School of Computer Science and Engineering

Odd Sem 2021-22

DMA Course Project Report

On

<u>GRAVITATIONAL WAVE DETECTION</u>

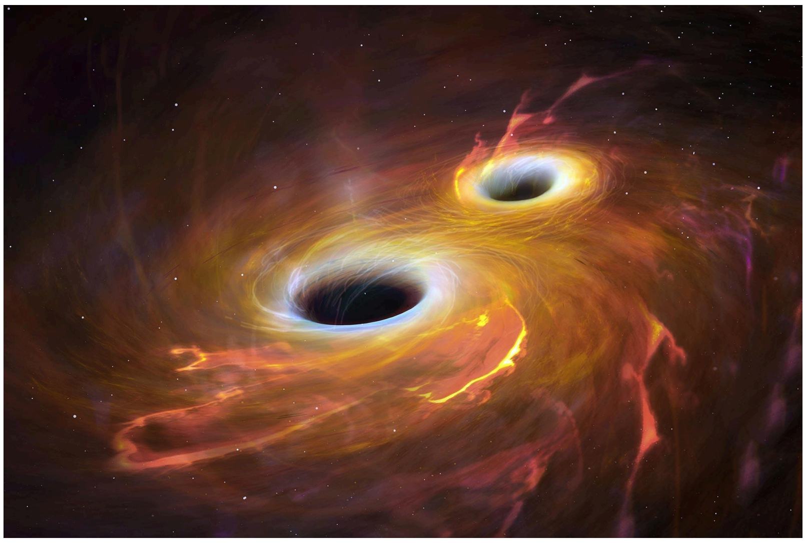
Submitted by : Team 5D03

Tanmayi Shurpali	01FE19BCS238
Shrinidhi Kulkarni	01FE19BCS241
Bhavana Kumbar	01FE19BCS244
Avantika Shrivastava	01FE19BCS253

Table of Contents

S.no	Content	
	Introduction	
1.	**Exploratory Data Analysis**	
	1.1	Understanding Data
	1.2	Raw Data Visualization
2.	**Data Preprocessing**	
	2.1	Windowing
	2.2	Whitening
	2.3	CQ-Transform
	2.4	Data Pipeline
3.	**Model Training**	
	3.1	Strategy
	3.2	Baseline Model
	3.3	Advanced Model
	3.4	Evaluation
4.	**Results & Conclusion**	
5.	**Future Scope**	
	References	

Introduction



Gravitational Waves have been discussed since the beginning of the 20th century, and scientifically researched since Einstein's General Theory of Relativity. They are caused by massive celestial bodies, like the Neutron Stars or Black Holes, when they accelerate they cause gravitational waves, in the form of waves, propagating through the curvature of space-time at the speed of light. These disturbances can be felt on the other side of the observable universe, but are extremely weak as they lose energy as gravitational radiation. It can be imagined similar to throwing a pebble in the pond, the site where the pebble hits water is the source of the disturbance and the outgoing ripples, are the gravitational waves, that get weaker as they move away from the source.In February 2015, the Laser Interferometer Gravitational-wave Observatory (LIGO) Scientific Collaboration and the Virgo Collaboration announced the first observation of a Gravitational-Wave (GW) signal from a stellar-mass Compact Binary Coalescence (CBC) system .Despite all the initial successes, the future of GW astronomy is facing many challenges. Because of the effectiveness of ML algorithms in identifying patterns in data, ML techniques may be harnessed to make all these searches more sensitive and robust. Applications of ML algorithms to GW searches range from building automated data analysis methods for low-latency pipelines to distinguishing terrestrial noise from astrophysical signals and improving the reach of searches.

The dataset is available from the G2Net Gravitational Wave Detection[1] on kaggle, but since this website is run in the Kaggle notebook environment, we use the local relative links to import the data. The entire project is uploaded in Kaggle [2].

Responsibilities

Avantika worked on exploring and visualizing data.She was also tasked with the preprocessing data, training model and final editing and formatting of this document and creating Kaggle notebook.

Tanmayi worked on exploring and handling data.She was also tasked with the preprocessing data, training model and final editing and formatting of this document and creating Kaggle notebook.

Shrinidhi worked on exploring data and it's analysis .She was also tasked with the training model and final editing and formatting of this document and creating Kaggle notebook.

Bhavana is tasked with doing necessary literature survey and exploring data.She was also tasked with the training model and final editing and formatting of this document and creating Kaggle notebook.

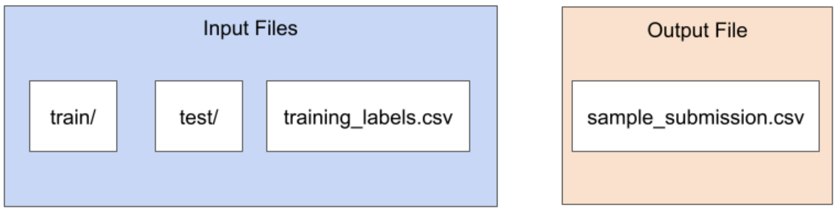
Problem Statement

To preprocess data then build, train & evaluate binary classification models to predict if the given set of signals has Gravitational Waves in them or not.

1.Explorative Data Analysis

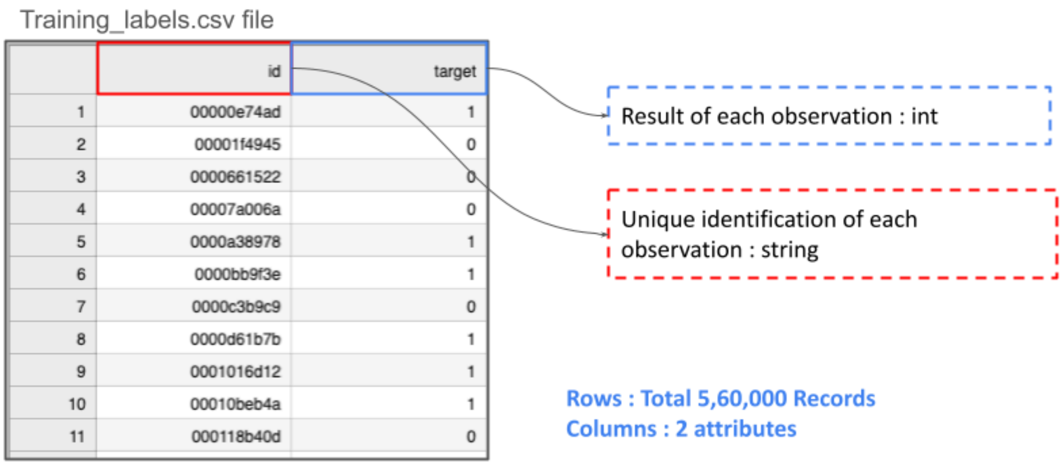
Exploratory Data Analysis is an approach in analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. EDA assists us in getting a better understanding of data, identifying various data patterns and getting a better understanding of the problem statement

<u>1.1 Understanding Data</u>



We were provided with two folders - train/ and test/ and two csv files- training_labels. Following is the description of each:

1. **train/** - the training set files, one npy file per observation; labels are provided in a files shown below
2. **test/** - the test set files; you must predict the probability that the observation contains a gravitational wave
3. **training_labels.csv** - target values of whether the associated signal contains a gravitational wave
4. **sample_submission.csv** - a sample submission file in the correct format.


Training_labels.csv file

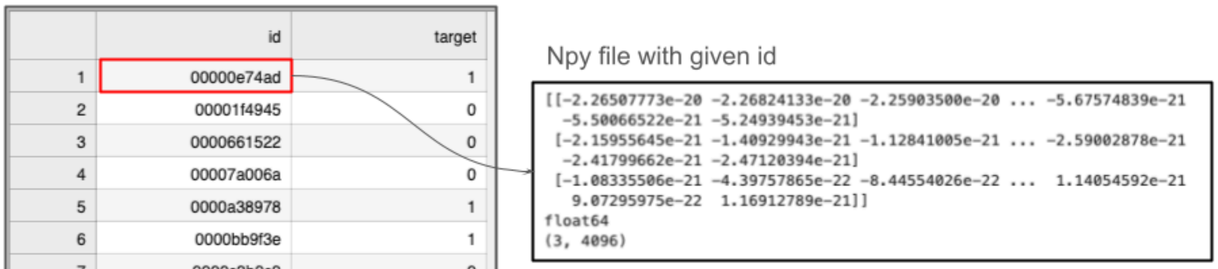
	id	target
1	00000e74ad	1
2	00001f4945	0
3	0000661522	0
4	00007a006a	0
5	0000a38978	1
6	0000bb9f3e	1
7	0000c3b9c9	0
8	0000d61b7b	1
9	0001016d12	1
10	00010beb4a	1
11	000118b40d	0

Result of each observation : int

Unique identification of each observation : string

Rows : Total 5,60,000 Records
Columns : 2 attributes

Training_labels.csv file



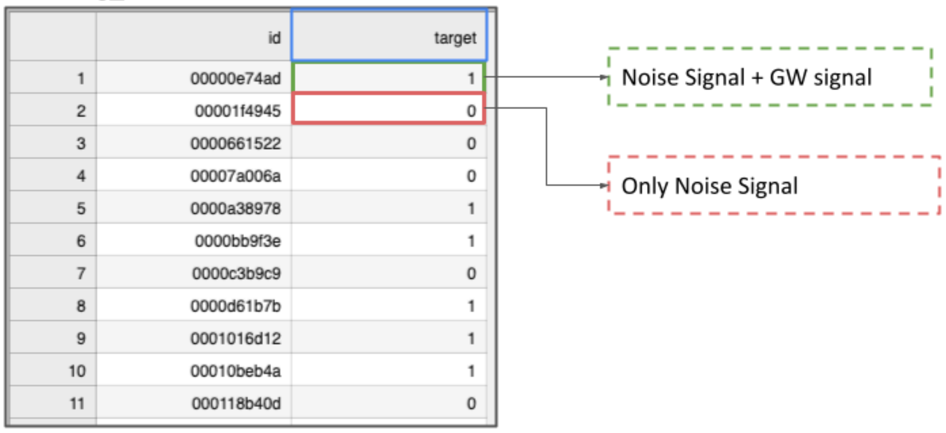
	id	target
1	00000e74ad	1
2	00001f4945	0
3	0000661522	0
4	00007a006a	0
5	0000a38978	1
6	0000bb9f3e	1
7	0000c3b9c9	0
8	0000d61b7b	1
9	0001016d12	1
10	00010beb4a	1
11	000118b40d	0

Npy file with given id

```
[[-2.26507773e-20 -2.26824133e-20 -2.25903500e-20 ... -5.67574839e-21
  -5.50066522e-21 -5.24939453e-21]
 [-2.15955645e-21 -1.40929943e-21 -1.12841005e-21 ... -2.59002878e-21
  -2.41799662e-21 -2.47120394e-21]
 [-1.08335506e-21 -4.39757865e-22 -8.44554026e-22 ...  1.14054592e-21
   9.07295975e-22  1.16912789e-21]]
float64
(3, 4096)
```

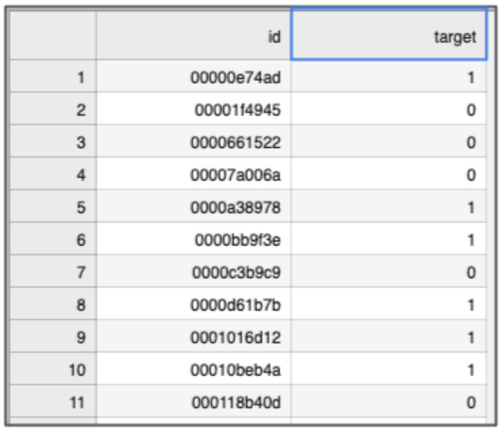
- 4096 is number of data points, each time series instance spans over 2 seconds and is sampled at 2048 Hz
- 3 is the number of detectors

Training_labels.csv file

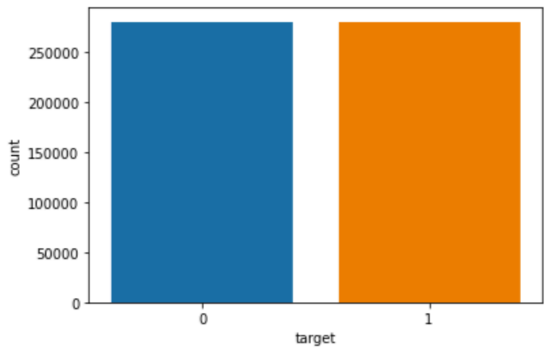


	id	target
1	00000e74ad	1
2	00001f4945	0
3	0000661522	0
4	00007a006a	0
5	0000a38978	1
6	0000bb9f3e	1
7	0000c3b9c9	0
8	0000d61b7b	1
9	0001016d12	1
10	00010beb4a	1
11	000118b40d	0

Training_labels.csv file



	id	target
1	00000e74ad	1
2	00001f4945	0
3	0000661522	0
4	00007a006a	0
5	0000a38978	1
6	0000bb9f3e	1
7	0000c3b9c9	0
8	0000d61b7b	1
9	0001016d12	1
10	00010beb4a	1
11	000118b40d	0



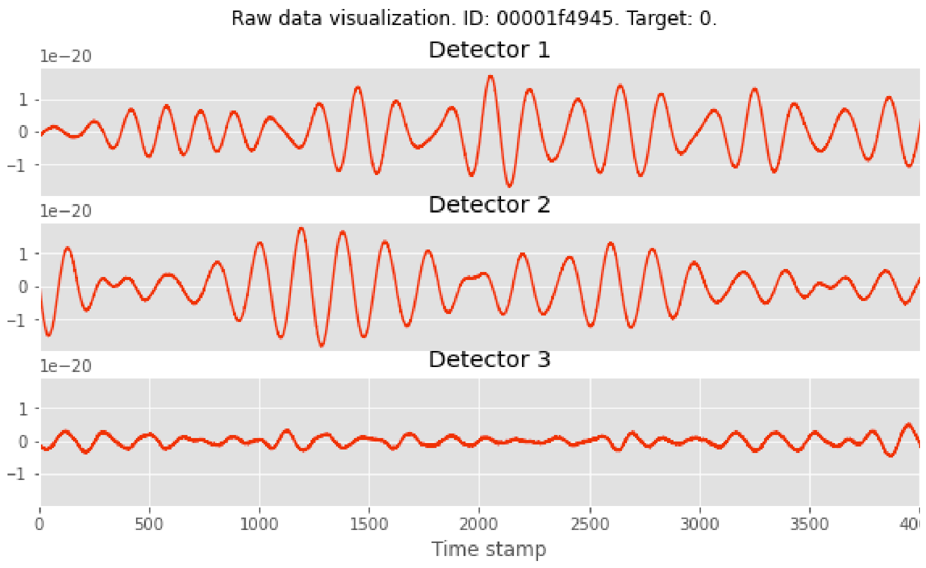
```
0    280070
1    279930
Name: target, dtype: int64
```

- Training_labels csv file has two columns id and target.
- There are 5,60,000 records in the file.
- There are no-null values in the file given
- The column - ‘id’ represents unique identification of npy files (samples) and ‘target’ represents probability of detecting GW wave signal

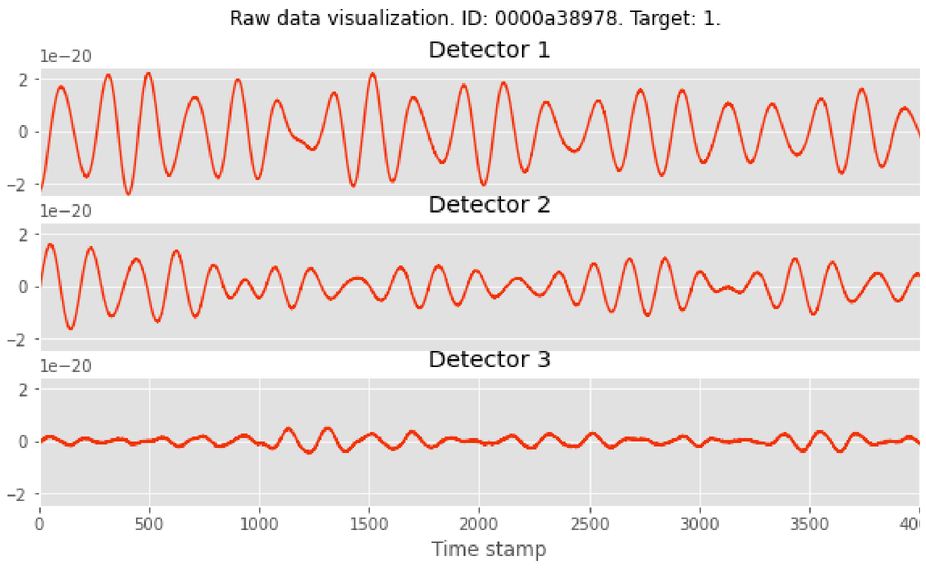
1.2 Raw Data Visualization

A single npy file is taken as an example strain to plot data points. Target '0' indicates the **negative sample** and target '1' indicates **positive sample.**

1. By Time-Domain Graphs



Time-Domain Graph for Negative Sample



Time-Domain Graph for Positive Sample

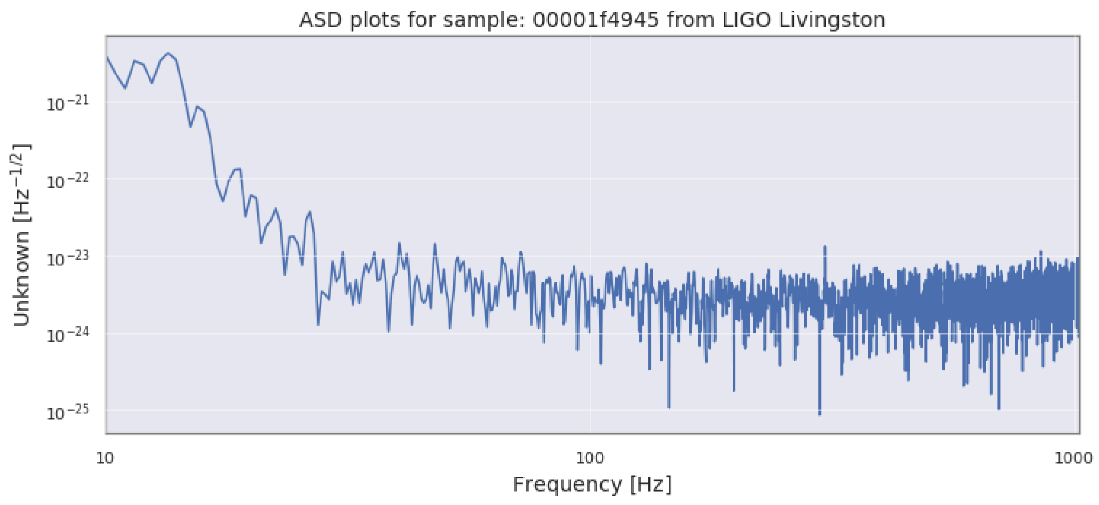
- The three signals originating from different detectors all look a bit different.
- It is difficult to infer just by looking whether the given wave has a GW signal or not as there is no notable difference between graphs of target - 0 and target - 1.
- The strain is of the order 10^ -20
- , which is extremely small and can be affected by many external factors. However, as seen in both the sample plots, the strain data is a combination of many frequencies and analysing the signals in frequency domain, instead of the time domain, might give us better insights.
- Depending on the location the amplitude recorded varies (smaller amplitudes indicating weak signal detection.
- Astrophysical signals have typical amplitudes comparable to the detector background noise. Therefore, characterization and reduction of detector noise is essential to GW searches.
- The interferometer is sensitive towards gravitational waves but unfortunately also for terrestrial forces and displacements.This may also include vibrations of the instruments themselves etc.. This kinds of forces cause stretching of the interferometer arms and this leads to constructive interference and the waves we can see above.

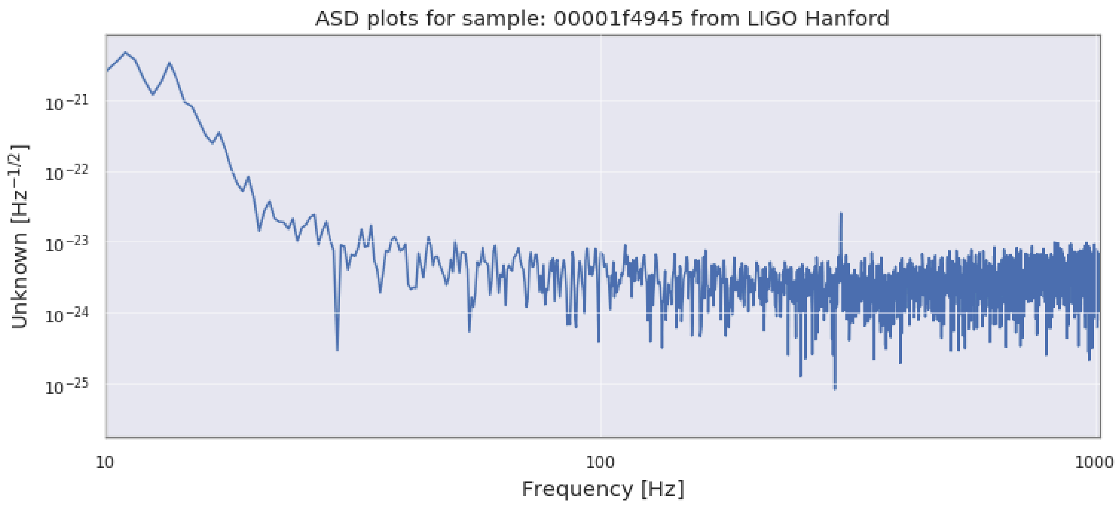
---

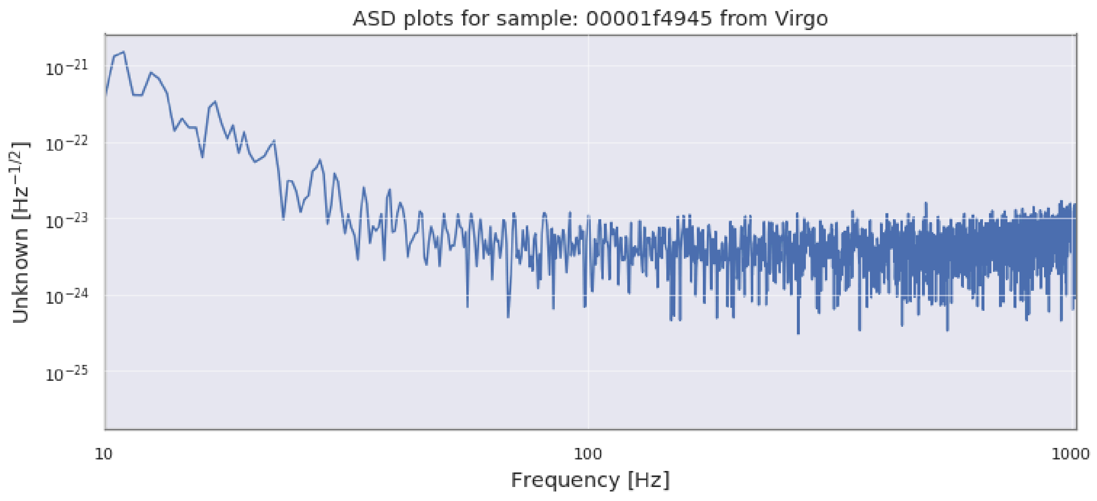
2. By Frequency Domain Graphs

ASD(Amplitude Spectral Density)

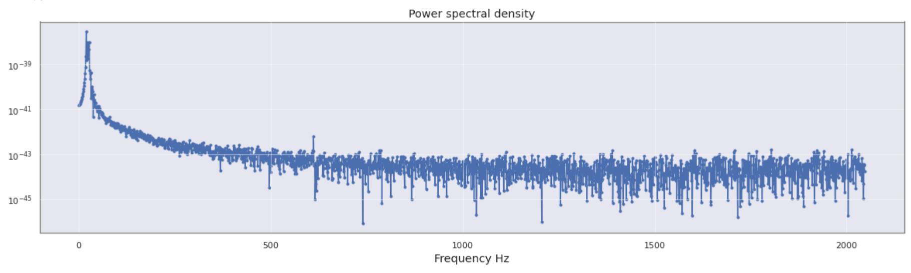
One of the ways to visualize a raw signal in the frequency domain is by plotting the amplitude spectral density (ASD).
These plots are plotted on a log scale for x-axis, and we see that it ranges from 10 Hz ~ 1000 Hz. Although these limits are for visualization purposes only, it helps us see some peaks for each observatory. A particular frequency can be peculiar in one measurement but remember that the GW signal has to be detected in all three waves to be confirmed. This data here still seems a bit noisy and as shown in the tutorial, if sampled for longer periods of time (on real data), it can give some valuable insights. However, the data in this competition is simulated and we try to find other ways to visualize it.


ASD plots for sample: 00001f4945 from LIGO Livingston
10^{-21}
10^{-22}
10^{-23}
10^{-24}
10^{-25}
Unknown [$Hz^{-1/2}$]
10
100
1000
Frequency [Hz]


ASD plots for sample: 00001f4945 from LIGO Hanford
10^{-21}
10^{-22}
10^{-23}
10^{-24}
10^{-25}
Unknown [$Hz^{-1/2}$]
10
100
1000
Frequency [Hz]


ASD plots for sample: 00001f4945 from Virgo
10^{-21}
10^{-22}
10^{-23}
10^{-24}
10^{-25}
Unknown [$Hz^{-1/2}$]
10
100
1000
Frequency [Hz]

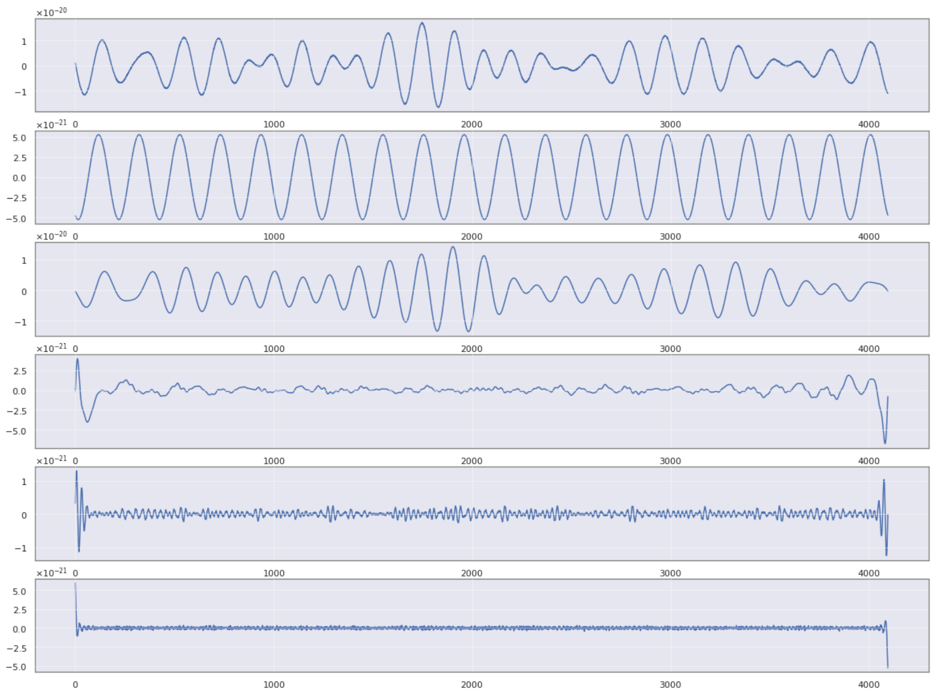
PSD(Power Spectral Density)



- The steep shape at low frequencies is dominated by noise related to ground motion.
- Above roughly 100 Hz, the Advanced LIGO detectors are currently quantum noise limited, and their noise curves are dominated by shot noise.
- High amplitude noise features are also present in the data at certain frequencies, including lines due to the AC power grid (harmonics of 60 Hz in the U.S. and 50 Hz in Europe), mechanical resonances of the mirror suspensions, injected calibration lines, and noise entering through the detector control systems.

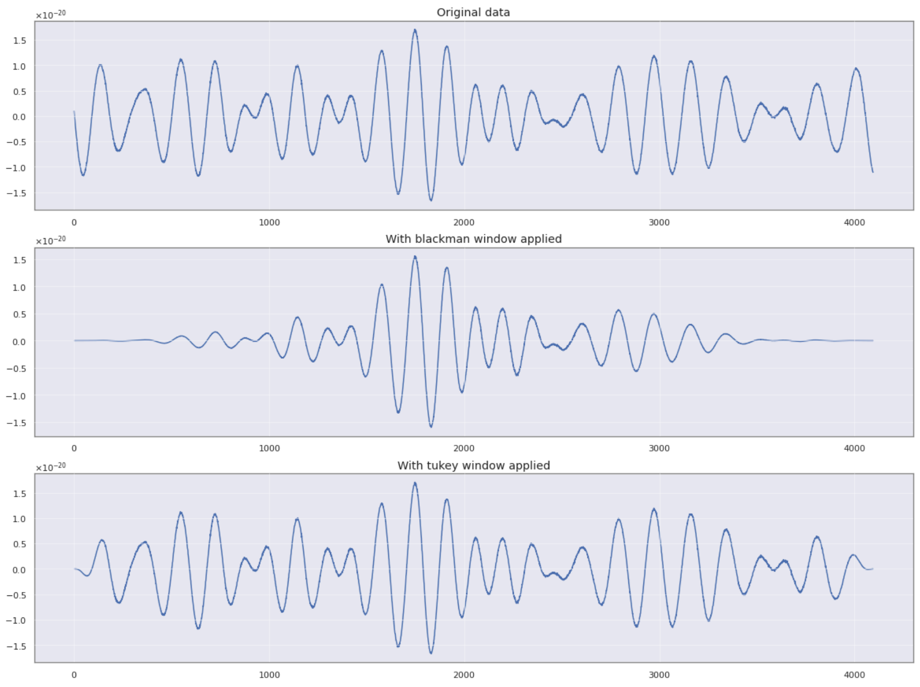
FFT(Fast Fourier Transform)

- The first step in many LVC analyses is to Fourier transform the time-domain data using a fast Fourier transform (FFT) . - - There are interesting kinds of side effects in the beginning and the end of each wave after doing the inverse Fourier transform.
- Since the FFT implicitly assumes that the stretch of data being transformed is periodic in time, window functions have to be applied to the data to suppress spectral leakage using e.g. a Tukey (cosine-tapered) window function.
- Failing to window the data will lead to spectral leakage and spurious correlations in the phase between bins.



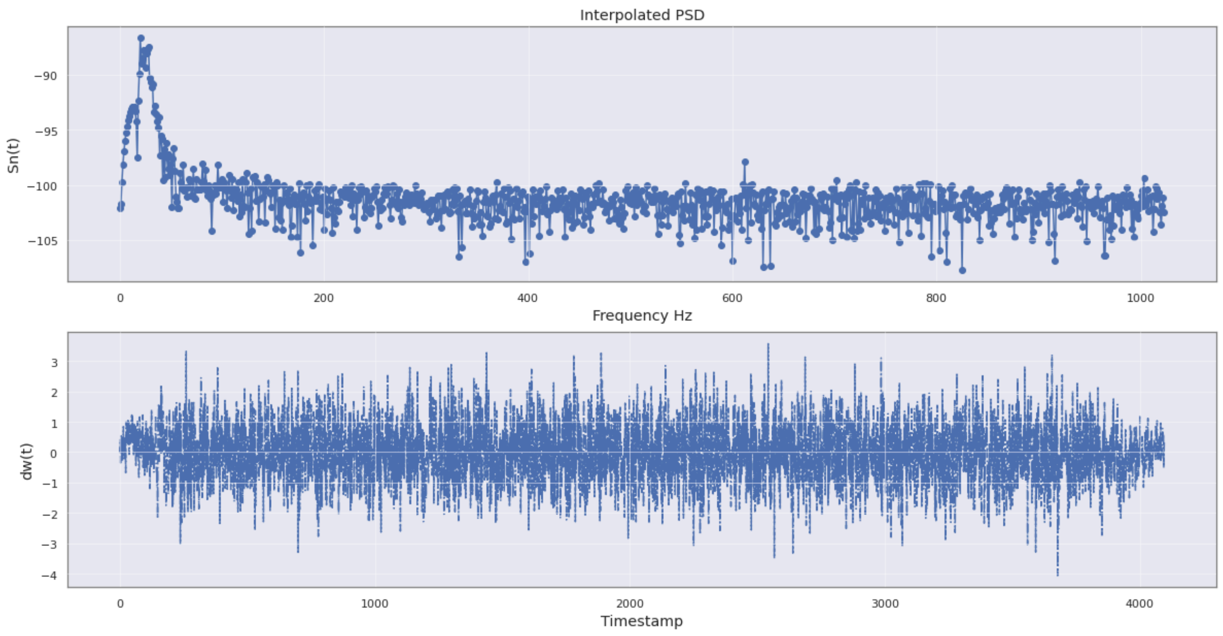
2. Data Preprocessing

2.1 Windowing



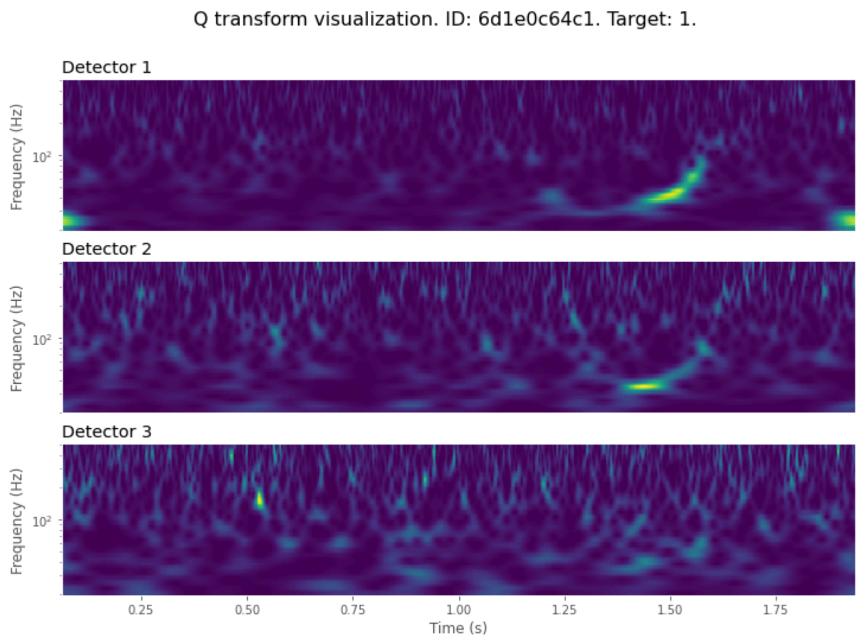
For the analysis of transient data the use of Tukey windows is advantageous as signals will suffer less modification than Blackman windows.

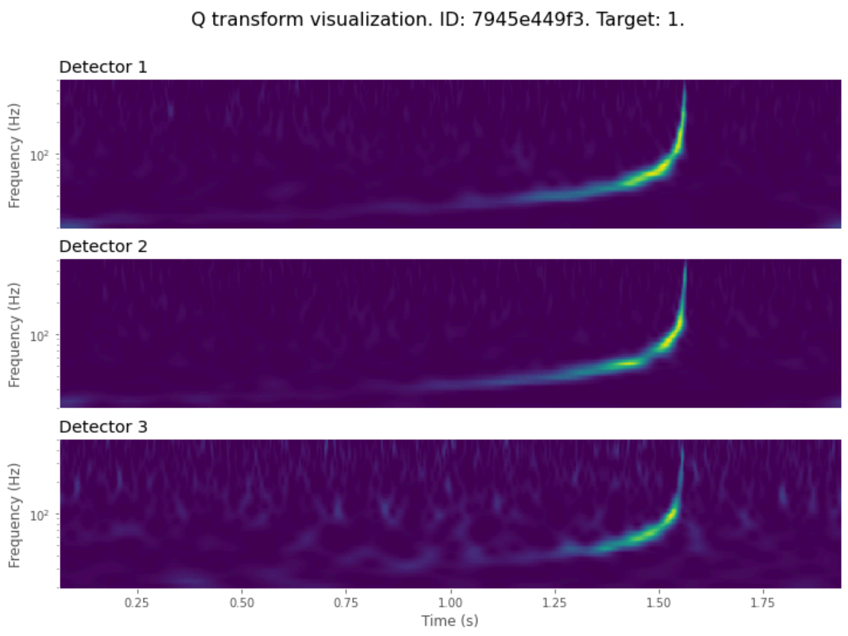
2.2 Whitening - Making Signal more uniform



- Whitening the data is suppressing the extra noise at low frequencies and at the spectral lines, to better see the weak signals in the most sensitive band.
- It is always one of the first steps in astrophysical data analysis (searches, parameter estimation).
- It requires no prior knowledge of spectral lines, etc; only the data are needed.

2.3 CQ-Transform





- The signal analysis didn't provide much insight, so let's try the second method in signal processing. Transforming the waves into spectrogram images, i.e. frequency-domain, and then visualizing them. This technique is widely used in audio analysis and since our data is a wave with a bunch of frequencies, we can use the same technique as well.
- The advantage of using a spectrogram, over a direct Fourier Transform where you lose time info, is that it captures the shift or change in frequencies over time and this removes white noise frequencies that are persistent, leaving the signals of interest. Constant Q-Transform is one way to visualize the spectrogram.
- Visibly, all three signals have different features and the above were plotted from a sample which has gravitational waves, and it shows the famous 'chirp' confirming the presence of gravitational waves.
- This transformation removes the unwanted noise frequencies, but still some of it remains, but a signal has to be detected in all three waves to be predicted as a gravitational wave.
- Next, we can compare how the Q-Transforms look for samples with and without gravitational wave signals.
- Apart from a few hints, we cannot say for sure that the difference between the waves with and without GW signals is obvious.
- There can be some cleaning or filtering we can apply to remove the noise further but that's where Deep Learning shines.
- The things we can't detect visually, machine learning can. Next, we build data pipelines, transform the data to spectrograms, and build models to make the predictions.

2.4 Data Pipeline : Next, we create the TensorFlow input data pipeline[6]. This is crucial as loading such a huge dataset can create a bottleneck on the entire workflow and can cause memory overload.

- A well structured data pipeline, if used, can create an efficient workflow by consolidating all the data imports, preprocessing & manipulations into cleanly defined functions.
- This can be unavoidable, if dealing with large datasets, such as in this project, which might not be possible to load at once due to memory limitations.
- TensorFlow's tf.data API can assist in building this complex yet flexible and modular pipeline.
- It also enables us to optimize our workflow for GPU and TensorFlow provides methods that can be run on GPU instead, while handling data as tf.tensor, that can run more than 5 times faster depending on the situation.

3. Training Models

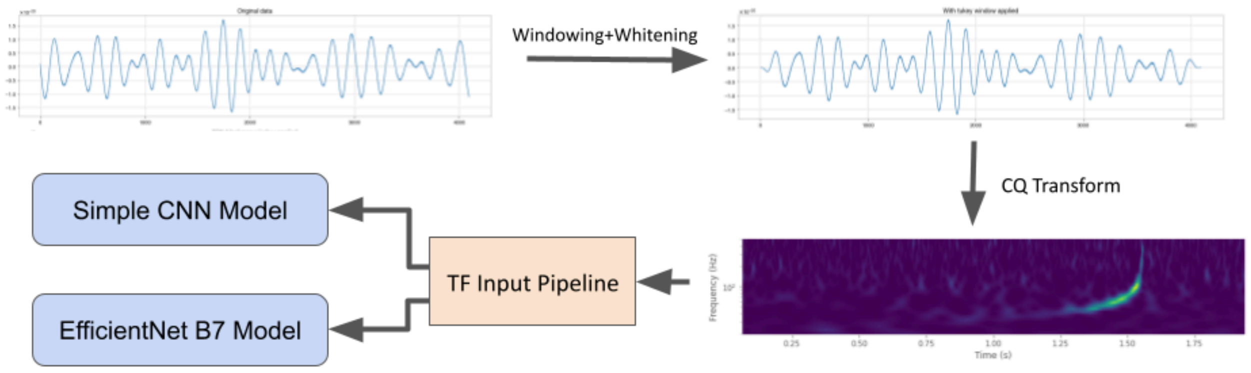
3.1 Modelling Strategy

This is essentially a signal processing problem with classification tasks; there can be two ways in which we can build models around this data, as also mentioned in the LIGO research paper - using "raw" signals with minimal pre-processing and using "images" by transforming the waves into spectrograms.

However, building models on raw signal data, by following the cleaning steps from respective publications, did not yield acceptable results. It is worth mentioning that only a part of the data was used during the strategy selection process, and it was concluded that more pre-processing was necessary, or rather proper pre-processing, if we were to use raw signal.

Eventually, the second method that we went with in this project, is used to transform the waves into the spectrogram image. We train two models to evaluate the results:

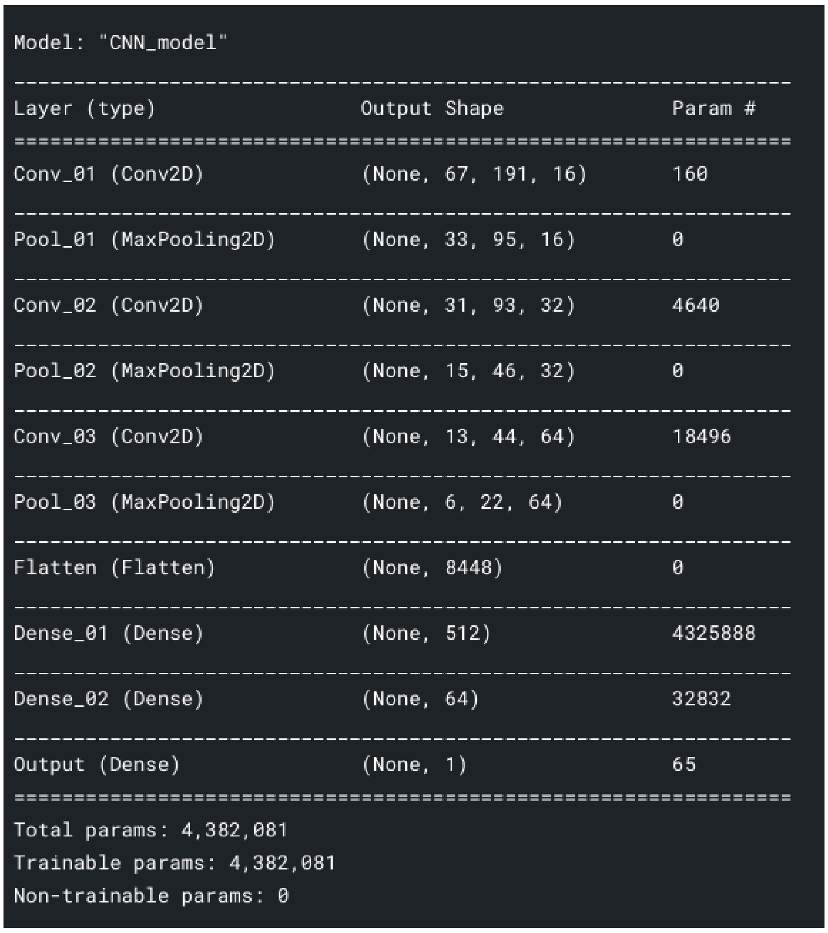
1. Simple CNN- a simple CNN architecture that is a modified version of the model usually used in MNIST Digit Recognizer tutorials. This acts as our baseline model.
2. EfficientNet-a EfficientNetB7 model that has been developed and pre-trained on ImageNet dataset. This model is chosen as it is known for its excellent performance with significantly fewer number of parameters, that can drastically improve the computational efficiency



3.2 Baseline Model - Simple CNN

Since now our problem is image classification we could use any algorithm that works well for us. We wanted our training model to study images in detail so we went with CNN as it uses multiple layers strategy. We built and trained a simple CNN model as our baseline model before we build more advanced models.

CNN is a type of neural network model which allows us to extract higher representations for the image content. Unlike the classical image recognition where we define the image features yourself, CNN takes the image's raw pixel data, trains the model, then extracts the features automatically for better classification.

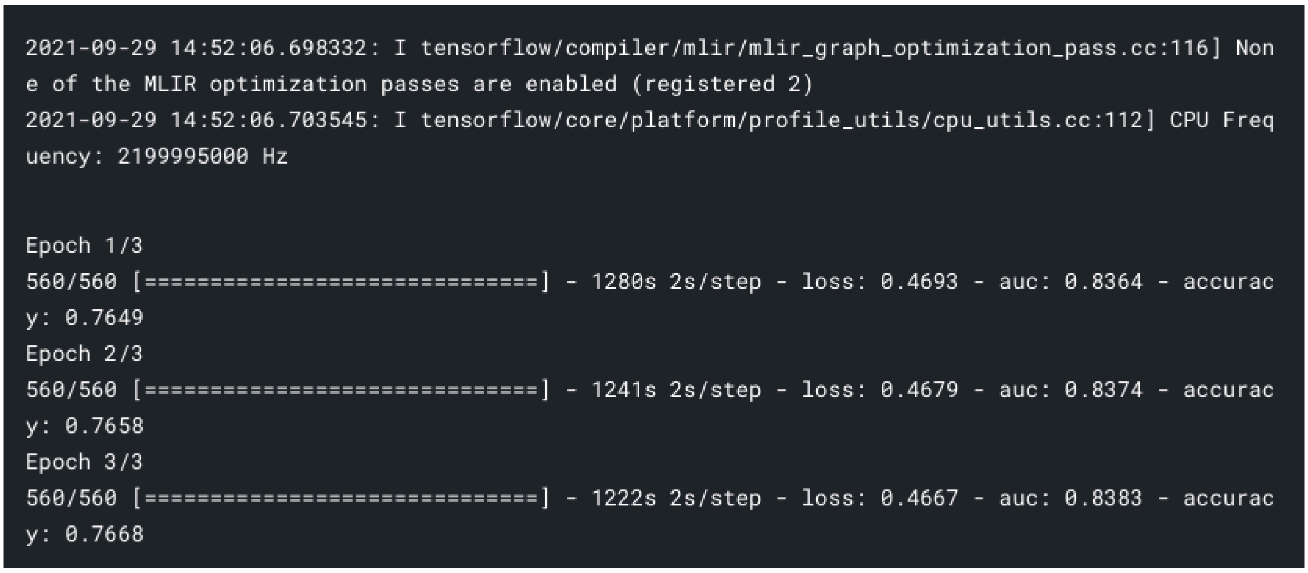


```
Model: "CNN_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv_01 (Conv2D)             (None, 67, 191, 16)       160
_________________________________________________________________
Pool_01 (MaxPooling2D)       (None, 33, 95, 16)        0
_________________________________________________________________
Conv_02 (Conv2D)             (None, 31, 93, 32)        4640
_________________________________________________________________
Pool_02 (MaxPooling2D)       (None, 15, 46, 32)        0
_________________________________________________________________
Conv_03 (Conv2D)             (None, 13, 44, 64)        18496
_________________________________________________________________
Pool_03 (MaxPooling2D)       (None, 6, 22, 64)         0
_________________________________________________________________
Flatten (Flatten)            (None, 8448)              0
_________________________________________________________________
Dense_01 (Dense)             (None, 512)               4325888
_________________________________________________________________
Dense_02 (Dense)             (None, 64)                32832
_________________________________________________________________
Output (Dense)               (None, 1)                 65
=================================================================
Total params: 4,382,081
Trainable params: 4,382,081
Non-trainable params: 0
```

CNN Architecture that we used

Since we wanted to validate our model we split the training data provided in the competition into train and test

```
Test-Train Split : 20-80
```



```
2021-09-29 14:52:06.698332: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] Non
e of the MLIR optimization passes are enabled (registered 2)
2021-09-29 14:52:06.703545: I tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Freq
uency: 2199995000 Hz

Epoch 1/3
560/560 [==============================] - 1280s 2s/step - loss: 0.4693 - auc: 0.8364 - accurac
y: 0.7649
Epoch 2/3
560/560 [==============================] - 1241s 2s/step - loss: 0.4679 - auc: 0.8374 - accurac
y: 0.7658
Epoch 3/3
560/560 [==============================] - 1222s 2s/step - loss: 0.4667 - auc: 0.8383 - accurac
y: 0.7668
```

Validating Model using test data

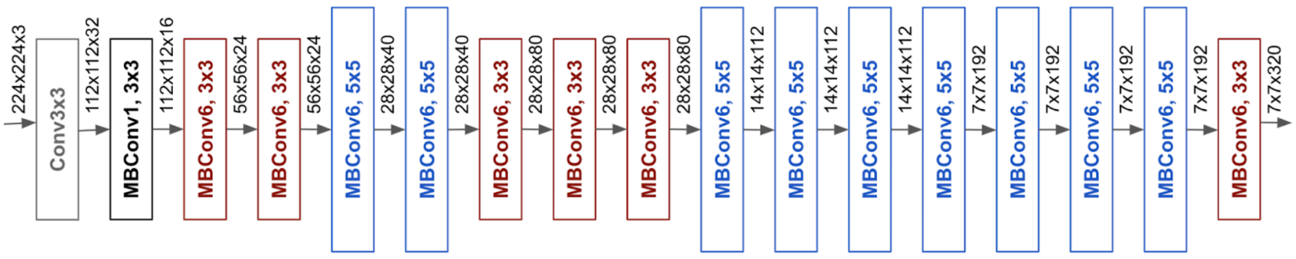
- The baseline model seems to be converging well only after about 3 epochs.
- But, it takes almost an hour to train each epoch, we can only say that the model can be improved with further training and fine-tuning the structure.
- At the end of 3rd epoch, we see 0.83 AUC score and 0.76 accuracy for the training dataset, while 0.84 AUC score and 0.77 accuracy for the validation dataset.

3.3 Advanced Model - EfficientNet B7

Convolutional neural networks (CNNs) are commonly developed at a fixed resource cost, and then scaled up in order to achieve better accuracy when more resources are made available. For example, ResNet can be scaled up from ResNet-18 to ResNet-200 by increasing the number of layers, and recently, GPipe achieved 84.3% ImageNet top-1 accuracy by scaling up a baseline CNN by a factor of four. The conventional practice for model scaling is to arbitrarily increase the CNN depth or width, or to use larger input image resolution for training and evaluation. While these methods do improve accuracy, they usually require tedious manual tuning, and still often yield suboptimal performance.

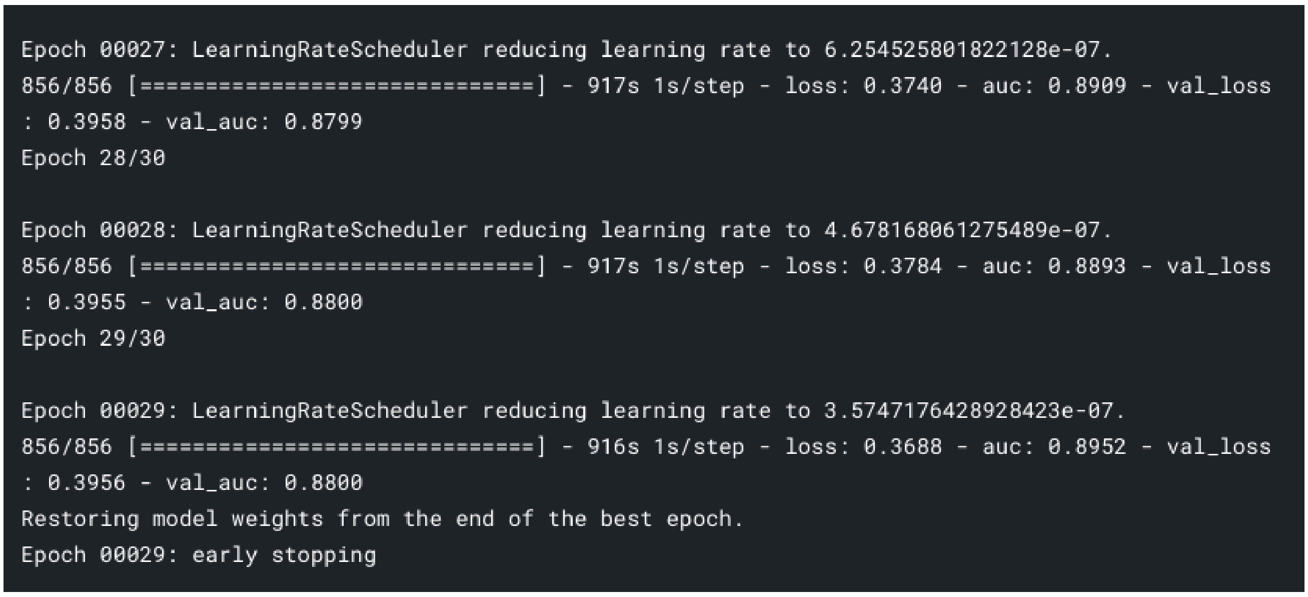
Compound Model Scaling: A Better Way to Scale Up CNNs

In order to understand the effect of scaling the network, we systematically studied the impact of scaling different dimensions of the model. While scaling individual dimensions improves model performance, we observed that balancing all dimensions of the network—width, depth, and image resolution—against the available resources would best improve overall performance. This is where EfficientNet [7] comes into picture. It uses compound scaling methods which gives better performance than traditional scaling up methods.



EfficientNet Model Architecture

- The baseline model performed quite well actually, but it was a very simple model which we trained for our particular dataset from scratch .
- However, there are more advanced and pre-trained state-of-the-art models that we can try to use for our classification task.



```
Epoch 00027: LearningRateScheduler reducing learning rate to 6.254525801822128e-07.
856/856 [==============================] - 917s 1s/step - loss: 0.3740 - auc: 0.8909 - val_loss
: 0.3958 - val_auc: 0.8799
Epoch 28/30

Epoch 00028: LearningRateScheduler reducing learning rate to 4.678168061275489e-07.
856/856 [==============================] - 917s 1s/step - loss: 0.3784 - auc: 0.8893 - val_loss
: 0.3955 - val_auc: 0.8800
Epoch 29/30

Epoch 00029: LearningRateScheduler reducing learning rate to 3.5747176428928423e-07.
856/856 [==============================] - 916s 1s/step - loss: 0.3688 - auc: 0.8952 - val_loss
: 0.3956 - val_auc: 0.8800
Restoring model weights from the end of the best epoch.
Epoch 00029: early stopping
```

- EfficientNet is one such model architecture that has been researched extensively recently, and has achieved state-of-the-art level accuracy as compared to other models on ImageNet data with significantly fewer number of parameters, which means faster training times.
- As we have a large dataset, we can use these models, with and without pretrained weights to see if we get better results than our baseline.
- The model converges after the 29th epoch where average time for each epoch was nearly 20 minutes with loss of 0.3688 which is in acceptable range, further training would have led to overfitting.
- The validation accuracy which we achieved after training this model was 0.8800

3.4 Evaluation

- We compiled both the models to keep track of ROC AUC
- The focus was on looking out for a good AUC value, which tells us that the model is good at separating the two classes well.
- We compared the two models later and also saw what kaggle submission scores we get from our predictions for the test dataset.
- The most basic direct comparison between the two models, our simple CNN and the EfficientNet, is summarized in the following table.

Model	Train AUC	Val AUC	Test AUC(Kaggle)	Avg time (per epoch)
Simple CNN	0.8363	0.8388	0.8435	3330s | 55 mins
EfficientNet B7	0.8952	0.8800	0.8754	860s | 15 mins

- Since the predictions made by the model are predicted values for the classes, we can look at the predicted values to judge how well our model did classifying those, specifically how confidently the model predicted those targets.
- Closer the predicted probabilities of the target are to 0 and 1, we can say more confident the model output is.

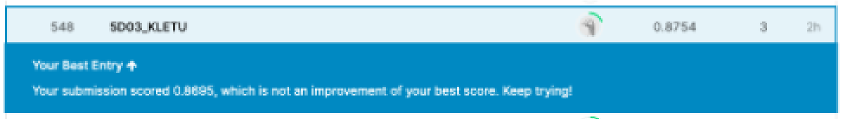
Probability(Presence of GW Signal)	Simple CNN	EfficientNet B7
20% to 80%	33%	61%
10% to 90%	22%	31%

- It was observed , out of the 226000 total test predictions, we can say that 74524, or ~33% of the values were predicted by the CNN model with high confidence (>80% probability) for either class
- 48979 or ~22% were predicted with more than 90% probability.
- In EfficientNet Model, out of the 226000 total test predictions, we can say that 74524, or ~61% of the values were predicted by the EfficientNet model with high confidence (>80% probability) for either class
- 70680 or ~31% were predicted with more than 90% probability.

4. Results & Conclusions

- Gravitational Waves are NOT EASY to detect! Once detected, they are hard to find.
- After sifting through a variety of preprocessing steps, we transformed the original strain wave data into frequency spectrograms, which are images that we then used to train deep learning models.
- One of the biggest challenges in this project was managing such a large dataset, which was solved by using the TensorFlow's tf.data API, and streamlining the entire workflow all the way from data import to model training & prediction tasks. This helped us achieve the goal of this project of building a pipeline that is flexible and can be reused in the future.
- Our simple CNN architecture, just after 3 epochs, was performing more than expected.
- The Efficient Net B7 model worked quite well with AUC score of 0.8754
- We evaluated the models for ROC AUC score, as we wanted our model to be good at separating the two classes, but also tracked accuracy scores for comparison. Overall, we achieved an AUC score of 0.8754 on the test dataset from kaggle.

Model	Kaggle AUC Score	Rank (On submission)	Rank (Current)
Simple CNN	0.8435	999	--
EfficientNet B7	0.8754	457	548



548 5DO3_KLETU 0.8754 3 2h

Your Best Entry

Your submission scored 0.8695, which is not an improvement of your best score. Keep trying!

5. Future Scope

- The modelling part of this project did not include any form of regularization.
- The signals were stacked side-by-side for this project, to form a single string of waves, and then transformed into a spectrogram which is unique per observation, but treating each signal from the respective observatory as different features could be interesting to look at.
- Explore further the reasons why attempts to use the "raw" signal did not give acceptable results when compared to the other method used.
- Computing efficiency was considered but there is room for further improvement.

References

[1].G2Net Gravitational Wave Detection Challenge[https://www.kaggle.com/c/g2net-gravitational-wave-detection]

[2].G2Net EDA + Preprocessing + Model Notebook[https://www.kaggle.com/avantikashrivastava/g2net-eda-preprocessing-model]

[3].Enhancing gravitational-wave science with machine learning[https://iopscience.iop.org/article/10.1088/2632-2153/abb93a/pdf]

[4]. Improving significance of binary black hole mergers in Advanced LIGO data using deep learning[https://arxiv.org/abs/2010.08584]

[5]. (GW Tutorials) [https://www.gw-openscience.org/LVT151012data/LOSC_Event_tutorial_LVT151012.html#Intro-to-signal-processing]

[6]. tf.data: Build TensorFlow input pipelines [https://www.tensorflow.org/guide/data]

[7]. Image classification via fine-tuning with EfficientNet[https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/]