DeepClean Neural Network for Gravitational Wave Noise Reduction

John Choi, Matthew Vigil, Laura Jian, Preethi Karpoor
University of California, San Diego
Group 5
Dated: 24th March, 2023

Processing of gravitational wave (GW) strain is necessary for astrophysical study of stellar bodies. Machine learning algorithms present an efficient tool to remove noise as demand for increased precision of GW diagnostic techniques increase. A replication of the noise subtraction pipeline developed by Ormiston et. al[1] designed to remove instrumental noise from a LIGO GW dataset is discussed and analyzed.

I. Introduction

Gravitational waves (GW) represent strains or distortions in space-time. These waves are currently being studied using the Laser Interferometer Gravitational Wave Observatory (aLIGO). The observations of these GW signals are heavily obscured by transient and periodic instrumental noise sources.

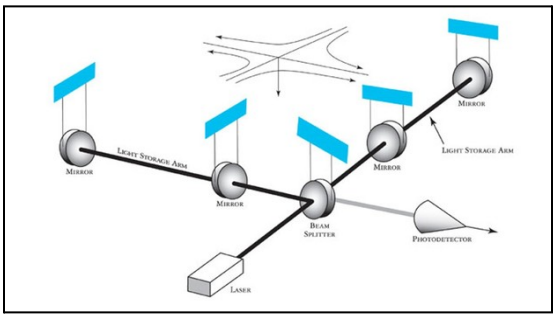

FIG. 1 - LIGO's working principle

Machine learning algorithms have become methods of interest for efficiently increasing instrumental detection sensitivity.

In this work, we attempted to replicate the DeepClean convolutional neural network (CNN), wherein the auxiliary channel signals (of non astrophysical origin) are passed through a regression pipeline to estimate the noise coupled with the strain signal incorporating machine learning techniques.

Window length and batch sample size have been noted as a strong influencer on model performance. Window size was used as a variable to test model loss, training time, signal-to-noise ratio (SNR), and ASD ratio.

II. Dataset

Our dataset consists of GW strain time series $h(t)$ along with 21 channels of auxiliary witness data $w_i(t)$ which records noise from several possible sources.

The witness channels could be sourced from the environment or could be auxiliary interferometric channels containing witnessed noise without the signal.

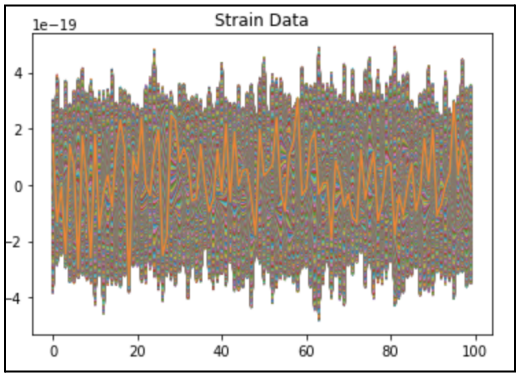

FIG. 2 - Strain data

The dataset was split into 80% training and 20% validation (test) sets. The training data was produced by randomly sampling the dataset and slicing portions of raw signal at a set window length. The sampling rate was 4096/s. It is good practice to validate time-series models on the latest data available. For this reason, validation data was sampled by slicing the largest fifth of indices from the original dataset.

III. Methods

Data Pre-processing

Following the literature, several pre-processing techniques were applied to the original dataset before feeding it into the network. The pre-processing was

applied to both the strain data and the witness channel data.

The pre-processing pipeline applied in this work differed from Ormiston's in several ways. The anti-imaging filter was not necessary in our reproduction as our strain data and our witness data were sampled at the same frequency. Furthermore, we did minimal windowing for the initial scope of the project. The processes we did adopt were the data normalization, band-pass filtering, and calculation of ASD for the witness data.

A Cross Spectral Density Analysis and Power Spectral Density Analysis was performed on each witness channel with respect to the strain data and each witness channel. After performing the CSD and PSD analysis, the frequencies that carried the largest contributions to the data were estimated to be between 0 and 0.3 Hz.

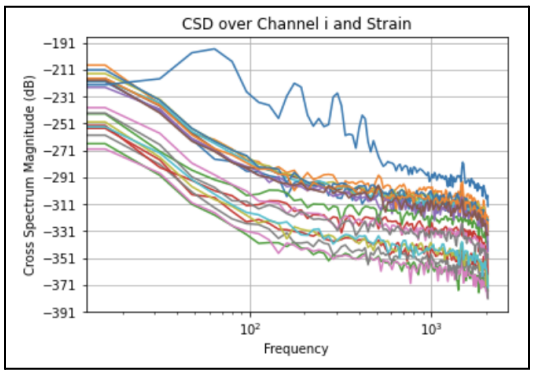

FIG. 3 - Finding pass-band frequencies via Cross Spectral Analysis

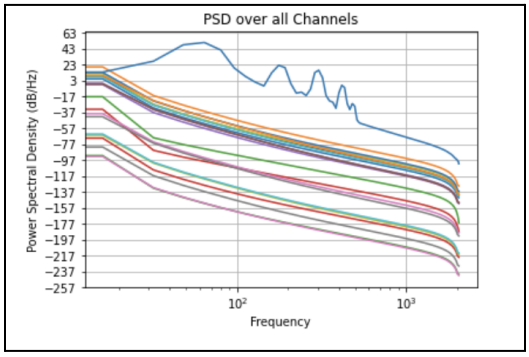

FIG. 4 - Finding pass-band frequencies via Power Spectral Analysis

After, an 8th order band-pass Butterworth filter was built. This is to account for nonlinear coupling between the witness data and the strain data. Therefore, this filtering was only applied to the witness channels, and not the strain data.This filtered any unnecessary power contributions from any of the witness channels outside the pass band of interest. As a result, we built the pass band to be between those with a slope decline of 160 dB/Dec.

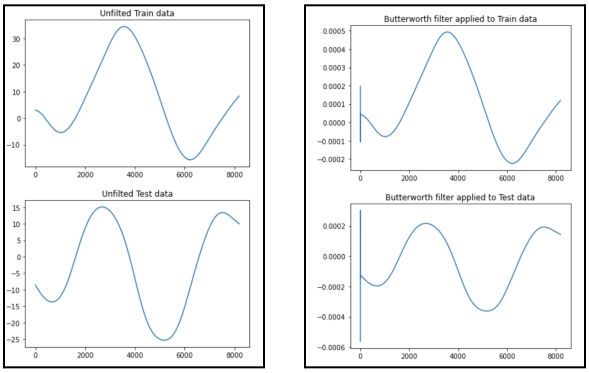

FIG. 5 - Training and Validation data before (left) and after (right) applying 8th order Butterworth Filter for 0 to 0.3Hz.

With the Butterworth filter built, we were able to pass the witness channels through the filter in order to reduce the power contributions from outside 0 Hz to 0.3 Hz.

Normalization was applied to both sets of strain and witness data using scipy's StandardScaler. This was to ensure that there was no single witness channel that would overpower the contributions to the SNR, and to reduce the possibilities of numerical complications in the computing of the custom loss function. Although the custom loss was not integrated into the file model (See Subsection Neural Network Architecture/Training), we maintained the normalization step to remove channel bias in model training.

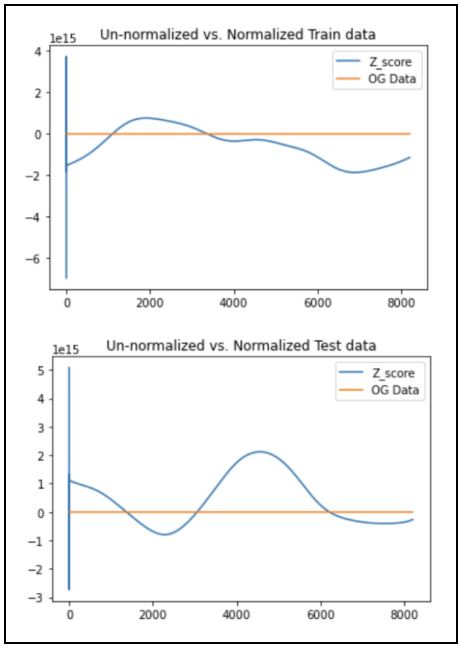

FIG. 6 - Training and Validation data after normalization.

Neural Network Architecture/Training

DeepClean is a 1-dimensional CNN designed to accept the pre-processed witness channel data, outputting a set of predicted noise. Data shape and size is given special care, due to the need for predicted noise to be subtracted from the test strain. For this reason, DeepClean is symmetrically built, with an equal amount of 1DConv and 1DConvTranspose layers. Each layer accepts the output from the layer before it, running a kernel across the sample. The 1DConv layers downsample the data, while the 1DConvTranspose layers upsample, preserving data shape. The result is the retention of model parameters as the data is passed from layer to layer. CNNs are known for their ability to retain long-term features and long time-series data, which makes this architecture particularly advantageous for noise reduction.

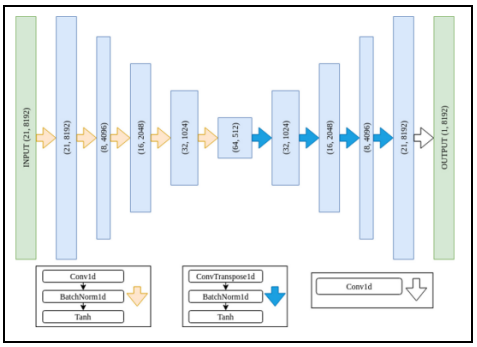

FIG. 7 - DeepClean architecture. This was the model replicated for this work[1].

Ormiston et. al. utilizes a custom loss function for DeepClean, composed of both an ASD and MSE term. These terms are weighted using a hyperparameter *w*, which can be adjusted based on the dataset. Due to the nature of our model implementation, this function would need to be written in TensorFlow, using only TensorFlow operations. Time did not permit the implementation of this function into our model so instead, we used the standard mean-squared error from TensorFlow. We expect this to negatively affect our testing and validation loss, and the output noise prediction.

Table 1: Constant DeepClean Hyperparameters

Kernel size	7
Activation function	tanh
Padding	same
Loss	MSE
Initial Learning Rate	$1 \cdot 10^{-3}$
Optimizer	ADAM
Batch size	32
Max epochs	10

Table 2: DeepClean Run data parameters

Run	1	2	3
Window Length (s)	2	4	8
Data size	8192	16384	32768

Three runs were conducted to test model performance. Window length was modified for each run, from 2s to 8s (Table 2).

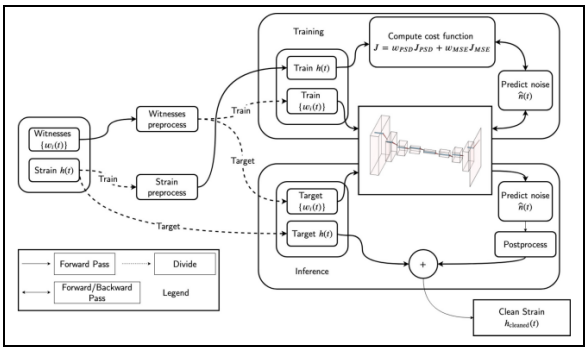

FIG. 8 - Overall workflow of noise subtraction pipeline[11].

Data Post-processing

The predicted noise generated from DeepClean must be post-processed before subtraction and analysis. This includes un-normalizing both the test strain data and the predicted noise, and applying an additional 8th order Butterworth band pass filter. Ormiston et. al. includes an anti-imaging filter to upsample the noise prediction. This was conducted due to the different sampling rates of the witness data and strain data. Since both our witness and strain data were sampled at the same rate, this step was not necessary.

The predicted noise and test strain data were un-normalized, returning the data to its original units and range. This is necessary to directly subtract the predicted noise from the test strain. To do this we multiplied our predicted noise by their standard deviations and added back their means. We then applied an 8th order Butterworth filter, to reduce possible instabilities and power contributions. Frequencies outside of 0 to 0.3Hz were filtered, similar to the pre-processing procedure. After post-processing, the predicted noise was subtracted from the strain validation data.

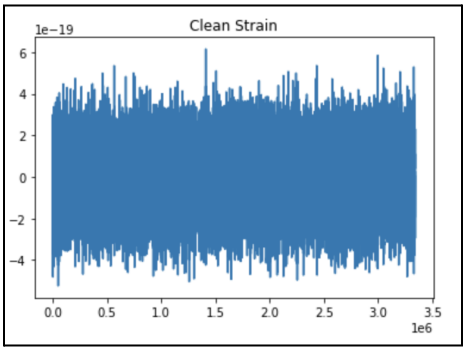

FIG. 9 - Clean strain after subtracting predicted noise.

IV. Results

The model training gave poor results, with both training and validation loss converging near 1 across 10 epochs. Validation loss was measured below training loss in some runs, with training and validation loss diverging in others. This indicates model underfitting. It is likely the lack of an ASD term in the loss function did not allow for proper training, leaving a skewed result. It is also possible there is an amount of data leakage, as the training set is a randomized sampling of time windows within the dataset, and validation is non-randomized samples of the last fifth of data indices. Ormiston et. al. implemented data randomization in their model to prevent possible contamination between training and validation, but that was not included in this work. Validation of the model on test witness data was successful, but implementation of the custom loss is necessary in order to validate the rigor of the following results.

Ormiston compared the performance of the DeepClean pipeline to the application of a Wiener filter, a non-ML method. In the interest of time this method was not implemented during study. Instead, we attempted to replicate multiple benchmarks within Ormiston, including the signal-to-noise ratio (SNR) and amplitude spectral density (ASD) ratio between the original and clean strain.

The SNR is calculated by:

$$(1) \quad SNR = \frac{P_{signal}}{P_{noise}}$$

where the power of the signal and noise is the root mean score of each respective dataset.

$$(2) \quad \frac{1}{N} \sum_{n=0}^{N-1} x(n)^2$$

Our final SNR across all three runs was ~143-147 dB.

To further compare our clean strain with the original, we generated an amplitude spectral density for each to compare. Since the changes in ASD were small, a ratio was generated to see the effects of the noise. he resulting model produced a minimum ASD ratio of 0.9994 at approximately 100 Hz, with a measured SNR between 143-147 dB between iterative runs. Compared to an ASD ratio spread from approximately 0.5 to 1.5 across all frequencies, our model replication showed low effectiveness in noise subtraction versus the literature.

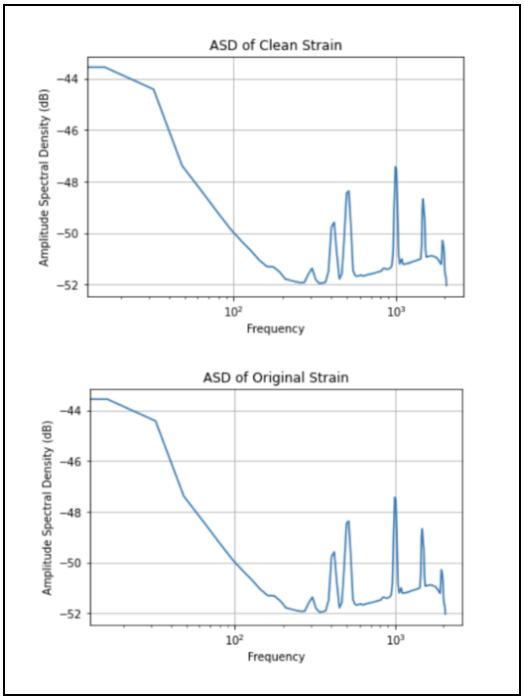

FIG.10 - ASD comparison of Clean and Original Strain

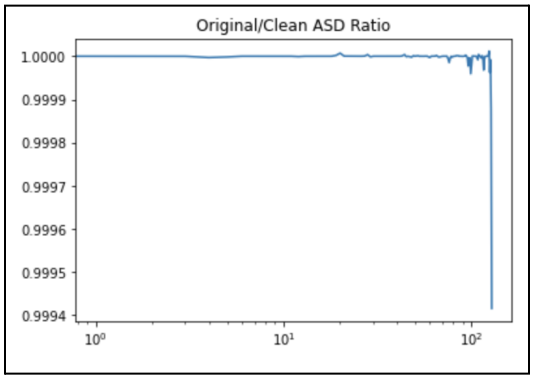

FIG.11 - ASD ratio between original and clean strain

V. Conclusion

An attempt at replicating the custom loss function described in Ormiston et. al. resulted in multiple errors and challenges including implementation of the custom loss function, data leakage, and high model error. In future work it would be necessary to compare the performance between the standard MSE loss and a successfully constructed custom loss.

Acknowledgments

Special thanks to Doctor Javier Duarte and Alec Gunny for their insight and guidance throughout this work.

VI. Data Availability

All code for this work can be found in the GitHub repository.

VII. References

1. R. Ormiston, T. Nguyen, M. Coughlin, R. X. Adhikari, and E. Katsavounidis, “Noise reduction in gravitational-wave data via deep learning,” Phys. Rev. Res., vol. 2, p. 033066, Jul 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033066613

2. Mohd Aszemi, Nurshazlyn & Panneer Selvam, Dhanapal Durai Dominic. (2019). Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms. International Journal of Advanced Computer Science and Applications. 10. 269 - 278. 10.14569/IJACSA.2019.010063