Friedrich-Alexander-Universität
Naturwissenschaftliche Fakultät

Bachelor Thesis
in Physics

Development of a Recurrent Neural Network for optimizing the muon simulation efficiency in IceCube

Benedikt Josef Mayer

Supervisor: Prof. Dr. Claudio Kopper

Erlangen Centre for Astroparticle Physics

Submission date: 4 March 2025

Abstract

The IceCube Neutrino Observatory detects high-energy astrophysical neutrinos, which are presumed to originate from active galactic nuclei (AGNs), gamma-ray bursts (GRBs), supernova remnants, and neutron star mergers. However, muons are also generated in atmospheric particle cascades, requiring IceCube to distinguish neutrino-induced muons from atmospheric background events.[1]

Efficiently simulating these muons is computationally demanding, necessitating optimization strategies. This thesis investigates the use of a Recurrent Neural Network (RNN) to improve the efficiency of muon simulation by predicting whether a muon event survives the veto selection process.

To facilitate training, a dedicated tool called ToyCube was developed to generate training data by simulating muons propagating through ice and IceCube, calculating the cumulative photon yield in IceCube's veto region. This tool provides full control over the simulation and demonstrates that an RNN can successfully learn to predict photon yield.

A dedicated dataset was generated using MuonGun simulations to train and evaluate a different RNN architecture. Systematic hyperparameter tuning—including adjustments to hidden layers, neuron counts, learning rates, thresholds, positive class weighting, and incorporating dropout—was performed to optimize model performance. The final model achieved a validation error rate of approximately 5.4 % or 6.1 %, depending on the chosen threshold, and demonstrated the potential to reduce computational costs by selectively filtering muon events before full simulation.

Contents

1 Introduction

The IceCube Neutrino Observatory is the world's largest neutrino detector, designed to observe high-energy neutrinos from astrophysical sources. Unlike charged cosmic rays, neutrinos travel undisturbed over cosmic distances, making them valuable messengers of extreme astrophysical events.[1]

The most likely sources of high-energy neutrinos include active galactic nuclei (AGNs), gamma-ray bursts (GRBs), supernova remnants, and neutron star mergers. These neutrinos are typically produced through the decay of pions, which decay into muons and neutrinos.[1]

A major challenge in neutrino detection is distinguishing astrophysical neutrino-induced muons from atmospheric muons. Simulating these muons is essential for background rejection, but is computationally expensive.

This thesis explores the use of deep learning, specifically Recurrent Neural Networks (RNNs), to enhance the efficiency of muon simulation in IceCube. RNNs are well-suited for processing sequential data, making them ideal for analyzing muon propagation tracks and predicting either the photon yield in the veto region or whether a given muon event survives the veto selection process. By training an RNN on simulated muon data, this study aims to reduce computational costs by filtering out background events before full simulation.

To achieve this, a dedicated simulation tool called ToyCube was developed to generate training data by modeling muon propagation and calculating photon yields within IceCube's veto region. A separate dataset, generated using MuonGun simulations, was then used to train and evaluate a more complex RNN architecture. Systematic hyperparameter tuning, including adjustments to the number of layers, neuron counts, learning rates, thresholds, positive class weighting, and incorporating dropout was performed to optimize the model's performance.

2 Theory

2.1 IceCube Neutrino Observatory

The IceCube Observatory is a particle detector located at the South Pole. It is a cubic-kilometer large and almost 2,500 meters deep in Antarctic ice. IceCube consists of several sub-detectors. At the surface is IceTop and under the ground is the inner sub-detector and DeepCore embedded in it. Figure 1 illustrates the structural design of the neutrino observatory. IceCube is primarily designed to study neutrinos from extreme cosmic phenomena.[2]

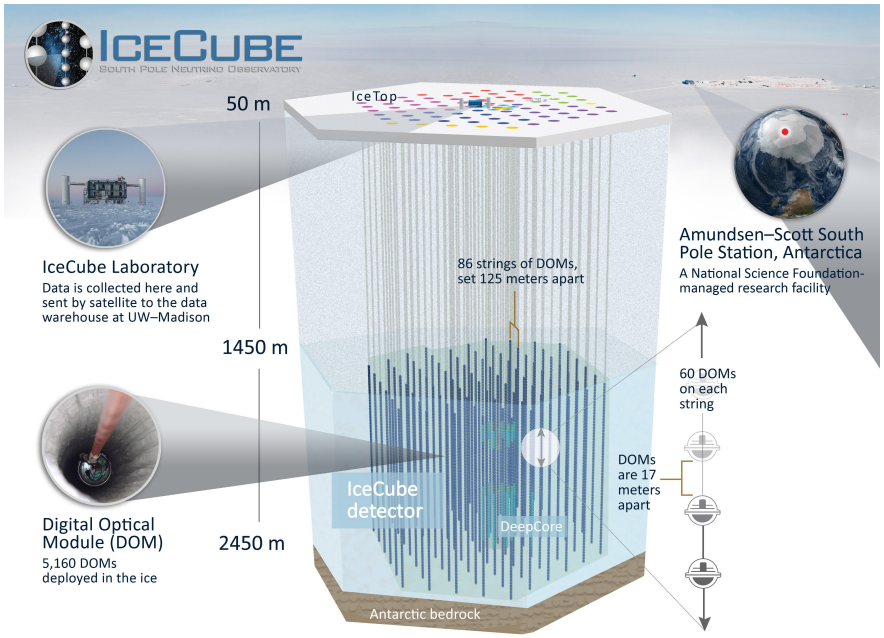


Figure 1: This schematic illustrates the structural design of the IceCube Neutrino Observatory. IceTop, located on the surface, is complemented by 5,160 Digital Optical Modules (DOMs) distributed along 86 strings, which are deployed in the ice at depths between 1,450 and 2,450 meters. The DeepCore sub-array is situated at the center of the detector. The image is sourced from [2].

There are 5,160 so-called Digital Optical Modules (DOMs) in the detection system which are attached to vertical cables, the so-called "strings". The strings are arranged on a hexagonal grid with 125 meters spacing and were lowered through 86 boreholes. Eight additional strings with a higher density of DOMs form the more compact DeepCore in the center. The function of DeepCore is to enhance IceCube's sensitivity to low-energy neutrinos and neutrino oscillation measurements.[2]

Above the main detector, IceTop is located at the surface and consists of stations with two ice-filled tanks, each equipped with two downward-facing DOMs. IceTop serves primarily as a veto and calibration detector, helping to identify and filter atmospheric muons that could interfere with the detection of neutrino-induced signals. Additionally, IceTop is used for studying extensive air showers (EAS).[2]

The innovative design and multi-component structure of IceCube allow it to detect neutrinos across a wide energy range.

2.2 Neutrino detection

Neutrinos are fundamental particles that belong to the lepton family. They are electrically neutral and therefore are not deflected by magnetic fields. Neutrinos are interacting only via the weak nuclear force and gravity, which allows them to pass through matter almost undisturbed, but making them challenging to detect. For a long time, their tiny rest mass was thought to be zero. Moreover, neutrinos exist in three flavors—electron, muon and tau neutrinos (ν_e, ν_μ, ν_τ)—and exhibit the phenomenon of flavor oscillation as they travel through space. Their ability to carry unmodified information over large distances makes them invaluable messengers in the exploration of the universe.[1]

Neutrinos are produced in a variety of astrophysical environments through different fundamental processes. In main sequence stars like the Sun, they are generated primarily via the proton-proton chain reaction, where hydrogen nuclei fuse to form deuterium, emitting electron neutrinos [1]:

$$p + p \rightarrow {}^2\mathrm{H} + e^+ + \nu_e.$$

In core-collapse supernovae, neutrinos are abundantly produced through electron capture, where electrons are absorbed by protons to form neutrons and electron neutrinos [1]:

$$e^- + p \rightarrow n + \nu_e.$$

High-energy neutrinos, in particular, are expected to originate from supernova remnants, but the most likely dominant sources are active galactic nuclei (AGNs). In these environments, protons are accelerated to extreme energies by relativistic jets and interact with surrounding photons, leading to pion production [1]:

$$p + \gamma \rightarrow \Delta^+ \rightarrow n + \pi^+.$$

The charged pions subsequently decay into muons and muon neutrinos:

$$\pi^+ \rightarrow \mu^+ + \nu_\mu.$$

Muons further decay, producing electron neutrinos and additional muon neutrinos:

$$\mu^+ \rightarrow e^+ + \nu_e + \bar{\nu}_\mu.$$

If the target photons originate from the cosmic microwave background, the resulting neutrinos are known as GZK neutrinos, which are potential candidates for the highest-energy neutrinos observed. Another possible source of high-energy neutrinos could be neutron star mergers, where extreme magnetic fields and relativistic outflows create conditions for hadronic interactions, leading to neutrino production through similar pion decay processes.[1]

High-energy neutrinos produced in astrophysical sources are difficult to detect and therefore only observed indirectly. When a neutrino interacts with the ice, it produces

electrically charged secondary particles. One key interaction mechanism is the Charged-Current (CC) interaction, in which a neutrino exchanges a W boson with a nucleon, transforming into its corresponding charged lepton.[1]

For example, a muon neutrino can interact with a neutron as follows:

$$\nu_\mu + n \rightarrow \mu^- + p.$$

If one of these charged particles travels through the medium faster than the speed of light in that medium, it emits Cherenkov radiation. This light is then detected by the photomultiplier tubes (PMTs) inside Digital Optical Module (DOMs) in the IceCube Neutrino Observatory. The observed light pattern allows reconstructing the energy and direction of the incoming neutrino. Muon neutrinos are particularly favorable for detection, as they produce long track-like signatures that can be used to determine their trajectory with high precision.

2.3 Muon detection

Muons are elementary particles and, just like neutrinos, they belong to the lepton family. Similar to electrons, they are charged particles, but with a rest mass of approximately $106\,\mathrm{MeV}/c^2$ they are roughly 207 times heavier than electrons. They are produced both in the Earth's atmosphere through cosmic ray interactions with nuclei in upper layers and within the detector via neutrino-induced interactions. Due to their relatively long lifetime of $2.2\,\mu\mathrm{s}$ and high penetration capability, muons can pass through significant amounts of matter, leaving distinct track-like signals in detectors. As the muon travel through matter, it causes ionization by knocking out electrons off atoms nearby, creating ionized particles along its path. At very high energies, the muon can also undergo Bremsstrahlung, in which the muon emits photons due to acceleration caused by its interaction with atomic nuclei. When traveling faster than the speed of light in a medium such as ice, muons emit Cherenkov radiation, which can then be captured by the DOMs of IceCube.[1]

The IceCube Neutrino Observatory detects muons because they serve as a crucial signal for the indirect detection of neutrinos. Muons are produced when high-energy neutrinos interact with atomic nuclei in the ice of the detector. Particularly relevant for IceCube are those muons that originate from interactions of astrophysical neutrinos, as they provide insights into cosmic events in which these neutrinos were generated.

However, muons are produced not only by neutrino interactions in the detector, but also as part of secondary particle cascades resulting from the interaction of cosmic rays with the Earth's atmosphere [1]. These atmospheric muons are also detected by IceCube, though they are considered background noise. Nevertheless, valuable information can be extracted from their analysis.

2.4 High-Energy Starting Event (HESE) veto

The High-Energy Starting Event (HESE) veto is a technique used in the IceCube Neutrino Observatory to separate astrophysical neutrinos from background events

caused by atmospheric muons and neutrinos. It works by using the outer parts of the detector as a veto layer, ensuring that only neutrino events that start within the fiducial volume of the detector are considered. The veto layer is defined as the DOMs in the outermost 90 m of the detector at the top, 90 m from the sides, the bottommost 10 m, and a 60 m thick horizontal layer below the region of the most dust-concentrated ice (cf. Figure 2).[3]

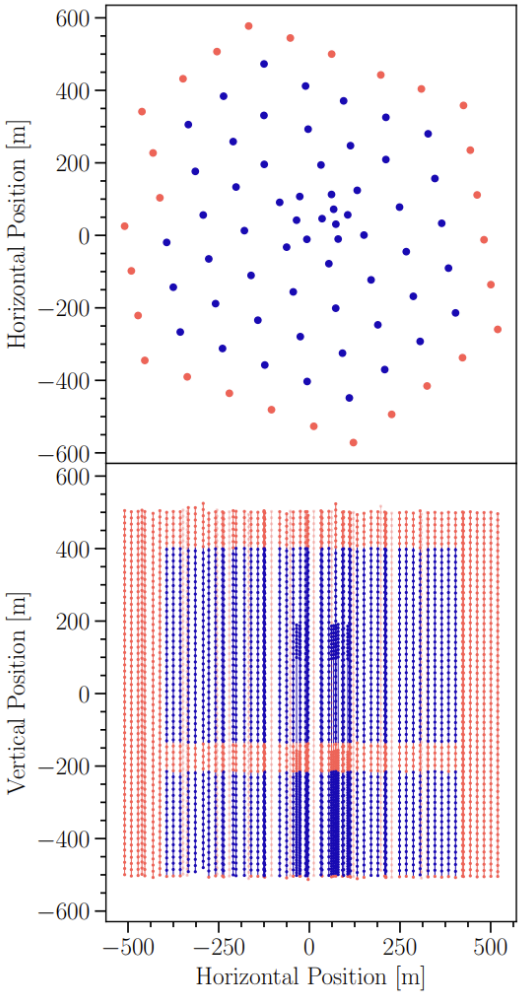


Figure 2: Schematic of the IceCube detector DOM positions. Above: The red dots represent strings with only veto DOMs, whereas the blue dots indicate the strings which also have non-veto DOMs. Below: This is a side view of the IceCube detector, where every red dot is a veto DOM and every blue dot is a non-veto DOM. This diagram was obtained from [3].

Since atmospheric muons enter the detector from outside, a set of veto criteria has been established to filter them out.

The first interaction of the event must start inside the fiducial volume to be considered. Also, the event must have three or fewer photo-electrons (PEs) of charge detected in the veto region, otherwise the event is rejected. In addition, if the event has hits on

more than two veto DOMs, it is also rejected. Any light detected in the veto region must have arrived within 50 ns after the event started to make sure that the light is coming from inside the detector and not from an outside source. The veto system is also able to ignore hits that happen too long before or after the event, by ignoring hits that occur more than 3 µs before or after the event start time. This helps remove accidental background noise that isn't related to the neutrino interaction. Finally, the detected veto hit must close enough to the event—within $3\,\mu s \cdot c \approx 899\,m$—to ensure that the hit is related to the neutrino event and not caused by something else entering the detector from the outside.[3]

Events failing these criteria are classified as external background and are removed from the HESE sample.

2.5 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a type of artificial neural network specifically designed to process sequential data by maintaining connections across time steps. Unlike traditional deep learning models, RNNs retain information from previous inputs through internal states, allowing them to capture contextual relationships and temporal dependencies. This makes them particularly effective for tasks makes them well-suited for tasks where context and order matter, like speech-related tasks, but also time series-dependent problems. Instead of treating each input independently, RNNs use past information to enhance predictions. However, they struggle with learning long-range dependencies due to issues like vanishing and exploding gradients, which can hinder effective training.[4]

The most basic model consists of three layers, the input layer, a hidden layer and an output layer, as shown in Figure 3. Recurrent connections allow information to cycle through the network, which maintains a hidden state, which in turn captures information.[4]

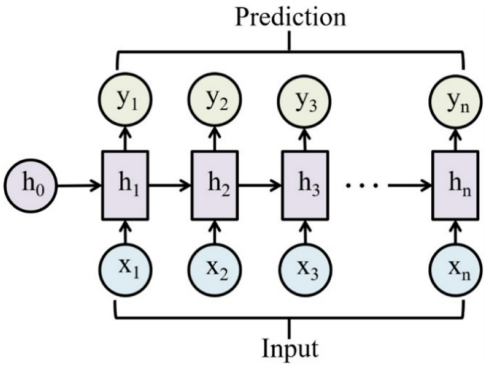


Figure 3: Basic RNN architecture, featuring an input layer, a hidden layer and an output layer. The internal state of the RNN is updated at each time step, enabling it to remember information from previous time steps. This schematic was taken from [4].

The following terms are mentioned in the subsequent research and necessary to understand the internal process and functions of an RNN.

Network architecture

Input Layer: The first layer, which receives the input data. It has the same size as the input vector.

Hidden layer: Layers between the input and the output layer, which maintain recurrent connections to store past information and processes sequential dependencies. They are called "hidden" because their outputs are not visible.

Output Layer: The layer which generates predictions based on the final hidden state.

Hidden size: The hidden size determines the number of neurons in the hidden layer of a neural network. This parameter affects the model's capacity to learn and retain complex patterns.

LSTM layers: An LSTM layer is a special type of RNN layer designed to handle sequential data and long-term dependencies. It avoids the vanishing gradient problem, which is a problem where the gradients become tiny, making it difficult for the model to learn and update weights.

Linear Layer (Fully Connected (FC) Layer): A layer where every neuron is connected to all neurons in the previous and next layers. They provide simple and fast mapping between inputs and outputs and are often used as the final layer before the output layer.

Activation function: Activation functions are used to enable the network to learn complex patterns, by introducing non-linearity to the outputs of neurons.

ReLU (Rectified Linear Unit): This popular activation function introduces non-linearity while being computationally efficient. If the input is positive, the output remains the same, but if the input is negative or zero, the output is set to zero. The function is defined as: $f(x) = \max(0, x)$.

Training parameters

Learning rate: Determines how much the weights of a neural network are updated during each iteration of training. A high learning rate speeds up training but may cause instability, while a low learning rate ensures more stable convergence but may take longer.

Batch size: Number of training samples processed together in a single training step. Larger batch sizes speed up training, but may generalize worse.

Early Stopping: A technique to prevent overfitting by stopping the training when the model's performance on a validation set stops improving.

Positive weight: A parameter used to adjust the loss function to handle class imbalance by assigning more weight to the minority class, ensuring the model doesn't become biased toward the majority class.

Dropout: A regularization technique where a fraction of neurons is randomly set to zero during training to prevent overfitting and improve generalization.

General Terms

Overfitting: A modeling error where the model learns not only the underlying patterns but also incorporates the noise in the training data, leading to poor performance on new, unseen data.

Loss: The loss is the error calculated as the difference between the model's predictions and the actual target values. The lower the loss, the better.

Training loss: The loss calculated on the training dataset, indicating how well the model is learning the training data.

Validation loss: The loss calculated on the validation dataset, used to monitor the model's performance and detect overfitting during training.

Testing loss: The loss is the error calculated on a separate test dataset that the model has never seen before, used to evaluate the final performance.

3 ToyCube Simulation

ToyCube is a simulation tool designed to generate training data for neural networks using the proposal package. It simulates the propagation of muons through ice and calculates the total photon yield in IceCube's veto region. The initial conditions of the muons, including energy, starting position, and direction, are randomly generated within a predefined framework. The number of samples (muon tracks) can be freely chosen. After generation, the training data is filtered and structured to minimize memory usage and computational overhead.

This chapter provides a detailed explanation of the development of this tool. The source code is available at [5].

3.1 Proposal package

The `proposal` python package is a dedicated simulation tool designed to model charged lepton propagation (especially muon propagation) within a predefined medium. It allows simulating the muon's trajectory and computes discrete energy losses due to interactions with the medium. Each interaction is characterized by its spatial coordinates, interaction type (e.g. ionization, bremsstrahlung, photonuclear interactions, electron pair production) and the associated energy deposit.[6]

3.2 Energy reconstruction and photon yield conversion

The muon propagation function of the `proposal` package only outputs the positions, types and energies of the individual energy loss events. To convert these losses into observable photon yields, we employ the following photon yield function for a point source:

$$\mu(r) = n_0 A \frac{1}{4\pi} e^{-\frac{r}{\lambda_p}} \cdot \frac{1}{\lambda_c r \tanh(\frac{r}{\lambda_c})}. \tag{1}$$

In this expression, the following medium-dependent parameters are defined:

$$\lambda_a = 98\,\mathrm{m},\ \lambda_e = 24\,\mathrm{m},\ \zeta = e^{-\frac{\lambda_e}{\lambda_a}},\ \lambda_p = \frac{\sqrt{\lambda_a \cdot \frac{\lambda_e}{3}}}{1.07} \text{ and } \lambda_c = \frac{\lambda_e}{3\zeta}. \tag{2}$$

The propagation length (λ_p) is defined by the absorption length (λ_a) and the effective scattering length (λ_e). The factor 1.07 is a correction factor used by [7] for a better fit to the experimental data. Equation 1 and its parameters (Equation 2) were taken from [7].

The prefactor $n_0 \cdot A$ in Equation 1 is determined via curve fitting to the experimental data in Table 1.

Figure 4 shows the curve of Equation 1 fitted to the values in the Table 1. In the analysis, the combined prefactor is found to be

$$n_0 \cdot A = (6.4 \pm 0.5) \cdot 10^8\,\text{number of photons} \cdot \mathrm{m}^2.$$

The curve does not fit perfectly to the data, but without the mentioned prefactor of 1.07 in Equation 2 it would be even worse.

distance [m]	30	50	78	94
photons	5500	780	229	68

Table 1: Experimental data used to fit the photon yield curve. This data was obtained from [8].

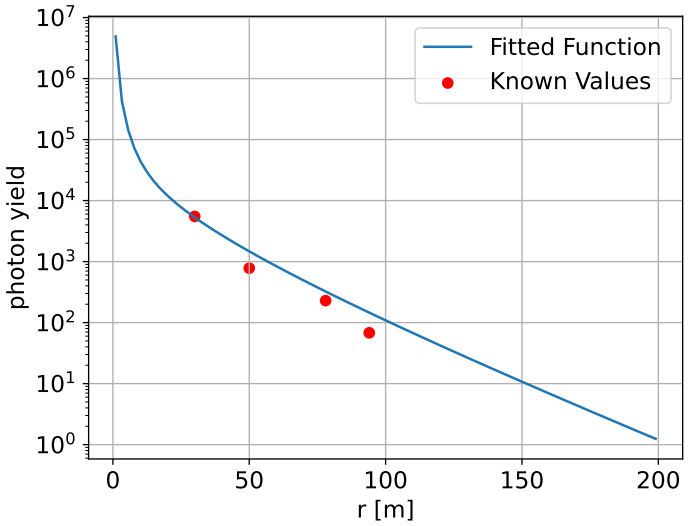


Figure 4: This plot illustrates the photon yield as a function of distance. The red dots represent the experimental data from Table 1 while the solid curve is the fitted model according to Equation 1 and the parameters in Equation 2.

The data in Table 1 are standardized to an energy of 300 TeV. For instance, a 300 TeV loss event leads to 5,500 hits after 30 m. Therefore, to calculate the number of photons, the energy loss itself does not directly contribute to the total number of hits. Rather, it must be divided by 300 TeV.

Thus, the final equation to compute the total number of photons produced by a given energy loss is:

$$N_{\text{photons}}(E, r) = \frac{E \cdot \mu(r)}{300\,\text{TeV}} \tag{3}$$

This formulation enables the calculation of the number of photons that would be detected by a Digital Optical Module as a function of both the distance r between the energy loss and the DOM, and the energy E deposited during the interaction.

3.3 Visualization of photon yield

To visually demonstrate the photon yield calculation, a simplified scenario is considered wherein the simulation volume is modeled as a sphere of ice with a radius of 10^{20} m. A positive muon is propagated through this medium under predetermined initial conditions. For a representative test case:

- an initial energy of 10^7 MeV,
- a starting position located 10^5 m above the IceCube center,
- and an initial propagation vector given by $(1, 0, -3)$

is chosen.

Figure 5 shows the complete track of energy losses along the muon's path, as it travels through the ice. The red diamond marks the position of a virtual observer, acting like a DOM, which observes the emitted photons of the individual energy losses as shown in Figure 6. Each energy loss event is color-coded according to its interaction type. The legend of these figures reveal that blue dots stand for ionization, orange dots for electron pair production, green dots for photonuclear reactions and red dots stand for bremsstrahlung.

In Figure 6, one can see the virtual observer again, as well as the proportion of photons it observes from which energy loss location. Size and color represent, as mentioned, the amount of photons, which reach the DOM and the type of energy loss the photons develop from. The closer the interaction, the more photons survive the distance traveled. It becomes evident that, at a certain distance, the photon yield drops to a point where no photons can be detected by the observer. This can be equated with the number of photons falling below the threshold of the detector, resulting in a track, that appears to be much shorter. The number of photons observed is calculated by Equation 3, where the distance for each photon yield calculation is the distance between the DOM and each point where an energy loss event happens.

Figure 5 and Figure 6 shows very well that the tool and the photon yield function (Equation 3) work successfully. Ultimately, the objective is to determine the amount of photons detected by each veto DOM in the IceCube observatory. This information allows calculating the total photon yield for each traversing muon. Later, this information

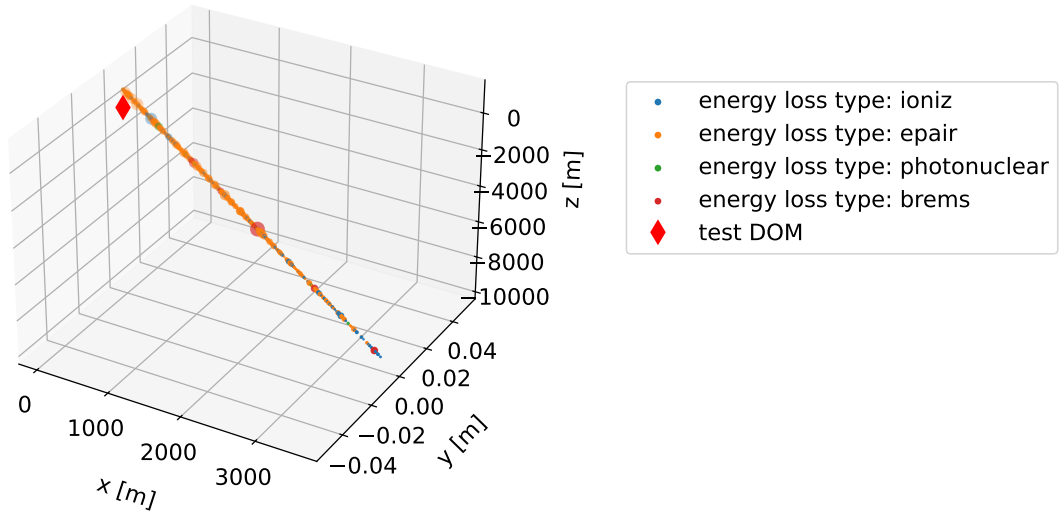


Figure 5: A 3D representation of the muon's energy loss events, where the dot size reflects the magnitude of the energy loss and the color indicates the interaction type. The red diamond at coordinates $(0, 0, 0)$ represents a virtual DOM observing the muon track. Please note that the circles have been scaled for better visibility.

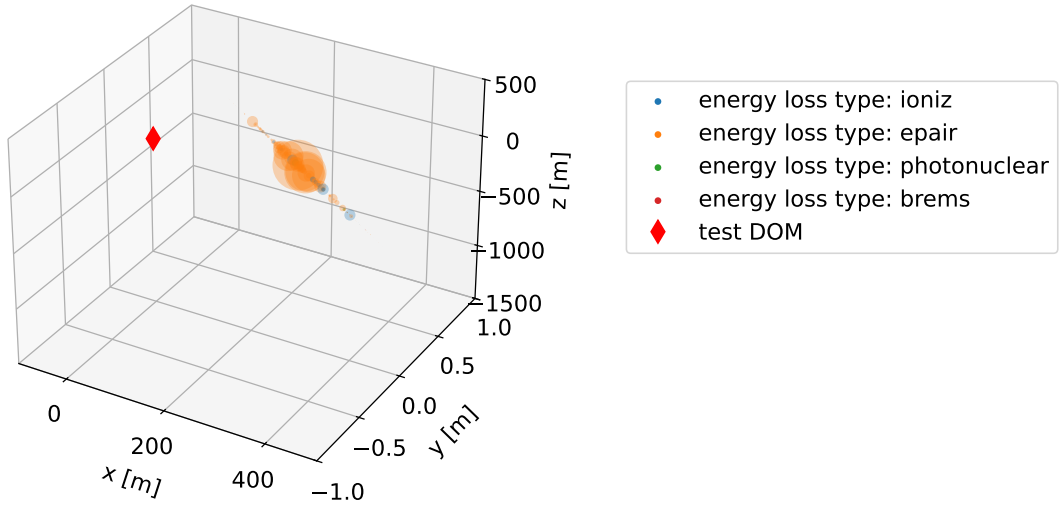


Figure 6: The photon yield, observed by a virtual DOM, represented by the red diamond at coordinates (0,0,0). The circle sizes correspond to the number of photons produced at each energy loss location, which were actually detected by the DOM. The interaction types are color-coded the same as in Figure 5. Here, too, the circles have been scaled for improved visibility.

is used to develop a machine learning framework designed to predict the total photon yield for each muon event.

Further, the geometry of IceCube's underground detector is needed. Specifically, for future purposes, only the veto region is needed. The veto region defined for the HESE veto contains a 60 m thick horizontal layer below the region of the most dust-concentrated ice. However, since this simulation uses a homogeneous ice sphere, a simplified version of the veto region is used, excluding the said horizontal layer. Only the veto region is relevant, because later this is the main target area for the neural network predictions.

In Figure 7 the red dots represent the veto region, whereas the blue dots mark the remaining DOMs. One can see, that the topmost four DOM layers, as well as the outer rim of strings and the bottommost DOMs belong to the veto region. The DeepCore is visible by the more densely packed dots in the center.

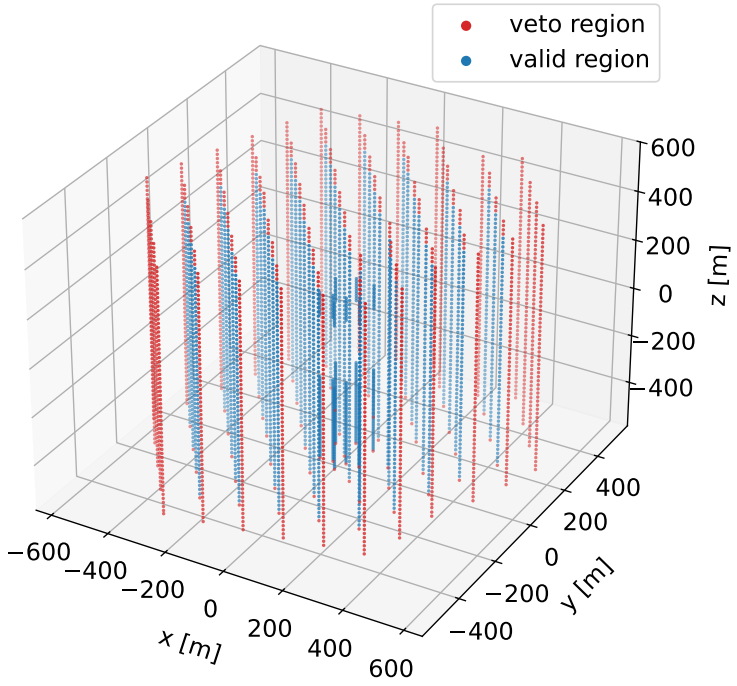


Figure 7: Geometry of the IceCube observatory (excluding IceTop). The red dots indicate the DOMs that are part of the veto region. The blue dots mark the remaining DOMs. The more densely packed DeepCore is visible in the center.

The Recurrent Neural Network, trained on the simulated muon tracks, should be able to predict the number of detected photons in the veto region. Later, a second RNN, based on the previous one, is trained on MuonGun simulation data, which then predicts whether the muon survives the veto decision or not.

The procedure illustrated in Figure 6 for a single DOM is now extended to all veto DOMs. The initial simulation parameters (sphere of ice, energy of the muon, starting position and direction) stay the same. Figure 8 visualizes the resulting cumulative photon yield across the veto region, which is placed inside the ice sphere. The positions

of the purple circles are equal to the positions of the veto DOMs. However, their sizes are determined by the total sum of the photons they observe. This sum is calculated by simply taking the sum of all photons each DOM observes (cf. Figure 6). Veto DOMs which do not detect any photon at all are not visible. The additional colored markers in the figure show a segment of the muon track, as in Figure 5. This time, the trace is cut to the spatial boundaries of the IceCube detector, which is why the rest of the track is not visible.

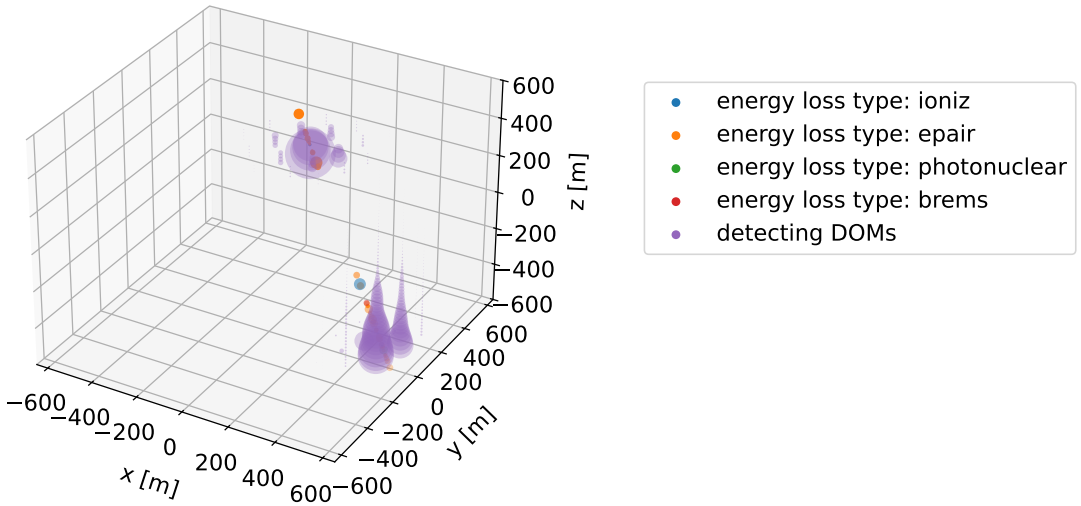


Figure 8: Spacial distribution of the photon yield across the veto DOMs. Each DOMs size is defined by the total number of photons it observes at that location. One can also see the trace of the propagating muon through the detector and its energy losses, as well as the color-coded type of loss. It should be mentioned, that the circle sizes are scaled to improve visibility.

A method is now established to calculate the number of photons that each veto DOM would detect as a muon passes through the observatory. The next step is to generate training data that can be used for the development and training of a Recurrent Neural Network.

3.4 Generation and preparation of training data

The training data should comprise two components:

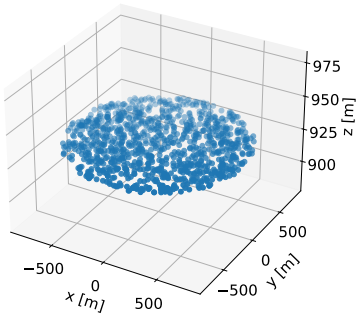
1. **Input data:** The positions of individual energy losses along the muon track and the corresponding energy deposits.
2. **Target data:** The total number of photons detected by all veto DOMs for that muon event.

To produce an appropriate amount of training data, many simulated muon tracks are generated. Each track is characterized by unique initial conditions, including random

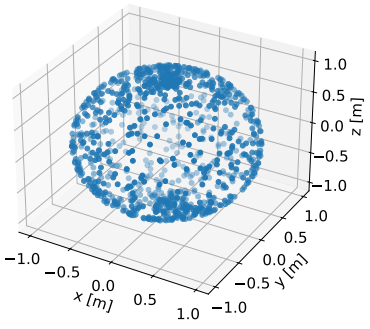
starting positions, propagation directions, and random starting energies. The first step is to create random starting positions.

Figure 9a illustrates 1000 evenly distributed random starting positions on a disk of 800 m radius, while Figure 9b displays the corresponding random propagation directions.

To avoid biasing the simulation with muons propagating predominantly in one direction (e.g., vertically downward), a rotation to the initial position is applied using the corresponding random propagation directions. These are generated evenly spread around a unit sphere as shown in Figure 9b.



(a) This figure shows a disk of 1000 evenly distributed randomly generated starting positions for muon tracks. The disk has a radius of 800 m and is located 930 m above the IceCube center.



(b) A plot of 1000 uniformly distributed dots on the unit sphere. Each of them is used as a random propagation direction for the muon track.

Figure 9: These plots illustrate the distribution of the random starting position (9a) and the random generated propagation directions (9b).

There is a function provided by [8] to calculate the three-dimensional rotation of a vector $\vec{x} = (x_1, x_2, x_3)$ to a new vector $\vec{x}' = (x_1', x_2', x_3')$. Equation 4 calculates the rotation matrix of e_z on the direction vector $\vec{d} = (d_1, d_2, d_3)$ and applies it to the position vector $\vec{x}$.

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} -d_1^2 \cdot \frac{1-d_3}{d_2^2+d_1^2} + 1 & -d_1 d_2 \cdot \frac{1-d_3}{d_2^2+d_1^2} & d_1 \\ -d_1 d_2 \cdot \frac{1-d_3}{d_2^2+d_1^2} & -d_2^2 \cdot \frac{1-d_3}{d_2^2+d_1^2} + 1 & d_2 \\ -d_1 & -d_2 & d_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{4}$$

The resulting positions after applying Equation 4 to all random initial positions is shown in Figure 10.

Figure 11 presents the non-rotated and rotated starting positions, including the IceCube veto region for scale.

Finally, for the initial muon energy, a range between 10^5 and 10^9 MeV is chosen.

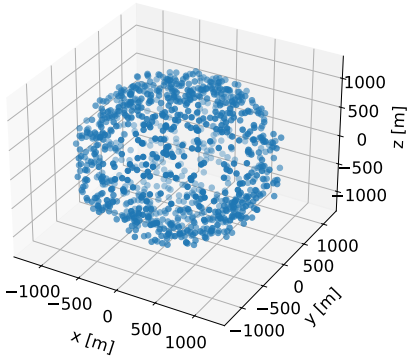
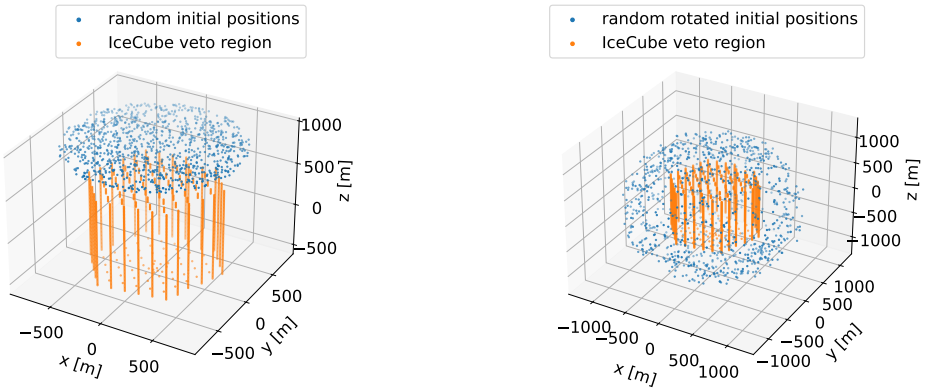


Figure 10: Random initial positions after rotating, demonstrating the effect of applying Equation 4 to the random initial positions shown in Figure 9a based on the random propagation direction shown in Figure 9b.



(a) Initial random positions (non-rotated) (same as Figure 9a) with the IceCube veto region.

(b) Rotated initial positions (same as Figure 10) including the IceCube veto region.

Figure 11: Comparison of non-rotated (11a) and rotated random initial positions (11b) with the IceCube geometry for spatial reference.

3.5 Data reduction and optimization

Some methods are needed to reduce the computing time of the simulation. Not only that, but also the training duration of the RNN needs improvement. To achieve this, unnecessary parts of the training data get removed, without compromising essential information.

One way to save on negligible training data is to choose a threshold for minimal detected photons per loss. This is to cut off all data which contributes virtually nothing to the total amount of photons in the veto region. This threshold is set to 0.01 photons. For example, when simulating 1,000 muons there are 1,059,606 energy loss events total from which only 76,908 are considered relevant due to the minimal emission criterion. The total saving is in this case about 92.7 %.

Another method is to reduce the memory consumption, by changing the structure of the stored data, which helps to load all the training data faster into the RAM for quicker access. It is also faster to restore the original structure of the data after loading it instead of loading the not optimized training data.

The training data is restructured to speed up RAM loading and processing. Instead of storing each energy loss event in the form of

$$E_1, \vec{x}_1, E_2, \vec{x}_2, \ldots, E_n, \vec{x}_n,$$

the data is reorganized to

$$\vec{x}_0, \vec{r}, E_1, d_1, E_2, d_2, \ldots, E_n, d_n.$$

For clarification, $\vec{x}_n$ is the position of an energy loss, whereas $\vec{x}_0$ is the starting position of the muon,E_n is the energy released, $\vec{r}$ is a normalized vector which points in the direction of the muons trajectory and d_n represents the distance along the track at which the n^{th} energy loss occurs. This distance is calculated by taking the norm of the difference between the starting position and the loss position, as in $d_n = ||x_0 - \vec{x}_n||$.

With the simplification of just storing the starting position and direction of the trajectory, everything else needed is the distance traveled to each energy loss.

This restructuring nearly halves the number of stored entries. For example, a muon track with 3000 interactions would require 12000 entries in the original format, but only 6006 entries in the optimized format (including the 6 entries for the starting position and direction).

The neural network does however require the first type of data structure for training. The original spatial coordinates for each energy loss event can be recovered using:

$$\vec{x}_n = \vec{x}_0 + \vec{r} \cdot d_n,$$

thereby enabling full reconstruction of the track if necessary.

The introduction of a minimal emission threshold and an optimized data format not only reduces storage requirements but also accelerates data loading, preprocessing, and the training duration.

4 Recurrent Neural Network Training

This section focuses on the development and training of a Recurrent Neural Network (RNN) designed to accurately predict whether a muon survives the veto decision or is rejected. The motivation behind this approach is to reduce the need for exhaustive simulations of all possible muon propagation scenarios, which would be computationally prohibitive. By effectively filtering out muons classified as atmospheric, the RNN enhances the efficiency of muon simulations in IceCube research.

4.1 Training with ToyCube data

The overarching goal of generating synthetic data using the ToyCube mini tool and training a neural network on this dataset is to establish a proof of concept. This approach ensures full control over the simulation while assessing the network's ability to learn from the structured data.

To determine whether the model can learn at all, the initial model parameters were determined through trial and error. The most important parameters are the number of hidden layers (in this case, LSTM layers) the number of hidden states (or neurons) in each layer, and the learning rate.

The structure of the training data created by the ToyCube tool reveals that the input layer size is four, corresponding to the energy and spatial position of each loss event. The training data is loaded sequentially, loss by loss, for each muon track sequence. The output layer size is one, as the target is the total amount of photons in the veto region. A batch size of 1,024 was chosen for training on the GPU cluster.

With picking a learning rate of 0.001, the next step is to investigate the impact of the number of layers and their size on the model's performance.

4.1.1 Number of layers and layer size testing

To identify the optimal parameter set for predicting the number of photons in the veto region of IceCube and determining whether the veto is triggered, nine different combinations of layer configurations (1, 2, or 3 layers) and sizes (64, 128, or 256 neurons) are tested. The trained RNNs are then evaluated to determine the best-performing parameter set.

In Figure 12, the RNN's predictions are compared with the actual values from the training dataset. In any case, more than 70 % of the data points are located within the 1σ confidence band. The sigma value is determined by calculating the standard deviation of the differences between the actual values and predicted values. If the absolute value of this difference is smaller than σ than the data point is located inside the confidence band. It turns out, that for two LSTM layers with 128 neurons, the most (78 %) predictions are located within $\pm 1\sigma$.

To find out how high the false prediction rates are, we assume, that to trigger a veto event, five photons are needed. When the RNN predicts more than five photons, but there are actually less than that, then this is called a false positive.

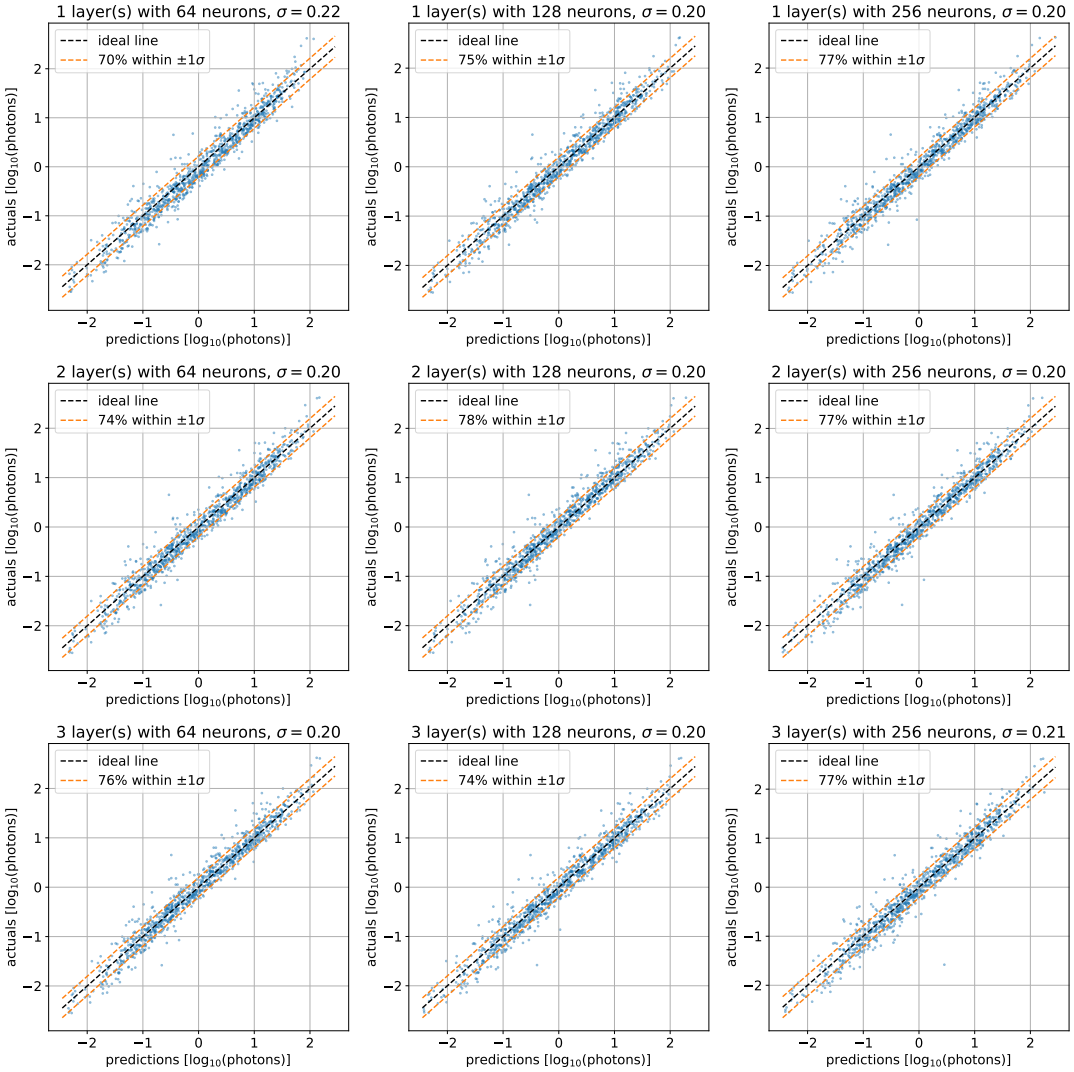


Figure 12: In these plots, the actual numbers of photons are plotted over the RNN's predicted number of photons. The axes are scaled with $\log_{10}$. The blue dots are the data points, the black line is the ideal line where the predictions match the true values, and the orange lines mark the confidence band of $\pm 1\sigma$. The number of LSTM layers used in the model is equal for each line, and the number of neurons is equal in each column.

In general, the interesting part is in how well the RNN can predict whether an event will survive the veto. To do that, the false positive rate needs to be as low as possible. The best fitting model is the one that is the least common to mistakenly predict more than five photons if there are actually less than that.

In Table 2 it is evident that the model variant with two hidden layers and 128 neurons has the lowest false positive rate (1.42 %). This model is the best fitting one.

number of layer	1			2			3		
size of layer	64	128	256	64	128	256	64	128	256
false positive rate [%]	3.41	2.27	2.98	3.12	1.42	2.13	2.13	3.27	2.84

Table 2: This tabel contains the false positive rates corresponding to all RNN models which are discussed in Figure 12.

The learning curves for the tested model variants are presented in Figure 13. These plots illustrate the progression of training and validation loss over increasing epochs. Some models terminate training early using the *Early Stopping* function to prevent overfitting.

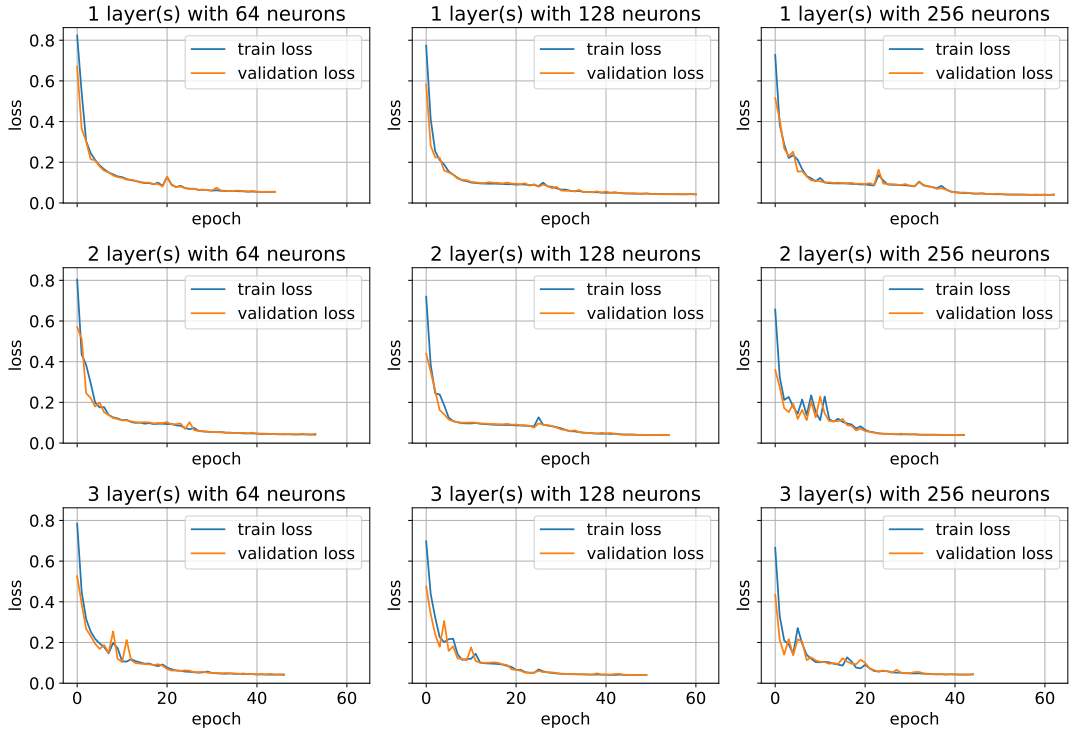


Figure 13: These are the learning curves corresponding to the models discussed in Figure 12. Each row contains the learning curves of the models with the same amount of LSTM layers. The size of the LSTM layer is the same for each column.

4.1.2 Testing learning rates

Previously, a learning rate of 0.001 was selected for training the model based on trial and error. This time, the goal is to evaluate whether an alternative learning rate yields better performance.

As shown in Figure 14, the learning curves for different learning rates indicate that 0.001 remains the optimal choice. A learning rate of 0.01 appears to be too high, likely resulting in unstable training or highly fluctuating loss. Vice versa, learning rates of 0.0001 and 0.00001 lead to excessively slow loss reduction, suggesting that they are too low. This results in inefficient training, as the model learns at an impractically slow pace.

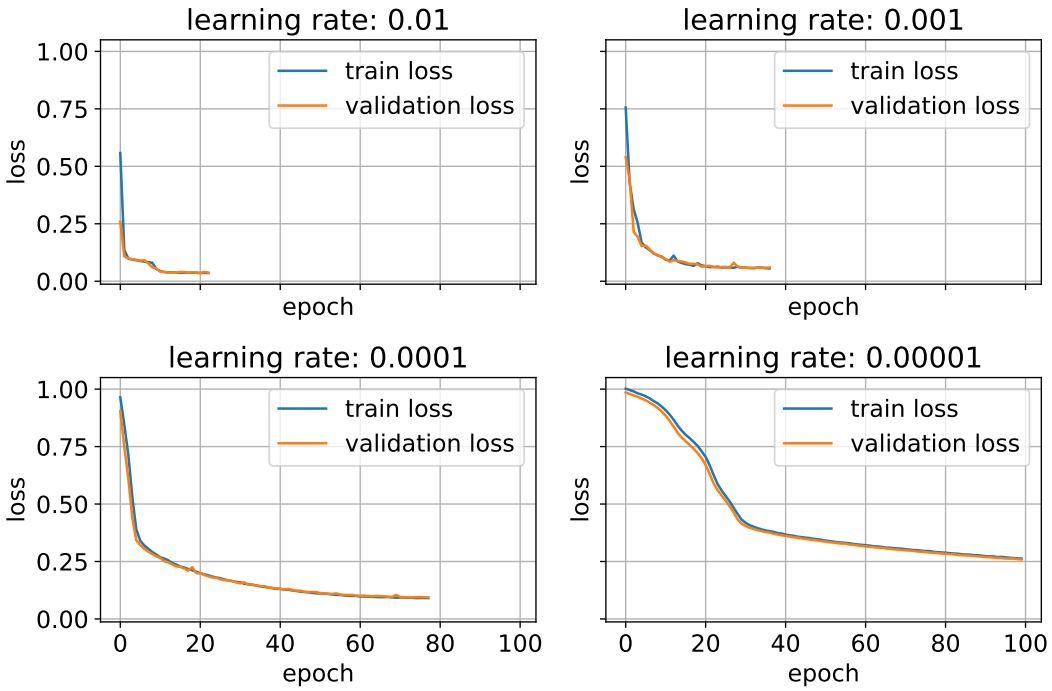


Figure 14: These plots show the learning curves for different learning rates: 0.01, 0.001, 0.0001, and 0.00001. The blue line represents the training loss, while the orange line represents the validation loss. Lower learning rates result in smoother curves but also lead to longer training times.

4.2 Training with MuonGun data

After successfully training a neural network on data simulated by ToyCube and demonstrating its effectiveness, the next step is to apply the model to MuonGun data. MuonGun is a simulation tool, integrated into IceCube's simulation framework, allowing researchers to compare simulated muon distributions with real data. A similar model architecture is used, but the target data for training is now binary: 1 if the event survives the HESE event selection (i.e., the muon is simulated), and 0 if it is rejected.

Previously, the target data represented the number of detected muons in the veto region. As before, the input training data consists of muon energy losses and their positions. All training data were provided by [8].

As with ToyCube data, different model parameters are tested to identify the most suitable configuration for training with MuonGun data. However, the optimal model may differ, as the data have changed.

Training with MuonGun data is divided into two main objectives:

1. The first goal is to fine-tune the model by adjusting various hyperparameters, such as the learning rate, hidden layer size, and the number of LSTM or fully connected (FC) layers. The optimization is done by monitoring the validation loss and minimizing the error rate as much as possible.

2. Once the model is optimized, the second objective is to improve the muon simulation efficiency by reducing incorrect predictions—particularly the false negative rate. Lowering the false negative rate ensures that that muons, which should be simulated, are not mistakenly rejected. This enhances the accuracy of the filtering process.

The *false negative rate* represents the proportion of muons that should have been simulated but were incorrectly rejected. A false negative occurs when a traversing muon, which would normally survive the veto (because it is not an atmospheric neutrino and was produced by a high-energy neutrino inside the detector), is incorrectly discarded. This results in lost information, which is undesirable.

The *false positive rate* refers to the proportion of muons that should have been rejected but were mistakenly passed by the veto. False positives occur when muons that should have been filtered out by the veto are instead included in the simulation. This wastes computational resources, which is also undesirable.

There is an inherent trade-off between false negatives and false positives: reducing one increases the other. Why this is the case will be explained in more detail in a later section. The goal is to strike a balance that maximizes simulation efficiency while minimizing both types of errors.

4.2.1 Parameter testing

This section examines the impact of varying the number of hidden layers, the number of neurons per layer, and the learning rate on the model's training performance.

As shown in Figure 15, increasing hidden layer size leads to a faster reduction in validation loss compared to increasing the number of layers—though the latter shows a rapid initial decline within the first two layers. Additionally, a higher learning rate generally improves the validation loss. A more detailed analysis of learning rate variations is presented in a later section.

The corresponding learning curves are provided in the appendix as Figure 28, Figure 29 and Figure 30. Based on these results, a model with two hidden layers, 512 neurons

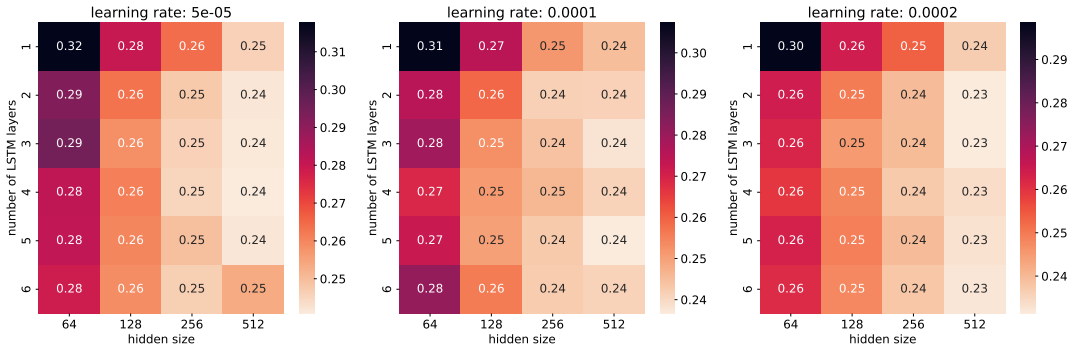


Figure 15: Parameter comparison of hidden size from 64 to 512, number of LSTM layers from 1 to 6 and learning rate from 0.00005 to 0.0002. The heatmaps are arranged in increasing order from left to right and visualize the validation loss. A brighter color indicates lower validation loss.

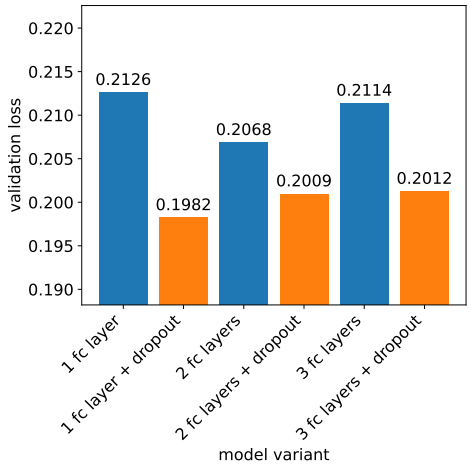
per layer, and a learning rate of 0.0002 is identified as the optimal configuration, as it achieves the lowest validation loss. Besides that, in Figure 30 the train loss declines noticeably further than in other configurations.

For consistency, validation loss is always evaluated at the highest possible common epoch across models. Since *Early Stopping* causes some models to terminate training earlier than others, the last shared epoch is used for comparison. This ensures a fair evaluation, favoring models that achieve faster loss reduction. This methodology will be applied in all upcoming loss comparisons.

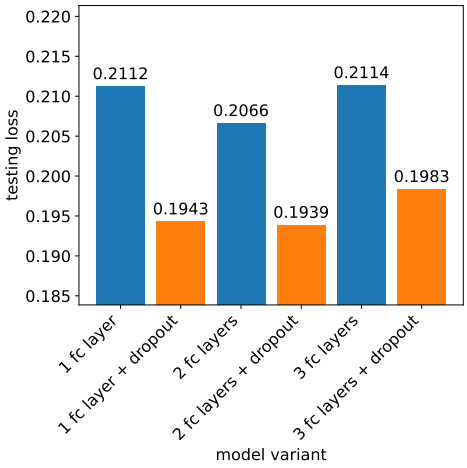
4.2.2 Testing fully connected (FC) layers

Figure 16a shows how well the dropout function improves the model's validation loss. The additional fully-connected layers and activation function (ReLU) before the output of the model help not so much. The linear layers used in this model have the same size as the LSTM layers that are placed before them. For the best result, one fully connected layer plus dropout is chosen. In Figure 16b the testing loss is slightly lower for two fully connected layers plus dropouts than for just one FC layer plus dropout, but that is not consistent with the results shown in Figure 16a and not much improvement for the cost of computational load.

Figure 17 and Figure 18 show some validation plots to the influence of the dropout function. With dropout enabled, the gap between positive and negative predicted probabilities widens, and the peaks become sharper. This is particularly evident in the peak of positive predictions. There is no noticeable difference in the right plots other than the smaller area under the curve (AUC), which drops from 0.04 to 0.03. A smaller are indicates that the curve comes closer to the corner, which means the skill of the RNN increases. In general, the right plot illustrates the performance of a binary classifier model at varying threshold values. The subsubsection 4.2.5 is dedicated to the threshold comparison. The remaining plots are provided in the appendix from Figure 31 to Figure 34.



(a) This plot shows the influence of the dropout function and the FC layers on the validation loss. The validation loss shown is the loss from the last common epoch for training with 1, 2 or 3 linear FC layers alternating with and without dropout.



(b) As in Figure 16a, the influence of the dropout function and the FC layers are shown. Instead of the validation losses, this plot compares the testing losses.

Figure 16: The influence of the number of fully connected layers and the dropout function to the validation loss (16a) and the testing loss (16b) is shown. In both cases, the dropout causes a significant drop in the losses.

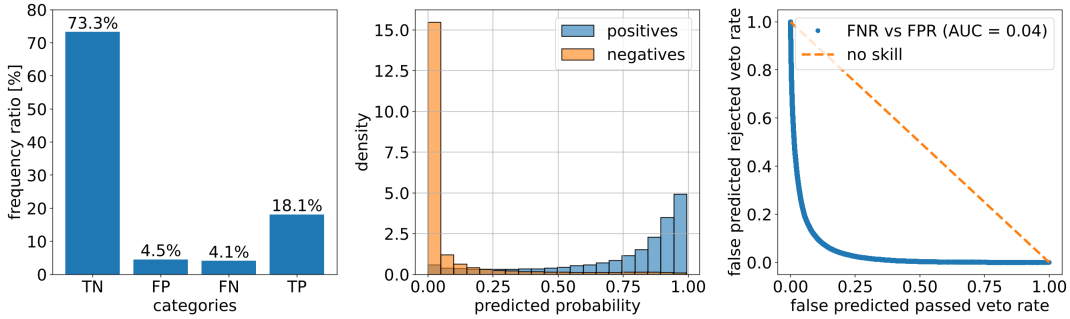


Figure 17: Left: Percentage of true negative (TN), false positive (FP), false negative (FN), and true positive (TP) predictions. Center: Distribution of positive and negative predictions. Right: False negative rate vs false positive rate. Model with one additional linear layer and no dropout function. Error rate: 8.6 %.

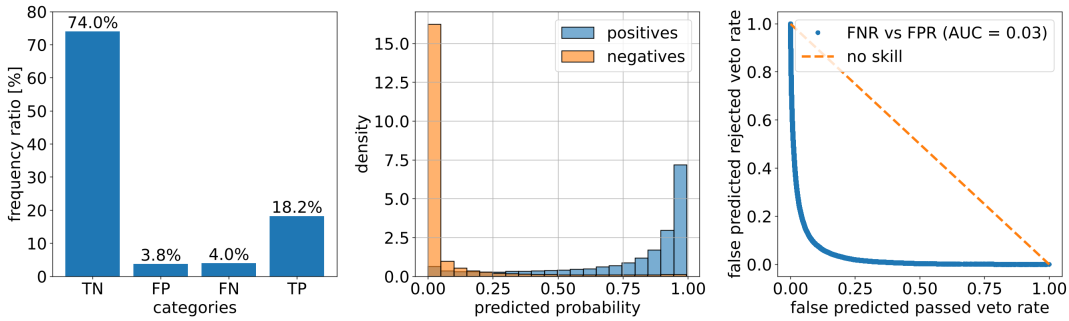


Figure 18: Left: Percentage of true negative (TN), false positive (FP), false negative (FN), and true positive (TP) predictions. Center: Distribution of positive and negative predictions. Right: False negative rate vs false positive rate. Model with one additional linear layer and no dropout function. Error rate: 7.73 %.

The *dropout* technique prevents overfitting by randomly dropping (setting to zero) a fraction of the neurons during training. This helps to improve the generalization by reducing the reliance on specific neurons and forces the network to learn more robust features.

With additional linear layers (FC layers) different sizes of them need to be tested alongside the size of the LSTM layers again. As seen in Figure 19, a LSTM layer size of 512 and one FC layer deliver the best results. It seems that the FC layer size does not have much of an influence to the validation loss and that the results from a FC layer size of 128 and 1,024 look the best, but 128 appears to be the slightly better choice overall. In the learning curves Figure 35 and Figure 38 this is not easy to see, but when comparing the error rates, 6.7 % is ever so slightly smaller than 6.8 %. Again, 1,024 neurons bring with them a lot of computational effort, which is not worth it for so few percentage points. The different error rates are provided in Table 3.

FC layer size	128							256							512							1024	
hidden size	128			256			512	128			256			512	128			256			512	256	512
num FC layer	1	2	3	1	2	3	1	1	2	3	1	2	3	1	1	2	3	1	2	3	1	1	1
error rate [%]	7.5	7.5	7.2	7.1	7	7.3	**6.7**	7.5	7.4	7.4	7	7.1	7	6.9	7.4	7.5	7.3	7.1	7.1	6.9	6.9	7.1	**6.8**

Table 3: This table contains the error rates for every model variant listed in Figure 19.

One can find the corresponding learning curves to Figure 19 in the Appendix (cf. Figure 35, Figure 36, Figure 37 and Figure 38).

The empty areas in Figure 19 are due to the fact that these models have not been trained. The same plot but for testing losses is located in the appendix (cf. Figure 39).

Since the size and number of FC layers have been optimized, one can take a quick look back at Figure 15. It was mentioned there that with increasing learning rate, the validation loss decreases. This means even higher learning rates should be tested.

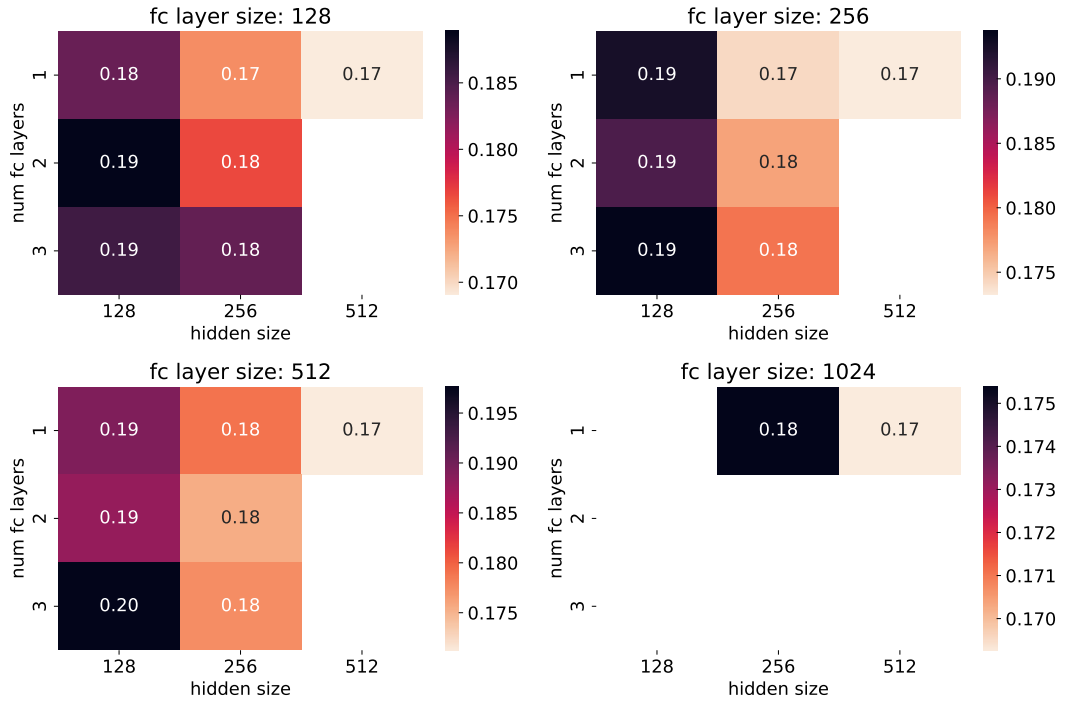
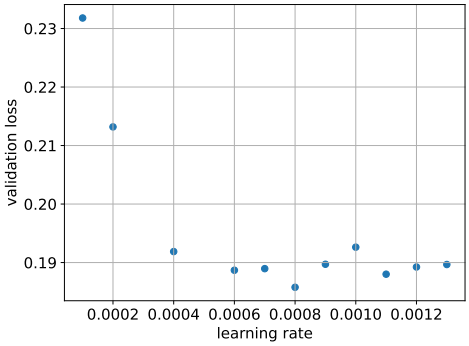


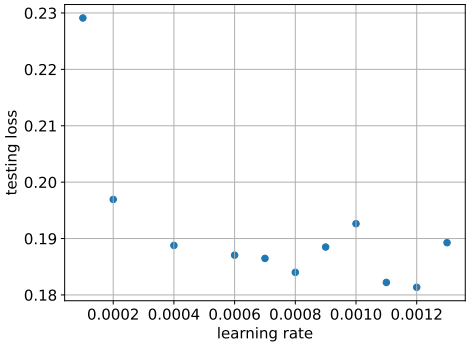
Figure 19: These plots illustrate the influence of the number of FC layers from one to three, the size of the LSTM layer from 128 to 512, and the size of the FC layer from 128 to 1,024 to the validation loss. The brighter the color, the lower the validation loss. There are empty spaces due to the fact that these models have not been trained.

4.2.3 Learning rate optimization

With increasing learning rate the validation loss and testing loss improved further as it is shown in Figure 20. This improvement stops approximately at around a learning rate of $6 \cdot 10^{-4}$. There is no obvious minimum. Figure 40 in the appendix shows all corresponding learning curves for the plots in Figure 20b. Comparing the trend of the learning curves, the optimal learning rate seems to be somewhere between $6 \cdot 10^{-4}$ and $1 \cdot 10^{-3}$ due to beginning fluctuations and lack of improvement. A learning rate of $8 \cdot 10^{-4}$ is considered the optimal choice.



(a) This plot compares the validation loss of RNN models with different learning rates. Higher learning rates lead to improvement up to a certain point. Learning rates from 0.0001 up to 0.0013 are compared.

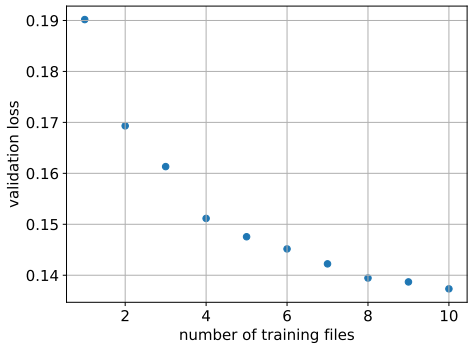


(b) As in Figure 20a, the losses of different learning rates are compared. Here, instead of comparing the validation losses, the testing losses are compared.

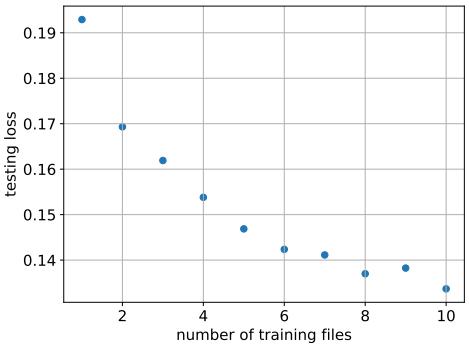
Figure 20: Both validation loss and testing loss behave reciprocal to the learning rate. They flatten out with increasing learning rate, but start to fluctuate towards the end. The testing loss varies more at the end than the validation loss. Again, each dot is the loss at the highest common epoch of the different models.

4.2.4 Influence of amount of training data

Using more training data is a great method to improve the model's performance even further. In Figure 21a, one can observe how the validation loss decreases with increasing amount of training data, from one to ten training data files. Each file contains around 400 k muon track sequences.



(a) One can clearly see, that the validation loss behaves reciprocal to the amount of training data.



(b) The testing loss behaves the same as the validation loss in Figure 21a.

Figure 21: Both plots show a downward going trend for the losses with increasing amount of training data. The x-axis describes the number of training files, with about 400 k muon tracks each. Each dot is the validation loss at the highest common epoch of the different models.

The curve clearly flattens out in Figure 21a with increasing amount of training data. In any case, the minimum has not yet been reached. However, due to time limitation, not more than ten files have been used for comparison. With ten files, the training took easily four hours on two "NVIDIA A100 Tensor Core-GPUs".

Not only the validation loss decreases with more training data, but also the error rate (cf. Table 4).

number of files	1	2	3	4	5	6	7	8	9	10
error rate [%]	7.72	6.71	6.51	6.06	5.86	5.73	5.58	5.41	5.53	5.32

Table 4: Table of error rates resulting from amount of training data.

4.2.5 Threshold comparison

At this stage of optimization, the model's performance is no longer measured by validation loss but by the accuracy of its predictions. The goal is to find out how many predictions are correct or incorrect. Therefore, the following categories can be determined:

- true negatives (TN): correctly predicted rejected muon
- false negatives (FN): false predicted rejected muon
- false positives (FP): false predicted passed muon
- true positives (TP): correctly predicted passed muon

The threshold determines whether the RNNs output, raging between zero and one, is classified as a rejected or passed muon. Depending on where the threshold is placed, the false negatives and false positives change reciprocal. For lower threshold values there are more false positives and less false negatives, and for higher threshold the opposite is the case. The total number of positives and negatives does not change.

In Figure 22 are some of the validation results showing the influence of the different thresholds. In the left column, the false positive rates and false negative rates change clearly depending on the threshold. In the center there is the distribution of total positive and negative predictions which does not changes. The plot in the right column illustrates the performance at varying threshold values. It compares the false negatives rate against the false positives rate, and shows where the threshold is (red dot) compared to the point closest to the lower left corner (optimal performance). The black diagonal helps to determine which threshold is closest to this corner, which in this case is between 0.25 and 0.3. The rest of the validation plots can be seen in the appendix from Figure 41 to Figure 43.

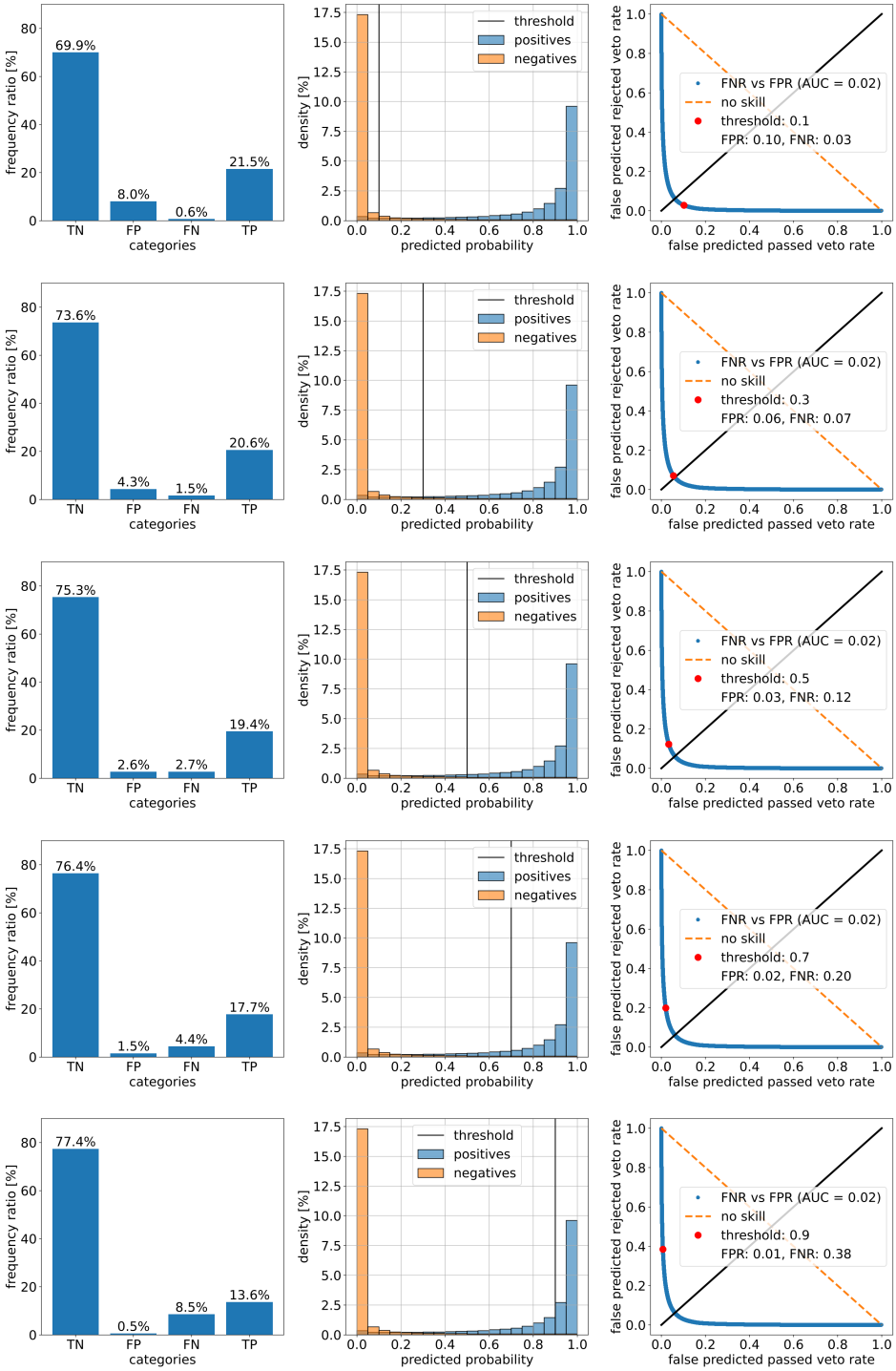


Figure 22: Left: Percentage of true and false predictions. Center: Histogram of positives and negatives with position. The black vertical line marks the position of the threshold. Right: False negatives rate vs false positives rate. The red dot marks the current threshold. The black diagonal helps to show how close the red dot (threshold) is to the optimum position. The closer the curve gets to the bottom left corner, the more skilled is the model.

In Figure 23, one can clearly see, that the closer the threshold comes to the edges, the lower the accuracy gets. A low threshold causes less incorrect predicted rejected muons and more incorrect predicted surviving muons. The same is the case for the total number of positive and negative predictions. A higher threshold causes the opposite. For optimal savings, the incorrectly rejected muon rate needs to be as low as possible by also keeping the incorrectly passed muon rate low.

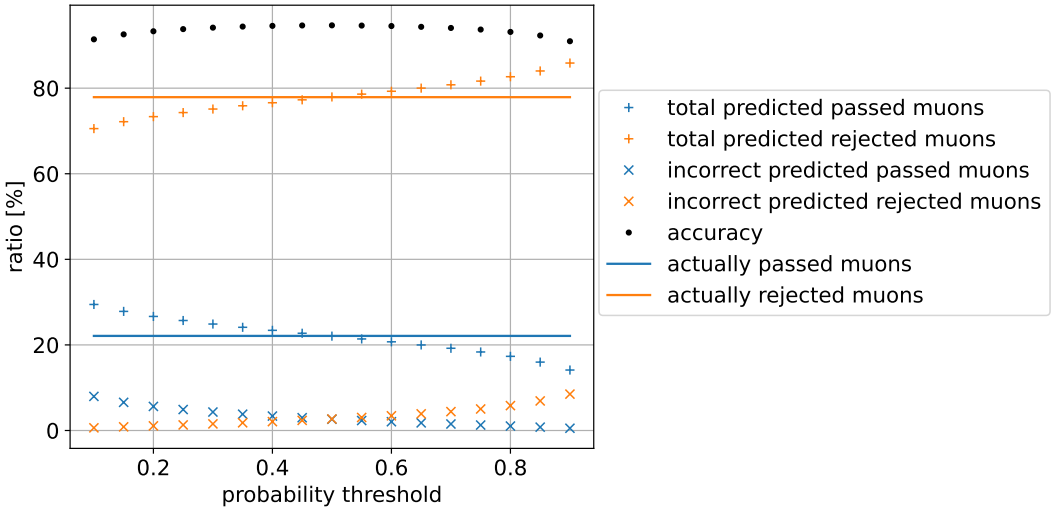


Figure 23: An overview over all results depending on the threshold. The plot shows different rates dependent on the threshold. The black dots illustrate the model's accuracy dependent on the threshold, whereas the solid lines are the actual rates of passed and rejected muons from the training data. The Xs display all incorrect predictions, whereas the plusses show the total number of positive and negative predictions.

Let's say we can accept a false passed rate of 5 % (we unnecessarily simulate 5 % muons that would normally be rejected). According to Table 5, this would correspond to a threshold of 0.25 where the false rejected rate is 1.3 % and the total saving is 74.29 %. Saving means (FN + TN)/total predictions. So the sum of all negative predictions (predicted rejected muons), which are therefore not simulated muons. At 1 % false rejected, the savings are still at 73 %.

error rate (%)	threshold	correctly passed (%)	false passed (%)	correctly rejected (%)	false rejected (%)	savings (%)
8.59	0.10	21.48	7.97	69.93	0.62	70.55
7.43	0.15	21.26	6.58	71.32	0.85	72.17
6.70	0.20	21.03	5.63	72.27	1.08	73.35
6.20	0.25	20.81	4.90	72.99	1.30	74.29
5.86	0.30	20.56	4.32	73.58	1.54	75.12
5.61	0.35	20.31	3.81	74.08	1.80	75.88
5.44	0.40	20.03	3.37	74.52	2.07	76.59
5.35	0.45	19.74	2.98	74.91	2.36	77.27
5.32	0.50	19.42	2.63	75.26	2.69	77.95
5.37	0.55	19.06	2.33	75.56	3.04	78.61
5.48	0.60	18.67	2.05	75.84	3.43	79.28
5.67	0.65	18.21	1.78	76.12	3.89	80.01
5.93	0.70	17.70	1.52	76.38	4.41	80.78
6.30	0.75	17.08	1.27	76.62	5.03	81.65
6.86	0.80	16.29	1.04	76.86	5.82	82.67
7.69	0.85	15.20	0.78	77.11	6.91	84.02
9.03	0.90	13.60	0.53	77.37	8.50	85.87

Table 5: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold.

4.2.6 Weight optimization

This section investigates whether applying positive weights can further improve savings.

Figure 24 compares different positive weight values ranging from 1 to 10, as well as the case where no specific positive weight is applied (`None`) and a weight of 3.5. The `None` setting implies that all targets receive the same weight. A weight of 3.5 can be considered "balanced" since it equalizes the weighting between positive and negative outcomes. This value is derived by dividing the number of negative outcomes by the number of positive outcomes in the training dataset, yielding approximately 3.5.

The fourth column of Figure 24 further illustrates that, for a weight of 3.5, the red dot is positioned closest to the lower left corner, which represents the optimal scenario.

It is obvious that the weighting of positive targets has a strong influence on the distribution of positive and negative predictions, as visible in Figure 24. The higher the positive weight, the more positive predictions there are. At very high positive weights, the number of false negatives vanishes, whereas the number of false positives continues to rise.

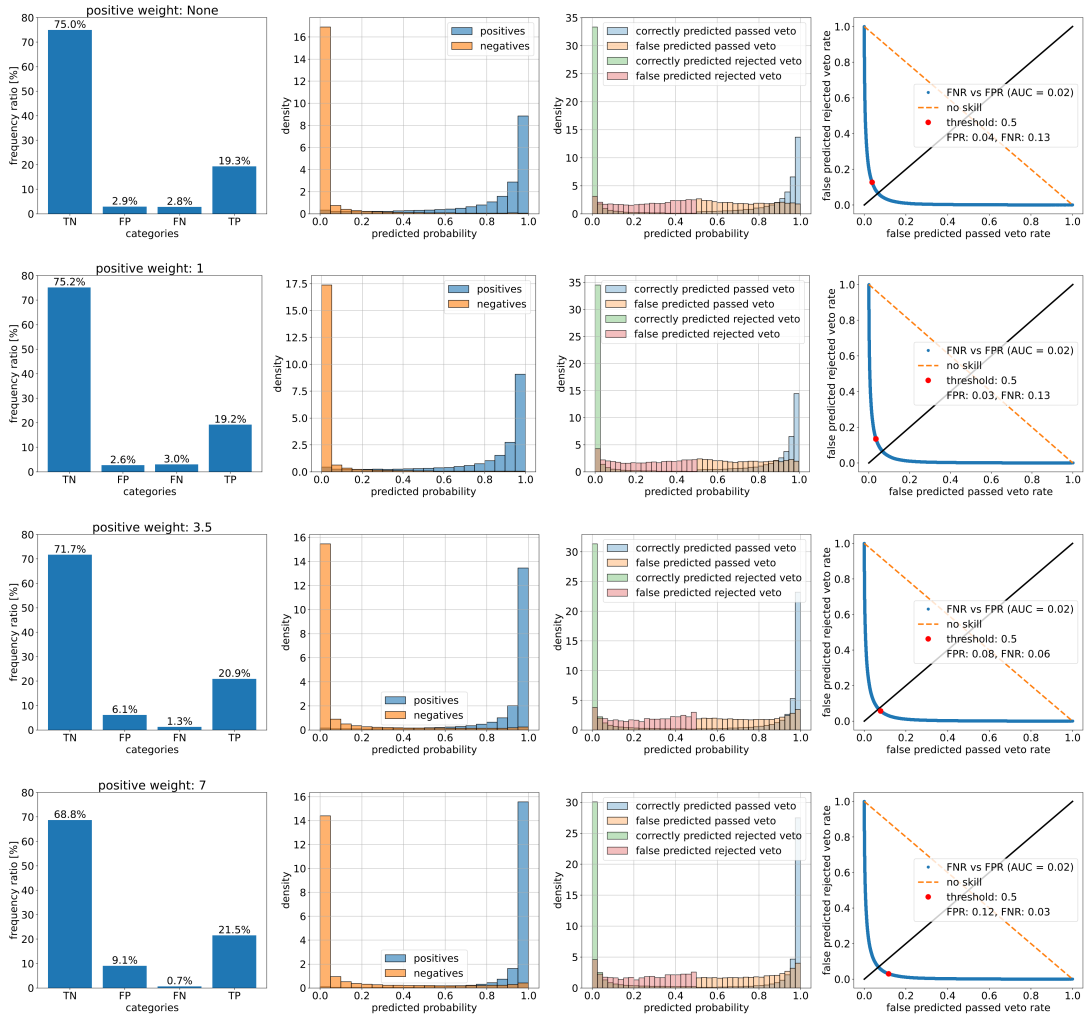


Figure 24: Validation plots for the positive weighting of `None`, 1, 3.5 (balanced) and 7. The rest of the plots (for weighting between one and ten) can be found in the appendix as Figure 44 and Figure 45. The first column shows the percentage of true and false predictions. In the second column, the distribution of positive and negative outcomes is displayed. The third column shows the probability distribution of true and false predictions. In the fourth column, the false negative rate is plotted over the false negative rate. The red dot shows the false positive rate and false negative rate for a threshold of 0.5.

Figure 25 and Figure 26 show how the different prediction ratios and the accuracy behave for the different positive weights. With increasing positive weights, the accuracy decreases for low thresholds but slightly increases for high thresholds. The total predicted passed rate decreases for all thresholds, whereas the total predicted rejected rate increases. As expected, the actually passed and rejected values do not change. The incorrect predictions behave similar to the total predictions. Their behavior can be better described by tilting. The false predicted passed rates tilt on the left side upwards and the false predicted rejected rates tilt on the right side downwards. The remaining nine plots are provided in the appendix as Figure 46 and Figure 47.

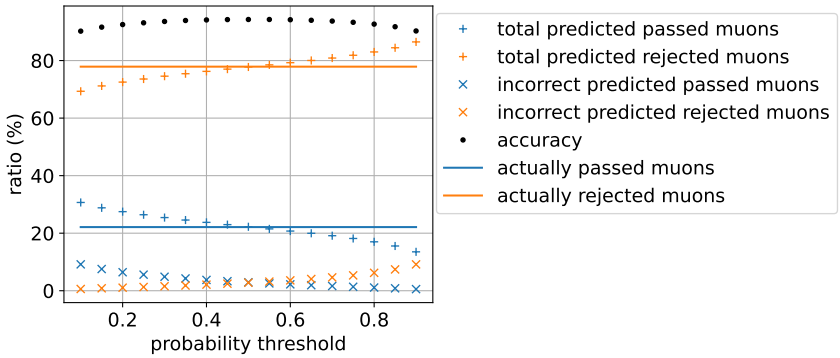


Figure 25: Threshold comparison at a positive weight of `None`. The black dots represent the model's accuracy dependent on the threshold, whereas the solid lines are the actual rates of passed and rejected muons from the training data. The Xs display all incorrect predictions, whereas the plusses show the total number of positive and negative predictions.

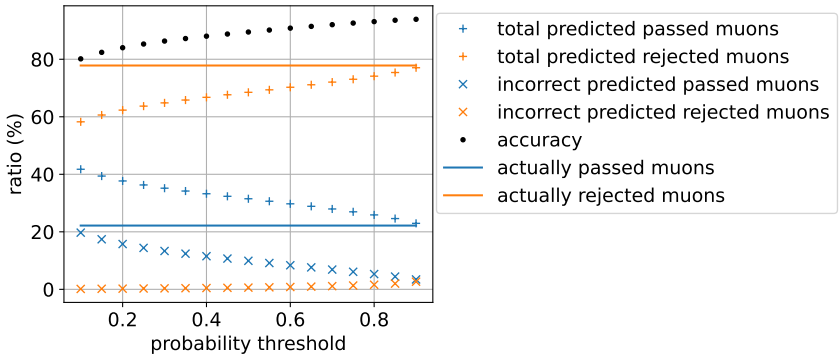


Figure 26: Threshold comparison at a positive weight of 10. The black dots represent the model's accuracy dependent on the threshold, whereas the solid lines are the actual rates of passed and rejected muons from the training data. The Xs display all incorrect predictions, whereas the plusses show the total number of positive and negative predictions.

Table 6 and Table 7 show the error rates, true and false prediction rates and the resulting savings of a model with positive weight of `None` and 10. As in subsubsection 4.2.5, 5 %

of false passed muons are acceptable, which means for Table 6 that less than 1.55 % of muons are mistakenly rejected and about 74 % of muons are not simulated (savings). The corresponding threshold is between 0.25 and 0.3. In Table 7 at 5.3 % false positive the false rejected rate is still at around 1.6 % with a total saving of 74 %. Here, the corresponding threshold is 0.8.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
9.77553	0.1	21.5109	9.1672	68.7136	0.608334	69.3219
8.38348	0.15	21.2768	7.54102	70.3397	0.842456	71.1822
7.4878	0.2	21.0587	6.42721	71.4535	1.06059	72.5141
6.88608	0.25	20.8245	5.59137	72.2894	1.29471	73.5841
6.41701	0.3	20.5659	4.86364	73.0171	1.55337	74.5705
6.07699	0.35	20.3066	4.2644	73.6164	1.81259	75.4289
5.88396	0.4	19.997	3.76169	74.1191	2.12227	76.2413
5.76345	0.45	19.6669	3.31109	74.5697	2.45236	77.022
5.72512	0.5	19.3142	2.92006	74.9607	2.80506	77.7658
5.72512	0.55	18.9441	2.54998	75.3308	3.17514	78.5059
5.81447	0.6	18.5219	2.21714	75.6636	3.59733	79.261
5.99923	0.65	18.0415	1.92152	75.9592	4.07771	80.037
6.27692	0.7	17.4842	1.64189	76.2389	4.63503	80.8739
6.70325	0.75	16.7896	1.37358	76.5072	5.32967	81.8369
7.3342	0.8	15.895	1.10995	76.7708	6.22425	82.9951
8.24339	0.85	14.7133	0.837493	77.0433	7.4059	84.4492
9.7121	0.9	12.9634	0.556214	77.3245	9.15589	86.4804

Table 6: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of `None`.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
19.8589	0.1	22.0322	19.7176	58.1089	0.14126	58.2502
17.5977	0.15	21.9803	17.4045	60.422	0.193172	60.6152
15.9888	0.2	21.9348	15.7502	62.0764	0.238673	62.315
14.7057	0.25	21.8825	14.4147	63.4118	0.290999	63.7028
13.6741	0.3	21.8336	13.3343	64.4923	0.33981	64.8321
12.7978	0.35	21.7782	12.4025	65.424	0.395238	65.8192
11.9757	0.4	21.7189	11.5211	66.3055	0.454596	66.7601
11.2303	0.45	21.6452	10.702	67.1245	0.528225	67.6527
10.5289	0.5	21.5706	9.92604	67.9005	0.602888	68.5034
9.8489	0.55	21.4769	9.15232	68.6742	0.696579	69.3708
9.17982	0.6	21.3654	8.37177	69.4548	0.808056	70.2628
8.57135	0.65	21.2351	7.633	70.1935	0.938354	71.1319
7.97839	0.7	21.0699	6.87479	70.9518	1.10361	72.0554
7.42225	0.75	20.8545	6.10334	71.7232	1.31891	73.0421
6.91139	0.8	20.572	5.30997	72.5166	1.60143	74.118
6.43446	0.85	20.1737	4.43469	73.3919	1.99977	75.3916
6.10292	0.9	19.5085	3.43801	74.3885	2.66491	77.0534

Table 7: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 10.

The remaining nine tables are provided in the appendix (cf. Table 8 to Table 17).

The general trend is, with higher positive weight the chosen threshold needs to be also higher with no noteworthy difference in the saving. This means the positive weight has no influence on the savings, but with different positive weight and threshold combinations some extreme false positive or false negative rates can be chosen which are not always obtainable.

It is even possible to pick a positive weight of 9 and a threshold like 0.05, which would result in a false rejected rate of 0.1 % and still 55 % savings, but a miserable false passed rate of 23 % (cf. Table 18).

Please note, that the values in Table 6 are ab bit worse than in Table 5, because the corresponding model was trained at an earlier point of time and therefore less training data was used to train the model.

4.2.7 Resulting model performance

The speed of the final model measured on a "NVIDIA A100 Tensor Core-GPU" is determined to be $(9 \pm 1)\,\mu$s. The 403928 predictions (size of training data file) took about 3.33 seconds. On CPU ("AMD EPYC 7662 @2.0GHz") the model's speed is determined to be (1.3 ± 0.1) ms. Here, the 403928 predictions took about 510 seconds.

The errors stated are caused by the fact, that the model predicts in batches of 1,024. The resulting time is the mean duration to predict a batch divided by the size of the batch to get the time per prediction, and the error is the corresponding standard deviation divided by the batch size.

The error rate of this model is 6.1 % with a threshold of 0.27 and 5.4 % with a threshold of 0.5. The resulting validation plots are illustrated in Figure 27.

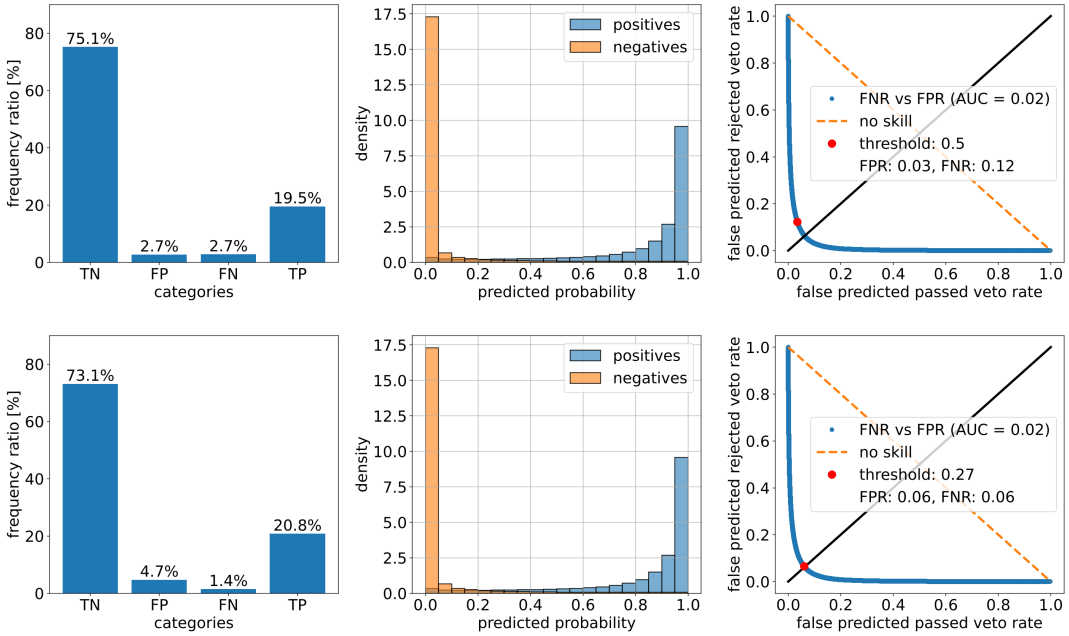


Figure 27: These are the validation results of the best performing model. For the upper plots the chosen threshold is 0.5 and for the lower ones the threshold is 0.27.

5 Conclusion

This research aimed to develop and optimize a Recurrent Neural Network (RNN) for improving the efficiency of muon simulation in IceCube research. By creating a dedicated tool to generate structured datasets, it was demonstrated that a neural network can effectively predict two key aspects: the total number of photons detected in IceCube's veto region from a track-like muon propagation event and whether a muon survives the veto decision. The latter, achieved using MuonGun simulation data and an optimized RNN architecture, allows for the efficient filtering of atmospheric muons, significantly reducing the computational cost of exhaustive simulations while maintaining accuracy.

Key findings highlight the importance of selecting an optimal set of hyperparameters, including the number of layers and neurons, fine-tuning learning rates, and incorporating dropout to balance generalization and performance. The final model, achieved a validation error rate of approximately 5.4 % or 6.1 %, depending on the chosen threshold and weighting. Additionally, thresholds tuning played a crucial role in optimizing false positive and false negative rates, enabling improved rejection of atmospheric muons at the cost of a higher false rejection rate.

While the developed model enhances simulation efficiency, further improvements could involve integrating additional physical constraints to enhance robustness. Future research could also explore alternative neural architectures, or expand the training dataset by incorporating additional features, such as time-dependent information, to further improve prediction accuracy.

In conclusion, this work demonstrates the potential of deep learning techniques to optimize particle physics simulations, paving the way for more efficient data processing for the IceCube Neutrino Observatory.

A Appendix

A.1 Parameter testing

These are the remaining learning curves from the modes discussed in subsubsection 4.2.1.

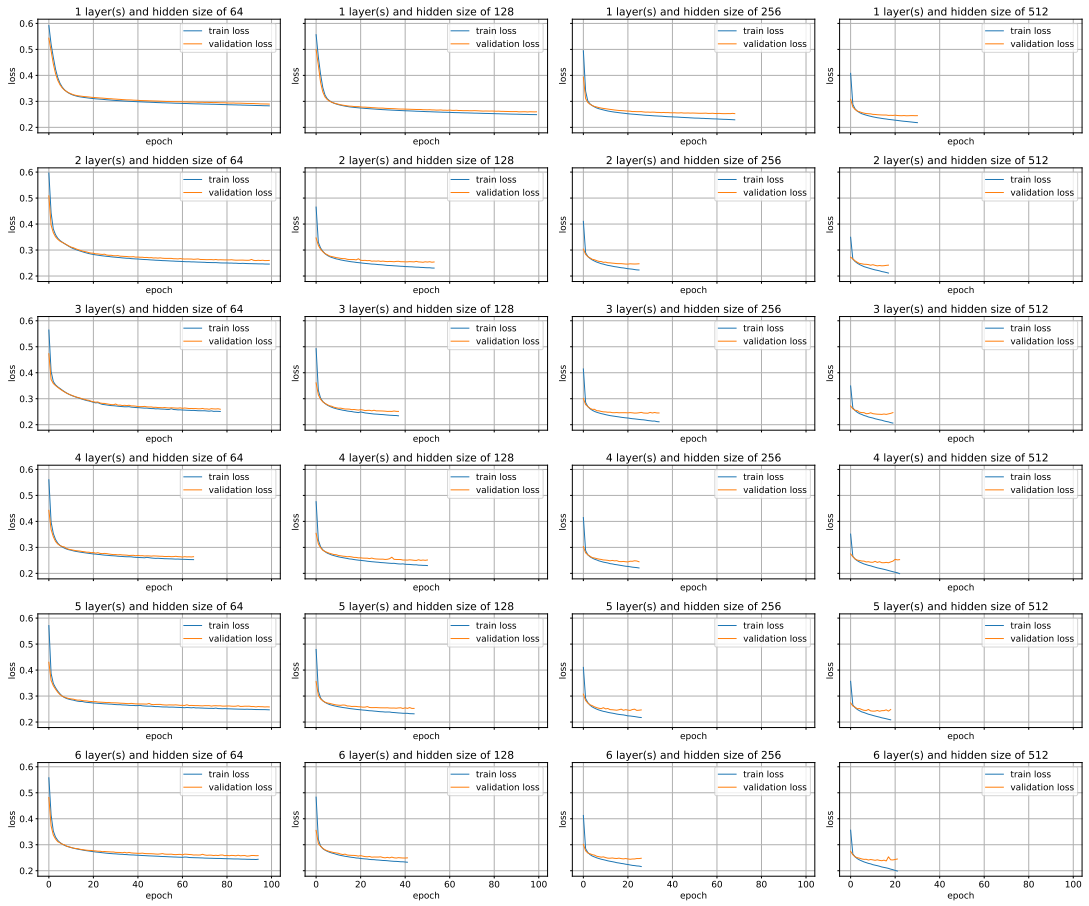


Figure 28: These are the learning curves of the models discussed in subsubsection 4.2.1. The rows include all models with the same amount of LSTM layers, and the columns include all models with the same hidden size. All of these models have a learning rate of 0.00005.

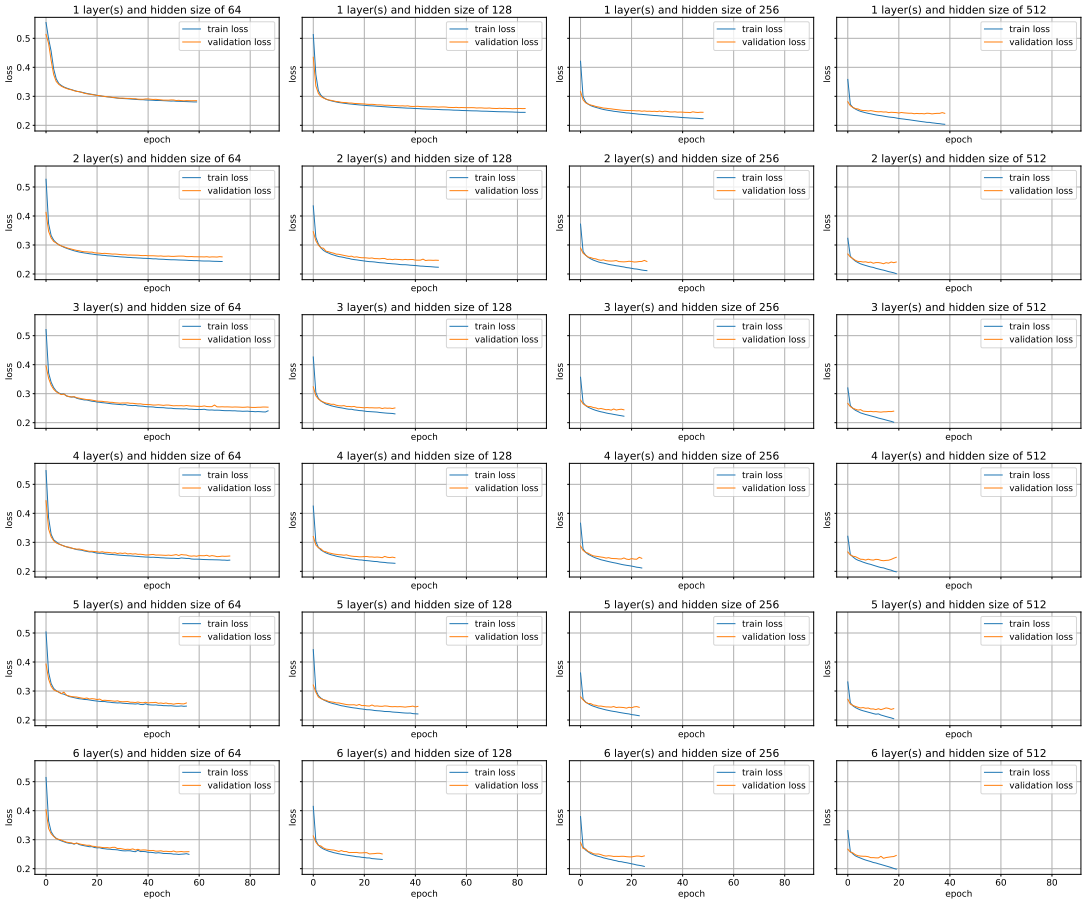


Figure 29: These are the learning curves of the models discussed in subsubsection 4.2.1. The rows include all models with the same amount of LSTM layers, and the columns include all models with the same hidden size. All of these models have a learning rate of 0.0001.

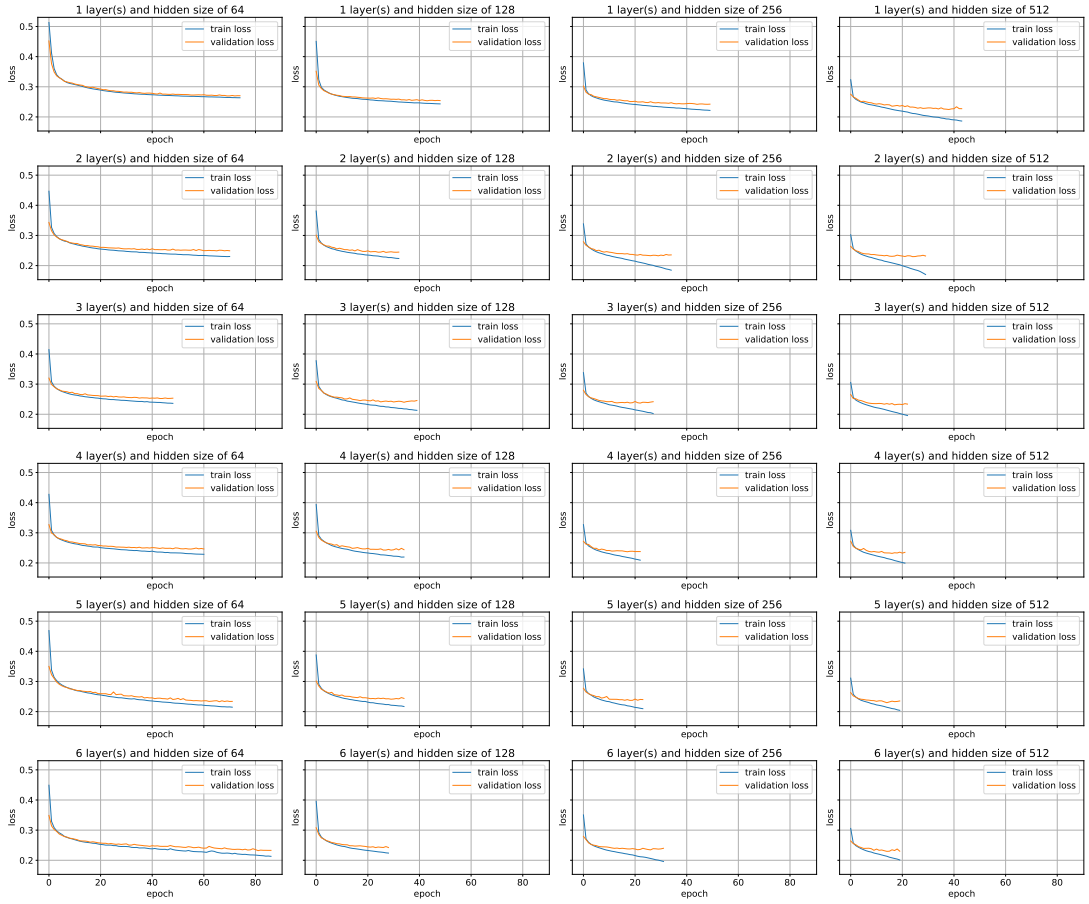


Figure 30: These are the learning curves of the models discussed in subsubsection 4.2.1. The rows include all models with the same amount of LSTM layers, and the columns include all models with the same hidden size. All of these models have a learning rate of 0.0002.

A.2 Testing fully connected (FC) layers

These are the validation plots corresponding to the models discussed in subsubsection 4.2.2, where the dropout function was tested.

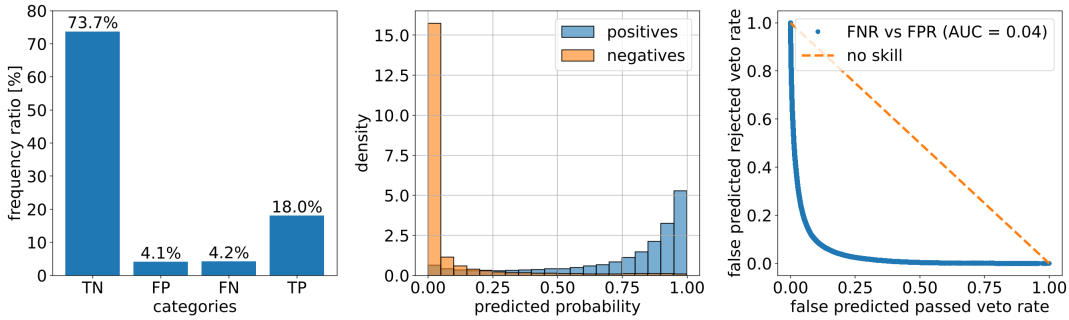


Figure 31: Left: percentage of true negative, false positive, false negative and true positive predictions. Center: distribution of positive and negative predictions. Right: false negative rate vs false positive rate. Model with two additional linear layers and no dropout function. Error rate: 8.25 %.

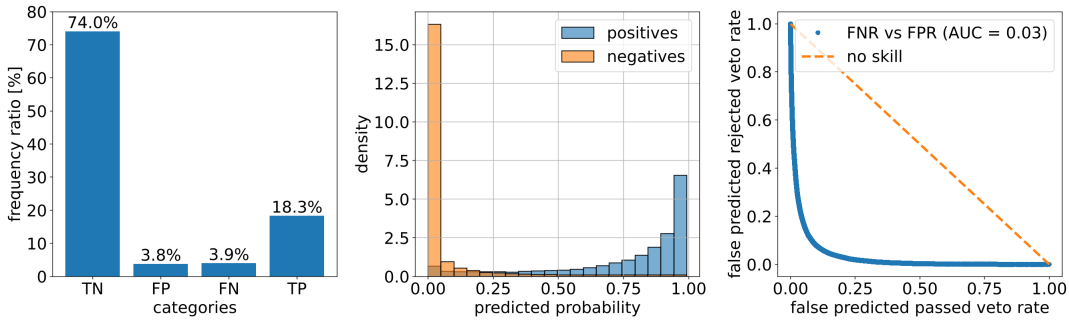


Figure 32: Left: percentage of true negative, false positive, false negative and true positive predictions. Center: distribution of positive and negative predictions. Right: false negative rate vs false positive rate. Model with two additional linear layers and a dropout function. Error rate: 7.69 %.

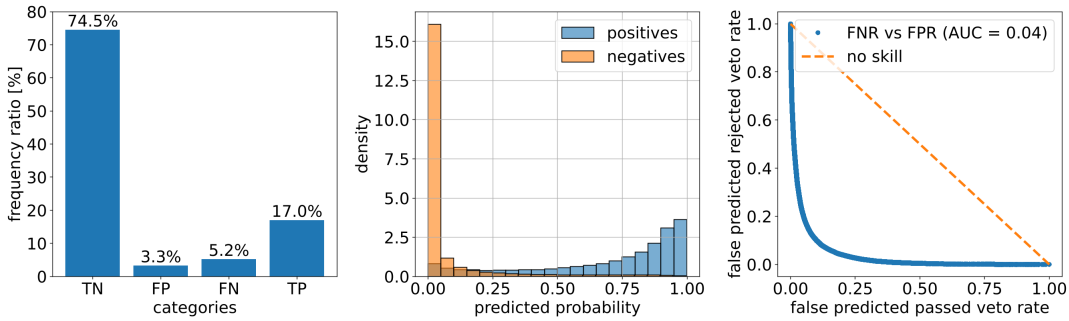


Figure 33: Left: percentage of true negative, false positive, false negative and true positive predictions. Center: distribution of positive and negative predictions. Right: false negative rate vs false positive rate. Model with three additional linear layers and no dropout function. Error rate: 8.47 %.

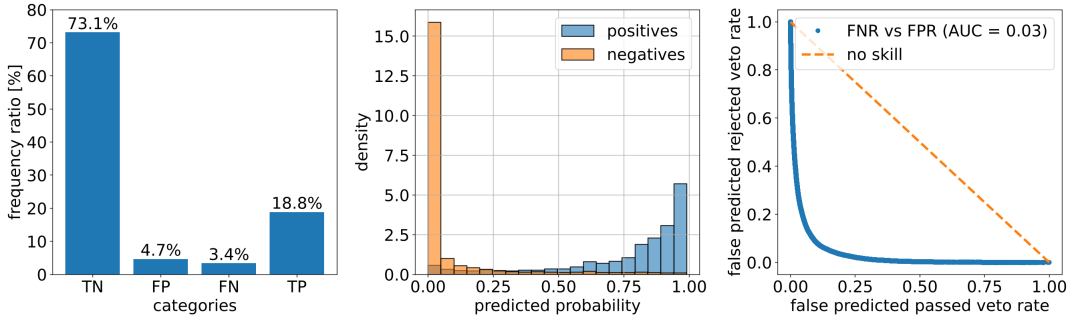


Figure 34: Left: percentage of true negative, false positive, false negative and true positive predictions. Center: distribution of positive and negative predictions. Right: false negative rate vs false positive rate. Model with three additional linear layers and a dropout function. Error rate: 8.02 %.

Here, one can find the learning curves corresponding to the RNN models discussed in Figure 19.

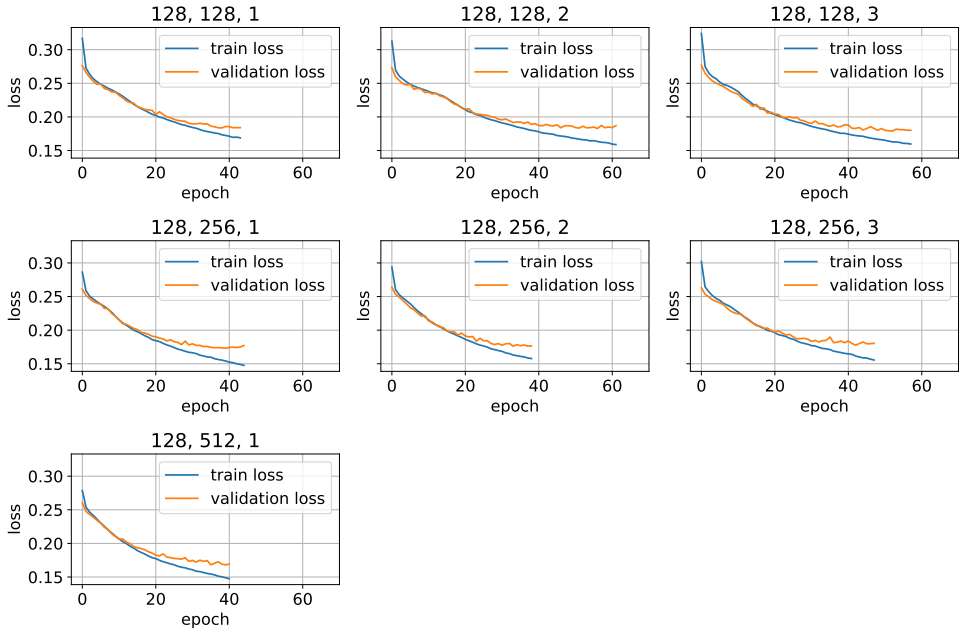


Figure 35: These are the learning curves of a RNN with a FC layer size of 128. The three numbers in the plot titles represent the FC layer size, the size of the LSTM layer and the number of LSTM layers.

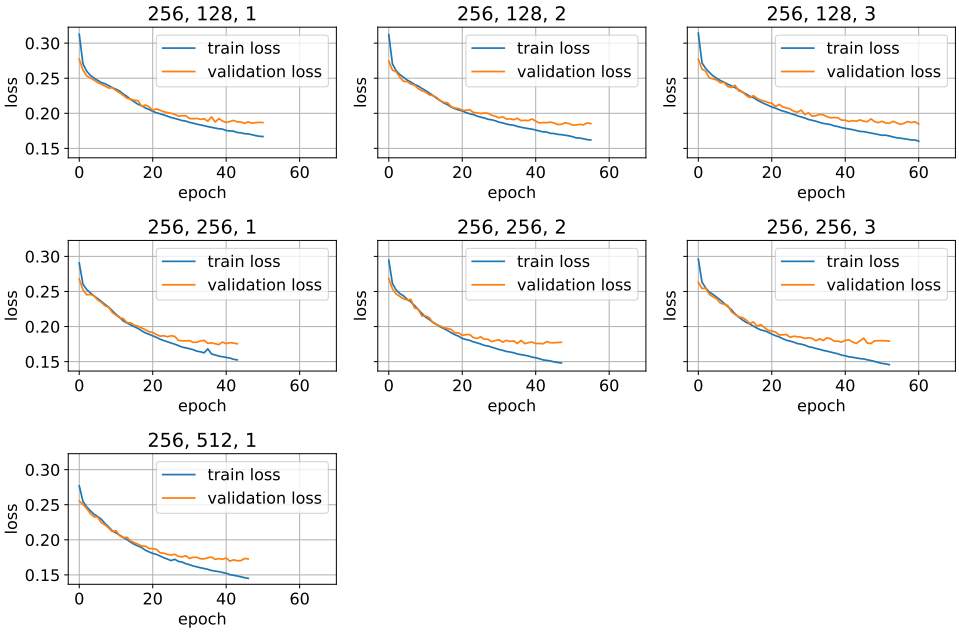


Figure 36: These are the learning curves of a RNN with a FC layer size of 256. The three numbers in the plot titles represent the FC layer size, the size of the LSTM layer and the number of LSTM layers.

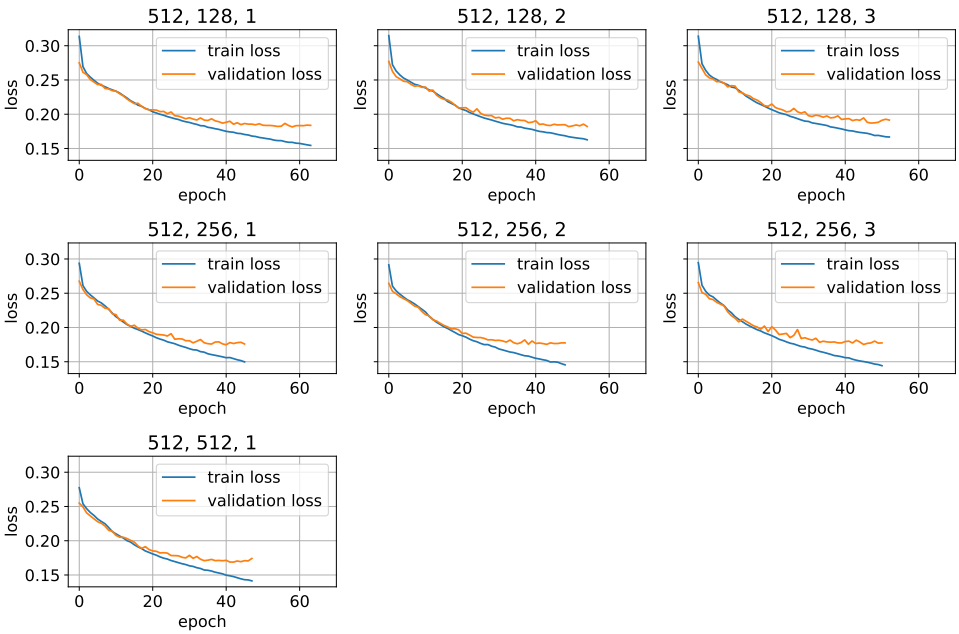


Figure 37: These are the learning curves of a RNN with a FC layer size of 512. The three numbers in the plot titles represent the FC layer size, the size of the LSTM layer and the number of LSTM layers.

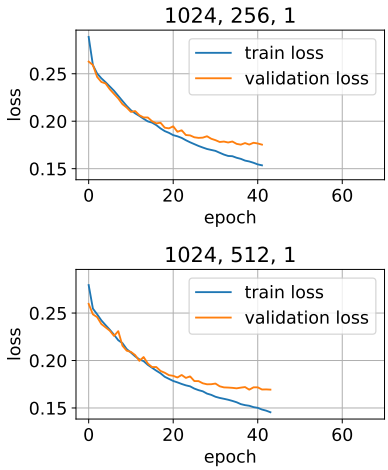


Figure 38: These are the learning curves of a RNN with a FC layer size of 1024. The three numbers in the plot titles represent the FC layer size, the size of the LSTM layer and the number of LSTM layers.

Here, one can find the testing loss comparison for the models discussed in Figure 19.

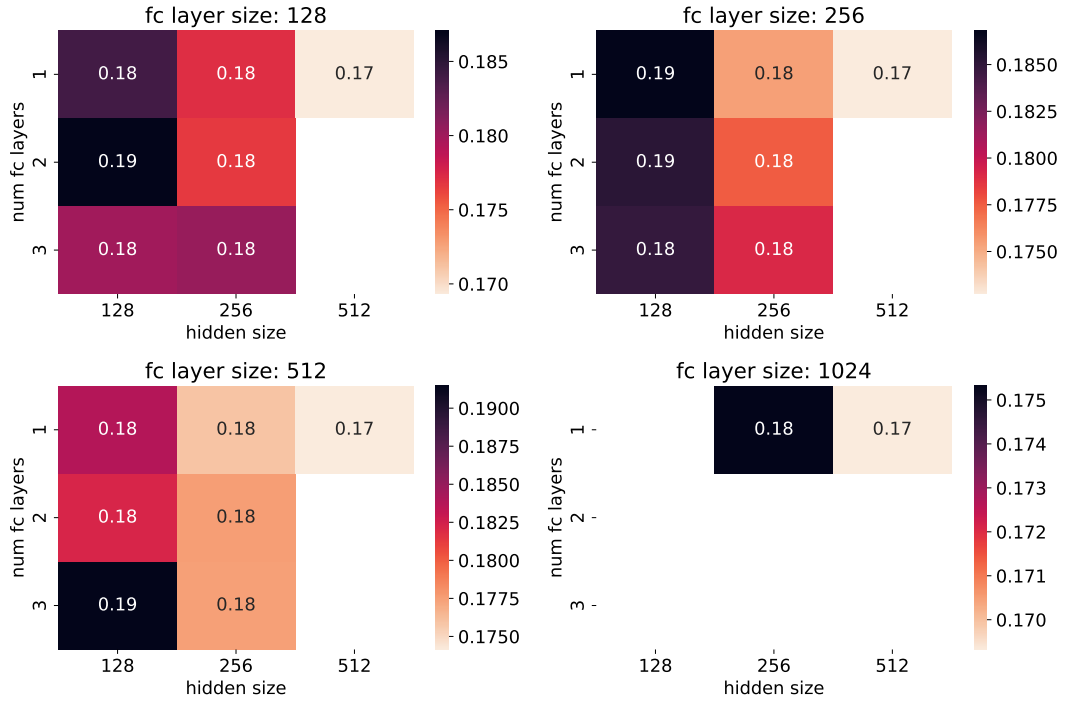


Figure 39: These plots illustrate the same as in Figure 19, but this time the testing loss is displayed. The brighter the color, the lower the validation loss.

A.3 Learning rate optimization

These are the learning curves, corresponding to the learning rate optimization in subsubsection 4.2.3.

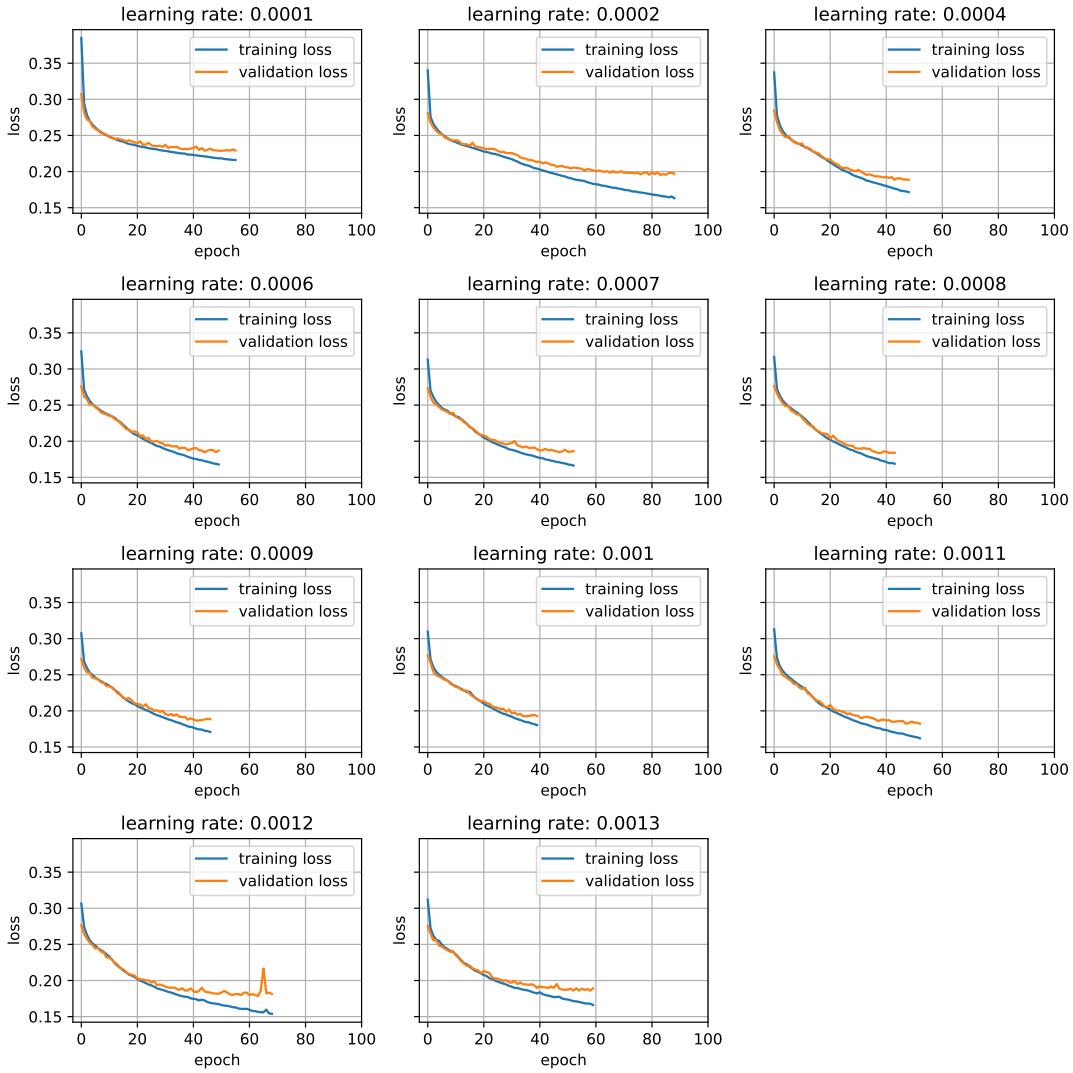


Figure 40: These are the learning curves for the model, where different learning rates are tested. The model has two LSTM layers with 128 neurons, 1 FC layer with also 128 neurons and dropout is enabled.

A.4 Threshold comparison

These are the remaining validation plots, which are used for the threshold comparison in subsubsection 4.2.5.

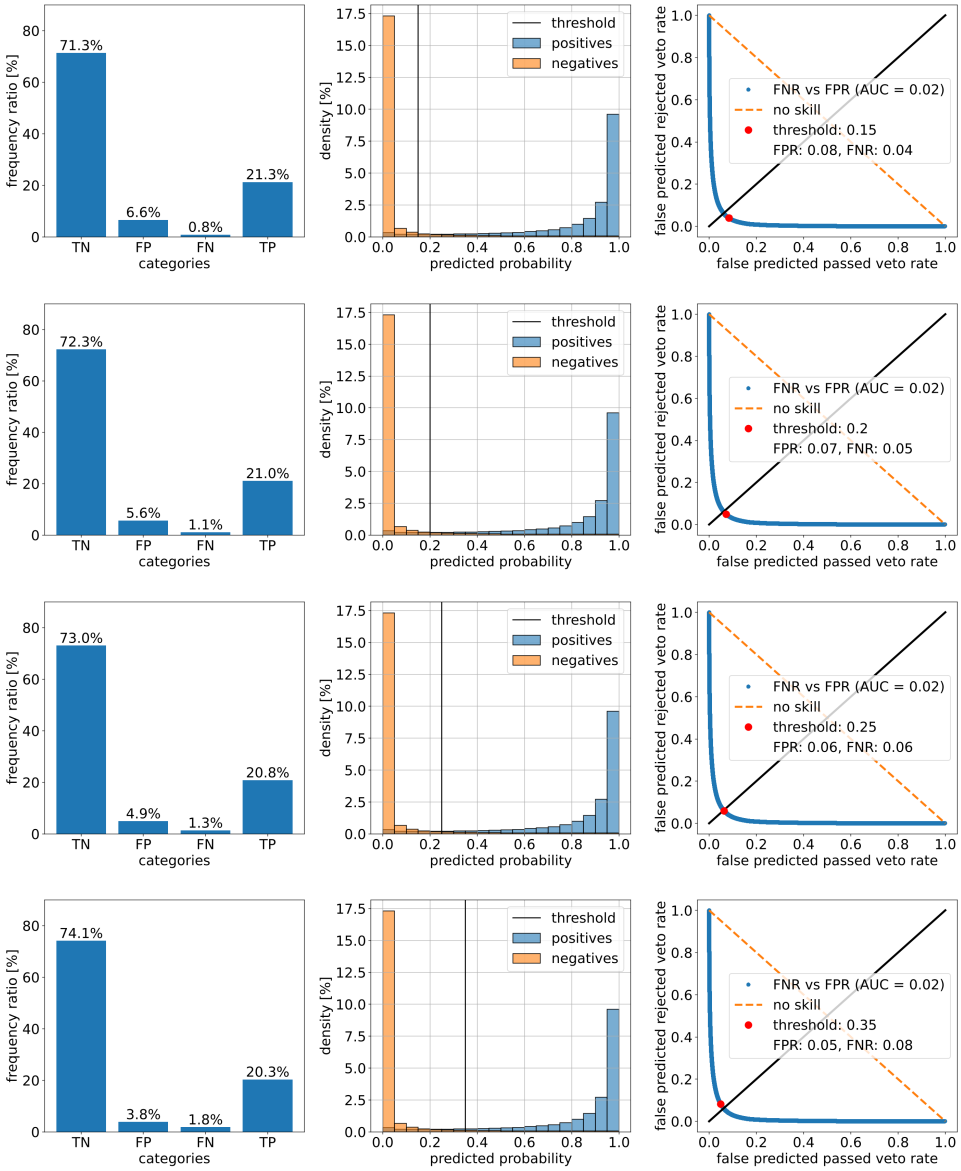


Figure 41: Left: Percentage of true and false predictions. Center: Histogram of positives and negatives with position. The black vertical line marks the position of the threshold. Right: False negatives rate vs false positives rate. The red dot marks the current threshold. The black diagonal helps to show how close the red dot (threshold) is to the optimum position. The closer the curve gets to the bottom left corner, the more skilled is the model.

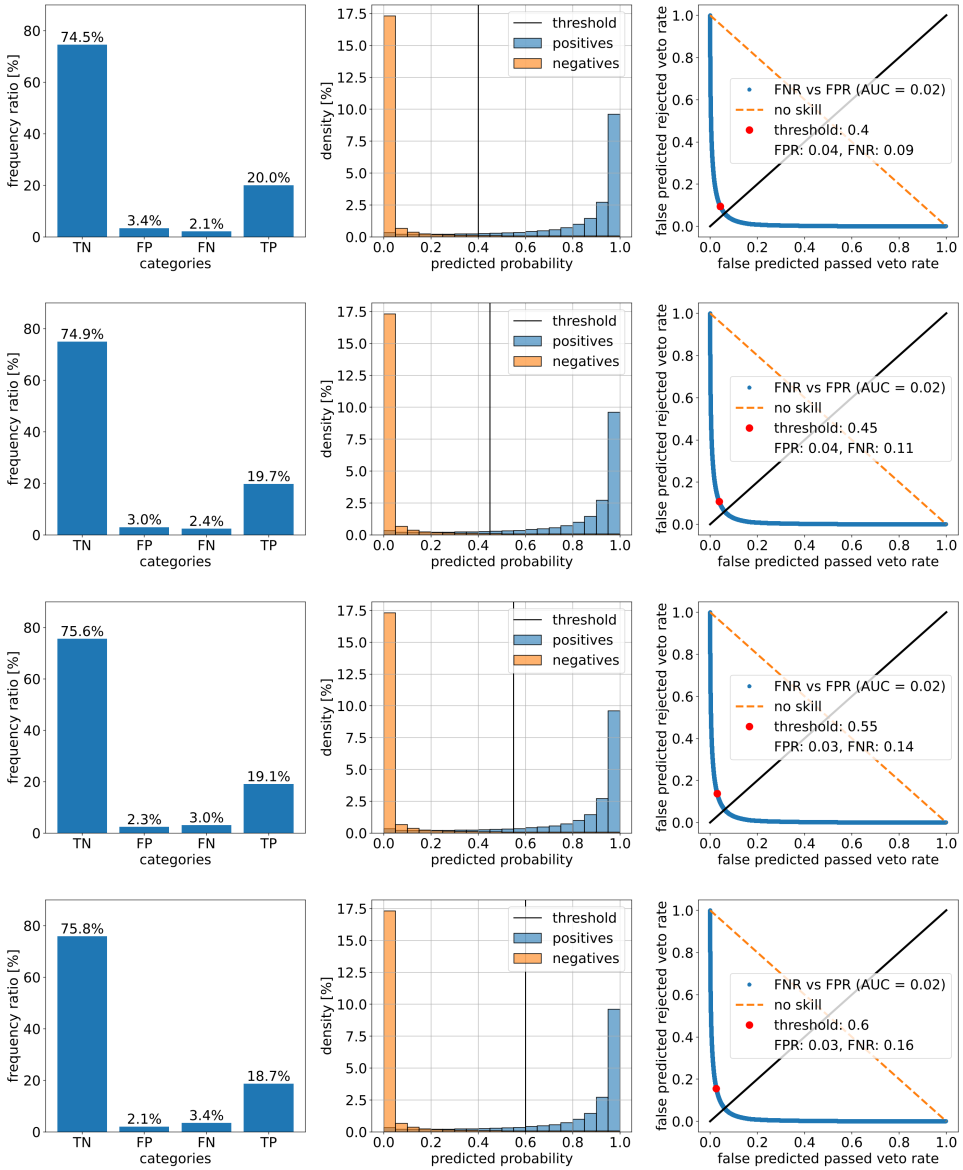


Figure 42: Left: Percentage of true and false predictions. Center: Histogram of positives and negatives with position. The black vertical line marks the position of the threshold. Right: False negatives rate vs false positives rate. The red dot marks the current threshold. The black diagonal helps to show how close the red dot (threshold) is to the optimum position. The closer the curve gets to the bottom left corner, the more skilled is the model.

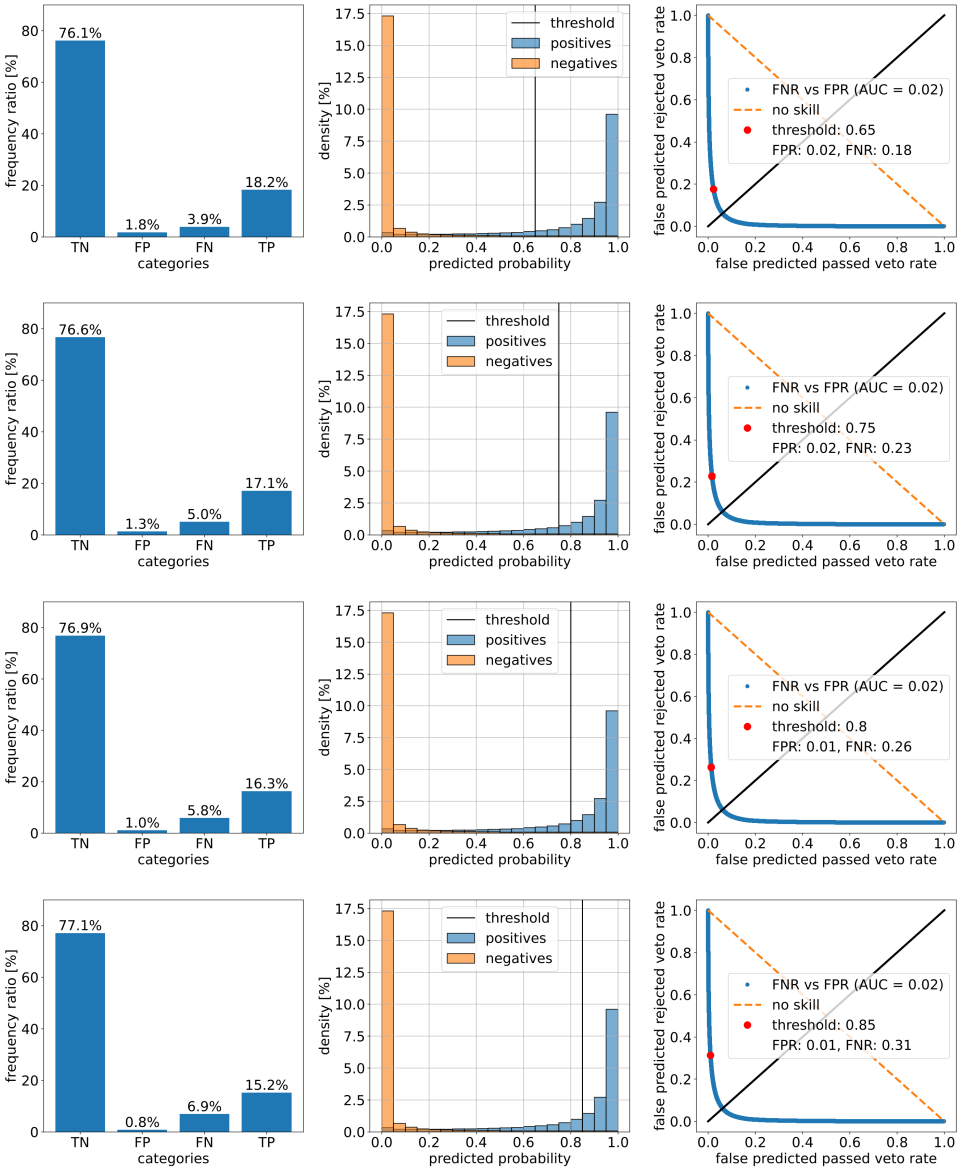


Figure 43: Left: Percentage of true and false predictions. Center: Histogram of positives and negatives with position. The black vertical line marks the position of the threshold. Right: False negatives rate vs false positives rate. The red dot marks the current threshold. The black diagonal helps to show how close the red dot (threshold) is to the optimum position. The closer the curve gets to the bottom left corner, the more skilled is the model.

A.5 Weight optimization

The remaining plots and tables discussed in subsubsection 4.2.6 are provided here.

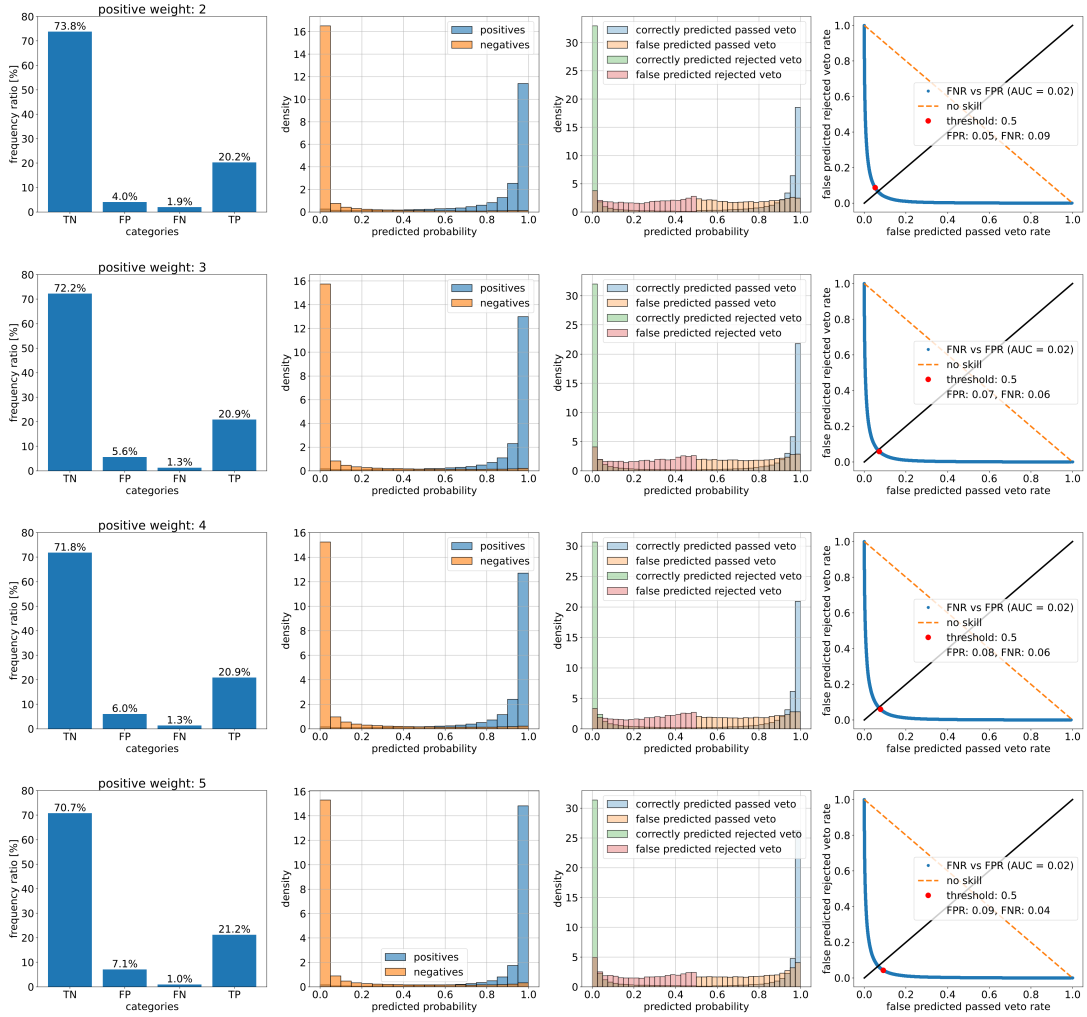


Figure 44: Validation plots for the positive weighting of 2, 3, 4 and 5. The first column shows the percentage of true and false predictions. In the second column, the distribution of positive and negative outcomes is displayed. The third column shows the probability distribution of true and false predictions. In the fourth column, the false negative rate is plotted over the false negative rate. The red dot shows the false positive rate and false negative rate for a threshold of 0.5.

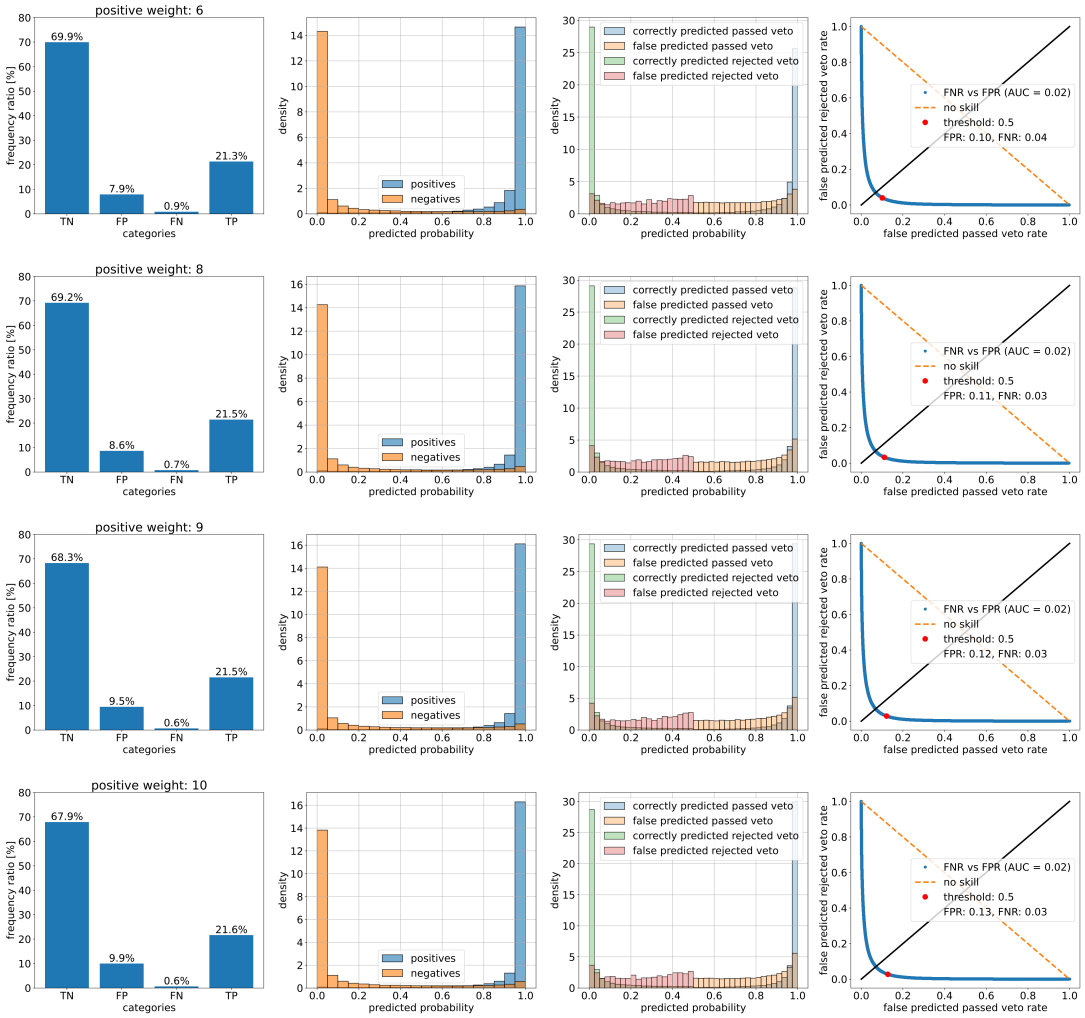


Figure 45: Validation plots for the positive weighting of 6, 8, 9 and 10. The first column shows the percentage of true and false predictions. In the second column, the distribution of positive and negative outcomes is displayed. The third column shows the probability distribution of true and false predictions. In the fourth column, the false negative rate is plotted over the false negative rate. The red dot shows the false positive rate and false negative rate for a threshold of 0.5.

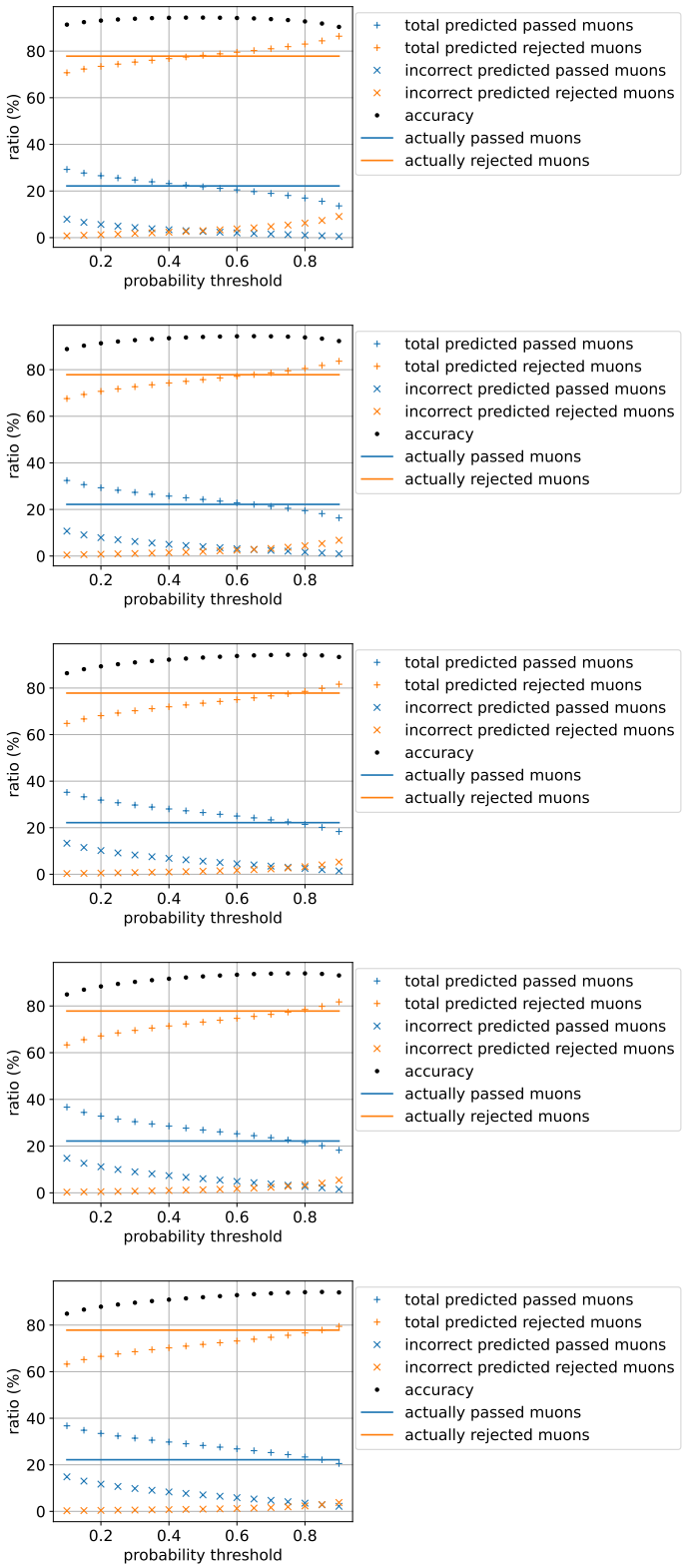


Figure 46: Threshold comparison for positive weights from 1 to 5.

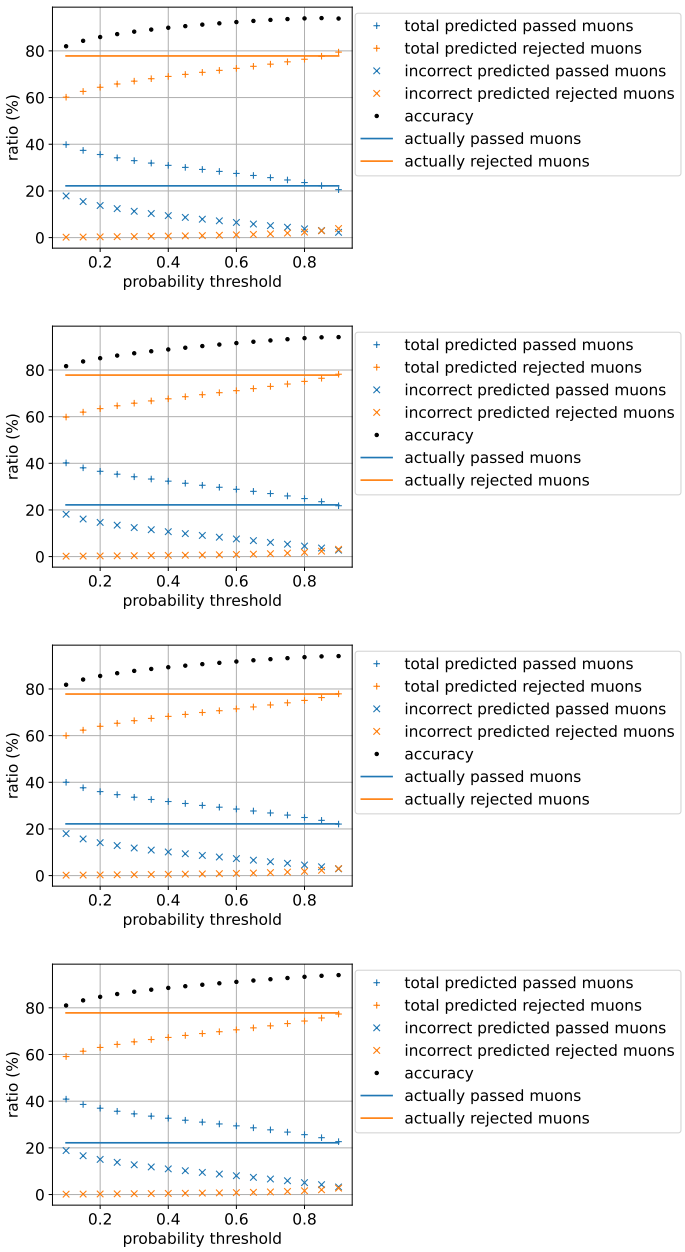


Figure 47: Threshold comparison for positive weights from 6 to 9.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
8.65553	0.1	21.3983	7.88036	69.9462	0.775171	70.7214
7.58398	0.15	21.1489	6.55938	71.2672	1.0246	72.2918
6.92277	0.2	20.9052	5.65453	72.172	1.26824	73.4402
6.44852	0.25	20.6612	4.93624	72.8903	1.51229	74.4026
6.11099	0.3	20.3933	4.33087	73.4957	1.78012	75.2758
5.86942	0.35	20.1269	3.82291	74.0036	2.04651	76.0501
5.72692	0.4	19.8359	3.38941	74.4371	2.33751	76.7746
5.64502	0.45	19.521	2.99252	74.834	2.6525	77.4865
5.63488	0.5	19.1882	2.64961	75.1769	2.98528	78.1622
5.69714	0.55	18.8169	2.34061	75.4859	3.35653	78.8425
5.81585	0.6	18.4145	2.05685	75.7697	3.759	79.5287
5.996	0.65	17.9671	1.78964	76.0369	4.20636	80.2433
6.29423	0.7	17.4329	1.55365	76.2729	4.74058	81.0135
6.70333	0.75	16.7754	1.30526	76.5213	5.39807	81.9194
7.28512	0.8	15.9355	1.04714	76.7794	6.23798	83.0174
8.18273	0.85	14.7957	0.804954	77.0216	7.37778	84.3994
9.62697	0.9	13.0815	0.53505	77.2915	9.09192	86.3834

Table 8: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 1.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
11.1581	0.1	21.7174	10.702	67.1245	0.456044	67.5805
9.70019	0.15	21.5577	9.08448	68.7421	0.615711	69.3578
8.66835	0.2	21.4047	7.89959	69.927	0.76876	70.6957
7.90828	0.25	21.2591	6.99392	70.8326	0.914363	71.747
7.32421	0.3	21.0901	6.24087	71.5857	1.08334	72.669
6.87437	0.35	20.9027	5.60365	72.2229	1.27072	73.4936
6.48927	0.4	20.7056	5.02145	72.8051	1.46782	74.2729
6.17697	0.45	20.5003	4.50377	73.3228	1.6732	74.996
5.96166	0.5	20.2488	4.03697	73.7896	1.92469	75.7143
5.78917	0.55	19.9766	3.5923	74.2342	2.19687	76.4311
5.66839	0.6	19.6692	3.16418	74.6624	2.50421	77.1666
5.62847	0.65	19.3292	2.78425	75.0423	2.84423	77.8865
5.68556	0.7	18.9288	2.44092	75.3856	3.24463	78.6303
5.84377	0.75	18.4126	2.08291	75.7436	3.76086	79.5045
6.14594	0.8	17.7574	1.72986	76.0967	4.41608	80.5128
6.69175	0.85	16.8395	1.35779	76.4688	5.33396	81.8027
7.72007	0.9	15.4087	0.955314	76.8712	6.76476	83.636

Table 9: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 2.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
13.6282	0.1	21.8767	13.3314	64.4952	0.296791	64.792
11.9064	0.15	21.7716	11.5045	66.322	0.401856	66.7239
10.6851	0.2	21.6742	10.1858	67.6407	0.49927	68.14
9.7701	0.25	21.5677	9.16431	68.6622	0.605784	69.268
8.99679	0.3	21.456	8.27932	69.5472	0.717468	70.2647
8.38335	0.35	21.3288	7.53869	70.2879	0.844664	71.1325
7.81293	0.4	21.207	6.84645	70.9801	0.966482	71.9466
7.3422	0.45	21.0492	6.21792	71.6086	1.12429	72.7329
6.91077	0.5	20.886	5.6233	72.2032	1.28747	73.4907
6.55421	0.55	20.6879	5.0686	72.7579	1.48561	74.2435
6.23612	0.6	20.4725	4.53521	73.2913	1.70091	74.9922
5.99724	0.65	20.1995	4.02332	73.8032	1.97392	75.7771
5.80944	0.7	19.8672	3.50316	74.3234	2.30628	76.6297
5.70748	0.75	19.4742	3.00824	74.8183	2.69924	77.5175
5.74636	0.8	18.9301	2.50297	75.3236	3.24339	78.567
6.00861	0.85	18.1499	1.98508	75.8415	4.02353	79.865
6.6601	0.9	16.9392	1.42584	76.4007	5.23427	81.635

Table 10: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 3.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
14.4598	0.1	21.8825	14.1688	63.6577	0.290999	63.9487
12.5992	0.15	21.7813	12.2071	65.6195	0.392136	66.0116
11.3403	0.2	21.6825	10.8493	66.9772	0.490997	67.4682
10.3635	0.25	21.5855	9.77547	68.0511	0.587997	68.6391
9.57361	0.3	21.4688	8.86897	68.9576	0.704645	69.6622
8.91385	0.35	21.3518	8.09214	69.7344	0.821706	70.5561
8.33764	0.4	21.2227	7.38688	70.4397	0.950764	71.3904
7.82286	0.45	21.0765	6.72587	71.1007	1.09699	72.1977
7.36744	0.5	20.9131	6.10706	71.7195	1.26038	72.9799
6.97261	0.55	20.7178	5.51699	72.3095	1.45562	73.7652
6.59537	0.6	20.4951	4.917	72.9095	1.67837	74.5879
6.30292	0.65	20.2277	4.35713	73.4694	1.94579	75.4152
6.07707	0.7	19.8986	3.80223	74.0243	2.27484	76.2992
5.93871	0.75	19.5015	3.26676	74.5598	2.67194	77.2317
5.94057	0.8	18.9704	2.7375	75.089	3.20306	78.2921
6.11388	0.85	18.2598	2.20018	75.6264	3.91371	79.5401
6.67355	0.9	17.1351	1.63514	76.1914	5.03841	81.2298

Table 11: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 3.5.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
15.0734	0.1	21.8938	14.7938	63.0327	0.279624	63.3123
13.0331	0.15	21.791	12.6507	65.1758	0.382415	65.5582
11.6327	0.2	21.6934	11.1527	66.6738	0.480035	67.1539
10.5529	0.25	21.5888	9.96823	67.8583	0.584688	68.443
9.68488	0.3	21.4701	8.98148	68.8451	0.703404	69.5485
8.96804	0.35	21.342	8.13661	69.6899	0.831427	70.5214
8.34943	0.4	21.1968	7.37281	70.4537	0.976617	71.4303
7.80714	0.45	21.0337	6.66734	71.1592	1.1398	72.299
7.34738	0.5	20.8634	6.03736	71.7892	1.31001	73.0992
6.96641	0.55	20.6543	5.4473	72.3792	1.51911	73.8984
6.63073	0.6	20.4198	4.87709	72.9495	1.75365	74.7031
6.35566	0.65	20.1449	4.32714	73.4994	2.02852	75.5279
6.15959	0.7	19.8076	3.79375	74.0328	2.36584	76.3986
6.04812	0.75	19.3814	3.25601	74.5705	2.79211	77.3626
6.04998	0.8	18.8209	2.69738	75.1292	3.3526	78.4818
6.28534	0.85	18.0296	2.14144	75.6851	4.1439	79.829
6.92649	0.9	16.7545	1.50753	76.319	5.41896	81.738

Table 12: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 4.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
15.1012	0.1	21.9071	14.8348	62.9918	0.266388	63.2582
13.3506	0.15	21.8291	13.0063	64.8203	0.34436	65.1647
12.1097	0.2	21.7582	11.6944	66.1322	0.4153	66.5475
11.1786	0.25	21.6887	10.6938	67.1328	0.484792	67.6176
10.3858	0.3	21.6107	9.82304	68.0035	0.562764	68.5663
9.6884	0.35	21.5294	9.04436	68.7822	0.644046	69.4262
9.08634	0.4	21.4353	8.34819	69.4784	0.73815	70.2165
8.53805	0.45	21.3373	7.70187	70.1247	0.836184	70.9609
8.03113	0.5	21.2221	7.07975	70.7468	0.951384	71.6982
7.57302	0.55	21.0899	6.48947	71.3371	1.08354	72.4206
7.12773	0.6	20.9393	5.89362	71.9329	1.23411	73.167
6.71905	0.65	20.7602	5.30583	72.5207	1.41322	73.9339
6.37159	0.7	20.5251	4.72321	73.1033	1.64838	74.7517
6.09031	0.75	20.2283	4.14514	73.6814	1.94517	75.6266
5.86549	0.8	19.8331	3.52509	74.3015	2.34041	76.6419
5.7689	0.85	19.2755	2.87091	74.9556	2.898	77.8536
5.97056	0.9	18.3729	2.16998	75.6566	3.80057	79.4571

Table 13: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 5.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
18.03	0.1	21.9948	17.8513	59.9753	0.178695	60.154
15.6945	0.15	21.9242	15.4453	62.3812	0.249221	62.6305
14.0737	0.2	21.85	13.7502	64.0764	0.323471	64.3998
12.8207	0.25	21.7716	12.4189	65.4077	0.401856	65.8095
11.7653	0.3	21.6909	11.2828	66.5437	0.482517	67.0263
10.8778	0.35	21.6111	10.3155	67.5111	0.562351	68.0734
10.0878	0.4	21.5185	9.43277	68.3938	0.655007	69.0488
9.40878	0.45	21.421	8.65636	69.1702	0.752421	69.9226
8.75129	0.5	21.3114	7.88925	69.9373	0.862037	70.7993
8.20714	0.55	21.1609	7.19453	70.632	1.0126	71.6446
7.66402	0.6	21.0016	6.49216	71.3344	1.17186	72.5062
7.17592	0.65	20.8084	5.81089	72.0157	1.36503	73.3807
6.73932	0.7	20.565	5.13086	72.6957	1.60846	74.3041
6.38089	0.75	20.2432	4.45062	73.3759	1.93028	75.3062
6.09941	0.8	19.8316	3.75755	74.069	2.34185	76.4108
5.96435	0.85	19.2424	3.03326	74.7933	2.93109	77.7244
6.14574	0.9	18.2798	2.25209	75.5744	3.89364	79.4681

Table 14: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 6.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
18.3323	0.1	22.0039	18.1628	59.6638	0.169595	59.8334
16.3289	0.15	21.9542	16.1096	61.7169	0.219232	61.9361
14.9347	0.2	21.8992	14.6604	63.1661	0.274247	63.4404
13.7998	0.25	21.8506	13.477	64.3496	0.32285	64.6724
12.8201	0.3	21.789	12.4356	65.3909	0.384483	65.7754
11.9603	0.35	21.7329	11.5198	66.3067	0.440532	66.7473
11.1881	0.4	21.6632	10.6778	67.1487	0.510232	67.6589
10.435	0.45	21.5908	9.85241	67.9741	0.582619	68.5568
9.73059	0.5	21.5114	9.06855	68.758	0.662039	69.42
9.08014	0.55	21.4055	8.3122	69.5143	0.767933	70.2823
8.45367	0.6	21.2864	7.56661	70.2599	0.887062	71.147
7.87581	0.65	21.1342	6.83652	70.99	1.03928	72.0293
7.32132	0.7	20.9398	6.08762	71.7389	1.2337	72.9726
6.7995	0.75	20.6903	5.31638	72.5102	1.48313	73.9933
6.34222	0.8	20.352	4.52073	73.3058	1.82149	75.1273
5.98669	0.85	19.8639	3.6771	74.1494	2.30959	76.459
5.88597	0.9	19.0475	2.76005	75.0665	3.12592	78.1924

Table 15: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 7.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
18.1847	0.1	21.9962	18.0074	59.8191	0.177247	59.9964
15.9518	0.15	21.9358	15.7142	62.1124	0.237639	62.35
14.4168	0.2	21.8808	14.1241	63.7024	0.292654	63.9951
13.2441	0.25	21.8154	12.8861	64.9405	0.35801	65.2985
12.2487	0.3	21.7584	11.8336	65.993	0.415093	66.4081
11.4179	0.35	21.6895	10.9339	66.8927	0.483965	67.3766
10.6859	0.4	21.6186	10.131	67.6955	0.554905	68.2504
10.0007	0.45	21.5416	9.36886	68.4577	0.631843	69.0895
9.35914	0.5	21.4516	8.63733	69.1892	0.721811	69.911
8.79307	0.55	21.3497	7.96929	69.8573	0.823775	70.681
8.23506	0.6	21.2322	7.29381	70.5327	0.94125	71.474
7.70601	0.65	21.0837	6.61626	71.2103	1.08975	72.3
7.23114	0.7	20.8967	5.95442	71.8721	1.27672	73.1488
6.78957	0.75	20.6603	5.27646	72.5501	1.51311	74.0632
6.36642	0.8	20.3567	4.54969	73.2769	1.81673	75.0936
6.04108	0.85	19.9077	3.77534	74.0512	2.26574	76.3169
5.91058	0.9	19.177	2.91413	74.9124	2.99645	77.9089

Table 16: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 8.

error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
19.0006	0.1	22.0223	18.8494	58.9771	0.151187	59.1283
16.8213	0.15	21.9743	16.6221	61.2044	0.19917	61.4036
15.2989	0.2	21.9284	15.0538	62.7727	0.245085	63.0178
14.095	0.25	21.8773	13.7988	64.0277	0.29617	64.3239
13.1041	0.3	21.8188	12.7494	65.0772	0.354701	65.4319
12.2315	0.35	21.7668	11.8249	66.0017	0.406613	66.4083
11.4683	0.4	21.7009	10.9957	66.8308	0.47259	67.3034
10.754	0.45	21.6287	10.2092	67.6174	0.544771	68.1621
10.113	0.5	21.5439	9.48344	68.3431	0.629568	68.9727
9.49957	0.55	21.4574	8.78355	69.043	0.71602	69.759
8.89131	0.6	21.3538	8.07167	69.7549	0.819638	70.5745
8.30496	0.65	21.2291	7.36061	70.4659	0.944352	71.4103
7.76371	0.7	21.0641	6.65431	71.1722	1.1094	72.2816
7.23176	0.75	20.8461	5.90437	71.9222	1.32739	73.2496
6.71657	0.8	20.5594	5.10252	72.724	1.61404	74.3381
6.28079	0.85	20.1253	4.23263	73.5939	2.04816	75.6421
5.98752	0.9	19.4264	3.2405	74.586	2.74702	77.3331

Table 17: Table of error rates (1 - accuracy), false and correct predictions and total savings (sum of negative predictions divided by the total number of predictions) for each threshold. This is the data for a positive weight of 9.

positive weight	error rate [%]	threshold	correctly passed [%]	false passed [%]	correctly rejected [%]	false rejected [%]	savings [%]
None	11.488	0.05	21.7222	11.0367	66.7899	0.451287	67.2412
	19.236	0.01	22.0264	19.0889	58.7376	0.147051	58.8847
	23.6549	0.005	22.0858	23.5672	54.2593	0.0876928	54.347
1	10.7792	0.05	21.6909	10.2967	67.5299	0.482517	68.0124
	17.1841	0.01	21.9917	17.0023	60.8243	0.181797	61.0061
	21.0068	0.005	22.0603	20.8936	56.9329	0.113132	57.046
2	13.9401	0.05	21.8918	13.6584	64.1682	0.281692	64.4499
	22.6355	0.01	22.0763	22.5383	55.2882	0.0972067	55.3855
	28.4375	0.005	22.1147	28.3788	49.4478	0.0587376	49.5065
3	16.7578	0.05	21.9801	16.5644	61.2621	0.193379	61.4555
	25.4934	0.01	22.0957	25.4156	52.4109	0.0777653	52.4887
	30.1583	0.005	22.1184	30.1032	47.7233	0.0550148	47.7783
3.5	17.8746	0.05	21.9915	17.6926	60.1339	0.182004	60.3159
	28.0255	0.01	22.1056	27.9577	49.8689	0.0678378	49.9367
	33.7309	0.005	22.1278	33.6852	44.1413	0.0457078	44.187
4	18.7251	0.05	22.0034	18.5551	59.2714	0.170008	59.4415
	29.9618	0.01	22.1122	29.9006	47.926	0.0612195	47.9872
	36.9474	0.005	22.1426	36.9166	40.9099	0.0308166	40.9408
5	18.466	0.05	21.9962	18.2887	59.5378	0.177247	59.7151
	28.6851	0.01	22.1029	28.6145	49.212	0.0705265	49.2825
	33.8666	0.005	22.1257	33.8188	44.0077	0.047776	44.0555
6	22.2659	0.05	22.0607	22.1532	55.6734	0.112718	55.7861
	35.9718	0.01	22.1402	35.9385	41.888	0.0332984	41.9213
	43.6998	0.005	22.1559	43.6822	34.1444	0.0175799	34.1619
7	21.9377	0.05	22.0603	21.8245	56.002	0.113132	56.1151
	32.9572	0.01	22.1271	32.9109	44.9157	0.0463283	44.962
	38.3149	0.005	22.141	38.2825	39.5441	0.0324712	39.5766
8	22.4535	0.05	22.0572	22.3373	55.4893	0.116234	55.6055
	34.9907	0.01	22.1296	34.9468	42.8797	0.0438464	42.9236
	40.5741	0.005	22.1464	40.547	37.2796	0.0270938	37.3067
9	23.038	0.05	22.0713	22.9358	54.8907	0.10217	54.9929
	34.9427	0.01	22.1362	34.9055	42.9211	0.0372281	42.9583
	40.5337	0.005	22.1495	40.5097	37.3168	0.0239914	37.3408
10	24.105	0.05	22.0818	24.0134	53.8132	0.0916224	53.9048
	37.0395	0.01	22.1437	37.0097	40.8169	0.0297825	40.8466
	43.0925	0.005	22.1557	43.0747	34.7518	0.0177867	34.7696

Table 18: Table of error rates, false and correct predictions and total savings for thresholds 0.05, 0.01, and 0.005 for positive weights between 1 and 10 plus `None`.

Bibliography

[1] A. D. Angelis and M. Pimenta, *Introduction to particle and astroparticle physics, Multimessenger astronomy and its particle physics foundations*, 2nd ed., Undergraduate Lecture Notes in Physics (Springer Cham, 2018).

[2] *IceCube*, `https://icecube.wisc.edu/science/icecube/`, [Online; accessed 07-February-2025].

[3] R. Abbasi, M. Ackermann, J. Adams, J. A. Aguilar, et al., "Icecube high-energy starting event sample: description and flux characterization with 7.5 years of data", Physical Review D **104**, `10.1103/physrevd.104.022002` (2021).

[4] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent neural networks: a comprehensive review of architectures, variants, and applications", Information **15**, `10.3390/info15090517` (2024).

[5] B. Mayer, *Git repository for toycube simulation and neural network training*, Accessed: February 28, 2025, 2025.

[6] J.-H. Koehne, K. Frantzen, M. Schmitz, T. Fuchs, et al., "Proposal: a tool for propagation of charged leptons", Computer Physics Communications **184**, 2070–2090 (2013).

[7] M. G. Aartsen, R. Abbasi, M. Ackermann, J. Adams, et al., "Energy reconstruction methods in the icecube neutrino telescope", Journal of Instrumentation **9**, P03009–P03009 (2014).

[8] C. Haack, Personal Correspondence, 2025.

Declaration of Originality

I, Benedikt Josef Mayer, student registration number: 22788651, hereby confirm that I completed the submitted work independently and without the unauthorized assistance of third parties and without the use of undisclosed and, in particular, unauthorized aids. This work has not been previously submitted in its current form or in a similar form to any other examination authorities and has not been accepted as part of an examination by any other examination authority.

Where the wording has been taken from other people's work or ideas, this has been properly acknowledged and referenced. This also applies to drawings, sketches, diagrams and sources from the Internet.

In particular, I am aware that the use of artificial intelligence is forbidden unless its use an aid has been expressly permitted by the examiner. This applies in particular to chatbots (especially ChatGPT) and such programs in general that can complete the tasks of the examination or parts thereof on my behalf.

Any infringements of the above rules constitute fraud or attempted fraud and shall lead to the examination being graded "fail" ("nicht bestanden").

Place, Date	Signature