DETECTION OF GRAVITATIONAL WAVES USING NEURAL NETWORKS

Shikha Bangar
Physics 594: Final Project
July 16th 2023

ABSTRACT

The gravitational waves (GW) signals are a different way to look at the Universe as they can help study massive objects that create ripples in space-time. These ripples are so small that identifying them has been a challenge. Isolating the signal from the noise is a complex task, even with the advancements made in sensitive detectors. It requires great skill and expertise to decipher the data and extract meaningful information accurately. In this project, we have used different neural network models to identify GW signals from mergers of binary black holes. Even after using complicated models, it took much work to identify the GW signals accurately. Due to the large amount of data with noise, this was a challenging task, and we could only achieve 0.76 accuracy. With the help of better computational resources and advanced techniques like transfer learning, better accuracy can be achieved.

1 Introduction

In 1916, Albert Einstein gave a remarkable theory on gravity - the general theory of relativity. According to this theory, gravity results from the curvature of spacetime caused by massive objects. The curvature (or gravitational pull in Newtonian Physics) is proportional to the object's mass. Also, when these massive objects move, they can create ripples in the fabric of spacetime, leading to Gravitational Waves (GW). An observer will feel distorted spacetime due to the propagation of the gravitational wave. The detection of these ripples is a challenging task due to their minuscule distortion of spacetime. Hence, there is a higher chance of detecting the GW from the black hole collisions, supernovae, and neutron star collisions as they are massive and will generate the most robust waves.

The GW is detected using an interferometer - studying the interference pattern for two or more light sources. As shown in fig. 1, the two light sources are placed perpendicular, and the interference pattern changes if the GW passes through them. Because GW will create only minor spacetime distortion, these interferometers must be very sensitive, making them very noisy. A bird landing on the arm of the interferometer or passing a heavy truck nearby can lead to a signal. One solution for such issues is placing multiple interferometers with long arms in isolated areas. One such initiative was taken in the form of LIGO, Laser Interferometer Gravitational-wave Observatory. It comprises two enormous laser interferometers located 3000 kilometers apart with around 4 km long arms.

In 2015, LIGO detectors - Livingston, Louisiana, and Hanford, Washington made the first direct observation of gravitational waves [2]. They detected the merging of binary black holes, a massive incident in astronomical

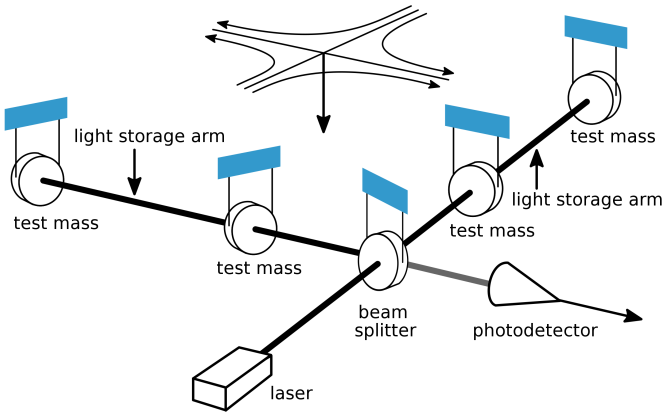


Figure 1: A schematic diagram of a laser interferometer [1]

events. This detection was marked as a scientific milestone not only because Einstein's theory was confirmed after 100 years, but now we have another way of "looking" at the Universe. In 2019, the data was collected again with more observatory, and in just six months, various merging of binary black holes, neutron stars, and black hole-neutron star has been observed. At this rate, the researchers are excited to study more astronomical problems like the measurement of Hubble's constant, test general relativity, and new GW sources like supernovae. However, this much data with loads of noise comes with challenges. Processing and analysis of data needs to catch up with the rising number of LIGO interferometers.

Machine Learning (ML) techniques significantly tackle the issues - large amounts of data with noise. In this paper [3], researchers have summarized many ML-based techniques that LIGO and Virgo's scientists have developed. The techniques range from detection and classification of noise, noise removal, and parameter estimation to search for GW signals. They also examine the potential of ML to improve GW science in general. Further work has been done for specific cases, such as binary black hole mergers. In this work [4], a time-domain model for gravitational waveforms was built using machine learning methods. With a similar goal in mind, in this work [5], the researchers presented a novel machine-learning strategy to search for binary black hole mergers in the data. They use transfer learning to classify the signals and glitches in the LIGO data.

In this project, we use the data from LIGO and Virgo to identify the gravitational signal from the noisy data. The goals of the project are to understand the GW time-simulated data from three different detectors, learn to manage massive data files, and use different machine learning techniques - linear regression, deep learning neural network, and convolution neural network to achieve decent accuracy.

The report is organized as follows. First, in section 2, we will discuss the data set and understand how to deal with time-series data from detectors. Then in section 3, the model is discussed, which was used to perform the classification. Next, the different hyperparameters with benchmarking methods in the sec **??**. This section is followed by the results obtained with different models. Finally, we discuss the results, limitations of the model, and future work that can be done using this project.

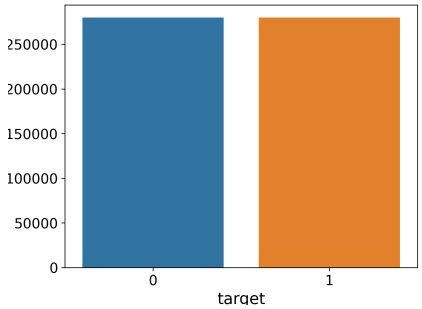
2 Dataset

The dataset for this project has been taken from Kaggle [6]. In 2021, Kaggle organized a "G2Net Gravitational Wave Detection" competition in collaboration with G2Net [7]. The competition aimed to build a model to

-

	id	target	path
0	00000e74ad	1	/content/g2net-gw/train/0/0/0/00000e74ad.npy
1	00001f4945	0	/content/g2net-gw/train/0/0/0/00001f4945.npy
2	0000661522	0	/content/g2net-gw/train/0/0/0/0000661522.npy
3	00007a006a	0	/content/g2net-gw/train/0/0/0/00007a006a.npy
4	0000a38978	1	/content/g2net-gw/train/0/0/0/0000a38978.npy

(a) Table of the data showing the id, target, and path. Target = 0 or 1 implies binary classification.



(b) Balanced Dataset and hence no need of under or oversampling of data.

Figure 2: Insights into the Dataset

analyze simulated GW time-series data from a network of Earth-based detectors. In this case, LIGO Hanford, LIGO Livingston, and Virgo.

The data contains a folder of the train which contains multiple folders with .npy files corresponding to a single data. Each data sample contains 3 time series (1 for each detector), and each spans for 2 sec and is sampled at 2048 Hz. There is another folder of tests with similar data. Moreover, there is a .csv file that contains labels associated with the gravitational wave.

The total size of the data is around 77GB which made it impossible to work on the local machine. We started working with the Kaggle notebooks but the entire process was very slow and hence we had to move to Google Colab. Although the colab had better RAM than Kaggle, it was not enough. We had to upgrade to Google Colab Pro to access more memory. We could not upgrade to GPU in colab because that reduces the RAM. Hence, all the models were trained using a CPU.

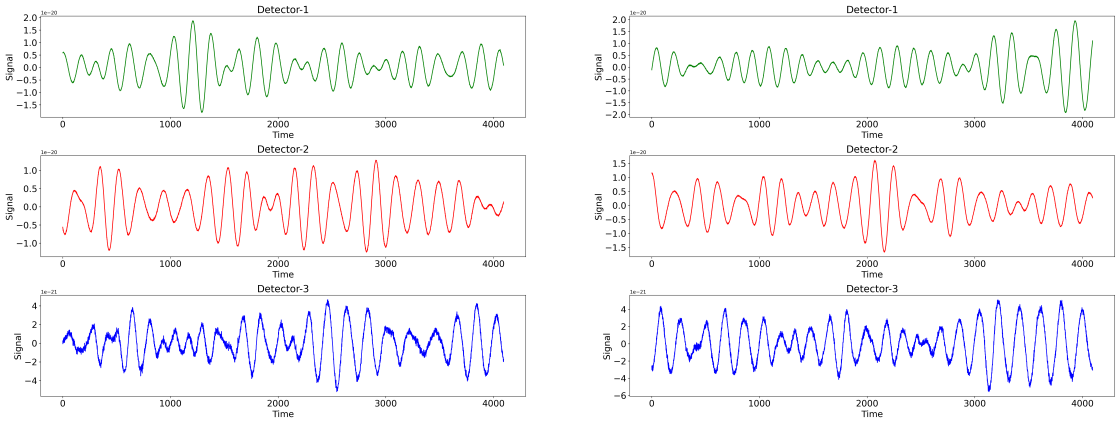
2.1 Data Visualization

We printed some of the data to get some insight. The table is shown in fig. 2 (a); the id corresponds to specific events, and the target is whether the event had a GW signal. The target=1 corresponds to the GW signal. The path corresponds to the path from where the data sample was picked. The total number of datasets is 560000. The next step is to know if the dataset is balanced or unbalanced. The fig. 2 (b) shows that the data is balanced.

We also plotted the data for the two cases - target = 0 and 1, shown in fig. 3. The first plot contains the time distribution of signals which spans for 2 sec, sampled at 2048 Hz. The different colors refer to 3 detectors. Since it was hard to understand the time distribution, we plotted probability distributions (fig. 4). Based on the plots, one can conclude that the GW signal is not visible. Hence we need an efficient machine-learning model to distinguish GW signals.

2.2 Data Augmentation

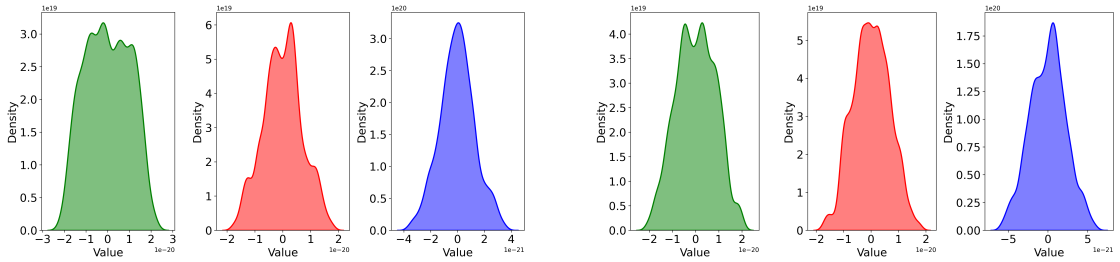
Using this data directly in a machine-learning model is challenging, so we used a spectrogram. A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. A way to reproduce the spectrogram is using Q-Transform, constant quality factor transform (CQT). It was first introduced in



(a) Target = 1, GW signal

(b) Target = 0, no GW signal

Figure 3: Comparison between the time distribution of signals from three detectors



(a) Target = 1, GW signal

(b) Target = 0, no GW signal

Figure 4: Comparison between the probability distribution of signals from three detectors

1988 in this work [8]. This is similar to the Fourier transform, but here, a data series is transformed to the frequency domain.

There are Python libraries that can perform the CQT. We used such libraries and plotted different images in fig .5. An interesting thing to notice here is, again, we cannot tell if there is GW present or not just by looking at it.

3 Model

After applying data augmentation, the data turned into images and hence the problem became a binary image recognition. The shape of the image is (56, 193, 1), which becomes the input for the neural network. One of the best ways to classify images is using Convolutional Neural Network (CNN). The CNN model used for this problem is shown in fig. 6.

We used two layers of 2D CNN, each with 64 and 32 filters. These layers use the activation Rectified Linear Unit (*relu*) and have kernel size of 3×3. Every 2D CNN is followed by the max pooling layer for downsampling the input along its spatial dimensions. It has a pool size of 2×2. Then there is flatten layer, which converts the 2D images into a 1D vector for the dense layer. This layer has only one node, which makes

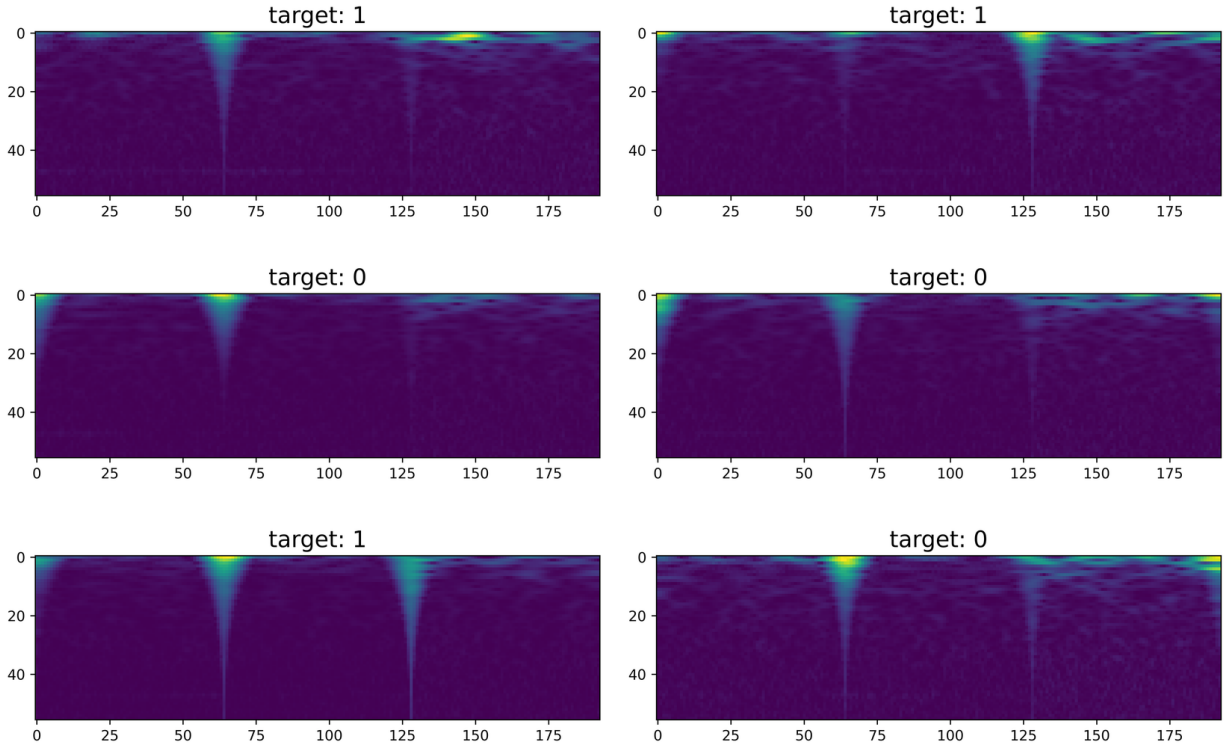


Figure 5: Spectograms of different targets plotted after randomly selecting 6 datasets.

it suitable for binary classification. It uses the sigmoid activation function, which output values between 0 and 1, perfect for binary classification probabilities. By constructing the model in such a way, we had almost 37,000 trainable parameters.

The model is compiled using the Adam optimizer. We use the binary cross-entropy as we have a binary classification problem. Finally, we measure 'accuracy' as the performance metric. The model was trained for 10 epochs, which might seem less, but that was the only option due to the lack of computational resources.

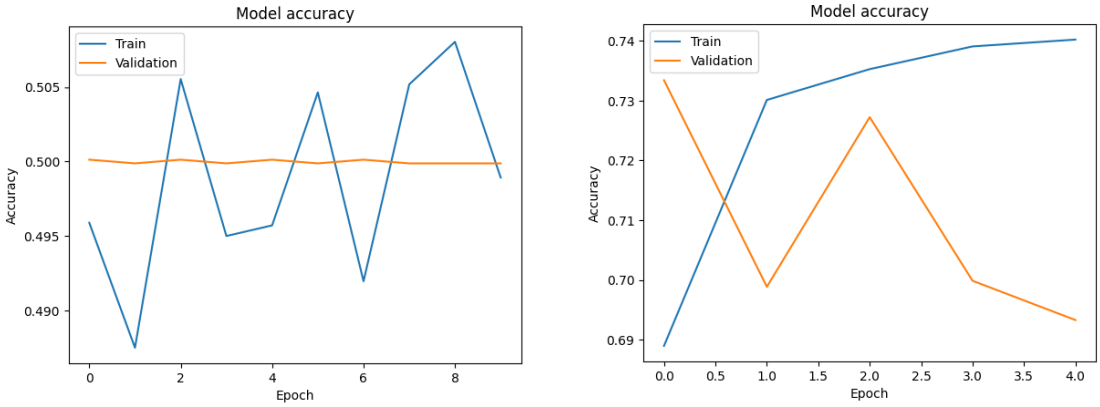
4 Benchmarking and Testing

We started with a simple Deep Neural Network (DNN) to get some insight into the complexity of data. This made us realize that we need a more complex DNN to make this work. So, we used a deep model with 6 layers and various nodes. Even that complicated DNN did not work. The best training accuracy achieved was 0.51, and the validation accuracy never changed from 0.5. The plotting of their accuracy also made us realize that the model was not training properly (fig. 7(a)).

The next step is choosing a Convolutional Neural Network (CNN) as it works best for image recognition. After looking at some of the solutions of participants of the Kaggle competition we realized that more complicated CNN models give good results when they are run for small epochs. So, we started with a complicated CNN model (fig. 8). We used the dropout layers and early stopping to avoid overfitting. After 5 epochs, training accuracy was 0.74 and validation accuracy was 0.6933. When we plotted these accuracy it was clear that

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)            (None, 54, 191, 64)       640

 max_pooling2d_2 (MaxPooling  (None, 27, 95, 64)        0
 2D)

 conv2d_3 (Conv2D)            (None, 25, 93, 32)        18464

 max_pooling2d_3 (MaxPooling  (None, 12, 46, 32)        0
 2D)

 flatten_1 (Flatten)          (None, 17664)             0

 dense_1 (Dense)              (None, 1)                 17665

=================================================================
Total params: 36,769
Trainable params: 36,769
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: The summary of the CNN model used for binary classification - detection of GW wave.



(a) DNN model

(b) CNN model

Figure 7: Results obtained in the process of optimization of the model, the training and validation accuracies are plotted at different epoch values for the case of DNN and CNN models.The results were not satisfactory and hence we needed a better model

```
Model: "CNN_model2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv_01 (Conv2D)            (None, 54, 191, 32)       320

 Pool_01 (MaxPooling2D)      (None, 27, 95, 32)        0

 Dropout_01 (Dropout)        (None, 27, 95, 32)        0

 Conv_02 (Conv2D)            (None, 25, 93, 32)        9248

 Pool_02 (MaxPooling2D)      (None, 12, 46, 32)        0

 Dropout_02 (Dropout)        (None, 12, 46, 32)        0

 Conv_03 (Conv2D)            (None, 10, 44, 32)        9248

 Pool_03 (MaxPooling2D)      (None, 5, 22, 32)         0

 Flatten (Flatten)           (None, 3520)              0

 Dense_01 (Dense)            (None, 64)                225344

 Dense_02 (Dense)            (None, 1)                 65

 Output (Dense)              (None, 1)                 2

=================================================================
Total params: 244,227
Trainable params: 244,227
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: The summary of a complicated CNN model used in the optimization process. This model did not produce good results and it corresponds to model - 4 in table 1

we were overfitting (fig. 7(b)). The same process of repeated many times, which also includes playing with the hyperparameters - changing the learning rate. But it was hard for the run to complete (due to RAM issues), and even when some completed the run, we did not get good results. Ultimately, we decided to try a simple CNN model, which gave around 0.76 accuracy. This was an improvement on the DNNs and all other complicated CNNs. The main features of the model used in this process are summarized in the table 1, and their accuracies are plotted in the fig. 9. For comparison, we have also included one of the good results from the participant of the Kaggle competition, where they used the Efficient Net B7 model.

Model No.	**Type**	**No. of layers**	**Epochs**	**Learning Rate**
1	DNN (simple)	4	5	0.0001
2	DNN (complicated)	4	10	0.01
3	CNN (simple)	2+1	10	0.01
4	CNN (complicated)	3+2	5	0.0001
5	EffNet	6	2	0.01

Table 1: Summary of all the different models tried in the optimization process. The last entry in the table corresponds to the results of one of Kaggle's competition participants (added for comparison). The accuracies of these models are plotted in the fig. 9

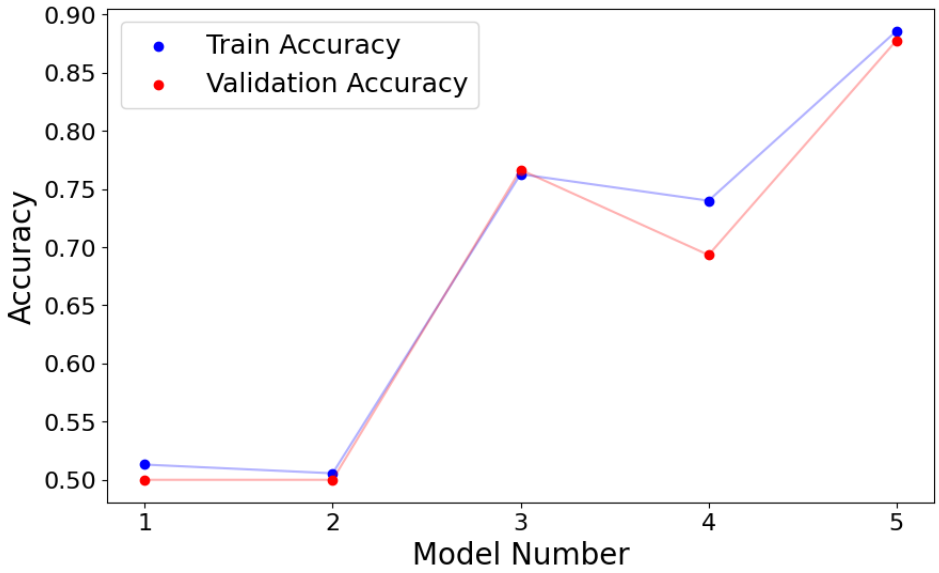


Figure 9: Training and validation accuracy plot for different models (summarized in table 1)

5 Results

The result achieved after training the simple CNN model had a training accuracy of 0.76 and a validation accuracy of 0.7649. When we plotted these accuracies it looked like they were about to converge (fig. 10). We understand the results don't look ideal but they were the best out of all the four models tried in this project.

6 Discussion

The validation accuracy achieved in this project could be better. Even a complicated CNN model with multiple layers and techniques to avoid overfitting, like early stopping, and regularization, could not achieve reasonable accuracy. The main challenge was the enormous amount of data intertwined, corresponding to time series from three detectors. Another factor that made this task challenging was noise. The integrated signal-to-noise ratio exceeds 8; this measures how detectable a signal is.

The data was taken from a Kaggle competition, and on average, participants achieved 0.8 accuracies, with the winner achieving 0.87. They used a different type of model - EfficientNet to get good results. This is a pre-trained model mainly used for images recognitions. It uses the concept of transfer learning which was also discussed in this work [5]. It is available in Keras in seven forms - EfficientNet B0 to B7 (with increasing complexity) [9]. We could not use this remarkable model because of a lack of computational resources. The Google colab kept crashing even with the simplest model - EfficientNet B0. Hence, we only worked with CNN models.

After conducting some experiments, we concluded that for the CNN model to perform well, it needs to be dense, and the learning rate should be extremely low. This allowed us to achieve decent results and improve the accuracy of our predictions. Another critical factor for getting good results is having good computational resources. Unlike the models we trained in class, this model training took around 3 hours for each epoch. This limits how many epochs we can run, as it seems unreasonable to go up to 100.

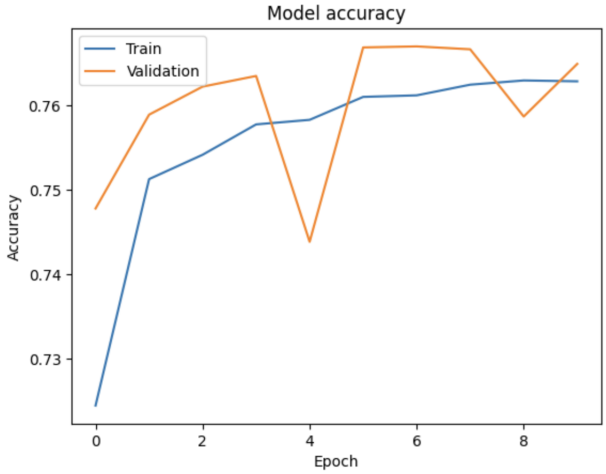


Figure 10: Training and validation accuracy plot for the simple CNN model. The training was performed for 10 epochs and we used Adam optimizer. The performance metric was accuracy, and binary cross entropy was used to solve the binary classification.

Although we faced many challenges and did not achieve the results we had hoped for, we could still gain much knowledge from this project. One of the most exciting things we learned about was how gravitational waves are detected. The detectors must be sensitive, which increases the noise, so finding a balance is important. The data was initially in the form of time series spanned over 2 seconds and sampled at 2048 Hz, and we used a spectrogram to convert this into images. This improved the learning process, as image recognition techniques are very advanced today. After visualizing the data, we also realized it is crucial to understand that one cannot identify a signal with the naked eye alone. That is why machine learning will play a significant role in detecting the signals accurately. This was very different from the MNIST data we studied in class. Another idea we learned from the project is that there is always a hierarchy of models that can perform better based on their features. After dealing with image recognition with the MNIST dataset, CNN works best for images, but that was not enough in this project. We would have to go to transfer learning to get good results.

There is much potential to take this project to the next level. The first thing that can be done is to use the transfer learning technique to improve the results. Another potential work can be using quantum machine learning. There is one-to-one correspondence in neural networks in classical and quantum machine learning. However, only a little work has been done in detecting GW signals using quantum machine learning. This project does have the prospect of becoming an entire research project.

In conclusion, the project was an exciting opportunity to learn about GW and how vital machine learning can be in distinguishing GW signals from noise. Also, CNN models are the best approach for image recognition but do not always perform well. If the data is enormous and contains much noise, advanced techniques like transfer learning are required for good results.

References

[1] L. Caltech, MIT. [Online]. Available: https://www.ligo.caltech.edu/page/what-is-interferometer

[2] B. P. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari *et al.*, "Observation of gravitational waves from a binary

black hole merger," *Physical review letters*, vol. 116, no. 6, p. 061102, 2016. [Online]. Available: https://journals.aps.org/prd/abstract/10.1103/PhysRevD.104.064051

[3] E. Cuoco, J. Powell, M. Cavaglià, K. Ackley, M. Bejger, C. Chatterjee, M. Coughlin, S. Coughlin, P. Easter, R. Essick *et al.*, "Enhancing gravitational-wave science with machine learning," *Machine Learning: Science and Technology*, vol. 2, no. 1, p. 011002, 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/2632-2153/abb93a/meta

[4] S. Schmidt, M. Breschi, R. Gamba, G. Pagano, P. Rettegno, G. Riemenschneider, S. Bernuzzi, A. Nagar, and W. Del Pozzo, "Machine learning gravitational waves from binary black hole mergers," *Physical Review D*, vol. 103, no. 4, p. 043020, 2021. [Online]. Available: https://journals.aps.org/prd/abstract/10.1103/PhysRevD.103.043020

[5] S. Jadhav, N. Mukund, B. Gadre, S. Mitra, and S. Abraham, "Improving significance of binary black hole mergers in advanced ligo data using deep learning: Confirmation of gw151216," *Physical Review D*, vol. 104, no. 6, p. 064051, 2021. [Online]. Available: https://journals.aps.org/prd/abstract/10.1103/PhysRevD.104.064051

[6] E. C. i. M. J. W. Chris Messenger, ChristopherZerafa, "G2net gravitational wave detection," 2021. [Online]. Available: https://kaggle.com/competitions/g2net-gravitational-wave-detection

[7] G2NET, "A network for gravitational waves, geophysics and machine learning." [Online]. Available: https://www.g2net.eu/

[8] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[9] Kaggle. [Online]. Available: https://keras.io/api/applications/efficientnet/