Gravitational Wave Detection

Ravi Satvik 202018008@daiict.ac.in
Siddhant Shah 202018013@daiict.ac.in
Rushvi Shah 202018039@daiict.ac.in
Keya Shah 202018056@daiict.ac.in

Department - M.Sc. Data Science
Dhirubhai Ambani Institute of Information and Communication Technology,
Gandhinagar, Gujarat, India (382007)

***Abstract*— Detecting and analyzing the information carried by gravitational waves allows us to observe the Universe in a way never before possible, providing astronomers and other scientists with their first glimpses of literally unseeable wonders. This project aims to experiment with various data visualization methods and deep learning techniques to classify Gravitational Wave signals from pure noise signals. To illustrate the complexity of the problem, Data visualization techniques namely, PyPlot, Parallel Data Plots, CQT Spectral Plots, Scipy Spectrograms, Welch Periodogram, Coherence Plots were used. The data set consists of time-series from three detectors sampled at a rate of 2048 hz for a period of 2 seconds. To improve performance of Basic CNN Model improvement techniques, efficient net, batch normalization and drop out layers were used. The performance of the model was evaluated using loss and AUC score. The proposed model achieved a maximum AUC of 0.85.**

Keywords—Gravitational Wave, Binary Black Hole, Deep Learning, Convolutional Neural Networks

I. Introduction

As with the multi disciplined approach to the invention of Gravitational-Waves, additional expertise will be needed to further research of Gravitational-Wave. In particular, social and natural sciences have taken an interest in ML, DL, classification problems, data mining, and also visualization to develop new techniques and algorithms to efficiently handle complex and massive data sets. The increase in computing power and the development of unique techniques for the quick analysis of data will be vital to the exciting new field of Gravitational-Wave Astronomy. Hypothetical outcomes may include increased sensitivity to Gravitational-Wave signals, data conditioning tools, application to control and feedback systems for next-generation detectors, noise removal, and signal characterization.

G2Net is a network of Gravitational-Wave, Geophysics and Machine Learning. Via an Action from European Cooperation in Science and Technology(COST), a funding agency for research and innovation networks, G2Net aims to create a large network of scientists. From four different areas of expertise, namely Gravitational-Wave physics, Robotics, Geophysics and Computing-Science, these scientists have agreed on a common aim of tackling challenges in data analysis and noise characterization for Gravitational-Wave detectors. In this project we use data provided with a training set of time series data containing simulated Gravitational-Wave measurements from a network of 3 Gravitational-Wave interferometers (LIGO Hanford, LIGO Livingston, and Virgo). Each time series contains either detector noise or detector noise plus a simulated Gravitational-Wave signal. The main task is to identify that when a signal is present in the data (which is target=1). The parameters that decide the exact form of a binary black hole waveform are the masses, distance, sky location, black hole spins, time of arrival, binary orientation angle, Gravitational-Wave polarization, and phase at coalescence (merger). These parameters (15 in total) have been randomized according to astrophysically motivated prior distributions and used to generate the simulated signals present in the data, but are not provided as part of the competition data. Each data sample which in the npy file contains 3 time series (1 for each detector) and each spans 2 sec and is sampled at 2,048 Hz. The integrated signal to noise ratio (SNR) is classically the most Instructive measure of how detectable a signal is and a typical level of detectability is when this integrated signal to noise ratio exceeds ~8.

II. Model-under-study

A. Matched filtering

Matched filtering works by multiplying the function of time by an output of the detector (called the template) that shows an anticipated wave shape, and summing (integrating) the result. If there's a signal matching the wave shape buried in the noise then the output of the filter will be more than anticipated for pure noise. The fundamental assumption of matched filtering is that the strain s(t) measured by the interferometric detector is made up of two additive components, namely the instrument noise n(t) and the (astrophysical) signal h(t):

$$s(t) = n(t) + h(t) \qquad (1)$$

For a given power spectral density Sn of n, we can then quantify the agreement between a given template T(t) in the template bank and the recorded strain s(t) at a time t0 by computing the signal-to-noise ratio (SNR). For an appropriate choice of normalization, the matched filtering signal-to-noise ratio is given by:

$$SNR(t_0) = \int_{-\infty}^{\infty} \frac{s(f).T^*(f).e^{2\pi i f t_0}}{S_n(f)} df \qquad (2)$$

where the tilde denotes the Fourier transform. For stationary Gaussian noise it can be shown that by design the SNR is indeed the optimal detection statistic for finding a signal h(t) if the time-reversed template T(−t) is equal to the signal. This is called the matched filter. In practice, the template bank should therefore contain exact simulated wave shapes that cover the space of expected signals in the recorded data sufficiently densely. Computing the SNR for every wave shape in the template bank and applying a threshold then produces a list of candidate event times[1].

In reality, the data is usually neither stationary nor exactly Gaussian. One particular challenge to data analysis are so called glitches. Glitches are nonstationary noise transients, which incorporate a range of different short time phenomena that affect the quality of the data measured by the detectors. They occur frequently, at rates up to many times per minute. As a consequence of these non-Gaussian and non-stationary effects, the real distribution of the SNR is not known and must be determined empirically in order to obtain calibrated statistical results from the computed SNR.

B. Neural Network

Neural network(NN) consists of a node layer, containing an input layer, it has one or more hidden layers, and also an output layer. Each node, or artificial neuron connects to another and it has an associated weight and threshold. If the output of any node is above the specified threshold value then that node is activated, sending data to the next layer of the network. Else, no data is passed along to the next layer of the network.Think of each node as its own linear regression model, composed of input data, weights, a bias (or threshold), and an output. Once an input layer is determined, weights are assigned. These weights help us to determine the importance of any given variable, with larger ones contributing more significantly to the output compared to other inputs. All inputs are then multiplied by their respective weights and then summed. Afterward, the output is passed through an activation function, which determines the output. If that output exceeds a given threshold, it "fires" (or activates) the node, passing data to the next layer in the network. This results in the output of one node becoming the input of the next node. This process of passing data from one layer to the next layer defines this neural network as a feedforward network. Some weights are assigned to determine importance. Larger weights signify that particular variables are of greater importance to the decision or outcome. The main goal is to minimize the cost function to ensure correctness of fit for any given observation. As the model adjusts its weights and bias, it uses the cost function and reinforcement learning to reach the point of convergence, or the local minimum. The process in which the algorithm adjusts its weights is through gradient descent, allowing the model to determine the direction to take to reduce errors (or minimize the cost function). With each training example, the parameters of the model adjust to gradually converge at the minimum.

C. CNN

A CNN is composed of interconnected layers of computational nodes. The nodes are known as neurons and the outputs of which are processed with an activation function. There are commonly three types of layers: convolutional layers, max-pooling layers, and fully connected layers. Convolutional layers perform the mathematical operation of convolution between the weights of the layer's neurons and the input to that layer. Max-pooling layers perform a down-sampling process that reduces the size of the data by selecting the maximum of data samples within fixed size bins. It can reduce the computational cost by decreasing the number of trainable parameters (weights and biases) of the CNN. Fully connected layers are layers that connect every neuron in its layer to every neuron of its immediate previous and next layer. When multiple layers of these three types are stacked and connected one after the other, a CNN has been formed, where the output of each layer is the input of the next layer. In a standard CNN, the first layer, also known as the input layer and often a convolutional layer, takes in raw values of the input (or time series from GW detectors in our case). The output layer is usually a fully connected layer with a softmax activation function with output used to represent the class scores or probabilities in the case of detection and classification. During the training stage, the weights of the neurons in a CNN are updated using an algorithm called back-propagation. The outputs of the CNN are used as an input to the loss function that is associated with the entire network. The value of the loss function is used to evaluate how well the algorithm models the input data of the CNN. After a subset of training data is fed through the CNN, the back-propagation algorithm then computes the gradient of the loss function with respect to the trainable weights and biases within the network. The size of the subset of the training data is referred to as batch. A gradient descent algorithm is then used to adjust the values of the weights and biases of the neurons in each layer in order to iteratively minimize the loss function. When the loss function is minimized and training is therefore complete, the CNN will be able to take in input in the form of new data and its output will best represent the probability of that data belonging to each of the trained classes. In this work, we employ a CNN of eight convolutional layers, three max-pooling layers, and three fully connected layers.

III. Description of the Problem

Although matched filtering techniques have proven highly successful in discovering binary black hole coalescences from the recordings of the Advanced LIGO and Advanced Virgo gravitational-wave observatories. Additional observatories in India and Japan are expected to become operational in the next five years forming an evolving detector network capable of observing hundreds of sources every year.

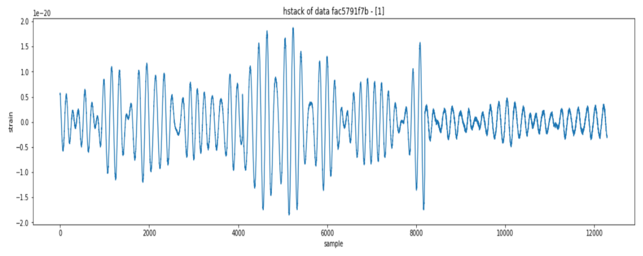
These sources will need to be expeditiously observed, localized in the sky and this information quickly disseminated to electromagnetic partners to maximize the chance of multi messenger observations. This requires reliable, real-time identification of potential compact binary coalescences (CBCs) to provide a time window and basic parameter estimate for slower, but more accurate Bayesian inference techniques to follow-up. Current matched filtering techniques are expensive computationally , with the computational cost scaling as a function of the broadness of the detector's sensitivity curve and the number of observatories; both of which are expected to increase in the coming years.

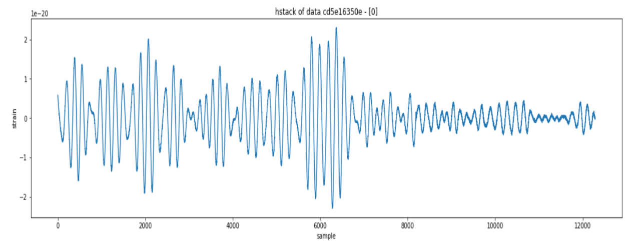
In recent years, ML techniques, in particular convolutional neural networks, have been investigated as a method to replace or complement traditional matched filtering techniques that are used to detect the gravitational-wave signature of merging black holes. These techniques have not yet been successfully applied to the analysis of long stretches of data recorded by the Advanced LIGO and Virgo gravitational-wave observatories.

IV. Solution to the Problem

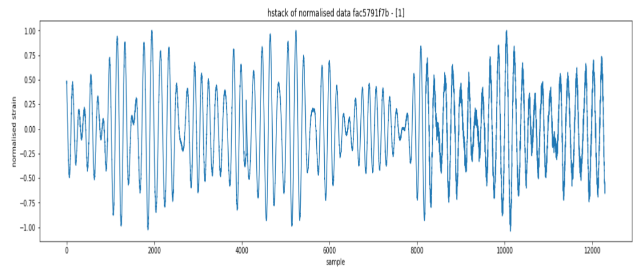
Exploratory Data Analysis

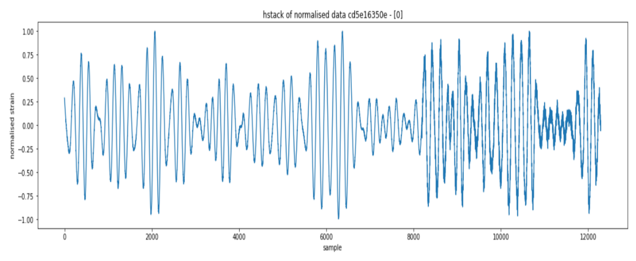
We begin by exploring the data through various data representation techniques to get a better understanding of the data. The first step was to plot the data of a sample with a gravitational wave [1] and a wave with pure noise [0].



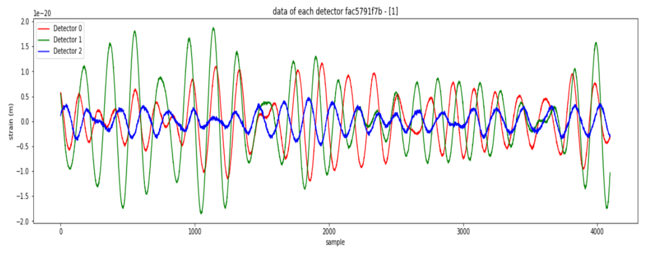


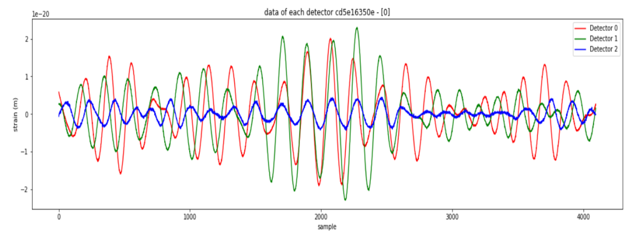
After which, we normalized the strains and plotted the data of a normalized sample with a gravitational wave [1] and a normalized wave with pure noise [0]



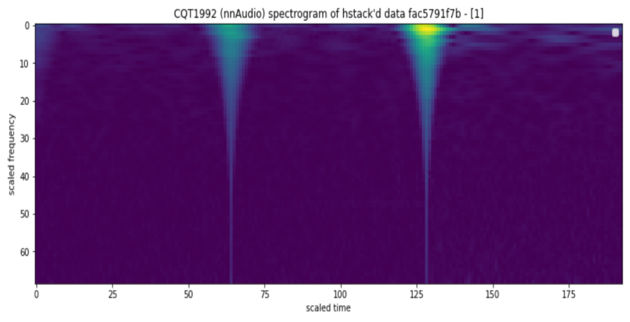


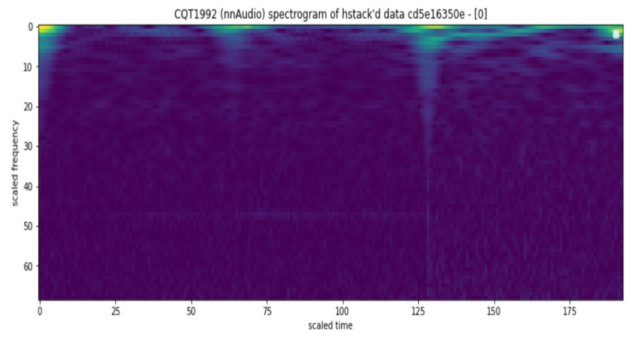
Then we plotted the strains from the three detectors (L1 = Detector 0, H1 = Detector 1 and V1 = Detector 2) parallely to each other to visualize the time delay between the detectors and the difference in the magnitudes of the waves detected.



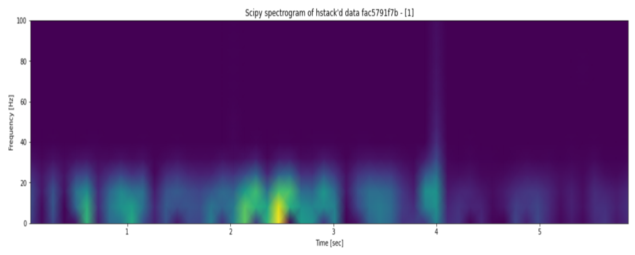


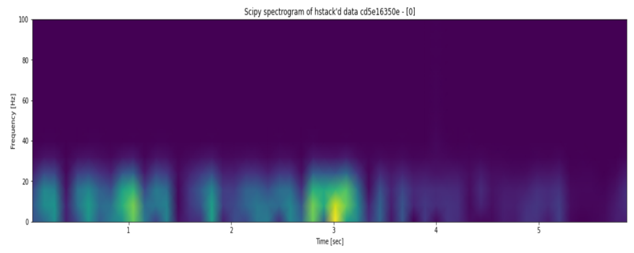
Then we created spectrograms of the strains using the nn Audio - Spectrogram toolkit. A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams. nnAudio is an audio processing toolbox using PyTorch convolutional neural network as its backend. By doing so, spectrograms can be generated from audio on-the-fly during neural network training and the Fourier kernels (e.g. or CQT kernels) can be trained. In these spectrograms, we can clearly observe obvious peaks at 1/3 and 2/3 of the total time due to discontinuity between detector data.





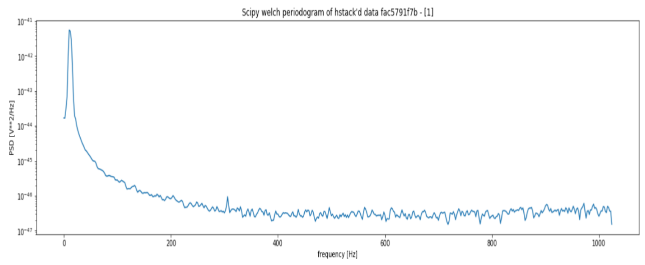
We then generated spectrograms using the Scipy Spectrogram toolkit which is used to compute a spectrogram with consecutive Fourier transforms. It is used as a way of visualizing the change of a nonstationary signal's frequency content over time.

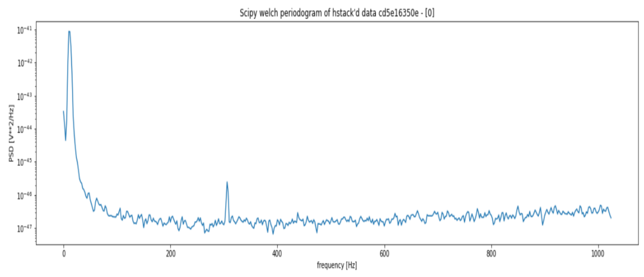




After which we Estimated power spectral density using Welch's method.

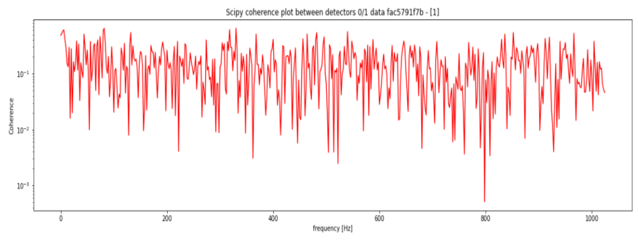
Welch's method computes an estimate of the power spectral density by dividing the data into overlapping segments, computing a modified periodogram for each segment and averaging the periodograms.

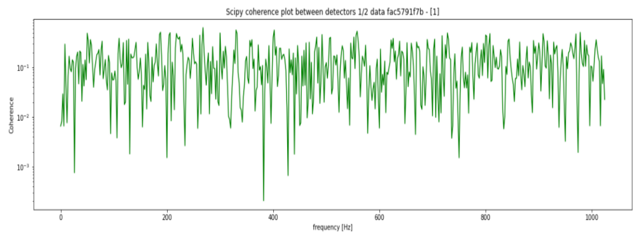


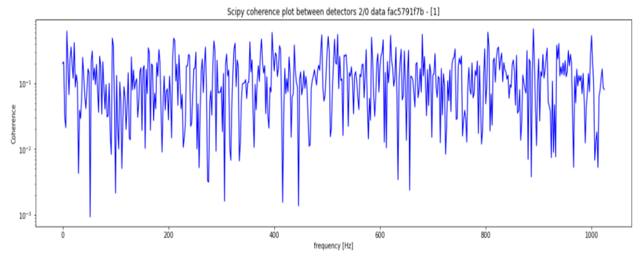


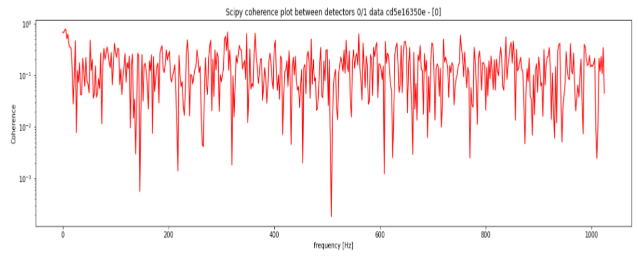
After which we estimated the magnitude squared coherence estimate, Cxy, of discrete-time signals using Welch's method.

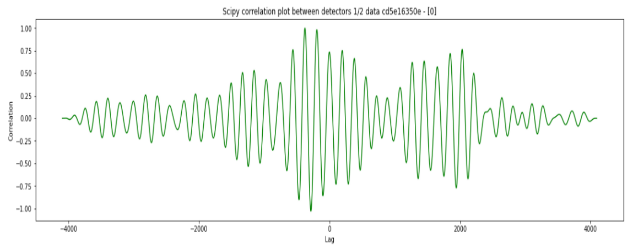
Cxy = abs(Pxy)**2/(Pxx*Pyy), where Pxx and Pyy are power spectral density estimates of X and Y, and Pxy is the cross spectral density estimate of X and Y.

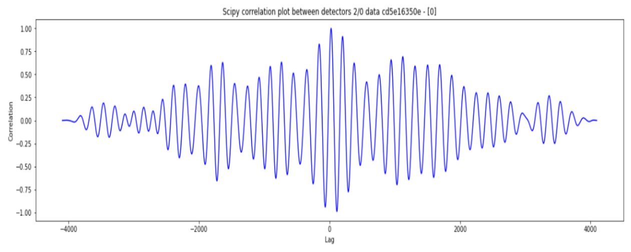






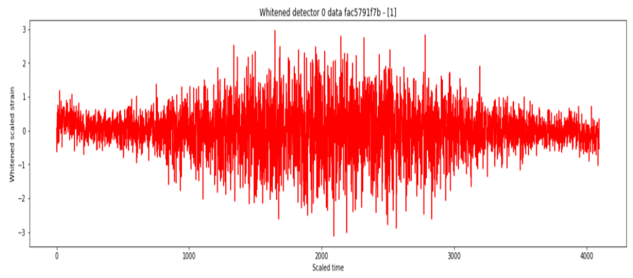


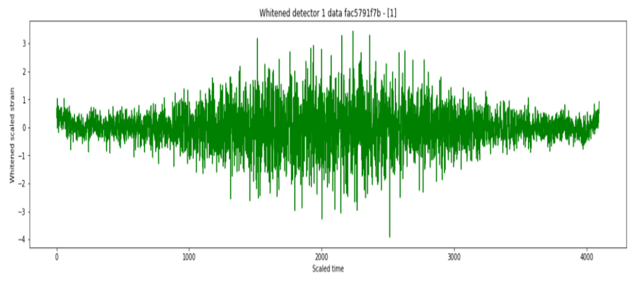


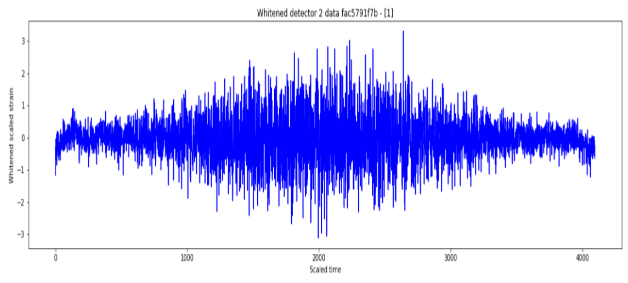


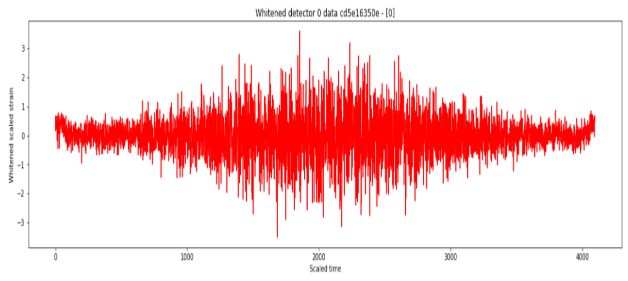
As we may observe from the above graphs, there is a significant correlation in the strains caused due to the detectors, hence we need to whiten the strains before we train the model.

Data Preprocessing and Data Sample Generation:
Spectral whitening is an attempt to make the spectrum of the signal "more uniform". One reason this might be a good thing to do is that it can have the effect of making the autocorrelation of the signal "narrower". This helps localize the signal in time.
A signal can be whitened by - fourier transform, divide by PSD, inverse fourier transform.

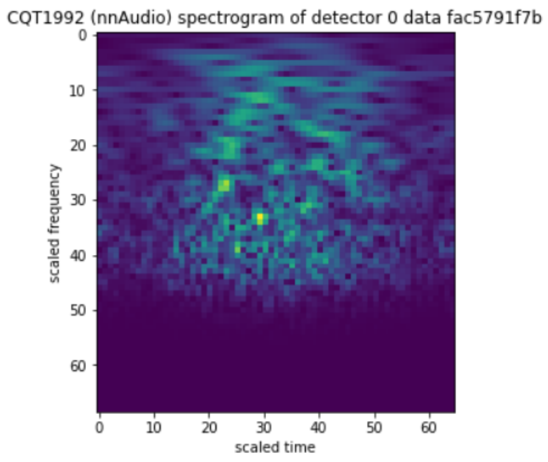


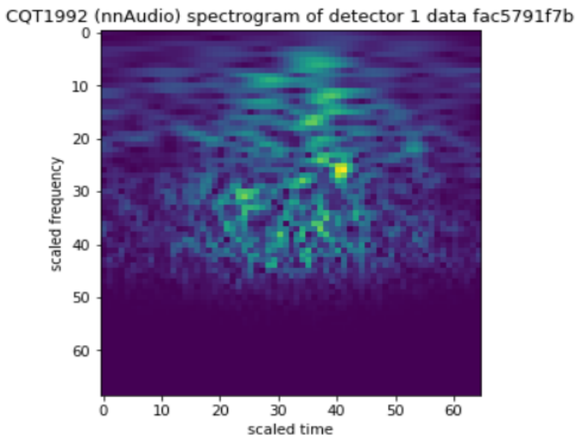


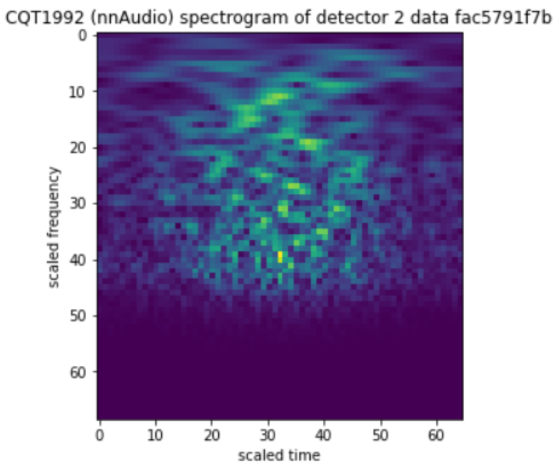




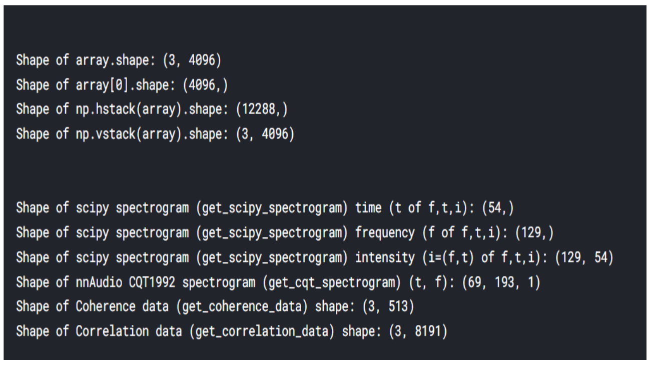
After whitening the strain, we plot the nnAudio Spectrogram of each of the detector strains to create the dataset used to train the model.







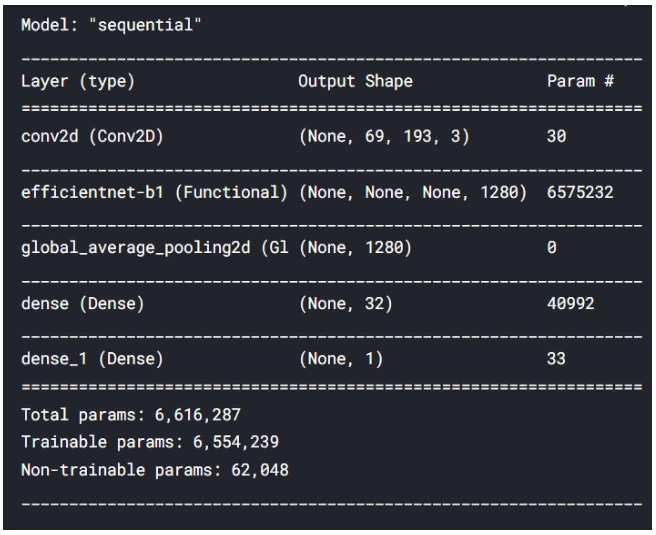
To summarize so far, no signal jumps out to the naked eye, by considering various time-domain / frequency-domain plots. This highlights the non-trivial nature of this data science problem. The sizes of the various data representations used so far are:



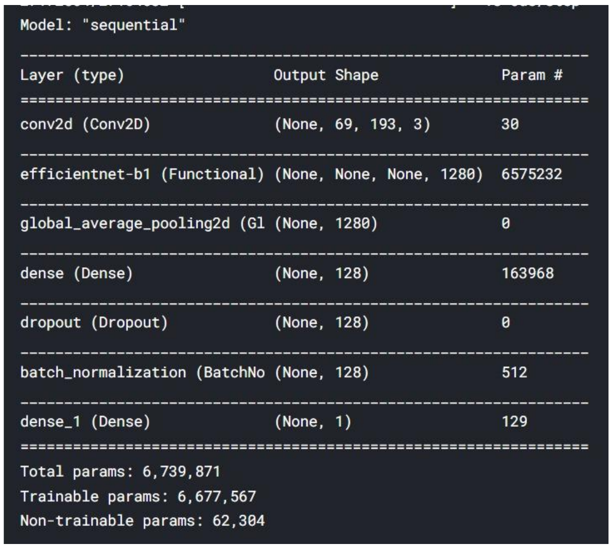
```
Shape of array.shape: (3, 4096)
Shape of array[0].shape: (4096,)
Shape of np.hstack(array).shape: (12288,)
Shape of np.vstack(array).shape: (3, 4096)


Shape of scipy spectrogram (get_scipy_spectrogram) time (t of f,t,i): (54,)
Shape of scipy spectrogram (get_scipy_spectrogram) frequency (f of f,t,i): (129,)
Shape of scipy spectrogram (get_scipy_spectrogram) intensity (i=(f,t) of f,t,i): (129, 54)
Shape of nnAudio CQT1992 spectrogram (get_cqt_spectrogram) (t, f): (69, 193, 1)
Shape of Coherence data (get_coherence_data) shape: (3, 513)
Shape of Correlation data (get_correlation_data) shape: (3, 8191)
```

Model Building:
The CQT Spectrogram dataset was trained on a 2 Convolutional Neural Network models created using the Keras library with the following architectures:

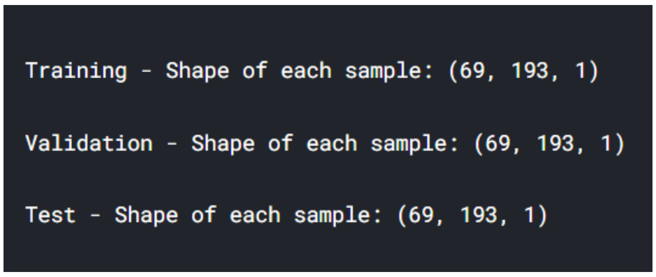


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 69, 193, 3)        30
_________________________________________________________________
efficientnet-b1 (Functional) (None, None, None, 1280)  6575232
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 32)                40992
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 6,616,287
Trainable params: 6,554,239
Non-trainable params: 62,048
_________________________________________________________________
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 69, 193, 3)        30
_________________________________________________________________
efficientnet-b1 (Functional) (None, None, None, 1280)  6575232
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 128)               163968
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
batch_normalization (BatchNo (None, 128)               512
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 6,739,871
Trainable params: 6,677,567
Non-trainable params: 62,304
```

The sizes of the training, testing and validation datasets given as input to the model are:



```
Training - Shape of each sample: (69, 193, 1)

Validation - Shape of each sample: (69, 193, 1)

Test - Shape of each sample: (69, 193, 1)
```

of a sample with a gravitational wave [1] and a wave with pure noise [0].

V. Results

The model was trained using the Model Checkpoint, Reduce on Plateau and Early Stopping Call backs to prevent overfitting the model. The first model completed training along with a validation loss of 0.4475 and a validation area under ROC curve of 0.8516. The second model completed training with validation AUC of 0.8299 and loss of 0.4723. Hence we can conclude that the dropout layer and batch normalization did not improve the model's performance.

VI. Conclusions

From this project and its results, it can be seen that the CNN model shows good generalization ability for data of different parameter ranges, which is a major advantage over the matched-filtering method in using the CNN model. The matched-filtering method is based on the existing template bank. Its search and response to GW signals are limited to the existing waveforms. The GW signals beyond the existing waveforms cannot be easily detected. The generalization ability of the CNN model in the task of GW signal detection will help us to discover signals beyond the existing templates. It may also be able to detect signals that imprint orbital eccentricity, orbital precession, and deviations from general relativity. Such a generalization ability will undoubtedly play an essential role in searches of GW signals beyond what we have in the template bank. More studies are needed along this direction.

Other areas of Gravitational Wave detection in which Deep Learning techniques may be explored are: the detection of the early inspiral of a gravitational-wave signal, Detection and classification of supernova gravitational wave signals, Detection of gravitational-wave signals from binary neutron star mergers, and Gravitational wave denoising of binary black hole mergers.

References

[1] Gregory Baltus, Justin Janquart, Melissa Lopez, Amit Reza, Sarah Caudill and Jean-Rene Cudell, "Convolutional neural networks for the detection of the early inspiral of a gravitational-wave signal", Phys. Rev. D 103, 102003 (2021), DOI: 10.1103/PhysRevD.103.102003.

[2] Daniel George, E.A. Huerta, "Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data", Physics Letters B, Volume 778, 2018, Pages

64-70, ISSN 0370-2693, https://doi.org/10.1016/j.physletb.2017.12.053.

[3] Man Leong Chan, Ik Siong Heng and Chris Messenger, “Detection and classification of supernova gravitational wave signals: A deep learning approach”, Phys. Rev. D 102, 043022 (2020), DOI: 10.1103/PhysRevD.102.043022.

[4] Timothy D. Gebhard, Niki Kilbertus, Ian Harry and Bernhard Schölkopf, “Convolutional neural networks: A magic bullet for gravitational-wave detection?”, Phys. Rev. D 100, 063015 (2019), DOI: 10.1103/PhysRevD.100.063015

[5] Marlin B. Schäfer, Frank Ohme and Alexander H. Nitz, “Detection of gravitational-wave signals from binary neutron star mergers using machine learning”, Phys. Rev. D 102, 063015 (2020), DOI: 10.1103/PhysRevD.102.063015.

[6] Wei Wei, E.A. Huerta, “Gravitational wave denoising of binary black hole mergers with deep learning”, Physics Letters B, Volume 800, 2020, 135081, ISSN 0370-2693, https://doi.org/10.1016/j.physletb.2019.135081.

[7] Heming Xia, Lijing Shao, Junjie Zhao and Zhoujian Cao, “Improved deep learning techniques in gravitational-wave data analysis”, Phys. Rev. D 103, 024040 (2021), DOI: 10.1103/PhysRevD.103.024040.

[8] Hunter Gabbard, Michael Williams, Fergus Hayes and Chris Messenger, “Matching Matched Filtering with Deep Networks for Gravitational-Wave Astronomy”, Phys. Rev. Lett. 120, 141103 (2018), DOI: 10.1103/PhysRevLett.120.141103.

[9] Alvin J. K. Chua and Michele Vallisneri, “Learning Bayesian Posteriors with Neural Networks for Gravitational-Wave Inference”, Phys. Rev. Lett. 124, 041102 (2020), DOI: 10.1103/PhysRevLett.124.041102

[10] G. Vajente, Y. Huang, M. Isi, J. C. Driggers, J. S. Kissel , M. J. Szczepańczyk and S. Vitale, “Machine-learning nonstationary noise out of gravitational-wave detectors”, Phys. Rev. D 101, 042003 (2020), DOI: 10.1103/PhysRevD.101.042003.