Project Report

Classification of LIGO Gravitational Wave data using Machine Learning

[Group 8]

In fulfilment for the requirement of the award of the degree of
Bachelor of Technology

By:
Divyansh Srivastava: 2018021050
Manish Kumar Yadav: 2018021064
Pankaj Kumar Singh: 2018021079
B.Tech IV year (C.S.E.)

Project Guide:
Manish Kumar Srivastava
Assistant Professor
C.S.E. Department
Madan Mohan Malaviya University of Technology

MADAN MOHAN MALAVIYA UNIVERSITY OF TECHNOLOGY (MMMUT)

GORAKHPUR, UTTAR PRADESH 273010

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report entitled

"Classification of LIGO Gravitational Wave data using Machine Learning"

submitted by Divyansh Srivastava (2018021050), Manish Kumar Yadav (2018021064), Pankaj Kumar Singh (2018021079) of Semester VIII is a bonafide account of the work done by them under our supervision.

Guide:
Mr. Manish Kumar Srivastava

Head of the Department:
Prof. Udai Shankar

Candidate's Declaration

I declare that this written submission represents my work and ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Divyansh Srivastava
Roll No: 2018021050

Manish Kumar Yadav
Roll No: 2018021064

Pankaj Kumar Singh
Roll No: 2018021079

B.Tech IV year (C.S.E.)

Approval Sheet

This project report entitled **Classification of LIGO Gravitational Wave data using Machine Learning** by Divyansh Srivastava, Manish Kumar Yadav and Pankaj Kumar Singh is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Examiner

_____

Supervisor
Mr. M. K. Srivastava

Head of Department
Prof. Udai Shankar

Date:

Place:

Acknowledgements

We would like to sincerely thank our guide Mr Manish Kumar Srivastava, Asst. Prof, CSE, for his support and valuable guidance. His timely advice, meticulous scrutiny, scholarly and scientific approach has helped us complete this project on time.

Also, we would like to thank our university, MMMUT, for providing the best facility and support.

We thank each and every staff of the Computer Science Department for lending us help and support. We express our heartfelt veneration to all who had been helpful and inspiring throughout this endeavour. Last but not the least, we thank the almighty for blessing us for completing the project.

Divyansh Srivastava
Roll No: 2018021050

Manish Kumar Yadav
Roll No: 2018021064

Pankaj Kumar Singh
Roll No: 2018021079

B.Tech IV year (C.S.E.)

Preface

I have made this report file on the topic **Classification of LIGO Gravitational Wave data using Machine Learning**. I have tried my best to elucidate all the relevant details of the topics included in the report, while in the beginning I have tried to give a overview. This is followed by the preprocessing and finally the main procedure and conclusion about this topic.

My efforts and wholehearted cooperation of each and every one has ended on a successful note. I express my sincere gratitude to Mr M.K. Srivastava who assisted me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Divyansh Srivastava
Roll No: 2018021050

Manish Kumar Yadav
Roll No: 2018021064

Pankaj Kumar Singh
Roll No: 2018021079

B.Tech IV year (C.S.E.)

Table of Contents

Abstract

The gravitational waves are the ripples in the fabric of spacetime that originates from a celestial body or phenomenon. The astronomers have made great achievements in observing the universe through telescope across the electromagnetic spectrum. But when it comes to observing bodies like neutron star that are extremely dense but very small in size or blackholes that have such strong gravity that not even electromagnetic waves can escape, it becomes a really tough work for a telescope. This is where gravitational waves come in. Neutron stars or blackholes are so dense that it distorts the spacetime around it in a very unique way. This distortion creates gravitational waves that travel to us on earth. Although the waves reaching the earth are so faint that it gets mixed with many type of noises. This project aims to access the gravitational wave data detected through LIGO/Virgo detectors and classify them. By training a machine learning model with already existing data of unique signatures of gravitational waves that come from different sources of noises, mergers of possible pairs of black hole and neutron stars, this project aims to classify the data in various classes of signal i.e. whether the data is actually coming from a merger event or is just useless noise.

Introduction

Observation through gravitational wave has been a growing science with 2017 Nobel price in Physics falling in the same field. The conventional observation through telescope becomes hard because the only feasible way to detect such superdense objects is to observe the light or different signals that come from the background of such objects that get curved due to the gravity of the foreground object since there are near to nothing signal that comes out of such objects. The hot shiny accretion disk surrounds the black hole but this is detectable only when it is known that where should a telescope be pointing at. The gravitational wave observation of such objects provides an insight of where the object could be located and it gives a head start to the astronomers to which part of the space should they observe for further analysis.

But the gravitational wave detection has the challenges of its own. Since the gravitational waves coming from the celestial objects are very faint, it becomes hard to prevent it from mixing among the various type of noises on earth such as the power transmission noise, the glitches from detection devices, the thermal noise, the brownian noise, the terrestrial noise and the seismic noises etc. The fact that makes it so hard to make out the difference between a signal and a noise is that a signal from a merger event has no defined characteristics which can be used to tag it as a signal. In fact, most of the times a signal looks similar to noise. In this project, we dedicated initial part for removal of those noise which could be identified. Then for other part of this project we focused on classifying the processed data into different class labels.

LIGO detectors

The gravitational waves engage with space by compressing it in one way while expanding them in the orthogonal way. This is the reason why modern gravitational wave detectors are L-shaped. They measure the comparative lengths of the arms using the produced interference patterns generated by the combination of two light lasers. The interferometer at Hanford and Livingston are jointly called LIGO (Laser Interferometer Gravitational-wave Observatory). LIGO is the largest of the gravitational wave detectors with arms of 4 km length. The other detectors are in VIRGO in Italy, GEO in Germany and TAMA in Japan.

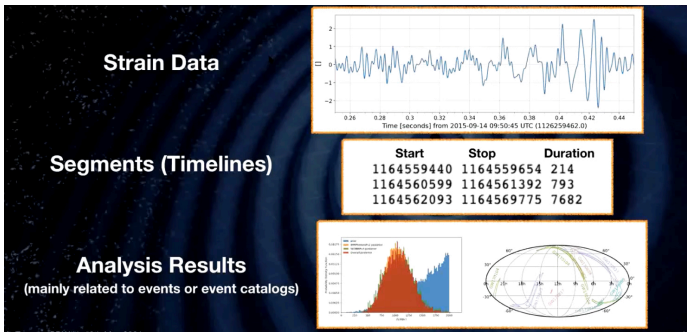
Aerial view of the LIGO detector in Hanford, WA.

LIGO technicians work on an internal optical components

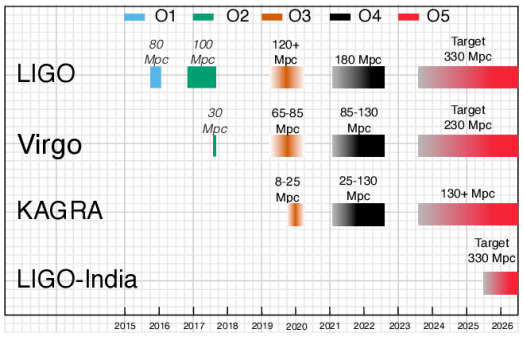
At LIGO's end, various measures are taken to remove the noise so that detector doesn't register them. This is done by placing the interferometer in complete vacuum, using an advanced system of hydraulics to counter seismic noises and a system to move the mirror against any air currents to nullify its effect. But even after such attempts, not all noises can be removed by physical measures. Our project is aimed to identify other sources of noise that appear in the data so that true signal never gets lost between the noise.

Working with data

1. Accessing public GW data using GWOSC: The Gravitational Wave Open Science Center (GWOSC) provides data from gravitational-wave observatories like LIGO Hanford, LIGO Livingston and Virgo etc.



Available forms of data at GWOSC



The detectors alternate period of data taking

The module **gwosc.datasets** provides tools for searching for datasets, including events, catalogs and full run strain data releases. The 'events' imply to the confident detection already made by the LIGO team. Whereas a segment is a dataset within a specified period of time.

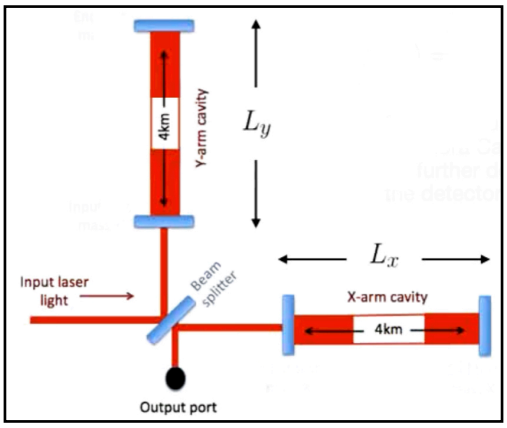
Our main focus is around the Strain data. In the next section, we will explain how to refine the data that has been accessed in this part.

2. Cleansing and band-passing of data : GWpy is a collaboration-driven Python package providing tools for studying data from ground-based gravitational-wave detectors.

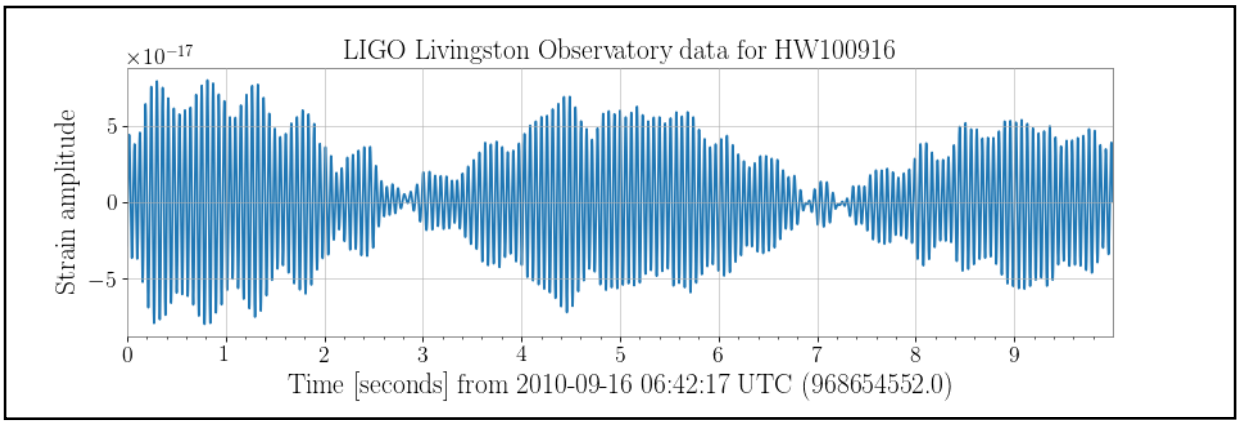
Our approach for refining data has been shown in the following steps:

- **Step 1: Obtaining the strain data -**

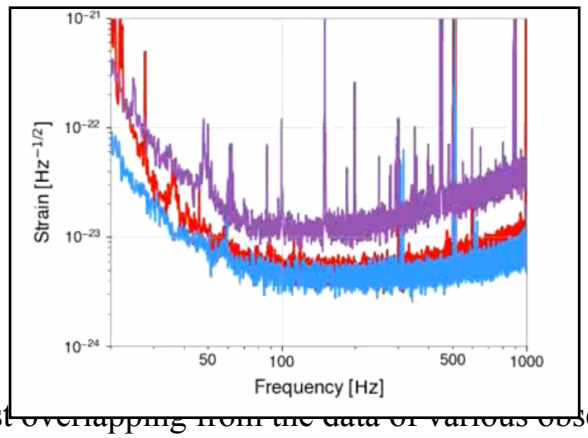
$$h(t) = (\Delta Ly - \Delta Lx)/L$$



- **Step 2: Plotting the strain data -**



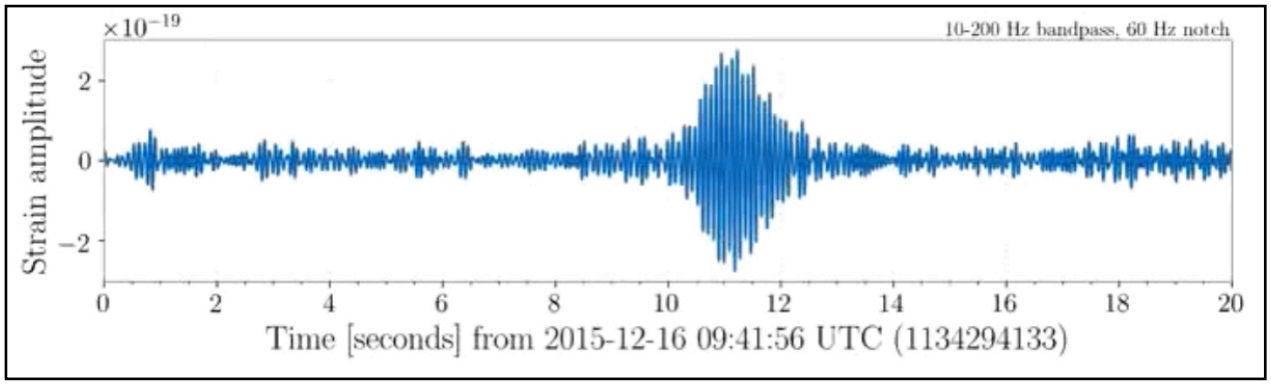
As we can see there is nothing much to infer from this plot. Here is the same plot but in the frequency domain.



The spikes can be seen almost overlapping from the data of various observatory in the world. But still the data makes no sense.

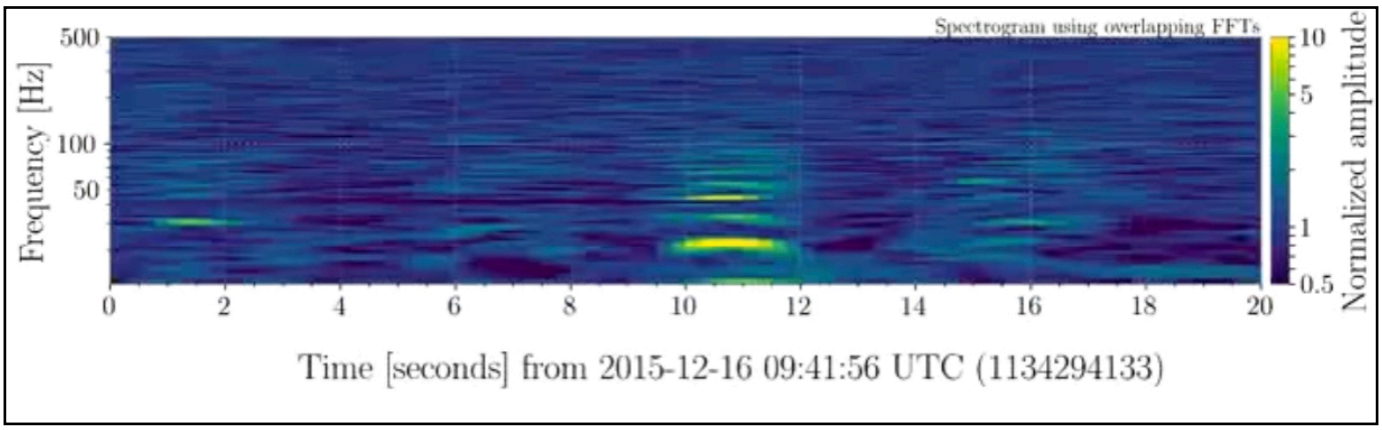
• Step 3: Band-passing the strain data -

If we pass the strain data plot through a bandpass method of TimeSeries object of the GWpy module, we get the following plot:

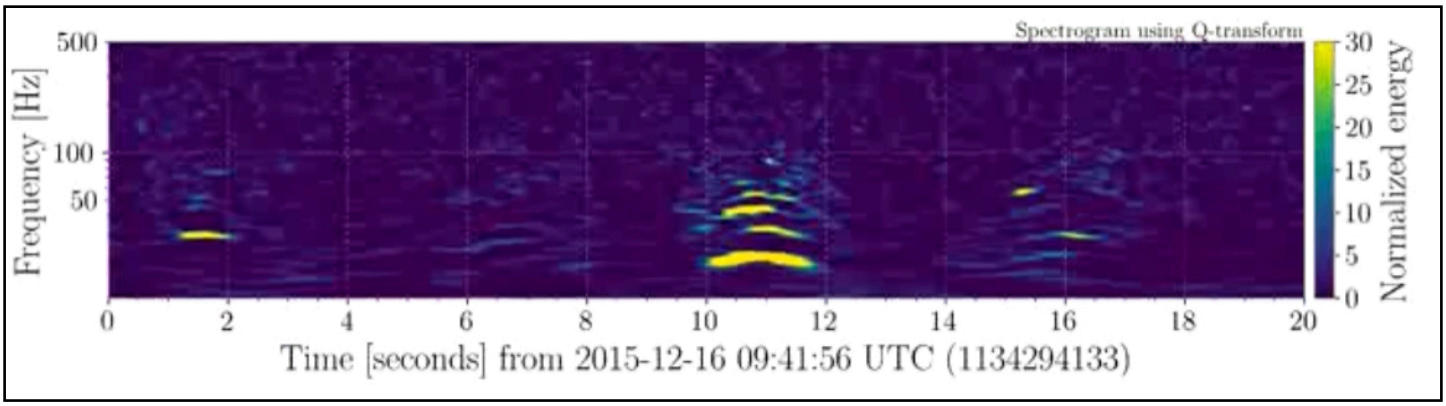


Now, the data strain amplitude shows a significant movement around a specific time i.e. it can be safely inferred that the detector has picked something. Now, it's upto us to investigate if this a signal or noise.

2. Q Transform - While the TimeSeries allows us to study how the amplitude of a signal changes over time, and the FrequencySeries allows us to study how that amplitude changes over frequency, the time-frequency Spectrogram allows us to track the evolution of the Frequency Series over over time.



In the above plot, the spectrogram has been obtained after using overlapping Fast Fourier Transforms. For detailed insight a similar spectrogram can be plotted using Q transform. The following plot shows the same.



Even now when we have a refined dataset we can still not conclude that whether the detection comes from a cosmic object/phenomenon or from the blips and glitches of the detector. For classifying the data to separate it from false detections, the data has to be fed to a machine learning model as discussed henceforth.

2. Notable attempt: For the initial phases, we have obtained strains and manipulated it as per above process to plot the spectrogram as shown in the following snapshots-

i. In first attempt, we tried to fetch the strain around already detected event GW170817

```
! pip install -q 'gwpy==2.0.2'
```

```
     |████████████████████████████████| 1.4 MB 28.3 MB/s
     |████████████████████████████████| 51 kB 7.8 MB/s
     |████████████████████████████████| 55 kB 4.5 MB/s
     |████████████████████████████████| 3.6 MB 46.7 MB/s
  Building wheel for ligo-segments (setup.py) ... done
```

```
%matplotlib inline
import gwpy
```

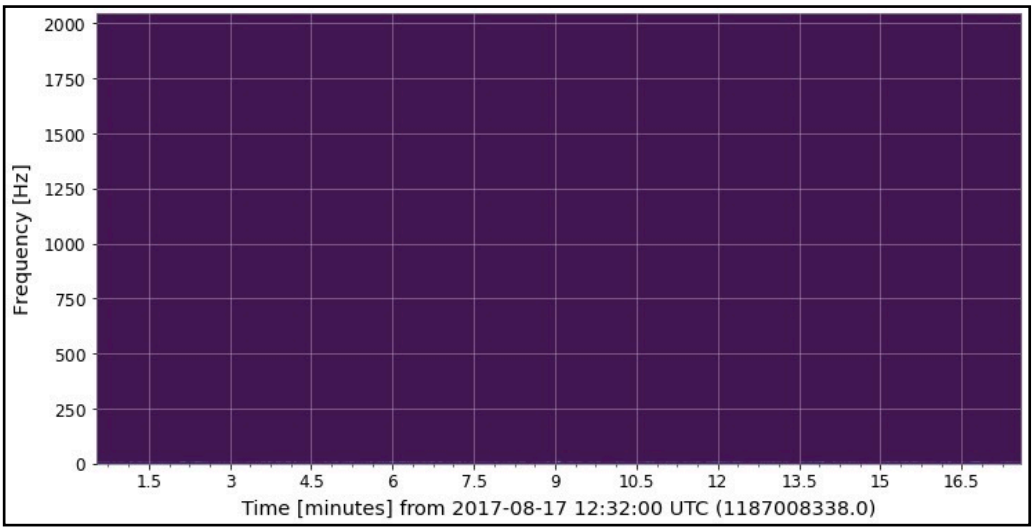
```
[3] from gwosc.datasets import event_gps
    from gwpy.timeseries import TimeSeries

    gps = event_gps('GW170817')
    print("GW170817 GPS:", gps)

    ldata = TimeSeries.fetch_open_data('L1', int(gps)-512, int(gps)+512, cache=True)
    print("GW170817 data")
    print(ldata)
```

```
GW170817 GPS: 1187008882.4
GW170817 data
TimeSeries([2.06056010e-20, 1.59181918e-20, 2.18438811e-20, ...,
            1.25504332e-19, 1.23976846e-19, 1.22231459e-19]
           unit: dimensionless,
           t0: 1187008370.0 s,
           dt: 0.000244140625 s,
           name: Strain,
           channel: None)
```

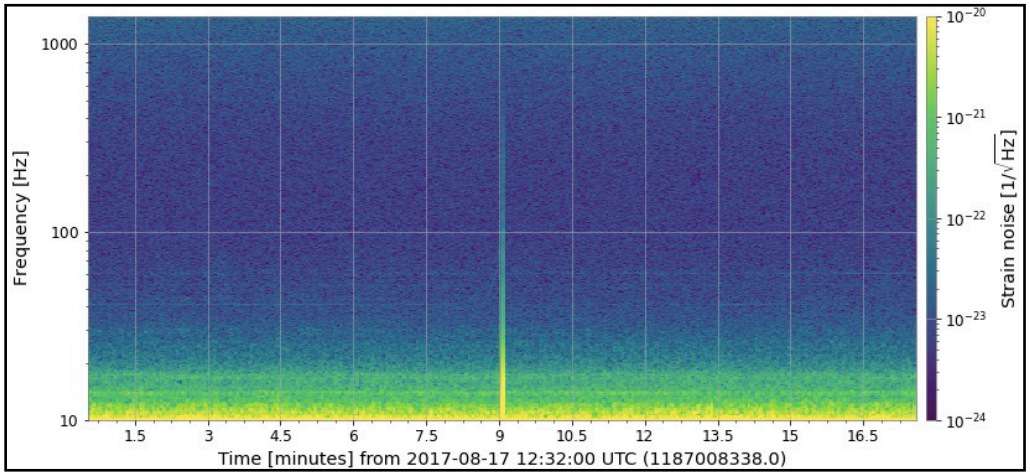
```
[4] specgram = ldata.spectrogram2(fftlength=4, overlap=2, window='hann') ** (1/2.)
    plot = specgram.plot()
```

```
✓ 2s  completed at 10:48 AM
```

But we got no detection on the spectrogram.



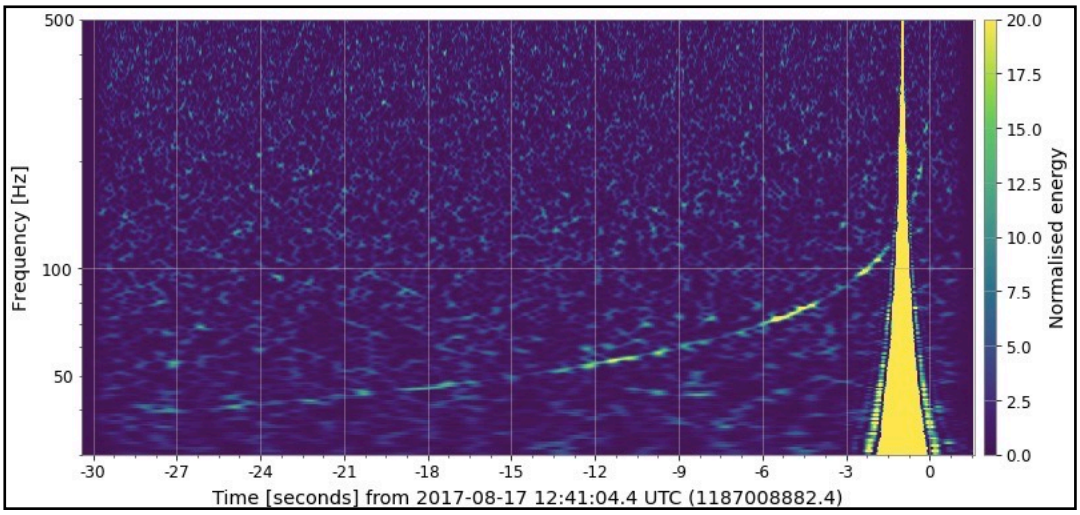
ii. We windowed the strain to a narrower value and changed the frequency scale to log.



We then adjusted the colorbar to obtain normalised frequency spectrogram.

```
[9] hq = hdata.q_transform(frange=(30, 500), qrange=(100, 110))
    plot = hq.plot()
    ax = plot.gca()
    ax.set_epoch(gps)
    ax.set_yscale('log')
    ax.colorbar(label="Normalised energy")
```

iii. We obtain a clear spectrogram with signs of a detection.

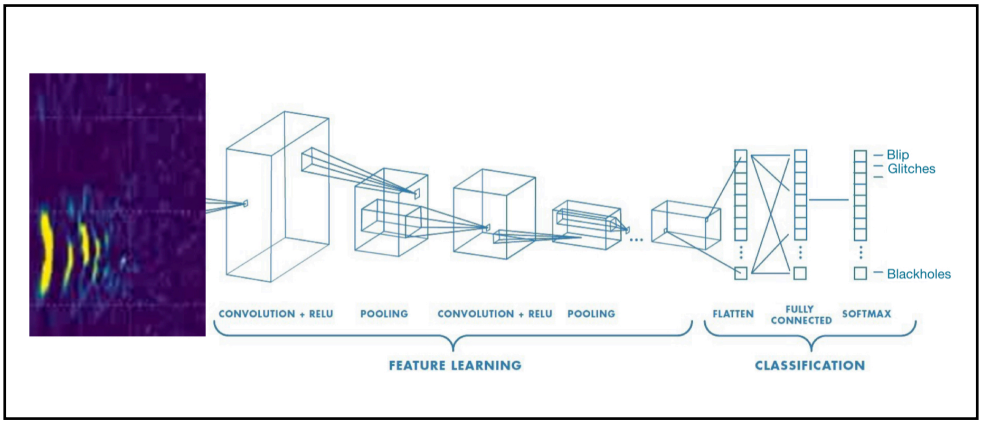


We aim to feed Q-transform like this to our model to classify it among the labels of signal and noise.

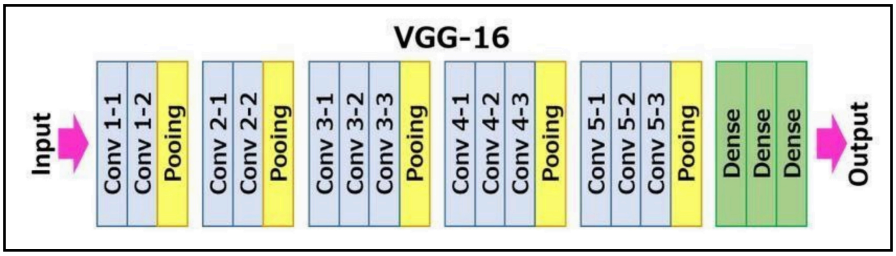
Classification of the data using Machine Learning

Gravitational wave detection requires a detailed understanding of the response of the LIGO and Virgo detectors to true signals in the presence of environmental and instrumental noise. Of particular interest is the study of anomalous non-Gaussian transients, such as glitches, since their occurrence rate in LIGO and Virgo data can obscure or even mimic true gravitational wave signals. Therefore, successfully identifying and eliminating these anomalies from gravitational wave data is of utmost importance for the detection and characterisation of true signals and for the accurate computation of their significance.

1. **Selection of model :** Upon reviewing similar projects in the field, it is found that Deep Transfer Learning method enables an optimal use of very deep convolutional neural networks for glitch classification given small and unbalanced training data sets, significantly reduces the training time and provides high accuracy. The best thing about this class of models is that it automatically picks up those features of the dataset that would train the model optimistically.



Upon trying various models of CNN, we decided to use **VGG16** model. This model was developed for the sole purpose of classification and localisation. Some of its features are:



1. It has a total of 21 layers out which 16 layers are weighted. Having a model where the data traverses through 16 layers of learnable parameters ensures high accuracy.
2. Along the entire architecture the convolution and pooling layers are consistently sandwiched.
3. Our model-ready data is a Q-transform where colormap is indicating the magnitude of a specific frequency at any given time.

2. **Optimiser:** Evidently, once we obtain a Q-transform, it becomes more or less like an image classification problem. So the best suited algorithm for our stochastic gradient descent was Adam's optimiser. In most simple terms, it can be understood as an algorithm that assigns weight dynamically to the layer over each iteration. **Adam's optimiser** is the most sought optimiser for computer vision problems.

3. **Loss function:** We intended to classify our Q-transform into different labels through our VGG16 model. This makes it a multi-label classification where no data can simultaneously correspond to two different labels. **Categorical cross-entropy** compares the predicted probability distribution with the target probability distribution.

$$\text{Loss} = \sum_{\substack{\text{output} \\ \text{size}}} -t_i \log(y_i)$$

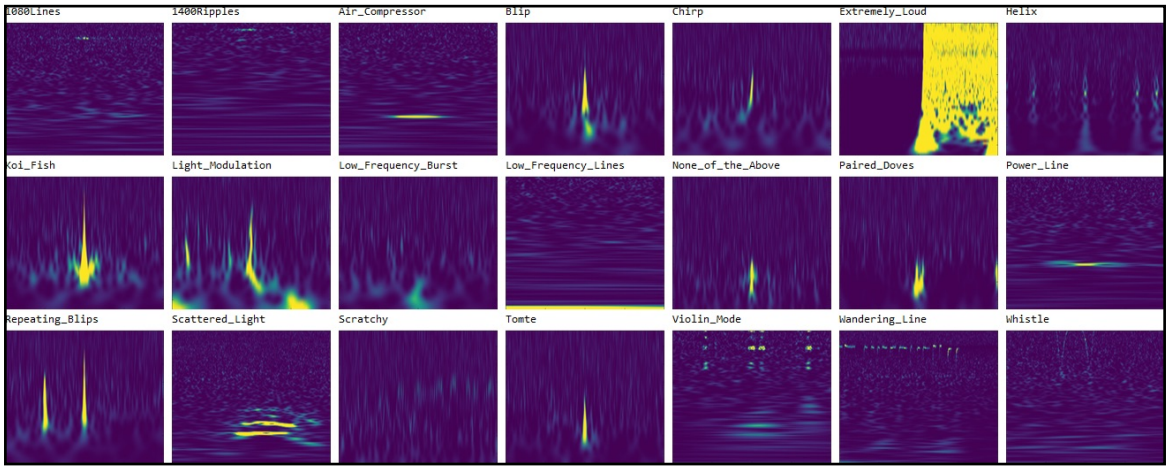
The negative sign ensures that the loss will reduce when the distributions get closer to each other.

4. **Activation function:** The categorical cross-entropy loss function accepts only a probability distribution. **Softmax** activation function is the most suited function for this cause because the softmax activation function rescales the model output in a way so that it tunes out the correct properties. It is also the most recommended activation function to be used with categorical-cross entropy loss function.

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_{k=0}^{K} e^{x_k}}$$

Training of the model

1. **Procurement of data:** The above model was trained by using a data set of following classes of glitches, curated and labeled by the Gravity Spy project using data collected during LIGO's first discovery campaign. Following classes of glitches are identified.



The Gravity spy project has separate dataset for training, testing and validation of data.

2. **Decision about data:** Batch size is a hyper-parameter that decides how much data a model should iterate through before updating its internal parameters. For our training of the model, we chose the **batch size to be 32.**

   As per the requirement of VGG16 model, we gave the input as a **tensor of size 224*224*3** converted from Q-transform that we created as discussed above by the help of pillow module.

3. **Data size for each class:** Upon reviewing the training dataset, we discovered that not all classes had enough number of training data. Since our model couldn't be trained over lesser amount of data due to the problem of overfitting, we decided to leave those classes where data was less than 100.

Testing and Validation of the model

1. **Validation split:** There have been three observation runs by LIGO since 2015 but the Gravity spy dataset available to us had data from the first run (O1) only. Having no abundance of data to spare for validation, we have opted for a validation split of 0.25.

2. **A note about labels:** Our model takes the Q-transform image as its input and classifies it into following four classes: blip, chirp, violin mode and koi_fish

Platform Code Snippet

```
import streamlit as st
import warnings
warnings.filterwarnings('ignore',category=DeprecationWarning)
st.set_option('deprecation.showPyplotGlobalUse', False)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import requests, os
from gwpy.timeseries import TimeSeries
from gwosc.locate import get_urls
from gwosc import datasets
from gwosc.api import fetch_event_json

from copy import deepcopy
import base64
import numpy as np
import tensorflow as tf
from PIL import Image
import matplotlib as mpl

mpl.use("agg")

from matplotlib.backends.backend_agg import RendererAgg

_lock = RendererAgg.lock


def preprocessing(pixel):
    pixel = pixel.resize((224, 224))
    pixel = pixel.convert('RGB')
    pixel = np.array(pixel)
    pixel = pixel.reshape((1, 224, 224, 3))
    pixel = pixel / 255.


    return pixel


def getresult(pixel):

    pixel = preprocessing(pixel)
    #plt.imshow(pixel[0])
    #st.pyplot()
    tfmodel = tf.keras.models.load_model('/Users/user/Downloads/LIGO/vggmodel.h5')
```

```
    #st.markdown(tfmodel.input)
    res = tfmodel.predict(pixel)
    res = list(res[0])
    res = res.index(max(res))
    res_map = {0:"Chirp",1:"Violin_Mode",2:"Koi_Fish",3:"Blip"}

    return (res_map[res], res)

apptitle = 'Classification of LIGO Gravitational Wave data using Machine Learing'

st.set_page_config(page_title=apptitle, page_icon=":eyeglasses:")

detectorlist = ['H1', 'L1', 'V1']

st.title('Classification of LIGO Gravitational Wave data using Machine Learning')

st.markdown("""
 Under the guidance of:
 * Manish Kumar Srivastava Sir\n
 By:\n
 * Divyansh Srivastava : 2018021050
 * Manish Kumar Yadav  : 2018021064
 * Pankaj Kumar Singh  : 2018021079
""")
@st.cache(ttl=3600, max_entries=10)
def load_gw(t0, detector, fs=4096):
    strain = TimeSeries.fetch_open_data(detector, t0 - 14, t0 + 14, sample_rate=fs, cache=False)
    return strain


@st.cache(ttl=3600, max_entries=10)
def get_eventlist():
    allevents = datasets.find_datasets(type='events')
    eventset = set()
    for ev in allevents:
        name = fetch_event_json(ev)['events'][ev]['commonName']
        if name[0:2] == 'GW':
            eventset.add(name)
    eventlist = list(eventset)
    eventlist.sort()
    return eventlist


st.sidebar.markdown("## Data Fetcher")

eventlist = get_eventlist()
```

```
select_event = st.sidebar.selectbox('Finding Method',['By event name', 'By GPS'])

if select_event == 'By GPS':
    str_t0 = st.sidebar.text_input('GPS Time', '1126259462.4')
    t0 = float(str_t0)

else:
    chosen_event = st.sidebar.selectbox('Select Event', eventlist)
    t0 = datasets.event_gps(chosen_event)
    detectorlist = list(datasets.event_detectors(chosen_event))
    detectorlist.sort()
    st.subheader(chosen_event)
    st.write('GPS:', t0)

    #  Experiment to display masses
    try:
        jsoninfo = fetch_event_json(chosen_event)
        for name, nameinfo in jsoninfo['events'].items():
            st.write('Mass 1:', nameinfo['mass_1_source'], 'M$_{\odot}$')
            st.write('Mass 2:', nameinfo['mass_2_source'], 'M$_{\odot}$')
            st.write('Network SNR:', int(nameinfo['network_matched_filter_snr']))
            st.write('\n')
    except:
        pass

detector = st.sidebar.selectbox('Detector', detectorlist)

fs = 4096
maxband = 2000
high_fs = st.sidebar.checkbox('Full sample rate data')
if high_fs:
    fs = 16384
    maxband = 8000

# -- Create sidebar for plot controls
st.sidebar.markdown('## Set Plot Parameters')
dtboth = st.sidebar.slider('Time Range (seconds)', 0.1, 8.0, 1.0)  # min, max, default
dt = dtboth / 2.0


freqrange = st.sidebar.slider('Band-pass frequency range (Hz)', min_value=10, max_value=maxband,
value=(30, 400))

# -- Create sidebar for Q-transform controls
st.sidebar.markdown('#### Q-tranform plot')
vmax = st.sidebar.slider('Colorbar Max Energy', 10, 500, 25)  # min, max, default
qcenter = st.sidebar.slider('Q-value', 5, 120, 5)  # min, max, default
```

```
qrange = (int(qcenter * 0.8), int(qcenter * 1.2))

# -- Create a text element and let the reader know the data is loading.
strain_load_state = st.text('Loading data...this may take a minute')
try:
    strain_data = load_gw(t0, detector, fs)
except:
    st.warning(
        '{0} data are not available for time {1}.  Please try a different time and detector pair.'.format(detector, t0))
    st.stop()

strain_load_state.text('Loading data...done!')

# -- Make a time series plot

cropstart = t0 - 0.2
cropend = t0 + 0.1

cropstart = t0 - dt
cropend = t0 + dt

st.subheader('Raw data')
center = int(t0)
strain = deepcopy(strain_data)

with _lock:
    fig1 = strain.crop(cropstart, cropend).plot()
    # fig1 = cropped.plot()
    st.pyplot(fig1, clear_figure=True)

st.subheader('Whitened and Band-passed Data')


white_data = strain.whiten()
bp_data = white_data.bandpass(freqrange[0], freqrange[1])

bp_cropped = bp_data.crop(cropstart, cropend)

with _lock:
    fig3 = bp_cropped.plot()
    st.pyplot(fig3, clear_figure=True)


st.subheader('Q-transform')

hq = strain.q_transform(outseg=(t0 - dt, t0 + dt), qrange=qrange)
```

```
with _lock:
    fig4 = hq.plot()
    ax = fig4.gca()
    cbar = fig4.colorbar(label="Normalised energy", vmax=vmax, vmin=0)
    ax.grid(False)
    ax.set_yscale('log')
    ax.set_ylim(bottom=15)
    st.pyplot(fig4, clear_figure=False)

    ax.set_axis_off()
    cbar.remove()
    #szwst.pyplot(fig4, clear_figure=False)
    fig4.savefig('photo.png',bbox_inches='tight',pad_inches = 0)

    img = Image.open('photo.png')


    #st.pyplot(img)
    #print(fig4)

st.subheader("Classification")
st.markdown("""
""")
output = getresult(img)
st.markdown("The above spectrogram most likely belongs to "+ output[0] + "  class.")


hide_streamlit_style = """
            <style>
            #MainMenu {visibility: hidden;}
            footer {visibility: hidden;}
            </style>
            """
st.markdown(hide_streamlit_style, unsafe_allow_html=True)
```

Model Snippet

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
        break
    break

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
In [2]:

# modules
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
# import keras
# from keras.layers import Conv2D
# from keras.models import Sequential
# from keras.layers import MaxPool2D
# from keras.layers import Flatten
# from keras.layers import Dense
# from keras.models import Model
# from keras.preprocessing.image import ImageDataGenerator
import tensorflow.keras as keras
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import accuracy_score
import cv2
import re
import random
import matplotlib.image as mpimg
from PIL import Image
random.seed(0)
np.random.seed(0)
In [3]:

# Variables
datasize = 100
totalclasses = 0
number_of_classes = -1
no_of_epochs = 100
validation_split = 0.25
batch_size = 32
ImagePixels = []
ImageLabels = []
take_folder = ['Blip','Chirp','Violin_Mode','Whistle','Koi_Fish']
In [4]:

def getImagePixel(filepath):
    pixel = Image.open(filepath)
    pixel = pixel.convert('RGB')
    pixel = pixel.resize((224,224))
    pixel = np.array(pixel)
    pixel = pixel/255.
    return pixel
In [5]:

totalclasses = 0
ImagePixels = []
ImageLabels = []
columns = []
for folderno,foldername in enumerate(os.listdir('/kaggle/input/gravity-spy-gravitational-waves/train/train')
[:number_of_classes]):
    if(foldername not in take_folder): continue
    folderpath= "/kaggle/input/gravity-spy-gravitational-waves/train/train/"+foldername

    for filename in os.listdir(folderpath)[:datasize]:
        filepath = folderpath+'/'+filename
        imagepixel = getImagePixel(filepath)
        ImagePixels.append(imagepixel)
        ImageLabels.append([folderno])
```

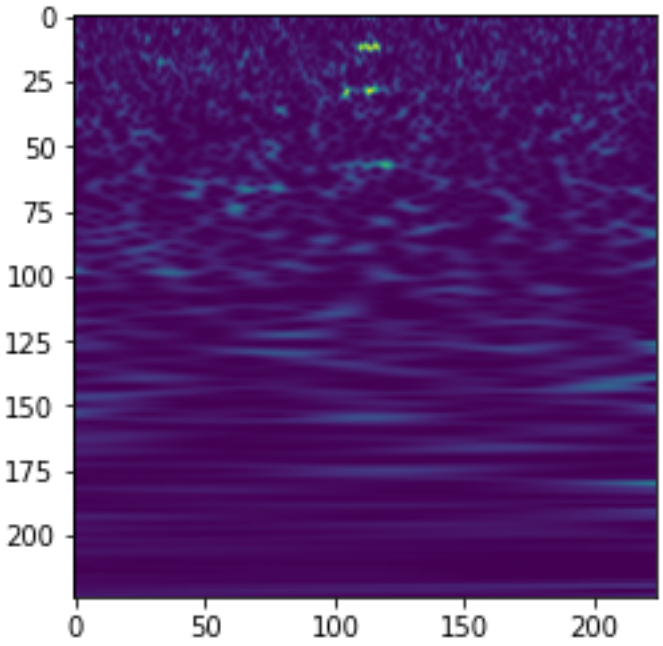
```
    totalclasses+=1
    columns.append(foldername)
    print(foldername,totalclasses,ImagePixels[0].shape)

ImagePixels = np.array(ImagePixels)
```

```
Chirp 1 (224, 224, 3)
Violin_Mode 2 (224, 224, 3)
Koi_Fish 3 (224, 224, 3)
Blip 4 (224, 224, 3)
In [6]:
```

```
ohencoder = OneHotEncoder(handle_unknown='ignore',sparse=False)
ImageLabels_df = pd.DataFrame(ImageLabels,columns=["label"])

ohlabel = pd.DataFrame(ohencoder.fit_transform(ImageLabels_df),dtype = 'float64',columns = columns)
print(ohlabel.head())

y_train = np.array(ohlabel)
print(y_train.shape)
```

```
   Chirp  Violin_Mode  Koi_Fish  Blip
0    1.0          0.0       0.0   0.0
1    1.0          0.0       0.0   0.0
2    1.0          0.0       0.0   0.0
3    1.0          0.0       0.0   0.0
4    1.0          0.0       0.0   0.0
(400, 4)
In [7]:
```

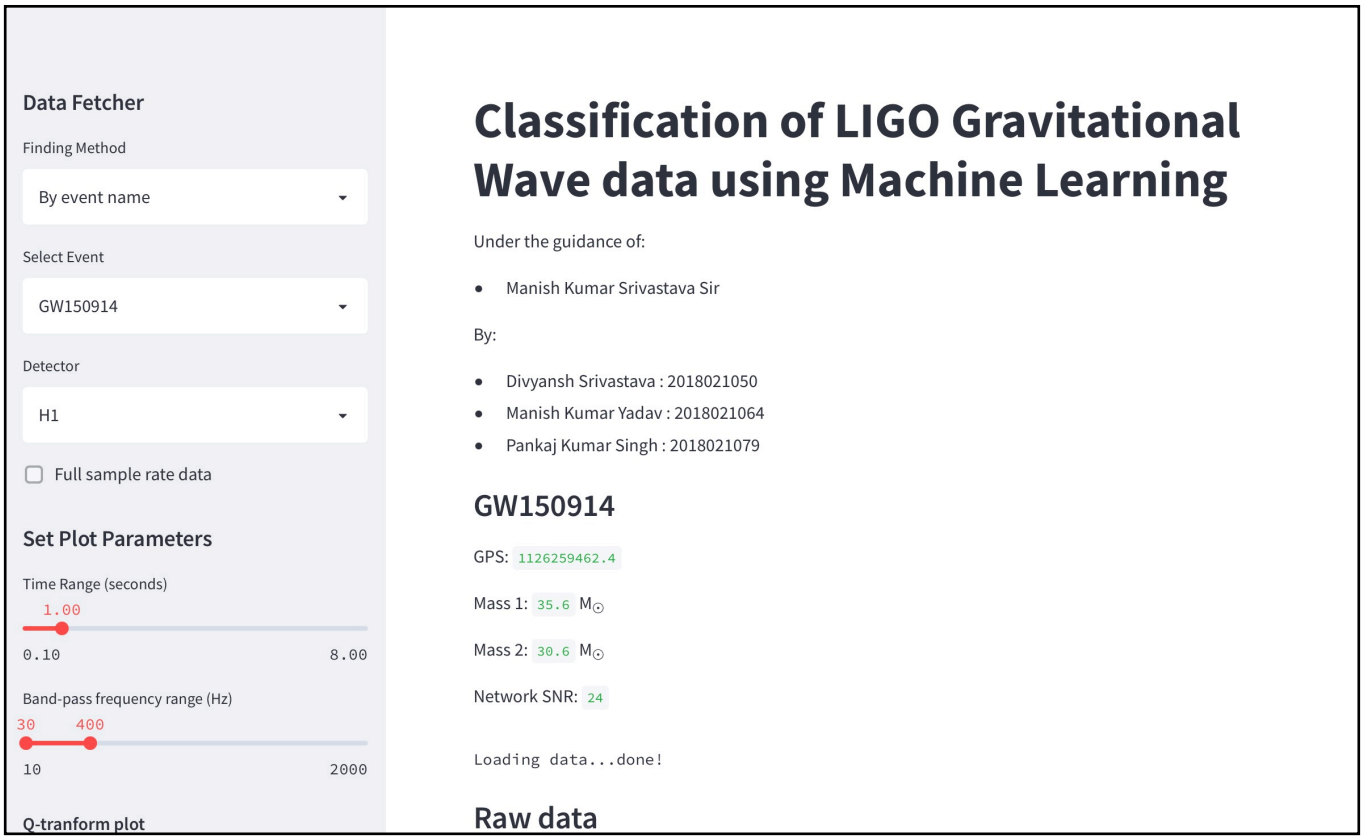
```
X_train,X_test,y_train,y_test = train_test_split(ImagePixels,ohlabel,test_size=0.0001,random_state=32,shuffle
= True)
In [8]:

plt.imshow(X_train[0])
plt.show()
```

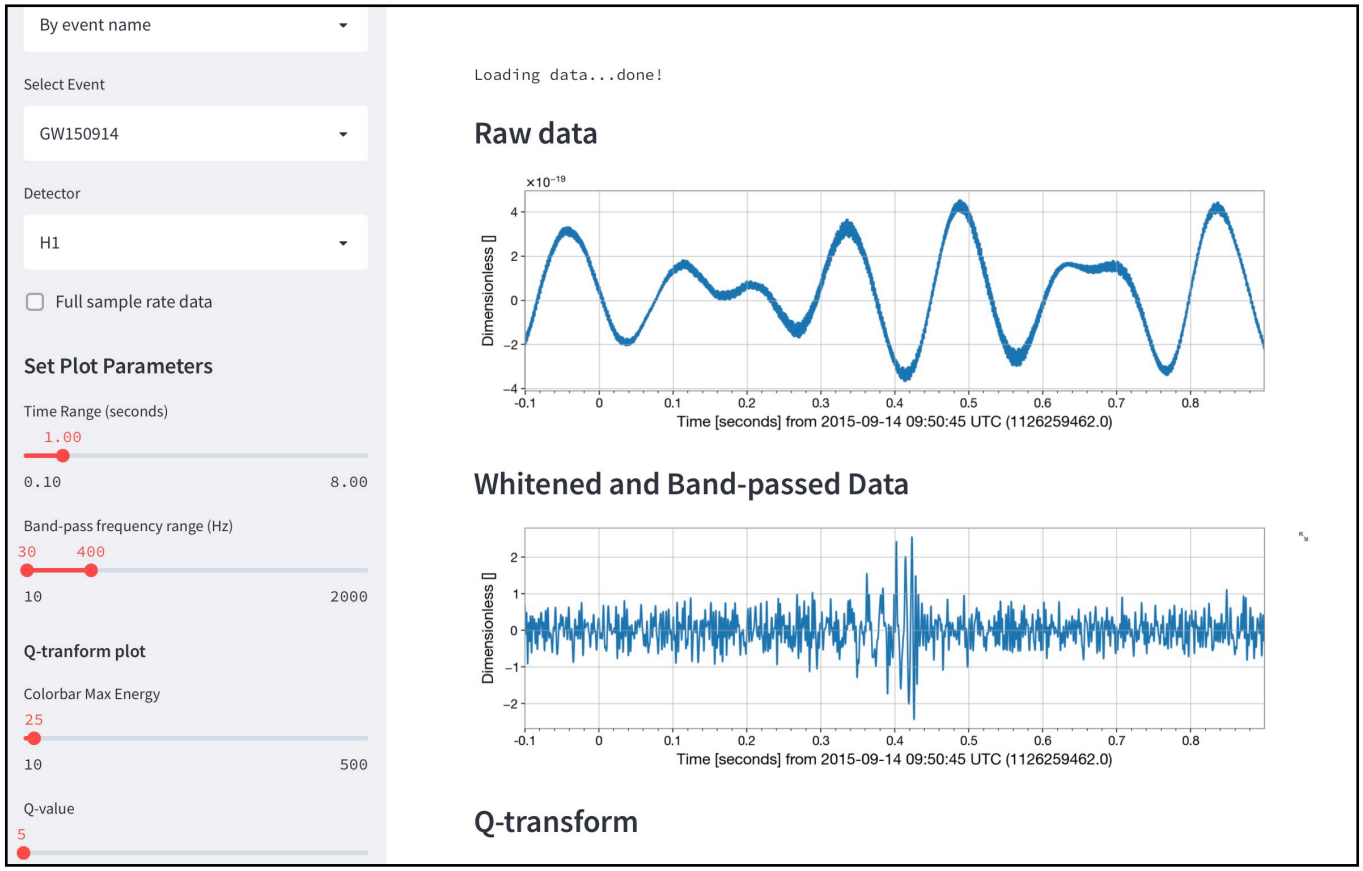


In [9]:

```
keras.backend.clear_session()
vgg = keras.applications.VGG16(input_shape=(224,224,3),include_top=False,weights =
'imagenet',pooling='max')
vgg.trainable = False
vggmodel = keras.Sequential([vgg
                  ,Dense(500,activation='tanh'),Dense(500,activation='tanh'),Dense(500,activation='tanh'),Dense(t
otalclasses,activation='softmax')])

vggmodel.compile(optimizer = 'adam',loss = 'categorical_crossentropy',metrics=['accuracy'])
vggmodel.summary()

hist =
vggmodel.fit(X_train,y_train,epochs=no_of_epochs,validation_split=validation_split,batch_size=batch_size)
```

Platform Demonstration Snippet





By event name

Select Event

GW150914

Detector

H1

Full sample rate data

Set Plot Parameters

Time Range (seconds)

1.00

0.10 8.00

Band-pass frequency range (Hz)

30 400

10 2000

Q-tranform plot

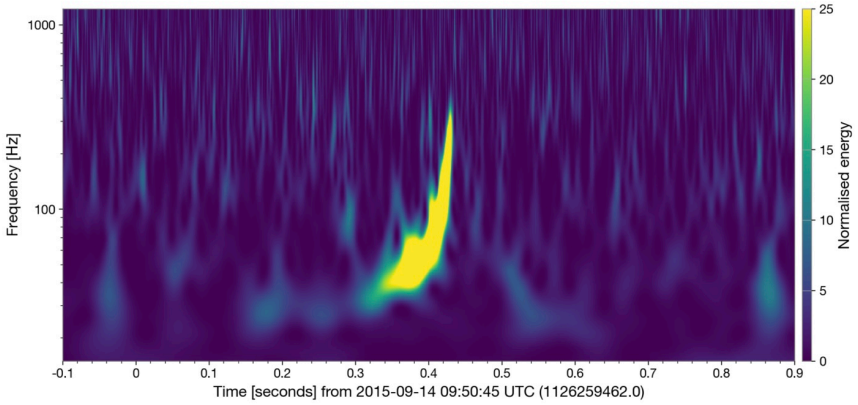
Colorbar Max Energy

25

10 500

Q-value

Q-transform



Classification

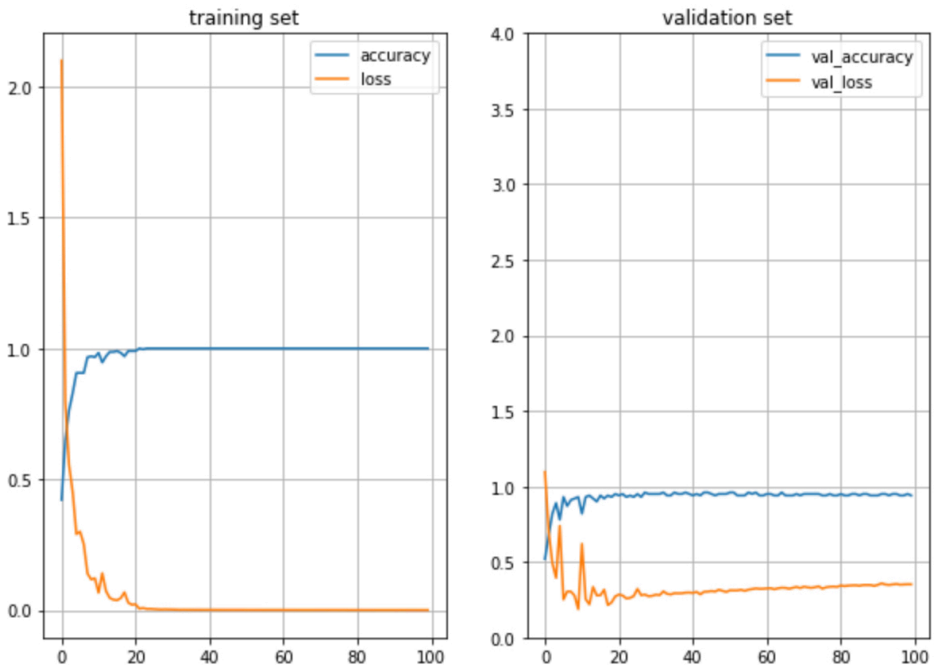
The above spectrogram most likely belongs to Chirp class.

Conclusion and future work

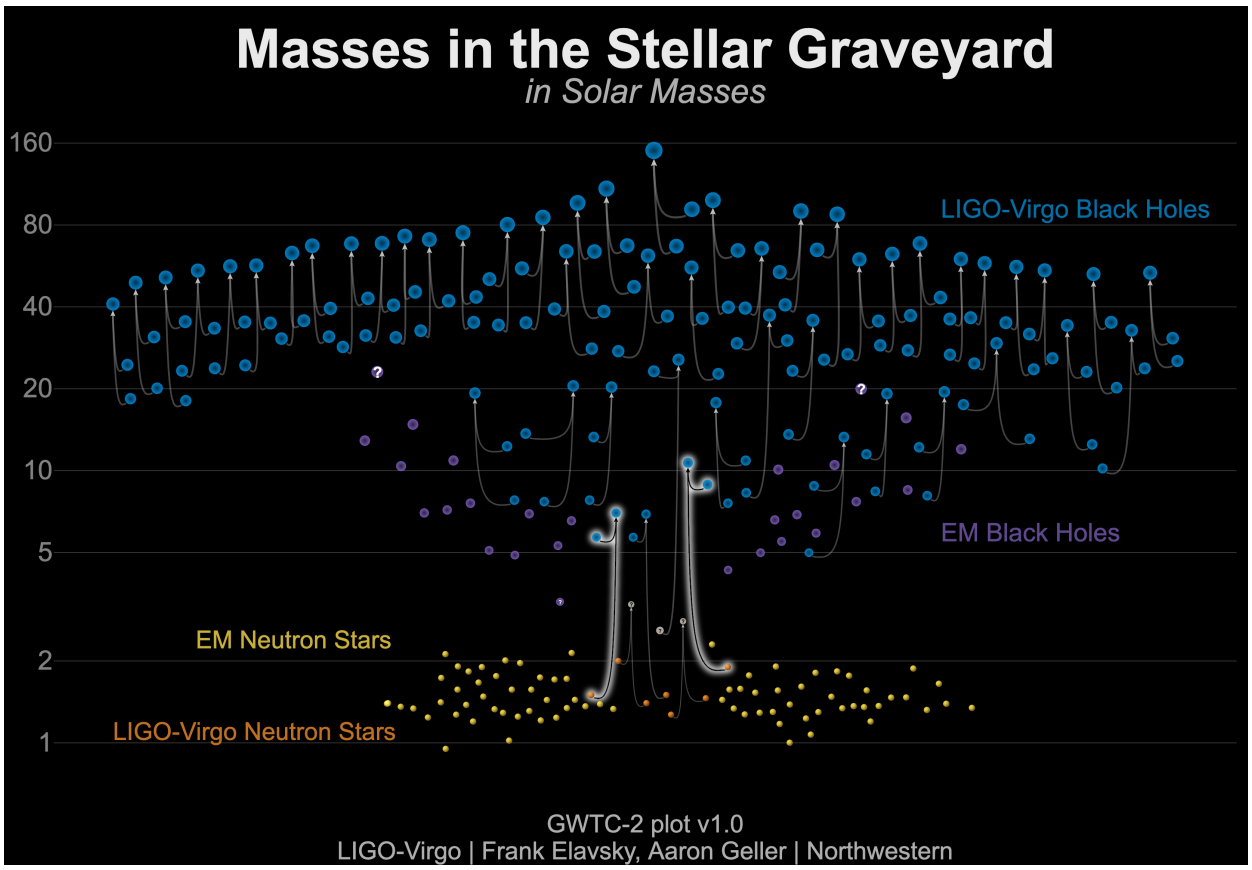
Our model was successfully able to classify the data into the aforesaid labels with validation accuracy of 0.94. Here are the details of a last five iterations of validating the model's performance.

```
Epoch 95/100
10/10 [==============================] - 1s 87ms/step - loss: 1.6398e-04
- accuracy: 1.0000 - val_loss: 0.3512 - val_accuracy: 0.9500
Epoch 96/100
10/10 [==============================] - 1s 88ms/step - loss: 1.6803e-04
- accuracy: 1.0000 - val_loss: 0.3540 - val_accuracy: 0.9500
Epoch 97/100
10/10 [==============================] - 1s 87ms/step - loss: 1.5792e-04
- accuracy: 1.0000 - val_loss: 0.3497 - val_accuracy: 0.9400
Epoch 98/100
10/10 [==============================] - 1s 85ms/step - loss: 1.5451e-04
- accuracy: 1.0000 - val_loss: 0.3509 - val_accuracy: 0.9400
Epoch 99/100
10/10 [==============================] - 1s 86ms/step - loss: 1.5409e-04
- accuracy: 1.0000 - val_loss: 0.3522 - val_accuracy: 0.9500
Epoch 100/100
10/10 [==============================] - 1s 85ms/step - loss: 1.5027e-04
- accuracy: 1.0000 - val_loss: 0.3518 - val_accuracy: 0.9400
```

The link to our model is https://www.kaggle.com/code/divyansh050/ligo-gwclassifier. A demonstration setup for this project has been coded in streamlit.



The capability of our machine learning model to classify the gravitational wave data can be extended in future to improve the detection sensitivity of the LIGO detectors by employing this model on real time data and by incorporating all the possible labels of data.



The gravitational wave programs have detected more neutron stars and blackholes than the telescopes that pick electromagnetic signals. Thus, making it clear that gravitational wave detection is more effective.

References

- LIGO Scientific Collaboration : https://www.ligo.org/detections/GW190814.php
- Unsupervised Clustering of LIGO data : https://journals.aps.org/prd/abstract/10.1103/PhysRevD.97.101501
- Deep Transfer Learning : https://ui.adsabs.harvard.edu/abs/2018PhRvD..97j1501G/abstract
- GW Open Data Workshop, May 10-14, 2021 : https://gw-odw.thinkific.com/courses/gw-open-data-workshop-4
- GW Open Data Workshop, May 23-25, 2022 : https://gw-odw.thinkific.com/courses/take/workshop-5
- Gravitational Wave Open Source Center : https://www.gw-openscience.org