**G2Net Detecting Continuous Gravitational Waves**

**CMPE - 257**

**Submitted By -**

**Name Student Id**

**Ketan Malempati 016695354**

**Pruthvi Raj Kotigari 016681522**

**Ashish Mernedi 016662360**

**Western Han 012261964**

**Sumanth Mungi 016648684**

## **TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
|  | **INTRODUCTION** | **01** |
|  | **RELATED WORK** | **02** |
|  | **DATA USED** | **03** |
|  | **PROPOSED METHODOLOGY** | **04** |
|  | **PRELIMINARY PROCESSING** | **05** |
|  | **CNN** | **05** |
|  | **LSTM** | **06** |
|  | **LGBM** | **07** |
|  | **ENSEMBLING MODELS** | **09** |
|  | **OPTIMIZATION** | **10** |
|  | **RESULTS AND DISCUSSION** | **10** |
|  | **LESSONS LEARNT** | **10** |
| **13.** | **FUTURE RESEARCH WORK AND LIMITATIONS** | **10** |
| **14.** | **ACKNOWLEDGMENT** | **10** |
| **15.** | **REFERENCES** | **11** |
|  |  |  |

## **LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| **Fig 3.1** | **Description of the data** | **03** |
| **Fig 3.2** | **Representation of dataset point (target=1)** | **03** |
| **Fig 3.3** | **Representation of dataset point (target=0)** | **03** |
| **Fig 3.4** | **Spectrogram of dataset[10]** | **03** |
| **Fig 6.1** | **CNN Architecture** | **05** |
| **Fig 6.2** | **CNN model building** | **06** |
| **Fig 7.1** | **LSTM model with forget gate** | **06** |
| **Fig 7.2** | **LSTM model** | **07** |
| **Fig 7.3** | **Loss vs Epochs for LSTM** | **07** |
| **Fig 8.1** | **Comparison between XGBoost and LGBM** | **07** |
| **Fig 8.2** | **Leaf-wise growth of LGBM** | **08** |
| **Fig 8.3** | **ENN for undersampling** | **08** |
| **Fig 8.4** | **SMOTE for Oversampling** | **08** |
| **Fig 8.5** | **Applying SMOTE** | **09** |

G2Net Detecting Continuous Gravitational Waves

Ketan Malempati  
*Master’s in Artificial Intelligence  
 San Jose State University* 016695354

Western Han   
*Master’s in Software Engineering  
 San Jose State University*   
 012261964

Sumanth Mungi  
*Master’s in Artificial Intelligence  
 San Jose State University*  
 016648684

Pruthvi Raj Kotigari  
*Master’s in Artificial Intelligence  
 San Jose State University*  
 016681522

Ashish Mernedi  
 *Master’s in Engineering  
 San Jose State University*  
 016662360

***Abstract*— The goal of this project is to find continuous gravitational-wave signals. We will develop a model sensitive enough to detect weak yet long-lasting signals emitted by rapidly-spinning neutron stars within noisy data. Our current approach to this dataset is to clean data, do exploratory data analysis, data-preprocessing, train machine learning models, perform hyperparameter optimisation, and evaluation and compare different results of machine learning models. The approach to solving this problem is subject to change since the topic of gravitational waves is vast, and the more research we do could grant us new ways to tackle our goal. We will use multiple models like CNN, LSTM and many more and compare them. We are going to do optimizations to these models like hyperparameter optimization and make comparisons with the same model but with different hyperparameters.**

***Keywords — CNN, LSTM, Ensemble, Gravitational waves, LGBM, Classification, SFT.***

# Introduction

Gravitational waves are ripples in space which can be associated with events such as black holes and exploding stars when cataclysmic waves are detected. Gravitational waves travel at the speed of light (186,000 miles per second). These waves squeeze and stretch anything in their path as they pass by. A gravitational wave is an invisible (yet incredibly fast) ripple in space. They have a frequency of 0.5 Hz, and a wavelength of about 600 000 km, or 47 times the diameter of the Earth. Humans can’t see these gravitational waves, but with the help of antennas such as LIGO (Laser Interferometer Gravitational-Wave), it can detect vibrations in the medium of space. The medium of space is something scientists are still researching to this day and could play an essential role in telling us more about what is in the universe.

The current approaches include techniques for improving the sensitivity of Advanced Laser Interferometer GW Observatory and Advanced Virgo GW searches, methods for fast measurements of the astrophysical parameters of GW sources, and algorithms for the reduction and characterization of non-astrophysical detector noise. These applications demonstrate how machine learning techniques may be harnessed to enhance the science that is possible with current and future GW detectors.

# Related Work

# **Signal estimation from modified short-time Fourier transform:**

An approach to estimate a signal from its modified short-time Fourier transform is presented in this study (STFT). This computationally straightforward approach is created by reducing the mean squared error between the estimated signal's STFT and the adjusted STFT. They create an iterative technique to estimate a signal from its adjusted STFT magnitude using this algorithm. The mean squared error between the estimated signal's STFT magnitude and the adjusted STFT magnitude is demonstrated to decrease with each iteration of the iterative approach. The discrete Fourier transform (DFT) computation is a key step in the iterative technique, and current hardware looks to allow for real-time implementation of the algorithm. Speech time-scale adjustment has been done using the technique proposed in this research. The resulting system appears to perform better than any current approach and produces speech that is of extremely high quality.

**Motor Fault Diagnosis Based on Short-time Fourier Transform and Convolutional Neural Network:**

The experimental results showed that the influence of the preprocessing method is small, and that the batch-size is the main factor affecting accuracy and training efficiency. By investigating feature visualization, it was shown that, in the case of big data, the extracted CNN features can represent complex mapping relationships between signal and health status, and can also overcome the prior knowledge and engineering experience requirement for feature extraction, which is used by traditional diagnosis methods. This paper proposes a new method, based on STFT and CNN, which can complete motor fault diagnosis tasks more intelligently and accurately. The experimental findings demonstrated that the preprocessing method has a limited impact and that batch size is the primary determinant of accuracy and training effectiveness. By looking into feature visualization, it was discovered that, in the case of big data, the extracted CNN features could represent intricate mapping relationships between signal and health status and could also get around the requirement for prior knowledge and engineering experience for feature extraction, which is used by conventional diagnosis methods. This research suggests a novel way for more intelligent and precise motor defect detection tasks, based on STFT and CNN.

**How effective is machine learning to detect long transient gravitational waves from neutron stars in a real search?**

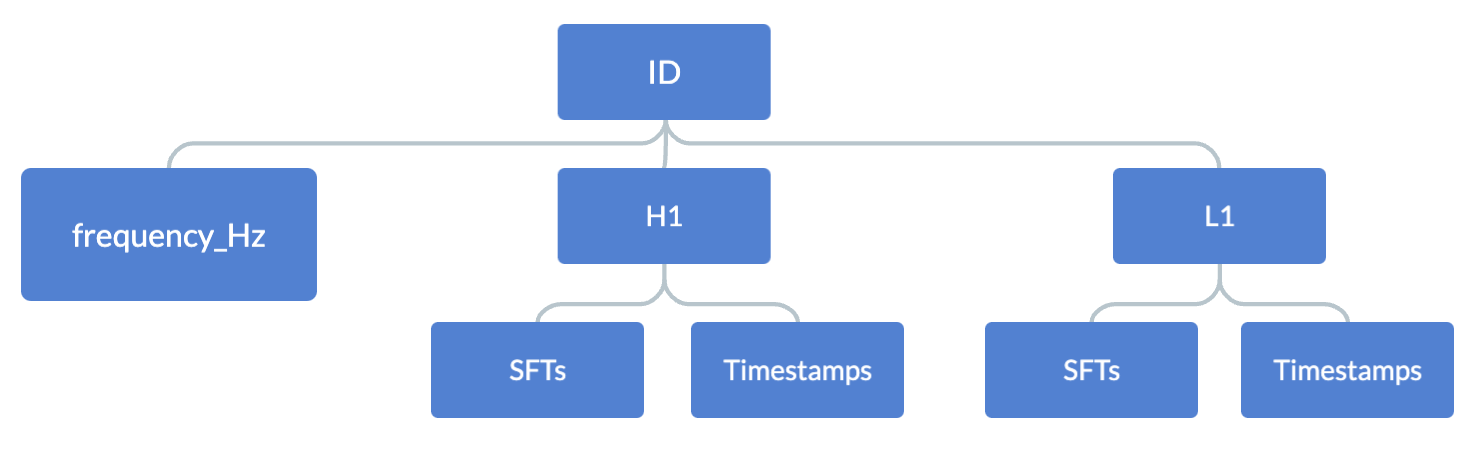
They provide a thorough analysis of convolution neural networks (CNNs') ability to recognize solitary neutron star long-duration transient gravitational-wave signals that last O(hours-days). They find that CNNs can handle signal morphologies that diverge from the training set and that a small amount of training data or injections is sufficient to ensure acceptable detection accuracy and a low false alarm probability. The truth is that they only need to train one CNN on signal/noise maps in a single 150 Hz band; after that, the CNN can discriminate signals/noise well in any band, but with varying efficiency and false alarm probabilities due to the nonstationary noise in LIGO/Virgo. Additionally, they show how we may limit the likelihood of false alarms for CNNs by choosing the best threshold for the CNN's outputs, which seems to rely on frequency. The Generalized FrequencyHough (GFH) is a well-known algorithm that maps curves in the time/frequency plane to lines in a plane that correspond to the initial frequency/spin-down of the source. Finally, we compare the detection effectiveness of the networks to that of the GFH. The networks are orders of magnitude faster to operate and can identify signals that the GFH is blind to, while having identical sensitivities to it. They suggest methods to apply CNNs to a real search utilizing LIGO/Virgo data in order to get over challenges we might face, including a limited supply of training data, by exploiting the findings of our investigation. Then, employing our networks and techniques, conduct a genuine search for a piece of GW170817. This is the first time a machine learning approach has been used to look for a gravitational-wave signal from a lone neutron star.

**Speaker Verification Model Using Short-Time Fourier Transform and Recurrent Neural Network:**

The need to correctly verify speakers is growing. As a result, a model for several speaker verification techniques has been proposed. In this study, we suggest a novel short-time Fourier transform-based technique for speaker verification (STFT). In contrast to the currently employed Mel-Frequency Cepstrum Coefficients (MFCC) extraction approach, we used a window function with an approximate 66.1% overlap parameter. In this situation, a deep running model called RNN (Recurrent Neural Network) containing LSTM cells is used to study the speech features of the speaker with the temporal characteristics. The accuracy of the suggested model is around 92.8%, which is 5.5% more accurate than the current speaker certification methodology.

# Data Used

Data was given in three files: Hierarchical Data Format (HDF5) train/test files, train\_labels CSV and submission\_sample CSV. The HDF5 files include data pertaining to time-frequency collected by LIGO Livingston Detector (L1) and LIGO Hanford Detector (H1), two gravitational-wave interferometers whose purpose is to detect gravitational waves. In addition to the time-frequency, each data sample includes Short-Time Fourier Transforms (SFTs) and GPS time which provides us with information on time-localized frequency. The CSV file contains features of the id of the dataset, and target, which is depicted as 1 for signal and 0 for no signal. When we map all of the features together we get a hierarchy of features (Figure 3.1). The ID feature is at the top of the hierarchy in the HDF5 file, as it links the data & data points to its label in the CSV. Frequency (Hz) is a subset that contains the range of frequency measured by the two gravitational interferometer detectors. H1 holds the data for LIGO Hanford Detector, consisting of SFTs frequencies and timestamps for measurement. L1 holds the data for LIGO Livingston Detector, consisting of SFTs frequencies and timestamps for measurement.

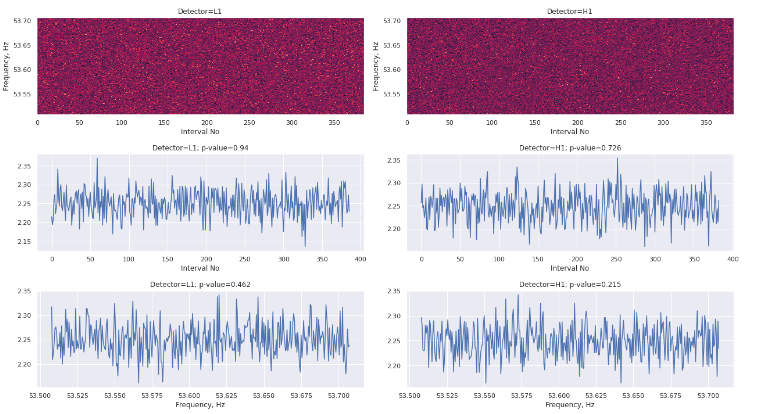


*Fig 3.1 Description of the data*

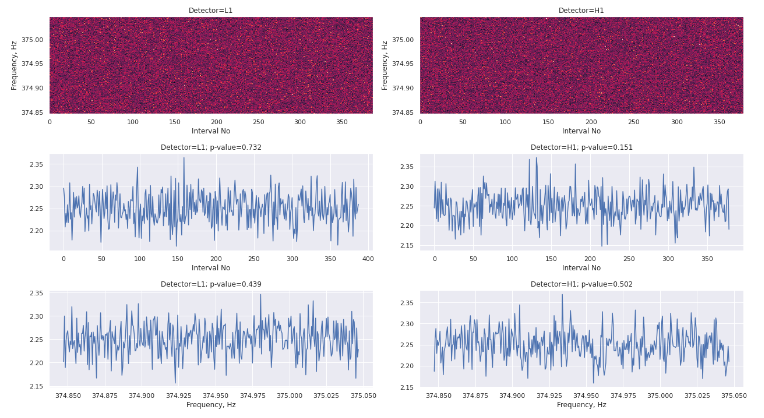
# Data Analysis

Upon observing the data, when we work with waves and signals, noise is susceptible in the data. Noise, itself, is a frequency and a signal in that frequency is denoted by its strength in the frequency field.

Id’s and target’s in the dataset have a correlation and connection to the entirety of the analysis. Target, although was depicted as 0, and 1 to determine whether there is a signal or not, has a value of 0.5 in the dataset as well. From the data, 0.5 more or less represents the fact that there is an essence of a signal in that dataset. Figure 3.2, is a good representation of what we see in the data plotting out the frequencies received from the two gravitational waves interferometer detector where the target = 1. However, when comparing Figure 3.2 (target =1) and Figure 3.3 (target = 0), it becomes difficult to differentiate between one with signal and the other without.

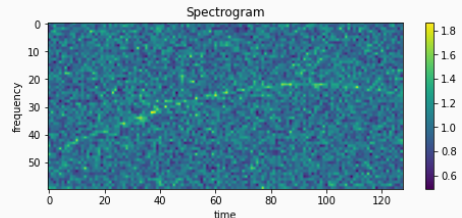


*Fig 3.2 Representation of dataset point (target=1)*

**

*Fig 3.3 Representation of a dataset point (target=0)*

Although comparing Figure 3.2 and 3.3 seem like static, if we zoom into the frequency we are able to see an essence of the signal (Figure 3.4). As you can see, the majority of the spectrum consists of noise; however, when zooming into a specific dataset we can observe this arch which signifies the signal. This specific plot is plotted against time and frequency in a suggested dimension.



*Fig 3.4 Spectrogram of dataset[10]*

# Proposed Methodology

# Preprocessing Data preprocessing is an essential step in Machine Learning because the quality of data and the useful information that can be extracted from it directly affects our model's ability to learn; thus, it is critical that we preprocess our data before feeding it into our model.

# CNN

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be. A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

# LSTM

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images) but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare. LSTM has become the most cited neural network of the 20th century.

# LGBM

Boosting algorithms are ensemble techniques where you design new models based on the errors observed on the previous models to do further analysis.

One such boosting algorithm is the Light Gradient Boosting Machine.

It is designed to be efficient with the following advantages:

* Faster training speed with higher accuracy
* Low memory usage
* Support of parallel, distributed and GPU learning

# Ensembling models (XGB, CATBOOST, LGBM, RANDOM FOREST)

In order to increase the precision of predictive analytics and data mining applications, ensemble modeling is the act of executing two or more related but different analytical models and then combining the results into a single score or spread..

A single model based on a single data sample can include biases, high variability, or plain mistakes that impair the validity of its analytical results in predictive modeling and other types of data analytics. Similar problems can result from using particular modeling methodologies. Data scientists and other data analysts can lessen the effects of those restrictions and give business decision makers greater information by combining different models or evaluating several samples.

A random forest model is a frequent illustration of ensemble modeling. Multiple decision trees, a sort of analytical model intended to forecast outcomes based on several factors and rules, are used in this data mining strategy. A random forest model combines numerous decision trees that may examine various sample data, assess various variables, or weigh universal variables differently. The outcomes of the different decision trees are then either averaged using a simple formula or combined using additional weighting.

Models that are being used:

XGB: Extreme Gradient Boosting, or XGBoost, is a concept that stems from Friedman's paper Greedy Function Approximation: A Gradient Boosting Machine. There are numerous resources on the issue of gradient boosted trees because they have been around for a while. Using the components of supervised learning, this lesson will explain boosted trees in a self-contained and principled manner. This justification, in our opinion, is clearer and more formal and serves to justify the model formulation utilized in XGBoost.

CATBOOST: CatBoost is a technique for decision trees that uses gradient boosting. It is created by Yandex researchers and engineers and is used for a variety of jobs at Yandex and in other businesses, such as CERN, Cloudflare, and Careem taxi, including search, recommendation systems, personal assistants, self-driving cars, weather prediction, and many more.

RANDOM FOREST: A large number of decision trees are built during the training phase of the random forests or random decision forests ensemble learning approach, which is used for classification, regression, and other tasks. The class that the majority of the trees chose is the output of the random forest for classification problems. The mean or average prediction of each individual tree is returned for regression tasks.

# Optimization Optimization is the process of iteratively training the model to produce a maximum and minimum function evaluation. It is one of the most important phenomena in getting better results in Machine Learning. In each iteration, we compare the results by changing the hyperparameters in each step until we achieve the best results. We develop an accurate model with a low error rate. There are numerous methods for optimizing a model. We use optimizers like Gradient Descent, Adam and Stochastic Gradient Descent Algorithms.

# Preliminary Processing

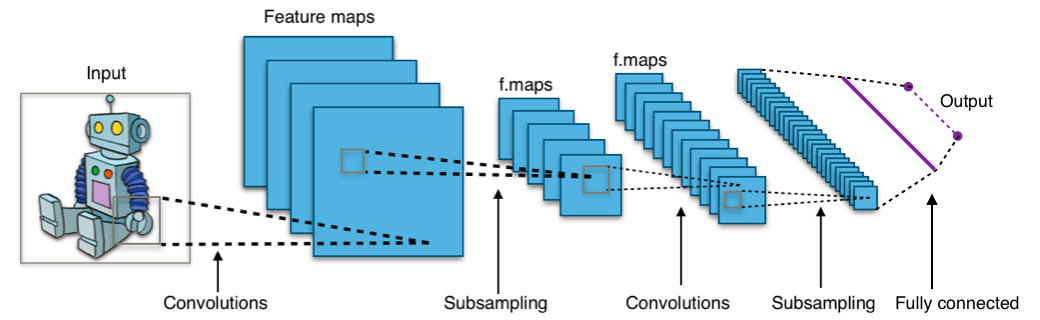
Pre-processing of the HDF5 files is essential in this project because the data in each file is huge and there are two different components in the data which are L1 and H1.

In this project we used different methods to pre-process the data. One is where we are converting all the data into an image of rgb values by mapping the SFT’s into RGB values ranging from 0 to 127. We are doing this because a change in the frequency means there would be a change in the RGB values associated to that time thus causing the color to be different, which allows us to see the gravitational waves in the generated images.

Another method that we are using in this project is where we combine all the data in the hdf5 file both from L1 and H1 and applying different mathematical functions on it like inverse fourier transforms, getting high impact values from both the features and bagging the data into different sets and taking average values from the sets as model input.

# CNN

An input layer, one or more hidden layers, and an output layer are the constituent parts of a convolutional neural network. The middle layers of any feed-forward neural network are referred to as hidden layers because the activation function and the final convolution obscure both the inputs and outputs of those levels. Within the hidden layers of a convolutional neural network are layers that are responsible for performing convolutions. In most cases, this will consist of a layer that carries out a dot product of the convolution kernel with the input matrix for the layer. In most cases, the Frobenius inner product is this product, and the activation function of this product is typically ReLU. The convolution procedure creates a feature map, which then contributes to the input of the subsequent layer as the convolution kernel slides along the input matrix for the layer. This process occurs when the convolution kernel moves from one layer to the next. After this comes subsequent layers, which may include pooling layers, completely connected layers, or normalizing layers, amongst others.



*Fig 6.1 CNN Architecture*

The learning process can be controlled using a variety of different variables known as hyperparameters. CNN's make use of a greater number of hyperparameters than standard multilayer perceptron (MLP).

The CNN model has many hyperparameters like

* Kernel size
* Padding
* Stride
* Number of filters
* Filter size
* Pooling type and size
* Dilation

The model we are using is tf\_efficientnet\_b7\_ns.

EfficientNet is a method and architecture for scaling convolutional neural networks that uses a compound coefficient to scale all dimensions of depth, width, and resolution in a convolutional neural network equally. In contrast to the common practice, which scales these elements freely, the EfficientNet scaling method scales network breadth, depth, and resolution uniformly using a set of predetermined scaling coefficients.

It is a generic model that implements a number of efficient models that use comparable DepthwiseSeparable and InvertedResidual blocks. The models that it implements are efficient because they use these blocks.

The hyper-parameters we are using are:

epochs = 16

batch\_size = 16

num\_workers = 2

weight\_decay = 1e-6

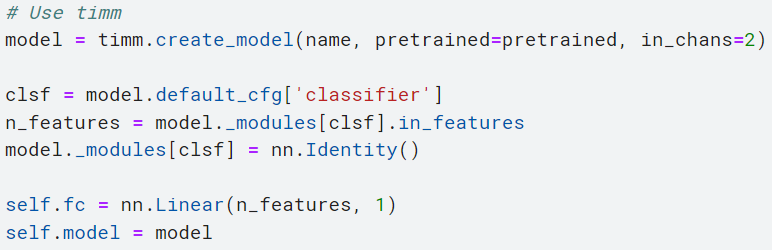
max\_grad\_norm = 1000

lr\_max = 4e-4

epochs\_warmup = 1.0

Optimiser - Adam

The below image shows creating the CNN model, first, we call the pre-trained model. Then we add a classifier to it as well as the features it needs to classify. Then finally we add a neural network to complete the model.



*Fig 6.2 CNN model building*

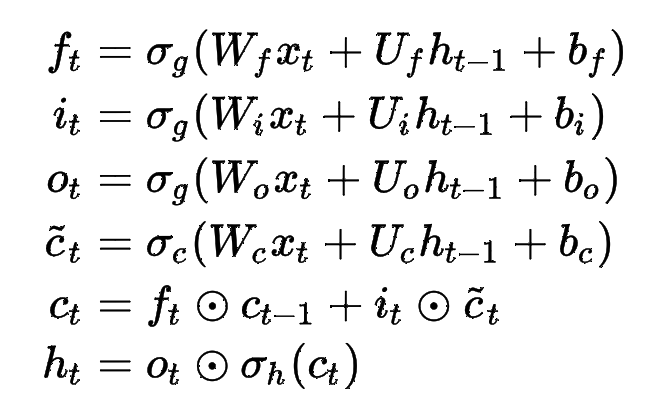
# LSTM

A standard RNN is analogized to have both "long-term memory" and "short-term memory" in the name of the LSTM. The activation patterns in the network change once per time-step, analogous to how physiological changes in synaptic strengths store short-term memories. The connection weights and biases in the network change once per episode of training, analogous to how physiological changes in synaptic strengths store long-term memories. The "extended short-term memory" of the LSTM architecture is intended to give RNN a short-term memory that can endure thousands of timesteps.

A cell, an input gate, an output gate, and a forget gate make up a typical LSTM unit. The three gates control the flow of information into and out of the cell, and the cell remembers values across arbitrary time intervals.

Since there may be lags of uncertain length between significant occurrences in a time series, LSTM networks are well-suited to categorizing, processing, and making predictions based on time series data. To solve the vanishing gradient issue that can arise when training conventional RNNs, LSTMs were created. The advantage of LSTM over RNNs, hidden Markov models, and other sequence learning techniques in many applications is their relative insensitivity to gap length.

The compact forms of the equations for the forward pass of an LSTM cell with a forget gate are:

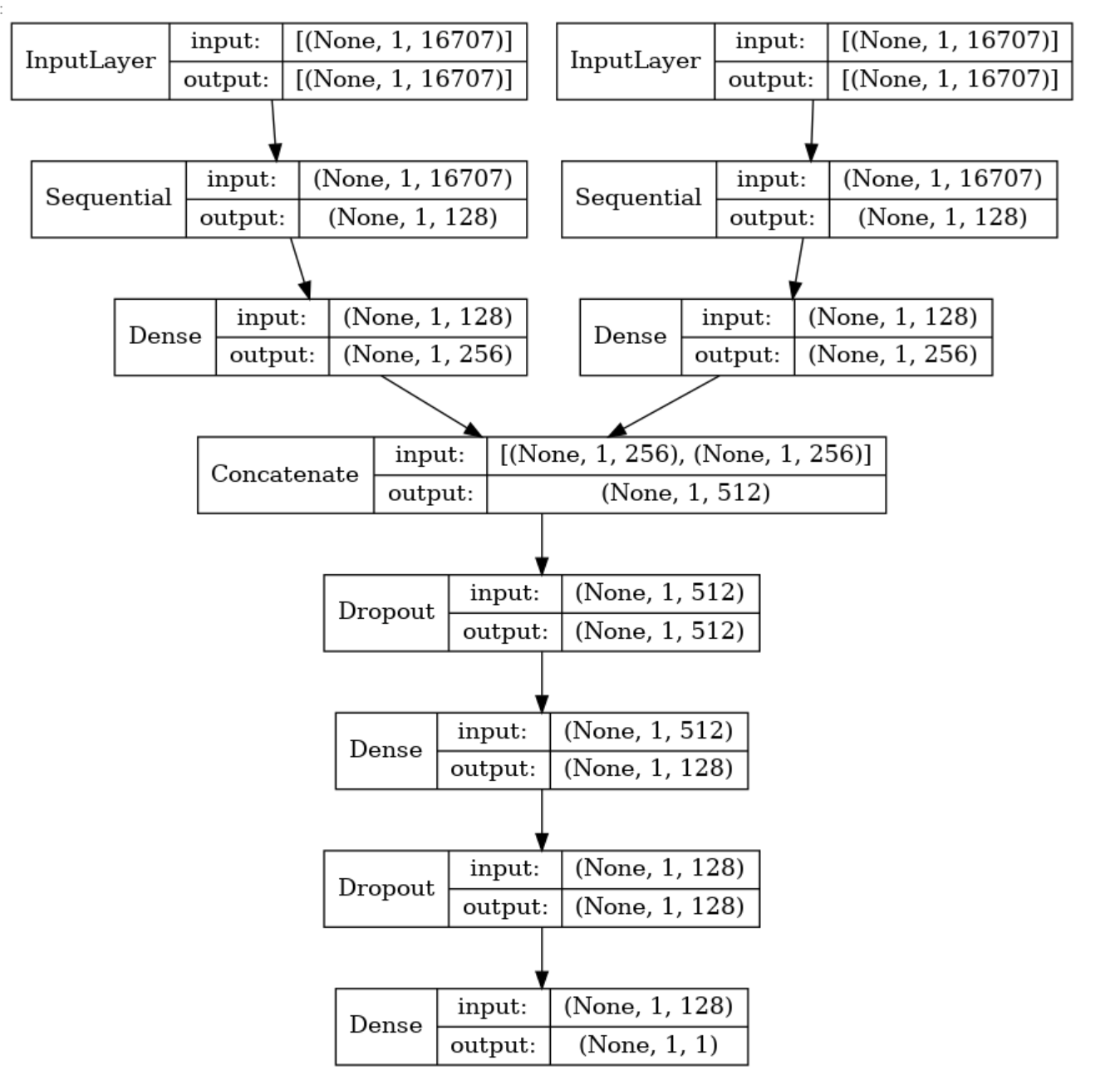


*Fig 7.1 LSTM model with forget gate*

Here is the image of how the model works.

We pass through 2 same LSTMs where the input for one is Hanford data and Livingston for the other.

Later, we concatenate the outputs of the LSTMs and pass them through a couple of dropout and dense layers. We again train them for a couple of epochs. The training flow is shown in *Fig 7.2*. The LSTM model goes as follows: we take the input data of size 16707 and map them to size 256 using sequential layers, later we scale them high to 256 with a dropout layer in the middle. We take the output from these layers and concatenate them as mentioned earlier and scale them to 128 and next to 1. Since the required size of the output of the model is 1 (a 0 or 1). Between each layer, we have a dropout layer with a probability of 0.2 to dropout. This helps us prevent the overfitting of the model. The current accuracy of the model is 50%.



*Fig 7.2 LSTM Model*

Hyper Parameters:

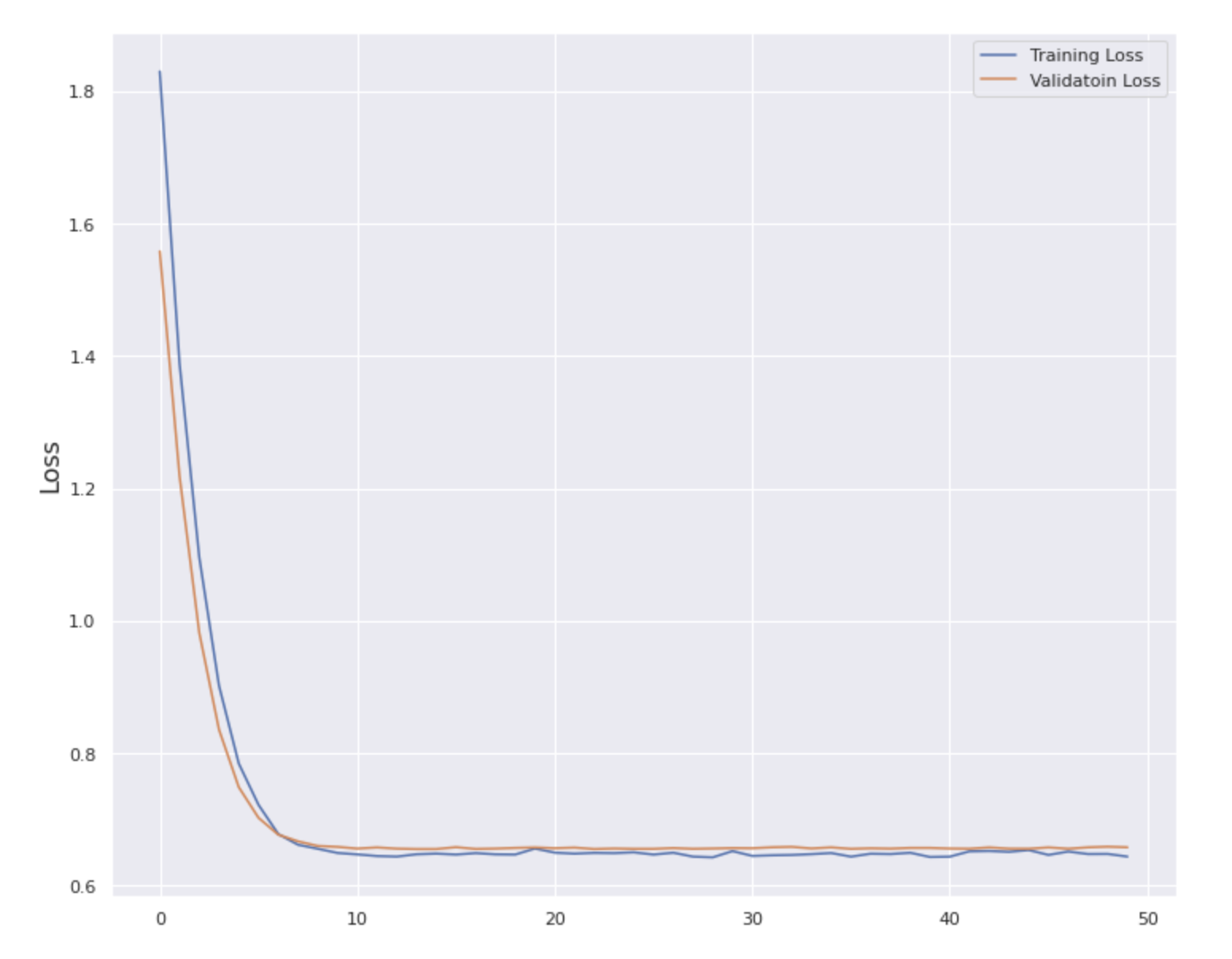
epochs = 50

Loss - Binary entropy loss

Validation split - 0.2

Optimiser - Adam

Metrics - AUC



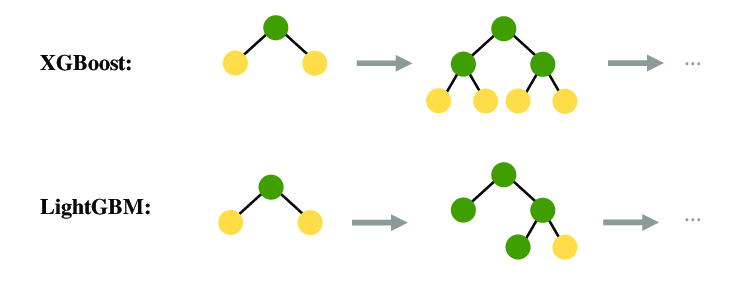
*Fig 7.3 Loss vs Epochs for LSTM*

# LGBM

Gradient boosting is a machine learning technique used in regression and classification tasks. It creates a prediction model as an ensemble of others- where you design new models based on the errors observed from the previous models and use those new models to do further analysis. XGBoost and LGBM are two widely used gradient boosting algorithms. While there are quite a few differences, the two work in a similar manner.

XGBoost is a type of gradient boosting model that uses tree-building techniques to predict its final value. This algorithm is designed to be highly efficient, flexible, and portable. The Light Gradient Boosting Machine Learning algorithm – also known as LGBM or LightGBM – is an open-source technique mainly used for machine learning tasks like classification and regression. It is quite similar to XGBoost as it too uses decision trees to classify data.

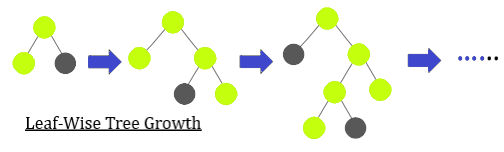
One of the main differences between these two algorithms, however, is that the LGBM tree grows leaf-wise, while the XGBoost algorithm tree grows depth-wise. LGBM is lightweight and requires fewer resources than its gradient booster counterpart, thus making it slightly faster and more efficient.



*Fig 8.1 Comparison between XGBoost and LGBM*

LGBM is designed to be distributed and efficient with the following advantages:

1. Faster training speed with higher accuracy
2. Low memory usage
3. Support of parallel, distributed and GPU Learning.

LGBM splits the tree leaf-wise as opposed to other boosting algorithms that grows tree level-wise. It chooses the leaf with maximum delta loss to grow. Since the leaf is fixed, the leaf-wise algorithm has lower loss compared to level-wise algorithm.

*Fig 8.2 Leaf-wise growth of LGBM*

LGBM does Smart feature engineering and smart sampling of data to build better models using following techniques:

1. Histogram or bundle way of splitting (smart optimization of splitting)
2. Exclusive feature bundling
3. Gradient based one side sampling - sampling data from analyzing their gradients. Low gradient samples require less training and high gradient samples require more training. LGBM combines both to give us more meaningful analysis.

Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) fulfills the limitations of histogram-based algorithm that is primarily used in all gradient boosting decision tree frameworks.

LGBM is almost 10x faster than XGBoost.

Problems Encountered:

The main problem encountered is imbalanced data. Imbalanced data is the data in which observed frequencies are very different across different possible values of categorical variables. Accuracy is a bad metric when working with imbalanced data.

Solution:

ENN (Edited Nearest Neighbor): Algorithm for finding ambiguous and noisy examples in a dataset. This rule involves using k=3 nearest neighbors to locate those examples in a dataset that are misclassified and that are then removed before a k=1 classification rule is applied.

* When used as an undersampling procedure, the rule can be applied to each example in the majority class, allowing those examples that are misclassified as belonging to the minority class to be removed, and those correctly classified to remain.
* It is also applied to each example in the minority class where those examples that are misclassified have their nearest neighbors from the majority class deleted.
* The Edited Nearest Neighbors rule can be implemented using the EditedNearestNeighbours imbalanced-learn class. The n\_neighbors argument controls the number of neighbors to use in the editing rule, which defaults to three.



*Fig 8.3 ENN for undersampling*

But ENN is used for undersampling of data. What about oversampling? Another technique comes handy for this- Synthetic Minority Oversampling Technique or simply SMOTE.

SMOTE (Synthetic Minority Oversampling Technique) is a machine learning technique that solves the problems that occur when using an imbalanced data set. SMOTE is an algorithm that performs data augmentation by creating synthetic data points based on the original data points.

The advantage of SMOTE is that you are not generating duplicates, but rather creating synthetic data points that are slightly different from the original data points.

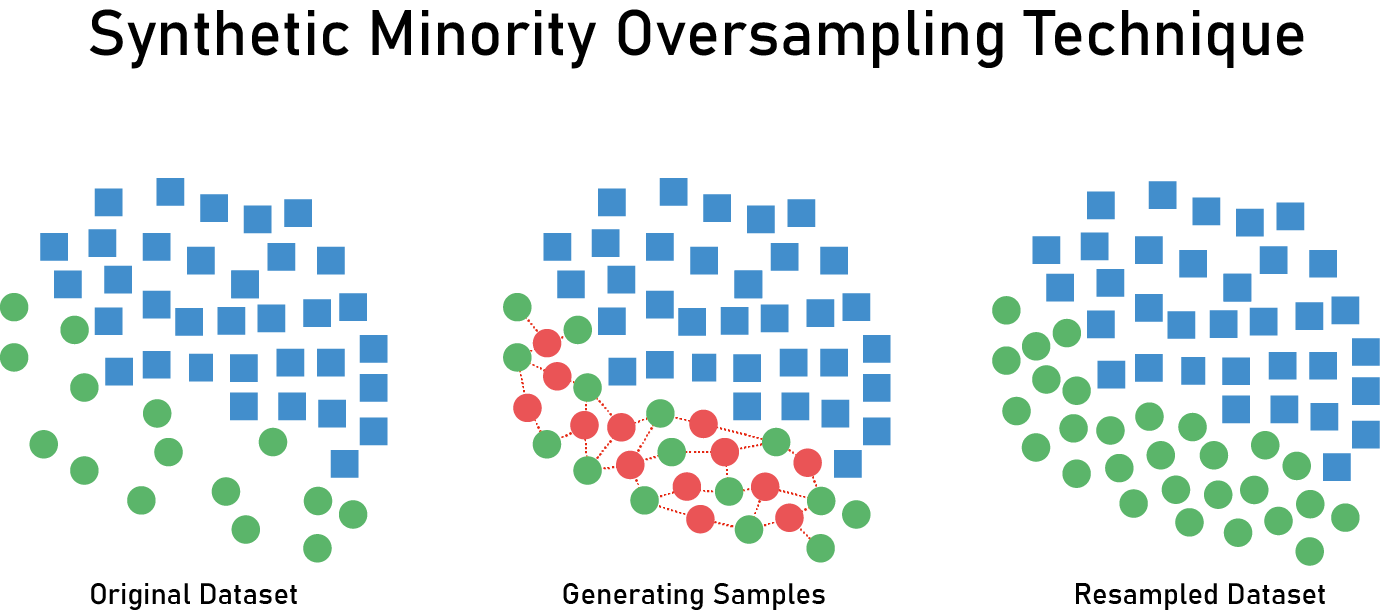
The SMOTE algorithm works as follows:

* You draw a random sample from the minority class.
* For the observations in this sample, you will identify the k nearest neighbors.
* You will then take one of those neighbors and identify the vector between the current data point and the selected neighbor.
* You multiply the vector by a random number between 0 and 1.
* To obtain the synthetic data point, you add this to the current data point.

Using SMOTE we can tweak the model to reduce false negatives, at the cost of increasing false positives. The result of using SMOTE is generally an increase in recall, at the cost of lower precision.

 *Fig 8.4 SMOTE for Oversampling*

Below is an image explaining the working of SMOTE in simplest way:



*Fig 8.5 Applying SMOTE*

Using these techniques, LGBM can be used for classification of the signals- whether they are gravitational wave signals or not. The input from both Hanford and Livingston stations is combined and a learning model is generated out of them. This model is then improvised using LGBM until we obtain a reasonably good model which we can depend on for accurate classifications.

# Ensembling models (XGB, CATBOOST, LGBM, RANDOM FOREST)

In this methodology we are going to train four different models which are XGB, Catboost, LGBM and Random Forest.

Processed the data by excluding the rows where the target values are not -1.

Using the stacking ensemble method, stacked generalization consists in stacking the output of individual estimators and using a classifier to compute the final prediction. Stacking allows us to use the strength of each individual estimator by using their output as input of a final estimator.

An ensemble machine learning approach called "Stacking," or simply "Stacking," uses generalization. It entails using techniques like bagging and boosting to combine the predictions from various machine learning models on the same dataset.

Stacking answers the following query:

How do you decide which machine learning model to utilize (and put your trust in) when there are several that are proficient at solving an issue, but in different ways?

Another machine learning model that learns when to utilize or trust each model in the ensemble will be used as the solution to this problem.

In contrast to bagging, stacking often uses different models that suit the same dataset (e.g., not all decision trees) (e.g. instead of samples of the training dataset).

In contrast to boosting, a single model is employed in stacking to figure out how to most effectively combine the predictions from the contributing models (e.g. instead of a sequence of models that correct the predictions of prior models).

The architecture of a stacking model consists of two or more base models, also known as level-0 models, and a meta-model, or level-1 model, which integrates the predictions of the base models.

Models fitted to training data and whose predictions are compiled are referred to as Level-0 Models (Base-Models).

Model at level 1 (meta-model) that learns the most effective way to combine predictions from base models.

The base models' extrapolations from non-sample data are used to train the meta-model. In other words, data that wasn't used to train the base models is supplied to the base models, which then make predictions and produce the expected results, which serve as the input and output pairs of the training dataset that the meta-model is fitted to.

In the case of regression, the outputs from the base models that are used as input to the meta-model can be real values, while in the case of classification, they can be probability values, probability like values, or class labels.

The most popular method for creating the training dataset for the meta-model is to perform k-fold cross-validation on the base models, using the out-of-fold predictions as the foundation.

The inputs to the base models, such as training data input elements, may also be included in the training data for the meta-model. This can give the meta-model extra context for deciding how to combine its predictions most effectively.

The basic models can be trained on the complete original training dataset whereas the meta-model can be trained separately on this dataset once it has been constructed.

When several separate machine learning models are competent on a dataset but are competent in various ways, stacking is acceptable. Another way to put it is that there is little or no association between the model predictions and the mistakes in those predictions.

Base-models are frequently intricate and varied. Therefore, it is frequently a good idea to utilize a variety of models, such as linear models, decision trees, support vector machines, neural networks, and more, that make quite diverse assumptions about how to accomplish the predictive modeling assignment. As base models, other ensemble techniques like random forests may also be employed.

Use a variety of base models with various presumptions about the prediction goal.

The meta-model is frequently straightforward, allowing for an easy interpretation of the basic model predictions. As a result, linear models are frequently used as the meta-model, such as logistic regression for classification tasks and linear regression for tasks requiring the prediction of numerical values.

Regression Meta-Model: Linear Regression.

Classification Meta-Model: Logistic Regression.

Stacking is sometimes referred to as "blending" in common parlance because the meta-model is a straightforward linear model. In other words, the prediction is an amalgamation of the basis models' projections, weighted on average.It's possible to classify the super learner as a particular kind of stacking.Although stacking is intended to enhance modeling performance, this gain is not always assured.

# Optimization

**Hyperparameter tuning with Keras Tuner**

The variables your chosen machine learning technique employs to adapt to your data are known as the model's parameters. A deep neural network (DNN), for instance, is made up of processing nodes (neurons), each of which performs an action on the data as it passes through the network. Each node in your DNN has a weight value during training that indicates to your model how much of an impact it will have on the outcome of the prediction. These weights are an illustration of a parameter in your model. The parameters of your model are, in many ways, what make it unique from other models of the same type applied to the same set of data.

The selection of effective hyperparameters may frequently make or break a machine learning project.

First, we define a model-building function. It takes a hp argument from which you can sample hyperparameters such as hp.Int ('units', min\_value=32)

Next, instantiate a tuner. You should specify the model-building function, and the name of the objective to optimize (whether to minimize or maximize is automatically inferred for built-in metrics.

HyperResnet and HyperXception are two built-in tunable models that Keras Tuner offers in addition to letting you design your own tunable models. Both of these models do searches across different combinations of the ResNet and Xception architectures.

A single training task is used to execute several trials for hyperparameter tweaking. Each trial consists of a full execution of the training application with selected hyperparameter values set within predetermined bounds. Each trial's findings are recorded by the AI Platform Training training service, which then makes adjustments for following trials. When the task is complete, we can obtain a summary of all the trials as well as the most advantageous configuration of values in light of the given criteria.

The AI Platform Training training service and your training application need to explicitly communicate in order to do hyperparameter tuning.

# Results and Discussions

After looking at the data visualization we conclude that the data comes under time series SFT’s data.

After considering all the models we come to the conclusion that the CNN model has the best accuracy and efficiency.

So, we conclude to use the CNN model for this project.

# Lessons Learnt

We learnt how to work with a team in doing a project, maintaining the data in a git repo and working with the same data with different processing and model training methodologies.

Working with huge data and HDF5 files of hundreds of GB size was also challenging and an important lesson that we learnt in this research paper which would be useful for all of us in the future.

We discovered how to train machine learning models using STF and Time series data whose values are in the form of frequencies.

We gained knowledge of gravitational waves, their benefits to us, and how to find them.

# Future Research works and Limitations

Main limitations of this project were the huge unbalanced data and the fact that there were more data points in the testing set that we needed to predict for than the training set.

The data is also huge which limited us to work on the models at basic settings with limited epochs.

In future as the next steps we would like to concentrate on the CNN model changing the layers in the neural network and focusing on processing the images like gray scaling the images or changing the saturation and contrast values so that the rgb values which show the waves are more enhanced and easily differentiated.

We are considering ensembling multiple models also as another approach where we will be using models which are more compatible to the G2net dataset.

# Acknowledgement

I would like to express my gratitude to Prof. Chandrasekhar Mukherjee, Department of Computer Engineering, San Jose State University, for his continuous guidance, assistance, and encouragement throughout the development of this CMPE - 257 Project.

1. References
2. <https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d>
3. <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=A%20Convolutional%20Neural%20Network%2C%20also,binary%20representation%20of%20visual%20data>.
4. <https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d>
5. [How to Use XGBoost and LGBM for Time Series Forecasting? | 365 Data Science](https://365datascience.com/tutorials/python-tutorials/xgboost-lgbm/)
6. [lightgbm.LGBMModel — LightGBM 3.3.3.99 documentation](https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMModel.html)
7. [SMOTE | Towards Data Science](https://towardsdatascience.com/smote-fdce2f605729)
8. [EditedNearestNeighbours — Version 0.9.1 (imbalanced-learn.org)](https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.EditedNearestNeighbours.html)
9. <https://en.wikipedia.org/wiki/Gravitational_wave>
10. <https://spaceplace.nasa.gov/gravitational-waves/en/>
11. <https://en.wikipedia.org/wiki/Long_short-term_memory>