UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Department of Electrical Engineering

Machine Learning for Radio Frequency Interference Flagging

Prepared by: **Kyle Harrison**
Prepared for: **Dr. Amit Mishra**
Dr. Russ Taylor

February 22, 2021

Submitted to the Department of Electrical Engineering at the University of Cape Town in part fulfilment of the academic requirements for a Master of Science degree in Electrical Engineering.

Declaration

I declare that this thesis is my own work. It is being submitted to the Department of Electrical Engineering, University of Cape Town, in fulfilment of the requirements for the degree of Master of Science in Electrical Engineering. It has not been submitted before for any degree or examination in any other university.

- I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.
- This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Names: Kyle Harrison

Date: February 22, 2021

Abstract

The field of radio frequency interference (RFI) flagging involves the identification of corrupted data within radio astronomy measurements. This work explores the application of supervised machine learning algorithms for RFI flagging, trained on real measurement data and simulated data with simulated RFI.

The goal of this work is to investigate the prediction of RFI using specific machine learning algorithms; Naive Bayes Classifier, K-Nearest Neighbours Classifier, Random Forest Classifier, the U-Net convolution neural network and the Multilayer Perceptron.

These algorithms are trained on real data, in which the ground truth includes inherent false positives, and simulated data where the ground truth positions of RFI are absolute. This is done through the use of time/frequency spectrogram data, relating to radio astronomy measurements, using the magnitudes and phases of each available polarization. Predictions for unseen test data are compared between algorithms, different implementations of those algorithms and each dataset.

A specific implementation for data pre-processing is designed and implemented, utilizing a two dimensional filtering technique for feature construction. The goal of this method is intended to implement a means of injecting a form of spatial information of nearby time/frequency samples for each sample in a spectrogram. The inclusion of this spacial information, which is relevant to broadband bursts and narrowband persistent RFI, is hypothesised to increase the level of information present in the processed dataset.

The use of feature construction using filtering techniques, demonstrates a noticeable improvement in the machine learning methods where each sample is treated individually during training and inference.

Acknowledgement

This thesis titled "*Machine Learning for Radio Frequency Interference Flagging*" was proposed by the author Kyle Harrison and supervisor Dr. Amit Mishra, in partnership with The Inter-university Institute for Data Intensive Astronomy (IDIA).

I would like to thank Prof Mishra for your encouragement of my ideas and help throughout my time at UCT. As well as Prof Taylor and IDIA for their fellowship, allowing me to pursue my own research using their cloud compute platform.

I owe a great deal of gratitude to Misha Mesarcik and Srikrishna Sekhar for their insights and help into RFI flagging. Our conversations gave me the confidence to keep going.

I would like to thank the endless support of my parents and Pia, without your encouragement I would never have been able to complete this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter briefly introduces the topic of this dissertation. It gives a summary into radio astronomy and the impact of radio frequency interference (RFI). The impact RFI can have on measurements and how it is dealt with. It goes on to outline the goals and scope of this research into machine learning for the detection of RFI and how results will be validated.

1.1 Opening remarks

Since the first detection of radio waves from celestial objects by Karl Jansky in 1932 the field of radio astronomy as increased exponentially. Owing to our atmospheres opacity to the radio spectrum, the study of radio emissions has lead to discoveries of entirely new classes of celestial objects such as quasars, pulsars and masers. Recently much work has gone into investigating the 21cm line visible in the radio spectrum. These studies aim to better explore the Epoch of Reionization (EoR), the period of time in which the creation of celestial objects such as stars and galaxies began.

The birth of modern radio telescopes and interferometry have allowed a plethora of organisations around the world to observe the universe at a higher resolution than optical astronomy.

One of the major difficulties shared between all of these organisations is the impact of man-made radio transmitters which overlap with the observed band-

width, referred to as radio frequency interference (RFI).

1.1.1 Background to the study

The strength of RFI is orders of magnitude more powerful than the faint celestial signals. If processed with the astronomical signals they are captured with, the resulting useful information will be negligible. This means the removal of RFI is of utmost importance. To prevent corruption of the observation data it is preferable to remove portions of the signals of interest to ensure all interference has been identified.

Many observatories take extensive steps to ensure their telescopes are located in isolated, radio quiet zones. Attempts are made to limit and control sources of interference such as electronic devices, communication and power transmission networks. Yet RFI is present everywhere, from passing air planes to satellite signals.

While RFI mitigation is applied at the data capture and pre-processing stages with techniques such as shielding and filtering. Inevitably the implementation of data post-processing for RFI removal is required.

With the scale of modern interferometers the amount of data to be processed is often in the scale of terabytes. These techniques are required to be robust, handling data which may vary in appearance as well as magnitude. Importantly the algorithms which deal with RFI identification must be able to do so without an extremely long time delay, as removal of RFI is one step in the pipeline to creating astronomical science images.

Post-processing of RFI can come in two distinct forms, flagging or excision. Excision is the process of subtracting the RFI signal from the data, retaining the underlying celestial information.

Flagging is the process of generating a Boolean map from time/frequency samples from each telescopes baseline which identify RFI vs non-RFI time/frequency samples. These flags are used in the imaging process in order to generate an image of the celestial radio source of emission.

1.1.2 Objectives of this work

The primary objectives of this research is to create and implement machine learning algorithms for the purpose of RFI flagging.

This is performed through the use of both real observation data and existing flagging techniques as well as simulated HERA data with simulated RFI. HERA (Hydrogen Epoch of Reionization Array) is a radio telescope in South Africa which functions as a precursor to the SKA (Square Kilometer Array). These two datasets are used training and testing data for multiple algorithms.

Comparisons between the performance of multiple machine learning and deep learning techniques are drawn. Experiments into optimisations of the models and improvements in pre-processing of the datasets are to be performed.

For the real dataset, predicted flags and then processed with the original data to generate a science image. This process of deconvolution and the resulting image metrics act as a separate measure of accuracy between the predictive algorithms and the existing flagging techniques.

Conclusions are drawn as to the effectiveness of the various algorithms, optimization steps and pre-processing between both datasets.

1.1.3 Motivations of this work

There exist a multitude of different RFI flagging algorithms used by each radio astronomy organisation. Each have their own unique advantages and disadvantages. The topic of machine learning is relatively recent and there is a distinct lack of research into its application for RFI flagging in this context.

The ability for a model to learn the implicit, underlying mechanics of one or more existing flagging techniques which is able to predict RFI in a fully autonomous fashion - would be a vital component in an image processing pipeline and for the theoretical understanding of RFI detection.

1.1.4 Scope and Limitations

The scope of data utilized in this work is restricted to a single real observation and a simulated HERA dataset. This work details post-processing techniques utilizing high latency, high complexity algorithms and is not intended for real-time applications.

The real dataset has been flagged with a combination of the AOFlagger algorithm and manual flagging techniques. The simulated dataset will have inherent limitations as to its uncertainty when compared to real data, likewise the simulated RFI will possess internal structure slightly different to that of the real dataset. Comparisons drawn between models trained and tested on each dataset is limited through this as a consideration.

The scope of learning algorithms explored in this work are the following.

Machine learning classifiers:

- Naive Bayes Classifier
- K-Nearest Neighbours
- Random Forest Classifier

The deep learning classifiers:

- Multilayer Perceptron
- U-Net convolutional neural network

1.1.5 Plan of development

This work begins with a background into the theory behind radio astronomy, its measurements and processing. A description into the relevant literature involving machine learning for RFI flagging is given. It goes on to describe the necessary techniques and algorithms involved in the machine learning algorithms utilised in this work. These algorithms are recreated from the parameters given in relevant literature and these algorithms are then optimised for the available datasets.

An approach to pre-processing through feature extraction is proposed and implemented. This processed data is used on the machine learning classifiers

and comparisons are drawn. Two deep learning approaches are designed and implemented, the first being the U-Net convolutional neural network and the second a multilayer perceptron network.

Conclusions are drawn into the reasons for differences between each algorithms performance. Finally recommendations are to be made as to extensions of this work.

1.1.5.1 Block diagram overview

A novel technique for pre-processing in the form of feature extraction is proposed in this work. This technique utilises a two dimensional sliding window, or filter, which performs statistical measurements over a subsection of samples. These features are used as additional information for the machine learning algorithms.

A block diagram for the pre-processing involved with the real dataset is shown in Figure 1.1. This diagram depicts the extraction of the four polarizations and flag for each baseline measurement. The time/frequency spectrogram data is pre-processed with the two dimensional filter procedure, normalized and flattened into a one dimensional array for each feature. This data is then stored in an HDF5 file which is used for long term storage and accessed by the machine learning algorithms.

The algorithms which utilize this data will treat each sample individually. Therefore the data shape $frequency \times time$ number of samples with N number of features for each filter extracted.

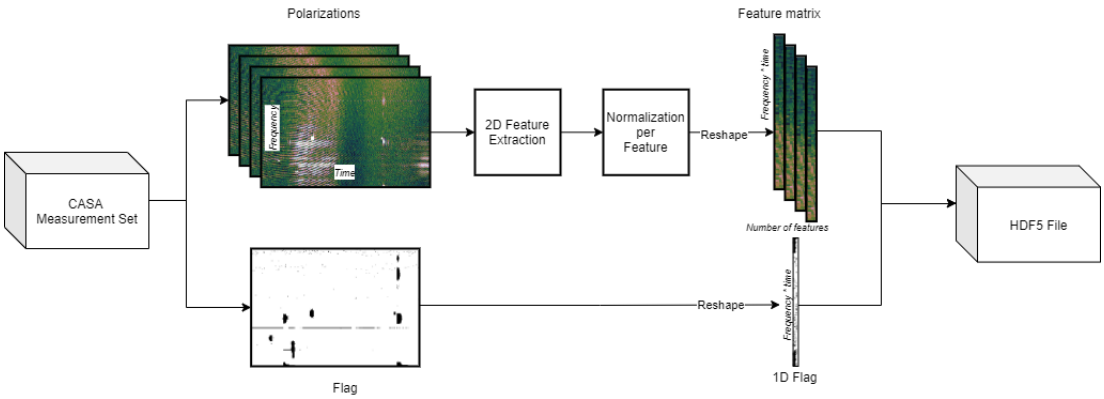


Figure 1.1: A block diagram overview of the data preprocessing pipeline. It shows the transformation from CASA measurement set files into HDF5 files through data reshaping and feature extraction.

Chapter 2

Theory

This chapter serves as a basis for understanding the fundamentals of radio astronomy as well as machine learning. It begins with a background into radio astronomy as a discipline and the theory behind celestial radiation. It goes on to describe modern interferometry and the process for converting measured antenna voltages into science images through convolution.

This chapter gives a foundation for machine learning. It covers the basis for processing data as well as methods to ensure algorithms are trained to their best ability. This chapter covers the theory behind the algorithms used in this work and how to ensure their optimisation.

2.1 Radio Astronomy

This section describes the concepts and fundamental basis for radio astronomy. It is necessary to give further detail into the data and methods to process it used in later implementations. It begins with a background into astronomy and the reception of celestial radiation. The fundamentals behind the visibility function and (u, v) plane are explored in depth. It finishes with an explanation of imaging, the process of transforming antenna measurements in a science image of radio sources.

2.1.1 Brief History of Radio Astronomy

The discovery of celestial radio emissions occurred serendipitously in the early 1930's by the American physicist Karl Janksy. Working with Bell Labs, Jansky was investigating unexplained interference in their transatlantic communication service. He observed radio emissions from active thunderstorms and an unknown source which repeated at the rate of the sidereal day.[1]

He soon found the source of the signals was emanating from the densest part of the Milky Way in the constellation Sagittarius. This would later be found to be Sagittarius A, one of the brightest radio sources in the sky.[2]

While the great depression prevented further research by Jansky, his pioneering in the field lead to the naming of the fundamental unit of flux density, the Janksy (Jy).

2.1.2 Celestial Radio Emissions

There are two main sources of continuum emissions in radio astronomy, thermal and non-thermal radiation. Thermal emissions are described by Planck's radiation law, describing how all objects with a temperature above absolute zero will radiate electromagnetic waves.

$$B_\nu(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{\frac{h\nu}{kT}} - 1} \tag{2.1}$$

where:

B_ν = source brightness

ν = electromagnetic frequency

T = temperature in Kelvin

h = Planck's constant

c = speed of light

k = Boltzmann's constant

From Equation 2.1 it can be inferred the brightness of a source is directly proportional to the temperature of the source. This is known as Wien's displacement law: $V_{max} \approx 5.789T$.

In radio astronomy with low frequencies it is more typical to reduce the Planck formula to the Rayleigh-Jeans approximation where $h\nu << kT$:

$$B_\nu(T) = \frac{2\nu^2}{c^2}kT \tag{2.2}$$

One of the more common forms of radiation found in radio astronomy is Synchrotron radiation. This is the process of a charged particle emitting radiation when subject to acceleration perpendicular to their velocity. In this case the acceleration is due to ambient magnetic fields. This causes continuum radiation over the entire spectrum and has a characteristic polarization.

This can be shown as:

$$P = \gamma^2 \frac{2}{3}\frac{q^4}{c^3}\frac{\nu^2 B^2 sin\alpha^2}{m^2c^2} \tag{2.3}$$

where:

P = radiated power

q = charge

v = velocity

α = angle between magnetic field and velocity

m = mass of the charged particle

It can be seen in Equation 2.1.2 that the total power radiated by the particle depends on the magnetic field strength, and that the higher the energy of the particle the higher power it will radiate.

2.1.3 Polarization

The state of polarization for incoming celestial radiation plays an important role in radio astronomy. Specific objects like pulsars can be almost completely polarized, allowing for analysis of the geometry of magnetic fields from the source. Other cosmic radio sources can be fully or partially polarized but are not monochromatic.

Describing the polarized state of radiation is done through the polarization ellipse in Equation 2.4 for E_x and E_y the vertical and horizontal components of

the electromagnetic wave represented as two orthogonal linearly polarized waves. [3]

$$E(z,t) = (\vec{x}E_x e^{i\phi 1} + \vec{y}E_y e^{i\phi 2})e^{i2\pi(vt-kz)} \tag{2.4}$$

Therefore the polarization of the electromagnetic wave is dependent on the relative amplitudes and phase of the orthogonal waves. Considering the case where each component has the same magnitude and their phases are separated by $\pi/2$, the resulting vector traces a circle - representing circular polarization.

Representing the polarization state of electromagnetic waves is simplified through the use of Stokes Parameters shown in Equation 2.5, used to evaluate direction and type of polarization.[4]

$$\begin{pmatrix} I \\ Q \\ U \\ V \end{pmatrix} = \begin{pmatrix} E_x^2 + E_y^2 \\ E_x^2 - E_y^2 \\ 2E_x E_y cos(\delta) \\ 2E_x E_y sin(\delta) \end{pmatrix} \tag{2.5}$$

2.1.4 Receiving Electromagnetic Radiation

To model a radio telescope it is assumed to have the ideal parameters of a plane receiving electromagnetic radiation emitted by a sphere. The total received power is dependent on the power and frequency of the transmission, and the area of the receiving plane. This received power can be expressed as in Equation 2.6 by an abstraction to an infinitesimal point on the plane:

$$dW = B_\nu B cos(\theta) d\Omega dv dA \tag{2.6}$$

where:

dW = infinitesimal power received by the plane

B_ν = brightness function at position dΩ

θ = solid angle from the vertical reference to the sphere

$d\Omega$ = infinitesimal solid angle of the source of the emission

dv = infinitesimal bandwidth of the emissions spectrum

dA = infinitesimal area of the plane

From this definition it is possible to express the received power of the entire plane for a solid angle. This is shown in Equation 2.7 by integrating over the area of the receiving plane and the relative bandwidth.[5]

$$W = A \int_{v}^{v+\delta v} \int_{\Omega} B cos(\theta) d\Omega dv \tag{2.7}$$

2.1.5 Radio Telescopes

Radio telescopes are can be thought of as highly directional antennas with the small beam width relative to the source radiation. Modelling an ideal antenna would be isotropic, giving equal gain to signals from all directions. In reality the beam of a parabolic reflector used in radio astronomy is shown in Figure 2.1. They possess a large gain in its main direction with multiple smaller side lobes radiating in other directions, including a back lobe. While side lobes have a lower gain than the main lobe and are attenuated, they are often the source of significant interference due to the high amplitudes of RFI.[6]

In order to express the power received by an antenna in terms of the antenna radiation pattern, it is taken into account that the sky brightness distribution depends on the direction of the antennas main lobe. The Equation 2.7 is adjusted to reflect this in 2.8 using spherical coordinates. The factor of 1/2 owes to the fact that an antenna is singularly polarized, receiving half the relative power of the fully polarized signal.

$$W = A_e \frac{1}{2} \int \int_{\Omega} B_\nu cos(\theta, \rho) P_n(\theta, \rho) d\Omega dv \tag{2.8}$$

where:

A_e = the effective area of the antenna

$P_n(\theta, \rho)$ = the antenna response for radiation pattern over angles θ and ρ

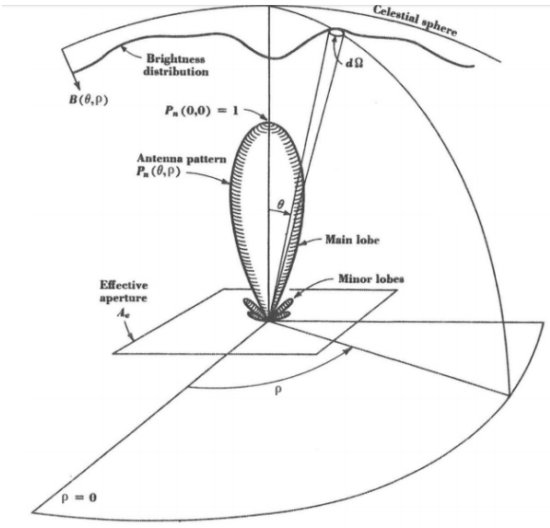


Figure 2.1: Example of an antennas radiation pattern showing main lobe and side lobes. The rotation in spherical coordinates is shown along with the infinitesimal solid angle of the source.[5]

2.1.6 Astronomical Interferometry

Interferometry in radio astronomy is the process of utilizing multiple telescopes separated by a distance, yet operating together. By using aperture synthesis their signals are combined into a single higher resolution image. The resolution of an antenna is defined by Equation 2.9, where λ is the observed wavelength and D is the diameter of the antenna.

$$\theta \approx \frac{\lambda}{D} \tag{2.9}$$

For a fixed wavelength this shows how the resolution of a radio telescope is directly dependent on the size of the antenna. The production of large receivers has been done with the Arecibo Observatory totalling $73000m^2$. Yet it entails high construction and maintenance costs.

The resolution of interferometric telescopes is dependent on the maximum baseline, B, the maximum distance between any who antennas. Resulting in the

Equation 2.10.

$$\theta \approx \frac{\lambda}{B} \tag{2.10}$$

Figure 2.2 shows a baseline B formed by two receivers. The sources of incoming radiation is shown as a plane wave when modelling the source as infinitely far away. As the phase measured between receivers differs by their distance apart. The geometrical time delay $T_g = \vec{B}\vec{s}/c$ determines the relative phase between the received signals $\phi = 2\pi\nu T_g$.

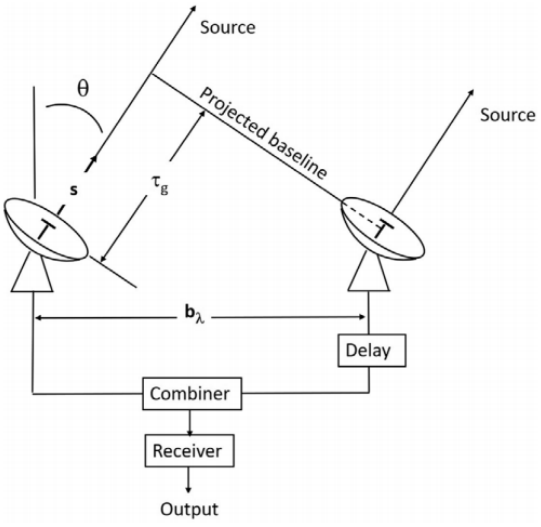


Figure 2.2: Depiction of the baseline formed by two receivers and their complex correlation adjusted for the geometric time delay.[3]

The receivers signals are aligned using the time delay T_i and are processed through a complex correlator, the output of which is a cross-spectral density by integrating over the sky brightness distribution.

$$S_{xy}(v_0, \sigma) = \int_{4\pi} A(\sigma) B_\nu(\sigma) e^{-j2\pi v_0 (T_g - T_i d\Omega} \tag{2.11}$$

where:

S_{xy} = cross-spectral density

v_0 = the specific frequency

σ = the vector from the phase centre to the source

$A(\sigma)$ = relative antenna area

$B_{v0}(\sigma)$ = sky brightness

T_g = geometric time delay

T_i = induced delay

Expressing the baseline in wavelengths, b_λ, letting the centre frequency be v and writing the geometric time delay in terms of the direction vectors yields Equation 2.12. This function applies to the *complex visibility* and is related to the Fourier transform of the brightness distribution $B_v(\phi)$.[3]

$$V(b_\lambda) = \int_{4\pi} A(\sigma)B_v(\sigma)e^{-j2\pi b_\lambda d\Omega}) \tag{2.12}$$

2.1.7 Visibility

In order to practically implement Equation 2.12 it is necessary to transform the coordinate system for the baseline vector b_λ and the offset vector σ. This is done through the utilising the observing uv plane as a source from the direction of the North Celestial Pole, and the sky plane represented by lm shown in Figure 2.3.

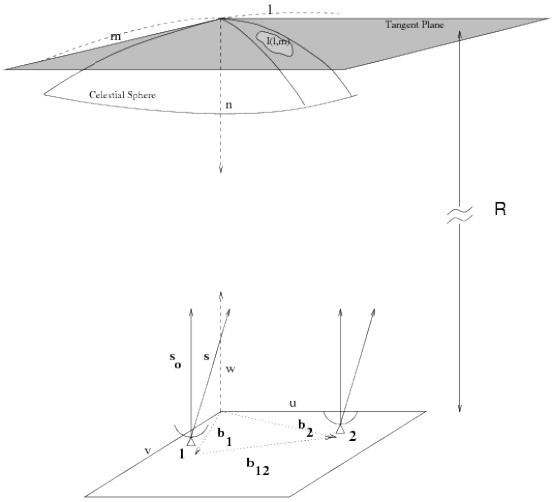


Figure 2.3: An image showing the relationship between the coordinate systems (u, v, w) and (l, m, n) with reference to antenna and source positions.[7]

Given that the offset vector $\sigma = s - s_0$, it can be expressed that:

$$\mathbf{b}_\lambda \cdot \boldsymbol{\sigma} = ul + vm + (n-1)w$$
$$d\Omega = \frac{dldm}{n} = \frac{dldm}{\sqrt{1-l^2-m^2}}$$

Which allows for the 2-D Fourier transform relationship to be expressed with (u, v) being Fourier pairs of (l, m).

$$\frac{A(l,m)B_\nu(l,m)}{\sqrt{1-l^2-m^2}} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} V(u,v)e^{+\imath 2\pi(ul+vm)}dudv \tag{2.13}$$

2.1.7.1 van Cittert–Zernike theorem

With a restricted field of view $A(l,m)/\sqrt{1-l^2-m^2} \to 1$. This leads to the formalization of the van Cittert–Zernike theorem which shows the response of an ideal baseline pq given by $V(u,v)$, the visibility function of pq, is the 2-D Fourier transform of the sky brightness for a small angle. This is also a representation of a sampling of the mutual coherence function between signals.

$$V_{pq}(u,v,0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} B_\nu e^{-2\imath\pi(ul+vm)}dldm \tag{2.14}$$

As the main beam of each antenna is too narrow to resolve individual components in the sky, each antenna is receiving a combination of all source signals. The van Cittert–Zernike theorem allows for processing received signals into a sky image. To this extent it states that within certain parameters the mutual coherence of incoherent, monochromatic sources are equal to the 2D Fourier transform of the complex visibility. As the wavefront of the source radiation emanates as far-field radiation, the sources appear coherent. The cross-correlation between antennas of the baseline can be represented as a sample of the mutual coherence function in the uv plane.[8]

2.1.8 UV Plane Coverage

An interferometer is accumulating measurements of the sky Fourier Transform by sampling the visibility function. Each sample from the projected baselines pro-

duce a UV track as the earth rotates, causing the corresponding spatial frequency coordinates (u, v) to vary with time.

Ideally in order to achieve a precise knowledge of the sky, the best possible visibility function is required. In order to accomplish this a complete sampling of the uv plane is required.

In reality the tracks formed by projected baselines as they change with time form portions of ellipses which can be derived from the expression of the projected baselines in the (u, v, w) reference frame. The samples of these uv tracks are referred to as uv coverage. An example of these tracks is shown in the first image of Figure 2.4.

These sampled positions give a sampling function $S(u, v)$. Which results in a discretized image owing to the discretized grid of visibilties, demonstrated in Equation 2.16 is a generalised summation of the sampled (u, v) plane measurements. This is a simplification, in reality the Fast Fourier Transform (FFT) requires the u and v coordinates to lie on a grid. To accomplish this a gridding kernel $S(u, v)$ is convolved before sampling.

$$S(u, v) * \sum_{k=1}^{M} \delta(u - u_k, v - v_k) \tag{2.15}$$

$$B_\nu(l, m) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} S(uv) V(u, v) e^{-2\imath\pi(ul+vm)} du dv \tag{2.16}$$

The Fourier transform of the sampling function is the response to a point source. This response forms an incomplete representation of the (u, v) plane and is referred to as the PSF (Point Spread Function) or dirty beam, $D(l, m)$. Using the convolution theorem leads to the dirty image of the true sky. The entire process is described visually in Figure 2.4.

$$B(l, m) * D(l, m) = B^d(l, m) \tag{2.17}$$

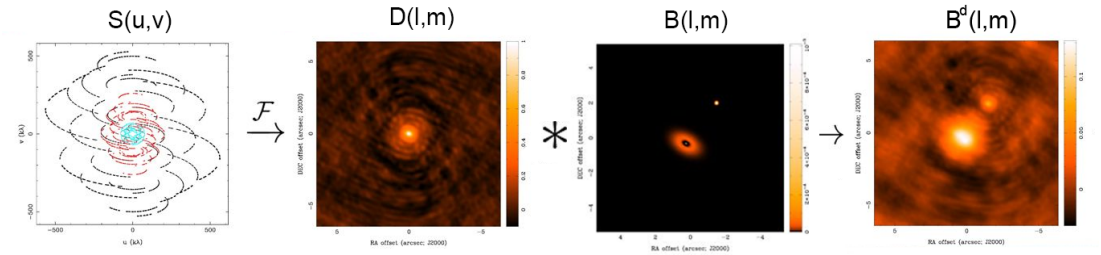


Figure 2.4: A demonstration of the process going from the sampled (u, v) plane which is Fourier transformed to the dirty image, convolved with the sky brightness function producing a dirty image.[9]

2.1.9 Deconvolution

The PSF has the effect of limiting the resolution of the true sky by acting as a type of low-pass spatial filter. The incomplete representation of the (u, v) plane causes detrimental distortions when the Fourier transform is applied, which results in the synthesised image. The maximum baseline determines the angular resolution while the shortest baseline limits the angular scale of lower brightness sources which are able to be imaged.

The incomplete representation of the (u, v) plane leads to 'holes' between baselines which produces sidelobes in the synthesised image. This prevents the dirty image from providing information on possible bright compact regions.

2.1.9.1 CLEAN Algorithm

Overcoming this effect was due in part to Jan Högbom with the development of the CLEAN algorithm. Known as 'deconvolution' it seeks to removed the negative impacts of convolution on the true sky brightness with the point spread function.

The process of deconvolution is to find the sources within the dirty image and subtract the PSF with a certain gain, to account for different flux values. This position and gain or flux multiplier are then added to a sky model. This prevent leakages from the noise present, flux due to the PSF or separate sources. The flux is not subtracted all at once, the process is done iteratively until a noise threshold is reached.

This iterative process is described below:

1. Create a residual image $I^R(l, m)$ by copying the dirty image $I^D(l, m)$.
2. Find the maximum value and position of a pixel in the residual image $I^R(l, m)$.
3. Multiply the PSF by a gain g and the pixel's value and subtract this from its position in the residual image.
4. Store the position and magnitude of the operation in (3.)
5. Iterate to (2.), unless all remaining pixel values are below a defined threshold or the number of iterations have reached some defined limit.
6. Convolve the accumulated point source sky model with a CLEAN beam created using a Gaussian fitted to the dirty beam.
7. To the CLEAN image in (6.) add the residual image to form a deconvolved image.

The CLEAN algorithm results in a de-noised, smoother image. As the positions of each source is identified the bright sources appear far more cleanly.

2.2 Radio Frequency Interference

The rate of expansion of radio astronomy is due in part to the opacity of Earth's atmosphere to radio wavelengths from around $30MHz - 300GHz$. Optical astronomy has many inherent limitations because of the atmospheric effects on μ wavelengths. While this serves as a major advantage for radio astronomy, in recent years the increased sensitivity and bandwidth of radio telescopes have given a major rise in the spectrum overlap between measurements of celestial radiation and man-made radio communication.

The nature of radio frequency interference (RFI) is inherently unpredictable. While known persistent sources exist, such as cellphone, airplane and satellite

communication signals, a plethora of RF emitting devices can cause interference. A major culprit in observatories is the presence of maintenance equipment, switch power relays and power tools cause high powered RFI bursts when activated. Less obvious sources exist, it was recorded at the Square Kilometer Array (SKA) site that nylon chairs and portable gazebos accumulating static charge and discharge were producing RFI.[10]

The presence of generated radio emissions unrelated to a system is referred to as RFI. In the context of radio astronomy the measurements of RFI, often observed in attenuated side-lobes, is several orders of magnitude more powerful than the indistinct celestial radiation. The presence of RFI not only corrupts usable data if processed but can saturate and overload sensitive receivers and cause calibration errors.

RFI in radio astronomy can generally be described in four distinct forms:

- Narrowband persistent
- Narrowband burst
- Broadband burst
- Single time/frequency blips

Narrowband occurs over a limited, discrete number of frequency bins while broadband occurs across multiple channels. Narrowband is often the most common and constitutes as the strongest sources in radio astronomy. Persistent and burst refer to the amount of time for which the RFI event occurs. Broadband bursts are rare, and can be caused by events like lightning. Blips are random, sporadic events which occur for a shorter period than the integration time, leading to hot-spots on specific time/frequency samples.[11]

One of the biggest factors in limiting exposure to RFI is maintaining a radio-quiet zone around the observatory. Huge steps in projects such as the SKA have been taken to remove all possible radio transmitters or sources of RFI within and around the site.

2.2.1 RFI Detection Techniques

The impact of RFI is not an issue specific to radio astronomy but falls into many radar domains. The work of the European Space Agency (ESA) into Soil Moisture and Ocean Salinity (SMOS) satellite measurements utilizing L-band passive radar, requires active steps to detect and mitigate RFI in a similar manner to that in radio astronomy. There are five broad categories of RFI detection algorithms which are applicable for SMOS and overlap with radio astronomy:

- Spectral
  - Incoming signals are divided into separate spectral bins and RFI is detected and mitigated with analog sub-band filters.
  - Showed competent detection of strong narrow-band continuous wave RFI.
  - Struggled to identify low-level noise.
- Temporal
  - Divides the signal into over-sampled temporal bins.
  - Easily detects pulsed RFI which would be shorter than the integration time.
  - Struggled to identify low-level RFI and requires high data bandwidth.
- Statistical
  - Statistical techniques utilize the Gaussian nature of the incoming signals compared to the non-Gaussian distribution of RFI. Primarily uses kurtosis identifying deviations of samples in the higher-order moments of the incoming signal.
  - Shown to identify low-level RFI which is undetected by other algorithms.
  - Struggles to identify pulse-sinusoidal RFI, a 'blind-spot' for 50/50 duty-cycle signals.
- Spatial

  - Operating in the spatial domain, an inverse FFT of the visibility measurement, this method functions as a thresholding method detecting pixels which deviate from the mean of a moving window.
  - Geophysical signals have inherent variability which causes inherent prediction errors within the averaging window.

- Stokes
  - Utilizes the 3rd and 4th Stokes parameters for anomalous behaviour.
  - Improved detection for RFI sources with polarized signatures.

These five categories have been used for developing independent techniques, or multiple categories are combined in a single detection algorithm in an attempt at making it more robust.[12]

2.2.2 RFI Mitigation in Radio Astronomy

Mitigation of RFI refers to the process of detecting and removing RFI from data. Primarily these can occur at two separate stages, pre-correlation and post-correlation. Pre-correlation refers to real-time or on-line algorithms which maintain a relatively small perspective of the overall receiver system with their scope limited to a single telescope or small time range. They must be low-complexity methods as the data throughput is always at a high rate due to the incoming data being at its highest time resolution. Conversely post-correlation methods occur offline on stored data which has been correlated between interferometer pairs, often already time averaged and/or calibrated.

There is a distinction to be made between the two methods of mitigation; flagging and excision. Excision of RFI involves identifying the signal of interest and subtracting it, recovering the underlying signal and retaining the uncorrupted information. This can be unpredictable and its validity can be difficult to measure, as no truth behind the underlying signal exists. For this reason flagging is more common.

Flagging of radio frequency interference is the process of generating a Boolean map for each time/frequency sample indicating whether it is usable data vs cor-

rupted data. The principles surrounding its implementation differ between each radio observatory and often between observations. Current flagging techniques will be explored in depth in Sections 3.1 and 3.2. Flagging in the first process in the imaging pipeline. It is not only used to remove RFI but errors associated with telescopes off source and equipment malfunctions. The theory behind its uses and techniques is rarely covered in general radio astronomy text books as it is often a telescope and observatory specific task.

Most imaging pipelines involve data averaging by binning nearby samples in the time/frequency domain, a similar process is performed during uv gridding during imaging. Samples which are flagged are omitted from these bins, causing a deviation from the mean of the sky visibility function. To account for this the flagged samples are essentially interpolated in order to match the mean of other samples in the bin. The resulting error is then negligible, but still accounts for spectral fluctuations.[13]

With the increasingly large volumes of data being created by modern interferometers it is common for large amounts of a data to be removed with flags. The text book "*An Introduction to Radio Astronomy*" states: '*using no data is better than using bad data*'.[3] Which is testament to the importance of removing any possible errors and corrupt data before the imaging pipeline.

2.3 Machine Learning

Machine learning is the study of statistical models and algorithms related to performing a specific task without using explicit instructions. The field uses mathematical models to make inferences or recognise patterns in order to transform input information into relevant predictions. This section details the fundamental theory behind how specific machine learning classifiers, neural networks operate as well as their extension to convolutional neural networks. It describes how to evaluate and optimize the performance behind machine learning algorithms.

The field of machine learning can be broadly divided into three subsections; supervised, unsupervised and reinforcement learning.

2.3.0.1 Supervised learning

Supervised learning is a paradigm of machine learning algorithms in which a given set of data $\mathbb{D} = \{(\vec{x}_n, \vec{y}_i)\}$ where $x_n \in \mathbb{R}^2$, $y_i \in \{-1, 1\}$ they are able to implicitly learn the relationship between $\vec{x}^n$ the input vectors and $\vec{y}^i$ the output vectors. Supervised learning can be further divided into *Regression* and *Classification* for continuous or discrete data respectively.

After training a model on a subset of $(\vec{x}_n, \vec{y}_i)$ it can be used to process the unseen subset of the data, generating predictions of $\vec{y}_i$ for the new data. A *loss function* is generated between the output predictions compared to the ground truth of each sample. Training of the model is done through minimising the loss function by adjusting the model parameters in order to converge to the most accuracy classifier.

2.3.0.2 Unsupervised learning

Algorithms in unsupervised learning seek to infer the output vectors $\vec{y}_i$ given only the input $\vec{x}_n$. They do so by finding implicit, underlying structures and relationships inside the data. Making predictions by identifying relevant similarities and differences between input samples.

2.3.0.3 Reinforcement learning

Reinforcement learning is a technique where the learner acts as an independent agent, making decisions and progressing to different states in order to maximise a reward.

2.3.1 Feature Engineering

Features are one of the most essential components of data in machine learning. They express the inherent relationships held within the data. The aim of pre-processing data is to optimise the representation of a datasets features in order for the learner to achieve the highest predictive capabilities.

Data is presented in a machine learning paradigm as an input matrix X containing m samples as row vectors $\vec{x}_n$ where each element of the vector $a_{m,n}$, or the columns of the matrix n represents a feature.

$$X_{m,n} = \begin{pmatrix} a_1^1 & a_1^2 & \cdots & a_1^n \\ a_2^1 & a_2^2 & \cdots & a_2^n \\ \vdots & \vdots & \ddots & \vdots \\ a_m^1 & a_m^2 & \cdots & a_m^n \end{pmatrix} \tag{2.18}$$

The output or target vector y_i corresponding to each input vector represents the class to which the sample belongs. In a binary classification scheme it would simply be $y_i \in \{0, 1\}$. For multi-class environments labels are one-hot encoded also known as the $1 - of - K$ scheme. Where the variable is represented in a vector of length K in which a single element $y_k = 1$ else 0.[14]

A feature in this context is described as a measurable property or characteristic which represents an identifiable, independent aspect of all samples. In this way features express the relationships between samples in the data. A machine learning algorithm is able to predict classes by identifying the similarities between the features of samples belonging to the same class. Subsequently the algorithm will inherently learn the differences between the same features from samples belonging to different classes.

Feature engineering refers to the process of generating features using *domain specific knowledge*.

2.3.1.1 Feature Extraction

The techniques involved around deriving new features from existing data is known as feature extraction. Where often explicit measurements and characteristics relating to the sample are unknown but can be calculated through transformations and manipulations of the current data. In order to derive new, useful, information the features identified must be non-redundant in order to represent a gain in information allowing better interpretation of the data.

2.3.1.2 Feature Selection

Feature selection refers to the process of identifying and isolating a subset of the given feature space which may be represent the underlying structure of the data. As feature extraction is intended to improve the predictive accuracy and performance of a learner, feature selection often employs *dimensionality reduction* techniques to select the optimum features in a trade-off between accuracy and complexity.

2.3.2 Overfitting and Underfitting

A fit describes the learners ability to generate a discriminator, which defines a boundary or regions in the output space to identify class membership. The goal of a learning algorithm is to generate predictions which best represent the inherent relationships within the data. If a learner is too strict in its approach it will generate predictions which closely match its training data and perform poorly on unseen data, this represents overfitting. If not strict enough a learner will fail to capture the underlying trends of its training data. Performing poorly on both datasets.

The images in Figure 2.5 represent examples of the types of fits possible. Underfitting is shown in the first frame. It shows how the discriminating line has not identified the inherent relationship of the curve of its input data. This is often the result of a model which is too simplistic and results in a low variance but high bias, insensitive to changes in the data. Overfitting is where the learner has captured a degree of noise within the training data, reducing its ability to generalize and therefore reducing its ability to predict unseen data. This is more common in machine learning and results in a low bias but high variance from an overly complex model which would change drastically if data were varied.

Evaluating the fit of a model may be done through dividing data into training, test and validation sets. Where the test data is left out of the training procedure. This unseen test data has no impact on the models generation of predictions and by using it for predictions one can evaluate the true accuracy of a model.

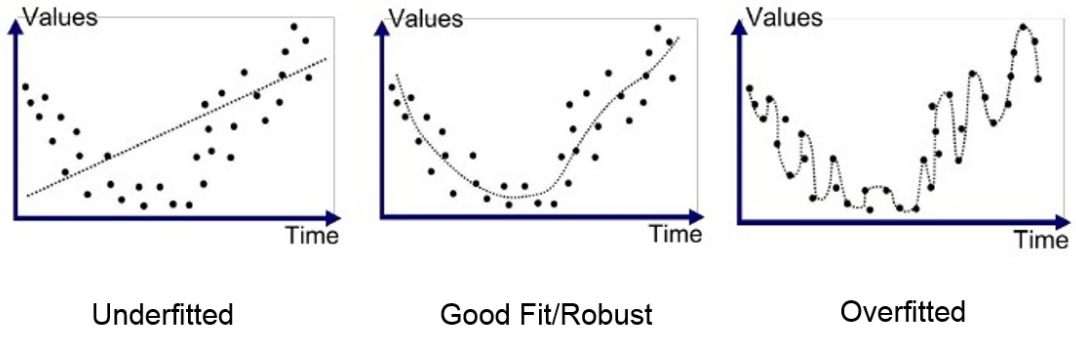


Figure 2.5: Depiction of the three types of fits to a 2D dataset.[15]

The validation set may be used when a learner employs optimisation based on preliminary results on data outside of its training information. This should not be utilised for assessing accuracy as there will be a slight relationship between the ability for the learner to distinguish this set more accurately than that of the truly unseen data.

2.3.3 Evaluation Metrics

Understanding the accuracy of a machine learning algorithms predictions must be more robust than simply measuring correct versus incorrect scores. While accuracy from this perspective may be useful in some instances, in a paradigm where the number of instances of each class is unbalanced it would prove unreliable. For example in unbalanced binary classes if a single class, 1 were to be represented in only 1% of the data and the learning algorithm predicts a zero for all data, it would be 99% accurate.

2.3.3.1 Confusion Matrices

A confusion matrix is a plot visualising the performance of predictions versus a known truth. Shown in Figure 2.6 is a binary confusion matrix shows the number of correct positive and negative predictions compared to their number of incorrect predictions respectively.

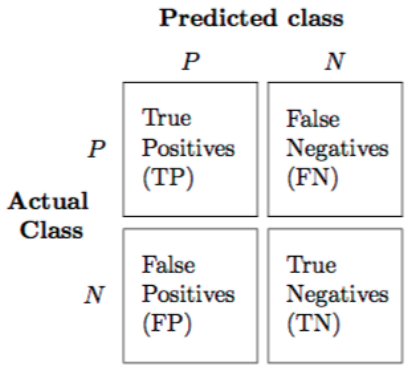


Figure 2.6: A confusion matrix plot, used to evaluate the performance of a classifier. [16]

2.3.4 Precision, Recall and F1-score

Further evaluation of a classifiers predictive accuracy is formulated with the use of Precision, Recall and F1-score in Equation 2.19 through **??**. These metrics take into account the positive and negative predictions in a binary class compared to whether they were true or false. Precision is known as the positive predictive value, and it represents the fraction of correct relevant predictions versus the total number of positives. Recall or sensitivity represents the fraction of correct positive predictions compared to the total number of relevant predictions. The F1-score is simply the harmonic mean of precision and recall, taking into account both metrics evenly weighted.

$$Precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{2.19}$$

$$Recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2.20}$$

$$F1score = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.21}$$

The predictions of a classifier are often a probabilistic determination of the confidence in a prediction. Where a threshold limit may be set to produce class labels it is practise to demonstrate a classifiers performance based on varying this threshold limit. This precision recall curve known as the Receiver Operating Characteristic (ROC) is shown in Figure 2.7, where a curve is generated by evaluating the precision and recall for this varying threshold. The area under the

curve (AUC) is the probability of a classifier predicting a positive over predicting a negative, or simply is ability to distinguish between classes.

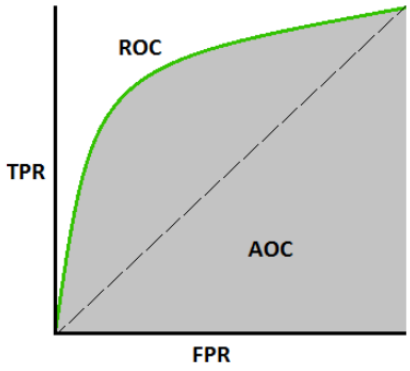


Figure 2.7: Figure depicting area under the curve (AUC) and the receiver operating characteristics (ROC) curve generated by varying the prediction threshold and evaluating the precision and recall.[17]

2.3.5 Decision Tree Learning

Decision trees are a tool used to represent a decision based on possible consequences in order to reach a given goal in a tree-like structure of nodes, branches and leaves. In machine learning, decision trees are a supervised predictive model which can be constructed using splits; sub-dividing the dataset's features in a manner which a path would best reach a leave node - representing a class label. This occurs through optimised "if-then-else" decisions relating to a samples features determining which class it belongs too.[18]

CART (Classification and Regression Trees) refers to the set of decision tree algorithms which are capable of using observations about the data, represented by branches, to reach conclusions about the target. Discrete values are processed in classification trees, while regression trees target values are continuous values.[19]

2.3.5.1 Classification Trees

A classification tree functions iteratively, dividing the input space into sub-populations with each step:

1. Create the root node

2. Split the data into two subsets
3. Set a decision for each subset
4. Move along the branches
5. Repeat (2) to (4) until a defined stopping criterion has been reached

The root node is chosen through the use of Entropy and Information gain or with the Gini Index. Information gain determines which features represent the maximum amount of information from a class. This is based on Shannon's Entropy which quantifies the degree of uncertainty or disorder. The amount of entropy is reduced from the root node to the leaf nodes. The formula for is shown in Equation 2.22 where the information entropy $H(T)$ for the relevant group is the negative logarithm of the probability mass function for the value. This probability p_i represents the percentage of each class from a child node resulting from a split.

The information gain for each possible split is calculated and the optimum first split is that which provides the best information gain, maintaining the highest purity for further nodes.[19]

$$H(T) = -\sum_{i=1}^{J} p_i log_2 p_i \tag{2.22}$$

The Gini impurity is another method which can be used as the cost function to evaluate optimum splits in the dataset. It represents the measure of the rate at which a randomly chosen sample would be incorrectly classified if randomly labelled based on the distribution of labels resulting from the labels in a split's subset. It is calculated by a sum of the probability p_i of an item of label i being chosen multiplied by the probability of it being incorrectly classified. The point at which it is minimised is where all cases of samples being classified by the node occur at a single target category.

$$I_G(p) = 1 - \sum_{i=1}^{J} p_i^2 \tag{2.23}$$

Each algorithm maintains the same goal, optimizing the decisions at each node which would maximise determining the class of each subset after the split.

A tree is terminated when it reaches a point at which each leaf node is an independent class. The tree may also be terminated during training by a user determined maximum depth or a minimum number of node records. Both user defined functions seek to reduce overfitting. The depth of a tree represents the inherent complexity it captures and predefining this limit can reduce the ability of a tree to generate training data specific splits further down.

2.3.5.2 Random Forests

A major disadvantage of conventional decision trees is their ability to overfit to training data. Random forests are an ensemble method which generates multiple trees with a degree of randomness and estimates a prediction from the average decision of all the trees generated. In this way the low bias, high variance in overfit trees is controlled by training multiple deep decision trees on separate subsets of the same training data.[20]

Bagging is the process of selecting random samples with replacement from the dataset. For the number of baggings B the method performs:

1. Sample and replace n training samples $\vec{x}_n, \vec{y}_i$ as $\vec{x}_b, \vec{y}_b$ for $b = \{1, ..., B\}$
2. Train tree f_b on $\vec{x}_b, \vec{y}_b$

Once B trees have been trained, predictions on test data x' are determined by the mode of all individual trees predictions.

Random forests further the bagging technique by performing 'feature bagging'. Selecting a random subset of features at each candidate split. The reduces the correlation of trees by selecting a strongly predictive feature.[20]

2.3.6 Neural Networks

Artificial Neural Networks (ANN) is the generalized name given to biologically inspired learning algorithms. They consist of a network of collected nodes called artificial neurons which are an abstraction, modelling biological neurons in a brain. Connections between neurons function similarly to synapses, receiving

and transmitting signals to and from neurons connected to it. The differences between a biological neuron and the artificial abstraction inspired by it, is shown in Figure 2.8 where the function of a basic neuron in an ANN is demonstrated.

The fundamental function of an ANN is a summation of inputs multiplied by an individual weight who's output is then manipulated by an activation function. This transformation from input to output represents a prediction, by comparing this to the target value an error function can be formulated. Minimizing this error by adjusting the weights represents training the network, as it learns a more accurate weight combination for a correct prediction.

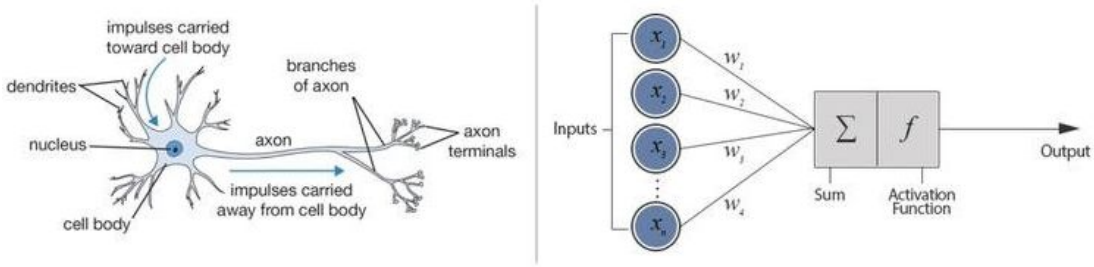


Figure 2.8: Figure demonstrating the differences between a biological neuron and an artificial neural network. The ANNs function is shown by inputs $\{x_1, ...x_n\}$ multiplied by the weights $\{w_1, ...w_n\}$. The output is the summation of this operation followed multiplication with an activation function. [21]

2.3.6.1 The Perceptron

The initial perceptron algorithm was devised as a supervised learning method inspired by the simplified model of a biological neuron. It maps a real-valued input vector $\vec{x}$ to an output value $f(x)$ as a single binary value.

$$f(x) = \begin{cases} 1 & \text{if } \vec{w}\vec{x} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

Where $\vec{w}$ is the weight vector and the input vector $\vec{x}$ for m is the number of inputs into the perceptron. In this case each element of the input vector represents the vectors feature space. The equation $w \cdot x > b$ defines a hyperplane dividing the feature space in two convex half spaces serving as the class decision

boundary. With b representing the bias shifting this boundary from the origin irrespective of input values.

This can be described as a linear discriminant model: $y(x) = f(w^T x_n)$ where f is the nonlinear activation function known as the Heavyside Step Function.

$$H(n) = \begin{cases} 1, \, n >= 0 \\ 0, \, n < 0 \end{cases} \tag{2.25}$$

Determining the optimum weights w is evaluated as a minimization of an error function known as the perception criterion. Which is defined as:

$$E_p(w) = -\sum_{n \in M} w^T x_n t_n \tag{2.26}$$

This assumes the target vector in the form $t \in \{-1, 1\}$, has all instances correctly classified as outputs with zero error associated with them. While incorrectly classified class labels of the quantity $-w^T x_n t_n$ can be attempted to be minimized over all classifications in the set M.

The drawback of the single perceptron model is the fact that it represents a linear classifier, if the input dataset is not linearly separable not all of the input vectors will be correctly classified according to the hyperplane decision boundary and it will not reach convergence. Along with this it does not provide probabilistic outputs, nor does it generalize with more than two classes.

2.3.6.2 The Multilayer Perceptron

The description of a multilayer perceptron (MLP) as a feed-forward neural network, is a model comprising of layers of multiple logistic regression models with continuous non-linearities. As opposed to perceptrons with discontinuous non-linearities owing to its use of the step function. These simplified linear models for regression and classification can be described as linear combinations of non-linear basis functions $\phi_j(x)$:

$$y(x, w) = f(\sum_{j=1}^{M} w_j \phi_j(x)) \tag{2.27}$$

Where the basis functions are themselves non-linear functions owing to the non-linear activation function $f(\cdot)$ and are able to be adjusted with the weights, as parameters w_j for $j = 1, ..., M$ number of neurons in the layer.

The linear combinations of the input features multiplied by each respective weight plus a bias: $a_j = \sum_{i=0}^{D} w_{ji} x_i$ for some sample feature vector x_i of D features are known as activations. The bias term can be contained in the weight parameters by defining $x_0 = 1$ and beginning the summation there.

This basic neural network is a series of functional transformations. Past the initial input layer the outputs of these basis functions are known as hidden units. The non-linear activation function $z_j = h(\cdot)$ is often chosen as a rectified linear unit (ReLU): $h(x) = x^+ = \max(0, x)$, but many possibilities and combinations exist depending on the application and target variables.

The second layer of the network takes inputs from the first layers activations z_j, where $j = 1, ..., M$ linear combinations, corresponding to the number of neurons constructed in the first layer and where $k = 1, ..., K$ is the total number of outputs.

$$a_k = \sum_{j=0}^{M} w_{kj} z_j \tag{2.28}$$

This entire process is shown in Figure 2.9, with the arrows showing the forward path of inputs to outputs. For a regression problem these outputs correspond to $y_k = a_k$ where the activation function is simply the identity. For classification problems a Soft-Max function or logistic sigmoid function is used: $y_k = \sigma(a_k)$ where $\sigma(a) = \frac{1}{1+e^{-a}}$.

Formalization of the MLP is done by grouping all the weights into a single vector w which can be adjusted to manipulate the set of inputs $\{x_i\}$ to outputs $\{y_k\}$. This network model shown in Equation 2.29 is therefore a nonlinear function which can be evaluated as a forward propagation through the network.

$$y_k(x, w) = \sigma(\sum_{j=0}^{M} w_{kj} h(\sum_{i=0}^{D} w_{ij} x_i)) \tag{2.29}$$

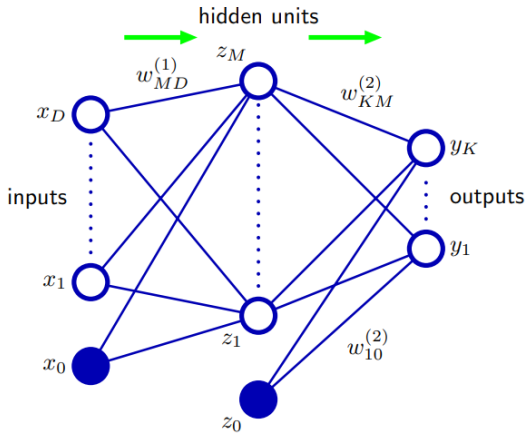


Figure 2.9: A depiction of a typical multilayer perceptron model or 'vanilla neural network'. The inputs x_i for $i = 1, ..., D$ represent the input vectors feature space where x_0 is the bias. A linear combination of the inputs multiplied by number of weights M are processed through the hidden units z representing activation functions. This is then repeated for the second layer to produce outputs y_j for $j = 1, ..., K$ where K depends on the number of classes predictions are generated for. [14]

2.3.6.3 Neural Network Training

As shown previously the neural network can be viewed simply as a series of functional transformations mapping an input to an output depending on a weight vector. Training a network involves adjusting the weight vector such that a chosen error function $E(w)$ associated with its prediction, is minimized. This represents a parameter optimization problem.

This error function $E(w)$ or cost function represents an Nth dimensional surface. Minimization of this surface involves finding a minima where the gradient of this function $\Delta E(w) = 0$. The goal is to therefore find a vector w such that the cost function is at its smallest value. By making a small 'step' in the weight space $w + \delta w$ the error function will equivalently change to $\delta E \approx \delta w \Delta E(w)$.

Gradient descent is a common optimization algorithm used for calculating the minimum of the error function. Which iteratively makes these small steps in the direction of the steepest slope until a minimum is reached. A method discussed for calculating these derivatives is the backpropagation algorithm.

As the error function is continuous, the point where the derivative is zero

does not necessarily correspond to a global minimum, which would represent an optimum weight vector. For any local minimum there will be equivalent local minima and saddle points.

2.3.6.4 Backpropagation

The method discussed above is performed by the family of algorithms known as backpropagation. They function by utilising gradient based optimization methods iteratively updating the weight matrix by small increments until convergence.

Deriving the backpropagation algorithm for an arbitrary error function $E(w)$ is done by defining a combination of the errors generated by each of N training samples $E(w) = \sum_{n=1}^{N} E_n(w)$ where the cost function is the sum of squares $E_n(w) = \frac{1}{2}\sum_k (y_{nk} - t_{nk})^2$.

As each error shown as δ_j depends on a_j the summed input to neuron j of the input layer, finding the partial derivative of this function is done by performing the chain rule with respect to each weight.

$$\delta_j = \frac{\partial E_n}{\partial w_{ij}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} \tag{2.30}$$

Similarly the hidden layer or output layer summation a_k for $k = 1, .., K$ from the connections of the previous layers activations $h(a_j)$ for $j = 1, .., M$, can be used to obtain the backpropagation formula in Equation 2.31 by omitting the derivation and manipulating previous equations:

$$\delta_j = h'(a_j)\sum w_{kj}\delta_k \tag{2.31}$$

The Equation 2.31 shows the essence of backpropagation. The error δ for a particular neuron can be derived by calculating the δ_k's of neurons further in the network and propagating them backwards.

2.3.6.5 Stochastic Gradient Descent

Backpropagation is a computationally effective method for evaluating derivatives in a neural network. A common method employing backpropagation in order to

train a network is stochastic gradient descent (SGD). It is an iterative method for minimizing a cost function by making small steps, the learning rate η, in the direction of greatest negative gradient, thus descending the slope for each increment τ.

$$w^{\tau+1} = w^{\tau} - \eta \Delta E(w^{\tau}) \tag{2.32}$$

The time for a minima to be reached depends on the error function itself and may be computationally intensive. To overcome one aspect of this *momentum* is implemented as an extension to the SGD algorithm. The intention is that by using a fixed learning rate the algorithm may progress slowly on certain slopes, by utilizing a linear combination of the gradient and the previous update a higher relative step size may be achieved.

$$w^{\tau+1} = w^{\tau} - \eta \Delta E(w^{\tau}) + \alpha \Delta w^{\tau-1} \tag{2.33}$$

There exist saddle points and local minima in the error function and the goal of an optimization algorithm reach as close to the global minima as possible. The optimization algorithm used must take this into consideration in order to find the best representation of the weight vector. To achieve this multiple extensions of SGD as well as different gradient evaluation methods have been designed.[22]

2.3.7 Convolutional Neural Networks

A variant of traditional deep neural networks implementing an input, output and multiple hidden layers. Known as the convolutional neural network (CNN) these are primarily used in image recognition and computer vision tasks. In a similar way to the biological inspiration behind perceptrons, CNNs architecture were modelled after neurons in the visual cortex. To this extent they apply to a restricted region where there exists a partial overlap between different neurons.

The advantage of CNNs in computer vision is their ability to take into account nearby pixels, which share an inherent relationship that traditional deep neural networks would require a massive number of connections between neurons to

identify. This property of an image, where the neighbouring pixels are more strongly correlated than pixels further away, is harnessed by extracting local features in the small sub-regions.

Multilayer perceptrons in a fully connected network are prone to overfitting, methods of loss function magnitude regularization are often used to overcome this. CNNs use a hierarchical pattern to transform highly complex features maps into progressively simpler ones. The process consists of a series of convolutional layers which convolve the input with a multiplication operation. The activations are followed by pooling layers, fully connected layers and/or normalization layers.

2.3.7.1 Convolutional layers

While referred to as *convolutional* the process is more in-line with a cross-correlation, the flip during the convolution operation adds unnecessary complexity to the algorithm and is generally omitted. For the purpose of this theory it will remain known as a convolution operation.

The goal of a convolutional layer is to map a defined region of the input, $x_{i,j}$ to a single neuron. The weights of the connection are represented by a kernel or convolutional filter matrix: K. These filters slide across the entire input space generating a feature map G_j for each filter, which is then applied to a non-linear activation function $h(\cdot)$.

$$G = h(\sum_i K_{i,j} * x_i + b_j) \tag{2.34}$$

By limiting the region each filter is convolved with, sparse local connectivity is produced in the network. This reduces the number of inputs to the neurons in each layer, reducing inherent complexity and computation.

The loss function can be used to adjust the trainable kernel using backpropagation for calculation of the gradient and a conventional optimization algorithm to minimize the function. These kernel matrices in the convolutional layer are in actuality synthesizing features from an input, representing feature generation.

2.3.7.2 Pooling

After each convolutional layer a pooling operation is applied as a means of implementing down-sampling into the network. This reduces the dimensionality and subsequently the complexity of the network and aids in reducing overfitting. It is a fundamental aspect to CNNs as it aims to merge generated features in the relevant sub-regions.

The pooling operations function with a defined *pooling window* which dictates the size of the individual, non-overlapping regions for which the convolutional layer is divided into. The pooling function implements down-sampling on each pooling window. One of the more common functions is max pooling which applies a max filter to the pooling window. This effectively selects the activation with the maximum amplitude in each region, reducing the number of activations propagated to the subsequent layer.

2.3.8 Regularization

It is far more likely for a model to present with overfitting than underfitting. An overfit model will perform poorly due to a lack of generalization and/or an over-complicated internal structure. Regularization can be used to restrict a model to a simpler form or inducing sparsity in order to reduce overfitting.

2.3.8.1 Generalization

By applying an additional term to the error function $E_n(w) + \frac{\lambda}{n}\sum_{i=1}^{N} w_i$ a bound can be imposed on the complexity of the function space available. A model with inherent noise in the feature space of x_n will incur overfitting in an overly complex model. This is L1 regularization whereby through a penalty term in the cost function, certain patters in the function space when building the model can bring about high generalization - balancing bias and variance.

Considering gradient descent shown in Equation 2.30 with the addition of a regularization term λ demonstrates how the biases remain unchanged while the weights are now constrained by a factor $1 - \frac{\eta\lambda}{n}$.

$$w = w - \eta \Delta E_n(w) - \frac{\eta\lambda}{n} w = (1 - \frac{\eta\lambda}{n}) w - \eta \Delta E_n(w) \tag{2.35}$$

This is known as L2 regularization and is used as a method of *weight decay* whereby the weights are constraint to smaller magnitudes while still able to increase if a subsequent decrease is achieved in the unregularized error term. L2 regularization reduces weights by in a proportion to the weight vector w, while L1 regularization is reducing weights by a constant.

2.3.8.2 Dropout

Regularization by dropout is where a defined fraction f of neurons in a layer are randomly selected to not provide propagation to the next layer. These neurons which are dropped do not contribute to forward propagation nor back propagation. The activations $a_k = h(\sum_{j=0}^{M} w_{kj} z_j)$ are now filtered via a dropout masking vector m becoming $m * h(w.x)$. The remaining activations are multiplied by $\frac{1}{f} = 2$ as a factor normalizing the reduction in the number of neurons.

The purpose of dropout is to reduce the complexity of further layers by limiting the scope of information they are presented with. By preventing additional inherent features from being generated a sparse, more generalized solution may be achieved.

2.3.8.3 Cross-validation

One of the fundamental aspects of preventing overfitting in a learning model is to train the model multiple times with different subsections of training and test data, and taking a mean of the results. This is known as K-fold cross validation. Where K is the number of partitions the dataset is divided into. An example of the algorithm is as follows:

- Randomize all the samples in the dataset
- Divide the dataset into K subsets
- Store a single subset as testing data
- For K-1 subsets:

  - Train the model on each subset
  - Record an evaluation of the models performance on the test subset
- Summarize the models true performance using the total evaluation scores

This process is repeated K times in order for each subset to serve as a test set while training can be performed on the entire dataset.

2.3.8.4 Early stopping

In the process of an optimization algorithm minimising the cost function during training, a limit is reached as the minima approaches convergence. Before convergence there may be a time where an evaluation of the test data would show a reducing in performance due to overfitting. Early stopping is a method of evaluating the test set after a defined number of epochs, and if a constraint as to its performance is met the training ends.

Chapter 3

Literature Review

This chapter covers the literature relevant to the topic of radio interference flagging as well as machine learning related to an application of RFI detection. Most articles described in this section which implement machine learning algorithms for RFI flagging are relatively recent at the time of this work.

This chapter begins with conventional, industry accepted practises for RFI flagging of astronomical data. It then details the published approaches to using convolutional neural networks treating time/frequency spectra as an image segmentation problem.

3.1 Pre-correlation RFI Identification Techniques

On-line or real-time identification of RFI at the detection stage is performed after digitization but before correlation between the separate telescopes. RFI identification techniques are near impossible to implement in analog electronics for radio astronomy. Analog-to-digital (ADC) conversion is performed in as short a time as possible to prevent the various types of noise possible from analog components, thermal noise in the SKA for example is reduced by cooling receivers to $70K$.[23]

Pre-correlation techniques can serve as powerful identification tools as the streaming data is at its highest time resolution, allowing for a fine grained approach. The drawback is that these algorithms must be real-time, having ex-

tremely low complexity. They will also have a limited scope of available data as it will come from a single telescope, or from a small time range.

3.1.1 Pulse Blanking

The presence of RFI is usually characterized as sharp impulses and in a real-time data streaming scenario where periodicity cannot be accounted for. An implementation known as pulse blanking is where a sample is 'blanked' or set to zero magnitude when it exceeds some defined threshold. With a high sample rate it is more than likely that the surrounding samples in a region around the triggering sample will also contain RFI. This strategy accounts for pre-detection and post-detection of an RFI pulse.

A simple strategy for this is after the ADC, an algorithm can detect whether a sample's power exceeds the mean power by a defined number of standard deviations. A defined number of samples before and after this trigger are then set to zero.

Asynchronous pulse blanking was a method implemented for detection and pulse blanking of L-band RFI. The pulses detected ranged from a length of 2 to 400 μs which occurred between $1 - 75$ms apart. The application was developed on an FPGA which maintained a running mean m and standard deviation α of samples magnitude with a limited FIFO memory in order to account for nearby samples. Detection occurred if a threshold an input samples magnitude, $||x||^2 > \beta\alpha + m$. [24]

3.2 Post-correlation RFI Identification Techniques

Radio astronomy presents extremely large volumes of data and any unnecessary or complex processing in the imaging pipeline can prove to be impractical. Flagging techniques in post-correlation are often the first step in the imaging pipeline and the data has likely been processed by some form of pre-correlation RFI identification. It is the last step to ensure no corrupted data is passed to the imaging pipeline.

Dealing with RFI can be in the form of excision, where the identified signal is subtracted in an attempt to recover the underlying information. Flagging is the process of generating a Boolean flag map corresponding to each sample identified as RFI in the time/frequency domain. Imaging pipelines rely on binning techniques for averaging in time/frequency and *uv* plane binning, where flagged samples are omitted they are often replaced with a mean value from the entire bin. This equates to an interpolation of lost values and has been show to result in errors in the form spectral fluctuations. Consequently if non-RFI samples are flagged, representing false-positives, they are not presented to subsequent data reduction steps in the pipeline. This can result errors or artefacts in the imaging plane such as incorrect flux densities.[13]

3.2.1 Thresholding Algorithms

The most common form of RFI identification methods are derived from various thresholding techniques operating on the magnitude of a signal, where RFI is most obvious. Thresholding techniques operate on a sample, channel or entire region in order to identify RFI if a defined quantity exceeded a determined or variable limit. The assumption of a thresholding algorithm relies on RFI having a higher magnitude exceeding that of the celestial data. To this extent RFI exists as a outlier in the dataset, with a higher deviation from the norm. The inherent application of a thresholding algorithm is therefore based around how the limit is derived, if set too low RFI will be present and propagate through the pipeline. If a threshold is set too high the excess data removed will reduce the validity of subsequent processing.

An example of a simple thresholding technique is to set the limit as a median of the magnitude time/frequency spectrogram, as it is invariant to the RFI outliers unlike the mean. Any sample above the median is considered corrupted, although this would lead to a loss of valid data and would be unable to capture transients which increase in magnitude until a peak.

3.2.2 Combinatorial Thresholding

The goal of combinatorial thesholding is to take advantage of the fact that RFI often appears across multiple localised time or frequency samples. Instead of a global threshold χ_1 being determined, the technique uses varying thresholds which can take into account whether multiple nearby samples both exceed a slightly lower threshold χ_2. This method prevents missing flags for RFI which have slightly lower magnitudes than that of the sharp peaks.[25]

3.2.2.1 VarThreshold

The VarThreshold technique falls under combinatorial thresholding as it takes a given set of N decreasing thresholds $\{\chi_i\}_{i=1}^{N}$ and iterates in both the time and frequency to detect if a combination of samples exceed the threshold χ_1. The algorithm for detecting if a single sample $R(v,t)$ should be flagged as RFI is shown in the equation below:

$$\begin{aligned}\text{flag}\nu_M(\nu,t) = \exists i \in \{0\dots M-1\} : \forall j \in \{0\dots M-1\} : \\ |R\left(\nu + (i-j)\,\Delta\nu, t\right)| > \chi_M \end{aligned} \tag{3.1}$$

For a number of samples M in either the time or frequency direction as a combination, a sample is flagged if any of the two rules applies. This functions as a flagging method taking localised, directional RFI into account, though limited to being orthogonal.[26]

3.2.2.2 SumThreshold

A variation of the VarThreshold method with improved performance and commonly used in radio astronomy is SumThreshold. It utilizes the method of combined samples as in VarThreshold but in this case the threshold criterion is determined by a sum of one or more samples in the combination. This implementation allows for samples to be flagged which fall below the initial thresholding value.

An added criterion is that samples are thresholded in an increasing order of thresholds, if a lower threshold has been reached the RFI contaminated samples

are substituted from the summation with that threshold value. This has the advantage of preventing the overflagging which occurs in VarThreshold from sharp spikes, which cause a drastic increase in the related sum.[27]

3.2.3 Surface Fitting

Relying on the assumption that the amplitude of most astronomical signals do not exhibit rapid changes in time/frequency a surface $\hat{V}(v,t)$ is defined. This smooth surface of the correlated visibilities $V(v,t)$ will show sharp RFI spikes which deviate from the surface, represented as $N_{RFI}(v,t)$, which can then be thresholded. As iteratively fitting two dimensional polynomials is a complex algorithm the spectrum is divided into k separate tiles and therefore k separate surfaces. The coefficients for the least square fit for a two dimensional, order N polynomial is shown as:

$$\hat{V}_k(v,t) = \sum_{i=1}^{N_v} a_{k,i} v^i + \sum_{i=1}^{N_t} b_{k,i} t^i + c_k, \tag{3.2}$$

The fit is produced iteratively by defining a weight function for the thresholding limit and minimizing the error for each tile. As each tile is an independent function there exist deviations on each boundary which will produce inherent errors. A more accurate method to tiling is to use a median sliding window approach around each sample as the local median would remain smooth compared to a mean.[26]

3.3 Machine Learning for RFI Flagging

The use of machine learning algorithms implemented to identify and flag RFI in radio astronomy datasets is relatively recent. Majority of the literature regarding its use involves training supervised algorithms on time/frequency data flagged by existing methods, or using simulated data where the ground truth is generated and therefore certain. While work has been done distinguishing and characterising different types of RFI using machine learning techniques, it is somewhat unrelated to the identification of RFI corrupted data against usable data.[8][28]

3.3.1 Data Simulation

There are advantages and disadvantages to using either real data or simulated data. Real data is advantageous as comparisons made between the performance of learning algorithms and known flagging techniques can be compared to the resulting images from known, certain measurements. Their shortcoming is by training a learner on the results of another flagging technique no certain ground truth is known, as any existing flagging algorithm will almost always contain errors. This is solved with simulated astronomical data and generated RFI, if the RFI is injected in a known manner it can function as an absolute ground truth for any flagging algorithm to compare. Yet simulated data will always have inherent uncertainties when it comes to modelling real measurements.

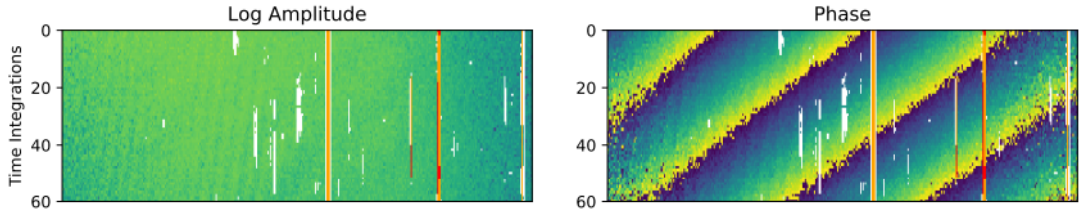
The simulation package *hera_sim* generates Hydrogen Epoch of Re-ionization Array (HERA) data by modelling a pseudo-sky from the discrete form of the visibility equation $\tilde{V}(t,v) = \sum_n [\tilde{A}(\tau,\hat{s}) * \tilde{S}_n(\tau) * \delta(\tau_n - \tau)]$ with point source flux densities sampled from a specific power-law distribution. Diffuse emissions are simulated with the Global Sky Model (GSM) combined with a model of the HERA primary beam. Baseline lengths are sampled from a half-normal distribution. RFI is then empirically modelled to resemble the RFI environment on-site, with 3% of the dataset containing RFI. The RFI is modelled as narrowband persistent, narrowband/broadband burst and single time/frequency blips. Their frequency of occurrence is empirically modelled from real HERA RFI expectations. An example of the generated time/frequency data is shown in Figure 3.1a.[11]

Generation of the effects produced during the HERA signal chain were modelled with three phenomenon:

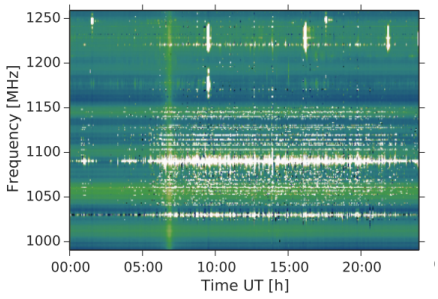
- Cross-talk: convolution between the simulated visibility and Gaussian noise
- Bandpass: fit of a 7th order polynomial to real bandpass measurements
- Gain fluctuations: phase delays between $-20 \leq \delta_{tau} \leq 20ns$

The HI Data Emulator (HIDE) package simulates time/frequency data, referred to as time-ordered data (TOD), using the GSM stored in Hierarchical

Equal Area isoLatitude Pixelation (Healpix) maps. Healpix maps are a transformation for the pixelisation of a two dimensional sphere where each pixel shares the same surface area, allowing for a spherical representation of a sky-map. The simulation capabilities of HIDE are extensive, allowing for adjustment of beam profiles and survey strategies. The GSM Healpix map is interpolated to the relevant frequency and convolved with the beam response, generating a TOD array. This TOD array is measured in kelvin (K) and is transformed by applying a gain and baseline offset. The RFI is simulated by an exponential or Gaussian with the same profile in time or frequency. For each frequency bin the magnitude of the RFI is sampled from a uniform distribution. The RFI then undergoes a 2D FFT convolution with the time-frequency plane. The issue with this implementation is that the generated RFI is then embedded within the underlying data. This is shown by the spectrogram in Figure 3.1b.[29]



(a) Magnitude and phase data generated with *hera_sim*.[11]



(b) Magnitude data generated with *HIDE*[30]

.

Figure 3.1: Example spectrograms of simulated data and RFI. Both are available as open source packages.[31][32]

3.3.2 Data Preprocessing

The preparation of time/frequency data for use in machine learning is a fundamental step for generating an accurate learner. Generating a flag map corresponding to every sample needs to be performed for each baseline. This can be achieved by convolutional methods with the output as a single 1×1 convolution performing segmentation by predicting each sample. This has an advantage of maintaining the relationship between nearby samples, as most RFI is likely to occur across multiple samples in time or frequency. The other method is to flatten the visibility plane into a single one dimensional array, treating each sample individually but losing their spatial relationship.

3.3.2.1 Feature Engineering

For both data shapes, the entire visibility or a flattened one dimensional array, the task of identifying features could be crucial to improving a learner. Unlike the majority of machine learning implementations where features are plentiful and dimensionality reduction techniques are required, a visibility $V(v,t)$ consists of only time samples at different frequencies. Identifying features for machine learning algorithms can be performed by using the magnitude and phase of a baseline's visibility. Doing so has shown improvements over using only the magnitude.[11]

When multiple features are employed its common practise to normalize features in order to prevent learning errors owing to effects in their related euclidean range difference. In this case the phase is always constrained between $-\pi \leq \phi \leq \pi$ but the range of possible RFI outliers influencing the magnitude must be normalized. This is achieved through Equation 3.3, which works to lessen the dynamic range issues caused by high magnitude RFI by centering the magnitude feature through the removal of its mean and standard deviation.[11]

$$\hat{V}(v,t) = \frac{(ln|V| - \mu_{ln|V|})}{\sigma_{ln|V|}} \tag{3.3}$$

Other work employed feature construction methods, using each polarization and stokes parameter to develop explicit statistical features. As a pre-processing

step the spectra are flattened to form a one dimensional array. Median Absolute Deviation is used in order to remove the high magnitude RFI outliers from the dataset.[33]

$$M_i = \frac{0.6745(x_i - \tilde{x}}{median(||x_i - \tilde{x}||} \tag{3.4}$$

Once removed the remaining data is normalized in order to increase the difference between low-level RFI and usable data. Statistical features were then generated on the pre-processed data using a sliding window of length 8. It is assumed the sliding window is performed on a per channel basis, and it is unknown how edge cases are handled ie: mirroring, zero padding. Deriving explicit statistical features may be a beneficial step in an implementation such as this where algorithms like K-Nearest Neighbours were used. Deep learning techniques generate implicit features in their hidden layers which work to further separate classes. [33]

The features generated using the sliding window approach are:

- channel mean
- kurtosis
- skewness
- cumulative sum
- variance
- upper and low percentiles
- sum of the window

In an unsupervised domain deriving useful features is of utmost importance. As opposed to generating features from existing data an alternative approach is to use additional meta-data relating to an observation. This work averaged the dataset across time in order to gain insight into the frequency structure of each possible RFI event, leading to each of the 96 frequency channels to function as features. This is likely a useful feature as majority of the man-made RFI from their location would fall into specific frequency bins. The meta-data associated with each observation which were used as features were; time of day and the

antenna direction. All of these features serve the same purpose, RFI events are likely to re-occur from specific directions such as cities where there would more likely be events during the day when the city is active.[34]

3.3.3 Classification Algorithms

The use of three classification algorithms; K-Nearest Neighbour, Random Forest Classifier (RFC) and Naive Bayesian (NB) showed significant results when trained on generated statistical features. The approach of using a sliding window for feature generation and labels generated by AOFlagger proved highly effective with the three methods achieving highly accurate results with RFC showing the highest accuracy.[33] Further work using the same pre-processing techniques investigated their implementation of RFC when trained and tested on separate polarizations, these results showed the horizontal polarization proved to be the most accurate for predicting RFI in other polarizations.[35]

3.3.4 Convolutional Neural Networks

The use of CNN's in the U-Net architecture have been implemented by treating the two dimensional visibility plane as an image segmentation problem. Both approaches have trained their architectures on simulated data.[11][30]

The U-Net architecture functions as a fully convolutional network, intended to be accurate with far less training data than traditional CNNs. The architecture is designed to have a contracting path of convolutional layers followed by an expanding path where pooling layers are replaced by up-sampling operations. This expanding path is intended to construct localised high resolution features from the contracting path which are combined with the up-sampling operations output. This combination functions similarly to an auto-encoder, representing the input in a lower dimensional subspace and then expanding to match the input size. The connections between the down-sampling and up-sampling layers function in a manner similar to residual connections, with a concatenation of features as opposed to an add. The purpose of which is to recover full spacial resolution

with the use of a crop function. The final output is a 1×1 convolution to map the high level features to each "pixel" or time sample of the spectrogram. The architecture is fully convolutional, the lack of fully connected layers improves the time complexity during training and inference. The original U-Net architecture is shown in Figure 3.2.[36]

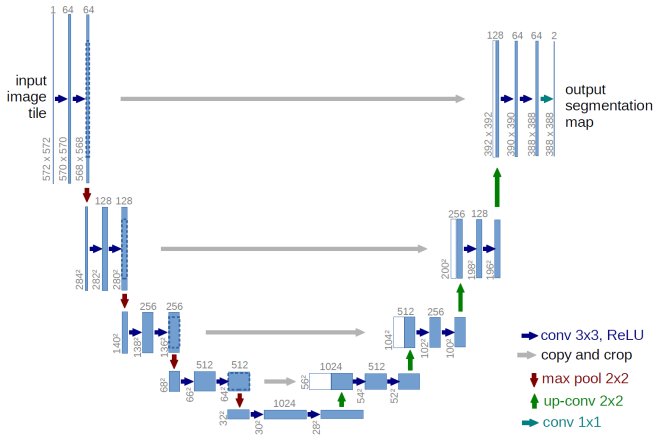


Figure 3.2: Full connected convolutional neural network in the U-Net architecture. The symmetric down-sampling and up-sampling layers give the architecture its name. The copy and crop connections shown between the two directions is a concatenation operation between those layers' feature maps, used for reconstructing the low dimensionality subspace back to the input size. The final operation is a 1×1 convolution, representing a segmentation for each pixel of the convolved input space.[37]

Utilizing the U-Net architecture for time/frequency spectrograms has been performed on a single polarization using only the magnitude, or both magnitude and phase. A U-Net trained and tested on SEEK simulated magnitude data flagged using SumThreshold was shown to achieve an F1-score of 0.85, while the ground truth generated by SumThreshold only achieved an F1-score of 0.75. This is indicative of the architectures learning capabilities, while trained on data with inherent false positives it is able to develop and understanding of the mechanisms behind how SumThreshold operates and even improve on it.

The use of a polarization's magnitude and phase compared to only the magnitude was investigated by modifying the U-Net architecture, shown in Figure 3.3. This implementation combines the downsampled and up-sampled convolutions from both the magnitude and phase with a concatenate operation. This function

serves the same purpose as the skip connections in U-Net, preventing overfitting to higher order non-linearities in deeper layers, but also functions by allowing separate learning to occur with the magnitude and phase simultaneously. The results showed slight improvement over training the U-Net on magnitude only data. With the recall, precision and F1-score of $0.9, 0.82, 0.88$ respectively. A major advantage of the *hera_sim* simulated RFI was it allows for evaluation of predictions per class of RFI. This showed the implementations ability to accurately predict narrowband RFI, yet difficulty in predicting broadband bursts and single sample blips.

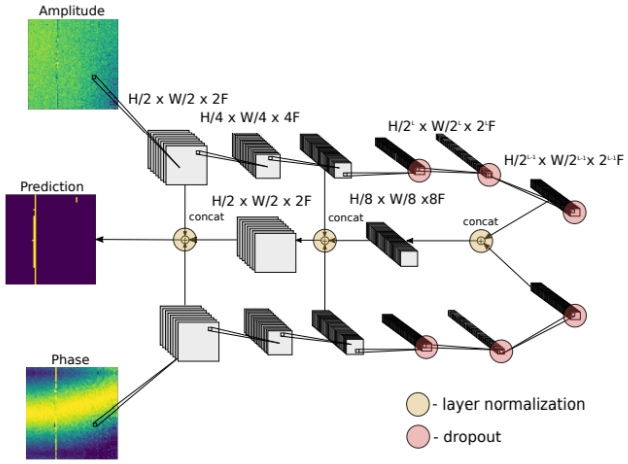


Figure 3.3: Figure depicting U-Net architecture utilizing magnitude and phase concurrently. H and W represent the input dimensions and F the number of filters.

Chapter 4

Design

This chapter conceptualises the work into machine learning for RFI flagging by defining the problem statement and identifying methods for its implementation. It begins with an exploration into the real dataset and its properties. The metrics necessary for evaluation of the machine learning algorithms are described. This chapter then gives an overview of the feature generation procedures which are to be used as a pre-processing step.

4.1 Problem Statement

Flagging of RFI involves creating a Boolean flag mask corresponding to a baselines visibility. For each time/frequency sample the flag mask identifies non-corrupted data as a 0 while 1s represent corrupted data. This may include RFI, measurement errors or artefacts. The scope of real data used in this research is limited to a single KAT-7 measurement. Data is stored in CASA measurement set (MS) files, a tabular format with multiple entries and varying datatypes relating to an observation.

This research is limited to investigating the ability for a learning algorithm to identify the underlying structure of RFI vs useful data by using flag arrays generated from existing techniques and simulated data. As much as possible calibrated data is to be used, this is due to sources of corruption such as; antenna or correlator failures, weather effects and shadowing. Existing flagging techniques are

able to identify these effects from relevant meta-data or by manual investigation.

Validation of the predicted flags is performed through true/false positive/negative evaluations. While for real data it is possible for a small amount of flags to represent incorrect classes, in the simulated dataset all the created RFI will have their corresponding flags correctly identified. Imaging through deconvolution of the visibility data is performed in order to generate images, of which source finding and blob detection can be performed as a comparison metric between algorithms.

4.2 Real Dataset Exploration

This section describes the type of data used and investigations into its properties.

4.2.1 Measurement Set

CASA utilises a table format describing a directory known as a measurement set (MS). The MS file storing the real dataset used in this work contains a total of 115500 time samples with a bandwidth of $156MHz$ and a centre frequency of $1371MHz$ with 200 separate channels after correlation. The co-ordinates of each of the 7 antenna is listed for a total of 21 baselines. From the number of time samples and baselines it can be inferred that each visibility will be $115500/21 = 5500$ samples over 200 channels. The tables contained within this measurement set are the following:
UVW, FLAG, FLAG_CATEGORY, WEIGHT, SIGMA, ANTENNA1, ANTENNA2, ARRAY_ID, DATA_DESC_ID, EXPOSURE, FEED1, FEED2, FIELD_ID, FLAG_ROW, INTERVAL, OBSERVATION_ID, PROCESSOR_ID, SCAN_NUMBER, STATE_ID, TIME, TIME_CENTROID, DATA, WEIGHT_SPECTRUM

Other work has shown the advantages of using added meta-data to measurements such as antenna direction and time of day.[34] For this work the fields of interest will be DATA and FLAG which correspond to the measured visibilities and generated flag maps respectively. An example baseline from the dataset is given in Figure 4.1a which demonstrates all the available polarizations and a flag mask generated by a combination of AOFlagger and manual flagging procedures.

Accessing CASA MSs is done via TaQL a high level SQL-like table query language. A query language is effective for extracting specifics of a dataset from a high level but has costly data access overheads. Data exploration and pre-processing is performed on the DATA and FLAG tables, these are extracted per baseline using a query such as: *taql('select DATA from $table where ANTENNA1=0 AND ANTENNA2=1').* For calibrated datasets not all antennas or antenna combinations will be present, in order to identify each baseline it is necessary to calculate all possible combinations using the ANTENNA1 and ANTENNA2 tables. Extracted baselines are then recorded to HDF5 files. Hierarchical Data Format (HDF) is a set of file formats designed for large data storage, allowing for rapid data access with inherent compression resulting in reduced storage requirements.

4.2.2 Exploratory Data Analysis

Exploratory data analysis is the process of investigation and analysis into datasets through statistical and visual methods in order to identify major characteristics. The goal of this section is to attempt to identify any underlying relationships and structures which may discriminate between the astronomical information and corrupted data points which include RFI.

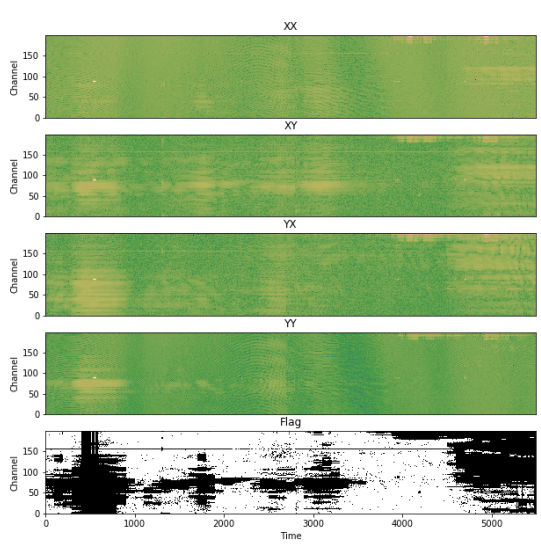
4.2.2.1 Magnitude Investigations

As the measurement data is recorded as complex values this section will explore the relevance of the magnitude vector in identifying useful information.

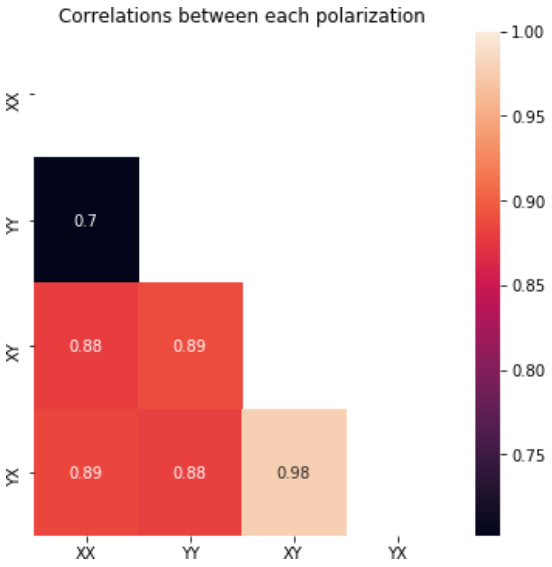
Shown in Figure 4.1a is the magnitude of the time/frequency domain from a calibrated baseline with each of the four polarizations and the flag representing the RFI identified by AOFlagger. Visibly the high intensity RFI is easy enough to see with the magnitude displayed logarithmically and the color palette showing it in a brighter yellow. Individual flags may be generated for each polarization as even visually there are slight differences between them, yet it is more common to distinguish a single flag array for the entire time/frequency spectrum. Another approach would be to combine each individual polarization's flag into one for the baseline.

The polarizations of the baseline itself are computed via auto-correlations and cross-correlations between the dual linear antenna feeds. Figure 4.1b shows the correlation

coefficients between each polarization. The orthogonal nature of the XX and YY auto-correlations is shown by their relatively low correlation compared to the others. Where the XY and YX cross-correlation outputs are relatively similar likely owing to their derived nature.



(a) Example data and flag columns from the fully polarized dataset. Each polarization of a single baseline is given with a flag mask corresponding to the baseline.



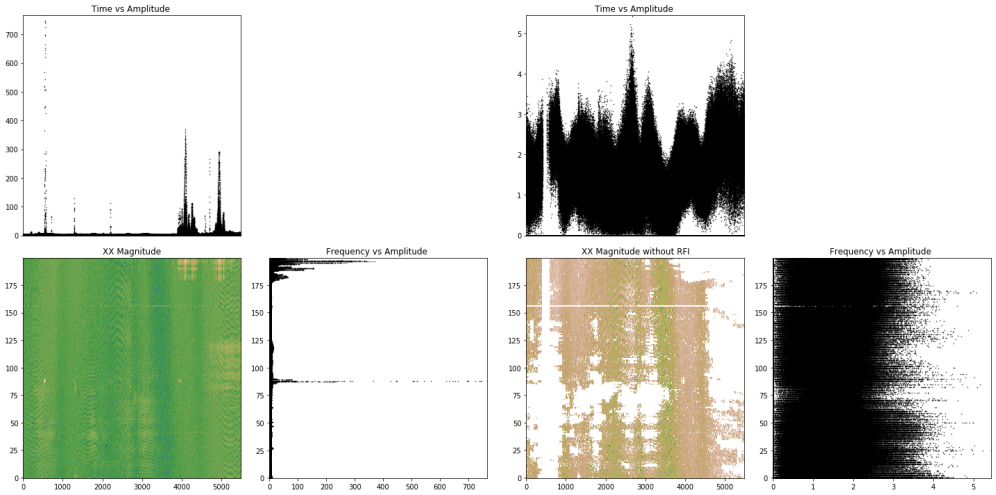
(b) The Pearsons correlation coefficient between each polarization of a single baseline. The most dissimilar polarizations are the XX and YY

Figure 4.1: Plots showing the visiblity for each polarization and the correlation coefficients between them. Visually it is difficult to distinguish differences between each visibility which is indicative of their high correlations, demonstrating their similarities.

Visualizing the impact the high intensity RFI has on the magnitude of a baseline is shown in the scatter plots in Figures 4.2. Once the RFI is removed the sinusoidal nature of the underlying waveform is visible in the time vs amplitude plot. This quasi-sinusoidal signal represents an ideal case of the cross-correlation output, also known as the "fringe". This fringe for a point source is given by:

$$r(\tau_g) = |V| \cos(2\pi\nu\tau_g + \phi) \tag{4.1}$$

Where τ_g is the geometric delay between the antennas of the baseline. This geometric delay varies as the Earth rotates. While the relative position of the source is static the fringe will change as a function of time. This causes the signal from the correlator to be a smooth quasi-sinusoidal function in the ideal case but various effects such a time-varying gain can cause the amplitude to vary as a function of time even while the source is stable.



(a) Subplots showing the amplitude of time samples over both time and frequency. This demonstrates the extent to which outliers effect the dataset owing to their far higher magnitudes.

(b) A scatter plot over time and frequency showing how a flagged baseline reduces outliers to the extent of which the sinusoidal nature of the radiation is visible in the time spectrum.

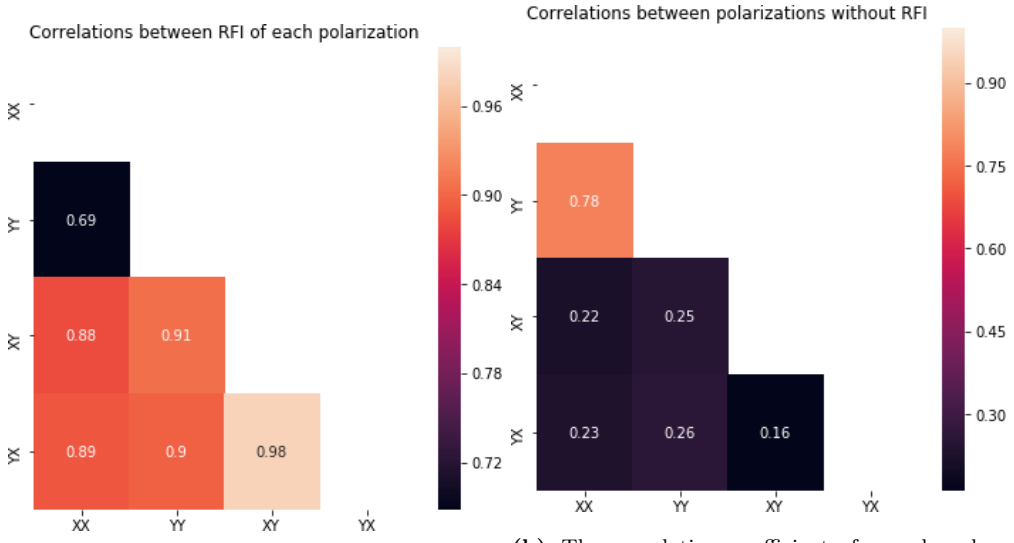
Figure 4.2: Subplots showing the amplitude of time samples over both time and frequency. For a baseline with RFI 4.2a and without RFI 4.2b.

The Figures in 4.3 show the correlation coefficients between each polarization with RFI and without RFI from the entire dataset. It is demonstrated that the correlations in Figure 4.3a are almost identical to the correlations shown in Figure 4.1b which are the correlations for the a single time/frequency spectrum across polarizations.

These plots demonstrate a key insight into using the information from each polarization in order to better identify RFI. As the magnitudes of RFI are vastly different to the underlying signal, and the correlation between these magnitudes show an inherent relationship between the RFI across polarizations. This serves as a means of identifying the RFI and distinguishing it compared to the celestial radiation, the apparent relationship of RFI across polarizations - if identified or learnt, may function as the discriminating factor.

4.2.2.2 Phase Investigations

The inclusion of phase information has been not shown significant benefit to the accuracy of machine learning classifiers.[11] Visualized in Figure 4.4 it can be seen that a similar relationship to that of the magnitude in Figures 4.1a. The RFI is also visually apparent, where localised regions have a sudden change in value, which again correlates



(a) The correlation coefficients between only the extracted RFI for each polarization. It shows the same trend and almost the same values to Figure 4.1b, with high values for the XY and YX components, while lower for XX and YY.

(b) The correlation coefficients for each polarization with the RFI flagged by AOFlagger completely removed. It shows the exact opposite trend to the RFI correlations, which is likely due to the sinusoidal nature of the underlying waveforms as shown in Figure 4.2b.

Figure 4.3: Subplots showing the correlation coefficients for the time/frequency measurements of each polarization over all the baselines combined. The correlations of the RFI only, is shown in Figure 4.3a and without RFI 4.3b. It is noted the drastic differences between the two subplots. There is a high correlation between RFI, almost the same values as the correlations between the actual polarizations shown in Figure 4.1b which is the combined RFI and non-RFI. There exist very low correlations between the underlying signals.

to the flags given in the previous Figure 4.3.

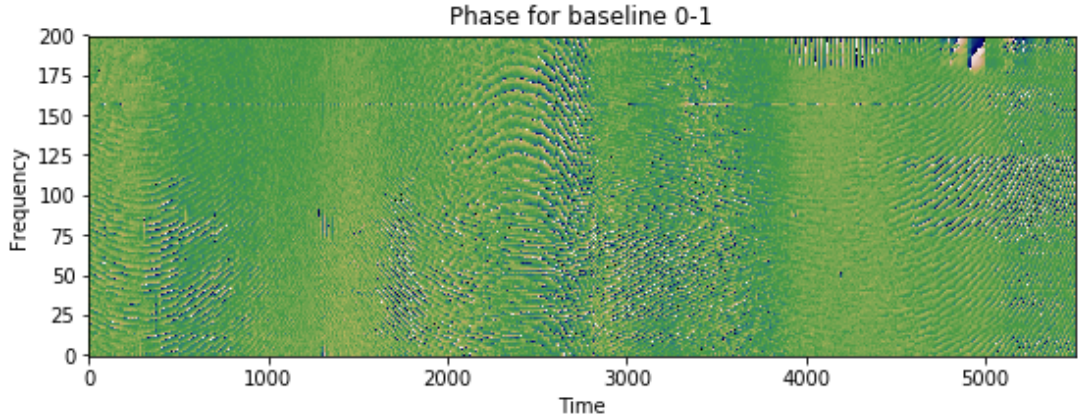


Figure 4.4: Plot of a baselines phase content in time/frequency.

Tests into the correlation between the phase values of polarizations showed little to no correlation, contrary to the same tests done using the magnitudes.

4.2.2.3 Statistical Properties

Investigating the statistical nature of the real dataset is done through normality tests. As the source of the celestial data are real world emissions they follow a Gaussian distribution. Conversely while natural RFI can be Gaussian in nature, many forms of man-made RFI are non-Gaussian in nature.[28] [38]

Applying normality tests to test the null hypothesis are done by a Shapiro-Wilks test and the Kolmogorov-Smirnov test for a normal distribution. This is performed for the real component of each polarization for every baseline with the flagged RFI removed and flattened into a one dimensional array. The results of these tests are shown in Table I which demonstrate that both result in a P-value of < 0.5 and they both reject the null hypothesis and that the non-RFI data is not normally distributed.

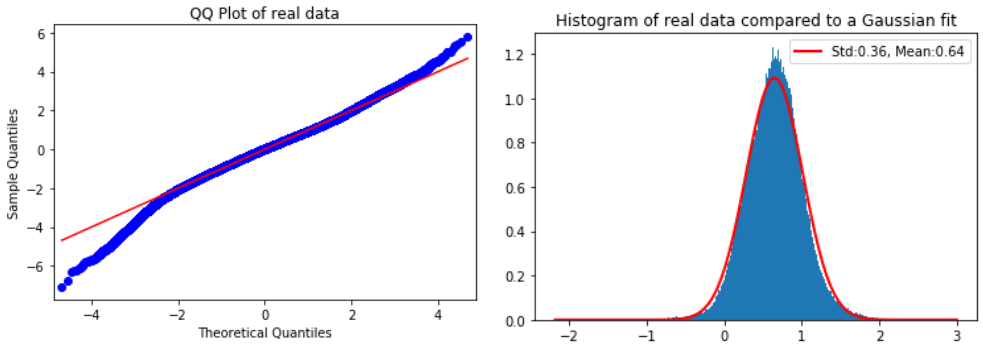
Table I: Comparison between the two normality tests.

Normality Test	Statistic	P
Shapiro-Wilk	0.992	0
Kolmogorov-Smirnov	0.0185	0

To further investigate this the mean of the real component for each polarization of a single baseline is used to generate a quantile-quantile (QQ) plot and a histogram,

both of the real values without the flagged RFI. Using the entire dataset would introduce additional uncertainty owing to additional effects such as gain calibration across baselines.

These plots shown in Figure 4.5 demonstrate visually that the data appears to follow Gaussian distributed in nature. The offset tails of the QQ plot indicate that while most extremely high magnitude values have been removed, some values still represent outliers from a normal distribution. The same is shown in the histogram, which extends to a higher range with a maximum value of 2.7 and minimum of -1.94.



(a) Quantile-quantile plot demonstrating that a Guassian distribution is plausible for the real component of the data with flagged RFI removed.

(b) A histogram of the real data with a normal distribution fit. The standard deviation and mean of the Gaussian plot are given in the legend.

Figure 4.5: Plots investigating the applicability of normality tests to the dataset. Celestial data follows a normal distribution for most point sources. By removing the RFI these two plots demonstrate the distribution visually.

As these plots use the mean of each polarizations' real components they were performed multiple times using different baselines and presented the same outcome - that the celestial information without RFI for this dataset is likely a normal distribution but presents with some outliers remaining as the flagging method would not be completely accurate.

4.3 Accuracy Metrics

The ability to differentiate between RFI and non-RFI samples could be abstracted to a trivial thresholding approach. It is crucial to identify a baseline accuracy for which future machine learning methods must drastically outperform. To accomplish this some simple heuristics are chosen and their accuracy as methods will help identify low

threshold for which any future implementation must outperform. These investigations are based on each sample which is higher than the statistic is identified as RFI, where the statistics are: mean, median, and lower percentile. The results of these investigations for a subset of 20000 samples of a single magnitude of the real dataset are shown in Table II. These results show some insights into the relationship between classes in the XX magnitude of the real dataset.

Precision can be viewed as the fraction of identified RFI that is actually RFI, while recall is the fraction of all RFI samples correctly identified. The high precision of *mean* therefore represents a high proportion of correctly identified RFI compared to the incorrectly identified non-RFI. The high recall for *median* means majority of RFI samples lie above the median of all samples. Similarly for the *lower percentile* where almost all RFI samples lie outside of the lower percentile, but with a low precision there are many non-RFI samples in this region as well.

Table II: Simple thresholding methods for accuracy comparisons.

Threshold method	Precision	Recall	F1-Score
Mean	0.86	0.51	0.64
Median	0.21	0.80	0.33
Lower percentile	0.13	0.99	0.23

The task many learning algorithms is to identify a decision boundary for which classes can be differentiated. Each of the above heuristics can be viewed as a linear decision boundary based on the magnitude of samples. When visualizing the distribution of magnitude samples from two different polarizations on each axis as shown in Figure 4.6 its apparent that a non-trivial and non-linear decision boundary is necessary for a higher accuracy. This is compounded by the fact that n features or polarizations would require an n-th dimensional non-linear decision boundary or discriminating hyperplane.

4.4 Feature Engineering

Posing the problem from a machine learning point of view can represent the data in two possible variations. Both share the common goal of generating a Boolean flag mask for each baseline.

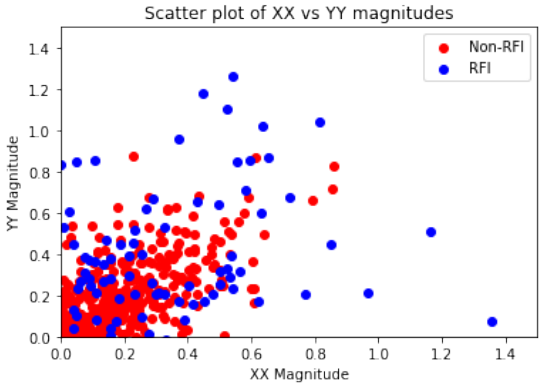


Figure 4.6: A demonstration of the separability between classes on a small subset the real data. The blue RFI points are higher magnitude and their presence as outliers can be seen visually. Yet the task is non-trivial, with some non-RFI instances still appearing with higher magnitudes and conversely RFI samples with lower magnitudes.

The first method for doing so is by treating the time/frequency domain in its two dimensional representation similar to an image. In this way a polarization can function as a image with its segmentation ground truth being represented as the flags. This is binary segmentation as performed by computer vision tasks.[11][30]

The second method treats each time/frequency sample individually with its binary class corresponding to its Boolean flag. Each individual sample then has features associated with its properties such as magnitude or phase.[33][22]

While this second method allows for a wider variety of machine learning methods besides those from computer vision tasks, it has the down side of dissolving the spatial relationship between RFI samples. As RFI bursts occur in both the spacial and temporal domains, when viewed as a 2D representation the locality of RFI samples demonstrates an inherent relationship within the data.

For this method, features are generated from each time/frequency sample in order to provide a learner with a wider variety of representations of that sample. This may then be used to identify relationships between the RFI and non-RFI samples.

4.5 Feature extraction

As the required result is a single flag mask for the entire time/frequency domain each polarization represents a feature for that sample. There are slight variations between polarizations. With high correlations for the RFI while lower correlations for the non-

RFI data, each polarization should provide a learner with a different representation from which to identify the inherent relationship between the classes.

Further features can be extracted by collecting multiple statistical measurements relating to a sample. These may be for the sample, across each polarizations or as performed in related work which used a sliding window approach to extract statistical features from a single polarization.[33] This method is useful for generating features which maintain the spatial relationship between RFI samples.

This can be performed in multiple ways. Using a one dimensional M sample length sliding window, traversing either time or frequency directions generating the statistical feature for the center sample of the window. This will result in some semblance of either temporal or spatial relationship between the nearby samples captured in the window depending on the chosen direction.

4.5.0.1 Filtering

In order to capture a more true representation of the spacial relationship an $M \times M$ two dimensional sliding window is used, also known as a filter. This maintains the relationship a sample has to neighbouring samples in both time and frequency directions. It is also possible to use a three dimensional $M \times M \times 4$ window for averaging the result between polarizations, the former method will generate $\times 4$ as many features. Modern machine learning and compute capabilities will not be drastically affected in terms of processing time by ± 100 features and the higher number of features allows for further optimization and theoretically improved accuracy.

Each of the statistical measures to be extracted as given below: [22][33]

- Median: Central value of the segment
- Mean: Numerical average of the segment
- Skew: Skewness of distribution of signal voltages in the segment
- Kurtosis: Approximate nature of Gaussianity of the segment
- Max sum: Maximum value of cumulative sum
- Slope of signal: The slope a linear fit to the segment
- Lower quartile: 25
- Upper quartile: 75
- Quartile range: Difference between upper and lower quartile

- Standard deviation: Indication of the segments variance
- Min: The minimum sample value
- Max: The maximum sample value
- Convolution: By row
- Convolution: By column
- Rank filter: Rank 1
- Rank filter: Rank 3
- Prewitt filter: Prewitt operator for edge detection
- Laplace filter: Discrete Laplace operator

This two dimensional sliding window, or filter, is performed on the time/frequency domains of a baselines polarizations. As this correlator output is complex valued it may be performed on the real and imaginary components or the magnitude and phase. Performing on all four is likely derivative as the latter are constructed by the former. It is more likely that useful information is present in the magnitude and phase information.

A visualization of a 3×3 median filter performed on a magnitude spectrum is shown in Figure 4.7. These plots both use the same colour map and are logarithmically normalized for visualization purposes. It can be seen how fringes are persevered in the background and the high magnitude values are far more pronounced. The spacial relationship between nearby samples is not only maintained but enhanced, for example the narrow band RFI between time 3000 and 4000 at frequency band 160 is still present and more visible.

It must be noted that this visualization is a reshaped version of the internal representation of the data. In reality each sample is independent, forming the rows of a matrix $5500 * 200 = 1100000$ long and N number of features forming the columns.

4.6 Simulated Data

The use of simulated data may is necessary for assessing the true accuracy of a possible flagging algorithms. Tests on real data can be validated against a ground truth generated by an existing algorithm or through manual flagging, but this has an inherent uncertainty owing to the fact that the these flags are not an absolute ground truth and will only carry the accuracy of the related flagging method which generated them. When using simulated visibility data combined with simulated RFI the exact ground

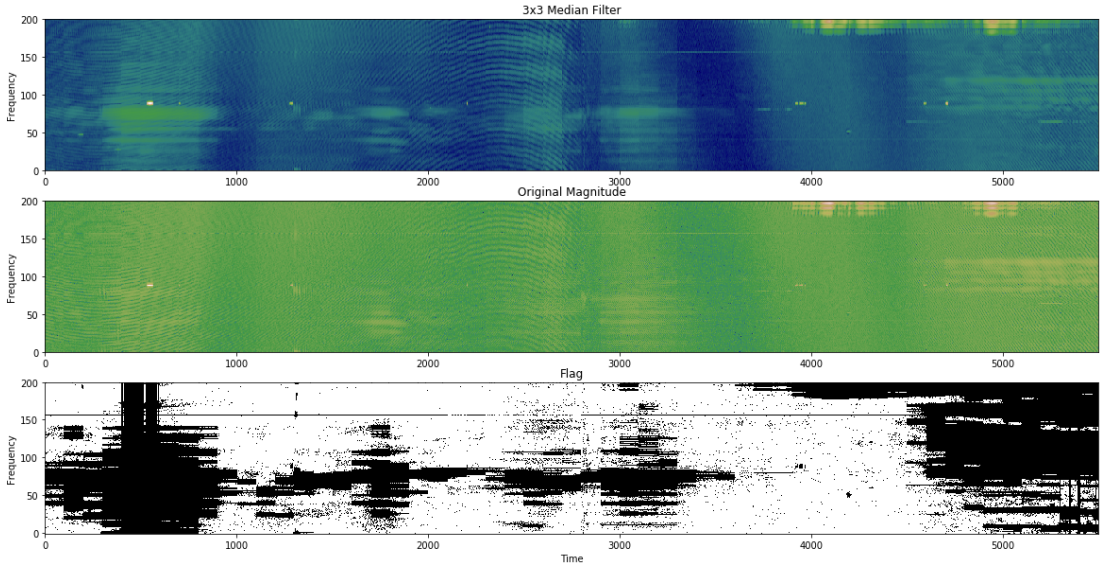


Figure 4.7: Plot visualizing the effect a 3x3 median filter has on exaggerating the RFI outliers.

truth can be measured as the simulated RFI will be associated with exact sample locations.

4.6.1 HERA Sim

The package *HERA_ Sim* is used for simulating visibilities and RFI.[32] These simulations have been used in research of deep learning for RFI flagging.[11] The time and frequency windows are able to be adjusted and the RFI simulation has three variants: narrowband persistent, narrowband/broadband burst and single sample blips. An example of the simulated time/frequency plot is shown in Figure 4.8, it has been adjusted to the same time scale and number of frequency bins and the real data to be used.

A disadvantage of this simulated data is that currently it is unable to simulate all four polarizations. In addition simulating RFI can only be done on the XX and YY polarizations separately. This limits the construction of tests which utilize multiple polarizations as additional features.

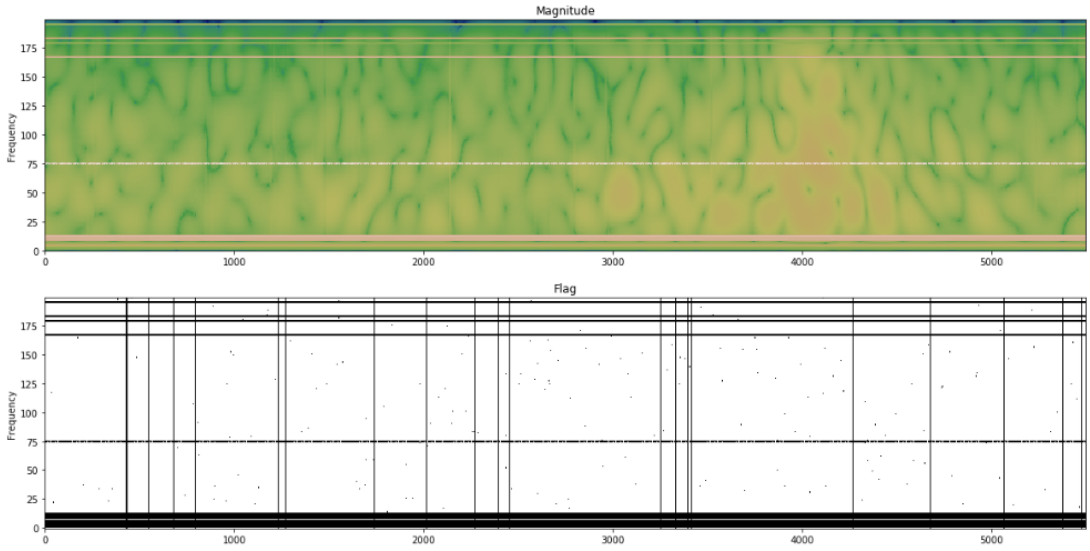


Figure 4.8: Simulated time/frequency for a single polarization with added RFI.

Chapter 5

Implementations

This chapter sets out to describe the implementation of machine learning and deep learning algorithms for RFI flagging in time/frequency data. It begins by showing the investigations which were designed and described in the previous chapter. These include experiments into reproducing the methods performed in literature. These investigations then progress into attempts at improving these methods for the specific datasets used in this work.

5.1 Scope and Limitations

The data to be used in these implementations is from two distinct sources, simulated HERA data and real KAT-7 data.

The KAT-7 data is full polarized and calibrated. It has been flagged to the best degree possible through a combination of AOFlagger and manual flagging. The pre-calibration data is not available.

The HERA simulated dataset is created to be the same shape as the real data in an attempt to make comparisons more likely to be relatable. Only one polarization is available at a time, the XX polarization is chosen to be used.

Owing to the limitation on the simulated data's polarizations, when performance is compared between the two - the real data will be limited to a single polarization.

5.2 Reproducing Previous Works

This section describes the processes that were preformed when attempting to reproduce results from literature. It initially investigates the machine learning methods and then goes on to demonstrate the results of specific deep learning methods. For both subsections the results for real and simulated data are shown.

5.2.1 Machine Learning Methods

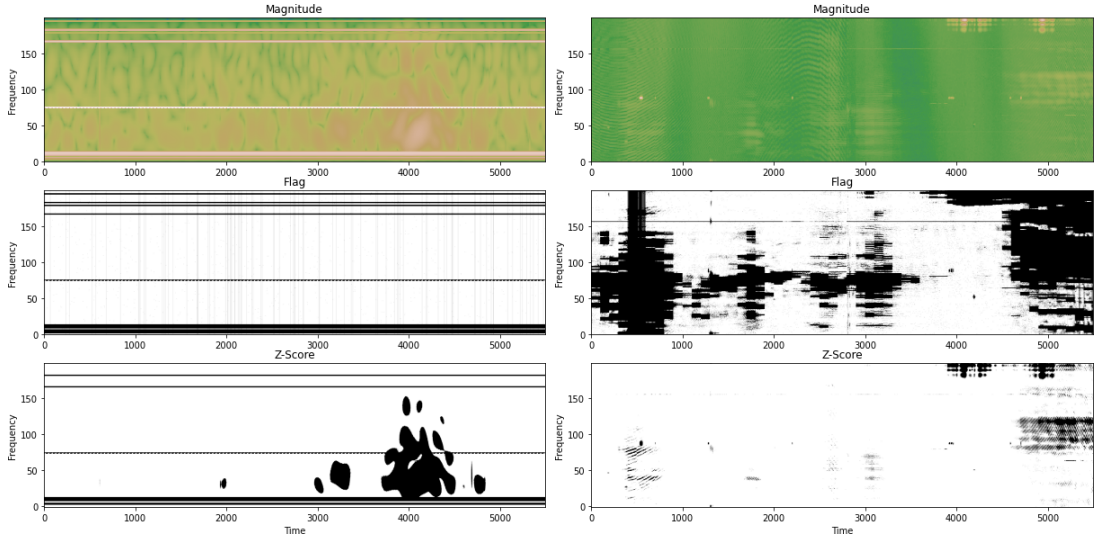
The machine learning methods used in *Mosiane, et al. (2017).* are initially implemented by applying specific pre-processing steps, the machine learning classifiers are then applied with assumed parameters which are maintained between experiments.[33]

Pre-processing is done initially by flattening the magnitude data into a one dimensional array. It is assumed a single polarization is taken, in this case XX. For both datasets this amounts to $21 baselines \times 5500 timesteps \times 200 frequency\ bins = 23\ 100\ 000 samples$.

Normalization is then performed on this data as per the literature by extracting outliers with the modified Z-Score given in Equation 5.1. The outliers identified by the modified Z-Score are shown in comparison to the magnitude and flag plots in Figure 5.1. Both subplots demonstrate that the modified Z-Score does identify some flag samples, namely the narrow-band persistent RFI in the simulated data. The modified Z-Score outliers also include a portion of the fridge in the simulated dataset.

The goal of extracting the datasets outliers prior to normalization is to allow the low-level RFI, which was not identified by the modified Z-Score, to be further amplified - for example the broad-band bursts shown in 5.1a. Evaluating the accuracy of the modified Z-Score as if it were a flagging method is shown in Table I. This can be seen as a measure demonstrating the differences between the levels of RFI present between the two datasets. While there are more flags in the real data, they do not present as such high value outliers as the flags in the simulated data do.

The precision in the simulated data is much lower as there is a large amount of non-RFI data identified as RFI, while in the real data a high proportion of RFI identified by the modified Z-Score is actual RFI. The opposite is true for the resulting recall. For the real data there are far fewer modified Z-Score flags identified than actual flags, unlike



(a) Simulated data showing the magnitude of a single polarization compared to its flag and the outliers identified by the modified Z-Score.

(b) Real data showing the magnitude of a single polarization compared to its flag and the outliers identified by the modified Z-Score

Figure 5.1: A comparison between the real and simulated data used in these investigations. These plots demonstrate the outliers identified by the modified Z-Score which is to be used in data normalization as per the literature.[33]

the simulated data which correctly identifies a higher proportion of the narrowband RFI.

Table I: Accuracy metrics between real and simulated data. [33]

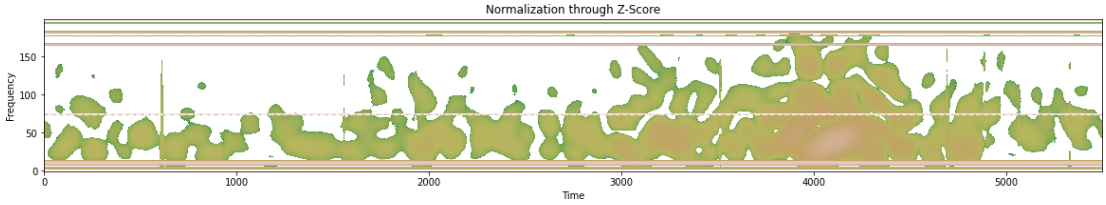
Dataset	Precision	Recall	F1-Score
Simulated data	0.55	0.56	0.55
Real data	0.87	0.17	0.28

It is assumed normalization in this case refers to the act of feature scaling or standardizing. While many forms of this exist, the Z-Score is used in this instance. The Z-Score must be differentiated from the *modified* Z-Score used previously. Shown in Equation 5.1 the Z-Score for normalization is performed on every sample by subtracting the mean and dividing by the standard deviation.

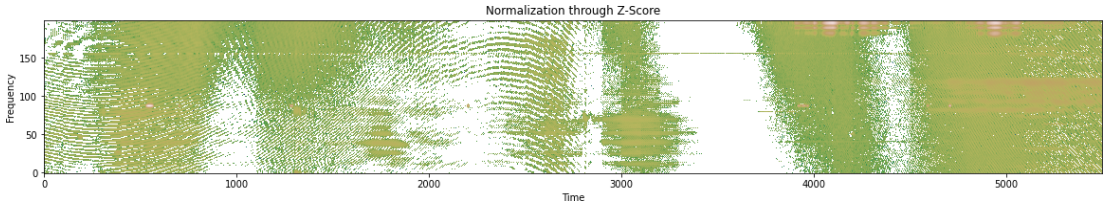
$$z = \frac{x - \mu}{\sigma} \tag{5.1}$$

The effect of normalizing the dataset after removing the modified Z-Score outliers in this manner has is demonstrated in Figure 5.2. These plots show the extent to which

the RFI, particularly in the simulated dataset, is been amplified. Yet a high proportion of the fringe alongside it.



(a) Simulated data showing the magnitude of a single polarization normalized after extracting the modified Z-Score outliers.



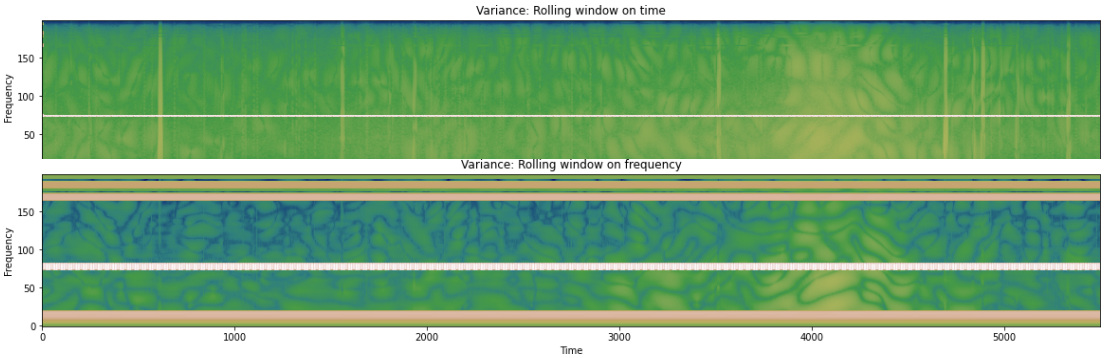
(b) Real data showing the magnitude of a single polarization normalized after extracting the modified Z-Score outliers

Figure 5.2: A comparison between the modified Z-Score normalized magnitudes of a baselines XX polarization for the simulated and real data. The goal was to increase the effect low-level RFI produces. There appears to be a few narrow band bursts visible in the simulated data. In general this effect has increased the visibilities internal structure.

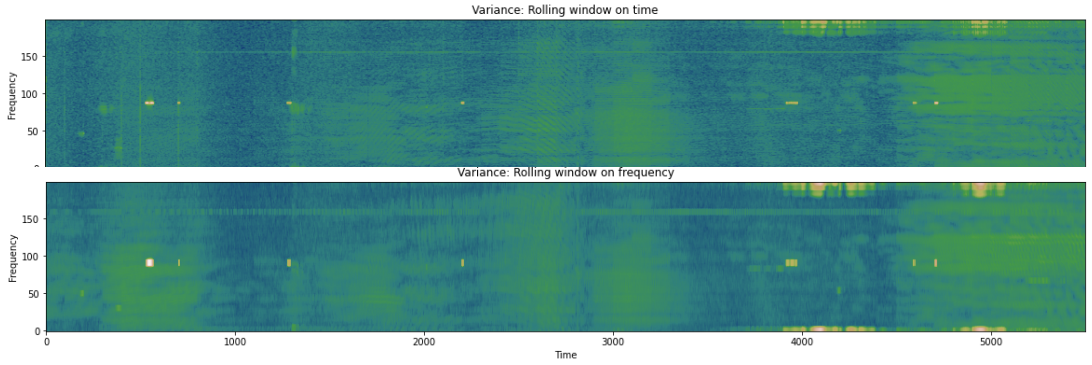
The feature extraction stage is performed with a one dimensional rolling window. The list of features is given by the items in 3.3.2.1. As the data is flattened into a one dimensional array there exist two methods for doing so, stacking by time or by frequency. The effect a rolling window has in extracting the variance statistic in both of these directions is given in Figure 5.3. These plots also serve to demonstrate visually the difference an extracted feature like variance can present in the data. The most obvious being the simulated data rolling on frequency, which shows the internal structure of the sky model far more clearly but at the same time extending the narrow band persistent RFI in the frequency bin around 75.

The machine learning methods are applied with the following parameters, kept as a control between training and evaluation on the real and simulated data.

- K-Nearest Neighbours
  - Elbow method to evaluate K
- Naive Bayes
  - Gaussian Naive Bayes



(a) Simulated data showing the difference between a rolling window operating on time versus frequency directions.



(b) Real data showing the difference between a rolling window operating on time versus frequency directions.

Figure 5.3: Subfigures showing the extracted variance using a rolling window 8 samples long. These plots also serve to demonstrate the difference between how a rolling window will perform depending on the direction in which data is flattened. When the direction taken is in the frequency, as the window treats all samples with equal weighting, the narrowband RFI can grow much larger - taking up extra bands.

- Random Forest
  - 100 estimators

The results of these evaluations are given in Table II. The results are as a whole are much lower than expected when compared to the related literature. This may be due in part to the difference in data, where these data sources represent largely imbalanced classes with far more non-RFI than RFI data. Without mechanism for combating this it may play a role in future experiments. The order of accuracy between methods is maintained from the source literature; RFC is the highest performing followed by K-NN and finally NB. With the RFC performing well on the simulated data in particular with a high AUC of 0.87 and NB performing very poorly at identifying RFI with a recall of 0.09 on the simulated data.

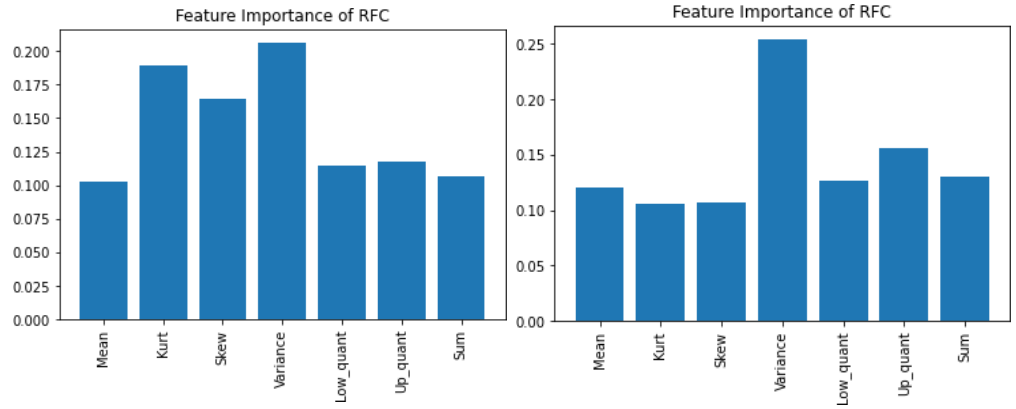
Table II: Accuracy metrics of each machine learning classifier on both datasets.

Classifier	Dataset	AUC	Precision	Recall	F1-Score
K-Nearest Neighbour	Simulated	0.72	0.83	0.45	0.58
	Real	0.66	0.57	0.36	0.44
Naive Bayes	Simulated	0.54	0.52	0.09	0.16
	Real	0.59	0.78	0.19	0.30
Random Forest	Simulated	0.87	0.95	0.76	0.84
	Real	0.66	0.74	0.33	0.46

Another metric of interest from these results is the ability for the RFC to generate a measure of "feature importance". Owing to the nature of how features function in a decision tree this can be generalized as the ability for each feature in distinguishing the class. The feature importance plots for each dataset are shown in Figure 5.4.

In the the real data tests, each feature aside from variance seem to have equal weighting. While for the simulated data, variance again shows prevalence yet kurtosis and skew also show a higher importance. The large magnitude difference between RFI and non-RFI likely leads to variance playing an important role in identifying the classes.

Literature has also shown the use of spectral kurtosis in identifying RFI in various applications.[39] The rolling window extraction technique in the frequency direction, implementing a kurtosis feature extraction method is visualized in Figure 5.5.



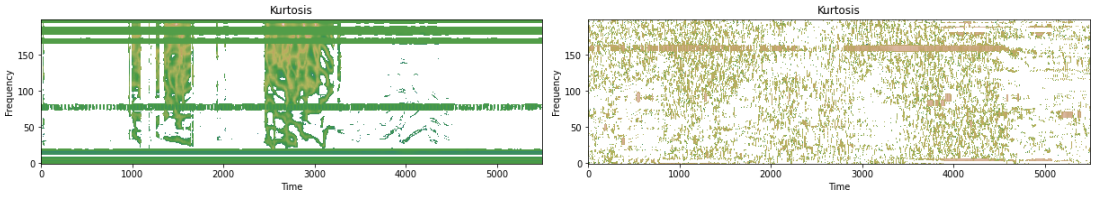
(a) Simulated data extracted feature importance from RFC

(b) Real data extracted feature importance from RFC

Figure 5.4: The feature importance calculated from the RFC result for each dataset. It indicates in each dataset that variance is a good discriminating factor in order to identify the classes. For the simulated data the kurtosis and skew also demonstrate high importance. These effects may be due to a variety of factors, the differences can most likely be attributed to the simulated HERA sky model's inherent distribution.

These plots show the underlying structure of the narrowband RFI clearly. This effect is more drastic in the simulated data, which also contains a higher amount of narrowband persistent RFI. The number of frequency samples is exaggerated, as demonstrated previously in the Figure 5.3 where a rolling window on frequency has the effect of elongation on the high magnitude samples across frequency bins.

While this effect may cause a higher number of false-positives or over-flagging to occur this is generally preferred in the task of RFI flagging to ensure no corrupted data may be propagated through the imaging pipeline.



(a) Simulated data showing the extracted kurtosis feature generated with a rolling window.

(b) Real data showing the extracted kurtosis feature generated with a rolling window.

Figure 5.5: Time/frequency plots of the extracted kurtosis feature from both datasets. Visually they are very distinct. The simulated data clearly displays the narrowband RFI while the real dataset tends more towards some narrowband being accentuated but mostly appears to be unstructured noise.

5.2.2 Deep Learning Methods

The deep learning method chosen for modelling is the U-Net architecture. This technique has been used for RFI detection in multiple sources.[30][11] The work of *Akeret, et al (2017)* uses the architecture as it was originally designed and this is attempted to be reproduced in this subsection.

As each visibility is size 5500×200, pre-processing is required for sub-dividing sections of the spectrogram. Each subset is generated as shape 100×100, dividing a visibility into 110 square "images" for each baseline for a total of 2310 subsets. Defining a U-Net architecture in Keras makes use of concatenation operations between the features in the down-sampling and up-sampling sections. Due to this fact only shapes multiples of x^2 can be used. To overcome this a zero padding operation is included from the input layer, reshaping the input to 128×128. A crop operation is used before the final 1×1 convolution to ensure the output shape matches the input.

These subsets are stored as HDF5 files and loaded on-the-fly during training using a custom data generator. Each subset's position during training is randomized with respect to their position in the baseline in an attempt to reduce over-fitting. Test, training and validation data is divided as a 70/15/15 split respectively and during training the positions are shuffled each epoch to prevent over fitting.

Recreating the parameters outlined in the relevant literature is done through the following means:

- Optimizer: Stochastic gradient descent
- Learning rate: 0.0001
- Learning rate exponential decay
- Epochs: 100
- Dropout: 0.5
- Regularization: L2 at 10^{-3}

The U-Net is to be generated with 3 layers and 64 filters, demonstrated in the relevant literature as the optimum architecture, and the defined given control parameters. During training it was apparent that the given learning rate of 0.2 was too high to

properly converge and required being lowered to 0.001 to initially begin training. The results on the test set are shown in Table III.

The high loss function for both sets indicates that the defined number of epochs at 100 is likely too low. This results in the low precision on both datasets. Recall can be considered an indication of the amount of correctly identified RFI. The recall is relatively high especially in the simulated results while the precision is generally low. This is an indication of over-flagging occurring, where an overwhelming amount of non-RFI samples are being incorrectly identified as RFI. This constitutes as a large number of false-positives.

Table III: Comparison of the given U-Net's accuracy metrics on both datasets.

Dataset	Precision	Recall	AUC	Loss
Simulated data	0.34	0.81	0.87	0.55
Real data	0.25	0.63	0.86	0.26

5.3 Expansion of Previous Works

This section explores mechanisms and techniques which serve as potential improvements upon the the previous implementations, specifically for the datasets used in this work. These attempts to improve the *"State of the art"* will function as a basis for the design and implementation of a unique contribution of work in the chapter to follow.

Primarily the investigations into improvements will revolve around pre-processing and feature extraction methods. The techniques implemented previously are easily extendible and it is intended that with further fine tuning, a higher accuracy can be achieved. The practical implications of these methods are unknown. With extremely large datasets in conventional radio astronomy, high complexity pre-processing methods are likely impossible. Due to this there exists a trade-off between the increased processing time and increased accuracy.

5.3.1 Feature Extraction Improvements

The feature extraction based on two dimensional filtering methods is defined and outlined in Section 4.5. The primary reasoning for performing feature extraction with a filter, is the ability for nearby samples to influence the center sample during processing.

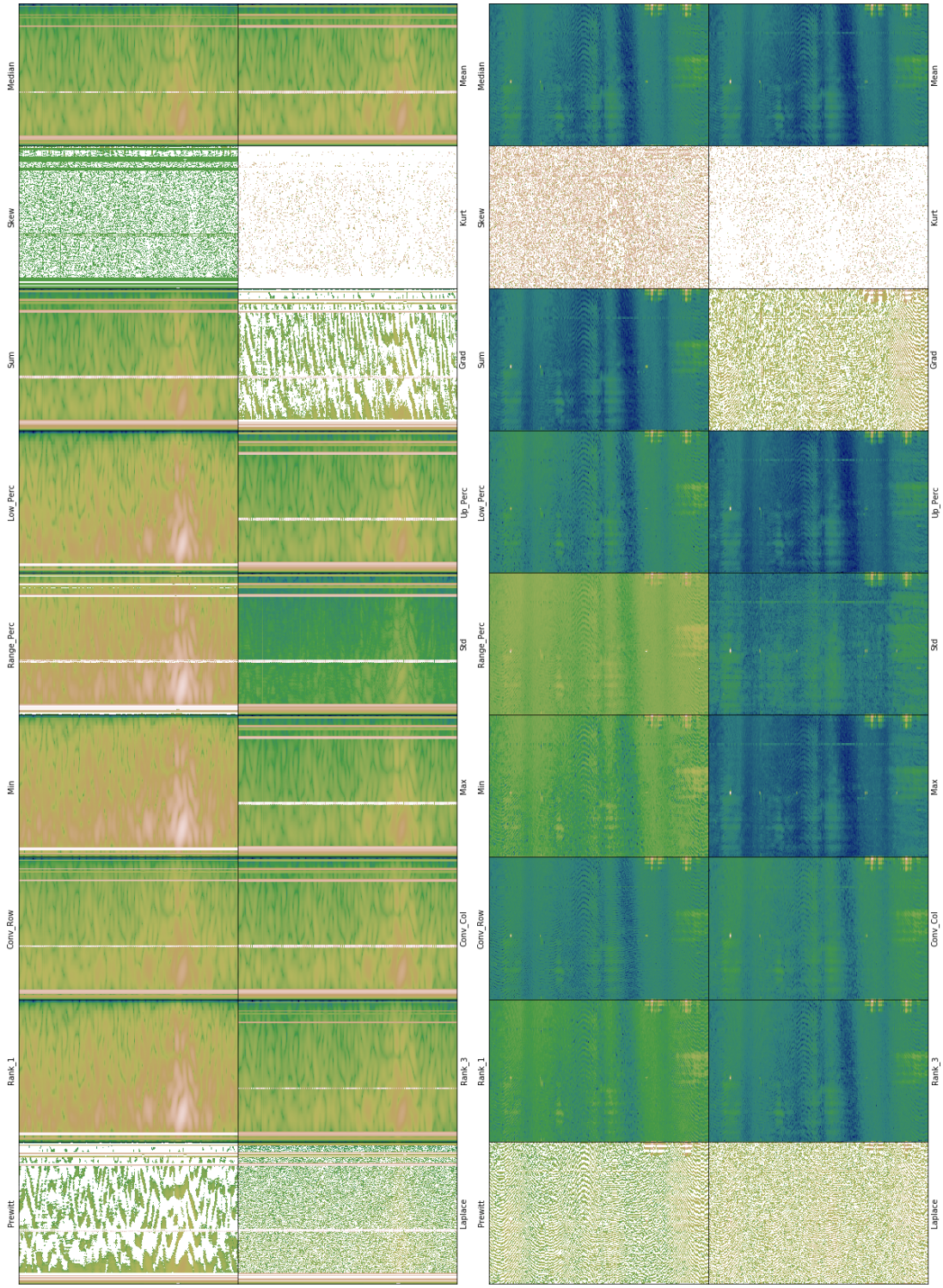
This allows each individual sample to maintain a semblance of its spatial relationship, as RFI is more likely to occur across multiple samples in the time or frequency direction. The disadvantage of a one dimensional sliding window is bias owing to its direction across time or frequency.

By applying a two dimensional filter of size 3×3 with a stride of 1 this spatial relationship present in narrowband or broadband persistent RFI is maintained. This may assist algorithms which treat each sample independently, which may only rely on a single samples features to identify its class. This feature construction method is similar to the function of a CNN filter. The difference is, where the features extracted by a CNN filter are learned, this method is constructing pre-defined features.

Implemented using the *scipy.ndimage* package the performance is highly optimized, yet still requires high processing time due to the vast amount of data and specific high complexity statistical algorithms. A visual demonstration of the resulting time/frequency data for each extracted feature for both datasets is shown in Figure 5.6. A visual distinction is clear for specific filters and the effect they have on both datasets, for example the gradient filter appears to amplify the underlying structure far more than others. A general observation is that the narrowband persistent RFI present in the simulated data is highly visible after these feature extractions, while for the real dataset only a few appear to present the same characteristic.

Investigations into how similar the extracted features are, is shown in Figure 5.7 by plotting the correlation between each individual feature for the simulated dataset. It is clear that not every feature is independent. With a correlation close to 1 between two features there is no practical advantage to keeping both as it would only lead to increased processing times. To this extent the following filters are identified as possessing strong linear correlations:

- Rank 1
- Convolution by column (vertical kernel)
- Minimum



(a) Simulated data showing all the extracted features generated by two dimensional filters.

(b) Real data showing all the extracted features generated by two dimensional filters.

Figure 5.6: Magnitude plots of a single baseline for each of the total number of extracted features across both datasets. Features are manually extracted using two dimensional rolling filters of size 3×3 with a stride of 1. It clear that some plots appear extremely similar with only minor differences between them.

- Mean
- Lower percentile
- Standard deviation

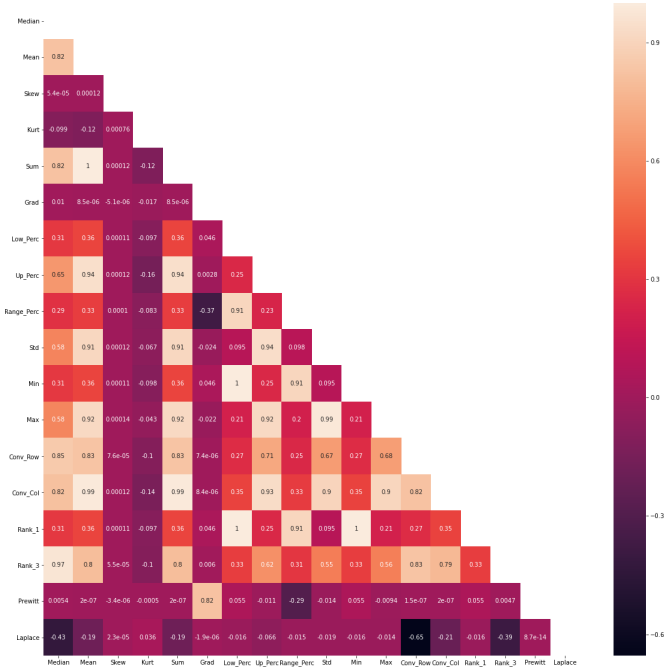


Figure 5.7: All Pearson-correlation coefficients compared between each feature for the simulated dataset. It can be seen that some features like *Mean* strongly correlate to other features. *Rank 1* also proves to have a total positive linear correlation to two other features, identifying this as an unnecessary versus redundant feature is required. It is likely the use of correlated features would not contribute to a learners ability to distinguish classes. Reducing the number of features will also reduce complexity and in turn training/inference time.

The simulated dataset is created with complex values making it possible to compute these constructed features on both magnitude and phase. It is chosen to initially only construct 18 features based on the magnitude samples of each baseline. This is done for a more direct comparison between the existing methods computed with a one dimensional filter on only the magnitudes.

For the fully polarized real dataset the final feature vectors are constructed by computing the filters for each polarization's magnitude and phase as well as 8 features originating from the computed Stokes visibilties. With 18 filters in total this amounts to $4 \times 18 + 4 \times 18 + 8 = 152$. With the total of 21 baselines at 5500×200 samples the resulting sample/feature matrix is 23100000×152.

As the previous results for each learning algorithm are substantially higher for the simulated dataset compared to the real dataset, an investigation in intended into whether constructing as many features as possible and selecting the features with the highest importance will improve the accuracy of the real dataset.

5.3.2 Feature Selection

The simulated data posses 36 features extracted using the 18 filters on the magnitude and phase of the single XX polarization. The constructed features for the real dataset contain 152 features extracted using the 18 filters on each polarization's magnitude and phase samples, as well as the 8 Stokes visibilities. In order to reduce the size of the data, feature selection is performed. This is intended to identify which features prove to be the most effective at identifying RFI while reducing the complexity of the model. This is akin to reducing a dataset without a loss of significant information.

Manual feature selection can be performed through identifying features with high correlations to each other, as shown in Figure 5.7, yet this method can prove to be time consuming for extremely large datasets and can have significant disadvantages to statistical methods. There is a major difference to be noted between *redundant* and *irrelevant* features. An irrelevant feature may show strong correlation to another feature of high importance, and determining the former to be redundant is vital.

Feature selection is performed through the use of the χ^2 (Chi-squared) test between each feature and their respective class. Utilizing the χ^2 test for feature selection is performed by first generating a contingency table between each feature and class. The χ^2 statistic is then computed against the expected frequency from each distribution over the classes and a uniform distribution over the features. To this extent if the χ^2 null hypothesis is rejected the features represent dependent variables, and those features independent of class are irrelevant for classification.

The top 10 resulting features from the χ^2 test on the simulated and real data are given in Table IV.

These results when compared to the feature importance plots generated by the RFC in Figure 5.4 demonstrate that variance/standard deviation is likely the strongest feature for learning algorithms. Yet is it noted that while kurtosis and skew are shown to benefit the RFC they are both missing from the top 15 features across both datasets.

Table IV: χ^2 test results for selection on real and simulated constructed features.

Ranking	Simulated Dataset	Real Dataset
1	Phase_Std	YY_Phase_Std
2	Mag_Min	XY_Mag_Min
3	Mag_Rank_1	YX_Mag_Rank_3
4	Mag_Low_Perc	YX_Mag_Median
5	Mag_Median	YY_Mag_Low_Perc
6	Mag_Conv_Col	YX_Mag_Min
7	Mag_Conv_Row	XY_Mag_Rank_1
8	Mag_Mean	XY_Mag_Low_Perc
9	Mag_Sum	YY_Mag_Rank_1
10	Mag_Rank_3	YY_Mag_Min
11	Mag_Up_Perc	YX_Mag_Rank_1
12	Mag_Max	YX_Mag_Low_Perc

For the simulated dataset the kurtosis filter for magnitude and phase are at index 18 and 20 respectively, while both skews are in the bottom 5.

It is also of interest that the standard deviation of phase is the variable most dependent on class according to the χ^2 test, yet no other phase measurement appears in the top 10. It is noted that literature has shown that the inclusion of phase information, specifically in a convolutional neural network, demonstrated an improvement in accuracy when the data was not calibrated - while both of these datasets under test are calibrated.[11]

5.3.3 Machine Learning Methods with New Features

To assess the changes in the datasets respective features the ML methods are repeated along with attempts to improve their selected parameters. It is hypothesised that the features represent more information that a learning method may use to improve its predictive capabilities.

5.3.3.1 Simulated Data

Experiments are carried out into the simulated dataset with 36 constructed features available. These are as follows:

- Test 1: All 36 features
- Test 2: 18 features, magnitude only
- Test 3: χ^2 test selected features
- Test 4: Top 3 χ^2 selected features

Initially all 36 features are utilized. Literature has shown that phase measurements may demonstrate improved results when included in CNN techniques so this is investigated in the ML techniques to uncover if this may be an underlying relationship.[11] The feature selection technique performed with the χ^2 test is then investigated, this is done in order to investigate how relevant these reduced features will be going forward. The motivation behind the final test is related to this, in investigating how reducing the dataset even further may affect the accuracy.

The resulting AUC, Precision, Recall and F1-Score for each of the four tests are shown in Table V. Noticeably the differences between the first three tests are negligible, yet the complexity of each of them is greatly reduced with Test 3 representing only 33% of the first tests data. The results do decrease in Test 4 with only 3 features being used.

When compared to the same tests done on the one dimensional sliding window features the results are over-all much higher. These results are shown in Table II where the highest AUC achieved was with the RFC at 0.87, this has now increased to 0.94 in Test 3 with the top 12 features. Each tests demonstrates an improvement over the features constructed with a one dimensional filter. The results for K-NN show a huge improvement compared to the original experiments, and while the NB classifier has improved its accuracy is still considerably lower.

The feature importance for Test 1 is shown in Figure 5.8. This plot demonstrates considerable differences to the features identified in the χ^2 test. While the criteria for considering a feature as important is different between these two methods it is noted that they seem to have almost opposite results. While the selected features in this figure show low feature importance in this particular classifier, the RFC generated with the selected features is still able to achieve high accuracy.

Table V: Results between simulated data ML tests.

	Classifier	AUC	Precision	Recall	F1-Score
Test 1	K-NN	0.93	0.99	0.86	0.92
	NB	0.70	0.64	0.47	0.54
	RFC	0.93	0.99	0.86	0.92
Test 2	K-NN	0.93	0.99	0.86	0.92
	NB	0.71	0.69	0.45	0.55
	RFC	0.93	0.99	0.87	0.93
Test 3	K-NN	0.91	0.97	0.83	0.90
	NB	0.62	0.56	0.27	0.37
	RFC	0.94	0.99	0.87	0.93
Test 4	K-NN	0.88	0.89	0.78	0.84
	NB	0.63	0.65	0.28	0.39
	RFC	0.84	0.90	0.68	0.78

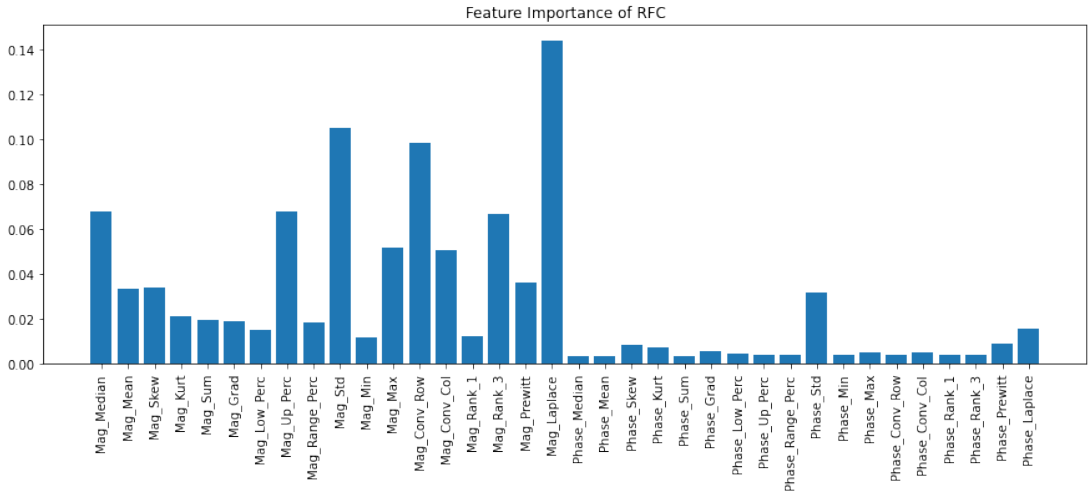


Figure 5.8: Feature importance generated by the RFC for each constructed feature in the simulated dataset. The features generated from the phase information prove to be considerably lower than those of magnitude. Compared to the top selected features there are significant differences. While the standard deviation of phase does show a high importance when compared to the other phase features it was calculated to be the feature with the highest linear dependence for each sample relative to their class. The *minimum* filter of magnitude is given little feature importance for the RFC and conversely the Laplace filter the highest - yet the opposite is true for the χ^2 tests identified features.

5.3.3.2 Real Data

The real dataset contains 152 constructed features. Utilizing each of these features for experimentation proves challenging from not only a complexity standpoint but also memory requirements. To overcome this it is chosen to make use of only the top 12 selected features, as performed in Test 3 of the simulated dataset experiments.

These results are demonstrated in Table VI. The results are a far higher improvement, with ≈ 0.3 improvement for each F1-Score and ≈ 0.15 improvement for each AUC. It is noted that again the NB classifier on the real dataset performs with higher accuracy relative to the simulated dataset. This is likely due to the statistical differences in how the non-RFI data is distributed.

The feature importance plot for the RFC is omitted as it displays nothing of significance. The selected features are relatively well distributed, with no particular feature drastically outperforming the rest. Yet the relationship remains where the order of selected χ^2 features is not maintained by the RFC owing to the different criteria in their importance metrics.

Table VI: Results for real data with selected features.

Classifier	AUC	Precision	Recall	F1-Score
K-NN	0.80	0.77	0.64	0.70
NB	0.81	0.78	0.65	0.72
RFC	0.82	0.85	0.65	0.74

5.3.4 U-Net Improvements

The application of the U-Net is again performed on 100×100 sub-regions of each baseline, treating the identification of RFI as an image segmentation task. Investigations are performed into attempts at improving the architecture from its original design as well as with the addition of the newly constructed features. As with previous investigations it is necessary to add zero padding and crop layers to the start and end of the network to deal with the irregular image size. The pre-processing of data is maintained, with the use of HDF5 files and a custom data generator to produce batches.

In each investigation the optimizer used will be *adam* with binary cross entropy

as the loss function. Generally SGD is considered to be more accurate with well constructed learning rates and momentum, but this proves time consuming and the adam optimizer generally reaches lower training loss earlier with little supervision. During training the call-backs applied are a learning rate reduction on plateau and early stopping. Reducing the learning rate when the validation loss has begun to plateau can be crucial in ensuring the best possible training. Learning rate decay simply as a function of the number of epochs does not take into account the current loss function. Early stopping is used to prevent overfitting to the training data. While training the validation loss may reach a point where it begins to gradually increase as the model is fine-tuning to the specific training data its receiving, stopping training when this point is reached should produce a model with less overfitting. This is the reason the dataset is split into 70/15/15 training/validation/testing subsets. If the model is updated according to the results of the validation set, assessing its accuracy on this same set would be disingenuous to a true reflection of an 'unseen' set.

The following tests are performed:

- Test 1: Using magnitude data only with new model parameters
  - Batch normalization
  - Convolutional transpose
  - Number of filters: 128, 64, 32
- Test 2: Full feature datasets
- Test 3: Top 12 selected features
- Test 4: Only real data magnitudes
  - Fully polarized magnitudes
  - Fully polarized magnitudes and phase
  - Fully polarized magnitudes and phase and Stokes magnitude and phase
  - Fully polarized magnitudes and Stokes magnitude
- Test 5: Pre-trained ResNet backbone

5.3.4.1 Test 1 - Magnitude only

The goal of batch normalization is to attempt to reduce internal covariate shift in a deep network. While this is still a debated topic and alternate hypotheses have been proposed as to how it function, reducing internal covariate shift was its initial application. This occurs during training where small changes to parameters in earlier layers can be amplified as they propagate to later layers, causing each subsequent layer to adjust to new input distributions. Normalizing each layers input by re-centering and re-scaling them can make the training of deep networks faster and more stable.[40]

The U-Net architecture functions as an encoder/decoder. The encoder's layers seek to reduce the tensor shape through down-sampling with a max pooling operation, and the original architecture uses an up-sampling operation in the decoder. This is simply a resizing, applying interpolation to calculate new pixels. The goal of using a convolutional transpose operation is to introduce a new learnt parameter which best reconstructs the new pixels with kernels.

The number of filters for each layer of the U-Net is based on the initial convolutional layer. In the previous investigation this was fixed at 64 filters, which is doubled after each max pooling operation until the shared layer between encoder/decoder. The number of filters will determine the depth of the extracted information, but will have an optimum number as too many would be detrimental to its learning.

The results for each iteration of the number of filters for both datasets (only XX magnitude) is shown Table VII. Is is obvious again that the simulated data proves to be much easier to identify to the real dataset. The number of filters appears to have no impact on the simulated dataset and they all appear to converge on the same solution, while the recall for 64 filters on the real dataset does show a slight improvement. The recall for the simulated data is extremely low for this application. Recall equates to the number of correctly identified RFI samples out of all the samples identified as RFI, for imaging it is crucial that *all* RFI is removed as any high magnitude samples will corrupt any further data processing.

5.3.4.2 Test 2 - Full feature datasets

This tests investigates the application of the full 152 feature real dataset and the 36 feature simulated dataset. These constructed features were applied to both magnitude

Table VII: Results for new U-Net model parameters.

Filter #	Dataset	AUC	Precision	Recall	F1-Score
128	Sim	0.93	0.98	0.85	0.91
	Real	0.89	0.95	0.34	0.50
64	Sim	0.92	0.99	0.85	0.91
	Real	0.93	0.93	0.46	0.61
32	Sim	0.93	0.99	0.85	0.91
	Real	0.91	0.97	0.32	0.48

and phase and while literature has shown the inclusion of phase information for a customized U-Net architecture demonstrated no conclusive improvement it is possible the feature construction methods applied to the phase information may amount to increased performance.

These results are given in Table VIII which show almost similar results for both datasets that were achieved by the RFC algorithm when trained on the top 12 features. The precision on the simulated dataset shows almost no false-positives are predicted, indicating a very high ability to identify non-RFI events.

While the simulated data shows no changes when comparing Test 1 on only magnitude data, the inclusion of features greatly improved the accuracy in training on the real dataset.

Table VIII: Results for the U-Net on all constructed features.

Dataset	AUC	Precision	Recall	F1-Score
Sim	0.92	0.99	0.84	0.90
Real	0.97	0.91	0.67	0.77

5.3.4.3 Test 3 - Top selected features

The top 12 features for each dataset were selected with the χ^2 test and proved to be valuable to improving the accuracy of the machine learning classifiers. For convolutional networks this is not necessarily true. The filtering method used in the statistical feature construction is akin to a convolutional filter with a kernel size of 3 and a stride of 1. The

major difference being the fact that the statistical features extracted are pre-defined, while for a CNN these features are learned. This feature learning may prove to be more valuable from the original data source.

The results for Test 3 are given in Table IX and while the simulated data shows minimal changes to using only the magnitude data in Test 1 the results for the real dataset are greatly improved. This is likely due to the additional information gained by using all four polarizations for the constructed features. For the single magnitude feature input to the first convolutional layer as in Test 1 each convolutional feature map is constructed using only this available information, which is then propagated down the network. For a sample input with multiple feature channels the convolutional kernel operates on each input channel individually, and each convolutional feature map is summation of these operations. With more information available through the input features is seems an intuitive step that the model is able to construct a better decision and therefore have higher accuracy as each convolutional feature map is learned through iterating on each input feature channel.

Table IX: Results for the U-Net on selected features.

Dataset	AUC	Precision	Recall	F1-Score
Sim	0.92	0.99	0.84	0.90
Real	0.97	0.93	0.67	0.78

5.3.4.4 Test 4 - Magnitude and phase of real data

As Test 2 and 3 investigate the application of the selected constructed features it is necessary to identify whether these constructed features offer any benefit in a convolutional network at all. As described in the test, these constructed features function similarly to how a convolutional network actually learns. This is best investigated through the fully polarized real dataset. Therefore it is imperative to investigate whether there is any benefit to the additional pre-processing by testing only the fully polarized information.

Along with this the use of Stokes visibilities are investigated. While these 8 feature vectors are a part of the 152 constructed features they are not represented in the top 12 selected features. If the additional information gained through the statistical feature construction proves to be of no benefit or less than beneficial it is important to identify

if the transformation to Stokes parameters can prove beneficial. This transformation represents a combination of the original complex vectors which are subsequently transformed themselves into magnitude and phase vectors. If any information is lost through this conversion which is maintained through the Stokes parameters this should reflect in the resulting accuracy metrics.

Table X demonstrates the results into these investigations. The AUC for each test is identical, meaning the ability for the model to distinguish between classes is relatively the same, which is a fairly skewed metric in such a unbalanced dataset. But demonstrates that in each test the classes are separating fairly evenly, only the amount of RFI vs non-RFI identified is shifting between the iterations. This is also shown where it appears that the inclusion of phase information results in a favouring of recall over precision as seen in row two and three of the table. This would imply that with the available phase vectors the model is correctly labelling a higher proportion of RFI yet with the decreased precision there is a higher proportion of non-RFI samples being identified as false-positives.

The inclusion of the F1-Score in these results show how the overall accuracy is the same between the first three tests. But the proportion of RFI correctly identified shifts depending on the features used. By this logic the inclusion of all polarizations magnitude and phase and the Stokes parameters achieves the best results. The hypothesis that the Stokes features provide additional information which may have been lost from the original complex data does not seem to apply when viewing these results, but far more testing would be required into this result for a definitive conclusion to be drawn.

Table X: Results for the U-Net on fully polarized real data.

Subset of fully polarized real data	AUC	Precision	Recall	F1-Score
Magnitudes	0.98	0.92	0.73	0.81
Magnitudes and phase	0.98	0.86	0.76	0.81
Magnitudes and phase and Stokes	0.98	0.87	0.77	0.81
Magnitudes and Stokes magnitudes	0.98	0.95	0.67	0.78

5.3.4.5 Test 5 - ResNet pre-trained

An investigation into the top 12 selected features applied to re-training the pre-trained weights with a different architecture is performed. The residual neural network (ResNet) architecture utilizing skip connections has been demonstrated to be state of the art (SOTA) in many domains. The use of skip connections is to provide deeper layers the activations from former layers, simplifying and speeding up the network but primarily reducing the impact of vanishing gradients. While a network without skip connections has the ability to access and construct more of the feature space, small changes in weight updates can produce the vanishing gradient problem where the gradient can be *vanishingly* small - preventing an update as it is proportional to the partial derivative of the error function and the weight.

In this test the ResNet34 architecture is used as an encoder for the U-Net. This ResNet with 34 layers is far deeper than the number of layers in the conventional U-Net approach, but has also been pre-trained on the imagenet dataset. The motivation behind this, is that convolutional feature maps posses similar characteristics between applications, identifying vertical and horizontal lines for example. Using pre-trained models for image recognition and segmentation tasks is a common technique, and its application to a task such as RFI identification is relevant.

As the generated square data set is 100×100 the model is again constructed with a zero-padding layer after the input, and a crop operation after the final convolution. This allows the loss function to compute on each reconstructed predicted pixel to their original position.

A possible limitation of this pre-trained model is that the imagenet dataset is made up of RGB images, and the model input is therefore 3 channels. This requires the addition of an initial convolutional layer with 3 filters. Meaning the 12 constructed features which are input to the model are being contracted to 3 immediately. This is converse to the U-Net architecture whereby the initial convolutional layer has a far high number of filters.

The results in Table XI show little improvement over the results for Test 2 on the same dataset. While for the real dataset the precision has increased the recall has decreased by over 0.2. These results seems to indicate the application of a pre-trained model would have little impact in the identification of RFI. While more investigations

would be required to draw conclusions, such as a pre-trained weights from other datasets or ResNet with 50 or 101 layers. These results tentatively indicate that the convolutional feature maps from imagenet offer no drastic improvements when re-trained with these datasests for RFI identification.

Table XI: Results for the pre-trained ResNet.

Dataset	AUC	Precision	Recall	F1-Score
Sim	0.92	0.99	0.84	0.90
Real	0.90	0.99	0.45	0.62

Chapter 6

Multilayer Perceptron Network (MLP)

This chapter explores the creation and optimization of a fully connected dense neural network or a *vanilla* neural network, referred to here as a multilayer perceptron (MLP). This ANN will be operating on separate features of both the real and simulated datasets. Experiments are to be conducted into identifying architectures and implementations which seek to maximize the predictive accuracy of a more simple neural network. Investigations into the learning capabilities of an MLP are to be demonstrated using results from the previous chapter in the application of machine learning methods and the U-Net CNN architecture on the constructed features.

6.1 Goals and Limitations

The goals of this chapter are to investigate the application of a fully connected dense network applied to the domain of RFI prediction in time/frequency data. To this extent it will be trained and tested on a variety of combinations of the simulated and real datasets in order to assess which of the constructed features prove to achieve the highest accuracy.

The goals for this chapter are as follows:

- Train a simple single hidden layer FCN on the magnitude spectra from the real and simulated dataset
- Design and optimize an FCN based on the real dataset

- Investigate an optimized network on the full number of constructed features from both datasets
- Investigate an optimized network on the top selected features

6.2 Design of Two MLP Models

Initially a multilayer perceptron model is proposed in which each time/frequency sample of a spectrogram is treated individually. This will simply be called the *Feature MLP* in this section. In this implementation each time/frequency sample is only represented by its relative features. The loss function is therefore computed with the single output neuron representing that specific samples class, RFI or non-RFI. This method will utilize the constructed features.

The goal of these investigations is to identify whether pre-processing using two dimensional filters can improve a models performance, as this method can preserve the spatial relationship between nearby samples - which would otherwise be lost when treating each sample independently.

The second approach, which is only briefly investigated, is inspired by MLP's use in image recognition tasks. In this case a subset of a visibility is flattened from a two dimensional representation into a one dimensional array to function as an input to the model. This will be referred to as the *Flat MLP* as each input is an $M \times M$ subset of the data. This model has the disadvantage of only being trainable on a single feature, as each input neuron is responsible for a single sample. The output in this case has the same number of neurons as the input, where each sigmoidal output neuron is represented the probabilistic prediction of the Boolean flag mask.

An example of a similar approach is shown in Figure 6.1 which depicts an image of a handwritten digit from the famous *mnist* dataset. In this example each pixel is connected to each input neuron, requiring $M \times N$ input neurons for image height M and width N. The difference being that in a example implementation such as this, the 10 output neurons correspond to each digit's class, having one-hot encoded each digit as a vector of this shape.

Applying this implementation directly would prove impractical as the visibilities in the sample data are 5500×200. Requiring a huge network size and increasing complexity drastically. This also has the implicit quality of assuming that each input

pixel is related, while in reality there is little to no relationship between pixels on opposite sides of the image.

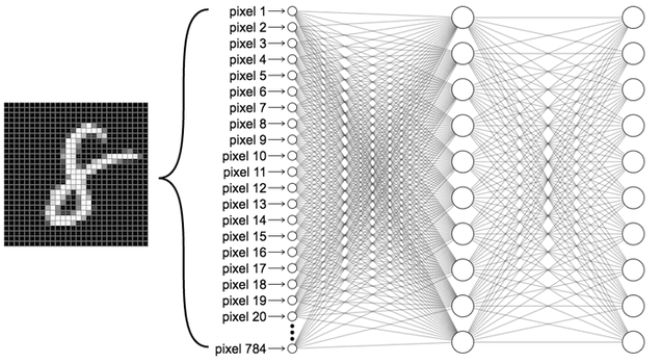


Figure 6.1: Example of a dense layer connecting to each input of an image. This is not how the proposed MLP functions, instead each sample is treated individually with the number of input neurons corresponding to the number of features for that sample.[41]

6.3 Implementation of the Feature MLP

This section outlines various tests into the implementation of the Feature MLP architecture. This architecture has the number of inputs equal to the number of features for a sample, with a single output node for that samples class.

It begins with initial tests into how the network performs on a single magnitude feature. With simple a trial and error approach to determine a functioning internal structure. It goes on to optimize the number of layers and neurons through a broad search. Different combinations of the constructed features are then investigated and finally an optimized architecture is then trained with a weighted loss function.

In each test the activation function of hidden layers is maintained as a ReLU, $f(x) = x^+ = \max(0, x)$. The ReLU function is generally regarded as the safest and most stable activation function for hidden layers in modern deep learning. It maintains sparsity within the graph and reduces the likelihood of vanishing gradient. The output activations are fixed as a sigmoid, $S(x) = \frac{1}{1+e^{-x}}$, as binary classification is to be performed with the loss function kept as binary cross entropy. The adam optimizer is maintained across tests as it is widely accepted in research over SGD, while SGD may produce lower validation loss through proper selection of learning rate this proves time consuming for the vast number of tests.

While investigating combinations of these constants could prove beneficial, for example the LeakyReLU for hidden layers, a full investigation across this entire parameter space would prove to time consuming.

6.3.1 Initial Training with Simulated Magnitude Spectra

Initial trial and error tests are carried out using the XX magnitude from the simulated dataset as a single feature input to various model shapes. These tests function as simply an outline into the logic behind some of the intuition used when investigating the Feature MLP architecture.

The reason these trial and error tests are performed is to distinguish some baseline information into how different implementations of the architecture may function. As the task of separating high magnitude RFI samples in visibility data can be abstracted to a simple implementation of a thesholding algorithm. Majority of the the RFI samples function as outliers to the dataset and achieving a low accuracy with a highly complex model would be a demonstration of severely over-complicating the approach.

6.3.1.1 Single Perceptron Model

To investigate this a single perceptron model is investigated as a baseline, with a single hidden neuron with a ReLU activation function and sigmoidal output neuron. This network is trained on the magnitude of the simulated data with SGD is on a 70/15/15 training, test and validation split. It achieves 0.10 and 0.49 precision and recall respectively as the validation loss struggles to minimize or converge over ≈100 epochs. This is expected, as demonstrated previously in Figure 4.6 the decision boundary requires a complex non-linear function.

6.3.1.2 Single Hidden Layer Network

Increasing the number of neurons in a single hidden layer to 32 and the precision reaches 0.95 while recall 0.46. The network converges rapidly, almost within 2 epochs due to the low complexity of the cost function and high number of samples per epoch. Increasing the number of neurons in a single hidden layer to $64, 128, 256$ has no effect on the resulting metrics from evaluating the test dataset. This leads to the conclusion

that a single hidden layer on this subset of data is likely resulting in the same decision boundary.

Inspecting the activations for the 32 and 256 hidden neurons on random test sample reveals that only 12 and 113 are active in this graphs respectively. This is interesting to note that while the number of active neurons is roughly half the size of the hidden layer, they converge on the same decision boundary. This effect is likely due to the limiting factor being the sigmoidal output.

With an F1-Score of 0.62 this single hidden layer is functioning with an accuracy relative to the mean threshold statistic as demonstrated in Table II, these heuristic threshold tests were also performed on the real dataset which has proven to be a more complex problem to solve that the simulated data these networks were trained on.

6.3.1.3 Two hidden Layer Network

Training two hidden layers with varying combinations of neurons between, them proves to have no effect on the accuracy metrics. Tests are carried out into changing the number of neurons in two hidden layers between combinations of $16, 32, 64, 128, 256$ with both layers the same number of neurons. Increasing or decreasing the number of neurons in the subsequent layer has no effect on the result. Interestingly the precision and recall resulting from the validation set is identical to the test set, indicating low overfitting. Its relatively safe to say this decision boundary can not be overcome with the Feature MLP architecture utilizing only the magnitude as a single feature.

6.3.2 Optimization Based on Real Dataset

Where the previous investigations showed no changes resulting from varying the number of neurons and layers in a single or two hidden layer network when trained on a single magnitude feature from the simulated dataset, this is not the case for increasing the number of features. This is best demonstrated through the real dataset, where the magnitude of each of the four polarizations is represented as an independent feature with moderate correlations between them.

In order to discern a model structure which achieves highest accuracy on selected features, the number of neurons and the number of layers are varied across multiple iterations. These tests were performed previously by the author.[42]

These investigations will be carried out into a dataset of 16 features. These are not constructed through the statistical filtering method as will be performed in later tests. The 16 features comprise of the 4 polarizations of the real dataset, with each polarization's magnitude and phase constituting for 8 features and each Stokes magnitude and phase the remaining 8.

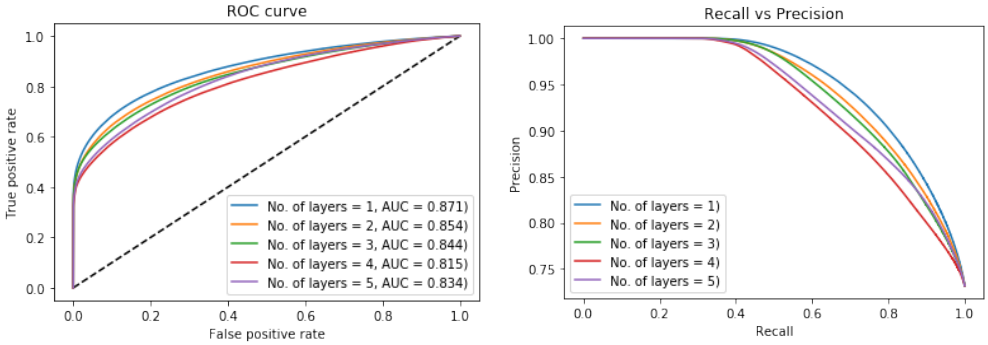
The evaluation metric for these investigations will be based on the ROC and precision recall curves for each iteration. Used primarily in binary classification it plots the true positive rate versus false positive rate as the discriminating threshold is varied. This threshold represents the point at which the probabilistic output of the sigmoid is classified as a binary value. Similarly the precision recall curves indicate the trade-off between the two metrics. This is relevant as in this paradigm, recall is deemed to be more important as it represents the models ability to correctly identify RFI samples, which far outweighs false-positives for incorrectly identified non-RFI samples.

6.3.2.1 Number of Layers

An MLP with a single hidden layer is said to be able to fit any non-linear function, each individual neuron in the hidden layers function as an independent perceptron and generate a linear decision boundary. For a single hidden layer the output layer can then aggregate the independent decision boundaries into non-linear separations. Yet increasing the number of layers allows for more complex non-linear features to be extracted, where each layer would represent higher complexity features.

Through trial and error is it found that when utilizing multiple hidden layers the number of neurons should be reduced in each subsequent layer. This may represent a means of reducing overfitting in the data by preventing an overwhelming number of high complexity features being generated.

The evaluation metrics for this experiment are shown in Figure 6.2. While the results do not show significant change between iterations, with a single layer the network is able to perform accurately. This may be in part due to the task of separating RFI not requiring highly complex features gained through additional layers, but may be due to the number of epochs tested over being enough for each network to reach convergence.



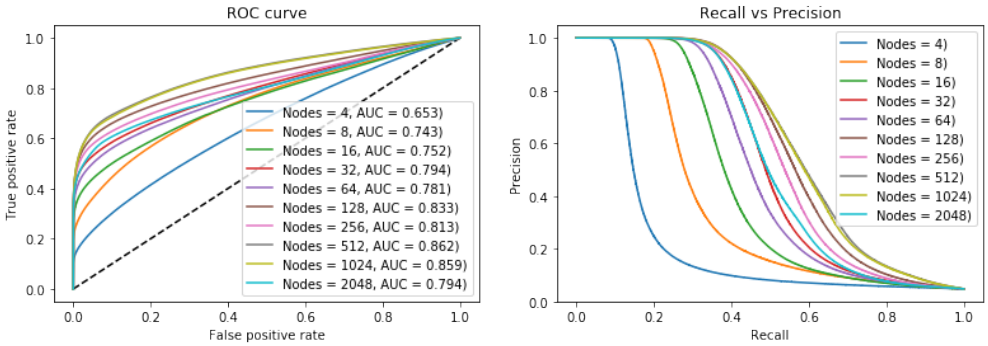
(a) Receiver operating characteristics (ROC) for a varying number of layers.

(b) Recall vs precision for a varying number of layers.

Figure 6.2: The evaluation metrics used to distinguish the accuracy between varying the number of layers with the number of nodes fixed at 512. The resulting AUCs are relatively similar, but both plots demonstrate that for 512 neurons the single layer architecture demonstrates the most promise.

6.3.2.2 Number of Neurons

In this experiment the number of nodes or neurons is varied across a single hidden layer. The accuracy metrics are demonstrated in Figure 6.3 which show an increase in relative accuracy up until 512 nodes are used at which point the accuracy begins to decline, likely due to overfitting on the training data.



(a) Receiver operating characteristics (ROC) for a varying number of layers.

(b) Recall vs precision for a varying number of layers.

Figure 6.3: The resulting metrics when varying the number of neurons across a single hidden layer. They demonstrate that the accuracy increases up until 512 nodes at which point is decreases, likely due to overfitting.

6.3.2.3 Optimized Network

The goal of these experiments is to attempt to identify an architecture of the MLP which would produce optimum results on the real test set. This was found to be a single hidden layer with 512 nodes. While further optimizations can take place, for example during training by implementing learning rate decay and early stopping. Additional model layers could be added to reduce overfitting, such as dropout and batch normalization. The presence of overfitting appears to be very low as the evaluation of the test set and the validation set achieves very similar results.

The precision, recall and F1-Score were 0.83, 0.69 and 0.75 respectively.

6.3.3 Training on Full Feature Set

Previous results appear to indicate that the additional feature information provides higher performance for the Feature MLP network. Testing this hypothesis is performed in this and the following sections.

This subsection defines tests into utilizing the full feature set for both datasets. These constructed features make use of a two dimensional filter which iteratively calculates 18 different statistical measurements with a 3×3 sample window with a stride of 1.

For the simulated dataset which contains only a single polarization, the filtering method applied to the magnitude and phase equates to 36 constructed features. While for the real dataset with 4 polarizations and 8 Stokes visibilities there are 152 features in total.

The previous investigation into optimization of the model indicated that a single hidden layer was sufficient. This hypothesis is again tested with the real dataset of 152 features as the huge increase in the number of weights to each neuron from the input layer may affect the structure of the network. To investigate this on the real dataset, tests are carried out into the accuracy metrics resulting from 1, 2 and 5 hidden layers with 512 initial neurons halving in each consecutive layer.

Each of these tests again results in a precision and recall within $\approx 1\%$ of each other. Yet the single hidden layer network achieves marginally higher results with 0.91, 0.76 and precision, recall for the real dataset respectively as shown in Table I. The real dataset results far outperform the optimized Feature MLP which was tested

on each polarizations magnitude, phase and Stokes parameters. Indicating the preprocessing of features appears to improve performance, likely due to the injection of spatial information from the filtering method.

Table I: Results for all features on a single hidden layer.

Dataset	AUC	Precision	Recall	F1-Score
Sim	0.92	0.97	0.81	0.88
Real	0.98	0.91	0.76	0.83

During these investigations tests are also performed into increasing the number of neurons in each layer, doubling them as opposed to halving them. This is in order to investigate whether with a higher number of input features performing dimensionality reduction initially may benefit the results. The logic previously as to why this didn't perform more accurately is that reducing the complexity of higher-order learned features may reduce overfitting. Testing this with the 152 feature real dataset proved to in-fact reduce accuracy, far higher than the multiple layer networks with decreasing the number of neurons each layer or when they remain the same.

6.3.4 Training on Top Selected Features

The χ^2 test was used to determine the classes, RFI or non-RFI, which possess a dependent relationship which features. The top 12 features with the highest p-values are deemed important and are extracted as a subset for testing. The goal is that by reducing the number of redundant or irrelevant features the complexity of training is reduced, as well the possibility that overfitting is minimized which would lead to increased accuracy.

The results of these tests are given in Table II. The simulated dataset results in slightly higher accuracy when compared to its full feature set, while the real data shows much lower accuracy. The recall of 0.58 for the real dataset demonstrates a lack of ability to successfully identify RFI. This is in-line with the U-Net test utilizing the same reduced features, where the accuracy also showed a decline. This result is indicative of deep learning treating the reduced features differently than the machine learning classifiers.

Table II: Results for top 12 features on a single hidden layer.

Dataset	AUC	Precision	Recall	F1-Score
Sim	0.92	0.98	0.82	0.89
Real	0.94	0.82	0.58	0.68

6.3.5 Weighted Loss Function

The task of RFI identification presents as an inherently unbalanced dataset. The amount of RFI present in the real dataset accounts for 10% of the data and in the simulated dataset is 13%. Methods exist for combating this through data adjustments, under-sampling the number of non-RFI events or duplicating a number of RFI events to balance the number of classes. This section investigates the application of adding class weights to the Feature MLP which adjust the loss function during training.

The method used for calculating class weights is: $\frac{num\_samples}{(num\_classes * num\_samples\_per\_class)}$

For the entire simulated data the class weights are 0.57 for non-RFI and 3.95 for RFI. For the entire real dataset these weights are 0.55 and 4.72. In practice these are calculated from the training dataset not the entire dataset, but with the amount of data present the values tend to change within 0.01 increments which will have little effect. These weights are applied during training to the best performing datasets so far, for simulated data the top 12 selected features and for the real data the full dataset of 152 features.

During training it the opposite occurs to when no class weights are used. The recall achieves a relatively stable value and the precision gradually increases each epoch. This is expected as the loss function is favouring correct RFI predictions. The results for evaluation on the test set are shown in Table III. For the simulated dataset the precision has decreased, meaning more non-RFI events are being incorrectly labelled as RFI, yet the slight increase in recall means the number of correctly identified RFI events has increased. This is likely a preferred outcome, even if the F1-Score would decrease slightly the major concern in this work is to correctly identifying as much RFI as possible.

For the real dataset the precision has decreased far more drastically than with the simulated dataset. The recall has also increased by an incredibly high margin. Both results demonstrate the trade off that class weights can introduce between precision and

recall, increasing both is unlikely. By increasing the loss function when incorrect RFI predictions are made the training can be optimized to a far higher degree to identify RFI.

Table III: Results for weighted loss function.

Dataset	AUC	Precision	Recall	F1-Score
Sim	0.92	0.91	0.84	0.87
Real	0.98	0.62	0.93	0.74

6.4 Implementation of the Flat MLP

This section outlines a brief investigation into a different concept for the MLP architecture referred to here at the *Flat MLP*. This architecture takes the 100×100 sub-division of the spectrogram and flattens them into a one dimensional tensor for input into an MLP. This has the drawback of not utilizing any features for the dataset as each neuron in the input layer is responsible for a single sample.

While not many tests are carried out into this architecture it is included as the results indicate a much higher proficiency for learning likely due to the spatial relationship which is available to each neuron as they receive all samples within a 100×100 subsection. This is demonstrated in Table IV with the simulated dataset which out performs the iteratively optimized and weighted Feature MLP network. The real dataset shows a distinct lack of ability to learn any real classification.

It is assumed this is due to the simulated dataset having a higher concentration of broadband RFI. By utilizing this spatial relationship in the Flat MLP, even with just the single magnitude this architecture has achieved results in-line with the top U-Net predictions on the same data.

Table IV: Results for Flat MLP trained on XX magnitude.

Dataset	AUC	Precision	Recall	F1-Score
Sim	0.92	0.99	0.84	0.90
Real	0.82	0.77	0.28	0.41

Chapter 7

Results and Discussion

This chapter summarizes some of the best performing models and implementations. It explores the reasoning behind why these implementations performed with a higher accuracy and details some further metrics and insights into their training and evaluation with a brief discussion on the assumptions as to how each architecture is inherently more suited to the specific dataset. It goes on to demonstrate the effect a model can have on the Boolean flag when trained on a real data ground truth with inherent false positives. A separate method is discussed which may be used to investigate the accuracy of RFI classification in the form of imaging.

7.1 Summary of Best Performing Learning Methods

Multiple implementations and configurations were investigated on both datasets. The best performing iterations are given in Table I. Each result is from a specific implementation of the algorithm and a specific feature set for each dataset. These details are given below:

- RFC
  - Simulated: 12 χ^2 top selected features
  - Real: 12 χ^2 top selected features
- U-Net
  - Simulated: 1 XX magnitude feature

  - Real: 8 magnitude and phase features

- Feature MLP
  - Simulated: 12 χ^2 top selected features, single hidden layer
  - Real: 152 features, single hidden layer

The results in Table I summarize the top performing methods from each dataset. While the RFC shows the highest F1-Score overall, this is specific to the simulated dataset and its accuracy on the real dataset is far lower than the others. This likely indicates the RFC is unable to handle the false-positives inherent in the real dataset due to the over-flagging performed by AOFlagger and manual flagging.

The interesting result in the U-Net model was that the highest accuracy achieved on the simulated dataset was for using only a single magnitude vector. Despite the feature construction methods showing promise in the other learning models it is clear the fully convolutional neural network did not draw much benefit from the additional pre-processing. While using the top 12 selected features for the simulated dataset did not detriment the U-Net learning they proved to provide no further accuracy improvements.

Table I: Results of the highest performing implementations.

Algorithm	Dataset	Precision	Recall	F1-Score
RFC	Sim	0.99	0.87	0.92
	Real	0.85	0.65	0.73
U-Net	Sim	0.99	0.85	0.90
	Real	0.87	0.77	0.81
Feature MLP	Sim	0.98	0.82	0.89
	Real	0.91	0.76	0.83

7.2 Results on Opposite Datasets

This section explores the results for each learning method when tested on the opposite dataset they were trained on. The goal of this is in order to assess the ability for each method to inherently learn a generalized form of RFI identification. The real dataset has lower results in each method due to the inherent false positives the AOFlagger

algorithm combined with manual flagging produced. As the goal in flagging a real dataset is to remove the highest amount of RFI possible, at the expense of mislabelling non-RFI events.

It is assumed the methods trained on the real dataset are learning the inherent mechanisms applied to labelling the RFI. While for the real dataset the ground truth is absolute, the exact position of each RFI sample known is correct. If a learning method was to be applied in practice it is required to be as generalized as possible, with the lowest degree of overfitting. In a real application an entire unseen dataset is required to be predicted.

The results into testing the overfitting of a subset of models is given in Table II. It is clear that these models are not in any way generalizing a predictor as the F1-Scores are very low. This could be the result of multiple effects. The most likely effect is the differences in magnitudes across features in the two datasets. While these values are standardized with respect to each feature vector, removing mean and shifting to unit variance, the sheer difference between the two datasets is extremely high. A few selected statistical measures for the top 12 constructed features from each dataset are shown in Table III.

It is immediately clear that the simulated dataset has far higher values. This is due to the nature of the broadband continuous RFI in this dataset. When the filtering method is applied in the real dataset the concentration of RFI is relatively low spatially. While for the simulated dataset there are multiple regions where the 3×3 filter will contain only RFI for all 9 samples.

While normalization or standardization seeks to minimize the impact that high magnitude samples can have on a classifier, the range that this induces between non-RFI and RFI samples is much higher in the simulated dataset than the real one. There are likely many ways to overcome this, such as a logarithmic normalization technique, which has been used in literature.[11]

The trend shown between the real and simulated test sets shown in Table II indicates that training on the real dataset and testing on the simulated dataset achieves far higher accuracy. This seems to confirm the hypothesis that this is due to the difference in magnitudes between RFI across both datasets. When trained on real data the discriminating hyperplane defined by the classifier would be closer to identifying a middle ground, thus when predictions are made on the simulated data the range between the

high magnitude RFI and low magnitude non-RFI samples is not as drastic. Yet when the opposite occurs and the real dataset is predicted, the relatively low magnitude RFI features when compared to the simulated ones are being mistaken for non-RFI and vice versa.

Table II: Results of evaluating a model on the opposite dataset.

Algorithm	Test Dataset	Precision	Recall	F1-Score
RFC	Sim	0.45	0.38	0.41
	Real	0.13	0.75	0.22
U-Net	Sim	0.17	0.88	0.28
	Real	0.98	0.01	0.02
Feature MLP	Sim	0.48	0.54	0.50
	Real	0.20	0.01	0.02

Table III: Highest and lowest values for measurements of features.

Dataset	Range	Mean	Std	Max
Real	Highest	1.39	2.89	260.84
	Lowest	0.30	0.45	3.06
Simulated	Highest	19.9	80.34	1732.34
	Lowest	0.35	0.65	3.11

7.3 Supervised Machine Learning Classifiers

Investigations into the three classifiers; K-Nearest Neighbour, Naive Bayes and Random Forest demonstrated almost unanimously that the RFC was able to best identify RFI with a recall of 0.87 on the simulated data and 0.65 on the reduced feature set of real data. These conventional algorithms particularly struggled with predicting the real data. While all the methods employed showed the simulated dataset was easier for RFI identification, the reason for this is inconclusive. It may be that the statistical nature of the RFI was very different between the datasets but more likely the fact that the flags designating RFI in the simulated dataset were a pure ground truth.

In the real dataset the flags were a combination of the AOFlagger algorithm and manual flagging techniques, a best attempt at producing the most accurate flag map. But false positives in the ground truth inherently existed. This may be the cause for reduced learning capabilities in these algorithms, where methods such as convolutional neural networks allow for some inconsistencies in class and limit overfitting in a manner which allows learning to proceed despite this.

The use of two dimensional rolling filters for feature construction greatly improved the predictions beyond what was capable with the sliding window approach. As all the classifiers treated each time/frequency sample individually, the ability for the two dimensional filtering method to give a sample a semblance of the inherent spatial relationship between samples gave the datasets a higher degree of information from which to learn.

7.4 U-Net

During training the U-Net reached convergence rapidly. It is well known that the fully convolutional networks without any dense layers are able to be accurately trained with smaller datasets. Each data subset of 100×100 created only 2310 samples over the entire dataset. The training metrics for a smaller network are shown in Figure 7.1 which demonstrate how rapidly the validation loss converges. The recall and precision reach their stable points after only one or two epochs, even with learning rate decay their values remain stable. This was likewise seen with a higher initial learning rate, the model simply trains extremely quickly.

The effect of convergence on the precision recall curves was converse to what was seen with other training techniques, the ResNet pre-trained network showed a similarity to the MLP architecture where the precision begins at a high point and begins to decay as the recall increases. This signals the networks trade off in identifying RFI, where more RFI samples are correctly identified at the cost of false negatives in the form of incorrectly identified non-RFI.

Demonstrating the accuracy metrics on a U-Net trained on the real data set is shown in Figure 7.2. The AUC of the receiver operating characteristics shows visually the ability for the model to separate classes with the elbow of the curve almost close to perfect. The recall precision plot shows the effect of varying the decision threshold

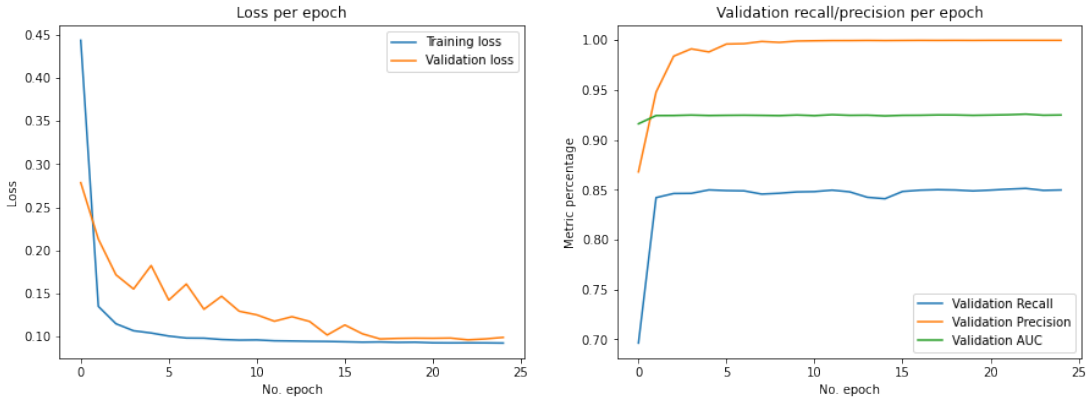
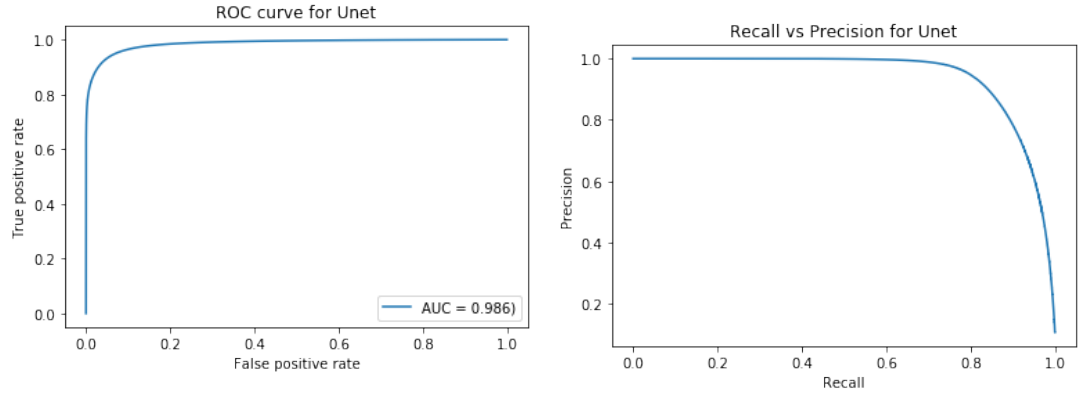


Figure 7.1: The U-Net training metrics. Demonstrating the loss and validation loss per epoch, which quickly reaches convergence. The precision and recall per epoch on the validation data show this convergence with stable values after almost a single epoch.

for a boolean output of the probabilistic prediction.



(a) Receiver operating characteristics (ROC) for U-Net.

(b) Recall vs precision curve for U-Net.

Figure 7.2: Resulting ROC and precision recall curves for the U-Net on the real data test set. The ROC shows a high degree of separation between classes and the precision recall curve demonstrates this effect.

The results of the U-Net architecture were the highest compared to the machine learning classifiers and the MLP architectures. While the recall for the real dataset was relatively low at 0.67 the precision of 0.93 demonstrates few false negatives. With the ROC curve in Figure 7.2 it can be seen that choosing an optimum threshold can greatly increase the amount of RFI identified, at the cost of increasing the number of incorrectly identified non-RFI samples.

7.5 Multilayer Perceptron

The main MLP investigated is referred to as the Feature MLP as it treated each time/frequency sample individually, but applied the feature vector as a tensor to the network. This approach for the data shape is identical to that for the machine learning classifiers but differed in terms of how batches were dealt with. The data generator developed for serving HDF5 files in the U-Net architecture made use of 100×100 subsets of the spectrogram. The same datasets were used in this instance, simply flattening each sample into individual rows with features as the arrays columns. The limitation of this restricted the minimum batchsize to 10000 samples. During training this proved to have little effect. As was demonstrated in the implementation of various network configurations in terms of number of layers and neurons the limiting factor was simply the number of features, batchsize had no observable effect on training and evaluation.

The training validation metrics during training of an implementation of the MLP are shown in Figure 7.3. These plots were generated after training the full 152 real data features with class weights to favour RFI prediction. The increase in validation precision at the expense of reducing recall is the opposite effect of training without class weights. Recall the metric of interest as it demonstrates the classifiers ability to correctly predict RFI events. While class weights increased the recall metric with a reduction to precision it is visible in the plot that the recall gradually reduces further than that of precision.

Early stopping prevents this effect from occurring further. There would be a point where the precision and recall begin to converge, but with a decreasing recall the cost function begins to increase owing to the class weighting. The large spike in the validation loss around epochs 24, 70, 80 and 160 are the repercussions of learning rate decay. As the learning rate is reduced by an order of magnitude the validation loss initially increases in response, but this allows the subsequent epochs to traverse the loss function more precisely, which in turn allows the validation loss to decay further.

7.6 Spectrogram Results

A brief investigation is performed into how the neural network MLP compares to the U-Net architecture on a single time/frequency spectrogram from the real dataset. This

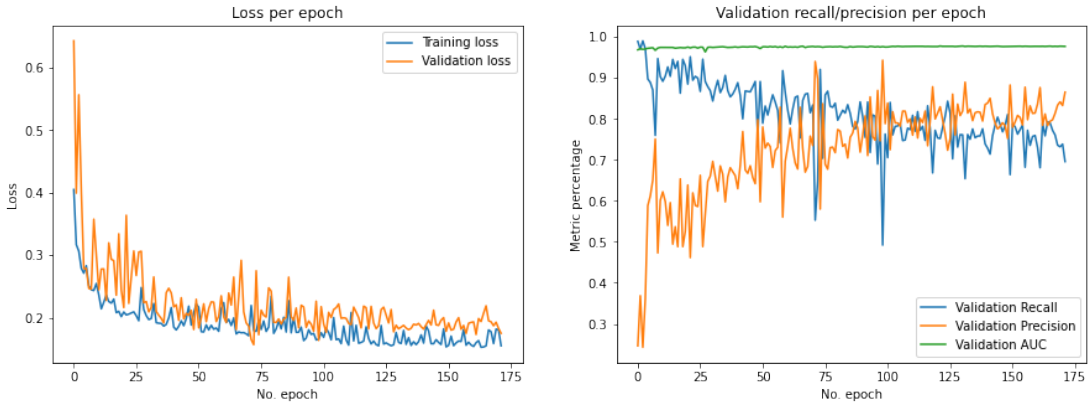


Figure 7.3: Validation metrics during training for the Feature MLP network on the full 152 features of the real dataset with class weights applied. The changes are relatively high each epoch, especially compared to the U-Net during training. This is caused by a high number of weight updates each epoch. The recall precision curve shows the trade-off between and increasing precision at the expense of recall. This is due to the class weighting applied during training which influences the cost function to favour predicting RFI, without class weights the opposite is seen.

configuration of the real dataset was composed of each magnitude and phase of each polarization as well the Stokes parameters. This investigation was carried out by the author in a related work.[42]

This is demonstrated visually in Figure 7.4 which shows the neural network MLP is able to identify a far higher number of RFI blips. These single sample RFI samples have been demonstrated in literature as one of the harder categories to identify.

This effect is likely due to the data shape, where the MLP treats each time/frequency sample individually the U-Net is far more reliant on the spacial relationship between nearby samples. This also lends some insight into results from the investigation into the Flat MLP architecture which operated on only a single magnitude but had each 100×100 subset and showed the same propensity for learning as the U-Net with specifically the simulated data.

7.7 Imaging and Blob Detection

While not investigated thoroughly in this work or literature related to ML on RFI identification, the goal of RFI flagging is to perform imaging on the baseline's visibilities. Each visibility and flag are combined in a measurement set file and CASA's CLEAN

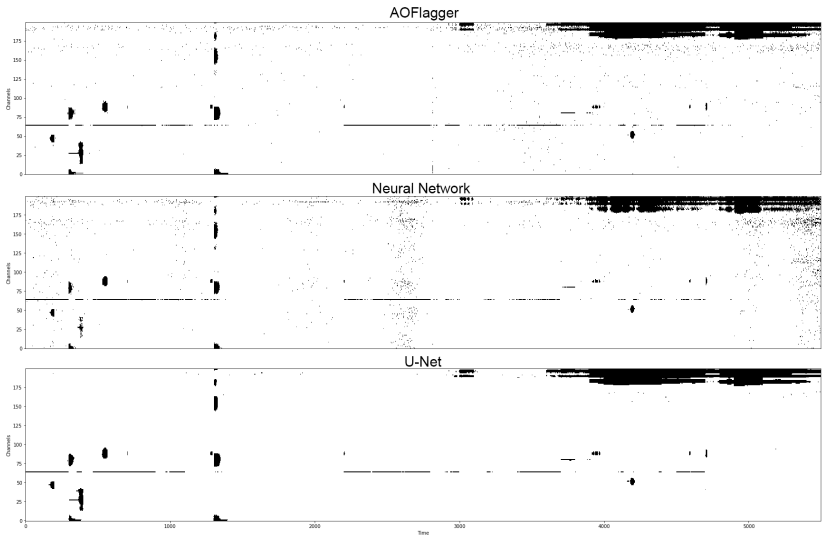


Figure 7.4: A single visibility left out of training and evaluated for the neural network MLP and U-Net architectures. While the U-Net shows overall an improved accuracy these spectrograms show a difference in the type of RFI identified. While the interference in the top right is more cleanly represented in the U-Net the neural network shows a higher proficiency for identifying the single sample RFI blips, which are known to be harder to predict.

algorithm is used in image deconvolution. This iteratively works on the highest values of an identified point source, applying a small gain convolved with the telescope point spread function. The results for imaging, blob detection and source finding were performed by the author in a related work.[42]

The limitation of this method is that each flag is required to be produced. For a learning method a subset of the entire dataset is required for training and validation, the method used is inherently flawed by applying the model to predict already seen training samples.

Once imaged, blob detection is performed in order to generate Gaussian and island models with which the number of identified sources may be identified. The results of the CLEAN algorithm and blob detection are seen in Figure 7.5. Both CLEAN and source finding have many parameters to be optimized, and this is in no way a representation of the perfect imaging procedure. Minor changes between the Gaussian and island model can be seen between the two learning methods.

The metrics associated with blob detection and source finding and shown in Table IV. The learning methods are shown to produce a higher flux density owing to the number of sources identified. Where the AOFlagger algorithm from which the ground

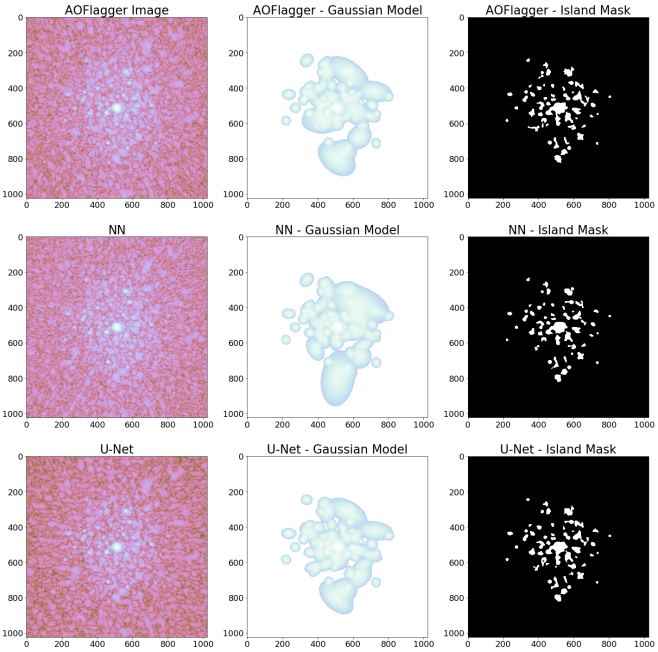


Figure 7.5: Results from the CLEAN algorithm for imaging the dataset. Little changes can be seen in the images themselves but the Gaussian models display minor differences between the identified point sources.

truth was generated is able to identify 59 point sources this increases in each of the learning methods.

This is not a conclusive result that the methods have distinctly identified as higher number point sources but indicate that while the ground truth from AOFlagger inherently contains false positives the algorithms may have learned the mechanism to identify RFI while reducing the number of false positives.

Table IV: Comparisons of PyBDSF Metrics.

	AOFlagger	**NN**	**U-Net**
Bg. mean (Jy/beam)	-6e-05	-1.3e-05	1.1e-04
Bg. rms (Jy/beam)	1.54e-3	1.65e-3	1.54e-3
Flux Density (Jy)	5.142	5.345	5.301
Source Count	59	65	64

Chapter 8

Conclusions and Future Recommendations

This chapter discusses the final conclusions into the application of the specific machine learning algorithms employed in this work for the detection of radio frequency interference. It then goes on to record possible expansions into this topic. Specific recommendations are made as to improvements of the methods explored in this work.

8.1 Conclusions

This section draws some final conclusions into the various experiments conducted in this work. It details some interesting findings for each category. A challenge in implementing learning algorithms on the real dataset was the inherent false positives, generated through the manual flagging and AOFlagger procedures. While this presented with lower accuracy compared to the simulated dataset, for which the ground truth was absolute, the metrics for each algorithm and both datasets achieved compatible results to current literature in machine learning for RFI flagging.

It was hypothesised early on that utilizing multiple polarizations for the real dataset would result in higher accuracy scores. This effect is seen when learning methods treated each sample individually, but this may be due to these methods functioning optimally with an increasing number of features. In the convolutional network the inclusion of phase and/or fully polarized data had little effect.

8.1.1 Feature Construction and Selection

A major goals of this work was the investigation of statistical feature construction methods on spectrogram data. Literature has demonstrated the use of 8 sample rolling windows for feature extraction along the time domain, which is a logical choice for data similar to that within this work. This is owing to the apparent independence of time series data across each frequency bin.[33]

The nature of RFI in the time/frequency data, particularly in the simulated HERA visibilities, was prone to a higher degree of narrow band persistent RFI. The application of a rolling window filter in this direction proves of little value, as this category of RFI is appears relatively straight forward for algorithms to identify as it generally maintains very high magnitudes.

The identification of broadband impulses proved less successful in identification, but still more so than the single sample RFI blips. It was this property which the two dimensional filtering techniques sought to accentuate. Thus the implementation of two dimensional filter methods was investigated in an attempt to capture a higher degree of the information present in time/frequency data for the machine learning algorithms. While the type and number of statistical features chosen may be improved it can be concluded that their use in the machine learning methods which treated each sample individually showed significant improvements over using only the magnitude vectors.

The reasons for this are not certain. In the case of the algorithms in which each samples was treated individually the filters injected a small semblance of spacial information into each extracted feature. As the two dimensional filter is applied to a sample it takes nearby samples into consideration, for these datasets there exists a relationship between the location of RFI events and their magnitudes. Broadband and narrowband RFI will differ in their magnitudes and distributions relevant to each other, but will maintain similar compositions for the events themselves across frequency and time respectively.

With the high number of constructed features generated, particularly in the real dataset, feature selection was performed with the χ^2 test. Many feature selection techniques are applicable and have their own advantages and disadvantages. With the simulated dataset the selected features had little influence compared to the results from utilizing all the constructed features, yet the real data often showed lower perfor-

mance. With the RFC feature importance metrics it is clear the evaluation of feature importance can differ highly between metric criteria.

It is hard to conclude broadly if the inclusion of phase information as a feature impacted the accuracy of classifiers. The χ^2 test identified the standard deviation of phase as a feature of high importance but no other generated phase features were present in either datasets top 12. The RFC feature importance measures also lacked any definitive proof of phase information acting as a valuable inclusion.

Literature has shown the inclusion of phase information to be of value in a convolutional U-Net which employed a unique architecture, treating the time and frequency domains with separate convolutional layers, normalizing and combining these convolutional features towards the end of the network.[11] Utilising phase as a feature vector in combination with magnitude features was obviously not as effective as a targeted, specific architecture for this goal and is the reason no demonstrable improvements were shown.

8.1.2 Machine Learning Classifiers

The addition of feature construction improved the machine learning classifiers accuracy far more than the other algorithms. Of the three classifiers selected; K-Nearest Neighbors, Naive Bayes and Random Forest, the RFC appeared to show the highest accuracy overall.

Particularly for the simulated dataset the RFC presented as the most accurate classifier out of all the techniques investigated. This was demonstrated again when testing the separate dataset that the model was trained on, which the RFC proved to achieve close to the highest accuracy for as well.

The RFC has been known to function well in a wide variety of applications. The use of feature construction showed clear improvements in the accuracy of the classifier.

8.1.3 U-Net

The U-Net convolutional neural network showed interesting results on the constructed feature sets. The filtering method for constructing features performs a similar function to a convolutional filter, but where the constructed features are a predefined statistical measure the convolutional filters are learned in a manner which will best increase the

accuracy.

The accuracy of the U-Net on the constructed features showed little to no improvement for the simulated dataset, which was able to achieve remarkably scores on only a single magnitude feature. Likewise the architecture was able to construct a better prediction using all real data including the polarization's magnitudes. This is indicative of the learned features being able to construct a parameter space more concisely from the raw datasets than those of the pre-processed ones.

Whether this conclusion can be drawn from the fact that the constructed features made use of nearby samples, which created an inherent loss of information for the convolutional layers is unknown.

8.1.4 Multilayer Perceptron

The Feature MLP network was implemented utilizing each sample independently in the same manner as the machine learning classifiers. As a learning algorithm the conventional neural network performed on par with the convolutional U-Net, which is a surprising result given the prevalence of deep convolutional networks in current literature. The additional constructed features showed great improvement compared to restricting the dataset to its raw magnitude, phase and Stokes vectors.

It was surprising the level of accuracy achieved using only a single hidden layer, it was anticipated that a deeper network would be able to learn higher dimensional features and construct a more accuracy classifier. Yet with a single hidden layer and limited neurons the network is conclusively an accuracy method for predicting RFI flags.

Whether these results suggest that a single hidden layer MLP could converge on any RFI flagging dataset is inconclusive. This is due to statistical differences between point sources and RFI sources between measurements. The prevalence of the difference types of RFI between measurements does seem to suggest, that the MLP lacks the ability to generalize a solution across different types of datasets. The overfitting which occurred when training and testing on separate datasets indicated more towards specific boundary conditions per dataset.

8.2 Future Improvements

This section outlines a subset of core concepts which this work was unable to cover. These implementations, improvements and ideas act as an explanation for possible extensions of the work.

8.2.1 K-Fold Cross Validation

A major limitation of this work is the lack of extensive K-Fold cross validation during evaluation and training. The sheer number of investigations performed and their high complexity leading to huge training times limited the number of tests which included cross validation. For the most part many of the algorithms converged on a solution which resulted in the same metrics across implementations of a test. Where inconsistencies occured the test was repeated which usually resulted in the expected measure.

Yet for a truly conclusive result each experiment should be conducted with K-Fold cross validation, utilizing differing subsection of the dataset to ensure a true test set is under investigation.

8.2.2 Augmentation

The convolutional U-Net algorithm utilized square subsets of the spectrogram, treating time/frequency data as an image segmentation problem. A common method for improving accuracy in this domain is the process of data augmentation. Whereby the subset undergoes a number of transformations during training such as scaling, rotating and adding Gaussian noise. This may result in a more robust learning method which is less prone to overfitting.

8.2.3 Feature Construction for Benchmark Datasets

The feature construction on time series data showed beneficial results with the methods which treated each sample individually. To investigate this pre-processing through feature construction using two dimensional filtering, a standard benchmark dataset should be processed with a similar method. Investigating the results should prove further insights into whether this means of feature construction provides additional

information to certain learning methods.

8.2.4 Dataset Balancing

As the amount of RFI in both the real and simulated datasets account for only 10% of the data the accuracy of a classifier may be improved if data balancing is performed. This can be achieved through undersampling the number of non-RFI events or oversampling the RFI events, effectively duplicating them such that the ratio is closer to 1:1.

8.2.5 Pre-processing Subdivision Overlap

During data pre-processing the spectrograms were subdivided into 100×100 sections in order for the two dimensional prediction techniques to be applied. For a convolutional network this requires the addition of a zero padding and crop operation which may have effected the edge cases. Investigations into creating 128×128 regions which overlap between each other, and averaging the result, may lead to improved accuracy.

8.2.6 Imaging on Unseen Data

An investigation was performed into imagining the real dataset with the CLEAN algorithm and performing blob detection and source finding in order to compare source counts between the ground truth flag, the U-Net and the Feature MLP. Owing to the nature of a learning algorithm utilizing a portion of the dataset for training it should be investigated how the learning methods perform when imaging is performed on a predicted dataset where all the samples are unseen.

The use of training data in a test dataset, while necessary for the practical application of deconvolution in this case, represents overfitting a solution. There is a higher chance that the exact flags which were used during training will simply be reproduced which does not account entirely for the learned behaviour.

8.2.7 U-Net Optimization

Attempts were made to investigate the effect of changing certain parameters of the U-Net architecture. These included the number of layers and number of filters per

layer. Further optimizations are possible such as the amount of dropout after each convolutional layer, or increasing/decreasing this level of dropout per layer. Additionally the optimizer during training can have a large impact on the result depending on the structure of the training data.

8.2.8 Complexity Investigations

The algorithms investigated were extremely time consuming, particularly the conventional machine learning classifiers which do not make use of a GPU. Training using the reduced selected feature set has a large impact on the amount of time required to train and make predictions, yet exact metrics for this were not established. This would be of significant importance of these methods were to ever be used in an imaging pipeline.

8.2.9 Constructed Feature Optimization

The method for constructing features utilized a two dimensional sliding window, or filter. This has inherent parameters which may be optimized such as:

- Number of samples - footprint size
- Type and number of statistical features
- How edge cases are handled ie: mirroring or zero-padding
- Type of window used ie: uniform, Gaussian, triangle

By varying combinations of these parameters a higher degree of information may be captured and higher accuracy achieved, but it is possible this is entirely dataset related.

8.2.10 Feature Selection and Dimensionality Reduction

The feature construction methods applied to the real dataset resulted in 152 individual features. Training and evaluating on this mean features can have a far higher time complexity than a reduced feature set. This work investigated an application of the χ^2 test as a feature selection criteria. There are many other feature selection methods which may prove to be more valuable in improving accuracy.

The investigation of dimensionality reduction methods on the real dataset could function as an additional pre-processing step. Applying an algorithm such as principal component analysis (PCA) as a transformation of the feature vectors may provide additional insights into the internal structure of the RFI within the time/frequency data.

Bibliography

[1] F. Ghigo. Pre-history of radio astronomy.

[2] K. G. Jansky, "Radio Waves from Outside the Solar System," *Nature*, vol. 132, no. 3323, p. 66, 1933. [Online]. Available: https://doi.org/10.1038/132066a0

[3] B. F. Burke and F. Graham-Smith, *An Introduction to Radio Astronomy*, 3rd ed. Cambridge University Press, 2009.

[4] T. Robishaw and C. Heiles, "The Measurement of Polarization in Radio Astronomy," 2018. [Online]. Available: http://arxiv.org/abs/1806.07391

[5] M. Mesarcik, "Discrete wavelet methods for interference mitigation: An application to radio astronomy," Master's thesis, University of Cape Town, 6 2019.

[6] T. L. Wilson, K. Rohlfs, and S. Hüttemeister, *Tools of Radio Astronomy*. Springer-Verlag, 2009.

[7] NRAO. The giant meterwave radio telescope. [Online]. Available: http://www.aoc.nrao.edu/~sbhatnag/Thesis/HTLATEX/SanB/thesisch5.html

[8] C. J. Wolfaardt, "Machine learning approach to radio frequency interference(rfi) classification in radio astronomy," Master's thesis, Stellenbosch University, 3 2016.

[9] D. WIlner. [Online]. Available: https://slideplayer.com/slide/6242288/

[10] R. Lord, "Rfi and radio astronomy why the fuss?." *Square Kilometer Array*, vol. K0000-2001V1-009 TR, 06 2019.

[11] J. e. a. Kerrigan, "Optimizing sparse rfi prediction using deep learning," *Monthly Notices of the Royal Astronomical Society*, Jul 2019. [Online]. Available: http://dx.doi.org/10.1093/mnras/stz1865

[12] S. Misra, "Development of radio frequency interference detection algorithms for passive microwave remote sensing," Ph.D. dissertation, The University of Michigan, 2011.

[13] A. Offringa, F. Mertens, and L. Koopmans, "The impact of interference excision on 21-cm epoch of reionization power spectrum analyses," *Royal Astronomical Society*, 01 2019.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[15] A. Bhande. What is underfitting and overfitting in machine learning and how to deal with it. [Online]. Available: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

[16] S. Raschka. [Online]. Available: http://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/

[17] S. Narkhede. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5/

[18] "Decision forest," *Inf. Fusion*, vol. 27, no. C, pp. 111–125, Jan. 2016.

[19] A. Liaw and M. Wiener, "Classification and regression by randomforest," *Forest*, vol. 23, 11 2001.

[20] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, Aug 1995, pp. 278–282 vol.1.

[21] B. Mwandau. [Online]. Available: https://www.researchgate.net/figure/Biological-Neuron-versus-Artificial-Neural-Network_fig4_325870973

[22] K. Staats, "Genetic programming applied to rfi mitigation in radio astronomy," Master's thesis, University of Cape Town, 12 2016.

[23] NRAO. Kat-7 telescope fitted with 'cold' radio receivers. [Online]. Available: https://www.skatelescope.org/news/kat-7-telescope-fitted-cold-radio-receivers/

[24] N. Niamsuwan, J. T. Johnson, and S. W. Ellingson, "Examination of a simple pulse-blanking technique for radio frequency interference mitigation," *Radio Science*, vol. 40, no. 5, 2005.

[25] L. Schoemaker, "Removing radio frequency interference in the lofar using gpus," Master's thesis, Vrije Universiteit Amsterdam, 5 2015.

[26] A. R. Offringa, A. G. de Bruyn, M. Biehl, S. Zaroubi, G. Bernardi, and V. N. Pandey, "Post-correlation radio frequency interference classification methods," *Monthly Notices of the Royal Astronomical Society*, Mar 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1365-2966.2010.16471.x

[27] A. Offringa, "Algorithms for radio interference detection and removal," Ph.D. dissertation, University of Groningen, 06 2012.

[28] D. Czech, A. Mishra, and M. Inggs, "A cnn and lstm-based approach to classifying transient radio frequency interference," *Astronomy and Computing*, vol. 25, 03 2018.

[29] J. Akeret, S. Seehars, C. Chang, C. Monstein, A. Amara, and A. Refregier, "Hide and seek: End-to-end packages to simulate and process radio survey data," *Astronomy and Computing*, vol. 18, pp. 8 – 17, 2017.

[30] J. Akeret, C. Chang, A. Lucchi, and A. Refregier, "Radio frequency interference mitigation using deep convolutional neural networks," *Astronomy and Computing*, vol. 18, pp. 35–39, 2017.

[31] cosmo ethz. [Online]. Available: https://github.com/cosmo-ethz/hide

[32] HERA-Team. [Online]. Available: https://github.com/HERA-Team/hera_sim

[33] O. Mosiane, N. Oozeer, A. Aniyan, and B. Bassett, "Radio frequency interference detection using machine learning." *IOP Conference Series: Materials Science and Engineering*, vol. 198, p. 012012, 05 2017.

[34] G. B. Doran, "Characterizing interference in radio astronomy observations through active and unsupervised learning," 2013.

[35] O. Mosiane, N. Oozeer, and B. A. Bassett, "Machine learning for radio frequency interference mitigation using polarization," in *2017 IEEE Radio and Antenna Days of the Indian Ocean (RADIO)*, Sep. 2017, pp. 1–2.

[36] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[37] D. WIlner. [Online]. Available: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/u-net-architecture.png

[38] M. Andrecut, S. Guram, S. George, and A. Taylor, "Galfacts rfi excision methods," 07 2011.

[39] G. Nita and D. Gary, "Statistics of the spectral kurtosis estimator," *Publications of The Astronomical Society of The Pacific - PUBL ASTRON SOC PAC*, vol. 122, pp. 595–607, 05 2010.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[41] ml4a.github.io. [Online]. Available: https://ml4a.github.io/ml4a/looking_inside_neural_nets/

[42] K. Harrison and A. K. Mishra, "Supervised neural networks for rfi flagging," in *2019 RFI Workshop - Coexisting with Radio Frequency Interference (RFI)*, 2019, pp. 1–8.