[to do: Abstract]

1 Neural network triggers

1.1 Motivation

The station-level triggers in the previous chapter have been shown to perform well for the science case of the Pierre Auger Observatory. However, it has also been concluded that a lot of potential data, especially at low energies is ignored. This is by intention in order to keep DAQ readout at feasible levels.

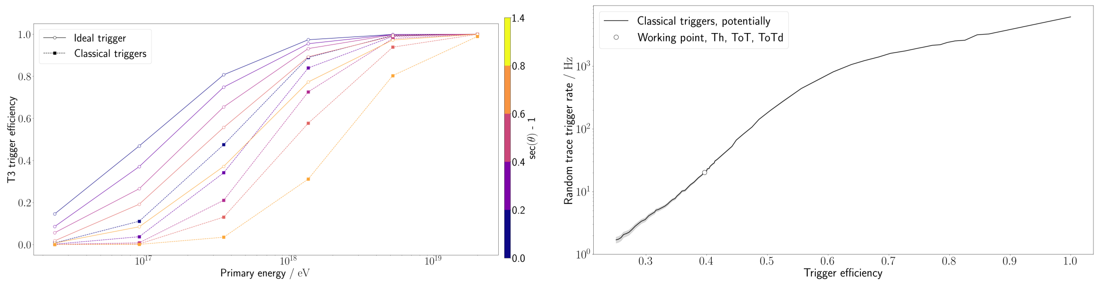
Attempts at improving the overal efficiency of the SD triggers can be made. This is only possible to a certain level. At lowest energies the particle cascade is not big enough to warrant coincident triggers in at least 3 WCD stations. As per **??**, the lateral trigger probability a given classification algorithm can maximally achieve is given by the LPP (c.f. **??**). The T3 detection probability of such an ideal trigger, and consequently the maximal efficiency for an array with 1.5 km spacing is compared to the efficiency of classical triggers in **??**.

Of course, efficiency can be improved simply by adjusting trigger thresholds of the algorithms in **??**. However, the more lenient these thresholds are, the more background events will be detected. This quickly results in trigger rates that are unmanagable for the infrastructure at the Pierre Auger observatory. The probability with which time traces correctly raise a T2 is shown alongside the resulting random-trace trigger rate for different thresholds of classical algorithms in **??**.

Ideally, neural network architectures developed in this chapter should undercut the random-trace trigger rate of classical triggers, while retaining an overall higher accuracy. That is, they lay below and right of the operating point in **??**. For any algorithm that achieves this, the corresponding LTP will be greater than that of classical triggers, resulting in higher event detection efficiency, while not exceeding the bandwidth limitations of the underlaying hardware.

1.2 Implementation

The python library TensorFlow [**tensorflow2015-whitepaper**] is used as a backend to implement the individual classifiers. All discussed architectures are built and trained with the release version 2.8.4 [**tensorflowversion**]. Adjustments to the trainable parameters are calculated according to a momentum-based stochastic gradient descent (Adam [**kingma2014adam**]) on a batch level. In this context, a single batch is made up of all traces that are recorded from a single air shower event. Since batch size grows quickly with increasing energy, a generative approach, where traces are created (c.f.



(a) T3 efficiency **(b) Classical trigger potential**

Figure 1.1: (a) Comparison of an ideal trigger sensitive to any shower signal from primary energies $E \geq 10\,\text{PeV}$ to classical triggers. **(b)** The noise level over calculated efficiency for classical triggers. The tail ends of the potential curve are calculated by adjusting the trigger thresholds from +250% to −95% of the nominal values.

??) at runtime upon requirement, is used in building training data in order to make the process as RAM-efficient as possible. This has important implications. As trace building relies heavily on randomization, the actual training data will not be the same if the random number generators are not seeded beforehand. This has been taken into account. All networks are - unless specifically stated otherwise - trained and validated using the same signal input data.

1.3 Design consideration

1.3.1 Architecture

The hardware specifications at the FPGA level, where trigger conditions are currently checked, are limited. For this reason, NN architectures should be kept as simple as possible. Most importantly, the number of weights, biases and other trainable parameters will need to be hardcoded into station software. Because of minimal available storage space, this number needs to be kept low.

This immediately disqualifies powerful candidates like autoencoders or transformers (compare **??**) from consideration. Only relatively simple dense-, convolutional-, or recurrent neural network architectures are viable contenders and might be implementable in the SD electronics.

1.3.2 Choice of input data

As stated in the previous chapters, neural networks learn by example. It is imperative that the input data propagating through the network during training resembles later use-case inputs as much as possible. However, some choices in how data is represented can and need to be made in order to benefit loss minimization.

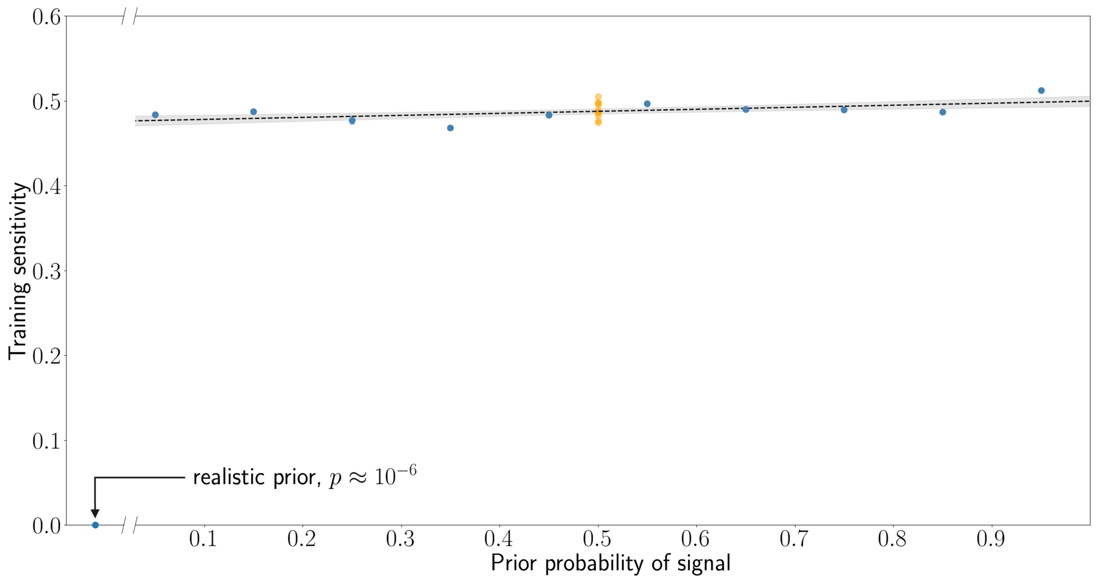


Figure 1.2: A very faint positive slope ($m = 0.02 \pm 0.01$) is observed when relating prior probability to trigger sensitivity (blue dots) of an example convolutional neural network. This could however be attributed to statistical fluctuations in the training fit. An ensemble of networks with the same architecture trained on the same data and with a single prior shows a comparable spread (orange dots).

Prior probability

The flux of cosmic rays with energies exceeding the proton knee is tiny ($\mathcal{O}(1\,\mathrm{m}^{-1}\,\mathrm{yr}^{-1})$ [**dembinski2017data**]). While the size of the SD guarantees decent exposure over the entire array, an individual station will mostly measure background. In fact, the prior probability p of encountering events beyond the kee in a random time trace is roughly one in one million. Of course, if this were reflected in the training dataset, neural networks would show very poor performance. On the notion of a broken clock being correct twice a day, a naive classifier would naively label every input as background and retain almost perfect accuracy. Such behaviour is not desired. The prior probability must be artificially inflated. The influence of prior probability on subsequent trigger sensitivity is shown in **??**. No strong correlation between prior and network performance is found in the range $0.05 \leq p \leq 0.95$. As long as the fraction of signal over entire training set is statistically relevant, the network learns to discern between signal and background events. For all further analysis a conservative prior of $p = 0.5$ is picked.

Filtering & Downsampling

Classical triggers are run in compatibility mode, where 3 UUB bins (8.3 ns) are filtered and downsampled to resemble a single UB bin (25 ns). Some information about the trace shape is lost in the process. Logically, a neural network should be able to identify original UUB signal traces with more ease than their filtered and downsampled counterparts. Ideal triggers should therefore have a higher efficiency when tested on full bandwidth traces

This does not render compatibility mode completely uninteresting for analysis. An effective way of minimizing the trainable parameters in a neural network is reducing its' number of input parameters. Filtering and downsampling encodes - albeit imperfectly - the input data and reduces the dimensionality by a factor of three. This is done in such a way that the UB-like trace contains trace information from a time window that is three times as long for a given number of bins.

Consequently, by filtering and downsampling input data it becomes possible to hand the classifiers additional information while keeping the network size at a fixed level. The effects this has on predictive power of neural networks is further discussed in **??**.

1.3.3 Further hyperparameters

The number of ways both input data and the network architecture can be tweaked to minimize loss is extensive. Further optimization methods, like the choice of labelling, are motivated and discussed in the performance reports in **??**.

One promising approach of optimizing trigger performance that was not examined in this thesis is the adjustment of the neural network loss function. Several reasons exist that motivate an analysis in this direction:

- **Classwise weighting:** The classes C_1 and C_2 are seen as equally important when calculating efficiencies by default. As noted in **??** an imbalance in distributions exists. Observing background is much more likely than observing an air shower. The loss function will reflect this to an extent if classification of background is weighted more heavily than classification of signal. Even a more complex loss sensitive to CR flux (compare **??**) or other parameters is possible.
- **Random-trace trigger rate pull term:** For the most part, the analysis presented here will introduce a neural network distinguished by some feature, optimize its performance, and check whether the resulting random-trace trigger rate is in an acceptable range. However, it is possible to go the opposite way and specify a target trigger rate first and then train the network. In this fashion, one could hope that the algorithms optimize efficiency for a given bandwidth by adding a pull term to the loss function. The challenge of this approach is weighting the pull term in a way that it does not dominate the loss function in different fringe cases. An important drawback of this method is that training will be very slow.

1.4 Convolutional neural networks

1.4.1 Architecture

The convolutional neural networks presented in this chapter all have a very similar architecture, which was shown to slightly outperform other candidates during prototyping tests. It is likely that this is due to design ideas that are closely related to the information structure in the time traces:

- A **2D convolutional input layer** (c.f. **??**) pools together the information of the three different WCD PMTs.
- A **1D convolutional layer** resolves temporal correlation in the output of the previous layer
- A

[to do: Architecture plot?]

1.4.2 Performance

- Conv2d layer to pool information from 3 PMTs
- Conv1d layer to pick up on trace shape
- Dense layer to reduce to binary output choice

[to do: plot of architecture?]

1.4.3 Input size

- Larger input size positively influences efficiency, same SNR level though

[to do: signal-to-noise-ratio plot for the same cut values]

1.4.4 Kernel size

- Larger kernel positively influence efficiency, same SNR level though

[to do: signal-to-noise-ratio plot]

1.4.5 Muon cut

- Muons not a viable hyperparameter for network training
- Does not decrease trigger rate by sufficient amount

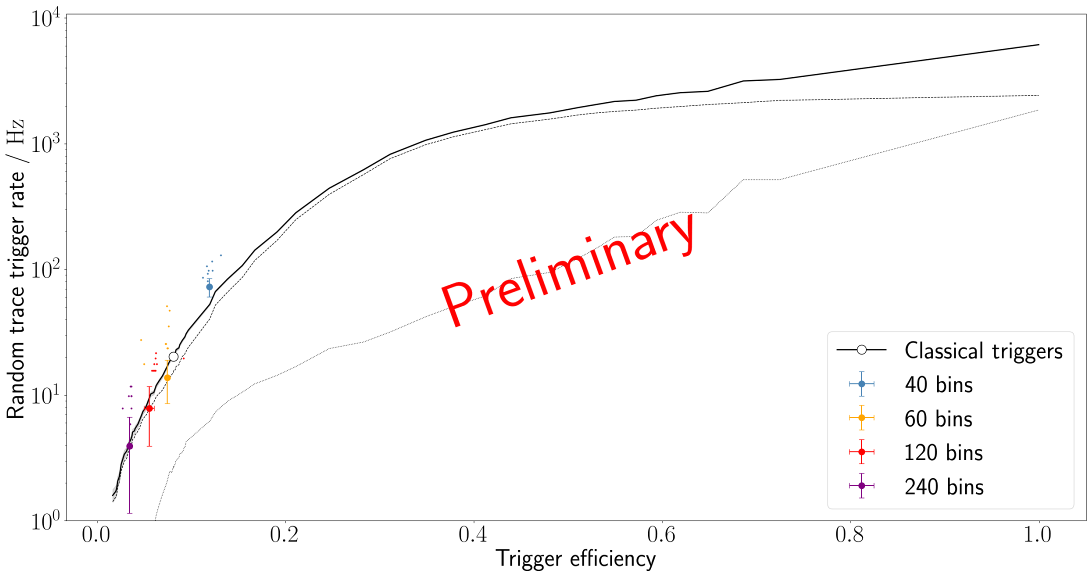


Figure 1.3: Input size vs. observed operation parameters

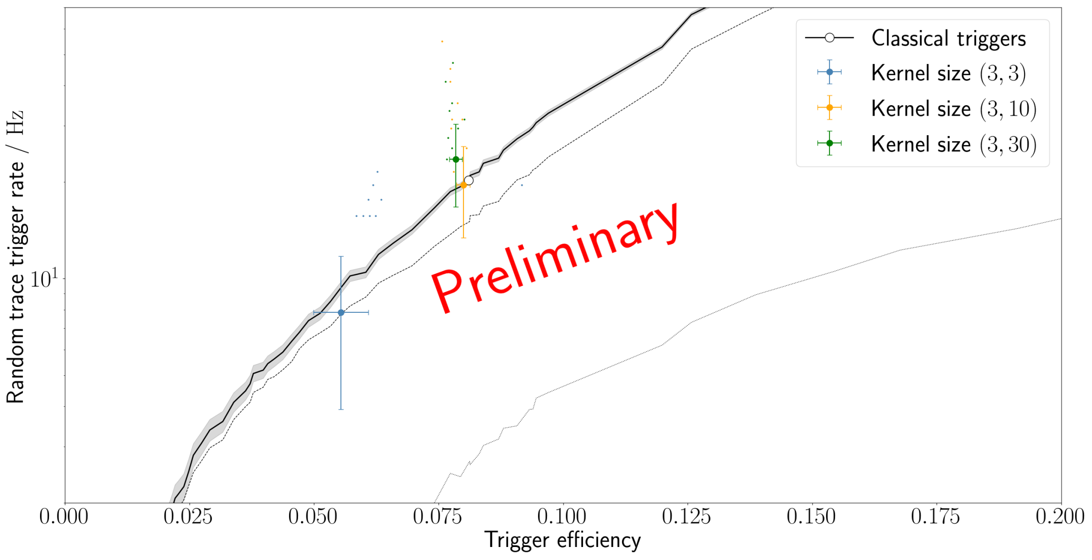


Figure 1.4: Convolutional kernel size vs. observed operation parameters

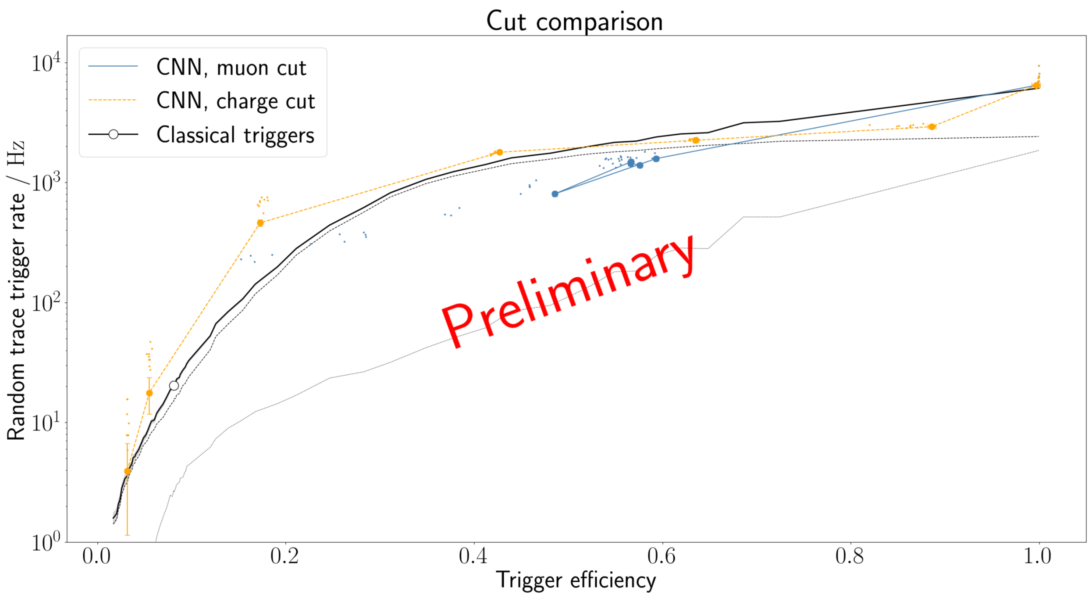


Figure 1.5: comparison between muon cut and charge cut

- Networks don't converge reproducibly

[to do: signal-to-noise-ratio plot]

1.4.6 Charge cut

- Works well with data at hand
- offers finer (infinite) resolution in setting cut threshold

[to do: signal-to-noise-ratio plot] [to do: discuss filtering & downsampling here?] [to do: signal efficiency ratio]

1.5 LSTM

1.5.1 Architecture

- Discuss choice of architecture
- Discuss possibility of using a single layer (instead of 3)

1.5.2 Charge cut

[to do: signal-to-noise-ratio plot]

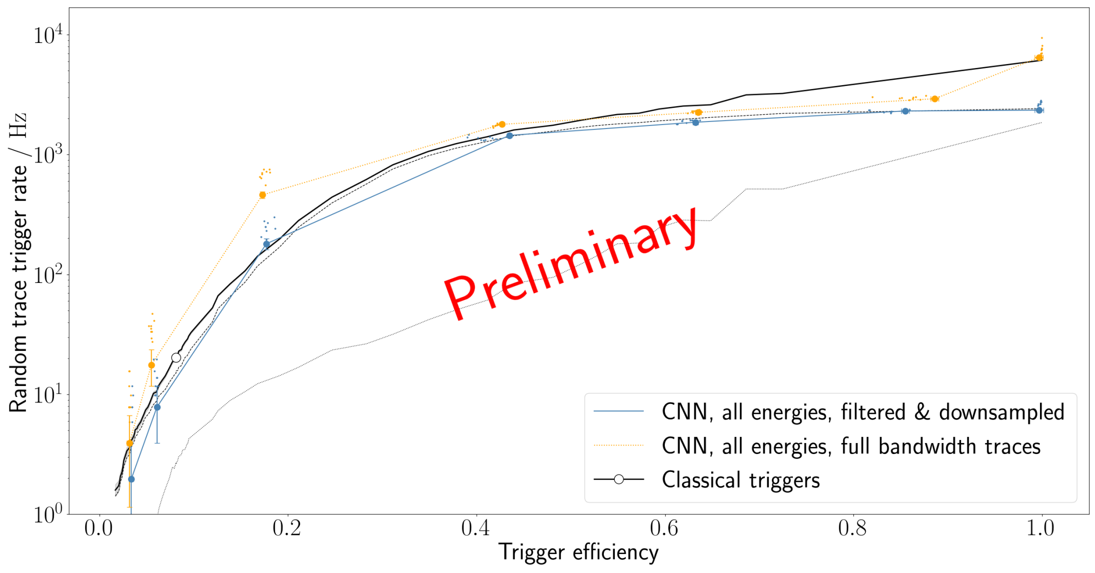


Figure 1.6: charge cut for full bandwidth and filtered & downsampled input data

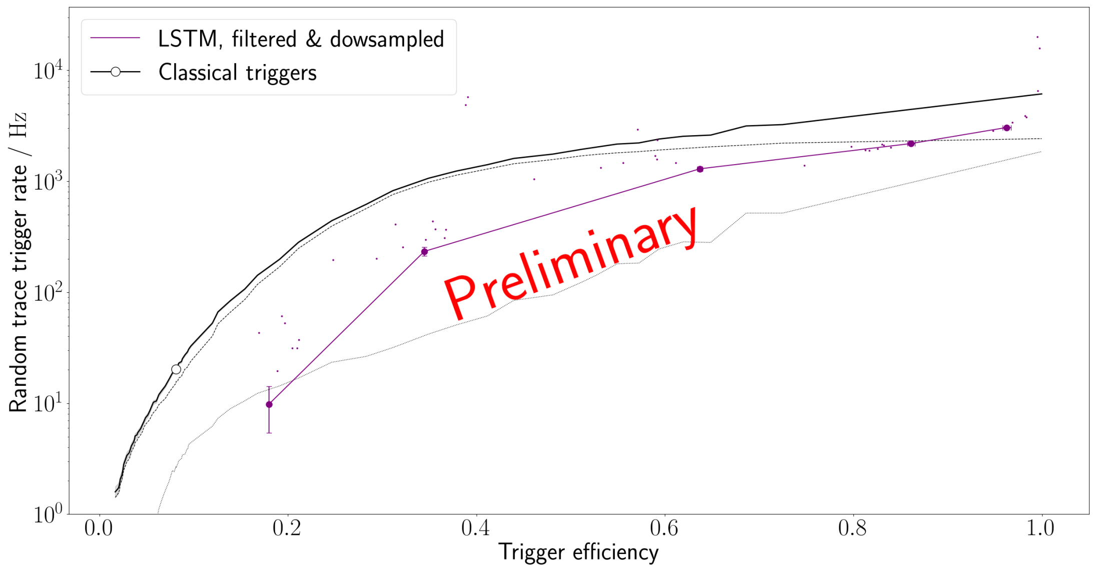


Figure 1.7: performance of LSTM

1.5.3 Choice of labelling

From comments in **??** it is clear that it is not a priori desirable to trigger on any kind of signal induced by extensive air showers. Especially at high separations between shower axis and station location recorded time traces look very similar to background muon events. Triggering on such events will induce an infeasible T2 rate. Certain types of signal traces must therefore be ignored, and labelled as background. Several ways of distinguishing interesting traces from uninteresting ones exist

Muon cut

As discussed in **??** atmospheric muons from very low-energy showers represent the dominant background (that is not just electronic noise) in measured data. The most straight forward approach of lowering the network sensitivity for such signals is requiring muonic coincidences. This means only data from tanks which were hit by at least 2, 3, ..., n_μ muons are assigned a signal label, where n_μ is a tweakable hyperparameter. Traces can be selected based on other particles as well. $\overline{\text{Off}}\underline{\text{line}}$ saves the number of muons, electrons and photons for simulated stations. All three, or even a (weighted) sum are candidate discriminators. In this work, only the muon cut is analyzed with greater detail in **??**, as it is thought to be more closely tied to the problem of distinguishing signal from background.

Charge cut

The integral of a (slice of a) time trace is directly correlated to the energy deposited in the WCD. By requiring the integral to exceed a value of S [to do: continue]