DOCTORAL THESIS

Feature Detection and Classification in streaming and non-streaming astronomical datasets

Author:
Zafiirah B. Hosenie

Supervisors:
Prof. Benjamin Stappers
Dr. Robert Lyon

A thesis submitted for the degree of
Doctor of Philosophy

in the

Department of Physics and Astronomy in the School of Natural Sciences
Faculty of Science and Engineering
The University of Manchester

2021

Contents

Word count: 77 176

List of Figures

List of Tables

Abbreviations

A.I	Artificial Intelligence
ANN	Artificial Neural Network
AUROC	Area Under the Receiver Operating Characteristic Curve
AGN	Active Galactic Nuclei
CNN	Convolutional Neural Network
CV	Cataclysmic Variable stars
DL	Deep Learning
DM	Dispersion Measure
F-T	Frequency and Time
FABLE	Fast rAdio Burst Localization and dEtection using Mask R-CNN
FRB	Fast Radio Burst
FRBID	Fast Radio Burst Intelligent Distinguisher
FoV	Field of View
FPR	False Positive Rate
GRB	Gamma Ray Bursts
GP	Gaussian Process
GPU	Graphics Processing Unit
ICVaS	Imbalance Learning for Variable Star Classification using Machine Learning
i.i.d	independent and identically distributed
ISM	Interstellar medium
JMI	Joint Mutual Information
*k*NN	*k* Nearest Neighbours
LC	Light Curves
LCM	Latent Class Model

MeerCRAB	MeerLICHT Classification of Real and Bogus using deep learning
MI	Mutual Information
ML	Machine Learning
MLP	Multi-layer perceptron
RF	Random Forest
RFI	Radio Frequency Interference
SP	Single Pulse
S/N	Signal-to-noise ratio
SKA	Square Kilometre Array
SMOTE	Synthetic Minority Over-sampling Technique
XGBoost	eXtreme Gradient Boosting

THE UNIVERSITY OF MANCHESTER

Abstract

Faculty of Science and Engineering

Department of Physics and Astronomy in the School of Natural Sciences

Doctor of Philosophy

FEATURE DETECTION & CLASSIFICATION

IN STREAMING AND NON-STREAMING ASTRONOMICAL DATASETS

by ***Zafiirah B. Hosenie***

Astronomy has always been at the cutting edge of data volumes but the growth of interest in short timescale variability is taking that to an entirely new level. The new generation of multi-epoch, multi-band and wide field-of-view astronomical surveys will generate a deluge of astronomical sources, including variables and transients. The sheer volume of this data suggests that transient detection and localization are transitioning from an offline process, to an online and real-time, automated decision-making procedure. In response, this thesis presents an interdisciplinary study of the transient and variable classification problem, with the aim of developing several machine learning (ML) based methods for both optical and radio astronomy.

This thesis starts with the classification of 11 types of periodic variable stars acquired with the Catalina Real-Time Transient Survey (CRTS). Based on our analyses, we find that accurate variable star classification is possible with just seven features - much fewer than in other works. In addition, we show that this classification problem cannot be solved with a 'flat' multiclass classification approach, as the data are inherently imbalanced. To partially alleviate the 'imbalanced learning problem', we convert a standard multiclass problem into a hierarchical classification problem, by aggregating subclasses into superclasses. This results in improved performance on rare class examples typically misclassified by multiclass methods. To further improve the hierarchical classification performance, we apply 'data-level' approaches to directly augment the training data so that they better describe under-represented classes. When combining the 'algorithm-level' together with the 'data-level' approach, we further improve variable star classification accuracy by 1-4%.

In addition, in order to have an early and rapid characterisation of interesting candidates in optical and radio surveys, it is fundamentally important to automate several steps within a transient detection pipeline, including the separation of transients/astrophysical events from 'bogus' or Radio Frequency Interference (RFI) detections, which has become a bottle-neck in

fast detection pipelines. To address this challenging task, we built `MeerCRAB` and `FRBID` - ML software based on a Convolutional Neural Network. `MeerCRAB` has been deployed in the MeerLICHT transient vetting pipeline and is designed to filter out the so called 'bogus' detections from true astrophysical sources. Optical candidates from the MeerLICHT telescope are described using a variety of 2D images. However, the relationship between the input images and the target classes is unclear, since the ground truth is poorly defined and often the subject of debate. This makes it a challenging task to find which numerical features or source of information should be utilised to build a classifier. We therefore used two methods for labelling our data (i) thresholding and (ii) latent class model approaches. Afterwards, variants of `MeerCRAB` models are deployed using different combinations of input images and training set choices, based on classification labels provided by volunteers. We found that the deepest network worked best with an accuracy of 99.5%.

`FRBID` is being used in MeerTRAP - a backend for the MeerKAT radio telescope that continuously searches for radio transients and pulsars. `FRBID` aims to remove any remaining RFI and classify new astrophysical candidates automatically in real-time. The performance of `FRBID` shows less than 1% false positive rate. Up till date, `FRBID` has detected more than half a dozen new FRB candidates, even in the presence of RFI. This discovery shows how accurate and efficient an ML algorithm can be, in real-time processes.

Lastly, this thesis provides a first proof of concept of a model, *F*ast R*a*dio *B*urst *L*ocalization & d*E*tection, `FABLE`, based on Mask R-CNN. `FABLE` is used for automatic detection, segmentation, and classification of FRBs with Dispersion Measure-Time (DM-T) images. We focus on improving the detection rate of FRBs, especially for low signal to noise ratio (SNR) in radio images and to identify 'single pulse' from background noise. The `FABLE` algorithm provides us with the probability that the detected source, is an FRB/Single Pulse and also the position of the FRB. In addition, `FABLE` outputs the DM and time parameters of the detected candidate, which allow us to characterise the FRB candidate detected. The time parameter is used to extract a piece of the original filterbank data and to de-disperse it to generate two further plots which astronomers will want to look at: a de-dispersed pulse, and the frequency - time plot. Moreover, `FABLE`'s performance is evaluated on the Average-Precision (AP) score metric. We measure a precision of 99.9% at 90% recall for FRB/SP with a minimum detection confidence threshold of 0.5. The `FABLE` model can serve as a new tool for real-time pipeline that can automatically detect FRB/SP and can be adopted in other domains in astronomy, for e.g. in source extraction software for detecting and localizing sources in astronomical images.

Declaration

I, ***Zafiirah. B. HOSENIE***, hereby declare that the contents of this thesis titled, " Feature Detection and Classification in streaming and non-streaming astronomical datasets", is my own work, except where specific reference is made to the work of others. The list of publications is detailed in §1.7 and the specific contributions in each chapter are listed below.

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Zafiirah. B. Hosenie

July 2021

Copyright Statement

[*]documents.manchester.ac.uk
[†]www.library.manchester.ac.uk/about/regulations/

Acknowledgment

First and foremost, I would like to express my sincere gratitude to my supervisors, **Prof. Ben Stappers** and **Dr. Rob Lyon**, for their unwavering support and belief in me. Their trust has given me confidence in myself and encouraged me in all the time of my academic research. They never failed to motivate me and support me in every decision I took during those 3 years. In addition, they have allowed me a lot of freedom in pursuing whatever ideas I wanted, something I greatly value.

This achievement would not have been possible without the support of the DARA BIG DATA project. Thanks to **Prof. Anna Scaife** for organisation such an incredible project, that supports us, the African's community to be owners and shapers of the advancement in technology and artificial intelligence. I would also like extend my sincere thanks to **Linzi Stirup** for being such an amazing DARA BIG DATA project manager and for her assistance with every support I needed.

I would like to offer my special thanks to the MeerLICHT team for such a great collaboration, especially **Prof. Paul Groot**, **Dr. Steven Bloemen**, **Dr. Paul Vreeswijk**, and **Dr Bart Scheers**. I would also like to thank **Dr. Cees Bassa** for great discussion on computing the bow-tie shape of a single pulse. Moreover, I thank the MeerTRAP team, mainly **Dr. Mateuz Malenta** and **Dr. Mayuresh Surnis** for the time spent on Zoom, critiquing the results obtained from our AI system. In addition, I would like to thank the Jodrell Bank team **Xiaoxi**, **Lin**, **Manisha**, **Kaustubh**, **Laura**, and **Laila** for our chit chat.

I would like to end with a special mention to my family. My **mom**, who is my constant source of inspiration, to always support my choices and believing in me. Thanks to my **brothers**, **sister** and **nephew** that keep me motivated in hard times. Last but not least, I express my profound gratitude to **Harry** for providing me with unfailing support, continuous encouragement, to be always by my side and for providing me unending inspiration.

Thanks for all your encouragement!

"Your life is like a balloon...if you never let yourself go, you will never know how far you can rise."

– Linda Poindexter

1

Machine Learning in Astronomy

"We can build a much brighter future where humans are relieved of menial work using Artificial Intelligence capabilities."

– Andrew Ng

1.1 Introduction

In recent years, there has been rapid progress in the development of astronomical instrumentation. An increasing number of instruments are dedicated to undertaking repeated observations of large swaths of the sky every few nights. This is not just restricted to optical astronomy but extends across the electromagnetic spectrum from radio to gamma-ray wavelengths. With an ever-increasing number of astrophysical transient surveys (surveys that observe short time scale sources in the sky), this has led to an avalanche of data, thus resulting in large databases. In these huge databases, a large of amount of information is hidden that cannot be obtained by manually analysing them. In astronomy, it is of crucial importance to develop efficient automated data analysis approaches to detect these interesting objects, otherwise much of the important information and potential science in the data, will be left unexplored.

To appreciate the extent of this problem, with the upcoming photometric surveys such as those conducted at the Vera C. Rubin observatory (LSST; LSST Science Collaboration et al. 2009), it is estimated that it will generate around 30 TB of raw data. These data will require processing and reduction, thus final archival data will be around 60 PB and the final catalogue will be approximately 10-20 PB in size (Borne, 2008). This challenging task calls for a new approach in astronomy, one that is able to deal with *Big Data*. By *Big Data*, it refers to data-sets that are so large that traditional data storage and processing methods are inadequate and no longer efficient. Many fields and industries are already in the era of *Big Data* and are currently facing this challenging problem.

As the challenges regarding *Big Data* continue to grow, many fields of study have shown a great interest in Machine Learning. *Machine Learning* (ML) is the application of statistics and mathematics to enable algorithms to be constructed that are capable of '*learning directly from data*'. For example, weather forecasts can be made by an algorithm that learnt from previous weather data or an algorithm can identify a cat in an image within milliseconds without the intervention of human. The surge in popularity of ML can largely be attributed to an exponential growth in computational power. The easy access of graphics processing units (GPUs) and tensor processing units (TPUs) via Google Collaboratory[*] (also known as Colab, a cloud service based on Jupyter Notebooks), facilitates the application of ML in various fields and industries. Recently, ML has achieved great success in Artificial Intelligence (AI) and Big Data applications, from the popular chatbots to self-driving cars (Hancock et al., 2019).

Various algorithms starting from traditional ML algorithms (e.g. Random Forest, (RF) Breiman (2001)) to *Deep Learning* (DL) algorithms, for example Convolutional Neural Network (CNNs, Lecun et al. 1999) have started to reach levels of human performance in image recognition tasks. DL has had incredible success in other fields too, from voice recognition to mixed-reality for medical diagnoses, thus showing its promise for many other domains.

1.2 Real-time Candidate Selection Challenge

Various surveys, for instance, the Catalina Real-time Transient Survey (CRTS [†], Drake et al. (2009)), the MeerLICHT optical wide-field telescope (Bloemen et al., 2016; Paterson, 2019), and the MeerKAT radio telescope (Jonas & MeerKAT Team, 2016), to name a few, are already generating vast amount of data, from petabytes to exabytes of data. Realistically, this data cannot be stockpiled indefinitely. Therefore, it is necessary to process the data as it arrives from these telescopes in real-time. Such a real-time system would allow the identification of candidates (interesting patterns detected in the data collected) and these candidates would be prioritized for storage and/or immediate follow-up observations. All other candidates, for instance noisy or corrupted data, must be discarded. The development of these advanced technological instruments have inadvertently introduced a real-time process to the pre-existing candidate selection challenge. Although it is feasible to apply existing sub-optimal selection techniques to

[*]https://colab.research.google.com/notebooks/intro.ipynb
[†]http://crts.caltech.edu/

reduce these problems, it is almost with certainty that this would result in missed discoveries. However, it is inconceivable for these instruments to miss potential interesting candidates in this way, due to inadequate technological software. Thus, it is a necessity to develop automatic candidate selection pipelines that will enable the instruments to deliver their science goals.

1.3 Research Scope

The research presented in this thesis is mostly focused on the development of various machine learning pipelines for optical and radio surveys. More specifically, machine learning frameworks that automate the classification of transients and variable stars with high levels of accuracy for CRTS, the MeerLICHT and the MeerKAT telescopes. This work addresses the challenges in classifying variable stars and transients in optical and radio data. In addition, it is concerned with the technical and practical challenges encountered in time-domain astrophysics, that is rejecting artefacts (which can occur as a result of saturated sources, convolution problems, defects in the detector, atmospheric dispersion, radio frequency interference and cosmic rays passing through the detector, amongst other things) in the imaging process for the MeerLICHT and MeerKAT telescopes. The rejection of artefacts and noise has become the most obstructive bottle-neck between fast transient detection and the ability to feed these interesting targets to follow-up observations for early classification. Central to these challenges is the imbalanced learning problem. Together these challenges make the existing classification methods hard to use in real-time. The classification challenges addressed by this thesis include,

- selecting the characteristics that best describe each class of variable stars and use them to train a ML algorithm,
- a severely class imbalanced distribution, that makes the classification difficult for certain types of variable stars due to their rarity,
- the need to maintain extremely low false positive rates so that time, observational and computational resources and effort are not wasted in analysing spurious detections or discarding interesting candidates,
- the need to maintain low false negative rates to avoid discarding interesting candidates and increase the chance of new discovery,
- developing an adaptive learning technique that can be trained anytime as data distribution may change with time due to noise and interference dominated environments.

1.4 Aims and Objectives

The aim of this study is to devise various machine learning pipelines for variable and transient classification and establish the positive role it plays in the future of astronomy. With new facilities and archives of astronomical databases, time domain astronomy requires these advanced statistical and machine learning techniques to enable scientific discovery. These include characterization, categorization and classification of candidates with high precision and accuracy. For our aims to be tractable, they are divided into key research objectives and challenges:

1. To analyse the effectiveness of features characterizing variable stars.
2. To assess the strengths and suitability of existing machine learning techniques for variable star classification, highlighting any weaknesses to overcome.
3. To develop a new variable star classification pipeline for CRTS data. The aim is to develop an algorithm that overcomes the challenges encountered with existing pipelines, for instance, the imbalanced learning problem.
4. To develop transient detection software for the MeerLICHT telescope for early detection and rapid follow-up, thus allowing the astronomical community to have an accurately labelled data set that will speed up the discovery of new transients.
5. To develop a Fast Radio Burst (FRB) and Radio Frequency Interference (RFI) detection software for the MeerKAT telescope. The software will automate the rejection process of RFI and noise candidates.
6. To evaluate the proposed automated frameworks by analysing the false positive and false negative rates.

1.5 Research Method

A detailed staged research methodology is employed. This research work is somewhat unusual as it bridges the gap between the field of computer science and astronomy.

1.5.1 Research Hypothesis

The goal of this research is to employ machine learning techniques to ease the candidate classification challenge. Success in this regard is evaluated upon the existence of features that well characterize a particular candidate. These features are used in automated machine learning algorithms to successfully *'learn'* boundary separation between candidates of real and bogus, FRBs and RFI, and among various types of variable stars. This leads to a hypothesis that can be formulated as follows. It is hypothesized that real transient candidates have certain features, that demarcate them out of the large number of noisy, spurious bogus detections during a transient search. A similar hypothesis is applied to FRBs & RFI/noise candidates and to various types of variable stars. Each candidate possesses certain salient features that differentiate them from other types of candidates. If such features prevail, it is plausible to learn a separation with high precision and accuracy using machine learning algorithms, for e.g., between real & bogus, FRBs & noise/RFI and various types of variable stars. If such features are easily distinguishable, then transient search and variable stars classification would be an entirely trivial process. However, if such prominent features are not available, it will be hard and perhaps impossible to classify transients and variable stars. This hypothesis shows a valid answer why classification of transients and variable stars are hard. Therefore, it is important to find a suitable set salient features for the hypothesis to hold.

1.5.2 Literature Review

Prior to undertaking each area of research, a survey of background literature has been carried out independently in Chapters 4, 5, 6, 7 and Chapter 8. Each of these chapters begins with a thorough analysis of the variable stars, real-bogus and FRB & noise/RFI classification literature respectively. The aim of the literature reviews in these chapters was firstly, to gain an understanding of the variable star classification problem and the challenges encountered in fast transient detection pipelines. The second stage provides an overview of the computer science domain for classification. The aim is to acquire a broader picture of the state of the art computer science research within astronomy and to better comprehend the transients and variable stars classification problems in a wider context. The third phase of these chapters focuses exclusively on the application of the core computer science techniques for classifying variable star and transient. This brought about an in-depth review in Chapter 3, of various machine learning algorithm used for classification.

1.5.3 Experimental Studies

Experimental studies carried out during this research varied in design and purpose, from developing machine learning pipelines for variable stars & real-bogus transients classification using data from optical telescopes, to analysing classifier performance of FRBs-RFI collected by the MeerKAT radio telescope. For each study, statistical methods will be utilized to analyse and evaluate the results obtained. The tools of information theory will be employed to study the importance of salient features, the latent class model will be deployed to examine labelling provided by volunteers, and standard machine learning metrics will be used to inspect classification performance of the softwares that will be developed in this study.

1.5.4 Development Work

Many of the outcomes from this research have been publicly shared with both the machine learning and astronomy community for wider use. This research encourages open-source software development that helps to advance research and lays a foundation for others to follow in the years to come.

1.6 Contributions

The work presented in this thesis has led to various contributions.

Contribution 1: Development of a hierarchical classification scheme for variable stars.
The study employed a number of machine learning algorithms to classify periodic variable stars using CRTS data. It concluded empirically that a '*flat multi-class*' scheme is an unsuitable approach for classifying CRTS data into different categories of variable stars. A '*flat multi-class*' scheme is inadequate as the CRTS data is subject to the imbalanced learning problem. This work illustrated that such classifier systems show a poor performance upon the imbalanced datasets. This contribution is proposed in Chapter 4, where a '*flat multi-class*' scheme is converted to a Hierarchical scheme as a plausible solution to tackle the problem of imbalanced data.

Contribution 2: Generating fake light curves of variable stars as a solution for imbalanced learning problem.
In this work, an attempt to further improve hierarchical classification performance is described by applying 'data-level' approaches to directly augment the training data so that they better describe under-represented classes. Three data-augmentation methods have been proposed. When combining the hierarchical scheme together with the 'data-level' approach, variable star classification shows an improvement both in terms of accuracy

and precision. The new proposed 'data-level' hierarchical classifier achieves a considerable increase on rare classes compared to existing approaches. This approach has been adopted recently by Sánchez-Sáez et al. (2020) to data streams produced by the Zwicky Transient Factory (ZTF, Bellm et al. 2019).

Contribution 3: a demonstration of sensitivity of machine learning algorithms when using noisy labelling during training and prediction phase.

This study demonstrates the fundamental importance of having a 'pure' data set when training a machine learning algorithm. By 'pure', this implies that the data (for e.g light curves or image data) shows distinguishable characteristics and the label associated with each candidate, is clearly defined. Latent class modelling has been introduced in this work to show the effect of human error while labelling large data sets. Incompleteness or wrong labels in the training set will lead to inaccuracy in the classifier. Chapter 6 demonstrates the performance drop of machine learning algorithm with the introduction of noisy labels.

Contribution 4: Developing an automated machine learning pipeline to filter bogus candidates for the MeerLICHT facility.

A Real-Bogus classifier, `MeerCRAB`, using deep learning has been developed for the MeerLICHT facility. Artefacts and noise rejections has been one of the bottle-neck in fast transient detection and our ability to feed these candidates to follow-up surveys for rapid classification. Artefacts/bogus candidates are picked up as potential real transient candidates by current software and then human expert must reject these artefact detections. Chapter 6 detailed the process of building the data set to train a deep learning pipeline using convolutional neural network. Given the success of `MeerCRAB`, the software has now been integrated in the MeerLICHT transient pipeline to alert the MeerLICHT team of interesting new transients detected.

Contribution 5: a method for classifying between Fast Radio Bursts/Single Pulses and Radio Frequency Interference for the MeerKAT facility.

`MeerCRAB` can be adapted to disentangle interesting objects from a noisy background. Similar models in radio astronomy are implemented that distinguished Single Pulses from Radio Frequency Interference for the MeerKAT telescope (`FRBID`: Fast Radio Burst Intelligent Distinguisher) as detailed in Chapter 7. `MeerCRAB` is a flexible software, thus we were able to easily modify it to integrate different images as its inputs and as result, achieve high levels of performance when using it for radio astronomy images.

Contribution 6: A demonstration of detecting and localizing fast radio burst or single pulses in radio images.
Classifying radio or optical images has made great breakthrough in astronomy that has helped to achieve important discoveries and the methods have been game-changing, thus allowing us to have so many discoveries. However, it is not only important to classify an image in a particular category or label, but it is important to also implement detection, segmentation and classification of candidate files. This process will improve decision making and will allow us to gain significantly more understanding about the characteristics of detected candidates. Candidate detection and segmentation will not only allow us to distinguish different sources in an image by drawing bounding box on a specific source, but also further mark and classify the pixels within the bounding boxes to a specific class/category. Fast rAdio Burst Localization and dEtection, `FABLE`, discussed in Chapter 8, is a deep learning software developed in this study to detect, localize, segment and classify single pulses/FRBs and RFI candidates.

1.7 Publications and Softwares

This research has led to the following publications.

Primary Author Publications

- **Z. Hosenie**., B. W. Stappers., R. J. Lyon., et al. *'`FABLE`: Fast rAdio Burst Localization and dEtection using Mask-RCNN'*. **Submitted in 35th Conference on Neural Information Processing Systems (NeurIPS) at the Machine Learning and Physical Sciences Workshop, 2021**. DOI:NeurIPS2021.
- **Z. Hosenie**., B. W. Stappers., R. J. Lyon., et al. *'`FRBID`: Fast Radio Burst Intelligent Distinguisher at the MeerKAT Telescope'*. **Submitted in 35th Conference on Neural Information Processing Systems (NeurIPS) at the Machine Learning and Physical Sciences Workshop, 2020**. DOI:NeurIPS2021.
- **Z. Hosenie**., S. Bloemen., P. Groot., R. J. Lyon., B. Scheers., B. W. Stappers et al. *'MeerCRAB: MeerLICHT Classification of Real and Bogus using Deep Learning'*. **Experimental Astronomy.** DOI:10.1007/s10686-021-09757-1.

- **Z. Hosenie**., P. Groot., R. J. Lyon., B. W. Stappers et al. *'Classification of Optical Transients at the MeerLICHT Telescope using Deep Learning'*. **Accepted in 34th Conference on Neural Information Processing Systems (NeurIPS) at the Machine Learning and Physical Sciences Workshop, 2020**. DOI:NeurIPS2020-ID25.
- **Z. Hosenie**., R. J. Lyon., B. W. Stappers., A. Mootoovaloo., V. McBride. *''Data Augmentation in a Hierarchical-Based Classification scheme for Variable Stars''*. **Accepted in 34th Conference on Neural Information Processing Systems (NeurIPS) at the Machine Learning and Physical Sciences Workshop, 2020**. DOI:NeurIPS2020-ID26.
- **Z. Hosenie**., R. J. Lyon., B. W. Stappers., A. Mootoovaloo., V. McBride. *'Imbalance Learning for Variable star classification using Machine Learning'*. **MNRAS, 493 (4): 6050-6059, 2020.** DOI:10.1093/mnras/staa642.
- **Z. Hosenie**., R. J. Lyon., B. W. Stappers., A. Mootoovaloo. *'Comparing Multi-class, Binary and Hierarchical Machine Learning Classification schemes for variable stars'*. **MNRAS, 488 (4): 4858-4872, 2019**. DOI:10.1093/mnras/stz1999.

Co-Author Publications

- A. Vafaei Sadr., E. E. Vos., B. A. Bassett., **Z. Hosenie**., N. Oozeer., M. Lochner. *'DeepSource: Point Source Detection using Deep Learning'*. **MNRAS, 484 (2): 2793-2806, 2019**. DOI:10.1093/mnras/stz131.
- Heavens, A., Fantaye, Y., Sellentin, E., Eggers, H., **Z. Hosenie**., Kroon, S., and Mootoovaloo, A. *'No evidence for extensions to the standard cosmological model'*. **PhysRevLett, 119 (10-8), 2017**. DOI:10.1103/PhysRevLett.119.101301.
- Heavens, A., Fantaye, Y., Mootoovaloo, A., Eggers, H., **Z. Hosenie**., Kroon, S., and Sellentin, E. *'Marginal Likelihoods from Monte Carlo Markov Chains'*. **arXiv:1704.03472v1, 2017**. DOI:arXiv:1704.03472.

Software

- **Z. Hosenie**, Fast rAdio Burst Localization and dEtection using Mask-RCNN, `FABLE`, 2021. This arose from the work described in Chapter 8. The code can be accessed on Github: https://github.com/Zafiirah13/FABLE and Zenodo: DOI: 10.5281/zenodo.4599248.
- **Z. Hosenie**, Multi-Input Fast Radio Burst Intelligent Distinguisher, `Multi-Input-FRBID`, 2021. This arose from the work described in Chapter 7. The code can be accessed

on Github: https://github.com/Zafiirah13/multi_input_frbid and Zenodo: DOI: 10.5281/zenodo.4434307.

- **Z. Hosenie**, Fast Radio Burst Intelligent Distinguisher using deep learning, `FRBID`, 2020. This arose from the work described in Chapter 7. The code can be accessed on Github: https://github.com/Zafiirah13/FRBID and Zenodo: DOI: 10.5281/zenodo.4049946.
- **Z. Hosenie**, MeerLICHT Classification of Real and Bogus transients using Deep Learning, `MeerCRAB`, 2020. This arose from the work described in Chapter 6. The code can be accessed on Github: https://github.com/Zafiirah13/meercrab and Zenodo: DOI: 10.5281/zenodo.4049943.
- **Z. Hosenie**, Latent Class Model, `LCM`, 2020. This arose from the work described in Chapter 6. The code can be accessed on Github: https://github.com/Zafiirah13/latent-class-model and Zenodo: DOI: 10.5281/zenodo.4049954.
- **Z. Hosenie**, Imbalance Learning for Variable Star Classification using Machine Learning, `ICVaS`, 2020. This arose from the work described in Chapter 5. The code can be accessed on Github: https://github.com/Zafiirah13/ICVaS and Zenodo: DOI: 10.5281/zenodo.4049970.

1.8 Thesis Structure

The rest of the thesis is structured as follows:

Chapter 1 provides a description of the aims and objectives of this study. It also elaborates its contributions to the astronomy and machine learning community.

Chapter 2 presents an introduction to the transient universe, with some description of variable stars, pulsars and fast radio bursts. In addition, it describes the various surveys utilized in this research.

Chapter 3 considers the classification of variable stars and transients with respect to wider machine learning research. It then provides a detailed introduction of the various machine learning algorithms utilized in Chapters 4, 5, 6, 7 & 8. This chapter helps to bridge the gap between computer science and astronomy.

Chapter 4 addresses the first research objective. It explores possible solutions to variable star classification found in the literature and this led to the development of a hierarchical classification scheme, first introduced in this thesis to tackle the imbalanced problem in the CRTS data.

Chapter 5 reflects on the poor performance on some of the categories of variable stars with small sample sizes highlighted in Chapter 4. This chapter introduces three methods of data-augmentation to overcome their deficiencies by generating fake variable light curves. A thorough analysis of the new approach lays out the solid evidence for its high accuracy and recall capabilities on the small sample categories.

Chapter 6 studies the application of a deep learning algorithm to classify optical transient image data for the MeerLICHT facility. The algorithm developed in Chapter 6 learns directly from the image data, rather than feeding salient features that characterizes a particular class/category of transients.

Chapter 7 shows the flexibility of the software developed in Chapter 6. The algorithm is adapted such that it is capable of taking radio images from the MeerKAT telescope as input and provides classification probabilities of whether it is an FRB or RFI candidate.

Chapter 8 proposes a new algorithm for not only classifying FRB-RFI candidate, but to also detect and localize the sources in the candidate image. It demonstrates the high level of recall and accuracy on real-world data.

Chapter 9 concludes the research and suggests possible avenues for future research.

2

Variable and Transient Astronomical Sources

"I believe there is no deep difference between what can be achieved by a biological brain and what can be achieved by a computer. It, therefore, follows that computers can, in theory, emulate human intelligence - and exceed it"

– Stephen Hawking

This chapter provides the reader with a background introduction to variable and transient universe. Terminology and concepts related to variable stars, transients, pulsars and fast radio bursts are presented. These concepts will help the reader to understand various data we used in other chapters.

2.1 The Variable and Transient Universe

Time-domain astronomy inspects the dynamic sky to enable the discovery and monitoring of transient and variable stars. The study of transients allows us to have wide-reaching and comprehensive information about the physics and evolution of many different types of stars. They also allow us to study the presence of materials along the line of sight and their surrounding environment. In addition, we can have a more comprehensive understanding of the effects of interactions of sources in binary systems and, also in some situations, identify unique sources in which general relativity and extreme physics theory can be tested. Therefore, time-domain astronomy opens many laboratories of astronomy, thus enabling us to explore the Universe in more details from a new perspectives.

The definition of 'transient' in astronomy is quite arbitrary. It usually concerns sources that were present in previously acquired data and that the sources afterwards disappear after a certain amount of time or vice-versa, i.e. never seen before but suddenly appear, for e.g. supernovae. In most situations, archival data may or may not reveal the quiescent counterpart to a new star/source due to the fact that the latter is largely dependent on the chosen wavelength or the instruments used during observations. For instance, in blue optical filters a bright flare may appear to be transient, however, with red filters during deep observations the flare can be revealed to be generated by an M-dwarf star. It is important to adopt a flexible definition of 'transient' in various contexts, as the origin of the definition is mostly observational. Generally, observable features and physical processes can be described as transient when they are short-lived when compared with stellar- and galaxy-evolution timescales.

The Universe can be analysed and explored across almost the whole electromagnetic spectrum. In astronomy when innovative instruments were crafted together with new 'window' to observe the sky at different wavelength range, unexpected phenomena were discovered that populate at phase-space regions formerly inaccessible. Spectacular examples are the discoveries of gamma-ray bursts (GRBs) at high energy (Klebesadel et al., 2004) and Jocelyn Bell's discovery of pulsars at radio wavelengths (Hewish et al., 1969). Most transient events are powered by mechanisms that make them visible at multi-wavelengths. The signatures of those mechanisms can reveal unexpected changes in duration and energy at different wavelengths. Thus only multi-wavelength observations both simultaneously and over time, can totally reveal the physics and the evolution of various astrophysical events, especially when they involve extreme objects, for example, massive stars, white dwarfs, neutron stars and black holes.

2.1.1 Observing transients in the optical

In the last two decades, the emergence of dedicated surveys have revolutionised our understanding of the transient sky. These include the Supernova Legacy Survey (Astier et al., 2006), the Australian SkyMapper (Keller et al., 2007), the Palomar Transient Factory (Rau et al., 2009), the Catalina Sky Survey (Drake et al., 2009), Pan-STARRS (Stubbs et al., 2010), the All-Sky Automated Survey for Supernovae (Shappee et al. 2014; Holoien et al. 2017), the Zwicky Transient Factory (ZTF, Bellm et al. 2019) among others. Such surveys often focus on transients evolving with characteristic timescales from minutes to several years and also include a broad range of events, both galactic and extragalactic. Example of transients include phenomena such as supernovae, novae, neutron stars, blazars, pulsars, cataclysmic variable stars (CV), gamma ray bursts and active galaxy nuclei (AGN). We provide a short description of a sample of these transient sources as follows.

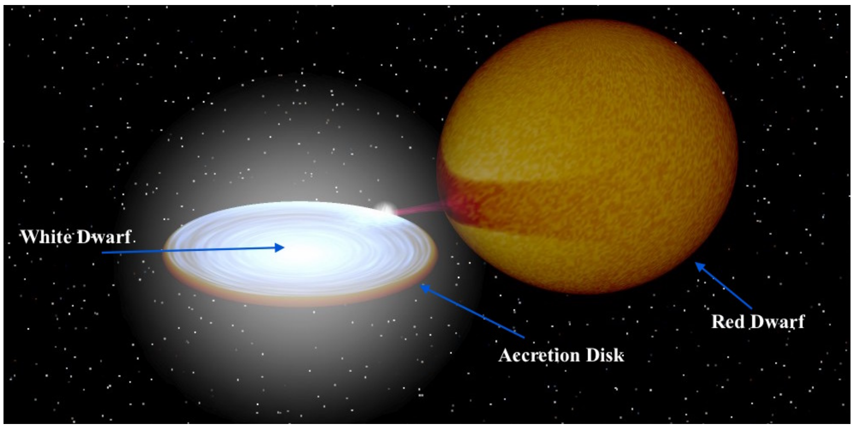


Figure 2.1 – The physical characteristic of a cataclysmic variable. The white dwarf is accreting materials filling its Roche-Lobe from companion star, in this example, a red dwarf. This figure was created by Rob Hynes using the BinSim binary visualization code (BinSim, 2021).

- **Supernovae**: they are astronomical events that occur in the last stages of a massive star's lifetime. When a star runs out of fuel and its fusion reaction pressure, that is the endothermic fusion of iron cannot withstand its own gravitational force, the core of the stars with masses $>8M_\odot$ (where $M_\odot$ is the mass of the Sun) collapses (Nadyozhin, 2008).

- **Cataclysmic Variable** (CV) Stars: CVs such as classical novae and dwarf novae, are binary star systems that consist of a primary white dwarf and a normal companion star. The stars are so close to each other that the white dwarf accretes matter from the companion stars, via a process called accretion (Knigge, 2011). This occurs due to the acceleration of matter by the strong gravitational field, from the companion star onto the white dwarf. A visualization of these kind of stars is illustrated in Figure 2.1. In addition, pulsating stars are common examples of variables detected in our Galaxy, for example Cepheids, RR-Lyrae, Mira-type, and RV-Tauri variables.

- **Active Galactic Nuclei** (AGN): Galaxies showing a very bright central core, with a luminosity that outshines the entire host galaxy are known as Active Galactic Nuclei, and galaxies hosting an AGN are named active galaxies. They are known to be the most luminous steady objects in the universe and their power output is variable on time scales of minutes up to years (Krawczynski & Treister, 2013).

- **Quasars** and **Blazars**: Both Blazars and quasars are types of AGNs. In addition to the fact that they emit high luminosity, they also emit a pair of perpendicular relativistic jets perpendicular to their accretion disks. These perpendicular jets, also known as beams, are rays of ionized matter expelled at bulk velocities of 95% - 99.9% the speed of light. These ionised materials are presumably generated by the active interaction of matter in

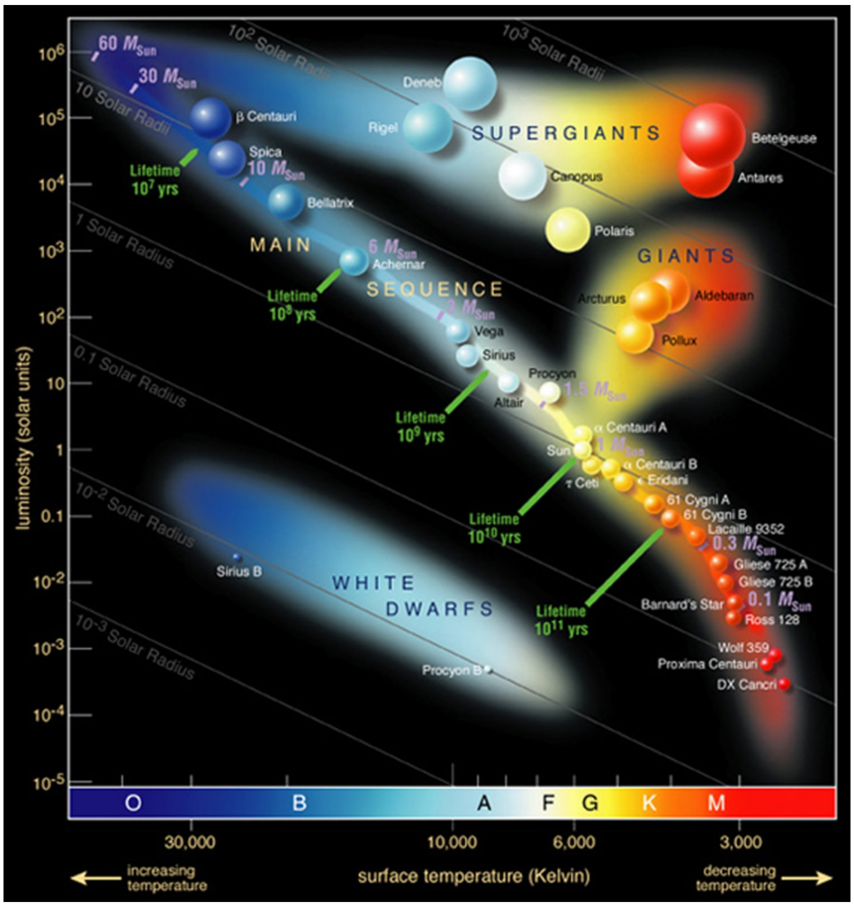


Figure 2.2 – In the Hertzsprung-Russell diagram the temperatures of stars are plotted against their luminosities. The position of a star in the diagram provides information about its present stage and its mass. Stars that burn hydrogen into helium lie on the diagonal branch, the so-called main sequence. When a star exhausts all the hydrogen, it leaves the main sequence and becomes a red giant or a supergiant, depending on its mass. Stars with the mass of the Sun which have burnt all their fuel evolve finally into a white dwarf (left low corner). **Credit**: European Southern Observatory (ESO).

the accretion ring that surrounds supermassive black holes.

In this chapter, we will mainly focus on some particular types of variables and transients (e.g. periodic variable stars, pulsars and fast radio bursts) as this thesis is mainly concerned with the classification and discovery of these transients. In the sections below, we will introduce a few terms and concepts about light curves, the Hertzsprung - Russell diagram and variable stars.

2.1.2 The Hertzsprung-Russell (H-R) Diagram

It was found that when the absolute magnitude (M_V), that is, the intrinsic brightness of stars is plotted against their temperature (stellar classification), we observe that stars are not randomly distributed but are mostly restricted to a few well-defined regions. However, another interesting fact shows that as the physical characteristics of a star change over time, its position on the

H-R diagram also changes. Therefore, the H-R diagram can be viewed as a graphical plot of stellar evolution. Knowing the location of a star on an H-R diagram, can reveal information about this particular star, for example, its luminosity, spectral type, color, temperature, and age among others.

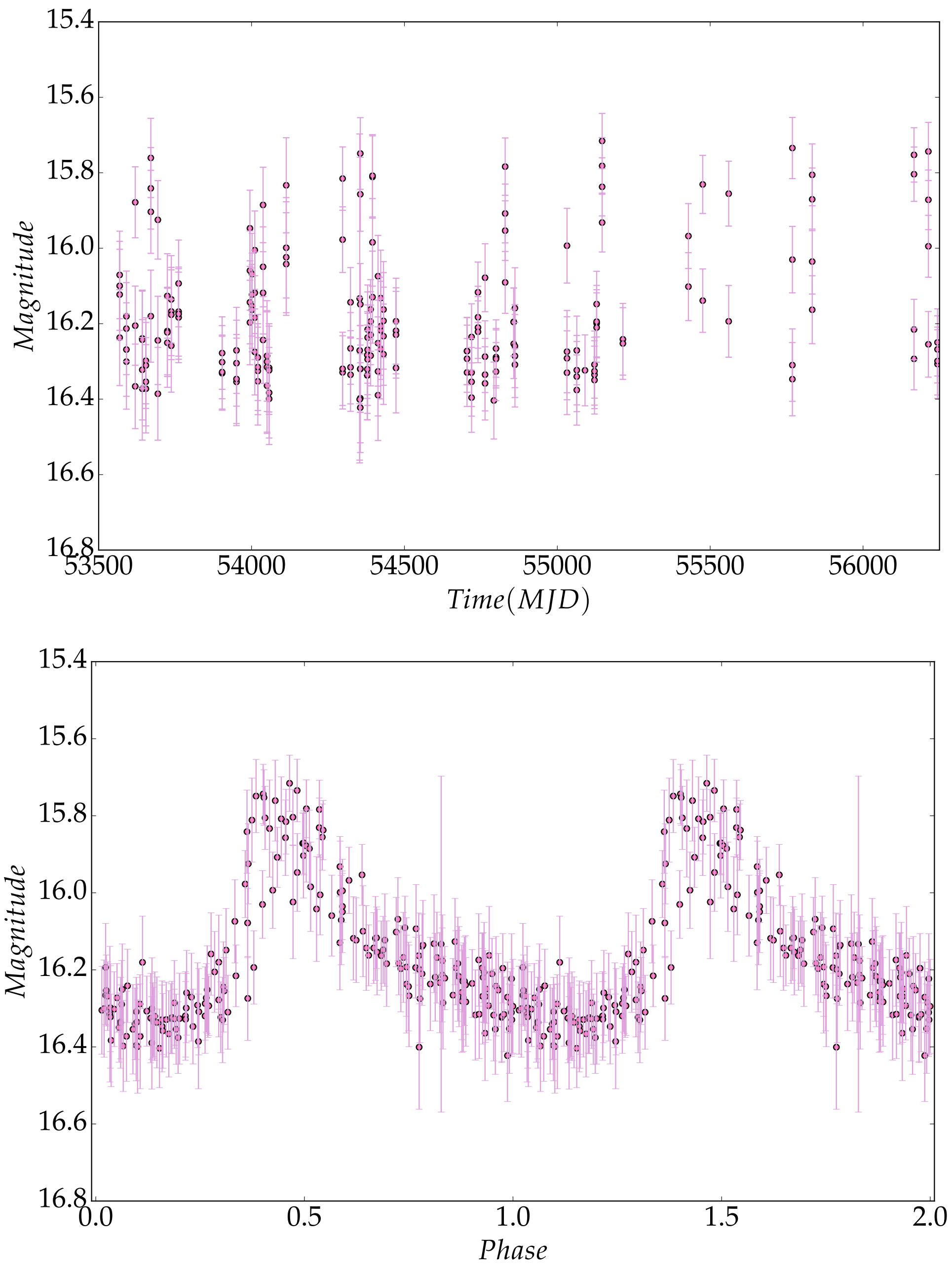


Figure 2.3 – An example of a δ-Scuti variable star light curve from the Catalina Real-Time Transient Surveys (CRTS, Drake et al. 2017) with magnitude on the y-axis and time on the x-axis (top panel). The bottom panel illustrates a phase-folded light curve for similar variable star, with a two phase-cycles on the x-axis.

Most stars can be classified by their:

- temperature, that is, spectral types from hottest to coolest,
- luminosity, that is, the differences in spectral lines among stars that have similar spectral

type are a function of the star's radius. These differences result in different luminosities. The luminosity (L) of a star is directly linked to its absolute magnitude (M_V). Therefore, luminosity can be defined as the total amount of energy radiated per second, that is, it is proportional to fourth power of temperature. It means that if two stars are at the same effective temperatures but have different luminosities, they must differ in size.

Figure 2.2 illustrates the H-R diagram. From the plot we can observe a main-band, also known as the main sequence, that starts at the lower right-hand corner and curving up to the left-hand corner. Almost 90% of all stars lie within the main sequence and these stars are burning their hydrogen into helium. Stars at the top left-hand corner are bright, hot and from O and B spectral classes whereas stars in the lower right-hand corner are cooler, dimmer and are of K and M spectral classes. On the main sequence, stars undergo an evolutionary process that depends on the rate of hydrogen fusion occurring in their cores. Stars on the main sequence remain at fixed position on the H-R diagram when there is a state of dynamic equilibrium. The latter occurs when the radiation pressure pushes outward from the hydrogen fusion mechanism and balances the inward pull of gravity. However, stars in the main sequence start to moves off along the band when the two forces no longer balanced. This happens when there is a depletion of hydrogen, thus the radiation pressure decreases and can no longer balances the gravitational forces. These evolutionary stages are also dependent on the initial mass of the star. In addition, there are specific locations on the H-R diagram that are occupied by giants and supergiants that exhibit types of variability. These stars have transitioned from the main sequence and are fusing heavier elements.

2.1.3 Light Curves

Stars show some variability in brightness/magnitude during the evolutionary process. The brightness that we observe on Earth depends on its distance and its absolute magnitude. The characteristics and behaviour of a star that show a variation in magnitude can be analysed by measuring their changes in brightness/magnitude over a period of time. We can therefore plot the variation on a graph also known as a *light curve* (LC).

Astronomical observations are taken with sophisticated telescopes from a region of the sky for a period of time. This produces a sequence of images at various times of observation. Using a technique in astronomy called *photometry*, which enables the precise measurement of the apparent magnitude of a source from the sequence of images over a period of time from which we can form a light curve, that is, a temporal sequence of brightness measurement over time for a particular source of interest. Light curves are usually plots of apparent magnitude (y-axis) over time (x-axis). An example of a light for a variable star is illustrated in Figure 2.3. If the source shows some periodic behaviour and its period can be determined, it can be useful to plot the phased light curve. The phase ϕ of an observation can be calculated as,

$$\phi = \left(\frac{t - t_0}{P}\right) - \mathbb{E}\left(t\right), \tag{2.1.1}$$

where P is the period, t_0 is an arbitrary starting point, t is the time when the observation was taken and $\mathbb{E}\left(t\right)$ is the integer part of $\left(\frac{t-t_0}{P}\right)$. Generally, the phase is expressed as the fraction of the period of variability, taking values in the range $[0, 1]$.

2.2 Variable Stars

A variable star is a star whose brightness fluctuates over time. Historically, the study of variable stars have been the principal method for determining the content and structure of stellar systems and the Universe. Among these stars, we can categorise them into two types: *intrinsic* and *extrinsic* variable stars.

Figure 2.4 shows a 'variability tree', which illustrates a visual summary of various types of variable phenomena. In this diagram there are three levels. In the first level, there is the classical division between extrinsic and intrinsic variables. In the second level a distinction is made based on the origin of the variability. In the extrinsic variability group, the stars considered are the rotation and eclipses by a companion or by a foreground object. Among the former, are the eclipsing binaries. The latter is composed of the classes: Beta Persei (EA), Beta Lyrae (Semi-detached) & the W Ursae Maj (contact) and rotational variables consist of Ellipsoidal (ELL) & RS Canum Venaticorum.

In the second level itself we move to the intrinsic variable objects. We find the eruptive variables, the cataclysmic variables and finally the pulsating variables. Arguably the most important group among the intrinsic variables is that of pulsating variables because it contains the RR Lyrae and Cepheids classes. These two classes exhibit relations between their periods and absolute luminosities that allow us to estimate distances, a quantity both fundamental and elusive in Astronomy (Feast & Walker, 1987; Freedman, 1988; Walker, 1988; Madore & Freedman, 1991). For a current review of the physics and phenomenology of pulsating stars, we refer the reader to Catelan et al. (2013).

As we can see variable stars are divided into several classes and subclasses, depending on

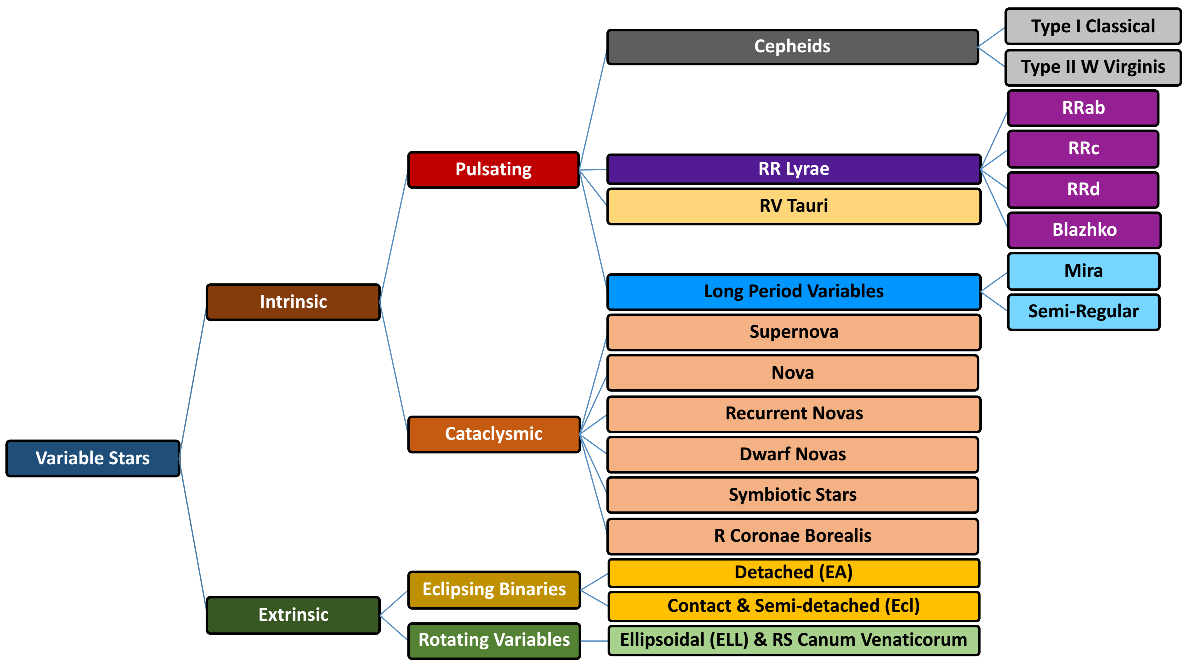


Figure 2.4 – Variability tree showing the many different types of stellar (and non-stellar) phenomena that are found in astronomy.

their period of pulsation and the shape of their light curves. In this thesis, we mainly focus on the classification of periodic light curves of pulsating stars, rotational variables and of binaries. The classification schemes are discussed in Chapters 4 & 5. In the next Section, we discuss mostly about the subset of variable stars that we use to build a classifier in Chapters 4 & 5.

2.2.1 Pulsating variables

The interior of a star is not directly observed, but by studying stellar pulsations one can determine the internal structure of stars. On the H-R diagram studied in §2.1.2, we observe that stars are not always in complete equilibrium, but seem to pulsate. The regions on the H-R diagram where pulsations occur, are called the *instability* regions. There are two major types of pulsation modes: gravity (g-modes) and pressure modes (p-modes). Pressure is the main driving force that causes a star to be perturbed from its equilibrium, this gives rise to acoustic waves, also known as pressure modes (p-modes). The latter largely produces radial motion, while for gravity modes, buoyancy is the restoring force that gives rise to perturbations, thus causing horizontal motions (Aerts et al., 2010).

A *radial* pulsation occurs when a star starts oscillating around its equilibrium state by changing its radius while maintaining its spherical shape. If a single overtone is excited, these radial pulsations will appear as mono-periodic variability in their light curves. Examples of well-known stars that show radial oscillations are the Cepheid and RR Lyrae stars. However, a *non-radial* pulsation occurs when some regions of the stellar surface of a star move inwards, while, simultaneously, other parts move outwards.

Pulsations occur in almost all stars and at various stages of stellar evolution. As discussed in §2.1.2, most stars are found along the main sequence, where hydrogen burning is occurring in their core. Along the main sequence, there are various types of variable stars, from low-mass to high-mass solar-like stars. When hydrogen is depleted in their core, helium burning takes over as an energy source. This process causes the helium core to contract while the outer regions of the star expands. These stars evolve towards cooler temperatures where classes of pulsating stars reside along the horizontal and red-giant branch.

We consider here the most common variables such as δ-Scutis, RR Lyrae, Cepheids, Long Period Variables (LPVs: Mira, and semi-regular variables). We focus in particular on periodic variable stars that are distinguishable with the CRTS (Drake et al., 2009).

- **δ Scuti stars**: The δ Scuti stars are Population I stars, found at the intersection of the instability strip with the main-sequence. Both radial as well as non-radial pulsations are observed in δ Scuti stars. Generally, they are of low order p-modes with a range of periods between 18 minutes to 8 hours. They have masses in the range from 1.5 to 2.5 $M_{\odot}$ and are often of A0 to F5 type giant or main sequence stars.

- **RR Lyrae**: RR Lyrae stars are of great galactic and cosmological importance as they are good distance indicators. They are population II, low metallicity, low-mass stars with masses varying in the range 0.6-0.8 $M_{\odot}$ and spectral types ranging from A2 to F6. They are found commonly in globular clusters, which are dense groups of old stars in the halos of galaxies. They are regarded as classical radial pulsators, having a pulsation period of nearly half a day. RR Lyrae stars are divided into RRab, RRc, RRd and Blazkho as they pulsate in different modes - fundamental radial mode or in the first-overtone radial mode respectively. For instance, RRab have longer pulsation periods (from 0.3 to 1.2 days) and asymmetric light curves, where as RRc have shorter periods (0.2 to 0.5 days) and more or less sinusoidal light curves. RRd stars are double-mode pulsators, in which both the fundamental and the first overtone are excited. However, many RR Lyrae are known to exhibit the Blazhko effect (long-term modulation; Blažko 1907a).

- **Cepheids**: Cepheids are Population I giant or supergiant stars with masses above 5 $M_{\odot}$ and spectral types ranging between F5 and G5. Cepheids expand and contract in a repeating cycle of size changes. The change in size can be observed as a change in apparent brightness (apparent magnitude). Together with the RR Lyrae, they are considered to be radial pulsators, with periods in the range 1-50 days and their light curves display a skewed characteristic distribution. Cepheids can also be divided in to the Classical Cepheids (Type I), the Double-mode Cepheids (Type II) and Anomalous Cepheids (ACEPs) classes. Double-mode Cepheids (Type II) have either both the fundamental and the first overtone, or the first and second overtone, excited. The different Cepheids types obey different period-luminosity relations. In general, Type I Cepheids are brighter than Type II Cepheids, if both have the same period.

- **Long Period variables (LPVs)**: They are Population I radial pulsators, located nearly at the red end of the instability strip and they have periods in the range longer than 80 days. Also, they show luminosities between the range of about $1 \times 10^3 L_\odot$ and $7 \times 10^3 L_\odot$ and have low effective temperatures between 2500 and 3500 K. LPVs can be categorised into two subclasses; **Mira** and **Semi-regular**. **Mira** variables are periodic pulsating red giants with a periods of 80 to 1000 days. It is a stage that most mid-sized main sequence stars transition through as they evolve to the red giant branch. **Semi-regular** variables are giants and supergiants showing periodicity accompanied by intervals of semi-regular or irregular light variation. Their periods range from 30 to 1000 days.

2.2.2 Rotational and Eclipsing Binaries

Many stars belong to a binary system. Binaries will always be present in any unbiased sample of stars. In this work, we will consider eclipsing binaries (contact plus semi-detached binary group (Ecl) and detached eclipsing binaries (EA)) and rotational variable stars such as ellipsoidal and RS Canum Venaticorum variables.

Stars in eclipsing binary systems have fundamentally allowed the computation of masses of their components based on their orbits. The calculation of the masses can further be used to constrain other fundamental parameters, such as density and radius and can also be used to determine an empirical mass-luminosity relationship (MLR) from which the masses of single stars can be calibrated. Hence, binaries play an important role in astrophysics.

- **Contact binaries**: They are also known as both W Ursae Majoris (W UMa's) stars or EW's. Usually these stars have very similar temperatures and spectral types and exhibit eclipses with symmetric shapes. However, we sometimes observe slight differences in temperature and size, and this gives rise to slight differences in eclipse depth. Generally, contact binaries are known to have minimum periods near 0.22 d (Rucinski 1996, 2007). However, Drake et al. (2014a) have discovered that a small number of systems may have shorter periods.

- **Detached binaries**: Detached eclipsing binaries are gravitationally bound systems. They are well-separated stars whose orbital plane is aligned closely along our line of sight. Detached binaries have highly elliptical orbits. Generally, their light curves show that the times of occurrence of their primary and secondary eclipse are not equally separated.

- **Semi-detached binaries**: The depths of the primary and secondary eclipses of semi-detached binaries are noticeably different. However, such systems can be difficult to distinguish from detached systems while those with smaller eclipse amplitudes are not easy to identify from contact binary systems.

- **Rotational variables**: For example **Ellipsoid variables (ELL)** are close binary systems with ellipsoidal shapes (not a 'perfect' sphere). These binary system do not show eclipses. Their combined brightness changes with period because of changes in emitting areas projected towards an observer. Another example is the **RS Canum Venaticorum variables (RS CVns)**. They are spotted stars with periods of less than 1 day for main-sequence stars, up to hundreds of days for giants. These stars have prominent bright and dark areas on their surface that result in small variations in brightness as they rotate in and out of our line of sight.

Up till now, we have discussed that various types of variable stars that are investigated in Chapter 4 & 5. In this thesis, we have also looked at the classification of radio data, for instance single pulses/pulsars/fast radio bursts and radio frequency interference in Chapters 7 & 8. In the next section, a background introduction on pulsars and fast radio bursts is given.

2.3 Observing transients at the radio wavelengths

The discovery of radio pulsars over a half century ago was a seminal moment in astronomy. It demonstrated the existence of neutron stars, and has allowed us to probe strong gravity, dense matter, and the interstellar medium.

2.3.1 Pulsars

Pulsars are rapidly rotating radio sources, that is, rotating neutron stars that possess typical magnetic field strengths that range between $10^8 - 10^{14}$ Gauss. The first pulsar was discovered in 1967 by Jocelyn Bell and Antony Hewish at Cambridge University (Hewish et al., 1968; Pilkington et al., 1968; Bell Burnell, 1977). They found a "pulsating" radio source while determining the angular sizes of compact sources using interplanetary scintillation (IPS). At first the source was thought to be terrestrial, for e.g. radio frequency interference (RFI). However, the dispersed nature of the signal and its frequency indicated that the source was of celestial origin and was characterized as either a white dwarf or a neutron star (Hewish et al., 1968). The pulses were theorised to be coming from radial pulsations of a compact star with a measured stable magnetic field strength of 10^7 Gauss.

The origin of the radio emission produced by pulsars, actually comes from their magnetosphere (Ghosh, 2007). A magnetosphere can be defined as a region of space that surrounds a pulsar in which charged particles are influenced by both open and closed magnetic field lines, known as the co-rotating magnetic field (Lorimer & Kramer, 2004). The radio emission is broadly explained by the acceleration of charged particles along the magnetic field lines of the rotating, not pulsating, neutron star (Gold, 1968). The magnetic field strength can be calculated by assuming the period, P and the slow-down rate, $\dot{P}$ of the pulsar, is due to magnetic dipole braking (Lorimer & Kramer, 2004). The surface magnetic field strength of a pulsar is expressed as:

$$B_s \equiv B\left(r = R\right) = \sqrt{\frac{3c^3}{8\pi^2}\frac{I}{R^6 \sin^2\alpha}P\dot{P}}. \tag{2.3.1}$$

By taking the values of the moment of inertia, $I = 10^{45}$ g cm^2 and radius, R= 10 km for a pulsar and assuming the angle between the magnetic moment and the spin axis, $\alpha = 90^\circ$, the surface magnetic field is re-written as:

$$\boxed{B_s = 3.2 \times 10^{19} G\sqrt{P\dot{P}} \approx 10^{12} G\left(\frac{\dot{P}}{10^{-15}}\right)^{\frac{1}{2}}\left(\frac{P}{1s}\right)^{\frac{1}{2}}.} \tag{2.3.2}$$

It is believed that the strong magnetic field causes particles to be extracted from the surface of the neutron star surface, thus creating a voltage gap. The extracted particles are then accelerated along the co-rotating magnetic field lines of the magnetosphere (Lorimer, 2008), which result in an increase in energy, hence the particles emit radiation in the form of high energy

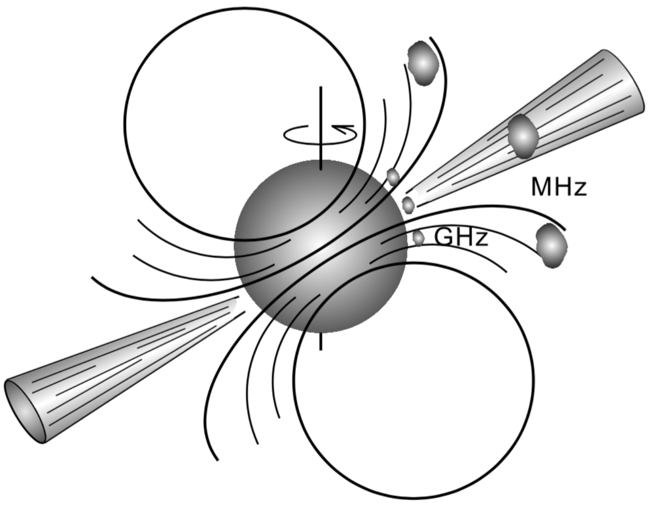


Figure 2.5 – Schematic diagram that illustrates the radius to frequency mapping in radio pulsars. At higher frequencies, the emission occurs closer to the neutron star surface as compared to the emission at lower radio frequencies, resulting in wider beams at low radio frequencies. Figure courtesy of Seiradakis & Wielebinski (2004).

photons. The photons now interact with the strong magnetic field of the pulsar and thus undergo the process called pair production (creating opposite elementary particles - an electron and positron). The particles radiate energy as they are accelerated along the open field lines. It causes the emission of radio waves and other various forms of radiation along the open field lines near a pulsar's magnetic pole, producing a coherent radiation beam. The magnetic axis of a pulsar is often inclined from the direction of its rotational axis. Therefore, the radiation beam produced at the magnetic poles is swept at an angle across the sky as the pulsar rotates. If the beam passes the line of sight of an observer, it is detected as a rise and fall in the broadband radio emission. This is known as the "Lighthouse effect" proposed by Gold (1968), that is, the radio pulses were attributed to the fact that the beam swept past our line of sight once per rotation.

2.3.1.1 Effects of the Interstellar Medium

The interstellar medium (ISM) consists of gas in the form of either plasma or neutral atoms. The most influential part of the ISM for radio signals from a pulsar, is the plasma. It affects the signals produced by pulsars in many ways and produces path differences and distorts the wave fronts of the pulsars such that the observed flux density varies randomly with time. The pulsar signal gets modulated through dispersion, Faraday rotation, scattering and scintillation. These effects impede our ability to observe and detect pulsars as their signals become indistinguishable from the background noise due to the smearing of the signals. In the following section, we discuss dispersion as it is the main interstellar effect that renders the search for pulsars difficult.

2.3.1.2 Dispersion

Dispersion causes the radio emission from a pulsar at lower frequencies to arrive later (Lorimer & Kramer, 2004). As pulsars are weak sources, modern pulsar searches use a wide bandwidth to improve sensitivity. Therefore, their pulse profile may be spread out by dispersion. When the radio emission from the pulsar travels through cold ionised plasma (ISM), it experiences a frequency dependent refractive index, where the refractive index μ for an observing frequency f is given by

$$\boxed{\mu = \sqrt{1 - \left(\frac{f_p}{f}\right)^2},} \tag{2.3.3}$$

where f_p is the plasma frequency given by

$$f_p = \sqrt{\frac{e^2 n_e}{\pi m_e}} \simeq 8.5 \left(\frac{n_e}{\mathrm{cm}^{-3}}\right)^{\frac{1}{2}} \mathrm{kHz}, \tag{2.3.4}$$

where e, m_e and n_e are the charge, mass and number density of the electron respectively. It is clear from Equation 2.3.3 that due to the differential speed of the propagating wave, the pulse at a higher frequency will arrive earlier than that at a lower frequency. As a consequence, the propagation of pulsar signal of frequency f to a distance d will be delayed in time with respect to a signal at infinite frequency. The time delay is

$$t = \left(\int_0^d \frac{dl}{v_g}\right) - \frac{d}{c}, \tag{2.3.5}$$

where d is the distance of pulsar from Earth and v_g is the group velocity of the propagating wave. We then substitute for $v_g = \mu c$ and considering that $f_p \ll f$, we obtain

$$t \simeq D \times \frac{DM}{f^2}, \tag{2.3.6}$$

where the Dispersion Measure (DM), that is, integrated free electron density along the line of sight is given by

$$\boxed{DM = \int_0^d n_e dl,} \tag{2.3.7}$$

which is expressed in pc cm^{-3} where the Dispersion constant, D is

$$D \equiv \frac{e^2}{2\pi m_e c} = (4.148808 \pm 0.000003) \times 10^3 \mathrm{MHz}^2\,\mathrm{pc}^{-1}\,\mathrm{cm}^3\,\mathrm{s}. \tag{2.3.8}$$

The delay between two observed frequencies f_1 and f_2 is given by

$$\Delta t \simeq D \times \left(f_1^{-2} - f_2^{-2}\right) \times DM. \tag{2.3.9}$$

As DM is not known a priori, it must be searched over to find a new pulsar. By measuring pulse arrival times at the two different frequencies, the DM of a pulsar can be estimated along the line of sight. Then, according to Equation 2.3.7, the distance of the pulsar can be estimated by integrating Equation 2.3.7 if n_e is known from standard models like the NE2001 model (Cordes & Lazio, 2002).

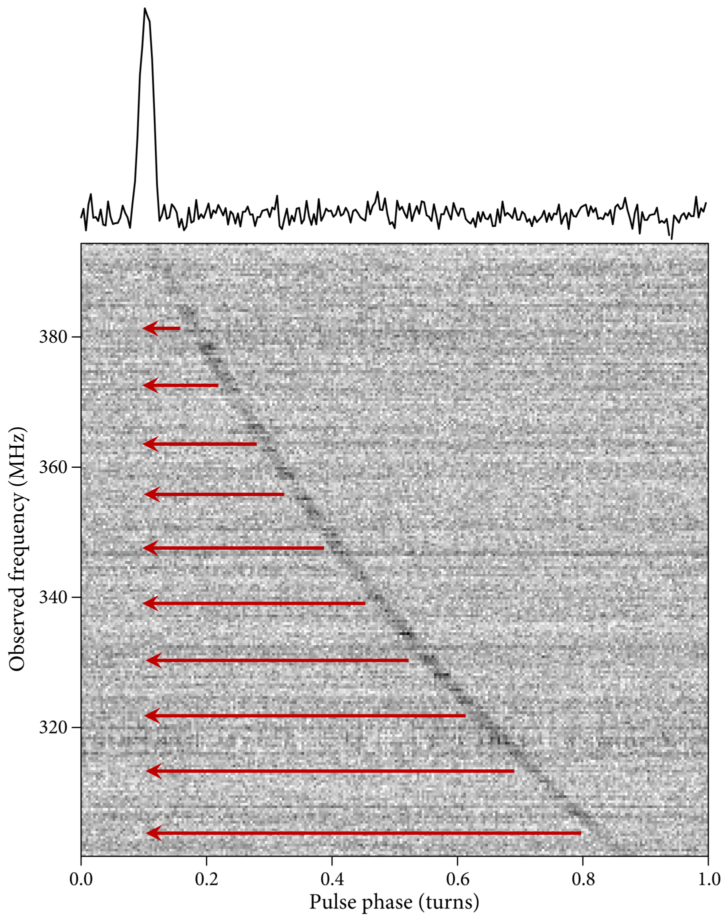


Figure 2.6 – Illustration of the importance of de-dispersion. The grey scale plot represents the uncorrected dispersive delay, the x-axis represents the pulse phase as a function of frequency from an observation of the pulsar J1800+5034. Directly integrating the data across the frequency dimension smears the pulse in phase/time and makes it difficult to detect. Hence, the dispersion delay must be removed by shifting every frequency channel individually by the correct number of time/phase samples (red arrows). The top panel illustrated the "de-dispersed" band-integrated pulse profile. Based upon diagram originally presented by Wang (2017) in Chapter 6.

2.3.1.3 Searching for Pulsars and Fast Radio Bursts

Searching for pulsars/fast radio bursts requires two fundamental search tools. A large radio telescope sensitive to weak radio emissions, and a signal search pipeline that is capable of isolating weak signals from noise/Radio Frequency Interference (RFI).

Radio telescopes comprise of typically an aperture that guides electromagnetic signals from the sky to a focus. The signals can be measured as a function of time using feeds. The feed response is measured over a range of radio frequencies. It can be imagined as a bandwidth, which is amplified and sampled discretely by a number of frequency channels (Petroff et al., 2019). High-time-resolution observations, similar to those utilized to search for pulsars and FRBs, record the stream of voltages in each channel over a period of time, and sampled the voltage stream at some finite time resolution.

These data are saved to disk in the voltage data format. It can further be downsampled by summing adjacent time or frequency channels, thus decreases the resolution and volume of the data. In the case of two orthogonal antennas, both polarizations are recorded, thus at this stage they may be summed. The resulting data cube of intensities at each time and frequency can be saved to disk in the format of 'filterbank' file. There are several steps that are required to search for dispersed pulses in these data cubes. In some situations, this involves a pre-processing step to sum the polarizations, if they are recorded separately. Afterwards, the total intensity data are analysed to generate a list of candidate FRB signals. The first step involve removal of noise/RFI. In this work, we developed an automated pipeline using artificial intelligence to perform distinction between noise/RFI and single pulse candidates in FRBs search pipeline (see Chapter 7).

2.3.1.4 Pre-processing step of radio frequency interference excision

Radio frequency interference (RFI) is ubiquitous in radio astronomical data. RFI can overwhelm the intensity of astrophysical signals and in some cases, can look sometimes fairly similar to an astrophysical signal by showing some of the expected characteristics (for example, a frequency-dependent sweep in time that appear like astrophysical dispersion, see Foster et al. 2018). In most FRBs searches and pipelines, before the data is searched for pulses an initial attempt is made to separate or remove RFI. The most common techniques entail masking time samples and frequency channels. For instance, if there are known in-band emitters, the corresponding frequency channels can be masked automatically (Petroff et al., 2019). Also, the data can be searched for impulsive RFI by searching for peaks in the DM = 0 cm^{-3} pc time series and masking the contaminated time samples (Kocz et al., 2012). The goal is to mask as much RFI as possible, without removing an astronomical signal. In Chapter 7, we provide a detail overview of the machine learning pipeline to automate the classification between real candidates (single pulses) and those that are noise or RFI for the MeerKAT facility.

2.3.1.5 Stages in the detection of pulses

Signals received from radio telescopes are readily interpreted by digital filter banks. Filter bank file is an array of band-pass filters. It is utilized to split up the pre-processed radio signal into a number of frequency channels. A received signal in the filter bank file is split up in to $n_{channels}$ frequency channels, each of width Δv. Each channel consists of a total number of samples of the signal, s_{total}, that is taken at a time interval, $t_{sampling}$ (μs), over a length of observational time, $t_{observe}$ (s), such that,

$$s_{total} = \frac{t_{observe}}{t_{sampling}}. \tag{2.3.10}$$

The filter bank file acts as input to the software search pipeline. The software comprises of various stages applied to the filter bank data. The first step is the RFI excision, which involves the removal of channels that contain known noise/RFI. Once RFI excision has been applied, the next step known as 'Clipping' (Keith et al., 2010). 'Clipping' is achieved by setting the samples, which exhibit higher intensities than some threshold intensities, to zero. After the data has undergone a cleaning process, the pre-processed data now enters a highly computationally expensive stage known as *de-dispersion*.

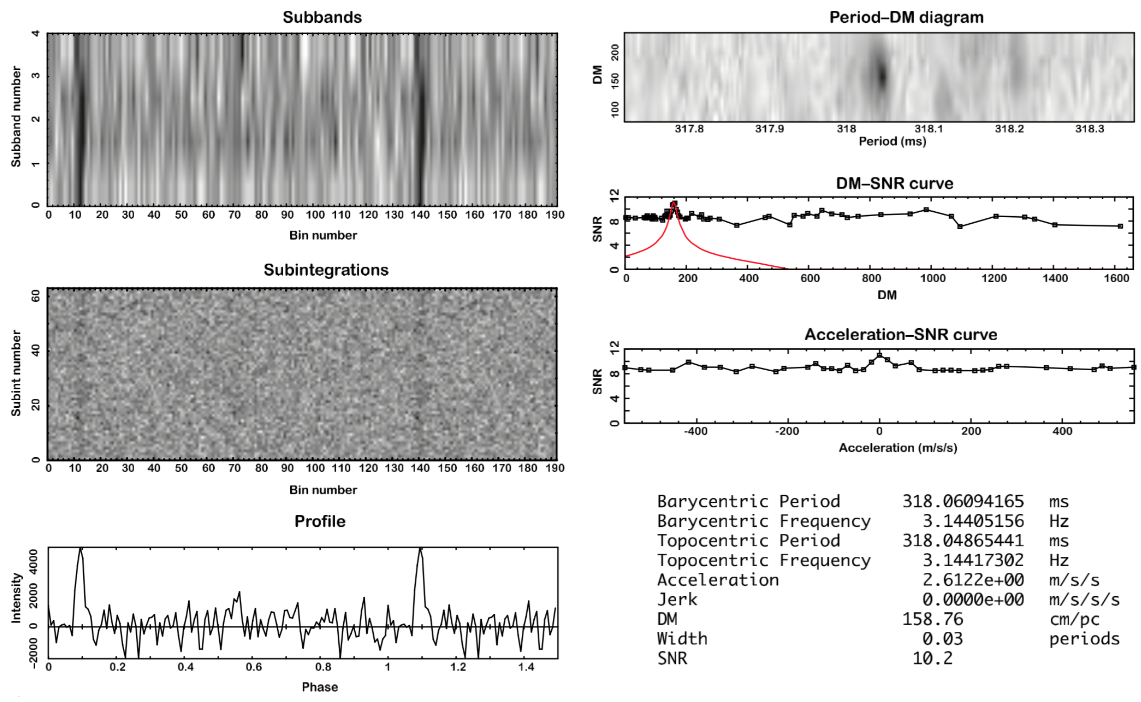
As described in §2.3.1.1 & 2.3.1.2, signals travelling through the ISM are affected in various ways. These signals interact with charged particles and cause a delay in the arrival of the signal on earth. This is viewed as a dispersive effect that causes a smearing effect in time of the pulsar signal. This smearing effect makes it hard to detect pulsar signals, as their pulses are less pronounced, i.e. their pulses show a reduction in Signal-to-noise ratio (S/N). Therefore, a signal receives an amount of dispersive smearing, and this quantity is proportional to the so-called dispersion measure (DM) (Lorimer & Kramer, 2004). However, the DM value of a particular signal is not known a priori, thus a number of DM tests or '*DM trials*' must be carried out to find this DM value as accurately as possible. Using an accurate DM value will remove the smearing effect of the signal, thus maximizing its S/N. For each DM-trial, each frequency channel is shifted by an appropriate delay as illustrated in Figure 2.6. Each successive DM-trial increases the delay in steps, until a maximum DM is reached. This process generates a single ‘de-disperse’ time-series per frequency channel. Then, per trial DM, these generated time-series are summed to produce a single de-disperse time-series. In overall, the process of de-dispersion generates a number of time-series that is equal to the total number of DM-trials.

The next step in pulsar searching is known as *periodicity search* (Lorimer & Kramer, 2004) and it involves a Fourier analysis. The periodicity search step filters the data to remove strong spectral features, also known as ‘birdies’ (Manchester et al., 2001; Hessels et al., 2007). Strong spectral features are commonly caused by periodic or quasi-periodic interference. The next step involves a summation of the amplitudes of harmonically related frequencies to their corresponding fundamentals. This is an important step as in the Fourier domain, the power from a narrow pulse is distributed between its fundamental frequency and its harmonics (Lorimer & Kramer, 2004). Thus it would have been unlikely to find a pulsar signal, if one would try to only detect the fundamental. After the summation process, periodic detections with Fourier amplitudes above certain threshold, are regarded as ‘suspect’ periods.

Hence, a further process is applied to the ‘suspect’ periods, also known as *sifting* (Stovall et al., 2013). Sifting removes duplicate detections of the same signal at slightly different DMs, along with their related harmonics. After the sifting process, a large number of ‘suspect’ periods are remained. The output of this Fourier search is a list of candidates that are characterised by some best-fit search parameters, for e.g. DM, frequency, number of harmonics summed and acceleration. These parameters are not sufficient to provide a distinct boundary and distinction between legitimate pulses and noise/RFI. To be able to detect pulses in the data, we need to return back to the original, multi-channel data and phase-fold them at the candidate period. This step will generates an integrated pulse profile of the candidate, along with additional diagnostic plots as shown in Figure 2.7 and detailed in §2.3.1.6.

2.3.1.6 Pulsar Candidates

A pulsar candidate is a signal detection that exhibits ‘pulsar like’ characteristics (Lorimer & Kramer, 2004; Eatough et al., 2010). These characteristics can be used for further analysis. Using graphical plots and statistics, each candidate summarises such a detection. By either an automated method, or a human expert, these plots and statistics must be inspected in order to determine a candidate’s probable origin. The causes of the vast majority of candidates are due to fluctuations in Galactic background noise, electronic noise of the receiving system and terrestrial RFI. However, only those with probable pulsar origin will be further analysed. The process of determining which candidates are deemed suitable to investigate, is known as candidate 'selection', which is a fundamental step in the search for pulsars. When correct selection decisions have been made, they allow valuable telescope time to be prioritized upon those detections and this yields a new discovery, and prevents pulsars from being missed.



```
Barycentric Period      318.06094165   ms
Barycentric Frequency     3.14405156   Hz
Topocentric Period      318.04865441   ms
Topocentric Frequency     3.14417302   Hz
Acceleration              2.6122e+00   m/s/s
Jerk                      0.0000e+00   m/s/s/s
DM                      158.76         cm/pc
Width                     0.03         periods
SNR                      10.2
```

Figure 2.7 – Example pulsar candidate plot of PSR J1926+0739. The bottom left of the plot displays: the integrated pulse profile, folded at the optimum period and dispersion measure (DM). The middle left shows the 64 temporal sub-integrations of the observation, illustrating how the pulse varies with time. Top left illustrates the stacked pulses across four frequency sub-bands, showing how the pulse varies with observing frequency. In addition, the period-DM diagram illustrates how the SNR varies with small changes in the folding period and DM. The DM-SNR curve shows the spectral SNR as a function of a wide range of trail DMs, and finally the acceleration-SNR curve illustrates the spectral SNR as a function of trial acceleration. The candidate was discovered by Eatough et al. (2010) using an Artificial Neural Network.

The basic candidate is described in terms of a small number of characteristic variables and plots. According to Eatough et al. (2010), common features of a pulsar candidate include: a narrow pulse width (~ 5% of the pulse period); the presence of a pulse over all the sub-integrations; pulse phase coherent emission across all frequency sub-bands; an 'island' in the period-DM diagram; good agreement between the real and theoretical DM-SNR curve, and a good agreement between the real and theoretical acceleration-SNR curve (Eatough et al., 2010). Figure 2.7 (upper left) shows the candidate plot of PSR J1926+0739. This candidate shows how the signal persists throughout both the time and frequency domains (Eatough et al., 2010). In addition, the integrated pulse profile is shown in Figure 2.7 (bottom left). The IP profile is the result of averaging across all observed frequencies and time. The period-DM plane and DM-SNR curves on the right of Figure 2.7 are used to describe the relationship between trial DM values, and the S/N as the DM value is not known a priori, thus it must be searched over to find a new pulsar.

2.3.2 The pulsar population

Given the very long lifetimes of pulsars, ~ 10's of millions of years, for 'normal' pulsars and a few billion years for milliseconds pulsars, what one essentially wants to do when studying pulsars is to analyse a large number of them to construct a population to find out how they form and evolve in various situations. The current pulsar population is mostly found in the Galaxy due to the weak nature of their emission. Currently, there are 28 pulsars that are the farthest and they are located in the Magellanic clouds (Ridley et al., 2013). Out of the remaining, 144 are located in Galactic globular and other clusters. This means that out of about 2500 pulsars currently detected (Manchester et al., 2005), almost 93% pulsars are in the Galactic disk of the Milky way Galaxy.

Due to the interstellar medium (ISM) effects, there would be an observational bias against detecting highly scattered and weak, distant pulsars. Another bias is due to the narrow radio beam of the pulsars which implies that there may be active radio pulsars not beaming towards the Earth.

Radio pulsars possess one unique property which allows us to construct a diagram for them called the $P - \dot{P}$ diagram. Pulsars are effectively large rotating dipole magnets in space and so they lose energy by spinning down and some of this energy manifests as radio emission. They therefore emit radiation at the expense of their rotational kinetic energy. This loss of energy causes the pulsars to slow down over time. This rate can be very precisely determined by doing pulsar timing measurements and is called the rate of slow-down or the period derivative $(\dot{P})$. These characteristics allow us to construct a diagram depicting the pulsar population as shown in Figure 2.8. The plot has been made using the psrqpy package[*], which is an interactive python tool within the ATNF pulsar catalogue[†].

The $P - \dot{P}$ diagram illustrates that normal pulsars and the millisecond pulsars (MSPs) are two distinct populations. It is observed that the normal pulsars are clustered towards the centre of the plot while, the MSPs form a cluster towards the bottom left, having much smaller periods as well as period derivatives when compared to normal pulsars. The lines of constant magnetic field strength which is given by Equation 2.3.2, characteristic age of detected pulsars is calculated as in Equation 2.3.11 and spin-down luminosity are also shown in the $P - \dot{P}$ diagram.

The characteristics age of pulsars can be inferred from

[*]http://psrqpy.readthedocs.io/en/latest/
[†]http://www.atnf.csiro.au/people/pulsar/psrcat/

$$\tau_c \equiv \frac{P}{2\dot{P}}. \tag{2.3.11}$$

From Figure 2.8, it is seen clearly that MSPs are much older and have much lower magnetic fields as compared to the normal pulsars but are still almost as luminous. This led to the theory that MSPs are born as normal pulsars. Later, they turn into MSPs through mass transfer from their binary companions (Bhattacharya & van den Heuvel, 1991).

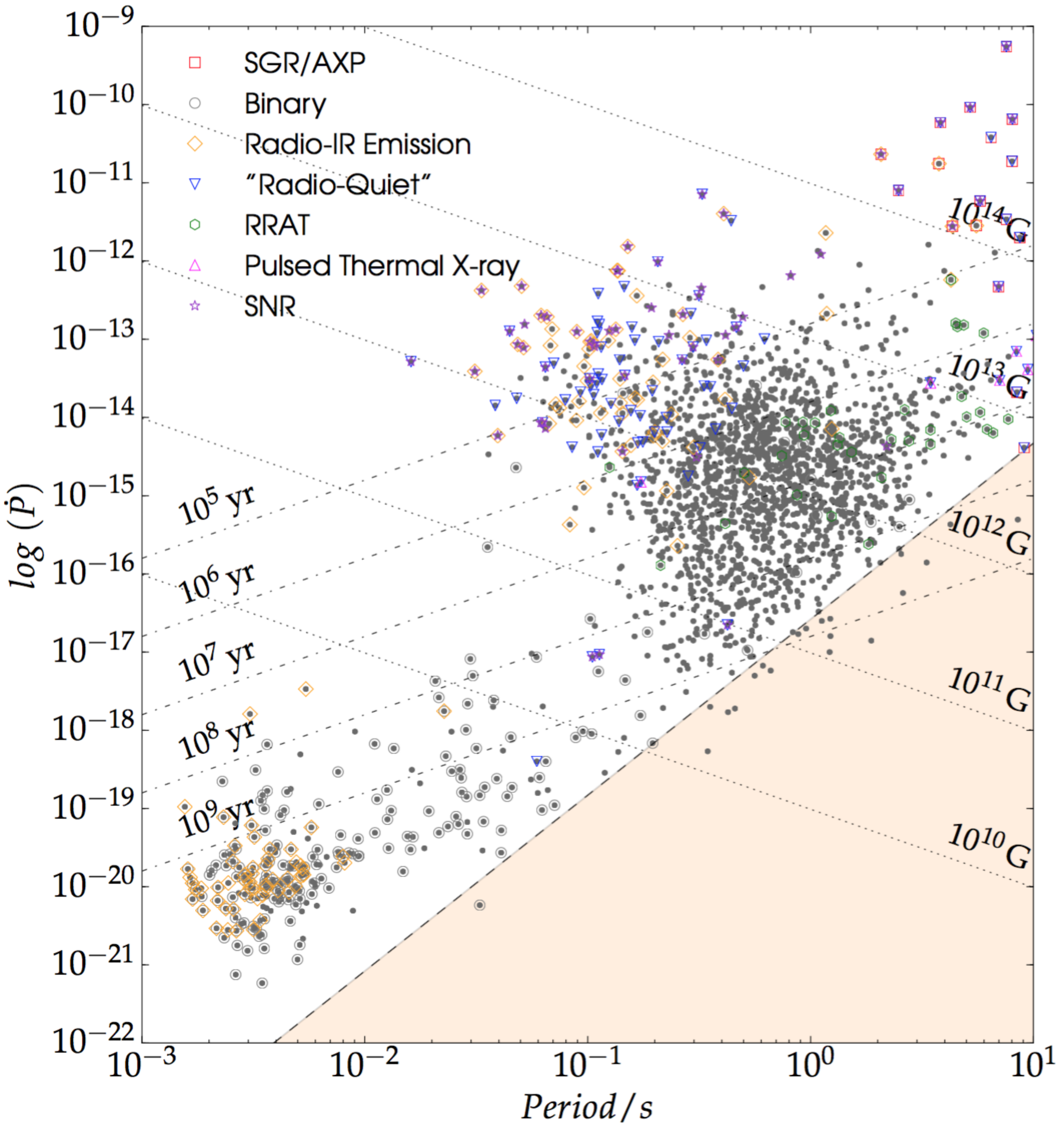


Figure 2.8 – A $P - \dot{P}$ diagram. The dot-dashed lines are lines of constant characteristic age and the dashed lines are lines of constant surface magnetic field strength, from $10^{10} - 10^{14}$ G. Circled points are pulsars in binary systems. The plot has been made using the psrqpy package, which is an interactive python tool combined with data from the ATNF pulsar catalogue.

Rotating RAdio Transients (RRATs; denoted by green circles in Figure 2.8) (McLaughlin et al., 2006) are transient pulse emitters and have slightly higher magnetic field strengths than normal pulsars. Various types of extreme pulsars also exist and their position in the $P - \dot{P}$ diagram helps in assessing how the evolution of neutron stars happens after the supernova explosion.

2.3.3 Fast Radio Bursts

Recently, pulsar surveys have resulted in the serendipitous discovery of fast radio bursts (FRBs). While FRBs seem identical to the individual pulses from pulsars, their large dispersive delays indicate that they originate from far outside the Milky Way and hence are many orders-of-magnitude more luminous (Petroff et al., 2019). FRBs are bright and powerful (50 mJy - 100 Jy) millisecond duration pulses of emission at radio frequencies between 100 MHz and 8 GHz. The origins of FRBs are still unknown, even though many more FRBs are discovered yearly, it is complicated to identify the galaxies in which they originate. The excitement about the discovery of FRBs has led to an increased in searches through new and archival data, not only at the Parkes telescope (Burke-Spolaor & Bannister, 2014; Ravi et al., 2015; Champion et al., 2016) but also at other telescopes, for example the discovery of FRBs at the Arecibo Observatory (Spitler et al., 2014), the Green Bank telescope (Masui et al., 2015), the Australian Square Kilometre Array Pathfinder (ASKAP, Bannister et al. 2017; Shannon et al. 2018), and the Canadian Hydrogen Intensity Mapping Experiment (CHIME, Boyle & Chime/Frb Collaboration 2018; CHIME/FRB Collaboration et al. 2019b). Since 2013, the rate of discovery of FRBs has increased and shows a doubling rate of the known population in the recent years (Shannon et al., 2018; CHIME/FRB Collaboration et al., 2019b). Discoveries from these telescopes have highlighted $\sim$ dozen repeating FRBs, two of which have been studied extensively, namely FRB 121102 (Spitler et al., 2016; Scholz et al., 2016; Chatterjee et al., 2017) and FRB 180814.J0422+73 (CHIME/FRB Collaboration et al., 2019a).

2.4 Astronomical surveys

In this thesis, several ML pipelines are developed to automate various steps in various surveys. Hence, in this section, we provide some details about the various surveys we used, ranging from optical to radio surveys. Several ground-based and space-based mission surveys have been undertaken during the last two decades. In this section, we elaborate on surveys that have been used in this work, for example, the Catalina Real-time Transient Survey, the MeerLICHT and the MeerKAT telescopes.

2.4.1 The Catalina Real-time Transient Survey (Optical)

The Catalina Sky Survey[‡] started back in 2004 and utilizes three telescopes to observe the sky between declination $\delta = -75$ and $+65$ degrees. Each of the telescopes is operated as a separate sub-survey and looks for optical transients. These comprise of the Catalina Schmidt Survey (CSS), the Mount Lemmon Survey (MLS) and the Siding Spring Survey (SSS). CSS and MLS are located in Tucson, Arizona and SSS is found in Siding Spring, Australia. Each telescope has allocated fields that cover the sky and avoids the Galactic plane region by 10 to 15 degrees because of reduced source recovery in crowded stellar regions (Drake et al., 2014b). All data observed by the Catalina Sky Survey is analysed for transient sources by the Catalina Real-time Transient Survey (CRTS [§], Drake et al. (2009)). In this work, we used data from the Siding Springs Survey. We utilized the periodic variable catalogue from the southern location within region $-20° < \delta < -75°$. More details about the data and methodology employed, are detailed in Chapter 4 and 5.

2.4.2 MeerLICHT Optical Telescope

MeerLICHT is an optical wide-field telescope that is operated robotically. The telescope is located at the Sutherland station of the South African Astronomical Observatory (SAAO). It consists of a 65 cm primary mirror and provides a 2.7 square degree field-of-view at a pixel scale of $0.56''$/pixel (Bloemen et al., 2016). MeerLICHT will co-observe with the MeerKAT radio telescope (Jonas & MeerKAT Team, 2016) on the same field. The combination of an optical and a radio telescope will enable the study of fast transient phenomena using simultaneous observations in two very distinct parts of the electromagnetic spectrum, whilst eliminating the delay introduced by triggering optical follow-up after the detection of a radio event. Both MeerLICHT and the BlackGEM array (Groot, 2019) (that is currently being installed at the La Silla Observatory in Chile) will yield about 500 observations per night, per telescope, thus generating tens of thousands of candidate alerts every clear night that could be spectroscopically followed up.

BlackGEM's main focus is on the detection of optical counterparts to gravitational wave events and MeerLICHT is designed to co-observe the sky as seen with the MeerKAT radio array. MeerLICHT and BlackGEM are technically identical with MeerLICHT being the prototype for the BlackGEM array. These arrays will enable the creation of a large database of transient

[‡]http://www.lpl.arizona.edu/css/
[§]http://crts.caltech.edu/

and variable sources. Such large databases are important for future analyses of data collected during upcoming photometric surveys like the Vera C. Rubin observatory (LSST; LSST Science Collaboration et al. 2009). In Chapter 6, we automate the transient detection pipeline for the MeerLICHT facility by developing a ML model to remove spurious sources detected in the MeerLICHT data.

2.4.3 MeerKAT Radio Telescope

The MeerKAT radio telescope is the (more) Karoo Array Telescope (Camilo et al., 2018) and is a precursor for the Square Kilometre Array (SKA) mid-frequency telescope, located in South Africa. It consists of 64 dishes with diameter 13.96 m each. MeerKAT has a field of view (FoV) of over a square degree at 1.4 GHz. This characteristic makes it an excellent instrument for searching for radio transients, for e.g. Fast Radio Bursts (FRBs). The nature of FRBs present various challenges, e.g compute power, real-time communication and in system design (Jankowski et al., 2020). Rapid data processing is fundamental to decrease the delay for follow-up observations and to retain high resolution data of the object of interest. In this work, a machine learning algorithm is constructed and implemented in a real-time triggering infrastructure for FRBs classification and localization at the MeerKAT telescope. More details can be found in Chapter 7.

2.5 Concluding Remarks

This chapter provides a background introduction to the transient universe, both in terms of radio and optical astronomy. The reader was introduced with various terminology and various types of transients and variable stars. The concepts behind various variable stars are introduced, thus allowing the reader to become familiar with these terms being used in other chapters. In this thesis, both optical and radio data are used independently. Hence, the various surveys we used are detailed.

3

Machine Learning Classification

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."

– Alan Turing

This chapter introduces the reader to Artificial Intelligence and various machine learning algorithms. Section 3.2 introduces important terminology and the steps to be undertaken when implementing machine learning techniques. An introduction to multiple machine learning algorithms is presented in Section 3.3. This will cover fundamental techniques whilst also explaining key concepts relevant throughout the remainder of the thesis.

3.1 Overview

Artificial Intelligence (A.I.) is a field of scientific study concerned with replicating human intelligence via artificial means. The field of A.I. is comprised of numerous sub-fields (computer vision, speech synthesis, machine learning, etc.) which together aim to create machines able to learn independently from experience. A.I. has had a long and rich history, arising from different philosophical points of view. Whilst there are far too many breakthroughs in the history of A.I. to elaborate in detail, there some key milestones worthy of mention. The first one, is the challenge of searching for patterns in data, an important one and has a successful history. For example, in the 16th century, the extensive astronomical observations carried out by Tycho Brahe, enabled Johannes Kepler to make the discovery of the empirical laws of planetary motion, which in turn provided a starting point for the development of classical mechanics. Similarly, in the early twenties, the discovery of regularities in atomic spectra played a fundamental role in the evolution and verification of quantum physics. Automated pattern recognition and learning was further explored by Alan Turing (Turing, 1937). The second occurred during the 1940's with the introduction of artificial neurons by McCulloch & Pitts (1943). They developed artificial nerve cells capable of learning and thereby recognizing patterns. It was not until 1956 that the field of A.I. was formally established at a conference in Hanover, New Hampshire (Benko & Lányi, 2009). The area of A.I. concerned with learning to recognize patterns is known as *Machine Learning* (ML). ML is concerned with the automatic detection of patterns in data through the use of computer algorithms and with the help of the discovery of these irregularities can take actions such as '*categorising*' the data into various categories/classes. Such systems capable of learning to recognize patterns can often also make predictions based on observed data. In other words they can use data collected about the past and today, to make predictions about tomorrow.

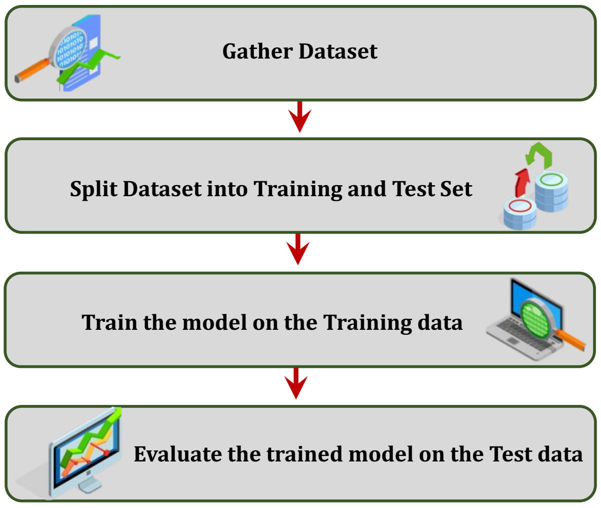

Gather Dataset
Split Dataset into Training and Test Set
Train the model on the Training data
Evaluate the trained model on the Test data

Figure 3.1 – The main stages required when implementing a machine learning algorithm. Some intermediate steps are skipped in this process, for instance, any pre-processing steps that need to be applied to the dataset and hyper-parameter tuning/optimisation are excluded too.

AI has seen a rise in popularity in recent years due to a number of factors. But perhaps most crucially, i) vast quantities of data are being generated by us and about us on a daily basis. Such data can be used to build automated ML systems capable of classification and prediction in a number of application domains, whilst, ii) advances in computing infrastructures (both hardware and software) have also improved. Nowadays, A.I. is being used in many areas, from fraud prediction, control system monitoring, stock trading, voice / facial recognition and within scientific fields. For example, computer vision is a sub-field of A.I. that is capable of mimicking the capabilities of human vision. It can be used to scan various categories of images and extract useful patterns. Similarly, A.I. has made it possible to undertake Natural Language Processing (NLP) tasks with some success and such tools are being more widely applied, from speech recognition to sentence completion.

The key research field within A.I. that is of central importance to this thesis is *Machine Learning* (ML). ML is a discipline that deals with the creation of computational machines which enable a computer program to speed up various processes, for instance, the recognition of patterns, and in doing so imitate the brain's ability to learn (Mitchell, 1997). These machines process information in the form of data as input, and output a decision which can be in the form of prediction, grouping (clustering) or classification. This chapter introduces some ML notation and terminology used throughout the thesis, and describes in detail the process of classification using ML algorithms.

3.2 Introduction to Machine Learning Terminology

Certain tasks are too challenging and too time-consuming for a human to tackle. Hence, ML is often deployed to solve various challenging problems, for example, a myriad of data classification tasks (Kotsiantis, 2007). It is necessary that certain concepts need to be understood and pre-determined steps need to be undertaken to train a machine learning algorithm. Figure 3.1 lists the stages that are followed during the building process.

The first stage in applying a machine learning algorithm is to acquire a dataset, D. A *dataset* refers to a collection of variables or records which contain some information about a particular problem. In this case, a data set D is described in the form of tabular data. Each row within D is known as an *instance* or *example*. Each *instance* is successively made up of a number of *features* otherwise known as *input variables* represented by X. The features represent the characteristics of the instances. These can be discrete or continuous. In addition, each example can also be assigned a continuous or discrete ground truth label, that describes it. The goal of ML is to use a subset of data from D, which we call experience E, to 'teach' an algorithm to solve a problem, using the past experience in E. This experience can then be used to solve a task T, where the ability of the algorithm to solve the problem is measured via some quantifiable performance measure P.

A machine learning model is therefore a mathematical model which can accurately map the input features to ground truth labels (sometimes called target labels, or just targets) (Cios et al., 2007). Any systems capable of mapping features describing instances to ground truth labels is known as a classification model[*], or a classification algorithm. Figure 3.2 shows an example of a typical dataset which consists of a number of instances. Each instance is represented by a

[*]Here a model refers to the mathematical representation of a learning system, that is implemented within an algorithm and able to 'learn' and make decisions over data.

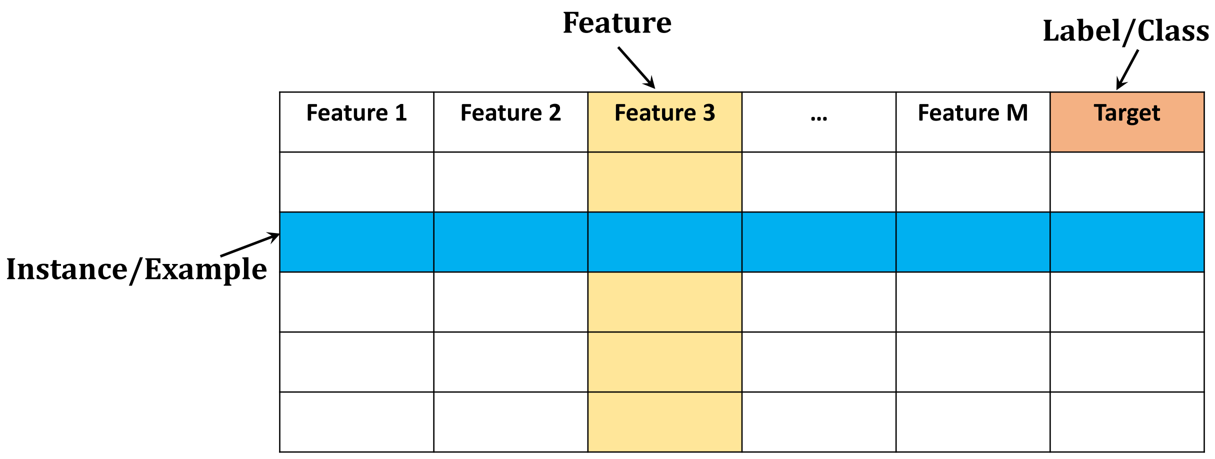


Figure 3.2 – Highlighting the primary step related to the implementation of a machine learning algorithm - the dataset. A dataset can be in a tabular form. Each row in the dataset is an instance/example, for which the examples are made up of a number of features and targets/labels. Each feature is represented as a column, and the right most column is the target.

number of features and a target class 'label' respectively.

A task, T, can be categorised as *supervised* or *unsupervised*. In supervised machine learning, a problem is defined in terms of input variables, X and associated outputs y which the algorithm uses to learn the inherent connection between features and class variables. Once it has learned, the algorithm can be applied to data comprised of features without labels (i.e., the ground truth is unknown), in order to predict what the class label should be. In many cases, the ground truth output labels y may be challenging to collect automatically and must be provided by a human. Thus, in the *supervised* approach, each instance is also characterised by a *ground truth* or *target*, y. Generally, a dataset comprises of a number of instances, characterised by a number of input features and corresponding targets. In the classification scenario (see §3.2.1), the target is also referred to as the *class* or *label*.

In *unsupervised* learning, input variables X are available but there are no labels y available for a variety of reasons - labelling cost, scarcity amongst others. In an informal way, unsupervised learning aims to use the inherent structure of the data, to guide prediction / categorization tasks without the need for human annotators to acquire ground truth labels normally required for supervised learning.

3.2.1 Classification Tasks

This thesis is mostly concerned with classification tasks. In such problems, the main objective is to build a model that can accurately predict the class or label for as many instances as possible. A perfect classifier is a model that can accurately predict the correct ground truth label on data for which there exists no pre-existing labels. In classification scenarios, the targets are usually discrete finite categories. Classification tasks dealing with two classes are often referred to as *binary* and if the task is concerned with more than two classes, then it is referred to as *multi-class*.

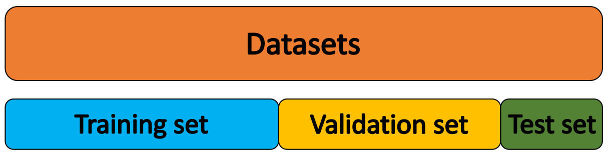


Figure 3.3 – Holdout splitting is accomplished by splitting the full dataset into training, validation and testing sets. Each set is disjoint. Models are trained on the training data, then evaluated on the validation data. The model hyper-parameters are optimised based on the validation set. The model which achieves the best performance on the validation data is then selected and employed to the test data.

3.2.2 Data Splitting

In general, machine learning algorithms use training data to build a mathematical model, where the training data is comprised of feature data. These features may have varying utility, that is, information rich features are desirable as they can be used to build more accurate classification systems. Let $S = \{X_1, \ldots, X_n\}$, represent a set of data available to us, then X_i is an individual object represented by variables known as $features$. An arbitrary feature of an object X_i is denoted by X_i^j, where $j = 1, \ldots, l$. Each object has an associated label y such that $y \in Y = \{y_1, \ldots, y_k\}$. For multi-class scenarios the possible labels assignable to these objects vary as $1 \leq y \leq \mathrm{N}_{\mathrm{class}}$. Meanwhile for binary class classification scenarios, we consider binary labels $y \in Y = [0, 1]$.

The goal is to build a machine learning algorithm that learns to classify objects described by features, from a labelled input vector, also known as the training set, X_{Train}. The training set consists of pairs such that $X_{Train} = \{(X_1, y_1), \ldots, (X_n, y_n)\}$. The learnt mapping function between input feature vectors and labels in X_{Train}, can then be utilised to label new unseen objects, in X_{Test}.

Data splitting is a fundamental stage when training and evaluating a machine learning algorithm. Essentially, the dataset needs to be split so that the model can be evaluated and tested to determine its performance. Commonly, the dataset is split into a *training set* and *test set*, where training data is a fully labelled subset of some dataset S for a supervised problem. It should contain instances representing all the different patterns we want an algorithm to recognize. Using this information a model is developed and this is known as the *training* phase. Furthermore, the test set which is also a subset of S that is completely distinct from X_{Train}, is used to evaluate the trained model. The *testing* phase is also referred to as an unbiased evaluation (Kotsiantis, 2007) as the model is being evaluated on data it has not seen before.

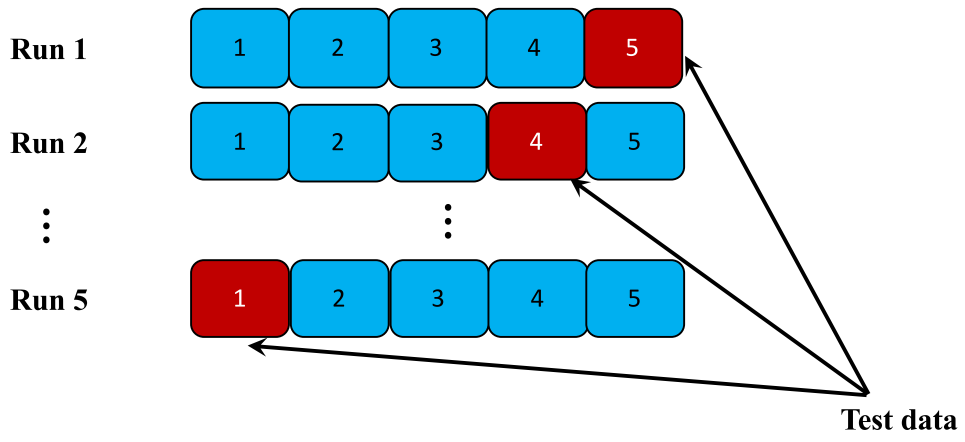


Figure 3.4 – Illustrating 5-fold cross-validation. The dataset is split into 5 disjoint partitions and the algorithm is trained on 4 partitions and tested on the last one. The training data is coloured in blue and the testing data in red. In the first run partition 5 is used as the test set. Then, the algorithm is executed again except this time the test set is another partition, in run 2 partition 4 is used as the test set. This is repeated 5 times until eventually the first partition is used for testing and the remaining ones are used for training.

Data splitting is necessary as performing training and evaluation on the same training data will not give a true impression of the generalization capabilities of the trained model on new data it has never before encountered. If data is shared between the training and testing data, we often obtain an unrealistic impression of model performance. It is therefore important that the training and test data are disjoint (Witten et al., 1999). The training set will be used to build the model and the test set will be used to evaluate the performance of the model. Two common approaches for splitting a dataset are *K-fold cross-validation* and *holdout* (Yadav & Shukla, 2016).

3.2.2.1 *K*-Fold cross validation and holdout splitting

When using machine learning algorithms, a major problem is an overoptimistic result, i.e the output results are too good to be true on training data, due to over/under fitting. Misleading performance mostly happens when we perform training and evaluation on the same data. Therefore, classification algorithms must be tested on data independent from the test set to avoid this problem. One method for avoiding this issue, involves splitting the data into training and testing sets. There are many potential split ratios, however 2/3 training and 1/3 test is common (Borovicka et al., 2012). A modified version of this approach is called the *holdout* test. In this case, the full dataset is split into three disjoint partitions, namely, the *training*, *validation* and *test* set. In the former approach involving 2 splits, the hyper-parameters[†] of the model are being optimised and selected based on the test set. While for the latter case, that is the 3 splits, the training data is first used to train the model. Once the training phase is completed, the

[†]Hyperparameter is a model configuration that is external to the model and whose value cannot be estimated from data. They are usually specified by the practitioner and tuned for a given predictive modelling problem.

model is evaluated and potentially optimised on the validation data. In this context, various models can be evaluated according to the results of parameter optimization. The model reaching the best performance measure on the validation data can be fully assessed on the test data. In the *holdout* approach, the hyper-parameters are optimised and fine-tuned on the validation data and not on the test data. An example of *holdout* splitting is illustrated in Figure 3.3.

Another common method for splitting datasets for evaluation is known as K-fold cross-validation, that is, the training dataset S is split randomly into K mutually exclusive subsets $(S_1, S_2, \ldots, S_K)$ of nearly the same size. The classification algorithm is trained and tested K times. For each time step $t \in \{1, 2, \ldots, K\}$, the algorithm is trained on $K-1$ folds and one fold S_t is used for validation. In addition, a stratification of the data can be applied such that for each of the $K - folds$, the data are arranged to ensure each fold preserves the proportion of samples for each class in the dataset at large. The overall balanced-accuracy (see §3.2.3) of an algorithm trained/tested via cross-validation is simply the average of each of the K balanced-accuracy measures obtained after each time step. A 5-fold cross-validation is illustrated in Figure 3.4.

3.2.3 Model Evaluation

The previous subsection introduced model evaluation on validation or testing data. In this section, we present the evaluation metrics terminology used in this thesis. The motivation of the metric is to evaluate the performance of the model, which eventually answers the question: how accurate is the model on a particular dataset?

In this thesis, the performance of machine learning algorithms used for various classification problems, is evaluated using measures such as the balanced-accuracy, precision, recall, F1 score, sensitivity and specificity (Chao et al., 2004). They are defined in terms of True Positive (T_P), False Positive (F_P), True Negative (T_N) and False Negative (F_N).

- **Sensitivity Measure**: Equation 3.2.1 also known as the true positive rate or "recall". It measures the proportion of actual positives correctly identified by the model.

$$\boxed{\text{Sensitivity/Recall} = \frac{T_P}{T_P + F_N}} \tag{3.2.1}$$

- **Specificity Measure**: Equation 3.2.2 also known as True Negative rate. It measures how well a model identifies negative results.

$$\boxed{\text{Specificity} \;=\; \frac{T_N}{T_N + F_P}} \tag{3.2.2}$$

- **Precision**: It is a measure of retrieved instances that are correctly labelled. Precision is described in Equation 3.2.3.

$$\boxed{\text{Precision} \;=\; \frac{T_P}{T_P + F_P}} \tag{3.2.3}$$

- **F1-score**: It is a metrics that aims to quantify overall performance, expressed in terms of precision and recall as shown in Equation 3.2.4.

$$\boxed{\text{F1-Score} \;=\; 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}} \tag{3.2.4}$$

- **Accuracy**: It is simply the ratio of correctly classified instances/examples to the total number of instances/examples used in the test set and is given as:

$$\boxed{\text{Accuracy} = \frac{T_P + T_N}{T_P + F_P + T_N + F_N}} \tag{3.2.5}$$

- **Balanced accuracy measure**: It is defined as the average recall rate obtained on each class and it is a metric that allows a representative impression of classifier performance in the presence of heavily imbalanced class distributions,

$$\boxed{\text{Balanced Accuracy (\%)} \;=\; \frac{\text{Sensitivity} + \text{Specificity}}{2} \times 100.} \tag{3.2.6}$$

- **Geometric Mean Score**: G-Mean is defined as the square root of the product of class-wise sensitivity. It maximizes the accuracy on each class in addition to keeping these accuracies balanced.

$$\boxed{\text{G-Mean} \;=\; \sqrt{\text{Sensitivity} \times \text{Specificity}}.} \tag{3.2.7}$$

- **Precision-Recall curve**: PR curve illustrates the trade-off between TPR and positive predictive value for a model at different probability thresholds.

- **Receiver Operating Characteristic Curves**, also known as **ROC** curves: It is a visualization of the true positive rate (recall) against false positive rate, defined as:

$$\boxed{\mathrm{FPR} = \frac{T_N}{T_N + F_P}} \tag{3.2.8}$$

  at various thresholds $\in$ [0,1]. The area under the ROC curve, also called as (AUC) is a metric used to evaluate the ability of the model to distinguish between two classes. AUC is independent of the threshold used to compute ROC.

- **Matthew's Correlation Coefficient**: It is mostly used in situations where we must evaluate classifier performance on class imbalanced data and it is the combination of the above metrics. MCC can be defined as:

$$\boxed{\mathrm{MCC} = \frac{T_P \times T_N - F_P \times F_N}{\sqrt{((T_P + F_P)(T_P + F_N)(T_N + F_N))}},} \tag{3.2.9}$$

  where an MCC of $+1.0$ indicates a perfect prediction, -1.0 a total disagreement between the model predictions and the true classes, and 0.0 is a "random guess".

- **Confusion Matrices**: The T_P, F_P, T_N, and F_N can be visualized via a confusion matrix as illustrated in Table 3.2.1, where the predicted class is indicated in each column and the actual class in each row. In this case, from Table 3.2.1, the true positives (T_P) are Class I examples that were correctly classified as Class I, false positives F_P correspond to Class II examples wrongly classified as Class I. In a similar way, false negatives F_N and true negatives T_N can be explained.

Table 3.2.1 – Confusion matrix implemented for our specific problem, where the positive class corresponds to Class I and the negative class to Class II for the two classification schemes. True/false positives/negatives are represented as T_P, F_P, T_N, and F_N respectively.

		Class II	Class I
Actual Class	Class II	T_N	F_P
	Class I	F_N	T_P
		Predicted Class	

3.3 Machine Learning Classification Algorithms

This thesis is concerned with the classification of astronomical data. This section provides a brief introduction to several ML classification algorithms used in this work.

Instances are characterized by the features extracted from the data. A rigorous way is needed to turn this information about each instance in to a quantifiable prediction. Our goal is to perform a supervised classification: given a sample of instances along with their respective class labels, a model is developed such that it learns the characteristics of each instance from the feature space and outputs the likelihood / prediction of an instance belonging to each class. The trained model can then be used to automatically make predictions regarding the true class variables represented by X of each new instance given.

The subsections below introduce several machine learning algorithms, for instance, decision trees (DTs), random forests (RFs), and XGBoost. These algorithms take as input features that describe the data and a class label. At the end of the process, a class prediction is output for each example in a test set.

3.3.1 Decision-Trees

A decision tree (Quinlan, 1986) is a graph-like structure comprised of nodes that represent features and edges that represent the possible values that features can take as illustrated in Figure 3.5. Decision trees map input features to output classes based on a series of selection rules (Breiman et al., 1984). They achieve this via a recursive binary partitioning that splits the feature space X into disjoint nodes in the tree. In our case, one node can correspond to one of the features listed in Figure 3.5, for example, one feature can be 'Gender', with two edges directing away from the node with two possible values, for instance, 'male' and '$female$'. For a given instance passed to the root of the tree for classification, it follows a path along one edge depending on the value of the feature at that node. The algorithm at each step chooses both the feature and split-point that provides the highest Information Gain (IG, Berrar & Dubitzky 2013) or produces the smallest impurity in the two resultant nodes in terms of classification. It is possible to measure the information content of a feature, using IG alone, or as part of a different metrics that incorporates it, such as GINI impurity. In our case, we use the GINI impurity for a set of instances as it is more efficient in terms of computational power and statistical power (Nembrini et al., 2018). To illustrate how tree-learning works, consider the following problem. Given data describing the survival rate of penguins in the wild as illustrated in Figure 3.5.

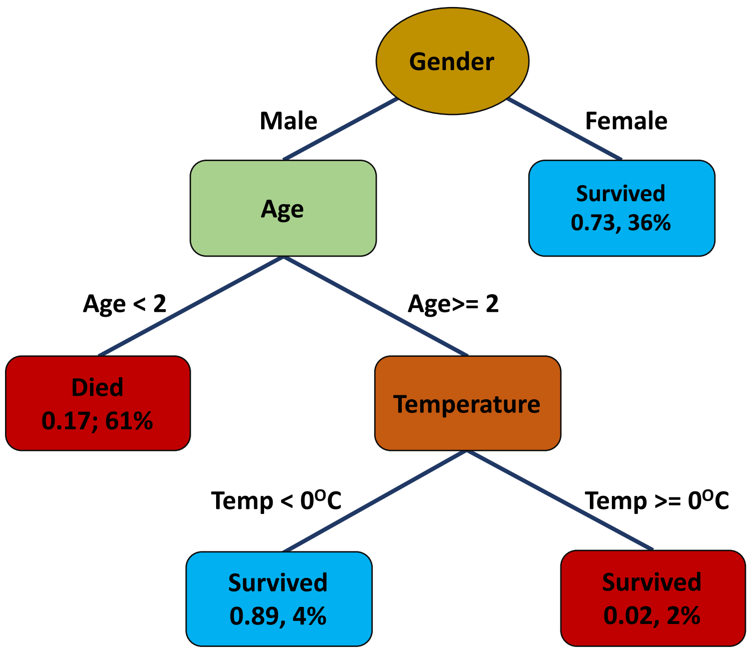


Figure 3.5 – A decision tree showing survival of penguins. The figures under the leaves show the probability of survival and the percentage of observations in the leaf. Summarizing: the chances of penguins survival were good if they were (i) a female or (ii) a male older than 2 years, living in an area where the temperature is below 0°C.

The penguins will move down the edges passing through many other nodes and will be filtered based on the values of other potential features. At the end of the path, the penguins end up at a particular leaf. The majority label at leaf determines the predicted class, for example, a leaf may correspond to the percentage of penguins 'survival' or 'death'. Therefore, decision trees present an automated way of performing classification of unknown objects by learning which path to take based on their features. However, building the best decision tree is a challenging task, for example, in our case, constructing an automated classification decision tree to distinguish between different types of variable stars (see Chapter 4). Generally, using a single decision tree for classification often leads to poor performance due to low or high variance, for instance, a small change in the training set can lead to a very different estimated tree structure. Instead of using a single decision tree, advanced techniques, for example, the ensemble methods that combined multiple individual trees have been developed that depend on the averaging of errors arising from between groups of trees.

3.3.2 Random Forest

A Random Forest (RF, Breiman, 2001) is simply a system that combines multiple of decision trees (shown in Figure 3.6) and evidence shows that ensemble methods tend to outperform single-model learners in real world scenarios. The RF approach is based on training several decision trees using sub-samples from the training set, and subsequently utilizing these decision trees to perform a classification of unknown objects.

Let us consider n_s samples in the training set having m features each. We then define the number of trees as T in the random forest and n_f is the number of features utilized in each tree where $n_f < m$. RF is then trained following two steps. First, T datasets with n_s samples are generated. The data set is constructed in such a way that the objects/stars are randomly sampled with replacement from the training set. Therefore, each set T consists of the same number of objects as the training set, but some objects are picked up more often (Carrasco et al., 2015). Finally, the individual decision tree is built from each T data sets. A set of features are randomly selected from the initial m features in Figure 3.2 whereby the best split at each node is based on those selected features.

The RF algorithm often performs well in practice because of two principles. The first is that each individual decision tree, created on different samples, acts as an independent and individual classifier. The second principle is based on the fact that only a subset of the features selected randomly, are utilized to construct an individual tree. These two principles allow the classifier to decipher patterns in each subset of features as they capture more of the variability in the data.

At the final stage, based on the feature value, every tree from the forest allocates a specific label to an object. The final prediction of the classifier for a given object is the one chosen by the majority of the T trees (Carrasco et al., 2015) though more complex ensemble voting systems are possible (e.g., votes weighted by individual classifier accuracy).

3.3.3 XGBoost

XGBoost (eXtreme Gradient Boosting) is a boosting algorithm and is a tree-based model which became popular since its inception in the ML community in 2016. XGBoost is an ensemble learning algorithm. Ensemble learning provides a systematic solution to combine the predictive power of multiple trainable models. The result is a single model which gives the aggregated output from several models. Bagging and boosting are examples of ensemble ML concepts. Bagging decreases the variance (variability of model prediction for a given data) in the prediction by generating additional data for training. New training datasets are produced by random sampling with replacement from the original data set. Boosting algorithm is an iterative process which adjusts the weight of the observations based on the previous classification, i.e., after each training step, the weights are redistributed. Misclassified examples increases its weights to emphasize the most difficult data to classify. In this way, subsequent learner will emphasize on them during the training process and decreases bias (the difference between the model prediction and the ground truth value/label).

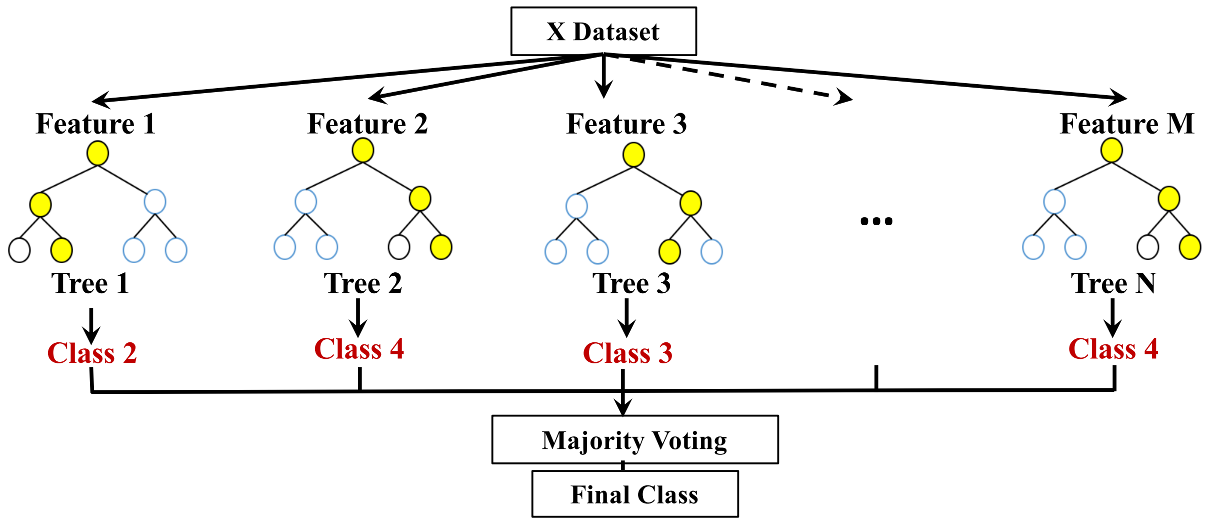


Figure 3.6 – Illustration of a random decision forest. A Random forest (RF) consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the RF outputs a class prediction and the class with the most votes becomes our final prediction.

XGBoost works in the same way as the Gradient Boosting Decision Tree (GBDT, Friedman, 2001) method. GBDT is an ensemble classification system that iteratively adds simple decision tree classifiers. The first classifier of the ensemble system is trained on the data, while the successive classifiers are trained on the errors of the predecessor classifiers. The base learners in boosting are weak learners which are highly biased, and the predictive power is no better than random guessing. Each of these weak learners contributes some vital information for prediction, enabling the boosting technique to produce a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance.

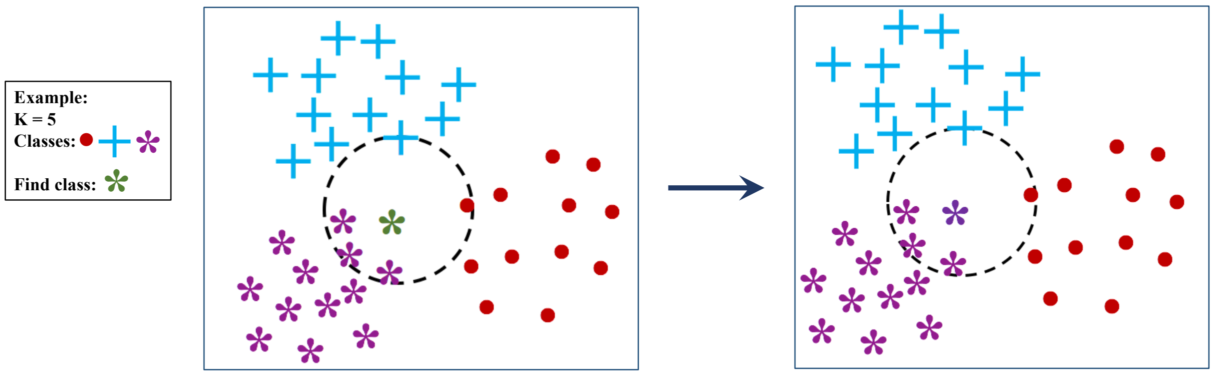


Figure 3.7 – Given N training points and there are three classes of objects. *k*NN will identify the class label of a test point regardless of its label. For $k = 5$, the algorithm will find the 5 Nearest neighbours to the test point based on the Euclidean distance. We note that for this test point, there are three training points in purple and one blue and red category closest to the test point. Therefore, *k*NN takes the majority class vote and assigned the test point to the purple class.

In addition, this classifier controls overfitting by using the regularization techniques, L1-

norm (Tibshirani, 1996) and L2-norm (Ng, 2004). In contrast to bagging techniques like Random Forest (an addition of decision trees that aggregate tree decisions), in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits. Such small trees, which are not very deep, are highly interpretable. Unlike, in GBDT, the beauty of XGBoost lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.

3.3.4 *k* Nearest Neighbours

k Nearest Neighbours (*k*NN, Buturovic, 1993) is a simple instance-based technique utilized for classification. *k*NN often performs very well and acts as a benchmark technique despite its simplicity (Ali et al., 2019). *k*NN classifies an unlabelled example/instance by applying the average class determined from majority class vote of the labels among the *k* nearest neighbours (Hastie et al., 2009). The *k* nearest neighbours of a test object are determined by computing the Euclidean distance measure. *k*NN outputs a discrete value that an object belongs to a particular class y. This output is computed by a majority vote of its neighbors, with the object belonging to the class most common among its *k* nearest neighbors. An illustration of the *k*NN algorithm is illustrated in Figure 3.7 and in Theorem 3.3.1.

Theorem 3.3.1: *k*NN Algorithm

Inputs:

Training set $X = \{x_1, \ldots, x_N\}$ with labels $Y = \{y_1, \ldots, y_N\}$,

An integer k where $0 < k \leq N$,

Test example x_0.

Output:

Predicted y_0 of x_0

For each point, $P' = (x', y')$

- Compute the distance $D(x', x_i)$, for e.g euclidean distance = $\sqrt{\sum_{i=1}^{k}(x' - x_i)^2}$ and append in S,
- Sort the $\mid S \mid$ distances by increasing order,
- Count the number of occurrences of each class y_i among the k nearest neighbours,
- Assign to x_0 the most frequent class.

*k*NN is robust to noisy training data (training data with corrupted or distorted information) and it performs effectively when the data set is sufficiently large. However, one disadvantage of *k*NN is that all the features are needed when computing the distance between data points. If a small portion of the data set consists of discriminatory information and the larger portion contains irrelevant features, the distance between the instances will be more influenced by the irrelevant samples (and their feature values) and this problem is known as the *curse of dimensionality* (Bellman, 1957), that is, in high dimensional space, the distances become meaningless. *k*NN is sensitive to this problem which can be overcome by weighting each feature differently when computing the distances between instances. Another problem with *k*NN is efficient memory indexing. Various techniques such as the *kd*−tree (Bentley, 1975a; Friedman et al., 1977) have been developed for more efficient memory indexing of the training data sets.

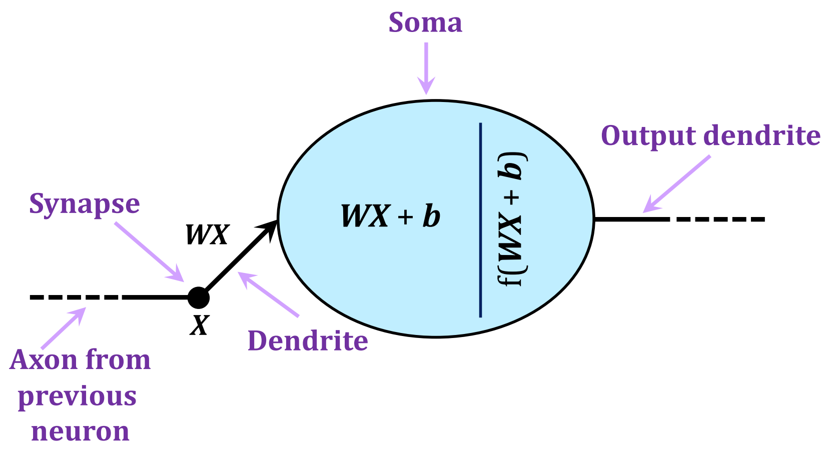


Figure 3.8 – The analogy of an artificial neuron in a neural network that mimics a neuron in the brain. The synapses in a neuron are represented as the, $\mathbf{X}$, the dendrite is the multiplication of the weights and the input, $\mathbf{W} \times \mathbf{X}$, the soma cellular body is similar to the application of the activation, $f(\mathbf{WX} + \mathbf{b})$, and the output dendrite represents the output of the function.

3.4 Introduction to Deep Learning

Machine learning algorithms for instance *k*NN, DT, RF, and XGBoost require features that represent specific and unique physical properties of the objects/sources they represent. The quality of the features utilized is a fundamental determinant in the success of these algorithms. This technique is known as shallow learning (Chen & Guestrin, 2016). Feature extraction is an important and difficult process that is crucial for this form of learning.

Deep learning is an area of machine learning in which the algorithms learn the features directly from the raw data, e.g, images. This becomes important in cases where feature extraction does not completely capture the physical characteristics of the raw data and also, helps reduce labelling cost. In recent years, with computationally powerful computing resources becoming readily available resources, deep learning has seen significant development and rapid deployment to numerous applications. Deep learning has received an enormous amount of attention in the computer vision community in particular.

LeCun et al. (2011) demonstrated that Deep Neural Networks (DNNs) are capable of outperforming shallow learning approaches. DNNs have been implemented in various domains such as object recognition (Szegedy et al., 2013), speech recognition (Ossama et al., 2014), image captioning (Vinyals et al., 2015) and voice recognition (Oord et al., 2016), to name a few. With these sophisticated advancements, DL is a fundamental field that can be used for the classification challenges that astronomy is facing with the deluge of data. In this thesis, the computer vision domain is explored to guide the classification of both radio and optical astronomical sources. More details can be found in Chapters 6, 7 and 8. In the next sections, we introduce some deep learning terminology before describing the various layers used to construct DL models implemented in future chapters.

3.4.1 Artificial Neural Network Learning

Artificial Neural networks (ANNs) (Bishop, 2006) dated back to the early 1940's, with the development of the artificial neuron by McCulloch & Pitts (1943). Their artificial neuron is inspired by the biological neuron based on an early understanding of the human brain.

A biological neuron works by propagating an electrical impulse received from other nearby neurons, only if the total sum of electrical input to that neuron is above some threshold level. If that level is met, the neuron fires an electrical impulse to those neurons it connects to via the synapse. In the artificial model, a neuron is represented by a mathematical function. This artificial neuron (function) accepts inputs from one or more nearby neurons, and will produce an output if the weighted sum of its inputs is above some threshold level.

An artificial neuron model is analogous to a biological neuron, that is comprised of inputs to the neuron, an activation function which can output a signal, similarly to a dendrite connected to the cellular body which outputs through the axon (Felten et al., 2015). Figure 3.8 presents a model of an artificial neuron.

The purpose of an ANN is to approximate some function f^* that maps some input $\mathbf{X}$ to a single output. For instance, for a classification scenario $\mathbf{y} = f^*(\mathbf{X})$ maps an input $\mathbf{X} \in \mathbb{R}^n$ to an output $\mathbf{y} \in \mathbb{R}^m$, which can also be written as

$$\boxed{\mathbf{y} = \alpha\left(\mathbf{W}\mathbf{X} + \mathbf{b}\right),} \tag{3.4.1}$$

where the matrix parameters $\mathbf{W} \in \mathbb{R}^{m \times n}$ are known as the *weights*, a vector of small positive values $\mathbf{b} \in \mathbb{R}^{m}$ are called the *biases* (can take values such as 0 or 0.1), and α is a non-linear activation function (see §3.4.1.1). Therefore, an ANN is simply a mapping of $y = f(\mathbf{X}; \mathbf{W}, \mathbf{b})$ and the algorithm learns the value of the parameters $\mathbf{W}$ and $\mathbf{b}$ that construct the best function approximation. Assuming α is a step function, then ANNs can be considered as binary classifiers and thus output either 0 or 1 as follows:

$$f(\mathbf{X}) = \begin{cases} 1, & \text{if } w_i x_i + b > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3.4.2}$$

where w_i denotes the i^{th} weight, and the input vector (x_i denotes the i^{th} vector component). Figure 3.8 illustrates an example of a single layer network along with the inputs, weights and bias. ANNs consist of many different types of layers, which are detailed below.

3.4.1.1 Activation function: Rectified Linear Units (ReLU)

The function $f(\mathbf{X}; \mathbf{W}, \mathbf{b})$ from the previous subsection was a step function which only outputs either 0 or 1. This thresholding/step function is often used for linear classifiers. Generally, the function which is applied to the inputs and weights is known as an activation function, α. Most activation functions can be viewed as taking an element-wise non-linear function α and apply it on top of an affine transformation $\mathbf{y} = (\mathbf{W}\mathbf{X} + \mathbf{b})$. One can use alternative non-linear functions, instead of limiting to only step functions and these include *tanh*, *ReLU*, *hyperbolic tangent*, *sigmoid*, and *softmax*, to name a few (Goyal et al., 2019). In this thesis, the activation function that is mostly used after each layer is the ReLU and at the end of most models, the softmax function is used. The ReLU function can be defined as,

$$\boxed{\text{ReLU}(x) = \max\{0, x\}.} \tag{3.4.3}$$

3.4.1.2 Softmax function

In a classification scenario, the aim is to represent the output of our model by a probability distribution over a discrete variable. For the case of a classifier, the probability distribution is represented over **n** different classes and for binary classification problem we want to output a single number:

$$\hat{y} = \mathrm{P}\left(y = 1|x\right), \tag{3.4.4}$$

as this number has to lie between 0 and 1. The softmax function is used at the end of the network to normalize the output of a network to a probability distribution over predicted output classes and it is given by:

$$\boxed{\operatorname{softmax}\left(x_d\right) = \frac{\exp\left(x_d\right)}{\sum_{j=1}^{D} \exp\left(x_j\right)},} \tag{3.4.5}$$

where $d \in \{1, \dots, D\}$ and D is the number of neurons in the last *dense* layer of the network. Formally, the softmax function is useful for multi-class-classification. For the case of binary classification, D will be equal to 2 and the softmax function will assign a probability to each neuron.

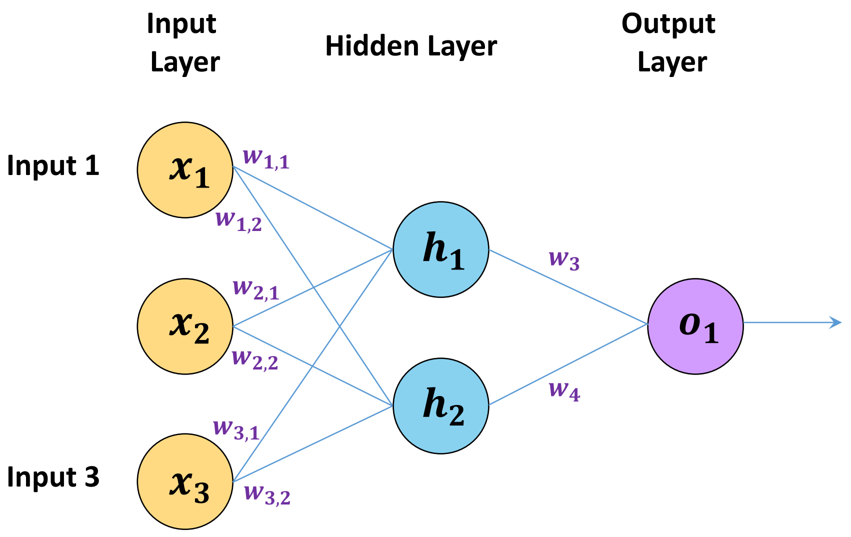


Figure 3.9 – Illustrating a multi-layer perceptron with 2 layers. There are a single input layer and output layer, and a hidden layer in between. Each layer is comprised of three input units, two hidden units and one output unit. In each layer, the units are stacked vertically. In the MLP, there are often many hidden layers where each unit is connected to every other unit in the previous layer.

3.4.2 Multi-layer network

In the previous section, a single layer network was discussed. However, for more complicated challenges, functions need to be more complex thus the network needs to consist of more than one layer. A network comprising of more than one layer is referred to as multilayer perceptron (MLP). A complication appears here, as inside the MLP many functions are connected in a chain. For example, consider the functions f_1 and f_2 are linked as follows: $f(x) = f_2(f_1(x))$. In this case, the function $f(x)$ is more complicated than the independent functions f_1 and f_2.

Goodfellow et al. (2016) refers to this chain as being widely used in the design of neural networks. In this particular example, the function f_1 represents layer 1 and f_2 refers to layer 2. Following similar reasoning, any layers can be added in the network, and hence this is referred to as *deep neural networks* (DNNs). The length of DNNs can be very large and the number of layers which are added together is known as the *depth* of the neural network. Figure 3.9 presents a two layer neural network. The first group of neurons/units is known as the *input* layer, and the last group of neurons/units is referred to as the *output* layer. All the layers in between the input and the output layers are known as the *hidden* layers. Figure 3.9 illustrates a two layer neural network with only one hidden layer.

Assuming that a *ReLU* activation function is used on each layer for the network presented in Figure 3.9, this implies there are three calculations which require computation. Primarily, h_1 is computed by applying a *ReLU* activation function on $(w_{1,1} \times x_1) + (w_{2,1} \times x_2) + (w_{3,1} \times x_3)$. Then h_2 is obtained by applying a *ReLU* activation function on $(w_{1,2} \times x_1) + (w_{2,2} \times x_2) + (w_{3,2} \times x_3)$. Lastly, the output of the network, O_1, is obtained by applying a *ReLU* function on the calculation of $(h_1 \times w_3) + (h_2 \times w_4)$. In this process, we note that the previous layer acts as input to the next layer and is referred to as a forward pass, which is also known as a *feedforward network*.

3.4.3 Deep neural network layers

A Deep neural network is composed of various layers, namely Convolutional layers plus pooling, dense/fully connected layers and activation layers among others. These layers can be combined in various ways to form different neural network architectures.

3.4.3.1 Convolutional layers

Convolutional layers (*CLs*) (Lecun, 1989) are used in DNN neural network (DNN), used to process 1-D grid data for example, time-series, 2-D grid or 3-D grid of image pixels. Such layers perform a kind of linear operation and apply a mathematical operation on the input, known as ‘convolution‘ which can be written as,

$$\boxed{f = \alpha\left(\mathbf{X} * \mathbf{W} + \mathbf{b}\right),} \tag{3.4.6}$$

where $\mathbf{W}$ is the kernel or filter, $\mathbf{b}$ is the vector of biases for each filter, α is the activation function and the output f is also known as the feature map. The aim of the *CL* is to extract features from the data. As can be noted in Equation 3.4.6, the convolution operation is employed on two functions **X** and **W** and at the end, the resulting function f is obtained.

In the context of deep learning, a convolutional neural network comprises of CLs that take as input the data **X**. The input **X** is typically an image and **W** is also known as a filter or kernel. A filter is described as having a depth, width and a height. The depth of a filter is similar to as the depth of the input image. The depth of an image is described by the number of channels, for instance, some images are made of red, green and blue channels. Thus for an RGB color image, the depth is 3. In astronomy, the channels are different parameters that describe astrophysical objects. For instance, for radio images of a pulsar, the channels are frequency-time and dispersion measure - time, thus the depth is 2. During the convolutional process, the filter convolves or "slides" over the input image in a horizontal and vertical pattern.

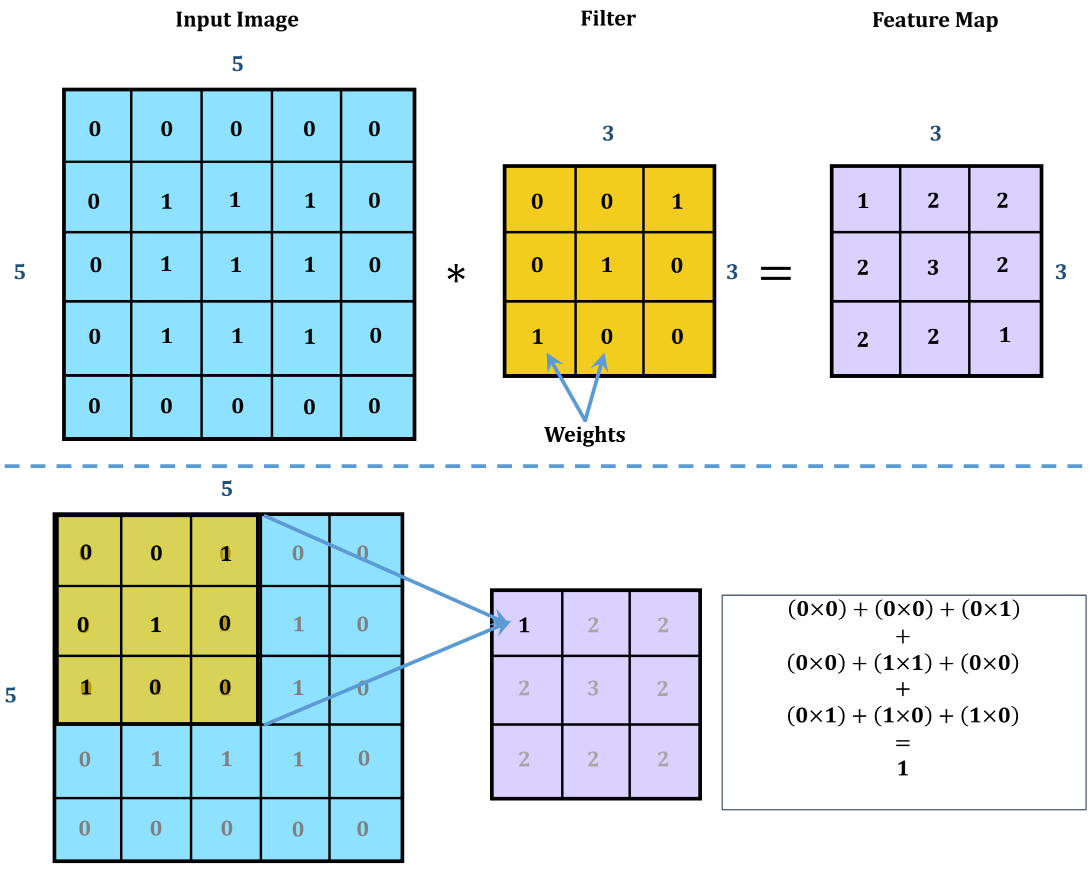


Figure 3.10 – Illustration of convolution process in the convolutional layer. On top, an input image, a filter and the resulting feature map after the convolution operation are presented. The bottom figure shows how the computation of the values are obtained in the feature. The filter slides over the image and the sum of the element-wise multiplication between the filter and the corresponding region of the image are computed. Thus for the top left region, the computed value is 1. After this step, the filter will shift by one pixel on the input image, and the computation is repeated. These sliding and computation steps are replicated until the resulting feature map is obtained.

Given an image I and a filter W_k, each pixel (i, j) is calculated using Equation 3.4.6 thus the resulting function f is known as the feature map. The convolution operation is computed each time the filter slides over the image. Figure 3.10 presents an example of the application of the convolutional operation to an input image. In this scenario, an input image with dimension 5×5 and a filter size of 3×3 are used. The figure shows how the value of 1 is computed at position (1,1) in the feature map. A 3×3 filter is positioned on the image and the dot product between both matrices are computed. Then, the 3×3 filter slides one by one pixel to the right of the input image and again the dot product is calculated to obtain a value of 2. There will be a time where the filter will no longer be able to move to the right, thus the filter moves down by one stride and the process of dot products continues. This process is repeated until all values in the feature map is calculated. Each value in the filter represents a weight that needs to be optimized by the neural network during training.

3.4.3.2 Pooling layers

A typical convolutional neural network comprises of three stages. The first stage of the process performs several convolution operations that produces a sample of linear activations or feature maps as discussed in the previous section. At the second stage, each linear feature map is passed through a hidden unit or a non-linear activation function, such as the *ReLU* activation function described in §3.4.1.1. The second stage is also known as the **detector stage**. Finally, the third stage deals with a **pooling** function that replaces the output of the network with the maximum output within a rectangular grid, also known as **Max Pooling**. The pooling function is an important step that enables the representation of the input *invariant* to small translations. This helps to provide useful information about the presence of some features in the input rather than exactly where the features are located in the image. In addition, the purpose of **Max Pooling** is to reduce the spatial size of the previous layer, thus decreasing the parameters in the model. Figure 3.11 shows an application of max pooling on a previous layer. Max pooling is typically done with a filter size width and height of 2×2. It strides over the previous layer and returns the maximum value as illustrated in Figure 3.11.

3.4.3.3 Dropout

Dropout refers to randomly 'omitting' or 'dropping out' units or neurons during the training phase. Dropout acts as a regularizer to prevent overfitting. More technically, during the training process, individual nodes/neurons are either ignored with probability 1-P or are kept with probability P, such that the complexity of the model networks is removed. During the training and validation process, dropout performs various operations which are defined by the dropout

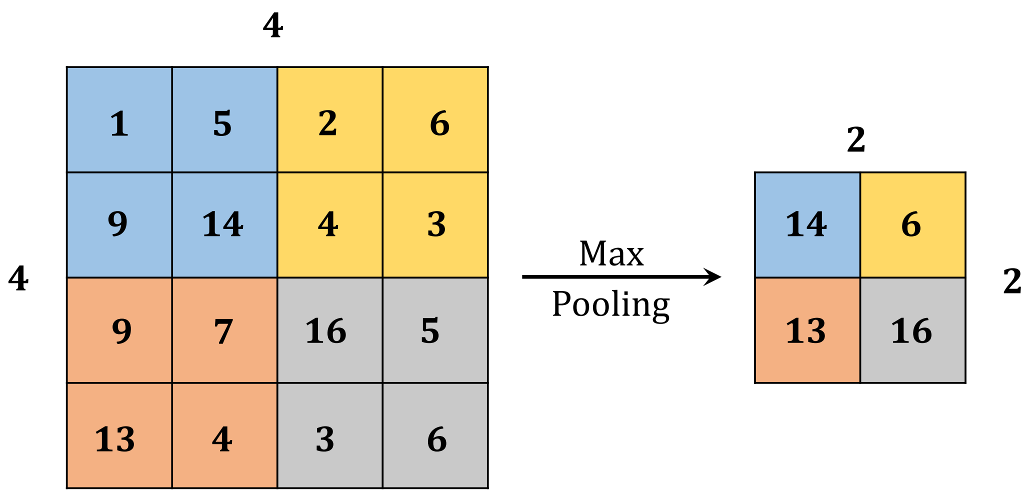


Figure 3.11 – Illustrating the dimensional reduction of the previous layer in a convolutional neural network. Spatial reduction is achieved by applying max pooling. The width and the height of the filter is 2×2, and the stride is set to 2. Max pooling works by computing the maximum value in each of the four regions.

rate, η. The latter is a parameter that needs to be defined during training and it takes values between 0 and 1. Dropout sets some of its input values to 0, and then multiplies all non-zero values by $\frac{1}{(1-\eta)}$, in such a way that the sum over all the input values remains similar. The advantage of using dropout is that it reduces the effective size of a layer to a certain percentage of the outputs from the CLs or dense layers.

3.4.3.4 Fully connected layers/Dense layers

Fully connected or dense layers work similarly to a multi-layer perceptron. Each neuron/unit in layer l has a weight connection with each unit in the previous layer $l-1$. A dot product operation is performed between the weights in layer l and the output from the units in layer $l-1$.

3.5 Training a neural network - Optimization

A neural network architecture can be built using the various layers described in the previous section. After assembling these layers, we then proceed with the training process. However, when training a ML algorithm, we need to specify an optimization function, a cost/loss function and a model structure. Therefore, during training we need to initialise all weights $\mathbf{W}$ to small random values and biases $\mathbf{b}$ to be initialised to 0 or small positive values. In the case of supervised learning for image classification, an input dataset with its associated labels are required. Hence labelled image data is used to train a model and then at the end of the process, classification probabilities are obtained. This stage is known as *forward propagation*. The difference between the true label and the classification probabilities is quantified by a *cost function*. An iterative gradient-based optimization algorithm is applied to train the DL models. The training process is mostly based on utilising the gradient to descend the cost function with respect to different weights. The gradient, together with a constant value, also known as the *learning rate*, is used to optimize the weights. This process is known as *back-propagation*.

As with many ML models, to be able to apply gradient-based learning, we need to choose a cost function to minimize and also at the end how to represent the output of the model. Most models are trained based on maximum likelihood. This implies that the cost function is simply the negative log-likelihood. One example of a cost function is the cross-entropy. Cross-entropy is a measure of difference between two probability distributions, for e.g., between the training data distribution and the predicted distribution. The cross-entropy or cost function is written as

$$\boxed{\mathrm{J}(\theta) = -\mathbb{E}_{x,y\sim\hat{\mathsf{P}}_{\text{data}}} \log \mathsf{P}_{\text{model}}\left(\mathbf{y}|\mathbf{X}\right),} \tag{3.5.1}$$

where $\mathbb{E}_{x,y\sim\hat{\mathsf{P}}_{\text{data}}}$ is the expectation of x, y with respect to an empirical distribution $\hat{\mathsf{P}}_{\text{data}}$. The form of the cost function varies from model to model, which depends on the specific form of $\log \mathsf{P}_{\text{model}}$.

The network is trained on the training data and is then evaluated on the validation data. The training process continues until we obtain the best performance on the validation set. At the end of the process, the trained model is saved and then can be used to perform prediction on the test data. It is important to emphasize that during training, both the validation and test sets have not been seen by the network.

In addition, the training process is done in small batches of training data as the forward and backward propagation are intensive computations. When the entire training set has been used once during forward and backward propagation, this is known as an *epoch*. The training process is computationally expensive and is done through several epochs. Afterwards, training stops when a specified performance criterion is met. It is fundamental to note that the training process often can under or over fit. $Underfitting$ occurs when performance on both training and validation data is poor. This usually implies that the model needs to have more parameters to fit the data. When training performance is outstanding and the performance on validation data is poor, it falls under the $overfitting$ regime. To overcome those issues, we use $regularisation$ (such as dropout), *data augmentation* and *early stopping*, which are discussed further below.

3.5.1 Data Augmentation

The best strategy for better generalisation of an ML model is to train it with large amounts of data. In practice, in a supervised learning approach the amount of labelled data is limited. One way to tackle this challenge is to create synthetic data and add it during the training process. For a classification task, this means that the classifier takes as input a complicated and high-dimensional vector $\mathbf{x}$ and maps it to a single target $\mathbf{y}$. This implies that the main task a classifier is facing is that it needs to be invariant to various transformations. We can generate new input $(\mathbf{X}, \mathbf{y})$ easily by applying a transformation on $\mathbf{X}$ inputs in our training sample.

Such data augmentation techniques have proven to be an effective method to alleviate problems caused by a lack of labelled data, where we could not otherwise train a model given insufficient data. Images being of high dimension and consisting of various variations/noise, they can be simulated by various data augmentation techniques. For example, operations like translating, scaling, and rotating the training input images by a few pixels can often improve generalization, although CNNs (the convolution and pooling techniques) already have this feature of being translational invariant.

3.5.2 Early Stopping

The training of large models shows over-fitting characteristics when we observe that the negative log-likelihood of the training process, or the training error decreases steadily with the number of epochs (number of training iterations over the datasets), but the validation set error starts to increase again. This scenario implies that a model with the best validation set error can be obtained (thus a better test set error) if the parameter setting is returned to a point in time where the lowest validation set error has been captured. For every epoch when the validation set error improves, a copy of the model parameters are stored. Therefore, for some pre-specified number of epochs, the training process stops when no improvement of the parameters over the best recorded validation set error is recorded. This strategy is called **early stopping**.

3.6 Concluding Remarks

This chapter introduced various concepts and terminology related to machine learning. This thesis is mostly concerned with supervised classification schemes, thus the concept of classification with traditional machine learning tools and deep learning concepts are illustrated in this chapter. The data splitting methodology into training and validation sets was described. In addition, techniques for evaluating the performance of a supervised machine learning classifiers was discussed. Next, the reader was presented with an introduction to deep learning, and multi-layer networks. Various types of layers and activation functions were discussed. Finally, the chapter ends with ways to avoid overfitting and underfitting while optimizing and training a neural network. Machine learning being a field that advances at a fast pace, we cover only topics related to work employed in this thesis.

4

A Hierarchical Classification Approach for variable star using Machine Learning

This chapter was originally published as a paper in MNRAS with the title: '**Comparing Multi-class, binary and Hierarchical classification of variable star using Machine Learning**'.

Zafiirah Hosenie , *Robert J. Lyon* , *Benjamin W. Stappers* and *Arrykrishna Mootoovaloo*

Accepted in MNRAS on **15 July 2019** and can be found at arXiv:1907.08189v1.

Summary of contributions: The work developed in this chapter is entirely my own. Co-authors provided supervision, suggestions of additional angles to discuss in the paper, and feedback on earlier drafts of it.

This chapter introduces the reader to the application of machine learning for variable stars classification. The following sections present an overview of the challenges in classifying variable stars and the idea of using machine learning to tackle the problem. The outline of this chapter is as follows. In §4.3, we provide a brief description of the dataset used and in §4.4 we present the feature generation techniques we employ here; while in §4.5 we explain how we build the classification pipeline. In §4.6 we apply state-of-the-art feature visualisation techniques to visualise how separable our features are before performing a multi-class classification. In §4.7, we provide an in-depth feature evaluation to determine the usefulness of our extracted features before performing a binary classification. In §4.8, we present a hierarchical taxonomy for classification and discuss our results; finally, we summarise our results and conclusions in §4.9.

4.1 Overview

Upcoming synoptic surveys are set to generate an unprecedented amount of data. This requires an automatic framework that can quickly and efficiently provide classification labels for several new object classification challenges. Using data describing 11 types of variable stars from the Catalina Real-Time Transient Surveys (CRTS), we illustrate how to capture the most important information from computed features and describe detailed methods of how to robustly use Information Theory for feature selection and evaluation. We apply three Machine Learning (ML) algorithms and demonstrate how to optimize these classifiers via cross-validation techniques. For the CRTS dataset, we find that the Random Forest (RF) classifier performs best in terms of balanced-accuracy and geometric means. We demonstrate substantially improved classification results by converting the multi-class problem into a binary classification task, achieving a balanced-accuracy rate of $\sim$99 per cent for the classification of δ-Scuti and Anomalous Cepheids (ACEP). Additionally, we describe how classification performance can be improved via converting a 'flat-multi-class' problem into a hierarchical taxonomy. We develop a new hierarchical structure and propose a new set of classification features, enabling the accurate identification of subtypes of cepheids, RR Lyrae and eclipsing binary stars in CRTS data.

4.2 Introduction

Astronomy has experienced an increase in the volume, quality and complexity of datasets produced during numerical simulations and surveys. One factor that contributes to the data avalanche is the new generation of synoptic sky surveys, for example, the Catalina Real-Time Transient Surveys (CRTS) (Drake et al., 2017). In addition, the Large Synoptic Survey Telescope (LSST, Ivezić et al. 2019) for example, which is now on the horizon, will produce ~ 15 Terabytes of raw data per night (Jurić et al., 2017). However, despite this data deluge, source variability is often still visually inspected to detect new promising candidates/variable stars. Visual inspection does have utility for detection and classification. Human experts can extract new useful information despite unevenly sampled data sets and also have the ability to distinguish noisy data from data exhibiting interesting behaviour/characteristics. They can also incorporate complex contextual information into their decision making. However, the efficacy of the manual approach decreases as the volume of data grows exponentially, as will be the case for the next generation of surveys. Visual inspection becomes inconsistent, consequently,

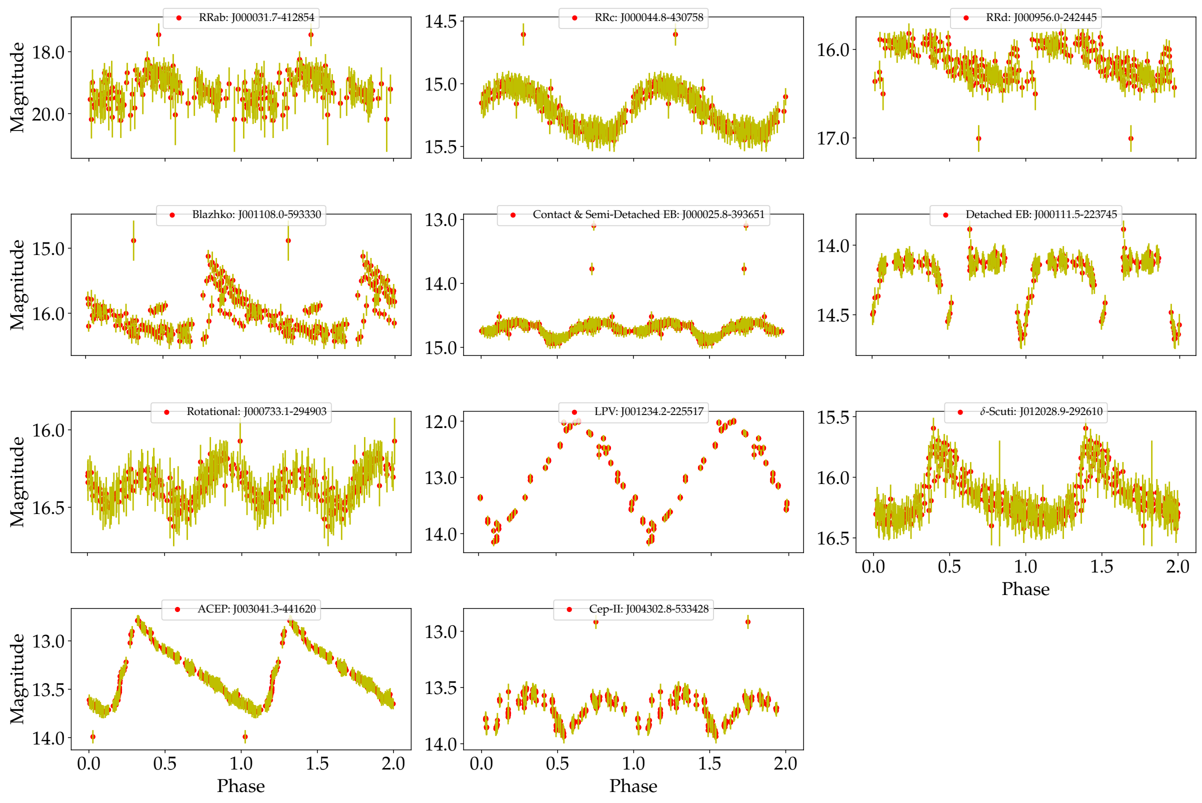


Figure 4.1 – Examples of folded light curves from the CRTS for the various types of variable stars considered in our analyses.

mistakes are made, and rare/interesting objects can be missed.

To address this problem, Machine Learning (ML) has been applied to variable-star classification in multiple time-series datasets (see Belokurov et al. 2003; Willemsen & Eyer 2007). In ML, variable stars are represented by features: independent measures that contain information useful for differentiating variable stars into their respective classes. Therefore, several developments have been made towards determining the best methods and features for describing variable-stars, including the Lomb-Scargle periodogram (Lomb, 1976; Scargle, 1982), Bayesian Evidence Estimation (Gregory & Loredo, 1992) as well as hybrid methods (Saha & Vivas, 2017). In addition, Eyer & Blake (2005) analysed the small sharp features of light curves and included them as input features to a Naïve Bayes classifier (Zhang, 2004). While Djorgovski et al. (2016) developed an automatic framework to detect and classify transient events and variable stars. They used a subset of the CRTS data to perform classification between two types of variable stars (W Uma and RR Lyrae) and obtained completeness rates of ∼96-97 per cent.

Kim & Bailer-Jones (2016) developed the `UPSILON` package to classify periodic variable stars using 16 extracted features from light curves, which achieves good results. Mahabal et al. (2017) developed a classifier based on the Convolutional Neural Network (CNN) model using labelled datasets of periodic variables from the CRTS (Drake et al., 2009; Djorgovski et al., 2011;

Mahabal et al., 2012; Djorgovski et al., 2016). They transformed a light curve (time series) into a two-dimensional mapping representation ($dm - dt$) which is based on the changes in magnitude (dm) over the time-difference (dt). Using multi-class classification, their algorithm achieved an accuracy of $\sim$83 per cent. Narayan et al. (2018) developed an ML approach to classify variable versus transient stars. Similarly, they performed a multi-class classification of combined variable stars & transients, and a "purity-driven" sub-categorisation of the transient class using multi-band optical photometry. Revsbech et al. (2018) used a data augmentation technique to mitigate the effects of bias in their data by generating additional training data using Gaussian Processes (GPs). They used a diffusion map method that calculates a pair−wise distance matrix that outputs diffusion map coefficients of the light curves. These coefficients act as feature inputs to a Random Forest (RF) classifier used to help identifying Type Ia supernova.

We found that it is fundamentally important to develop accurate and robust automated classification methods for this problem using machine learning and other statistical approaches. This work describes a new automatic classification pipeline for the classification of variable stars via application to archival data. To our knowledge, this is the first time the southern CRTS (Drake et al., 2017) data set has been used to build/evaluate an automatic classification system.

Similar work has been completed in recent years (Kim & Bailer-Jones, 2016; Mahabal et al., 2017; Narayan et al., 2018), though the features used for learning are rarely evaluated in a statistically rigorous way. We found that using a large set of features does not imply higher classification metrics. We therefore perform an in-depth analysis of ML features to understand their information content, and determine which give rise to the best classification performance. We utilize various visualization techniques and the tools of Information Theory to achieve this.

Based on our analyses we find that accurate variable star classification is possible with just seven features - much fewer than in other works. In addition, we show that this classification problem cannot be solved with a 'flat' multi-class classification approach, as the data is inherently imbalanced. To partially alleviate the 'imbalanced learning problem' (Last et al., 2017), we developed an approach inspired by earlier work in this area (Richards et al., 2011a). This involved converting a standard multi-class problem in to a hierarchical classification problem, by aggregating sub-classes in to super-classes. This results improved performance on rare class examples typically misclassified by multi-class methods. We adopt a similar methodology to Richards et al. (2011a), however we i) propose a different hierarchical classification structure, ii) use a different feature analysis/selection methodology resulting in different feature choices,

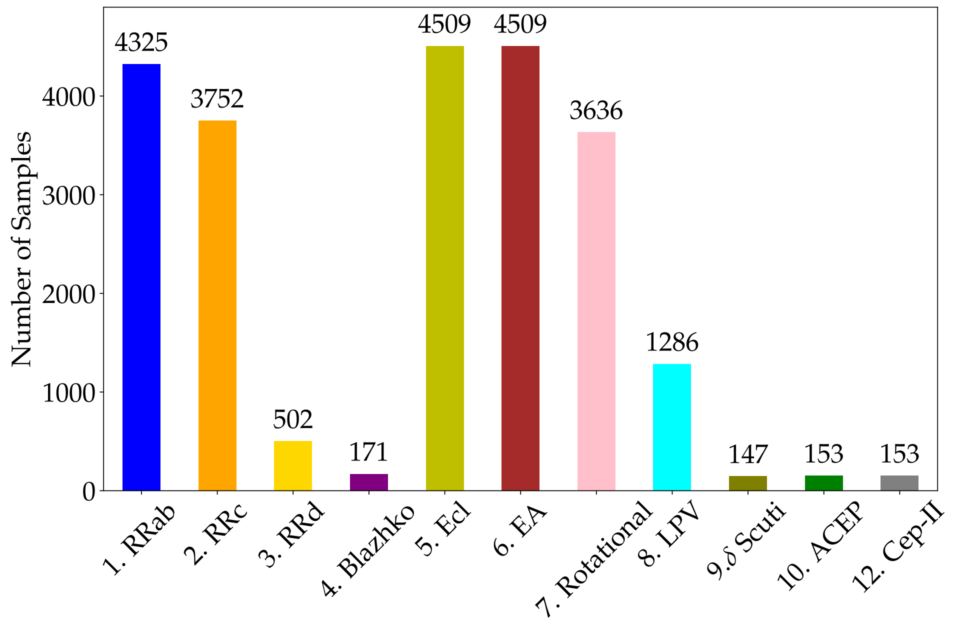


Figure 4.2 – Class distributions for the CSDR2 datasets. We downsample Type 5: semi-detached binary stars to 4,509 samples to prevent larger classes from dominating the training sets. The excluded samples ∼14,294 for Type 5: Ecl are then included in the test set for prediction. These distributions highlight the remaining imbalance in the datasets.

iii) apply hyper-parameter optimisation to build optimal classification models, and finally iv) apply the resulting approach to CRTS data.

4.3 Data

We use the publicly available CRTS (Drake et al., 2017) dataset that covers the sky with declinations between $-20°$ and $-75°$. The sources in the dataset have median magnitudes in the range $11 < V < 19.5$. The dataset contains different forms of periodic variable stars, these are stars that undergo regular changes in brightness every few hours or within a few days or weeks. The periodic variable stars in the dataset can generally be classified into three broad classes: namely eclipsing, pulsating, and rotational. A detailed overview of each type of variable star is provided in §2.2. The classes can be further divided into sub-types, for example pulsating stars consist of δ Scutis, RR Lyrae, Cepheids, and the Long Period Variables (LPVs) group which includes both semi-regular variables and Mira variable stars. The RR Lyrae class consists of RRab's (fundamental mode), RRc's (first overtone mode), and RRd's (multimode). However, many RR Lyrae stars are known to exhibit the Blazhko effect (long-term modulation) (Blažko, 1907b). In addition, the Cepheids include type-II Cepheids (Cep-II), Anomalous Cepheids (ACEPs), and Classical Cepheids. The eclipsing binary variables in the data are divided into detached binaries (EA) and contact plus semi-detached binaries (Ecl). The rotational class consists of variable stars including the ellipsoidal variables (ELL) and spotted RS Canum Venaticorum (RS CVn) systems. A more detailed overview of the data set is given in Drake et al. (2017) and Alonso-Garcia et al. (2015) gives a more detailed overview of the properties of the various types of pulsating stars.

Table 4.3.1 – The seven features used as inputs to our classification scheme. Six features based on simple statistics, are extracted directly from light curves using FATS. Note that the period feature is obtained from the Drake et al. (2017) catalogue.

Features	**Description**	**Symbol**
Mean	$\mu = \frac{1}{N}\sum_{i=1}^{N} m_i$ where m is the magnitude and N is the number of data points.	μ
Standard Deviation	$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(m_i-\mu)^2}$	σ
Skewness	$\gamma = \frac{N}{(N-1)(N-2)}\sum_{i=1}^{N}\left(\frac{m_i-\mu}{\sigma}\right)^3$	γ
Kurtosis	$kurt = \frac{N(N+1)}{(N-1)(N-2)(N-3)}\sum_{i=1}^{N}\left(\frac{m_i-\mu}{\sigma}\right)^4 - \frac{3(N-1)^2}{(N-2)(N-3)}$	$kurt$
Mean-variance	This is a simple variability index and is defined as the ratio of the standard deviation, σ, to the mean magnitude, μ.	$\frac{\sigma}{\mu}$
Period	Drake et al. (2017) used the Lomb-Scargle (L-S) periodogram analysis together with an Adaptive Fourier Decomposition (AFD) method to calculate the period of unevenly sampled data. More information about this feature can be found in Drake et al. (2017).	T
Amplitude	The amplitude is half of the difference between the median of the maximum 5% and the median of the minimum 5% magnitudes.	Amp

The dataset contains about 37,745 periodic variable stars (Catalina Surveys Data Release 2,[*] CSDR2). For our analysis, we use a sample of 37,437 out of the 37,745 stars from the CSDR2. We have excluded Type 11: Miscellaneous variable stars (periodic stars that were difficult to classify as presented in Drake et al. (2017)) and Type 13: LMC-Cep-I which as a group consists of only 10 examples. We remove the smallest classes as there are too few samples to characterise those classes, as we also know that classifiers given such data will struggle to categorise them accurately due to the imbalanced learning problem (Last et al., 2017). Furthermore, we downsample Type 5: semi-detached binary stars to 4,509 samples as this class of object originally consisted of 18,803 samples. We perform this downsampling to prevent the large class from dominating the training sets, which could otherwise potentially bias a classifier. The excluded samples of Type 5 are then included in the test set. Fig. 4.1 shows examples of folded light

[*]Catalina Surveys Data Release 2

curves for each class under consideration. We also present the number of samples considered for the 11 different types of variable stars in Fig. 4.2. Note that ∼14,294 samples of Type 5: Ecl, unused during training, were eventually used in the test set.

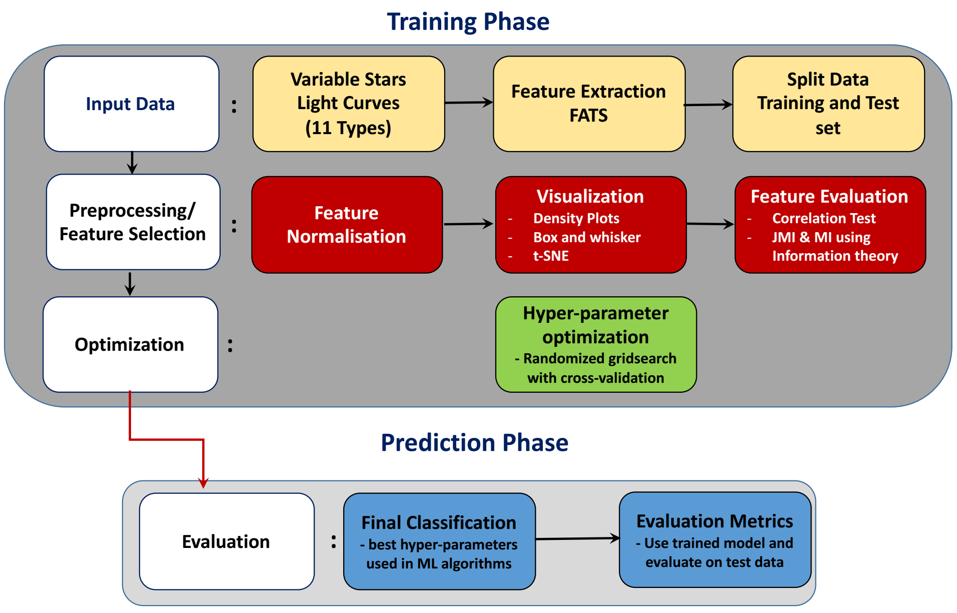

Training Phase
Input Data
Variable Stars Light Curves (11 Types)
Feature Extraction FATS
Split Data Training and Test set
Preprocessing/ Feature Selection
Feature Normalisation
Visualization
- Density Plots
- Box and whisker
- t-SNE
Feature Evaluation
- Correlation Test
- JMI & MI using Information theory
Optimization
Hyper-parameter optimization
- Randomized gridsearch with cross-validation
Prediction Phase
Evaluation
Final Classification
- best hyper-parameters used in ML algorithms
Evaluation Metrics
- Use trained model and evaluate on test data

Figure 4.3 – The different stages of our classification framework. First, we use light curves of variable stars as input data. For the first stage process, features are extracted using FATS and then are later split into training and test sets. The second stage involves data preprocessing and feature selection. In this process, the extracted features are normalised, visualised and selected based on various techniques. Afterwards, the third stage covers hyper-parameter optimisation using the randomized grid search with cross-validation methods. Finally, the last stage uses the best hyper-parameter to re-train the ML algorithms using the entire normalised training set in stage 2 and evaluate it on the normalised test set in stage 2 using various metrics to quantify the models.

4.4 Feature Generation

In general, machine learning algorithms use training data to build a mathematical model, where the training data is comprised of feature data. These features may have varying utility, that is, information rich features are desirable as they can be used to build more accurate classification systems. In this work, we generate statistical features from the light curves to characterize and distinguish different variability classes. Let $S = \{X_1, \ldots, X_n\}$, represent the set of variable star data available to us, then X_i is an individual star represented by variables known as $features$. An arbitrary feature of star X_i is denoted by X_i^j, where $j = 1, \ldots, l$. Each variable star has an associated label y such that $y \in Y = \{y_1, \ldots, y_k\}$. For multi-class scenarios the possible labels assignable to variable stars vary as $1 \leq y \leq 12$ but since we do not consider Type 11 examples for reasons given at the end of §4.3, we note that $y \neq 11$. Meanwhile for binary class classification scenarios, we consider binary labels $y \in Y = [0, 1]$.

The goal is to build a machine learning algorithm that learns to classify variable stars described by features, from a labelled input vector, also known as the training set, X_{Train}. The training set consists of pairs such that $X_{Train} = \{(X_1, y_1), \ldots, (X_n, y_n)\}$. The learnt mapping function between input feature vectors and labels in X_{Train}, can then be utilised to label new unseen stars, in X_{Test}.

In this work, we focus mainly on using the statistical properties of the data with no preconceived notions of their suitability or expressiveness as input features to our ML algorithms. As a result we focus on 7 features, of which 6 are intrinsic statistical properties relating to location (mean magnitude), scale (standard deviation), variability (mean variance), morphology (skew, kurtosis, amplitude), and time (period). These features are highly interpretable, and robust against bias. Note that we remove data points from light curves that are 3σ above or below the mean magnitude, where σ is the standard deviation and it is an important step to remove any outliers in the data. This cleaning does not alter the light curves significantly as it removes less than 1 per cent of their data points.

Afterwards, we used the FATS[†] (Nun et al., 2015) Python Library to extract these features. FATS takes as input the unfolded light curves and it outputs various statistical features: the mean, standard deviation, skew, kurtosis, mean-variance, and amplitude. We also incorporate the period for each star given in the catalogue as a feature to our ML algorithms. The description of the input features used for classification is listed in Table 4.3.1. Practitioners should be cautious when applying the mean magnitude as a feature in combination with data obtained at another telescope. It has the potential to bias a training set against fainter/brighter sources. We note that adding the telescope label as a feature may overcome this issue, but we leave that to future work.

One important aspect of the training process used to build a classification model is data preprocessing. Some ML algorithms, e.g functions/classifiers that calculate the distance between data points, will not work properly without normalisation or standardisation since the range of values of the features/raw data varies widely. We therefore employ a normalisation method, $\hat{\mathbf{S}}$, to standardize the feature data such that all values in the feature space are scaled between 0 and 1.

4.5 Classification Pipeline

In this section, we describe the process used to perform the classification of variable stars, for instance, training, hyper-parameter optimisation and prediction. We first extract features and

[†]FATS: Feature Analysis for Time Series

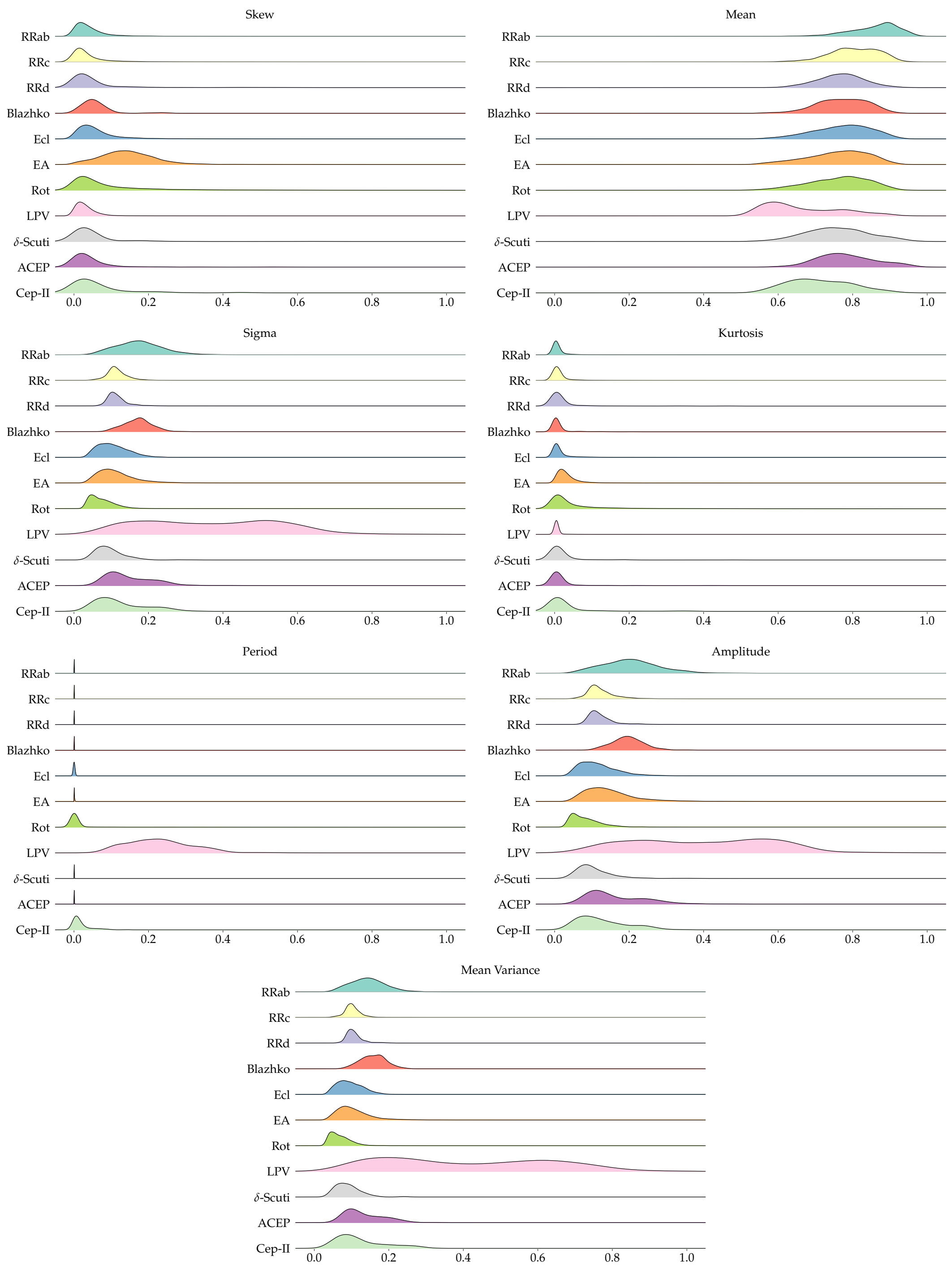


Figure 4.4 – One-dimensional density estimates of the selected features by class. The features considered are (a) Skew, (b) Mean, (c) Sigma, (d) Kurtosis, (e) Period, (f) Amplitude, (g) Mean Variance.

then split the feature data into the training (70 per cent) and the test (30 per cent) data. The training data is used to train the model while the test data is used to evaluate the performance of the trained model. Once the data is split into two, we perform a simple normalization where each feature X_i^j is divided by its maximum, $max(X_i^j)$ (see Eq. 4.5.1). The goal of normalisation is to ensure that all features use a common scale. This is beneficial for ML algorithms that are sensitive to feature distributions, for instance, distance-based algorithms that require Euclidean distance computation (for e.g. $k-$Nearest Neighbours (kNN)). Some models (e.g. Decision Tree (DT), Random Forest (RF)) are less sensitive to feature scaling. However, it is good practice to standardize when comparing between classification systems to rule out potential sources of disparity in our results. Note that in this context, we found that this simple normalisation was enough to yield good performance. For X_{Test}, the features are then rescaled using $max(X^{j}_{Train_i})$,

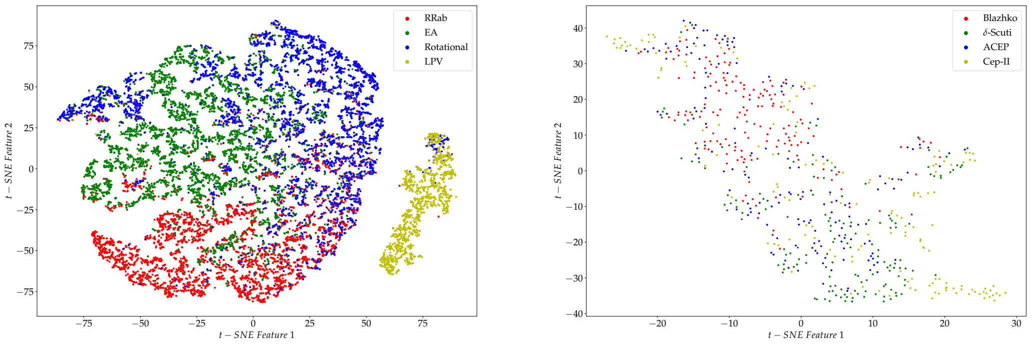
$$\boxed{X_{Train} = \left|\frac{X_{Train_i}}{max\left(X_{Train_i}\right)}\right|; \qquad X_{Test} = \left|\frac{X_{Test_i}}{max\left(X_{Train_i}\right)}\right|.} \tag{4.5.1}$$

Afterwards, we apply some feature evaluation strategies to check whether the features show some separability. Then, for our classification purposes, we apply three different classification algorithms: DT, RF and kNN, accomplished with Scikit-Learn (Pedregosa et al., 2012). A detailed overview of each classifier can be found in §3.3.1, §3.3.2 and §3.3.4.

A major problem faced when using various classifiers is hyper-parameter optimization. This is crucial for finding the hyper-parameters which yield the best overall classification performance for a specific problem. The most widely used techniques are `Hyperopt` (Bergstra et al., 2015), a Bayesian optimisation approach, grid search and manual search.

In our study, we adopt a randomized search that iterates a number of times through prespecified hyper-parameters and finds the optimum parameter that outputs the best balanced-accuracy for a classifier. Together with the randomized grid search for hyper-parameter optimization, we apply 5-fold stratified cross-validation (see §3.2.2.1) on the training set to evaluate model performance, that is, the 70 per cent training set is further split into training and validation sets. We cross-validate to ensure any observed results are real and not just due to some dataset specific effect. We note that CRTS data exhibits distributional disparities - some types of star are common in the data, while others are relatively rare. Traditional machine learning classifiers perform poorly on such data (He & Garcia, 2008). They become biased towards correctly classifying the common classes, a strategy that typically yields the greatest overall accuracy. In some cases, this bias can be overcome by re-weighting training examples so that rare examples are weighted higher than common ones. However where/when imbalance is manifested via complex data characteristics (class overlap, small disjuncts, sub-class inseparability, see (He & Garcia, 2008)), re-weighting alone is insufficient. Based on our analysis of our data presented in §4.6.1 & §4.6.2, we see enough imbalanced characteristics to suggest that our problems cannot be solved by weighting alone, nonetheless we apply re-weighting with the aim of mitigating such problems.

We then proceed with cross-validation, use the best model hyper-parameters found during this process and re-train the model with the entire 70 per cent training set. The trained model is then evaluated on the test set (30 per cent), X_{Test} using various evaluation metrics described in §3.2.3. The performance of our pipeline is evaluated on balanced-accuracy, the Geometric-mean score, F1-score, recall and standard confusion matrices. The classification pipeline is summarised in Fig. 4.3.



(a) t-SNE with large sample data

(b) t-SNE with small sample data

Figure 4.5 – (a) shows the t-distributed stochastic neighbour embedding (t-SNE) visualization for the feature set after normalisation with large sample data (RRab, EA, Rotational and LPV). We note that the classes are quite well-separated in the embedding space. (b) illustrates t-SNE visualization for the feature set after normalisation with the small sample size data (Blazhko, δ-Scuti, ACEP and Cep-II). No distinct separation is seen within the small sample dataset.

4.6 Multi-class Classification

Our main goal is to perform a multi-class classification using our classification pipeline previously described. We first apply some feature evaluation strategies to check whether our extracted features in §4.4 are good for classification. The most common methods that characterise 'good' features: look for the presence of linear correlations between the features and the class labels, and/or we sometimes indirectly measure feature utility by using classification performance (for e.g Bates et al. (2011)) as a proxy. If performance is good, we assume the features have utility. However, it is often misleading to evaluate features based on classifier performance as it varies according to the classifier used (Brown et al., 2012).

In this work, we employ the three primary considerations as in Lyon et al. (2016) to evaluate our features. A feature must i) show importance in discriminating between the different classes/types of variable stars, ii) maximise the separation between the various variable stars, and iii) lastly yield a good performance when used in conjunction with a classification algorithm. We have therefore applied two feature visualisation strategies to our features given in Table 4.3.1 before performing a multi-class classification.

4.6.1 Visualisation of feature space with one-dimensional density estimates

One way to visualise the features we extracted in §4.4, is to plot one-dimensional density estimates of the observed distribution as shown in Fig. 4.4. This plot allows us to visually compare the distributions of the normalised features for the different classes of variable stars. The plots depict distinct feature-by-feature characteristics of each class. It enables us to identify outliers and determine if there is multi-modality in the feature space. For example, the RR Lyrae skew distributions are mostly narrow and peaked, showing that these classes are well-characterised by the skewness. In addition, the density plots provide us with information of which features are fundamentally important in discriminating different sets of classes. Some classes have overlapping distributions for one or more feature variables. This applies to the RR Lyrae, eclipsing binary and the Cepheid stars, whilst other classes, such as the LPVs have trivially separable distributions, suggesting that the LPV class should be easy to classify. However, more rigorous investigation is needed to determine with confidence whether these features are useful for classification purposes.

4.6.2 t-SNE

After visualising the individual features separately, we wish to understand the relationship between our features but this is difficult to do given the feature dimensionality we are dealing with. Yet it is possible to overcome this problem by reducing the dimensionality so that it is representable in a 2- or 3-D representation space. t-Distributed Stochastic Neighbour Embedding (t-SNE, van der Maaten & Hinton (2008)) is a tool for performing this reduction and visualisation[‡].

More specifically, similar objects in high-dimensional space are clustered together with nearby points using a k-D tree (Bentley, 1975b) of all the points. Using a Student-t distribution (same as the Cauchy distribution (Cauchy, 1853)), the Euclidean distance between each point and its k-nearest neighbours is computed. This distance is further converted into a probability distribution. Similar points have a high probability of being assigned to the same class and different points have a low probability of being picked. Afterwards, t-SNE constructs a similar probability distribution using a gradient descent method, in the low-dimensional space over the points, thus minimizing the Kullback-Leibler divergence between the two probability distributions (Kullback & Leibler, 1951).

[‡]sklearn.manifold.TSNE

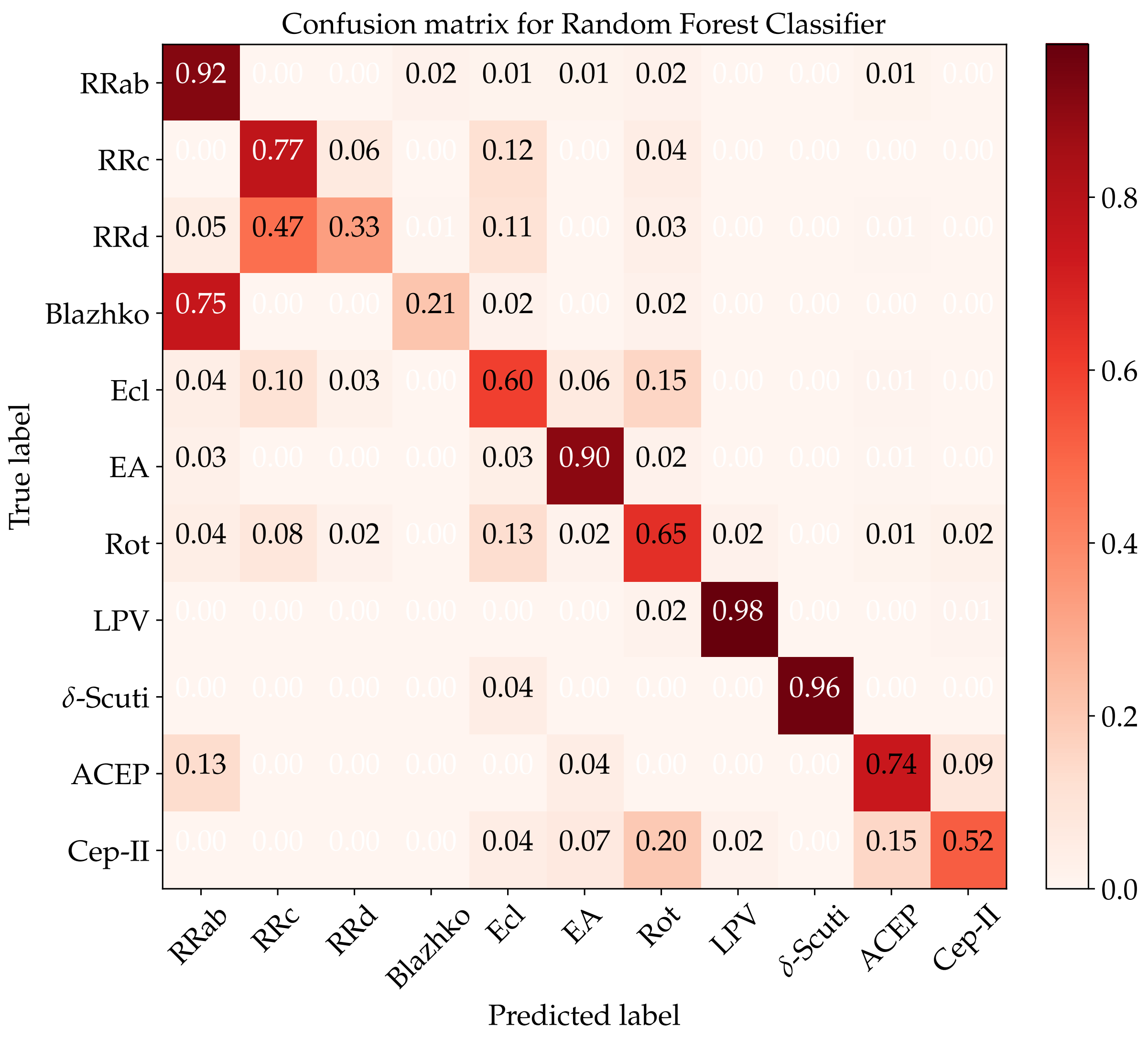


Figure 4.6 – Normalised confusion matrix for multi-class classification with the RF classifier using the best hyper-parameters from a randomized grid search cross-validation. The RF classifier was applied on a 70% training set and evaluated on a 30% test set. The classifier performs poorly on under-represented class labels.

Generally, classes that are well separated in a t-SNE visualization, yield good levels of classification performance by machine learning systems. However, the converse is not strictly true (Lochner et al., 2016). We use t-SNE only for the visualization of class separability as there are various limitations with t-SNE, as neither the sizes of the clusters nor distance between the points may be informative (Wattenberg et al., 2016). For our high dimensional visualisation, we partition the data into two categories: i) data with large sample sizes that consists of RRab, RRc, Ecl, EA, Rotational and LPV stars and ii) data with a small sample size containing RRd, Blazhko, δ-scuti, ACEP and Cep-II. Fig. 4.5(a) shows quite well-separated classes for the large sample sizes and we would expect our ML algorithms to perform well using this data. Fig. 4.5(b) strongly suggests inseparability of the classes for the small sample data. This inseparability may be attributed to the fact that there are few examples of each class. Also, another

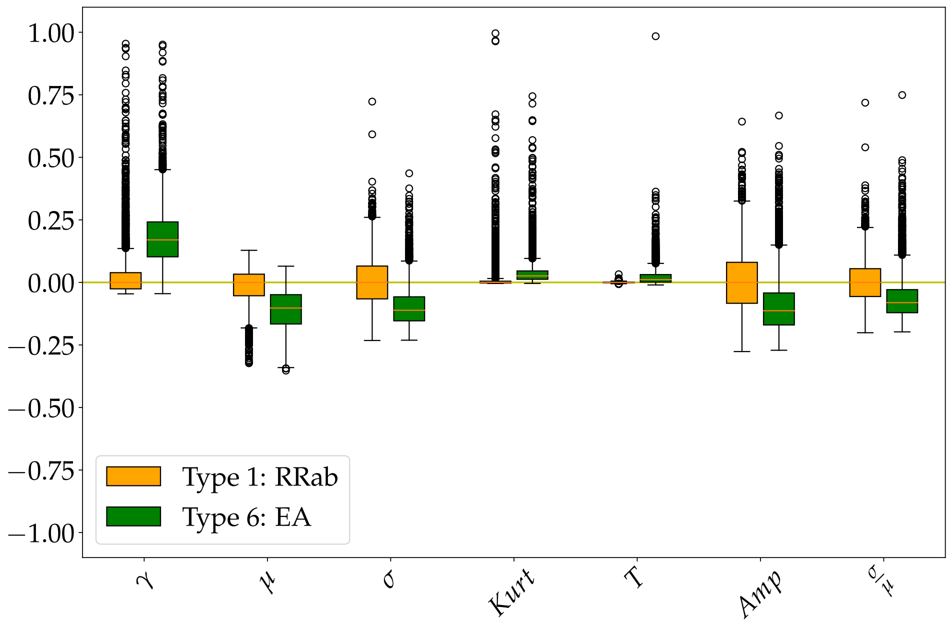


Figure 4.7 – The box and whisker plots show how the features separate RRab (Type 1) and EA (Type 6) examples. The orange coloured boxes describe the feature distribution for RRab sources, where the corresponding black circles represent extreme outliers. The box plots in green describe the EA distributions. There is no clear separation of the period (T) between the two classes. The other features show varying degrees of improved separability, and are generally easily separable at a visual level between the two different types. Refer to Table 4.3.1 for definitions of the features used.

possible explanation is that, for these stars, other features might be required to enable separability. After performing some feature visualisation, we decided to use all of the features as inputs to perform a multi-class classification.

4.6.3 Performance of Multi-class classification

We implement three classifiers, RFs, DTs and *k*NN to perform an 11-classes classification. Of these, RF achieves the best performance with a balanced-accuracy rate of $\sim$70 per cent on the 30 per cent test data. This poor performance is not surprising, given the large class imbalance present in the data, i.e. the classes with small sample sizes make up just $\sim$6 per cent of the whole dataset. The results presented here are mainly focused on the RF classifier as this achieves the best performance balanced-accuracy. In Fig. 4.6, we plot the confusion matrix of the RF classifier which depicts the predicted class versus the true class in a tabular form. The results obtained by a perfect classifier would result in values only along the diagonal of the confusion matrix. The off-diagonals give us information about the various types of errors that the classifier makes.

We note that some of the mistakes made by the classifier can be attributed to the fact that some of the science classes are physically similar, for example, the RR Lyrae, eclipsing binaries and Cepheids subclasses. We also note that most of the misclassified examples arise from classes for which there are few training examples, hence indicating bias toward larger classes, which illustrates that class imbalance is an issue. Classes comprising of many examples are more likely to be correctly classified, implying that having more examples of the under-represented science classes will help the classification.

4.7 Binary Classification

Given the complication of the multi-class classification scheme, decomposing the multi-class problem into smaller binary class problems may alleviate this issue. This may reduce the higher complexity inherent to an 11-class problem. Therefore, in this section we consider binary cases for the classification scheme.

In addition, the poor performance of the multi-class classification in the previous section can be attributed not only to class imbalance but also to the relative importance of features. Therefore, to further investigate whether the features we used are important for classification, we perform an in-depth analysis of the features used for binary classification in §4.7.1, 4.7.2 & 4.7.3 by using roughly balanced classes. Feature selection strategies can be grouped in various categories: classifier-independent (filter methods) and classifier-dependent (wrapper and embedded methods) are the most common. Whilst it is possible to evaluate feature importance using some classification models (e.g. using a random forest), we instead decouple feature analysis from any specific classifier model. We do this as wrapper methods are susceptible to overfitting, which can lead to feature choices that may not generalise well beyond the classifier used during selection (Brown et al., 2012). Thus in this work, we used mostly filter methods. These methods rank the features based on statistical measures of information content, determined by calculating the correlations/relationships between them.

Recall that in §4.6.2, we sub-divided the data set into two subsets, namely small and large samples, for which each has roughly equal number of examples. We therefore consider pair-wise combinations of each class within the small-sample dataset and perform feature selection and binary classification. This is repeated for the large-sample dataset. Here, we report on results for Type 1 (RRab) & Type 6 (EA) only for in-depth feature analyses. However, similar performance is obtained for the other pair-wise combinations. For the performance of binary classification, we report the results obtained with Type 1 (RRab) & Type 6 (EA) from the large sample dataset and Type 9 (δ-Scuti) & Type 10 (ACEP) from the small-sample dataset.

Table 4.7.1 – The point-biserial correlation coefficient and the Mutual Information $MI(X;Y)$ for each feature for the binary class pair: RRab (Type 1) and EA (Type 6) is illustrated. Higher mutual information is desirable. A high MI value implies that there is a strong correlation between the features and the class labels. In addition, the Joint Mutual Information (JMI) rank is given for each feature. A lower JMI rank is preferred. The JMI ranking illustrates the importance of each feature after taking into account the redundancy between features.

Features	Classes		
	Type (1 & 6)		
	r_{pb}	MI	JMI
Skew	0.648	0.503	2
Mean	- 0.547	0.260	6
Std	- 0.481	0.402	4
Kurtosis	0.248	0.486	3
Period, T	0.019	0.336	1
Amplitude	- 0.375	0.307	5
Mean Variance	- 0.368	0.174	7

4.7.1 Data visualisation for binary classes

The discriminating capabilities of the features are examined for some binary cases. To determine if there is some amount of separability between the two classes, we plot the box and whisker plots for the different features extracted.

The data has been pre-processed by performing the normalization described in §4.4. Here, we take Type 1 (RRab) as the negative class and Type 6 (EA) as the positive class. For each individual feature, the median value of the negative class is subtracted. This ensures a fair separability scale and centres the plots around the median, hence a distinct separation is seen more clearly between the two classes. The box plots centred on the median value represent the feature distribution of the negative class. Fig. 4.7 shows a reasonable amount of separability between Type 1: RRab and Type 6: EA. For each individual feature, we note a clear separability between the two classes, except for the Period, T. At this point, we are tempted to write-off this feature, however, for a more rigorous analysis of feature selection, we extend our investigation by looking for any linear correlation between the features and the class label.

4.7.2 Point Biserial Correlation Test

The point biserial correlation coefficient, r_{pb} (Gupta, 1960) is applied to find the relationship between a continuous variable x, and binary variable, y. Similarly to other correlation coefficients, it varies between -1 and +1, where +1 corresponds to a perfect positive relation and -1 corresponds to a perfect negative relation while a value of 0 means there is no association at all. The point biserial correlation coefficient is similar to the Pearson product moment (Pearson, 1895). More detailed information can be found in Lyon et al. (2016).

Table 4.7.1 illustrates the correlation between the seven features studied and the target class variable, for one binary class. From Table 4.7.1, it is observed that there are three features that show strong correlations ($>| 0.45 |$), for instance, the skew, the mean and the standard deviation. It can also be seen from Table 4.7.1 that the T exhibits a weak correlation for this binary class pair. This means that T might not be useful for classification. However, again one should be careful when judging the feature importance based upon their linear correlations. Features having linear correlations close to zero, may have useful non-linear correlations.

4.7.3 Information Theory

To investigate the relative importance of the features, we implement the Information Theoretic method that Lyon et al. (2016) applied to the pulsar search problem. According to Guyon & Elisseeff (2003), features that appear to be information poor, may provide new and meaningful information when combined with one or more other features. Information theory is mainly based on the entropy of a feature, X. The Mutual Information (MI, Brown et al. (2012)), which measures the amount of information between the input feature X and the true class label Y is defined as:

$$\boxed{I\,(X;Y) = H\,(X) - H\,(X \mid Y)\,,} \tag{4.7.1}$$

where $H(X)$ is the entropy and $H\,(X \mid Y)$ is the conditional entropy. The conditional probability represents the amount of uncertainty remaining in X, after knowing Y. Hence, the mutual information is indicative of the amount of uncertainty in X that is removed by knowing Y. MI can be zero if and only if X and Y are statistically independent. Therefore, it is useful for features to have high MI. A high MI indicates that a feature is correlated with the target variable, and thus can in principle be used by a classifier to yield accurate classifications.

Table 4.7.2 – A summary of the performance results of the various classifiers, ordered from best-performing to worst-performing based on the balanced-accuracy and G-mean for binary classification. Two values are given for all metrics. For each binary case presented, the first values represent metric values for Type 1 (RRab) and Type 9 (δ-Scuti). The second values are the metrics values for Type 6 (EA) and Type 10 (ACEP). In addition, the average of those metrics are summarised in single value.

Classifiers	Precision	Recall	F1-Score	G-mean	Balanced Accuracy
Type 1 (RRab) and Type 6 (EA) Classification					
RF	0.97/0.97	0.97/0.97	0.97/0.97	0.97/0.97	0.94/0.94
	∼0.97	∼0.97	∼0.97	∼0.97	∼0.94
DT	0.96/0.96	0.96/0.96	0.96/0.96	0.96/0.96	0.92/0.92
	∼0.96	∼0.96	∼0.96	∼0.96	∼0.92
***K*NN**	0.95/0.97	0.97/0.95	0.96/0.96	0.96/0.96	0.92/0.91
	∼0.96	∼0.96	∼0.96	∼0.96	∼0.92
Type 9 (δ-Scuti) and Type 10 (ACEP) Classification					
RF	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
	∼1.0	∼1.0	∼1.0	∼1.0	∼1.0
DT	1.00/0.96	0.96/1.00	0.98/0.98	0.98/0.98	0.95/0.96
	∼0.98	∼0.98	∼0.98	∼0.98	∼0.96
***K*NN**	0.98/0.98	0.96/0.98	0.97/0.97	0.97/0.97	0.93/0.94
	∼0.97	∼0.97	∼0.97	∼0.97	∼0.93

The MI value of the seven features are listed in Table 4.7.1. Using the MI method, we are able to choose the features that best describe the difference between two types of classes. However, the selected features might have redundant information, for example two features that give high MI might have the same information, in which a single selected feature could be all that is required to achieve the same classification results.

Therefore, a ranking system can be applied, which is also known as the Joint Mutual Information (JMI) criterion (Yang & Fong, 2011). The JMI enables the detection of complimentary information, and thereby a reduction in the number of features by eliminating redundancy. The JMI performs a selection from a sample of feature sets based on the amount of complementary information and ranks them. It starts from the feature that possesses the largest MI value, X^1.

A greedy iterative process is used to decide which features complement X^1 (Guyon & Elisseeff, 2003). The JMI score is given by,

$$\boxed{\mathrm{JMI}\left(X^J\right) = \sum_{X^K \in F} I\left(X^J X^K ; Y\right),} \tag{4.7.2}$$

where $X^J X^K$ is the joint probability of two features and F is the selected features. The iterative process continues until a set of features are selected or all features are ranked. The use of the JMI enables a reduction in the amount of redundancy within the features.

We have applied the JMI to our feature data as shown in Table 4.7.1. We note that features which appear to be of low information content, as determined via visual analysis and study using the point-biserial correlation coefficient, now appear to be useful. It is tempting to write off features that show low scoring linear correlations. However, it can be seen that even though the Period, T, exhibits a low linear correlation, it is ranked as the '1^{st}' best feature by the JMI. Given that we have shown that all the seven features have utility, we apply them all for binary classification.

4.7.4 Performance of Binary Classification

For binary classification, we train three classifiers independently using roughly balanced pair-wise class combinations from the large-sample and small-sample datasets. We present the results for i) Type 1: RRab & Type 6: EA and ii) Type 9-δ-Scuti & Type 10: ACEP only to show the difference between utilizing the two different sized datasets. Similar results are obtained with other pair-wise combinations of binary classes achieving balanced-accuracy and G-Mean values that vary by $\sim \pm 0.03$.

We found that the RF performs best with an overall balanced-accuracy of 0.94 with a G-mean value of 0.97 for Type 1: RRab & Type 6: EA. In addition, we observe that the two science classes in the small dataset samples (Type 9: δ-Scuti and Type 10: ACEP) have some level of misclassification in multi-class classification, while in the case of binary classification, the RF (being the best performing algorithm) outputs a balanced-accuracy of 1.0 with a G-mean value of 1.0. The results for both cases studied are summarised in Table 4.7.2.

As discussed in §4.6.3, multi-class classification gave undesirable results. On the other hand, using binary classification (§4.7.4) we show how to obtain improved balanced-accuracies for the science classes, achieving balanced-accuracies > 90 per cent in all cases we investigated. We thus now attempt to build upon the result by following an hierarchical approach to classification which will allow us to break the problem into sets of binary comparisons or smaller multi-class comparisons.

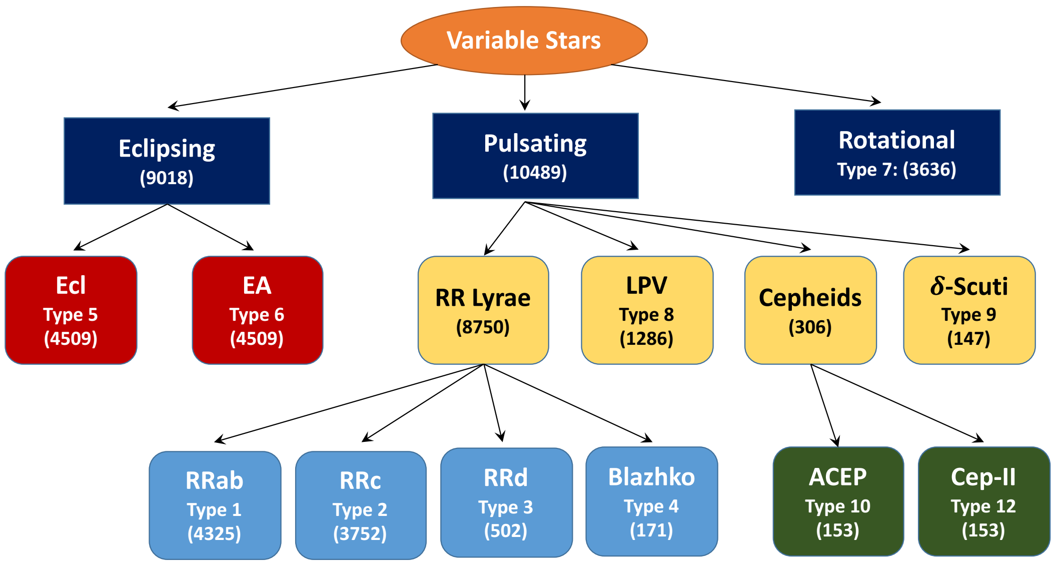


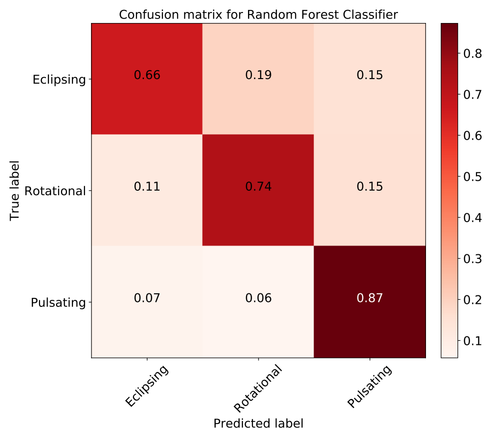
Figure 4.8 – A hierarchy of variable-star classification for data used in §4.3 which is constructed based on the understanding of their physical properties. At the top layer, the variable-stars can be split into three major classes: eclipsing, rotational and pulsating systems. The number in parenthesis represents the number of samples in each particular class.

4.8 Hierarchical Classification Concepts

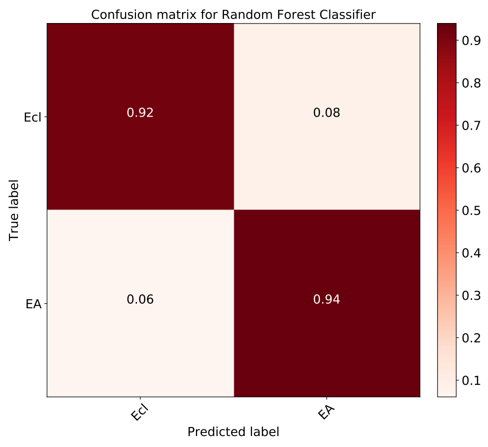
Using some astrophysical properties of the variable stars in our dataset, we group the variable sources into categories as shown in Fig. 4.8. The first level of the hierarchy is split into three broad science classes: eclipsing, rotational and pulsating. From the top level, we continue to subdivide the classes, for instance, at the third level, we are left with exactly two main classes: RR Lyrae and Cepheids with 6 science sub-classes in the final sub-node. For further details on hierarchical classification, we refer the reader to the excellent review by Silla & Freitas (2011).

Firstly, we perform a multi-class classification on the first hierarchy layer to distinguish between eclipsing, rotational and pulsating stars. This method helps to address some of the class imbalance issues. However, this separation is not perfect because the rotational class has many less examples than the other two classes. The same evaluation methodology as in §4.5 is employed and the results for the top layer is summarised in Table 4.8.1. We obtain a balanced-accuracy rate of $\sim$ 61 per cent with a G-mean value of $\sim$ 0.79 for the first layer.

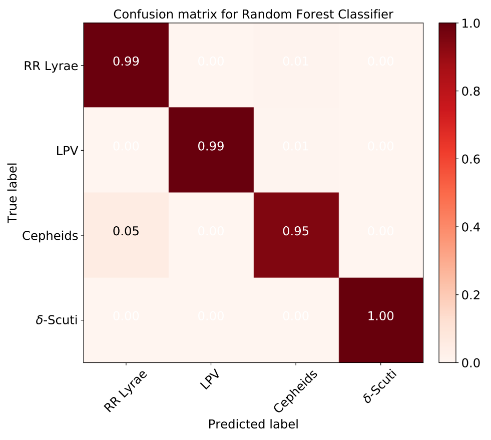
We then move down the hierarchy to perform two separate classifications for layer 2, keeping the same examples in the training sets and the test sets as in the top layer. We subdivide the eclipsing binary group into Ecl and EA classes and perform a binary classification. The result of this experiment shows that we are successful in classifying between the two classes of objects with a 0.86 balanced-accuracy and 0.93 G-mean value. In the second layer, we undertake a multi-class classification of four distinct types of classes: RR Lyrae, LPV, Cepheids and δ-Scuti. We achieve a balanced-accuracy of 0.98 in distinguishing between those classes.



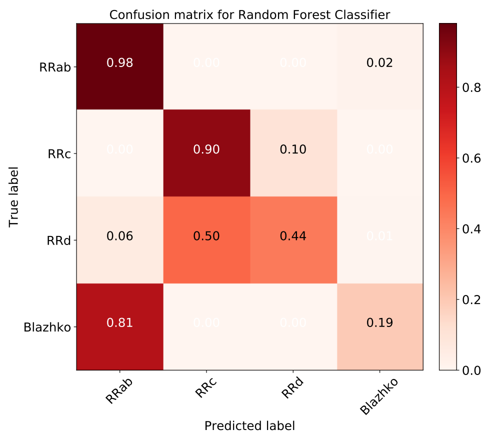
(a) First level



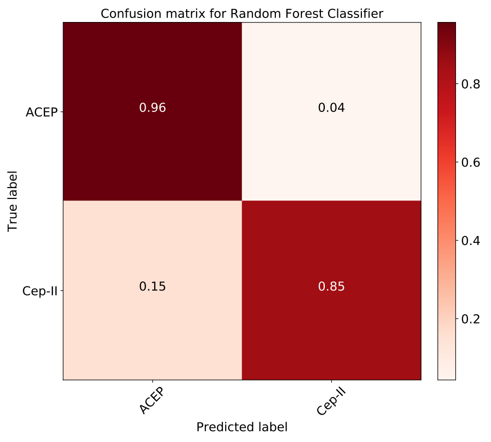
(b) Second level



(c) Second level



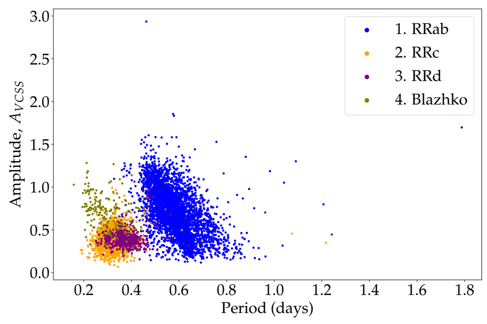
(d) Third level



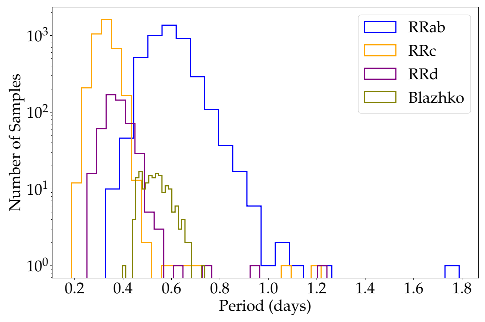
(e) Third level

Figure 4.9 – The normalized confusion matrices for the best classifier (RF) for hierarchical classification.

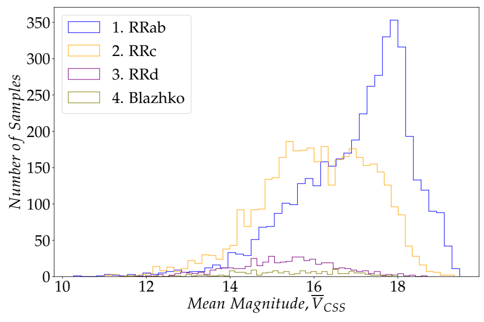
Eventually, we follow the hierarchy down to the third layer where we investigate the categorization of two classes: RR Lyrae and Cepheids. Our aim here is to find to what extent we will be successful in distinguishing between the sub-classes of RR Lyrae (RRab, RRc, RRd and Blazhko) and Cepheids (ACEP and Cep-II). The analysis shows that we are successful in distinguishing between RRab and RRc with high balanced-accuracy. However, most of the RRd sources are classified as RRc and Blazhko sources are classified as RRab. To check whether our results are affected by class imbalance, we downsample RRc in the training set to the same number of samples as RRd and perform a binary classification. This process is repeated for RRab, whereby the number of objects is decreased to the same number as in Blazhko class. We train a binary classifier, keeping the test set the same for RRc & RRd and RRab & Blazhko. We found that using balanced classes does increase the performance of the classifier significantly, for instance, we are able to distinguish between RRab & Blazhko and RRc & RRd with a balanced-accuracy rate of 80 per cent and 75 per cent respectively.



(a) The period-amplitude distribution



(b) The period distribution



(c) The magnitude distribution

Figure 4.10 – We plot the distribution of RR Lyrae sub-types (RRab, RRc, RRd and Blazhko) using available data from the ascii catalogue of the sources in Drake et al. (2017).

For an in-depth analysis of the RR Lyrae sub-classification, we use the data provided by Drake et al. (2017) that utilised the Adaptive Fourier Decomposition (AFD) method (Torrealba et al., 2015) to determine the period of each source. To see how the visually selected periodic variables differ from the initial candidates; we plot the amplitude and the period distribution of the variable stars available in the catalogue. From Fig. 4.10, we note that it is a challenging task to separate the subclasses: RRab & Blazhko and RRc & RRd. We found that our ML classifier also struggles to separate those classes. In addition, we observe a clear separation between

RRab and RRc, and our classification pipeline is also able to separate these two classes. In the same vein, Malz et al. (2018) point out it is generally challenging to have a clear separation between RRc and RRd even though they have different pulsating modes and due to the rarity of RRd sources, they are often subsumed by RRc labels. Moreover, Drake et al. (2017) argued that RRd phased light curves often appear like those of RRc and Blazhko stars often resemble RRab's when phase-folded over multiple cycles.

Furthermore, we analyse the classification of Cepheids and we obtain an error classification rate of ∼19 per cent. On the same note, Drake et al. (2017) argued that classifying ACEP and BL Her (a sub-groups of Cep-II) is difficult for field stars because of the mixture of stellar populations in the field and the inadequate distance information.

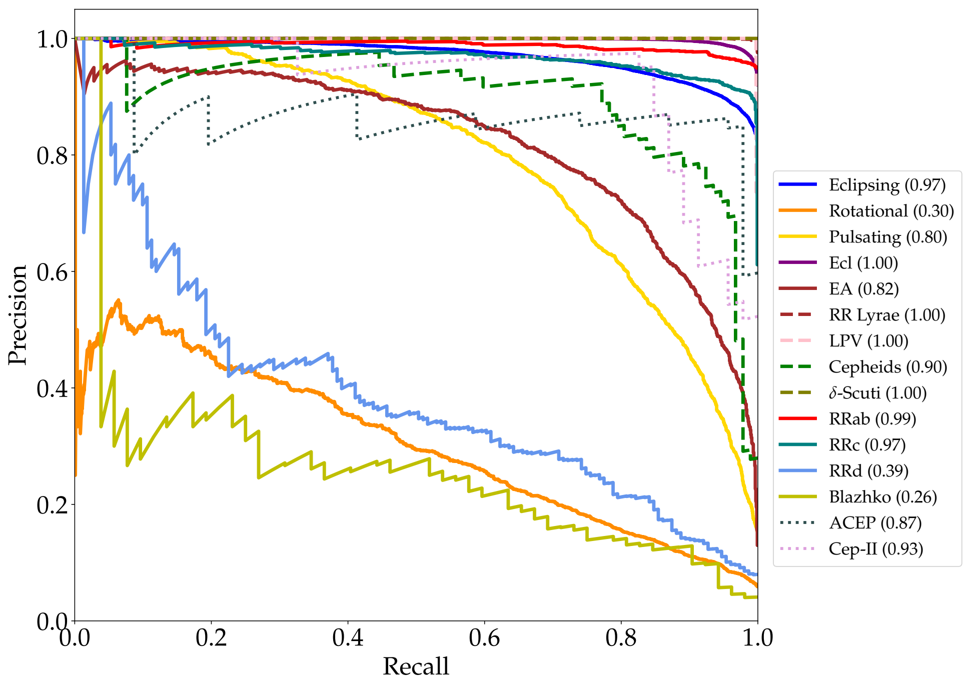


Figure 4.11 – Precision-Recall curves for each node in the hierarchical model. Each curve represents a different variable stars with the area under the precision-recall curves score in brackets. This metric is computed on the 30% of the dataset used for testing, except that Type 5: Ecl stars has ∼15647 samples.

For our hierarchical model, we present the RF classifier results only as it performs well compared to other classifiers discussed in this chapter. From the results in Table 4.8.1 and Fig. 4.9, we see immediately a disparity in performance among the classes. This discrepancy in misclassification is due to the comparative size of each of the science classes. We note that we obtain high performance with classes that are data-rich. To alleviate the class-imbalance problem, one can either gather different catalogues for the under-sampled classes or augment the existing available catalogues via sophisticated statistical machine learning approaches.

Table 4.8.1 – A summary of the performance results of the RF classifier for the variable-star hierarchical classification. We report metrics per class, separated by '/'. Also, we compute the average metrics taking into consideration the overall classes, summarised in a single value.

Precision	Recall	F1-Score	G-Mean	Balanced Accuracy
First Level: Eclipsing, Rotational and Pulsating Classification				
0.97/0.19/0.51	0.66/0.74/0.87	0.79/0.30/0.64	0.78/0.78/0.86	0.59/0.60/0.75
∼0.86	∼0.70	∼0.74	∼0.79	∼0.61
Second Level: RR Lyrae, LPV, Cepheids and δ-Scuti				
1.00/1.00/0.75/0.96	0.99/0.99/0.95/1.00	0.99/1.00/0.84/0.98	0.99/1.00/0.97/1.00	0.98/0.99/0.93/1.00
∼0.99	∼0.99	∼0.99	∼0.99	∼0.98
Second Level: Ecl and EA				
0.90/0.50	0.92/0.94	0.95/0.65	0.93/0.93	0.86/0.86
∼0.95	∼0.92	∼0.93	∼0.93	∼0.86
Third Level: RRab, RRc, RRd and Blazhko				
0.96/0.93/0.36/0.29	0.98/0.90/0.44/0.19	0.97/0.91/0.40/0.23	0.97/0.92/0.65/0.44	0.94/0.85/0.40/0.18
∼0.90	∼0.90	∼0.90	∼0.92	∼0.86
Third Level: ACEP and Cep-II				
0.86/0.95	0.96/0.85	0.91/0.90	0.90/0.90	0.82/0.80
∼0.91	∼0.90	∼0.90	∼0.90	∼0.81

Table 4.8.2 – Comparison of our Hierarchical model, Multi-class model and UPSILON package (Kim & Bailer-Jones, 2016). For a fair comparison, we test these models on 8 classes and report the scores in terms of recall and F1-score. The model with higher classification score in highlighted in blue.

Types of Variable Stars	**UPSILON Model with 16 Features**		**Our Multi-Class Model with 7 Features**		**Our Hierarchical Model with 7 Features**	
	Recall	**F1-Score**	**Recall**	**F1-Score**	**Recall**	**F1-Score**
RRab	0.75	0.86	0.92	0.73	**0.98**	**0.97**
RRc	0.78	0.87	0.77	0.46	**0.90**	**0.91**
RRd	0.11	0.20	0.33	0.14	**0.44**	**0.40**
Ecl (EC & ESD)	0.75	0.73	0.60	0.74	**0.92**	**0.95**
EA	**0.97**	**0.98**	0.90	0.68	0.94	0.65
LPV	0.92	0.96	0.98	0.95	**0.99**	**1.00**
δ-Scuti	0.86	0.93	0.96	0.70	**1.00**	**0.98**
Cep-II	0.38	0.55	0.52	0.32	**0.85**	**0.90**

In Fig. 4.11, we plot the precision-recall curve for each class and we note that the classification performance is very good. The area under the precision-recall curve values are greater than 0.85 for several classes, except for Type 7: Rotational, Type 3: RRd and Type 4: Blazhko. This correlates with the results presented in Table 4.8.1. In addition, we compare our proposed hierarchical approach to an already implemented machine learning package known as UPSILON (Kim & Bailer-Jones, 2016). The latter used a RF classifier, trained on 16 features extracted from OGLE (Udalski et al., 1997) and EROS-2 (Tisserand et al., 2007) periodic variable stars light curves; while our model is trained on 7 features from CRTS data. The comparison is made with only 8 classes out of 11 as UPSILON has not been trained on all the variable stars available in CRTS data. We report the results in terms of recall and F1-score in Table 4.8.2. It is seen that our hierarchical model outperforms the UPSILON model in classifying most of the variable stars, except for Type 6: EA. We can plausibly say that our hierarchical classifier yields good performance with 7 features in classifying sub-groups of stars as compared to the UPSILON package. For a comparison in terms of features used, we report results when using a 'flat' multi-class model (RF) in §4.6.3 with 7 features and compare it with the hierarchical model. The hierarchical model outperforms both the UPSILON model and the multi-class model developed in this work. This clearly shows that it is not necessary to extract many features to obtain higher classification metrics. In addition, we have shown that converting a 'flat' multi-class problem into a hierarchical system improves variable star classification. Comparing the classification performance of the HC and multi-class approach is difficult and nuanced as the multi-class approach seeks to improve overall classification performance (accuracy, recall), while our HC approach aims to improve rare class recall in particular. Before deciding on which approach to use, practitioners should consider their goal and choose an appropriate evaluation metric based upon this.

4.9 Conclusion and Future work

With the upcoming synoptic surveys (for e.g. LSST, Ivezić et al. (2019)), automated transient and variable-star classification is becoming an increasingly important field in astronomy. It presents a difficult computational challenge which we have nonetheless attempted to tackle in this work. We describe the core challenges associated with automatic variable-star classification and subsequently explored the nature of the data to be processed. We then applied various approaches with the aim of accurately classifying 11 types of variable star. Some of the methods we applied are similar to current techniques, while others are new to this field. Compared to other related work, we utilized only 7 input features during classification, where 6 features are based on statistical properties of the unfolded data and the Period, T, is obtained directly from the data catalogue. We used these features as inputs to three separate commonly employed ML algorithms. We demonstrate that the RF classification algorithm performed best in all test cases. Treating the variable-star classification as a multi-class problem with 11 classes, results in a poor performance. In doing so, the RF algorithm is efficient at identifying RRab, RRc, Ecl, EA, Rotational and LPV classes, but is unsuccessful in distinguishing, for example, δ-Scuti and ACEP. However, by decomposing the multi-class problem into several binary classification problems, we gain a significant improvement in balanced-accuracy - with a balanced-accuracy rate of 1.0 for the classification of δ-Scuti and ACEP. Our results suggest that decomposing a multi-class problem into several binary classification steps could yield improved results.

We therefore developed a hierarchical approach for classifying the 11-variable-star classes in our data, by aggregating those classes that show similarities. We show that a hierarchical taxonomy for n-class objects improves the classification rate. We are now successful in identifying sub-types of cepheids and eclipsing binaries with a balanced-accuracy rate of 81 per cent and 86 per cent respectively. Whilst the hierarchical approach does not work well for all classes, this is understandable in cases with high class-imbalances and a lack of training examples.

We have presented an approach to analysing variable star data in such a way, that is beneficial to machine learning feature analysis and extraction. This process yields insights that allow us to obtain improved classification performance. In other words, we have taken a principled approach to feature analysis and design, especially for multi-class problems with a highly imbalanced datasets. We employ new methods for feature selection and evaluation and we show that converting a multi-class problem towards a hierarchical classification scheme helps to reduce the class-imbalance problem as well as provide a significant improvement for variable stars classification. In the near future, we wish to investigate the impact of data augmentation on the performance of our ML classifiers. Moreover, flagging data (as we did for the "Miscellaneous class") is often undesirable. One school of thought is to treat them as outliers but another might argue that these could perhaps indicate new discoveries. It is a major task with such a classification scheme, since we now have to tackle the problem in an unsupervised way, a topic currently in its infancy.

5

Imbalance Learning for variable star classification using Machine Learning

This chapter was originally published as a paper in MNRAS with the title: '**Imbalance Learning for variable star classification using Machine Learning**'.

Zafiirah Hosenie , *Robert J. Lyon* , *Benjamin W. Stappers*, *Arrykrishna Mootoovaloo* and *Vanessa McBride*

Accepted in MNRAS on **27 February 2020** and can be found at arXiv:2002.12386v1.

Summary of contributions: The work developed in this chapter is entirely my own. Co-authors provided supervision, suggestions of additional angles to discuss in the paper, and feedback on earlier drafts of it.

This chapter introduces the reader to imbalance learning problem encountered in the previous chapter for variable stars classification. The following section presents an overview of the challenges in classifying variable stars and provides three methods of data augmentation to tackle the imbalance learning problem. The structure of this chapter is as follows. In §5.3, we describe the data set used in our analysis; while in §5.4, the three data augmentation algorithms used, are explored. In §5.5, we provide a description of the various stages in the hierarchical classification pipeline; in §5.6 we present the classification results and finally, we conclude in §5.7.

5.1 Overview

The accurate automated classification of variable stars into their respective sub-types is difficult. Machine learning based solutions often fall foul of the imbalanced learning problem, which causes poor generalisation performance in practice, especially on rare variable star sub-types. In previous work, we attempted to overcome such deficiencies via the development of a hierarchical machine learning classifier. This 'algorithm-level' approach to tackling imbalance, yielded promising results on Catalina Real-Time Survey (CRTS) data, outperforming the binary and multi-class classification schemes previously applied in this area. In this work, we attempt to further improve hierarchical classification performance by applying 'data-level' approaches to directly augment the training data so that they better describe under-represented classes. We apply and report results for three data augmentation methods in particular: *R*andomly *A*ugmented *S*ampled *L*ight curves from magnitude *E*rror (`RASLE`), augmenting light curves with Gaussian Process modelling (`GpFit`) and the Synthetic Minority Over-sampling Technique (`SMOTE`). When combining the 'algorithm-level' (i.e. the hierarchical scheme) together with the 'data-level' approach, we further improve variable star classification accuracy by 1-4%. We found that a higher classification rate is obtained when using `GpFit` in the hierarchical model. Further improvement of the metric scores requires a better standard set of correctly identified variable stars and, perhaps enhanced features are needed.

5.2 Introduction

Astronomy is now in an era dominated by an explosion of big data, produced with current and future surveys, such as OGLE (Udalski et al., 2008, 2015), CRTS (Drake et al., 2017) and Kepler (Koch et al., 2010) among others, thus, relying solely on visual inspection for classification is becoming impractical. Therefore, automatic classification pipelines are required to categorize an unprecedented amount of variable star light curves into known or unknown classes for various astrophysical applications. Accordingly, machine learning has heavily been studied to solve classification problems, for instance, uncovering aberrant phenomena encountered in observations, also known as unsupervised anomaly detection (Chen et al., 2018; Zong et al., 2018) and automatic classification of variable stars (Kim & Bailer-Jones, 2016; Benavente et al., 2017; Mahabal et al., 2017; Narayan et al., 2018; Pashchenko et al., 2018; Tsang & Schultz, 2019; Zorich et al., 2020).

However, a major issue that impedes the successful automated classification of astronom-

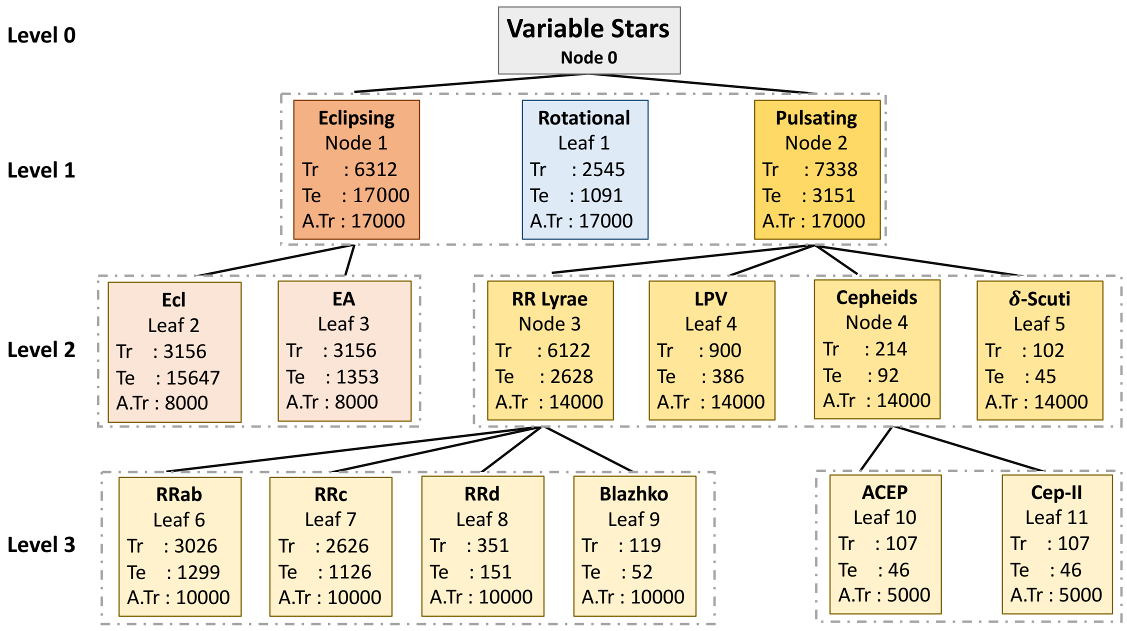


Figure 5.1 – Hierarchical Tree classification with automated light curves augmentation for CRTS Data. The number of training examples (real LCs) is represented by *Tr*, the number of training examples after augmentation (both real and synthetic LCs) is represented by *A.Tr* and the number of test examples (real LCs) is represented by *Te*. At level 1, the real LCs in the training set are augmented and the dotted square represents a trained model (RF/XGBoost classifier). During testing phase, the classified examples in the test set move down the hierarchy at level 2. Afterwards, real LCs in the training set in level 1 moves to their respective branches at level 2. The real LCs are augmented and features are extracted. This process is repeated until it reaches all leaves in the hierarchy.

ical data is known as the imbalanced learning problem. This occurs when we wish to organise data into distinct groups known as "classes", using examples to guide a process known as "classification". When there is a large distributional difference between the number of examples belonging to each class, minority and majority classes form. When the imbalance between the minority and majority classes is large, problems can arise when attempting to build standard machine learning classification algorithms, ultimately resulting in poor categorisation performance. This happens as such algorithms are usually optimised to achieve maximum accuracy. However this is trivially achievable in imbalanced datasets by always assigning the majority class label when making predictions. This leads to biased classifiers that obtain high predictive accuracy for majority class, but poor predictive accuracy for minority classes, which are more often than not, the focus of our interest.

Imbalanced learning problems occur in many domains, for instance in fraudulent phone call identification (few calls are fraudulent, (Fawcett & Provost, 1996)), or text classification (in cases where there is either more positive or more negative sentiments). In astronomy, this issue becomes acute given that datasets must often be searched for rare or unusual phenomena which may not be accurately defined in advance. This problem impacts the classification of variable stars in particular, as some types of variable star are uncommon, making it difficult to build systems to be able to recognise them. In astronomy, several works have tried to address the problem of class imbalance to date (Hoyle et al. 2015; Lochner et al. 2016; Narayan et al. 2018; Revsbech et al. 2018; Agarwal et al. 2020).

There are two approaches for dealing with class imbalance problems (He & Garcia, 2008). The first are generally known as 'algorithm level' approaches. These seek to modify classification algorithms directly, to better accommodate imbalanced class distributions. This can involve, for example, adapting the learning function at the heart of the algorithm to favour metrics other than accuracy during training and also applying hyperparameter tuning while training the algorithm (See §5.5.4). Algorithm level approaches make an implicit assumption - that the data is sufficiently descriptive and statistically characteristic of the classes under consideration, and changes to the algorithm alone will enable this data to yield good classification performance.

Alternatively, 'data level' approaches seek to modify the data given to a classification algorithm, with the aim of improving classification performance. Data level approaches can be as simple as balancing training data artificially via an appropriate sampling method, or as complex as generating artificial samples to balance the training set. Data level approaches assume that classification algorithms will be capable of separating the classes under consideration, given appropriate training data. Hybrid approaches mix the two techniques when faced with difficult problems. For instance, in some cases modifying an algorithm will not produce the improvement expected, if the classification problem at hand exhibits excessive class overlap, disjuncts, or is affected by small sample sizes (i.e. some classes are genuinely rare), whilst in some cases trying to balance training sets will not work if the information content of the training samples is too low to allow a classifier to delineate effective class boundaries.

In previous work, we attempted to develop a variable star classifier together with various techniques of feature selection and feature importance, and ran into the imbalanced learning problem. To overcome this, we attempted to modify the algorithms used for classification, and ultimately proposed a successful hierarchical classification system. We compared the hierarchical system (using 7 features) with the `UPSILON` package (Kim & Bailer-Jones, 2016) (using 16 features). Whilst hierarchical system was effective, recall on minority classes could be stubbornly low relative to majority classes. In other domains such problems are overcome by balancing the training distribution directly. This approach implies the minority class is sufficiently described in the training data to solve the imbalance, and further that the classifier used is sensitive to the class size. We believe this to be the case, thus we proceed similarly. We present a hybrid approach to overcoming imbalances, which represents a principled and pragmatic approach to this problem. Thus in this work, we improve the Hosenie et al. (2019a, hereafter H19) classification scheme by adding a sufficient amount of data, such that each class has an equal amount of training examples. This can be achieved by simulating more data or gathering more real data (which is often difficult).

Balancing training sets directly can be difficult. Fortunately, techniques such as Synthetic Minority Over-sampling Technique (`SMOTE`, Chawla et al. 2011), random values drawn from the Gaussian distribution (Peterson et al., 1998) and Gaussian Processes (GPs, Rasmussen 2003) modelling (`GpFit`) can simplify the problem to a large extent by simulating lightcurves. GPs have been used in several works to synthetically augment biased supernova training sets (Lochner et al., 2016; Narayan et al., 2018; Revsbech et al., 2018), variable stars (Faraway et al., 2016; Castro et al., 2018; Martínez-Palomera et al., 2018) and lightcurve de-trending (Aigrain et al., 2016).

In this work, we are concerned only with periodic variable star classification and we present GPs for augmenting periodic variable star data using folded light curves. Second, we propose a new method, *R*andomly *A*ugmented *S*ampled *L*ight curves from magnitude *E*rror (`RASLE`[*]), to periodic variable star data for the first time, which synthetically augments the training set by sampling from the magnitude errors. We then compare the three data augmentation methods (`SMOTE`, `GpFit` & `RASLE`) and their utility for improving variable star classification, trained with either a Random Forest (RF, Breiman, 2001) classifier or eXtreme Gradient Boosting (XGBoost, Chen & Guestrin, 2016) classifier. Finally, we incorporate a Bayesian Optimisation approach to find the best hyper-parameters for the RF and XGBoost in the hierarchical classification (HC) scheme. We achieve an improvement of 1-4 percent in terms of balanced-accuracy and G-mean scores at all levels in the HC, compared to the results of H19.

5.3 Data

The Catalina Real-Time Transient Survey (CRTS, Drake et al. 2017) has produced a catalogue of periodic variable stars from 6 years of optical photometry from the Siding Spring Survey (SSS). We consider only 11 classes from the CSDR2[†] dataset as presented in Table 5.3.1 for our analysis. From Table 5.3.1, we observe that the data is heavily imbalanced. Thus to simplify

[*]After the preparation of this manuscript, we learnt that another team Gabruseva et al. (2020), has come up with a similar method independently.

[†]Catalina Surveys Data Release 2

Table 5.3.1 – Sample size of classes in CRTS data. The class distribution is extremely imbalanced, such as Ecl are over-represented.

Types of variable stars	**N_{Objects}**
RRab (fundamental mode)	4325
RRc (first overtone mode)	3752
RRd (multimode)	502
Blazhko (long-term modulation)	171
Contact & Semi-Detached Binary: Ecl	18803
Detached Binary: EA	4509
Rotational: Rot	3636
Long Period Variable: LPV	1286
δ-Scuti	147
Anomalous Cepheids: ACEP	153
Type-II Cepheids: Cep-II	153

our experimentation, we reduced the size of the largest class (Ecl) via random under-sampling. We down-sample this class to 4509 (this makes the number of Ecl examples comparable to the next biggest class, EA) and the remaining Ecl light curves (LCs) are then used for prediction. This is why the number of samples available for testing exceeds those for training as shown in Fig 5.1.

5.4 Data Augmentation

While the under-sampling methods (i.e. downsample Ecl and developing the hierarchical system) help to address some of the class imbalance issues, they are themselves insufficient, as minority class performance was not good enough for our purposes. We therefore provide three ways to over-sample the data, a form of data augmentation necessary as some of the classes still outnumber other classes (see *Tr* values in Fig 5.1). We augment the data via the generation of artificial data, in order to increase the number of training samples by generating similar but not identical examples. In principal the more data we have, the better our ML models will be as this technique helps to reduce overfitting. In this work, we consider three methods of augmentation, (i) `SMOTE`, (ii) `RASLE`, and (iii) `GpFit`.

5.4.1 Synthetic Minority Over-sampling Technique

The Synthetic Minority Over-sampling Technique (`SMOTE`) inserts artificially generated minority class examples into a dataset, by operating in "*feature space*" rather than "*data space*". This technique helps to balance the overall class distribution. The standard implementation of `SMOTE` utilizes $k-$nearest neighbours (Buturovic, 1993) to group similar class objects and to determine

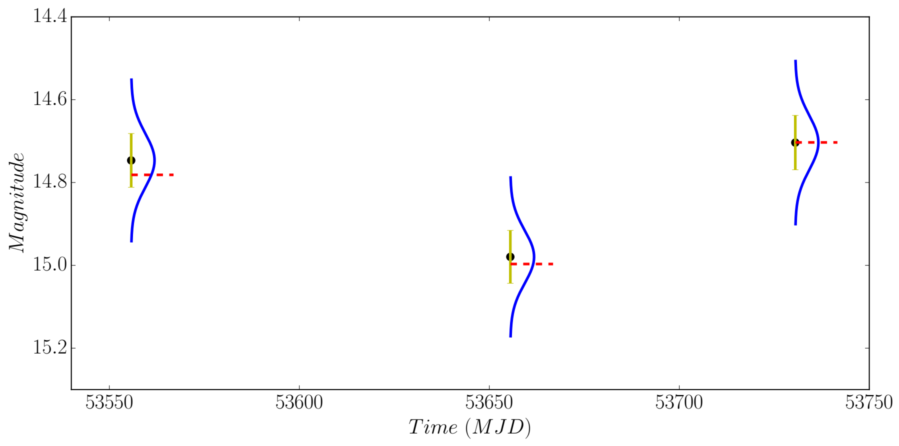


Figure 5.2 – RASLE: generating new light curves by random sampling from a normal distribution. The true magnitude along with its error bars is shown in black and yellow. We assume a normal distribution with mean equal to the true magnitude and with sigma equal to the error in magnitude. We randomly draw one sample (red-dashed line) from each normal distribution to produce a completely new light curve.

which class categories are in the minority class and need over-sampling. To generate a new synthetic example, the $k-$nearest neighbours method is further used by first selecting an example in the minority class. The collection of feature values describing this example, its feature vector, is then combined with the feature vectors of one of its k nearest neighbours chosen at random. The difference between the vectors of these two examples is computed and subsequently multiplied by a random number drawn between 0 and 1. This produces an entirely new synthetic feature vector. This process is repeated until enough synthetic examples have been created. Finally, the new augmented training set is comprised of both the synthetic examples and the real minority examples. In our pipeline, we utilize the 'regular-SMOTE' algorithm from the imbalanced-learn[‡] (Lemaitre et al., 2016) package.

5.4.2 *R*andomly *A*ugmented *S*ampled *L*ight curves from magnitude *E*rrors

The artificial examples generated by standard SMOTE, may not truly represent data recorded during observations. One way around this is to generate artificial samples from existing data points in a more scientifically valid way. That is we randomly sample a selection of rare class examples, take their primary characteristics, and generate new examples from them by perturbing them in a principled way. We do this using the *R*andomly *A*ugmented *S*ampled *L*ight curves from magnitude *E*rrors (RASLE) method.

The application of RASLE is employed on unfolded-LCs, that is, each variable star is described by its time, magnitude and error in magnitude. Using this information, we generate

[‡]https://imbalanced-learn.readthedocs.io/en/stable/index.html

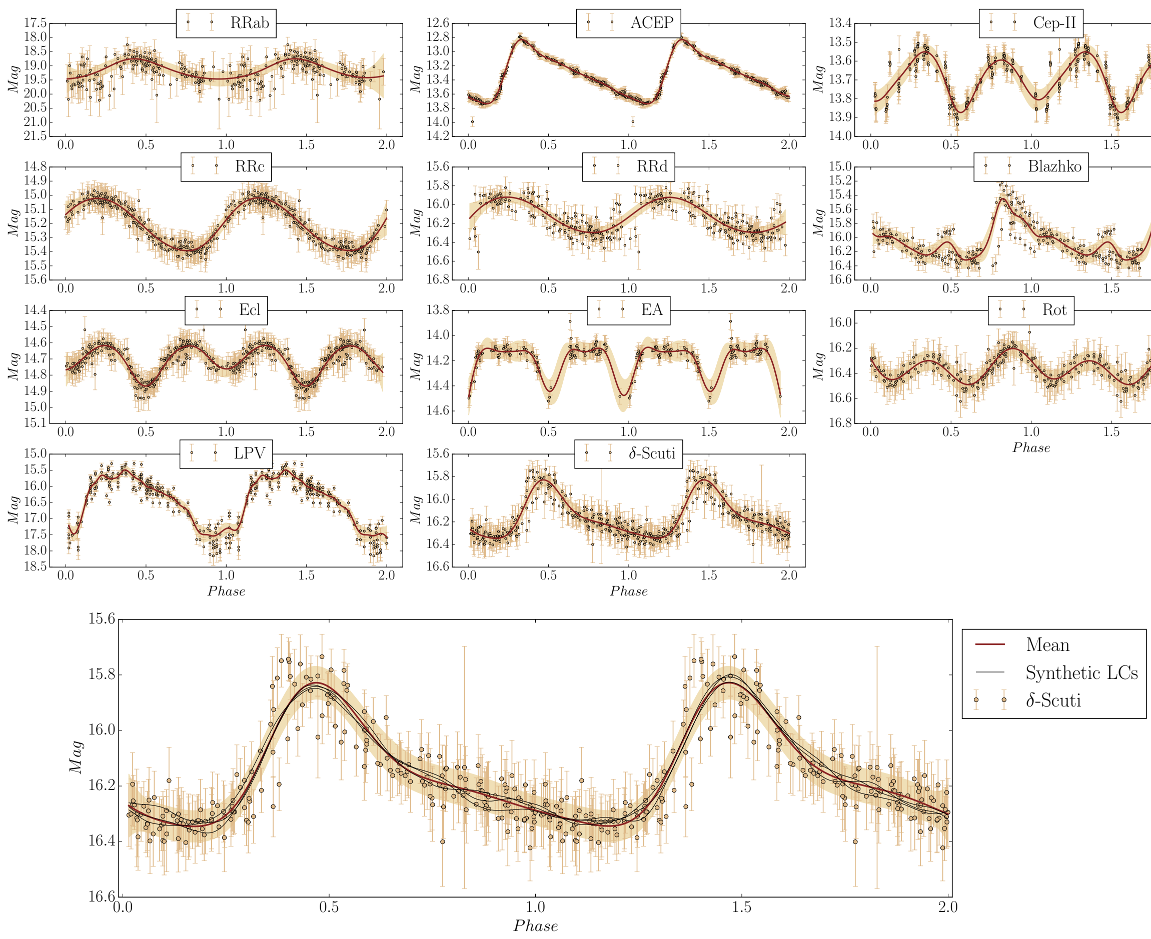


Figure 5.3 – Gaussian Processes offer a flexible approach to produce a smooth model of periodic light curves reported in magnitudes as a function of phase. This is demonstrated with model fits for each example of variable stars considered in the CRTS dataset. The data points are illustrated in black-rounded dots along with the error bars. The mean of the GP fit is shown in brown with three standard deviation away from the mean, shown in shaded pale brown. In the bottom panel, the black lines represent three randomly drawn samples from the GP fit. These randomly sampled light curves, also known as synthetic LCs together with real LCs, are used in the training set.

new light curves in the following way. Let us consider a probability distribution which can be concisely represented by a normal distribution. The probability distribution function (*pdf*) can be interpreted as going over the magnitude space vertically with the horizontal axis showing the probability that some value will occur. To construct the *pdf*, we make an assumption that the magnitude follows a normal distribution with mean, μ, to be equal to the true magnitude and the standard deviation, σ, to be equal to the error in magnitude. For each data point at a specific time, we sample a single magnitude from the *pdf*. Each sampled magnitude is assigned the same time as in the original data. Fig 5.2 shows an example of a light curve with the magnitude and error bars drawn for three specific times. The *pdf* of the magnitude is shown in blue and one magnitude is sampled randomly from the *pdf* shown in dotted red lines. The generated light curve is given the new (random) sampled magnitude with the same time value as in the original data.

5.4.3 Modelling Light Curves with Gaussian Process

An ideal case for data augmentation is to use a well defined model of the classes under consideration to create synthetic data. However, there is no available model valid for all the different variable stars considered. We therefore build a model describing variable stars using Gaussian Processes (GPs, Rasmussen 2003) applied to CRTS data. We then use this model to generate artificial light curves, allowing us to augment our training data through the addition of new examples in a principled way, using the distributions of existing data to create them.

A GP is a distribution over functions. It is defined by a mean $\mu(t)$ and a covariance (kernel) function $c(t, t')$, and is given as

$$\boxed{f(t) \sim \mathrm{GP}\left(\mu(t), c(t, t')\right).} \tag{5.4.1}$$

When the function f is computed at points t, the marginal distribution follows a multivariate normal distribution (Rasmussen, 2003). The kernel function, c, takes two inputs and shows the similarity between them. When evaluating Bayesian inference, having the set of known function values for the training sets $\boldsymbol{f_x}$, and the set of known function values for the test sets $\boldsymbol{f_y}$, are normally distributed and is given as follows:

$$\begin{bmatrix} \boldsymbol{f_x} \\ \boldsymbol{f_y} \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{f_x} \\ \boldsymbol{\mu}_{f_y} \end{bmatrix}, \begin{bmatrix} \boldsymbol{C}_{f_x f_x} & \boldsymbol{C}_{f_x f_y} \\ \boldsymbol{C}_{f_x f_y} & \boldsymbol{C}_{f_y f_y} \end{bmatrix} \right), \tag{5.4.2}$$

where the means of the training and test set are denoted by $\boldsymbol{\mu}_{f_x}$ and $\boldsymbol{\mu}_{f_y}$ respectively and likewise $\boldsymbol{C}_{f_x f_x}$, $\boldsymbol{C}_{f_y f_y}$, $\boldsymbol{C}_{f_x f_y}$ represent the training, test and train-test covariances/kernels. The conditional distribution, $\boldsymbol{f_x} \mid \boldsymbol{f_y} = \mathcal{P}$ is given by

$$\mathcal{P} \sim \mathcal{N}\left(\boldsymbol{C}_{f_x f_y} \boldsymbol{C}_{f_y f_y}^{-1} \left(\boldsymbol{f_y} - \boldsymbol{\mu}_y \right) + \boldsymbol{\mu}_{f_x}, \boldsymbol{C}_{f_x f_x} - \boldsymbol{C}_{f_x f_y} \boldsymbol{C}_{f_y f_y}^{-1} \boldsymbol{C}_{f_x f_y}^{\intercal} \right). \tag{5.4.3}$$

For a specific set of testing samples, Eq 5.4.3, represents the posterior distribution. For a set of training examples $\mathcal{D}$, the posterior distribution is described by (Rasmussen, 2003)

$$\begin{aligned} f_y \mid \mathcal{D} &\sim \mathrm{GP}(\mu_{\mathcal{D}}, c_{\mathcal{D}}), \\ \mu_{\mathcal{D}}(t) &= \mu(t) + \boldsymbol{c}_{T_s t}^{\intercal} \boldsymbol{C}^{-1} \left(f_y - \mu \right), \\ c_{\mathcal{D}}\left(t, t'\right) &= c\left(t, t'\right) - \boldsymbol{c}_{T_s t}^{\intercal} \boldsymbol{C}^{-1} \boldsymbol{c}_{T_s t'}^{\intercal}, \end{aligned} \tag{5.4.4}$$

where the covariance vector between every training sample, T_s and t is $\boldsymbol{c}_{T_s t} = c(T_s, t)$. The choice of the covariance function is established, based on the knowledge of the domain. In our case, we want to model light curves, so we require a kernel that can demonstrate both small fluctuations and smooth variations. Given the different characteristics of the various stars, an appropriate choice of the kernel in this work is the Matern 5/2 kernel given by,

$$\boxed{C_{\mathrm{Matern52}}(\Upsilon) = \left(1 + \frac{\sqrt{5}\Upsilon}{\ell} + \frac{5\Upsilon^2}{3\ell}\right) \exp\left(-\frac{\sqrt{5}\Upsilon}{\ell}\right),} \tag{5.4.5}$$

where Υ and ℓ are the kernel hyperparameters, that is, Υ controls the degree of smoothness and ℓ is the characteristic length scale. We employ the GP regression using `George` (Ambikasaran et al., 2015) with kernel hyper-parameters randomly initialised. Using our data and these randomly initialised hyper-parameters, the negative log likelihood is calculated. Afterwards, these hyper-parameters for the kernel are optimised (i.e., finding the best values for these parameters) using the Limited memory Broyden-Fletcher- Goldfarb-Shanno (L-BFGS, Fletcher 1987) optimization algorithm by minimizing the negative log likelihood.

The kernel with the optimized parameters is then used to fit the GP from which we sample synthetic light curves to augment our training set. Before fitting a GP to our data, we first convert the LCs from time distribution to phase distribution (folded-light curves) where the data is at the detected period for each variable star. We then randomly sampled synthetic LCs from the GP model to form the augmented training set. We show an example of `GpFit` on the folded-LCs for the different variable stars in Fig 5.3 and the bottom plot illustrates 3 synthetic LCs randomly drawn from `GpFit`. We then unfolded the phases back into time space and used those synthetic LCs together with the original LCs as the training set.

5.5 METHOD DESCRIPTION

Drawing heavily from H19, we outline the general approach used to classify variable stars. In this study, we use RF and XGBoost classifiers. We use these classifiers for two reasons. Firstly, to ensure that results presented here are comparable with previous work. Secondly, because they have proven to be robust against the issues associated with class imbalance (Chen et al. 2004; Wang et al. 2020). We then provide an overview of the HC scheme, together with the various stages we adopt to build the ML pipeline. Similar to H19, we pre-processed the lightcurves by applying a sigma-clipping method prior to any analysis.

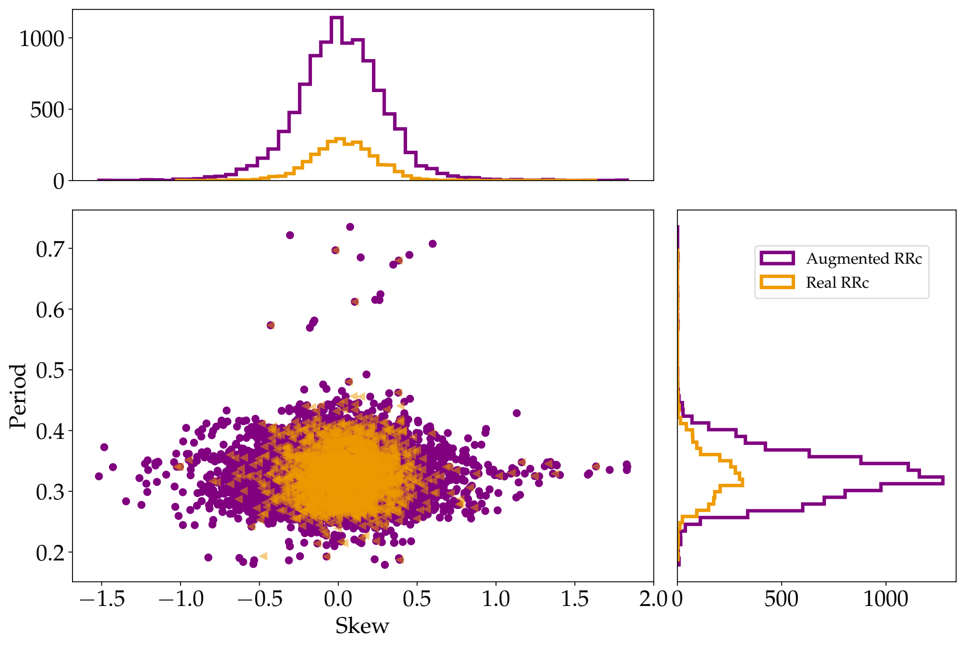


Figure 5.4 – Period versus Skew distribution for real and synthetic LCs generated using `GpFit`.

5.5.1 Stage 1: Hierarchical Tree Classifiers

H19's HC uses the astrophysical properties of the various sources to construct a tree-based structure to represent the different classes (Fig 5.1). Each node/leaf represents a class - identified by the label inside the node/leaf - and the edges represent the relationship between the super-class and sub-class. For the HC, we use XGBoost and RF and then report the one that provides the best classification performance. A detailed overview of XGBoost and Random Forest can be found in §3.3.3 and §3.3.2 respectively. In astronomy, XGBoost has recently been used by Mirabal et al. (2016) who implemented this classifier for unknown point source classification in the Fermi-LAT catalogue and for the separation of pulsar signals from noise (Bethapudi & Desai, 2018). In addition, XGBoost has also been applied for variable star classification (Sesar et al., 2017; Pashchenko et al., 2018; Kgoadi et al., 2019).

5.5.2 Stage 2: Level-wise Data Augmentation in HC

Since the training set is still imbalanced after aggregating sub-classes into super-classes, we use the three data augmentation techniques described in §5.4. Each technique is applied and tested independently in our HC based ML pipeline. For the `SMOTE` approach, features (the mean magnitude, standard deviation, skewness, kurtosis, mean-variance, amplitude and period) described in H19 are extracted from the real LCs. Then, `SMOTE` automatically balances the class distribution via the creation of synthetic examples sampled over the feature space, such that the size of the minority class equals the size of the majority class, cancelling the imbalance out.

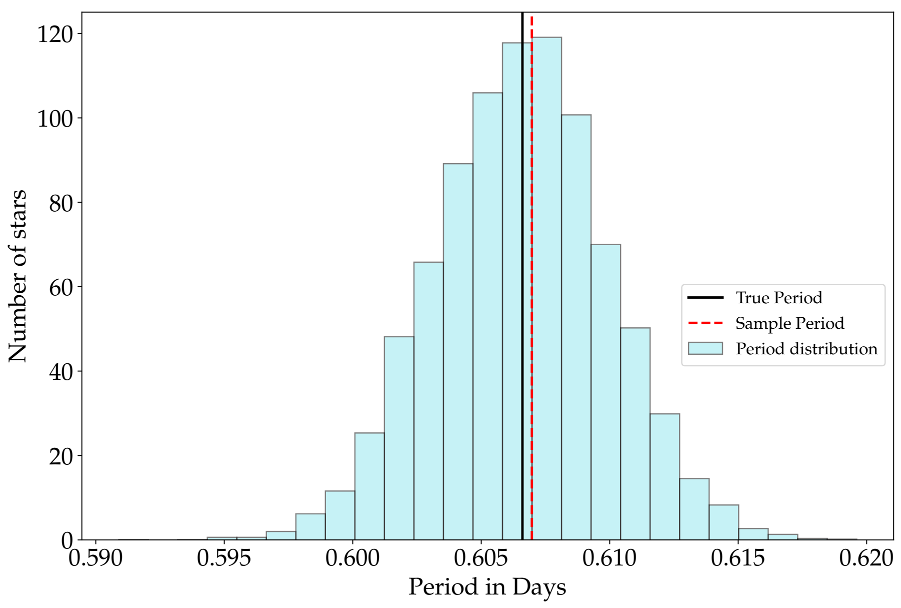


Figure 5.5 – For each synthetic LC, a period value (red vertical line) is randomly sampled from a normal distribution, with mean T being the true period of the real LC and σ_T being the computed uncertainty of the period, T.

For example, considering level 1 in Fig 5.1, the majority class is Pulsating, consisting of 7338 examples. Therefore, `SMOTE` adds new examples of the other two minority classes (eclipsing 6312 and rotational 2545) ensuring they both contain 7338 examples. This process is repeated for each branch and level in the HC, where the training set is directly balanced according to the size of the majority class prior to data augmentation.

While for the `GpFit` and `RASLE` cases, we are generating new light curves based on real LCs, thus generating new synthetic LC examples. Therefore, our training set will consist of both real and synthetic LCs, whilst we test our ML pipeline with only real LCs. These two techniques can be used to over-sample both the majority and minority class. The number of training examples after augmentation, *A.Tr* used for each level is given in Fig 5.1. Afterwards, features are extracted from these LCs as discussed below.

5.5.3 Stage 3: Feature Extraction

In this work, similarly to H19, our features are based on 6 intrinsic statistical properties relating to location (mean magnitude), scale (standard deviation), variability (mean variance), morphology (skew, kurtosis, amplitude), and time (period). These features are highly interpretable, and robust against bias (Hosenie et al., 2019a). For the `GpFit` and `RASLE` approach, the first six features are extracted directly from the augmented training set (containing both real and synthetic LCs) using the FATS library (Nun et al., 2015). Whilst for the period feature, the real LCs in the training set are assigned their respective period from the ascii-catalogue (Drake et al., 2017) and the synthetic LCs are assigned a period calculated by the method discussed in §5.5.3.1. For the test set we use only real LCs, hence the six features are extracted directly from the LCs and their period is obtained from the data catalogue. Therefore, we have 7 features that describe each variable star. Fig 5.4 shows the distribution of the two most important features as investigated in H19 (period and skew) for real and synthetic LCs. We observe that the synthetic LCs show similar characteristics compared to the real LCs.

5.5.3.1 Period for augmented LCs

A synthetic LC is given a period based on the uncertainty in the estimated period of the real LC. In this case, the estimated period, T, is obtained from Drake et al. (2017). The associated uncertainty, σ_T for a given period is calculated as follows. A periodic signal is detected in a periodogram by the presence of a peak with a certain width and height. In Fourier perspective, we assume that there is a direct relationship between the precision with which a peak's frequency can be detected and the width of this peak; often known as the half-width at half-maximum (VanderPlas, 2018) and is given by:

$$v_{\frac{1}{2}} \approx \frac{1}{\mathrm{T}} \tag{5.5.1}$$

This can be viewed as interpreting the periodogram with the least-square method, that is, the inverse of the curvature of the peak is determined with the uncertainty (Ishak, 2017). In the Bayesian perspective, this translates to a Gaussian curve fit to the exponentiated peak (Smith & Erickson, 2012; Bretthorst, 2013). Let us consider a periodogram with maximum value $\mathrm{P}_{max} = \mathrm{P}\left(v_{max}\right)$, such that

$$\mathrm{P}\left(v_{max} \pm v_{\frac{1}{2}}\right) = \frac{\mathrm{P}_{max}}{2}. \tag{5.5.2}$$

Hence, the Bayesian uncertainty is calculated by approximating the exponentiated peak as a Gaussian, that is,

$$\exp\left[\mathrm{P}\left(v_{max} \pm \delta v\right)\right] \propto \exp\left[\frac{-\delta v^2}{(2\sigma_v^2)}\right]. \tag{5.5.3}$$

The above equation can then be written as follows and we obtain the uncertainty in frequency in Eq 5.5.4.

$$\frac{\mathrm{P}_{max}}{2} \approx \mathrm{P}_{max} - \frac{v_{\frac{1}{2}}^2}{(2\sigma_v^2)}; \qquad \frac{v_{\frac{1}{2}}^2}{2\sigma_v^2} \approx \frac{\mathrm{P}_{max}}{2};$$

$$\sigma_v \approx \frac{v_{\frac{1}{2}}}{\sqrt{\mathrm{P}_{max}}}, \tag{5.5.4}$$

where $\delta v \approx v_{\frac{1}{2}}$. Considering the signal-to-noise ratio $\varphi = \text{rms}\left[\frac{y_n - \mu}{\sigma_n}\right]$, where μ is the mean magnitude, y_n and σ_n is the magnitude and error in magnitude for each data point respectively. We can then write the following equation for a well-fitted model,

$$\mathrm{P}_{max} \approx \frac{\varphi^2 N}{2}. \tag{5.5.5}$$

We then substitute Eq. 5.5.5 in Eq. 5.5.4 and the uncertainty in frequency can be written as:

$$\sigma_v \approx v_{\frac{1}{2}} \sqrt{\frac{2}{\varphi^2 N}}, \tag{5.5.6}$$

where $v_{\frac{1}{2}} \approx \frac{1}{\mathrm{T}}$, N is the number of data points and φ is the signal to noise. We now compute the uncertainty in period by taking the derivative of σ_v,

$$\frac{\partial v}{\partial \mathrm{T}} \approx -\frac{1}{\mathrm{T}^2}; \qquad \partial \mathrm{T} = -\mathrm{T}^2 \sigma_v; \qquad \sigma_{\mathrm{T}}^2 = \mathrm{T}^4 \sigma_v^2.$$

Hence, the uncertainty in period is then obtained using Eq 5.5.7.

$$\boxed{\sigma_{\mathrm{T}} = \mathrm{T}^2 \sigma_v} \tag{5.5.7}$$

where σ_{T} will be Gaussian if σ_v is very small. A period value is given to each synthetic LC (generated either with `GpFit` or `RASLE`), by randomly sampling from a normal distribution with mean, T (the true period of the real LC from which the synthetic LCs are generated) and within 1 σ-confidence interval, being σ_{T} using Eq. 5.5.7. An example of associating a period to an augmented LC is shown in Fig 5.5.

5.5.4 Stage 4: Training with Bayesian Optimization

We first randomly split our data into training (70 per cent) and testing sets (30 per cent). The training set moves through the first level in the HC scheme discussed in §5.5.1. The training examples are then augmented using one of the three data augmentation techniques and features are extracted where appropriate. Afterwards, the model (see dotted square at level 1 in

Fig 5.1) is trained using either the RF or XGBoost classifier, as required. We then use a Bayesian Optimization approach to find the best hyper-parameters for the ML algorithm. It has been demonstrated for large parameter spaces that Bayesian Optimization, also known as Sequential Model-Based Optimization (SMBO, Hutter et al. 2011) performs better than either manual or randomized grid searches (Bergstra et al., 2015). It is one of the most efficient techniques for hyper-parameter optimization of ML algorithms.

In this work, we used SMBO techniques compared to H19, who used a randomized grid-search for hyper-parameter optimization. Before applying the above methods, we perform a stratified cross validation. The training data is split into 5 folds, where 4 different folds are kept for training each time and an independent fold is used for validation. We then use the SMBO method, `HyperOpt` (Bergstra et al., 2015) to find the best hyper-parameters on the 4 folds and validated the model on the independent fold. We then evaluate our trained model based on balanced-accuracy, G-mean, precision, recall, and F1-scores, on real LCs in the test set.

5.6 Analysis and Results

This work is mostly concerned with learning from an imbalanced class distribution. The problem is typically addressed using the following approaches.

1. *Data level*: We employ three approaches to the HC scheme in such a way that the class distributions are rebalanced directly; that is, it is a first proof of principle application of a level-wise augmentation in Hierarchical taxonomy, where we resample the original dataset to achieve a desired balancing.

2. *Algorithm level*: We focus on using two different algorithms (RF and XGBoost), together with a Bayesian Optimization algorithm for hyper-parameter tuning, to achieve improved performance on the minority class examples.

The HC algorithm is trained on both real and artificially augmented data and tested on real data. We show the results of the three data augmentation techniques in Table 5.6.1. We assess the consistency of the results based on balanced-accuracy and G-mean scores. The shaded blue color represents the augmentation methods, which when applied together with the HC classifier, yielded improved results over H19. We found that `GpFit` achieves the best performance measures compared to H19 at all levels in the HC. When using the `GpFit` method, we found that our RF implementation performs best at all HC levels when compared to H19 and we highlight this result in grey. In addition, we found that XGBoost, similarly to the RF, provides good

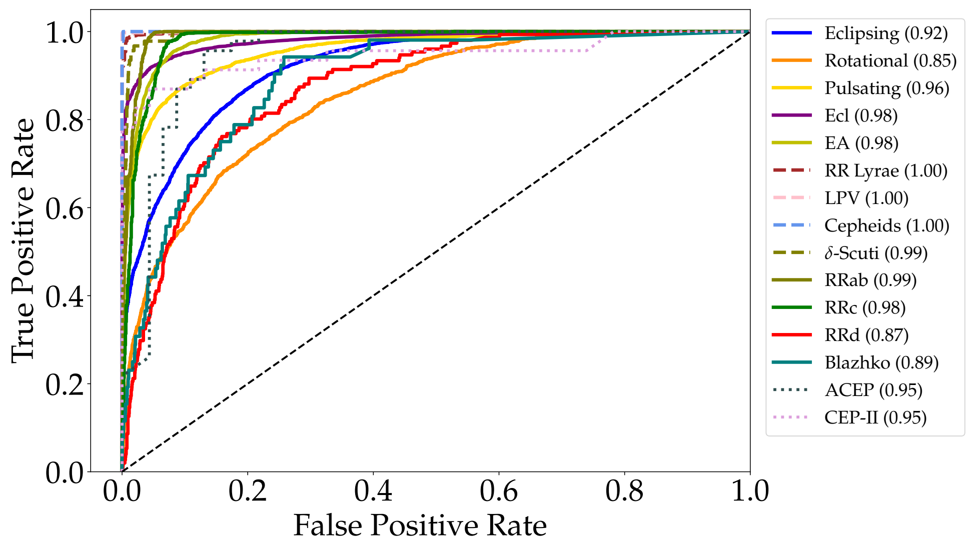


Figure 5.6 – Receiver operating characteristic (ROC) curves for each node in the hierarchical model. Each curve represents a different variable star class with the area under the ROC curve (AUC) score in brackets. This metric is computed on the 30% of the dataset used for testing.

performance for variable star classification. Moreover, in H19, we show that the HC model is neither underfitting nor overfitting by plotting precision-recall curves at different levels. In this work, we assess the consistency of the results using `GpFit` and RF by plotting the Receiver Operator Characteristic (ROC) curve for each class (see Fig 5.6). We note that classification performance is very good. The area under the ROC curve (AUC) values are greater than 0.95 for several classes, except for Rotational, RRd, and Blazhko. The reasons for these misclassification are further discussed in §5.6.1.

We improve upon the result obtained in H19. For instance, the balanced-accuracy increases from 61 to 65 per cent in level 1, from 86 to 88 per cent at level 2 for the eclipsing node, from 86 to 87 per cent for sub-classes of RR Lyrae at level 3, and finally from 81 to 83 per cent for Cepheids at level 3. To check the consistency and robustness of our new approach, we perform an additional step. We use different splits ($K = 5, 6, \ldots, 10$) during cross-validation and predict on the 30% test set. With these analyses, we obtain an uncertainty on the metric scores considered, for example for Cepheids at level 3, a 0.83 ± 0.02 balanced-accuracy and 0.91 ± 0.01 G-mean score are obtained. We obtain similar results at different levels in the hierarchy. In these analyses, we observe that we have not made a huge improvement to H19, in terms of minority classes and we explain the various reasons that might lead to this outcome in §5.6.1.

5.6.1 Impact of imbalance on classification performance

Training a classifier upon imbalanced data does not guarantee poor generalisation performance (Galar et al., 2011). Regardless of imbalance, if the features or the training data themselves are discriminative enough to provide a clear separation between the different classes, then classifiers will likely generalize well. However, there are three main characteristics of imbalanced data sets that make it hard for a classifier to discriminate the minority from the majority classes. These are

1. small sample sizes (Galar et al., 2011; He & Garcia, 2008),
2. class inseparability (Galar et al., 2011; Japkowicz & Stephen, 2002) (see Fig 5.7(a) & 5.8) and,
3. small disjuncts (see Fig 5.7(b)).

Ultimately, the training data showing these characteristics conspire to make it hard for any classifier to build an optimal decision boundary leading to sub-optimal classifier performance. These characteristics are seen at some levels in the HC. In this section, we illustrate these effects at level 3 using the sub-classes of RR Lyrae. Fig 5.7(a) shows that some classes have overlapping characteristics, which leads to poor performance. We observe similar characteristics (class-overlapping) for the sub-classes of RR Lyrae in Fig 5.8(a), even after balancing the classes in the training set. These overlapping characteristics are due to the fact that there are no physical distinction between some of the subclasses. As can be seen in Fig 5.8(a), RRab and RRc classes can reasonably be separated based on their period alone. RRab are variable stars pulsating in fundamental mode, RRc stars pulsate in the first overtone while RRd stars simultaneously pulsate in the fundamental and first overtone. Therefore, RRd's form part of both RRab and RRc variable stars at the same time. In addition, Blazhko stars are found among RRab stars (Jurcsik et al., 2009), RRc stars (Netzel et al., 2018) and even RRd stars (Jurcsik et al., 2015). This explains the poor performance of the classifier for separating RRd and Blazhko stars, even after balancing the classes. In addition, we also present a t-distributed stochastic neighbour embedding (t-SNE, van der Maaten & Hinton 2008) of the minority classes (Blazhko, δ-Scuti, ACEP & Cep-II) in Fig 5.8(b) after augmenting them using the `GpFit` method. The result shown in Fig 5.8(a) does not differ when we perform multiple runs with different parameters. Each time we find small disjuncts in the feature space, showing characteristics similar to those shown in Fig 5.7(b), thus making it difficult for the classifier to construct a decision boundary.

In this work, we found that training the HC with class-balanced data has the effect of improving balanced-accuracy and G-mean scores. However, the minority classes are still misclassified. Although these results suggest that balancing the class distribution is not sufficient for classifying the minority classes, their capacity to prevent overfitting and increase the recall rate makes them appealing.

Another reason that leads to misclassification - the lack of a standard set of correctly classified (i.e. where the ground truth is certain) variable star example useful for training. Drake et al. (2017) investigated the level of agreement of their classifications with the International Variable Star Index (VSX, Watson et al. (2007)). They found that

1. VSX has not classified any of their Blazhko stars, but instead simply classify them as RRab stars,
2. VSX classified many of their contact binaries as detached and semi-detached binaries,
3. most of their Rotational stars (spotted or ellipsoidal variables) have been classified as contact binaries, and
4. most of their RRd stars have been misclassified as other stars (RRab, RRc) by VSX.

We observe similar misclassifications when using our automated HC pipeline, even after balancing the classes. With the presence of so many misclassified objects, we can plausibly say neither Drake et al. (2017) or VSX can be considered as providing ground truth. Therefore, there is a real need to have a standard set of correctly identified variable stars that can be utilized for training automated machine learning methods. It is imperative to train these sophisticated ML based algorithms with accurately calibrated priors in order to obtain reliable classification outputs.

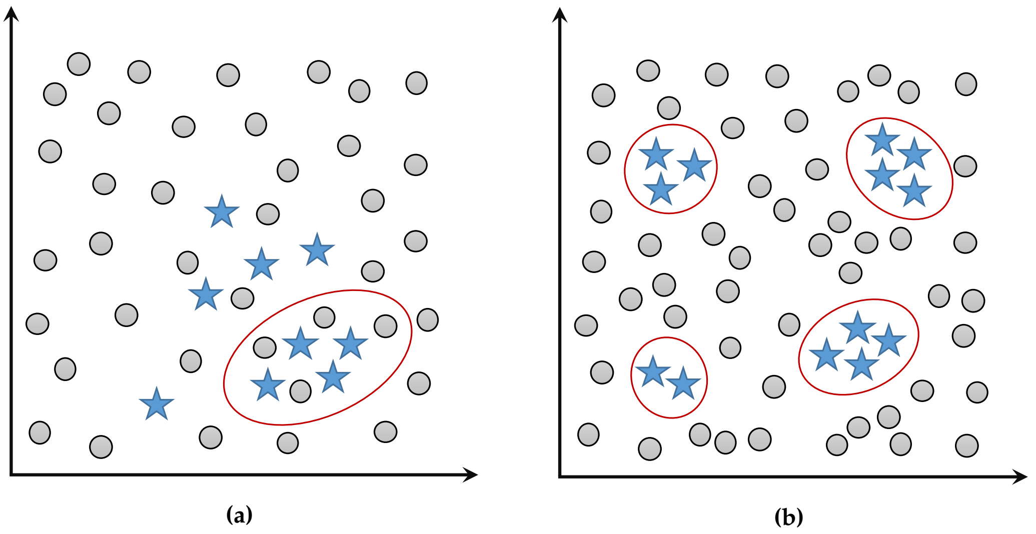


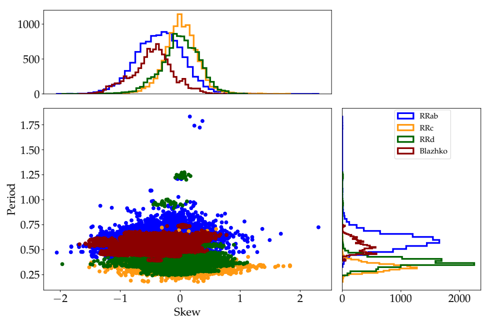
Figure 5.7 – Demonstration of (a) Class inseparability and (b) small disjuncts in feature space.

Table 5.6.1 – Evaluation metrics used to summarize the HC pipeline with the application of three methods of data augmentation. We present the balanced-accuracy and G-mean scores level-wise to evaluate our model. H19 results are presented in bold text in the table. It is seen that the HC pipeline performs fairly well with data augmentation, achieving G-mean scores above ∼ 80% at all levels. The shaded blue represents the augmentation methods that outperform H19. We observe that at all levels, `GpFit` together with RF, performs better than H19 and it is represented in shaded grey. The '∼' represents a single value for the computed average metrics by taking into consideration the overall classes and '*No aug*' means no augmentation is applied on the data.

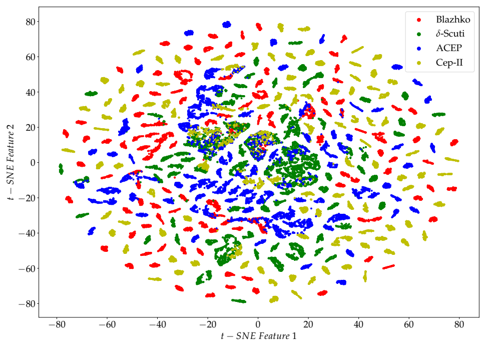
Augmentation Methods	**Classifiers**	**G-Mean**	**Balanced-accuracy**
First Level: Eclipsing, Rotational and Pulsating Classification			
H19 (No aug)	**RF**	**0.78/0.78/0.86 (∼ 0.79)**	**0.59/0.60/0.75 (∼ 0.61)**
SMOTE	XGBoost	0.80/0.77/0.89 (∼ 0.81)	0.63/0.59/0.80 (∼ 0.65)
	RF	0.80/0.78/0.89 (∼ 0.81)	0.63/0.60/0.79 (∼ 0.65)
RASLE	XGBoost	0.82/0.76/0.89 (∼ 0.83)	0.66/0.57/0.79 (∼ 0.68)
	RF	0.82/0.77/0.89 (∼ 0.83)	0.66/0.58/0.79 (∼ 0.68)
GpFit	XGBoost	0.80/0.75/0.89 (∼ 0.81)	0.63/0.56/0.79 (∼ 0.65)
	RF	0.80/0.75/0.89 (∼ 0.81)	0.63/0.56/0.78 (∼ 0.65)
Second Level: RR Lyrae, LPV, Cepheids and δ-Scuti			
H19 (No aug)	**RF**	**0.99/1.00/0.97/1.00 (∼ 0.99)**	**0.98/0.99/0.93/1.00 (∼ 0.98)**
SMOTE	XGBoost	0.99/1.00/1.00/0.95 (∼ 0.99)	0.97/0.99/1.00/0.90 (∼ 0.97)
	RF	0.99/1.00/1.00/0.96 (∼ 0.99)	0.97/0.99/1.00/0.92 (∼ 0.97)
RASLE	XGBoost	0.99/1.00/0.99/0.93 (∼ 0.99)	0.98/1.00/0.98/0.85 (∼ 0.98)
	RF	0.99/1.00/1.00/0.94 (∼ 0.99)	0.98/0.98/1.00/0.88 (∼ 0.98)
GpFit	XGBoost	0.99/1.00/0.99/0.95 (∼ 0.99)	0.97/0.99/0.97/0.99 (∼ 0.98)
	RF	0.99/1.00/1.00/0.97 (∼ 0.99)	0.97/0.99/1.00/0.93 (∼ 0.98)
Second Level: Ecl and EA			
H19 (No aug)	**RF**	**0.93/0.93 (∼ 0.93)**	**0.86/0.86 (∼ 0.86)**
SMOTE	XGBoost	0.94/0.94 (∼ 0.94)	0.88/0.88 (∼ 0.88)
	RF	0.94/0.94 (∼ 0.94)	0.88/0.88 (∼ 0.88)
RASLE	XGBoost	0.93/0.93 (∼ 0.93)	0.85/0.85 (∼ 0.85)
	RF	0.93/0.93 (∼ 0.93)	0.85/0.86 (∼ 0.86)
GpFit	XGBoost	0.93/0.93 (∼ 0.93)	0.88/0.88 (∼ 0.88)
	RF	0.94/0.94 (∼ 0.94)	0.87/0.88 (∼ 0.88)
Third Level: RRab, RRc, RRd and Blazhko			
H19 (No aug)	**RF**	**0.97/0.92/0.65/0.44 (∼ 0.92)**	**0.94/0.85/0.40/0.18 (∼ 0.86)**
SMOTE	XGBoost	0.95/0.92/0.67/0.58 (∼ 0.91)	0.91/0.83/0.42/0.31 (∼ 0.83)
	RF	0.95/0.82/0.47/0.33 (∼ 0.91)	0.91/0.82/0.47/0.33 (∼ 0.83)
RASLE	XGBoost	0.96/0.95/0.56/0.53 (∼ 0.92)	0.93/0.89/0.30/0.26 (∼ 0.87)
	RF	0.97/0.95/0.52/0.52 (∼ 0.92)	0.94/0.90/0.25/0.25 (∼ 0.87)
GpFit	XGBoost	0.97/0.93/0.57/0.44 (∼ 0.92)	0.94/0.86/0.30/0.17 (∼ 0.85)
	RF	0.97/0.93/0.56/0.41 (∼ 0.92)	0.94/0.87/0.32/0.26 (∼ 0.87)
Third Level: ACEP and Cep-II			
H19 (No aug)	**RF**	**0.90/0.90 (∼ 0.90)**	**0.82/0.80 (∼ 0.81)**
SMOTE	XGBoost	0.88/0.88 (∼ 0.88)	0.78/0.76 (∼ 0.77)
	RF	0.88/0.88 (∼ 0.88)	0.78/0.76 (∼ 0.77)
RASLE	XGBoost	0.88/0.88 (∼ 0.88)	0.77/0.78 (∼ 0.77)
	RF	0.88/0.88 (∼ 0.88)	0.77/0.78 (∼ 0.78)
GpFit	XGBoost	0.88/0.88 (∼ 0.88)	0.78/0.78 (∼ 0.78)
	RF	0.91/0.91 (∼ 0.91)	0.84/0.82 (∼ 0.83)

5.7 Conclusion

In this work, we present a new approach for tackling the problem of imbalanced data: a level-wise data augmentation in a hierarchical classification framework. Through an empirical investigation, we demonstrate three techniques for augmenting data, that is, `SMOTE`, `RASLE` and `GpFit` are applied to variable star data. We show that using RF and `GpFit` together can effectively improve recall rates, hence increasing the balanced-accuracy and G-mean scores by 1-4 per cent. Although, the results show that even after balancing the training set level-wise, such approaches do not prevent the misclassification of the minority class, though their capacity to increase other metrics (e.g. recall) still makes their application appealing. Perhaps, the misclassification occurs because these objects are just not easily separable and we observe similar misclassifications in this work as determined by Drake et al. (2017) when they compared their results with VSX. Therefore, it is imperative to have correctly labelled data that can accurately be used to train automated ML pipeline in order to output reliable classification performance.



(a)



(b)

Figure 5.8 – (a) shows the Period-Skew distribution of RRab, RRc, RRd and Blazhko after augmenting each respective class to 10,000 examples. We note that the classes are still overlapping in the feature space, even after the augmentation process. (b) illustrates small disjoints in feature space using t-distributed stochastic neighbour embedding (t-SNE) visualization in the small sample size data (Blazhko, δ-Scuti, ACEP and Cep-II), after augmentation. No distinct separation is seen within the feature space.

6

Real-Bogus Classification using Deep Learning for the MeerLICHT facility

This chapter was originally published as a paper in Experimental Astronomy and NeurIPS conference 2021 with the title: '`MeerCRAB`**: Real-Bogus Classification using Deep Learning for the MeerLICHT facility**'.

Zafiirah Hosenie, *Steven Bloemen*, *Paul Groot*, *Robert Lyon*, *Bart Scheers*, *Benjamin Stappers*, *Fiorenzo Stoppa*, *Paul Vreeswijk*, *Simon De Wet*, *Marc Klein-Wolt*, *Elmar Kording*, *Vanessa McBride*, *Rudolf Le Poole*, *Kerry Paterson*, *Danielle L. A. Pieterse* and *Patrick Woudt*

Accepted in Experimental Astronomy on **22 April 2021** and can be found at arXiv:2104.13950v1.

Summary of contributions: The work developed in this chapter is entirely my own. Co-authors provided supervision, suggestions of additional angles to discuss in the paper, and feedback on earlier drafts of it. In this work, we present two labelling strategies to label our data (i) *thresholding* which removes noisy labelling and (ii) the *Latent class model*, L_{lcm} (Formann, 1984) which incorporates labelling uncertainty in our model. Afterwards, we constructed three models based on CNNs to build a new robust system that separates real candidates from their bogus counterparts for the MeerLICHT-transient search pipeline. In §6.3 we provide an overview of the MeerLICHT telescope and we detail the data used for training and testing the `MeerCRAB` algorithms. In §6.4, the methods, network set-up and architectures are described. Results and experimental set-up are detailed in §6.5, followed by our main conclusions in §6.6.

6.1 Overview

Astronomers require efficient automated detection and classification pipelines when conducting large-scale surveys of the (optical) sky for variable and transient sources. Such pipelines are fundamentally important, as they permit rapid follow-up and analysis of those detections most likely to be of scientific value. We therefore present a deep learning pipeline based on the convolutional neural network architecture called `MeerCRAB`. It is designed to filter out the so called "bogus" detections from true astrophysical sources in the transient detection pipeline of the MeerLICHT telescope. Optical candidates are described using a variety of 2D images and numerical features extracted from those images. The relationship between the input images and the target classes is unclear, since the ground truth is poorly defined and often the subject of debate. This makes it difficult to determine which source of information should be used to train a classification algorithm. We therefore used two methods for labelling our data (i) thresholding and (ii) latent class model approaches. We deployed variants of `MeerCRAB` that employed different network architectures trained using different combinations of input images and training set choices, based on classification labels provided by volunteers. The deepest network worked best with an accuracy of 99.5% and Matthews correlation coefficient (MCC) value of 0.989. The best model was integrated to the MeerLICHT transient vetting pipeline, enabling the accurate and efficient classification of detected transients that allows researchers to select the most promising candidates for their research goals.

6.2 Introduction

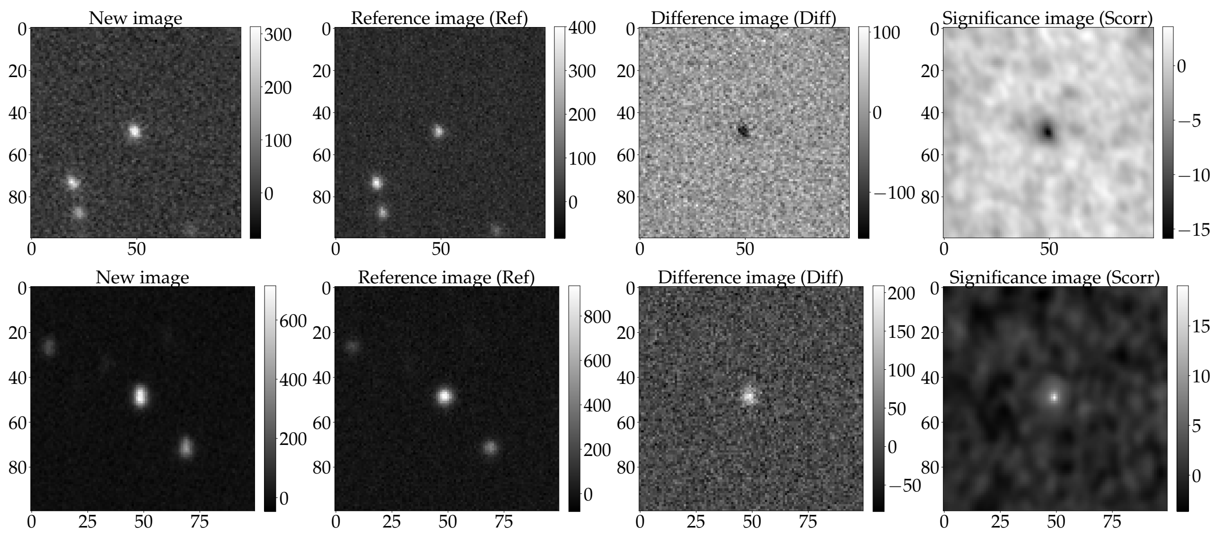
Contemporary large-scale optical surveys such as Skymapper (Keller et al., 2007), the Palomar Transient Factory (PTF, Rau et al. 2009), the Catalina Real-time Transient Survey (CRTS, Drake et al. 2009), the Panoramic Survey Telescope and Rapid Response System (Pan-STARRS1, Kaiser et al. 2010), the All-Sky Automated Survey for SuperNova (ASASSN, Shappee et al. 2014), Gaia (Gaia Collaboration et al., 2016), the MeerLICHT telescope (Bloemen et al., 2016; Paterson, 2019) and the Zwicky Transient Factory (ZTF, Bellm et al. 2019) are generating a plethora of transient events originating from a wide range of sources. These instruments enable us to observe and explore changes in millions of sources/candidates, thus unlocking new opportunities for interpreting and understanding large families of sources.

The MeerLICHT facility provides a 2.7 square degree field-of-view at a pixel scale of 0.56″/pixel (Bloemen et al., 2016) that maximises the volume of astrophysical candidates with brightnesses appropriate for spectroscopic follow-up using current large-aperture optical facilities. More details regarding the survey can be found in Bloemen et al. (2016). Both MeerLICHT and the BlackGEM array (Groot, 2019) (that is currently being installed at the La Silla Observatory in Chile) will yield about 500 observations per night, per telescope, thus generating hundreds of candidate alerts every clear night that could be spectroscopically followed up. BlackGEM's main focus is on the detection of optical counterparts to gravitational wave events and MeerLICHT is used to co-observe the sky as seen with the MeerKAT radio array (Jonas & MeerKAT Team, 2016). MeerLICHT and BlackGEM are technically identical with MeerLICHT being the prototype for the BlackGEM array.

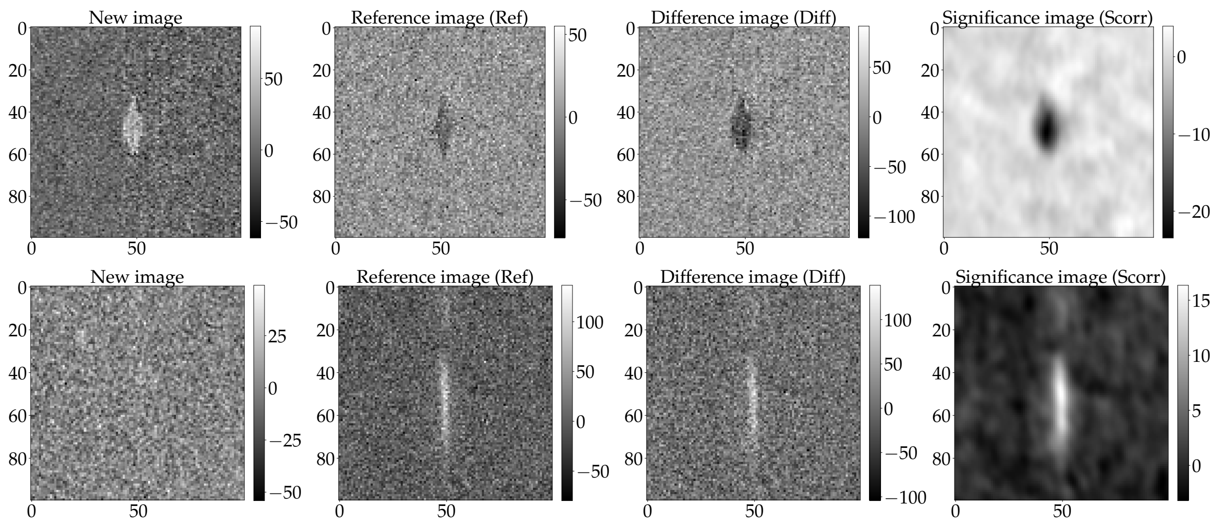
Transients and variables are sources that vary on all timescales (from milliseconds up to years) and they vary significantly from a reference image - either an increase or decrease in brightness. Transients include phenomena such as supernovae, gamma-ray bursts, tidal disruption events and flare stars, to name a few. A successful transient follow-up program enables the creation of a large database of transient and variable sources. Such large databases are important for future analyses of data collected during upcoming photometric surveys such as those conducted at the Vera C. Rubin observatory (LSST; LSST Science Collaboration et al. 2009). While we possess a reasonable understanding of many transient sources, achieved via consideration of their spectra, the main goal of surveys undertaken with MeerLICHT is to find and select the subset of sources that are not well understood. This will help us to increase our knowledge of the different families of transients and variable stars. Secondly, given that transients are rapidly fading sources due to their often destructive nature, MeerLICHT aims to identify transients rapidly, as they are only visible for a limited amount of time for follow-up.

In order to have an early and rapid characterisation of these sources, it is fundamentally important to automate several steps within a transient detection pipeline, including the separation of transients/astrophysical events from "bogus" detections, which has become a bottle-neck in fast detection pipelines. So called "bogus" detections can occur as a result of saturated sources, convolution problems, defects in the detector, atmospheric dispersion, unmodelled differences at the subtraction stage and cosmic rays passing through the detector, amongst other things.

Most surveys use three images for transient event detection and extraction: (i) an early observation of the relevant sky (also known as the *template/reference* image), (ii) a calibrated recent image (*New/Science* image), (iii) the *difference* image which is formed by subtracting the *reference* image from the *new/science* image. Using the difference image, one can, in principle, effectively detect transients, however, in many cases, the subtracted image contains bogus sources.



(a) Real examples.

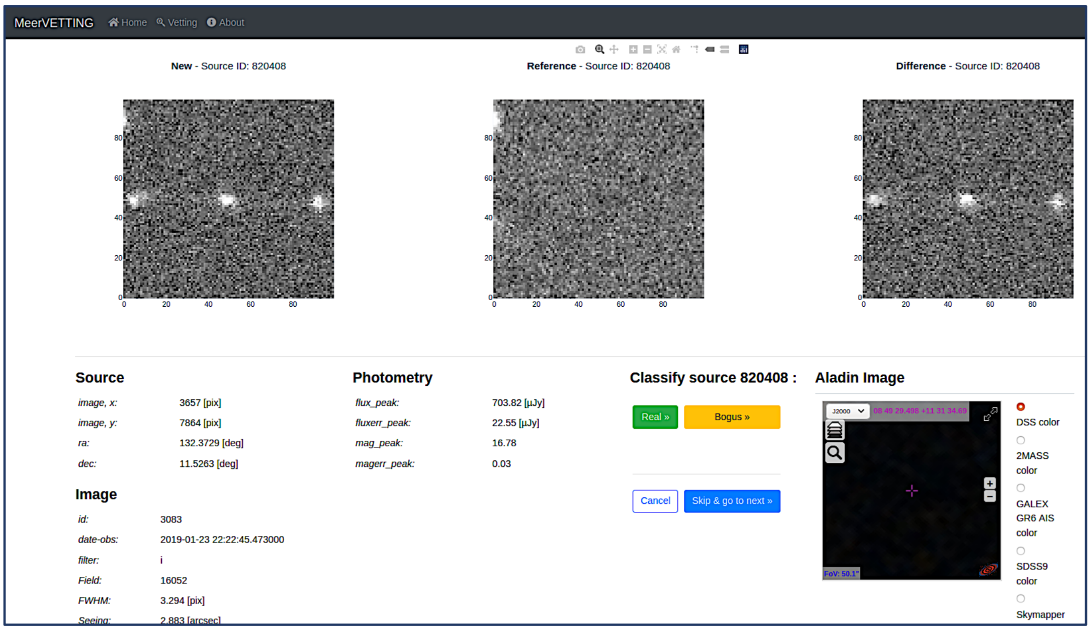


(b) Bogus examples.

Figure 6.1 – Examples of bogus and real transients in the MeerLICHT database. Each column represents the new (N), reference (R), difference (D) and significance (S) images and the rows are the different fields.

Therefore to be successful, surveys require an automated way to distinguish between real and bogus candidates. To address this challenging task, most of the time-domain surveys mentioned previously have adopted machine learning (ML) algorithms to perform real-bogus classification. Convolutional neural networks (CNNs, Lecun et al. 1999) have been used in the image domain as feature extractors for automatic vetting algorithms, for example, during the Skymapper Survey (Gieseke et al., 2017), the High cadence Transient Survey (HiTS, Cabrera-Vives et al. 2017) and the ZTF (Bellm et al., 2019) similarly utilized deep learning techniques. Other ML techniques such as Random Forest (RF) and *k*-Nearest Neighbour (*k*-NN) classification approaches have been employed to classify light curve transients from CRTS (Richards et al., 2011b; Hosenie et al., 2020, 2019b).

The classification task in these surveys is usually separated into two distinct steps. Firstly,



Source

image, x:	3657 [pix]
image, y:	7864 [pix]
ra:	132.3729 [deg]
dec:	11.5263 [deg]

Photometry

flux_peak:	703.82 [µJy]
fluxerr_peak:	22.55 [µJy]
mag_peak:	16.78
magerr_peak:	0.03

Image

id:	3083
date-obs:	2019-01-23 22:22:45.473000
filter:	i
Field:	16052
FWHM:	3.294 [pix]
Seeing:	2.883 [arcsec]

Figure 6.2 – `MeerVETTING` web-interface used to label MeerLICHT candidates as either real, bogus or skip confused candidates. Vetters are provided with three images (new, reference, difference) that are cut-outs of the science images.

bogus candidates are filtered out from real sources immediately after acquiring data, that is, the classification between real and bogus. The second stage involves assigning an astrophysical category/class label to each detected transient based on its spectroscopic or photometric information (e.g. Muthukrishna et al. 2019). In this thesis, we focus on the automation of the first stage, that is, the classification of sources as either Real or Bogus using deep learning methods developed for the MeerLICHT facility.

We note that when using ML based automated classification systems, we should not use models trained on data acquired at one telescope, to make predictions upon data acquired by another. Doing so constitutes a violation of the i.i.d principle, which ultimately limits classification performance and consistency. In addition, labelling mistakes are often even more costly - especially on rare sub-classes of transient phenomena. The performance of an ML system is thus entirely dependent upon the quality and distributional properties of the input data and the associated labels it is given. For a system to perform well for a given task, it must be built using data and labels that are distributionally similar to the data it must process in practice.

6.3 The MeerLICHT facility

MeerLICHT is an optical wide-field telescope that is operated robotically. The telescope is located at the Sutherland station of the South African Astronomical Observatory (SAAO). It consists of a 65 cm primary mirror and provides a 2.7 square degree field-of-view at a pixel

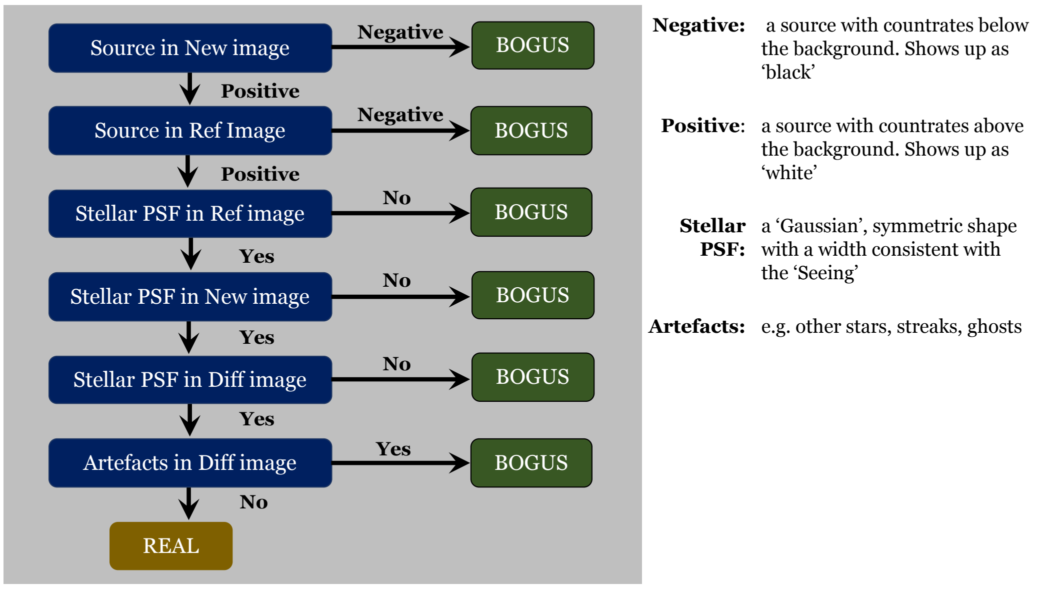

Source in New image
Negative
BOGUS
Positive
Source in Ref Image
Negative
BOGUS
Positive
Stellar PSF in Ref image
No
BOGUS
Yes
Stellar PSF in New image
No
BOGUS
Yes
Stellar PSF in Diff image
No
BOGUS
Yes
Artefacts in Diff image
Yes
BOGUS
No
REAL
Negative: a source with countrates below the background. Shows up as 'black'
Positive: a source with countrates above the background. Shows up as 'white'
Stellar PSF: a 'Gaussian', symmetric shape with a width consistent with the 'Seeing'
Artefacts: e.g. other stars, streaks, ghosts

Figure 6.3 – Decision tree characterising real and bogus candidates. Vetters used this guide to label each candidate and to construct a large training set for `MeerCRAB`.

scale of 0.56"/pixel (Bloemen et al., 2016). MeerLICHT will co-observe with the MeerKAT radio telescope on the same field. The combination of an optical and a radio telescope will enable the study of fast transient phenomena using simultaneous observations in two very distinct parts of the electromagnetic spectrum, whilst eliminating the delay introduced by triggering optical follow-up after the detection of a radio event.

MeerLICHT (and also BlackGEM) images are processed by the BlackBOX package[*] to produce image products and catalogues of all objects detected as well as transient candidates resulting from optimal image subtraction. The raw MeerLICHT images are automatically transferred from SAAO to the Inter-university Institute for Data Intensive Astronomy (IDIA[†]) in Cape Town, South Africa, and processed by BlackBOX.

First, the images are gain and overscan-corrected, and flat-fielded using a set of twilight flats. Cosmic rays and satellite trails are detected using the astroscrappy[‡] implementation of LA Cosmic (van Dokkum, 2001) and STSDAS satdet[§] modules, respectively. Subsequently, the following steps are performed: object detection using Source-Extractor (Bertin & Arnouts, 1996), astrometric calibration using Astrometry.net (Lang et al., 2010), estimation of the Point Spread Function (PSF) as a function of position using PSFEx (Bertin, 2011) and photometric calibration. The latter is done using a custom-built catalogue of calibration stars in the MeerLICHT photometric system based on fitting stellar spectral templates to Gaia, SDSS, PanS-

[*]see https://github.com/pmvreeswijk/BlackBOX and https://github.com/pmvreeswijk/ZOGY
[†]see https://www.idia.ac.za/
[‡]see https://github.com/astropy/astroscrappy
[§]see https://acstools.readthedocs.io/en/latest/satdet.html

TARRS, SkyMapper, 2MASS and GALEX photometry.

Finally, optimal image subtraction is performed, comparing the new image with a reference image, closely following the prescriptions of Zackay, Ofek & Gal-Yam, a.k.a. ZOGY (Zackay et al., 2016). To allow for the PSF to vary across the image, the full MeerLICHT images are divided into 8 by 8 sub-images, on which the ZOGY calculations are applied separately, before inserting the sub-images back into a full image. The following images are produced: a difference (D) image (see Eq. 13 in the ZOGY paper) and a statistics (also known as significance / *Scorr*, S) image (see Eqs. 16 and 17 in the ZOGY paper) providing the probability of a transient being present at a particular position. The *Scorr* image, S, is normalized by the Poisson noise of the input images and the error resulting from the astrometric uncertainty when remapping the reference image to the new image frame; this leads to the *Scorr* image (see Eq. 25 in the ZOGY paper), which we also refer to as the significance image. The unit of this *Scorr* image is standard deviations (sigma) and transients above an adopted significance threshold (we used *Scorr*$\geq$12 for the data presented in this thesis) are normally selected on the basis that they are potentially significant. In practice, many significant but artificial transients are present in collected data due to cosmic rays, saturated stars, bad pixel regions or other image artefacts; many of these can be filtered out with some basic constraints applied to the size and shape, but for each image, tens of transient candidates remain where we only expect a few astrophysical transients per image.

The MeerLICHT/BlackGEM database ingests the transient catalogues produced by BlackBOX, including 1×1 arcminute thumbnail cut-outs around each transient of the new (N), the reference (R), the difference (D) and significance / *Scorr* (S) image as shown in Figure 6.1. The process of creating a training set for `MeerCRAB` is detailed in the next section.

6.3.1 MeerLICHT dataset and data labelling

Our goal is to automate the separation of real candidates from bogus objects for the MeerLICHT transient detection pipeline. The main challenge faced when building a supervised automated system is that we need to construct a large labelled data set that can be used to train a ML model. In addition, the data set needs to be representative, that is, we should have a fairly balanced number of real and bogus candidates. If the latter are unavailable, ML algorithms built from such unrepresentative data tend to be biased towards the majority class (e.g. Hosenie et al. 2019b).

We therefore construct a large representative training dataset for the Real-Bogus challenge

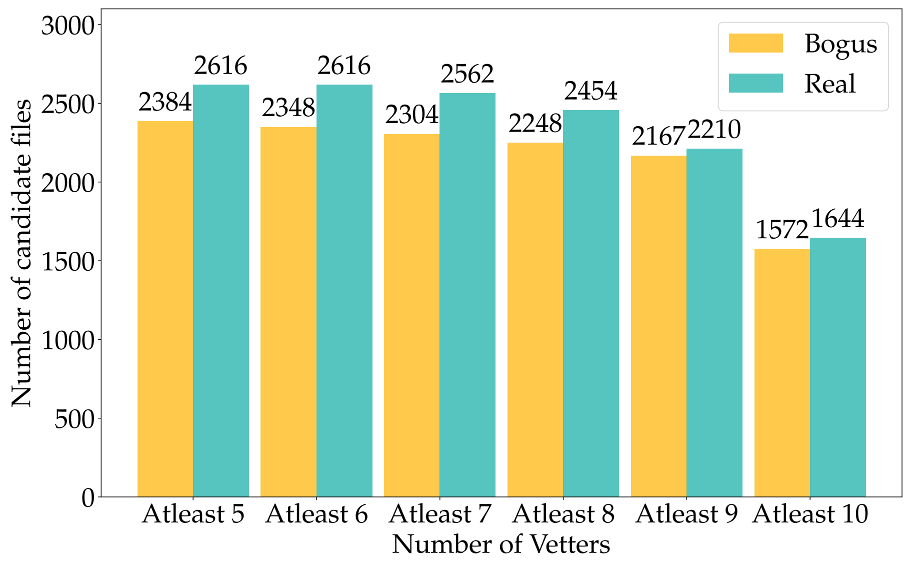


Figure 6.4 – Thresholding method used to analyse the labelling provided by vetters. The x-axis represents criteria we applied on the labelling. For *Atleast 9 (T9)*, this indicates that 9 out of the 10 vetters have agreed on the labelling. Similar strategy applies for *Atleast 5 (T5)* to *Atleast 10 (T10)*. The y-axis is the number of candidates with a given label, either real or bogus.

by manually vetting a selection of transients, using a web-interface, known as `MeerVETTING` as shown in Figure 6.2. Using the MeerLICHT database, a set of 5000 transient candidates were randomly selected from MeerLICHT data taken between 2017 and 2020. A team of 10 people ('vetters') were presented with three 100×100 pixels images during vetting, i.e. the new (N), reference (R), and difference (D). The properties for real and bogus candidates were defined as a phenomenological distinction based solely on the MeerLICHT data, not an astrophysical distinction. In the context of MeerLICHT data, 'positive' implies positive pixel values and 'negative' points to negative pixel values located at the centre of the 100×100 images.

- Real is any source that is of astrophysical origin, and variable in time and/or position. A real source therefore

    1. has a shape that reflects a point-source. Most MeerLICHT data is taken in decent focus conditions, so it implies that the source is round, and has a (visual) extent of $\sim$ 5-10 pixels,
    2. is positive in either the new or the reference image,
    3. can be variable in both directions, e.g. fading or brightening between the new and reference image, and is therefore positive or negative in the difference image and the significance image,
    4. can (dis)appear between the new and the reference image. This means that in one of the two images there is no source at all, and in the other there is a clear point-source.

- Bogus is any source that is not of astrophysical origin. A bogus source therefore generally has

  1. a shape that is not a point-source: not round, not 'Gaussian', with a size $\lesssim 5$ pixels or $\gtrsim 10$ pixels,
  2. is negative in the new image,
  3. is positive in the new image but negative in the reference image.

Before using the `MeerVETTING` web-interface, vetters were provided with a visual guide of the various properties of real and bogus candidates. By providing vetters with a guide, we in principle create better annotators, who should produce better labels which in turn should yield improved ML models - as long as the guide itself is not inherently biased in some way. The above characteristics are summarised in a decision-tree as shown in Figure 6.3. Using the `MeerVETTING` web-interface, vetters based their decision to manually vet a source as either Real or Bogus by following the decision-tree.

Despite these guidelines, the decision is still subjective whilst there remain boundary cases that are hard to label. Therefore large training datasets will almost always contain examples with inaccurate labels. We test the performance of the `MeerCRAB` models by (i) removing confused candidates (noisy labels) using a *thresholding* method, and (ii) including the entire dataset with labels based on the latent class model, L_{lcm}. In the following sections, we provide a brief discussion of the two methods.

6.3.2 Labelling data with Thresholding Method

Our sample of real-bogus training samples is constructed from a pool of 5000 candidates that has been selected randomly. Each object is classified by 10 vetters as either real, bogus or can be skipped if they are unsure how to classify it. Each vetter's ability to classify a particular candidate may vary according to the class, images, criteria and experience. This may result in very different classifications for the same candidate. We therefore assign a probability $\mathcal{P}\left(Real\right)$ and $\mathcal{P}\left(Bogus\right)$ to each vetted candidate as follows:

$$\boxed{\mathcal{P}\left(\text{Real}\right) = \frac{n\left(R\right)}{n\left(T\right)},} \tag{6.3.1}$$

$$\boxed{\mathcal{P}\left(\text{Bogus}\right) = \frac{n\left(B\right)}{n\left(T\right)},} \tag{6.3.2}$$

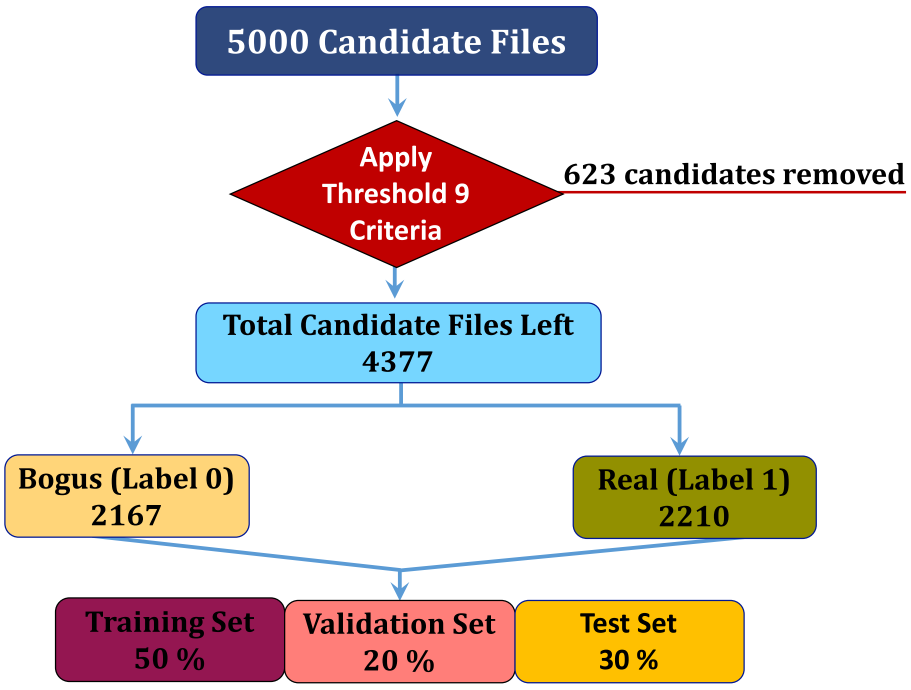
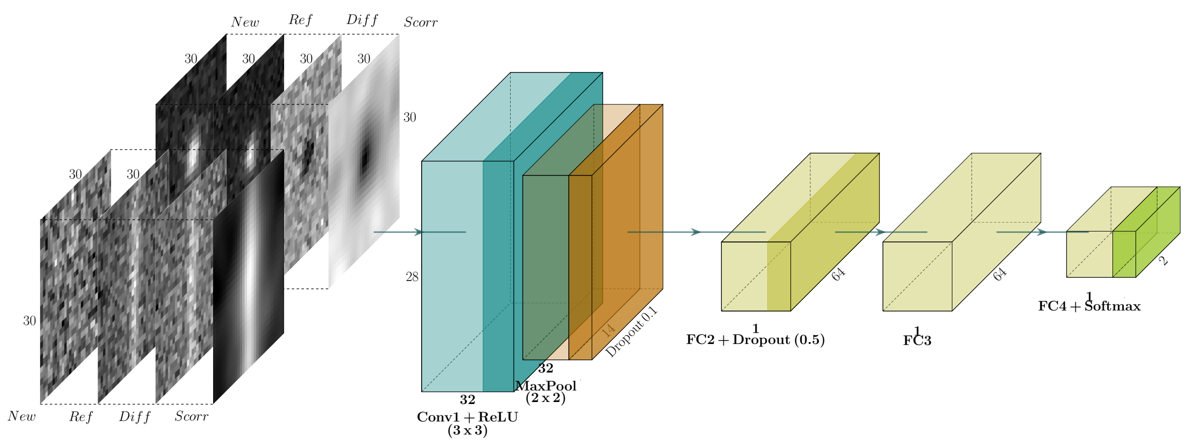


Figure 6.5 – An example of the procedure for selecting the candidates for training the CNN using the thresholding approach, applied on the 5000 candidates. In this example, *Atleast 9 (T9)* is applied and we note that 623 candidates are discarded. Then, the remaining candidates (4377) are split into training, validation and test set for training and evaluation processes.

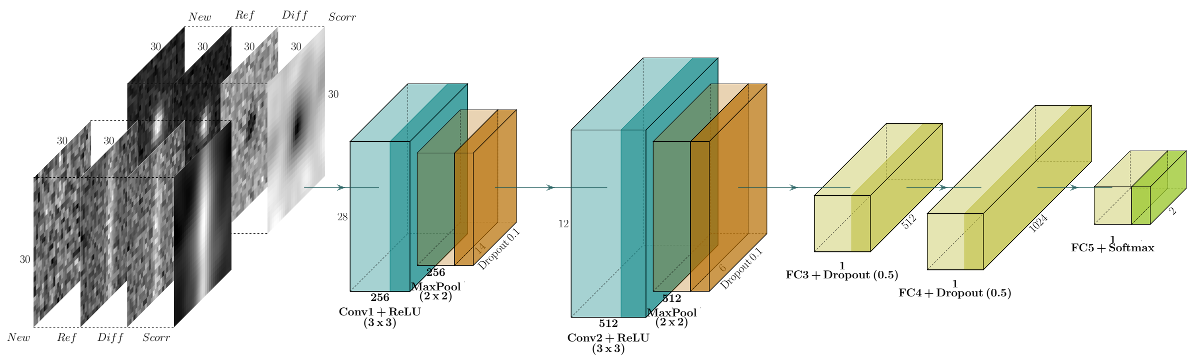
where $n\left(R\right)$ is the total number of vetters who classified a candidate as real, $n\left(B\right)$ is the total number of vetters who classified a candidate as bogus, $n\left(T\right)$ is the total number of vetters classifying a particular candidate and in this case $n\left(T\right) = 10$ as none of the vetters skipped a particular candidate. The vetters classification results are illustrated in a bar plot as shown in Figure 6.4. If a candidate has $\mathcal{P}\left(Real\right) \geq 0.9$, it will be given the label real or if $\mathcal{P}\left(Bogus\right) \geq 0.9$, it will be assigned as bogus in the bar plots with x-axis "*Atleast* 9" (**T9**). Each bar corresponds to a threshold, e.g, the last bar indicates that out of 5000 candidates, 3216 labelled objects are agreed upon by all the 10 vetters, of which 1572 are bogus and 1644 real. Therefore, there are 1784 sources remaining for which vetters did not agree completely and these confused candidates are removed from the data when using **T10**. In the "*Atleast 9*" (**T9**) case, there are 2167 sources where vetters agreed they are bogus and 2210 sources where "*Atleast 9*" vetters say they are real, and so on. Going down to **T5**, that is, "*Atleast 5*" vetters where all the 5000 candidates have been assigned a class, with 2384 bogus and 2616 real. In this thesis, we will analyse what happens to the classification results when varying thresholds from **T8** to **T10**.

6.3.3 Labelling data with Latent Class Model, $\mathbf{L}_{lcm}$

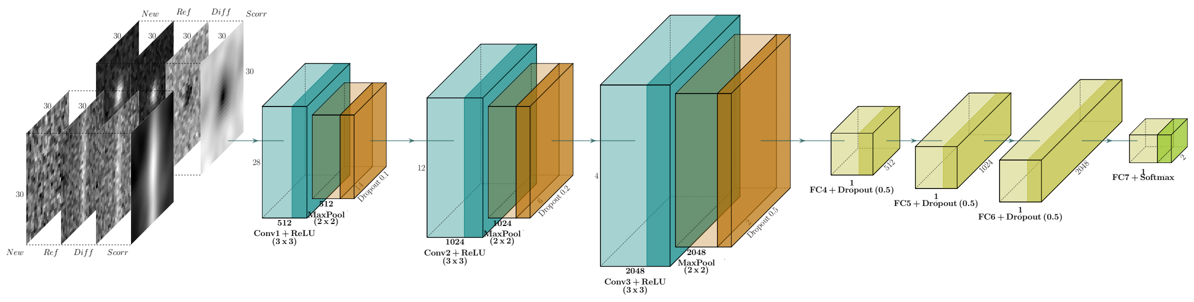
Latent class model (LCM) is a statistical technique used to classify candidates into mutually exclusive, or latent classes. When data is in the form of a series of categorical responses,



(a) MeerCRAB1



(b) MeerCRAB2

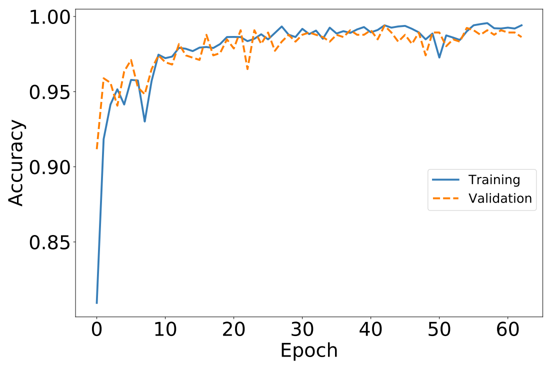


(c) MeerCRAB3

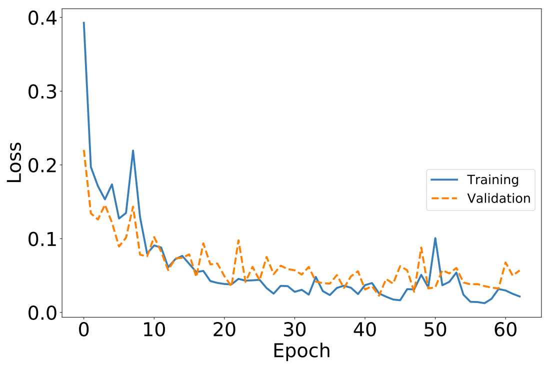
Figure 6.6 – The three network architectures considered in this work: `MeerCRAB1`, `MeerCRAB2` and `MeerCRAB3`. We show four images grouped together (new, reference, difference and significance) to form the input of the networks, followed by convolutional layers, max-pooling, dropout and dense layers. At the end, the network outputs a probability whether a candidate is either real or bogus during the prediction phase.

for example individual-level voting data as in the case of real-bogus classification, it is often an interesting technique to identify and characterize clusters of similar cases. In this work, some confused sources[¶] were removed from the data when using the *thresholding* method as shown in Figure 6.5. The process outlined in Figure 6.5, is exactly the same irrespective of the threshold used, the only difference lies in the number of candidates removed when the threshold is applied. However, these confused sources are useful for determining how the system will perform in a real-world scenario. Therefore, confused examples will also be used

[¶]5 vetters labelled them as bogus and the other 5 as real.



(a) Accuracy



(b) Loss

Figure 6.7 – Learning curves of the `MeerCRAB3` model with *T9* and *NRD* as input. The left panel shows the training and validation accuracy over iterations/epochs. The right panel shows the change in negative log-likelihood/loss with epochs. It can be observed that the training objective decreases consistently over iterations, but at some point (around 48 epochs) the validation set loss eventually starts to increase again. An early-stopping technique is applied to avoid overfitting by terminating that training process. At this stage, the algorithm picks the best parameters at 48 epochs.

during the evaluation phase, and this is achieved by using L_{lcm} to assign them their most likely labels. Therefore, for the L_{lcm} technique we used all 5000 candidates during the training and evaluation phase.

LCM relates a set of observed multivariate variables to a set of latent variables. The latent variable is usually discrete. A class is identified by a pattern of conditional probabilities that provide the chance that variables are given certain values.

Let us take the situation of real versus bogus. We want to use LCM to understand the labels provided by the vetters and give a final label to each source. Imagine that class 0-1 is given to a range of candidates with characteristics a, b, c, and d and that class 0 is associated with the presence of characteristics a, b, and c, and class 1 with characteristics b, c and d. LCM will try to detect the presence of latent classes (the candidates entities), generating patterns of association in the characteristics. Then LCM is used to classify candidates according to their maximum likelihood class membership.

The introduction of a latent variable ensures conditional independence within each latent class, the observed variables, in this case the vetters' labelling, are statistically independent. The association between the observed variables is explained by the classes of the latent variable (McCutcheon, 1987). The latent class model can be formulated as follows:

$$\boxed{\mathcal{P}_{i_1,i_2,\ldots,i_N} \approx \sum_{c}^{C} \mathcal{P}_c \prod_{n}^{N} \mathcal{P}^{n}_{i_n,t},} \tag{6.3.3}$$

where C is the number of latent classes and in our case, $C = 2$, i.e. real and bogus classes. N is

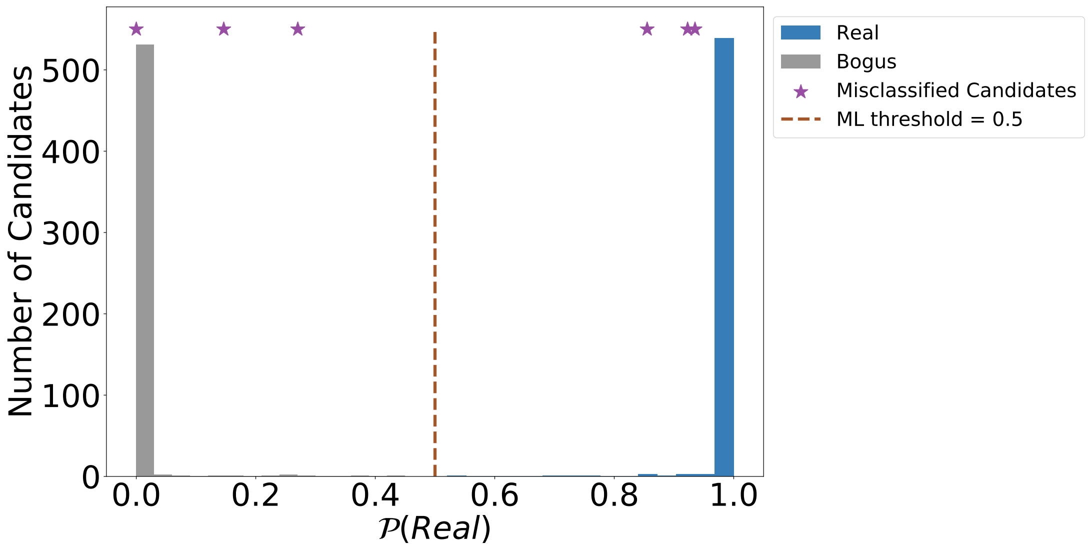


Figure 6.8 – Probability distribution on test data with 1095 candidates, trained with model configuration `MeerCRAB3` with $T9$ and NRD as input.

the number of observed binary variables (in this case, $n = 1, \ldots, 10$, since we have 10 vetters) and $\mathcal{P}_c$ are the unconditional probabilities that should sum to one. $\mathcal{P}^n_{i_n,t}$ are the marginal/conditional probabilities.

6.4 `MeerCRAB`: A Real-Bogus Intelligent Distinguisher for the MeerLICHT facility using Deep Learning

Before providing further details on the model used, we define a few important terms that will be used in this work. We labelled bogus examples as 0 and real examples as 1, therefore,

- TP: true positives are real candidates correctly classified as real,
- TN: true negatives are bogus candidates correctly classified as bogus,
- FP: false positives are bogus candidates that are classified as real,
- FN: false negatives are real candidates that are classified as bogus.

From the above definition, our classifier must minimize the number of FP and FN. This is because we do not want to lose many real candidates falsely classified as bogus (FN) and we want to minimize contamination (FP) by any bogus candidates in our final sample.

To overcome this challenge, we employed a Convolutional Neural Network as it has been proven by various studies to have excellent classification performance (Gieseke et al., 2017; Cabrera-Vives et al., 2017; Bellm et al., 2019; Vafaei Sadr et al., 2019; Lin et al., 2020). In this work, we construct three CNN models, referred to as `MeerCRAB1`, `MeerCRAB2`, and `MeerCRAB3`. The details of the CNN models are illustrated in Figure 6.6. A brief overview of each layer in the `MeerCRAB` architecture is explained in details in §3.4.3. `MeerCRAB1` is a network with a single convolutional layer (*CL*), `MeerCRAB2` comprises two *CLs* while `MeerCRAB3` is a deeper network with three *CLs*. Each *CL* is made of (3×3) pixel filters, together with a Rectified Linear Unit (ReLU, Agarap 2018) function, followed by a pooling layer with filter size of (2×2). After the *CLs*, we used fully connected layers (also known as dense layers). In addition, we use a dropout rate varying from (0.1 to 0.5) after each of the pooling and dense layers as seen in Figure 6.6 to avoid over-fitting. For the output layer, we used a softmax function that outputs a probability value between 0 and 1.

The implementation of `MeerCRAB` is made using the `Tensorflow`[||] and `Keras` (Chollet & others, 2018) API with *Python v3.6*. For training the `MeerCRAB` models, we used an Nvidia GeForce GTX 1080Ti 11GB GPU. During training, the binary cross-entropy loss function, Adam optimizer (Kingma & Ba, 2014) with a low learning rate (lr = 0.0002) and a batch-size of 64 were used. We then split our data for each and every experiment as follows: 50% training, 20% validation and 30% testing. As input to the `MeerCRAB` models, we cropped the (100×100) pixel images that were analysed by vetters to (30×30) pixel images. We also utilized the (100×100) pixel images during training, but we observed a drop in performance and therefore only use the (30×30) pixel images cropped from the centre. This pre-processing step increases the likelihood that the models will retain useful information and not be distracted by noise or spurious patterns.

As described above, the MeerLICHT database provides four images N, R, D and S. We perform several analyses which are presented in §6.5, where we use a combination of these images. This work allows us to understand which image/s is/are important for helping the CNN to output better classification results. We therefore use cut-outs of N, R, D, S and stack them to form either of these input images: $NRDS$ $(30 \times 30 \times 4)$, NRD $(30 \times 30 \times 3)$, NR $(30 \times 30 \times 2)$, D $(30 \times 30 \times 1)$, or S $(30 \times 30 \times 1)$.

Afterwards, we apply data augmentation to mitigate the risk of over-fitting which may result in poor generalisation performance upon data outside the training set. Therefore, at each training step, the images are augmented by flipping them randomly in a horizontal and/or vertical direction. We do not apply any rotation or translation to the images. This data augmentation step is important as it helps to increase the training sample size and the probability

[||] https://github.com/tensorflow/tensorflow

Table 6.4.1 – Contingency table used as a visual aid for model selection. The sum of A, B, C, and D represents the total number of instances in the test set. A is the number of instances correctly classified by both models. D is the number of instances misclassified by both models. B is the number of objects that model 1 correctly classified, but has been misclassified by model 2. C is the number of examples misclassified by model 1 but being correctly classified by model 2.

	Model 2 Correct	Model 2 Wrong
Model 1 Correct	A	B
Model 1 Wrong	C	D

that the CNN models will encounter similar images twice, will decrease.

Moreover, to further avoid any over-fitting during training, we employ an early stopping technique to stop the training process if no further decrease in validation loss is observed for several epochs. The various cases and models are trained for a number of epochs varying from 40 to 150. We show the optimisation curves (the accuracy and loss curves) during training and validation in Figure 6.7 for the best models with NRD and **T9** as input. We observe that both the training and validation accuracy reaches a range between 98.5% to 99.5% where the algorithm picks the best parameters at 48 epochs.

6.4.1 Evaluation Performance

We use different evaluation techniques as detailed in §3.2.3, for instance, the accuracy, the precision, the recall and the Matthew correlation coefficient (MCC) metrics to evaluate the `MeerCRAB` models. In addition, we utilize the McNemar test for model comparison.

6.4.1.1 McNemar's Test

The McNemar test (McNemar, 1947) is a statistical test used to check marginal homogeneity in the context of statistical models. It is used to compare the predictive accuracy of two models' predictions and it is based on a contingency table as shown in Table 6.4.1. The latter provides insights for model selection, in contrast to a typical confusion matrix. It shows the number of instances/predictions model 1 and model 2 got right or wrong given a fixed test set. In McNemar's test, a null hypothesis, H_0 is formulated such that $\mathcal{P}(B)$ and $\mathcal{P}(C)$ are similar or it can be interpreted as two models perform equally well. Therefore, the alternative hypothesis H_1 is that $\mathcal{P}(B) \neq \mathcal{P}(C)$ or the two models do not perform equally well. Edwards (1948) proposed a corrected McNemar test statistic that can be computed as given below:

$$\chi^2 = \frac{(|B - C| - 1)^2}{B + C}. \tag{6.4.1}$$

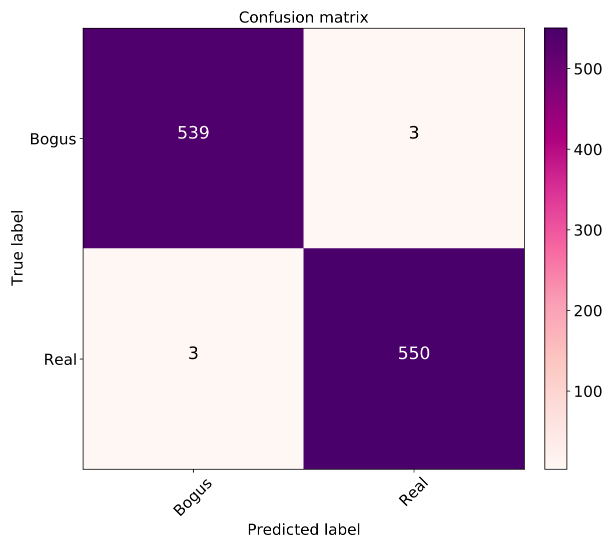


Figure 6.9 – Confusion Matrix without normalisation from a test set of 1095 candidates. The model is trained using `MeerCRAB3` with *T9* and *NRD* as input. The diagonal represents the correctly classified instances in the test. The off-diagonals represent the number of instances that are misclassified. We note that we have a very low FP and FN with this model configuration.

If the sum of B and C is greater than 25 or sufficiently large, under H_0 the χ^2 value follows a chi-squared distribution with one degree of freedom. If we set a significance threshold for example, $\alpha = 0.05$, the p-value can be computed. Assuming that the null-hypothesis is true, the p-value implies the probability of observing a larger chi-squared value. However, if the p-value is less than α, then the null hypothesis is rejected, that is, the two models do not perform equally well.

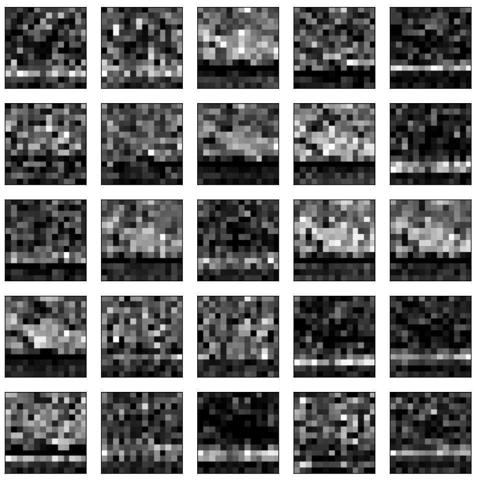
In the case where the sum of B and C is less than 25, an exact binomial test is used instead, since the chi-squared value may not be well approximated by the chi-squared distribution. Thus, the exact p-value is calculated as follows:

$$\boxed{\text{p-value} = 2 \times \sum_{i=B}^{n} \binom{n}{i} 0.5^{i} (1-0.5)^{n-i},} \tag{6.4.2}$$

where $n = B + C$ and a factor 2 indicates the computation of a two-sided p-value. For model selection, if the p-value is less than 0.05, we reject the null hypothesis, thus one model is outperforming the other. However, if the p-value is greater than 0.05, we do not reject the null hypothesis and it indicates that both models perform equally well.

6.5 Results and Analysis

The classification performance of the `MeerCRAB` models are analysed as follows:


(a) Bogus feature maps

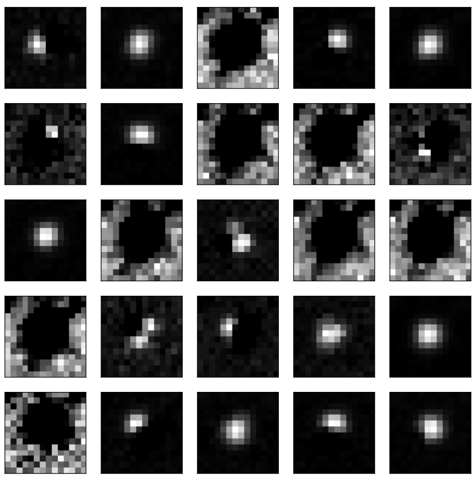

(b) Real feature maps

Figure 6.10 – Feature maps induced by the convolutional layer in `MeerCRAB3` given a "bogus" and "real" source, respectively. We observe that the model distinguishes between background noise and the source at the centre of an image.

- The `MeerCRAB` models are trained on a subset sample of input images and validated on an unseen image sample.
- During the prediction phase, the trained models are then used to output probabilistic predictions for unseen images in the test set as shown in Figure 6.8. The probability distribution from the output of the `MeerCRAB` models spanned the range of $\mathrm{P}_{\texttt{MeerCRAB}} \in [0;1]$. Therefore, a candidate is predicted as bogus if $\mathrm{P}_{\texttt{MeerCRAB}} < 0.5$ and as real if $\mathrm{P}_{\texttt{MeerCRAB}} \geq 0.5$. $\mathrm{P}_{\texttt{MeerCRAB}} = 0.5$ indicates a random guess and the `MeerCRAB` models are confused between real and bogus candidates.

We investigated various scenarios for training and evaluating the pipeline. We made use of three network structures: `MeerCRAB1`, `MeerCRAB2`, & `MeerCRAB3`. We also varied the number of images used as input to the three model architectures. We use these combinations of input images *NRDS*, *NRD*, *NRS*, *NR*, *D* and *S* independently. In addition, we investigated the effect of varying the thresholding applied on data labelling, ie., the effect of noisy data labels as discussed in §6.3.2 and §6.3.3.

For all the experiments considered, it is worth mentioning that we train the networks with and without data augmentation. We found that employing data augmentation decreases the number of misclassifications. Therefore, we report results with the models trained with data augmentation only in this work. Using data augmentation, we found the objective values steadily decreased during the fitting process on both the training set and validation set. Figure 6.7 provides an illustration of the objective functions and we observe that both the training and validation loss are close to each other, indicating that the models did not overfit on the training

Table 6.5.1 – The results for various labelling methods are presented in terms of precision, recall, accuracy and MCC values using *NRD* as input to the three models.

Methods of labelling	Precision	Recall	Accuracy	MCC
		`MeerCRAB1`		
L_{lcm}	0.96	0.96	0.960	0.920
T8	0.98	0.98	0.980	0.958
T9	0.98	0.98	0.979	0.958
T10	0.99	0.99	0.991	0.983
		`MeerCRAB2`		
L_{lcm}	0.97	0.97	0.967	0.936
T8	0.99	0.98	0.977	0.953
T9	0.99	0.99	0.986	0.973
T10	0.99	0.99	0.994	0.988
		`MeerCRAB3`		
L_{lcm}	0.97	0.97	0.968	0.936
T8	0.99	0.99	0.988	0.976
T9	**0.99**	**0.99**	**0.995**	**0.989**
T10	1.00	1.00	0.998	0.995

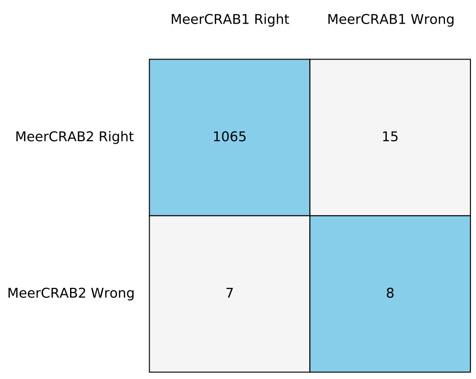
sample. Hence, the dropout layers as well as various regularization techniques used in the models are effective measures to prevent overfitting.

6.5.1 Case studies

We start by providing an overall comparison of various scenarios used and will subsequently analyse some of the models along with certain modifications in more details. The `MeerCRAB` models are evaluated on various metrics. Based on the $P_{\texttt{MeerCRAB}}$ value, we construct a confusion matrix to have an overview of the classification results. The confusion matrices display the fraction of correctly classified candidates as TP, TN along the diagonal. The off-diagonal values in the confusion matrices show the misclassified examples (FP and FN). We also evaluated the models based on precision, recall, accuracy, and MCC as discussed in §6.4.1. The results for the various scenario cases are summarised in Table 6.5.1, 6.5.2, 6.5.3, 6.5.4 and Figure 6.9.

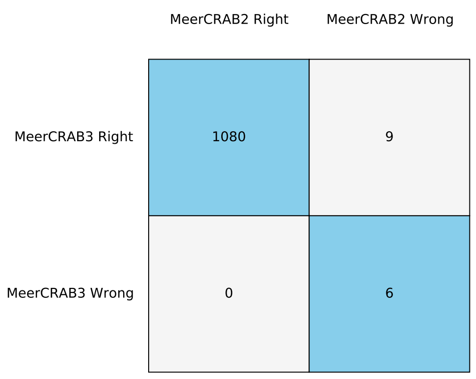
6.5.1.1 Data labelling based on Thresholding criteria

In supervised learning algorithms, the success of deep neural networks depends highly on the availability and accessibility of high-quality labelled training data. In this work, we found that the presence of label errors (label noise) in the training data greatly reduced the accuracy of all `MeerCRAB` models on test data. Unfortunately, large training datasets almost always contain examples with inaccurate or incorrect labels. It is a challenging task to train deep neural net-



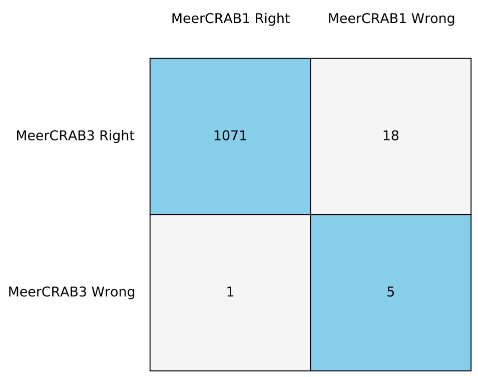
	MeerCRAB1 Right	MeerCRAB1 Wrong
MeerCRAB2 Right	1065	15
MeerCRAB2 Wrong	7	8

(a) `MeerCRAB1` vs `MeerCRAB2`



	MeerCRAB2 Right	MeerCRAB2 Wrong
MeerCRAB3 Right	1080	9
MeerCRAB3 Wrong	0	6

(b) `MeerCRAB2` vs `MeerCRAB3`



	MeerCRAB1 Right	MeerCRAB1 Wrong
MeerCRAB3 Right	1071	18
MeerCRAB3 Wrong	1	5

(c) `MeerCRAB1` vs `MeerCRAB3`

Figure 6.11 – The contingency tables for the three models under consideration with *T9* and *NRD* as input. For model selection, we compare the performance accuracy of each model on a similar test set. From the plots, we observe that `MeerCRAB3` is a better model compared to `MeerCRAB1` and `MeerCRAB2`.

works (DNNs) robustly with noisy labels (Han et al., 2018) as DNNs have a high capacity to fit noisy labels (Zhang et al., 2016), and this results in poorer model performance in practice.

In this work, in order to sanity check the potential generalisation performance of our models, we utilised latent class models, L_{lcm} to label our data - including those samples for which vetters could not reach agreement. This approach allows us to introduce labelling noise. Whilst this noise reduces the performance of our models causing accuracy to drop to 0.968, it helps us obtain an impression of real-world performance where imperfect labelling and noise cannot be controlled for. Results show that models trained using such data still perform very well. This indicates that the networks are robust to noise.

Our analysis also involves comparing various thresholding criteria: **T8**, **T9**, and **T10** along with the `MeerCRAB3` model with *NRD* as input. The results for labelling techniques are summarised in Table 6.5.1. We observe that as the threshold increased from **T8** to **T10**, the accuracy of the model increases from 0.988 to 0.998 and MCC values increase from 0.976 to 0.995. However, when using the L_{lcm} method, we note a significant drop in L_{lcm} accuracy (0.968) as deep networks tend to memorize training label noise, resulting in poorer model performance. There-

Table 6.5.2 – The results with Threshold 8 (**T8**) are presented in terms of precision, recall, accuracy and MCC values using various combinations of input images (new (N), reference (R), difference (D), and significance (S)) to the three models.

Number of Images	Precision	Recall	Accuracy	MCC
`MeerCRAB1`				
NRDS	0.98	0.98	0.980	0.958
NRD	0.98	0.98	0.977	0.954
NRS	0.97	0.97	0.971	0.942
NR	0.96	0.96	0.962	0.923
D	0.88	0.88	0.877	0.762
S	0.83	0.83	0.831	0.662
`MeerCRAB2`				
NRDS	0.97	0.97	0.975	0.948
NRD	0.98	0.98	0.977	0.953
NRS	0.99	0.99	0.986	0.973
NR	0.99	0.99	0.987	0.975
D	0.91	0.91	0.910	0.825
S	0.89	0.89	0.887	0.777
`MeerCRAB3`				
NRDS	0.98	0.98	0.983	0.966
NRD	**0.99**	**0.99**	**0.988**	**0.976**
NRS	0.98	0.98	0.983	0.968
NR	0.98	0.98	0.981	0.963
D	0.92	0.92	0.921	0.843
S	0.89	0.89	0.891	0.786

fore, it is necessary to obtain fairly high-quality labels for a CNN to work appropriately, thus removing noisy labelling from the model yields better model performance.

However, from here on we focus on experimentation that uses labels obtained via thresholding criteria : **T9** only, and not the L_{lcm}. we do this as we aim to have the best optimized model with an adequate number of candidates along with a good level of agreement on the labelling, such that we obtain fewer false positives and false negatives.

6.5.1.2 Network architectures

We trained the `MeerCRAB` pipeline with three different architectures using **T9** data. `MeerCRAB1` consists of a single CL, `MeerCRAB2` is trained with 2 CLs and `MeerCRAB3` with 3 CLs. From Table 6.5.2, Table 6.5.3 and Table 6.5.4, we note that `MeerCRAB1` which is a shallow network yields a surprisingly good performance for **T8** to **T10**. Looking at Table 6.5.3 with **T9**, we note that `MeerCRAB1` achieves an accuracy of 0.980 and MCC = 0.960 on the test set. When using deeper networks (`MeerCRAB2` and `MeerCRAB3`), we found that we obtain a better performance. The accuracy for `MeerCRAB2` and `MeerCRAB3` using NRD as input are 0.986 & 0.995 and MCC values

Table 6.5.3 – The results with Threshold 9 (**T9**) are presented in terms of precision, recall, accuracy and MCC values using various combinations of input images (new (N), reference (R), difference (D), and significance (S)) to the three models.

Number of Images	Precision	Recall	Accuracy	MCC
`MeerCRAB1`				
NRDS	0.98	0.98	0.980	0.960
NRD	0.98	0.98	0.979	0.958
NRS	0.98	0.98	0.978	0.956
NR	0.97	0.97	0.972	0.946
D	0.86	0.83	0.823	0.690
S	0.86	0.85	0.853	0.708
`MeerCRAB2`				
NRDS	0.99	0.99	0.988	0.976
NRD	0.99	0.99	0.986	0.973
NRS	0.99	0.99	0.986	0.973
NR	0.99	0.99	0.989	0.978
D	0.91	0.91	0.912	0.827
S	0.89	0.87	0.865	0.751
`MeerCRAB3`				
NRDS	0.99	0.99	0.987	0.974
NRD	**0.99**	**0.99**	**0.995**	**0.989**
NRS	0.99	0.99	0.990	0.980
NR	0.99	0.99	0.989	0.978
D	0.93	0.93	0.931	0.863
S	0.89	0.87	0.868	0.760

are 0.973 & 0.989 respectively.

To have a better understanding of why `MeerCRAB1` yields a good performance, we plot in Figure 6.10 the feature maps of the *CL* for a bogus and real example. We observe that there appears to be feature maps that activate on the background ("dark centre"), while other maps activate on different parts of the centre. This suggests that the network can distinguish between the source itself and the background, thus it is able to classify images relatively unhindered by different levels of noise.

To determine which network performs best, we employ the McNemar statistical test as discussed in §6.4.1.1. We plot the contingency tables for the three models in Figure 6.11. The sample size in the B and C cells are relatively small and $(B + C) < 25$ to approximate the chi-square value from the chi-square distribution. We therefore compute the p-value in this case from a binomial distribution. Assuming we use $\alpha = 0.05$, if the p-value is less than 0.05, we reject the null hypothesis that both models perform equally well on the test set and if the p-value is greater than 0.05, we do not reject the null hypothesis, we then conclude that the two models perform equally well. From Table 6.6.1, we note that when comparing `MeerCRAB1`

Table 6.5.4 – The results with Threshold 10 (**T10**) are presented in terms of precision, recall, accuracy and MCC values using various combinations of input images (new (N), reference (R), difference (D), and significance (S)) to the three models.

Number of Images	Precision	Recall	Accuracy	MCC
		`MeerCRAB1`		
NRDS	1.00	1.00	0.995	0.990
NRD	0.99	0.99	0.991	0.983
NRS	0.98	0.98	0.985	0.970
NR	0.99	0.99	0.986	0.973
D	0.92	0.92	0.920	0.841
S	0.93	0.93	0.934	0.868
		`MeerCRAB2`		
NRDS	1.00	1.00	0.995	0.990
NRD	0.99	0.99	0.994	0.988
NRS	0.98	0.98	0.984	0.968
NR	0.99	0.99	0.992	0.985
D	0.94	0.94	0.944	0.888
S	0.94	0.94	0.940	0.881
		`MeerCRAB3`		
NRDS	0.99	0.99	0.990	0.980
NRD	**1.00**	**1.00**	**0.998**	**0.995**
NRS	0.99	0.99	0.994	0.988
NR	0.99	0.99	0.985	0.970
D	0.94	0.94	0.939	0.878
S	0.94	0.94	0.943	0.886

with `MeerCRAB2`, the p-values are greater than 0.05, we therefore conclude that the models have an equal performance. However, when comparing `MeerCRAB1` with `MeerCRAB3` and `MeerCRAB2` with `MeerCRAB3`, the p-value is less than 0.05 this indicates that one model is performing better. When we analyse Figure 6.11(b,c), we note that `MeerCRAB3` has less instances being misclassified compared to `MeerCRAB1` and `MeerCRAB2`. Therefore, we conclude `MeerCRAB3` is a better model compared to `MeerCRAB1` and `MeerCRAB2`.

6.5.1.3 Input Images

In this section, we investigate the effect of adding and removing images from the input to the CNN models. Vetters were shown only three images during vetting: N, R and D images. However, when training and evaluating the three networks, we compare the different groups of images, to see whether a competitive performance can be achieved with more or less input data. Focusing on **T9** and `MeerCRAB3`, we note that NRD input yields the best performing model with an accuracy of 0.995 and MCC value of 0.989. Similar results are obtained with **T8** and **T10**. The second best model in **T9** is with input NRS yielding an accuracy of 0.990

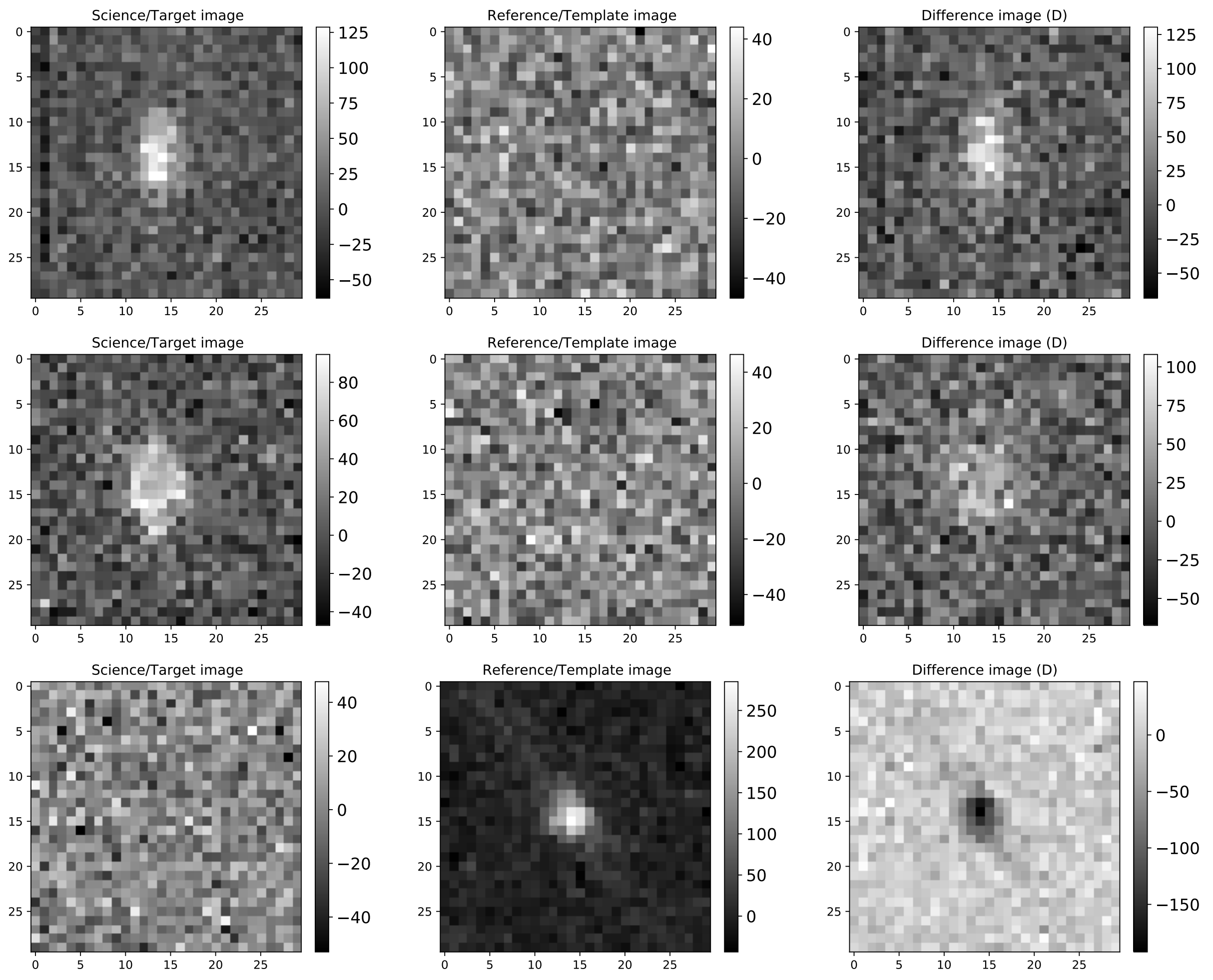


Figure 6.12 – The false positives obtained with `MeerCRAB3`, trained with *T9* labelling and *NRD* as input. These candidates are classified as bogus by humans. However, the best performing network misclassified them as real.

and MCC value of 0.980. With only *NR* as input, we note that `MeerCRAB3` performs equally well. Therefore, using a reduced image input set (*NR*) yields a competitive performance and indicates that a reduced set of images is sufficient for separating real and bogus. However, we note that using only *D* or *S* as input worsens the classification performance and this indicates that using information only from the difference or significance imaging is insufficient.

6.5.2 Analysis of misclassification

With the various investigations, it is worth mentioning that the classification performance is, in general, very similar for the different `MeerCRAB` models, i.e., neither a particular network structure nor the involved parameters seem to have a significant influence on the final classification performance. The conclusion one can draw at this point is that a standard CNN model seems to be well-suited for the task at hand and that even relatively simple networks yield a performance that is competitive with state-of-the-art approaches.

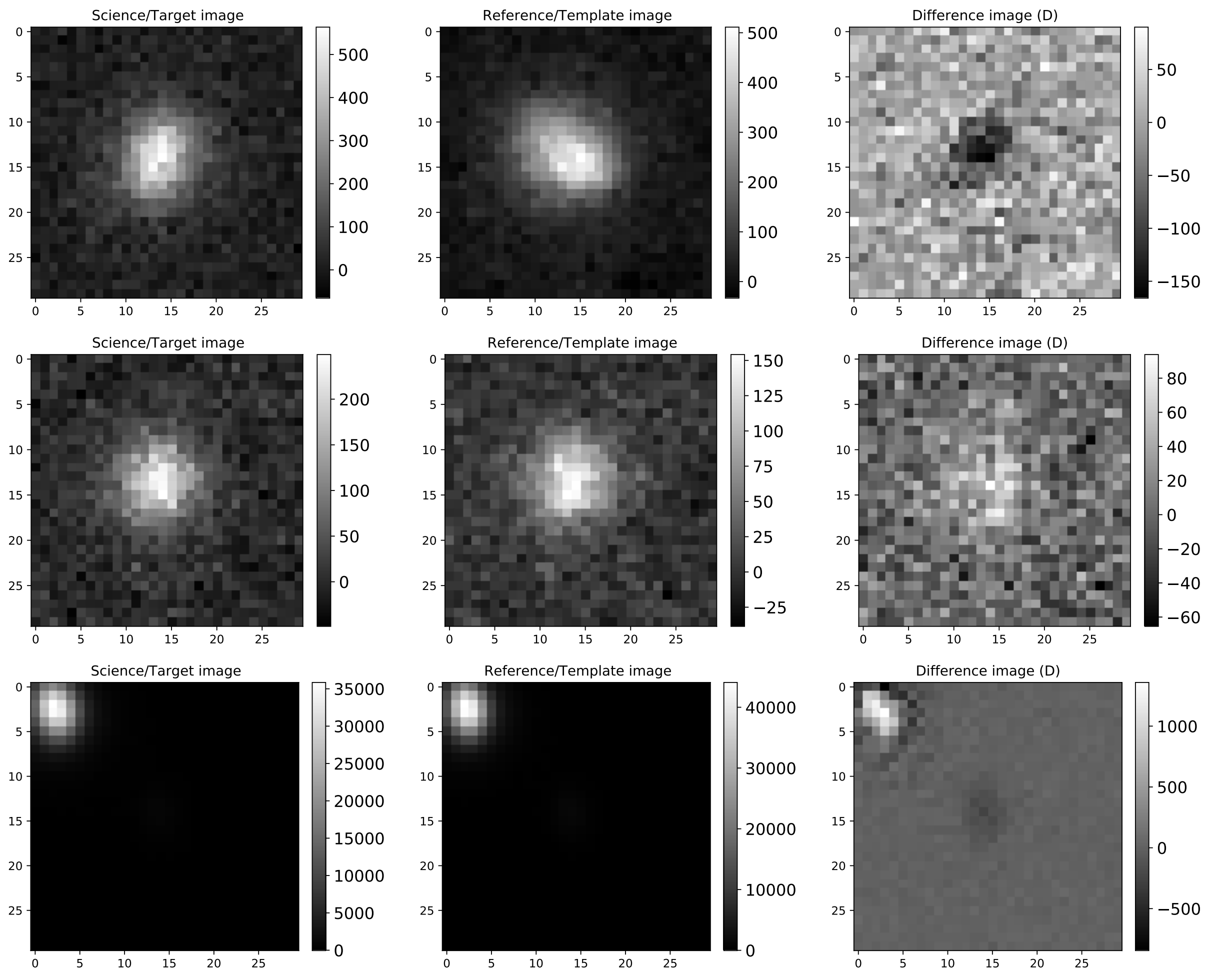


Figure 6.13 – The false negatives obtained with `MeerCRAB3`, trained with *T9* labelling and *NRD* as input. These candidates have been classified as real by vetters. However, the best model misclassified them as bogus candidates. The first two rows illustrate two real candidates with characteristics of an extended source with > 10 pixels extent. The large extent of the sources can be attributed to that fact that the seeing can be very poor.

In this section, we focus on our best model, `MeerCRAB3` network with the *NRD* configuration and **T9** as input. In Figure 6.12 and Figure 6.13, we present all misclassifications it made

In Figure 6.12, the top two rows show bogus candidates with the presence of elongated spikes but the CNN found high pixel values at the centre, thus got confused and classified them as a real. The last row shows bogus candidates caused by blinking tail light of planes which can be recognized by the stripe. However, the `MeerCRAB` models seemingly recognized it as real due to the presence of a strong point source at the centre. However, we find that the decision made by `MeerCRAB` fulfils our requirements as we do not feed information related to elongated spikes and tail-light-trails when training the ML algorithm. These are not commonly occurring bogus events at present but may be added to our training set in the future and it will enable `MeerCRAB` to identify them.

Figure 6.13 shows real candidates being misclassified as bogus. The top two rows are likely to be real point sources because the Seeing can be very poor, thus leading to the large extent of > 10 pixels. The fact that the source in the top row is oriented differently in the science/new and reference image suggests that it is not a Galaxy. However, these misclassifications are expected as these characteristics of real objects were not provided when training the network.

6.6 Conclusions

A deep learning framework, `MeerCRAB`, integrated in the MeerLICHT facility is a step forward in the automation and improvement of the transient vetting process. In practice, by using `MeerCRAB` we can significantly reduce the number of missed transients per night and this may have a great impact on detecting and classifying the unknown unknowns of our universe.

In this thesis, we detailed the process of developing `MeerCRAB`. To be able to train a deep neural network, we construct a large, high-quality labelled and representative dataset. To do so, we developed a vetting guidelines for vetters and taught them how real or bogus candidates in the MeerLICHT data appear. Then, a sample of 5000 candidates were provided to 10 vetters for labelling. Based on the vetters labels, we applied two methods to assign the final labelling to each candidate: (i) the *thresholding* method (**T8**, **T9** & **T10**) and (ii) latent class model L_{lcm}. At **T9**, a source is labelled as real if *atleast 9* out of the 10 vetters labelled it as real or vice-versa. We found that **T9** is a good threshold criteria to be used since we have enough samples for training and testing the models, hence providing high quality labelled data. We found that going lower than this (i.e. **T7**, **T8**) or using L_{lcm}, added noisy labels. When used to train the network, such data decreased the performance of the models.

Moreover, we demonstrated that by increasing the depth of the network, (`MeerCRAB1` to `MeerCRAB3`), the performance of the model increases as well. McNemar's statistical test showed that `MeerCRAB3` performs better than `MeerCRAB1` and `MeerCRAB2`. In addition, we used a combination of input images (new (N), reference (R), difference (D), significance (S)) as input to the three networks. We found that with only NR, we obtained competitive results. We also observed that adding the difference and significance images improves network performance.

In summary the best performing model has the following configuration: **T9** with `MeerCRAB3` having NRD as input. This model yields an accuracy of 99.5 % and MCC value of 0.989. This performance achieves an acceptable false positive and false negative rate for the real-time MeerLICHT transient detection pipeline requirements.

`MeerCRAB` is a crucial component of the MeerLICHT project which aims to detect and identify transient and variable sources. With the streaming data coming from MeerLICHT, the vast ma-

Table 6.6.1 – McNemar's Test using *T9* and *NRD* results for model selection. `MeerCRAB3` has the best performance than `MeerCRAB1` and `MeerCRAB2` as the p-value is less than 0.05.

Models	P-value
`MeerCRAB1` vs `MeerCRAB2`	0.13400
`MeerCRAB2` vs `MeerCRAB3`	0.00390
`MeerCRAB1` vs `MeerCRAB3`	0.00008

jority of astrophysical data are not only challenging to store, but also to classify efficiently and effectively. Therefore, `MeerCRAB` will enable the rapid identification of promising astrophysical sources in a timely-manner. In addition, `MeerCRAB` can be adapted to be a system that disentangles interesting objects from a noisy background. We have already implemented similar models in radio astronomy that distinguish Single Pulses from Radio Frequency Interference for the MeerKAT telescope (`FRBID`**: Fast Radio Burst Intelligent Distinguisher). `MeerCRAB` is a flexible software system, thus we were able to easily modify it to integrate different images as its inputs and as result, achieved high levels of performance when using it for radio astronomy images. Given the performance of `MeerCRAB` on both optical and radio image sources in astronomy, the method may have utility for those working in related areas.

**See https://github.com/Zafiirah13/multi_input_frbid and https://github.com/Zafiirah13/FRBID

7

FRBID: FAST RADIO BURST INTELLIGENT DISTINGUISHER AT THE MEERKAT TELESCOPE

This chapter has been submitted as a conference paper in NeurIPS conference 2021 with the title: '**FRBID: Fast Radio Burst Intelligent Distinguisher at the MeerKAT Telescope**'.

Zafiirah Hosenie, *Robert Lyon*, *Benjamin Stappers* et al.

Submitted in NeurIPS conference 2021 on **27 September 2021**.

Fast Radio Bursts (FRBs, Lorimer 2008) are luminous radio pulses, originating from extragalactic distances and have a duration of just a few milliseconds. This work is part of the MeerTRAP (more TRAnsients and Pulsars) project*. The aim is to use the MeerKAT telescope located in South Africa to search for, detect and localise those bursts (pulsars and fast radio transients) accurately in real-time. MeerTRAP will detect dozens of new bursts and localise them with high precision, thus will enable us to identify their hosts and distances. This information will enhance their use as cosmological probes. In this chapter, we provide a short overview of a single stage in the MeerTRAP pipeline, that is, the design, implementation and exploitation of state-of-the-art machine learning techniques (FRBID) to classify between Single Pulses (can be periodic pulsars (1ms - 30s), rotating radio transients (RRATs) or FRBs) and noise, and Radio Frequency Interference jointly referred to as N-RFI in real time at the MeerKAT telescope. At the moment, FRBID is currently being used 'off-line' but will be moved 'online' soon.

*https://www.meertrap.org/

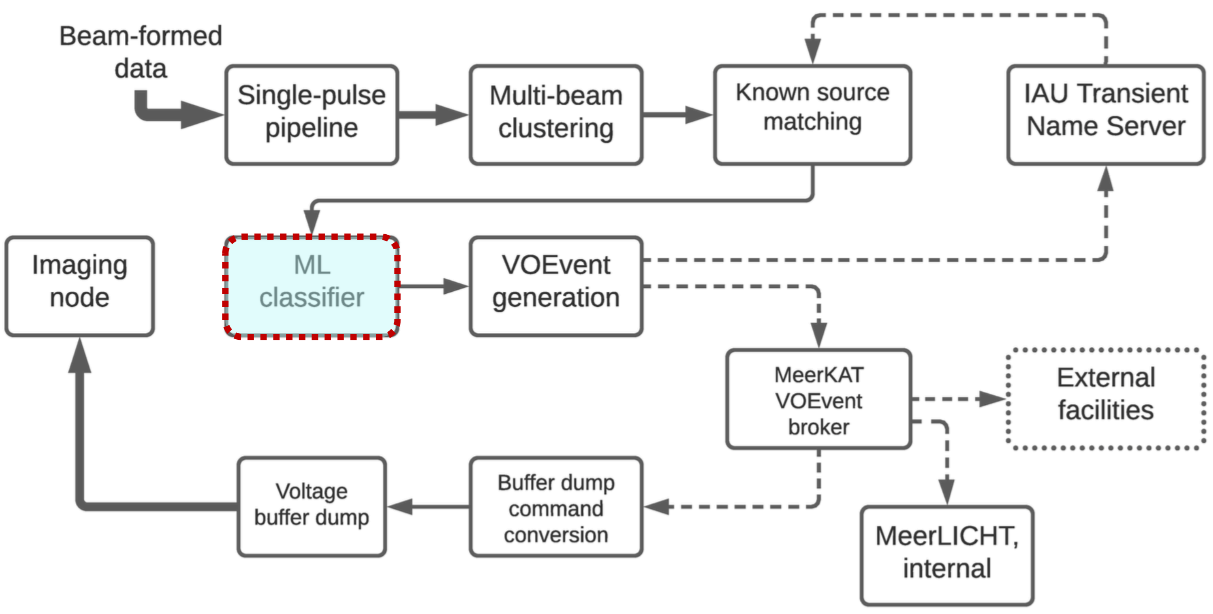


Figure 7.1 – An overview of the MeerTRAP real-time triggering pipeline at the MeerKAT telescope. The red highlighted region is the ML classifier developed in this work. Figure courtesy of Jankowski et al. (2020).

7.1 Introduction

Detecting and understanding the origin of FRBs is extremely challenging as their sporadic nature demands vast amounts of observing time and/or wide fields of view. In this work, we use data from the 64-dishes MeerKAT telescope (see §2.4.3 for more details) to classify single pulses/FRBs in real time as part of the MeerTRAP project. Real-time FRB detections will be performed using a single pulse search pipeline. For the time being, single pulses and N-RFI classification is being carried out 'offline'.

As the dispersion measure of candidates is unknown (see §2.3.1.2), this necessitates searching for pulses at different dispersion measures (DMs). The MeerTRAP team is currently working towards constructing a fully-automated, real-time transient detection pipeline. The pipeline is split into five main parts as follows:

1. Data reception.
2. Real-time GPU based time-domain search.
3. Candidate classification.
4. Transient trigger and voltage capture.
5. Database storage and data exploration.

Figure 7.1 illustrates an overview of the MeerTRAP real-time processing infrastructure. More information about the various steps are detailed at https://www.meertrap.org/about-1/real-time-transient-detection-pipeline/ and in the Jankowski et al. (2020); Rajwade et al. (2021) papers. There are various challenges in the real-time triggering pipeline illustrated in Figure 7.1. The first challenge is the software must deal with the processing of the candidate data in ≤ 10 seconds after it is received by the single pulse pipeline and in ≤ 40 seconds overall (Jankowski et al., 2020). The second challenge deals with the reduction of the number of single-pulse candidates caused by noise, radio frequency interference (N-RFI, mostly from planes, satellites, maintenance work, lightning), and from known radio sources, above a certain candidate noise level. The MeerTRAP team have developed a set of python tools that execute multi-beam clustering and source matching algorithms from given catalogues[†]. This combination will typically reduce the candidate rate by 70 to 90 per cent (Jankowski et al., 2020). After this step, a machine-learning classifier is anticipated to decrease this rate further to < 1 per cent once fully deployed.

Although MeerKAT is very sensitive, because it is an interferometer its field of view for one beam is small so MeerTRAP makes hundreds of beams. Hence, the candidate rate generated can be tens of thousands per day. Recently with the advent of state-of-the-art Graphic Processing Units (GPUs) - accelerated pipelines (e.g., `heimdall` [‡] (Barsdell et al., 2012) it is now possible to implement real-time FRB searches. However, these searches generate up to thousands of candidates per day and they are highly affected by a high false positive rate, due both to Gaussian noise and the presence of RFI. Hence, the last stage of visual inspection by a human becomes challenging and ultimately infeasible. It is imperative to reduce the sheer volume of candidates that require inspection. Therefore, various techniques are presently being implemented. These often include N-RFI mitigation techniques to remove DM = 0 pc cm^{-3} signals. k-Nearest neighbours as a clustering technique (Cover & Hart, 1967) has also been deployed to identify single, bright events that trigger many candidates (see Burke-Spolaor et al. (2011)).

However, the above-stated techniques cannot provide a real-time classification of candidates, for example as N-RFI, FRB or pulsars. Traditionally classification of candidates has been done visually by astronomers. However, this limits the ability to trigger real-time multi-wavelength follow-ups, and forces a requirement to record and store large data volumes. This is one of the challenge that the MeerTRAP real-time transient pipeline is facing. Machine learning has the potential to provide an automated solution to this problem.

[†]https://bitbucket.org/jankowsk/meertrapdb/
[‡]https://sourceforge.net/pro jects/heimdall-astro

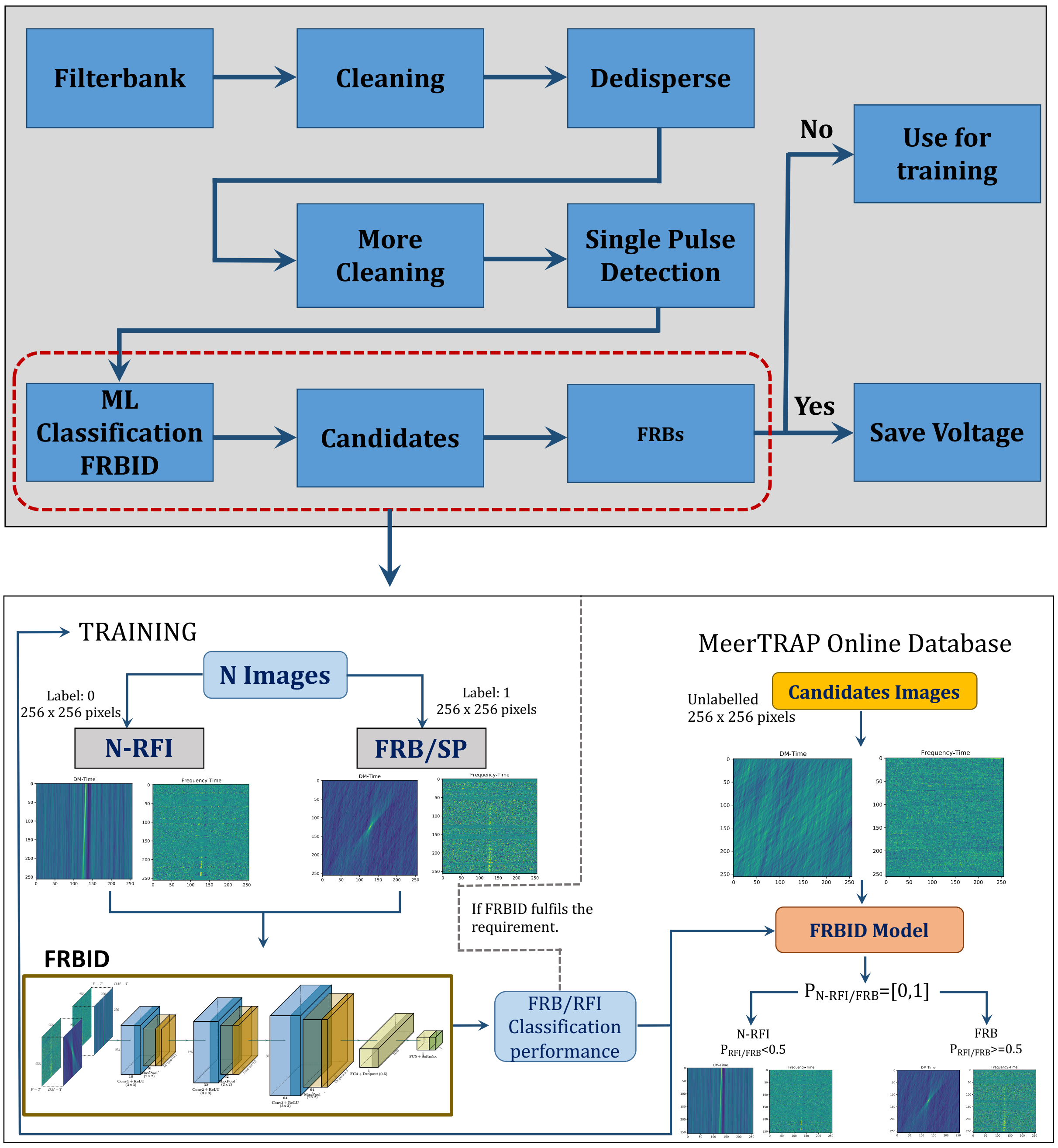


Figure 7.2 – An overview of various stages in single search pipeline and an illustration of the process of `FRBID` model for classification of FRB and N-RFI. First data is received from Filterbank Beam-Former User Supplied Equipment (FBFUSE). Once we receive the data, we have to deal with noise and RFI (N-RFI). A static frequency channel mask and an Inter-Quartile Range mitigation (IQRM) algorithm are applied to the data before the de-dispersion. This cleaning process removes corrupted data from the pipeline. Then the data is de-dispersed and is searched for single pulses using a standard Boxcar filter. Afterwards, a multi-beam clustering and source matching algorithm is implemented to reduce the candidate rate by 70 to 90 per cent. After this step, `FRBID` is anticipated to decrease this rate further to < 1 per cent once fully deployed.

Table 7.1.1 – Total Number of FRB and N-RFI candidates in our labelled dataset. Candidates marked as 'FRB' also include multiple detections of the same pulsar/FRB.

	N-RFI	FRBs	Total
Training Set	8235	7852	16087
Validation Set	2059	1963	4022
Testing Set	477	468	945

Applying a ML classifier at this stage has been inspired by the work of Agarwal et al. (2020). The latter developed a ML classifier based on a transfer learning approach for transient classification. We tried their publicly available software but found that some aspects did not match our data. Therefore, in this chapter, we adopted a similar approach to that presented in Chapter 6 and in Agarwal et al. (2020). We developed a state-of-the-art machine learning classifier, `FRBID`: Fast Radio Burst Intelligent Distinguisher, that will distinguish between single pulses, FRBs and N-RFI. In the next section, we discuss in detail the multiple stages in the MeerTRAP pipeline.

7.2 How does MeerTRAP search for single pulses?

The overview of the pipeline is described in Figure 7.2 and is detailed below.

1. First we need to receive data from Filterbank BeamFormer User Supplied Equipment (FBFUSE). FBFUSE is a separate compute cluster developed by the Max-Planck Institüt für Radioastronomie in Bonn, Germany. It generates 768 coherent beams and an incoherent sum of the power from all the available (maximum of 64) antennas, and these are received via the network interface cards (NIC) on each compute node (Rajwade et al., 2021). The data are sent over the network to MeerTRAP. Multiple modes are possible - we can observe in ultra high frequency (UHF) and L-Band (1-2 GHz) in the range of 1024-4000 channels, at multiple sampling times. Currently the real-time processing operates best at a sampling time of 360 μs at L-band. With 12 beams per node, the task is quite challenging as we can generate up to 2.4 GB of data per second per node. The challenge is that we need to receive the data, process it and output it in usual format, all in real time.

2. The next step is the cleaning process. Once we receive the data, we have to deal with noise and RFI. Even though MeerKAT is located in the desert, the N-RFI environment is quite dynamic because of planes, satellites, maintenance work, and lighting among others. To reduce the number of N-RFI detections, a static frequency channel mask was applied to the data before the de-dispersion and single-pulse search. Recently, a new N-RFI mitigation algorithm called Inter-Quartile Range mitigation (IQRM, Morello et al. in prep) has been implemented. These methods combined, allow the removal of most of the corrupted data from the pipeline (Rajwade et al., 2021). In addition, the 'standard' zero-DM excision (Eatough et al., 2009) is implemented to discard any remaining N-RFI that

was too broad band to be masked by the static channel mask or was not frequent enough to be detected by the IQRM algorithm.

3. The next stage is to de-disperse the data. As single pulse signals travel through the interstellar medium (ISM) before arriving at the telescope, their radio emission are 'dispersed' by the free electron content in the ISM. The amount of dispersion is proportional to the dispersion measure (DM). DM is described as the integrated column density of free electrons between the source (pulsar/FRB) and the observer. This generates a frequency-dependent delay in such a way that lower frequency signals arrive later in comparison to higher frequency signals. Since the true DM of a pulsar/FRB is not known beforehand, the data is de-dispersed into multiple trial values (trial-DMs, see §2.3.1.2). Therefore, the DM sweep needs to be removed from the data obtained after cleaning. Some decisions on the number of steps of DMs need to be made. DM steps need to be chosen in such a way that it is small enough that intra-channel smearing is not a major concern, but should be big enough so that we do not de-disperse unnecessary DMs. In this step, we end up with (tens of) thousands of DMs. This is the most intensive part of the process and has to be run on GPUs using a highly optimised ASTRO-ACCELERATE [§] algorithm.

4. The next process is to search for single pulses. A standard Boxcar filter (convolve the signal with a simple rectangular function - up to a maximum boxcar width of 0.67s) is used. And when we take the number of DMs and the number of waves, sometimes this leads to a few hundreds candidates per node in just a few minutes only.

5. Then, we use a first stage vetting. The candidates are clustered in the DM and time plane. Only the most significant candidates are kept.

6. Even after clustering or after the first stage vetting, there are still tens of thousands of candidates sometimes and we do not want to have scientists to look at those candidates one by one, and many of them are just N-RFI. And, per day we can get more than 100 thousands of candidates. This process can get tiring and inefficient.

7. An additional vetting also known as multi-beam clustering is used to remove common N-RFI and known sources. But even after that, we can still end up with 10,000+ candidates per day and this amount of data can take up 250TB archive storage in no time.

8. Therefore, we need an additional post-processing step that can sift through the data as

[§] https://github.com/AstroAccelerateOrg/astro-accelerate quickly and efficient as possible. Inspired by the work carried out by Agarwal et al. (2020), in this chapter we implement a ML classifier, `FRBID`, that can actually make a decision of what this candidate actually is, whether there is anything useful (is a pulsar/-FRB/RRAT) or do we want to just get rid of it.

The next section provides a description and processing of the data we used to train the `FRBID` machine learning algorithm.

7.3 MeerKAT Data and Data Labelling Description

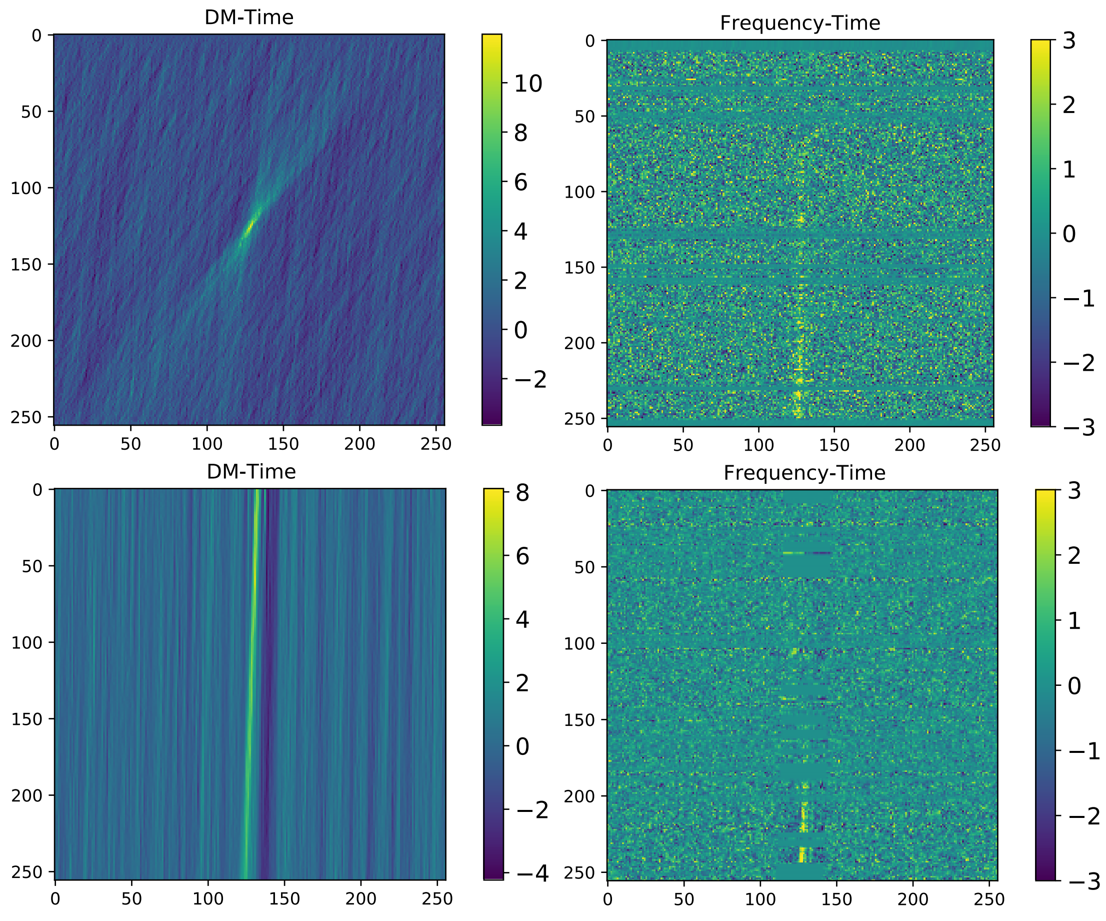


Figure 7.3 – Example images of high S/N candidates from the MeerKAT telescope. The top row shows an example of a single-pulse/FRB candidate, while the bottom row is an example of an N-RFI candidate. The first column corresponds to the DM-time image and the second column is the Frequency-time image of the candidates.

We used observations from the MeerKAT telescope at UHF band and L-band modes, to generate single pulse and N-RFI candidates. Single pulse candidates are obtained from known periodic pulsars (1ms - 30s), rotating radio transients (RRATs) and FRBs. The candidates have a dispersion measure range of 0-5000 pc/cc at L-band and 0-1490 pc/cc at UHF band. Also, the candidates have a width range from 0.3 to 300 ms with S/N threshold at 8 sigma. In this work, we use only two plots: the DM-time and Frequency-time plot. An example of DM and Frequency plots of a single pulse/FRB and N-RFI candidate is shown in Figure 7.3.

The MeerTRAP data is standardised such that `FRBID` becomes agnostic to observing frequency. Hence, de-dispersed data in the frequency-time spectrogram is utilised as input. This de-dispersion process makes the data to be independent of the true DM of the candidate and observation frequency. Afterwards, a resizing process is performed by averaging the frequency axis and removing the extra pixels, that is, the frequency-time is resized to 256×256 pixels. We chose 256 frequency bins to maintain the frequency modulation of recently reported FRBs (Shannon et al., 2018).

DM-time images are constructed by performing an averaging along the frequency-axis, after de-dispersing the frequency-time at different DMs. The DM range lies from zero to twice the DM of the candidate, spanning across 256 steps. The DM-time images are resized to 256×256 pixels. As can be observed in Figure 7.3, a typical DM-time image of a single-pulse/FRB candidate looks like a bow-tie shape centred around a non-zero DM value, while N-RFI is mostly a stripe in the DM-time plane and in this case peaks at DM=0. The extent of the bow-tie shape is dependent on the pulse width and S/N. The angle between the edges depends on the DM, the width of the candidate and its bandwidth. The filled area between the edges is controlled by the spectra of the candidate.

For training a supervised deep neural network, we need to provide accurate labels to the model. We therefore select candidates that were visually labelled by three vetters. We ensure to alleviate the imbalanced-learning problem by acquiring highly balanced dataset of single pulses/FRB and N-RFI. Since it is a supervised learning algorithm, we used the label '0' for N-RFI candidates and label '1' for FRB candidates. We assigned a label '0' if all three vetters labelled it as N-RFI and a label '1' is given to candidate where three vetters agree that it is a single pulse/FRB. For training and evaluating `FRBID`, we split the data into 76% training, 19% validation and 5% test set. The number of candidates for N-RFI and FRBs in the training, validation and test sets, are detailed in Table 7.1.1.

7.4 `FRBID`: Fast Radio Burst Intelligent Distinguisher for the MeerKAT telescope using Deep Learning

`FRBID`: Fast Radio Burst Intelligent Distinguisher, is a deep learning model based on a convolutional neural network (CNN, see §3.4.3). `FRBID` is developed based on `MeerCRAB` discussed as in Chapter 6. We developed two model architectures for the classification of FRB and N-RFI: `FRBID` and `Multi-Input FRBID`.

- `FRBID`

  In `FRBID`, the input (i.e. Frequency-time and DM-time) images are stacked together and a CNN model is trained using the stacked images as shown in Figure 7.4.

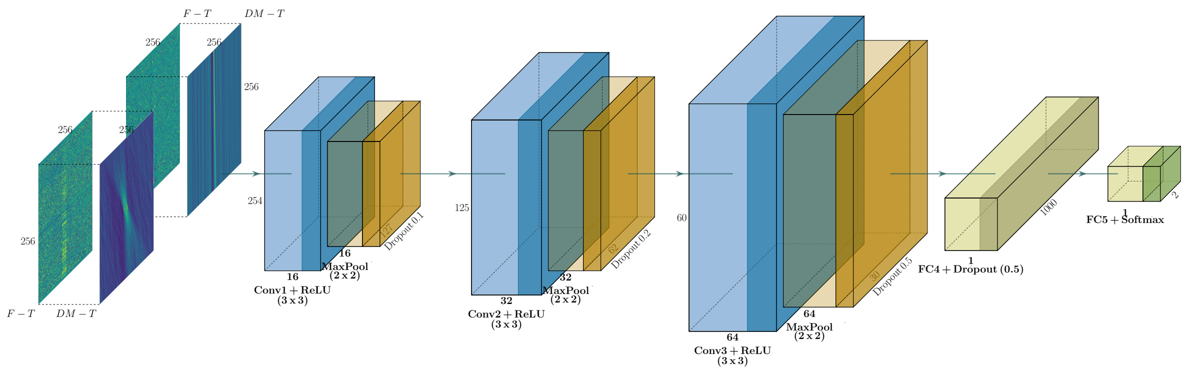


Figure 7.4 – Illustration of `FRBID` architecture. We show two images grouped together (DM-time and Frequency-time) to form the input of the network, followed by convolutional layers, max-pooling, dropout and dense layers. At the end, the network outputs a probability whether a candidate is either an N-RFI or SP/FRB during the prediction phase.

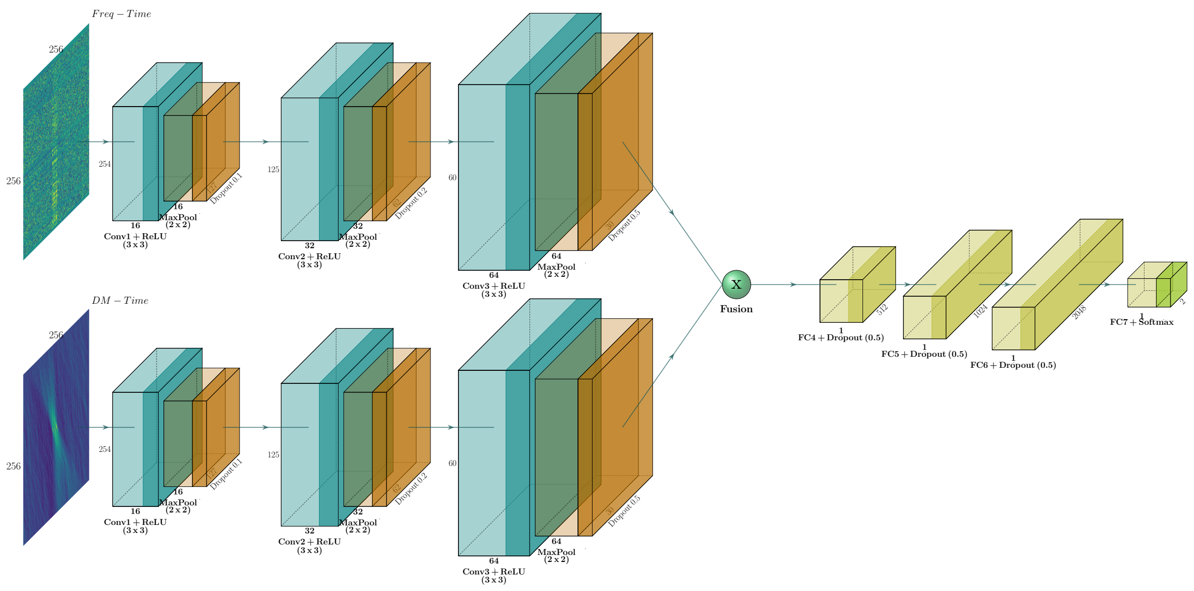


Figure 7.5 – Illustration of `Multi-Input FRBID` architecture. Here the frequency-time and the DM-time array are used as inputs to two separate convolutional neural networks. These models are then fused at then and perform a binary classification. Both models are trained at the same time and the overall model will learn the relative importance of each input separately.

- `Multi-Input FRBID`

  While in `Multi-Input FRBID`, two separate CNN models are trained on the Frequency-time & DM-time images respectively and once the Frequency-time & DM-time models are trained, they are combined at the end to obtain a more robust model. This process is also known as '*Network fusion/hierarchical hybrid neural network*'. The idea of using two separate CNN models has been inspired from Agarwal et al. (2020). Figure 7.5 illustrates the `Multi-Input FRBID` architecture. Two CNN models are concatenated after the feature extraction layers and then are later passed through a large fully connected layer and this results in a binary classification at the end of the model.

Both `FRBID` and `Multi-Input FRBID` models are based on three convolutional layers (CLs). Each CL is made of (3×3) pixel filters, together with a Rectified Linear Unit (ReLU) function, followed by a pooling layer with filter size of (2×2). After the CLs, we used fully connected layers (also known as dense layers). In addition, we use a dropout rate varying from (0.1 to 0.5) after each of the pooling and dense layers as seen in Figure 7.4 & 7.5 to avoid over-fitting. For the output layer, we used a softmax function that outputs a probability value between 0 and 1. Both `FRBID` and `Multi-Input FRBID` are built with `Keras` and `TensorFlow` as the backend. Both models are trained and tested on the same dataset and are compared in §7.5.

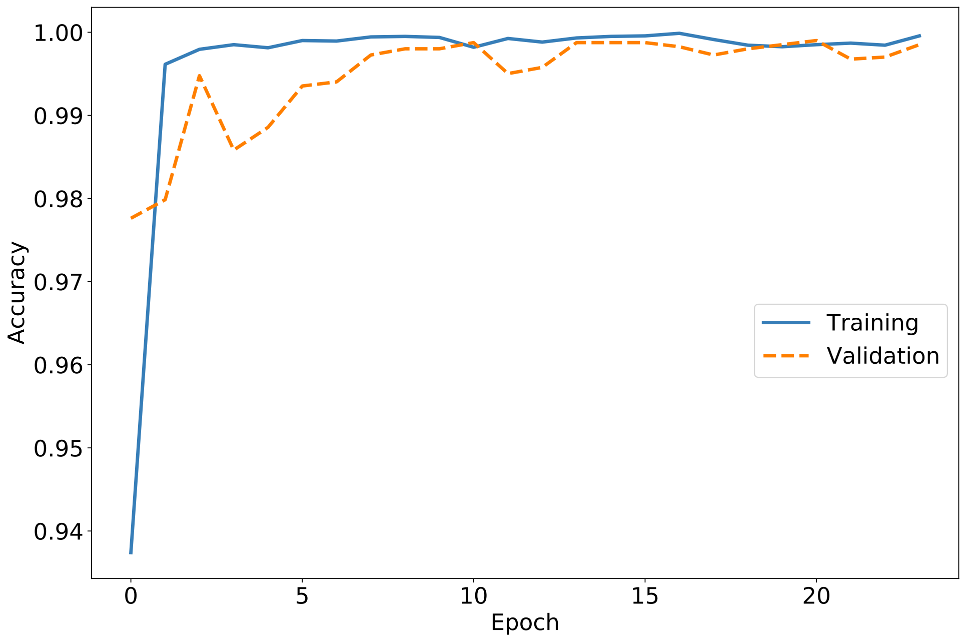
7.4.1 Training process

For training both models independently, we used an Nvidia GeForce GTX 1080Ti 11GB GPU. Both networks are trained and tested independently. We do not apply data augmentation to our data since we have sufficient examples of each class considered. We apply an early stopping technique (see §3.5.2) while training the networks. The training process continues until the validation accuracy stops increasing for at least 5 consecutive epochs. At this iteration, we considered this model to be the best representative model for prediction purposes.

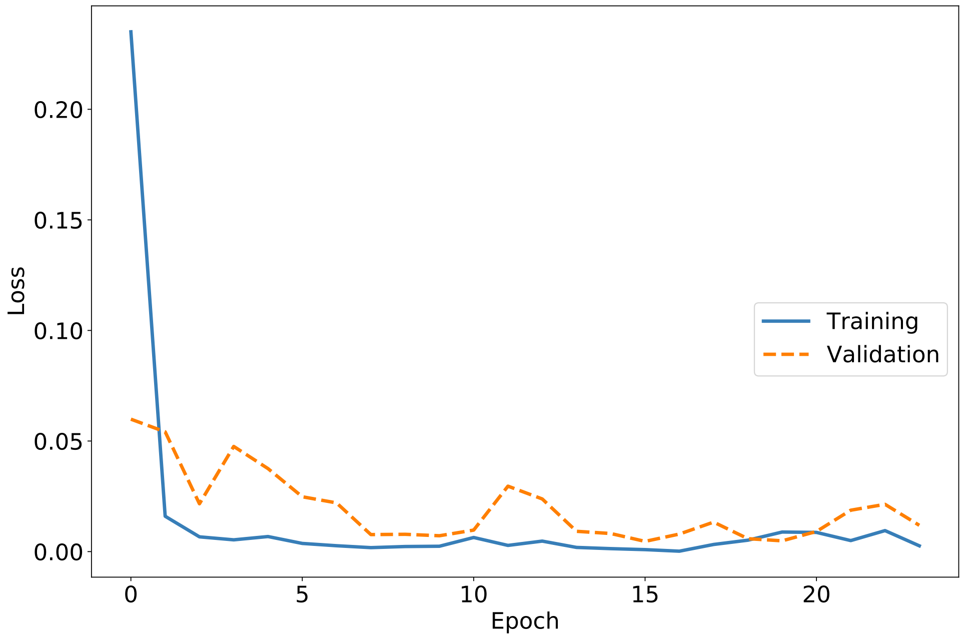
For training purposes, we employed the Adam optimiser (Kingma & Ba, 2014) with a binary cross-entropy loss function. The parameters used while training the models are: the learning rate is set to 0.0002, batch size of 64 and epochs 30. Figure 7.6 shows the optimisation curves of `FRBID`. As can be seen in Figure 7.6, `FRBID` stops training when the validation loss starts increasing.

7.4.2 Evaluation metrics

Various metrics are employed to evaluate the performance of the models. Our main goal is to have an algorithm that accurately identifies FRB and N-RFI candidates while minimising the false positives and false negatives. In this chapter, we have used balanced accuracy, the Matthew Correlation Coefficient, precision, and recall to analyse the performance of the classifier. These metrics are detailed in §3.2.3.



(a) Accuracy

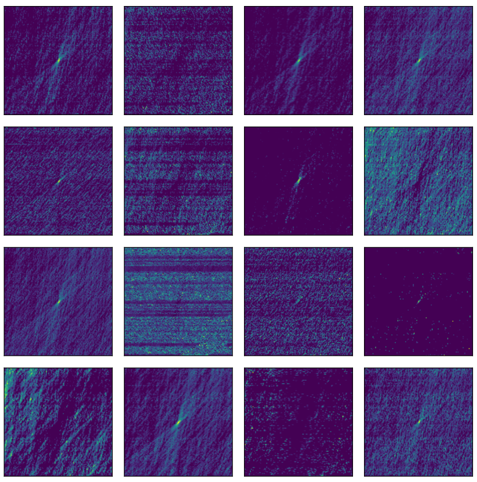


(b) Loss

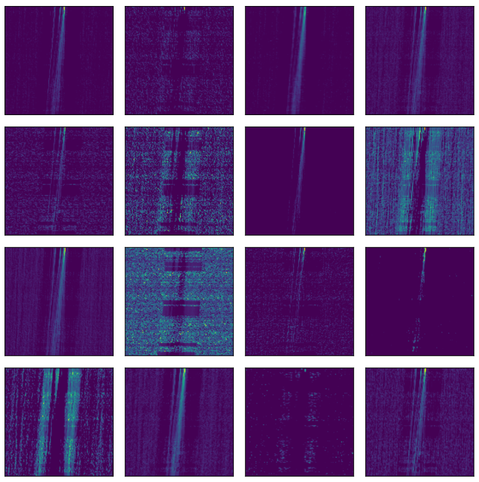
Figure 7.6 – Learning curves of the `FRBID` model. The upper panel shows the training and validation accuracy over iterations/epochs. The lower panel shows the change in negative log-likelihood/loss with epochs. It can be observed that the training objective decreases consistently over iterations, but at some point (around 20 epochs) the validation set loss eventually starts to increase again. An early-stopping technique with (*patience* = 8) is applied to avoid overfitting by terminating that training process. At this stage, the algorithm picks the best parameters at 20 epochs. *Patience* parameter is the number of epochs when no improvment is observed after which the training process will be stopped.

7.5 Results and Analysis

We start by providing an overall comparison of various scenarios used and will subsequently analyse some of the models along with certain modifications in more detail. The models are



(a) FRB feature maps



(b) RFI feature maps.

Figure 7.7 – Feature maps induced by the convolutional layer in `FRBID` given an FRB and N-RFI candidate, respectively. We observe that the model distinguishes between background noise and the bow-tie shape at the centre of the image.

Table 7.4.1 – The results with `FRBID` and `Multi-Input FRBID` are presented in terms of precision, recall, balanced accuracy and MCC values using the DM-time and frequency-time images as input.

Model	Precision	Recall	Balanced Accuracy	MCC
`FRBID`	1.00	1.00	0.999	0.998
`Multi-Input FRBID`	1.00	1.00	0.999	0.998

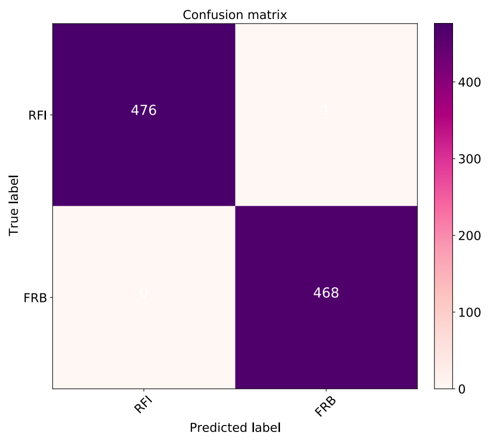
evaluated on various metrics. Based on the P_{FRB} value, we construct a confusion matrix to have an overview of the classification results.

7.5.1 Model Selection

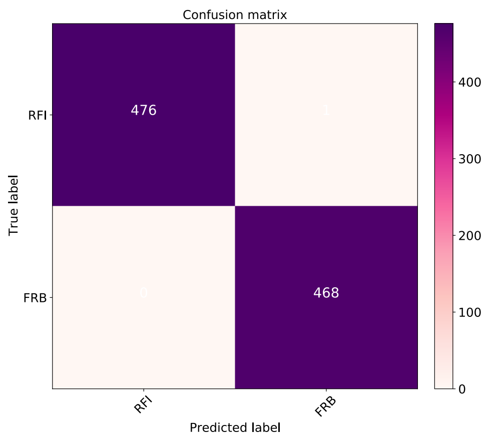
As discussed in the previous sections, we trained two models independently on DM-time images, and frequency-time images. We used the *validation accuracy* and *test accuracy* as metrics to decide the best performing model. We evaluated the performance of the models on an independent test data. This step demonstrates how well the models perform and generalise on real and unseen FRB and N-RFI data. The results are detailed in Table 7.4.1 and we note that with both `FRBID` and `Multi-Input FRBID` we obtain an accuracy and MCC values of ~ 99.8 %. Given that both models have comparable performance, therefore in the MeerTRAP pipeline, we integrated the `FRBID` as it requires less computational power, compared to `Multi-Input FRBID`.

7.5.2 Feature Extraction

The satisfactory performance of both models shows reasonable confidence that they have learnt features of both N-RFI and FRB candidates, based only on the frequency-time and DM-time



(a) `FRBID` model.



(b) `Multi-Input FRBID` model.

Figure 7.8 – (a) shows the confusion matrix obtained with `FRBID` model. (b) illustrates the confusion matrix obtained with `Multi-Input FRBID` model. Both models perform equally well with only one misclassification of N-RFI candidate.

images. Figure 7.7 illustrates the feature maps of an example of FRB and N-RFI candidate. We observe that there appears to be feature maps that activate on the background, while other maps activate on the bow-tie shape of the FRB candidate. This suggests that the network can distinguish between the morphology of the FRB candidate (bow-tie shape) and the background noise, thus it is able to classify images relatively unhindered by different levels of noise.

7.5.3 Analysis of Misclassification

Figure 7.8 illustrates the confusion matrices of both `FRBID` and `Multi-Input FRBID` models. The number of incorrect classifications of FRB & N-RFI and the number of correct classification of FRBs & N-RFI are reported. As noted in Table 7.4.1 and Figure 7.8, the rate of misclassifying N-RFI candidates is relatively low. The false positive rate varies between 0.015% and 3% (<1% most of the time). Both models show a single incorrect classification of N-RFI candidate and all FRB candidates are correctly classified. Figure 7.9 presents a single misclassification made by both models. This misclassification can be attributed to the fact that when viewed closely, a faint point source in the middle of the image can be observed.

7.5.4 Output of `FRBID`

During the prediction phase, the trained models are then used to output probabilistic predictions for unseen images in the test set as shown in Figure 7.10. The probability distribution from the output of the `FRBID` model spanned the range of $\mathrm{P}_{FRB} \in [0;1]$. Therefore, a candidate is predicted as N-RFI if $\mathrm{P}_{\mathtt{FRB}} < 0.5$ and as FRB if $\mathrm{P}_{FRB} \geq 0.5$. $\mathrm{P}_{FRB} = 0.5$ indicates a random guess and the `FRBID` model is confused between FRB and N-RFI candidates.

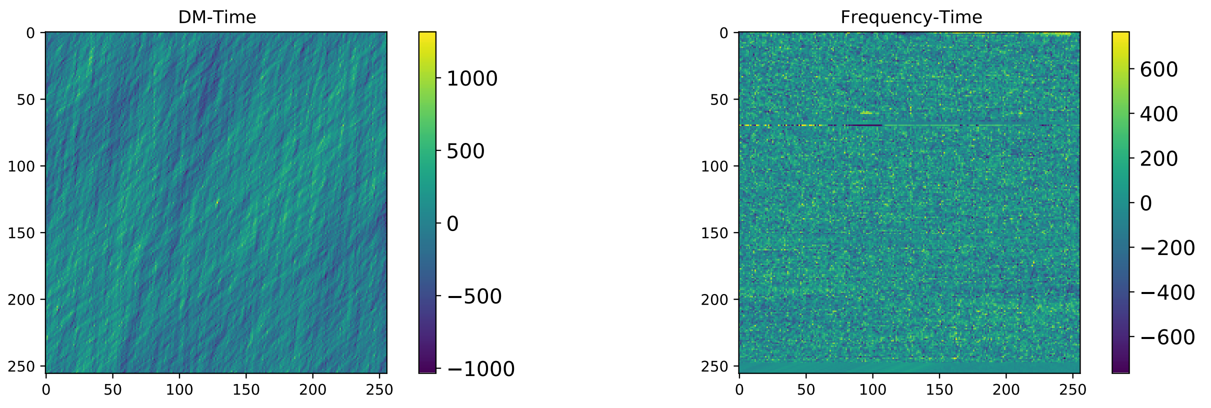


Figure 7.9 – Illustration of a misclassified N-RFI candidate by `FRBID` and `Multi-Input FRBID`. This candidate has been classified by a human as N-RFI. However, both models identified it as an FRB candidate. When looked at closely, a point-like source at the centre of DM-time image can be observed, thus this feature confused the networks.

7.6 Conclusion

We provide a user-friendly open-source python package `FRBID`: Fast Radio Burst Intelligent Distinguisher [¶], currently integrated in single pulse search pipelines MeerTRAP for real-time classification of candidates. Currently `FRBID` is being used in 'offline' mode and will soon move to 'real-time' mode. The input of `FRBID` is a candidate file with the DM-time and frequency-time data. The output of `FRBID` is the probability of a candidate to be an FRB in the range of [0;1]. We compared two model architectures in this chapter: `FRBID` and `Multi-Input FRBID`[||]. The latter trains two CNN models in parallel and it takes as input the DM-time and frequency-time images separately. At the end of the process, both models are concatenated and `Multi-Input FRBID` outputs the probability of a candidate being an FRB. For training `FRBID` and `Multi-Input FRBID`, it is recommended to use a balanced dataset, that is one with a comparable amount of FRB and N-RFI candidates.

We found that both architectures, `FRBID` and `Multi-Input FRBID`, perform best on our chosen metrics, achieving an accuracy of > 99.8% with a false positive rate of <1% most of the time. However, `Multi-Input FRBID` requires more computational resources since we are training two CNN models simultaneously. Given that `FRBID` is less computationally expensive, hence it is currently being integrated into the MeerTRAP pipeline for commensal FRB searches at the MeerKAT telescope. The performance of `FRBID` shows a false positive rate of less than 1%. Up till date, `FRBID` has detected more than half a dozen new single pulse candidates, even in the presence of N-RFI and follow-ups are still undergoing for these discoveries.

[¶]https://github.com/Zafiirah13/FRBID
[||]https://github.com/Zafiirah13/multi-input-frbid

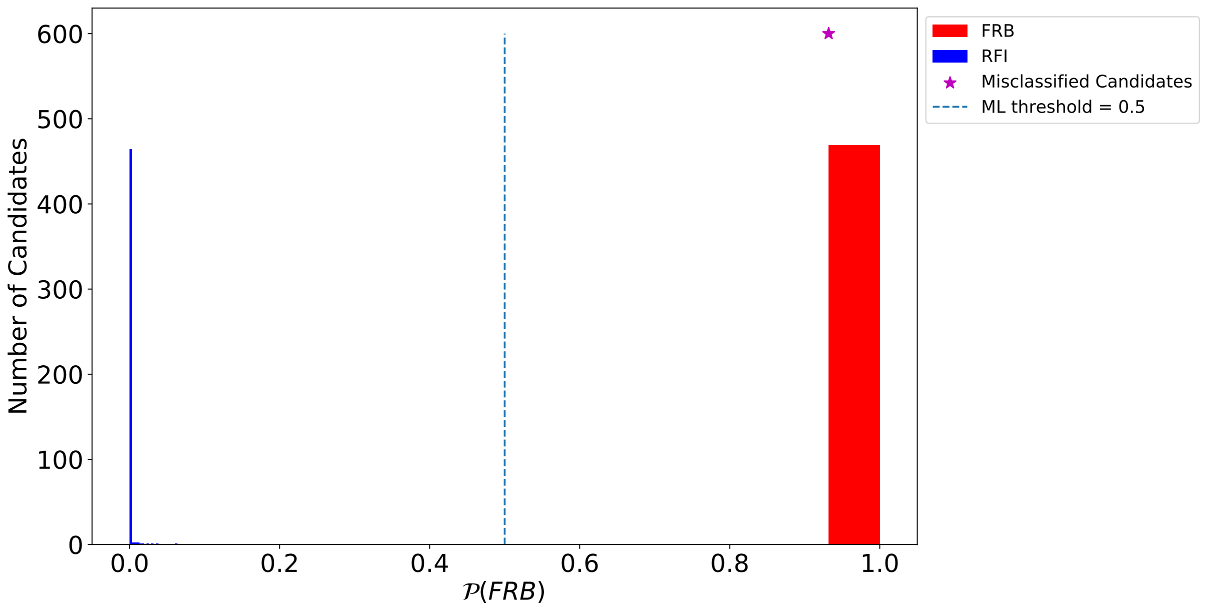


Figure 7.10 – Probability distribution on test data with 945 candidates, trained with model configuration `FRBID` with DM-time and frequency-time as input. The only one misclassified N-RFI candidate is shown in purple *.

8

FABLE: Fast rAdio Burst Localization & dEtection using Mask-RCNN

This chapter has been submitted as a conference paper in NeurIPS conference 2021 with the title: '`FABLE`: **Fast rAdio Burst Localization and dEtection using Mask-RCNN**'.

Zafiirah Hosenie, *Robert Lyon*, *Benjamin Stappers* et al.

Submitted in NeurIPS conference 2021 on **27 September 2021**.

This chapter provides a first proof of concept of an implementation of localization, detection and segmentation algorithm using Artificial Intelligence for Fast Radio Bursts or single pulses. The main aim of this chapter is to provide a general idea of detecting and localizing sources in astronomical images. This automated ML algorithm can be adopted to other domains in astronomy, for instance source extraction software. In addition, the analysis in this chapter was carried out well before Chapter 7. Hence, we use a different dataset obtained from Agarwal et al. (2020) and we did not attempt to provide a comparison between these two chapters. The following section presents the general idea of detecting and localizing FRBs using machine learning. Section 8.3.1 introduces the terminology and steps to undertake when implementing the '*Fast rAdio Burst Localization & dEtection using Mask-RCNN*': `FABLE` algorithm.

8.1 Introduction

We focus on improving the detection rate of Fast Radio Bursts (FRB), especially for low signal to noise ratio (SNR) in radio images. Recently, deep learning (LeCun et al., 2011, 1998; Li et al.,

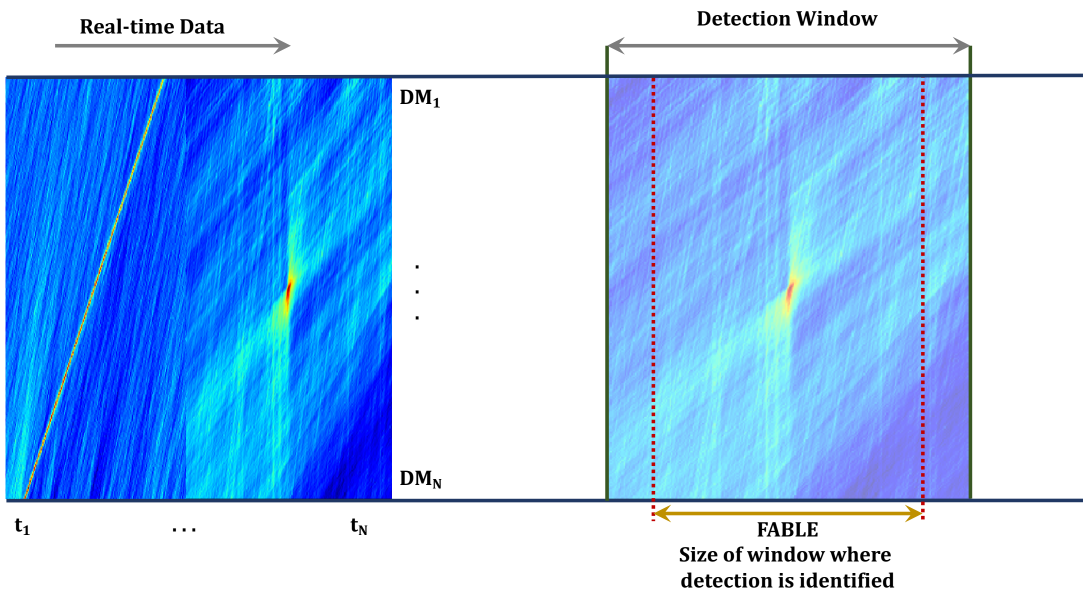


Figure 8.1 – The workflow of real-time Single Pulse/Fast Radio Burst detection, classification and segmentation. Real-time sequence of DM-time images will be the input to the pipeline. A detection window will be defined, for example 300 × 300 pixels and another smaller window where the detection of an FRB will be activated, for example 256 × 256 pixels. The output of the network will be the probability that the window contains an FRB, the location of the FRB in pixel coordinates, the DM and time value.

2018) methods such as the convolutional neural network (CNN) with hierarchical feature learning abilities has made great breakthroughs in various fields, for instance, for face recognition, in biomedical image analysis for cancer detection, and disease classification.

One method is to use detection and segmentation algorithms. They are useful to distinguish different sources in an image and draw bounding boxes over a specific source of interest. Mask Region Convolutional Neural Network (R-CNN) is one of the methods of object detection and segmentation. It can not only draw a bounding box for the target object, but also further mark and classify whether the pixels in the bounding box belong to the source or not. Mask R-CNN can be used to identify the source, mark the boundary of the source, and detect key points (Chiao et al., 2019), before classifying the source into other classes (whether it is a pulsar, variable stars, FRBs ...).

The aim of this chapter is to build a model, `FABLE`, based on Mask R-CNN for automatic detection, segmentation, and classification of FRBs with Dispersion Measure-Time (DM-T) radio images. The aim for this project is to detect FRBs in real-time without having to use a thresholding technique. In this chapter, we are dealing with static images. However, these images can be imagined as a sequence of images that we will access in real time. We want the `FABLE` algorithm to provide us with the probability that it is an FRB/Single Pulse and we also need the position of the FRB. The latter will allow us to work out the DM and time value of the candidate.

With `FABLE`, we can perform:

- **FRB detection**, giving us the (x,y) bounding box coordinates for each FRB in a candidate file,
- **Instance segmentation**, enabling us to obtain a pixel wise mask for each individual FRB in an image,
- **Output Dispersion measure of FRB** - calculate the dispersion measure and time based on pixel-wise mask for each individual FRB in an image.

The output (for instance the DM and time parameters) of `FABLE` is fundamental as it allows us to characterise the FRB candidate detected. We want to use the time to extract a piece of the original filterbank data and to de-disperse it to generate two further plots which astronomers will want to look at:

1. a de-dispersed pulse, and
2. the frequency - time plot.

In the next sections, we will elaborate on the data acquisition and process we used to annotate the labelled data. These annotation, labels and the DM-time images will act as input to the `FABLE` algorithm.

8.2 Dataset Preparation

In this section, we elaborate on the data we used and the technique we developed to automate the annotation step.

8.2.1 Dataset Used

One major problem when training supervised deep neural networks is the insufficiency of labelled data. We overcome this issue by using candidate files generated by Agarwal et al. (2020). We provide a brief summary of the data in this section. More information about the candidate files generation can be found in Agarwal et al. (2020) and the sample size used in this paper is detailed in Table 8.2.1. In this chapter we utilised data from observations using the Green Bank Telescope (GBT) and the 20m telescope both located at the Green Bank Observatory (GBO) (Agarwal et al., 2020). In order to generate a uniform dataset, Agarwal et al. (2020) used `Heimdall`[*] (Barsdell et al., 2012) with the following parameters on all the data in Table

[*]https://sourceforge.net/pro jects/heimdall-astro

Table 8.2.1 – Size of dataset used to train, validate and test the FABLE model.

Splits	Single Pulse
Training	17907
Validation	1121
Test	3357

8.2.1: S/N $\geq$ 8, 10 < DM < 10, 000 pc cm^{-3} and width < 32 ms. A brute force de-dispersion is performed to transform data from frequency-time to DM-time space. Each de-dispersed time series is baselined to zero mean and afterwards sliding box-car filters of various widths are implemented. The boxcar filtered time series is normalised to unit root mean squared deviation. Therefore, the peaks in the time series correspond to S/N. Afterwards, a threshold is utilized to select the candidates. The generated candidates were manually labelled. In this work, from Agarwal et al. (2020) paper, we used only 22 385 single pulse candidates.

8.2.2 Contouring and classification of FRB process

For training and validating the model, we need to provide the FABLE algorithm the original images and labels with exact indication of the types of objects present in the image, known as *annotation*. The process of annotation produces polygon objects which are coordinates of the location of the targets of interest. In our case, we are interested in separating FRB/Single pulses from the background noise in the DM-time image. The annotation is saved in the format of JavaScript Object Notation (.json). For annotating candidate files, we split our dataset of 22385 candidates into 80% training and 20% validation and test set. The 20% is further split into a 25% validation set and a 75% test set. The number of candidates available for training, validating and testing the FABLE model is detailed in Table 8.2.1. As noted in Table 8.2.1, the amount of available data is quite large. Hence the process of annotating the 256 $\times$256 pixels DM-time images for 22385 candidates will be time consuming and will require human labour. We therefore built an algorithm that will automate this labelling process. This is achieved by computing a *Model of Summed Dispersed Pulse* (MSDP) of the bow-tie shape for a single pulse (see §7.3 for detailed description). Hence, this annotation process is carried out automatically without involving a human in the loop.

8.2.3 Automatic annotation of FRB candidate

In this section, we elaborate on the process of automatically annotating an FRB candidate. This is based on finding an approximation of a de-dispersed pulse in the DM-time plane, which we call a *Model of Summed Dispersed Pulse* (MSDP). Analytically, we can assume a pulse depends on time, t, and in general can modelled as a Gaussian or a delta function and the dynamic spectrum of the dispersed pulse can be generated as a function of frequency and time, as shown in Figure 8.2. Because it is expensive to de-disperse the data for each given dispersion measure, we compute the MSDP that can be used to model each line in the DM-time plane using the dispersion equation. In Figure 8.3, we plot the amplitude of the analytical dispersed pulse (based on Gaussian assumption) and the MSDP (based on the dispersion equation) as a function of time, after de-dispersing at the given DM and summing across all frequency channels. We found that the analytical dispersed pulse (after summing and scrunching along the frequency channels) and MSDP are almost identical. We therefore conclude that we can use the MSDP to create an annotation/mask for FRB candidate. The description and derivation of the MSDP are detailed below.

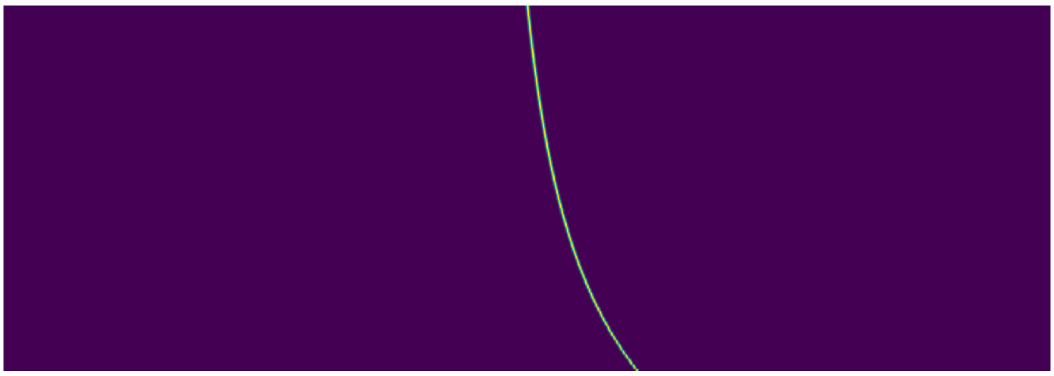


Figure 8.2 – An example of a dynamic spectrum of a dispersed pulse is shown in the plot, where the pulse can be assumed as a Gaussian or delta function as a function of frequency (y-axis) running from low to high and time (x-axis).

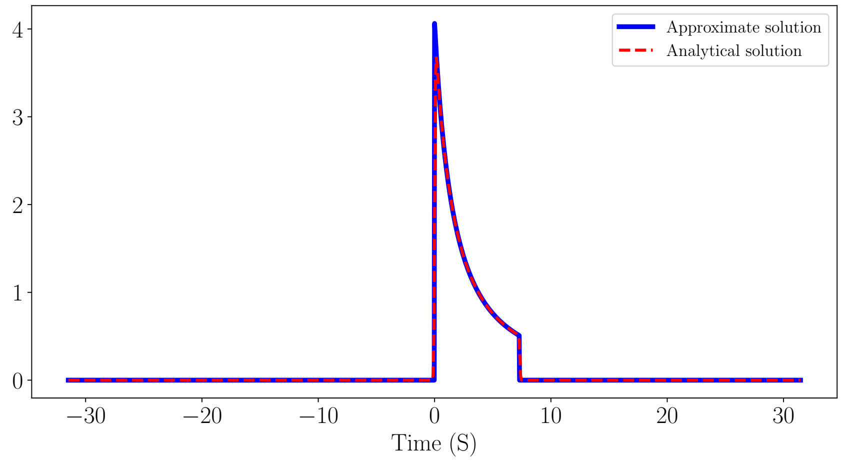


Figure 8.3 – Comparison of an analytical dispersed pulse and MSDP. The plot shows the amplitude of the analytical pulse (based on Gaussian assumption) and the derived pulse (based on the dispersion equation) as a function of time, after de-dispersing at the given DM and summing across all frequency channels.

- **Model of Summed Dispersed Pulse**

  Our goal is to generate a model of bow-tie shape of a pulse in the DM-time plane. In this section, we derive an approximation of the model of summed dispersed pulse (MSDP) based on the dispersion equation. We found that the derivative of the dispersion equation is a good approximation/model of the pulse in the DM-time plane as detailed below.

  Consider the dispersion equation;

$$t = 4149 \times DM \left[f^{-2} - f_0^{-2}\right] \tag{8.2.1}$$

  where f_0 is the reference frequency and the value 4149 is the dispersion constant, $K_{DM} = \frac{e^2}{2\pi M_e c}$. M_e and c is the mass of electron and the speed of light respectively. Using Equation 8.2.1, we solve it for f as follows:

$$f^{-2} = \frac{t}{4149DM} + f_0^{-2}, \tag{8.2.2}$$

$$\boxed{f = \left(\frac{t}{4149DM} + f_0^{-2}\right)^{-\frac{1}{2}}.} \tag{8.2.3}$$

  We then compute the derivative of f, which is almost similar to an analytical dispersed pulse when summing and scrunching along frequency channels. Taking the derivative of Equation 8.2.3 as follows:

$$\frac{\partial f}{\partial t} = -\frac{1}{2}\left[\frac{t}{4149DM} + f_0^{-2}\right]^{-\frac{3}{2}} \left(\frac{1}{4149DM}\right), \tag{8.2.4}$$

$$\partial f = \left(-\frac{1}{2 \times 4149DM}\right)\left[\frac{t}{4149DM} + f_0^{-2}\right]^{-\frac{3}{2}} \partial t, \tag{8.2.5}$$

  where $\partial t = \Delta t = t_{max} - t_{min}$ and therefore

$$\partial t = \frac{1}{f_{max} - f_{min}}. \tag{8.2.6}$$

  Replacing Equation 8.2.6 in Equation 8.2.5, we obtain the derived solution for the MSDP as,

$$\boxed{\partial f = \frac{\left| \frac{1}{2\times 4149DM} \left[\frac{t}{4149DM} + f_0^{-2} \right]^{-\frac{3}{2}} \right|}{f_{max} - f_{min}}.} \tag{8.2.7}$$

We found an approximation model of MSDP by taking the dispersion equation, $\left(\Delta t = 4149 \times DM \left[f^{-2} - f_0^{-2}\right]\right)$, solving the latter for f and then we compute ∂f. We found that the derivative is a good model that characterises the bow-tie in the DM-time plane. Using this concept, we are able to reconstruct the bow-tie shape of a pulse and this is further detailed in the next section.

8.2.4 Pseudo-code for Automatic Annotation of FRB candidates

We provide a pseudo-code on the process of generating the bow-tie shape of a single pulse candidate.

Step 1: Function to compute the derivative of frequency, f.

The following is a Python code to perform the derivative of frequency f using Equation 8.2.7 for the MSDP. The function takes as inputs an array of time (`t`), a reference frequency (`f0`), the true dispersion measure of the FRB candidate (dm) and the lowest and highest frequency (`fmin`, `fmax`). These variables are stored in the candidate header file.

```python
def derivative_frequency(t,f0,dm,fmin,fmax):
    t0   = 4149*dm*(fmin**(-2)-f0**(-2)) # Time limits
    t1   = 4149*dm*(fmax**(-2)-f0**(-2)) # Time limits
    tmin = np.min([t0,t1])
    tmax = np.max([t0,t1])
    # Derivative
    d = np.absolute((2*4149*dm)**(-1)*( f0**(-2)+t/(4149*dm))**(-1.5) )/(fmax-fmin)
    # Apply time limits
    d = np.where((t>=tmin) & (t<tmax),d,0)
    return d
```

Step 2: Load the variables from the candidate header file.

We load a candidate file stored in `hdf5` format. We read in the header file to have the appropriate information to compute the analytical and the MSDP shape. A filterbank file defines the frequencies as follows: `fch1` is the frequency of the top of the band. `foff` is the channel width, which will always be negative. So the frequency of channels are `fmax = fch1` and `fmin`

= fch1+nchan×foff. For de-dispersion we are free to choose the reference frequency (f0 in $\Delta t = 4149 \times DM \left[f^{-2} - f_0^{-2} \right]$). Most software packages choose the top of the band. The choice of this frequency changes the shape of the bow-tie. Here, we choose f0 = fmax. The pseudo-code shows how to extract the appropriate parameters describing the pulse in the candidate file. Here, f0 is the reference frequency used for de-dispersion. 't0' is the location of the bow-tie centre, and 'a' is the flux.

```
# Read the header information
with h5py.File(h5_file,'r') as f:
    filename = os.path.basename(h5_file) # The filename of the candidate
    labels = check_word_snr(filename) # The labels of the candidate: FRB
    dm_time      = np.array(f["data_dm_time"])
    if labels == "FRB":
        dm     = f.attrs["dm"] # in cm-3 pc
        tsamp  = f.attrs["tsamp"]; tstart = f.attrs["tstart"]; w = 2*tsamp # in s
        nchan  = f.attrs["nchans"]; nsamp = 2000
        fmax   = f.attrs["fch1"]; channel_width = f.attrs["foff"] # in MHz
        fmin   = fmax + (nchan*channel_width); f0 = fmax;
        t0     = 4149*dm*( (fmax**-2) - (f0**-2) )
        # Intensity of highest pixel
        a = dm_time[np.unravel_index(dm_time.argmax(),dm_time.shape)]
```

Step 3: Compute the derivative of frequency, f

In the code below, we compute the derivative of the frequency, f, as illustrated in Equation 8.2.7. We obtain a pulse shape when computing the derivative of f that looks similar to the analytical dispersed pulse when summing and scrunching along frequency channels. This is shown in Figure 8.3 when we plot 'za' and 'd' variables against 't' in the pseudo code below.

```
t     = np.linspace(-0.5*nsamp*tsamp,0.5*nsamp*tsamp,nsamp,endpoint=False) # time
f     = np.linspace(fmax,fmin,nchan,endpoint=False) # frequency
tm,fm = np.meshgrid(t,f) # 2D grid of frequency and time
dtm   = 4149*dm*(fm**(-2)-f0**(-2)) # Dispersion Equation
z     = a*np.exp(-0.5*((tm-t0-dtm)/w)**2)/(w*np.sqrt(2*np.pi))# Analytical
    dispersed pulse
# Computing derivatives of the analytical dispersed pulse and MSDP
za    = np.sum(z,axis=0)/nchan # Analytical pulse collapsed in freq space
d     = derivative_frequency(t,f0,dm,fmin,fmax)*a # derive dedispersed pulse
```

Step 4: Compute the bow-tie shape of the pulse.

The bow-tie shape can be obtained by using the following pseudo-code. The candidate file we used in this work is 256 $\times$ 256 pixels and the true DM is almost always found at the centre of the image. The 'mask' variable in the code below is the bow-tie shape of the pulse. The mask is obtained when the derivative of f is computed at different DM values. The middle panel in Figure 8.4 illustrates an example of the mask computed at different DM values. As we note, the algorithm has successfully reconstructed the bow-tie shape of the pulse in the real DM-time image plane (Figure 8.4 left panel). The mask variable creates a binary matrix with zeros and ones as shown in Figure 8.4. Zeros represent the background noise and ones illustrate the region where the pulse resides.

```
# Compute the mask
n  = nchan
time = np.linspace(-0.5*n*tsamp,0.5*n*tsamp,n,endpoint=False)
dm_values = np.linspace(2*dm,0,256)
mask     = np.zeros(n*n).reshape(256,n)
for i in range(len(dm_values)):
   mask[i] = derivative_frequency(time,f0,dm_values[i]-dm,fmin,fmax)
```

Step 5: Extract the contours of the bow-tie shape of the pulse

For the FABLE algorithm, we only need the contours of the bow-tie shape or the mask computed above. We first find the coordinates where the pixels > 0.

```
# Find the coordinates whose pixel values greater than 0
 snr   = np.where(mask>0.0)
 y_pix = snr[0]; x_pix = snr[1]
 ymin = []; ymax = []; x = []
 image = mask
 for i in range(image.shape[0]):
     white_pix = np.array(np.where(image[i,:]>0.0))
     if (white_pix.shape[1]==0):
         first_pix = (image.shape[0]/2)
         last_pix = (image.shape[0]/2)
     else:
         first_pix = min(white_pix[0,:])
         last_pix = max(white_pix[0,:])
     x.append(i/1.); ymin.append(first_pix); ymax.append(last_pix)
```

Then, for each channel, we find the coordinate of the first pixel > 0 and the coordinate of the last pixel > 0. We know from the data that the FRB is located at the centre of the 256×256 pixels. Therefore at the centre, the white pixel list would be zero, then we put the coordinate manually to be 256/2.

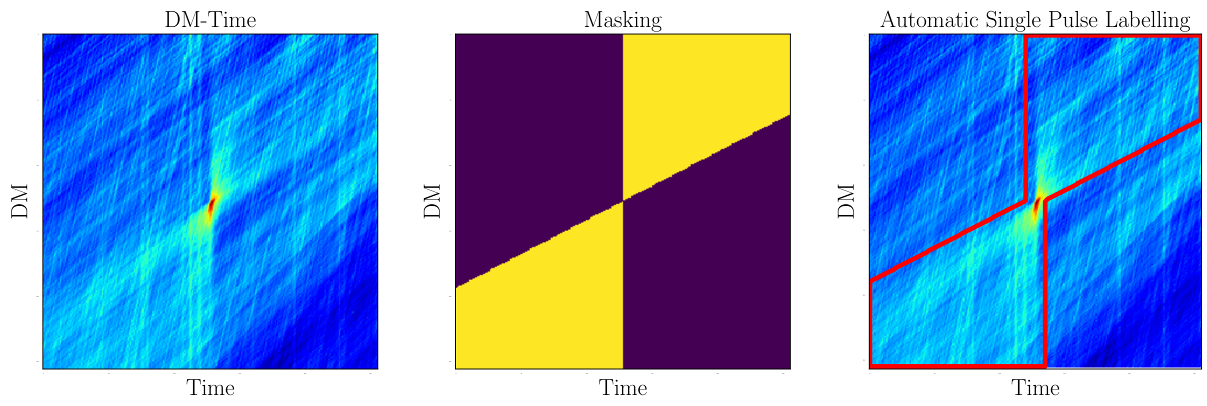


Figure 8.4 – Automatic annotation of single pulse candidate. The left panel shows the original DM-time image of a single pulse candidate. The middle panel illustrates the masking process of the bow-tie shape of the single pulse which is in a binary format. The right panel presents an overlay of a derived approximation of the bow-tie shape and this is shown as a contouring in red on top of the DM-time image. The polygon region indicates the region of interest and is an approximate computation of the bow-tie shape to facilitate the annotation process.

```
shift_pixel = 8
ymin[:]    = [ymin_ - shift_pixel for ymin_ in ymin]
ymax[:]    = [ymax_ + shift_pixel for ymax_ in ymax]
# we add all the x-pixel and y-pixel. We reverse the right x-pixel and y-pixel of
    the FRB, so that the list become continuous and then add the first coordinate
    of the left part of the FRB to complete the loop.
 time_coor = x + x[::-1] + [x[0]]
 dm_coor  = ymin + ymax[::-1] + [ymin[0]]
 # When performing the shifting of the pixel to the left and the right, the mask
     will become larger than the image size, so we need to crop those pixels<0 and
     >255 by the following command:
 dm_coordinates = []
 for l in dm_coor:
     if l < 0: dm_ = 0
     elif l > 255: dm_ = 255
     else: dm_ = l
     dm_coordinates.append(dm_/1.)
```

After deriving the MSDP, we obtain the bow-tie shape of the pulse in DM-time space. We then convert the DM-time space into pixel coordinates and we shift the right and left extent by 8 pixels so that we can capture full the entire FRB bow-tie shape without losing information. When performing the shifting of the pixels to the left and the right, the mask will become larger than the image size, so we need to crop those at pixel locations < 0 and > 255. The contours after applying an 8 pixel shift and cropping are illustrated in Figure 8.4.

Step 6: Store the contouring coordinates of the bow-tie shape in `JSON` format.
We need to save those contouring coordinates of the bow-tie shape of the pulse candidate in a `JSON` file. In addition, we store the filename, the size of the file, the x-y coordinates of the bow-tie and the label of the candidate (for e.g in this case 'SP'). This file is known as the '*annotation*' file of the data. The code to prepare for the annotation file in `JSON` format is shown below.

```
if h5_file.endswith('.h5'):
    size = os.path.getsize( h5_file)
    filesize = str(size)
    filename = h5_file[len(images_folder_name)::]
    img_annotation_key = filename + filesize
    x_coor = dm_coordinates; y_coor = time_coor; id="SP"; x = x_coor; y = y_coor;
    region =
        {'shape_attributes':{'name':'polygon','all_points_x':x,'all_points_y':y},\
             'region_attributes':{'name':id}}
    img_annotation =
        {img_annotation_key:{'filename':filename,'size':size,'regions':[region]}}
```

8.3 FABLE algorithm based on Deep Learning

In this chapter, we employ an instance segmentation algorithm to identify a 'single pulse' from background noise. Compared to other computer vision task, it is the hardest possible task. We want the algorithm to output the location of the FRB/SP, the probability it is an FRB/SP, and the pixels that belong to the candidate.

8.3.1 Mask R-CNN Methods

We chose Mask R-CNN (He et al., 2017) to implement detection, segmentation and classification of candidate files due to its simplicity and effectiveness (Pešek, 2018). Mask R-CNN is a convolutional neural network technique and is an extended algorithm based on Faster R-CNN (a fast and effective segmentation algorithm for object detection (Ren et al., 2015)). The stages of Mask R-CNN for processing candidate files is illustrated in Figure 8.5.

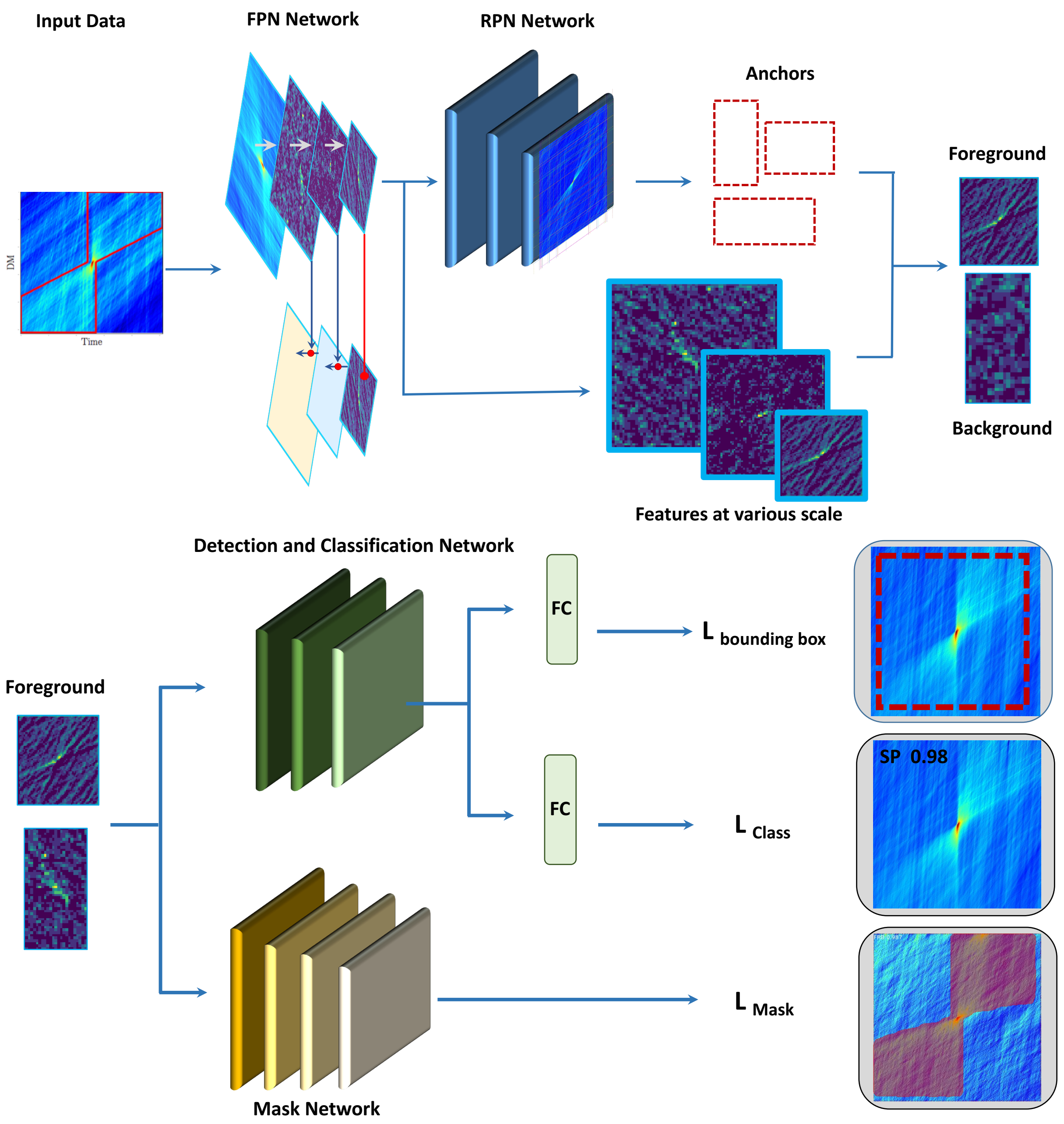


Figure 8.5 – The workflow of instance FRB/SP detection, classification and segmentation. There are four modules: Feature pyramid network, Region proposal network, Detection network and Mask network. The DM-time image and the annotation (contour coordinates of the bow-tie shape along with its label) act as input to the FPN. The FPN acts as a feature extractor and the feature maps act as input to RPN. The latter scans region of interest (RoI) and outputs anchors. The final proposal anchors act as input to the detection and classification network. The last stage predicts the class of the object in the RoI and refine further the size and the location of the bounding box to encapsulate the object.

The Mask R-CNN algorithm is a two-stage methodology. (I) Mask R-CNN creates proposals (area likely to contain the object), that is, it generates bounding boxes of the candidate file after scanning the image; (II) It also classifies the bounding boxes as FRB/SP, outputs binary mask for the region of interest (RoI) and the class of the detected candidate (He et al., 2017).

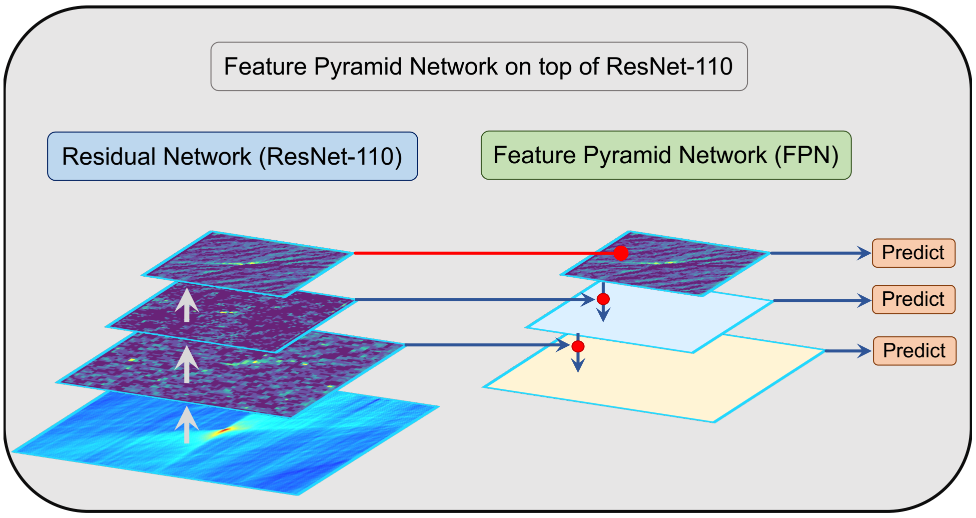

Feature Pyramid Network on top of ResNet-110
Residual Network (ResNet-110)
Feature Pyramid Network (FPN)
Predict
Predict
Predict

Figure 8.6 – Illustration of the Feature Pyramid Network. FPN consists of two pyramid networks. The second pyramid network uses high level features from the first pyramid network and passes them down to lower layers. This allows better representation of objects at multiple scales.

The structure of Mask R-CNN is described as follows:

- **Feature Pyramid Network**

  Feature Pyramid Network (FPN, Lin et al. (2017)) is made up of a backbone architecture, in our case, we use Residual Learning Network (ResNet) (He et al., 2016), for feature extraction. This is a standard convolutional neural network (CNN). The top layers detect low level features (edges) and the lower layers successively detect higher level features (SP). The candidate files pass through the backbone architecture where the image is converted from 256×256×3 to a feature map of shape 32×32×512. The feature will act as input for the following stages in the Mask R-CNN network. FPN is a standard feature extraction pyramid by adding a second pyramid network to achieve better representation of objects at multiple scales. The second pyramid network takes high level features from the first pyramid (the backbone architecture, ResNet101) and passes them down to lower layers. This enables features at all levels to have access to both higher and lower level features. The feature pyramid network is illustrated in Figure 8.6.

- **Region Proposal Network (RPN)**

  RPN is a lightweight convolutional neural network that scans the candidate image in a sliding-window way and locates areas of interest (that is the areas that contain the FRB/SP). These regions that the RPN scans over are known as *anchors*. Anchors are bounding boxes of various sizes and aspect ratios, distributed over the candidate file/im-

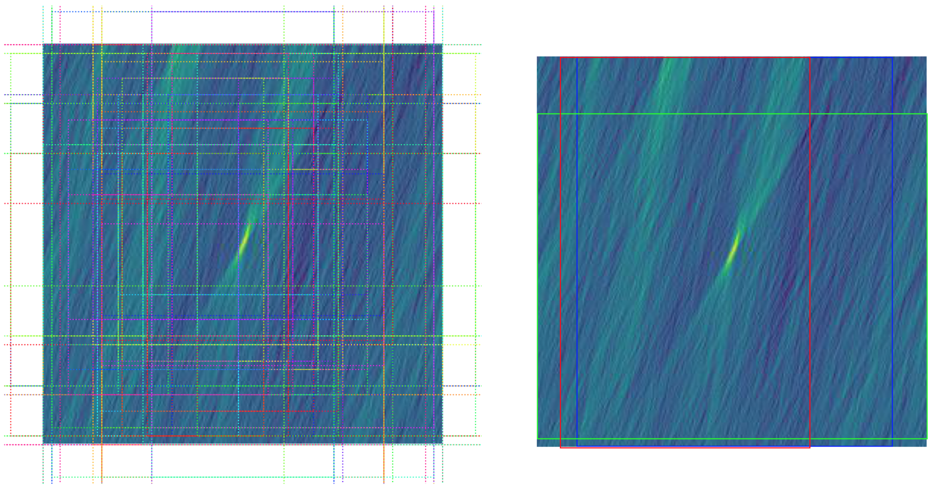


Figure 8.7 – The Region Proposal Network (RPN) runs a lightweight binary classifier on a lot of boxes (anchors) over the image shown in the left panel. Then the intersection of union (IoU) metrics of the anchors with ground truth object is computed. Positive anchors are those that have an IoU ≥ 0.7 with any ground truth object, and negative anchors are those that do not cover any object by more than 0.3 IoU. To train the RPN regressor, the shift and resizing needed are computed to make the anchor cover the ground truth object completely as illustrated in the right panel. Anchors with high objectness score (positive anchors) are passed to the stage two to be classified.

age as shown in Figure 8.7. The sliding window scanning is carried out by the convolutional nature of the RPN and this process is handled in parallel on a GPU. This process is fast since the RPN scans over the backbone feature map instead of the image directly, thus allows the reuse of the extracted features from the backbone architecture. Two outputs are generated from the RPN for each anchor: (i) the Anchor class- either of the two classes: background (BG) or foreground (FG). The FG class means that there is likely an FRB/SP in that box; (ii) Bounding Box Refinement: a FG anchor (also known as positive anchor) might not be centred perfectly over the FRB/SP. Therefore, the RPN computes an estimate of a percentage change in x, y, width and height to refine the anchor box to better fit the FRB/SP. Then, using the RPN prediction, the top anchors containing the FRB/SP are chosen and their sizes and locations are refined. If there are too much overlapping between anchors, the anchor with the highest foreground score is kept and the rest are discarded. This process is known as *Non-max suppression*. At the end of the stage, the final proposals anchor (region of interest) are passed to the next stage (see Figure 8.7 and Figure 8.8 for illustrations).

- **RoI classifier and Bounding Box Regressor (BBR)**

  This process works on the regions of interest (RoIs) proposed by the RPN network. Similar to the RPN, it generates two outputs for each RoI: (i) Class: It predicts the class of the object in the RoI. This network is deeper and has the ability to classify regions to specific

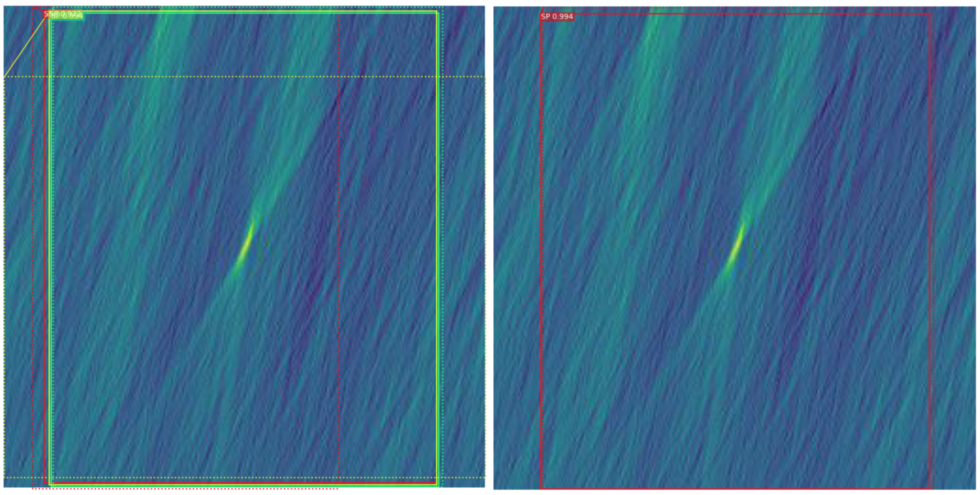


Figure 8.8 – Illustration of the proposal classification of anchors from the RPN network. The classifier head runs on proposals to generate class probabilities and bounding box regressions. Out of 1000 ROIs, we found three valid proposals, that is 997 are classified as background and 3 ROIs are classified as 'FRB/single pulse'. Then the ROIs which show confidence less than 0.5 are removed and this is illustrated in the left panel. Finally, a non-max suppression is applied per-class and the ROI with the best confidence is kept as shown in the right panel.

class, in this case, FRB/SP or background class. (ii) Bounding refinement: It refines further the size and the location of the bounding box to encapsulate the object. One problem encountered by the RoI classifier is that it requires a fixed input size. However, due to the bounding box refinement step in RPN, bounding boxes have different sizes. Therefore, an important step is necessary to refine the feature map to a fixed size, also known as *RoI Alignment (RoIAlign)* (He et al., 2017) or *RoI pooling*. It refers to as cropping a region of a feature map, apply a bilinear interpolation and resize it to a fixed size.

- **Segmentation Masks**

  This stage involves a convolutional network that takes as input the positive regions selected from the RoI classifier and constructs masks for those regions. The mask branch is implemented with 4 convolutional layers with kernel size 3×3 and 256 feature maps for each layer. The binary cross entropy loss is used to train the mask network. During training, these masks are scaled to low resolution (28×28 pixels) to compute the loss and during inferencing, the predicted mask are scaled up to the size of the RoI bounding box and this gives us the final masks, one per object/class. An illustration of the predicted mask is shown in Figure 8.9.

8.3.2 Implementation of FABLE

FABLE is based on the Mask R-CNN algorithm. As input to the FABLE model, we used 256×256 pixels DM-time images and we provide a JSON file consisting of the annotation (region of the bow-tie shape and its labels)

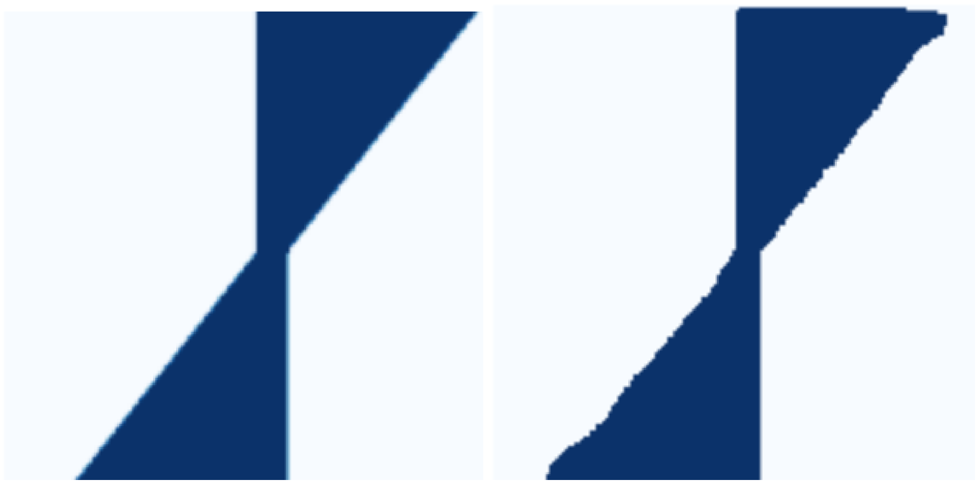


Figure 8.9 – Illustration of the segmentation mask. The training target for the mask branch is illustrated in the left panel. This stage takes the detections (refined bounding boxes and class IDs) from the previous layer and runs the mask head to generate segmentation masks for every instance. The predicted mask is shown in the right panel.

of each candidate file used for training and validation. We implement the Mask R-CNN algorithm using an open-source package built on Tensorflow and Keras, which is publicly available on Github: Mask R-CNN for Object Detection and Segmentation[†]. We conducted various experiments on a server equipped with Nvidia GeForce GTX 1080Ti 11GB GPU. In the training process, the GPU was used to train the adopted ResNet-101 backbone with a mini-batch size of 2 images, 100 steps per epoch, learning momentum of 0.9, and a weight decay of 0.0001. We modified the loading dataset function for our customized training data and left other parameters at the default settings. Instead of training the Mask R-CNN from scratch, we apply a transfer learning approach where we trained our model using the pre-trained weights for the COCO dataset (http://cocodataset.org/#home) as the latter has a large amount of training data for the Mask R-CNN to learn common and discriminative features. We then train our network in several stages using stochastic gradient descent (Kingma & Ba, 2015), where the latter updates the weights (model parameters, w_j) by minimizing the loss function $J(w)$,

$$w_{j+1} = w_j - \lambda \frac{\partial}{\partial w_j} J(w_j), \tag{8.3.1}$$

where λ is the learning rate. λ is a fine-tuned hyperparameter used to prevent the model from falling into 'local minima' and hence achieves convergence rapidly. At first, we train the head layers of the model with $\lambda = 10^{-3}$ for 30 epochs. We then re-trained the full model with all layers with $\lambda = 10^{-4}$ for 45 epochs and then progressively decreasing the learning rate from

[†]https://github.com/matterport/Mask_RCNN

$\lambda = 10^{-5}$ to $\lambda = 10^{-6}$ for 80 epochs.

In order to ensure the accuracy and stability of the model, we utilized a set of validation datasets fed to the most generalized Mask R-CNN. The value of the loss function $L = L_{\text{class}} + L_{\text{box}} + L_{\text{mask}}$, in Mask R-CNN was minimized, and the most suitable model through the minimization of the loss function on the training data was used as the best model. We record the full learning curve of the model as shown in Fig. 8.10. As can be seen from the plots, progressively adjusting the learning rate makes the training and validation loss values consistent with each other and reached their lowest at 80 epochs without overfitting.

In the inference/testing process, we use the optimized, trained model and set a detection confidence threshold at 50% (i.e., detections with confidence less than 50% were ignored).

8.4 Metrics for performance evaluation of `FABLE`

In this chapter, we used different evaluation metrics to evaluate `FABLE` compared to Chapters 4, 5, 6 & 7. The metrics are based on Intersection over Union (IoU) and Average Precision (AP).

8.4.1 Intersection over Union (IoU)

Intersection over Union (IoU) is commonly used in the evaluation of semantic segmentation and is simply a normalized intersection of the ground-truth and the detected segmentation. More specifically, given gt_i and dt_i to be the ground-truth and detected segmentation, respectively,

$$\mathrm{IoU}_c\left(\hat{y}, y^*\right) = \frac{\mid \{\hat{y} = c\} \cap \{y^* = c\} \mid}{\mid \{\hat{y} = c\} \cup \{y^* = c\} \mid}; \tag{8.4.1}$$

$$= \frac{\text{area}\left(gt_i \cap dt_i\right)}{\text{area}\left(gt_i \cup dt_i\right)}$$

where c is either the FRB/SP or background class, $y_i^* \in \{1, 2\}$ is the ground-truth label of pixel i. $y_i^* = 1$ means pixel i is in the FRB/SP area while for $y_i^* = 2$ means pixel i is in the background area. $\hat{y}_i \in \{1, 2\}$ is the prediction for pixel i of whether it belongs to class c. IoU is fundamental when we want to evaluate the overlapping percentage of two objects. It is also used in the next subsection when we compute the average precision.

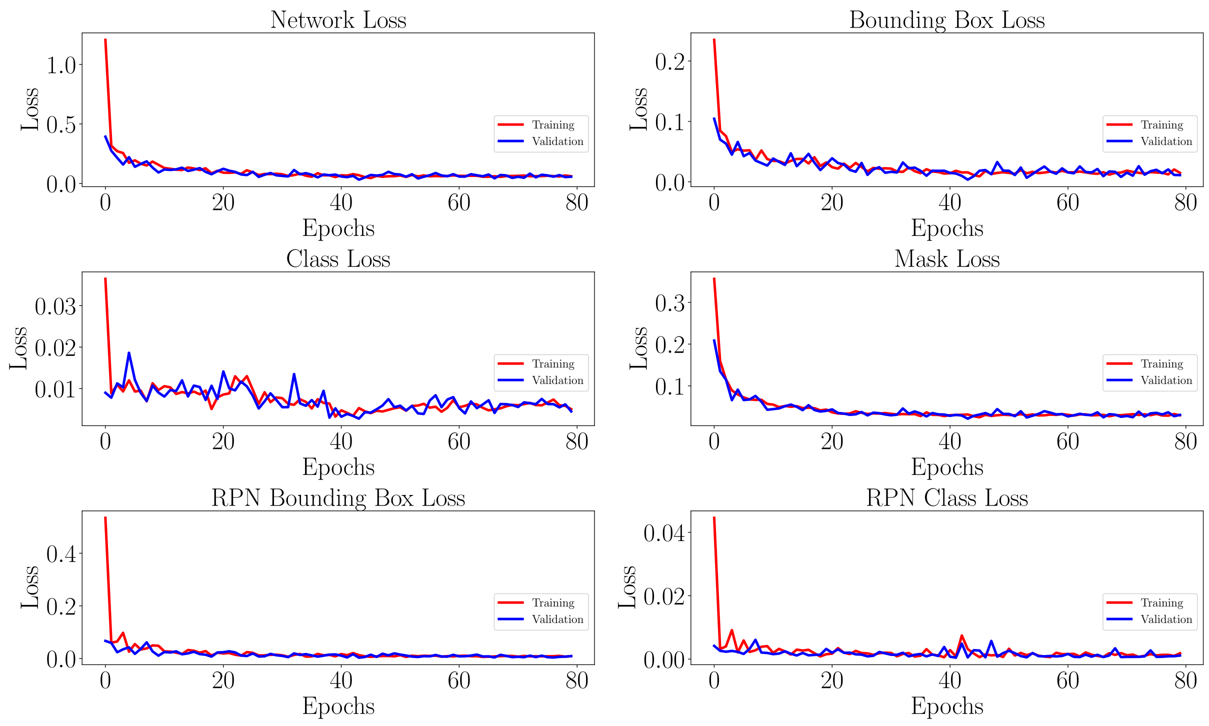


Figure 8.10 – The loss functions for `FABLE` optimization. The loss values for the different networks in `FABLE` during training and validation process are recorded. From the plots, using progressively decreasing learning rate help the optimization process.

8.4.2 Average Precision (AP)

For the detection and classification task, there are two stages we need to carry out:

1. Determine whether the object of interest (FRB/SP) exists in the candidate file/image;
2. Find the corresponding location of the object of interest if it exists.

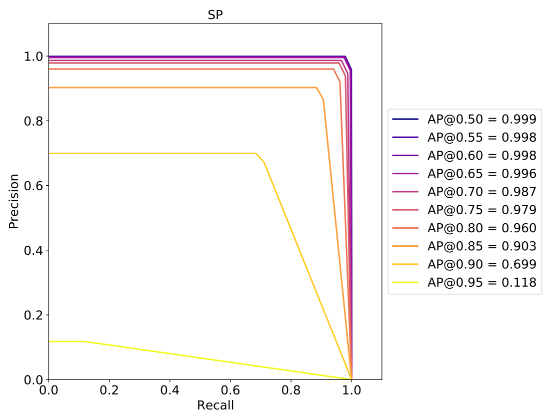
The IoU serves as a good indicator for segmentation, however it will not quantify how accurate our detection results are. Therefore, we use the *average precision* (AP) to measure the performance and quality of our detection algorithm. Let us assume there are two sets: set of labelled objects S_l and set of predicted objects S_p. Given an IoU threshold T_{IoU}, if the object labelled in S_l matched the object prediction in S_p such that IoU(S_l, S_p) > T_{IoU}, then this is either considered as True Positive (TP, if the label = 1) or True Negative (TN, if the label = 2). For instance, when the threshold is set to 0.6, for each detection i that $T_{IoU_i} \geq 0.6$, the detection counts as a true positive or true negative.

Average Precision, or AP, is an approximation of the area under the curve of *precision* (P = $\frac{TP}{TP+FP}$) against *recall* $\left(R = \frac{TP}{TP+FN}\right)$. As the model progress in its classification of objects, recall always increases by occasional incorrect classifications, setting recall as the x-axis and monitoring the relative changes of precision could be summarized by the area under the curve.

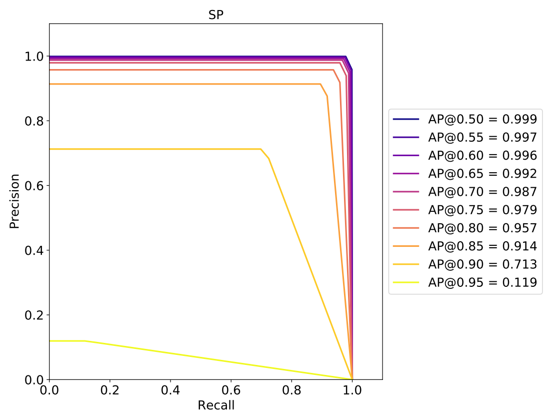
This value could then be averaged over all categories and depending on the chosen $t \in T_{IoU}$, it can be denoted by $AP^{IoU=t}$,

$$\boxed{AP = \frac{1}{N_{test}} \sum_{t \in \{0,0,01,\ldots,1.0\}} p(t).} \tag{8.4.2}$$

We average the precision–recall curves and mean AP scores for all N_{test} images in the test data set. This procedure is then done for IoU thresholds $T_{IoU} \in \{0.5, 0.55, \ldots, 0.95\}$.



(a) Precision-recall curves on validation set



(b) Precision-recall curves on test set

Figure 8.11 – Precision-recall curves and mean AP scores calculated at varying IOU thresholds averaged over the validation and test data set for FRB/SP. We find IOU = 0.5 gives a mean AP score of 99.9.

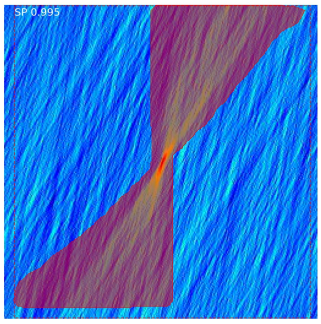
8.4.3 Model Optimization and Accuracy of `FABLE`

We conducted a two-step assessment for the trained `FABLE` model. First, we assessed the average precision (mAP: the mean of average precision values of each class) of the trained `FABLE` model with the hold-out validation dataset (comprising of FRB/SP candidates). Second, we make inference of the trained `FABLE` model on a test data.

8.5 Analysis and Discussion of `FABLE` results

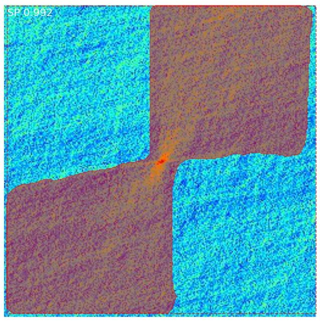
To validate our trained `FABLE` network, we test its performance against DM-time images from the validation and test data set. The test set is not used during the training process, thus it can be utilized to provide an unbiased estimate of the performance of `FABLE`. The validation is used for optimizing the network while training the network.

To quantify the performance of our `FABLE`'s classification capability, we calculate the precision and recall for each image in the validation and test data set. A detection is considered positive if the detection confidence is greater than a given threshold. In this work, we varied cand_tstart_58146.525462999998_tcand_37.4414824_dm_2140.15372_snr_-1.00000.h5



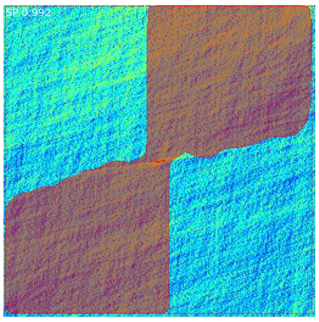
```
Detected Class:SP
Probability FRB: 0.995
Calculated DM: 2140.154 ± 33.440
Calculated time: 58148957.113
Brightest Pixel: 11.002
Brightest Pixel at position:(128, 129)
```

cand_tstart_58153.503043999997_tcand_261.2492582_dm_2737.43912_snr_-1.00000.h5



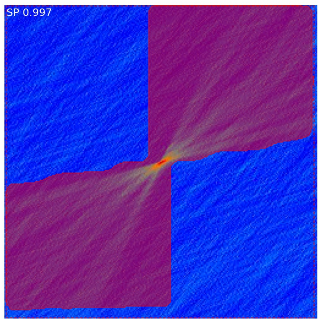
```
Detected Class:SP
Probability FRB: 0.992
Calculated DM: 2737.439 ± 42.772
Calculated time: 58153976.244
Brightest Pixel: 7.921
Brightest Pixel at position:(128, 130)
```

cand_tstart_58153.503043999997_tcand_261.6381856_dm_3682.33058_snr_-1.00000.h5



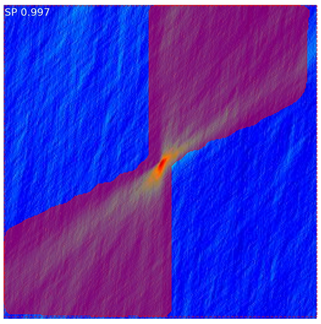
```
Detected Class:SP
Probability FRB: 0.992
Calculated DM: 3682.331 ± 57.536
Calculated time: 58154338.424
Brightest Pixel: 7.628
Brightest Pixel at position:(128, 126)
```

cand_tstart_58146.525462999998_tcand_37.5567159_dm_3439.98839_snr_-1.00000.h5



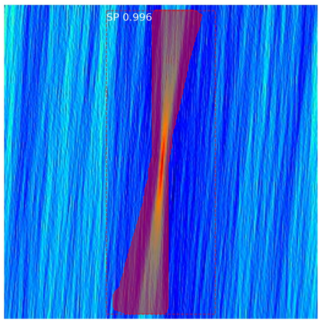
```
Detected Class:SP
Probability FRB: 0.997
Calculated DM: 3439.988 ± 53.750
Calculated time: 58148339.223
Brightest Pixel: 17.785
Brightest Pixel at position:(128, 128)
```

cand_tstart_58153.503043999997_tcand_262.1407068_dm_976.42477_snr_-1.00000.h5



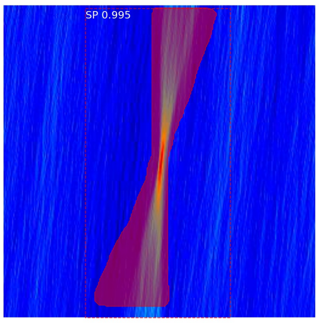
```
Detected Class:SP
Probability FRB: 0.997
Calculated DM: 968.856 ± 15.257
Calculated time: 58156331.844
Brightest Pixel: 14.395
Brightest Pixel at position:(129, 128)
```

cand_tstart_58153.503043999997_tcand_263.4258254_dm_503.13618_snr_-1.00000.h5



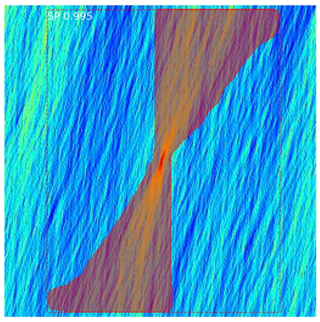
```
Detected Class:SP
Probability FRB: 0.996
Calculated DM: 511.122 ± 7.862
Calculated time: 58156353.944
Brightest Pixel: 9.849
Brightest Pixel at position:(126, 129)
```

cand_tstart_58153.503043999997_tcand_264.0258125_dm_1514.30518_snr_-1.00000.h5



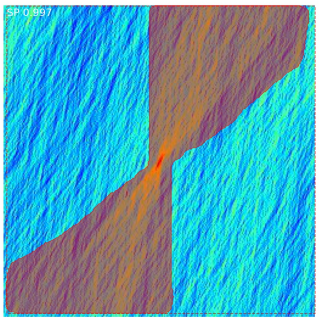
```
Detected Class:SP
Probability FRB: 0.995
Calculated DM: 1514.305 ± 23.661
Calculated time: 58156052.084
Brightest Pixel: 14.094
Brightest Pixel at position:(128, 129)
```

cand_tstart_58153.503043999997_tcand_264.6782308_dm_4287.84890_snr_-1.00000.h5



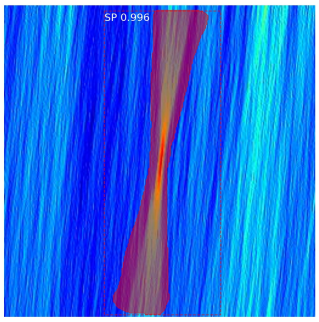
```
Detected Class:SP
Probability FRB: 0.995
Calculated DM: 4355.910 ± 66.998
Calculated time: 58156706.114
Brightest Pixel: 8.043
Brightest Pixel at position:(126, 129)
```

cand_tstart_58153.503043999997_tcand_264.6969164_dm_2005.48662_snr_-1.00000.h5



```
Detected Class:SP
Probability FRB: 0.997
Calculated DM: 2021.278 ± 31.336
Calculated time: 58155548.984
Brightest Pixel: 8.313
Brightest Pixel at position:(127, 129)
```

cand_tstart_58153.503043999997_tcand_246.9910994_dm_1534.83830_snr_-1.00000.h5



```
Detected Class:SP
Probability FRB: 0.996
Calculated DM: 1571.674 ± 23.982
Calculated time: 58156353.944
Brightest Pixel: 9.977
Brightest Pixel at position:(125, 129)
```

Figure 8.12 – Prediction examples in the test set using the trained FABLE model. The outputs are a generated mask indicating the exact shape of the FRB/SP. Also, a bounding box is produced to find the boundary location of the source of interest and the probability that it is an FRB/SP. The predicted DM and time values have been determined by using the mask that allows us to determine the brightest pixel of the candidates.

Table 8.5.1 – Summary table of our AP score metrics calculated on the test data set. Although FABLE performs well for small T_{IoU}, its performance rapidly decreases for larger T_{IoU}. This is likely due to the low SNR of candidates in the test images.

Class	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}	AP_{95}
FRB/SP	0.999	0.996	0.987	0.957	0.713	0.119

the intersection over union (IOU) thresholds $T_{IoU} \in \{0.5, 0.55, \ldots, 0.95\}$. We evaluate the precision and recall for FRB/SP classes and used the average precision (AP) for all 1121 and 3357 images in the validation set and test data set respectively.

The results presented in Figure 8.11 summarize the performance of our `FABLE` network on ground truth images in the validation and test data. In Table 8.5.1 and Figure 8.11, we calculate the AP metrics to evaluate the performance of `FABLE` for FRB/SP detection. As we note from the plots and table, we expect fewer positive detections of FRB/SP at greater T_{IoU} and more FRB/SP candidates are detected at lower T_{IoU}. The best performance of `FABLE` is obtained at $T_{IoU} = 0.5$ with an AP value of 0.999. In addition, we also show some examples of the detection of FRB/SP on the DM-time images in the test set at $T_{IoU} = 0.5$ in Figure 8.12. `FABLE` has successfully detected all the FRB/SP present in the test set at $T_{IoU} = 0.5$. During prediction, as can be noted in Figure 8.12, the `FABLE` model outputs a mask indicating the exact shape of the FRB/SP. Also, a bounding box is generated to find the location boundary of the source of interest and the probability that it is an FRB/SP. The mask has been useful to find the brightest pixel of the FRB/SP detected. Using the index/coordinates of the brightest pixel, we can work out the DM value and time of the candidate file. During prediction, we obtain information about the predicted class, the probability of the candidate being a FRB/SP, calculated DM and time and the position of the brightest pixel of the FRB/SP. This information is detailed on the right side of each candidate file.

8.6 Concluding Remarks

In this work, we develop a new deep-learning method for detecting, localizing and classifying FRB/SP sources in radio images. We used data from Agarwal et al. (2020) as a ground truth comparison for supervised machine learning. Our code, `FABLE` efficiently performs all tasks of FRB/SP detection, localizing and classification in one pipeline. The network is robust enough and we are able to use the generated mask of the bow-tie shape to locate the brightest pixel coordinate of the detected FRB/SP to compute an approximate DM and time for the detected candidate.

We briefly describe the contributions of this work: We

1. establish a novel technique for instance segmentation in radio images for Fast Radio Burst/single pulse searches,
2. demonstrate how existing deep learning techniques in the field of computer vision can be utilised to solve challenging problems such as detection, localization and classification in radio astronomy,
3. use of the DM-time image of a candidate, extracting the bow-tie shape information to perform FRB/SP classification, detection and localization,
4. develop an open-source software, available on GitHub: FABLE: Fast rAdio Burst Localization and dEtection using Mask R-CNN[‡].

We evaluate the performance of `FABLE` using the AP score metric, and show the precision and recall curves for FRB/SP against background noise. `FABLE` shows good performance at moderate IOU thresholds T_{IoU}. We measure a precision of 99.9 per cent at 90 per cent recall for FRB/SP with a minimum detection confidence threshold of 0.5. The proposed method can improve the consistency and accuracy of FRB/SP against background noise classification. In the future, the number of cases in the image database is expected to increase and the `FABLE` model can serve as a new tool for real-time pipeline that can automatically detect FRB/SP.

[‡]https://github.com/Zafiirah13/FABLE

9 Conclusions & Future Work

With the advent of the largest telescopes, the MeerLICHT, BlackGEM, Catalina Sky Survey (CSS), MeerKAT and SKA telescopes, we found that it is hard to perform classification of transients not because they cannot be detected by optical and radio telescopes, but rather because of the data deluge received per second that makes it difficult to sift through them quickly. As an avalanche of data is being processed, there is no way of knowing beforehand which are most likely to be certain types of variable stars, or likely to be interesting candidates, and which are most likely to be noise or interference in both optical and radio data. As analysing these data is expensive, both in terms of human effort and financially, it is important to develop automated techniques to sift through the enormous volumes of data as quickly and accurately as possible. Hence, this thesis aimed to devise several automated machine learning techniques that are capable of overcoming the classification problems for both optical and radio astronomy. Here we summarize the work accomplished and illustrate how these techniques can be, or are being used in real-time surveys.

This thesis approached jointly complementary domains of optical transients (variable stars), radio transients (single pulses, pulsars, fast radio bursts) and machine learning. In Chapter 1, the aims, objectives and contributions of this study are detailed. Chapter 2 introduced the transients phenomena in both optical and radio astronomy. A review of various types of variable stars, pulsars, fast radio burst candidates were conducted. As this study deals mostly with classification problems, Chapter 3 provided a review of several machine learning techniques for classification. Based upon the review of machine learning research, various ML classifiers were identified as promising avenues of investigation for variable light curves and image domain classification. The classifiers developed in this study, are able to process large volumes of data rapidly and efficiently. The characteristics demonstrated by these ML classifiers are attractive to current surveys.

9.1 Contributions

This work set out to develop transient classification algorithms for use during MeerLICHT and MeerKAT transient and single pulse searches. Not only have such algorithms been created, but they have been actively deployed to both MeerLICHT and MeerKAT surveys. The primary aims of this study have been achieved, and thus this research can be viewed a success in that regard. From this study, some general conclusions can be drawn from this study and these are summarised below.

1. The research hypothesis proposed that there existed some salient features of each type of variable stars, which differentiate them from other types of variable stars. Based on the results presented in this thesis, we made this conclusion in Chapter 4. The levels of classification performance achieved using a Random Forest classifier, can only be attained if the data possessed features that characterise well the separation between the various types of variable stars.

2. In Chapter 4, we identify a set of features which can be used in ML algorithms, that are capable of learning a separation between various types of variable stars. During this work, we provide evidence of the separability of features using various statistical techniques, such as point-biserial correlation test and the use of information theory. In this study, we found that using these salient features with a 'flat multi-class' classifier, we encounter several drops in performance for certain types of variable stars which has been attributed to the imbalanced learning problem.

   The imbalanced learning problem is encountered more often when we deal with datasets that are strongly dominated by certain majority classes. When using such data, there are three factors that contribute to the challenging task of differentiating between minority and majority class examples. The first factor is associated with class-overlap - which causes a lack of separation between classes. In addition, small-disjuncts and small sample size data are the remaining two factors, that make it difficult for ML algorithms to perform accurately. These factors lead classifiers to becoming biased towards predicting the majority class label and thus exhibit poor performance in terms of recall on the minority class. In attempting to overcome the imbalanced learning problem when using the Catalina Real-Time surveys of variable stars (CRTS), we developed a hierarchical ML classifier based on Random Forest, that classifies variable stars based on their astrophysical properties. We found that this hierarchical concept of classification greatly improved our classifier recall on minority classes. We recently found that similar ideas have been adopted for variable star classification in the Zwicky Transient Factory (ZTF, Bellm et al. 2019).

3. For further mitigation of the effects of the imbalanced learning problem on the CRTS data, there likely exists other methods which have the potential to greatly improve classification performance. In Chapter 5, we therefore develop three independent methods of balancing the sample size of CRTS data in the hierarchical classifier. At each level in the hierarchical tree, we balance the sample size of the minority class to the sample size of the majority class. This balancing of the data can be achieved either at the feature-level or directly on the raw-data (light curves). We therefore used `SMOTE` as a first approach that works directly on features extracted from the light curves. And in the second approach, we simulated examples of light curves using Gaussian Process (`GpFit`) or `RASLE`. We found that balancing the sample size of minority classes at each level in the hierarchy, improves recall on minority classes.

4. Chapter 6, 7 & 8 studied the problem of noise candidates for both optical and radio surveys, and their increasing volumes in more details. While in the past it may have been easy to filter out noise candidates (bogus or RFI) using S/N cuts assuming that most noise candidates possess a low S/N. However, this approach is not reliable nowadays as survey specifications are much improved and getting better. In recent surveys, noise candidates have high S/N, which implies that we require a high S/N threshold to be able to remove them. However, with the application of a high S/N threshold, the algorithm will reject weak legitimate candidates (real sources, pulsars, or fast radio bursts). In addition, with great improvements in telescope design and survey specifications, it becomes more difficult to select candidates in more practical ways. Firstly, the data deluge captured by sensitive telescopes is becoming impractical to store. Secondly, the rate of data capture is increasing at a rapid pace, thus great emphasis is made on rapid selection decisions, so that interesting and promising candidates can be prioritised for storage, viewing and science follow-up. These practical problems, similar to candidate classification and selection, are difficult to solve. Therefore, in Chapter 6, 7 & 8, we introduced the application of machine learning techniques to perform automatic candidate classifications for both optical and radio telescopes.

5. In Chapter 6, we presented `MeerCRAB`, a deep learning algorithm to classify between real and bogus candidates for the MeerLICHT telescope facility. As elaborated above, it is important to develop automated algorithms that are capable of selecting interesting candidates rapidly and efficiently. `MeerCRAB` is based on a convolutional neural network that works on image data directly. The MeerLICHT transient pipeline characterises a candidate with four images, New, Reference, Difference, and Significance. These images act as inputs to `MeerCRAB`, along with the labels provided by vetters. In this Chapter, we demonstrate the importance of having good labelling that is representative of the data by applying two methods: *thresholding* and *latent class model*, L_{lcm}. We found that `MeerCRAB` yields an accuracy of 99.5 % and MCC value of 0.989. This performance achieves an acceptable false positive and false negative rate for the real-time MeerLICHT transient detection pipeline. In addition, `MeerCRAB` can be adapted to be a system that disentangles interesting objects from a noisy background. We implemented a similar model to perform candidate selection for radio astronomy in Chapter 7.

6. In Chapter 7, we illustrate a user-friendly open-source python package `FRBID`: *Fast Radio Burst Intelligent Distinguisher*. We provide a short overview of a single stage in the MeerTRAP pipeline, that is, the design, implementation and exploitation of state-of-the-art machine learning techniques to classify between single pulses (can be periodic pulsars (1ms - 30s), rotating radio transients (RRATs), FRBs) and Radio Frequency Interference (RFI) in real time at the MeerKAT telescope. The input of `FRBID` is a candidate file with the DM-time and frequency-time data. The output of `FRBID` is the probability of a candidate of being an FRB in a range of [0;1]. After training `FRBID`, we obtain a performance achieving an accuracy of > 99.8% on the test data with a false positive rate of less than 1%. With such a performance, `FRBID` has been integrated to the single pulse search pipeline, MeerTRAP, for real-time classification of candidates. To date, `FRBID` has recently discovered more than half a dozen of new real candidates.

7. In Chapter 8, we provide a first proof of concept of an implementation of localization, detection and segmentation algorithm using Artificial Intelligence for Fast Radio Bursts or single pulses. The main aim of Chapter 8 is to provide a general idea of detecting and localizing sources in astronomical sources images. We develop `FABLE`: *Fast Radio Burst Localization & detection using Mask R-CNN* that efficiently performs all tasks of FRB/SP detection, localizing and classification in one pipeline. The network is robust enough and we are able to use the generated mask of the bow-tie shape to locate the brightest

pixel coordinate of the detected FRB/SP to compute an approximate DM and time for the detected candidate. We evaluate the performance of `FABLE` using the Average-Precision (AP) score metric and we obtain a precision of 99.9% at 90% recall for FRB/SP with a minimum detection confidence threshold of 0.5. With the success of `FABLE`, this automated ML algorithm can be adopted in other domains in astronomy, for instance source extraction software.

9.2 Future work

Machine learning is becoming increasingly useful in astronomy as it helps astronomers to identify interesting sources/candidates in large surveys. However, these learning algorithms (especially supervised methods) require a large amount of labelled data. As we have seen in this thesis, the process of labelling data is expensive both in terms of time, human labour and precision. If the labelling process is performed in an illogical way, ML algorithms tend to become biased, hence affecting performance. Therefore, there is a need to develop algorithms that eliminate human annotations and learn directly with larger data sets. These algorithms will potentially mitigate some of the biases that come into play with data curation. Below we detail recommendations for future work in terms of semi-supervised and unsupervised learning algorithms that would be useful to explore for future surveys.

1. **Semi-supervised learning approach for classification**

   Semi-supervised learning can be applied where data consists of a few labelled examples and a large fraction of unlabelled examples. The ML model must learn directly from the small fraction of labelled examples and somehow learn on the additional larger dataset of unlabelled examples. This concept allows the model to generalise by improving the performance of the supervised task and is capable of classifying new unlabelled examples in the future.

   The semi-supervised GAN (Generative Adversarial Network, Goodfellow et al. 2016) is an example of a model that can be used to address semi-supervised learning problems. In a traditional GAN model, the discriminator is trained to predict whether an image is *real* (arises from the same distribution as the training data set) or *fake* (generated by the generator). This concept allows the model to learn features from unlabelled images. Hence, via a transfer learning approach the discriminator can be utilized when developing a classifier for similar dataset, thus enabling the supervised prediction task to make the most from the unsupervised training of the GAN.

In a semi-supervised GAN, the discriminator is updated in such a way that it can predict $N+1$ classes, where N is the number of class labels in the classification task and the extra class label is added for a new '*fake*' class. The concept of semi-supervised GAN involves training the discriminator simultaneously for both the unsupervised GAN problem and the supervised classification problem.

Hence, the discriminator is trained in two ways: a supervised and unsupervised approaches.

- **Unsupervised training mode**: The training process of the discriminator works similarly as in a traditional GAN and the model is used to predict whether an example is either '*real*' or '*fake*'.
- **Supervised training mode**: The training process of the discriminator allows it to predict the class label of real examples.

The unsupervised training approach allows the model to learn useful feature extraction capabilities from a large unlabelled dataset. Hence, using a semi-supervised approach for classification could in principle achieve state-of-the-art results when trained on a few labelled examples, such as hundreds or thousands of subjects. In addition, the training process of a semi-supervised GAN can also produce better quality images output by the generator model.

In practice, as we noted in this thesis, the number of available labelled examples for certain classes of variable stars or number of interesting candidates or pulsar detections is small. We have learnt from the literature review, that other practitioners either undersampled or over-sampled the training data to avoid biasing the ML model. In this practical scenario, a semi-supervised or unsupervised approach would be helpful to overcome deficiencies in these workarounds.

2. **Unsupervised learning approach for classification**

   Unsupervised or self-supervised learning is another approach that works directly on a vast amount of data without labels. This approach could allow us to build AI models that work well in real-time surveys and adapt quickly to changing noise levels in data streams. The elimination of human annotation will not only reduce the need for human labour, but also it will allow models to be created and deployed much quicker, thus enabling more rapid and more accurate responses to evolving real-time surveys.

An example of a self-supervised model is the `SEER`[*](SElf-supERvised) model. It is a self-supervised computer vision model that can learn from any random group of images - without the need for careful curation and labels during training (Goyal et al., 2021). `SEER` is based on two new algorithms known as `SwAV` (Caron et al., 2020) and `RegNets` (Radosavovic et al., 2020). `SwAV` is a clustering algorithm that rapidly group images with similar visual concepts and leverages their similarities. And `RegNets` are ConvNet models that show the ability to scale billions or even trillions of parameters, and the model can be optimized such that it can fit different runtime and memory limitations. With the combination of these models, `SwAV` and `RegNets` in `SEER`, the model shows an improvement over the previous state-of-the-art in self-supervised learning - and did so with 6 times less training time (Goyal et al., 2021).

Therefore, with the SKA coming online soon, such an unsupervised learning algorithm is an avenue of research that may be worth exploring.

[*]https://github.com/facebookresearch/vissl

BIBLIOGRAPHY

Aerts, C., Christensen-Dalsgaard, J., & Kurtz, D. W. 2010, Asteroseismology

Agarap, A. F., Deep Learning using Rectified Linear Units (ReLU). 2018, arXiv e-prints, arXiv:1803.08375

Agarwal, D., Aggarwal, K., Burke-Spolaor, S., Lorimer, D. R., & Garver-Daniels, N., FETCH: A deep-learning based classifier for fast transient classification. 2020, *MNRAS*, 497, 1661

Aigrain, S., Parviainen, H., & Pope, B. J. S., K2SC: flexible systematics correction and detrending of K2 light curves using Gaussian process regression. 2016, *MNRAS*, 459, 2408

Ali, N., Neagu, D., & Trundle, P., Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. 2019, SN Applied Sciences, 1, 1

Alonso-Garcia, J., Minniti, D., Angeloni, R., et al., Infrared detection of a dwarf nova eruption in the globular cluster M22 by the VVV survey. 2015, The Astronomer's Telegram, 7238, 1

Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M., Fast Direct Methods for Gaussian Processes. 2015, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38, 252

Astier, P., Guy, J., Regnault, N., et al., The Supernova Legacy Survey: measurement of Ω_M, Ω_Λ and w from the first year data set. 2006, *A&A*, 447, 31

Bannister, K. W., Shannon, R. M., Macquart, J. P., et al., The Detection of an Extremely Bright Fast Radio Burst in a Phased Array Feed Survey. 2017, *ApJ*, 841, L12

Barsdell, B. R., Bailes, M., Barnes, D. G., & Fluke, C. J., Accelerating incoherent dedispersion. 2012, *MNRAS*, 422, 379

Bates, S. D., Bailes, M., Bhat, N. D. R., et al., The High Time Resolution Universe Pulsar Survey - II. Discovery of five millisecond pulsars. 2011, *MNRAS*, 416, 2455

Bell Burnell, S. J. 1977, in Eighth Texas Symposium on Relativistic Astrophysics, ed. M. D. Papagiannis, Vol. 302, 685

Bellm, E. C., Kulkarni, S. R., Graham, M. J., et al., The Zwicky Transient Facility: System Overview, Performance, and First Results. 2019, *PASP*, 131, 018002

Bellman, R., Dynamic programming. 1957, Princeton University Press, Princeton

Belokurov, V., Evans, N. W., & Du, Y. L., Light-curve classification in massive variability surveys - I. Microlensing. 2003, *MNRAS*, 341, 1373

Benavente, P., Protopapas, P., & Pichara, K., Automatic Survey-invariant Classification of Variable Stars. 2017, *ApJ*, 845, 147

Benko, A. & Lányi, C. S., History of artificial intelligence. 2009

Bentley, J. L., Multidimensional Binary Search Trees Used for Associative Searching. 1975a, Communications of the ACM, 18(9), 18(9)

Bentley, J. L., Multidimensional binary search trees used for associative searching. 1975b, Communications of the ACM, 18 (9), 18 (9)

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D., Hyperopt: a python library for model selection and hyperparameter optimization. 2015, Computational Science & Discovery, 8, 8

Berrar, D. & Dubitzky, W., Information gain. 2013, Encyclopedia of Systems Biology; Dubitzky, W., Wolkenhauer, O., Yokota, H., Cho, K.-H., Eds

Bertin, E. 2011, in Astronomical Society of the Pacific Conference Series, Vol. 442, Astronomical Data Analysis Software and Systems XX, ed. I. N. Evans, A. Accomazzi, D. J. Mink, & A. H. Rots, 435

Bertin, E. & Arnouts, S., SExtractor: Software for source extraction. 1996, A&AS, 117, 117

Bethapudi, S. & Desai, S., Separation of pulsar signals from noise using supervised machine learning algorithms. 2018, *Astronomy and Computing*, 23, 15

Bhattacharya, D. & van den Heuvel, E. P. J., Formation and evolution of binary and millisecond radio pulsars. 1991, *Phys. Rep.*, 203, 1

BinSim. 2021, Binary star visualisations

Bishop, C. 2006, Pattern Recognition and Machine Learning (Springer, New York)

Blažko, S., Mitteilung über veränderliche Sterne. 1907a, *Astronomische Nachrichten*, 175, 325

Blažko, S., Mitteilung über veränderliche Sterne. 1907b, *Astronomische Nachrichten*, 175, 327

Bloemen, S., Groot, P., Woudt, P., et al. 2016, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 9906, Proc. SPIE, 990664

Borne, K. D., A machine learning classification broker for the LSST transient database. 2008, *Astronomische Nachrichten*, 329, 255

Borovicka, T., Jirina Jr, M., Kordik, P., & Jirina, M., Selecting representative data sets. 2012, Advances in data mining knowledge discovery and applications

Boyle, P. C. & Chime/Frb Collaboration, First detection of fast radio bursts between 400 and 800 MHz by CHIME/FRB. 2018, The Astronomer's Telegram, 11901, 1

Breiman, L., Random Forests. 2001, Machine Learning, 45, 45

Breiman, L., Friedman, J., Stone, C., & Olshen, R., Classification and Regression Trees. 1984, Taylor and Francis

Bretthorst, G. L. 2013, Bayesian spectrum analysis and parameter estimation, Vol. 48 (Springer Science & Business Media)

Brown, G., Pocock, A., Zhao, M.-J., & Luján, M., Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. 2012, Journal of machine learning research, 13, 13

Burke-Spolaor, S., Bailes, M., Johnston, S., et al., The High Time Resolution Universe Pulsar Survey - III. Single-pulse searches and preliminary analysis. 2011, *MNRAS*, 416, 2465

Burke-Spolaor, S. & Bannister, K. W., The Galactic Position Dependence of Fast Radio Bursts and the Discovery of FRB011025. 2014, *ApJ*, 792, 19

Buturovic, L. J., Improving K-nearest neighbour density and error estimates. 1993, Pattern Recognition, 26, 26

Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., & Maureira, J.-C., Deep-HiTS: Rotation Invariant Convolutional Neural Network for Transient Detection. 2017, *ApJ*, 836, 97

Camilo, F., Scholz, P., Serylak, M., et al., Revival of the Magnetar PSR J1622-4950: Observations with MeerKAT, Parkes, XMM-Newton, Swift, Chandra, and NuSTAR. 2018, *ApJ*, 856, 180

Caron, M., Misra, I., Mairal, J., et al., Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. 2020, arXiv e-prints, arXiv:2006.09882

Carrasco, D., Barrientos, L. F., Pichara, K., Anguita, T., et al., Photometric classification of quasars from RCS-2 using Random Forest. 2015, Astronomy & Astrophysics, A44, A44

Castro, N., Protopapas, P., & Pichara, K., Uncertain Classification of Variable Stars: Handling Observational GAPS and Noise. 2018, *AJ*, 155, 16

Catelan, M., Minniti, D., Lucas, P. W., et al., Stellar Variability in the VVV survey. 2013, arXiv e-prints, arXiv:1310.1996

Cauchy, A., Sur les resultats moyens d'observations de mes nature, et sur les resultats les plus probables. 1853, C.R. Acad. Sci, 37, 37

Champion, D. J., Petroff, E., Kramer, M., et al., Five new fast radio bursts from the HTRU high-latitude survey at Parkes: first evidence for two-component bursts. 2016, *MNRAS*, 460, L30

Chao, C., Liaw, A., & Breiman, L., Using random forests to learn imbalanced data. 2004, University of California, Berkeley

Chatterjee, S., Law, C. J., Wharton, R. S., et al., A direct localization of a fast radio burst and its host. 2017, *Nature*, 541, 58

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P., SMOTE: Synthetic Minority Over-sampling Technique. 2011, arXiv e-prints, arXiv:1106.1813

Chen, C., Liaw, A., Breiman, L., et al., Using random forest to learn imbalanced data. 2004, University of California, Berkeley, 110, 110

Chen, H., Diethe, T., Twomey, N., & Flach, P. A., Anomaly detection in star light curves using hierarchical Gaussian processes. 2018

Chen, T. & Guestrin, C., XGBoost: A Scalable Tree Boosting System. 2016, ArXiv e-prints:1603.02754

Chiao, J.-Y., Chen, K.-Y., Liao, K. Y.-K., et al., Detection and classification the breast tumors using mask R-CNN on sonograms. 2019, Medicine, 98, 98

CHIME/FRB Collaboration, Amiri, M., Bandura, K., et al., A second source of repeating fast radio bursts. 2019a, *Nature*, 566, 235

CHIME/FRB Collaboration, Amiri, M., Bandura, K., et al., Observations of fast radio bursts at frequencies down to 400 megahertz. 2019b, *Nature*, 566, 230

Chollet, F. & others, Keras: The Python Deep Learning library. 2018, ascl:1806.022

Cios, K., Pedrycz, W., Swiniarski, R., & Kurgan, L. 2007, Data mining: a knowledge discovery approach (Springer Publishing Company, Incorporated)

Cordes, J. M. & Lazio, T. J. W., NE2001.I. A New Model for the Galactic Distribution of Free Electrons and its Fluctuations. 2002, arXiv e-prints, astro

Cover, T. & Hart, P., Nearest neighbor pattern classification. 1967, IEEE transactions on information theory, 13, 13

Djorgovski, S. G., Donalek, C., Mahabal, A., et al., Towards an Automated Classification of Transient Events in Synoptic Sky Surveys. 2011, arXiv e-prints, arXiv:1110.4655

Djorgovski, S. G., Graham, M. J., Donalek, C., et al., Real-Time Data Mining of Massive Data Streams from Synoptic Sky Surveys. 2016, arXiv e-prints, arXiv:1601.04385

Drake, A. J., Djorgovski, S. G., Catelan, M., et al., The Catalina Surveys Southern periodic variable star catalogue. 2017, *MNRAS*, 469, 3688

Drake, A. J., Djorgovski, S. G., García-Álvarez, D., et al., Ultra-short Period Binaries from the Catalina Surveys. 2014a, *ApJ*, 790, 157

Drake, A. J., Djorgovski, S. G., Mahabal, A., et al., First Results from the Catalina Real-Time Transient Survey. 2009, *ApJ*, 696, 870

Drake, A. J., Graham, M. J., Djorgovski, S. G., et al., The Catalina Surveys Periodic Variable Star Catalog. 2014b, *ApJS*, 213, 9

Eatough, R. P., Keane, E. F., & Lyne, A. G., An interference removal technique for radio pulsar searches. 2009, *MNRAS*, 395, 410

Eatough, R. P., Molkenthin, N., Kramer, M., et al., Selection of radio pulsar candidates using artificial neural networks. 2010, *MNRAS*, 407, 2443

Edwards, A. L., Note on the 'correction for continuity' in testing the significance of the difference between correlated proportions. 1948, Psychometrika, 13, 13

Eyer, L. & Blake, C., Automated classification of variable stars for All-Sky Automated Survey 1-2 data. 2005, *MNRAS*, 358, 30

Faraway, J., Mahabal, A., Sun, J., et al., Modeling lightcurves for improved classification of astronomical objects. 2016, Statistical Analysis and Data Mining: The ASA Data Science Journal, 9, 9

Fawcett, T. & Provost, F., Combining Data Mining and Machine Learning for Effective User Profile. 1996, In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining

Feast, M. W. & Walker, A. R., Cepheids as distance indicators. 1987, *ARA&A*, 25, 345

Felten, D. L., O'Banion, M. K., & Maida, M. E. 2015, Netter's atlas of neuroscience (Elsevier Health Sciences)

Fletcher, R., Practical methods of optimization (2nd ed.). 1987, New York- John Wiley & Sons

Formann, A. K. 1984, Die latent-class-analyse: Einführung in Theorie und Anwendung (Beltz)

Foster, G., Karastergiou, A., Geyer, M., et al., Verifying and reporting Fast Radio Bursts. 2018, *MNRAS*, 481, 2612

Freedman, W. L. 1988, in Astronomical Society of the Pacific Conference Series, Vol. 4, The Extragalactic Distance Scale, ed. S. van den Bergh & C. J. Pritchet, 24–31

Friedman, J., Greedy function approximation: A gradient boosting machine. 2001, Ann. Statist, 29, 29

Friedman, J., Bentley, J., & Finkel, R., An Algorithm for Finding Best Matches in Logarithmic Expected Time. 1977, ACM Transactions on Mathematical Software, 3(3), 3(3)

Gabruseva, T., Zlobin, S., & Wang, P., Photometric Light Curves Classification with Machine Learning. 2020, *Journal of Astronomical Instrumentation*, 9, 2050005

Gaia Collaboration, Brown, A. G. A., Vallenari, A., et al., Gaia Data Release 1. Summary of the astrometric, photometric, and survey properties. 2016, *A&A*, 595, A2

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F., A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. 2011, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42, 42

Ghosh, P. 2007, Rotation and Accretion Powered Pulsars, Vol. 10

Gieseke, F., Bloemen, S., van den Bogaard, C., et al., Convolutional neural networks for transient candidate vetting in large-scale surveys. 2017, *MNRAS*, 472, 3101

Gold, T., Rotating Neutron Stars as the Origin of the Pulsating Radio Sources. 1968, *Nature*, 218, 731

Goodfellow, I., Bengio, Y., & Courville, A. 2016, Deep Learning (MIT Press), `http://www.deeplearningbook.org`

Goyal, M., Goyal, R., & Lall, B., Learning Activation Functions: A new paradigm for understanding Neural Networks. 2019, arXiv e-prints, arXiv:1906.09529

Goyal, P., Caron, M., Lefaudeux, B., et al., Self-supervised Pretraining of Visual Features in the Wild. 2021, arXiv e-prints, arXiv:2103.01988

Gregory, P. C. & Loredo, T. J., A New Method for the Detection of a Periodic Signal of Unknown Shape and Period. 1992, *ApJ*, 398, 146

Groot, P. J., The multi-colour dynamic Universe explored. 2019, *Nature Astronomy*, 3, 1160

Gupta, S., Point biserial correlation coefficient and its generalization. 1960, Psychometrika, 25(4), 25(4)

Guyon, I. & Elisseeff, A., An Introduction to Variable and Feature Selection. 2003, Journal of Machine Learning Research, 3, 3

Han, B., Yao, Q., Yu, X., et al., Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels. 2018, arXiv e-prints, arXiv:1804.06872

Hancock, P. A., Nourbakhsh, I., & Stewart, J., On the future of transportation in an era of automated and autonomous vehicles. 2019, Proceedings of the National Academy of Sciences, 116, 116

Hastie, T., Tibshirani, R., & Friedman, J. 2009, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. (Springer, New York, NY)

He, H. & Garcia, E. A., Learning from imbalanced data. 2008, IEEE Transactions on Knowledge & Data Engineering

He, K., Gkioxari, G., Doll?, P., & Girshick, R., Mask R-CNN. 2017

He, K., Zhang, X., Ren, S., & Sun, J., Deep Residual Learning for Image Recognition. 2016

Hessels, J. W. T., Ransom, S. M., Stairs, I. H., Kaspi, V. M., & Freire, P. C. C., A 1.4 GHz Arecibo Survey for Pulsars in Globular Clusters. 2007, *ApJ*, 670, 363

Hewish, A., Bell, S. J., Pilkington, J. D. H., Scott, P. F., & Collins, R. A., Observation of a Rapidly Pulsating Radio Source. 1968, *Nature*, 217, 709

Hewish, A., Bell, S. J., Pilkington, J. D. H., Scott, P. F., & Collins, R. A., Observation of a Rapidly Pulsating Radio Source (Reprinted from Nature, February 24, 1968). 1969, *Nature*, 224, 472

Holoien, T. W. S., Brown, J. S., Stanek, K. Z., et al., The ASAS-SN bright supernova catalogue - III. 2016. 2017, *MNRAS*, 471, 4966

Hosenie, Z., Lyon, R., Stappers, B., Mootoovaloo, A., & McBride, V., Imbalance learning for variable star classification. 2020, *MNRAS*, 493, 6050

Hosenie, Z., Lyon, R. J., Stappers, B. W., & Mootoovaloo, A., Comparing Multiclass, Binary, and Hierarchical Machine Learning Classification schemes for variable stars. 2019a, *MNRAS*, 488, 4858

Hosenie, Z., Lyon, R. J., Stappers, B. W., & Mootoovaloo, A., Comparing Multiclass, Binary, and Hierarchical Machine Learning Classification schemes for variable stars. 2019b, *MNRAS*, 488, 4858

Hoyle, B., Rau, M. M., Bonnett, C., Seitz, S., & Weller, J., Data augmentation for machine learning redshifts applied to Sloan Digital Sky Survey galaxies. 2015, *MNRAS*, 450, 305

Hutter, F., Hoos, H. H., & Leyton-Brown, K., Sequential model-based optimization for general algorithm configuration. 2011

Ishak, B. 2017, Statistics, data mining, and machine learning in astronomy: a practical Python guide for the analysis of survey data, by Željko Ivezić, Andrew J. Connolly, Jacob T. VanderPlas and Alexander Gray: Scope: reference. Level: specialist

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al., LSST: From Science Drivers to Reference Design and Anticipated Data Products. 2019, *ApJ*, 873, 111

Jankowski, F., Berezina, M., Stappers, B. W., et al., Real-time triggering capabilities for Fast Radio Bursts at the MeerKAT telescope. 2020, arXiv e-prints, arXiv:2012.05173

Japkowicz, N. & Stephen, S., The class imbalance problem: A systematic study. 2002, Intelligent data analysis, 6, 6

Jonas, J. & MeerKAT Team, The MeerKAT Radio Telescope. 2016, 1

Jurcsik, J., Smitola, P., Hajdu, G., et al., Overtone and Multi-mode RR Lyrae Stars in the Globular Cluster M3. 2015, *ApJS*, 219, 25

Jurcsik, J., Sódor, Á., Szeidl, B., et al., The Konkoly Blazhko Survey: is light-curve modulation a common property of RRab stars? 2009, *MNRAS*, 400, 1006

Jurić, M., Kantor, J., Lim, K. T., et al. 2017, in Astronomical Society of the Pacific Conference Series, Vol. 512, Astronomical Data Analysis Software and Systems XXV, ed. N. P. F. Lorente, K. Shortridge, & R. Wayth, 279

Kaiser, N., Burgett, W., Chambers, K., et al. 2010, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 7733, Proc. SPIE, 77330E

Keith, M. J., Jameson, A., van Straten, W., et al., The High Time Resolution Universe Pulsar Survey - I. System configuration and initial discoveries. 2010, *MNRAS*, 409, 619

Keller, S. C., Schmidt, B. P., Bessell, M. S., et al., The SkyMapper Telescope and The Southern Sky Survey. 2007, *Publ. Astron. Soc. Australia*, 24, 1

Kgoadi, R., Engelbrecht, C., Whittingham, I., & Tkachenko, A., General classification of light curves using extreme boosting. 2019, arXiv e-prints, arXiv:1906.06628

Kim, D.-W. & Bailer-Jones, C. A. L., A package for the automated classification of periodic variable stars. 2016, *A&A*, 587, A18

Kingma, D. P. & Ba, J., Adam: A Method for Stochastic Optimization. 2014, arXiv e-prints, arXiv:1412.6980

Kingma, D. P. & Ba, J., Adam: A Method for Stochastic Optimization. 2015, International Conference on Learning Representations

Klebesadel, R. W., Strong, I. B., & Olson, R. A. 2004, in American Institute of Physics Conference Series, Vol. 727, Gamma-Ray Bursts: 30 Years of Discovery, ed. E. Fenimore & M. Galassi, 3–6

Knigge, C. 2011, in Astronomical Society of the Pacific Conference Series, Vol. 447, Evolution of Compact Binaries, ed. L. Schmidtobreick, M. R. Schreiber, & C. Tappert, 3

Koch, D. G., Borucki, W. J., Basri, G., et al., Kepler Mission Design, Realized Photometric Performance, and Early Science. 2010, *ApJ*, 713, L79

Kocz, J., Bailes, M., Barnes, D., Burke-Spolaor, S., & Levin, L., Enhanced pulsar and single pulse detection via automated radio frequency interference detection in multipixel feeds. 2012, *MNRAS*, 420, 271

Kotsiantis, S. B., Supervised machine learning: A review of classification techniques. 2007

Krawczynski, H. & Treister, E., Active galactic nuclei — the physics of individual sources and the cosmic history of formation and evolution. 2013, *Frontiers of Physics*, 8, 609

Kullback, S. & Leibler, R. A., On Information and Sufficiency. 1951, The Annals of Mathematical Statistics, 22(1), 22(1)

Lang, D., Hogg, D. W., Mierle, K., Blanton, M., & Roweis, S., Astrometry.net: Blind Astrometric Calibration of Arbitrary Astronomical Images. 2010, *AJ*, 139, 1782

Last, F., Douzas, G., & Bacao, F., Oversampling for Imbalanced Learning Based on K-Means and SMOTE. 2017, arXiv e-prints, arXiv:1711.00837

Lecun, Y. 1989, Generalization and network design strategies, ed. R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Elsevier)

LeCun, Y., Bengio, Y., & Hinton, G., Deep learning. 2011, Nature, 521, 521

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P., Gradient-based learning applied to document recognition. 1998, Proceedings of the IEEE, 86, 86

Lecun, Y., Haffner, P., Bottou, L., & Bengio, Y., Object Recognition with Gradient-Based Learning. 1999

Lemaitre, G., Nogueira, F., & Aridas, C. K., Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. 2016, arXiv e-prints, arXiv:1609.06570

Li, Y., Wang, G., Nie, L., Wang, Q., & Tan, W., Distance metric optimization driven convolutional neural network for age invariant face recognition. 2018, Pattern Recognition, 75, 75

Lin, H., Li, X., & Zeng, Q., Pulsar Candidate Sifting Using Multi-input Convolution Neural Networks. 2020, arXiv e-prints, arXiv:2007.14843

Lin, T., Doll?, P., Girshick, R., et al., Feature Pyramid Networks for Object Detection. 2017

Lochner, M., McEwen, J. D., Peiris, H. V., Lahav, O., & Winter, M. K., Photometric Supernova Classification with Machine Learning. 2016, *ApJS*, 225, 31

Lomb, N. R., Least-Squares Frequency Analysis of Unequally Spaced Data. 1976, *Ap&SS*, 39, 447

Lorimer, D. R., Binary and Millisecond Pulsars. 2008, *Living Reviews in Relativity*, 11, 8

Lorimer, D. R. & Kramer, M. 2004, Handbook of Pulsar Astronomy, Vol. 4

LSST Science Collaboration, Abell, P. A., Allison, J., et al., LSST Science Book, Version 2.0. 2009, arXiv e-prints, arXiv:0912.0201

Lyon, R. J., Stappers, B. W., Cooper, S., Brooke, J. M., & Knowles, J. D., Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach. 2016, *MNRAS*, 459, 1104

Madore, B. F. & Freedman, W. L., The Cepheid Distance Scale. 1991, *PASP*, 103, 933

Mahabal, A., Sheth, K., Gieseke, F., et al., Deep-Learnt Classification of Light Curves. 2017, arXiv e-prints, arXiv:1709.06257

Mahabal, A. A., Donalek, C., Djorgovski, S. G., et al. 2012, in New Horizons in Time Domain Astronomy, ed. E. Griffin, R. Hanisch, & R. Seaman, Vol. 285, 355–357

Malz, A., Hlozek, R., Allam, T. J., Bahmanyar, A., et al., The Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC): Selection of a performance metric for classification probabilities balancing diverse science goals. 2018, arXiv:1809.11145

Manchester, R. N., Hobbs, G. B., Teoh, A., & Hobbs, M., The Australia Telescope National Facility Pulsar Catalogue. 2005, *AJ*, 129, 1993

Manchester, R. N., Lyne, A. G., Camilo, F., et al., The Parkes multi-beam pulsar survey - I. Observing and data analysis systems, discovery and timing of 100 pulsars. 2001, *MNRAS*, 328, 17

Martínez-Palomera, J., Förster, F., Protopapas, P., et al., The High Cadence Transit Survey (HiTS): Compilation and Characterization of Light-curve Catalogs. 2018, The Astronomical Journal, 156, 156

Masui, K., Lin, H.-H., Sievers, J., et al., Dense magnetized plasma associated with a fast radio burst. 2015, *Nature*, 528, 523

McCulloch, W. S. & Pitts, W., A logical calculus of the ideas immanent in nervous activity. 1943, The bulletin of mathematical biophysics, 5, 5

McCutcheon, A. L. 1987, Latent class analysis No. 64 (Sage)

McLaughlin, M. A., Lyne, A. G., Lorimer, D. R., et al., Transient radio bursts from rotating neutron stars. 2006, *Nature*, 439, 817

McNemar, Q., Note on the sampling error of the difference between correlated proportions or percentages. 1947, Psychometrika, 12, 12

Mirabal, N., Charles, E., Ferrara, E., et al., 3FGL demographics outside the galactic plane using supervised machine learning: Pulsar and dark matter subhalo interpretations. 2016, The Astrophysical Journal, 825, 825

Mitchell, T. M. 1997, Machine Learning (New York: McGraw-Hill)

Muthukrishna, D., Narayan, G., Mandel, K. S., Biswas, R., & Hložek, R., RAPID: Early Classification of Explosive Transients Using Deep Learning. 2019, *PASP*, 131, 118002

Nadyozhin, D. K., Physics of Supernovae: theory, observations, unresolved problems. 2008, arXiv e-prints, arXiv:0804.4350

Narayan, G., Zaidi, T., Soraisam, M. D., et al., Machine-learning-based Brokers for Real-time Classification of the LSST Alert Stream. 2018, *ApJS*, 236, 9

Nembrini, S., König, I. R., & Wright, M. N., The revival of the Gini importance? 2018, Bioinformatics, 34, 34

Netzel, H., Smolec, R., Soszyński, I., & Udalski, A., Blazhko effect in the first overtone RR Lyrae stars of the OGLE Galactic bulge collection. 2018, *MNRAS*, 480, 1229

Ng, A. Y. 2004, Feature selection, L 1 vs. L 2 regularization, and rotational invariance, 78

Nun, I., Protopapas, P., Sim, B., et al., FATS: Feature Analysis for Time Series. 2015, arXiv e-prints, arXiv:1506.00010

Oord, A. v. d., Dieleman, S., Zen, H., et al., Wavenet: A generative model for raw audio. 2016, arXiv preprint arXiv:1609.03499

Ossama, A., Abdel-rahman, M., Hui, J., et al., Convolutional Neural Networks for Speech Recognition. 2014, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22(10), 22(10)

Pashchenko, I. N., Sokolovsky, K. V., & Gavras, P., Machine learning search for variable stars. 2018, *MNRAS*, 475, 2326

Paterson, K. 2019, in IAU Symposium, Vol. 339, Southern Horizons in Time-Domain Astronomy, ed. R. E. Griffin, 203–203

Pearson, K., Note on Regression and Inheritance in the Case of Two Parents. 1895, Proceedings of the Royal Society of London, 58, 58

Pedregosa, F., Varoquaux, G., Gramfort, A., et al., Scikit-learn: Machine Learning in Python. 2012, arXiv e-prints, arXiv:1201.0490

Pešek, O., Mask R-CNN v prostředí GRASS GIS. 2018

Peterson, B. M., Wanders, I., Horne, K., et al., On Uncertainties in Cross-Correlation Lags and the Reality of Wavelength-dependent Continuum Lags in Active Galactic Nuclei. 1998, Publications of the Astronomical Society of the Pacific, 110, 110

Petroff, E., Hessels, J. W. T., & Lorimer, D. R., Fast radio bursts. 2019, *A&ARv*, 27, 4

Pilkington, J. D. H., Hewish, A., Bell, S. J., & Cole, T. W., Observations of some further Pulsed Radio Sources. 1968, *Nature*, 218, 126

Quinlan, J. R., Induction of Decision Trees. 1986, Machine Learning, 1, 1

Radosavovic, I., Prateek Kosaraju, R., Girshick, R., He, K., & Dollár, P., Designing Network Design Spaces. 2020, arXiv e-prints, arXiv:2003.13678

Rajwade, K., Stappers, B., Williams, C., et al., MeerTRAP in the era of multi-messenger astrophysics. 2021, arXiv e-prints, arXiv:2103.08410

Rasmussen, C. E., Gaussian processes in machine learning. 2003

Rau, A., Kulkarni, S. R., Law, N. M., et al., Exploring the Optical Transient Sky with the Palomar Transient Factory. 2009, *PASP*, 121, 1334

Ravi, V., Shannon, R. M., & Jameson, A., A Fast Radio Burst in the Direction of the Carina Dwarf Spheroidal Galaxy. 2015, *ApJ*, 799, L5

Ren, S., He, K., Girshick, R., & Sun, J., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2015, arXiv e-prints, arXiv:1506.01497

Revsbech, E. A., Trotta, R., & van Dyk, D. A., STACCATO: a novel solution to supernova photometric classification with biased training sets. 2018, *MNRAS*, 473, 3969

Richards, J. W., Starr, D. L., Butler, N. R., et al., On Machine-learned Classification of Variable Stars with Sparse and Noisy Time-series Data. 2011a, *ApJ*, 733, 10

Richards, J. W., Starr, D. L., Butler, N. R., et al., On Machine-learned Classification of Variable Stars with Sparse and Noisy Time-series Data. 2011b, *ApJ*, 733, 10

Ridley, J. P., Crawford, F., Lorimer, D. R., et al., Eight new radio pulsars in the Large Magellanic Cloud. 2013, *MNRAS*, 433, 138

Rucinski, S. M. 1996, in Astronomical Society of the Pacific Conference Series, Vol. 90, The Origins, Evolution, and Destinies of Binary Stars in Clusters, ed. E. F. Milone & J. C. Mermilliod, 270

Rucinski, S. M., The short-period end of the contact binary period distribution based on the All-Sky Automated Survey. 2007, *MNRAS*, 382, 393

Saha, A. & Vivas, A. K., A Hybrid Algorithm for Period Analysis from Multiband Data with Sparse and Irregular Sampling for Arbitrary Light-curve Shapes. 2017, *AJ*, 154, 231

Sánchez-Sáez, P., Reyes, I., Valenzuela, C., et al., Alert Classification for the ALeRCE Broker System: The Light Curve Classifier. 2020, arXiv e-prints, arXiv:2008.03311

Scargle, J. D., Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data. 1982, *ApJ*, 263, 835

Scholz, P., Spitler, L. G., Hessels, J. W. T., et al., The Repeating Fast Radio Burst FRB 121102: Multi-wavelength Observations and Additional Bursts. 2016, *ApJ*, 833, 177

Seiradakis, J. H. & Wielebinski, R., Morphology and characteristics of radio pulsars. 2004, *A&ARv*, 12, 239

Sesar, B., Hernitschek, N., Mitrović, S., et al., Machine-learned identification of RR Lyrae stars from sparse, multi-band data: the PS1 sample. 2017, The Astronomical Journal, 153, 153

Shannon, R. M., Macquart, J. P., Bannister, K. W., et al., The dispersion-brightness relation for fast radio bursts from a wide-field survey. 2018, *Nature*, 562, 386

Shappee, B. J., Prieto, J. L., Grupe, D., et al., The Man behind the Curtain: X-Rays Drive the UV through NIR Variability in the 2013 Active Galactic Nucleus Outburst in NGC 2617. 2014, *ApJ*, 788, 48

Silla, C. N. J. & Freitas, A. A., A survey of hierarchical classification across different application domains. 2011, Data Mining and Knowledge Discovery, 22, 22

Smith, C. R. & Erickson, G. 2012, Maximum-Entropy and Bayesian Spectral Analysis and Estimation Problems: Proceedings of the Third Workshop on Maximum Entropy and Bayesian Methods in Applied Statistics, Wyoming, USA, August 1–4, 1983, Vol. 21 (Springer Science & Business Media)

Spitler, L. G., Cordes, J. M., Hessels, J. W. T., et al., Fast Radio Burst Discovered in the Arecibo Pulsar ALFA Survey. 2014, *ApJ*, 790, 101

Spitler, L. G., Scholz, P., Hessels, J. W. T., et al., A repeating fast radio burst. 2016, *Nature*, 531, 202

Stovall, K., Lorimer, D. R., & Lynch, R. S., Searching for millisecond pulsars: surveys, techniques and prospects. 2013, *Classical and Quantum Gravity*, 30, 224003

Stubbs, C. W., Doherty, P., Cramer, C., et al., Precise Throughput Determination of the Pan-STARRS Telescope and the Gigapixel Imager Using a Calibrated Silicon Photodiode and a Tunable Laser: Initial Results. 2010, *ApJS*, 191, 376

Szegedy, C., Toshev, A., & Erhan, D., Deep Neural Networks for Object Detection. 2013, NeurIPS

Tibshirani, R., Regression shrinkage and selection via the lasso. 1996, Journal of the Royal Statistical Society: Series B (Methodological), 58, 58

Tisserand, P., Le Guillou, L., Afonso, C., et al., Limits on the Macho content of the Galactic Halo from the EROS-2 Survey of the Magellanic Clouds. 2007, *A&A*, 469, 387

Torrealba, G., Catelan, M., Drake, A. J., Djorgovski, S. G., et al., Discovery of 9000 new RR Lyrae in the southern Catalina surveys. 2015, Monthly Notices of the Royal Astronomical Society, 446, 446

Tsang, B. T. H. & Schultz, W. C., Deep Neural Network Classifier for Variable Stars with Novelty Detection Capability. 2019, *ApJ*, 877, L14

Turing, A. M., On Computable Numbers, with an Application to the Entscheidungsproblem. 1937, *Proceedings of the London Mathematical Society*, s2-42, s2-42

Udalski, A., Kubiak, M., & Szymanski, M., Optical Gravitational Lensing Experiment. OGLE-2 – the Second Phase of the OGLE Project. 1997, Acta Astron., 47, 319

Udalski, A., Szymanski, M. K., Soszynski, I., & Poleski, R., The Optical Gravitational Lensing Experiment. Final Reductions of the OGLE-III Data. 2008, Acta Astron., 58, 69

Udalski, A., Szymański, M. K., & Szymański, G., OGLE-IV: Fourth Phase of the Optical Gravitational Lensing Experiment. 2015, Acta Astron., 65, 1

Vafaei Sadr, A., Vos, E. E., Bassett, B. A., et al., DEEPSOURCE: point source detection using deep learning. 2019, *MNRAS*, 484, 2793

van der Maaten, L. & Hinton, G., Visualizing High-Dimensional Data Using t-SNE. 2008, Journal of Machine Learning Research

van Dokkum, P. G., Cosmic-Ray Rejection by Laplacian Edge Detection. 2001, *PASP*, 113, 1420

VanderPlas, J. T., Understanding the Lomb-Scargle Periodogram. 2018, *ApJS*, 236, 16

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D., Show and Tell: A Neural Image Caption Generator. 2015, Proceedings of the IEEE

Walker, A. R., Calibration of the Cepheid period-luminosity relation. 1988, 4, 89

Wang, C., Deng, C., & Wang, S., Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. 2020, Pattern Recognition Letters, 136, 136

Wang, C.-Y., Essential radio astronomy, by James J. Condon and Scott M. Ransom. 2017, *Contemporary Physics*, 58, 278

Watson, C. L., Henden, A. A., & Price, A., The International Variable Star Index (VSX). 2007, Journal of the American Association of Variable Star Observers (JAAVSO), 35, 414

Wattenberg, M., Fernanda, V., & Johnson, I., How to Use t-SNE Effectively. 2016, *Distill*

Willemsen, P. G. & Eyer, L., A study of supervised classification of Hipparcos variable stars using PCA and Support Vector Machines. 2007, arXiv e-prints, arXiv:0712.2898

Witten, I. H., Frank, E., Trigg, L. E., et al., Weka: Practical machine learning tools and techniques with Java implementations. 1999

Yadav, S. & Shukla, S., Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. 2016

Yang, H. & Fong, S., Moderated VFDT in stream mining using adaptive tie threshold and incremental pruning. 2011, Proceedings of the 13th international conference on Data warehousing and knowledge discovery

Zackay, B., Ofek, E. O., & Gal-Yam, A., Proper Image Subtraction-Optimal Transient Detection, Photometry, and Hypothesis Testing. 2016, *ApJ*, 830, 27

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O., Understanding deep learning requires rethinking generalization. 2016, arXiv e-prints, arXiv:1611.03530

Zhang, H., The Optimality of Naive Bayes. 2004, FLAIR, 2, 2

Zong, B., Song, Q., Min, M. R., et al., Deep autoencoding gaussian mixture model for unsupervised anomaly detection. 2018

Zorich, L., Pichara, K., & Protopapas, P., Streaming classification of variable stars. 2020, *MNRAS*, 492, 2897