Particle Track reconstruction using a recurrent neural network at the $\mu - 3e$ experiment

Bachelor thesis of
Sascha Liechti

06.04.2018

Supervised by
Prof. Nicola Serra
Dr. Patrick Owen

Abstract During the $\mu - 3e$ experiment we faced the challenge of reconstructing the paths of certain low momentum particles that curled back into the detector and cause additional hits. To face this, a recurrent neural network was used which found the right track for 87% of these particles.

Contents

1 Standard Model

1.1 Elementary particles and forces

The Standard Model(SM) describes all known elementary particles as well as three of the four known forces[1].
The elementary particles that make up matter can be split into two categories, namely quarks and leptons. There are 6 types of quarks and six types of leptons. The type of a particle is conventionally called flavour. The six quark flavours and the three lepton flavours are separated over 3 generations (each which two quarks and two leptons in it). Experimental evidence suggests that there exist exactly three generations of particles [1]. Each particle of the first generation has higher energy versions of itself with the similar properties, besides their mass, (e.g. $e^- \rightarrow \mu^- \rightarrow \tau^-$) as in other generations. For each following generation, the particles have a higher mass than the generation before.

Table 1: Quarks in the Standard Model

Quarks				
	Particle		Q[e]	$\frac{mass}{GeV}$
1. Gen.	up	u	$-\frac{1}{3}$	0.003
	down	d	$\frac{2}{3}$	0.005
2. Gen.	strange	s	$-\frac{1}{3}$	0.1
	charm	c	$\frac{2}{3}$	1.3
3. Gen.	bottom	b	$-\frac{1}{3}$	4.5
	top	t	$\frac{2}{3}$	174

One category consists of quarks(q)(see Table 1). In this, we differentiate between up-type quarks, with charge $-\frac{1}{3}e$, and down-type, quarks with charge $\frac{2}{3}e$. Quarks interact with all fundamental forces.
Each quark carries a property called colour-charge. The possible colour charges are red(r), green(gr), blue(bl) in which anti-quarks carry anti-colour. Quarks can only carry one colour, whilst every free particle has to be colourless[2]. In conclusion we cannot observe a single quark.
Free particles can achieve being colourless in two ways. Either by having all three colours present in the same amount (one quark of each colour), which creates the characteristic group of baryons(qqq) and anti-baryons($\bar{q}\bar{q}\bar{q}$)

[1] Strong, weak and electromagnetic forces
[2] Colour confinement

or by having a colour and its anticolour present, which creates the group of mesons($q\bar{q}$).

Table 2: Leptons in the standard model

Leptons				
	Particle		Q[e]	$\frac{mass}{GeV}$
1. Gen.	electron	e^-	-1	0.005
	neutrino	ν_e	0	$< 10^{-9}$
2. Gen.	muon	μ^-	-1	0.106
	neutrino	ν_μ	0	$< 10^{-9}$
3. Gen.	tau	τ^-	-1	1.78
	neutrino	ν_τ	0	$< 10^{-9}$

The other group consists of leptons(l)(see Table 2). They only interact through the weak and the electromagnetic force. Each generation consists of a lepton of charge -1 and a corresponding EM neutrally charged neutrino. The electron has the lowest energy of all charged leptons. This makes the electron stable while the higher generation particles decay to lower energy particles.

The leptons of one generation, namely the charged lepton and its corresponding neutrino are called a lepton family. A lepton of a family counts as 1 to its corresponding lepton family number whilst a anti-lepton counts as -1.

Table 3: Fundamental forces

Force	Strength	Boson		Spin	Charge	$\frac{mass}{GeV}$
Strong	1	gluon	g	1	0	0
Electromagnetism	10^{-3}	photon	γ	1	0	0
Weak	10^{-8}	Z boson	Z	1	0	80.4
	10^{-8}	W boson	$W^\pm$	1	± 1	91.2

The particles of the SM interact through the 3 fundamental forces of the SM. In these interactions, particles called bosons are being exchanged which are the carriers of their respective force (see Table 3).
As mentioned above, only quarks can interact through the strong force, in which they exchange gluons. Gluons are massless and EM neutrally charged. The strong force has the biggest coupling strength of 1 (though it decreases

with higher energies as a result of gluon-gluon self interaction loops, which interfere negatively in perturbation theory)[2]. A gluon carries colour charge and hence can change the colour of a quark but it conserves its flavour. The strong interaction has an underlying gauge symmetry of SU(3). Therefore, it can be derived that colour charge is conserved through the strong interaction[3].
The electromagnetic(EM) force is propagated through the photon. It carries zero charge and no invariant mass. Exclusively charged particles can interact through the electromagnetic force. The coupling strength is $\alpha \approx \frac{1}{137}$, contrary to the strong force the coupling constant increases with higher energies[2]. This difference stems from the fact that photon-photon interaction loops are not allowed whereas gluon-gluon interaction loops are. In perturbation theory this results in only positive terms being added to the coupling strength. The underlying gauge symmetry is of SU(1). The electromagnetic force also conserves flavour.
The weak force has two types of bosons. The bosons of the weak force are the only fundamental bosons to have an inertial mass.
First we will discuss the EM neutral Z boson. Even though the Z boson belongs to the weak force it, it also has an electromagnetic part additionally to the weak force part[4]. It follows directly, that the Z boson couples weaker to uncharged particles.
The other boson of the weak force is the W boson. In the classical SM, the only way particles can change flavour is through the weak force by emitting or absorbing W boson. It is important to notice that, besides of having an invariant mass, the W boson is the only boson with a non zero charge ($Q_{W^\pm} = \pm 1e$). In the gauge symmetry of the weak force the $W^\pm$ are actually the creation and annihilation operators of said symmetry[5].
An important characteristic of the weak force is that it exclusively couples to lefthanded(LH) particles and righthanded(RH) antiparticles (describing chirality states)[6].
The chirality operators for left- and righthandedness are:

LH: $\frac{1}{2}(1 - \gamma^5)$, RH: $\frac{1}{2}(1 + \gamma^5)$

As a consequence RH particles and LH anti-particles can't couple to the W boson at all. This also results in charged RH particles and LH anti-particles

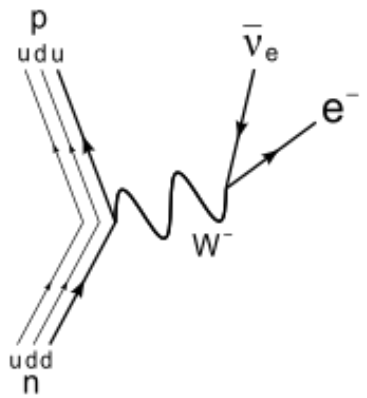
[3]E.g. through Gell-Mann matrices
[4]$Z \rightarrow EM_{part} + W^3$, [2]
[5]$W^\pm = W_1 \pm iW_2$
[6]In the ultrarelativistic limit helicity and chirality eigenstates are the same

to couple to the Z boson only through the electromagnetic part of the itself, while uncharged RH particles and LH anti particles (e.g. RH ν, LH $\bar{\nu}$) don't couple with the EM force nor the weak force.

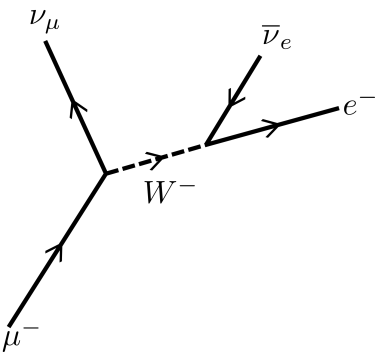
1.2 Interaction rules

Now, we will establish the general rules for interactions in the SM.

Baryon number is conserved
As we already established before, the only interaction that can change flavour is the weak force through the W boson. We directly see that all other interactions baryon number has to be conserved. So any up-type quark can be changed to a down-type quark and backwards by emitting or absorbing a W boson. In the end however, there are still 3 quarks which form a baryon[7], even though it changed its type and charge. A well known example is the beta decay, where a down quark in a neutron decays into an up quark to form now a proton(e.g. see Figure 1a). We easily see that the baryon number is conserved.



(a) Feynman diagram of the β-decay



(b) Feynman diagram of a μ-decay

Figure 1: Certain diagrams of decays

Lepton family number is conserved
According to the SM lepton family number is conserved. As all interactions

[7]Pentaquarks($qqqq\bar{q}$) and other exotic states excluded

beside the W conserve particle flavour, it is easy to see that lepton family number is conserved.
Whenever a lepton interacts with a W boson, it just changes a lepton to its corresponding lepton neutrino and or the other way around (e.g. see Figure 1b).

2 Physics beyond the SM

2.1 Neutrino Oscillation

Classically the SM considers neutrinos to be massless. While this assumption works well for a lot of cases, we know nowadays that at least two of the three neutrinos have to have mass[8]. Neutrinos are known to oscillate between all three states of flavour, as the eigenstates of flavour are not eigenstates of mass. As a consequence ν_e, ν_μ and ν_τ are not fundamental particle states but a mixture of the mass eigenstates ν_1, ν_2 and ν_3. They are connected through the PMNS matrix:

$$\begin{pmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \end{pmatrix} = \begin{pmatrix} U_{e_1} & U_{e_2} & U_{e_3} \\ U_{\mu_1} & U_{\mu_2} & U_{\mu_1} \\ U_{\tau_1} & U_{\tau_2} & U_{\tau_3} \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{pmatrix} \tag{1}$$

As a result, neutrinos propagate as a superposition of all mass eigenstates. Additionally, we can describe the PMNS matrix through three mixing angles θ_{12}, θ_{13} and θ_{23} and a complex phase δ [9]. The electron superposition looks then like this:

$$|\nu_e\rangle = U_{e_1} |\nu_1\rangle e^{-i\Phi_1} + U_{e_2} |\nu_2\rangle e^{-i\Phi_2} + U_{e_3} |\nu_3\rangle e^{-i\Phi_3} \text{ with } \Phi_i = E_i \times t$$

As a result lepton family number is not a conserved quantity anymore as neutrino flavour oscillates over time.
We can calculate the probability for a neutrino to transition from flavour α to β like:

$$\begin{aligned} P(\nu_\alpha \to \nu_\beta) = &2Re\left(U_{\alpha_1} U^*_{\beta_1} U^*_{\alpha_2} U_{\beta_2} e^{-i(\Phi_1-\Phi_2)}\right) \\ &+2Re\left(U_{\alpha_1} U^*_{\beta_1} U^*_{\alpha_3} U_{\beta_3} e^{-i(\Phi_1-\Phi_3)}\right) \\ &+2Re\left(U_{\alpha_2} U^*_{\beta_2} U^*_{\alpha_3} U_{\beta_3} e^{-i(\Phi_2-\Phi_3)}\right) \end{aligned} \tag{2}$$

An important thing to note is, that if any elements of the PMNS matrix are complex, this process is not invariant under time reversal ($t \to -t$)[10]
$P(\nu_\alpha \to \nu_\beta) \neq P(\nu_\beta \to \nu_\alpha)$.

[8] The mass difference between neutrinos is non zero: $m_i - m_j = \Delta m_{i,j} \neq 0$, $\forall j \neq i$

[9] Measurements: $\theta_{12} \approx 35°$, $\theta_{13} \approx 10°$, $\theta_{23} \approx 45°$ [3], [4]

[10] The probability does not change, if we add a complex phase to the PMNS matrix, just if one of the elements has a phase different from the others

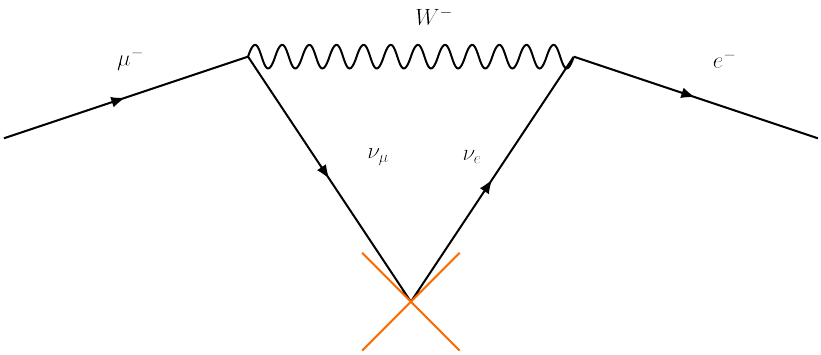


Figure 2: Process that violates lepton family number conservation through neutrino oscillation

Nowadays it's a well accepted fact that lepton family number gets violated through neutrino oscillation.
But why should flavour oscillation be exclusive to neutrinos?
Maybe there are ways for the EM charged leptons as well to directly transition to another lepton family[11]?

2.2 New physics

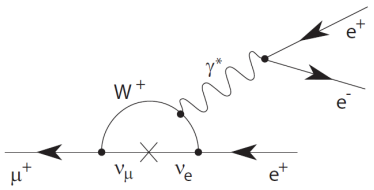
As a consequence of neutrino oscillation, lepton flavour is a broken symmetry. The SM has to be adapted to include lepton flavour violation (LFV) and massive neutrinos. LFV is also expected for charged neutrinos.
Although, it has yet to be determined how LFV violation exactly works to which scale it exists.

This may raise the question on why charged LFV has never been observed yet. This is especially surprising as the mixing angles of the neutrinos have been measured to be big.
There are two reasons why charged LFV is strongly surpressed: The first is that charged leptons are much heavier than neutrinos and the other that the mass differences between neutrino flavour are tiny compared to the W boson mass.

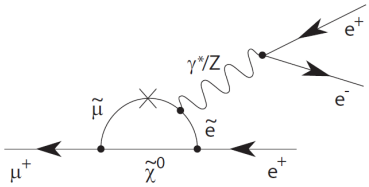
In the classical SM, charged LFV is already forbidden at tree level. Though it can be induced indirectly through higher order loop diagrams (using neutrino oscillation). By adding new particles beyond the SM, we generate new ways

[11] Maybe also possible for quarks?

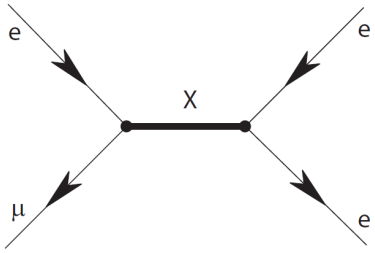
for LFV in the charged sector to happen. As LFV is naturally generated in many models beyond the SM, finding charged LFV is a strong hint for new physics.



(a) LFV through neutrino oscillation



(b) LFV by using supersymmetric particles



(c) LFV at tree level

Figure 3: Charged LFV

One way charged LFV can occur is through supersymmetric particles (see Figure 3b). By observing charged LFV supersymmetry would gain new importance.
Together with supersymmetric models, other extensions of the SM such as left-right symmetric models, grand unified models, models with an extended Higgs sector and models where electroweak symmetry is broken dynamically are all good candidates to explain charged LFV and most importantly experimentally accessible in a large region of the parameter space.

3 $\mu \to eee$ decay

3.1 Kinematics

The two most prominent charged LFV decays are $\mu \to e\gamma$ and $\mu \to eee$. Here the latter is chosen as more diagrams beyond the SM contribute. Namely tree diagrams, Z penguin and box diagrams. This offers the possibility to test more models.

Possible ways for the decay $\mu \to eee$ to occur are shown in Figures 3a, 3b, 3c.

Still some simplifications are made as it is assumed that only the tree and the photon diagram are relevant. [5]

This gives us a Lagrangian of:

$$L_{LFV} = \left[\frac{m_\mu}{(\kappa+1)\Lambda^2}\overline{\mu_R}\sigma^{\mu\nu}e_L F_{\mu\nu}\right]_{\gamma-\text{penguin}} + \left[\frac{\kappa}{(\kappa+1)\Lambda^2}(\overline{\mu_L}\gamma^\mu e_L)(\overline{e_L}\gamma_\mu e_L)\right]_{\text{tree}} \tag{3}$$

If we neglect signal and background we can use momentum conservation as the decay happens rather quickly. As a result the total sum of all particle momenta should be equal to zero:

$$|\vec{p}_{tot}| = \left|\sum \vec{p}_i\right| = 0 \tag{4}$$

The particles resulting in the decay lie all in a plane. The resulting positrons and electrons are in the energy range of (0-53)MeV.

3.2 Background events

Below is a summary of all the different types of background considered in the experiment.

3.2.1 Internal conversions

The event $\mu \to eee\nu\nu$ results in the same particles seen by the detector as the event we are searching for[12]. As a result it proves to be quite challenging to separate the two.
By using momentum conservation, it becomes possible to differentiate the

[12]Neutrinos are invisible to our detector

$\mu \to eee$ and the $\mu \to eee\nu\nu$ events. In the muon rest frame the total momentum is zero and the energy of the resulting particles is equal to muon rest energy.
By reconstructing the energy and momenta of the three e we can check if their momenta add up to zero and their energies equal the muon rest energy. If not we can assume that there are additional neutrinos. This differentiation between the two events is crucial for the experiment as the $\mu \to eee\nu\nu$ events pose the most serious background for $\mu \to eee$ decay measurements.
As a result, our detector needs a very good energy resolution to consistently make it possible to differentiate between the two events as neutrino energies and momenta are very small.

3.2.2 Michel decay

The biggest contributing background however stems from another decay called Michel decay, that is also allowed in the classical SM. As we use a beam of positive muons the corresponding Michel decay looks as follows: $\mu^+ \to e^+\nu\bar{\nu}$.
Contrary to the events before this one does not produce any em negatively charged particles. This makes these events easily distinguishable from our wanted events. As a result they only enter our data in form of a potential background through wrongly constructed tracks.

3.2.3 Radiative muon decay

This is the case where $\mu \to e^+\gamma\nu\nu$. If the photon produced in this event has high enough energies and creates a matter antimatter pair in the target region ($\gamma \to e^-e^+$), it can create a similar signature than the searched event. They contribute to the accidental background, as equal to the searched event no neutrinos are produced. To minimize these effects, the material in both the target and detector is minimized and a vertex constraint is applied.

3.2.4 BhaBha scattering

Another way how background can get produced is when positrons from muon decays or the beam itself scatter with electrons in the target material. Consequently they share a common vertex and together with an ordinary muon decay it can look similar as our searched $\mu \to eee$ event. This contributes to the accidental background.

3.2.5 Pion decays

Certain pion decays also lead to indistinguishable signature as our searched event, the most prominent being the $\pi \to eee\nu$ and $\pi \to \mu\gamma\nu$ decays. The later only produces a similar signature if produced photon converts through pair production to an electron and a positron.
However, as only a negligible portion will actually contribute to the background, as there is only a small branching fraction and the momenta and energy of the produced particles have to match up with the criteria mentioned in section 3.1.

3.2.6 Analysis of the background

The results of simulations indicate that the effect of purely accidental background contributions are small for a high enough energy resolvent detector. [5]
The most relevant background stems from the $\mu \to eee\nu\nu$ events. This problem can only be tackled by using a very precise total energy resolution of $\sigma_E = 1MeV$ at the aimed sensitivities.

4 Mu3e experiment

4.1 Requirements

The ultimate goal of this experiment is to observe a $\mu \to eee$ event. As we strive for a sensitivity of 10^{-16} , we should be able to observe this process if its branching ratio would be higher than our sensitivity. Otherwise, we want to exclude a branching ratio $> 10^{-16}$ with a 90% certainty.
To get to this sensitivity, more than $5.5 \cdot 10^{16}$ muon decays have to be observed. To reach this goal within one year, a muon stopping rate of $2 \cdot 10^{9} Hz$ in combination with a high geometrical acceptance as well as a high efficiency of the experiment is required.

4.2 Phase I

Phase I of the experiment serves as an exploratory phase to gain more experience with the new technology and validate the experimental concept. At the same time it already strives to produce competitive measurements with a sensitivity of 10^{-15}. [13] This will be done, by making use of the already existing muon beams at PSI with around 1-1.5 $\cdot 10^{8} Hz$ of muons on target. The lowered sensitivity also allows for some cross-checks as the restrictions on the system are much more relaxed than in phase II.

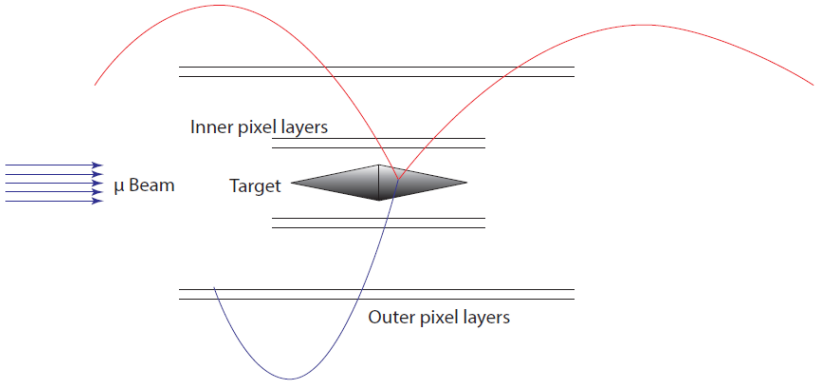
4.3 Phase II

Phase II strives to reach the maximum sensitivity of 10^{-16}. To achieve this in a reasonable timeframe, a new beamline will be used which delivers more than $2 \cdot 10^{9} Hz$ of muons.

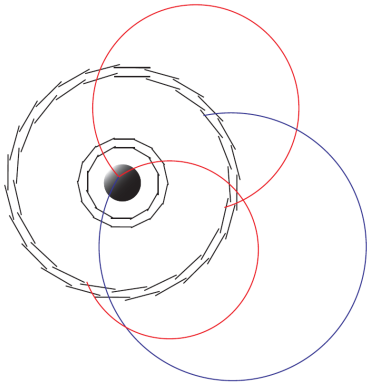
4.4 Experimental setup

The detector is of cylindrical shape around the beam. It has a total length of around $2m$ and is situated inside a $1T$ solenoid magnet with $1m$ of inner radius and a total length of $2.5m$. This form was chosen to cover as much phase space as possible. For an unknown decay such $\mu \to eee$, it crucial to have a high order of acceptance in all regions of phase space. There are only two kind of tracks that get lost. The first one are up- and downstream tracks and the second one are low transverse momenta tracks (no transversing of enough detector planes to be reconstructed).

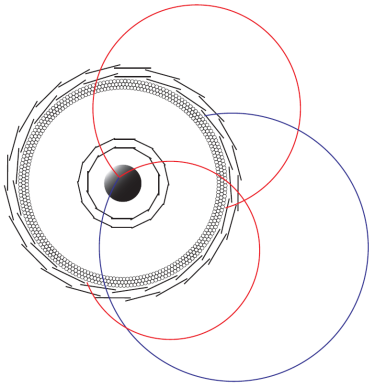
[13] Current experiments are in the 10^{-12} sensitivity range



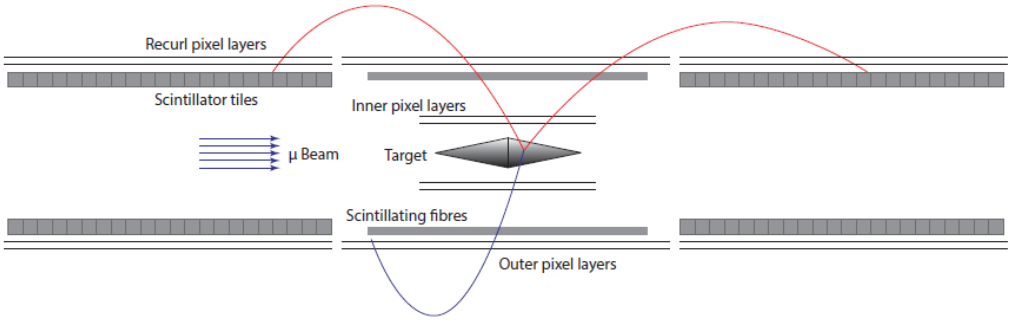
(a) Setup of the detector in the first part of phase I



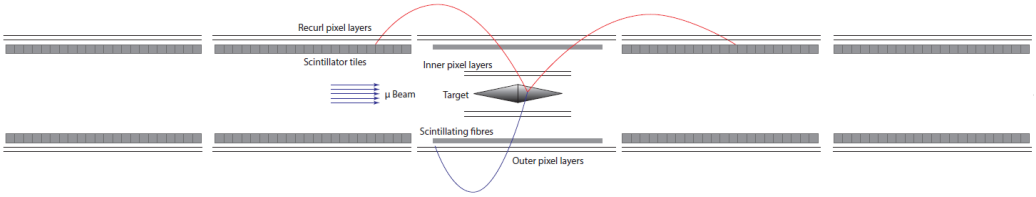
(b) Tracks in the detector in the first part of phase I



(c) Tracks in the detector in the second part of phase I and phase II



(d) Setup of the detector in the second part of phase I



(e) Setup of the detector in phase II

Figure 4: Setup of the detector during different phases of the experiment

As seen in figure 4e, the final version of the detector can be divided into 5 separate parts in the longitudinal direction. There is the central part with the target, two inner silicon pixel layers, a fibre tracker and two outer silicon layers. The forward and backward parts, called recurl stations, consist only of a tile timing detector surrounded by two silicon recurl layers. A big advantage of this layout is that even a partially constructed detector (gradually over phase I to phase II parts get added) can give us competitive measurements.

The target itself is a big surfaced double cone with a surface length of $10cm$ and a width of $2cm$. The target was chosen specifically to be of this shape to facilitate separating tracks coming from different muons and hereby also helping to reduce accidental background.
The two inner detector layers, also called vertex layers, span a length $12cm$. The innermost layer consists of 12 tiles while the outer vertex layer consists of 18 tiles. The tiles are each of $1cm$ width, with the inner layer having an average radius of $1.9cm$, respectively $2.9cm$, and a pixel size of $80 \times 80\mu m^2$. [6], [7], [8]. They are supported by two half cylinder made up of $25\mu m$ thin Kapton foil mounted on plastic. The detector layers itself are $50\mu m$ thin and cooled by gaseous helium. The vertex detectors are read out at a rate of $20MHz$, giving us a time resolution of $20ns$.
After the vertex layers the particles pass through the fibre tracker (see Figure 4c, 4e). It is positioned around $6cm$ away from the center. Its main job is to provide accurate timing information for the outgoing electrons and positrons. It consists of three to five layers, each consisting of $36cm$ long and $250\mu m$ thick scintillating fibres with fast silicon photomultipliers at the end. They provide us a timing information of less than a $1ns$.
Next the outgoing particles encounter the outer silicon pixel detectors. They are mounted just after the fibre detector with average radii of $7.6cm$ and $8.9cm$. The inner layer has 24 and the outer has 28 tiles of $1cm$ length. The active area itself has a length of $36cm$. Similarly to the vertex detectors, they are mounted on $25\mu m$ thin Kapton foil with plastic ends.
The stations beam up- and downwards only consist of the outer pixel detector layers as well as a timing detector. While the silicon detector are the same as in the central station, the timing tracker was chosen to be much thicker than the fibre detector in the central station. It consists of scintillating tiles with dimensions of $7.5 \times 7.5 \times 5mm^3$. They provide an even better time resolution than the fibre tracker in the center. Incoming particles are supposed to be stopped here. The outer stations are mainly used to determine the momenta of the outgoing particles and have an active length of $36cm$ and a radius of around $6cm$.

4.5 The problem of low longitudinal momentum recurlers

As explained in section 4.4, the outgoing particles are supposed to recurl back into the outer stations of the detector to enable a precise measurement of the momentum. A problem arises if the particles have almost no momentum in the beam direction. Then they can recurl back into the central station and cause additional hits there. As the the central station is designed to let particles easily pass through, they can recurl inside the central station many more times without getting stopped. As we have a $20ns$ time window for the readout of the pixel detectors, we need a very reliable way to identify and reconstruct these tracks as recurling particles as otherwise they look exactly like newly produced particles coming from our target. As one can imagine, this influences the precision of our measurements by a big margin. So, finding a way to identify these low beam direction momentum particles consistently is of great importance as it is crucial for the experiment to reduce the background as much as possible.

There is already an existing software to reconstruct particle tracks. However, it struggles to find the right tracks for a lot of the particles recurling back into the center station.
These recurlers will typically leave eight hits or more, four (one on each silicon pixel detector layer) when initially leaving the detector and another four when initially falling back in. It is possible for these recurlers to produce even more hits when leaving the detector again but for this thesis we will be only focusing on these 8 hit tracks.
The current reconstruction algorithm works by fitting helix paths with a χ^2 method onto the 8 hits.
However, experience has shown that often the fit with the lowest χ^2 isn't necessarily the right track. If we increase the χ^2 limit value to some arbitrary limit, we get a selection of several possible tracks per particle. Without any additional tools however, it is impossible to figure out if the right track is in the selection[14] and if yes which one of them correct track is.

[14] Based on detector efficiency it is possible for a particle to leave less than 8 tracks and therefore not be reconstructed by the algorithm

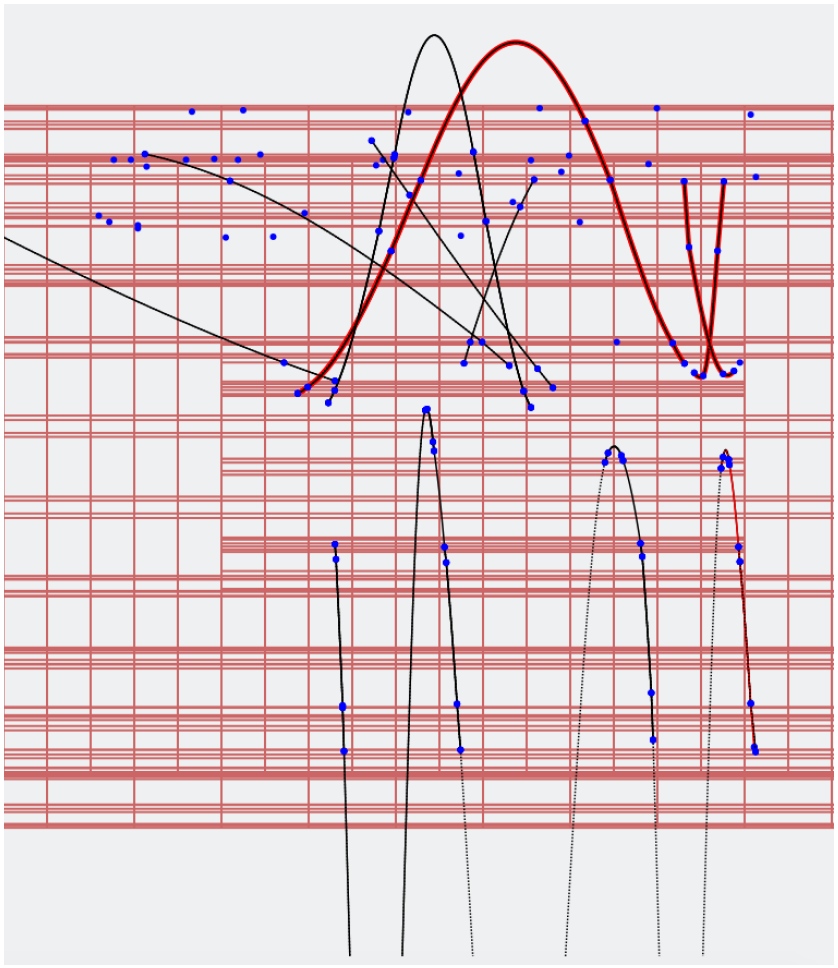


Figure 5: Particle recurling back into the center station (highlighted)

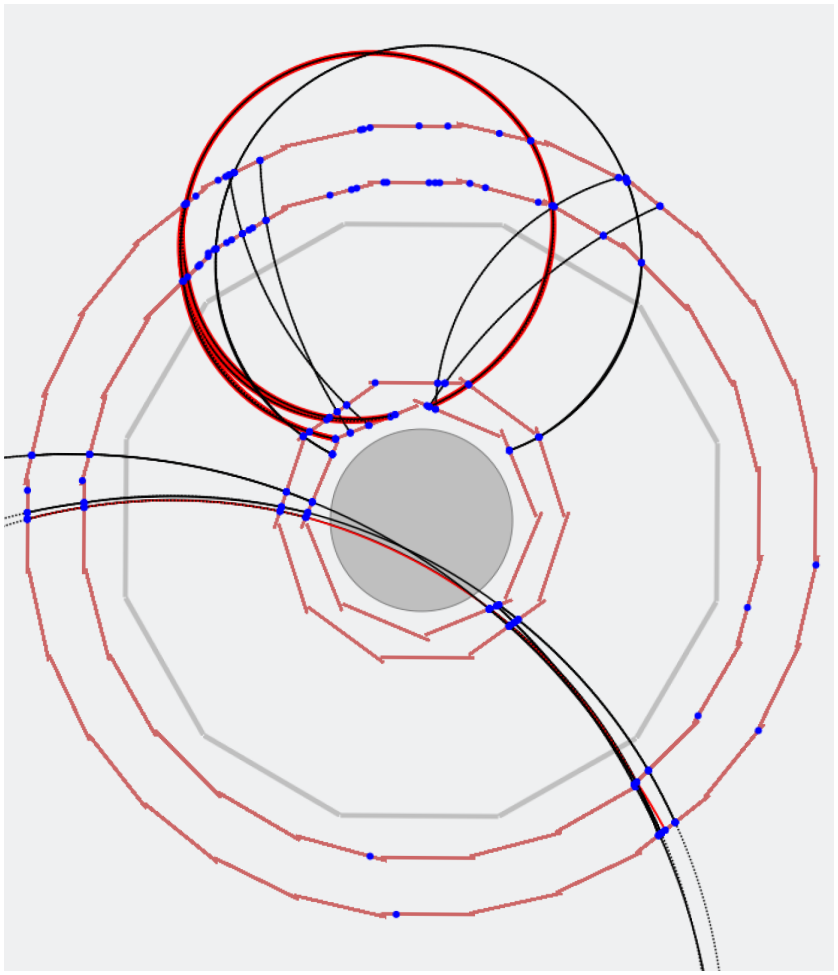


Figure 6: Particle recurling back into the center station (highlighted)

5 Machine learning

5.1 Introduction

Machine learning has already proven itself to be very successful in resolving many problems in numerous other areas of science and also in the private sector. Based on these promising results, scientists are eager to study the potential of machine learning in physics.

There are several sections of machine learning. In this paper, we will focus mainly on neural networks(NN), with special attention to recurrent neural networks(RNN) [9], [10] and XGBoost(XGB) [11] models with boosted decision trees.

5.2 Artificial neural networks

5.2.1 General concepts

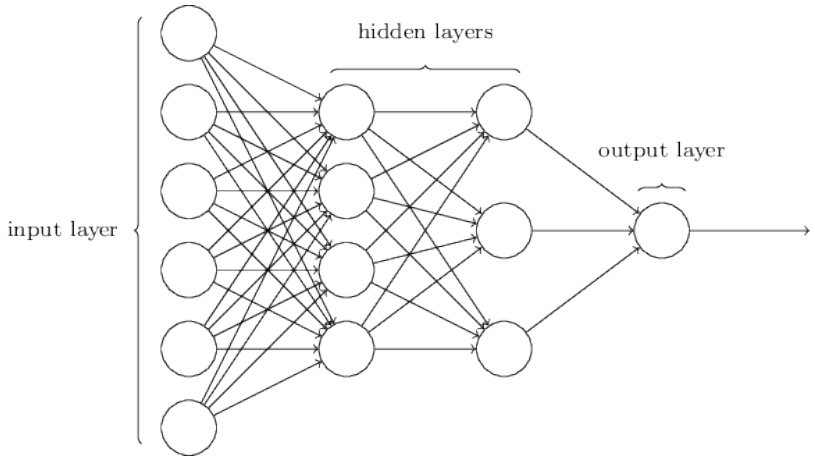
The fundamental concept behind artificial neural networks is to imitate the architecture of the human brain. They can be used for classification problems as well as regression problems. In its most simple form, it can be thought of some sort of mapping from some input to some target. For this thesis two neural networks of a special subtype of neural networks, called recurrent neural networks, were used. All of the networks used in this thesis were written in the python library Keras [12] with a Tensorflow [13] backend. In this section the basic principles of neural networks will be explained.

A neural network consists of many neurons organized in layers as seen in figure 7a. Each neuron is connected to every neuron in the neighbouring layers, while each of these connections has a specific weight assigned to it.
In its most basic form, each neuron calculates a weighted sum to all of its inputs and then applies a bias to it . In addition, each neuron has an activation function, which will be applied at the end of the calculation (see also figure 7b):

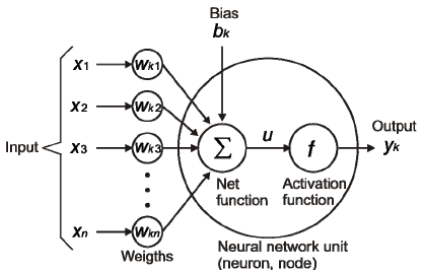
$$y = f_{activation}\left(\sum_{i=1}^{n}(x_i \cdot w_i) + b\right) \tag{5}$$

This is done to create non linearity in the system. Later, some more complex architectures of neurons will be presented.
The first layer, also called input layer, is always defined by the number of inputs, with one dimension for each input. The dimensions of the following

layers (excluding the last one), which are also called hidden layers, can be chosen to be an arbitrarily number. The number of dimensions of the last layer, also called output layer, is determined by the dimensionality of the prediction. The number of hidden layers, and their corresponding dimension, changes the performance of the system.



(a) Architecture of a neural network



(b) Neuron

Figure 7: Neural network architecture

There is no way of knowing how many dimensions and layers will give you the best performance, as one can only define general effects of what happens when they are being modified. Generally, increasing the number of layers enables the system to solve more complex problems, while more dimensions make the system more flexible. However, even these general guidelines are to be applied with caution. For example, adding too many layers can cause the system to train exceedingly slow, whilst adding to many neurons with a too small training set can result in overfitting[15]. Depending on the problem

[15]When a system performs well on the training set but poorly on the test set

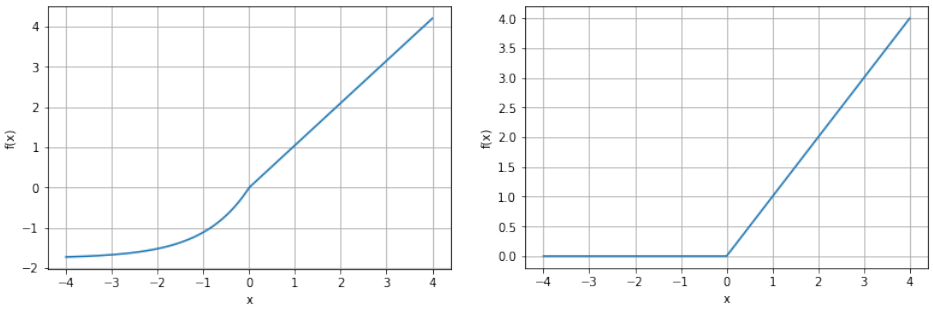
and the data given, each has its own optimal configuration. By gaining more experience with NN, people can take better guesses where to start. However, in the end it always results in some sort of systematic trial and error to find the optimal configuration.

5.2.2 Activation functions

The two RNN's used in this thesis both use *selu*'s [14] as well as *tanh* and *relu*'s as their activation functions.

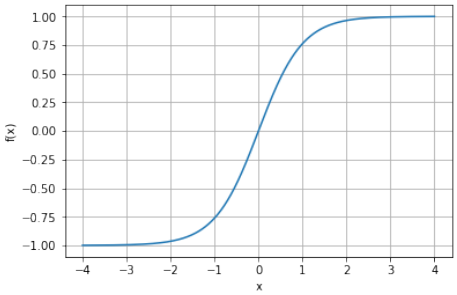
$$selu(x) = \lambda \begin{cases} x, & \text{if } x < 0 \\ \alpha e^x - \alpha, & \text{otherwise} \end{cases} \tag{6}$$

$$relu(x) = MAX(0, x) \tag{7}$$



(a) Selu, elu activation function

(b) Relu activation function



(c) Tanh activation function

Figure 8: Activation functions

Where λ and α are fixed parameters[16]. Selu's have the advantage of nor-

[16]For standard scaled inputs (mean = 0, stddev. = 1.0): $\alpha \approx 1.6732$, $\lambda \approx 1.0507$

malizing the output. As a rule of thumb, normalized inputs usually tend to give better results (The output of the neurons are the input of other neurons). Using a *tanh* was the standard approach for a long time although it has some disadvantages over the other activation functions. This is because its slope becomes really small for large numbers, which slows down training noticeably.

5.2.3 Concepts of training

The neural network is trained with a sample of events. This sample consists of a few input parameters and a training target, which is the value the neural network will be trained to predict. Three important terms for the training of a neural network are epochs, batch size and loss function.
An epoch refers to one training iteration, where all of the training samples get used once and the weights and biases get modified to fit the wanted targets better. Usually a system is trained over many epochs until the weights and biases stay approximately constant at their optimal values.
Batch size refers to the number of examples that are given to the system at once during the training. Batch size should neither be chosen too small, e.g. small batch sizes train slower, nor too big, some randomness is wanted. Experience shows, that a reasonable batch size usually lies between 10 to 100 examples per batch. It is important to note that by decreasing batch size we make the minimum of the mapping we want to find wider. This makes finding the general area of the minimum easier. However if the minimum gets too wide, the slope gets to small to reach the minimum in a reasonable time. On the other side by increasing the batch size too much, the minimum gets exceedingly narrower and it possible to continuously keep "jumping" over the minimum with every training step performed.

5.2.4 Loss functions

To train the system, we need some way to parametrize the quality of our predictions. To account for that we use a loss function. A loss function takes the predicted values of the system and the targeted values to give us an absolute value of our performance. There are various loss functions. In the two RNN's "mean squared error"(MSE, formula 8) and "binary crossentropy"(BC, formula 9) were being used. The goal of every NN is to minimize the loss function.

$$L(w,b) = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i(w_i,b_i) - Y_i)^2 \tag{8}$$

$$L(w,b) = -\frac{1}{n}\sum_{i=1}^{n}\Big(Y_i log(\hat{Y}_i(w_i,b_i)) + (1-Y_i) log(1-\hat{Y}_i(w_i,b_i))\Big) \tag{9}$$

With:

- w_i are the weights of the system
- b_i are the biases of the system
- $\hat{Y}_i(w_i,b_i)$ are the predicted values of the system
- Y_i are the targets for the predictions
- $L(w,b)$ is the loss over n events

5.2.5 Stochastic gradient descent

There exist several methods to minimize the loss. The most simple one being stochastic gradient descent(SGD). When performing SGD we can calculate the gradient and just apply it to our weights and biases. By doing this repeatedly, we will eventually end up in a minimum[17].

5.2.6 Stochastic gradient descent with Momentum

Trainings algorithm working with momentum are basically an improved version of SGD. To circumvent the problem of getting stuck in any minimum, our gradient can build up momentum of the past gradients. This is done by adding a momentum term to the applied changes to the weights and biases. The momentum is an exponentially decaying average over past gradients. This generally trains faster than SGD and has less potential to get stuck in local minima.

5.2.7 RMSProp

Another improved version of SGD is RMSProp. The RMSProp algorithm scales the learning rate of each individual parameter by an exponentially decaying average of the past squared gradients. This has the effect, that the

[17]It is very possible to also just get stuck in a local minimum

learning rate increases if the past gradients were small to increase step size and vice versa. Additionally the average of the past squared gradients decays exponentially to prevent the step size from getting too small.

5.2.8 Adam

The most commonly used algorithm however, is the Adam algorithm [15], which stands for Adaptive Moment estimation, training algorithm (see formulas 10). Is is essentially a combination of Momentum and RMSProp and takes the best of both. It is also the one used to train both RNN's of this thesis as it converges the quickest and most reliable to the global minimum. The algorithm contains two moments. The first moment is an exponentially decaying average of past gradients as in Momentum while the second moment is an exponentially decaying average of past squared gradients as in RMSProp.

Initial state:

$$V_{d_W} = 0, S_{d_W} = 0, V_{d_b} = 0, S_{d_b} = 0$$

On iteration t:

$$\begin{aligned}
V_{d_W} &= \beta_1 V_{d_W} + (1-\beta_1)dW, V_{d_b} = \beta_1 V_{d_b} + (1-\beta_1)db \\
S_{d_W} &= \beta_2 S_{d_W} + (1-\beta_2)dW^2, S_{d_b} = \beta_2 S_{d_b} + (1-\beta_2)db^2 \\
V_{dW}^{corrected} &= \frac{V_{dW}}{1-\beta_1^t}, V_{db}^{corrected} = \frac{V_{db}}{1-\beta_1^t} \\
S_{dW}^{corrected} &= \frac{S_{dW}}{1-\beta_2^t}, S_{db}^{corrected} = \frac{S_{db}}{1-\beta_2^t} \\
W_{t+1} &= W_t - \alpha \frac{V_{dW}^{corrected}}{\sqrt{S_{dW}^{corrected}} + \epsilon} \\
b_{t+1} &= b_t - \alpha \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}} + \epsilon}
\end{aligned} \tag{10}$$

With:

- V_{d_W}, V_{db} correspond to the Momentum part
- S_{d_W}, S_{db} correspond to the RMSProp part
- ϵ this constant is chosen to be very small and only there to prevent division by 0 (usually $\epsilon = 10^{-8}$)

- α: learning rate (needs to be tuned according to the problem)
- β_1 decay constant of the Momentum part (usually $\beta_1 = 0.9$)
- β_2 decay constant of the RMSProp part (usually $\beta_2 = 0.999$)

5.2.9 Decaying learning rate

To counteract the problem of "jumping" over the minimum repeatedly, some NN also use a decaying learning rate during their training. By using this, the step size gets smaller with every consecutive step which should in principle result in the step size converging to zero when reaching the global minimum. Most NN's, as well as the two RNN's used in this thesis, usually don't use a decaying learning rate as the Adam algorithm on its own already performs well enough.

5.2.10 Batch normalisation

Another important technique often used in NN is Batch Normalisation [16], [17]. By performing Batch Normalization we normalize and center the input around zero in between every layer of the NN. Batch Normalization has proven to be a potent technique to make NN train faster and even perform better.

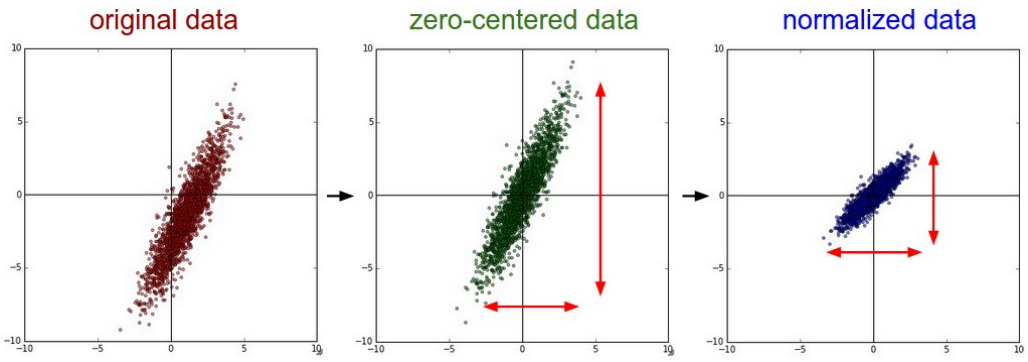


Figure 9: The effects of Batch Normalization on data

5.3 Recurrent Neural Networks

5.3.1 General concepts

Recurrent Neural Networks(RNN) are subclass of neural networks and are specialised to deal with sequential data structures. There are various appli-

cations for RNN's such as speech recognition, music generation, sentiment classification, DNA sampling and so on. Generally normal NN don't perform that well on sequential data. One of the reasons is for example that it doesn't share features learned across different positions in the data[18]. Another problem is that the input and output don't necessarily have to have the same length every time.
It is important to note that when using RNN's what the units we called neurons before are usually called cells.
RNN's pose a much better representation of the data which also helps reducing the number of variables in the system and hereby make it train more efficiently.

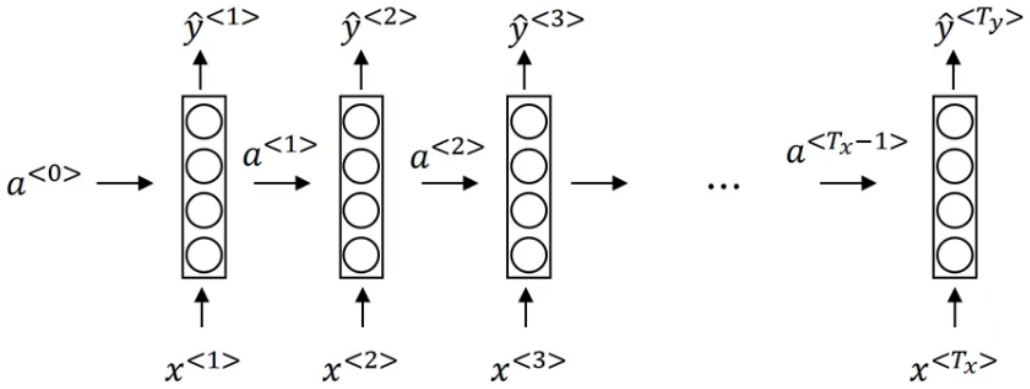


Figure 10: General RNN architecture

With:

- $x^{\langle t \rangle}$: Input at timestep t with T_x total steps
- $\hat{y}^{\langle t \rangle}$: Output at timestep t
- $a^{\langle 0 \rangle}$: Initial value given to the RNN in the first step
- $a^{\langle t \rangle}$: Information passed over from the last step

In figure 10 the general architecture of a RNN can be seen. Every step of the input data ($x^{\langle t \rangle}$) gets sequentially fed into the RNN which then generates some output $\hat{y}^{\langle t \rangle}$ after every step of the input. To share already learned information and features for future steps, $a^{\langle t \rangle}$ gets passed down as additional input into the RNN for the next step.

[18]In our experiment positions of the particles with x,y,z in the detector

5.3.2 Most common architectures

There are two concepts of how the data is fed into the system and three structures of RNN's depending on the input and output of the system.

Usually, the data is fed into the system step by step. For problems, where not the entire sequence is known already at the start, this is the only way to feed the data into the system.
If however, the entire sequence is already known at the beginning, e.g. in sequence classification, often the information is read by the system forwards and backwards. Networks with this specific architecture are called bidirectional RNN's [18]. This often increases the systems performance.
However, as with the first RNN, we wanted to predict particle tracks after leaving the detector, we could only use a one directional RNN as the whole track wasn't available. The second RNN is actually a classifier of the tracks. With the whole information available from the start, it was designed to be a bidirectional RNN.

A system has a "many-to-one" architecture, if we have a sequential input but we only care about the final output of the system, e.g. classification problems. This is the architecture used for both RNN's. With the same reasoning, if we have sequential inputs and want care about the output generated at each step, e.g. speech recognition, the architecture is called "many-to-many". A "one-to-one" architecture is basically just a regular NN.

5.3.3 Cell types

Besides the basic RNN cell type, which shall not be discussed in detail in this thesis, the two most influential and successful cell types are Long-Short-Term-Memory(LSTM) [19] cells and Gated Recurrent Units(GRU) [20]. However, in this thesis only LSTM cells will be explained in greater detail as the were the only cells used in the RNN's.

GRU's were invented with the intention to create a cell type with a similar performance to the LSTM cell while having a simpler internal structure. By being less complex as an LSTM cell a GRU cell has also less parameters to modify during training which also speeds up training.

LSTM cells (see figure 11) have many useful properties such as a forget gate, an update gate as well as an output gate. With this cell type, it is easy to pass down information for the following steps without it being altered in

a big way (Long term memory). However, there are also ways built in to update this passed down information with new one (Short term memory). Even though GRU's are gaining more and more traction, LSTM-cells are still widely considered to be the most successful type of cells.

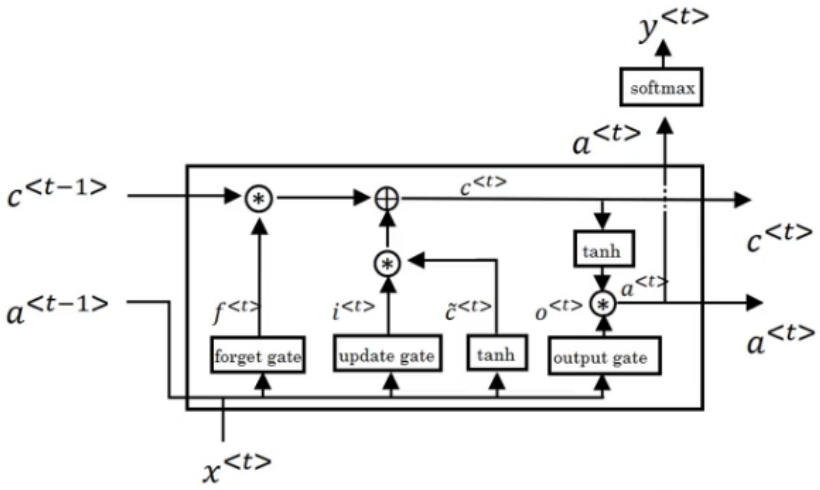


Figure 11: Architecture of a LSTM cell

The math behind the LSTM cell looks as follows[19]:

$$\begin{aligned} \tilde{c}^{\langle t \rangle} &= tanh(W_c \left[a^{\langle t \rangle}, x^{\langle t \rangle}\right] + b_c) \\ \Gamma_u &= \sigma(W_u \left[a^{\langle t \rangle}, x^{\langle t \rangle}\right] + b_u) \\ \Gamma_o &= \sigma(W_o \left[a^{\langle t \rangle}, x^{\langle t \rangle}\right] + b_o) \\ c^{\langle t \rangle} &= \Gamma_u \cdot \tilde{c}^{\langle t \rangle} + \Gamma_o \cdot tanh(c^{\langle t-1 \rangle}) \\ a^{\langle t \rangle} &= \Gamma_o \cdot tanh(c^{\langle t \rangle}) \end{aligned} \tag{11}$$

5.4 XGBoost

XGBoost[11] is based on boosted decision trees (extreme gradient boosting). In this approach, the data samples get split using a decision tree. With every step a new tree gets created to account for the errors of prior models, which are then added to create the final prediction. A gradient descent algorithm is used to minimize loss when adding new trees.

[19]The notation used is the same as in figure 11

It's is often used as a classifier. However, it can also used in regression models. In this thesis, an XGBoost classifier was used to determine a baseline and have some comparison for our bidirectional RNN classifier.

6 Data

6.1 General information

There were two sets of data used in this thesis. First, each of the datasets were shuffled to counteract any bias given by the sequence of the data and then split into two parts. 80% was used to train the model(training set) while the remaining 20% were later used to test the model(test set).
The sets were created using a Geant4 [21] based simulation with the specific configuration of the $\mu \rightarrow 3e$-experiment configuration.

The first dataset(dataset 1) contained 46896 true 8-hit tracks of recurling particles, and each hit consisting of 3 coordinates (x,y,z).

The second dataset(dataset 2) contained 109821 tracks. These were exclusively tracks that the current track reconstruction algorithm wasn't conclusively able to assign to an event. As a result, every event contained all the preselected tracks, computed by the already existing algorithm, that were calculated to be a possible track. It is important to note that only for around 75% of the events, the true track was in this preselection. This posed an additional challenge, as one could not just simply chose the best fitting track. To assign the tracks to their corresponding events, they all carried an event number with them matching them with their event.[20]. Each track contained the coordinates of the 8 hits (x,y,z), the value of the χ^2-fit performed by the reconstruction algorithm, the event number as well as a label which told us if the track was true or false[21].

6.2 Preprocessing

6.2.1 Dataset 1

To optimize the data fed into the RNN, dataset 1 was preprocessed. In a first step, a min-max scaler with a range of $[-0.9, 0.9]$ from the python library Scikit-learn [22] was used. This particular choice of range was based on the fact that a *tanh* activation function was used in the output layer. To accommodate for its properties of being asymptotically bounded by ± 1 we chose a range of $[-0.9, 0.9]$ to make all the data easily reachable by the system. In a second step, the data got shuffled and split into the training

[20] One number for all tracks of the same events
[21] Only used for training and testing of the system

and test sets. The first four steps were used as an input for the RNN while the second four steps were our prediction target.

6.2.2 Dataset 2

Analogously to dataset 1, first the coordinates of the tracks as well as the χ^2 were scaled with a min max scaler (separate ones) with a range of $[-0.9, 0.9]$ from the python library Scikit-learn. Then the first four steps of every track were taken and fed into our first track predicting RNN. For each of the last four steps of a track we then had two sets of coordinates. One were the predicted coordinates of our RNN and the other one the coordinates given by the reconstructing algorithm. To have the information of the χ^2 fit available at each step, we created an array of shape $(\#tracks, steps, 4)$ (1 dimension for each of the coordinates and another for the χ^2 fit). However, at the spot of the x,y,z coordinates there were neither the predicted coordinates of our RNN nor the coordinates given by the reconstructing algorithm but instead the difference of the two. Our target was the truth value of each track[22].

[22] 1 = true, 0 = false

7 RNN's used

7.1 RNN for track prediction

The first RNN had the task to predict the positions of the recurled 4 hits. As input the 4 hits of an outgoing particle are used.

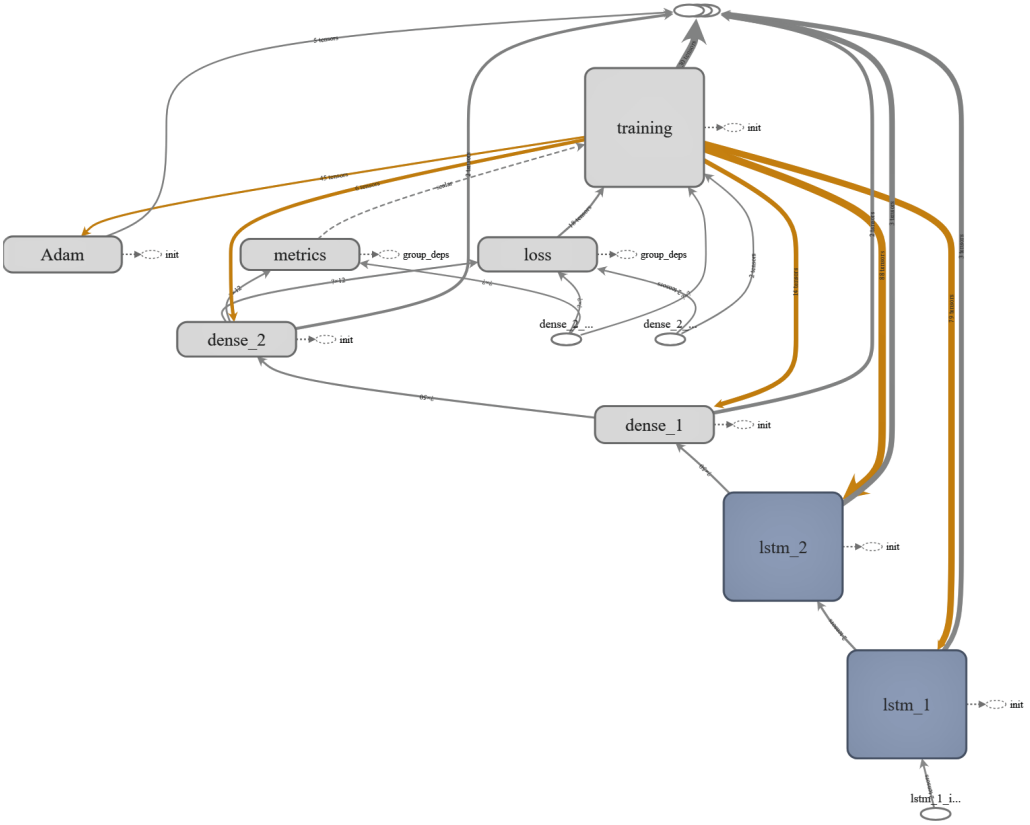


Figure 12: RNN Prediction architecture

Figure 12 shows the architecture used for the RNN track prediction. It is a one directional RNN with following layout for its layers:

1. Layer: 50 LSTM cells

2. Layer: 50 LSTM cells

3. Layer: Dense layer (50 cells)[23]

4. Layer: Dense layer (12 cells)

The optimal number of layers, cells and cell-type was found by systematically comparing RNN's that are equal besides one property (e.g. Using GRU's instead of LSTM cells). Also all the activation functions were chosen to be selu's.
The loss and metric function used were the mean squared error(mse) as this had the most similarity with an euclidian distance. The model itself was trained by an Adam algorithm.
The output was a 12 dimensional vector of the shape: $(x_5, y_5, z_5, x_6, y_6, z_6, ..., z_8)$. Note that the numeration starts with 5 as the 5^{th} hit of the track is the first one to be predicted.

7.2 RNN for classification of tracks

The second RNN was used a classifier to find the right tracks. As already described in section 6.2.2, the input data was of shape $(batchsize, 4, 4)$ with $(\Delta x_i, \Delta y_i, \Delta z_i, \chi^2)_{\text{at step i}}$.

Where:

- $\Delta x_i = x_{i,\text{preselected}} - x_{i,\text{predicted}}$, the difference between the by the original tracking algorithm preselected track and the by the RNN predicted one
- $\Delta y_i, \Delta z_i$ same as for Δx_i
- Value of the χ^2 fit

The output was then just a one dimensional vector, where 1 stands for a true track and 0 stands for a false track. The RNN itself is going to predict a number between 0 and 1, which can be interpreted as amount of confidence that it is a true track.

[23] Dense layer cells are basically just basic NN cells as explained in section 5.1

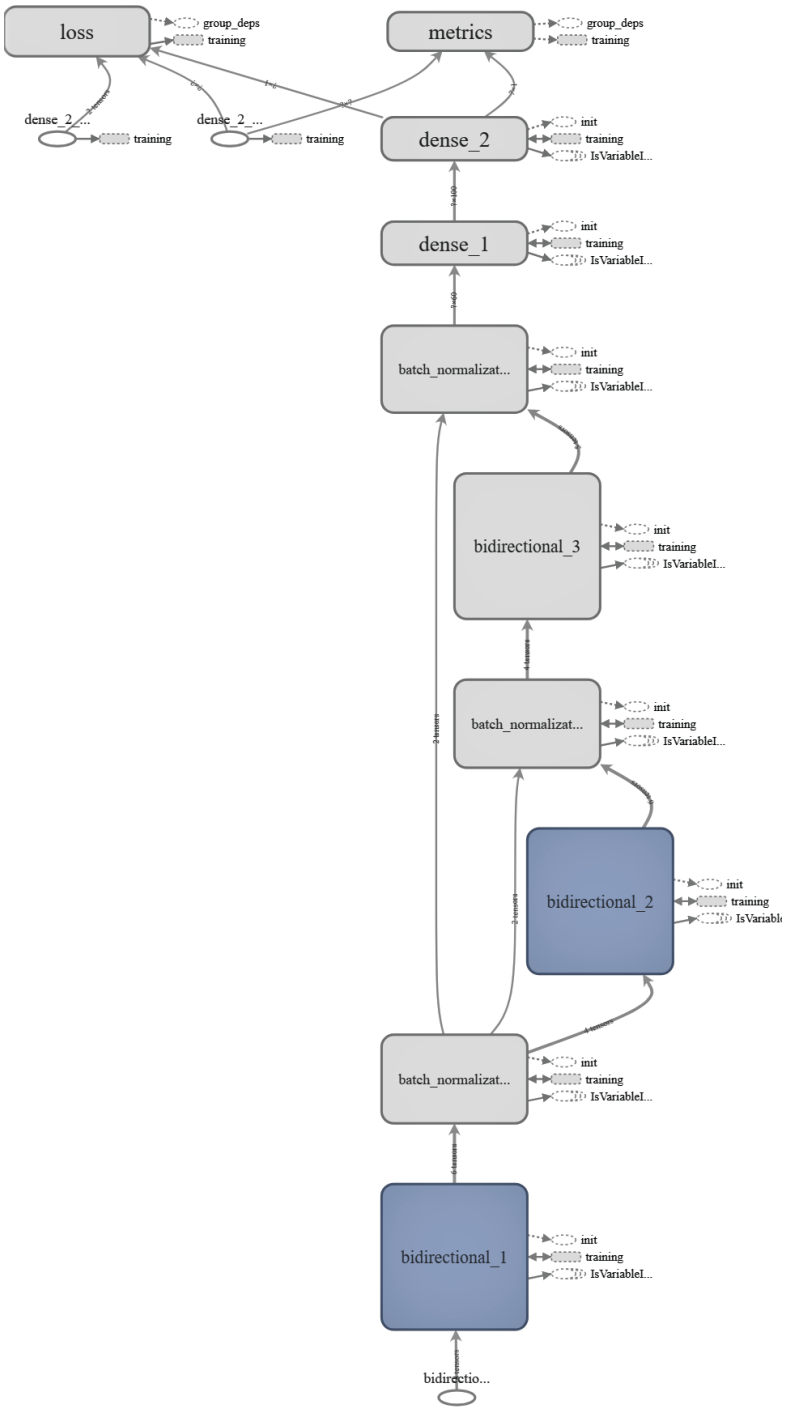


Figure 13: RNN classifier architecture

The RNN for the classification was chosen to be bidirectional and as in the RNN before LSTM cells were used. Here, a tanh was used for all the activation functions besides the last one. The last layer used a softmax activation function[24]. As tanh doesn't automatically do batch normalization, between every layer of cells a batch normalization layer was added.
The layout of the layer was as follows:

1. Layer: 30 LSTM cells (bidirectional, batch normalization)
2. Layer: 30 LSTM cells (bidirectional, batch normalization)
3. Layer: 30 LSTM cells (bidirectional, batch normalization)
4. Layer: Dense layer (50 cells, batch normalization)
5. Layer: Dense layer (1 cell, softmax activation function)

The optimal number of layers, cells and cell-type was found by systematically comparing different RNN architectures. Also, it is important to note that the second RNN is directly dependant of the first RNN. When changing the first RNN one would also have to retrain the second.

[24]Similar to a tanh but bounded between [0,1]

8 Results

8.1 Best χ^2

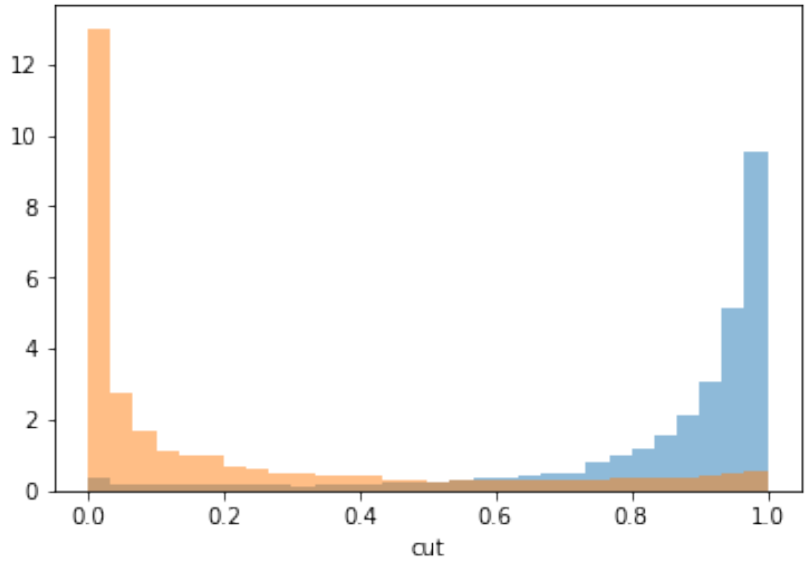
The most simple version to try to classify which one is the right path out of the preselection would be to just take the path with the smallest χ^2. Like this, we would choose the path that agrees the most with the track reconstructing algorithm that gives us our preselection. However, as already mentioned, in dataset 2 only around 75% of the events even have the true track among the ones preselected by the reconstruction[25]. In this case we would have to label all the tracks as false tracks. By simply choosing the best χ^2 we don't account for this at all. So, by default our maximum accuracy would be around 75% if the true track would really always just be the one with the best χ^2.

It turns out the accuracy of this method is only at 52.01%. So, there is a need for better algorithms to classify this problem.

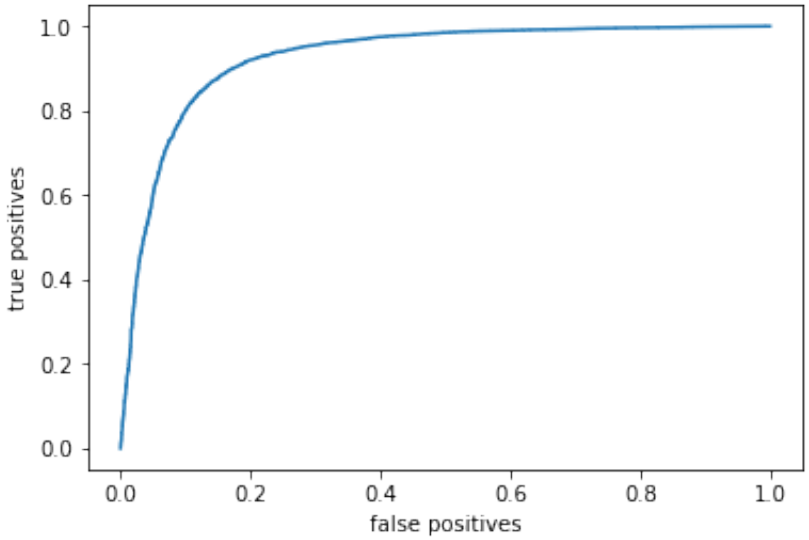
8.2 RNN classifier with RNN track prediction input

The RNN's that we put in sequence (first track prediction then classification) are a much more complex model. When trained, they were able to label all the tracks right with an accuracy of around 87.63%. Note that the 75% limit of always choosing one track for every event was exceeded[26].

[25] E.g. by not having all 8 hits as a result of detector efficiency (searches for 8 hits)
[26] Usually the one that is considered the best by the corresponding algorithm



(a) Number of false positives and false negatives depending cut



(b) ROC curve for the RNN model

Figure 14: XGBoost classifier figures

As shown in figure 14a, depending on where we apply the cut, we have a changing number of false positives and false negatives. In figure 14a, the blue bins are false positives and the orange bins are false negatives. Depending

on what is more important for the experiment[27]. One can also qualitatively judge the performance here, as in the optimal case all the false positives would gather at the area where the cut goes to 0. Analogously, we want all the false negatives to gather at the cut around 1. Here we see that this is fulfilled really well. So, already by this graph we see that the system will perform well.

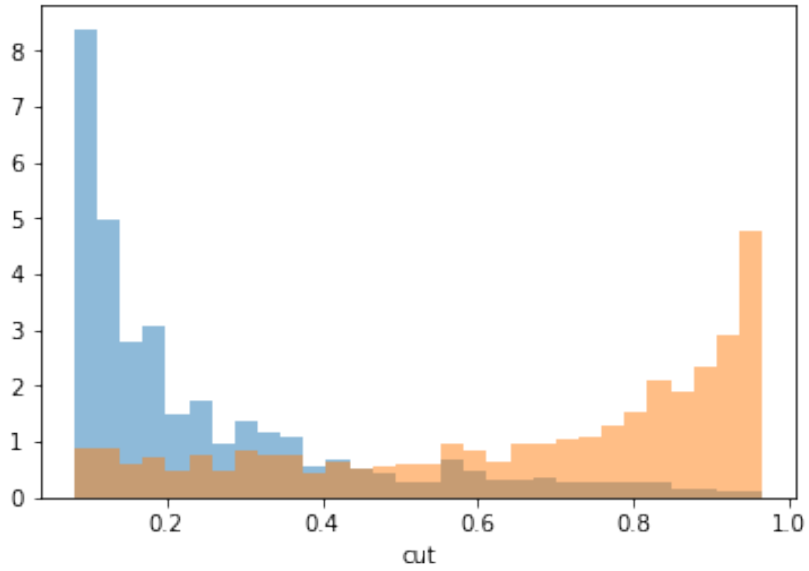
Figure 14b shows the ROC curve [23] of the RNN classifier. Generally, the more area under the ROC curve the better the classifier. In the perfect case, where everything gets labelled 100% correctly, the area under the curve(ROC AUC) would be 1 and random guessing would be around 0.5. Here, we have an area of 0.93. This is already really close to the optimal case.

8.3 XGBoost

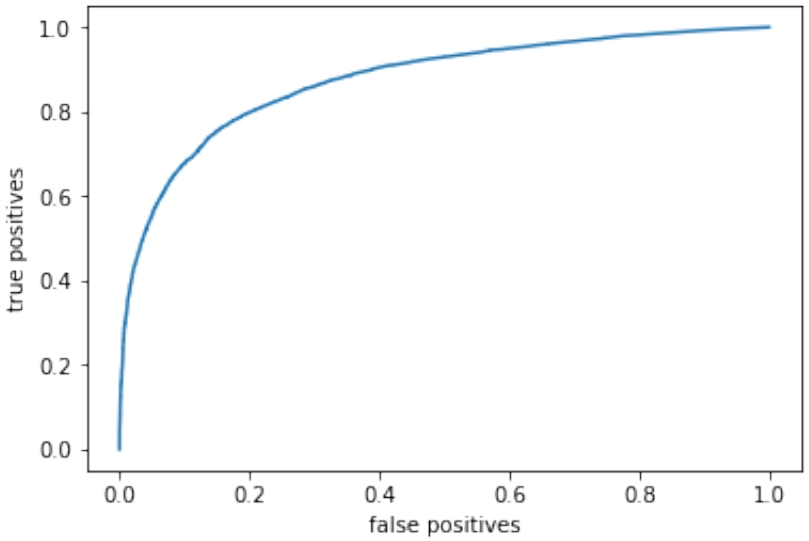
Also an XGBoost classifier[28] was implemented and trained to have some more comparison to the performance of our RNN classification. XGBoost models train much faster than NN and are often a serious competitor to them as often they reach similar performances. Based on that, they are often used as baselines for RNN classifiers and a RNN classifier is considered good if they surpass the XGBoost model. The input of XGBoost model was the same as for the RNN classification. The accuracy of this classifier of labelling the tracks was at 80.74% with a cut applied at 0.5. Note that here we also exceeded the 75% even though with a smaller accuracy than the RNN.

[27] E.g. all positives have to be correct $\rightarrow$ increase cut

[28] Depth = 3 and number of estimators =3



(a) Number of false positives and false negatives depending cut



(b) ROC curve for the XGBoost model

Figure 15: XGBoost classifier figures

In figure 15a the blue bins are false positives and the orange bins are false negatives. Here we see that the bins are more evenly spread and gather less at the edges. So, already qualitatively we can guess that it will perform worse than our RNN's.

Figure 15b shows the ROC curve of the XGB classifier. Generally, the more area under the ROC curve the better the classifier. In the perfect case, where everything gets labelled 100% correctly, the area under the curve would be 1. Here we have an area of 0.88.

8.4 Comparison in performance of the RNN and XGBoost

The RNN classifier performs with around 6% better accuracy than the XGBoost classifier. Also, by comparing the the ROC curves in figure 16, one can clearly see that the area under the RNN ROC curve is bigger. In numbers we have around 0.05 > more area under the curve for the RNN model. The RNN classifier performs significantly better in labelling the 8 hit tracks than the XGBoost model.

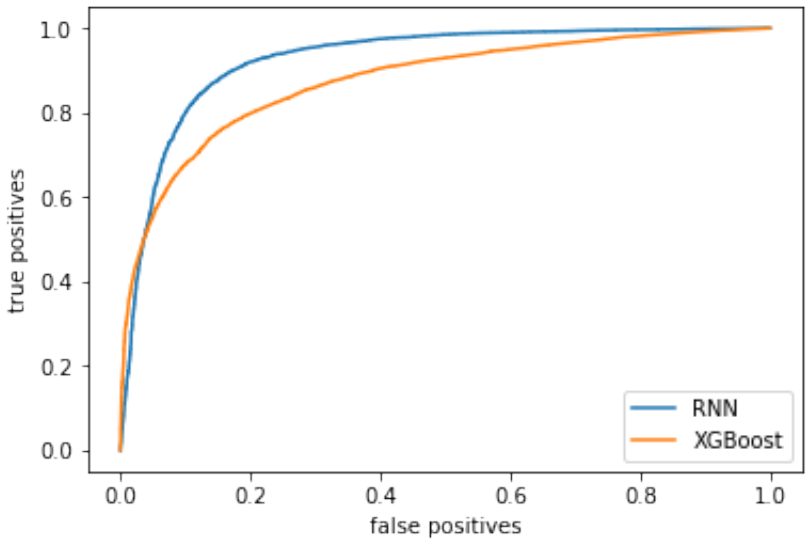


Figure 16: Comparison ROC curves of RNN and XGBoost model

9 Results

9.1 Results

The RNN models perform significantly better at labelling the 8 hit tracks than all other classifiers and methods.

Model	Accuracy with cut at 0.5 [%]	ROC AUC
Best χ^2	52.01%	/
XGBoost	80.74	0.88
RNN	87.63%	0.93

Using this system of RNN's proves to be viable solution to this problem and brings a huge jump in accuracy also over other machine learning solutions.

9.2 Outlook and potential

Where do we want to go from here? One way to improve the algorithm would for example be to create a fully connected neural network [24]. By doing this both RNN's would be connected and would train as a unit. This would have the positive effect of not having to retrain the classifying RNN as well whenever the first on gets changed.
Another goal could be to make this type of RNN appliable to more types of problems. So for example, instead of being restricted to tracks of a specific length (here eight hits) one could make it more general to be able to deal with an arbitrary length of the track. This would be especially useful for this experiment, as a lot of particles don't just recurl once but many times over (in the central station). Hereby, they are creating a lot of background, which minimalizing is crucial to reach our desired sensitivity of 10^{-16}.
The ultimate goal however, would be to replace the current track reconstruction algorithm altogether and put a RNN in its place. This could for example be done by an RNN performing beam search[29] [25] to find the true track of a particle. In other areas, beam search has proven to be a powerful tool and there is a lot of potential for this sort of algorithm in physics as well, especially in track reconstruction

[29]Both inside out and outside in

10 Acknowledgements

I would like to thank the Physics Department of the University of Zurich. Special thanks goes to Prof. Nicola Serra of the University of Zurich who let me do this thesis in his research and introduced me into the world of neural networks.
I would also like to express my gratitude towards Dr. Patrick Owen for providing me with data and always being here to help when I had questions. Also special thanks goes to Jonas Eschle, who was always there to help me with programming the RNN's and discuss techniques and tricks.

References

[1] MZ Akrawy, G Alexander, J Allison, PP Allport, KJ Anderson, JC Armitage, GTJ Arnison, P Ashton, G Azuelos, JTM Baines, et al. Measurement of the z0 mass and width with the opal detector at lep. *Physics Letters B*, 231(4):530–538, 1989.

[2] Mark Thomson. *Modern particle physics*. Cambridge University Press, 2013.

[3] S Abe, T Ebihara, S Enomoto, K Furuno, Y Gando, K Ichimura, H Ikeda, K Inoue, Y Kibe, Y Kishimoto, et al. Precision measurement of neutrino oscillation parameters with kamland. *Physical Review Letters*, 100(22):221803, 2008.

[4] P Adamson, C Andreopoulos, R Armstrong, DJ Auty, DS Ayres, C Backhouse, G Barr, M Bishai, A Blake, GJ Bock, et al. Measurement of the neutrino mass splitting and flavor mixing by minos. *Physical Review Letters*, 106(18):181801, 2011.

[5] A Blondel, A Bravar, M Pohl, S Bachmann, N Berger, M Kiehn, A Schöning, D Wiedner, B Windelband, P Eckert, et al. Research proposal for an experiment to search for the decay $\mu \to eee$. *arXiv preprint arXiv:1301.6113*, 2013.

[6] Heiko Augustin, Niklaus Berger, Sebastian Dittmeier, Carsten Grzesik, Jan Hammerich, Qinhua Huang, Lennart Huth, Moritz Kiehn, Alexandr Kozlinskiy, Frank Meier Aeschbacher, et al. The mupix system-on-chip for the mu3e experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 845:194–198, 2017.

[7] Jonathan Philipp, Lennart Huth, Heiko Augustin, Raphael Philipp, Dirk Wiedner, Niklaus Berger, Mu3e Collaboration, et al. Das hv-maps basierte mupix teleskop. Technical report, Detector RD at DESY Test beam, 2015.

[8] Heiko Augustin, N Berger, S Bravar, Simon Corrodi, A Damyanova, F Förster, R Gredig, A Herkert, Q Huang, L Huth, et al. The mupix high voltage monolithic active pixel sensor for the mu3e experiment. *Journal of Instrumentation*, 10(03):C03044, 2015.

[9] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[10] Stephen Grossberg. Recurrent neural networks. *Scholarpedia*, 8(2):1888, 2013.

[11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[12] François Chollet et al. Keras: Deep learning library for theano and tensorflow. *URL: https://keras. io/k*, 7(8), 2015.

[13] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[14] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980, 2017.

[15] Trishul M Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *OSDI*, volume 14, pages 571–582, 2014.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[17] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

[18] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[19] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[20] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[21] S Agostinelli. S. agostinelli et al.(geant4 collaboration), nucl. instrum. methods phys. res., sect. a 506, 250 (2003). *Nucl. Instrum. Methods Phys. Res., Sect. A*, 506:250, 2003.

[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[23] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July 1997.

[24] CR Gent and CP Sheppard. Special feature. predicting time series by a fully connected neural network trained by back propagation. *Computing & Control Engineering Journal*, 3(3):109–112, 1992.

[25] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.