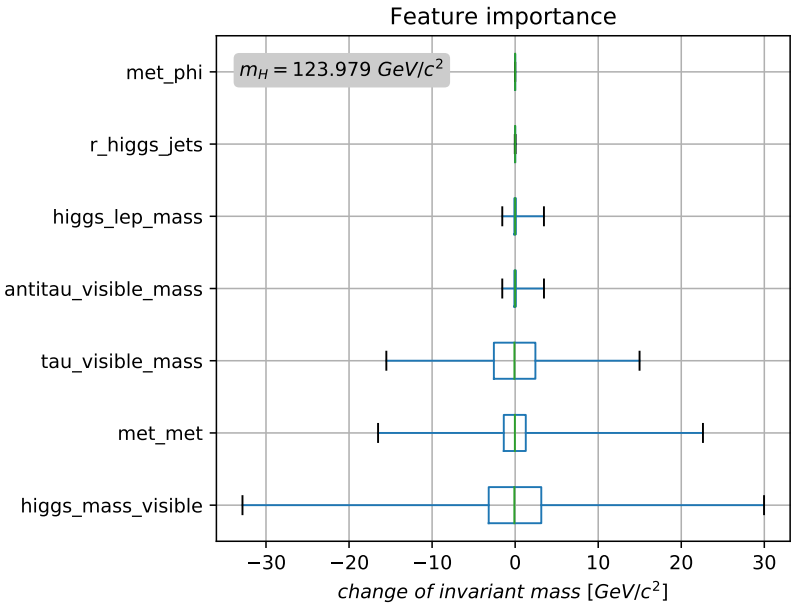
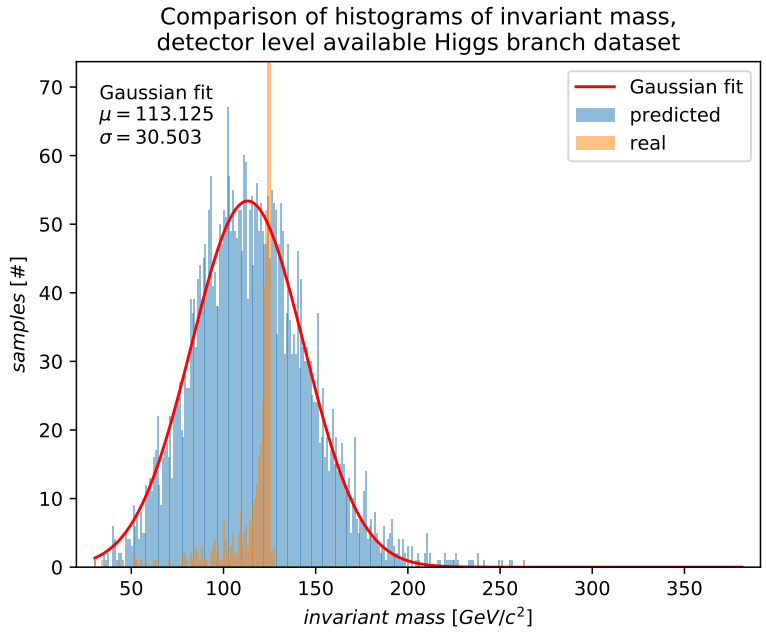
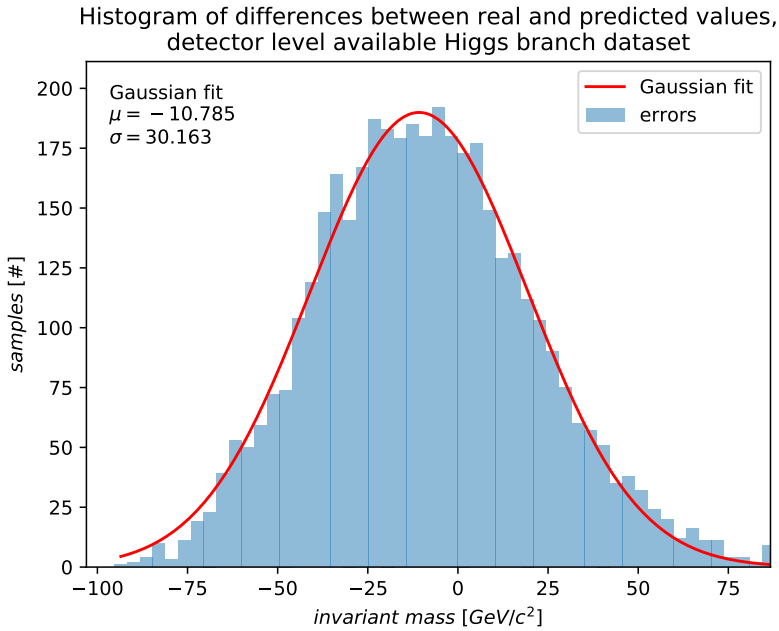


Figure 4.19: Feature importance – detector available part of the truth dataset (Higgs branch).



(a) Comparison of the values predicted by the neural network (blue) and the values computed by exact method (orange).



(b) Errors – differences between values computed by exact method and values predicted by the neural network.

Figure 4.20: Invariant mass of the Higgs boson – detector available part of the truth dataset (Higgs branch).

Table 4.2 summarizes the results on the truth level. As expected, the error is increasing with decreasing number of available features. For the purpose of further analysis, the third row is the most important, because "detector available part" is the same dataset (in terms of number and type of features) that will be available on the detector level.

Table 4.2: Summary of the truth level results.

Dataset	**Features**	**Error**	μ $[GeV/c^2]$	σ $[GeV/c^2]$
full	23	2.37 %	126.06	1.06
visible part	20	2.53 %	123.89	1.14
detector available part	17	2.63 %	126.06	1.09
full (Higgs branch)	12	3.26 %	127.74	2.34
visible part (Higgs branch)	10	10.21 %	121.59	11.36
detector available part (Higgs branch)	7	21.00 %	113.13	30.50

4.6 Further research

Next part of the research will be focused on estimating the invariant mass of the Higgs boson on the detector level. For this, the neural networks trained on the truth level dataset will be used once the events from the detector level dataset will be prepared for studying.

After that, another neural network – classifier will be trained to differentiate signal events from background events. The skeleton of the network is already prepared and can be used once the datasets are prepared.

Another idea for future research, that might bring better results, is studying the photon emissions that influence the total energy of the decay system.

Conclusion

There are several outputs of this thesis. First, the algorithm for selecting events belonging to the $2\ell SS + 1\tau_{had}$ channel and extracting selected features.

Another output is the group of three neural networks. Out of them, one has been fully tested – the regression model estimating the invariant mass of the Higgs boson on the truth level. Six modifications of this neural network have been trained and tested on datasets with varying number of features.

One of these models is suitable for estimating the invariant mass of the Higgs boson on the detector level (it is trained on the part of the truth level dataset that is also available on the detector level and its estimations are accurate enough).

This network will be tested on the detector level. In the future, the invariant mass reconstruction on the detector level could be used in a neural network classifier and contribute to increasing the signal/background ratio.

Bibliography

[1] Aaboud, M., et al. Evidence for the associated production of the higgs boson and a top quark pair with the atlas detector. *Phys. Rev. D 97* (Apr 2018), 072003. Available at https://link.aps.org/doi/10.1103/PhysRevD.97.072003.

[2] Aaboud, M., et al. Observation of Higgs boson production in association with a top quark pair at the LHC with the ATLAS detector. *Physics Letters B 784* (2018), 173 – 191. Available at http://www.sciencedirect.com/science/article/pii/S0370269318305732.

[3] Alitti, J., et al. A Measurement of two jet decays of the W and Z bosons at the CERN $\bar{p}p$ collider. *Z. Phys. C49* (1991), 17–28. Available at https://lib-extopc.kek.jp/preprints/PDF/1990/9008/9008480.pdf.

[4] Bärtschi, P., Galloni, C., Lange, C., and Kilminster, B. Reconstruction of τ lepton pair invariant mass using an artificial neural network. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 929* (2019), 29 – 33. Available at http://www.sciencedirect.com/science/article/pii/S0168900219303377.

[5] Brownlee, J. How to choose loss functions when training deep learning neural networks, Jan 2019. Available at https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[6] CERN. The higgs boson. Available at https://home.cern/science/physics/higgs-boson.

[7] CERN. Histograms. Available at https://root.cern.ch/root/htmldoc/guides/users-guide/Histograms.html.

[8] CERN. Integration with other languages. Available at https://root.cern.ch/integration-other-languages.

[9] CERN. PyROOT. Available at https://root.cern.ch/pyroot.

[10] CERN. ROOT a Data analysis Framework. Available at https://root.cern.ch/.

[11] CERN. TLorentzVector Class Reference. Available at https://root.cern.ch/doc/master/classTLorentzVector.html.

[12] CERN. TTree Class Reference. Available at https://root.cern.ch/doc/v614/classTTree.html.

[13] CERN. Calorimeter, Nov 2017. Available at https://atlas.cern/discover/detector/calorimeter.

[14] CERN. The inner detector, Jul 2017. Available at https://atlas.cern/discover/detector/inner-detector.

[15] CERN. Magnet system, Aug 2017. Available at https://atlas.cern/discover/detector/magnet-system.

[16] CERN. Muon spectrometer, Dec 2017. Available at https://atlas.cern/discover/detector/muon-spectrometer.

[17] CERN. Mass / invariant mass, Jan 2018. Available at https://atlas.cern/glossary/mass-invariant-mass.

[18] Fortuner, B. Activation functions. Available at https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html.

[19] Glaysher, P. BDTs in the Search for $t\bar{t}H$ Production with Higgs Decays to $b\bar{b}$ at ATLAS. Available at https://cds.cern.ch/record/2275063.

[20] Hao, Z. Loss functions in neural networks, Jun 2017. Available at https://isaacchanghau.github.io/post/loss_functions/.

[21] Higgs, P. W. Broken symmetries and the masses of gauge bosons. *Phys. Rev. Lett. 13* (Oct 1964), 508–509. Available at https://link.aps.org/doi/10.1103/PhysRevLett.13.508.

[22] Karpathy, A. Convolutional neural networks. Available at http://cs231n.github.io/convolutional-networks/.

[23] Keras developers. Keras: The python deep learning library. Available at https://keras.io/.

[24] Lerner, L. *Physics for Scientists and Engineers*. No. sv. 1 in Physics for Scientists and Engineers. Jones and Bartlett, 1996. Available at https://books.google.cz/books?id=pEduPwAACAAJ.

[25] Lönnblad, L., Peterson, C., and Rögnvaldsson, T. Mass reconstruction with a neural network. *Physics Letters B 278*, 1 (1992), 181 – 186. Available at http://www.sciencedirect.com/science/article/pii/037026939290731I.

[26] Mehta, A. A complete guide to types of neural networks, Jan 2019. Available at https://www.digitalvidya.com/blog/types-of-neural-networks/.

[27] Mrenna, S. Particle codes, Oct 2007. Available at http://home.fnal.gov/~mrenna/lutp0613man2/node44.html.

[28] Nobel Media. The Nobel Prize in Physics 2013. Available at https://www.nobelprize.org/prizes/physics/2013/press-release/.

[29] NumPy developers. NumPy. Available at https://www.numpy.org/.

[30] Pandas developers. Python Data Analysis Library. Available at https://pandas.pydata.org/.

[31] PDG physicists. Review of particle physics. *Phys. Rev. D 98* (Aug 2018), 030001. Available at https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[32] Sabinasz, D. Coding games and programming challenges to code better. Available at https://www.codingame.com/playgrounds/9487/deep-learning-from-scratch---theory-and-implementation/gradient-descent-and-backpropagation.

[33] Sharma, H. Activation Functions : Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep..., Jan 2019. Available at https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e.

[34] Sjöstrand, T. Introduction to PYTHIA. Available at http://home.thep.lu.se/~torbjorn/Pythia.html.

[35] Stapf, B., Wermes, N., and Dingfelder, J. Studies of $t\bar{t}H(\tau\tau)$ event topologies and kinematics with the ATLAS experiment, 2015. Available at "http://cds.cern.ch/record/2317012".

[36] Stutz, D. Backpropagation algorithm. Available at https://github.com/davidstutz/latex-resources/blob/master/algo-backprop/backprop.tex.

[37] Tch, A. The mostly complete chart of neural networks, explained, Aug 2017. Available at https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464.

[38] TensorFlow developers. TensorFlow. Available at https://www.tensorflow.org/.

[39] The ATLAS Collaboration. The ATLAS experiment at the CERN large hadron collider. *Journal of Instrumentation 3*, 08 (aug 2008), S08003–S08003. Available at https://doi.org/10.1088%2F1748-0221%2F3%2F08%2Fs08003.

[40] Venkatachalam, M. Recurrent neural networks, Mar 2019. Available at https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce.

[41] Wong, C. *Introduction to High-energy Heavy-ion Collisions*. Introduction to High-energy Heavy-ion Collisions. World Scientific, 1994. Available at https://books.google.cz/books?id=Fnxvrdj2NOQC.

Appendix A

Acronyms

ANN	Artificial neural network
ATLAS	A Toroidal LHC ApparatuS
BDT	Boosted Decision Trees
BP	BackPropagation
CERN	European Organization for Nuclear Research (French: Conseil Européen pour la Recherche Nucléaire)
CMS	Compact Muon Solenoid
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
FNN	Feed-forward Neural Network
GeV	GigaelektronVolt
LHC	Large Hadron Collider
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
MSLE	Mean Squared Logarithmic Error
NN	Neural Network
PDG	Particle Data Group
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
XOR	eXclusive OR

Appendix B

Contents of enclosed CD and GIT repository

Following directory tree describes the contents of enclosed CD and GIT repository at `https://gitlab.fit.cvut.cz/urbanp13/masters-thesis---petr-urban.git`.

- `readme.txt` ..... the file with CD/GIT contents description
- `data` ..... the directory of datasets
- `src` ..... the directory of source codes
  - `preprocessing` .... the directory of source codes for feature extraction
  - `nn` ..... the directory of source codes for neural networks
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format