Noise Removal and Recovery of Sample Signals

A Comparison of Neural Network Applications

In this report, we will address various methods of noise removal from a given signal on sample data. We are given a specific sinusoidal signal, then we apply various combinations of noise to the true signal. Then, different data augmentation techniques are applied and neural networks attempt to learn the relationship between the applied noise itself and the signal with the applied noise. First, the notation and signals are defined.

$$
\begin{aligned}
s(t) &= \sin(2\pi f_s t) \text{ (true signal)} \\
n_1(t) &= A_{n_1} \sin(2\pi f_{n_1} t + \phi_{n_1}) \text{ (noise signal 1)} \\
n_2(t) &= A_{n_2} \sin(2\pi f_{n_2} t + \phi_{n_2}) \text{ (noise signal 2)} \\
n_3(t) &= A_{n_3} \sin(2\pi f_{n_3} t + \phi_{n_3}) \text{ (noise signal 3)} \\
d_1(t) &= s(t) + n_1(t) \text{ (additive noise)} \\
d_2(t) &= s(t) + n_1(t) + n_2(t) * n_3(t) \text{ (multiplied noise)}
\end{aligned}
$$

We will experiment on additive and multiplied noise to the true signal. A baseline linear neural network will be applied on both types of signals, then the focus lies exclusively on recovering the true signal from the multiplied noise with more advanced neural networks. We experiment with different activation functions and data structures that are fed into the neural network.

Initially, we apply a neural network with 1 hidden layer and a linear activation function to $d_1(t)$. This approximates the true signal incredibly well, as it should. Realistically, a neural network is not needed for this approximation, since it is as simple of an additive noise signal as we can get. We see the approximation of this neural network in the figure below.

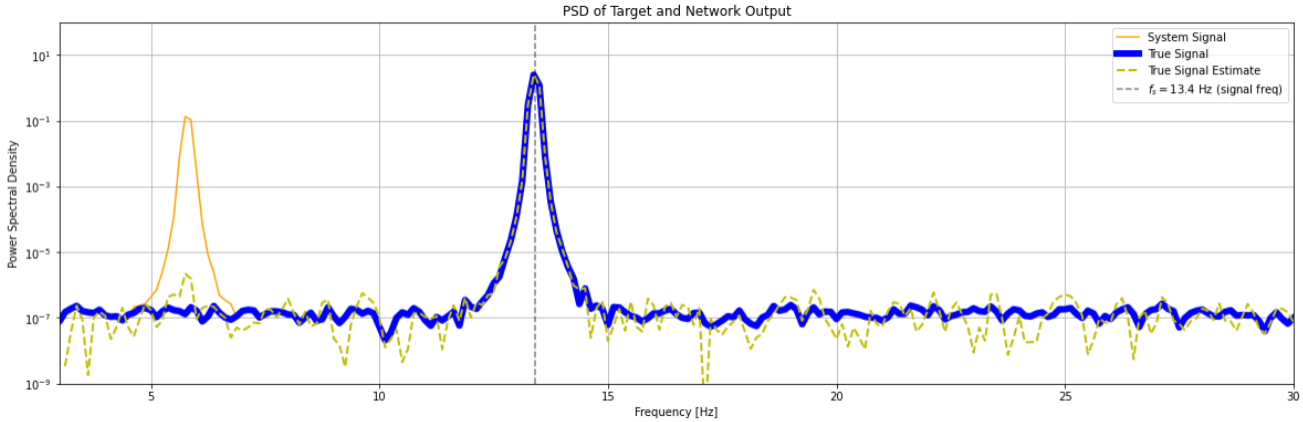

Figure 1: Linear network results with additive noise.

For $d_2(t)$, the multiplied noise signal, the same linear neural network performs poorly. The estimated output from the network ends up approximating the multiplied noise rather than the

true signal, effectively making it ineffective for detecting true signals when we need to apply it to real data. We experiment with a ReLU activation function, which generally is a good choice for neural networks when dealt with a classification problem. Unfortunately, for time series data, the results were only slightly better than the linear network. We now experiment with some data augmentation and apply different neural network structures to our data.

Rather than feeding in individual data points for the noise, the signals are augmented to be fed in to the network as chunks of 1-second intervals. When we apply the ReLU activation function, our results are fairly poor once again, as expected for time series data. We instead turn to methods that are designed specifically to work with time series data, such as recurrent neural networks. We implement a simple recurrent neural network with a hyperbolic tangent activation function. The results of this network are shown below.

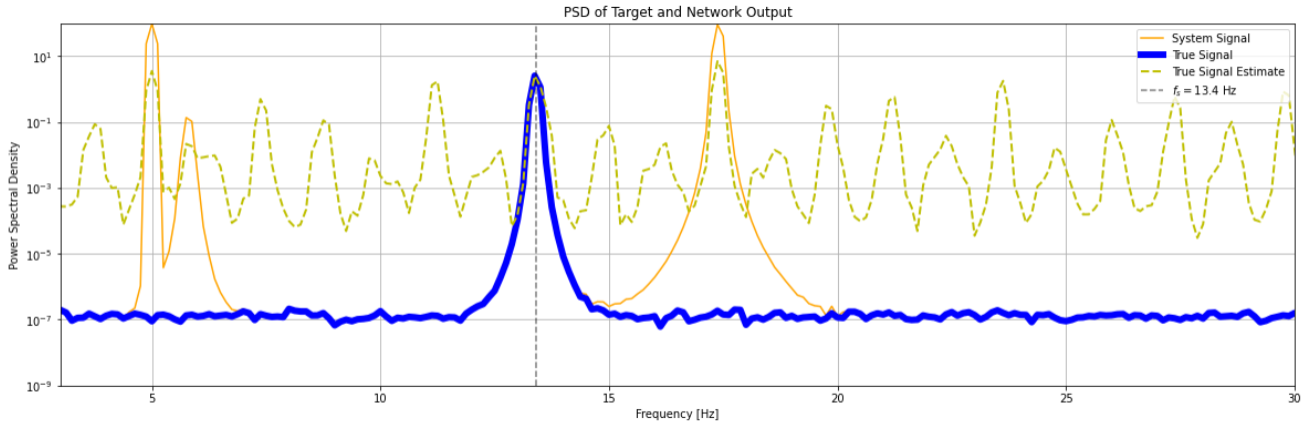

Figure 2: SimpleRNN results with multiplied noise and chunked data augmentation.

We see that the estimate includes peaks that are roughly on the same order of magnitude as our true signal. This makes it very difficult to tell which peaks are the true signal and which are misrepresented by the network. This prompted the question of combining multiple hidden layers in the network and observing performance then. The results of a simple RNN and a ReLU layer afterwards are displayed below.

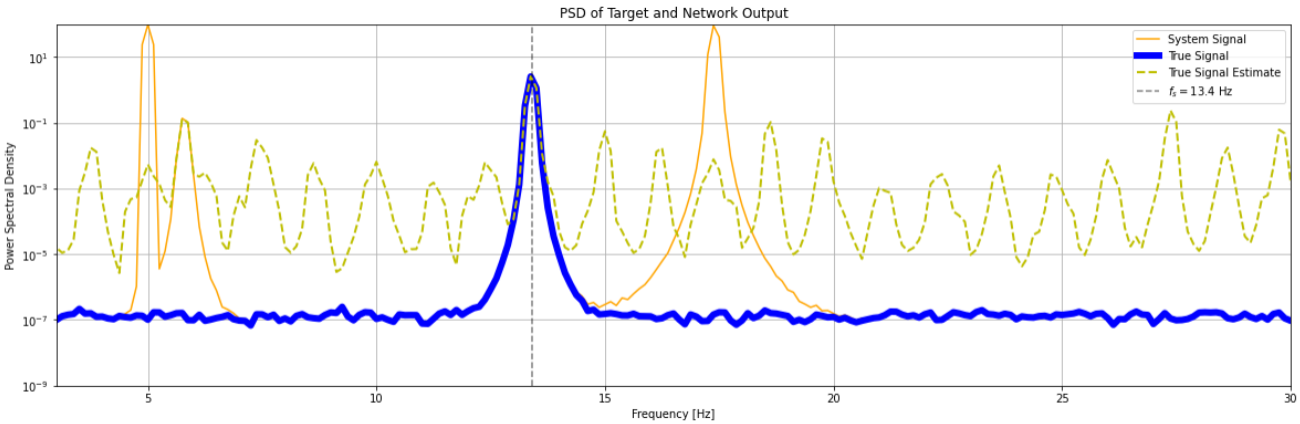

Figure 3: SimpleRNN + ReLU results with multiplied noise and chunked data augmentation.

This does significantly better than the simple RNN alone. We can see that the estimated peaks are about two orders of magnitude below our estimate for the true peak. This indicates that including additional layers will help improve the associations the network can make after implementing a layer intended to deal with time series data.

Moving on to the final, and best performing network, we have a special version of an RNN. The LSTM (long short-term memory) architecture functions by "remembering" important data chunks that approximate our true signal well and using that to aid the performance of the network. This was expected to perform better than our previous attempts, and our visualization of our estimate confirms that idea.

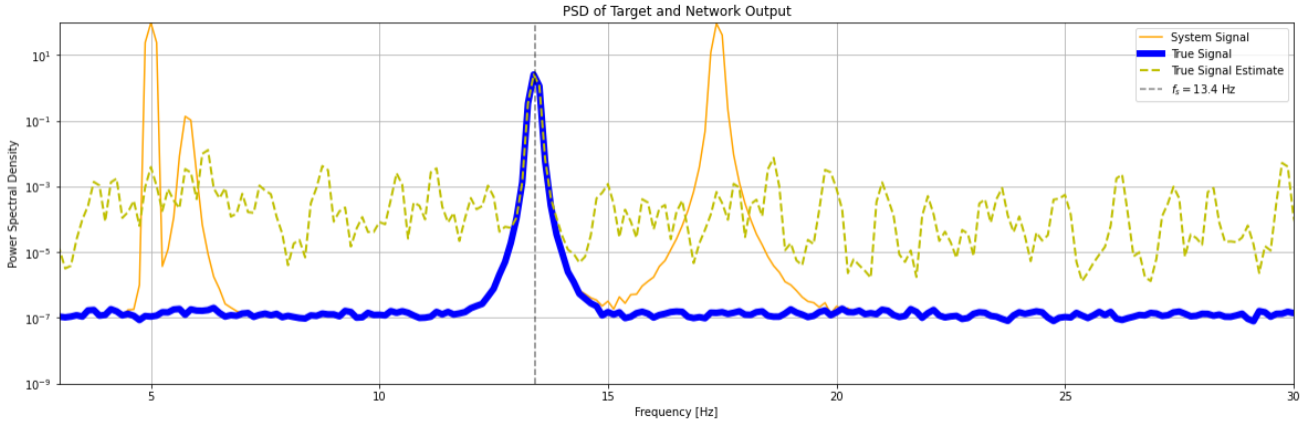

Figure 4: LSTM Results with multiplied noise and chunked data augmentation.

From the LSTM estimate, we see that the multiplied noise peaks are underapproximated by about 2-3 orders of magnitude, a very promising sign. The true signal stands out, making peak detection significantly easier and more efficient.

To recap, we have applied various neural networks to different noise signals to extract a true signal from sample data. The LSTM performed the best, a promising sign for our analysis on real LIGO data. The results from working on the sample data should provide a useful template for working with the true data and removing external noise factors.