Pileup Detection

The overall goal of this work is to create an algorithm or framework which is able to distinguish between pileup and non-pileup events, while also measuring the time difference between the two pileup events and the ratio of their amplitudes. Additional goals is to create a secondary model/algorithm which is able to “deconvolute” the two pulses in a pileup event.

This desired algorithm will be constructed in the supervised machine learning framework, meaning the neural network will be trained on data with known values for the output.

Dataset

There are two main subsets for the data used in training and testing, the first is real data created by splitting a signal from the dynode of an LYSO based detector and adding a delay cable to one of the signals before connecting them back together. The second data subset comes from taking a single dynode trace and artificially creating pileup. This is done to get a more refined scale for the pileup, instead of the discrete values from the delay pileup. The importance of having two different subsets, both used in training, is so that the model does not specifically find attributes related to the artificial creation of pileup. This could be small kinks at beginning and end, or even differences in the tail.

Data Augmentation

Data augmentation is used to further extend the training set, and further help the model find features unrelated to the position of the features in the trace. To do this, the traces are “cropped” multiple times, shifting the center of the trace throughout. An example of this is removing the first 20 bins of a trace, then using the original trace remove the last 10 bins, and while still using the original trace, remove the first and last 10 bins of the trace. All 3 of these new traces would then be used as inputs instead of the original trace.

Additional changes to the data include adding random gaussian noise to the traces. This can be done similar to the earlier mentioned augmentation, where the same trace but with varying levels of noise can be used as input. Once again, this will increase the training data size, and hopefully help to reduce any over fitting.

Model(s)

The full model will act similar to an auto encoder, where the encoder creates a feature vector, which the decoder then interprets to attempt to replicate the input. In this case, the encoder portion will be composed of multiple different sub models, each with a specific focus, namely detecting the pileup (with a boolean output), detecting the phase of the pileup, and detecting the relative amplitude of the two traces in the pileup. The decoder will then take the feature vector from the output of the encoder and try to create a decomposition of the two pileup signals. This is slightly different to a regular auto encoder, which tries to reproduce the input exactly. The goal of this separation is to use the encoder to independently measure whether there is a pileup event, and return its phase and relative amplitudes. These values will either be a part of the feature vector, or be a single layer away from the feature vector that is passed to the decoder. The separation of the encoder and decoder also allows for the encoder to be trained separately from the decoder, allowing for quicker results. The full output of the model will also append the results from the encoder (pileup, phase, and amplitude) along with the deconvoluted traces.

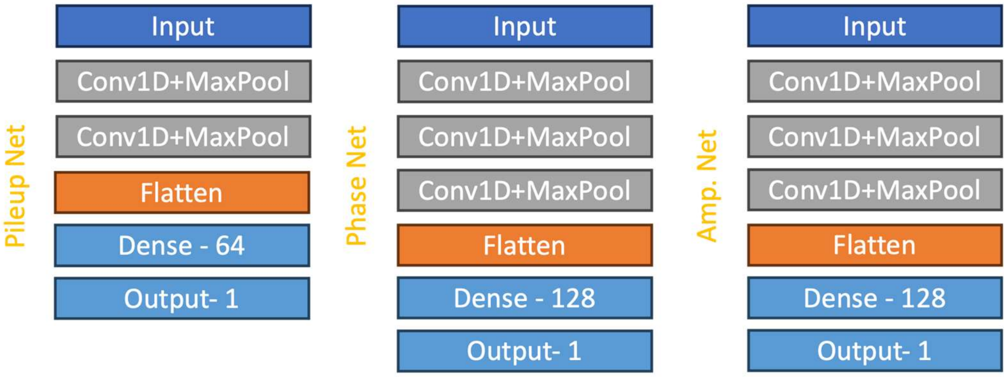

Figure 1. Diagrams for the Pileup Net, Phase Net, and Amp. Net used for measuring the pileup of two signals in a single event. These blocks are implemented into the full model.

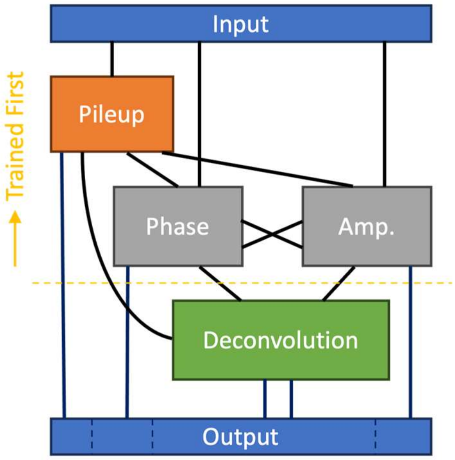

Figure 2. Diagram for the autoencoder net, where the encoder is above the yellow dashed line and the decoder is below the line.

The blocks of neural networks shown in Figure 1 can be used individually, or compiled together to form the encoder. Figure 2 shows the diagram for the complete auto encoder, including where certain blocks are interconnected or connect to the output. Above the yellow line represents the encoder portion of the model, which can be trained separately from the decoder to predict the pileup phase and amplitudes.