

# Comparative Analysis of Machine Learning Algorithms for Detection of Fast Radio Bursts

Parth Thakur  
[10520930]

Dissertation submitted in partial fulfilment of the requirements for the degree of  
MSc in Data Analytics  
at Dublin Business School

August 2020

# Declarations

I, Parth Thakur, declare that this dissertation that I have submitted to Dublin Business School for the award of Master's in Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.

Signed: Parth Thakur

Student Number: 10520930

Date: 24/08/2020

# Acknowledgements

I would like to thank my supervisor, Dr. Mehran Rafiee, for his guidance and feedback throughout my research. His suggestions have helped me shape my research and helped me stay motivated to complete the dissertation.

I am also grateful towards the academic staff of Dublin Business School, who have given me the knowledge and inspiration to carry out this research.

Most importantly, I am thankful to my parents Prachi and Pravin Thakur and my friends Archita Jha, Avika D'Souza and Vatsal Chorera. Without their support, I would not have been able to complete my master's degree.

# Abstract

Fast radio bursts are a new astronomical phenomenon that have scientists baffled. To detect and analyse these FRBs, upcoming radio telescopes will capture tremendous amount of data. At the scales of these new telescopes, traditional methods of manually analysing the data fail. There is a need for extensive research in the use of machine learning for this task. This dissertation compares the performance of various machine learning algorithms, so they can be scaled to meet the demands of these new telescopes. To that extent, this research explores ways to simulate FRBs for analysis. The study compares four algorithms, namely, K-Nearest Neighbours, Random Forests, Deep Learning, and Convolutional Neural Networks.

# Table of Contents

<b>Acknowledgements .....</b>	<b>ii</b>
<b>Abstract.....</b>	<b>iii</b>
<b>Chapter 1. Introduction.....</b>	<b>1</b>
1.1 Aims and Objective.....	3
1.2 Structure of the dissertation.....	4
1.3 Dissertation Scope and Limitations.....	4
<b>Chapter 2. Literature Review .....</b>	<b>6</b>
2.1 Overview .....	6
2.2 Deep Learning .....	7
2.3 Convolutional Neural Networks.....	8
2.4 Traditional ML Algorithms .....	9
2.5 Conclusion.....	10
<b>Chapter 3. Background .....</b>	<b>11</b>
3.1 Introduction .....	11
3.2 Properties of Fast Radio Bursts.....	11
3.2.1 Dispersion .....	12
3.2.2 Propagation effects.....	13
3.3 Algorithms.....	14
3.3.1 Random Forest Classifier.....	14
3.3.2 k-Nearest Neighbours Classifier .....	16
3.3.3 Deep Learning.....	17
3.3.4 Convolutional Neural Networks .....	18
3.4 Comparison Parameters.....	20
3.4.1 Confusion matrix .....	20
3.4.2 Receiver Operating Characteristic (ROC) curve .....	23
3.4.3 Time Complexity .....	24
3.5 Summary .....	26
<b>Chapter 4. Data Preparation.....</b>	<b>27</b>
4.1 Hierarchical Data Format .....	29
4.2 Feature Extraction .....	29
4.3 Cross Validation.....	30

<b>Chapter 5. Implementation Methods .....</b>	<b>32</b>
5.1 Introduction .....	32
5.2 The scikit-learn Library .....	32
5.3 Keras and TensorFlow .....	32
5.4 Google Cloud Platform .....	33
<b>Chapter 6. Modelling and Evaluation .....</b>	<b>34</b>
6.1 Introduction .....	34
6.2 K-Nearest Neighbours .....	34
6.2.1 Evaluation .....	35
6.3 Random Forest .....	36
6.3.1 Evaluation .....	37
6.4 Deep Learning .....	38
6.4.1 Evaluation .....	39
6.5 Convolutional Neural Network .....	41
6.5.1 Evaluation .....	42
6.6 Summary .....	43
<b>Chapter 7. Results and Discussion.....</b>	<b>44</b>
7.1 Introduction .....	44
7.2 Comparison of models .....	44
7.2.1 Time Complexity of the models .....	45
7.3 Discussion .....	45
7.4 Limitations .....	46
<b>Chapter 8. Conclusion .....</b>	<b>47</b>
8.1 Future Work .....	47
<b>Bibliography .....</b>	<b>48</b>
<b>Appendix A .....</b>	<b>51</b>
<b>Appendix B .....</b>	<b>53</b>

# List of Figures

Figure 1. Lorimer Burst - First FRB to be discovered.....	1
Figure 3.1 Survival of passengers on the Titanic.....	15
Figure 3.2 Decision boundaries of the 1-NN rule.....	16
Figure 3.3 Illustration of deep learning model.....	17
Figure 3.4 Typical CNN Architecture .....	20
Figure 3.5. Confusion matrix .....	20
Figure 3.6. ROC Curve .....	24
Figure 3.7. Big-O Complexity Chart .....	25
Figure 4.1 Simulated FRB .....	28
Figure 4.2 Simulated FRB with Scintillation.....	28
Figure 4.3 Reading the HDF5 file.....	29
Figure 4.4 Splitting the data into training and testing sets.....	31
Figure 6.1 Code for k-NN Classifier.....	34
Figure 6.2 Confusion matrix for k-Nearest Neighbours Classifier.....	35
Figure 6.3 ROC Curve for k-Nearest Neighbours Classifier.....	36
Figure 6.4 Code for Random Forrest Classifier.....	36
Figure 6.5 Confusion Matrix for Random Forest Classifier .....	37
Figure 6.6 ROC Curve for Random Forest Classifier.....	38
Figure 6.7 Code for Deep Learning .....	38
Figure 6.8 Confusion Matrix Deep Learning Classifier .....	39
Figure 6.9 ROC Curve for Deep Learning Classifier .....	40
Figure 6.10 Code for CNN.....	41
Figure 6.11 Confusion Matrix Convolutional Neural Network Classifier .....	42

Figure 6.12 ROC Curve for Convolutional Neural Network Classifier .....	43
Figure 1.1 Time complexity of the models .....	45

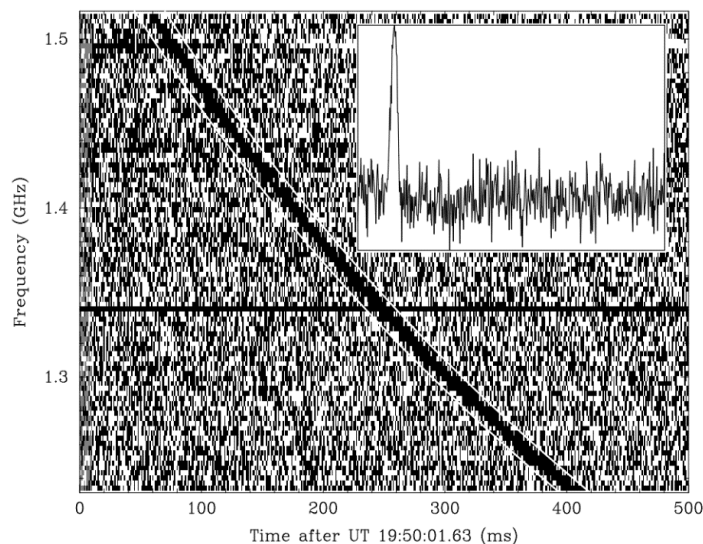
## List of Tables

Table 4.1 Simulation Parameters .....	27
Table 5.1 GCP Virtual Machine Specifications.....	33
Table 6.1 Evaluation parameters for k-NN classifier .....	35
Table 6.2 Evaluation parameters for random forest classifier. ....	37
Table 6.3 Evaluation parameters for deep learning classifier .....	40
Table 6.4 Evaluation parameters for CNN classifier .....	42
Table 1.1 Machine learning models with parameters .....	44

# Chapter 1. Introduction

*"Do not look at stars as bright spots only. Try to take in the vastness of the universe."*

Since Martha Mitchell said that more than a century ago, humanity has uncovered many of the universe's secrets. Similarly, when Karl Jansky detected radio signals coming from the Milky Way, radio astronomy became the new frontier. This discovery opened new avenues for observing the universe and soon many previously unknown objects were identified. These include stars and galaxies and previously unknown categories of objects such as quasars and pulsars. These objects provided novel methods to test many scientific theories, most importantly Einstein's theory of relativity. The recent discovery of Fast Radio Bursts (FRB) however, has scientists baffled.



*Figure 1. Lorimer Burst - First FRB to be discovered (Lorimer et al., 2007)*

FRBs are bright pulses of radio waves with durations in milliseconds or less. They were first reported by Lorimer et. al. in 2007. The main characteristic of an FRB is the dispersion it experiences as it travels through the inter-galactic medium. This is illustrated in figure 1 where the radio pulse shows a sweep across time in frequency domain. This is because, as light travels

through a medium, each frequency of the spectrum experiences a difference in refraction index. This tiny difference makes longer wavelengths of light travel slower than shorter wavelengths. This causes the pulse to broaden over extremely long distances. The amount by which a pulse broadens is known as the dispersion measure (DM). This broadening of the beam requires it to travel over extremely long distances which makes them distinct from terrestrial Radio Frequency Interference (RFI) (Fluke and Jacobs, 2020).

Since 2007, FRBs were discovered in many radio telescopes around the world. In 2014 at Arecibo (Jones et al., 2012), then in 2015 at the Green Bank Telescope (Connor and van Leeuwen, 2018). The Australian Square Kilometre Array has detected many FRBs (Agarwal et al., 2020). All these detections however work on offline data analysis. FRB searches are typically performed on high time resolution data by first accounting for dispersion measures over many trials. A time series is then generated for the de-dispersed data, which is convolved with kernels of various widths to look for broad pulses. Finally chunks of data are flagged for human inspection. This process implies that there is room for improvement in detection rates and accuracy.

It is predicted that there are almost a 100 fast radio bursts per day, but only 117 have been discovered so far due to this technological block (Zhang et al., 2018). This method limits the data to what can be stored, and thus the sensitivity of the telescopes is limited. Hence, with a limited sensitivity, many weak FRBs are undetected. To address this problem, a newer generation of radio telescopes are being constructed. One such telescope, known as the Square Kilometre Array (SKA) will be completed in 2027. Looking for fast radio bursts is one of its primary objectives. The SKA is estimated to produce data at rates higher than 1 terabyte per second (Czech et al., 2018). At these rates, conventional techniques of offline analysis fail. Therefore, new techniques for detecting FRBs in real-time must be developed. More recently, with improvements in the de-dispersion algorithm and modern GPUs, have made real-time

FRBs searches possible. Although these searches are plagued by high false positive rates, caused by terrestrial noise (ex. TV broadcast, WiFi, Satellite Communication, etc.). The high false positive rate presents a challenge as the data still needs to be inspected by humans. To reduce the sheer number of data to be analysed, many algorithms are applied to detect Radio Frequency Interference (RFI). Clustering algorithms to detect common types of RFI have been applied to reduce false positive rates. Techniques like K-Nearest Neighbours (KNN) and Gaussian Mixture Model (GMM) (Wagstaff et al., 2016), Convolutional Neural Networks (CNN) and Long Short Term Memory Models (LSTM) (Wolfaardt, 2016) have been demonstrated to adequately identify single bright pulses from RFI. There are numerous other techniques which use more complex models like Deep Learning.

## 1.1 Aims and Objective

Machine learning has been used in radio astronomy for a long time. The absolute amount of available data lends itself very easily to this task. With new radio telescopes like the SKA and the Canadian Hydrogen Intensity Mapping Experiment (CHIME) starting operations, the available data is going to rise exponentially. Hence, there is a need to build many machine learning algorithms that can handle the enormous data accurately.

The primary objective of this study is to compare a few models for detecting Fast Radio Bursts. Radio Frequency Interference is the main source noise and any machine learning model should be able to detect an FRB and classify it separately from background noise. The newer telescopes also have large data rate capacity. Hence, the primary research question that this study will focus on are:

1. Which widely used machine learning model is best suited for identifying fast radio bursts in real-time?
2. How can the characteristic features be extracted from a radio spectrogram array?

3. How to analyse the real-time performance of these models?

## 1.2 Structure of the dissertation

**Chapter 1 Introduction:** This chapter outlines the motivation for the research and details the aims and objectives of the dissertation.

**Chapter 2 Literature Review:** This chapter is a detailed review of related work, and about talks about various machine learning methods used in the industry.

**Chapter 3 Background:** This section of the study describes some background information of FRB and machine learning necessary for conducting the research.

**Chapter 4 Data Preparation:** This portion of the thesis describes the tools and methods used to simulate FRB.

**Chapter 5 Implementation Methods:** This chapter describes the tools and techniques used to build and train the machine learning algorithms.

**Chapter 6 Modelling and Evaluation:** This section describes the training process of ML algorithms and evaluates them based of conventional evaluation metrics.

**Chapter 7 Results and Discussion:** This chapter compares the results obtained from the previous section and draws comparisons between ML algorithms.

**Chapter 8 Conclusion:** This chapter concludes the thesis by stating the results. It also talks about the future scope of this research.

## 1.3 Dissertation Scope and Limitations

The data rates for radio telescopes are expected to reach 1 terabyte per second. Simulating such high data rates on a personal computer is an impossible task. Hence, this dissertation will work

on a scaled down version of the data. This dissertation also assumes that the data has been passed through RFI mitigation software. Despite that, the simulated data is based on reality and the models can be expected to work on real data as well.

## Chapter 2. Literature Review

### 2.1 Overview

Exploration in the use of machine learning in astronomy has been carried out since the 1980s (Fluke and Jacobs, 2020). In their paper “*Surveying the reach and maturity of machine learning and artificial intelligence in astronomy*” Fluke and Jacobs talk about the use of machine learning in astronomy. This is mainly due to a history of archiving massive amounts of data. This archive crossed the 100-terabyte mark near the end of the 20<sup>th</sup> century and soon it will reach the exabyte scale. Hence the use of artificial intelligence will keep getting more prominent. Their paper lists the use of many ML algorithms for several tasks. They document the use of ANN, CNN, SVM, RF, and k-M on spectroscopy data. In conclusion, they talk about how ML algorithms are necessary in the future as telescopes start producing an ever-increasing amount of data. (Fluke and Jacobs, 2020)

Newer radio telescopes like the Canadian Hydrogen Intensity Mapping (CHIME) and the upcoming Square Kilometre Array (SKA) are expected to produce data at a rate of 100 petabytes per year. At such a high rate, several challenges must be addressed. These challenges include:

1. Low power digital processing.
2. Adaptive machine learning algorithms.
3. Scalable data archives for storage.

Since the SKA is anticipated to generate such a high volume of data, storing it is impossible. This creates a problem for offline processing of this data. In the paper “*Big Data Challenges for Large Radio Arrays*” Jones et. al. outline these challenges for large radio telescope arrays.

They also suggest a few currently in-use techniques to aid in the processing of the data produced by the SKA. One such technique is a Data Triage installed in the Very Long Baseline Array (VLBA). A data triage uses machine learning algorithms instead of hardwired systems to classify radio signals. Data triages can be used to extract information before the data stream is compressed and vital information is lost. It can also adapt the sampling rate in real-time based on RFI characteristics (Jones et al., 2012). These data triages need robust classification algorithms which can reliably classify radio pulses. Hence testing a variety of solutions is important. They conclude by telling about the importance of research in this field. Exploration done in the context of large radio telescopes like the SKA can have huge impact on other industries like biotech, climate research, telecommunications, and many others.

Since their paper was published in 2012, many researchers have proposed models to detect and classify radio transients. This chapter will highlight the use of ML in detection of radio transients.

## 2.2 Deep Learning

Connor and Leeuwen proposed a deep learning model for classifying Fast Radio Bursts in their 2018 paper. They construct a tree like deep neural network that takes multiple or individual data products as input. Their train dataset was built using data from real telescopes, mainly CHIME and Apertif. However, most of the data used is in the form of simulated FRB since there is a lack of confirmed real events. Nevertheless, their model produces high accuracy and recall rates and demonstrate that Deep Learning can be implemented to detect FRB in real telescopic data (Connor and van Leeuwen, 2018).

Furthering Connor and Leeuwen's work, a deep learning classifier known as FETCH (Fast Extragalactic Transient Candidate Hunter) was released by Agarwal et. al. in 2019. Their classifier uses deep learning to analyse vast amount of radio spectroscopy data and finds

transient radio pulses. They use a technique of transfer learning to train a neural network to classify FRB from RFI. The main advantage of this model is that it uses real RFI data from the Green Bank Observatory instead of simulating gaussian noise which often differs from real noise vastly. They produced 11 models each with a recall over 99.5% on their test data. Another benefit of this model is that it is frequency and telescope agnostic. That is, the model can be adapted to other telescopes with a different bandwidth. As such, they demonstrate their model by finding FRB in ASKAP (Australian SKA Pathfinder) and Parkes data. The FETCH models have a classification time of  $12 \pm 1$  ms per candidate on NVIDIA GTX-1070Ti GPU. Hence, their model can classify candidates in real time if the candidate rate stays below  $10^8$  per hour (Agarwal et al., 2020).

### 2.3 Convolutional Neural Networks

FRB121102 is currently the only known source of repeating FRB pulses. In their paper, “*Fast Radio Burst 121102 Pulse Detection and Periodicity: A Machine Learning Approach*”, (Zhang et al., 2018) describe the use of Convolutional Neural Networks (CNN) to find 72 new pulses from this source. Their main argument focuses on the fact that FRB are simple compared common objects like cars and faces where CNN are used. The dispersed profile is detected in the early layers of a CNN. However, the network must be robust enough to manage highly noisy data, especially to detect candidates near the noise amplitude. The model must also be effective at classifying FRB from RFI. They demonstrate their model on a set of simulated pulses with a range of signal to noise ratio (S/N) and obtain a recall of 88% and a precision of 98%.

CNN have also been used in tandem with Long Short-Term Memory (LSTM) for detection of transient RFI. Transient RFI is a type of interference that is cause by devices like florescent light, air conditioners, etc. Such RFI is detrimental to radio surveys as it triggers many false

positives. In their paper “*A CNN and LSTM-Based Approach to Classifying Transient Radio Frequency Interference*”, (Czech et al., 2018) the authors tackle this problem. Their solution is to build a simple ML model that can detect RFI in the false positive triggers for FRB. Their model consisting of a 1D convolutional layer, followed by a bidirectional LSTM layer and finally a fully connected layer, presenting the output in a 1-hot configuration. Their model was able to classify signals from seven devices with a precision and recall of 84% and 91% respectively.

## 2.4 Traditional ML Algorithms

The V-FASTR method of classifying FRB from RFI is discussed in the paper “*A Machine Learning Classifier for Fast Radio Burst Detection at the VLBA*” (Wagstaff et al., 2016). V-FASTR (VLBA Fast Transient) was created to search data collected by VLBA (Very Long Baseline Array) for FB. The model employs a trained random forest classifier to predict the class for each new candidate, then consults a database of known pulsars to further refine its predictions. The classifier retains predictions with confidence over 90% only which reduces the workload on human reviewer by at least 80%. With this, they achieved an accuracy of over 99%. Hence, the Random Forest classifier was shown to detect radio transients reliably.

Further, in his thesis n “*Machine learning approach to radio frequency interference (RFI) classification in radio astronomy*” (Wolfaardt, 2016) describes the use of K-Nearest Neighbours (KNN) and Gaussian Mixture Model (GMM) to classify RFI. His research concluded with demonstrating that KNN and GMM can classify 15 types of RFI with 70.80% and 65.56%, respectively.

## 2.5 Conclusion

The literature review has shown various ML algorithms can reliably detect transient radio pulses as well as transient RFI. Algorithms like Deep Learning and CNN can easily reach 95% accuracy on simulated data. There are many other studies that test other algorithms for FRB detection (Fluke and Jacobs, 2020). However, a recent study that compares the performance of these algorithms could not be found.

## Chapter 3. Background

### 3.1 Introduction

The fast radio burst catalogue is currently very limited. With only 172 known sources, and even considering samples from repeating sources, the dataset is not large enough to train a machine learning algorithm. Hence, it is necessary to simulate FRB with the correct properties and train machine learning models on this dataset. Fortunately, FRB are simple in nature and it is effortless to simulate them. This chapter is a summarisation of the paper “*Fast Radio Bursts*” (Petroff et al., 2019) They explain fast radio bursts in detail, which provides hints on the means to simulate them. Further, the main objective of this research is to compare for commonly used machine learning models. As discussed in earlier sections, many ML algorithms are used in radio astronomy. The most frequently used models and approaches to compare them are explained in detail in this chapter.

### 3.2 Properties of Fast Radio Bursts

Interstellar and intergalactic space is considered a vacuum by all. However, it is only close to a perfect vacuum. There are various things dispersed throughout, like stellar dust, free electrons, etc. As light travels through space over very large distances, the minute effects of the interstellar medium add up. Although, the research community currently has no strict standard for defining FRB. Any radio pulse that broadly meets the criteria is considered an FRB. These criteria include the pulses dispersion measure (DM), pulse duration, brightness, and broadband-ness, etc.

### 3.2.1 Dispersion

Dispersion is the property of light, where the speed of light is a function of its frequency. Hence, when a broadband ray travels through a medium, it causes light of lower frequencies to be slower than that of higher frequencies. This phenomenon is most evident in a prism. When white light enters one side, it is split into seven colours due to dispersion.

Even though space is considered a vacuum, it is not a perfect void. There is a large amount of dust and free electrons in interstellar and intergalactic space. These add up over large distances and disperse light as it travels. The amount by which light is dispersed is known as the Dispersion Measure (DM). The amount of dispersion in a radio pulse is calculated as a measure of time delay between the highest ( $\nu_{hi}$ ) and lowest ( $\nu_{lo}$ ) frequencies.

Hence, the time delay ( $\Delta t$ ) is given by:

$$\Delta t = \frac{e^2}{2\pi m_e c} (\nu_{lo}^{-2} - \nu_{hi}^{-2}) DM \approx 4.15 (\nu_{lo}^{-2} - \nu_{hi}^{-2}) DM \sim \text{ms} \quad (1)$$

Where,  $m_e$  is the classical mass of an electron and  $c$  is the speed of light in vacuum. The dispersion measure is given as,

$$DM = \int_0^d n_e(l) dl \quad (2)$$

In this expression,  $n_e$  is the electron number density,  $l$  is a path length and  $d$  is the distance to the FRB.

The measured pulse width,  $W$ , is generally considered for each FRB to be a combination of an intrinsic pulse of  $W_{int}$  period and instrumental spreading contributions. Usually for a top-hat pulse,

$$W = \sqrt{(W_{int}^2 + t_{samp}^2 + \Delta t_{DM}^2 + \Delta t_{DMerr}^2 + \tau_s^2)} \quad (3)$$

where  $t_{samp}$  is the sampling time as above,  $\Delta t_{DM}$  is the dispersive delay across an individual frequency channel and  $\Delta t_{DMerr}$  represents the dispersive delay due to dedispersion at a slightly incorrect DM.  $\tau_s$  is the temporal broadening factor caused due the FRB travelling through a turbulent medium.

### 3.2.2 Propagation effects

#### 3.2.2.1 Scintillation.

Given the fact that FRB originate from different galaxies and their implied small emitting regions, FRB are considered a point source of light. Thus, the radio waves will scintillate.

Scintillation is caused by refractive and diffractive effects as light passes through the clumpy and turbulent intervening material which has variations in the density of electrons on a variety of spatial scales. The signal delays can cause destructive or constructive interference when these waves come together again. That generates a complex frequency structure in the observer's plane that varies with time. The relative motion between the medium of observer, source, and scatter dominates the time variation of the pattern of scintillation observed at Earth. The characteristic frequency scale is called the scintillation bandwidth, while the characteristic timescale for a scintle to persist is called the scintillation time. The scintillation is approximated by the positive half of the cosine function with a random phase:

$$A = \cos\left(2\pi N_{scint} \left(\frac{\nu}{\nu_{ref}}\right)^{-2} + \phi_{scint}\right) \quad (4)$$

where  $A$  is the amplitude of the pulse,  $N_{scint}$  is the number of scintillations (uniformly distributed from 0 to 10), and  $\phi_{scint}$  is some random phase (uniformly distributed from 0 to 1).

### 3.2.2.2 Scattering.

FRB are temporally broadened by scattering. This induces multi-path propagation and thus, a later arrival time for parts of the signal. Scattering is the reason the sky is blue during the day. As white light interacts with particles in the atmosphere, light of higher frequency gets scattered more. This effect can also be observed in intergalactic signals, as they travel through a host of mediums.

In the simple case of a thin and infinitely extended scattering screen, this effectively convolves the FRB pulse with a one-sided exponential decay. In this simple picture, the decay time of this exponential tail scales strongly with frequency. The scattering is defined as:

$$s(t) = \frac{1}{\tau_\nu} e^{-t/\tau_\nu} \quad (5)$$

$$\tau_\nu = \tau_0 \left( \frac{\nu}{\nu_{ref}} \right)^{-4} \quad (6)$$

Here,  $\tau_\nu$  is the scattering timescale at frequency  $\nu$ , with  $\nu_{ref}$  acting as a reference frequency.

There are various other propagation effects that FRB experience as they travel over large distances. Effects like faraday rotation, angular broadening, plasma lensing, etc. However, these effects are minute and will not cause much difference to the ability of a machine learning algorithm in detecting an FRB. Hence, they are not considered during the simulation of FRB dataset.

## 3.3 Algorithms

### 3.3.1 Random Forest Classifier

The random forest classifier is an ensemble model that uses a multitude of decision trees and uses the mode of the classes predicted by these trees as its output. Random forests overcome the overfitting drawback of decision trees.

### 3.3.1.1 Decision Tree learning

The decision tree model uses a series of conditions on the input to arrive at the output class. These conditions create a tree like structure with the tree branching at each condition. The leaves of the tree form the output classes.

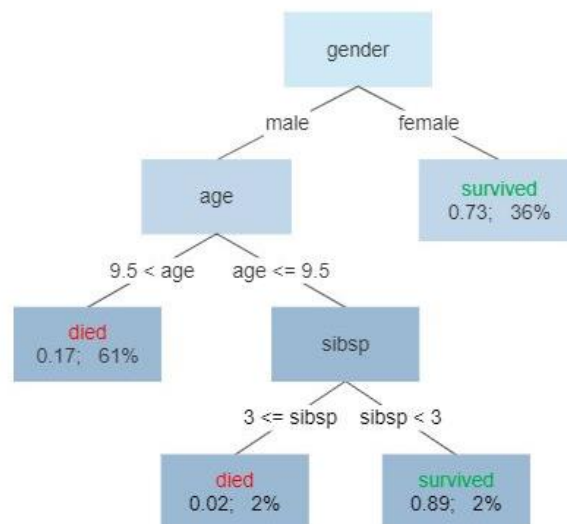


Figure 3.1 Survival of passengers on the Titanic (Gilgoldm, 2020)

Figure 3.1 illustrates a decision tree model created for the titanic dataset. A tree is built by splitting the source set consisting the root node of the tree into subsets – which are split into constituting children. The splitting is carried out in a recursive manner until no more splits can be performed (Shalev-Shwartz and Ben-David, 2014a). Although, decision tree models can be very non-robust and a small change in training data will result in large changes in the outcome. To over-come this limitation, the random forest algorithm is used.

### 3.3.1.2 Bootstrap Aggregating

During training, bootstrap aggregating or bagging is used on individual tree learners. Given a training set  $X$  with  $x_i$  input points corresponding to  $y_i$  outputs, bagging repeated creates a random sample with replacement. The trees are then trained on the bagging samples. Random forests, however, change the bagging algorithm to apply it to apply to the feature set. If one or

fewer features induce a bias in the decision tree algorithm, feature bagging will select these features in many trees, and their output will be correlated. (James et al., 2013).

After training, predictions for input data is carried out by taking the majority vote from all the individual classification trees.

### 3.3.2 k-Nearest Neighbours Classifier

k-NN algorithms are one of the simplest of all machine learning algorithms. They memorise the entire training dataset and then predict any new instance based on the labels of its closest neighbours in the training data. Generally, the distance metric used is the Euclidian distance ( $\rho$ ) which is given by:

$$\rho(x, x') = \|x - x'\| = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2} \quad (7)$$

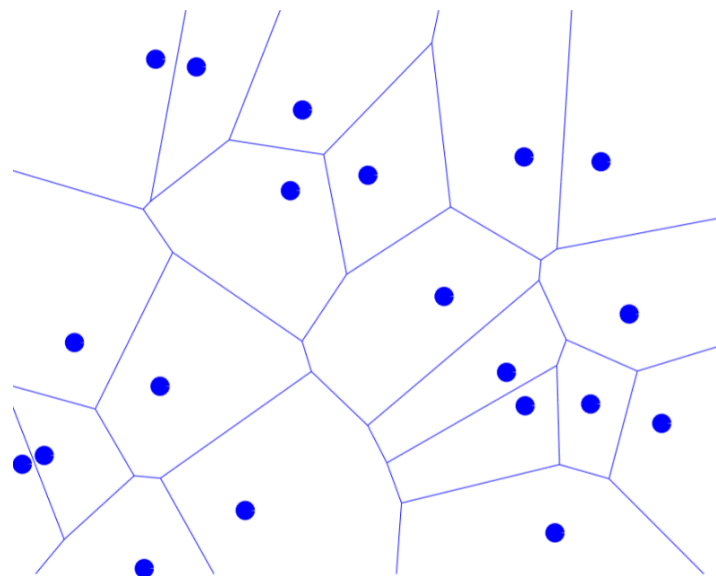


Figure 3.2 Decision boundaries of the 1-NN rule (Shalev-Shwartz and Ben-David, 2014b)

Figure 3.2 illustrates the decision boundaries of the 1-NN rule. “The points shown are sample points and the predicted class of any new label will be the label of the sample point in the centre of each cell it belongs to.” (Shalev-Shwartz and Ben-David, 2014b). Similarly, n-Dimensional

data can be classified using n-dimensional decision boundaries. Hence, when dealing with data with more than 10 dimensions, it is necessary to perform dimension reduction.

### 3.3.3 Deep Learning

An artificial neural network is an algorithm inspired by the structure of neurons in the human brain. Basically, in the human brain, computation devices (neurons) perform simple calculations on the inputs they receive. These calculations are fed to other neurons which in turn feed their calculations more. Until finally, through a complex structure of simple calculations, the brain can carry out highly complex computations. ANN are formal computation constructs that are modelled after this computation paradigm (Goodfellow et al., 2016a).

Deep learning is a machine learning method based on the artificial neural network. The “deep” part in Deep Learning refers to the depth of the neuron network. An example of a deep learning method is the Multi-Layer Perceptron (MLP) which is illustrated in Figure 3.3.

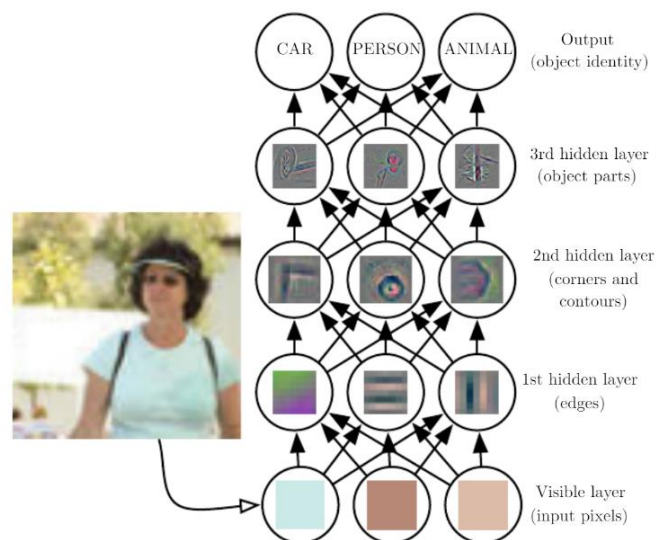


Figure 3.3 Illustration of deep learning model (Goodfellow et al., 2016a)

The rationale is that, since a computer cannot be easily taught the high-level features of objects in the real world, it should be taught the relationships between the simple characteristics of the

object. In Figure 3.3, the task of identifying a person, car or an animal in a picture is broken down to three tasks. Computers can easily identify edges in an image when pixel information is provided. With the edge information provided by the first layer, the second layer can then detect corners and other patterns. With the details of patterns in edges, the third layer can easily find parts of objects. Finally, with the parts of the image identified, the output layer can identify the object in the image (Goodfellow et al., 2016a).

### 3.3.4 Convolutional Neural Networks

From (Goodfellow et al., 2016b),

*Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*

Convolutional neural networks (CNN) have been very successful at solving complex tasks that algorithms like deep learning fail to solve. The reason behind this is the convolutional layers applied to the input. Under the hood, CNN are just deep learning models but with convolutional filters before the MLP inputs. Convolutional filters use a mathematical operation called “convolution”. The operation used in CNN for the convolutional layer is known as a “kernel”. This operation acts as a filter on input data, and with every step, information is abstracted from the input. Convolutional layers also reduce the dimensions in the input space considerably, without losing information.

#### 3.3.4.1 The Convolution Operation

The convolution operation when performed on two functions produces a third function describing how they both interact. Specifically, how the second function changes the shape of the first. It is given by the formula:

The convolution operation when performed on two functions produces a third function describing how they both interact. Specifically, how the second function changes the shape of the first. It is given by the formula:

$$s(t) = (x * w)(t) = \int x(a)w(t - a)da \quad (8)$$

Where,  $x(a)$  is the input function and  $w(a)$  is the kernel. These functions are continuous which is hardly the case in computing. The inputs to a CNN will be discretized in time. Hence, when  $x$  and  $w$  are discrete, their convolution is given by:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{a=\infty} x(a)w(t - a) \quad (9)$$

For two-dimensional input, as used in this research, the kernel  $K$  must be two dimensional too:

$$s(t) = (I * K) = \sum_m \sum_n I(m, n)K(i - m, i - n) \quad (10)$$

Where,  $m$  and  $n$  are the dimensions of the input.

#### 3.3.4.2 Pooling

A CNN convolutional layer typically consists of three stages. The first is the convolutional layer which applies the kernel to the input and produces a set of linear activations. The second is the activation layer, which translates the linear activation to a non-linear activation – generally using the rectified linear activation function. Then finally a pooling layer which replaces the output at certain places with a summary statistic. This is generally the local maximum and is known as the max pooling layer.

The convolutional layers output is then applied to a fully connected – *dense* – layer, which produces the output label. The typical architecture of a CNN is shown in Figure 3.4

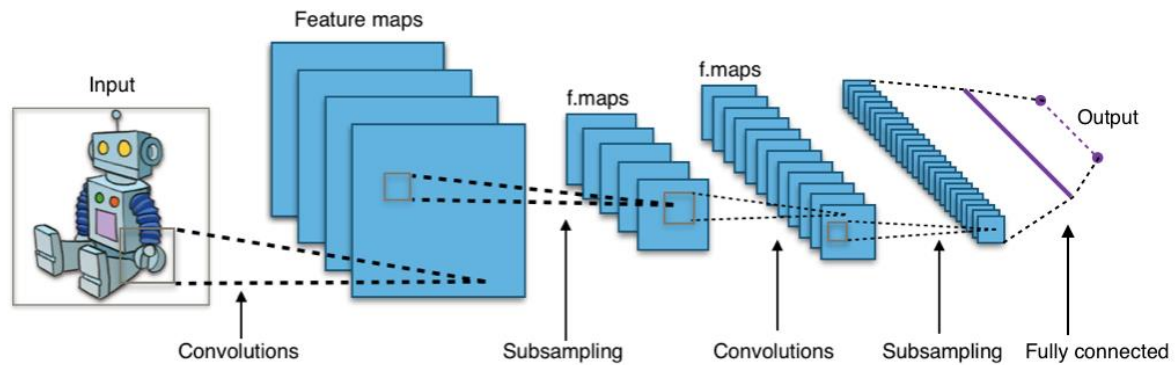


Figure 3.4 Typical CNN Architecture (Aphex34, 2015)

These additional convolutional layers, however, take a long time to process. It takes powerful Graphic Processing Units (GPU) to train CNN quickly. Although, once trained, they show extreme compatibility in real-time applications.

### 3.4 Comparison Parameters

#### 3.4.1 Confusion matrix

Every classification algorithm generates a confusion matrix. It relates the predicted class to the true class of the data. Confusion matrices are very useful in inferring the performance of a machine learning model.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 3.5. Confusion matrix (Narkhede, 2018)

As illustrated in Figure 3.5, a confusion matrix provides four measures for a model: True positives (TP), False positives (FP), False Negatives (FN), True Negatives (TN). These four measures are crucial for evaluating a machine learning model.

True positives are the number of items the model correctly guess as a positive case. In case of the models classifying FRB, a TP will be the correct labelling of an FRB event. Likewise, a True negative is when the model correctly labels a negative case (RFI in this case). Conversely, false positives and false negatives measure the incorrectness of the model. Using these measures, four additional quantities can be computed. Namely: Accuracy, Specificity, Recall and precision.

#### 3.4.1.1 Accuracy

Classification accuracy of a machine learning classifier is the ratio of correct predictions to the number of total predictions made. With respect to the confusion matrix, accuracy is given by,

$$accuracy = \frac{true\ positive + true\ negative}{total\ sample} \quad (11)$$

Accuracy is a good metric to gauge the performance of a model, although it can be misleading in many cases. Consider a model that classifies all candidates as RFI. In a dataset with 99% RFI, this model will be accurate 99% of the time. Such a model is not of any use. To overcome this drawback, other metrics are used along with accuracy to evaluate a machine learning model.

#### 3.4.1.2 Recall

Recall is the ability of a model to detect all the relevant classes in a dataset (Koehrsen, 2018). Naturally, labelling all the data points as a single class is not useful, nevertheless, such a model will have a 50% accuracy score. Recall can be calculated as true positives divided by the sum of true positives and false negatives and is given by the equation:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (12)$$

If all cases are labelled as FRB, the recall goes to 1.0. That does not mean we have a perfect classifier. The trade-off here is precision which decreases as recall increases.

### 3.4.1.3 Precision

Precision measures the proportion of the data points a model states were relevant, were relevant (Koehrsen, 2018). It is given as the ratio of true positives to the sum of true positives and false positives. Precision is expressed by the equation:

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (13)$$

In the case that a model labels all data points as FRB, the precision score is 0.5. Further in the case where it labels all data points as RFI, precision and recall are 0 as there are no true negatives.

Precision and recall must be observed in tandem. Say if the model is tweaked so that it detects only one FRB. The precision in this case will be 1.0 as there are no false positives. However, the recall will be very low as there are many false negatives. On the other hand, if all candidates are classified as FRB, the recall will be 1.0, but there will be many false positives and human workload will increase. Thus, precision and recall have an inverse proportionality relationship.

### 3.4.1.4 F1 Score

In many cases a model is built to maximise either recall or precision. For example, in medical screening, it is preferable to have a recall of one – catching all positive cases – if the cost of follow up examination is not high. In this case, an optimal blend of precision and recall is required. Ideally, FRB and RFI must be classified with the highest accuracy such that all FRB are detected, but RFI are not, reducing the workload of scientists.

The F1 score is a harmonic mean of precision and recall (Koehrsen, 2018), and is given by the formula:

$$F_1 = 2 * \frac{\textit{precision} * \textit{recall}}{\textit{precision} + \textit{recall}} \quad ( 14 )$$

A harmonic mean is preferred, as it penalizes extreme values. A simple mean of 1 and 0 is 0.5, but the harmonic mean is 0. Hence, when precision and recall must be given equal weightage, the F1 score is used.

### 3.4.2 Receiver Operating Characteristic (ROC) curve

Typically, a model will a value between 0 and 1 for each class. A threshold can then be set to classify between FRB and RFI. A ROC curve shows how the relationship between precision and recall changes, as the threshold is changed. It plots the true positive rate on the y-axis and the false positive rate on the x-axis. The true positive rate (TPR) is the recall and the false positive rate (FPR) is the probability of a false alarm (Stephanie, 2016). Both can be calculated from the confusion matrix:

$$\textit{true positive rate} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false negatives}} \quad ( 15 )$$

$$\textit{false positiverate} = \frac{\textit{false positives}}{\textit{false positives} + \textit{true negatives}} \quad ( 16 )$$

An example ROC curve is shown in Figure 3.6

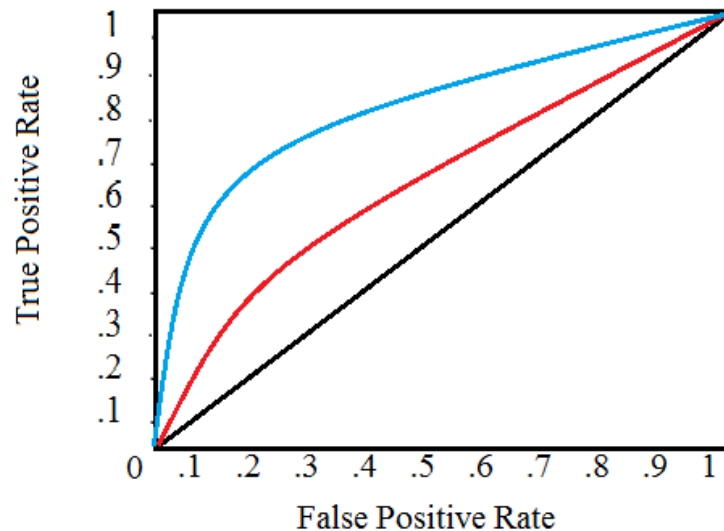


Figure 3.6. ROC Curve (Stephanie, 2016)

The black diagonal line represents a random classifier, while the blue and red lines represent a machine learning model. For a classifier if the threshold is set to 1, all cases are identified as negative. Hence, the true positive rate (TPR) and false positive rate (FPR) are both 0. As the threshold is decreased, more data points are identified as positive, until the threshold reaches 0, where all cases are positive and TPR and FPR are both equal to 1.

Consequently, if the curve is above diagonal line, the classifier is better than a random classifier. This can be quantified by calculating the area under the curve (AUC) of the ROC curve. Hence, a higher AUC indicates a better classifier. A random classifier achieves an AUC of 0.5 (Streiner and Cairney, 2007).

### 3.4.3 Time Complexity

Time complexity of an algorithm simply refers to how long it takes for the algorithm to execute. Such a metric is extremely important in real-time cases, as the algorithm needs to keep up with incoming data. If the execution time is too high, a trade-off must be made by either discarding data or down sampling it, or higher operational memory must be installed. Using computational complexity has the benefits of ignoring the differences like the computer power and

architecture used at runtime and the underlying programming language, allowing users to focus on the fundamental differences of the elementary operations of the algorithms (garychl, 2020).

### 3.4.3.1 Big O Notation

Computational complexity of an algorithm is commonly represented by the Big O notation. Simply, it defines the worst-case computation time for an algorithm. Figure 3.7 visualizes different cases of complexities.

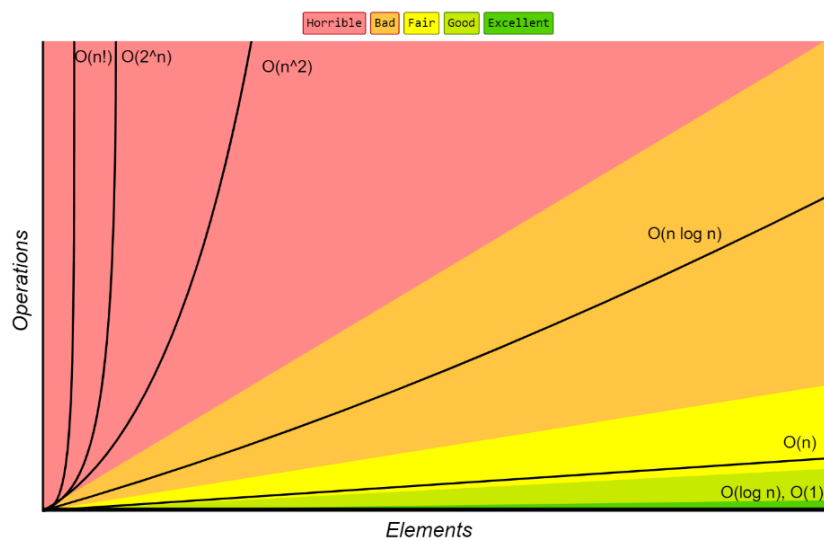


Figure 3.7. Big-O Complexity Chart (Rowell, 2012)

Elements in Figure 3.7 represents the number of data points supplied to an algorithm. This can be interpreted as the size of the test dataset. Operations on the y-axis reveal the computational cost of the algorithm with respect to the number of elements in the dataset. As observed, the  $O(1)$  complexity is in the “excellent” category. Algorithms with  $O(1)$  time complexity run independently from input size. This an idealistic condition and machine learning algorithms cannot achieve it. ML algorithms aim to be as good as  $O(n)$ , where time complexity scales linearly with input size.

It should be noted that calculating the Big-O for a complex algorithm is extremely difficult, if not impossible. Calculating Big-O requires users to keep track of operation time of every step

in the algorithm. The algorithm needs to be broken down to individual steps, and the Big-O must be calculated for each of them (Yse, 2020). ML procedures execute hundreds of steps under the hood which make it unfeasible to calculate the Big-O for it. Nevertheless, it can be inferred from the wall-time of the prediction process. By supplying the algorithm with inputs of different sizes, it is possible to understand the underlying complexities of the algorithm.

### 3.5 Summary

Fast radio bursts in their journey across the universe go through many mediums that have numerous effects on them. The most prominent is dispersion, wherein light of lower frequencies arrive later than light of higher frequencies. This gives FRB their characteristic sweep across the frequency – time plot.

There are also several machine learning algorithms that can be used to detect these FRB. Methods like Random Forrest, k-Nearest Neighbour, Deep Learning and Convolutional Neural Networks each have their own pros and cons of detecting objects. To compare their performance, metrics like the F1 score and the Big-O notation is used.

## Chapter 4. Data Preparation

The FRB simulation dataset is created using the *injectFRB* tool used in the paper *Applying Deep Learning to Fast Radio Burst Classification* (Connor and van Leeuwen, 2018). In their simulation process, they first create a 1-dimensional array of gaussian numbers. This array is then cloned across a 2<sup>nd</sup> dimension to create the gaussian profile of an FRB. They then apply the exponential scattering function to the array, creating a scattered profile for the pulse. Finally, frequency scintillation is added. The scintillated FRB is then dispersed, giving it the structure of a real FRB. The output of the above simulation is added to gaussian noise background making the simulation closer to real FRB events.

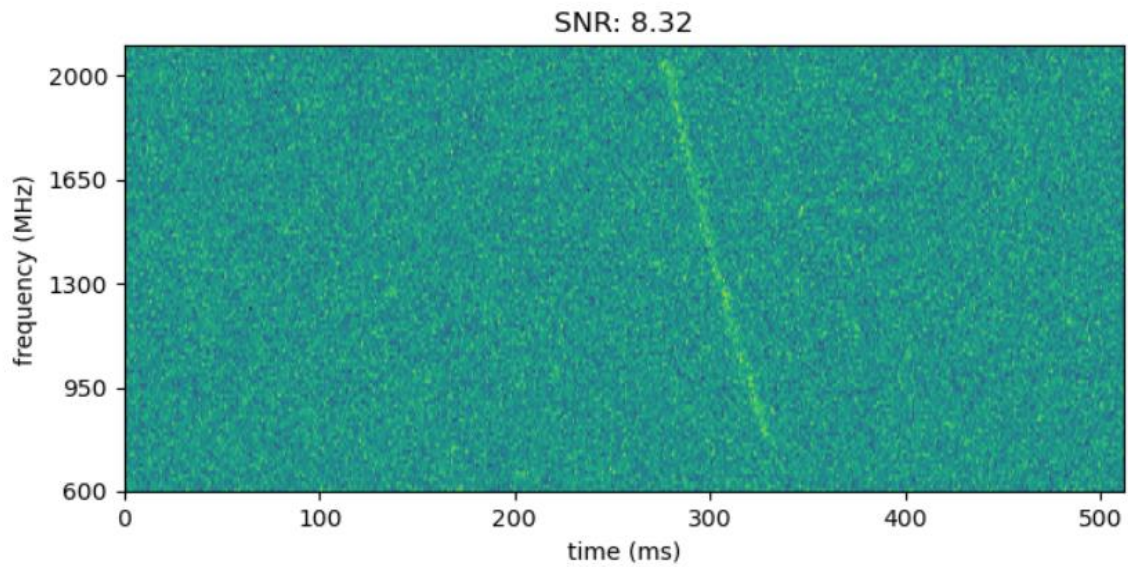
The following table shows the parameters used to simulate FRB based on the equations above:

Parameter	Value
Array shape	(128, 512)
Max duration	250 ms
$\tau_v$	0.1
$\nu_{min}$	600 MHz
$\nu_{max}$	2100 MHz
$\nu_{ref}$	1350 MHz

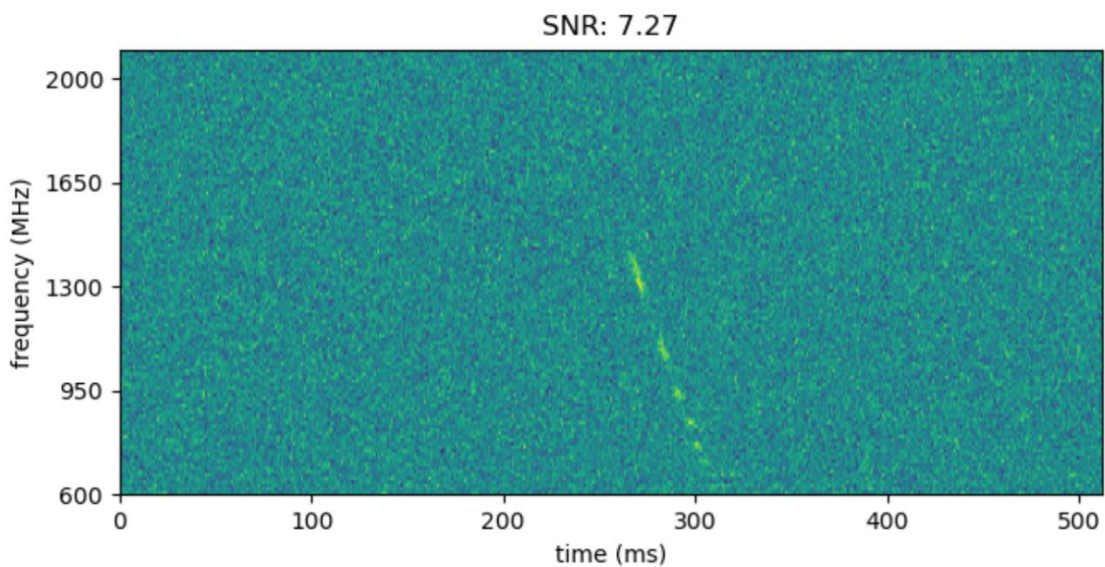
*Table 4.1 Simulation Parameters*

Although the frequency range of the SKA is from 50 MHz to 20 GHz (“SKA Technology - SKA UK,” n.d.) observations of FRB under 400 MHz and over 2 GHz have not been made. Many of the FRB detected are around 1400 MHz (The HDF Group, 2006). Hence, the frequency range for the simulation was chosen to be 600 MHz – 2100 MHz. The reference frequency ( $\nu_{ref}$ ) is the middle frequency.  $\tau_v$  is the dispersion index – discussed in section 3.2.1. The Maximum duration is the maximum duration of any FRB in the array.

Figure 4.1 and Figure 4.2 show the frequency – time plot of the simulated FRB. Figure 4.2 clearly shows the scintillation that is observed in many FRB.



*Figure 4.1 Simulated FRB*



*Figure 4.2 Simulated FRB with Scintillation*

The simulations are stored as 2D NumPy arrays with the shape 128, 512. To train a machine learning model reliably, 5000 simulations were created. A few more examples are available in Appendix A.

## 4.1 Hierarchical Data Format

The Hierarchical Data Format is a set of file system formats used to store complex and large datasets. Version 5 of HDF, known in short as HDF5, supports storage of multidimensional arrays of homogenous data. It also supports creation of groups, which act as containers for data or other groups (The HDF Group, 2006). The HDF5 format is a perfect container to store the simulated FRB and background NumPy arrays. The file contains two groups named ‘FRB’ and ‘BAK’. The FRB group stores the simulated FRB while the BAK group stores the simulated background. During the modelling process, the HDF5 file can be read as shown in Figure 4.3.

```
import h5py
with h5py.File('data/200823-150951.h5', 'r') as data:
    print(data.keys())
    print(data['FRB'].shape)

    X = np.empty((10000, 128, 512))

    t = tqdm(data['FRB'])
    t.set_description('Reading FRBs')
    X[::2] = [a for a in t]

    t = tqdm(data['BAK'])
    t.set_description('Reading backgrounds')
    X[1::2] = [a for a in t]

    y = np.empty((10000,))
    y[::2] = np.ones(data['FRB'].shape[0])
    y[1::2] = np.zeros(data['BAK'].shape[0])
```

*Figure 4.3 Reading the HDF5 file*

Here, the  $X$  variable contains alternating FRB and Background arrays and the  $y$  variable contains alternating 1s and 0s.

## 4.2 Feature Extraction

Traditional machine learning algorithms like k-NN and Random Forests are not designed to take raw input in the form of 2D arrays. Hence, it is necessary to extract features from the input data in such a way that it is representative of the input. Transforming the raw input into a set

of features is known as feature extraction. For this research, seven statistical features were selected. Namely:

1. Maximum mean of frequency channels.
2. Maximum standard deviation of frequency channels.
3. Maximum skewness of frequency channels.
4. Maximum kurtosis of frequency channels.
5. Maximum variance of frequency channels.
6. Mean of maximum of frequency channels.
7. Mean of the entire channel.

Like skewness, kurtosis is a measure that describes the shape of the data. It measures how spread out the data is from the mean. If kurtosis is high, it implies that a significant portion of the data is further than  $3\sigma$  from the mean.

The feature extraction process is not applied to Deep Learning and CNN as their intrinsic structure is designed to learn the patterns in raw input data.

### 4.3 Cross Validation

Cross validation is a technique used in ML evaluation to test the robustness of an algorithm. Generally, when a ML technique is trained on a dataset, it quickly learns the patterns in it and achieves high accuracy. However, it is necessary to test the model on data that it has not seen. Hence, cross validation is used to check how well a machine learning model with generalize to separate data.

A form of cross validation involves partitioning the input dataset into training and testing data. Usually the split is, 80% of the input as training and 20% as testing. The ratio of the split is completely depended on the ML architect. This research applies the train – test split method on the simulated FRB – RFI dataset in a 70/30 manner. The code for the splitting process is

illustrated in Figure 4.4. This means, that a model will be supplied 7,000 arrays containing 3,500 FRB and RFI each.

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=260795)
```

*Figure 4.4 Splitting the data into training and testing sets.*

Where, X is an array containing 10,000 FRB and backgrounds, and y is an array of 1s and 0s.

For modelling, an output of 1 is considered detection of FRB, and 0 is considered background.

Hence, the training and testing dataset is created with \*\_train arrays containing the training data, and \*\_test arrays containing the testing data.

## Chapter 5. Implementation Methods

### 5.1 Introduction

The machine learning models chosen for comparison are built using the python programming language. Python is a highly versatile language, allowing users to quickly build ML models and evaluate them. Python has a host of data science tools and libraries which are simple to use. The libraries used in this research are `scikit-learn` and `keras`. Scikit-learn is used to implement the traditional ML methods, while keras is used to build the neural network models.

This research makes use of a virtual machine created on the Google Cloud Platform which is discussed in more detail in section 5.4 of this chapter.

### 5.2 The scikit-learn Library

Scikit-learn, known in short as sklearn, is a software package that provides methods to build various ML models. Along with these, it also provides numerous functions for evaluation and cross validation of the models. The most useful ability of sklearn is its universally applied *fit* method. Every machine learning model built using sklearn has the fit method which takes two arguments. First is an array of training features, and the second is an array of corresponding outputs. To make predictions using the model, the *predict* method is used, which is supplied and array of features for prediction (scikit learn, n.d.).

### 5.3 Keras and TensorFlow

*Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow.*

(Keras, n.d.). Keras was designed to be fast and easy to use. The main advantage of using keras is its ability to compute using the Graphics Processing Unit (GPU). GPUs are exponentially more powerful at computing than CPUs, which reduces the training time for neural networks drastically. Another great feature is the *Sequential* method, which provides a user-friendly way to build neural networks. It allows users to add layers to a network in a stepwise manner, making it easy to edit the network. Keras uses a TensorFlow backbone for building neural networks. TensorFlow is a symbolic math library developed at Google.

## 5.4 Google Cloud Platform

Google Cloud Platform (GCP) is a group of services provided by Google, that enables users to run virtual machines on Google's internal infrastructure. GCP makes it easy to build virtual machines with state-of-the-art hardware inexpensively. One of the most important features offered by GCP for this research is the ability to add a high-end GPU. GCP allows addition and removal of GPUs on demand. The specifications of the virtual machine used to test the ML algorithms is provided in Table 5.1.

Component	Specification
CPU	Intel Skylake Platform, 8 virtual cores
GPU	NVIDIA Tesla K80
RAM	30 GB
Operating System	CentOS 8

*Table 5.1 GCP Virtual Machine Specifications*

# Chapter 6. Modelling and Evaluation

## 6.1 Introduction

As discussed in section 5 of this research, ML algorithms are built using the scikit-learn and Keras libraries in python. The models are evaluated on a virtual machine hosted on GCP. This chapter details the process of building and training each of the chosen algorithms, as well as evaluating their performance. The performance metrics used for evaluation are discussed in section 3.4. The ‘.train’ and ‘.evaluate’ functions used below are describes in a separate python package created for this research.

## 6.2 K-Nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier
from helpers import BenchmarkTM
knn_clf = KNeighborsClassifier(n_neighbors=3)
knn_clf, train_params = BenchmarkTM.train(knn_clf, X_train, y_train,
                                         label='K-Nearest Neighbours Classifier')
test_params, roc_curve, big_o_curve = BenchmarkTM.evaluate(knn_clf, X_test, y_test,
                                                           label='K-Nearest Neighbours Classifier',
                                                           sample_range=SAMPLE_RANGE)
```

*Figure 6.1 Code for k-NN Classifier*

The k-NN classifier is first initialised with the *n\_neighbors* parameter set to 3. This signifies that the model will look at three closest data points to label an input. The model is then fit to the training data and evaluated.

## 6.2.1 Evaluation

## 6.2.1.1 Confusion Matrix

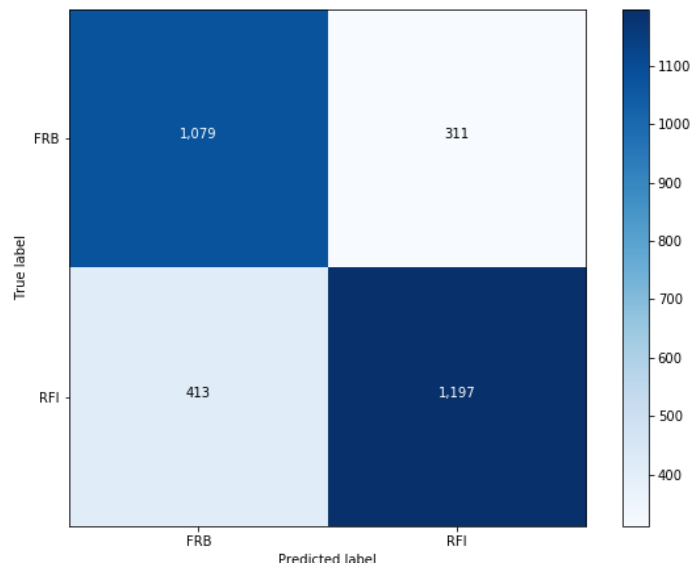


Figure 6.2 Confusion matrix for *k*-Nearest Neighbours Classifier

The confusion matrix for *k*-NN classifier is shown in Figure 6.2. From the figure, it can be inferred that, True positives: 1,079, True negatives: 1,197, false positives: 311, false negatives: 413. These parameters help to determine the evaluation metrics for the model. They are listed in Table 6.1.

Parameter	Value
Accuracy	75.87%
Precision	0.78
Recall	0.72
F1 Score	0.75

Table 6.1 Evaluation parameters for *k*-NN classifier

From Table 6.1 and Figure 6.2, it is understood that the *k*-NN classifier is a fine classifier. However, there are many false positives which will increase human workload, while many FRB are classified as RFI.

### 6.2.1.2 ROC Curve

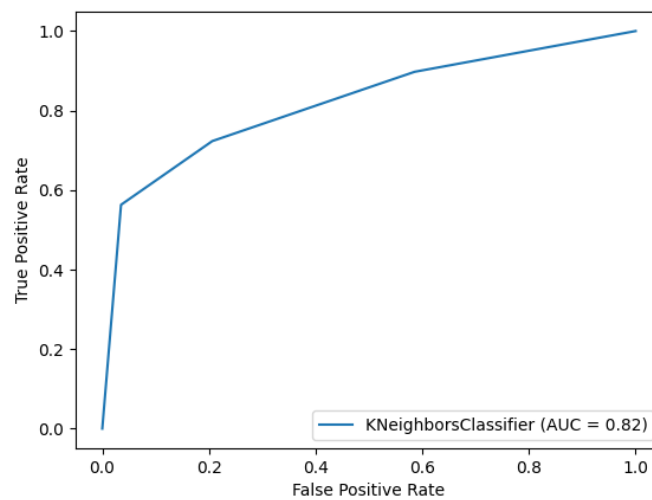


Figure 6.3 ROC Curve for *k*-Nearest Neighbours Classifier

The ROC curve for *k*-NN classifier is shown in Figure 6.3. The area under the curve (AUC) for the model is 0.82. This means there is an 82% chance that the model will be able to distinguish between RFI and FRB.

## 6.3 Random Forest

```

from sklearn.ensemble import RandomForestClassifier
from helpers import BenchmarkTM
rf_clf = RandomForestClassifier(n_estimators=10)
rf_clf, train_params = BenchmarkTM.train(rf_clf, X_train, y_train,
                                         label='Random Forest Classifier')
test_params, roc_curve, big_o_curve = BenchmarkTM.evaluate(rf_clf, X_test, y_test,
                                                           label='Random Forest Classifier',
                                                           sample_range=SAMPLE_RANGE)

```

Figure 6.4 Code for Random Forrest Classifier

The random forest classifier is first initialised with *n\_estimators* parameter set to 10. This the number of internal decision trees used by the algorithm to classify the input. Since the FRB data is simplistic in nature, the number of estimators is set low to prevent over fitting, and to

reduce the computation cost of the model. The model is then fit to the training data and evaluated.

### 6.3.1 Evaluation

#### 6.3.1.1 Confusion Matrix

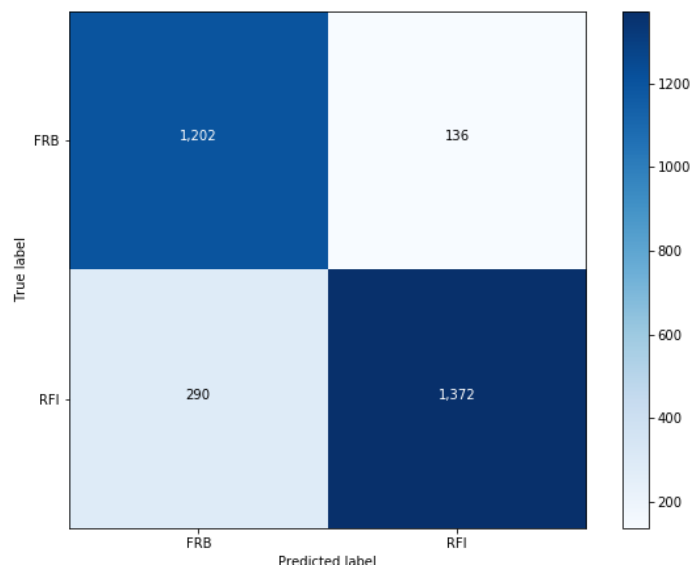


Figure 6.5 Confusion Matrix for Random Forest Classifier

The confusion matrix for random forest classifier is shown in Figure 6.5. From the figure, it can be inferred that: true positives: 1,202, true negatives: 1,372, false positives: 136, false negatives: 290. These parameters determine the model evaluation metrics listed in Table 6.2.

Parameter	Value
Accuracy	85.80%
Precision	0.9
Recall	0.81
F1 Score	0.85

Table 6.2 Evaluation parameters for random forest classifier.

From Figure 6.5 and Table 6.2 it is understood that the random forest classifier is a good classifier. With more than 85% accuracy it can distinguish between FRB and RFI fairly. The false positives are also low, although a significant portion of true FRB are classified as RFI.

### 6.3.1.2 ROC Curve

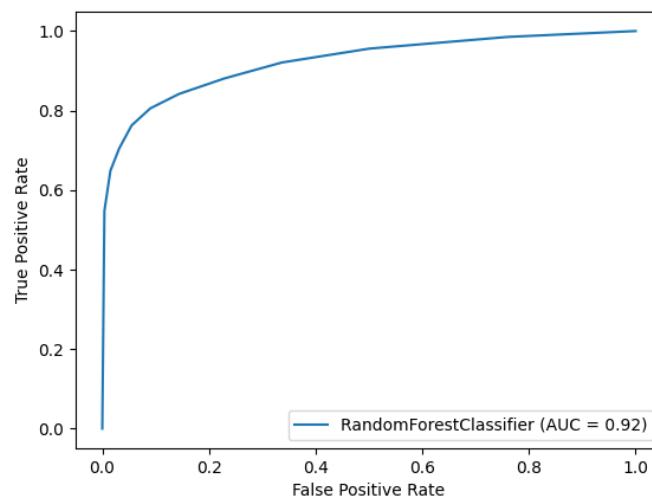


Figure 6.6 ROC Curve for Random Forest Classifier

The ROC curve for random forest classifier is shown in Figure 6.3 Figure 6.6. The area under the curve (AUC) for the model is 0.92. This means there is a 92% chance that the model will be able to distinguish between RFI and FRB.

## 6.4 Deep Learning

```

from keras.models import Sequential
from keras.layers import Dense

dl_clf = Sequential()
dl_clf.add(Dense(12, input_dim=128 * 512, activation='relu'))
dl_clf.add(Dense(24, activation='relu'))
dl_clf.add(Dense(1, activation='sigmoid'))
dl_clf.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

dl_clf, train_params = BenchmarkNN.train(dl_clf, X_train, y_train,
                                         label='Deep Learning Classifier')
test_params, roc_curve, big_o_curve = BenchmarkNN.evaluate(dl_clf, X_test, y_test,
                                                           label='Deep Learning Classifier',
                                                           sample_range=SAMPLE_RANGE)

```

Figure 6.7 Code for Deep Learning

The deep learning model is constructed using the *Sequential()* function for the keras library. The network is made up of four layers, with the first and last layers being the input and output layers, respectively. There are two fully connected hidden layers. The first hidden layer has 12 neurons with *input\_dim* parameter set to  $128 * 512 = 65536$ , which is the dimensionality of the input arrays. The second hidden layer is again a fully connected (dense) layer with 24 neurons. Both the hidden layers use the rectified linear activation function (relu), which forces the output of each neuron to be positive. Any negative output is casted to 0. Finally, a dense layer with 1 neuron acts as the output layer. The activation function used here is the *sigmoid* function with output between 0 and 1. Hence, the network outputs the probability of positive class.

#### 6.4.1 Evaluation

##### 6.4.1.1 Confusion Matrix

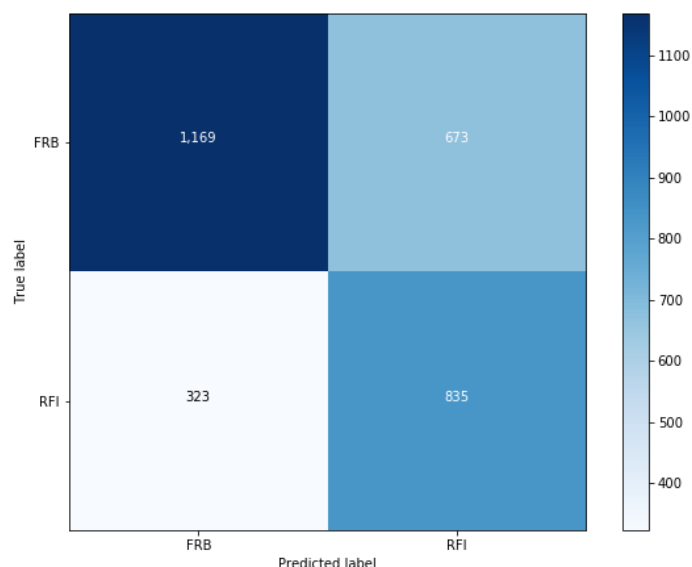


Figure 6.8 Confusion Matrix Deep Learning Classifier

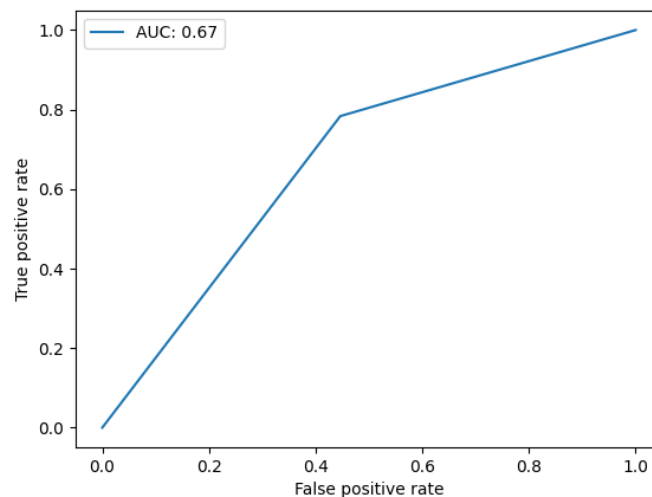
The confusion matrix for deep learning classifier is shown in Figure 6.8. From the figure, it can be inferred that: true positives: 1,196, true negatives: 835, false positives: 673 and false negatives: 323. These parameters determine the model evaluation metrics listed in Table 6.3

Parameter	Value
Accuracy	66.80%
Precision	0.63
Recall	0.78
F1 Score	0.70

*Table 6.3 Evaluation parameters for deep learning classifier*

From Figure 6.8 and Table 6.3 it can be concluded that deep learning classifier cannot reliably on the FRB simulation data. With an accuracy of 66.80%, it cannot consistently classify FRB and RFI. Its main disadvantage is the high false positives, which is 673.

#### 6.4.1.2 ROC Curve



*Figure 6.9 ROC Curve for Deep Learning Classifier*

The ROC curve for deep learning classifier is shown in Figure 6.9. The area under the curve (AUC) for the model is 0.67. This means there is only a 67% chance that the model will be able to distinguish between RFI and FRB.

## 6.5 Convolutional Neural Network

```

from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Flatten, Dropout
from keras.optimizers import SGD

X_train = X_train.reshape(-1, 128, 512, 1)
X_test = X_test.reshape(-1, 128, 512, 1)

conv_clf = Sequential()

conv_clf.add(Conv2D(filters=16, kernel_size=(3,3), activation='relu', input_shape=(128, 512, 1)))
conv_clf.add(MaxPooling2D(strides=(2,2)))

conv_clf.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu'))
conv_clf.add(MaxPooling2D(strides=(2,2)))

conv_clf.add(Flatten())
conv_clf.add(Dropout(0.2))

conv_clf.add(Dense(12, activation='relu'))
conv_clf.add(Dense(1, activation='sigmoid'))

opt = SGD(lr=0.05)
conv_clf.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
conv_clf, train_params = BenchmarkNN.train(conv_clf, X_train, y_train, label='CNN Classifier')

```

Figure 6.10 Code for CNN

CNN models use the *Sequential()* and *Dense()* in the same way as deep learning model. A major difference though, is the addition of the convolutional layers before the fully connected dense layers. As shown in Figure 6.10, the CNN model built for this research uses two convolutional layers. The first layer has 16 filters of size 3 x 3 with activation set to *relu*. The *input\_dim* parameter here is two dimensional, as CNN are great at recognising patterns in 2D arrays. The second layer has 32 filters of the same shape. A max pooling layer is added after each convolutional layer. The max pooling layer looks at a 4 x 4 section of the output of previous layer and keeps the maximum activation in that section. This reduces the dimensionality of the output significantly. Finally, a dense layer with 12 neurons is added which is connected to the output layer. The model's output is the probability of true class.

## 6.5.1 Evaluation

## 6.5.1.1 Confusion Matrix

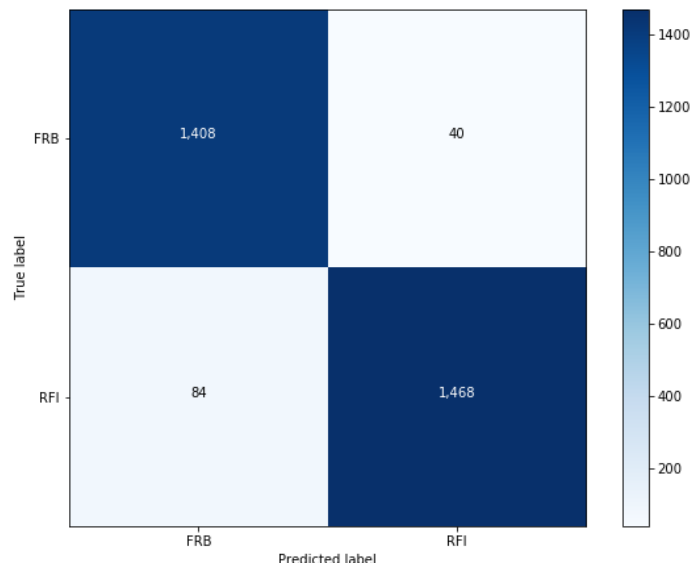


Figure 6.11 Confusion Matrix Convolutional Neural Network Classifier

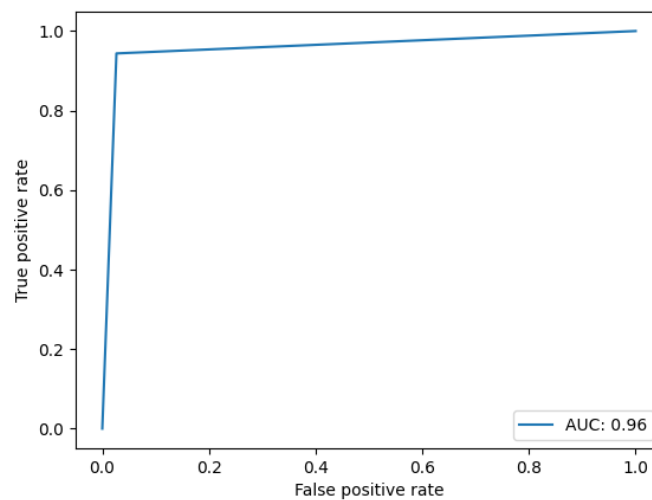
The confusion matrix for CNN classifier is shown in Figure 6.11. From the figure, it can be inferred that: true positives: 1,408, true negatives: 1,468, false positives: 40, false negatives: 84. These parameters can be used to determine the model evaluation parameters listed in Table 6.4

Parameter	Value
Accuracy	95.87%
Precision	0.97
Recall	0.94
F1 Score	0.96

Table 6.4 Evaluation parameters for CNN classifier

From Figure 6.11 and Table 6.4, it is deduced that the CNN classifier is a great model. It can correctly classify 95% of the candidates as FRB and RFI. The false positives and false negatives are also very low.

### 6.5.1.2 ROC Curve



*Figure 6.12 ROC Curve for Convolutional Neural Network Classifier*

The ROC curve for CNN classifier is shown in Figure 6.12. The area under the curve (AUC) for the model is 0.96. This means there is a 96% chance that the model will be able to distinguish between RFI and FRB.

## 6.6 Summary

Hence, in this section, four machine learning algorithms were trained and evaluated. The parameters used to construct the models were determined by trial and error, and the best ones are presented here.

# Chapter 7. Results and Discussion

## 7.1 Introduction

In this chapter the findings of this research are presented and discussed. This section also aims to answer the questions posed during the preliminary research in a succinct manner. This chapter also addresses the research questions posed during the preliminary research. Finally, a brief description of the limitations is described.

## 7.2 Comparison of models

Table 7.1 Machine learning models with parameters provides an at-a-glance comparison of the tested machine learning models.

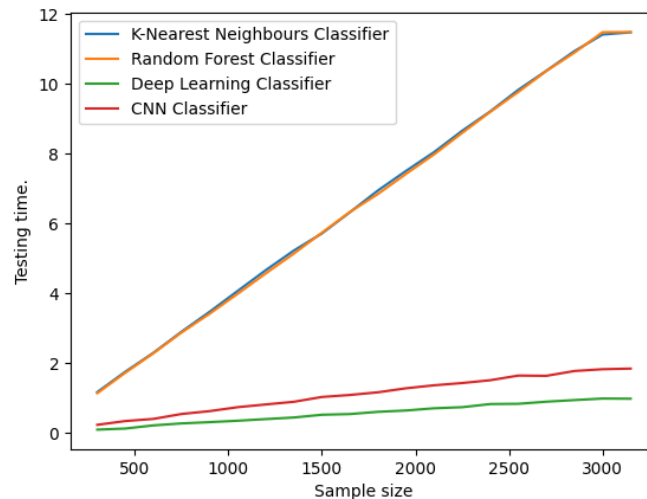
<b>Model</b>	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F1 Score</b>	<b>Runtime (s)</b>
k-Nearest Neighbours	75.87%	0.78	0.72	0.75	11.69
Random Forest	85.80%	0.90	0.81	0.85	11.42
Deep Learning	66.80%	0.63	0.78	0.70	2.43
Convolutional Neural Network	95.87%	0.97	0.94	0.96	2.48

*Table 7.1 Machine learning models with parameters*

From Table 7.1, Convolutional Neural Networks have the best performance among all. CNN demonstrate a very high 95% accuracy, which even the deep learning model does not achieve. K-NN and Deep Learning have similar performance in terms of recall and F1 score. Although, deep learning model is quicker to run, its accuracy is low. Random Forest achieve a considerable score too.

### 7.2.1 Time Complexity of the models

The time complexity of the models was calculated by supplying the model with varying sizes of testing data. Figure 7.1 shows the graph between number of samples and the time taken by a model to predict their labels.



*Figure 7.1 Time complexity of the models*

From Figure 7.1, it is inferred that the traditional ML algorithms are very slow to predict. Their speed drastically drops with increasing sample size. The neural networks, however, have a much better performance. A reason for this is the feature selection step that need to be carried out on the input space for the traditional ML algorithms.

Further, CNN are observed to be slightly slower than deep learning models. This is due to the convolutional layers adding to the computation cost of the network. Nevertheless, their precision at classifying FRB from RFI is drastically better than Deep Learning.

## 7.3 Discussion

From the results in section 6 and the analysis in this section, the best suited model to detect FRB from RFI is a Convolutional Neural Network. Not only are they significantly better at

classifying FRB, they can be scaled easily with increasing data rates. This is the most important feature to have for upcoming radio telescopes like the Square Kilometre Array.

Further, it was observed that the Random Forest classifier is decent at the task. However, their time complexity is high, and they will not scale with the data rates produced by large radio telescopes. Nevertheless, they can be used as a last stop check to reduce the number of candidates required to be scanned by humans.

To train the machine learning algorithms, statistical feature extraction was used. Features like kurtosis, skewness, and standard variance of the frequency channels were used to train and predict using traditional machine learning models. For the neural network models, no feature extraction was carried out, as their internal structure is designed to find patterns in raw input data.

The real-time performance of the models was analysed by inferring the big-o characteristics of the models. This was done by plotting a graph of time required for testing with respect to number of samples tested.

#### 7.4 Limitations

One of the main limitations of this thesis is its simulation of RFI data. Radio Frequency interference is assumed to be gaussian which is rarely the case in real life. RFI is plagued with signals from terrestrial radio sources like TV towers and satellites. These signals were ignored in this thesis to limit the scope.

Another limitation is that the feature extraction process is simplistic in nature. For this thesis, only statistical features of the frequency channels were used. However, there may be other features, characteristic to fast radio bursts that may potentially boost the performance of the models.

## Chapter 8. Conclusion

This research aimed to identify the best performing machine learning model for classification of fast radio bursts and Radio Frequency Interference. Many machine learning algorithms are used for this task, out of which four were chosen for comparison. Namely: K-Nearest Neighbours, Random Forest, Deep Learning, and Convolutional Neural Networks. For this, fast radio burst data was simulated after thoroughly understanding their properties, and random gaussian noise was used for radio frequency interference. It was found that convolutional neural networks have the best performance in terms of accuracy score and precision. Although deep learning had faster run time on sample data, their performance was sub-optimal. The other two models' performance was fair, but the overhead of feature extraction caused them to be slow during prediction. Hence, it was concluded that the convolutional neural network is the best to be used in large radio telescopes like the square kilometre array.

### 8.1 Future Work

The main assumption of this research is that radio frequency interference is gaussian in nature. However, gaussian noise is too simple to simulate real RFI. Real RFI is very erratic and cannot be modelled easily. A future extension of this research will use real examples of RFI and test the robustness of the algorithms. These examples must come from real telescopes currently in operation. There is also scope of improvement in the feature extraction for fast radio bursts. More research can be carried out on the intrinsic properties of FRB and a feature extraction algorithm can be developed. Such algorithms need to have a better time complexity than simply extracting the statistical features.

# Bibliography

- Agarwal, D., Aggarwal, K., Burke-Spolaor, S., Lorimer, D.R., Garver-Daniels, N., 2020. FETCH: A deep-learning based classifier for fast transient classification. *Monthly Notices of the Royal Astronomical Society*. <https://doi.org/10.1093/mnras/staa1856>
- Aphex34, 2015. Typical CNN Architecture [WWW Document]. CC BY-SA 4.0. URL <https://commons.wikimedia.org/w/index.php?curid=45679374>
- Connor, L., van Leeuwen, J., 2018. Applying Deep Learning to Fast Radio Burst Classification. *The Astronomical Journal* 156, 256. <https://doi.org/10.3847/1538-3881/aae649>
- Czech, D., Mishra, A., Inggis, M., 2018. A CNN and LSTM-based approach to classifying transient radio frequency interference. *Astronomy and Computing* 25, 52–57. <https://doi.org/10.1016/j.ascom.2018.07.002>
- Fluke, C.J., Jacobs, C., 2020. Surveying the reach and maturity of machine learning and artificial intelligence in astronomy. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. <https://doi.org/10.1002/widm.1349>
- garychl, 2020. How to Compare Machine Learning Algorithms - Towards Data Science [WWW Document]. *Towards Data Science*. URL <https://towardsdatascience.com/how-to-compare-machine-learning-algorithms-ccc266c4777> (accessed 8.21.20).
- Gilgoldm, 2020. Decision Tree [WWW Document]. CC BY-SA 4.0. URL <https://commons.wikimedia.org/w/index.php?curid=90405437> (accessed 8.21.20).
- Goodfellow, I., Bengio, Y., Courville, A., 2016a. Introduction, in: *Deep Learning*. pp. 1–26.
- Goodfellow, I., Bengio, Y., Courville, A., 2016b. Convolutional Networks, in: *Deep Learning*. MIT Press, p. 326.
- James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. Bagging, Random Forests, Boosting, in: *An Introduction to Statistical Learning*. pp. 316–321.

- Jones, D.L., Wagstaff, K., Thompson, D.R., D’Addario, L., Navarro, R., Mattmann, C., Majid, W., Lazio, J., Preston, R., Rebbapragada, U., 2012. Big data challenges for large radio arrays, in: IEEE Aerospace Conference Proceedings. <https://doi.org/10.1109/AERO.2012.6187090>
- Keras, n.d. About Keras [WWW Document]. URL <https://keras.io/about/> (accessed 8.24.20).
- Koehrsen, W., 2018. Beyond Accuracy: Precision and Recall [WWW Document]. [towardsdatascience.com](https://towardsdatascience.com). URL <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c> (accessed 8.21.20).
- Lorimer, D.R., Bailes, M., McLaughlin, M.A., Narkevic, D.J., Crawford, F., 2007. A bright millisecond radio burst of extragalactic origin. *Science* 318, 777–780. <https://doi.org/10.1126/science.1147532>
- Narkhede, S., 2018. Understanding Confusion Matrix. When we get the data, after data... [WWW Document]. [Towards Data Science](https://towardsdatascience.com). URL <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62> (accessed 8.24.20).
- Petroff, E., Hessels, J.W.T., Lorimer, D.R., 2019. Fast radio bursts. *Astronomy and Astrophysics Review*. <https://doi.org/10.1007/s00159-019-0116-6>
- Rowell, E., 2012. Big-O Algorithm Complexity Cheat Sheet [WWW Document]. Big O Sheet. URL <https://www.bigocheatsheet.com/> (accessed 8.21.20).
- scikit learn, n.d. An introduction to machine learning with scikit-learn — scikit-learn 0.23.2 documentation [WWW Document]. URL <https://scikit-learn.org/stable/tutorial/basic/tutorial.html> (accessed 8.24.20).
- Shalev-Shwartz, S., Ben-David, S., 2014a. Decision Trees, in: *Understanding Machine Learning: From Theory to Algorithms*. pp. 250–256.
- Shalev-Shwartz, S., Ben-David, S., 2014b. Nearest Neighbor, in: *Understanding Machine Learning: From Theory to Algorithms*. pp. 258–265. <https://doi.org/10.1017/cbo9781107298019.020>

SKA Technology - SKA UK [WWW Document], n.d. URL <https://unitedkingdom.skatelescope.org/ska-technology/> (accessed 8.25.20).

Stephanie, 2016. C-Statistic: Definition, Examples, Weighting and Significance - Statistics How To. Statistics How To.

Streiner, D.L., Cairney, J., 2007. What's under the ROC? An introduction to receiver operating characteristics curves. *Canadian Journal of Psychiatry* 52, 121–128. <https://doi.org/10.1177/070674370705200210>

The HDF Group, 2006. Introduction to HDF5 (Slides) [WWW Document]. URL <https://support.hdfgroup.org/HDF5/doc/H5.intro.html> (accessed 8.23.20).

Wagstaff, K.L., Tang, B., Thompson, D.R., Khudikyan, S., Wyngaard, J., Deller, A.T., Palaniswamy, D., Tingay, S.J., Wayth, R.B., 2016. A machine learning classifier for fast radio burst detection at the VLBA. *Publications of the Astronomical Society of the Pacific* 128, 084503. <https://doi.org/10.1088/1538-3873/128/966/084503>

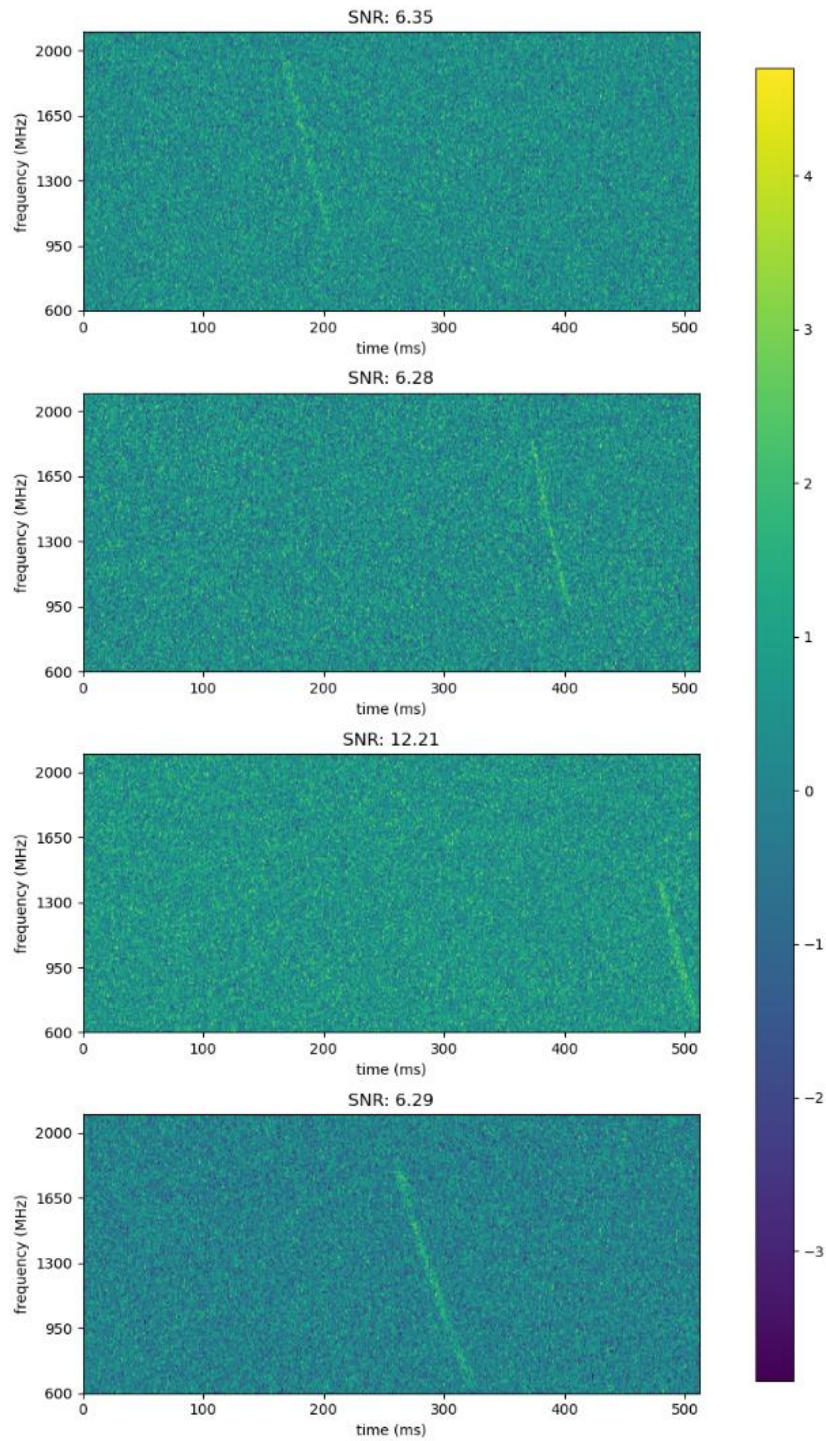
Wolfaardt, C.J., 2016. Machine learning approach to radio frequency interference(RFI) classification in radio astronomy.

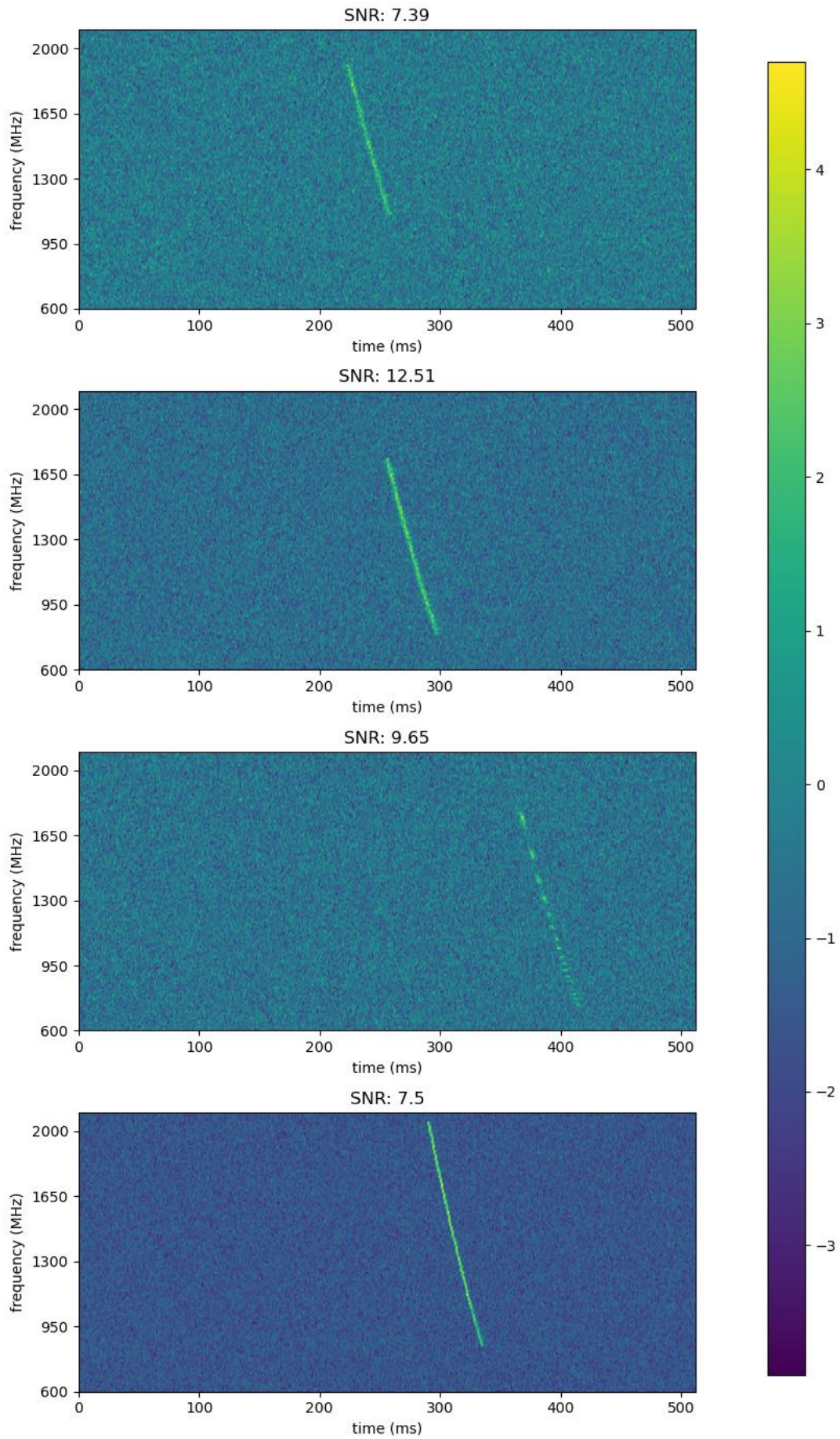
Yse, D.L., 2020. Time Complexity: How to measure the efficiency of algorithms [WWW Document]. KDnuggets. URL <https://www.kdnuggets.com/2020/06/time-complexity-measure-efficiency-algorithms.html> (accessed 8.21.20).

Zhang, Y.G., Gajjar, V., Foster, G., Siemion, A., Cordes, J., Law, C., Wang, Y., 2018. Fast Radio Burst 121102 Pulse Detection and Periodicity: A Machine Learning Approach. *The Astrophysical Journal* 866, 149. <https://doi.org/10.3847/1538-4357/aadf31>

# Appendix A

## Examples of Simulated FRB





## Appendix B

The simulated data is available to download at:

[https://mydbs-my.sharepoint.com/:u:/g/personal/10520930\\_mydbs\\_ie/Efv1UtOh40JGvH\\_akOkM8dcB18aCqwoxvsj0\\_bWigDLvoQ?e=DC6c5c](https://mydbs-my.sharepoint.com/:u:/g/personal/10520930_mydbs_ie/Efv1UtOh40JGvH_akOkM8dcB18aCqwoxvsj0_bWigDLvoQ?e=DC6c5c)

The entire artefact is available at:

[https://mydbs-my.sharepoint.com/:f:/g/personal/10520930\\_mydbs\\_ie/En4Ld1k1w8VDus2sd2eR3E8BjDrchYNVVrmA2asQwtf8yw?e=FepKRu](https://mydbs-my.sharepoint.com/:f:/g/personal/10520930_mydbs_ie/En4Ld1k1w8VDus2sd2eR3E8BjDrchYNVVrmA2asQwtf8yw?e=FepKRu)





