Denoising of Single Antenna RNO-G Neutrino Event Signals

Bastian Setter

February 28, 2023

1 Measurements of neutrinos with radio emission

A promising approach for the measurement of ultra-high-energy neutrinos is the detection via their radio emission when interacting in the atmosphere. By connecting many radio antennas, radio arrays are constructed that are supposed to extend over hundreds of square kilometers to obtain sufficient exposure for successfully detecting these rare cosmic messengers. For accurate measurements, the antennas have to be placed in a radio-quiet environment, like Greenland, where the background noise is small. However, the size of the noise can still significantly disturb the measurement. Therefore, the reduction of this noise in neutrino measurements will be investigated in the following project.

2 Provided data

Before a algorithm to denoise the signals is developed and optimised, the data provided for this project will be analysed and preprocessed. This includes some basic statistical properties, like distribution evaluation and data set construction.

2.1 First look and statistics

The data provided consists of 25600 simulated neutrino signals and realistic background measured at Greenland with a length of 640 ns and 2048 data points. An example of both can be seen in Figure 1.

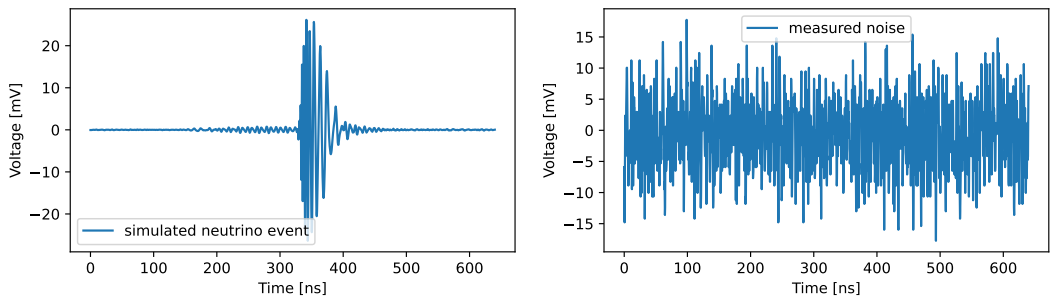

Figure 1: Left: Example of a simulated neutrino event signal. Right: Example of realistic measured noise trace.

2.1.1 Noise

As the noise was actually measured, the analog signal has been digitised at some point. This can be seen in the data, since the unique quantized values of the noise follows $noise_i = i \cdot 0.591\,715\,98\,\text{mV}$ with $i \in Z$. The maximum and minimum are $41.420\,118\,34\,\text{mV}$ and $-39.053\,254\,44\,\text{mV}$ respectively, however not all possible values are populated near the edges of the noise range.
A histogram of noise values is shown in Figure 2. The bins are of width $0.591\,715\,98\,\text{mV}$ containing one unique value each. As can be seen, the noise follows a normal distribution $\frac{a}{\sqrt{2\pi\sigma^2}} \cdot exp\left(-0.5 \cdot \left(\frac{x-\mu}{\sigma}\right)^2\right)$ with the fitted parameters from Table 1. From the fitted mean it is clear, that the noise is not centered around 0 mV.

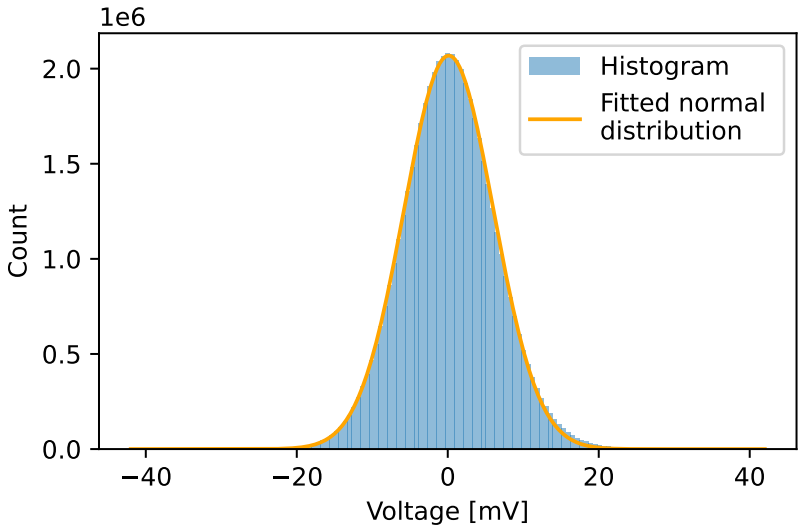

Figure 2: Histogram of all 52428800 value in the noise samples in blue, following a normal distribution with fit parameters in Table 1.

Parameter	Value
μ	0.126(11) mV
σ	5.940(11) mV
a	3080(5)

Table 1: Fit parameters for the fitted normal distribution $\frac{a}{\sqrt{2\pi\sigma^2}} \cdot \ exp\left(-0.5 \cdot \left(\frac{x-\mu}{\sigma}\right)^2\right)$ in Figure 2. μ and σ are the mean an variance as for a standard normal distribution. a is a scaling factor to account for the non-normalized histogram.

Fitting a normal distribution to the histogram of noise at every point in time separately and plotting the mean and 1-σ of the fitted distributions gives Figure 3. Especially from the 1-σ values, but also from the means we can observe a repeating shape with a frequency of about 25 MHz. This is probably also caused by the electronics of the measurement station.

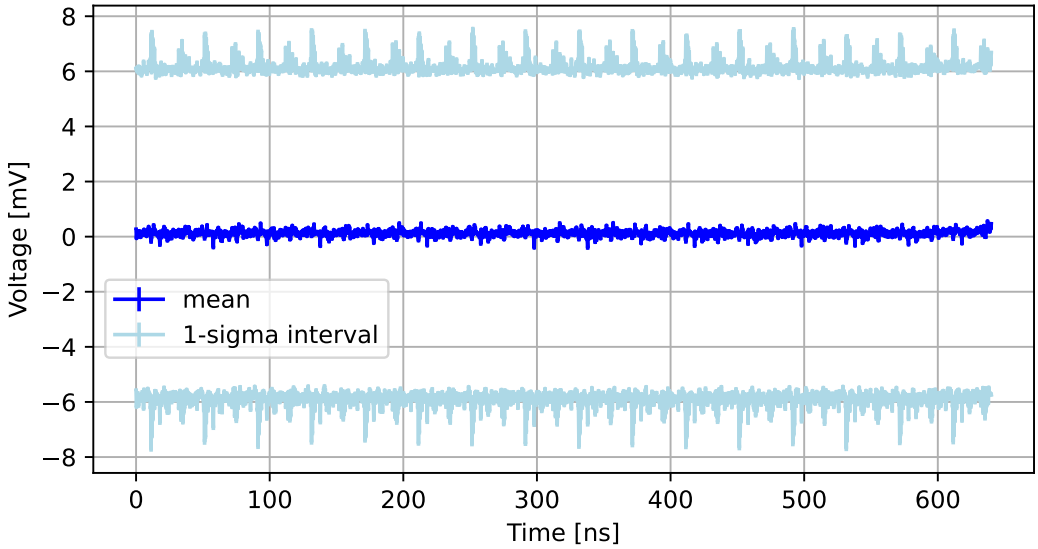

Figure 3: Mean and 1-σ interval of a normal distribution fitted to a histogram for every point in time.

Calculating the mean and standard deviation of noise traces, instead of points in time, reveals a similar picture. Histograms of the mean and standard deviation of every trace are shown in Figure 4. Here the means are shifted as well, with the mean of means at 0.25 and a standard deviation of 1.3.

2.1.2 Signal

In contrast to the noise, the simulated neutrino events are not quantized. Even more so, under all 52428800 values no duplicates are present. This however is most likely, since the values have a precision of 16 digits. Still a common value would be okay, if the simulation had a fixed starting point such as 0, which is apparently not the case. How the difference in quantization between noise and signal would become relevant is outlined in subsubsection 2.2.1.
Due to the large variations in signal strength, a largely distribution based analysis as for the noise is not feasible. Instead the events will be characterised by their amplitude, position and such. In order to extract those features, the peaks of any given signal have to be determined. To do this,

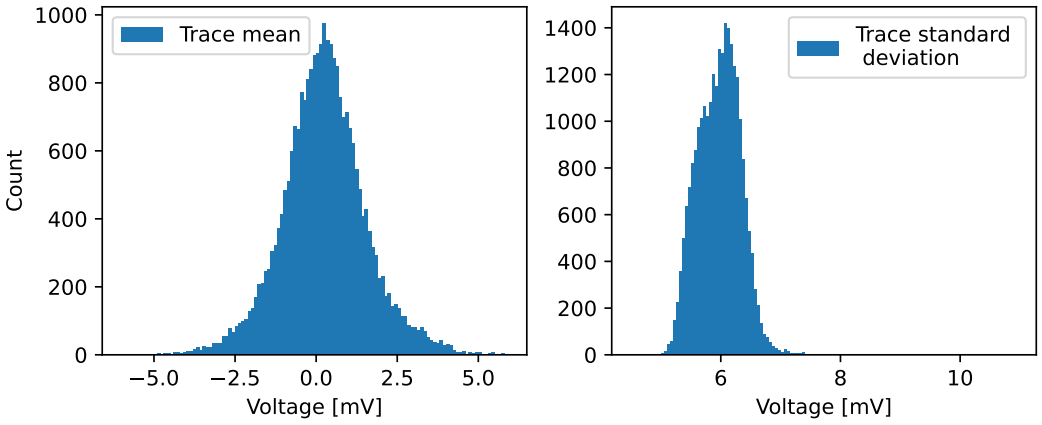

Figure 4: Histograms of mean (left) and standard deviation (right) of every noise trace.

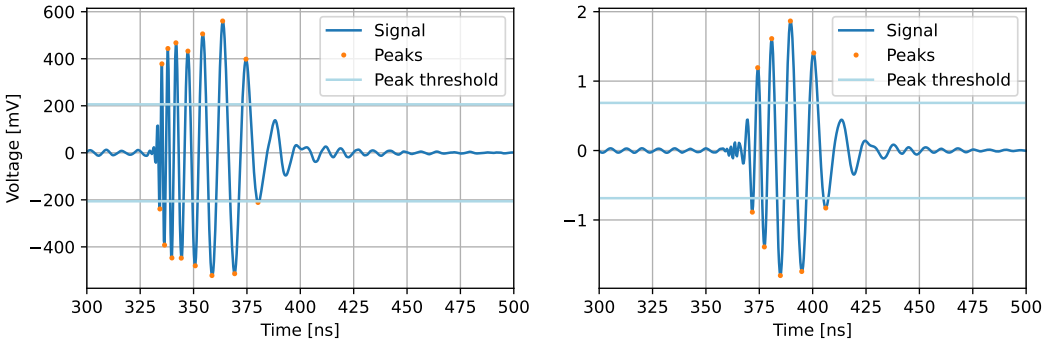

Figure 5: Example signals with high (left) and low (right) amplitude of the peak finding algorithm used. Blue shows the true signal, orange the found peaks, light blue the threshold for peaks to be accepted.

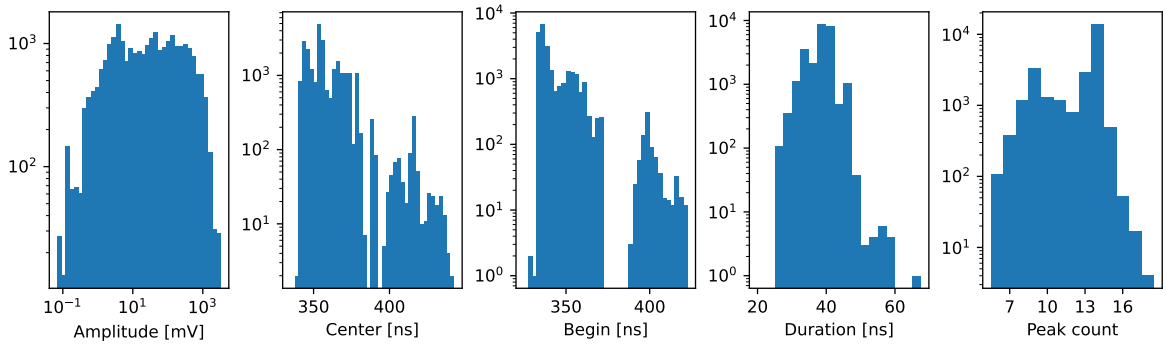

Figure 6: Histograms of the 5 discussed metrics for all signals. For the amplitude a logarithmic x-scale and binning is used. The orange line shows the 1-σ interval of the noise.

`scipy.find_peaks()` with a threshold of $\frac{peakmax}{e}$ was used. Examples can be seen in Figure 5. Since the neutrino events are not quantized, small signals do not perform worse.

The metrics chosen to analysis the signals are:

- amplitude - the maximum voltage found in the whole array
- center - the position of the largest voltage point
- begin - the time of the first peak
- duration - the difference in time of the first and last peak
- peak count - the number of peaks found in the signal

A histogram of the just mentioned criteria is displayed in Figure 6. All metrics show a highly concentrated distribution. This is caused by the high degree of similarity in the generated signals. The amplitude distribution varies more than the other metrics in a range from 0.07 mV to 3235 mV. 31% of the signals have an amplitude below 5.94 mV, the 1-σ interval of the noise. This means that a significant amount of the provided signals will be lost in the variance of the noise. The begin is separated in two distinct blobs around 350 ns and 400 ns. This separation can also be seen in the center position, but more fanned out, due to the differing duration of the signals.

2.2 Preprocessing

This section will deal with the way of construction the different needed data set. First how a single pair of input and output is constructed and second the split of data into train, test and validation data sets.

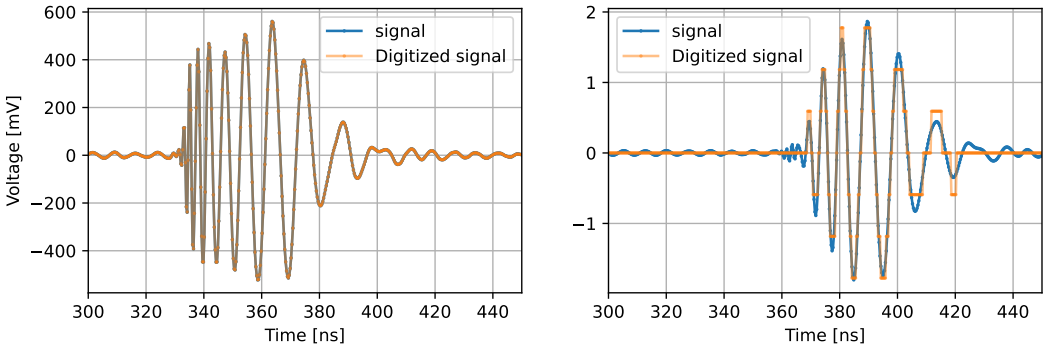

Figure 7: Two examples for the quantization applied to the signals. Blue shows the true signal, orange the quantized signal.

2.2.1 Single samples

As stated in the beginning, the noise and signals are provided separately. To construct the input signals for the coming algorithms, a simulated neutrino event can simply be added to any given noise trace. The output is then the neutrino event used in the construction of the input signal.
When assuming that a provided input signal is always the composition of a neutrino event and signal, this should not be a problem. This however is not the case in actual measured data, where long stretches of pure noise are measured. Therefore any signal algorithm would need to be able to distinguish pure noise from a signal that needs to be reconstructed. When using the simple input construction of adding noise and simulation, an algorithm could just memorise the quantisation of the noise and reconstruct a signal only, if the input does contain values not memorised. This would than reduce the effectiveness of the algorithm, especially for signals with low amplitudes, since the algorithm would not have to find a clever way to decide whether a small signal is present or not. For high amplitudes it should not matter, since there it would be obvious due to the sheer size of the values in the input data.
In practise the algorithm to detect a signal and the one to denoise the signal are two separate algorithms. Therefore we could assume for the denoising algorithm, that most input signals to be denoised will contain a signal. Still it is best to use data of the same type in training as in the actual application. By quantizing the signals 324 neutrino events will become 0, since they are below 0.295 857 99 mV in amplitude and therefore half of a noise step. Examples of quantized signals can be seen in Figure 7. Due to the high sample rate also relatively small events are approximated very well and for larger ones almost no difference can be noticed.

2.2.2 Data set creation

First the provided signals are quantized and then together with noise shuffled to ensure randomness in the data. Afterwards the data is split into train, test and validation set. The train set is used to adjust parameters of the algorithm, the validation set is used to estimated the performance of the algorithm during the training and the test data set is used in the final performance analysis of the trained algorithm.
To increase the number of samples, any given signal is applied to multiple noise traces within their given set, until the desired amount of samples is achieved. For this application the training data set contains 70000, the validation set 8000 and the test data set 12000 samples. By separating the sets beforehand, cross-contamination and risk of not detecting overfitting is reduced.

3 Trained networks

The task of denoising input signals seems like a good application for autoencoder deep neural networks. Since the important features for any given signal hidden under the noisy inputs have to be extracted by the encoder and reduced to a few nodes, used for reconstruction of the pure neutrino event in the encoder afterwards. Useful layers for the network to be designed might include dense and convolutional layers. More advanced structures, such as graph neural networks, transformers and many others, are

Layer (type)	Output Shape	Parameters #
dense_1 (Dense)	(None, 512)	1049088
activation_1 (Sigmoid)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
activation_2 (Sigmoid)	(None, 128)	0
dense_3 (Dense)	(None, 512)	66048
activation_3 (Sigmoid)	(None, 512)	0
dense_4 (Dense)	(None, 2048)	1050624

Table 2: Architecture of the starting model. The model has 2231424 total parameters, all of which are trainable.

not necessary in this application, since they are designed to address the issue with unevenly spaced data as is the case in the Km3net telescopes or variable input sizes for text and language applications. Factors to be optimized include activation functions, learning rates, loss, size of latent space and kernel size of convolutional layers.

3.1 First Network

As a starting point a basic autoencoder with dense layers and activation function "Sigmoid" was used. For exact specifications see Table 2. The Optimizer used is "Adam" and the loss "Mean squared error". Both are defaults in reconstruction tasks and therefore used as standard options.

The training of the model is plotted in Figure 8, it is plotted in logarithmic fashion, to view the development in the more optimized regions of the training in more detail. The orange and blue curve show the mean squared error of the train and validation set for every epoch. A smoothed version of both curves is plotted in red and green respectively. See the attached script for details on the smoothing applied. The horizontal black line shows the best epoch in the training. It is determined by the minimum of the purple curve, which is the smoothed validation curve, but penalises epochs trained by a small factor. This ensures, that if no significant progress is being made for the validation set, no unnecessary overfitting is allowed. If the determined best epoch lies within the first 90% of the total amount of epochs trained, meaning overfitting could have taken place, a second network is trained to the optimal number of epochs. If the best epoch lies within the last 10% the current network is used. To reduced the variance of a single trained model, 5 more epochs are trained with the latest model and the model with lowest mean squared error is saved, for further analysis.

Still the model has learned already quite much, as can be seen in Figure 9 showing two examples of reconstructed signals. The general shape of the signals is kept in both images, as is the position and length of the reconstructed signal, with a slightly worse performance in the small signal. Only the amplitude is significantly diverging from the original signal. This observation is true for both examples, independent of the fact the one is clearly visible above the noise and the other about 1/3 of the 1σ interval of the noise.

A more thorough investigation of the trained model can be seen in Figure 10. It shows a correlation plot and histogram of all 5 metrics described in subsubsection 2.1.2. The plotted data is split into two sets, one containing signals with an amplitude above, the other with amplitudes below 17.82 mV. This value has been chosen, since it is the 3-σ value of noise distribution and therefore selecting signals clearly visible over the noise. In the correlation plots the blue dots show all signals and green and orange only the selected data set. This gives a better understanding, of how the particular set performs in comparison to the whole data. In the histograms the true and reconstructed distributions are overlaid, but in contrast to the correlation plots not the distributions of the whole data set, as it would not add much information.

For the amplitude a clear and wide correlation of reconstructed and true signals can be seen. For the high amplitude signals a underestimation can be observed. This is most prominent in signals with amplitudes up to 100 mV. For the signals with low amplitude a overestimation can be observed, focused at a peak of 3 mV. This is also the low end of the underestimation in the high amplitude data.

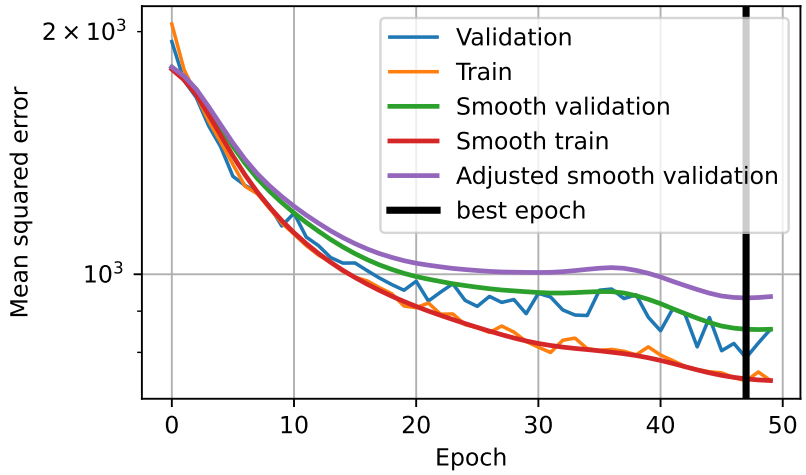

Figure 8: Training history of the first model (Table 2). Orange and red show the original and smoothed mean squared error for the test data set in every epoch. Blue and green show the original and smoothed mean squared error for the validation data set at the end of every epoch. Purple is the smoothed validation curve, but adjusted by 10% over the whole training. The vertical black bar indicates the best epoch for the training. It is placed at the minimum of the purple curve.

For the time related metrics - center, begin and duration - a good agreement in the distributions of high amplitude data can be observed. For the low amplitude signals generally the same range of values is found in the reconstructed signals. Still for the begin metric a complete focus of the most prominent peak in the true signals is found and the duration metric shows much larger values and no resemblance of the peaks in the true signals is reproduced.

In the correlation of the center metric it is apparent, that the signals follow some sort of underlying system, since only very distinct values are present in the true and reconstructed data. This is apparent for the high amplitude data in lines with slope 1 and for the low amplitude data in vertical and horizontal lines. The diagonal lines indicated mostly correct reconstruction mistaking the correct value only by a few discrete steps. The complete grid in the low amplitude data is produced due to the discretized centers in true and reconstructed signals, but as no correlation between low and high values is established, no diagonal lines emerge in a prominent fashion. The quantized values can also be seen clearly in the true signal duration.
For the peak amount metric no meaningful correlation can be observed, however in the high amplitude data the outer points of the correlation plot are not populated indicating a slightly better reconstruction for high amplitude data. The histograms do not show a better reconstruction of the high amplitude data.
In addition to the graphical analysis, 2 numerical estimators are calculated for every training's run. To ensure comparability the metric computed will always be mean squared error, independent of used loss. The first estimator evaluates the test data set on the finished model. The second estimator is the "middle mean" of the best epoch in the validation set of the first training (see Figure 8). To calculate the middle mean, the lowest and highest error in any given list are dropped and the remaining values averaged.

3.2 Network optimizations

In this chapter the performance of the model will be shown and optimized for different factors, such as hyper parameters, model architectures, compile options and such. To ensure an optimally trained network the number of epochs is increased to 500.

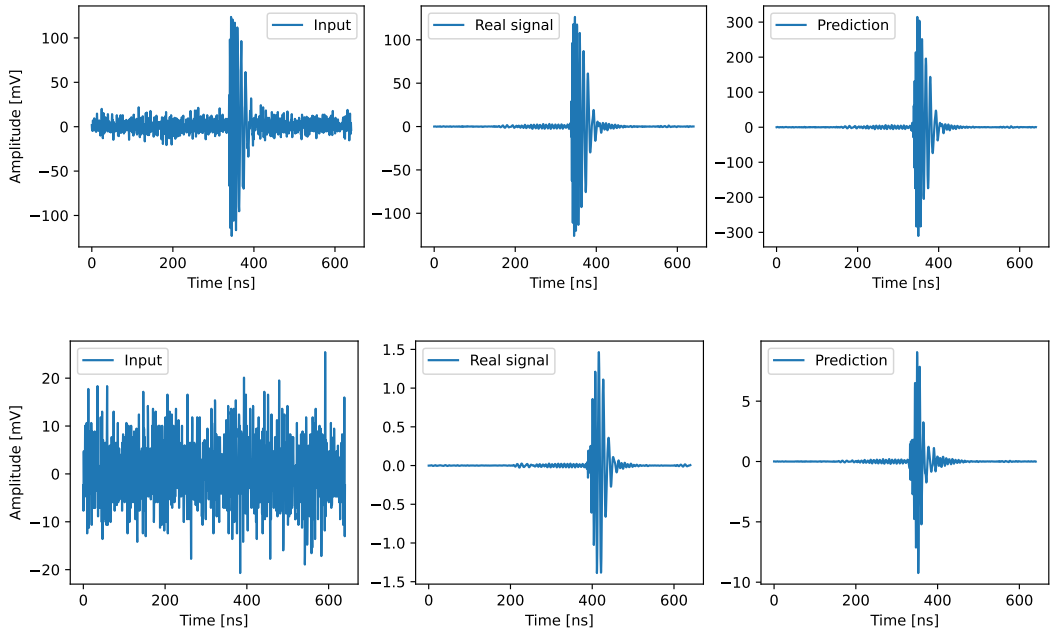

Figure 9: Example of two reconstructed signals. The top row shows a signal with high amplitude and the bottom row a signal with low amplitude, vanishing in the noise. From left to right the columns show the input signal with noise, the original signal and the reconstructed signal.

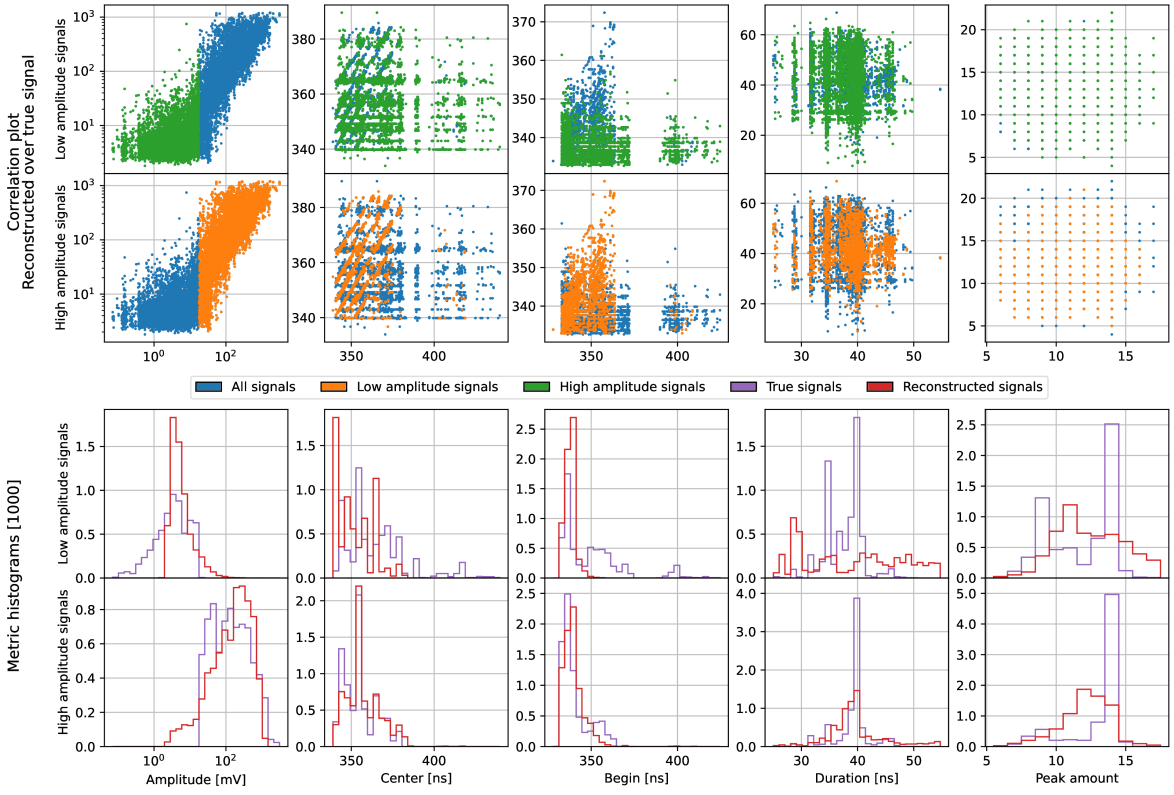

Figure 10: Correlation plots (top) and histogram for true and reconstructed signal metrics for the first trained network (Table 2). The metrics calculated are amplitude, center, begin, duration and peak amount of signal displayed in one column each, note that the label at the bottom is true for the whole column. The analysis is performed separately for signal with low and high amplitude.

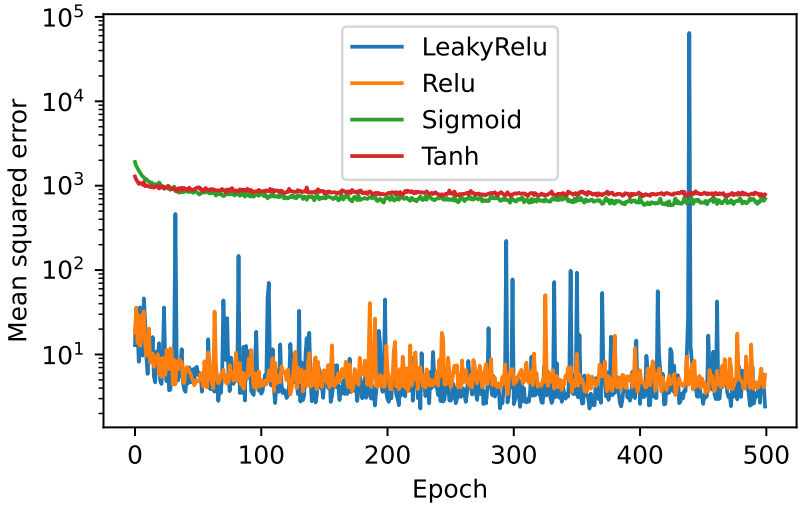

Figure 11: Mean squared error of the validation set during the training with different activation functions applied throughout the whole model (Table 2). The mean squared error is plotted in a logarithmic manner to show to low error rates better. Functions applied are leakyRelu (blue), relu (orange), sigmoid (green) and tanh (red).

3.2.1 Activation function

The first parameter to be changed is the activation function used throughout the whole model. "Sigmoid" has been compared to "tanh", "relu", "leakyRelu" and "exponential". The training history of these is plotted in Figure 11. "Exponential" is missing, since in the training, the loss diverged in the first epoch. The performance of "sigmoid" and "tanh" is clearly worse than "relu" and "leakyRelu". Even though the training of both is very volatile, the lower bound of "relu" is clearly above "leakyRelu". Due to the volatile training both "relu" versions are tested in the next step.

3.2.2 Learning rate

The second parameter to be optimised is the learning rate. As stated in the last paragraph the high default rate causes the training to be volatile indicating a learning rate, which is to high. By reducing the learning rate, the network can adjust its weigh more slowly and is hence not constantly jumping over the minima. In Figure 12 the estimators for several different rates are plotted. A significant improvement of the network can be observed until 3×10^{-5}. The increase at 1×10^{-6} is due to the network not converging in a timely manner. For instance the convergence of 2×10^{-6} has taken about 2500 epochs. For 3×10^{-5} it took only about 600. As for the lack of improvement and a shorter training the learning rate of 3×10^{-5} with "leakyRelu" as activation function will be use in the further optimization.

3.2.3 Loss

The losses to be compared are the "huber", "mean squared error" and "mean squared logarithmic error" loss. The first two losses use the difference of the data points to assign their loss values. The mean squared error resembles a parabola, severely punishing outliers. The huber function is comprised of a parabola near 0 and linear slope elsewhere, resulting in less punished outliers than the mean squared error. The mean squared logarithmic error is more of a relative error between true and predicted value. This results in severely punishing underestimation and allows for overestimation. In regression tasks with a lower bound, such as energy regression, this is often the best loss to use.

The comparison of different losses is difficult to accomplish with numerical estimators, since the estimators themselves are based on the mean squared error. Still they are shown in Figure 13. In addition to the estimators, the amount of overfit, meaning the ratio of validation set to test set at the optimal epoch, is given. It is clear, that the best choices are huber and mean squared error, both

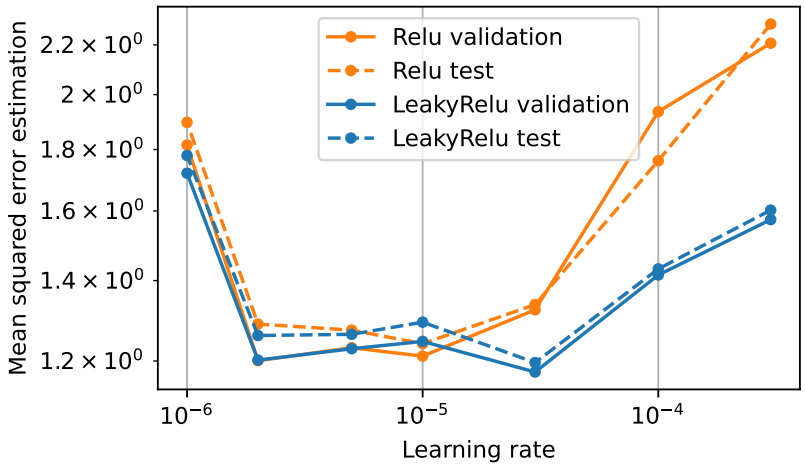

Figure 12: Validation (solid) and test (dashed) estimator for different learning rates and the activation functions relu (orange) and leakyRelu (blue).

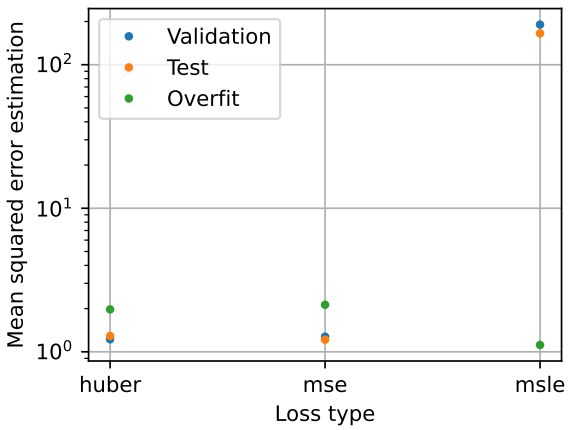

Figure 13: Validation and test estimators, as well as overfitting for 3 different losses. The losses are huber, mean squared error (mse) and mean squared logarithmic error (msle).

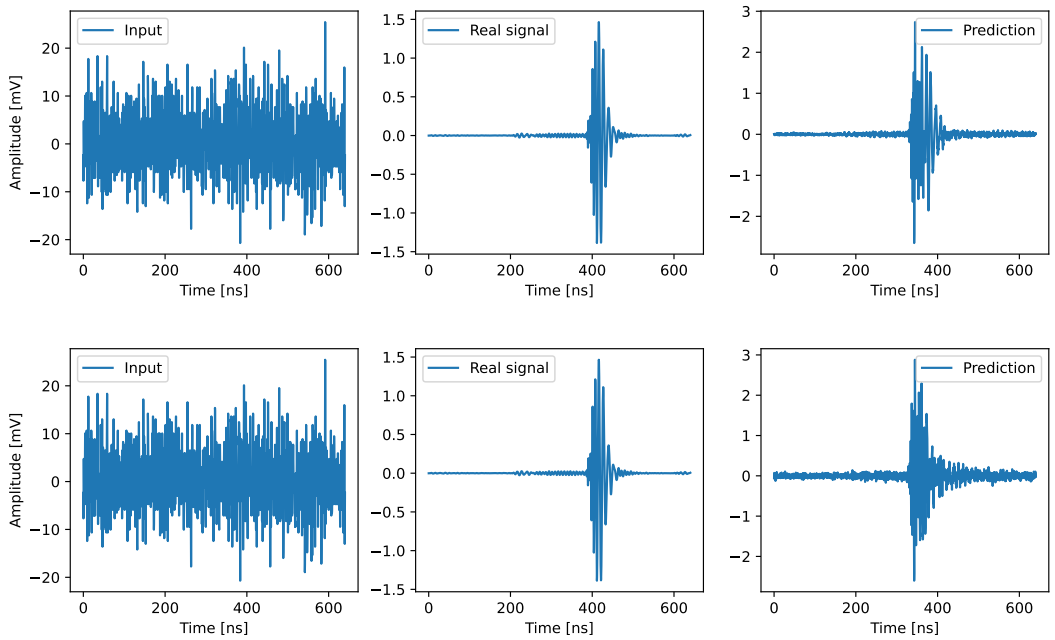

Figure 14: The same signal reconstructed with the huber loss (top row) and mean squared error (bottom row). From left to right the columns show the input signal with noise, the original signal and the reconstructed signal.

show a far superior error, compared to the mean squared logarithmic error. The significantly lower overfitting is caused by the general bad performance of the network. The huber loss is also showing slightly less overfitting, than the mean squared error. A visual comparison of two reconstructed signals is shown in Figure 14. Here the denoising of the huber network is clearly superior. Therefor the huber loss will be used in the further optimization.

3.2.4 Size of latent space

Next the size of latent space of the autoencoder will be varied. By reducing the number of neurons in the smallest layer, the network can remember less details and has to rely more on the general shape of the neutrino events. On the other hand, allowing the network to remember more details can promote overfitting by just memorising every input output pair. The size of latent space of the network is varied in a range of 4 to 512. Also the amount of reduction (skip) is tested, meaning by what factor the amount of neurons is de-/increase from layer to layer.
The results are shown in Figure 15. As expected, the error is decreasing and overfitting is increasing with size of latent space. In generel a skip of 4 produces better and more consistent results. The improvement of error at the lower end of latent space with skip 2 is slower than with a skip of 4. It takes until a latent space of 256 to achieve the performance skip 4 at a latent space size 48, while introducing considerably more overfit. Therefor a skip of 4 and latent space size of 48 is considered the best overall solution.

3.2.5 Convolutional layer

At last the effects of convolutional layers of different kernel sizes in the beginning of the autoencoder are being tested. The configurations include a single layer with a kernel of size 8 to 256. The two layer setup consists of the base layer as in the one layer setup and an additional upstream layer with kernel size between 8 and 128. The measured estimators for every configuration are shown in Figure 16.

For most configurations a clear decrease in performance can be observed in comparison to no convolutional layer. In the 2 layer setups, the decrease is correlated with kernel size independent of the layer. The one layer setup on the other hand does increase in performance with larger kernel size, so much so that a single kernel of size 256 has to be considered the best network layout.

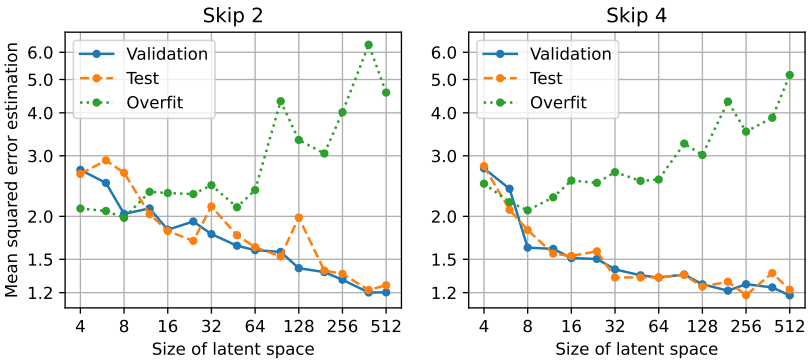

Figure 15: Validation (solid blue) and test (dashed orange) estimators, as well as overfit (dotted green) over different latent space sizes for two skips. Skip gives the factor, by which two number of neurons de-/increases from layer to layer. Left show a skip of 2 and right a skip of 4. The x and y axis are plotted in a logarithmically.

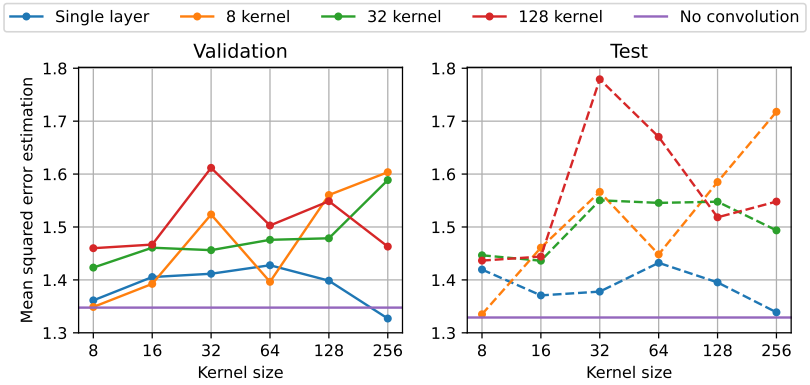

Figure 16: Validation (left, solid) and test (right, dashed) estimators of network layouts with convolutional layers of varying kernel size. Purple gives the baseline for a no convolution setup. The kernel size is plotted logarithmically. Blue shows the estimator for a single layer setup. The two layer setup consists of one layer as in the one layer setup and a second upstream layer of kernel size 8 (orange), 32 (green) or 128 (red). The best setup is a single layer with kernel size of 256.

3.3 Final Network

After the optimizations, the layout of the final model is shown in Table 3. It consists of a dense autoencoder with latent space size of 48 following a single convolutional layer with kernel size 256. The optimizer is Adam with a learning rate of 3×10^{-5} and the loss is Huber.

The graphical analysis is given in Figure 17. Comparing it to the analysis of the starting network in Figure 10, vast improvements can be observed. The distributions of reconstructed high amplitude data match the true values almost perfectly. Also the low amplitude signal match their true shapes significantly better. The width of the amplitude correlation plot has shrunken for low and high amplitude data. High amplitude data becoming a single line. Low amplitude signals no longer overestimating and reconstructing signals with generally lower amplitude. The center reconstruction for high amplitude signals is now misconstructed by only a single quantization step, down from 2 or 3 by the original model. For low amplitude centers, the model is no longer showing horizontal lines, meaning the quantization of the center has no longer been learned. Independent of amplitude still no signal is being reconstructed with a center of around 400 ns. The vanishing of the horizontal lines in the low amplitude correlation plot can be observed for the begin and duration metric as well. For both metrics the high amplitude correlation has developed from a mostly uncorrelated point cloud to a contained line of origin, showing very good reconstruction. In the low amplitude begin distribution

Layer (type)	Output Shape	Parameters #
conv1d (Conv1D)	(None, 2048, 1)	257
leaky_re_lu (LeakyReLU)	(None, 2048, 1)	0
reshape (Reshape)	(None, 2048)	0
dense (Dense)	(None, 2048)	4196352
leaky_re_lu_1 (LeakyReLU)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
leaky_re_lu_2 (LeakyReLU)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
leaky_re_lu_3 (LeakyReLU)	(None, 128)	0
dense_3 (Dense)	(None, 48)	6192
leaky_re_lu_4 (LeakyReLU)	(None, 48)	0
dense_4 (Dense)	(None, 128)	6272
leaky_re_lu_5 (LeakyReLU)	(None, 128)	0
dense_5 (Dense)	(None, 512)	66048
leaky_re_lu_6 (LeakyReLU)	(None, 512)	0
dense_6 (Dense)	(None, 2048)	1050624
leaky_re_lu_7 (LeakyReLU)	(None, 2048)	0
dense_7 (Dense)	(None, 2048)	4196352

Table 3: Layout of the final model. It has 10636849 total parameters, all of which are trainable. The optimizer is Adam with a learning rate of 3×10^{-5} and the loss is Huber.

higher reconstructed values can be seen, however similar to the center metric, no reconstruction of high values is taken place. The most increase in performance has been made by the low amplitude duration metric. Coming from an almost uniform distribution with slight focus at a wrong value, the improved model now follows the trend shown in the true distribution, focusing on the strongest true peak at 40 ns. Still a second strong peak at 34 ns is completely missed. Also a substantial amount of signals are still reconstructed overestimated. The development of the low amplitude peak amount metric is similar to the low amplitude duration metric, showing a good but still lacking development.

4 Conclusion

In this report the denoising of single antenna neutrino events was discussed. The provided samples of realistic noise and simulated neutrino events have been analysed. It was found, that the noise still contains fragments of the electronic setup at the measuring site. The signals have been found to be very similar to each other and been characterised with the five metrics amplitude, center, begin, duration and peak amount.

During the construction of the train, validation and test sets the simulations have been quantized to match the noise, so the input resembles possible actual measurements. The number of input samples has been increased by using noise and signals in different combinations, while keeping the different data sets separate.

A autoencoder was chosen to act as the denoising algorithm. A first standard model had already produced good results in reconstruction denoised signals, keeping the shape and general position of signals right. To optimize the model the activation function, learning rate, loss, size of latent space and convolutional kernel size have been varied. The final optimized model reproduces signals with high amplitude almost perfectly and also gives good results for signals with low amplitudes, partly vanishing in the noise.

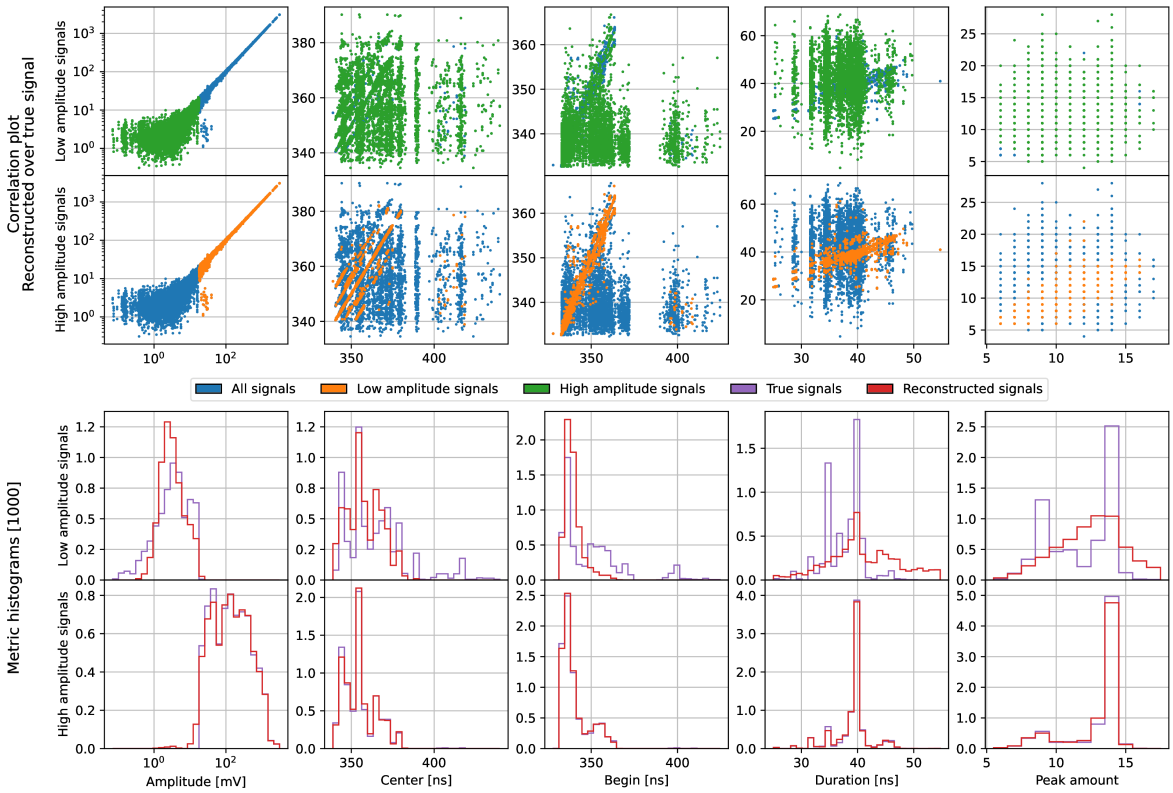

Figure 17: Correlation plots (top) and histogram (bottom) for true and reconstructed signal metrics for the final trained network (Table 3). The metrics calculated are amplitude, center, begin, duration and peak amount of signal. The analysis is performed separately for low and high amplitude signals.