GRAVITATIONAL WAVES DETECTION

A PROJECT REPORT

Submitted by

NADAM DHEERAJ KUMAR REDDY
[RA1811033010066]
SATTI VIJAY NAGA SRI DHANANJAY REDDY
[RA1811033010085]

Under the Guidance of

Dr. Selva Kumara Samy
Assist. Professor, Department of Computational Intelligence

in partial fulfillment of the requirementsfor the degree of

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING
WITH SPECIALIZATION IN SOFTWARE
ENGINEERING**

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report [18CSP110L semester internship report] titled "**GRAVITATIONAL WAVES DETECTION** " is the bonafide work of "**NADAM DHEERAJ KUMAR REDDY [RA1811033010066], SATTI VIJAY NAGA SRI DHANANJAY REDDY [RA1811033010085]**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. SELVA KUMARA SAMY	DR. R. ANNIE UTHRA
SUPERVISOR	**Head Of The Department**
Assist. Professor	Dept. COMPUTATIONAL INTELLIGENCE
Dept. COMPUTATIONAL INTELLIGENCE	

Signature of Internal Examiner **Signature of External Examiner**

Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must besigned and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course	**: B.Tech / CSE with specialization in Software Engineering**
Student Name	**: Nadam Dheeraj Kumar Reddy**
Registration Number	**: RA1811033010066**
Title of Work	**: Gravitational Waves Detection**

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with theUniversity policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date forevery student in your group.

Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must besigned and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course	**: B.Tech / CSE with specialization in Software Engineering**
Student Name	**: Satti Vijay Naga Sri Dhananjay Reddy**
Registration Number	**: RA1811033010085**
Title of Work	**: Gravitational Waves Detection**

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with theUniversity policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date forevery student in your group.

ACKNOWLEDGEMENTS

I/We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extendedfor the major project work and his continued support. We extend our sincere thanks to **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her invaluable support. We wish to thank **Dr. Annie Uthra**, Professor & Head, Department of Computational Intelligence, SRM Institute of Science and Technology, for her valuable suggestions and encouragement throughout the period of the internship work.

I/We are extremely grateful to our Panel Head, **Dr.C.Lakshmi**, Assistant Professor, Department of Computing Technologies, SRM Institute of Scienceand Technology, for his / her inputs during the monthly reviews. We register ourimmeasurable thanks to our Faculty Advisor, **Jeyasudha**, Assistant Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. Selvakumarasamy**, Assistant Professor, Department of Computational Intelligence, for providing me an opportunity to pursue my major project under his/her mentorship. He / She provided me the freedom and support to explore the workof my interest.

I/We sincerely thank the staff and students of the Computational Intelligence Department, SRM Institute of Science and Technology, for their help during myinternship work. Finally, we would like to thank my parents, our family members,and our friends for their unconditional love, constant support, and encouragement.

Dheeraj Reddy

Dhananjay Reddy

ABSTRACT

According to popular scientific theories, the universe is said to consist of 4-dimensions (3 space co-ordinates and a time co-ordinate). It states that space and time are interlinked together to form a space-time continuum. Here the universe is said to be a fabric of space- time and that mass bends this fabric and any object accelerating through it produces ripples in space-time. These are gravitational waves. LIGO observatories are constructed for the sole purpose of detecting these waves. LIGO's two four-kilometer-long arms are placed in an L-shape, allowing one arm to stretch while the other shortens as a wave passes through. Although the forecast for this prediction was made over 100 years ago, equipment sensitive enough to detect them has only recently become available. Not only is the effect insignificant, but the signal is so weak that it might be overpowered by something as insignificant as a cricket ball being dropped near LIGO's mirrors. There are numerous factorsthat can obstruct detection or cause a misleading signal. Detection of these waves is a complex process that requires highly sensitive massive equipment. The computational complexity of typical approaches like matched-filtering and Bayesian inference is a major barrier to real-time detection and parameter estimation of gravitational waves from compact binary mergers. Time-series data from three LIGO Observatories are collected and preprocessed by taking the time difference and alignment into consideration. These signals are then whitened by removing Gaussian and white noise from the signals by applying signal processing filters like Constant-Q transform, Wiener filter, and bandpass filtering. The SNR of these signals is increased. Fourier transform is applied to the time-series data to convert it into the Frequency domain to efficiently figure out the essential frequencies. The waves are then used as training data for Deep Learning models like 1-D CNN, and regression CNN for detection and parameter estimation of the gravitational waves signal in the data. These findings highlight the importance of using realistic gravitational-wave detector data in machine learning approaches and represent an important step toward real-time detection andinference of gravitational waves.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

DL	Deep Learning
CNN	Convoluted Neural Network
ML	Machine Learning
GW	Gravitational Waves
LIGO	Laser Interferometer Gravitational-wave Observatory
BH	Black Hole
BBH	Binary Black Hole
BNS	Binary Neutron Star
NN	Neural Network
ASD	Amplitude Spectral Density
PSD	Power Spectral Density
FT	Fourier Transform
FFT	Fast-Fourier-Transform
DFFT	Discrete-Fast-Fourier-Transform
CQT	Constant Q-Transform
ReLU	Rectified Linear Unit
FC	Fully Connected Layer
RMS	Root Mean Square
DF	Data Frame
UI	User Interface
FS	Frequency Sample
Hz	Hertz
s	Seconds

CHAPTER 1

INTRODUCTION

1.1 Background

i. Einstein’s General Theory of Relativity

Einstein's General Theory of Relativity was his attempt to redefine gravity. He established that the laws of physics apply to all non-accelerating observers, and he demonstrated that the speed of light ina vacuum remains constant regardless of the speed at which an observer travels.

As a result, he came to the conclusion that space-time could be intertwined into a single-continuum known as space-time. Events that happen at the same time for one observer may occur at alternativetimelines for others.

According to this theory, the universe has four dimensions (3 space co-ordinates and a time co- ordinate). It asserts that space and time are inextricably linked to form a space-time continuum. Theuniverse is described as a fabric in this context.

ii. Space-Time continuum

The space-time continuum has four dimensions: three of space (length, width, and height...or, depending on how you want to think of them, up/down, left/right, and forward/backward) and one oftime.

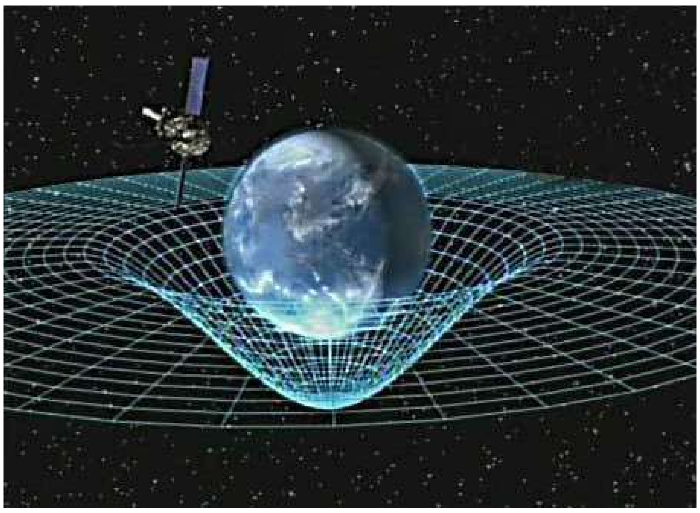


Fig 1.1: Curvature caused in the fabric of space-time by accelerated motion of Earth

iii. Black Holes

Nothing, not even light, can escape from this cosmic body of extreme gravity. Black holes are formed from the death of massive stars. When internal-thermonuclear-fuels run out in its core the stars walls start collapses in. After the fuels are depleted at the end of its life, the core becomes unstable and gravitationally collapses inward upon itself, thus causing the star's outer layers to be blown away. Matterstarts compressing in from all directions and what's left of all of this a singularity that is so potent not even light can escape its gravitational pull.

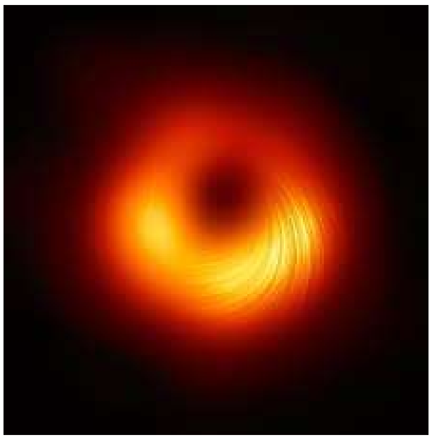


Fig 1.2: First image of a black hole captured in 2018

1.2 Gravitational Waves

Gravitational waves are 'ripples' in space-time caused by some of the Universe's most violent and energetic processes. In his general theory of relativity, Albert Einstein predicted the existence of gravitational waves in 1916. Einstein's mathematics demonstrated that massively accelerating objects(such as neutron stars or black holes orbiting each other) disrupted space-time in such a way that 'waves' of undulating space-time propagated in all directions away from the source. These cosmic ripples would travel at light speed, carrying information about their origins as well as hints about the nature of gravity itself.

Despite Einstein's prediction of gravitational waves in 1916, the first proof of their existence did not arrive until 1974, 20 years after his death. In that same year, two astronomers working at Puerto Rico'sArecibo Radio Observatory discovered a binary pulsar, which is exactly the type of system predicted by general relativity to emit gravitational waves. Knowing that this discovery could be used to put Einstein's audacious prediction to the test, astronomers began measuring how the orbits of the stars changed over time. After eight years of observations, they discovered that the stars were approaching each other at the rate predicted by general relativity if they were emitting gravitational waves.

On September 14, 2015, LIGO detected undulations in spacetime caused by gravitational wavesgenerated by two colliding black holes 1.3 billion light-years away.

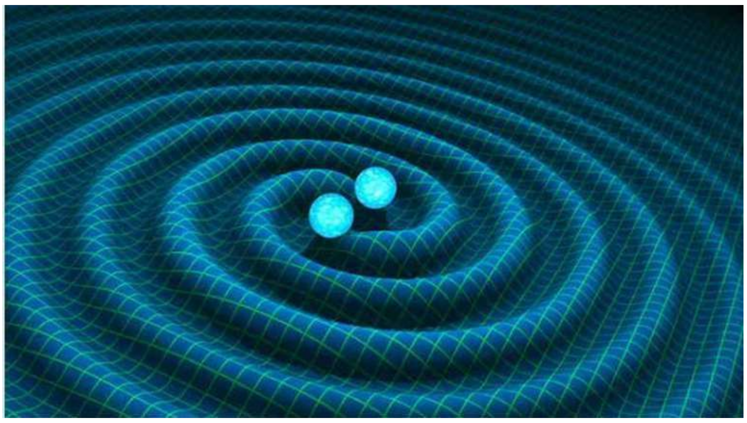
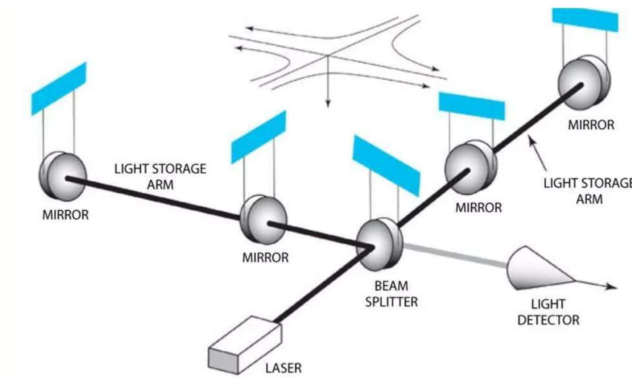


Fig 1.3: Simulation of gravitational waves being emitted from compact binary mergers

1.3 Laser Interferometer Gravitational-Wave Observatory

LIGO is an acronym that stands for "Laser Interferometer Gravitational-wave Observatory." It is a large observatory built for the sole purpose of detecting GW signals. It could be considered to be a marvelouspiece of engineering. LIGO, which is made up of multiple massive laser interferometers spaced 3000 kilometers apart, uses the physical-aspects of light/space itself to make



a detection. It in-turn helps physicists understand the origins of GW and the universe itself.

Fig 1.4: Functioning of LIGO equipment in detecting GW

The LIGO experiment looks for distortions in space-time that indicate the passage of gravitational wave signals through the Earth. A laser beam is split into two 4 km long arms that contain mirrors. The laser beamsreflect off of mirrors and converge at the crux of the arms, causing them to travel different distances. A light detector can detect this mismatch. This time series data from these observatories is used to build a model capable of predicting whether or not a LIGO detector signal contains a gravitational wave.

1.4 Problem Statement

Gravitational waves, despite having been theoretically predicted around 100 years ago, were discovered very recently. The reason for it being the effects experienced by us on Earth is miniscule.

Even current technology is barely sensitive enough to detect them. There is a necessity for highly efficient models that are capable of analyzing large amounts of data and effectively predict the existence of such a wave in a specific data sample.

1.5 Related Work

Related Work There has been use of different signal processing techniques and ML/DL algorithms for detecting these waves. There has been implementation of signal processing techniques like wiener filter, bandpass filter, low pass filter, Deep Learning model to filter out noise from the data like deep filtering, etc. ML algorithms like Decision Tree, SVM's, etc. were also used to classify the data. But these models aren't complex enough to accurately predict what is necessary. Variations in data is great enough to increase false positives and negatives. Thus a combination of these noise filtering techniques are to be used to increase SNRand the data in frequency domain is passed as input to a suitable ML/DL model for prediction

CHAPTER 2

Literature Survey

Ref.no	Title	Overview	Published in	Year of publication
[1]	**Observation of Gravitational Waves from a Binary Black Hole Merger**	The first observation of gravitational waves at LIGO observatories is explained	APS Physics	2016
[2]	**Matching Matched Filtering with Deep Networks for Gravitational-Wave**	Use of matched filtering for detection of gravitational wave is explained	APS Physics	2018
[3]	**Enhancing gravitational-wave science with machine learning**	Different signal processing techniques an waveform modeling is explained	IOP Science	2021
[4]	**Detection and parameter estimation of Gravitational waves using DL**	A regression CNN for detection and 1-D CNN for parameter estimation is explained	ScienceDirect	2021
[5]	**Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data**	Deep Filtering method for noise reduction is provided.	ScienceDirect	2018

Ref.no	Title	Overview	Published in	Year of publication
[6]	**Barlow Twins: Self-Supervised Learning via Redundancy Reduction**	Barlow Twins Self-Supervised Learning Neural network is defined	ARXIV	2018
[7]	**nnAudio: An on-the-Fly GPU Audio to Spectrogram Conversion Toolbox Using 1D Convolutional Neural Networks**	nnAudio Python module for signal processing is explained	IEEE	2019
[8]	**Fourier transform of the continuous gravitational wave signal**	Fourier transform for analyzing the wave in frequency domain so as to get relevant frequency range in which the wave could be present.	APS	2021
[9]	**Insight into the noise reduction Wiener filter**	Wiener filter a signal processing technique for noise reduction is explained.	IEEE	2020
[10]	**Deep learning searches for gravitational wave stochastic backgrounds**	1D and a 2D convolutional neural network (CNN) and a Long Short Term Memory (LSTM) networkfor gravitational wave analysis	IEEE	2021

Ref.no	Title	Overview	Published in	Year of publication
[11]	**From One to Many: A Deep Learning Coincident Gravitational-Wave Search**	In this section, the model uses a complex neural network (NN) trained on the data froma single detector like a single LIGO Observatory and this is used to make a prediction for a GW in a signal	ARXIV	2021
[12]	**PyDub: An on-the-Fly GPU Audio to Spectrogram Conversion Toolbox Using 1D Convolutional Neural Networks**	PyDub Python module for signal processing is explained	IEEE	2019
[13]	**Fourier transform of the continuous gravitational wave signal**	Fourier transform for analyzing the wave in thefrequency domain to get the relevant frequency	APS	2021
[14]	**Insight into the noise reduction Wiener filter**	Wiener filter for noise reduction is explained	IEEE	2020
[15]	**Deep learning searches for gravitational-wave stochastic backgrounds**	1D and a 2D convolutional neural network (CNN) and a Long Short Term Memory (LSTM) network for gravitational wave analysis	IEEE	2021

CHAPTER 3

3 SYSTEM ARCHITECTURE AND DESIGN

The main objective of the project is to develop a ML/DL model that is capable ofpredicting if a LIGO detector signal contains a gravitational wave or not.
The system architecture and modules description is as follows:

3.1 System Architecture

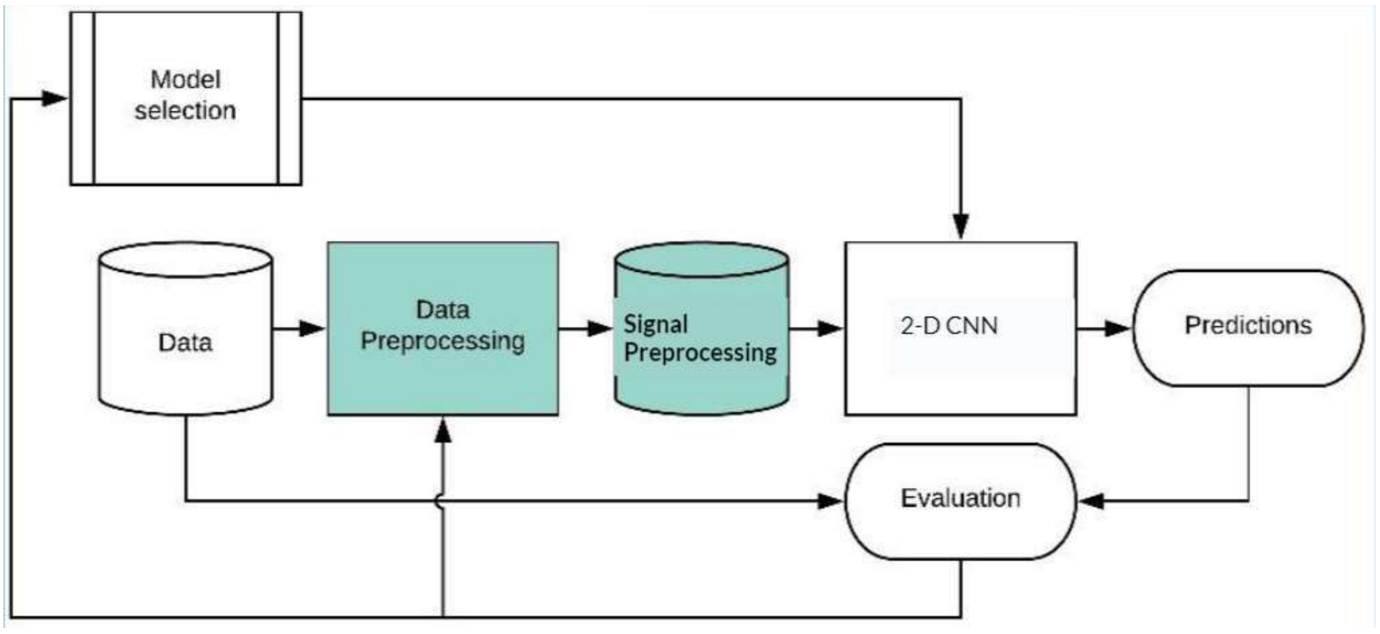


Fig 3.1: Architecture Diagram of the system employed

The system architecture is as follows.

3.1.1 Data:

The dataset used was time-series data gathered from LIGO observatories that contain either detector noise or signal along with detector noise. The signals are extremelyweak having a very less signal to noise ratio. Most of the data is from simulated gravitational waves rather than actual waves. These waves are derived from relativistic equations and use real parameters.
The dataset consists of more than 20000 data samples divided into train and test samplesets that are later used as such.

3.1.2 Data Preprocessing:

The data gathered as mentioned as above are preprocessed to convert it into the format that we could later use. These preprocessing methods involve converting the data into the required format, converting it into frequency domain to analyze the relevantfrequencies, etc.

3.1.3 Signal Preprocessing:

As mentioned above the preprocessed data have a very low SNR. Thus poses a greatobstacle in detecting these underlying signals. To increase the SNR the data is applied upon by a few signal processing techniques. These methods include Wiener filter, whitening the signal, smoothening the signal using methods like centered moving avg,
low-pass filter to filter out unnecessary high frequency signals, etc.

To explain the architecture as a whole, the data is first preprocessed and later signal processing techniques are applied to increase SNR. This data is later passed as input to thesuitable ML/DL model. The models validation metrics like accuracy, precision, etc. are evaluated and compared with other similar models. The model that works the best is chosen as the prediction model.

3.2 2-D CNN Architecture

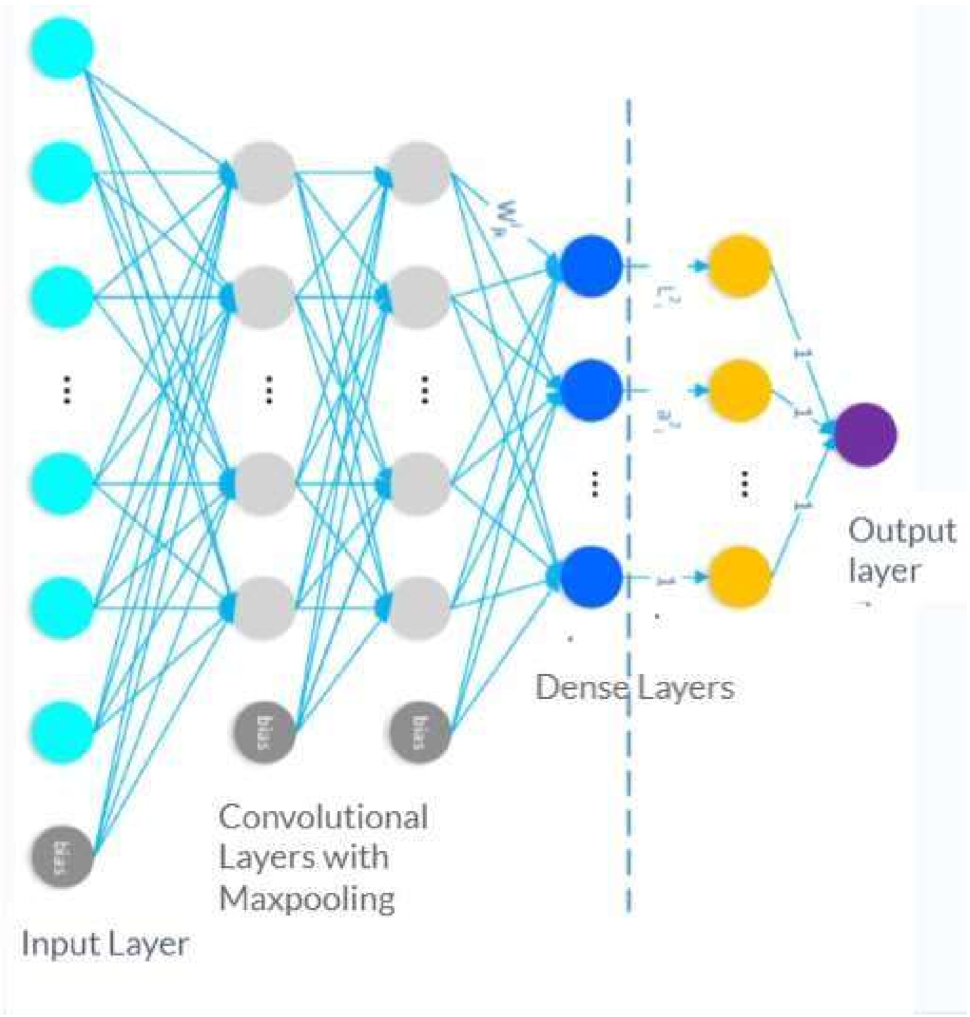


Fig 3.2: CNN architecture diagram used for this project

3.2.1 Basic Architecture

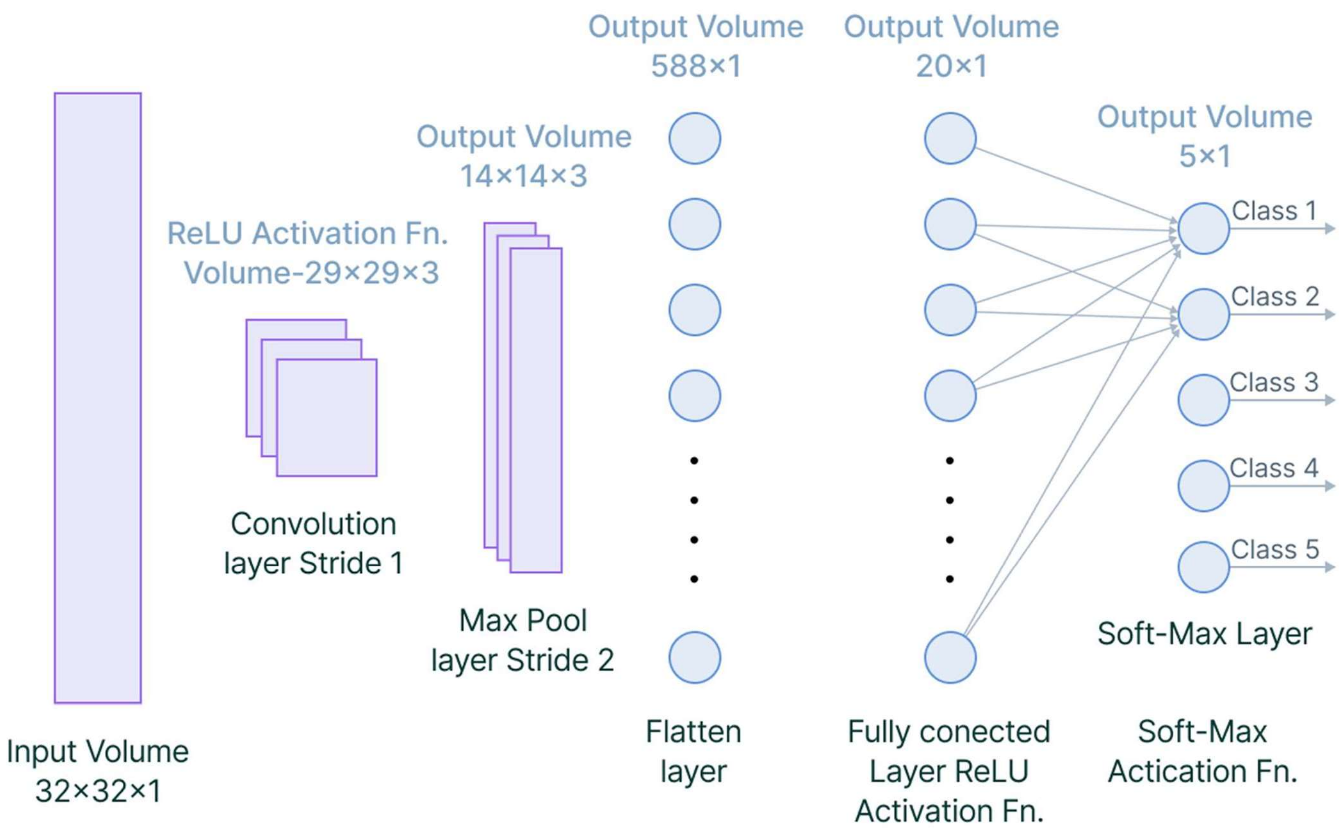


Fig 3.3: Basic Architecture of a CNN

A CNN architecture is made up of two major components:

- A convolution function that is capable of separating/identifying the feature vector of atime-series data for analysis, which in a process is called Feature Extraction

- A FC layer that makes use of the o/p of the process to make predictions for the class ofthe provided data on basis of the features that were extracted in previous stages of the process.

Convolution-layers, max-pooling layers, and FC-layers comprise the CNN. These layers are stacked over and over to form a complex/dense CNN architecture. In addition to theselayers, two additional estimators must be considered: the dropout-layer and the activation-function which are provided later on.

3.2.2 Convolutional Layer

This is the first layer that is used to extract the various features from the input images. Inthis layer, the mathematical operation of convolution is performed between the input image and a filter of size. By swiping the filter over the input image, the dot product is calculated between the filter and the parts of the input image that are proportional to thesize of the filter.

The result is known as the Feature map, and it contains information about the image such as the corners and edges. Later, this feature map is fed to other layers, which learn severalother features of the input image.

3.2.3 Pooling Layer

A Convolutional Layer is usually followed by a Pooling Layer. This layer's primary goal isto reduce the size of the convolved feature map in order to reduce computational costs.
This is accomplished by reducing the connections between layers and operating independently on each feature map. Pooling operations are classified into several typesbased on the method used.
The largest element from the feature map is used in Max Pooling. Average Pooling computes the average of the elements in a predefined image section size. Sum Poolingcomputes the total sum of the elements in the predefined section. The Pooling Layer iscommonly used as a link between the Convolutional Layer and the FC Layer.

3.2.4 Fully Connected Layer

The Fully Connected (FC) layer, which includes weights and biases as well as neurons, is used to connect neurons from different layers. These layers are typically placed prior to theoutput layer and constitute the final few layers of a CNN Architecture.
The previous layers' input images are flattened and fed to the FC layer in this step. The flattened vector is then passed through a few more FC layers, where the mathematical function operations are typically performed. At this point, the classification procedure isinitiated.

3.2.5 Dropout

When all of the features are connected to the FC layer, the training dataset is prone to overfitting. Overfitting occurs when a model performs so well on training data that it has anegative impact on the model's performance when applied to new data.
To address this issue, a dropout layer is used, in which a few neurons are removed fromthe neural network during the training process, resulting in a smaller model. When a dropout of 0.3 is reached, 30% of the nodes in the neural network are dropped out at random.

3.2.6 Activation Functions

Finally, the activation function is a critical parameter in the CNN model. They are used to learn and approximate any type of continuous and complex relationship between network variables. In other words, it determines which model information should be fired forward and which should not at the network's end.

It introduces nonlinearity into the network. There are several activation functions that are commonly used, including the ReLU, Softmax, tanH, and Sigmoid functions. Each of these functions has a specific application. The sigmoid and softmax functions are preferred for a binary classification CNN model, and softmax is generally used for a multi-classclassification.

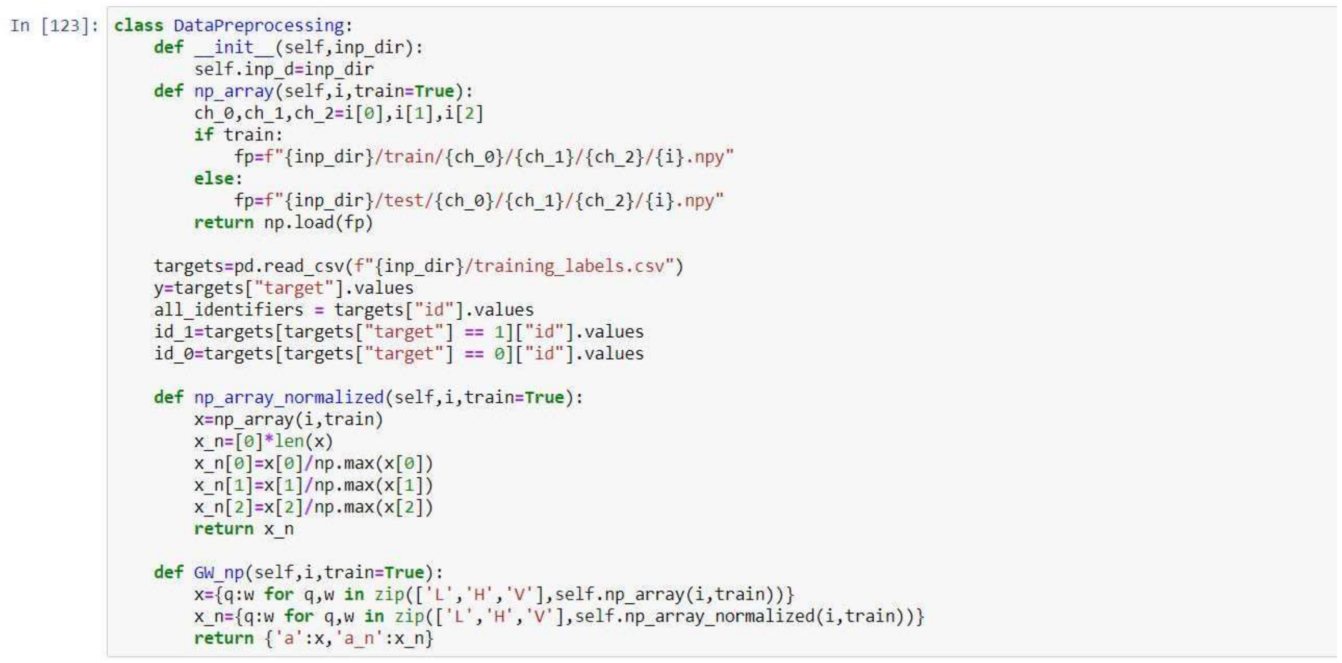
3.3 Modules Description

3.3.1 Data Preprocessing

Dataset containing time series data from three LIGO observatories aregathered.

HDF5 files are converted to numpy arrays and are sorted into their respective 3Observatories.

Data is normalized and categorized.



```
In [123]: class DataPreprocessing:
              def __init__(self,inp_dir):
                  self.inp_d=inp_dir
              def np_array(self,i,train=True):
                  ch_0,ch_1,ch_2=i[0],i[1],i[2]
                  if train:
                      fp=f"{inp_dir}/train/{ch_0}/{ch_1}/{ch_2}/{i}.npy"
                  else:
                      fp=f"{inp_dir}/test/{ch_0}/{ch_1}/{ch_2}/{i}.npy"
                  return np.load(fp)

              targets=pd.read_csv(f"{inp_dir}/training_labels.csv")
              y=targets["target"].values
              all_identifiers = targets["id"].values
              id_1=targets[targets["target"] == 1]["id"].values
              id_0=targets[targets["target"] == 0]["id"].values

              def np_array_normalized(self,i,train=True):
                  x=np_array(i,train)
                  x_n=[0]*len(x)
                  x_n[0]=x[0]/np.max(x[0])
                  x_n[1]=x[1]/np.max(x[1])
                  x_n[2]=x[2]/np.max(x[2])
                  return x_n

              def GW_np(self,i,train=True):
                  x={q:w for q,w in zip(['L','H','V'],self.np_array(i,train))}
                  x_n={q:w for q,w in zip(['L','H','V'],self.np_array_normalized(i,train))}
                  return {'a':x,'a_n':x_n}
```

3.3.2 Signal Processing

Multiple signal processing methods (SQT, wiener filters, bandpass filteringetc)are described.

This is to remove gaussian and white noise from the signals to increase SNR

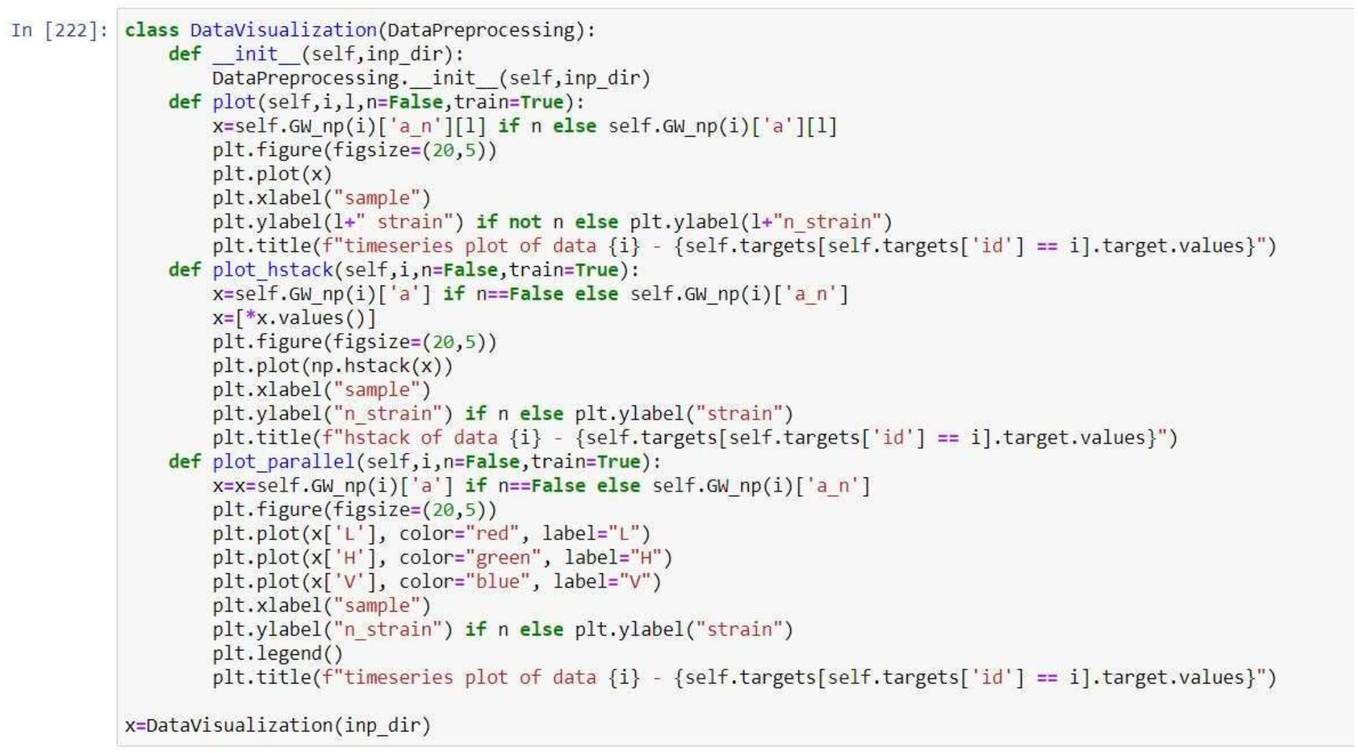
The data is aligned and rotated according to the orientations of theobservatories.

3.3.3 Data Visualization

Here the time series data is plotted on a graph to better understand the signals.

The signals are plotted individually stacked horizontally and overlapped overeach other

The signals are converted to frequency domain by performing Fourier transform on the waves to understand the frequency range of the signal.



```
In [222]: class DataVisualization(DataPreprocessing):
              def __init__(self,inp_dir):
                  DataPreprocessing.__init__(self,inp_dir)
              def plot(self,i,l,n=False,train=True):
                  x=self.GW_np(i)['a_n'][l] if n else self.GW_np(i)['a'][l]
                  plt.figure(figsize=(20,5))
                  plt.plot(x)
                  plt.xlabel("sample")
                  plt.ylabel(l+" strain") if not n else plt.ylabel(l+"n_strain")
                  plt.title(f"timeseries plot of data {i} - {self.targets[self.targets['id'] == i].target.values}")
              def plot_hstack(self,i,n=False,train=True):
                  x=self.GW_np(i)['a'] if n==False else self.GW_np(i)['a_n']
                  x=[*x.values()]
                  plt.figure(figsize=(20,5))
                  plt.plot(np.hstack(x))
                  plt.xlabel("sample")
                  plt.ylabel("n_strain") if n else plt.ylabel("strain")
                  plt.title(f"hstack of data {i} - {self.targets[self.targets['id'] == i].target.values}")
              def plot_parallel(self,i,n=False,train=True):
                  x=x=self.GW_np(i)['a'] if n==False else self.GW_np(i)['a_n']
                  plt.figure(figsize=(20,5))
                  plt.plot(x['L'], color="red", label="L")
                  plt.plot(x['H'], color="green", label="H")
                  plt.plot(x['V'], color="blue", label="V")
                  plt.xlabel("sample")
                  plt.ylabel("n_strain") if n else plt.ylabel("strain")
                  plt.legend()
                  plt.title(f"timeseries plot of data {i} - {self.targets[self.targets['id'] == i].target.values}")

          x=DataVisualization(inp_dir)
```

3.3.4 Prediction Module

Have designed a 2-D CNN with three convolution layers each followed by amax pooling layer and three dense layers two of them with RELU activation and the last one with a sigmoid activation so that the output layer consists of a value in the range of [0,1] indicating the probability of the signal containing a gravitational wave signal.

CHAPTER 4

Methodology

For this project we've used the data set from Kaggle. The data set used has time-series data obtained from three LIGO (Laser Interferometer Gravitational-Wave Observatory) namely LIGO Hanford, LIGO Livingston, VIRGO. These three observatories are sampled at 4096Hz for 2s. These data are either a combination of detector noise or detector noise + GW signal.

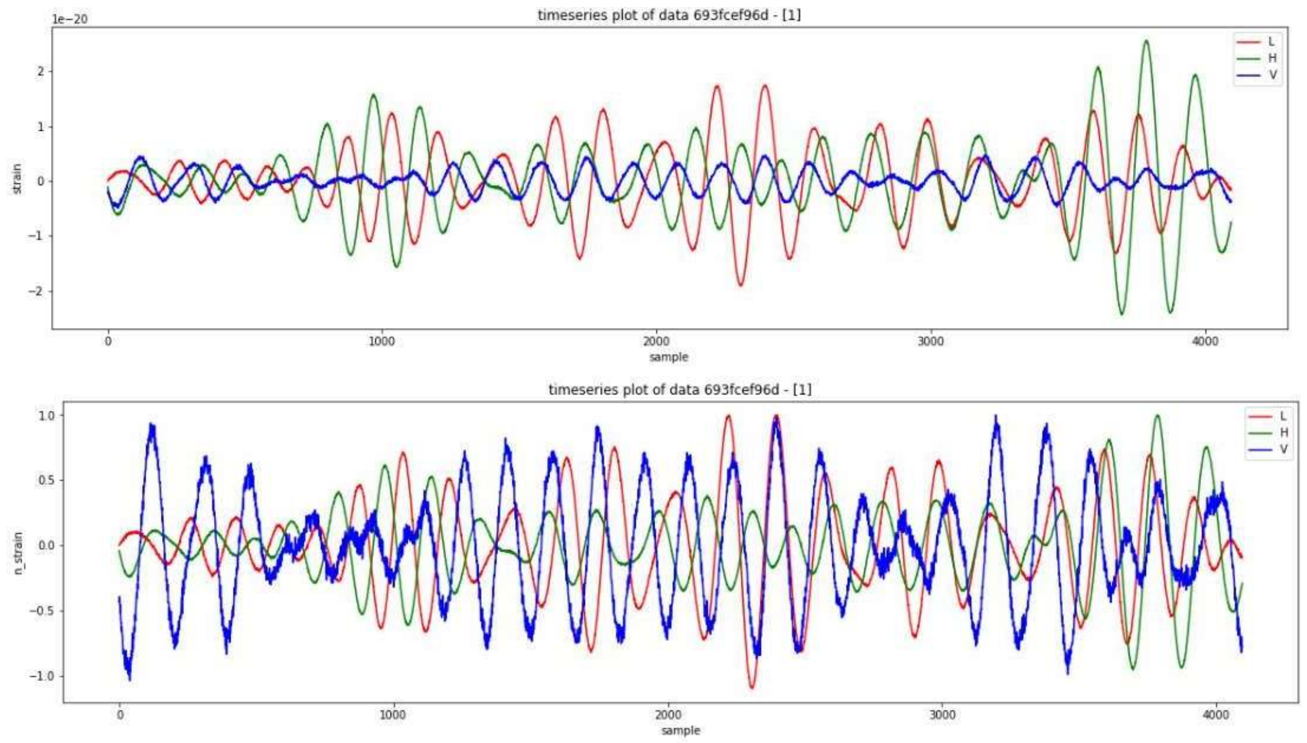


Fig.4.1: Data strains for normal vs normalized plotted for three observatories in time domain

The signal strength for most signals is very low for which we use multiple signal processing algorithms like whiten to remove gaussian noise, a low pass filter to filter out irrelevant higher frequencies. CQT filter is later applied on it and is passed as input to the 2-D CNN that we designed.

4.1 Discrete Fast Fourier Transform

Discrete-fast-Fourier-transform(DFFT) is a method that is used to convert discrete time-series data-signals to discrete frequency-domain signals. (DFFT) is used to calculate the frequency-spectrum of a signal.

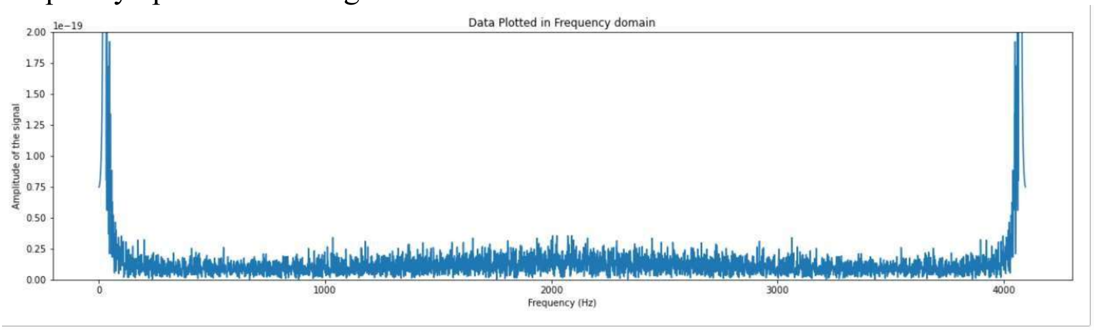


Fig.4.2: LIGO strain transformed to frequency domain

4.2 Power Spectral Density (PSD)

The Power Spectral Density (PSD) of a signal is the ratio of its power level to its frequency (Hz). PSDs are frequently used to characterise randomised broadband signals. The spectral resolution used to digitise the signal normalises the PSD's amplitude.

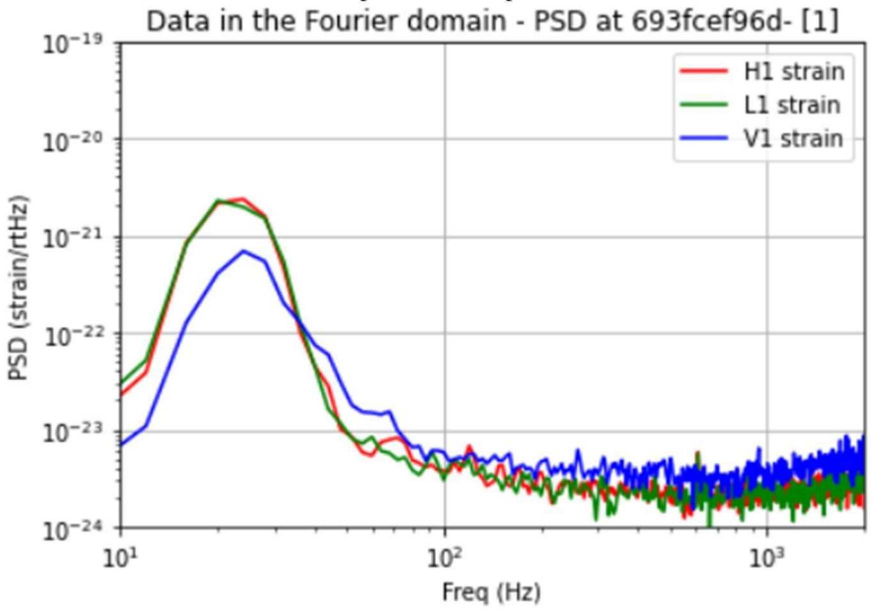


Fig.4.3: PSD for a sample signal

4.3 Amplitude Spectral Density (ASD)

The ASDs (Amplitude Spectral Density) are the square root of the PSD, which are the averages of the squares of the data's FFTs. They are an estimate of the detectors' "strain-equivalent noise" versus frequency, which limits the detectors' ability to identify GW signals.

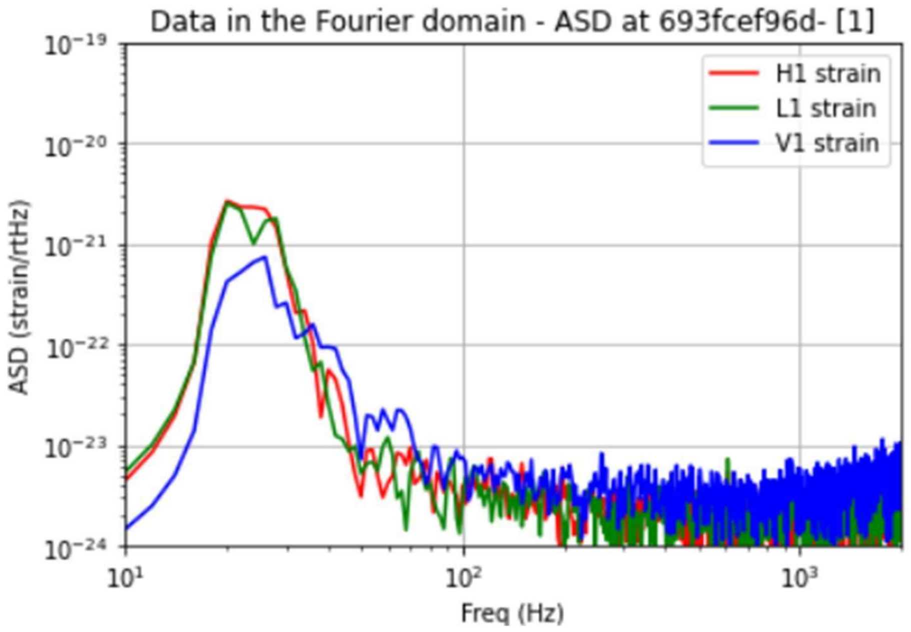


Fig.4.4: ASD for a sample signal

4.4 Low-Pass Filter

A low-pass filter is used to attenuate the signals that supersede the predetermined cutoff frequency. It is a concept taken from electronics and communication systems. It allows the lower frequencies to pass through while blocking the higher frequency signals. The response of this filter is determined during its initial design.

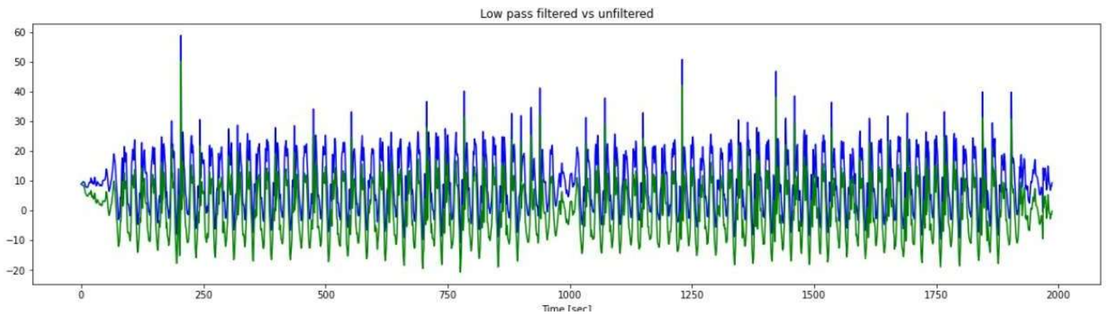


Fig 4.5: Lowpass filtered vs unfiltered plotted parallelly

4.5 Whitten

It is a linear transform that converts random variable vectors with a specified co-variance matrix to a set of new variables whose co-variance matrix is I. Thus implying each has a specific variance. This function is used to remove Gaussian or white noise from detector signal data.

4.6 CQT Filtered Signal

The constant-Q transform, also known as CQT, converts time-series data to the frequency domain.
The transform can be thought of as a series of logarithmically spaced filters-fk, with the k-th-filter having a spectral width of sigma-fk, which is equal to a multiple of the width of the previous-filter:

$$\delta f_k = 2^{1/n} \cdot \delta f_{k-1} = \left(2^{1/n}\right)^k \cdot \delta f_{\min},$$

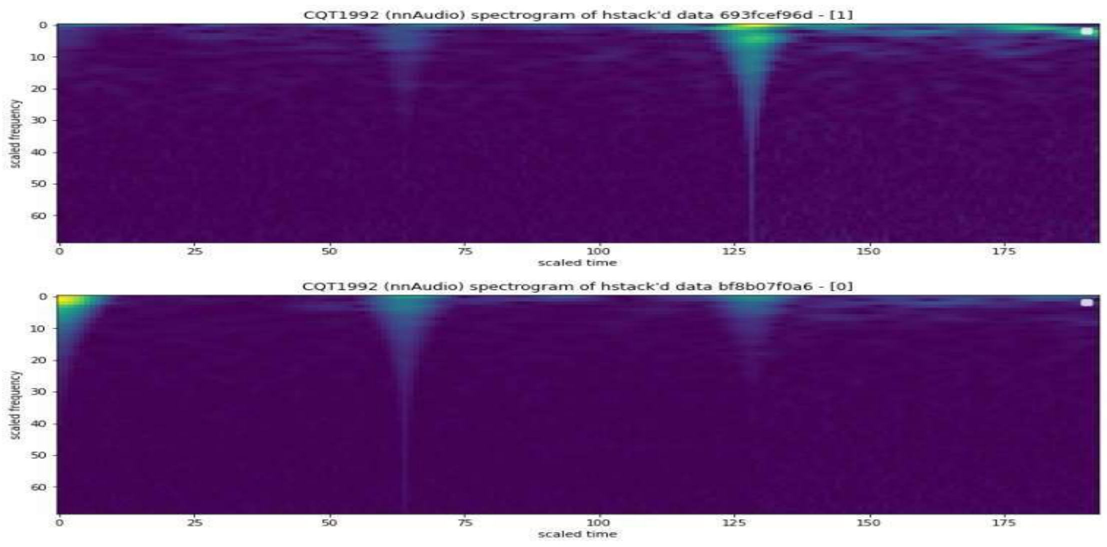


Fig.4.6: CQT Filtered Signal

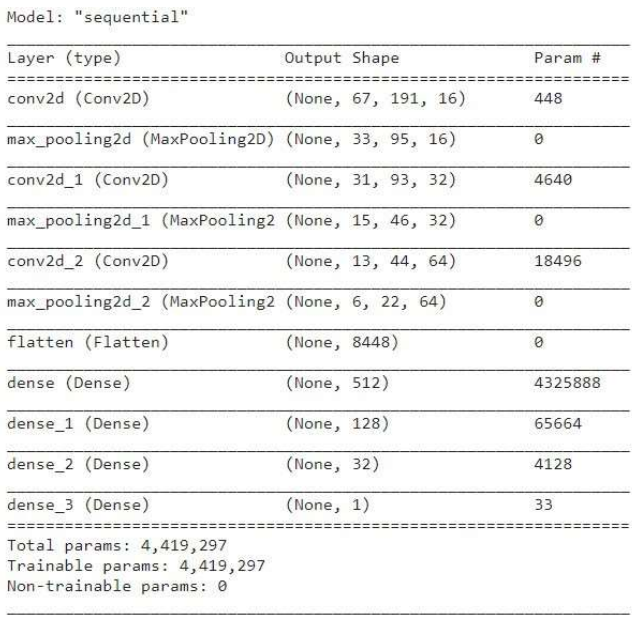
4.7 DL Model Used

We've used a 2-D CNN for this. The 2-D CNN takes the preprocessed CQT passed LIGO data of shape (1,69,193,3). It outputs a regression value in the range [0,1].

The model has 3 Convolution layers of filter sizes (16,32,64). Max pooling is applied on every Conv layer to extract features from the data.

The outputs from Conv layers are flattened and passed as input to dense layers which reduce the output vector size. ReLu is used on the last but one dense layer. While Sigmoid activation is used to get a value in the range of [0,1].

To compare the performances of the model we have compared it against a 2-D CNN that also uses pre-trained weights from Efficient net. But this model outperformed it when it was compared for 1 epoch. This efficient net model had an accuracy of 52.85%.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 67, 191, 16)       448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 33, 95, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 31, 93, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 46, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 44, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 22, 64)         0
_________________________________________________________________
flatten (Flatten)            (None, 8448)              0
_________________________________________________________________
dense (Dense)                (None, 512)               4325888
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 32)                4128
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 4,419,297
Trainable params: 4,419,297
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.7: 2-D CNN layers

The model also had a comparatively higher accuracy than multiple - epoch-ed versions and 1-D CNN .

CHAPTER 5

Coding and Testing

5.1 Raw Code

5.1.1 Import the req libraries

Imported the libraries that we would be using for this project

```
import pandas as pd


import math
import numpy as np
import random


from tensorflow.keras.utils import Sequence
from tensorflow.keras import Sequential, utils, optimizers, metrics
import tensorflow as tf
import tensorflow.keras.layers as layers
from tensorflow.keras.models import load_model


import matplotlib
matplotlib.use('nbagg')


from matplotlib import pyplot as plt,mlab as mlb


from sklearn.model_selection import train_test_split


from random import shuffle
from nnAudio.Spectrogram import CQT1992v2
import torch



from efficientnet.tfkeras import EfficientNetB0, EfficientNetB1


from scipy import signal, fft
from scipy.interpolate import interp1d
from scipy.signal import butter, filtfilt, iirdesign, zpk2tf, freqz


from scipy.signal import welch


import os
import time
import seaborn as sns
from copy import copy,deepcopy
```

5.1.2 Data Preprocessing module

Has functions to transform the data as required.

```
class Data:
  def __init_(self,in_dir):
    self.inp_d=in_dir
  def np_array(self,i,train=True):
    c_0,c_1,c_2=[i[0],i[1],i[2]]
    if train:
      fp=f"{config.TRAIN_ROOT}{c_0}/{c_1}/{c_2}/{i}.npy"
    else:
      fp=f"{config.TEST_ROOT}{c_0}/{c_1}/{c_2}/{i}.npy"
```

```
        return np.load(fp)
    targets=pd.read_csv(f"{config.inp_dir}/training_labels.csv")
    y=targets["target"].values
    all_identifiers = targets["id"].values
    id_1=targets[targets["target"] == 1]["id"].values
    id_0=targets[targets["target"] == 0]["id"].values
    sample_submission=pd.read_csv(f"{config.inp_dir}/sample_submission.csv")
def np_array_normalized(self,i,train=True):
        x_=self.np_array(i,train)
        x_n_=[0]*len(x)
        x_n_[0]=x_[0]/np.max(x_[0])
        x_n[1]=x_[1]/np.max(x_[1])
        x_n[2]=x_[2]/np.max(x_[2])
        return x_n_
def GW_np(self,i,train=True):
        x_={q:w for q,w in zip(['L','H','V'],self.np_array(i,train))}
        x_n_={q:w for q,w in zip(['L','H','V'],self.np_array_normalized(i,train))}
        return {'a':x_,'a_n':x_n_}
def cqt_spectrogram(self,i,train=True):
        cqt = CQT1992v2(sr=config.s_freq,hop_length=64, fmin=20, fmax=1024, bins_per_octave=12,
norm=1, window='hann', center=True, pad_mode='reflect', trainable=False,
output_format='Magnitude', verbose=False)
        waveform=np.hstack(self.np_array(i,train))
        waveform=waveform/np.max(waveform)
        waveform=torch.from_numpy(waveform).float()
        cqt_image=cqt(waveform)
        cqt_image=np.array(cqt_image)
        cqt_image=np.transpose(cqt_image, (1,2,0))
        return cqt_image
def spectrogram(self,i,train=True):
        waveform=np.hstack(self.np_array(i,train))
        waveform=waveform/np.max(waveform)
        (freq,time,intensity)=signal.spectrogram(waveform,config.s_freq, mode="magnitude",
 scaling="spectrum", window=('kaiser', 14))
        return (freq,time,intensity)
```

5.1.3 Data Visualization Module

Has functions to plot the data in graphs and visualize it. It is used to analyze the data to identify relevant properties that could worked on

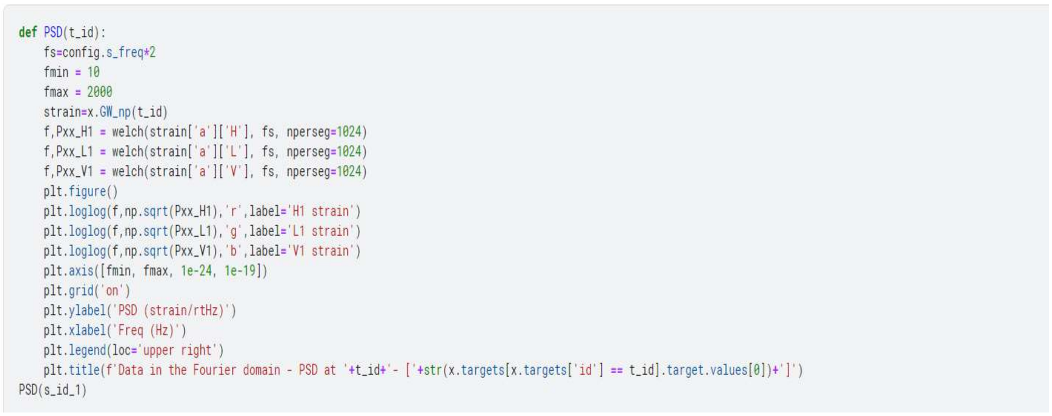
```
class DataVisualization(Data):
    def__init_(self,in_dir):
        Data.__init__(self,in_dir)
    def plot(self,i,l,n=False,train=True):
        x=self.GW_np(i)['a_n'][l] if n else self.GW_np(i)['a'][l]
        plt.figure(figsize=(20,5))
        plt.plot(x)
        plt.xlabel("sample")
        plt.ylabel(l+" strain") if not n else plt.ylabel(l+"n_strain")
        plt.title(f"timeseries plot of data {i} - {self.targets[self.targets['id']==i].target.values}")
 def plot_hstack(self,i,n=False,train=True):
        x=self.GW_np(i)['a'] if n==False else self.GW_np(i)['a_n']
        x=[*x.values()]
        plt.figure(figsize=(20,5))
        plt.plot(np.hstack(x))
        plt.xlabel("sample")
        plt.ylabel("n_strain") if n else plt.ylabel("strain")
        plt.title(f"hstack of data {i} - {self.targets[self.targets['id']==i].target.values}")
def plot_parallel(self,i,n=False,train=True):
        x=self.GW_np(i)['a'] if n==False else self.GW_np(i)['a_n']
        plt.figure(figsize=(20,5))
        plt.plot ( x['L'], color="red", label="L")
        plt.plot ( x['H'], color="green", label="H")
        plt.plot ( x['V'], color="blue", label="V")
        plt.xlabel ("sample")
        plt.ylabel ("n_strain") if n else plt.ylabel("strain")
        plt.legend()
plt.title(f"timeseries plot of data {i} - {self.targets[self.targets['id']==i].target.values}")
def plot_cqt_spectrogram(self,i,train=True):
        image=self.cqt_spectrogram(i,train)
        plt.figure(figsize=(20,5))
        plt.imshow(image)
        plt.xlabel("scaled time")
        plt.ylabel("scaled frequency")
        plt.legend()
        plt.title(f"CQT1992 (nnAudio) spectrogram of hstack'd data {i} -
{self.targets[self.targets['id']==i].target.values}")
    def plot_spectrogram(self,i,train=True):
        (f,t,it)=self.spectrogram(i,train)
        plt.figure(figsize=(20,5))
        plt.pcolormesh(t,f,it,shading='gouraud')
        plt.ylabel('Frequency [Hz]')
        plt.xlabel('Time [sec]')
        plt.ylim(0,100)
        plt.title(f"Scipy spectrogram of hstack'd data {i} - 
{self.targets[self.targets['id']==i].target.values}")


x=DataVisualization(config.inp_dir)
```

5.1.4 Signal Processing

Has functions to reduce noise from the signal data and improve its SNR

5.1.4.1 PSD

```
def PSD(t_id):
   fs=config.s_freq*2
   fmin = 10
  fmax = 2000
   strain=x.GW_np(t_id)
   f,Pxx_H1 = welch (strain['a']['H'], fs, nperseg=1024)
   f,Pxx_L1 = welch (strain['a']['L'], fs, nperseg=1024)
   f,Pxx_V1 = welch (strain['a']['V'], fs, nperseg=1024)
   plt.figure()
   plt.loglog (f,np.sqrt (Pxx_H1), 'r',label='H1 strain')
   plt.loglog (f,np.sqrt (Pxx_L1), 'g',label='L1 strain')
   plt.loglog (f,np.sqrt (Pxx_V1), 'b',label='V1 strain')
   plt.axis ([fmin, fmax, 1e-24, 1e-19])
   plt.grid ('on')
   plt.ylabel ('PSD (strain/rtHz)')
   plt.xlabel ('Freq (Hz)')
   plt.legend (loc='upper right')
   plt.title (f'Data in the Fourier domain - PSD at '+t_id+'- ['+str(x.targets[x.targets['id'] == 
t_id].target.values[0])+']')
PSD(s_id_1)
```



```python
def PSD(t_id):
    fs=config.s_freq*2
    fmin = 10
    fmax = 2000
    strain=x.GW_np(t_id)
    f,Pxx_H1 = welch(strain['a']['H'], fs, nperseg=1024)
    f,Pxx_L1 = welch(strain['a']['L'], fs, nperseg=1024)
    f,Pxx_V1 = welch(strain['a']['V'], fs, nperseg=1024)
    plt.figure()
    plt.loglog(f,np.sqrt(Pxx_H1),'r',label='H1 strain')
    plt.loglog(f,np.sqrt(Pxx_L1),'g',label='L1 strain')
    plt.loglog(f,np.sqrt(Pxx_V1),'b',label='V1 strain')
    plt.axis([fmin, fmax, 1e-24, 1e-19])
    plt.grid('on')
    plt.ylabel('PSD (strain/rtHz)')
    plt.xlabel('Freq (Hz)')
    plt.legend(loc='upper right')
    plt.title(f'Data in the Fourier domain - PSD at '+t_id+'- ['+str(x.targets[x.targets['id'] == t_id].target.values[0])+']')
PSD(s_id_1)
```

Fig 5.1 PSD implementation

5.1.4.1 ASD

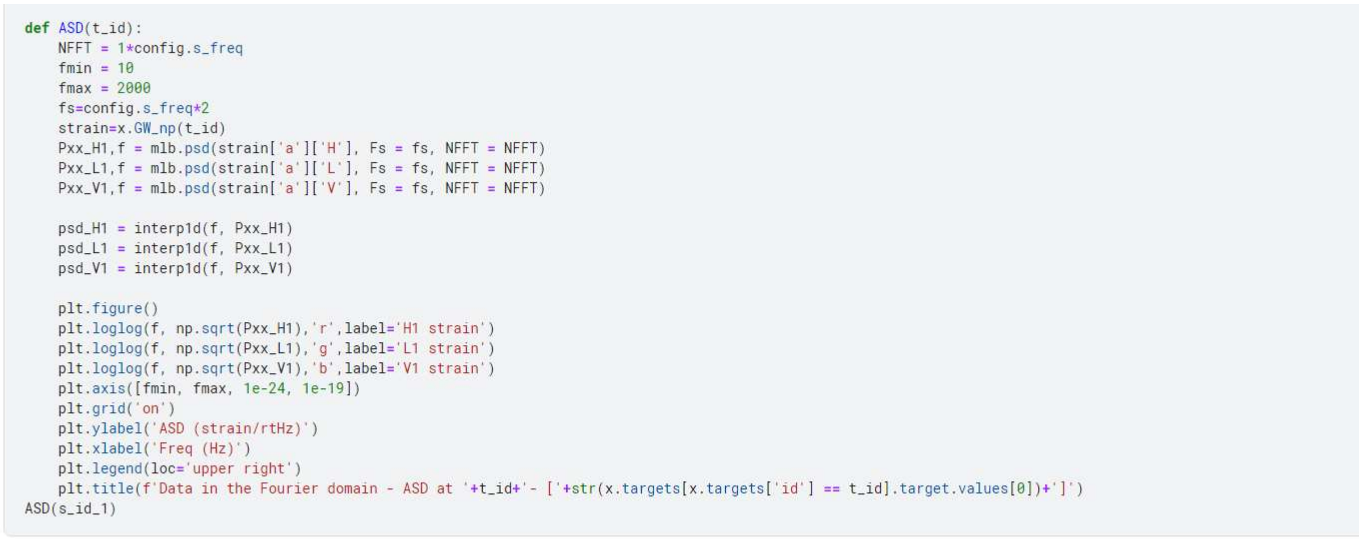
```
def ASD(t_id):
    NFFT = 1*config.s_freq
    fmin = 10
    fmax = 2000
    fs=config.s_freq*2
    strain=x.GW_np(t_id)
    Pxx_H1,f = mlb.psd ( strain ['a'] ['H'], Fs = fs, NFFT = NFFT)
  Pxx_L1,f = mlb.psd ( strain ['a'] ['L'], Fs = fs, NFFT = NFFT)
  Pxx_V1,f = mlb.psd ( strain ['a'] ['V'], Fs = fs, NFFT = NFFT)
  psd_H1 = interp1d ( f, Pxx_H1)
    psd_L1 = interp1d ( f, Pxx_L1)
    psd_V1 = interp1d ( f, Pxx_V1)
    plt.figure()
    plt.loglog (f, np.sqrt(Pxx_H1),'r',label='H1 strain')
    plt.loglog (f, np.sqrt(Pxx_L1),'g',label='L1 strain')
    plt.loglog (f, np.sqrt(Pxx_V1),'b',label='V1 strain')
    plt.axis ([fmin, fmax, 1e-24, 1e-19])
    plt.grid ('on')
    plt.ylabel ('ASD (strain/rtHz)')
```

plt.xlabel ('Freq (Hz)')

plt.legend (loc='upper right')

plt.title (f'Data in the Fourier domain - ASD at '+t_id+'- ['+str(x.targets[x.targets['id'] == t_id].target.values[0])+']')

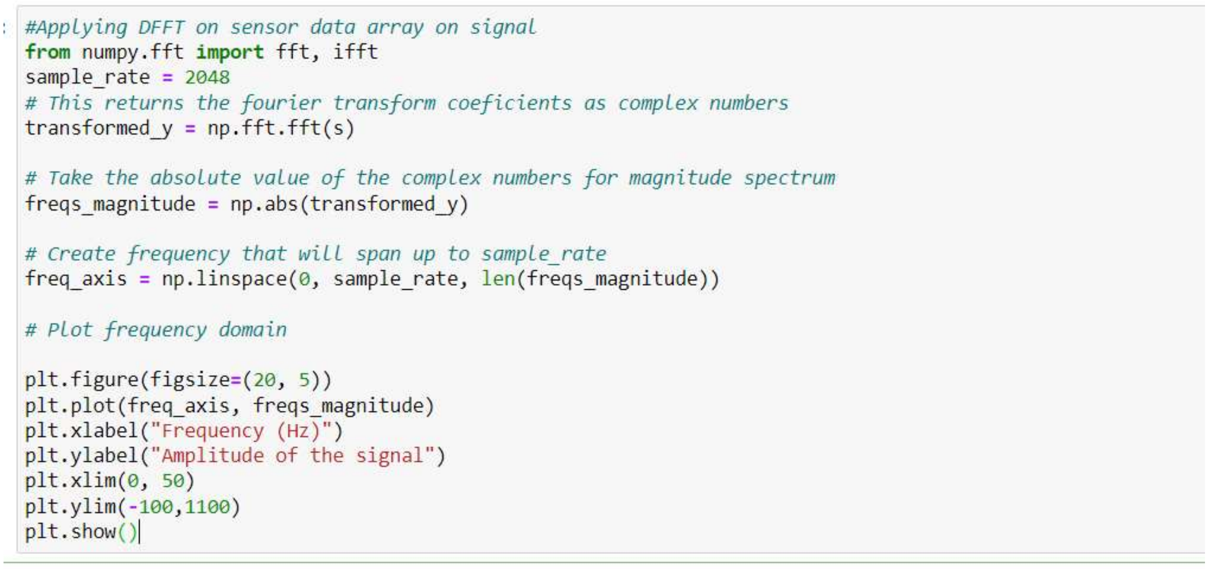
ASD(s_id_1)



```
def ASD(t_id):
    NFFT = 1*config.s_freq
    fmin = 10
    fmax = 2000
    fs=config.s_freq*2
    strain=x.GW_np(t_id)
    Pxx_H1,f = mlb.psd(strain['a']['H'], Fs = fs, NFFT = NFFT)
    Pxx_L1,f = mlb.psd(strain['a']['L'], Fs = fs, NFFT = NFFT)
    Pxx_V1,f = mlb.psd(strain['a']['V'], Fs = fs, NFFT = NFFT)

    psd_H1 = interp1d(f, Pxx_H1)
    psd_L1 = interp1d(f, Pxx_L1)
    psd_V1 = interp1d(f, Pxx_V1)

    plt.figure()
    plt.loglog(f, np.sqrt(Pxx_H1),'r',label='H1 strain')
    plt.loglog(f, np.sqrt(Pxx_L1),'g',label='L1 strain')
    plt.loglog(f, np.sqrt(Pxx_V1),'b',label='V1 strain')
    plt.axis([fmin, fmax, 1e-24, 1e-19])
    plt.grid('on')
    plt.ylabel('ASD (strain/rtHz)')
    plt.xlabel('Freq (Hz)')
    plt.legend(loc='upper right')
    plt.title(f'Data in the Fourier domain - ASD at '+t_id+'- ['+str(x.targets[x.targets['id'] == t_id].target.values[0])+']')
ASD(s_id_1)
```

Fig 5.2: PSD implimentation

5.1.4.1 DFFT



```
#Applying DFFT on sensor data array on signal
from numpy.fft import fft, ifft
sample_rate = 2048
# This returns the fourier transform coeficients as complex numbers
transformed_y = np.fft.fft(s)

# Take the absolute value of the complex numbers for magnitude spectrum
freqs_magnitude = np.abs(transformed_y)

# Create frequency that will span up to sample_rate
freq_axis = np.linspace(0, sample_rate, len(freqs_magnitude))

# Plot frequency domain

plt.figure(figsize=(20, 5))
plt.plot(freq_axis, freqs_magnitude)
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude of the signal")
plt.xlim(0, 50)
plt.ylim(-100,1100)
plt.show()
```

Fig 5.

```
#Applying DFT on the signal to convert it to frequency domain
def dfft(s):

    from numpy.fft import fft, ifft

    sample_rate = 4096

    # This returns the fourier transform coeficients as complex numbers

    transformed_s = np.fft.fft(s[2])

    # Take the absolute value of the complex numbers for magnitude spectrum

    freqs_magnitude = np.abs(transformed_y)

# Create frequency that will span up to sample_rate

    freq_axis = np.linspace(0, sample_rate, len(freqs_magnitude))

    # Plot frequency domain

    plt.figure(figsize=(20,  5))

    plt.plot(freq_axis, freqs_magnitude)

    plt.title('Data Plotted in Frequency domain')

    plt.xlabel("Frequency (Hz)")

    plt.ylabel("Amplitude of the signal")

    plt.ylim(0,.2e-18)

    plt.show()
```

5.1.4.1 Low Pass Filter

```
def b_low_pass(c, fs, o):

    nyq=0.5*fs

    n_c=c/nyq

    return butter(o, n_c, btype='low', analog=False)



def b_low_pass_filter(d, c, fs, o):

    return lfilter(*butter_lowpass(c, fs, o=o),d)

def lowpass(r_a_d):
    fs=4096
    nyq=0.5*fs
    c=2.24e-25
    n_c=c/nyq
    o=1
    #'f' for filtered signal and 'r' for removed signal
    #numpy arrays are used for this
return {'f':b_low_pass_filter(r_a_d,c,fs,o),'r':r_a_d-b_low_pass_filter(r_a_d,c,fs,o)}
```

5.1.4.1 Whitten

```
def whiten(s, interp_psd, dt):
    fq=np.fft.rfftfreq(len(s),dt)
    FT=np.fft.rfft(s)
    w_FT=FT/(np.sqrt(interp_psd(fq)/dt/2))
    w_FT=np.fft.irfft(w_FT,n=len(s))
    return w_FT
```

5.1.4.6 CQT transform

```
def cqt_1992(d):
    d_c=np.hstack(d)
    d_c=np.hstack(d)/np.max(d)
    spectrogram_creator=CQT1992v2(
        sr=2048, hop_length=64, fmin=20, fmax=1024, verbose=False
    )
    im=spectrogram_creator(torch.from_numpy(d_c).float())[0]
    r=np.stack([im,im,im],axis = -1)
    return r


f=cqt_1992
```

5.1.5 Data Generator

This module is to convert the dataset into the required form before passing the data as input to the DL model.

```
class DataGenerator_(tf.keras.utils.Sequence):
    def__init_(self,path_,list_ID_,d,batch_size):
        self.path_=path_
        self.list_ID_=list_ID_
        self.d=d
        self.batch_size=batch_size
        self.indexes=np.arange(len(self.list_ID_))

   def __len_(self):
      len_=int(len(self.list_ID_)/self.batch_size)
      if len_*self.batch_size<len(self.list_ID_):
         len_+=1
      return len_

   def __data_generation(self, list_IDs_temp):
      X=np.zeros((self.batch_size,l,w,h))
      y=np.zeros((self.batch_size, 1))
      for i,ID in enumerate(list_IDs_temp):
         id_=self.data.loc[ID, 'id']
         file=id_+".npy"
         path_in='/'.join([self.path, id_[0], id_[1], id_[2]]) + '/'
         data_array=np.load(path_in+file)
        X[i, ]=f(data_array)
        y[i, ]=self.data.loc[ID, 'target']
    return X,y

  def___getitem__(self, id):
    indexes = self.indexes[id * self.batch_size : (id+1) * self.batch_size]
    list_ID_temp = [self.list_ID_[k] for k in indexes]
     X, y = self._data_generation(list_ID_temp)
    return X, y
```

5.1.6 2-D CNN model

This is the 2-D CNN that was trained to classify a LIGO signal as detector noise or detector noise + GW signal

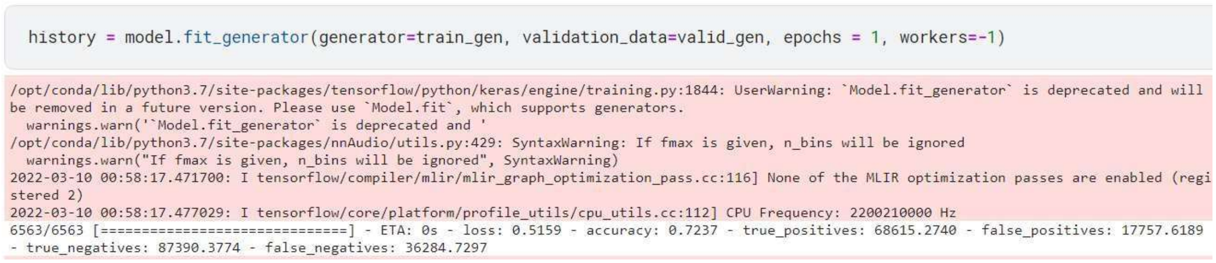
```
def get_model():
  model = tf.keras.Sequential()


  model.add(tf.keras.layers.Conv2D(filters=16,kernel_size=3,input_shape=(l,w,h),activation='relu',))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=2))



model.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,input_shape=(l,w,h),activation='relu',))
  model.add( tf.keras.layers.MaxPooling2D( pool_size =2) )

model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,input_shape=(l,w,h),activation='relu',))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=2))


  model.add(tf.keras.layers.Flatten())
```

```
    model.add(tf.keras.layers.Dense(512,activation="relu"))
    model.add(tf.keras.layers.Dense(128,activation="relu"))
    model.add(tf.keras.layers.Dense(32,activation="relu"))
    model.add(tf.keras.layers.Dense(1,activation="sigmoid"))
    model.compile(
optimizer = tf.keras.optimizers.Adam(learning_rate=config.l_r),
        loss = 'binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.TruePositives(), tf.keras.metrics.FalsePositives(),
tf.keras.metrics.TrueNegatives(), tf.keras.metrics.FalseNegatives()]
    )

    return model

model=get_model()
model.summary()

Train=True
if Train:
    history = model.fit(train_gen, validation_data=valid_gen, epochs = 1)
    model.save('/kaggle/working/trained_cnn.h5')
    model.save_weights('/kaggle/working/trained_cnn_weights.h5')
else:
    model=load_model('../input/cnn-weights/trained_cnn1.h5')
    model.load_weights('../input/lolnoob/trained_cnn_1.h5')
```

5.1.7 Prediction module

This function takes the path of the data signal file as input and outputs the predicted confidence score of the signal containing a GW signal.

```
def Predict(path):
    t=np.load(path)
    return model.predict(cqt_1992(t).reshape(1,69,193,3))
Target_1_test=['0000bb9f3e.npy','00026119ef.npy','0003920ef5.npy']
  Target_0_test=['0003a9d0b0.npy','0003d34cd0.npy','000a3072ab.npy']
  #qq=Target_1_test[1]
  qq=Target_0_test[2]
  Predict('../input/g2net-gravitational-wave-detection/train/0/0/0/{}'.format(qq))
```



```
history = model.fit_generator(generator=train_gen, validation_data=valid_gen, epochs = 1, workers=-1)
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will
be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
/opt/conda/lib/python3.7/site-packages/nnAudio/utils.py:429: SyntaxWarning: If fmax is given, n_bins will be ignored
  warnings.warn("If fmax is given, n_bins will be ignored", SyntaxWarning)
2022-03-10 00:58:17.471700: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (regi
stered 2)
2022-03-10 00:58:17.477029: I tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Frequency: 2200210000 Hz
6563/6563 [==============================] - ETA: 0s - loss: 0.5159 - accuracy: 0.7237 - true_positives: 68615.2740 - false_positives: 17757.6189
- true_negatives: 87390.3774 - false_negatives: 36284.7297
```

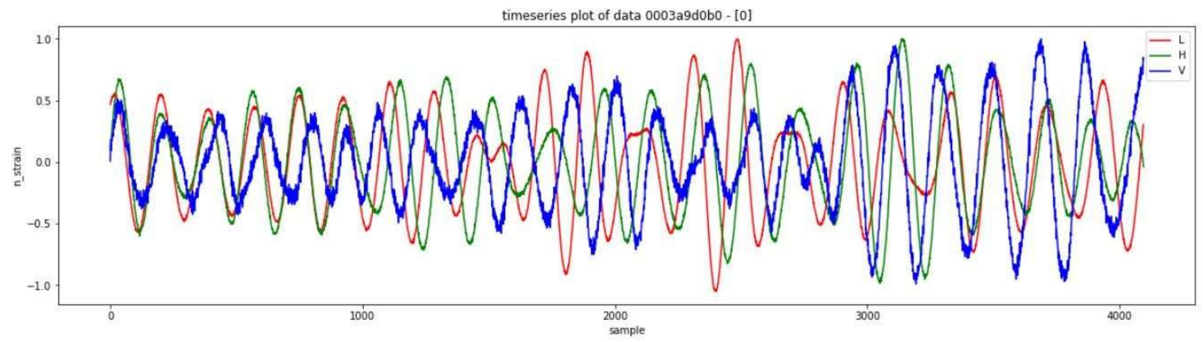
Fig 5.1: Accuracy of the model

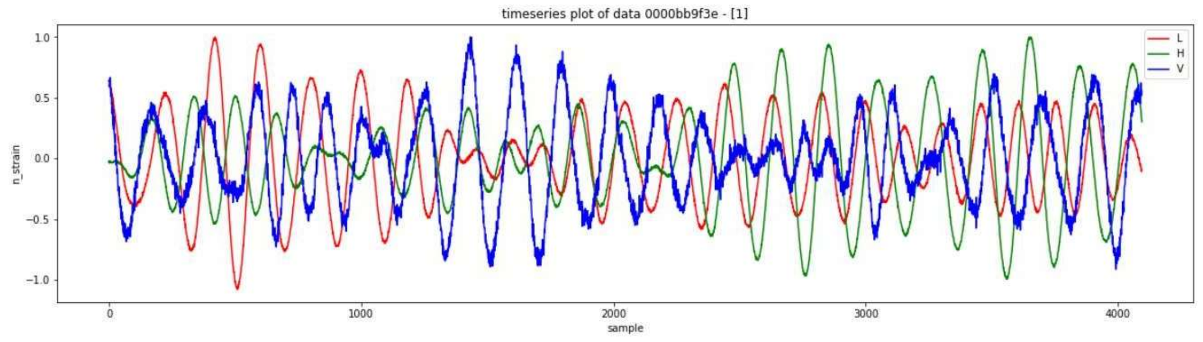
CHAPTER 6

Results and Discussion

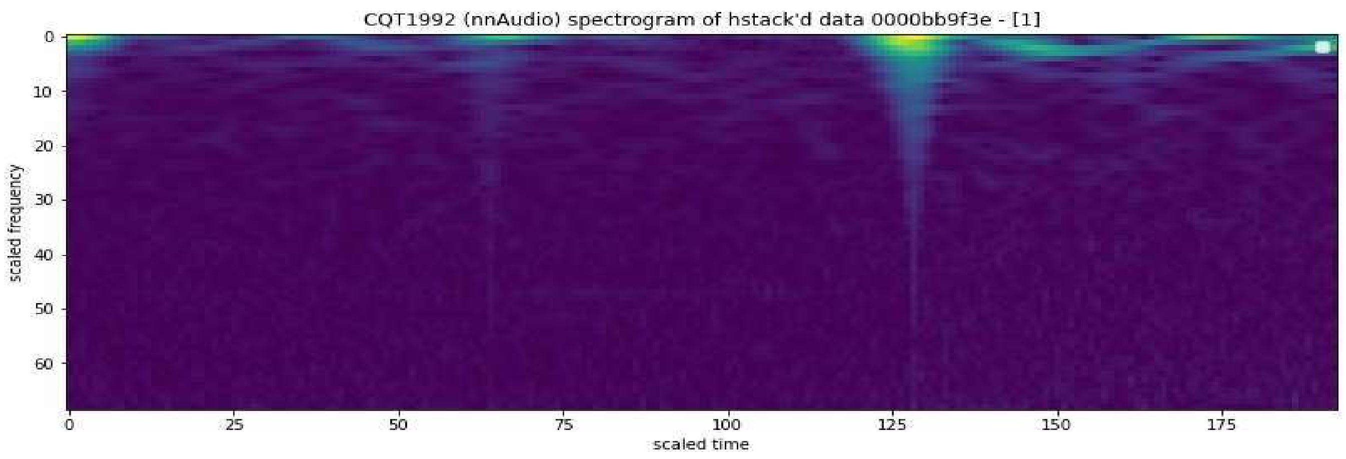
This project is aimed at reducing the time complexity of conventional methods like conventional matched-filtering and Bayesian inference approaches by providing a method to perform real time detection of the signals. This would in-turn be a lot helpful to physicists who can analyze these waves to better understand the universe.

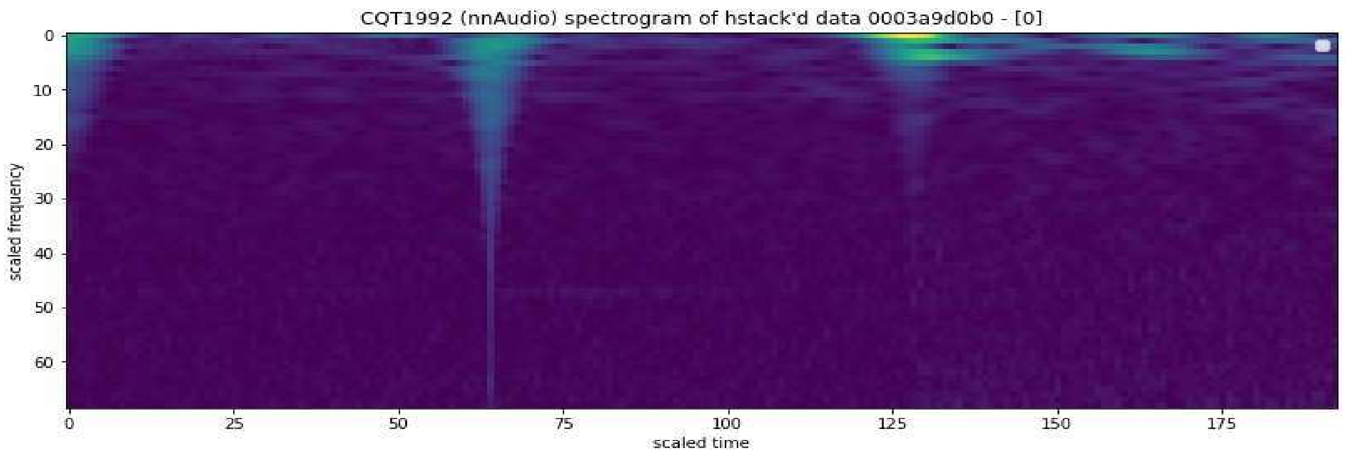
Normalized and filtered signal plotted for three LIGO observatories:





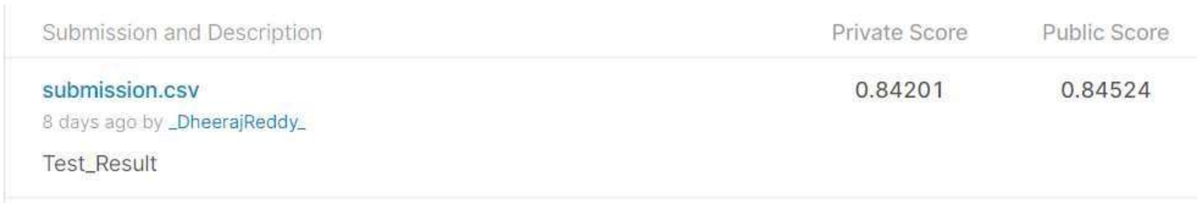
CQT filtered signals that is to be passed as input to the CNN model:





Output of this model is the confidence score of the gravitational wave's existence in a signal:

For this project we have implemented a Deep Learning model that is capable of predicting if a signal contains a GW or not. The had a validation accuracy of 84.209% and an accuracy of 72.37%.



Submission and Description	Private Score	Public Score
submission.csv 8 days ago by _DheerajReddy_ Test_Result	0.84201	0.84524

Fig 6.2: Score of the model determined from mean error of the predicted value

CHAPTER 7

Conclusion and Future Enhancement

7.1 Conclusion

The developed 2-D CNN model provides an alternative to conventional computationally expensive approaches. The model has a validation accuracy of 84.209% and an accuracy of 72.37% when tested for a data set consisted over 100000 files. Thus, we can conclude that DL approaches provide an efficient approach to detecting and recognizing gravitational wave parameters.

7.2 Future Enhancement

The model can be further be enhanced using complex neural network architectures that consume a lot of memory and time. The enhancement of the equipment could also help in improving the efficiency of the model. A large variety of data if fed to the neural network could also improve the accuracy of the model.
The development of signal processing techniques that could effectively remove noise from the signal data would further help in analyzing the signals that would effectively lead to developing a model that could perform better parameter estimation in predicting the underlying signal.

REFERENCES

[1] B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration)Phys. Rev. Lett. 116, 061102 – Published 11 February 2016

[2] Hunter Gabbard, Michael Williams, Fergus Hayes, and Chris Messenger Phys. Rev. Lett. 120, 141103 – Published 6 April 2018.

[3] Elena Cuoco et al 2021 Mach. Learn.: Sci. Technol. 2 011002.

[4] Plamen G.KrastevaKiranjyotGillbV. AshleyVillarbcEdoBergerb; 2012.13101; 2021.

[5] DanielGeorgeabE.A.Huerta;1711.03121;2018.

[6] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, Stéphane Deny; "Barlow Twins: Self-Supervised Learning via Redundancy Reduction"-arXiv:2103.03230; 2021.

[7] https://ieeexplore.ieee.org/document/9174990%5C%22/authors; IEEE Access (Volume: 8); 2020.

[8] S. R. Valluri, V. Dergachev, X. Zhang, and F. A. Chishtie Phys. Rev. D 104, 024065 – Published 26 July 2021

[9] Jingdong Chen, J. Benesty, Yiteng Huang and S. Doclo, "New insights into the noise reduction Wiener filter," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 14, no. 4, pp. 1218-1234, July 2006, doi: 10.1109/TSA.2005.860851.

[10] A. Utina et al., "Deep learning searches for gravitational wave stochastic backgrounds," 2021 International Conference on Content-Based Multimedia Indexing (CBMI), 2021, pp. 1-6, doi: 10.1109/CBMI50038.2021.9461904.

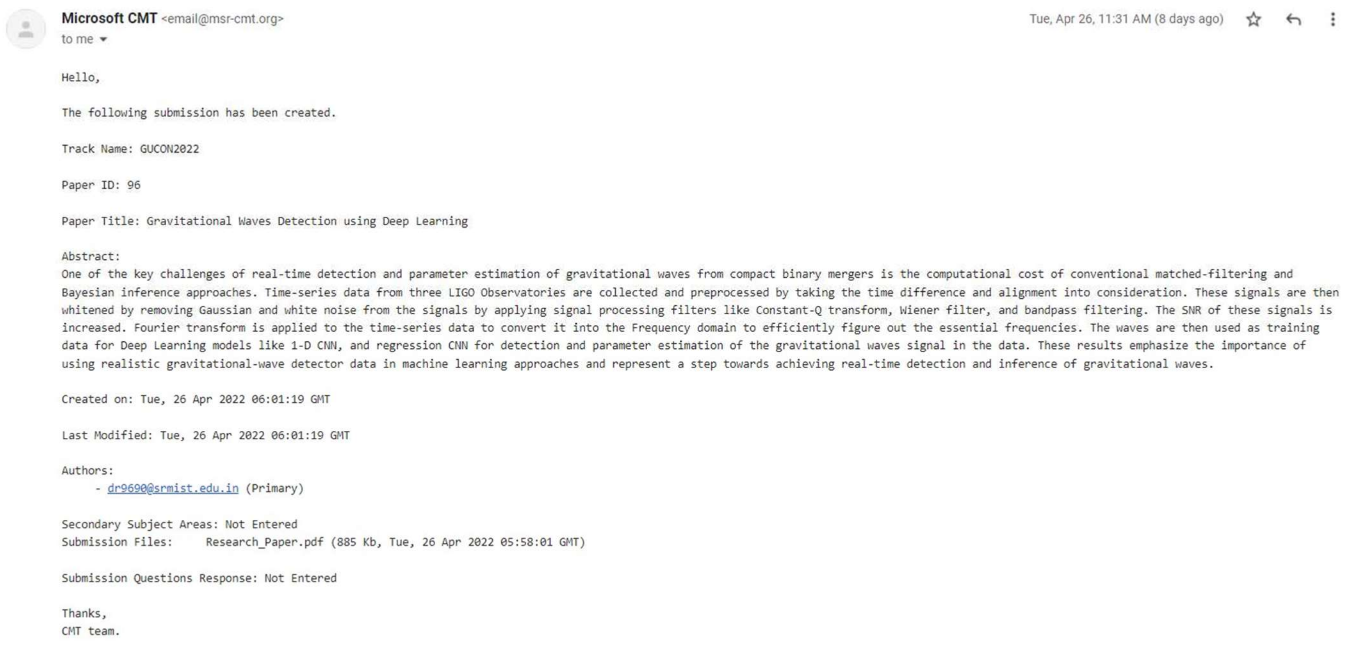
[11] Schörkhuber, Christian & Klapuri, Anssi. (2010). Constant-Q transform toolbox for music processing. Proc. 7th Sound and Music Computing Conf.

Appendix

APPENDIX A

CONFERENCE SUBMISSION

Our paper “Gravitational Waves Detection using Deep Learning” was successfully submitted to IEEE IAS GUCON 2022. Our paper got the submission id “paper id : 96”



Microsoft CMT <email@msr-cmt.org>
to me

Tue, Apr 26, 11:31 AM (8 days ago)

Hello,

The following submission has been created.

Track Name: GUCON2022

Paper ID: 96

Paper Title: Gravitational Waves Detection using Deep Learning

Abstract:
One of the key challenges of real-time detection and parameter estimation of gravitational waves from compact binary mergers is the computational cost of conventional matched-filtering and Bayesian inference approaches. Time-series data from three LIGO Observatories are collected and preprocessed by taking the time difference and alignment into consideration. These signals are then whitened by removing Gaussian and white noise from the signals by applying signal processing filters like Constant-Q transform, Wiener filter, and bandpass filtering. The SNR of these signals is increased. Fourier transform is applied to the time-series data to convert it into the Frequency domain to efficiently figure out the essential frequencies. The waves are then used as training data for Deep Learning models like 1-D CNN, and regression CNN for detection and parameter estimation of the gravitational waves signal in the data. These results emphasize the importance of using realistic gravitational-wave detector data in machine learning approaches and represent a step towards achieving real-time detection and inference of gravitational waves.

Created on: Tue, 26 Apr 2022 06:01:19 GMT

Last Modified: Tue, 26 Apr 2022 06:01:19 GMT

Authors:
- dr9690@srmist.edu.in (Primary)

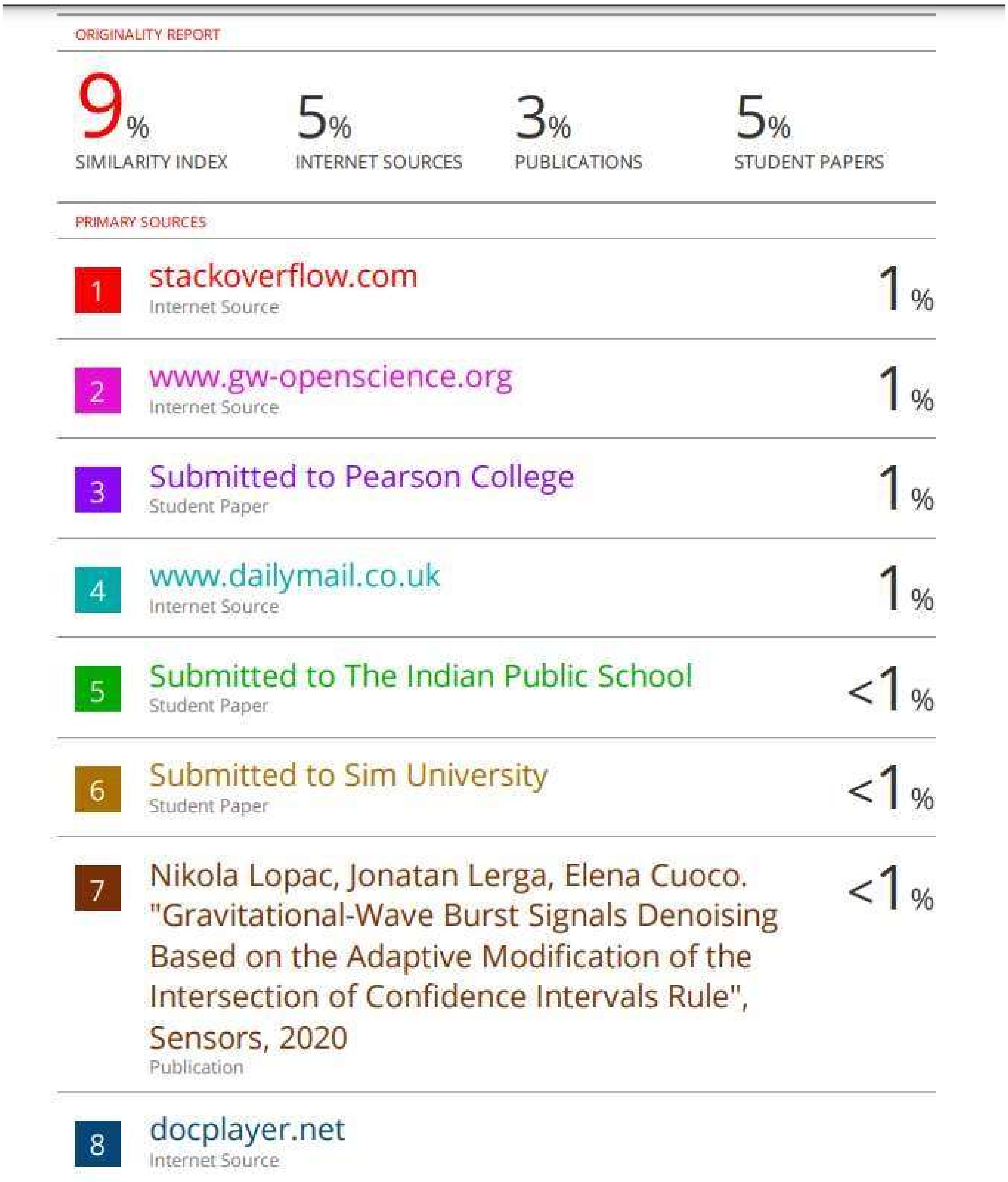
Secondary Subject Areas: Not Entered
Submission Files: Research_Paper.pdf (885 Kb, Tue, 26 Apr 2022 05:58:01 GMT)

Submission Questions Response: Not Entered

Thanks,
CMT team.

APPENDIX B

PLAGAIRISM REPORT

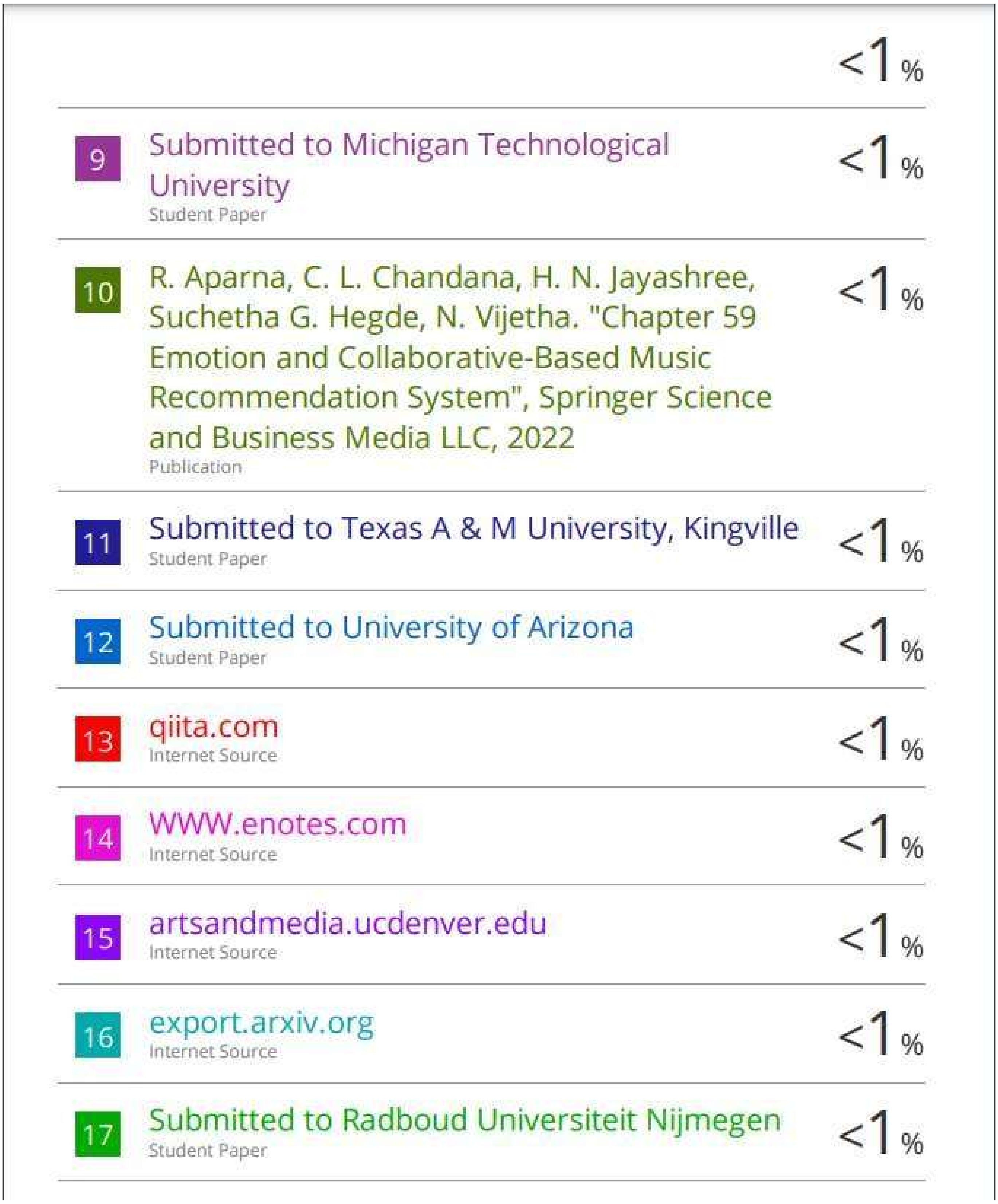


ORIGINALITY REPORT

9%	5%	3%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	stackoverflow.com Internet Source	1%
2	www.gw-openscience.org Internet Source	1%
3	Submitted to Pearson College Student Paper	1%
4	www.dailymail.co.uk Internet Source	1%
5	Submitted to The Indian Public School Student Paper	<1%
6	Submitted to Sim University Student Paper	<1%
7	Nikola Lopac, Jonatan Lerga, Elena Cuoco. "Gravitational-Wave Burst Signals Denoising Based on the Adaptive Modification of the Intersection of Confidence Intervals Rule", Sensors, 2020 Publication	<1%
8	docplayer.net Internet Source	



		<1%
9	Submitted to Michigan Technological University Student Paper	<1%
10	R. Aparna, C. L. Chandana, H. N. Jayashree, Suchetha G. Hegde, N. Vijetha. "Chapter 59 Emotion and Collaborative-Based Music Recommendation System", Springer Science and Business Media LLC, 2022 Publication	<1%
11	Submitted to Texas A & M University, Kingville Student Paper	<1%
12	Submitted to University of Arizona Student Paper	<1%
13	qiita.com Internet Source	<1%
14	WWW.enotes.com Internet Source	<1%
15	artsandmedia.ucdenver.edu Internet Source	<1%
16	export.arxiv.org Internet Source	<1%
17	Submitted to Radboud Universiteit Nijmegen Student Paper	<1%

18	Submitted to Rochester Institute of Technology Student Paper	<1 %
19	Submitted to University of Hertfordshire Student Paper	<1 %
20	Y. Takahashi, M. Takahara, T. Makabe, D. Inami, M. Ohno, F. Nakagawa, T. Koyama, A. Sugiyama, M. Chatani, R. Ikeda. "An ISDN echo-cancelling transceiver chip set for 2B1Q coded U-interface", IEEE Journal of Solid-State Circuits, 1989 Publication	<1 %
21	Submitted to Republic Polytechnic Student Paper	<1 %
22	"Gravity wave", Salem Press Encyclopedia of Science, 2016 Publication	<1 %
23	d1002391.mydomainwebhost.com Internet Source	<1 %
24	documents.mx Internet Source	<1 %
25	pure.port.ac.uk Internet Source	<1 %