Application of Statistical Pattern Recognition and Deep Learning for Morphological Classification in Radio Astronomy

by

Adolf Burger Becker

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in the Faculty of Science at Stellenbosch University

Supervisor: Dr. T. L. Grobler

April 2022

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....April 2022.....

Abstract

Application of Statistical Pattern Recognition and Deep Learning for Morphological Classification in Radio Astronomy

A. B. Becker

Computer Science Division,
Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MSc

April 2022

The morphological classification of radio sources is important to gain a full understanding of galaxy evolution processes and their relation with local environmental properties. Furthermore, the complex nature of the problem, its appeal for citizen scientists and the large data rates generated by existing and upcoming radio telescopes combine to make the morphological classification of radio sources an ideal test case for the application of machine learning techniques. One approach that has shown great promise recently is Convolutional Neural Networks (CNNs). Literature, however, lacks two major things when it comes to CNNs and radio galaxy morphological classification. Firstly, a proper analysis to identify whether overfitting occurs when training CNNs to perform radio galaxy morphological classification is needed. Secondly, a comparative study regarding the practical applicability of the CNN architectures in literature is required. Both of these shortcomings are addressed in this thesis. Multiple performance metrics are used for the latter comparative study, such as inference time, model complexity, computational complexity and mean per class accuracy. A ranking system based upon recognition and computational performance is proposed. MCRGNet, ATLAS and ConvXpress (novel classifier) are the architectures that best balance computational requirements with recognition performance.

Uittreksel

Toepassing van Statistiese Patroon Herkenning en Diep Leer vir Morfologiese Klassifikasie in Radio Astronomie

("Application of Statistical Pattern Recognition and Deep Learning for Morphological Classification in Radio Astronomy")

A. B. Becker

Rekenaar Wetenskap Afdeling,
Departement Wiskundige Wetenskappe,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MSc

April 2022

Die morfologiese klassifikasie van radiobronne is belangrik om 'n volledige begrip van die evolusieprosesse binnein sterrestelsels te ontwikkel, asook die rol wat hul plaaslike omgewings hierin speel. As gevolg van die ingewikkelde aard van die probleem, asook die aantrekkingskrag daarvan vir "burgerwetenskaplikes" en die groot hoeveelhede data wat deur bestaande en opkomende radioteleskope gegenereer word, maak die morfologiese klassifikasie van radiobronne 'n ideale proefgebied vir die toepassing van masjienleertegnieke. 'n Benadering wat belowend lyk, is Konvolusionele Neurale Netwerke (KNNe). Literatuur ontbreek egter twee belangrike dinge as dit kom by KNNe en die morfologiese klassifikasie van radio sterrestelsels. Eerstens is daar 'n analise nodig rondom die identifikasie van oorpassing wanneer KNNe afgerig word om radio sterrestelsels volgens morfologie te klassifiseer. Tweedens word 'n vergelykende studie oor die praktiese toepaslikheid van die KNN-argitekture in literatuur benodig. Albei hierdie tekortkominge word in hierdie tesis aagespreek. Veelvuldige prestasiemetings word vir laasgenoemde vergelykende studie gebruik, soos inferensietyd, modelkompleksiteit, berekeningkompleksiteit en gemiddelde akkuraatheid per klas. 'n Rangorde skema word voorgestel gebaseer op herkenning en berekeningsprestasie. MCRGNet, AT-

LAS en ConvXpress (nuwe bydrae) is die argitekture wat berekeningsvereistes en herkenningsprestasie die beste balanseer.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations: the National Research Foundation of South Africa, the Computer Science Department of Stellenbosch University, Shane Josias for his insights and guidance, Lisa van Staden for her support and optimism, Prof Mattia Vaccari for his help as both a mentor and a guide through the technicalities of astronomy and my supervisor, Dr Trienko Grobler, for all his sacrifices and contributions without which this thesis would not have been possible. I would like to thank my parents, Koos and Vini Becker, for their support throughout my academic studies. Lastly I thank my wife, Elize, who has beared with me through my worst and has allowed me to give my best for this thesis.

Dedications

Hierdie tesis word opgedra aan my vrou, Elize, en my ouers, Koos en Vini.

"The heavens declare the glory of God; and the firmament sheweth His handywork. Day unto day uttereth speech, and night unto night sheweth knowledge." – Psalm 19:1-2

Contents

List of Figures

List of Tables

Glossary

AGN Active Galactic Nucleus.

AI Artificial Intelligence.

ANN Artificial Neural Network.

ASKAP Australian Square Kilometre Array Pathfinder.

CNN Convolutional Neural Network.

CoNFIG Combined NVSS-FIRST Galaxies.

EMU Evolutionary Map of the Universe.

FIRST Faint Images of the Radio Sky at Twenty-Centimeters.

FITS Flexible Image Transport System.

FLOPs floating point operations.

FN False Negative.

FP False Positive.

FR0CAT FR0 Catalogue.

FRI Fanarof-Riley Type I Galaxy.

FRICAT FRI Catalogue.

FRII Fanarof-Riley Type II Galaxy.

FRIICAT FRII Catalogue.

GPU Graphics Processing Unit.

LRG dataset Labeled Radio Galaxy dataset.

MeerKAT Meer Karoo Array Telescope.

MPCA Mean per Class Accuracy.

MURG dataset Modified Unlabeled Radio Galaxy dataset.

NRAO National Radio Astronomy Observatory.

NRF National Research Foundation.

NVSS NRAO VLA Sky Survey.

PB Peta Byte.

PyBDSF Python Detector and Source Finder.

SARAO South African Radio Astronomy Observatory.

SKA Square Kilometre Array.

SMBH Super Massive Black Hole.

SWIRE The Spitzer-Space-Telescope Wide-area Infrared Extragalactic Survey.

TN True Negative.

TP True Positve.

unLRG dataset unlabeled Radio Galaxy dataset.

VLA Very Large Array.

VLASS Very Large Array Sky Survey.

XOR Exclusive OR function.

Chapter 1

Introduction

> *"The time will come when diligent research over long periods will bring to light things which now lie hidden. A single lifetime, even though entirely devoted to the sky, would not be enough for the investigation of so vast a subject... And so this knowledge will be unfolded only through long successive ages. There will come a time when our descendants will be amazed that we did not know things that are so plain to them... Many discoveries are reserved for ages still to come, when memory of us will have been effaced."*
>
> — Lucius Annaeus Seneca, *Quaestiones Naturales*

1.1 Overview

Radio astronomy is undergoing a significant change in observational capabilities in the buildup to the Square Kilometre Array (Braun *et al.*, 2015; Bourke *et al.*, 2018, SKA). This rapid development comes with a unique set of challenges, primary of which is the massive data output from these radio telescopes. SKA precursors such as the Australian Square Kilometre Array Pathfinder (ASKAP) (Johnston *et al.*, 2008) and the enlarged Karoo Array Telescope (named MeerKAT) (Jonas and MeerKAT Team, 2016) output several petabytes (PBs) of data per year. The SKA is eventually expected to produce approximately 300 PB per telescope per year during full operation, resulting in a 8.5 Exabyte archive over the 15-year lifespan of the project (Scaife, 2020).

The SKA's pathfinders and precursors (Norris *et al.*, 2013) could potentially revolutionize the field of radio astronomy. Very Large Array Sky Survey (VLASS) (Lacy *et al.*, 2020) and Evolutionary Map of the Universe (EMU) (Norris *et al.*, 2011) are two ongoing surveys that are expected to detect up to roughly 5 and 70 million radio sources, respectively. To date, only roughly 2.5 million radio sources are known. Historically, scientific analysis used catalogues compiled by either individuals or small teams (Fanaroff and Riley, 1974). However, the increasingly large samples of radio sources detected by modern radio telescopes means that

the classification of full catalogues by subject matter experts is no longer a viable option (Hocking *et al.*, 2015).

The recent advances in machine learning tasks related to image classification are a potential solution, providing near human accuracy. Convolutional Neural Networks (CNNs) are a popular choice for image recognition problems in both academia and industry (LeCun *et al.*, 1989; Goodfellow *et al.*, 2016). A CNN is a special type of neural network that learns which features are important to extract from images. It then employs these learned features to perform classification. A prime example of a CNN architecture that has been successful in classifying images is AlexNet, which was one of the first CNN architectures to achieve human-level performance on the ImageNet Challenge. This involved classifying 1.2 million images into 1000 different classes (Deng *et al.*, 2009; Krizhevsky *et al.*, 2012).

Several CNNs have been developed specifically for radio astronomy, starting with Aniyan and Thorat (2017)'s classifier Toothless which was inspired by AlexNet. The rising popularity of Deep Learning led to a series of articles which proposed novel CNN architectures for classification of radio sources, most of these were not comparable with their predecessors for various reasons, mostly due to being trained on different datasets. Furthermore, the models developed during these studies were often trained on small sample sizes, which make it difficult to assess generalization to a larger dataset. Only one of these studies report on computational requirements beyond trainable parameter count. An additional problem caused by the classifiers being trained on different datasets is that we cannot compare which of these architectures are more prone to overfitting than the others. Significant research has been done in creating new classifiers, but with no means to compare these studies, it is difficult to assess the performance and practicality of these methods at scale.

1.2 Problem Statement

Large datasets are expected in the Exabyte range from new radio astronomy surveys, which would limit the practicality of subject matter expert image classification. A method of automatic classification is needed that can classify such large datasets reliably and fast. Significant research has been done to automate the morphological classification of radio galaxies, with several studies proposing new classifiers for this task. However, no study has compared the performance of these classifiers. Most classifiers have been trained on datasets of different sizes and composition, making comparison even more difficult. As such, an assessment of which classifiers are more prone to overfitting is also lacking. The relevant studies in radio astronomy has given in-depth information regarding classification performance metrics but focuses very little on computational requirements.

In short, no comparison study exists to assess the recognition and computational performance of the CNN architectures that have been proposed for the morphological classification of radio galaxies. When considering such a study, an assessment of overfitting and the effectiveness of regularization interventions is also crucial. Such a study will make it possible to select the best architectures for deployment and aid in the future development of better performing architectures.

1.3 Objectives

To address these problems, several CNN architectures from the radio astronomy literature were retrained on the same dataset. We then assess how prone the architectures are to overfitting relative to each other and test various regularization interventions on those that overfit. The most effective interventions then replace the original models in the recognition performance comparison study. Architectures were also assessed on their computational requirements. A simple ranking system is proposed based on recognition performance and computational requirements.

The objectives of this study are:

1. **Identifying proneness to overfitting:** After all the architectures from the selected studies are retrained on the same dataset, we assess how prone the architectures are to overfitting relative to each other.

2. **Assessing regularization intervention effectiveness:** After identifying which architectures are more prone to overfitting, we test the various regularization interventions from the literature of selected studies on those that overfit. The most effective interventions are then applied to these overfitting prone architectures and replace the original models in the recognition performance comparison study.

3. **Assessing recognition performance:** Classifier performance is then assessed and compared with regards to precision, recall, F1-score and mean per class accuracy.

4. **Reporting computational requirements:** Besides recognition performance, it is important to assess the computational requirements of various CNNs when considering which to use in practice. We assess and report the computational requirements of each architecture in literature in this thesis.

5. **Ranking system:** A ranking system has been developed to assess and compare CNNs that take into account the recognition performance and computational requirements of each architecture.

1.4 Contributions

1.4.1 Publications

- B. Becker and T. Grobler, Classification of Fanaroff-Riley Radio Galaxies using Conventional Machine Learning Techniques, *2019 International Multi-disciplinary Information Technology and Engineering Conference (IMITEC)*, Vanderbijlpark, South Africa, 2019, pp. 1-8 (Becker and Grobler, 2019).

- B. Becker, M. Vaccari, M. Prescott, M. and T. Grobler, 2021, CNN Architecture Comparison for Radio Galaxy Classification. *Monthly Notices of the Royal Astronomical Society*, 503(2), pp.1828-1846 (Becker *et al.*, 2021).

1.4.2 Software

Software contributions that have been made are available publicly on Github at: https://github.com/BurgerBecker/rg-benchmarker

1.5 Thesis Outline

The thesis outline is summarized below:

- Chapter 2 discusses the background and theory surrounding radio astronomy. The radio astronomy section of this chapter discusses the radio galaxy morphology taxonomy that subject matter experts use to classify radio sources.

- Chapter 3 discusses the background and theory of CNNs. This chapter starts with a historical overview of the development and adoption of CNNs. The chapter then discusses the layers that a CNN is composed of. Overfitting is defined and methods of addressing overfitting through regularization is discussed. Lastly the metrics we used to evaluate the models from the experiments we conducted in this thesis are presented.

- Chapter 4 introduces the data used in this thesis and the acquisition thereof.

- Chapter 5 provides an overview of the radio galaxy CNN literature. The chapter starts by listing the different architectures we considered in this study. We then discuss the contributions of each study, i.e how did each study address overfitting and which regularization strategies did they utilize. We also mention the minor modifications made to some architectures to be suitable for the comparison study.

- Chapter 6 presents the experimental setup. The various experiments are laid out and discussed, starting with the identification of architectures prone to overfitting in Experiment 1 (see point 1 from Section 1.3). We then discuss Experiment 2 in which we assess the effectiveness of the various regularization strategies used in the literature (point 2, Section 1.3). Lastly, we discuss Experiment 3 which compares the architectures' performance on the test set (point 3, Section 1.3) and their computational complexity (Section 1.3, point 4). This experiment also discusses and introduces the ranking system (point 5, Section 1.3).
- Chapter 7 presents the results of the experiments in Chapter 6.
- Chapter 8 offers our conclusion and closing remarks.

1.6 Grant Acknowledgement

This work is based on the research supported wholly or in part by the National Research Foundation (NRF) of South Africa grant number 117275.

Chapter 2

Background: Radio Astronomy

> *"The total amount of energy from outside the solar system ever received by all the radio telescopes on the planet Earth is less than the energy of a single snowflake striking the ground."*
>
> — Carl Sagan, *Cosmos*

This chapter gives the background context which is needed to understand the material relating to Radio Astronomy presented in the thesis. In Section 2.1 the instruments that are used to observe radio sources are described. In Section 2.2 the taxonomy scientists use to group the different radio sources is presented.

2.1 Radio Telescopes

Earth's low atmospheric opacity within the radio frequency range, from 15MHz to 300MHz (20m to 1mm wavelength) provides a unique opportunity to study radio-loud astronomical sources from the planet's surface with minimal loss of signal quality. The field of radio astronomy focuses on studying sources detected in the radio window of the electromagnetic spectrum via radio telescopes.

Radio telescopes are radio receivers made up of specialized antennae that observe radio waves emitted by astronomical sources. These astronomical sources are extremely distant, ranging from exosolar objects to galaxies at the edge of the observable universe, and therefore have very weak radio signals by the time it reaches earth. Radio telescopes require sensitive receivers and large surface areas to collect and/or focus enough energy from the signal in order to study it meaningfully. This can either be done by increasing the dish size (such as the Arecibo Observatory dish with a 305 meter diameter in Figure 2.1) or combine the signal of an array of smaller telescopes to emulate a larger telescope through radio interferometry, such as the Very Large Array (VLA) (see Figure 2.2).

The SKA radio telescope is of the latter type and will after completion have a collecting area of more than one square kilometre. The SKA radio telescope will

Figure 2.1: The Arecibo Observatory dish in Puerto Rico is an example of a single dish radio telescope. Operations began in 1963, the 305 meter diameter dish claimed the title of largest single filled aperture telescope for 53 years until 2016 with the completion of the Five-hundred-meter Aperture Spherical Telescope (FAST) in Kedu, Peoples Republic of China. The Arecibo Observatory dish was decommissioned on 19 November 2020 due to safety concerns. Before controlled demolition could commence, on 1 December 2020 the main telescope collapsed.

be co-located in South Africa and Australia. The South African Radio Astronomy Observatory (SARAO) is responsible for managing all the South African radio astronomy facilities and initiatives, including the South African contribution to the SKA project. SARAO is a facility of the NRF.

2.2 Morphological Classes of Radio Galaxies

Since the days of Hubble, astronomers have been progressively creating more sophisticated classification schemes to group galaxies based on their shapes observed at optical wavelengths into seperate classes. Morphological classification has become a fundamental aspect of studies relating to galaxy formation and evolution, therefore, critical to modern astronomy as a whole. The shape of a galaxy is inti-

Figure 2.2: The VLA located in Socorro, USA, is an example of an array telescope, consisting of 27 radio antennas. Each antenna in the array measures 25 meters (82 feet) in diameter and weighs about 230 tons.

mately tied to the dynamical and physical processes at play within and around it (Hubble, 1926; de Vaucouleurs, 1959; Sandage, 1961; Elmegreen and Elmegreen, 1987). Current understanding of galaxy formation and evolution state that every massive galaxy is believed to contain a supermassive black hole (SMBH) which undergoes periods of accretion throughout cosmic time to produce an Active Galactic Nucleus (AGN). AGNs are often detected in radio surveys via their synchrotron emission produced by accelerated electrons in their cores, lobes and jets (see Figure 2.3). These are referred to as radio-loud AGN.

Radio galaxies are morphologically distinguished by the shape of their jets and lobes and the position of their hotspots in these jets, relative to their core. Figure 2.3 illustrates the mechanism whereby energy is expelled from the core of the AGN in jets that feed into lobes around the galaxy.

The first morphological classes of radio galaxies were split based on the ratio of inter-lobe "hotspot" distance to total length. A hotspot is an area of high intensity in either one of the lobes. This is called the Fanaroff-Riley ratio (Fanaroff and Riley, 1974) (see Figure 2.4). The division was postulated to lie at a ratio of 0.5 with a corresponding division in radio luminosity. Fanaroff-Riley Type I (FRI) were more likely to have the hotspots near the core and Fanaroff-Riley Type II (FRII) with the hotspots near the edges. Figure 2.5 depicts an example of an FRI and FRII galaxy. There are also Bent-tailed sources whose lobes do not form in a straight line. Finally there are Compact sources, which are simply one

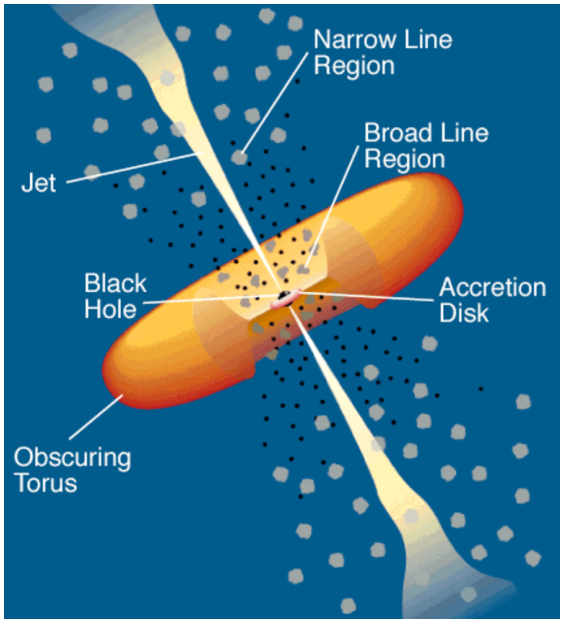

Figure 2.3: Schematic representation of the structure of a radio-loud Active Galactic Nucleus

undifferentiated core with no jets and lobes. These are the morphological classes that are considered in this study.

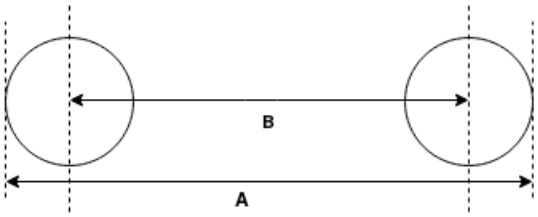

Figure 2.4: A diagram of the Fanaroff-Riley ratio. The ratio is defined as the inter-lobe hotspot distance (B) divided by the total source length (A)

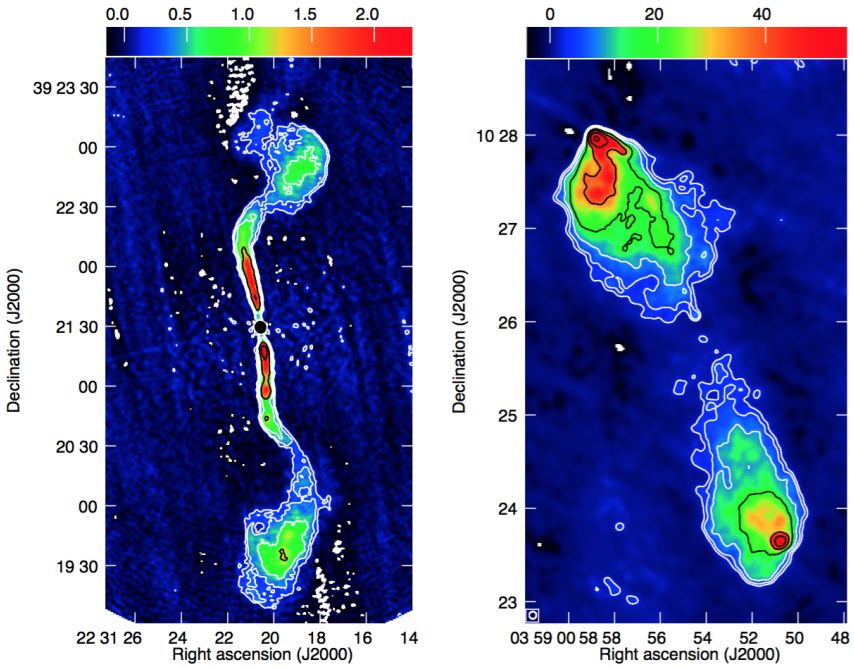

Figure 2.5: The FRI galaxy 3C 449 (left) and the FRII galaxy 3C 98 (right) illustrate the FR morphology. The red area indicates the brightest radio emission. For the FRII's the edges are much brighter in radio emissions, while the FRI's have hotspots closer to the core region. Reproduced from *Extragalactic Jets from Every Angle*, Proceedings of the International Astronomical Union, IAU Symposium, Volume 313, pp. 211-218 with permission (Kharb *et al.*, 2015)

Chapter 3

Background and Theory: Convolutional Neural Networks

> *"The central challenge to solid mathematical, scientific understanding of intelligence today is to understand and replicate the highest level of intelligence that we can find even in the brain of the smallest mouse. This is clearly far beyond the qualitative level of capabilities or functionality that we find in even the most advanced engineering systems today."*
>
> — Paul Werbos, *A Scientific/Engineering View of Consciousness and How to Build It*

This chapter gives the background context which is needed to understand the material related to CNNs presented in the thesis. In Section 3.1 a concise introduction to machine learning and its aims are given. In Section 3.2 a brief historical overview is given of Artificial Neural Networks (ANNs) and CNN development and adoption. The building blocks and training of ANNs are discussed in Section 3.3. The basics of CNNs are discussed in Section 3.4. The problem of overfitting is discussed in Section 3.5. Regularization and methods to address overfitting is discussed in Section 3.6. The metrics we used in this thesis to evaluate the CNNs from radio astronomical literature are summarized in Section 3.7.

3.1 Machine Learning

Machine learning is a subset of the field of Artificial Intelligence (AI). By using sample data, machine learning algorithms develop models that can perform a specific task with data that it previously has not been exposed to. This ability to perform a specific task well on previously unobserved inputs is referred to as generalization. The model learning from sample data is called training and the sample data is referred to as training data.

Mitchell (1997) provides an operational definition of machine learning:

- "The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience."
- "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Various machine learning models have been developed in which some task is learned based on experiences or data. For the purposes of this thesis we will be focusing on the task of image classification. During a classification task, the model is required to state whether the input item is part of one of k classes. Machine learning algorithms can be broadly categorized into supervised or unsupervised learning tasks, based on the way the model has been trained, although this is not a strict definition since neither are formally defined terms and degrees of overlap occur between both (Goodfellow *et al.*, 2016).

3.1.1 Unsupervised Learning

In unsupervised learning, the training data the model is exposed to has no corresponding labels. The model learns a probability distribution that can generate representative examples of the training set that it was exposed to.

3.1.2 Supervised Learning

When using a supervised learning approach, the model is given input-output pairs to learn the correct associations for specific data. Supervised learning requires an input $\boldsymbol{x}_j$ and a corresponding label $\boldsymbol{y}_j$ that the model can learn from. The label represents the desired output or class of the input.

All of the studies considered in this thesis follow a supervised learning approach.

3.1.3 Task: Image Classification

A specific task is required to assess performance of the learning algorithm. Image classification is one such task and the focus of our study. Algorithms performing image classification attempt to place an image into one of a set of predefined classes based on the contents of the image as a whole (Wang and Su, 2020). This is done by assigning the image a class label. This task is typically performed on images where only a single object is present, in contrast to object detection that performs both classification and localization of multiple objects in an image. We do not, however, train any models for this task.

3.2 Historical Overview

CNNs have within the last decade revolutionized the field of computer vision, attaining human-like accuracy levels when it comes to recognition and classification tasks. Although these recent innovations have made headlines in mainstream media and captured the popular imagination as the "ascendancy" of AI, CNNs have a history stretching back to the mid-20th century, long before computers became ubiquitous.

3.2.1 McCulloch-Pitts Neuron

Walter Pitts was pivotal to the development of the idea of an artificial neuron. He had taught himself Greek, Latin, mathematics and number theory before reaching puberty, despite a difficult childhood with a family who thought it better that he join the workforce than stay in school. An anecdote tells that while a 12 year old Pitts was being chased through the streets of Detroit, he went and hid in a public library. There he found Alfred Whitehead and Bertrand Russel's three volume *Principia Mathematica*, spending the next three days in the library reading all 2000 pages. The *Principia* was an attempt to reframe mathematics from the perspective of pure logic. The young Pitts sent Russel a letter listing errors he had found. Russel invited Pitts to Cambridge University, which he turned down due to his young age. Pitts did however run away from home three years later when he found out Russel would be visiting the University of Chicago. Pitts was destitute, when he met Jerome Lettvin, a young medical student. Lettvin would end up introducing him to Warren McCulloch, a neurophysiologist with a background in mathematics. Pitts and McCulloch shared a common interest in the work of the great German polymath Gottfried von Leibniz. One particular shared interest was for what Davis (2002) would later refer to as: *Leibniz Dream*. According to Davis, Leibniz "dreamt of an encyclopedic compilation, of a universal artificial mathematical language in which each facet of knowledge could be expressed, of calculational rules which would reveal all the logical interrelationships among these propositions. He also dreamt of machines capable of carrying out calculations, freeing the mind for creative thought." McCulloch, being further inspired by Alan Turing's work showing the possibility of a machine capable of computing any function (given that the computation can be carried out within finite steps), had developed the belief that the human mind was such a machine and was interested in mathematically formulating the mechanism by which it functioned, from the neuron upward. McCulloch explained his theory to Pitts, whose in depth understanding of number theory proved useful in underpinning the idea mathematically.

Following this, McCulloch invited both Pitts and Lettvin to live with him and his family. Pitts was destitute at this point in his life.

McCulloch and Pitts went about formalizing their theory which culminated in their seminal work: "A Logical Calculus of the Ideas Immanent in Nervous Activity" (McCulloch and Pitts, 1943), which proposes an artificial neuron that sums the inputs to it and activates if the values are above a certain threshold. This activation mechanism is similar to what McCulloch refers to as the "all-or-none" activation characteristic of how biological neurons were believed to work at the time, enough of the neighbouring neurons have to send a signal to a neuron for it to fire itself. The artificial neuron described in their work has become more widely known as the McCulloch-Pitts neuron. The McCulloch-Pitts neuron has served as a foundational principle for how nearly all ANNs work today.

Despite the best efforts of McCulloch, Pitts would never hold any official position at the Massachusetts Institute of Technology (MIT) beyond research associate nor finish his Doctoral dissertation. His work with Levitt on "What the Frog's Eye Tells the Frog's Brain" (Lettvin *et al.*, 1959) found that the frog's eye preprocessed a lot of the visual information before conveying it to the brain, rather than the brain processing all the information pixel by pixel. This pushed Pitts to despair, he refused to sign the paperwork to receive his Ph.D. and he set fire to his Doctoral dissertation. Pitts started drinking heavily and died in 1996 at age 43, McCulloch died 4 months later. Figure 3.1 shows McCullogh, Pits and Lettvin.

McCulloch and Pitts provided a mechanistic view of the human brain, rather than the less quantifiable views popularised by Freud and Jung's psycho-analytic school. A more mechanistic view set the stage for further development of ANNs as a more concrete field of academic study, rather than the more abstract views of cognition held in psycho-analysis at the time.

3.2.2 Perceptron

The perceptron was developed by Frank Rosenblatt and was inspired by the McCulloch-Pitts neuron, a perceptron neuron will fire if the inputs to it are greater than zero. It used a single neuron for classification and had a learning rule (Rosenblatt, 1958). It was eventually shown that the perceptron could not learn the XOR function (Minsky and Papert, 1969).

3.2.3 From Hubel and Wiesel to AlexNet

Hubel and Wiesel (1968) found that mammalian visual cortices primarily consist of two types of cells, i.e. simple cells (that would activate when straight edges had a certain orientation) and complex cells (with a larger receptive field and lower sensitivity to orientation). This inspired the Neocognitron (Fukushima, 1980) which combined layers consisting wholly of one of two types of "cells" into a hierarchical model that could perform handwritten character recognition. One type of cell

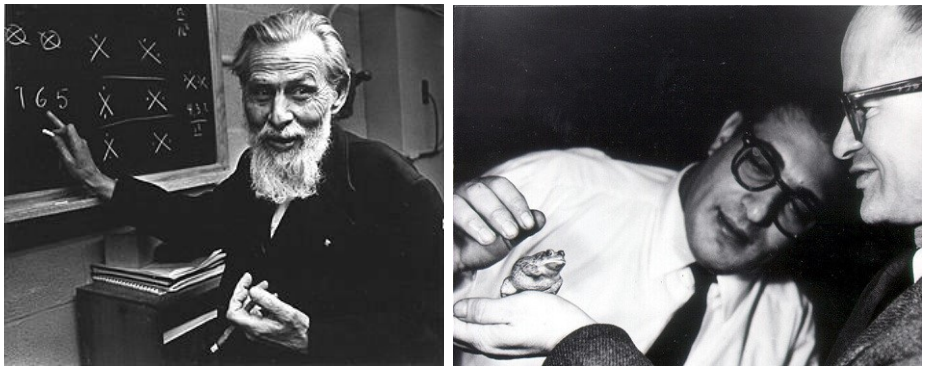

Figure 3.1: Warren McCulloch lecturing [left image]. Walter Pitts (right) and Jerome Lettvin inspect a frog [right image]. Both were co-authors of "What the Frog's Eye Tells the Frog's Brain" (1959) together with McCulloch, one of the most cited papers in history, the findings of which pushed Pitts to despair and alcoholism. McCulloch and Pitts developed the logical underpinnings that have been foundational to the development of artificial neural networks.

would apply a convolutional operation to the input, while the other would down-sample the input. The weights for the convolutional operation would be learned from input examples. The first deep CNN (LeCun *et al.*, 1998) had seven layers and was primarily used for handwritten character recognition, although training such deep models was computationally very expensive and time consuming. The widespread advent of Graphics Processing Units (GPU) within desktop computers, however, resulted in more people being able to quickly train deep CNNs (Cireşan *et al.*, 2010). Furthermore, Deep Learning in general and CNNs in particular became the *de facto* standard for image classification after the 2012 ImageNet Challenge was won by a CNN, i.e. AlexNet (Krizhevsky *et al.*, 2012).

3.2.4 History of Backpropagation

The back propagation of error (referred to commonly as backpropagation) is a method developed to efficiently calculate the gradient of the loss function and is described further in Section 3.3.5. Using backpropogation to train neural networks was first proposed by Paul Werbos in 1971 (Werbos, 1974). This was published in his 1974 Doctoral dissertation and was initially used to train a model to predict social communication and nationalism.

Sadly, the work Werbos did remained mostly unnoticed by the scientific community in the early 1980's. The aforementioned backpropagation algorithm was independently rediscovered by Rumelhart *et al.* (1986). In "Backpropagation through time: what it does and how to do it" Werbos (1990) clarifies and makes concise the central ideas of backpropagation. Werbos' original thesis was later reprinted

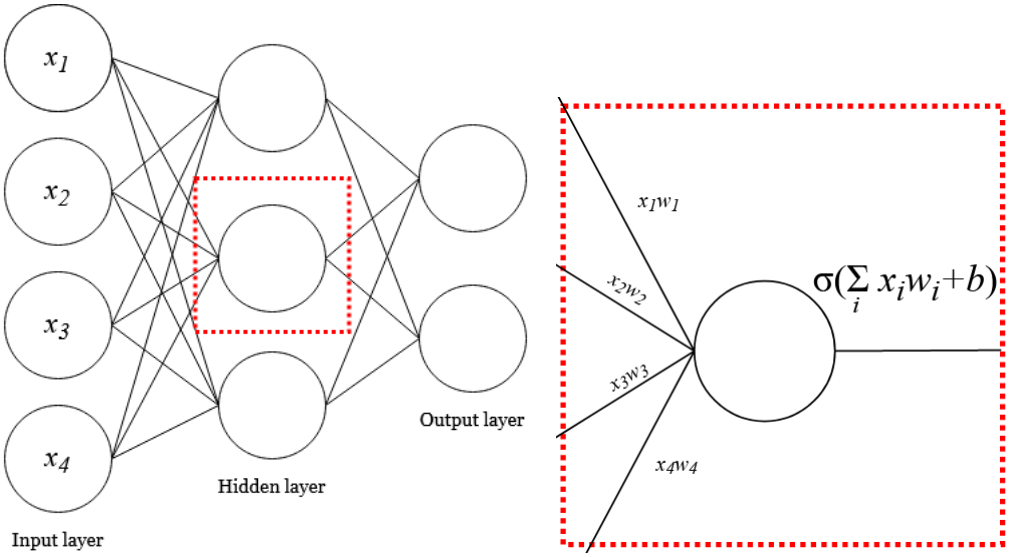

Figure 3.2: An ANN example of a 3 layer structure, with one input layer, one hidden layer and an output layer. The input layer receives and distributes the data to the hidden layer neurons. Each connection between neurons has a weight factor (depicted as w on the right hand side of the figure) and a bias term. The sum of all the inputs to a neuron and its bias term have an activation function applied to it (depicted as σ in the figure; σ is usually a non-linear function). This is the output of this neuron, which in this case gets sent to the output layer's neurons, where the same process is repeated, except that the output of this neuron's layer represents the probability of the data being in this class.

as part of Werbos' larger work "The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting" (Werbos, 1994). In 1995 Werbos was awarded the IEEE Neural Network Pioneer Award for his discovery of backpropagation.

The first application of backpropagation to a Neocognitron-esque CNN was done by LeCun *et al.* (1989).

3.3 Artificial Neural Networks

Neural networks can in general be visualized as graph-like structures in which nodes are referred to as neurons (see Figure 3.2). Each edge or connection to another neuron has a weight term, which represents the strength of its connection. All non-input neurons also have a bias parameter. The weights and biases affect the propagation of information through the network. With the right combination of weights, the network can match input from a certain class to its respective class

label reliably. The neurons are arranged in layers, with an input being propagated from the input layer, through intermediate layers (called hidden layers) until it reaches the output layer.

ANNs are normally composed of fully connected or dense layers. The neurons of a fully connected layer are connected to all the neurons of the previous layer. The output layer is also a fully connected layer. When the ANN is being used for classification the output layer has as many neurons as the number of classes that have been provided. The main difference, within the context of the current deep learning regime, between the output layer and other layers is the activation function being applied to that layer's input. For example, for all the architectures selected in this study the softmax activation function was used in the output layer while either ReLU or a linear activation was used in the

3.3.1 Deep Learning Frameworks: Keras and Tensorflow

The Keras deep learning framework (Chollet *et al.*, 2015) was made use of during our study, which is an open-source software library with a Python language Application Programming Interface (API) for the Tensorflow deep learning framework (Abadi *et al.*, 2015). Keras is useful for fast experimentation and prototyping with ANNs. Several technical terms have different implementations within the various deep learning frameworks (such as dropout, see Section 3.6.2) and all such terms are discussed within the context of the Keras and Tensorflow implementations.

3.3.2 Activation Functions

In an ANN, a neuron typically has an activation function that is applied to the neuron's inputs to provide an output. When a multi-layer ANN has only linear activations, the output is a linear transformation of the input. Since these activations are commutative, the network structure of the ANN with only linear activations can be condensed to two layers: the input and the output layers. A two layer neural network with non-linear activation has been shown to be a universal function approximator (Cybenko, 1989). For ANNs non-linear activation functions are commonly used (although linear activations are also used), which can be used to model more complex relations. Some commonly used activation functions in ANNs are discussed in the following subsections.

3.3.2.1 Logistic sigmoid

The logistic sigmoid (or simply 'sigmoid') function was one of the earliest activation functions used in ANNs. The sigmoid function applied to any real value maps it to a value between zero and one. As the magnitude of the input increases,

the sigmoid's gradient decreases rapidly, which leads to the vanishing gradient phenomenon (see Section 3.3.5.5). This has lead to the waning of the activation function's popularity. Equation 3.1 is the activation function mathematically expressed, with x representing the input to the function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{3.1}$$

Equation 3.2 represents the derivative of the sigmoid function:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)). \tag{3.2}$$

3.3.2.2 ReLU

Another activation function used in ANNs (which has become more commonly used within the current deep learning dispensation) is the rectified linear unit (ReLU) which is mathematically defined as:

$$\sigma(x) = \max(0, x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{3.3}$$

$$\sigma'(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{3.4}$$

Although the derivative of ReLU is technically undefined at zero, Equation 3.4 is commonly used. ReLU is much less prone to the vanishing gradient problem (see Section 3.3.5.5) than the sigmoid activation function is and is part of the reason why it has seen such widespread adoption (see Section 3.3.5.5).

3.3.2.3 Softmax

The softmax function is the activation function which is commonly used within the current deep learning context in the output layer for multi-class classification. This activation function normalizes an input vector's values to the range between zero and one to represent a probability associated with each class. The neuron with the highest output value determines the classification result, with a higher value meaning greater model confidence in the classification. The softmax function $\boldsymbol{\sigma} : \mathbb{R}^n \to [0, 1]^n$ is defined as:

$$\sigma_i(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \text{ for } i = 1, ..., n \text{ and } \boldsymbol{x} = (x_0, ..., x_n) \in \mathbb{R}^n \tag{3.5}$$

with n being the size of the input vector x. The derivative of Equation 3.5 is expressed as:

$$\frac{\partial \sigma_i}{\partial \boldsymbol{x}} = \begin{cases} \sigma_i(\boldsymbol{x})(1 - \sigma_i(\boldsymbol{x})) & i = j \\ \sigma_i(\boldsymbol{x})\sigma_j(\boldsymbol{x}) & i \neq j \end{cases} \tag{3.6}$$

The softmax function takes a vector as input and as output gives a vector with each element in the input affecting each element in the output. The softmax function is a generalization of the sigmoid function to an n-dimensional categorical probability distribution. As such, softmax is used as the output layer for multiclass classification.

3.3.3 Loss Function

The loss function C measures how well the model performs on a specific task. A loss function is used to calculate the error between the current predicted output y'_j and the correct label y_j for a specific set of weights and biases. Alternative terms for the loss function include the error or cost function. The loss function used for classification is categorical Cross-entropy loss. This is also the loss function used in this thesis.

$$C(\boldsymbol{y'}, \boldsymbol{y}) = -\sum_{j=1}^{n} y_j \log(y'_j) \tag{3.7}$$

3.3.4 Forward Pass

Consider a small ANN consisting of an input layer, hidden layer and an output layer such as the one pictured in Figure 3.3. The example ANN has labelled weights w^l_{jk}, neuron outputs a^l_j, activation function σ and the model prediction y'. The generalized example ANN we refer to in the rest of Section 3.3 has L layers.

The input layer's neurons take as input the individual elements of the data instance $\boldsymbol{x}$. The input layer has the same number of neurons as the input dimension n.

Let z^l_j correspond to the cumulative inputs to the jth neuron in layer l, i.e. the sum of the weighted input to that neuron and the jth neuron's bias parameter b^l_j. Weights are represented as w^l_{jk} with k representing the neurons in layer $l-1$. More formally,

$$z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j \tag{3.8}$$

The output of neuron j in layer l is given in Equation 3.9 after an activation function σ is applied to what we will refer to from here on as the pre-activation term z^l_j.

$$a^l_j = \sigma(z^l_j) \tag{3.9}$$

For input in the first layer there is no activation function.

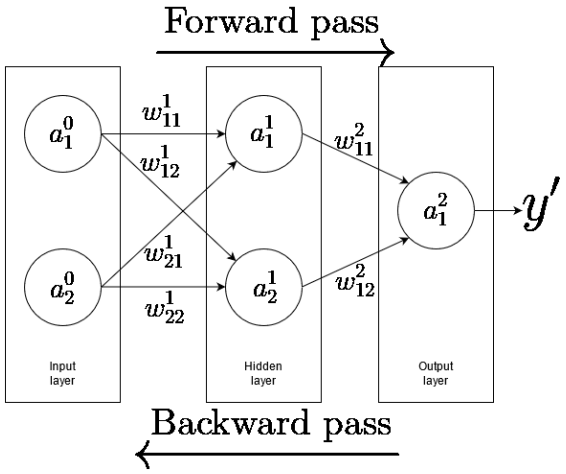

Figure 3.3: Example ANN with labelled weights (w_{jk}^l), neuron outputs (a_j^l) and the model prediction (y'). The arrows indicate the flow of the network during a forward pass (as in the case of inference) or a backward pass (during parameter updates).

$$a_j^0 = x_j \tag{3.10}$$

For a forward pass the input values (x_j) are fed into the input neurons in layer $l = 0$. The output of the input neurons are a_1^0 and a_2^0. These values are then passed to the hidden layer neurons, where their weighted sum is calculated and a bias term is added (see Equation 3.8) to which the activation function is applied (Equation 3.9). The neuron outputs are then passed to the output layer, where the weigted sum of the activations in the previous layer is computed and a bias term is added. The final activation function is then applied which provides the output probabilities. We designate this last layer's outputs as a_j^L (alternatively as y'_j).

During training, validation and testing an additional step takes place to calculate the error using a loss function and given labels.

3.3.5 Backpropagation

ANN training is done by minimizing the loss function with respect to the parameters of the neural network. Gradient descent is used to minimize the loss by iteratively moving in the direction of the gradient's downward slope until a local or global minimum is reached. This gradient calculation is done efficiently with the backward propagation of errors algoritm (backpropagation).

3.3.5.1 The Chain Rule

The chain rule allows us to decompose the gradient of a function. Given two functions $y = f(u)$ and $u = g(x)$ the derivative with respect to x takes the form of Equation 3.11.

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x} = f'(u)g'(x) \tag{3.11}$$

3.3.5.2 Derivatives

In order to minimize the loss function with respect to the ANNs parameters, we apply gradient descent. To do this, we require the gradient of the network's loss function with respect to the network's parameters. Making use of the chain rule in Equation 3.11 we calculate the derivative of the loss function C with respect to the weights and the biases as follows:

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j}\frac{\partial z^l_j}{\partial w^l_{jk}} \tag{3.12}$$

The derivative of the pre-activation term z^l_j from Equation 3.8 with respect to the weight w^l_{jk} simplifies to a^{l-1}_k since no other term in Equation 3.8 has the w^l_{jk} factor.

We will use the following shorthand notation to represent the loss term's derivative w.r.t. the pre-activation term:

$$\Delta^l_j := \frac{\partial C}{\partial z^l_j} \tag{3.13}$$

We can now rewrite Equation 3.12 as:

$$\frac{\partial C}{\partial w^l_{jk}} = \Delta^l_j a^{l-1}_k \tag{3.14}$$

Similarly, the derivative of the loss function with respect to a specific bias term can be computed via Equation 3.15 and simplifies to Equation 3.16.

$$\frac{\partial C}{\partial b^l_j} = \frac{\partial C}{\partial z^l_j}\frac{\partial z^l_j}{\partial b^l_j} \tag{3.15}$$

$$= \Delta^l_j \tag{3.16}$$

Application of the chain rule to the loss function's derivative w.r.t the pre-activation term of the last layer L; we obtain Equation 3.19.

$$\Delta_j^L = \frac{\partial C}{\partial z_j^L} \tag{3.17}$$

$$= \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{3.18}$$

$$= \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \tag{3.19}$$

Likewise, applying the chain rule to the loss function's derivative w.r.t each pre-activation term in layer $l+1$, we get Equation 3.22.

$$\Delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{3.20}$$

$$= \sum_i \frac{\partial C}{\partial z_i^{l+1}}\frac{\partial z_i^{l+1}}{\partial z_j^l} \tag{3.21}$$

$$= \sum_i \Delta_i^{l+1}\frac{\partial z_i^{l+1}}{\partial z_j^l} \tag{3.22}$$

Recall that the ith pre-activation term in layer $l+1$ is given by Equation 3.24.

$$z_i^{l+1} = \sum_k w_{ik}^{l+1}a_k^l + b_i^{l+1} \tag{3.23}$$

$$= \sum_k w_{ik}^{l+1}\sigma(z_k^l) + b_i^{l+1} \tag{3.24}$$

Since only one of the terms in Equation 3.24 has a factor parameterized by z_k^l , the partial derivative $\frac{\partial z_i^{l+1}}{\partial z_j^l}$ simplifies to Equation 3.25.

$$\frac{\partial z_i^{l+1}}{\partial z_j^l} = w_{ij}^{l+1}\sigma'(z_j^l) \tag{3.25}$$

Substitution of Equation 3.25 into Equation 3.22 we can write the gradient Δ_j^l of as Equation 3.26.

$$\Delta_j^l = \sum_i \Delta_i^{l+1}w_{ij}^{l+1}\sigma'(z_j^l) \tag{3.26}$$

The gradient of each layer is a function of the gradient of the next layer, since the output of any layer is a function of the previous layer's outputs.

Clearly, the gradients associated with layer l depend only on the gradients associated with layer $l+1$. We can, thus, efficiently calculate the gradients with respect to the parameters (Equations 3.12 and 3.15).

We elaborate bellow on how these are used to update the weights and biases.

3.3.5.3 Gradient Descent

Training takes place by updating the parameters for each training sample in an iterative fashion, the weight update mathematically expressed as:

$$w_{jk}^{l} = w_{jk}^{l} - \lambda \Delta_{j}^{l} a_{k}^{l-1} \tag{3.27}$$

$$b_{j}^{l} = b_{j}^{l} - \lambda \Delta_{j}^{l} \tag{3.28}$$

Please note that the weight and bias variables on the left side of the equations represent the updated values, while on the right side the weight and bias variables represent the previous values. The learning rate λ is a hyperparameter that scales the step size of gradient descent. We discuss hyperparameter tuning in Section 6.1.2.

We will now discuss a high level walk-through of the backpropagation algorithm using the gradients and parameter updates.

3.3.5.4 Algorithmic Description

Below follows a description of the backpropagation algorithm:

- During a forward pass of training data in the network, we store the pre-activation z_j^l and activation a_j^l for all layers.
- Then we calculate Δ_j^L using Equation 3.19.
- The gradients Δ_j^l are then calculated for each layer using Equation 3.26, while moving backward from the output layer to the input layer. As we move backward we reuse the gradients associated with layer $l+1$ as well as the z_j^l values computed during the forward pass.
- We calculate the gradient of the loss function with respect to the weights (Equation 3.14) and biases (Equation 3.16) of each layer.
- The weights and biases are updated using the step size in Equations 3.27 and 3.28. The gradients calculated for each layer Δ_j^l and the outputs of each layer a_j^l (computed during the forward pass) are re-used in this step.

To summarize, for each training sample the backpropagation algorithm makes a forward pass (inference), then measures the error using the loss function, goes through the neural network in reverse (backward pass) to measure the error contribution from each parameter and updates the parameters to reduce the error (gradient descent).

3.3.5.5 Vanishing Gradient Problem

As we have shown above, during gradient descent the weight updates are done proportionally to the gradients calculated for each layer. Sometimes this gradient becomes extremely small which prohibits the weights from updating. In extreme cases, this stops the network from learning effectively. This is known as the vanishing gradient problem and since the effect compounds, deep neural networks are especially more prone to it (Goodfellow *et al.*, 2016). Some activation functions (sigmoid, Section 3.3.2.1) are more likely to be affected by this than others (ReLU, Section 3.3.2.2).

3.3.6 Optimisation Algorithm

Variations of optimisation algorithms change the default update equation of gradient descent to something more advanced to improve convergence. Otherwise slow convergence could occur depending on the type of loss landscape or massive oscillations could occur, should there be several local minima.

3.3.6.1 Root Mean Squared Propagation

Root Mean Squared Propagation (RMSProp) (Hinton *et al.*, 2012*a*) was one of the first optimisation algorithms that makes use of adaptive learning rates. An adaptive learning rate is when the step size of each parameter is scaled individually. An adaptive learning rate is used to better achieve convergence since larger learning rates might overstep local/global minima.

3.3.6.2 ADAM

Adaptive moment estimation (ADAM) is an optimization algorithm that can be used to iteratively update a neural network's weights (Kingma and Ba, 2017). A learning rate is kept for every network parameter and is adapted separately as training continues.

ADAM was chosen as the optimisation algorithm to be used in this study since it is efficient. ADAM is recommended by several Deep Learning experts, namely Andrej Karpathy (former OpenAI researcher and Head of AI at Tesla) and Sebastian Ruder. In "An Overview of Gradient Descent Optimization Algorithms", Ruder (2016) found that while there are more similarities than differences between Adam and other adaptive learning rate optimizers. For example, ADAM also stores a decaying average of previous squared gradients in the way that RMSProp does, but ADAM also stores the previous gradients. ADAM, however, does perform better towards the end of optimization and lists Adam as the overall best choice.

3.4 Convolutional Neural Networks

Next we provide a technical overview of CNN structure and functionality starting with layer composition. Although many different types of ANNs have developed over time, CNNs in particular have become some of the top performing classifiers for image recognition.

CNNs have three main types of layers: Convolutional layers, pooling layers and fully connected (or dense) layers. The convolutional layers are usually grouped together to form convolutional blocks. These convolutional blocks are often followed by pooling layers. Most CNNs consist of several convolutional blocks with pooling layers. The output of these layers are then flattened to a single output vector which is then input into dense layers. We have already discussed these so called dense layers in some detail in Section 3.3. The last dense layer is an output layer which has the same number of neurons as there are classes, each representing a probability that the input is part of that class (and usually employs the softmax activation function).

Broadly speaking, the convolutional and pooling layers select the features which are then classified by the dense layer.

3.4.1 Convolutional Layers

In convolutional layers, a convolution operation is performed on a small neighbourhood of pixels which then outputs a single value for the neuron in the next layer. This operation is realized using a small matrix containing trainable weights. This small matrix is known as the kernel. The aforementioned kernel is then moved over the image in strides, with a stride length of 1 moving the centre of the kernel one pixel across or down until the entire image has been covered. The average kernel size that is used is 3×3 pixels, although some networks use larger kernel sizes of up to 11×11 pixels early on in the network to reduce input size while the image still contains a high ratio of noise to information (we will see later that AlexNet and Toothless make use of this). In practice, the kernel is three dimensional and can, therefore, be applied to the red, blue and green channels of images. In later convolutional layers, the kernel is moved across the outputs of the previous layer and not the original image, i.e. the outputs of the previous layer become the input pixels of the current convolutional layer. Furthermore, more than one channel of pixels can be created in subsequent layers if more than one three dimensional kernel is used per layer. In the literature, these "channels" are also known as feature maps.

This many-to-one mapping during convolution leads to the output of a convolutional layer to be downsampled (i.e. to have a smaller output dimension than the input). If downsampling happens too suddenly, this can potentially lead to

the loss of too much information without it being incorporated into the model. One workaround for the downsampling problem is padding the input with zeros around the edges, to ensure the output shape is the same size as the input shape. This padding is often referred to as same or zero padding, whereas the absence of padding is known as valid padding. Kernel size, stride length and padding type are all examples of hyper parameters of a CNN, each of which could affect recognition performance.

The output of the final convolutional layer is then flattened into a one dimensional vector, which then becomes the input of the first fully connected layer.

Please note that for the sake of clarity we often used the term "pixel" in the above descriptive paragraphs instead of "neuron" (they are in fact almost interchangeable concepts in this context). This is important to bear in mind, whilst reading through Section 3.4.2.

3.4.2 Convolution

The convolutional operation used within the convolutional layers of a CNN (as mentioned in the previous section) is defined as:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f-1} \sum_{v=0}^{f-1} \sum_{k'=0}^{f'_n-1} x_{i',j',k'} \cdot w_{u,v,k',k} \text{ with } \begin{cases} i' = i \times s + u \\ j' = j \times s + v \end{cases} . \tag{3.29}$$

In Equation 3.29, $z_{i,j,k}$ is the output of row i and column j's neuron in the kth feature map in convolutional layer l. Moreover, s is the stride length (horizontal and vertical strides are the same for all architectures in this study). Furthermore, f is the size of the receptive field (i.e. the kernel size used) and f'_n is the number of feature maps in the previous layer $(l-1)$. Note that all the architectures we considered in this study made use of a square receptive field. The symbol $x_{i',j',k'}$ represents the output of row i' and column j''s neuron in the k'th feature map and b_k is the bias term in feature map k of layer l. Furthermore, $w_{u,v,k',k}$ is the connection weight between neurons in feature map k of layer l and their input located at row u and column v at feature map k'. Note that, in practice, $z_{i,j,k}$ is first passed through an activation function before it is passed on to the next layer of the network. Convolution is depicted graphically in Figure 3.4.

3.4.2.1 Max Pooling Layers

Pooling layers are used to deliberately downsample the input, reducing the input size while preserving the salient features we want the network to learn. Max pooling layers perform downsampling by moving a kernel across the input and

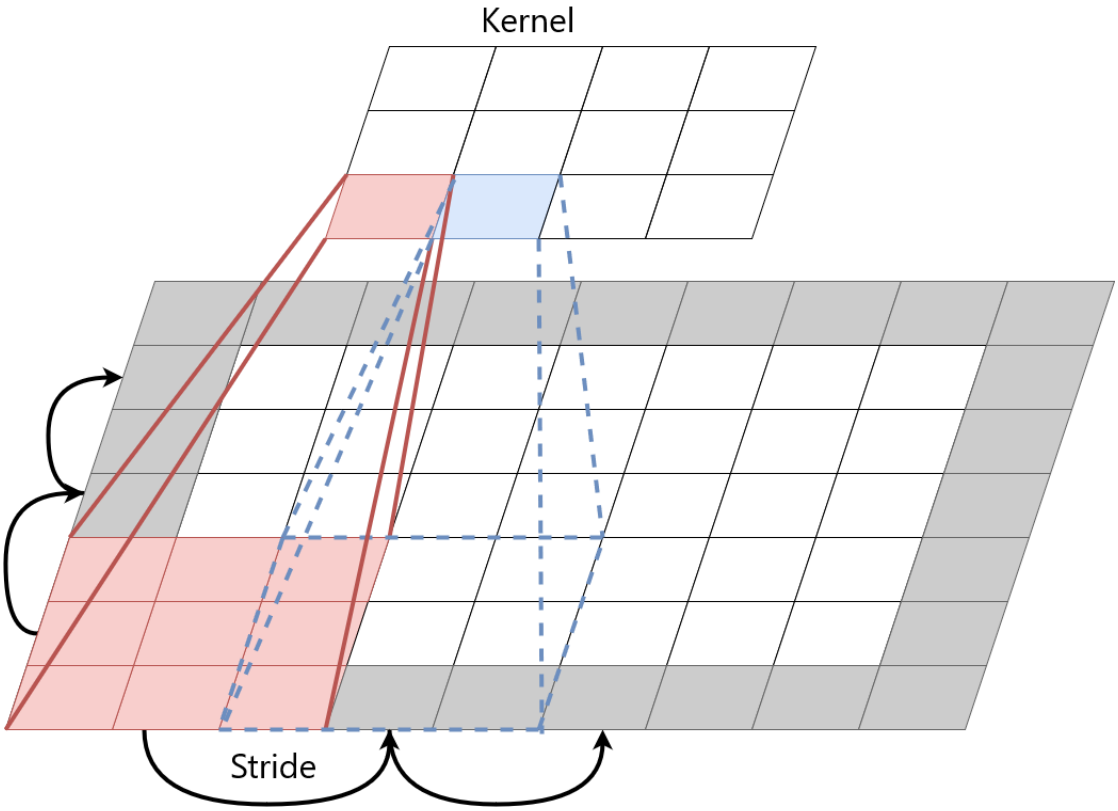

Figure 3.4: A visual representation of a single convolutional layer with a stride length of 2 and a filter size (kernel size) of 3 by 3.

returning only the maximum pixel value within a kernel. Since a max pooling layer's kernel size is normally 2×2, this leads to a size reduction of the input by half.

A reduction in input size is needed so that the computational requirements of deeper layers can be reduced or kept constant.

3.4.3 Rotational Invariance

When a CNN can classify an image irrespective of orientation, it is said to have rotational invariance. CNNs are normally not fully rotationally invariant (Lukic *et al.*, 2019*b*). Convolutional layers enforce translation equivariance and pooling layers add translation invariance but both of these usually allow only limited invariance to rotations, normally not more than a few degrees (Marcos *et al.*, 2016). Other Deep Learning architectures are rotationally invariant by design, such as Capsule Networks (Sabour *et al.*, 2017) which have been considered by Lukic *et al.*

(2019*b*) in a radio astronomy study with the results in favour of traditional CNNs. This is expanded more on in Chapter 5.

3.4.3.1 Receptive Field and Effective Stride

The final convolutional layer's output is not necessarily the result of a transformation applied to every pixel in the input image (unlike a dense/fully connected layer). Rather, each output pixel has a limited "field of view", a limited region in the input image that trickles down through the convolutional layers to become a single output. This is the architecture's theoretical receptive field, as opposed to its effective receptive field (the pixels in that limited region that had the largest impact on the output) (Luo *et al.*, 2017).

3.5 Overfitting

In this section we give a brief introduction to over fitting. We also briefly discuss the bias-variance trade-off and how this pertains to overfitting.

3.5.1 Overfitting a model

When a statistical model corresponds/fits too closely to noise in a particular dataset that it cannot extrapolate or interpolate accurately to new data, the model is said to have been overfitted on the dataset. When training ANNs overfitting is a significant concern. The model will fit to noise that is specific to the training data that is not present in other samples pulled from the same distribution. This will lead to poor generalization performance. The ability to adapt to new previously unseen data by a model is referred to as its ability to generalize.

Overfitting occurs in a model when the loss of the model on the training set is low but high on the validation/test set. Overfitting will likely result in high training accuracy and low validation/test accuracy.

Another method of reducing overfitting is to add regularization techniques (discussed in Section 3.6), which place constraints on your model with regards to the type of information and quantity that can be stored. The idea behind this is that with less capacity, only the most salient features and patterns can be learned.

3.5.2 Capacity

Loosely speaking, the ability of a machine learning model to fit a variety of functions is called capacity. One way of reducing the probability that a model will overfit or underfit is by adjusting its capacity. A model will generally perform

better when its capacity is suitable for the complexity of the task at hand and the number of training samples which are available. Increasing a model's capacity will increase the functions a model can learn and fit to data. Low capacity models are unable to represent all the relations of complex tasks within their hypothesis space and are more likely to underfit. High capacity models can produce models that solve these complex tasks but the capacity might be higher than what is required, at which point the model fits to noise and results in overfitting.

3.5.3 Bias-variance tradeoff

From a theoretical perspective a model's generalization error is expressed as the combination of its bias, variance and irreducible errors (Geman *et al.*, 1992). Generalization error is an accuracy measure of a model on previously unseen data.

Bias errors come from the incorrect assumptions a model makes about a given dataset (e.g. finding a linear trend when the actual function is polynomial). Low capacity models tend to have a high bias error, their limited hypothesis space means that the function fitted to the data will likely be too simplistic. This is when a model is said to be underfitting. Underfitting is when a model insufficiently learns and fits to the training data and has adverse effects on generalization performance.

High capacity models have low bias error, fitting more complex functions to represent the data. However, this may lead to the model fitting too closely to noise present in the data, resulting in poor generalization on new unseen data. This is the variance error that models with large capacity suffer from, being sensitive to small variations in the training data. This is when a model is said to be overfitting.

Irreducible error is a result of the noise present in the data set. This error can be mediated by cleaning up the data through preprocessing and curating or changing the data collection method.

The bias-variance tradeoff comes in to play when increasing a model's capacity, this will reduce its bias but increase its variance. A decrease in the one leads to an increase in the other. Since the generalization error is dependant on both the bias and variance error, an optimal capacity range exists where the sum of the errors is at a minimum where the model is less prone to overfitting and underfitting.

3.6 Regularization

Here we will present some of the most widely used approaches to avoid overfitting, namely dropout, kernel regularization, batch normalization and early stopping.

3.6.1 Batch Normalization

Batch normalization (often shortened to batch norm) reparametrizes the model to introduce multiplicative and additive noise in the hidden neurons during training (Ioffe and Szegedy, 2015). The primary purpose of batch normalization was to improve optimization, but the additive noise can have a similar regularizing effect to dropout.

3.6.2 Dropout

Dropout, as originally proposed by Hinton *et al.* (2012*b*), "severs" some off the weighted connections within the network during training to reduce the reliance of the network on all the neurons. This creates smaller sub-networks within the larger network, effectively creating simpler networks within a larger more complex one. The generalized form developed by Srivastava *et al.* (2014) randomly sets the inputs to a neuron to zero with a probability of p. The dropout rate p is usually set to 0.5 which works for a wide variety of network configurations (Srivastava *et al.*, 2014). The Keras implementation used in this study scales the weighted connections that have not been "cut"/set to zero by a factor of $1/(1-p)$ (Chollet *et al.*, 2015). After training, dropout is no longer active and as such Srivastava *et al.* (2014) proposes that the weights are scaled by the dropout rate when training finalizes. In the Keras framework (Chollet *et al.*, 2015) the weight rescaling is done at training time, at the end of a batch after the weight updates.

Dropout has been shown to work well in practice particularly in combination with ReLU (Dahl *et al.*, 2013) but does not perform well in combination with batch normalization (Li *et al.*, 2019).

3.6.3 Kernel Regularization

Model complexity is sometimes described as a function of weights. Models that have weights with a large magnitude can be viewed as being more complex. These models are also very sensitive to changes in the input variables and thus more sensitive to noise. Kernel regularization adds a penalty to the loss function. This encourages smaller weights, but does not necessarily enforce them (Géron, 2019; Ng, 2004). L_2 kernel regularization is mathematically defined as:

$$C_{L_2} = C + \lambda \sum_i w_i^2 \tag{3.30}$$

This is implemented in Keras as: `loss = l2 * reduce_sum(square(x))`

This result is then added as a penalty to the loss function. The Keras function `reduce_sum` adds all the elements of the kernel together, while the `square` function

computes the square of the input kernel element-wise. This result is then scaled with the `l2` factor (also referred to as λ).

3.6.4 Early Stopping

When a moving average over the validation loss increases more than a threshold, then the training is stopped. This is called overfitting and is an effective counter to overfitting.

3.7 Description of Metrics

The different metrics we used to evaluate the performance of the CNN architectures we considered are summarized in this section.

3.7.1 Confusion Matrix and F_1-Score

A useful tool when reporting the results of classifiers in general is the confusion matrix. The ij^{th} entry of a confusion matrix shows the number of images that were classified as belonging to class j even though they actually belong to class i. For the case where $i = j$ (i.e. along the diagonal), the classification is in the correct class. In practice, when we depict confusion matrices we often use annotated interpretable labels instead of the aforementioned integer labels. A hypothetical confusion matrix which was obtained after classifying a dataset consisting of radio sources is depicted in Figure 3.5. This confusion matrix depicts how well our classifier could distinguish FRI sources from non-FRI sources. The class for which a classifier's performance is currently being assessed is known as the positive class (the class currently under consideration). The remaining classes are known as the negative classes. In the case of Figure 3.5 FRI is our positive class. The depiction in Figure 3.5 will of course be different if another class becomes the positive class. Furthermore, the confusion matrix in Figure 3.5 also graphically depicts the definition of the following concepts in the case of a multiclass scenario: true positives, false positives, true negatives and false negatives. The general definition of the above concepts and examples thereof (from Figure 3.5) are listed below:

- True Positives (TP): images belonging to the positive class being classified as such *(sources that were annotated as FRI and classified as such)*.
- True Negatives (TN): images from the negative classes that are not classified as belonging to the positive class (*sources that were annotated as FRII*

and correctly classified as such or sources that were annotated as Bent, but incorrectly classified as FRII).

- False Negative (FN): images belonging to the positive class not classified as such (*sources annotated as FRI, but incorrectly classified as FRII*).
- False Positives (FP): images belonging to the negative classes that are classified as belonging to the positive class (*sources annotated as FRII, but incorrectly classified as FRI*).

Recall refers to the ratio of the number of images that were correctly classified as belonging to the positive class to the total number of images in the positive class, i.e.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.31}$$

Recall is also referred to as the true positive rate. Precision is the ratio of images that were correctly classified as belonging to the positive class to the total number of images that were classified as belonging to the positive class, i.e.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.32}$$

The weighted average of recall and precision is known as the F_1-score,

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3.33}$$

A larger F_1-score is considered a preferable metric.

3.7.2 Mean per Class Accuracy

Overall accuracy is the ratio of correct classifications for all classes to the total number of samples tested on. Based on the confusion matrix in Figure 3.5, this is calculated by dividing the main diagonal (the TP of each class) by the sum of the entire matrix.

Overall accuracy can be a misleading metric, especially when a significant class imbalance is present.

For this purpose we use Mean per Class Accuracy (MPCA), calculated as the mean of the main diagonal of a normalized confusion matrix. The confusion matrix is normalized by dividing each row with the number of samples in that row (which corresponds to the number of samples per class). This metric is less susceptible to class imbalances than overall accuracy.

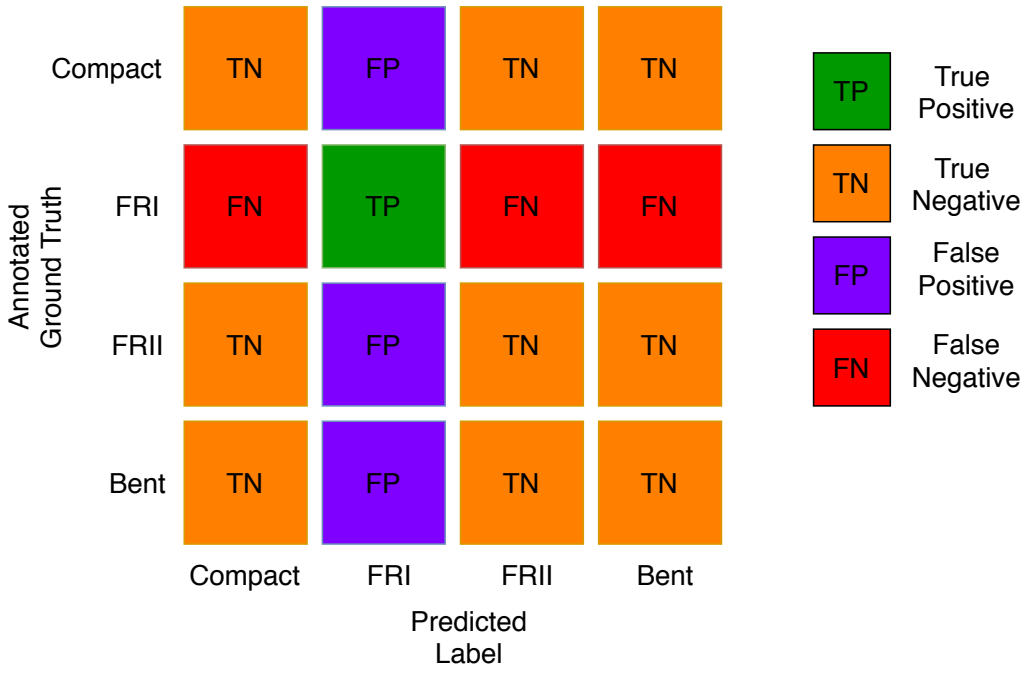

Figure 3.5: Confusion Matrix Layout. This specific example showcases an assessment of the FRI class.

3.7.3 Model Complexity

We define model complexity as the model's number of trainable parameters. The trainable parameters of a CNN are its weights and bias terms. The number of trainable parameters of all model instances associated with a particular architecture will be the same as long as they were created using the same set of hyperparameters.

3.7.4 Computational Complexity

The computational complexity of the architectures considered was measured using Tensorflow's version 1 profiler (https://www.tensorflow.org/api_docs/python/tf/compat/v1/profiler/profile). The aforementioned profiler measures the number of floating point operations (FLOPs) used by the model in a single forward pass.

3.7.5 GPU Memory Usage

Theoretical GPU memory usage was estimated by first determining the memory footprint of the CNNs parameters and then adding to that the amount of active memory the CNN would require when processing a batch of data (a batch size,

the number of samples classified at the same time by the CNN, of 32 was used in this case).

3.7.6 Inference Time and Classification Rate

Inference time is the time that a CNN requires to classify a single image. Classification rate is the number of images that are classified per second, obtained by inverting inference time. In this thesis, inference time was estimated by taking the average of 10 timed runs in which we classified 3072 images (with a batch size of 32).

3.7.7 Training to Validation Loss Ratio

We use the ratio between a model's training and validation loss as a metric to assess which architectures are more prone to overfitting comparative to other architectures. The metric is calculated by dividing the training loss with the validation loss.

When the training loss is significantly lower than the validation loss, this indicates that overfitting is present. With the training and validation loss closer to each other, we assume that less overfitting is present.

3.7.8 Training to Validation Accuracy Ratio

Similarly to the loss ratio discussed above, we use the ratio between a model's training and validation accuracy as a metric to assess which architectures are more prone to overfitting than other architectures. The metric is calculated by dividing the training accuracy with the validation accuracy.

When the training accuracy is significantly higher than the validation accuracy, this indicates that overfitting is present. With the training and validation accuracy closer to each other, we assume that less overfitting is present.

Chapter 4

Data Description and Acquisition

> *"Errors using inadequate data are much less than those using no data at all."*
>
> — Charles Babbage, *Attributed to Charles Babbage in William Kenneth Richmond's "The Education Industry", 1969.*

The data that we used for our experiments is discussed in this chapter. In Section 4.1 and 4.2 two datasets used by Ma *et al.* (2019) are introduced. The dataset we used in this study is presented in Section 4.3.

4.1 unLRG Dataset

We have modified a dataset that Ma *et al.* (2019) made use of for their study. Their original study is shortly summarized in Section 5.2.8. The authors referred to the forementioned dataset as the Unlabelled Radio Galaxy (unLRG dataset) dataset, which contains 14254 sources from the Best-Heckman sample (Best and Heckman, 2012). Although this dataset is referred to as being "unlabelled", it was manually labelled by the authors. They grouped the data into six classes; this included the compact, FRI, FRII, two bent-tail types, X-shaped and ringlike classes.

4.2 Excluded Data

Why is unLRG then called "unlabelled" when it has been manually labelled by Ma *et al.* (2019)? The moniker "unlabelled" was used to distinguish it from another mutually exclusive dataset which Ma *et al.* (2019) used. This other dataset is referred to as the Labelled Radio Galaxy (LRG dataset) dataset. This curated dataset contains well attested sources from the Combined NVSS-FIRST Galaxies (CoNFIG) catalogue (Gendre and Wall, 2008; Gendre *et al.*, 2010), the Groups catalogue (Proctor, 2011), the FRI catalogue (FRICAT) (Capetti *et al.*, 2017*a*), the FRII catalogue (FRIICAT) (Capetti *et al.*, 2017*b*), the FR0 catalogue (FR0CAT)

Source Name	Right Ascension (degrees)	Declination (degrees)	Classification
J000001.57-092940.3	0.00044	-9.49453	Compact
J000025.55-095752.8	0.00710	-9.96467	FRI
J000027.89-010235.4	0.00775	-1.04317	Compact
J000049.32-005042.9	0.01370	-0.84525	FRI
J000052.92+003044.6	0.01470	0.51239	FRII

Table 4.1: The first 5 rows of the MURG dataset, the full table is available online at https://github.com/BurgerBecker/rg-benchmarker

(Baldi *et al.*, 2018) and the X-shaped catalogue (Cheung, 2007). NVSS is the National Radio Astronomy Observatory (NRAO) VLA Sky Survey. The LRG dataset contains 1442 sources and consists of 6 classes (compact, FRI, FRII, two bent-tail types, X-shaped and ringlike). To avoid confusion, the LRG dataset has been excluded from the experiments that was conducted as part of this thesis, i.e. we exclusively work with the unLRG dataset from this point onward.

4.3 Modified unLRG (MURG) dataset

As mentioned before, our study uses a modified version of unLRG dataset. Since only MCRGNet classified X-shaped and ringlike sources of all the studies in Table 5.1 these sources were removed. A number of error-prone images were also removed from the unLRG dataset (i.e. images consisting only of `NaN` values). Three compact, two FRI and two FRII sources were also removed. Moreover, all FR0 sources were added to the compact class, since these classes were used interchangeably in some of the selected studies (Alhassan *et al.*, 2018). We grouped the two bent-tail classes together into a single bent-tail class. From this point on in the thesis the modified unLRG dataset that we made use of in our study is referred to as the MURG dataset. The MURG datasets contains 14093 sources. All images from the MURG sample are drawn from the VLA FIRST survey (Becker *et al.*, 1995).

The final catalogue that was used for our experiments is partially presented in Table 4.1. The rest of the catalogue is available on our Github repository (https://github.com/BurgerBecker/rg-benchmarker). The sources can be downloaded from our Github repository. We acquired the dataset by downloading FIRST cutouts (Becker *et al.*, 1995) in Flexible Image Transport System (FITS) format (300 by 300 pixels) via the Skyview tool (McGlynn *et al.*, 1998) (available online via https://skyview.gsfc.nasa.gov). The final class breakdown of the MURG dataset is presented in Table 4.2. An example source from each of the morphological classes we considered in this thesis can be found in Figure 4.1 to Figure 4.4.

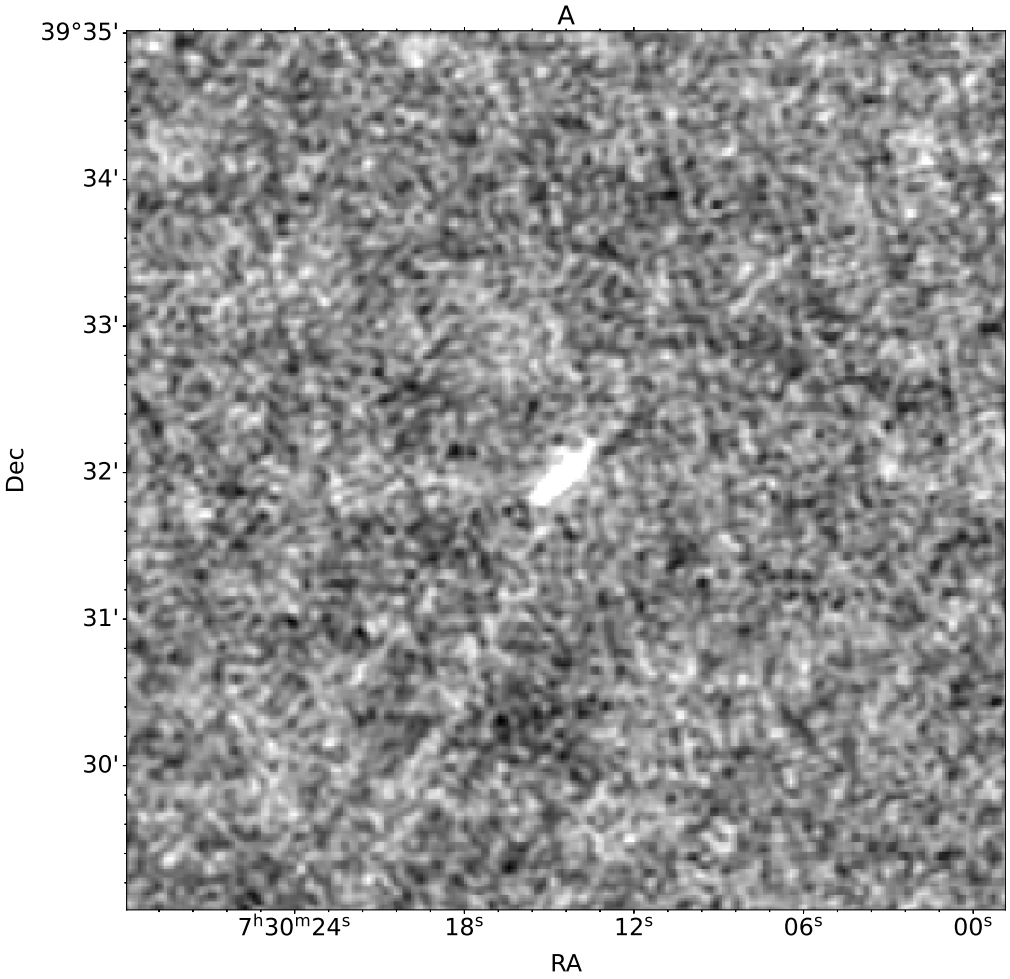

Figure 4.1: Example of the FRI (A) radio morphology from the FIRST survey (Becker *et al.*, 1995). The example shown is before the preprocessing step discussed in Section 6.1.1.

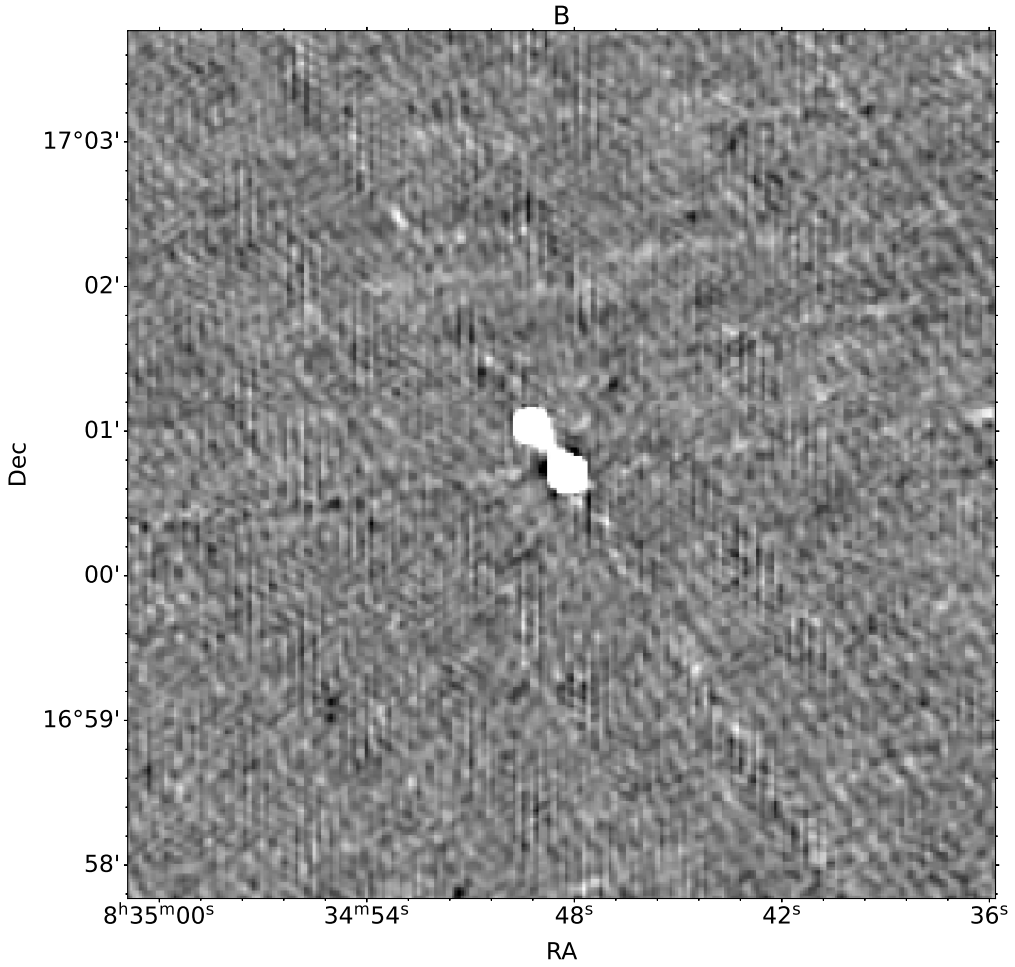

Figure 4.2: Example of the FRII (B) radio morphology from the FIRST survey (Becker *et al.*, 1995). The example shown is before the preprocessing step discussed in Section 6.1.1.

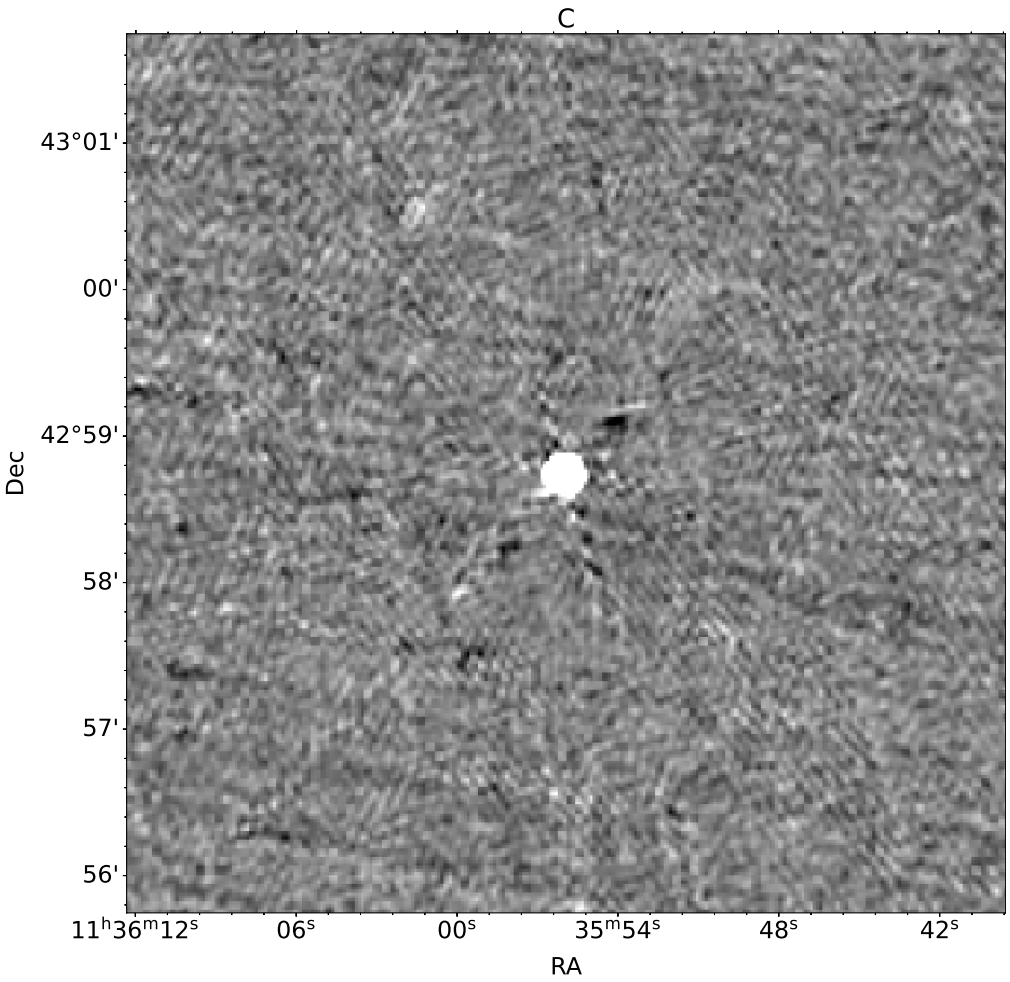

Figure 4.3: Example of the compact (C) radio morphology from the FIRST survey (Becker *et al.*, 1995). The example shown is before the preprocessing step discussed in Section 6.1.1.

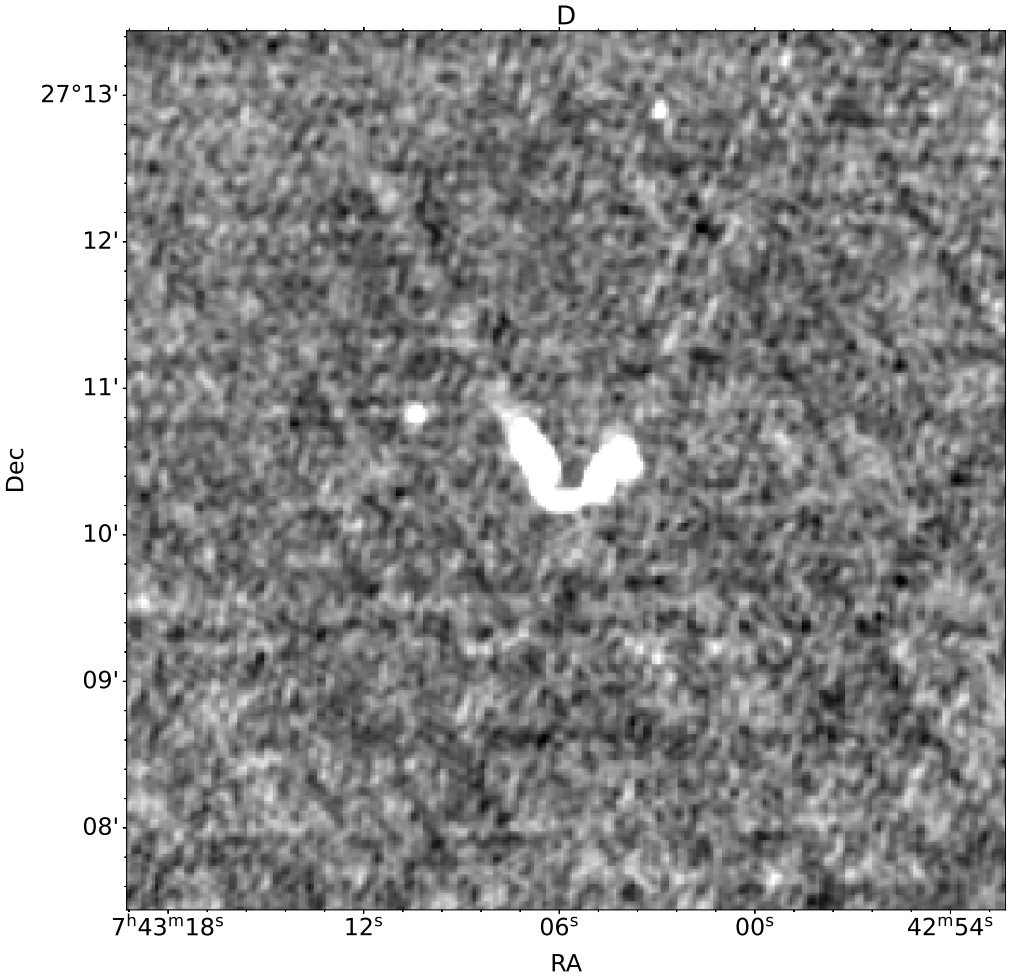

Figure 4.4: Example of the Bent tail (D) radio morphology from the FIRST survey (Becker *et al.*, 1995). The example shown is before the preprocessing step discussed in Section 6.1.1.

Class	MURG
Compact	6093
FRI	5039
FRII	2072
Bent	889
Total	14 093

Table 4.2: The MURG dataset breakdown per class.

Chapter 5

Literature Review

> *"The role of radiologists will evolve from doing perceptual things that could probably be done by a highly trained pigeon to doing far more cognitive things."*
>
> — Geofrey Hinton, *The New Yorker "A.I. Versus M.D."(Mukherjee et al., 2017)*

In this Chapter, we review the CNN architectures that are currently used to perform the radio morphological classification task. In Section 5.1 we briefly discuss the history of CNNs in astronomy and how this influenced the research done in radio astronomy. In Section 5.2 we discuss each of the selected studies in more depth as well as give a layout of their architectures (as implemented in this study). In Section 5.3 we discuss the impact of the various modifications that have been made during our study on the performance of the various architectures. In the penultimate section, we discuss our contribution in Section 5.4. This is a novel architecture and comparison study. In this thesis we have expanded upon the comparison study and added an overfitting experiment (discussed in the next Chapter). Lastly, we briefly mention the excluded architectures in Section 5.5 as well as other machine learning methods developed for radio astronomy.

5.1 CNNs and Radio Astronomy

The application of deep CNNs to radio astronomy for image classification is a relatively young research area which roughly started five years ago, and was sparked by the succesfull adoption of deep CNNs in the field of optical astronomy. Optical galaxy morphology is an ideal image classification task with clear class distinctions and simple rules to follow for classification (Lukic *et al.*, 2018).

Willett *et al.* (2013) proposed the Kaggle Galaxy Zoo competition with the purpose of training a machine learling solution that could classify a crowd sourced annotatet optical astronomy dataset. The winning solution was a CNN developed

by Dieleman *et al.* (2015). The work of Dieleman *et al.* (2015) influenced the development of similar research in radio astronomy, such as the CNN developed by Alger (2016) to determine the presence of a SMBH in a radio galaxy. The first architecture specifically designed for morphological classification of radio galaxies was developed by Aniyan and Thorat (2017). This largely shaped morphological classification in radio astronomy and provided a research framework that many other studies expanded upon. Below we expand on several studies that have had an impact on radio galaxy classification.

5.2 Selected Architectures

The terms architecture and model are not used interchangeably in the context of this study. We refer to an architecture as the layout of the network's structure, whereas a model is a specific trained instance of the architecture. Models of the same architecture are differentiated by the data it was trained on and other hyper-parameters such as different learning rates or the optimizer used during training.

The focus of this thesis is on what we refer to as "traditional" CNN architectures which is comprised of single-input, single-output layers in a stacked structure. As such, we briefly mention but do not consider capsule networks, residual networks, attention gated networks, steerable group equivariant CNNs, CNNs that use parallel convolutional blocks or architectures that use inception modules (Goodfellow *et al.*, 2016; Géron, 2019). This is beyond the scope of this thesis but would be useful approaches that could be considered as part of a future endeavour.

Table 5.1 provides the architecture names, the corresponding studies from the literature as well as shortened keys assigned for use in plots. Some studies have contributed more than one architecture (Lukic *et al.*, 2019*a*). All of the architectures listed in Table 5.1 were modified to enable them to discern between four types of radio sources (i.e. the number of output classes were changed to four). An example source from each of the morphological classes we considered in this thesis can be found in Figure 4.1 to Figure 4.4. Figure 5.1 outlines a timeline of these studies. Below we briefly discuss the contribution of each study.

5.2.1 AlexNet

The architecture of AlexNet is described in Table 5.2 (Krizhevsky *et al.*, 2012). This architecture has influenced a lot of the other architectures that are considered in this work.

Architecture Name	Key	Study
AlexNet	ALN	Krizhevsky *et al.* (2012)
ATLAS X-ID †	ATL	Alger *et al.* (2018)
ConvNet4	CN4	Lukic *et al.* (2019*a*)
ConvNet8	CN8	Lukic *et al.* (2019*a*)
FIRST Classifier	1stC	Alhassan *et al.* (2018)
FR-Deep	FR-D	Tang *et al.* (2019)
Hosenie	H	Hosenie (2018)
MCRGNet †	MCRG	Ma *et al.* (2019)
Radio Galaxy Zoo	RGZ	Lukic *et al.* (2018)
SimpleNet	CNs	Lukic *et al.* (2019*a*)
Toothless †	TLS	Aniyan and Thorat (2017)
CLARAN † (VGG16D)	VGG	Wu *et al.* (2019)
ConvXpress	CXP	Becker *et al.* (2021)

Table 5.1: List of architectures and their keys for all figures. Architectures marked † have been modified from their original form.

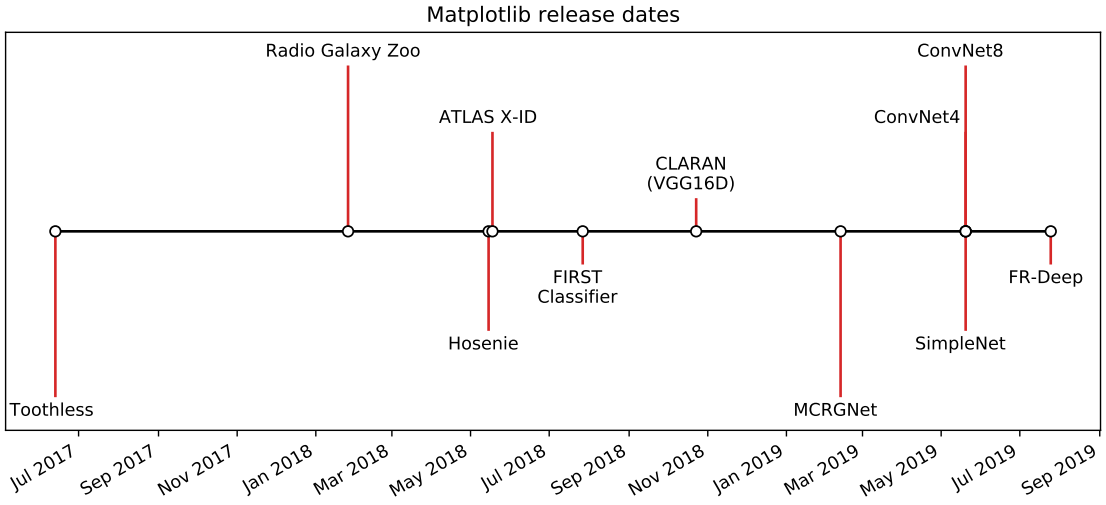

Figure 5.1: Timeline of selected publications relating to radio galaxy classification.

5.2.2 Toothless

Just as AlexNet has been a seminal work in image classification for CNNs, so too has Toothless (Aniyan and Thorat, 2017) made its mark on the classification of radio galaxies for being the first CNN developed for image classification. As mentioned before, Toothless has influenced the general direction of morphological classification in radio astronomy. It has thus been referenced in almost all of the

Layer Type	Filter size/ Dropout rate/ Neurons	Kernel Size	Stride	Pad	Activation
Convolutional	96	11x11	4	Valid	ReLU
Max Pooling	-	3x3	2	Valid	-
Batch Norm.	-	-	-	-	-
Convolutional	256	5x5	1	Valid	ReLU
Max Pooling	-	3x3	2	Valid	-
Batch Norm.	-	-	-	-	-
Convolutional	384	3x3	1	Valid	ReLU
Batch Norm.	-	-	-	-	-
Convolutional	384	3x3	1	Valid	ReLU
Batch Norm.	-	-	-	-	-
Convolutional	256	3x3	1	Valid	ReLU
Max Pooling	-	3x3	2	Valid	-
Batch Norm.	-	-	-	-	-
Flatten	-	-	-	-	-
Fully Connected	4096	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Batch Norm.	-	-	-	-	-
Fully Connected	4096	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Batch Norm.	-	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.2: AlexNet Architecture Layout

subsequent works listed in Table 5.1. Toothless is based on AlexNet's architecture and does not differ much other than the type of padding used and the addition of batch normalization. The original AlexNet design using valid padding (no padding is applied around the edges of the input of a layer) rather than same padding (zero padding around the edges of the input of a layer to ensure there is no size reduction other than that caused by stride length). Valid padding reduces the input size steadily after each convolutional layer, while same padding keeps the original input size to the layer. This means that the majority of downsampling (reduction in input size) in Toothless is a result of max pooling.

Aniyan and Thorat (2017) originally implemented Toothless as three binary classifiers each one either classifying an image as being either FRI/FRII, FRI/Bent

and FRII/Bent. If two classifiers would predict a source as the same class with a 60% probability, the classification would be accepted, if both predicted with less than the 60% confidence threshold, a '?' would be appended to the classification. Additionally, should none of the classifiers give the same class output, the source is labelled as "Strange". To reduce computational requirements, only a single classifier instance of Toothless (not three) is considered in our study. The architecture used to represent Toothless in this study can be found in Table 5.3.

Toothless was trained on 177 Bent-tails, 125 FRIs and 227 FRIIs (a data set similar in nature to the one reported in Section 4.3). Traditional CNNs require large amounts of data to train. To overcome this obstacle, Aniyan and Thorat (2017) augmented each class by employing one-degree augmentations of the original dataset. Aniyan and Thorat (2017) do raise concerns that this may affect generalization performance and may reduce the feature space that the CNN can choose from, leading to a model which fits to the noise present in the data. To address this, batch normalization was used between the convolutional layers and dropout was added between the dense layers. A dropout rate of 50% was used.

The models were also very sensitive to the preprocessing procedure which was used. They employed sigma clipping, which entails setting pixles that reside three sigma above or below the mean to zero (see Chapter 6.1.2 for more information).

Another concern was the class imbalance, which was addressed by creating more augmented samples for the underrepresented FRI class. Testing was done on 187 samples (77 Bent-tails, 53 FRI, 57 FRII). The worst precision was for FRII, which the authors argue is due to a subset-population of the Bent-tail class that are exhibiting FRII-like morphology and as such they are misclassified.

5.2.3 Radio Galaxy Zoo Classifier

The issues of overfitting faced by Aniyan and Thorat (2017) due to the relatively few representative samples of each class prior to augmentation drove Lukic *et al.* (2018) to consider a different data collection approach. Instead of using the small datasets which were prepared by domain experts in catalogues, Lukic *et al.* (2018) developed a different four class system based on the number of components each source has (compact, single-component extended, two-component extended and multi-component extended). These labels are automatically generated by the Python Blob Detector and Source Finder (PyBDSF) if it is applied to an interferometric image. PyBDSF was developed by Mohan and Rafferty (2015) and is an automated source finding tool that can aid the labeling of radio sources by fitting Gaussian mixture models to the data. The use of PyBDSF allowed Lukic *et al.* (2018) to construct much larger datasets. The new classification scheme used by Lukic *et al.* (2018) is not interchangeable with the classification system used by Aniyan and Thorat (2017). This approach made it possible to construct

Layer Type	Filter size/ Dropout rate/ Neurons	Kernel Size	Stride	Pad	Activation
Convolutional	96	11x11	4	Valid	ReLU
Batch Norm.	-	-	-	-	-
Max Pooling	-	3x3	2	Valid	-
Convolutional	256	5x5	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Max Pooling	-	3x3	2	Valid	-
Convolutional	384	3x3	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Convolutional	384	3x3	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Convolutional	256	3x3	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Max Pooling	-	3x3	2	Valid	-
Flatten	-	-	-	-	-
Fully Connected	4096	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4096	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.3: Toothless Architecture Layout

a training set which contained twenty one thousand samples. The quality of the labelling was, however, questionable. Another dataset was also used, the first data release from the Radio Galaxy Zoo citizen science project was used, providing almost fifteen thousand training samples labelled by human volunteers (also as either compact, single-component extended, two-component extended and multi-component extended). Roughly seven thousand samples were used for testing.

Lukic *et al.* (2018) developed six classifiers, removing those that overfit and comparing the rest based on classification accuracy. The E variant is proposed by the authors as the best architecture and can be found in Table 5.4. Dropout was used between the fully connected layers with a dropout rate of 50%.

Lukic *et al.* (2018) was the first paper from the Radio Galaxy Zoo research group which explicitly deals with radio galaxy classification task. Their study was done with two experiments. The results of the first experiment was then used to

fine tune the hyperparameters of their models and to determine the models that were overfitting. The models that were found to overfit were excluded from the second experiment. The fine tuned hyperparameters from from the first experiment were then re-used in the second experiment.

Compared to Toothless, the reliance on preprocessing was reduced significantly. Recall, Toothless required pre-processing in order for it to classify with an acceptable classification accuracy (Aniyan and Thorat, 2017). Lukic *et al.* (2018) showed that training and validation loss were quite close for their models and that overfitting does not seem to be present. The authors address the presence of other sources (a second or third etc. source surrounding the primary target source) in 44% of the multi-component extended samples, which they attempted to remove manually but many such superimposed images remained. Radio Galaxy Zoo Classifier was not modified in any further way for our comparison study. The architecture is given in Table 5.4.

Layer Type	**Filter size/ Dropout rate/ Neurons**	**Kernel Size**	**Stride**	**Pad**	**Activation**
Convolutional	16	8x8	3	Same	ReLU
Convolutional	32	7x7	2	Same	ReLU
Max Pooling	-	3x3	3	Valid	-
Convolutional	64	2x2	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Flatten	-	-	-	-	-
Fully Connected	1024	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	1024	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.4: Radio Galaxy Zoo Classifier Architecture Layout

5.2.4 Hosenie

This architecture was originally a binary classifier (for the FRI and FRII classes) but has been modified for the four class scheme we adopt in this thesis. Hosenie (2018) developed a much smaller network (29 thousand parameters) compared

to Aniyan and Thorat (2017) (87 million parameters) to address the overfitting concerns they brought up. The architecture proposed is represented in Table 5.5. A more stringent selection criteria was also applied to the dataset used by Aniyan and Thorat (2017) to remove noisy samples. Additionally, dropout was applied as regularization techniques between all layers, with a dropout rate of 50% in the dense layers and 25% in the convolutional layers.

Training was done on 48 FRIs and 185 FRIIs sources with rotational augmentation applied. Testing was done on 116 samples.

Layer Type	**Filter size/ Dropout rate/ Neurons**	**Kernel Size**	**Stride**	**Pad**	**Activation**
Convolutional	6	11x11	1	Same	ReLU
Max Pooling	-	3x3	3	Valid	-
Dropout	0.25	-	-	-	-
Convolutional	19	5x5	1	Same	ReLU
Max Pooling	-	3x3	3	Valid	-
Dropout	0.25	-	-	-	-
Convolutional	38	3x3	1	Same	ReLU
Convolutional	26	3x3	1	Same	ReLU
Convolutional	26	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Dropout	0.25	-	-	-	-
Flatten	-	-	-	-	-
Fully Connected	40	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	40	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.5: Hosenie Architecture Layout

5.2.5 ATLAS

ATLAS was not designed explicitly for radio galaxy classification, but rather to find the host galaxies of radio sources in the infrared spectrum. Alger *et al.* (2018) did a comparative study between a CNN, a logistic regression classifier and a random

forest classifier for this task. The CNN and the logistic regression classifier were less sensitive to changes in brightness when tested on fainter components from another survey than the training set.

The CNN described in Alger *et al.* (2018) had an additional input vector of 10 features from the candidate host in the Spitzer Wide-area Infrared Extragalactic (SWIRE) survey, which has not been included in the modified architecture we used in this study. The original CNN model was a binary classifier. The output layer's activation function was, therefore, changed from sigmoid to softmax. The hidden dense layer also made use of a sigmoid activation, which was changed to a ReLU activation.

Similarly to Lukic *et al.* (2018), experiments were done with both domain expert labelled and crowd sourced labelled data. The authors comment that the models that were trained with the crowd sourced data performed comparable to the models that were trained using the data that were labelled by domain experts. The domain expert labelled data contained 468 samples while the crowd sourced data contained 2460 samples. The authors also comment that larger datasets would be helpful to help rectify the large class imbalance which was present in their data. The architecture of ATLAS is presented in Table 5.6. ATLAS has dropout layers between all the convolutional and fully connected layers with a dropout rate of 25% and 50%, respectively.

Layer Type	**Filter size/ Dropout rate/ Neurons**	**Kernel Size**	**Stride**	**Pad**	**Activation**
Convolutional	32	10x10	1	Same	ReLU
Max Pooling	-	5x5	5	Same	-
Dropout	0.25	-	-	-	-
Convolutional	32	10x10	1	Same	ReLU
Max Pooling	-	5x5	5	Same	-
Dropout	0.25	-	-	-	-
Flatten	-	-	-	-	-
Fully Connected	64	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.6: ATLAS Architecture Layout

5.2.6 FIRST Classifier

Alhassan *et al.* (2018) introduces the four class scheme of Compact, Bent-tail, FRI and FRII for automatic classification (the same scheme we use in this study). Previous studies up to this point (with the exception of Lukic *et al.* (2018) which did not use the Fanarof-Riley classification scheme) excluded Compact sources because of their relatively simple morphology. Numerous methods that can identify compact sources already exist (Mohan and Rafferty, 2015). The FIRST Classifier was developed by utilizing experiments that varied the number of convolutional layers (between 2 to 10) and the different types of activation functions. The best performing model based on precision, recall and F1-score was selected at the end. No modifications were made to this architecture.

The architecture consists of three convolutional layers with 3x3 kernels and two fully connected layers. To combat overfitting the authors added a dropout layer was added after the first fully connected layer (a dropout rate of 50% was used).

Training was done on 530 samples and testing on 154 samples. Validation and training loss was equal at the end of training, although it is unknown whether the loss was rectified for known effects that make validation loss appear lower. For example, since validation loss is calculated at the end of each epoch and training loss is calculated at the end of each batch, the training loss calculated at the end of each epoch might be lower than the value reported in the article. The architecture of FIRST Class is given in Table 5.7.

Layer Type	**Filter size/ Dropout rate/ Neurons**	**Kernel Size**	**Stride**	**Pad**	**Activation**
Convolutional	32	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Convolutional	64	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Convolutional	194	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Flatten	-	-	-	-	-
Fully Connected	194	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.7: FIRST Classifier Architecture Layout

5.2.7 CLARAN (VGG16D)

Wu *et al.* (2019) differs from all the studies selected for this thesis in that it does not perform a pure image classification task but performs object detection instead. This means that it performs object localization in addition to image classification and as such is less susceptible to the issues that arise when an image contains more than one source. The shift of task to object detection rather than image classification is in part a response to address the concerns of Lukic *et al.* (2018) who pointed out that a significant number of images in their dataset contained multiple sources. Wu *et al.* (2019) used a Faster Region Based Convolutional Neural Networks (R-CNN) model as the basis of this architecture (Ren *et al.*, 2015) with a VGG16D backbone for image classification (Simonyan and Zisserman, 2015) which was pretrained on ImageNet. The architecture of VGG16D is described in Table 5.8. A morphological classification system different from the Fanarof-Riley scheme is used in this study as well, based on the number of components C and peaks P; which results in a six class classification scheme (1C-1P, 1C-2P, 1C-3P, 2C-2P, 2C-3P and 3C-3P). This was done in response to the concerns of the small feature space in the data sample used by Aniyan and Thorat (2017) and to build on the classification system used in Lukic *et al.* (2018).

In their study, training was done on 6141 samples and testing on 4603 samples. No rotational augmentation was used. To assess whether the model was overfitting an additional experiment was performed which entailed training model variants that utilized less parameters. This was done by decreasing the dimensions of the two dense layers from 4096 to 256 for the one experiment and to 64 for the other. This, respectively, leads to an 83% and 86% decrease in model parameters. These models score poorer on test accuracy than the original. Wu *et al.* (2019) concludes from this result that the CLARAN model is not overfitting, since test error is still decreasing with an increase in capacity (Goodfellow *et al.*, 2016).

The authors attribute the relatively better performance of the original network architecture to several factors that help prevent overfitting: the two dropout layers in the dense layers (with a rate of 50%), "augmentation" resulting from small areas selected in each image by the region of interest proposals that increase training samples from the thousands to the millions and lastly transfer learning, where the weights of a pretrained network is used rather than initialising weights and training from scratch. Since our data does not have bounding box labels and our study focusses on comparing traditional CNNs, we only train the VGG16D model.

5.2.8 MCRGNet

Morphological Classification of Radio Galaxy Network (MCRGNet) was developed with the aim of making use of large amounts of unlabelled radio data to pre-train

Layer Type	**Filter size/ Dropout rate/ Neurons**	**Kernel Size**	**Stride**	**Pad**	**Activation**
Convolutional	64	3x3	1	Same	ReLU
Convolutional	64	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Convolutional	128	3x3	1	Same	ReLU
Convolutional	128	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Convolutional	256	3x3	1	Same	ReLU
Convolutional	256	3x3	1	Same	ReLU
Convolutional	256	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Convolutional	512	3x3	1	Same	ReLU
Convolutional	512	3x3	1	Same	ReLU
Convolutional	512	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Convolutional	512	3x3	1	Same	ReLU
Convolutional	512	3x3	1	Same	ReLU
Convolutional	512	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Valid	-
Flatten	-	-	-	-	-
Fully Connected	4096	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4096	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.8: VGG16D Architecture Layout

a CNN. To do this a convolutional autoencoder (CAE) was trained on the 14254 unlabelled images (described in Chapter 4 as the unLRG dataset). An autoencoder is a type of ANN used to learn efficient codings of unlabeled data. After the training step the convolutional layer's weights were used in a CNN model that was then fine-tuned on a smaller well curated dataset of 1442 labelled sources (described in Chapter 4 as the LRG dataset). The architecture originally classified images into six classes; Compact, FRI, FRII, two Bent types, X-shaped and Ringlike sources.

In our study, the dropout rate of the dropout layers between the convolutional layers have been reduced from the original 50% to 25%. This was adapted due to both poor initial results and to be more consistent with the dropout rate of the dropout layers between the convolutional layers of the other architectures. The architecture has a dropout layer between all the convolutional layers and all the fully connected layers. The architecture of MCRGNet is given in Table 5.9.

Layer Type	**Filter size/ Dropout rate/ Neurons**	**Kernel Size**	**Stride**	**Pad**	**Activation**
Convolutional	8	3x3	2	Same	ReLU
Dropout	0.25	-	-	-	-
Convolutional	8	3x3	2	Same	ReLU
Dropout	0.25	-	-	-	-
Convolutional	16	3x3	2	Same	ReLU
Dropout	0.25	-	-	-	-
Convolutional	16	3x3	2	Same	ReLU
Dropout	0.25	-	-	-	-
Convolutional	32	3x3	2	Same	ReLU
Dropout	0.25	-	-	-	-
Flatten	-	-	-	-	-
Fully Connected	64	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.9: MCRGNet Architecture Layout

5.2.9 FR-Deep

The original focus of the study by Tang *et al.* (2019) was to assess the viability of transfer learning in radio astronomy specifically across surveys with different resolutions. For this a CNN with five convolutional layers and four fully connected layers was developed. This CNN was then pretrained on lower resolution surveys and then fine-tuned on a higher resolution survey. The FIRST survey was one of the datasets used to pretrain the CNN with fine-tuning and testing being done on data from MeerKAT. The authors found that pre-training with lower resolution data does not have any real benefit, if the new data has a much better resolution.

No modifications was made to this classifier's architecture, however we do not apply any pre-training. The architecture has batch normalization between each convolutional layers and dropout layers with a 50% dropout rate between the fully connected layers. The modified architecture is presented in Table 5.10.

5.2.10 SimpleNet, ConvNet4 and ConvNet8

The authors compared the performance of three traditional CNNs with that of a capsule network. The classifiers originally classified three classes: unresolved sources, FRI and FRII. Lukic *et al.* (2019*a*) found that the traditional CNNs outperformed the capsule networks.

The three traditional CNNs are referred to in this thesis as SimpleNet, ConvNet4 and ConvNet8. SimpleNet consists of five convolutional layers followed immediately by the output layer. This was done to assess the necessity of an intermediate dense layer. ConvNet4 has four convolutional layers and two fully connected layers. ConvNet4 used a larger kernel size (5x5) throughout its convolutional layers, rather than a reduction after initially utilizing large kernels (like in AlexNet) to reduce the number of convolutional parameters that are required. ConvNet8 has eight convolutional layers and two fully connected layers. The convolutional layers are structured as four pairs each followed by a max pooling layer. The convolutional layers also use a smaller kernel size of 3×3.

SimpleNet was found to perform significantly worse than ConvNet4 and ConvNet8. This result prompted the authors to postulate that and intermediate dense layer is only useful if more complex patterns are present in the data that comes from the convolutional layers (which is not the case for the radio galaxy morphology classification task).

No modifications have been made to any of the architectures, other than the number of output classes which were changed to four. The dropout rate of the dropout layers between the convolutional layers have been set to 25% and to 50% between the fully connected layers. L2 regularization is added to the first fully connected layers of ConvNet4 and ConvNet8.

Layer Type	Filter size/ Dropout rate/ Neurons	Kernel Size	Stride	Pad	Activation
Convolutional	6	11x11	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Max Pooling	-	2x2	2	Valid	-
Convolutional	16	5x5	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Max Pooling	-	3x3	3	Valid	-
Convolutional	24	3x3	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Convolutional	24	3x3	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Convolutional	16	3x3	1	Same	ReLU
Batch Norm.	-	-	-	-	-
Max Pooling	-	5x5	5	Valid	-
Flatten	-	-	-	-	-
Fully Connected	256	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	256	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	256	-	-	-	ReLU
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.10: FR-Deep Architecture Layout

The modified architectures is presented in Tables 5.11, 5.12 and 5.13.

5.3 Impact of Modifications

At this point in time we should take a moment to consider the potential impact that the modifications we discuss in Section 5.2 will have on the performance of the architectures presented in the studies from Table 5.1. It should be noted that the modifications proposed by us are a necessity as these modifications make it possible to perform a meaningful comparison of these architectures. Three major modifications were discussed:

Layer Type	Filter size/ Dropout rate/ Neurons	Kernel Size	Stride	Pad	Activation
Convolutional	16	4x4	1	Same	ReLU
Convolutional	16	4x4	1	Same	ReLU
Convolutional	16	4x4	1	Same	ReLU
Max Pooling	-	4x4	4	Same	-
Dropout	0.25	-	-	Same	-
Convolutional	16	4x4	1	Same	ReLU
Convolutional	16	4x4	1	Same	ReLU
Max Pooling	-	4x4	4	Same	-
Dropout	0.25	-	-	-	-
Flatten	-	-	-	-	
Fully Connected	4	-	-	-	Softmax

Table 5.11: SimpleNet Architecture Layout

Layer Type	Filter size/ Dropout rate/ Neurons	Kernel Size	Stride	Pad	Activation
Convolutional	16	5x5	1	Same	ReLU
Convolutional	16	5x5	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Convolutional	16	5x5	1	Same	ReLU
Convolutional	16	5x5	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Fully Connected	500	-	-	-	Linear
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.12: ConvNet4 Architecture Layout

Layer Type	Filter size/ Dropout rate/ Neurons	Kernel Size	Stride	Pad	Activation
Convolutional	32	3x3	1	Same	ReLU
Convolutional	32	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Convolutional	64	3x3	1	Same	ReLU
Convolutional	64	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Convolutional	128	3x3	1	Same	ReLU
Convolutional	128	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Convolutional	256	3x3	1	Same	ReLU
Convolutional	256	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Flatten	-	-	-	-	-
Fully Connected	500	-	-	-	Linear
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.13: ConvNet8 Architecture Layout

Output Classes The number of the output classes and in some cases even the output-labels of the output classes were altered (Toothless as an example of the former, CLARAN as an example of the latter). This alteration, however, is standard practise within the field of Deep Learning. Take AlexNet as an example it was originally designed for the ImageNet Challenge, but it is nowadays used to solve many other types of image recognition problems (i.e. the number of classes and the output labels it can produce differ from its original use case). Generally speaking, if a CNN architecture is identified that can discern between N different classes, then its recognition performance will normally not deteriorate significantly if the number of classes that one considers is either reduced or increased by one (given that it is properly re-trained). Moreover, neither would considering N completely different labels

have a significant impact on its performance. There are of course exceptions to this, if the nature of the problem is changed completely or the inherent separability of the dataset changes significantly this generalization might not necessarily remain true.

Architecture Instances Only single architecture instances were considered (as an example only a single architecture instance of Toothless (Aniyan and Thorat, 2017) was used). Multiple instances of any architecture can be incorporated into a more complex classifier (like Toothless). This will certainly improve the recognition performance of a particular architecture. However, knowing how a single instance of the architecture performs enables us to identify which architectures will ultimately perform better if they are chosen to create a more complex classifier.

Data The same dataset was used to evaluate each architecture.

5.4 Contributions

The unique contributions made by our study is presented in this section.

5.4.1 ConvXpress

The novel architecture of ConvXpress is based on the architecture of MCRGNet, ConvNet8 and VGG16D. This architecture was introduced formally in our article "CNN Architecture Comparison for Radio Galaxy Classification" (Becker *et al.*, 2021). ConvXpress is deeper than ConvNet8 (11 vs 8 convolutional layers) and uses the convolutional stack structure introduced by VGG16D. Each stack is comprised of three convolutional layers (except for the last stack, which consists of only two layers) and a max pooling layer. This was developed to match or enlarge the receptive field size (see Section 3.4.3.1) of AlexNet's convolutional layers without having to use AlexNet's large kernel size. The effective receptive field of ConvXpress's first convolutional stack is 11×11, the same size as AlexNet's first convolutional layer. However, the stack structure has applied three non-linear rectification functions compared to AlexNet's single activation, making the model more discriminative (Simonyan and Zisserman, 2015).

In addition to this, the number of parameters required are reduced by stacking: a layer with an 11×11 kernel with C input channels require $11^2C^2 = 121C^2$ parameters, while 3 layers with a 3×3 kernel and C input channels require only $3(3^2C^2) = 27C^2$ parameters.

ConvXpress has a non-standard stride length, similar to MCRGNet, Toothless, AlexNet and Radio Galaxy Zoo. In particular, it makes use of a stride length of 2 in

the first convolutional layer of the first and second convolutional stacks. The dense (or fully-connected) layers are the same as ConvNet8's, using a linear activation in the second last layer with an L2 kernel regularizer. ConvXpress also contains five dropout layers. The dropout rate for the dropout layers in ConvXpress is 25% between the convolutional layers. The only exception is the dropout layer between the fully connected layers which uses a dropout rate of 50%. The architecture of ConvXpress is presented in Table 5.14.

Layer Type	**Filter size/ Dropout rate/ Neurons**	**Kernel Size**	**Stride**	**Pad**	**Activation**
Convolutional	32	3x3	2	Same	ReLU
Convolutional	32	3x3	1	Same	ReLU
Convolutional	32	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Convolutional	64	3x3	2	Same	ReLU
Convolutional	64	3x3	1	Same	ReLU
Convolutional	64	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Convolutional	128	3x3	1	Same	ReLU
Convolutional	128	3x3	1	Same	ReLU
Convolutional	128	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Convolutional	256	3x3	1	Same	ReLU
Convolutional	256	3x3	1	Same	ReLU
Max Pooling	-	2x2	2	Same	-
Dropout	0.25	-	-	-	-
Flatten	-	-	-	-	-
Fully Connected	500	-	-	-	Linear
Dropout	0.5	-	-	-	-
Fully Connected	4	-	-	-	Softmax

Table 5.14: ConvXpress Architecture Layout

5.4.2 Comparison Study

One of the central contributions to the literature that we have made is discussed in detail in Becker *et al.* (2021). In this work we present a comparison framework to allow easier comparison of classifiers with each other. While some other studies compare their CNNs with previous CNNs (mostly with Toothless), they do not retrain any of these CNNs on the dataset used in their study. Only Lukic *et al.* (2019*a*) trained several novel CNN architectures they had developed as well as a Capsule Network, but did not retrain any previous architectures for comparison. Additionally, the results from Lukic *et al.* (2019*a*) is based off of a single run.

Our contribution in Becker *et al.* (2021) involved retraining all of the architectures mentioned in Section 5.2 on the same dataset with three iterations, each iteration trained on a randomized subset of the larger dataset. We expand on that work in this thesis, increasing the number of iterations to ten, while simultaneously place a larger emphasis on identifying and reducing overfitting. Another study that further expands on the research we've done is the work done by Fielding *et al.* (2021) in which a comparison study is presented which focusses on the classification of optical galaxies instead of radio galaxies.

5.4.3 Overfitting and Regularization

The regularization methods used by the architectures are listed in Table 5.15. All studies used dropout between the hidden layers of their fully connected/dense layers. SimpleNet (CNS) has only an output layer, hence why no dropout layer is present. We can broadly categorize the regularization strategies used by the various studies as either dropout between the convolutional layers (convolutional dropout), L2 kernel regularization or batch normalization.

Batch normalization showed inconsistent results in terms of regularization for preliminary experiments, which have been excluded from our study. Batch normalization is not widely used in the selected studies. As such, we've excluded batch normalization in our consideration of regularization effects.

The architectures that have implemented L2 kernel regularization have also implemented convolutional dropout. In our previous study (Becker *et al.*, 2021) it was found that ConvNet8 and ConvXpress were among the best performing architectures. As such, we perform several tests to determine which of these two methods (or the combination of them) are the most effective regularization intervention.

5.5 Excluded Architectures

Some architectures were excluded from the study for either being originally designed to perform a task other than classification or in order to restrict the scope

Key	Dropout (Conv)	Dropout (Dense)	Total Dropout	Batch Norm	Kernel Regulizer	Dropout Rate (Conv)	Dropout Rate (Dense)
CXP	4	1	5	0	L2(0.01)	0.25	0.5
CN4	2	1	3	0	L2(0.01)	0.25	0.5
CNS	2	0	2	0	0	0.25	0
MCRG	5	1	6	0	0	0.25	0.25
H	3	2	5	0	0	0.25	0.5
ATL	2	1	3	0	0	0.25	0.5
FR-D	0	3	3	5	0	0	0.5
1stC	0	1	1	0	0	0	0.5
RGZ	0	2	2	0	0	0	0.5
CN8	4	1	5	0	L2(0.01)	0.25	0.5
VGG	0	2	2	0	0	0	0.5
TLS	0	2	2	5	0	0	0.5
ALN	0	2	2	2†	0	0	0.4

Table 5.15: Regularization Methods used in each classifier. The columns with dropout in the title indicate the number of dropout layers each architecture has. This has further been subdivided between convolutional and dense layers, a total is also given. † AlexNet has batch normalization between it's dense layers, while the other classifiers have it between the convolutional layers.

of the study to traditional CNN architectures. The following architectures were excluded:

- Convosource: Designed for source-finding, to extract the pixels that belong to an astronomical source from an image with background noise (Lukic *et al.*, 2019*b*).
- COSMODEEP: Designed to perform a combination of source finding and classification by breaking up larger images into smaller tiles that are then individually classified as either containing no signal or containing a radio source (Gheller *et al.*, 2018).
- DEEPSource: Aimed at source-finding in low signal-to-noise ratio cases (Sadr *et al.*, 2019).
- Capsule Networks: This study limits the focus of comparison to traditional CNN architectures described in the literature. Lukic *et al.* (2019*b*) compared Capsule Network performance with traditional CNN architectures.

- Attention gated CNNs: Bowles *et al.* (2020) used attention gated CNNs for radio galaxy classification and was inspired by the succesfull application thereof to the classification of sonograms task (Schlemper *et al.*, 2018). The architecture Schlemper *et al.* (2018) developed uses the convolutional layers of VGG16D but replaces the fully connected layers with an attention gated mechanism, thereby reducing the number of parameters required by almost 90% (696K parameters required vs. VGG16D's 138m parameters). Bowles *et al.* (2020) reports results comparable to Tang *et al.* (2019).

- Steerable group equivariant CNNs: Scaife and Porter (2021) introduced steerable group equivariant CNNs to radio astronomy, which was developed by Weiler and Cesa (2019). Scaife and Porter (2021) used steerable group equivariant CNNs with convolutional layers that are rotation and reflection equivariant. These steerable group equivariant CNNs requires less parameters and according to Scaife and Porter (2021) they perform similarly to traditional CNN designs.

- Mask R-CNN: Arslan (2020) trained a Mask R-CNN for radio galaxy localization and classification, with the Mask R-CNN architecture originally being developed by He *et al.* (2017). Arslan (2020) reports worse results than CLARAN in all but the compact class.

- Maslej-Krešňáková *et al.* (2021) developed a CNN with three parallel convolutional blocks that feeds into a fully connected layer and performed a comparative study on augmentation techniques. The study found that rotations, reflections and increasing the brightness within a specific pixel range had positive effects on recognition performance while translation and zooming/cropping the image had detrimental effects.

- Samudre *et al.* (2021) used a pre-trained DenseNet model with a cyclical learning rate and discriminative learning to increase training speed.

- Polsterer *et al.* (2016) used self organizing Kohonen maps to construct prototype classes of radio galaxy morphologies.

- Ma *et al.* (2018) developed a deep neural network autoencoder and Gaussian mixture models to generate FRI and FRII samples.

- Ralph *et al.* (2019) developed a self organizing map in conjunction with a convolutional autoencoder for the unsupervised clustering of radio galaxies.

- Ntwaetsile and Geach (2021) used Haralick features to automatically sort and cluster radio galaxies based on their morphology.

- Giant Radio Galaxy Classification using Multi-Domain Deep Learning. This work by Tang *et al.* (2021) uses multi-domain multi-branch CNNs to identify giant radio galaxies.

Chapter 6

Experiment Description

> *"As the true method of knowledge is experiment, the true faculty of knowing must be the faculty which experiences."*
>
> — William Blake, *All Religions Are One, "The Argument", 1788.*

The experiments we conducted are described in this chapter. In Chapter 5.1, we list and discuss all the architectures we considered in our study. The experimental setup is presented in Section 6.1, with image preprocessing discussed in Section 6.1.1 and other preliminaries such as hyperparameter selection discussed in Section 6.1.2. The experiments conducted are presented in Sections 6.2 to 6.4. Experiment 1 (assessment of overfitting) is presented in Section 6.2. Experiment 2 (assesment of regularization intervention) is discussed in Section 6.3. Experiment 3 (ranking) is discussed in Section 6.4.

6.1 Experimental Setup

This section describes the experimental setup that we have used. The image prepossessing procedure that we have adopted is described in Section 6.1.1. The hardware used as well as other important overarching experimental information is presented in Section 6.1.2.

The experiments conducted are described in Section 6.2 through Section 6.4.

6.1.1 Image Preprocessing

The preprocessing steps used are that the images were first normalized and then thresholding was applied. The images were normalized by taking the minimum pixel value of each image, subtracting this from every other pixel value and then dividing the result by the difference between the maximum and minimum pixel value of the same image. The thresholding method used assigns a zero value to all pixels with a value below the threshold of three standard deviations above the mean pixel value of the specific image. Otherwise, the pixel value is kept the same.

6.1.2 Preliminaries

All training was performed on a Nvidia Tesla V100 32 GB. Our architectures were constructed using the deep learning framework, Keras (Chollet *et al.*, 2015). To ensure replicable results we provided a random seed to all non-deterministic processes. A customized version of the Keras data generator class was used to load images during training, validation and testing.

Each architecture was trained for 20 epochs with a learning rate selected for each architecture based on preliminary runs. Adam was used as the optimizer with a learning rate scheduler. Loosely speaking, a learning rate scheduler is a function called during training which reduces the learning rate if the loss has not decreased by a given threshold for x epochs. For the experiments in this thesis, the learning rate is reduced by half after three epochs in which the validation loss has not decreased by a threshold of 0.01. After training is done, we save the model resulting from 20 epochs of training which we refer to as the "final" model (these models are used in Experiments 1 to 3 in Sections 6.2 to 6.4).

Furthermore, the experiments below were repeated ten times (different seed values for the random processes and the subset selection were assigned during each of the runs). Using different seed values results in a different weight initialization for the CNNs and a different subset selection for the training, validation and testing sets during each run. This is done to get a more accurate representation of an architecture's performance. All the dropout rates of dropout layers between the convolutional layers were set to 25% and to 50% between the fully connected layers (see Section 5.2 for architecture specific implementation).

6.1.3 Training and Validation

The training and validation data are sampled from the MURG dataset. Training is performed on a random selection of 250 sources per class (6000 after augmentation) and validation on 100 sources per class (2400 after augmentation). The training set is augmented by rotating each source at 15 degree intervals, leading to 24 rotated samples of each image. This results in 24,000 and 9,600 sources for training and validation respectively after augmentation (as shown in parentheses in Table 6.1). Again, exactly which sources are selected is determined by the random seed that was used during each experimental run. While this is a much larger training and validation split (in absolute terms) than what was used by the selected studies trained on datasets labelled using the Fanaroff-Riley scheme (see Chapter 5), it is significantly smaller (in relative terms) than what is normally used when training most conventional CNNs. The selected studies that were trained on datasets labelled using the Fanaroff-Riley scheme trained on up to 60% of the

Class	Training Set (Augmented)	Validation Set (Augmented)	Test Set
Compact	250 (6000)	100 (2400)	5743
FRI	250 (6000)	100 (2400)	4689
FRII	250 (6000)	100 (2400)	1722
BENT	250 (6000)	100 (2400)	539
Total	1000 (24 000)	400 (9600)	12 693

Table 6.1: MURG Random Split Experiment: Training, validation and test set break down per class. The test set is a subset of the MURG samples. The values in parentheses are the number of augmented samples.

available data and tested on 20%, which translates to less than 40 unique samples in the test set on average.

As we have already alluded to, we have taken a smaller percentage of the training set size than the selected studies used, but the absolute number of samples is in line with what the selected studies used. Hence, this smaller selection was chosen to assess the efficacy of model performance on a larger dataset when training on a relatively small subset of the data, since this is how models, that will perform radio galaxy morphological classification in practice, are to be trained (at least at first). It should also be noted that there is a large class imbalance present in the dataset. Moreover, there is a second reason we considered the aforementioned data split: This data spilt ensures that each class is equally represented in the training set (i.e it allows us to perform undersampling). Testing was performed on a MURG test split. The total number of samples in this test set is given in Table 6.1.

6.2 Experiment 1: Assessment of Overfitting

In this experiment all the architectures from Table 5.1 are trained according to the experimental setup described in Section 6.1. To summarize, each architecture is trained for 20 epochs over ten separate runs. The available data is uniquely split for each run. For each run, for each architecture, a "final" model (the model obtained after 20 epochs of training) is produced. The "final" models are then used in this experiment.

Both the loss and accuracy of the training and validation sets are calculated and stored. We use these values to calculate the loss and accuracy ratios. The loss ratio is calculated by dividing the training loss by the validation loss (see Section 3.7.7). Similarly, we calculate the accuracy ratio by dividing the training accuracy with the validation accuracy (see Section 3.7.8). The loss and accuracy

ratios serve as a proxy for measuring overfitting. In order to take both metrics into account, we plot these metrics versus each other and determine overfitting based on how far a datapoint is from the (1,1) point. The (1,1) point is the best score the models can achieve in terms of loss and accuracy ratios (although there is a point to be made that being closer to this point does not necessarily mean the model is better, more is elaborated on this in Section 7.2). We do not attempt to postulate a threshold with regard to either ratio when deciding which architectures overfit and which do not, but rather take the five architectures that are the furthest from the (1,1) point (this is done using a positional ranking scheme, discussed in the next paragraph). We assume that these architectures are more prone to overfitting than the other architectures in our study, based on the discussion of these metrics in Sections 3.7.8 and 3.7.7. These are the architectures we use to test the efficacy of various regularization methods as is discussed in the next experiment.

The position ranking is calculated as follows for a specific run: sorting the distances of the models in ascending order (i.e. furthest first, closest last), the numerical value of their position minus one is then added to their rank. This is then repeated for each run, with each subsequent ranking added, after which we have the position ranking. In our case, since there are 13 models with 10 runs each and we subtract one from the numerical value of their position the maximum position ranking which is obtainable is 120. This is done per run rather than using the mean distances of each architecture to ensure we give the same weight to each individual run. The top five models in the position ranking (i.e. the furthest) are the most prone to overfitting and will be used in the next experiment.

6.3 Experiment 2: Intervention of Regularization Techniques

Having selected candidate architectures that are likely overfitting, we apply several interventions to test which one is the most effective regularization method with regards to our metrics.

We regularize the 5 models selected in Experiment 1 using either L2 kernel regularization or convolutional drop out or a combination of both. To have a baseline comparison, the models that made use of regularization were retrained. The retraining was done after having removed the above two interventions from them if an architecture employed them. These baseline architectures, however, still make use of dropout in their dense layers. An `l2` factor of 0.01 was used.

The experiment described above provides three intervention strategies and the baseline for comparison. The methods are then compared based on the effect that they have on the loss and the accuracy of each architecture instance, both

with reference to training and validation error. For each architecture (over all 10 runs), we then select the regularization strategy that resulted in the best validation accuracy and validation loss for the next experiment. Moreover, a model is only included in the next experiment if it does not overfit (which is determined by evaluating the loss and accuracy ratios defined in Section 3.7.7 and Section 3.7.8).

6.4 Experiment 3: Architecture Comparisons

The best intervention strategy is then selected to replace the default regularization strategies of the five most overfitted models. These models' results from the previous experiments are then used in the ranking based on classifier accuracy (classifier ranking) and ranking based on computational cost (computational ranking).

The aforementioned rankings are calculated using a round-robin "tournament" in which each architecture is compared to every other architecture (excluding itself) in several different categories. If an architecture achieves a higher or lower score (which depends on the metric of the category under consideration) than a "competing" architecture does in a specific category then the former architecture's ranking is incremented by one, while the latter architecture's ranking is decremented by one. As alluded to before, this comparison is repeated for every category and every architecture-pair. A higher category score is better in the case of recognition performance metric categories, while a lower category score is better in the case of computational requirement metric categories. To establish a classifier ranking, the MPCA, the per class F1-score, precision and recall of the different architectures are compared with one another (i.e. a total of 13 categories are considered). To establish a computational ranking, GPU memory usage, floating point operations and inference time are compared with one another (i.e. a total of 3 categories are considered).

Chapter 7

Results

> *"The process may seem strange and yet it is very true. I did not so much gain the knowledge of things by the words, as words by the experience I had of things."*
>
> — Plutarch

In this chapter we present the results of the experiments described in the previous chapter. The results of Experiment 1 (assessment of overfitting) is presented in Section 7.1. The results of the Experiment 2 (assesment of regularization interventions) is discussed in Section 7.2. The results of Experiment 3 is discussed in Section 7.3 in which the ranking of the different architectures has been established. The aforementioned ranking takes into account the recognition performance and the computational requirements of the different architectures.

7.1 Results of Experiment 1: Assessment of Overfitting

The results of Experiment 1 (Section 6.2) are given in Figures 7.1, 7.2 and 7.3.

The main metrics that are being considered for this experiment are two ratios: the ratio of the training loss to the validation loss, and the ratio of the training accuracy to the validation accuracy. We will simply refer to these ratios as the loss ratio and accuracy ratio further in the text. In Figure 7.1, we plot the accuracy ratio versus the loss ratio. Figure 7.1 provides us with an estimate of whether the architectures are overfitting on the data. Each individual datapoint represents a model which was trained during a specific run, with the colour indicating the architecture. The triangles indicate the mean values of that architecture.

A model that exhibits a higher accuracy ratio is more prone to overfitting than a model with a lower accuracy ratio. A higher accuracy ratio is indicative of a model that scores better on the training set than on the validation set. Conversely, a model that exhibits a lower loss ratio is more prone to overfitting than a model

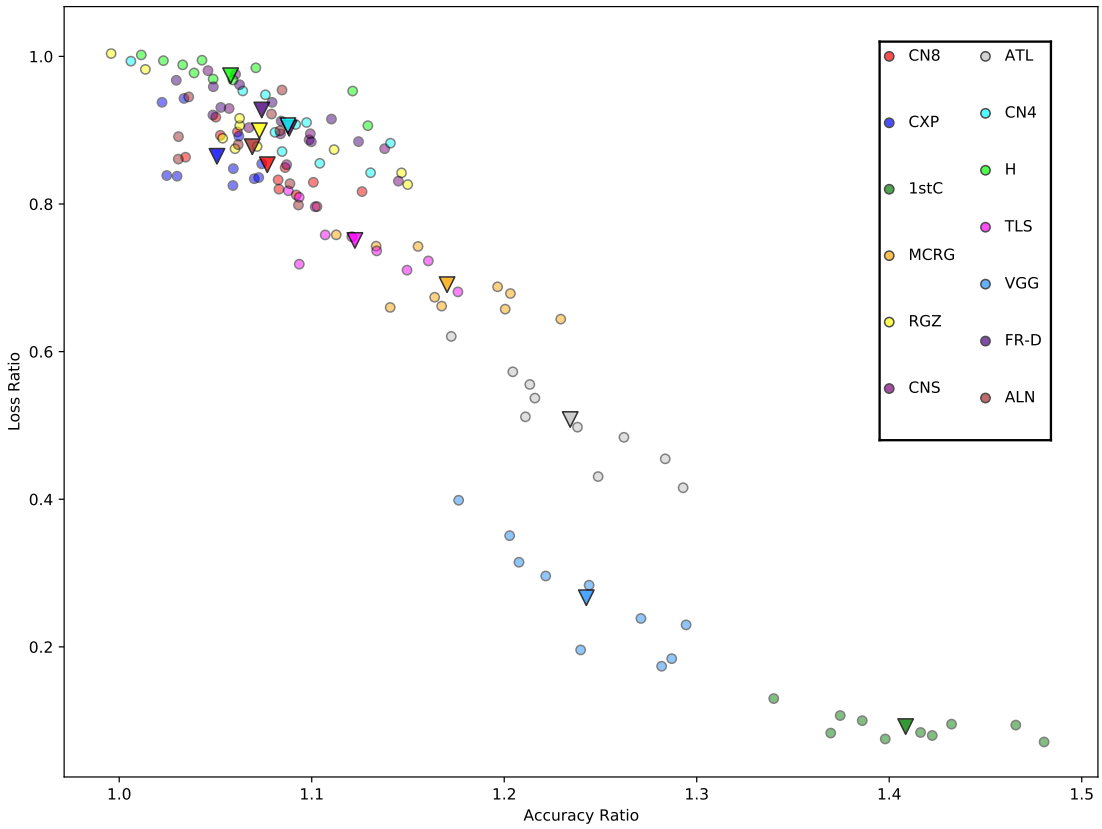

Figure 7.1: Loss Ratio vs Accuracy Ratio for each of the models after having been trained for 20 epochs.

with a higher loss ratio. A higher validation loss relative to a lower training loss is indicative of overfitting.

The further a model is from (1,1) the more prone it is to overfitting. In order to calculate which five architectures is most prone to overfitting we use a ranking scheme. The results of this ranking is given in Table 7.1 as well as the mean distance each architecture is from (1,1).

Architecture Key	**Position Ranking**	**Mean Distance**	**Std Dev**
1stC	0	0.9963	0.03
VGG	10	0.7728	0.08
ATL	20	0.5451	0.07
MCRG	30	0.3537	0.05
TLS	40	0.2782	0.05
CN8	65	0.1664	0.04
CXP	78	0.1456	0.05
ALN	78	0.1438	0.05
CNS	81	0.1332	0.05
RGZ	82	0.1262	0.07
CN4	82	0.1294	0.06
FR-D	97	0.1045	0.04
H	117	0.0646	0.05

Table 7.1: Positional Ranking and Mean Distance from (1,1) with standard deviation included.

The architectures furthest from (1,1) are Toothless, MCRGNet, ATLAS, VGG16D and FIRST Class. These are the architectures to be considered in Experiment 2. For the sake of completeness, we also investigate the relationships between the performance on the training and validation sets for the loss and accuracy metrics.

Figure 7.2 shows the training loss versus the validation loss of each architecture. Each individual datapoint represents a model trained for a run, with the colour indicating the architecture. The triangles indicate the mean values of that architecture. The crescent shape has models that overfit on the left end and models that likely underfit on the right end, with the best trade-off models near the middle. Architectures with a much lower training loss than validation loss on the left side of the figure are the candidates identified for overfitting with our distance metric in Figure 7.1. On the right side of the figure, architectures with a training loss similar to their validation loss can be found. This shows one of the limitations

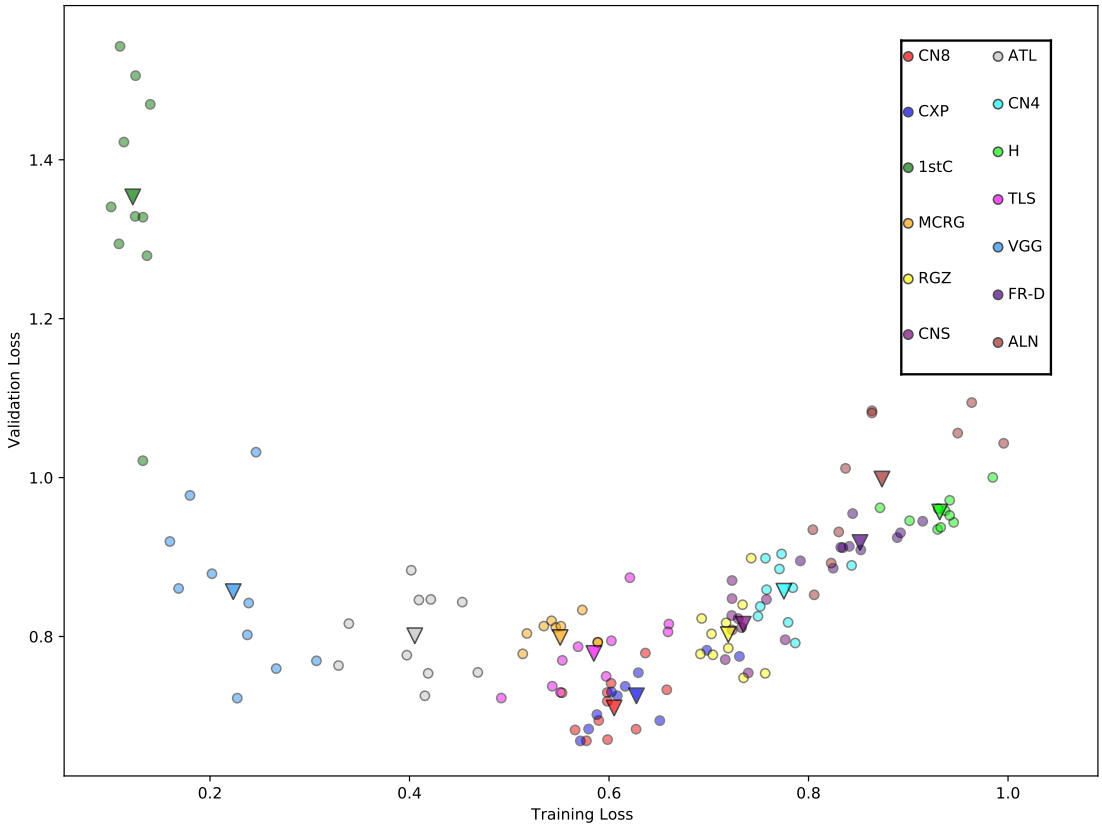

Figure 7.2: Training vs Validation Loss for model after 20 epochs of training

of the ratio plot in Figure 7.1, it is difficult to identify models that underfit or are overregularized.

Figure 7.3 shows the training accuracy versus the validation accuracy of each architecture. Each individual datapoint represents a model trained for a run, with the colour indicating the architecture. The triangles indicate the mean values of that architecture.

Next we look at the five models identified as most prone to overfitting and the effect regularization has on them.

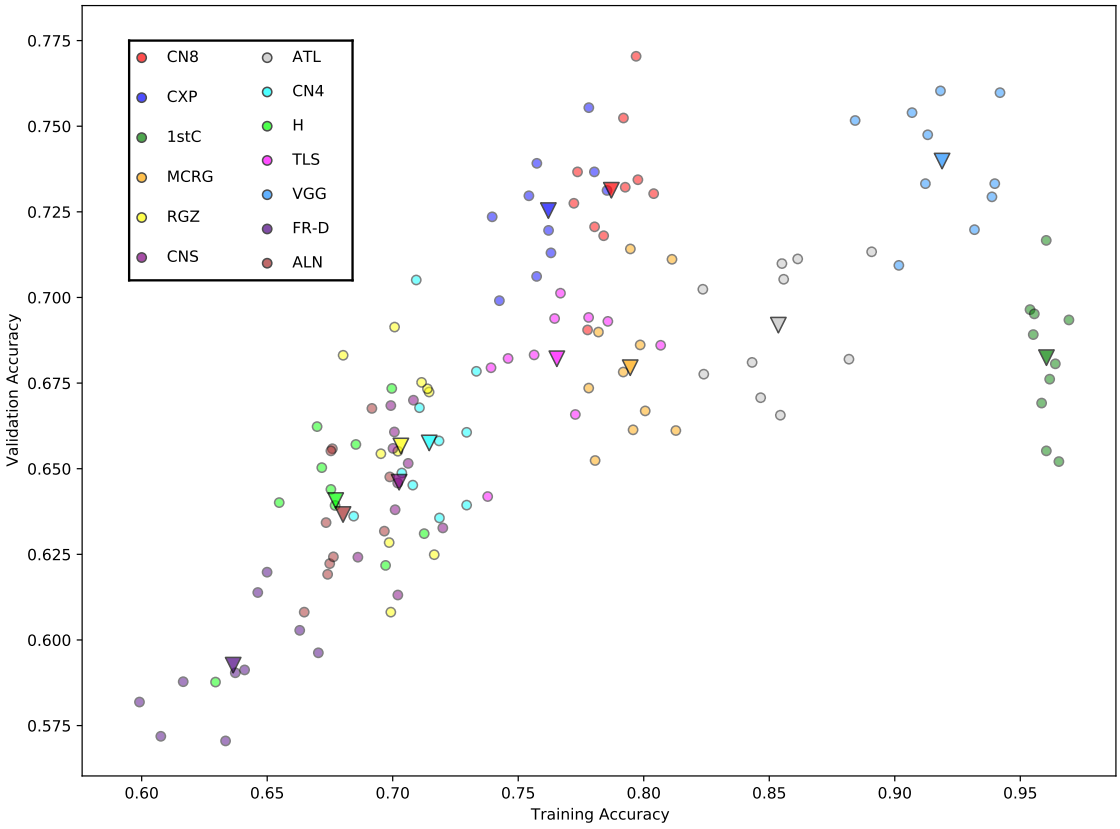

Figure 7.3: Training vs Validation Accuracy for model after 20 epochs of training

7.2 Results of Experiment 2: Intervention of Regularization Method

We regularize the 5 models selected in Experiment 1 using either L2 kernel regularization or convolutional drop out or a combination of both. To have a baseline comparison, the models with some regularization were retrained without utilizing either regularization method. The baseline models, however, still make use of dropout between their dense layers. Each of the regularization methods are indicated by a shorthand in the figures: with L2 kernel regularization as L2, convolutional drop out as D, the combination of both as L2D and the baseline model with neither method as NCD (no convolutional dropout).

Figure 7.4 shows the loss and accuracy ratio of the results from Experiment 2 over all the runs. We provide some additional information is to better understand the figures in this section: the method of regularization used on a specific architecture is indicated by the marker shape and the colour indicates the architecture, the smaller markers are the results for each individual run and the larger markers with the black edges are the means of each architecture.

Table 7.2 shows the average distance of each architecture from (1,1). The original architectures indicated with an asterisk in the table are the original architecture given in Table 7.1. Note that the Toothless models in Table 7.1 are derived from an architecture that uses batch normalization which were removed from all the Toothless variants in Table 7.2, since batch normalization can potentially affect regularization and complicate the comparison study. The architecture that closest resembles the original Toothless here is the no convolutional dropout (NCD) variant with the only difference being the lack of batch normalization. To illustrate this, we mark Toothless NCD with a dagger in Table 7.2. This exclusion of batch normalization from the architecture caused an increase in mean distance from (1,1). We also note that further research into the effect of batch normalization within the context of radio galaxy classification should be done, but that this is beyond the scope of the current study.

Figure 7.5 shows the means of the loss and accuracy ratio of the results from Experiment 2, with the addition of the means for each regularization method. The crimson markers with hatches (for example see Figure 7.5) show the mean associated with a regularization method.

We note that the combination of both methods (L2D) shows the greatest decrease in distance relative to the baseline models (NCD) with the L2 kernel regularization (L2) close by (see Table 7.3). While convolutional dropout (D) does show an improvement relative to the baseline models, it does not have the same effect on First Class's models (which is severly overfitting).The other intervention regularization techniques do significantly improve the performance of First Class's models. This trend is true for ATLAS's and to a lesser extent for MCRGNet's models, but does not hold for Toothless's and VGG16D's models which both show that using convolutional dropout results in an overall significant decrease in distance to (1,1). This is likely because both of these architectures have significantly more convolutional parameters than the other three.

Figure 7.6 shows the training loss compared to the validation loss. A similar crescent shape as seen in Figure 7.2 is visible. We see that no convolutional dropout (NCD) results in a significantly lower training loss with a much higher validation loss, convolutional dropout (D) does have a significant regularizing effect, moving much closer to the middle. The effects are significantly more varied per architecture. As an example, convolutional dropout does have much of an

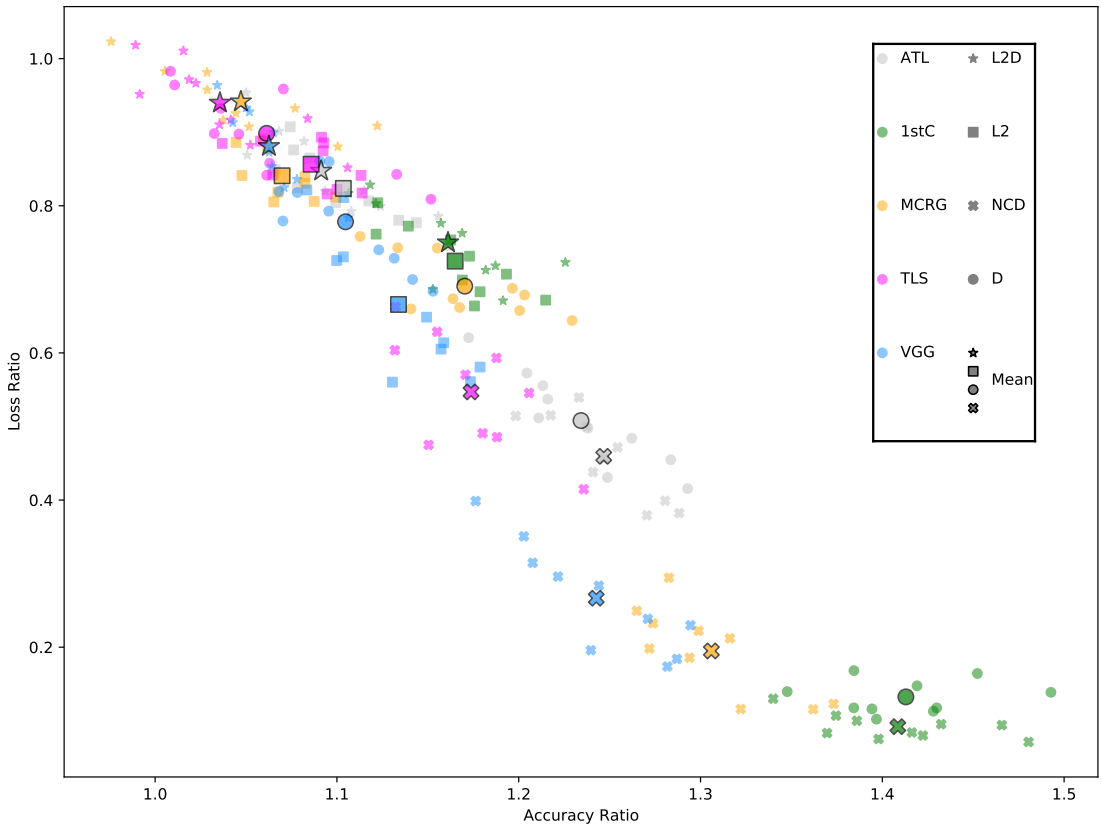

Figure 7.4: Regularization Interventions: Accuracy Ratio vs Loss Ratio of All Runs

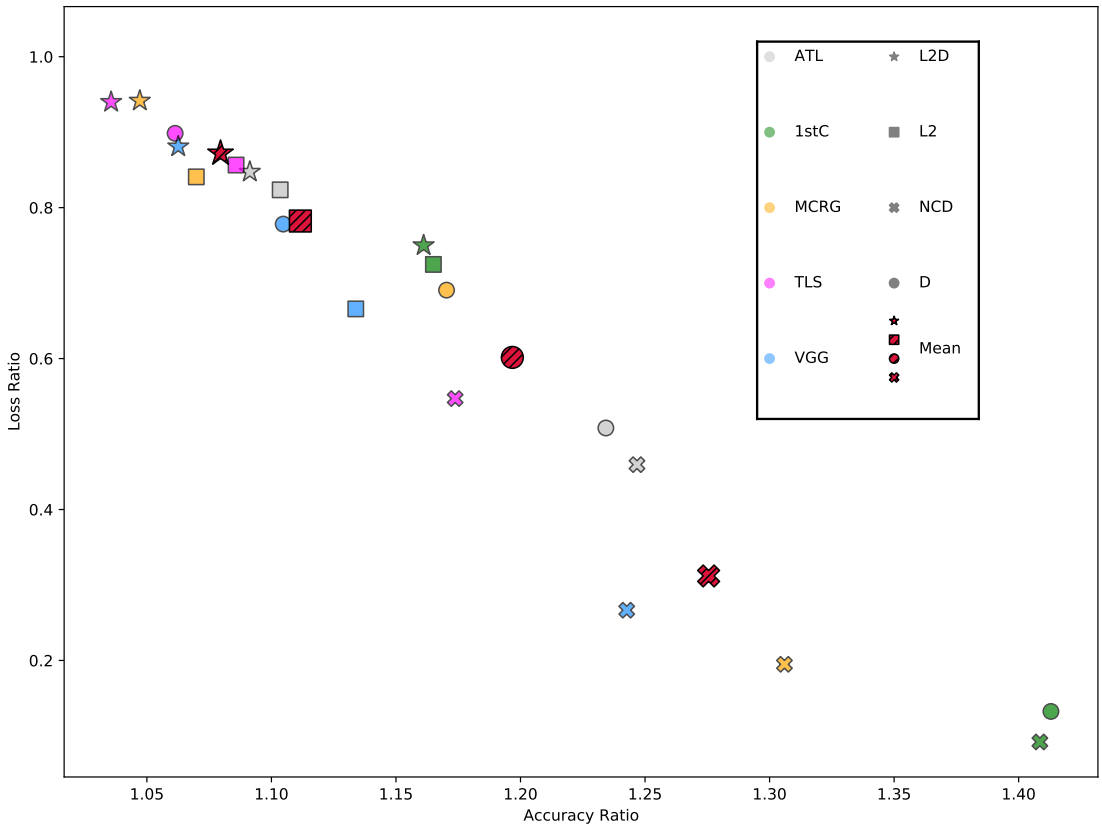

Figure 7.5: Regularization Interventions: Accuracy Ratio vs Loss Ratio of Means per Architecture

Architecture	Position Rank	Mean Distance	Std Dev
1stC NCD*	0	0.9963	0.03
1stC D	11	0.9616	0.03
MCRG NCD	20	0.8615	0.07
VGG NCD*	29	0.7728	0.08
ATL NCD	43	0.5944	0.06
ATL D*	51	0.5451	0.07
TLS NCD†	60	0.4857	0.08
MCRG D*	79	0.3537	0.05
VGG L2	80	0.3604	0.10
1stC L2	89	0.3212	0.05
1stC L2D	97	0.2981	0.06
VGG D	111	0.2459	0.07
ATL L2	125	0.2050	0.06
TLS L2	137	0.1683	0.04
MCRG L2	138	0.1743	0.04
ATL L2D	140	0.1787	0.06
VGG L2D	158	0.1351	0.06
TLS D	165	0.1210	0.07
MCRG L2D	182	0.0832	0.05
TLS L2D	185	0.0781	0.05

Table 7.2: Position Ranking and Distance of each Regularized Architecture

impact on First Class, while Toothless sees a significant increase in validation accuracy. L2 kernel regularization without convolutional dropout (L2) produces the lowest average validation loss without an accompanying lower training loss (which is normally indicative of overfitting). L2 kernel regularization with convolutional dropout (L2D) results in an increase of both training and validation loss when it is compared to L2 without convolutional dropout. Figure 7.7 shows the training accuracy compared to the validation accuracy, with L2 scoring the highest validation accuracy.

The average distances and positional rank of each of the regularization methods is given in Table 7.3. The average validation accuracy, loss and accuracy ratios of each of the regularization methods are given in Table 7.4.

In Table 7.3, we note that the positional ranking now includes the sum of all the models' ranks that use a specific regularization method. L2D has a much larger effect on the distance metric than the other methods do, although as we have mentioned before this can be misleading with regards to loss and accuracy values because the ratios do not take the absolute values of accuracy and loss into

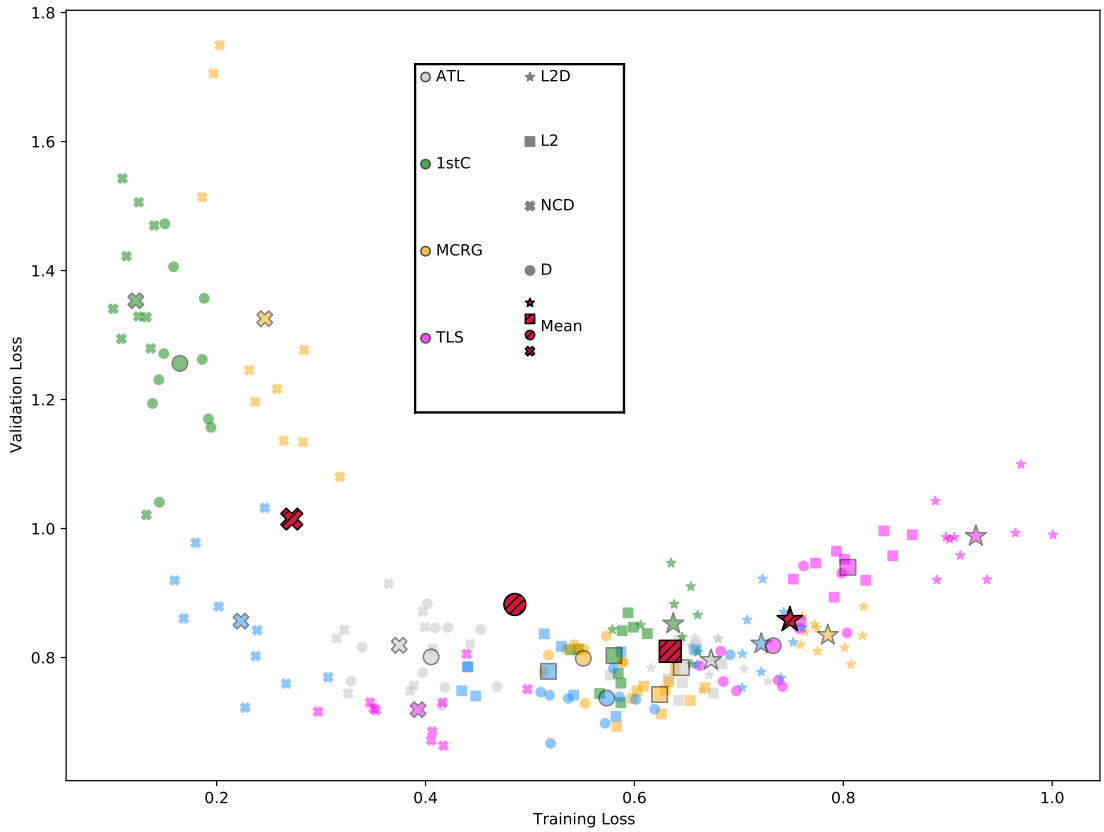

Figure 7.6: Regularization Intervention: Training Loss vs Validation Loss

Regularization Method	**Position Rank**	**Average Distance**
No Convolutional Dropout (NCD)	152	0.7421
Dropout (D)	417	0.4454
L2 Kernel Regularization (L2)	569	0.2458
L2 Kernel Regularization & Dropout (L2D)	762	0.1547

Table 7.3: Position Ranking and Mean Distance of each Regularization Method

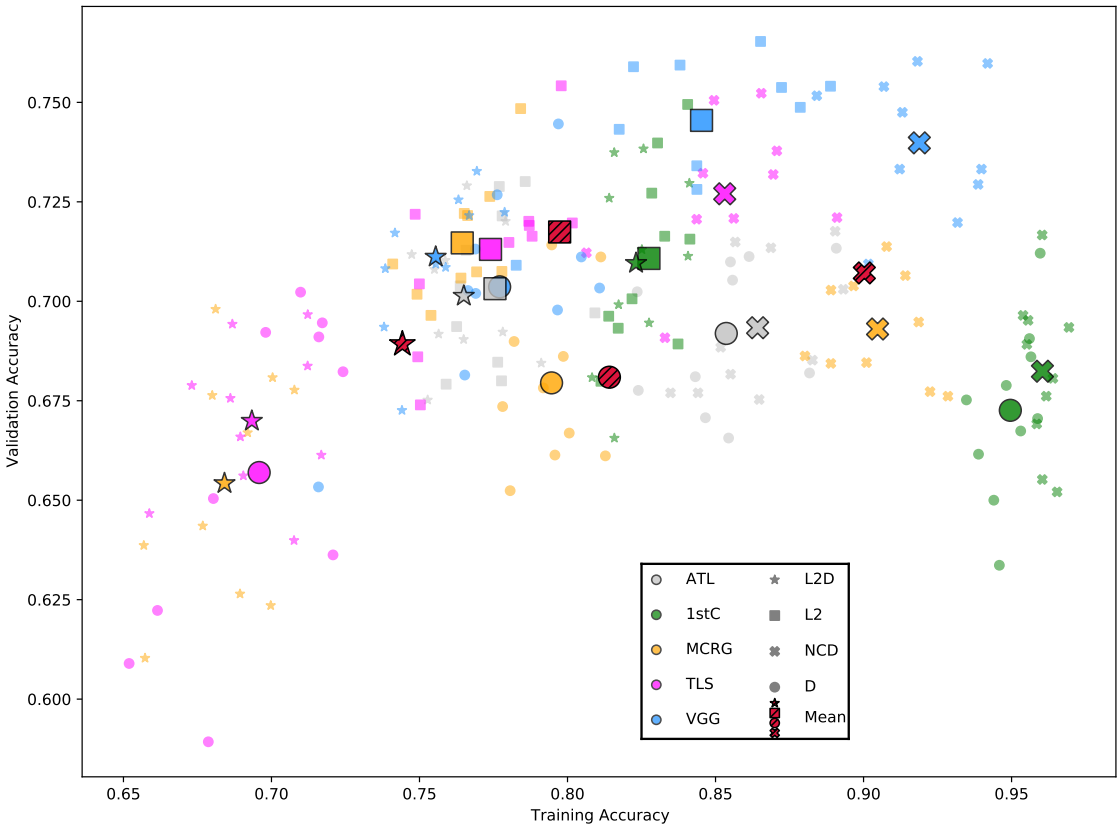

Figure 7.7: Regularization Intervention: Training Accuracy vs Validation Accuracy

Reg. Method	**Accuracy**			**Loss**		
	Train	**Val.**	**Ratio**	**Train**	**Val.**	**Ratio**
NCD	90.03	70.71	1.2755	0.2719	1.0146	0.3120
D	81.41	68.09	1.1967	0.4855	0.8822	0.6016
L2	79.73	71.74	1.1116	0.6343	0.8097	0.7823
L2D	74.42	68.93	1.0796	0.7488	0.8585	0.8719

Table 7.4: Regularization Method Mean Accuracy, Loss and Ratios

account.

In Table 7.4 we see that L2 has associated with it the lowest validation loss and the highest validation accuracy of all regularization methods. While the L2D method has associated with it the best loss ratio and accuracy ratio, we see signs of underfitting relative to the other methods with a more than 5% lower training accuracy than L2 and a nearly 3% lower validation accuracy. NCD exhibits clear signs of having overfitting with a very high training accuracy. D performs worse than L2D on validation accuracy. We select L2 kernel regularization as the intervention we will apply, based on its performance on the validation set and since it does not show any clear signs of overfitting.

From the results of Table 7.3 and 7.4 we conclude that the most effective regularization intervention that does not compromise performance is the addition of L2 kernel regularization in the first dense layer.

7.3 Results of Experiment 3: Architecture Comparisons

All the results presented in the subsequent subsections were obtained from Experiment 3 (see Section 6.4). The results of this experiment is summarized in Table 7.6. In Section 7.3.1, we report on the MPCA versus the computational complexity of the architectures in Table 5.1. Note that, for the sake of brevity we sometimes only use the shortened phrase "architectures" instead of "architectures in Table 5.1" when referring to the architectures that we considered in this thesis. Section 7.3.2, looks at the per class F1-score performance of the architectures (which serves to showcase the trade-off in class performance for each classifier and highlights the shortfalls of a metric such as MPCA). The memory requirements and the classification speeds of the architectures are discussed in Section 7.3.3 and Section 7.3.4. The overall ranking of the architectures is presented in Section 7.3.5.

7.3.1 Accuracy-rate vs Computational Complexity vs Model Complexity

Figure 7.8 reports the MPCA versus the computational complexity of the architectures in Table 5.1; for a single forward pass (measured in floating point operations or FLOPs). The size of the markers in Figure 7.8 represents the model complexity of the architectures (measured in the number of trainable parameters). These values are presented in Table 7.5.

The model with the highest MPCA (74.38%) is CLARAN (i.e. VGG16 L2) (Wu *et al.*, 2019; Simonyan and Zisserman, 2015). The best performing models

from the existing literature ConvNet8 (Lukic *et al.*, 2019*b*) (73.18%), FIRST Class L2 (Alhassan *et al.*, 2018) (71.67%), MCRGNet L2 (Ma *et al.*, 2019) (71.38%) and Toothless L2 (Aniyan and Thorat, 2017) (71.34%). The novel classifier produced for this thesis, ConvXpress, has the third highest MPCA (72.69%). These results are presented in Table 7.6.

We note that all of the best performing classifiers make use of an L2 kernel regularization layer and that nearly all of the architectures that have been regularized are in this group.

The figures in this section was in part inspired by work done by Bianco *et al.* (2018) who performed a benchmark analysis of various deep neural network architectures at the time. We drew inspiration from the various plots to represent the data, such as Figure 7.8.

Key	**Learning Rate**	**Parameters**	**FLOPs**	**GPU Mem. (GB)**
MCRG L2	0.001	213916	8406674	63.49
RGZ	0.00001	5283444	75825974	84.99
H	0.0001	261239	109649469	315.39
FR-D	0.0001	479996	141486958	280.58
ATL L2	0.0001	1385988	546585022	411.65
CXP	0.0001	3415944	764997460	393.22
ALN	0.00001	37302980	1107736302	412.67
CNS	0.0001	37460	797660134	573.44
CN4	0.0001	165910168	1036529876	897.02
TLS L2	0.00001	71906180	1634645742	782.34
1stC L2	0.0001	51655412	1128841236	1715.20
CN8	0.0001	42646184	4612528724	1844.22
VGG L2	0.00001	201384644	27231525758	4061.18

Table 7.5: Learning Rate, Parameters, FLOPs and GPU Memory

7.3.2 Per Class F1-score

Figures 7.9 through 7.12 report the F1-score of each class sorted by mean F1-score. The results are also given in Table 7.7. The results are represented as violin plots for each architecture. Violin plots use kernel density estimation (KDE) to estimate a probability distribution of the data. The middle tick present on each architecture's violin represents the median value. F1-score encapsulates recall and

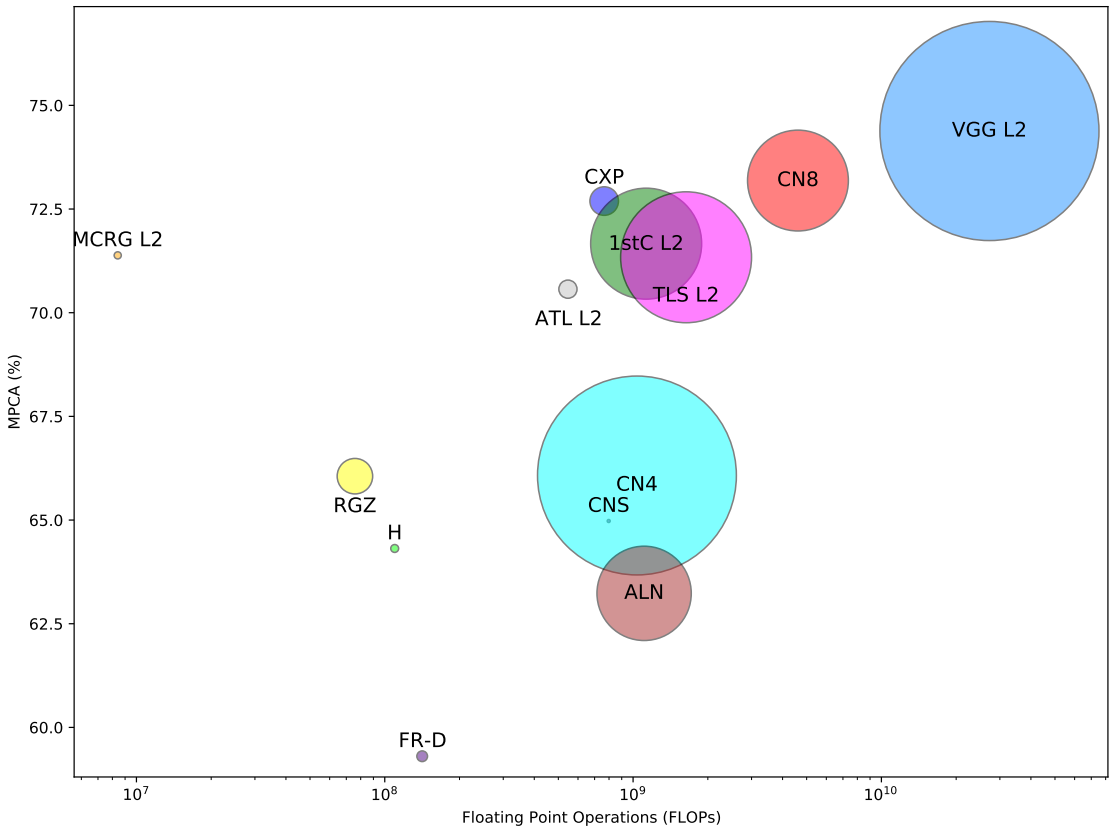

Figure 7.8: MPCA vs Parameters vs FLOPs: All exact values are available in Table 7.5.

Key	Computational Rank	Classifier Rank	MPCA
1stC L2	-18	488	71.67
ALN	4	-884	63.23
ATL L2	8	249	70.57
CN4	-14	-470	66.07
CN8	-28	857	73.19
CNS	-12	-738	64.98
CXP	6	636	72.69
FR-D	10	-975	59.30
H	20	-611	64.31
MCRG L2	36	474	71.38
RGZ	30	-577	66.06
TLS L2	-14	459	71.34
VGG L2	-36	1092	74.38

Table 7.6: Rankings and MPCA Table: Computational Ranking, Classifier Ranking and MPCA is given. Architectures are sorted alphabetically according to their keys.

precision into a single metric. More general metrics, such as MPCA and overall accuracy, can be misleading metrics, since a model might score high in either of these metrics by doing exceptional well in one class, whilst underperforming in another class. These results are presented in Table 7.7. Classifiers should in general not be evaluated using a single metric. However, a single metric is sometimes necessary as it can convey information in a concise and succinct manner.

Most architectures fair well classifying Compact sources, with even the worst performing architecture in Figure 7.9 outperforming the best F1-score for the other classes. Compact sources have a more straightforward morphology (for the most part) than the other classes. This more straight forward morphology allows the classifiers to learn the class distinction easier. VGG L2 has the highest mean and median score for this class, although some VGG L2 models performed worse than MCRGNet L2's models, which score close to each other. We note that MCRGNet L2 (213916 parameters) is a much less complex model than VGG16 L2 (201384644 parameters) in terms of parameter size with accuracy scores quite comparable for the compact class.

The FRI class was more difficult for the classifiers to learn than Compact sources, although the worst performing classifier for the FRI class does not drop below 0.5, as it does for FRII and Bent-tails. The results are given in Figure 7.10 VGG L2 has the highest mean and median score for this class. VGG L2 does, however, have greater variability in model performance than the 2nd and 3rd best

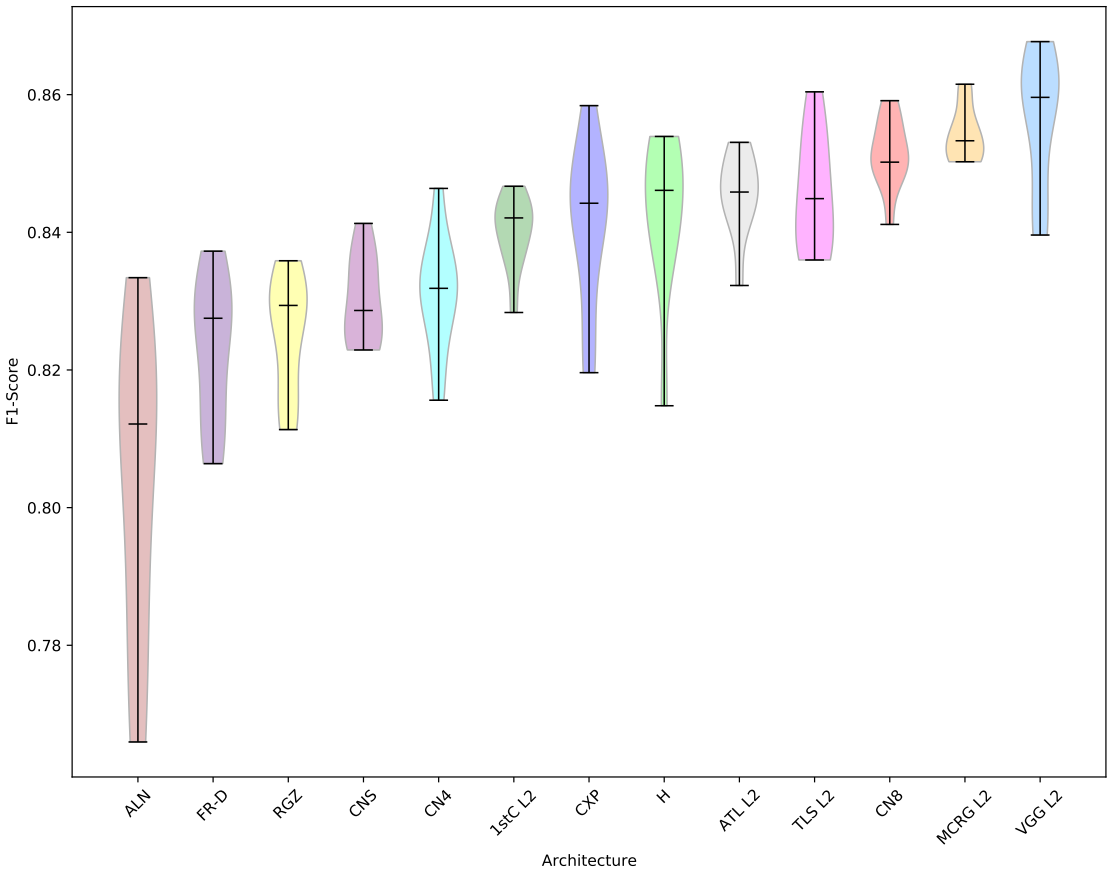

Figure 7.9: Compact Source F1-score Violin Plot: Architectures are sorted according to the means with the median value is displayed on the plot.

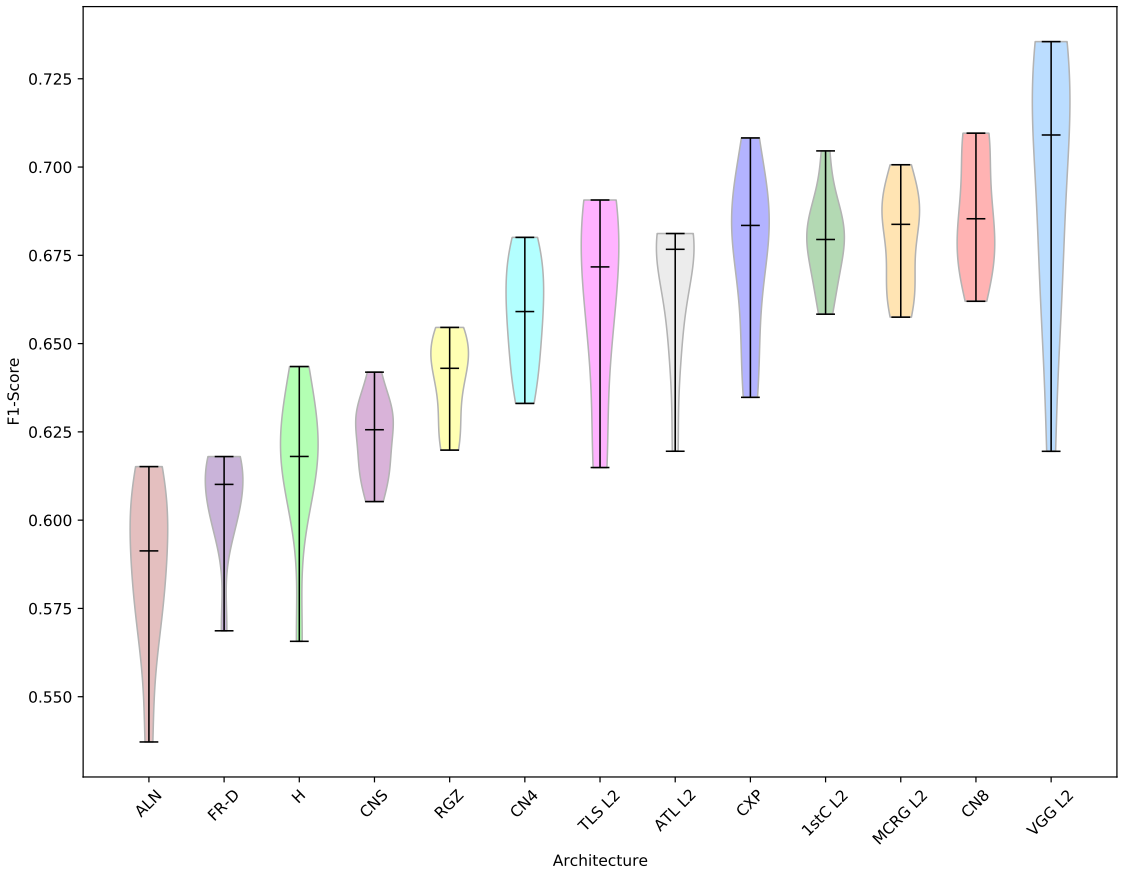

Figure 7.10: FRI F1-score Violin Plot: Architectures are sorted according to the means with the median value is displayed on the plot.

architectures, i.e. ConvNet8 and MCRGNet L2, respectively.

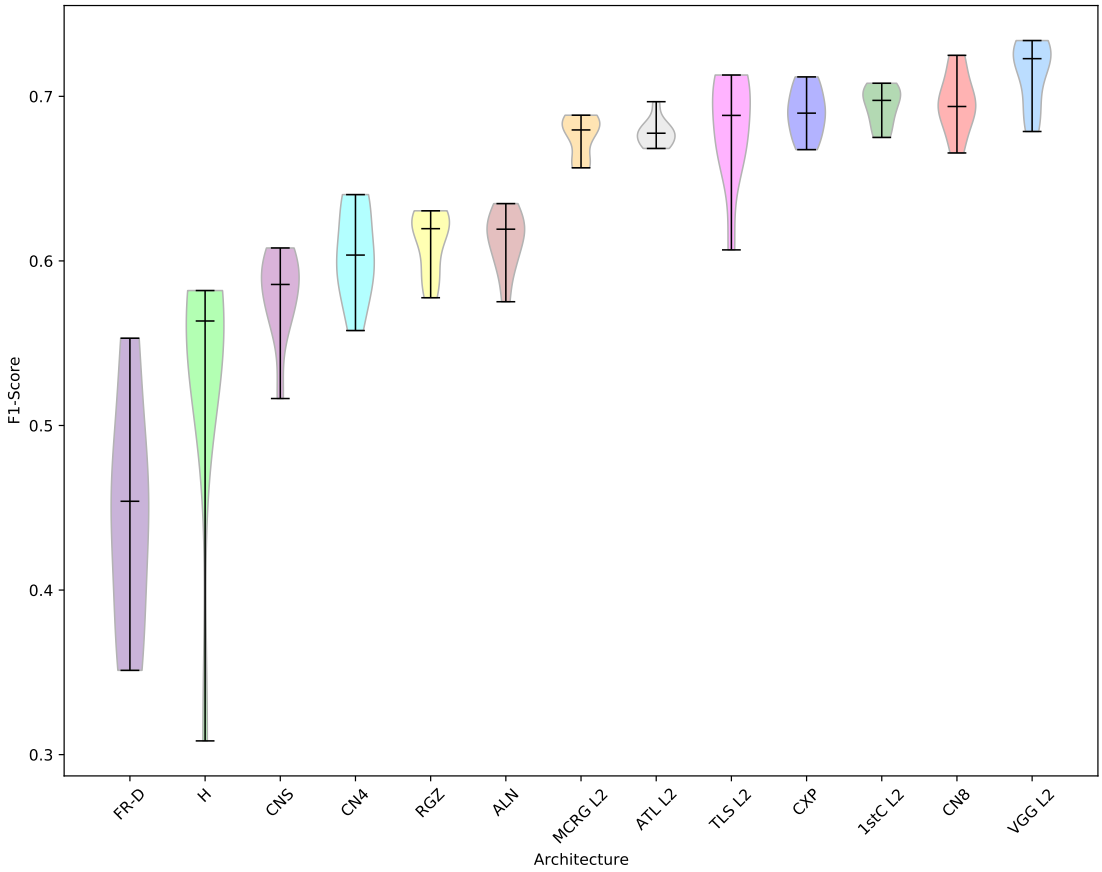

Figure 7.11: FRII F1-score Violin Plot: Architectures are sorted according to the means with the median value is displayed on the plot.

The results for the FRII class show some models getting a F1-score below 0.5, likely because of confusion between FRII images and Bent-tail images. In Figure 7.11 a divide is present between architectures that use L2 kernel regularization and those that have not (with the exception of ConvNet4), with a sudden rise in F1-score between AlexNet and MCRGNet L2. VGG L2 is the best performing architecture in this class. The models of the architectures with L2 kernel regularization show less variability in F1-score (with the exception of Toothless and ConvNet4) than the architectures without it.

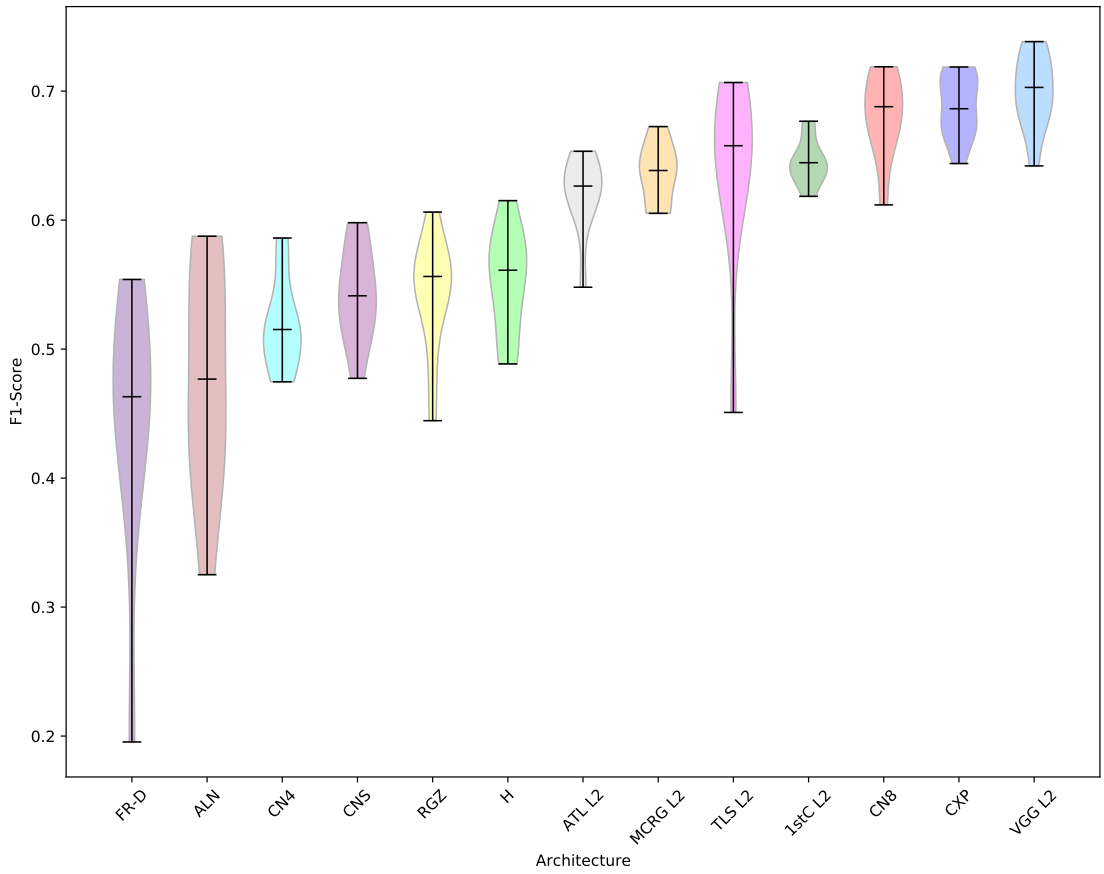

Figure 7.12: Bent-tail F1-score Violin Plot: Architectures are sorted according to the means with the median value is displayed on the plot.

Figure 7.12 reports a violin plot of the F1-score for the Bent-tail class. As with the results from the FRII class, we see some architectures getting an average F1-score below 0.5, likely because of confusion between FRII images and Bent-tail images. The architectures with L2 kernel regularization (with the exception of ConvNet4) are also the best performers for the Bent-tail class, although not as pronounced as in the FRII class. VGG L2 is the best performing classifier.

7.3.3 Classification Speed

Classification speed is the number of images an architecture can classify per second at a certain batch size. A batch size of 32 has been used for this experiment. The classification speed of the different architectures are obtained by taking the inverse of the inference times reported in Figure 7.13. Figure 7.14 reports the MPCA versus classification speed of the different architectures. It is evident from Figure 7.14 that computationally efficient models generally have faster classification speeds. A trade-off, therefore, exists between faster classification and higher recognition performance, at least for standard CNN architectures. Moreover, MCRGNet has the fastest classification speed at 1270 images per second with Radio Galaxy Zoo close by at 1246 images per second. VGG16 has the slowest classification speed at 291 images per second. These results are presented in Table 7.8.

In comparison, the classification speed of an average person is "about 250 images in 5 minutes" or roughly 0.833 images per second (Markoff, 2012). The classification task from which this result was obtained is complex, a human classifier had to choose a label from a large number of possibilities, and as such the aforementioned result should be regarded as a lower bound estimate of how fast an average person would be able to classify 250 images.

7.3.4 Inference Time vs. GPU Memory Usage

Figure 7.13 reports inference time versus theoretical GPU memory usage (at a batch size of 32). As memory usage increases, inference time dramatically increases. The standard deviation associated with the inference time is also depicted in Figure 7.13. Inference time is tabulated in Table 7.8 and GPU memory in Table 7.5.

7.3.5 Ranking

All the architectures listed in Table 5.1 have been ranked in Table 7.6 according to their recognition performance (given as classifier ranking) and their computational performance (given as computational ranking). An overall rank is calculated based on the sum of these two rankings. Please note that this ranking is a relative ranking

Key	**Comp. Prec.**	**FRI Prec.**	**FRII Prec.**	**Bent Prec.**
VGG L2	0.8283	0.7196	0.7145	0.7349
CN8	0.8077	0.7040	0.6867	0.7269
CXP	0.7926	0.7023	0.6917	0.7276
1stC L2	0.8161	0.6625	0.6933	0.6922
TLS L2	0.8215	0.7063	0.6693	0.7151
MCRG L2	0.8283	0.6815	0.6691	0.6745
ATL L2	0.8198	0.6681	0.6661	0.6644
RGZ	0.7863	0.6292	0.5994	0.6195
H	0.8215	0.5709	0.5996	0.5924
CNS	0.7824	0.6055	0.5984	0.5926
ALN	0.7566	0.5663	0.5862	0.6401
CN4	0.7768	0.6354	0.6286	0.5801
FR-D	0.8254	0.5347	0.5378	0.4890
	Comp. Recall	**FRI Recall**	**FRII Recall**	**Bent Recall**
VGG L2	0.8894	0.6785	0.7208	0.6866
CN8	0.9008	0.6724	0.7082	0.6460
CXP	0.9012	0.6517	0.6919	0.6629
1stC L2	0.8662	0.6976	0.6963	0.6065
TLS L2	0.8758	0.6330	0.7202	0.6245
MCRG L2	0.8829	0.6785	0.6869	0.6071
ATL L2	0.8747	0.6664	0.6924	0.5892
RGZ	0.8713	0.6514	0.6301	0.4894
H	0.8686	0.6740	0.5041	0.5260
CNS	0.8848	0.6427	0.5673	0.5043
ALN	0.8649	0.6117	0.6573	0.3954
CN4	0.8956	0.6857	0.5876	0.4740
FR-D	0.8235	0.7009	0.4029	0.4449
	Comp-F1	**FRI-F1**	**FRII-F1**	**Bent-F1**
VGG L2	0.8564	0.6963	0.7142	0.6995
CN8	0.8515	0.6874	0.6959	0.6818
CXP	0.8412	0.6748	0.6903	0.6883
1stC L2	0.8401	0.6794	0.6943	0.6452
TLS L2	0.8457	0.6632	0.6819	0.6425
MCRG L2	0.8542	0.6795	0.6761	0.6362
ATL L2	0.8456	0.6669	0.6779	0.6218
RGZ	0.8253	0.6396	0.6118	0.5429
H	0.8429	0.6170	0.5342	0.5517
CNS	0.8301	0.6232	0.5797	0.5422
ALN	0.8037	0.5873	0.6128	0.4780
CN4	0.8310	0.6586	0.6043	0.5178
FR-D	0.8229	0.6059	0.4458	0.4499

Table 7.7: Average results of the F1-score, Precision and Recall are given in this table.

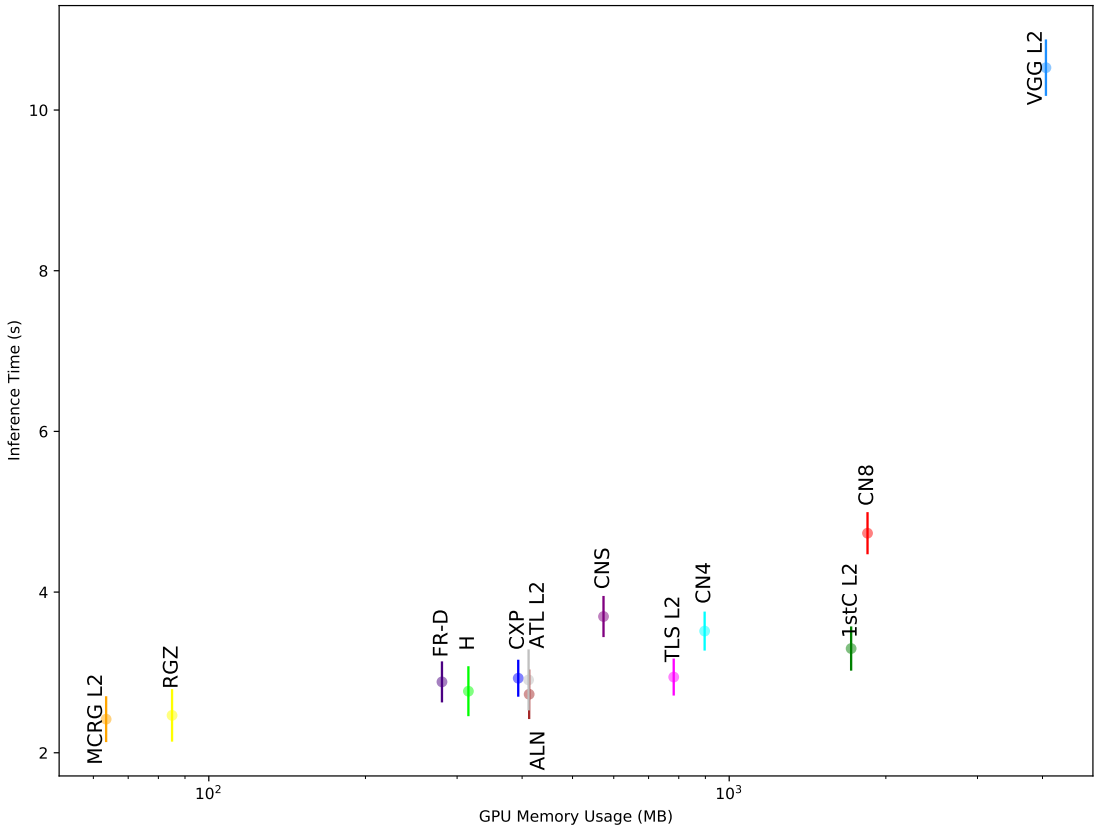

Figure 7.13: GPU Memory Usage vs Inference Time

and that it is limited to the architectures within this study (and the datasets used) and as such cannot be seen as an absolute reflection of architecture standing.

Figure 7.15 shows the computational and classifier rankings of the architectures and replaced counterparts. For precise values see Table 7.6. The computational rank takes into account memory requirements, classification rate and FLOPs (see Table 7.5). This is calculated by comparing the models of each architecture per run. For each of these metrics the architecture gains a point for having the lower value and loses one for having the higher value (except for the classification rate where a higher value gains a point and lower loses), draws result in no increase or decrease for either party. This same scoring system is applied during each run for both the computational and the recognition performance metrics; the scoring is applied cumulatively.

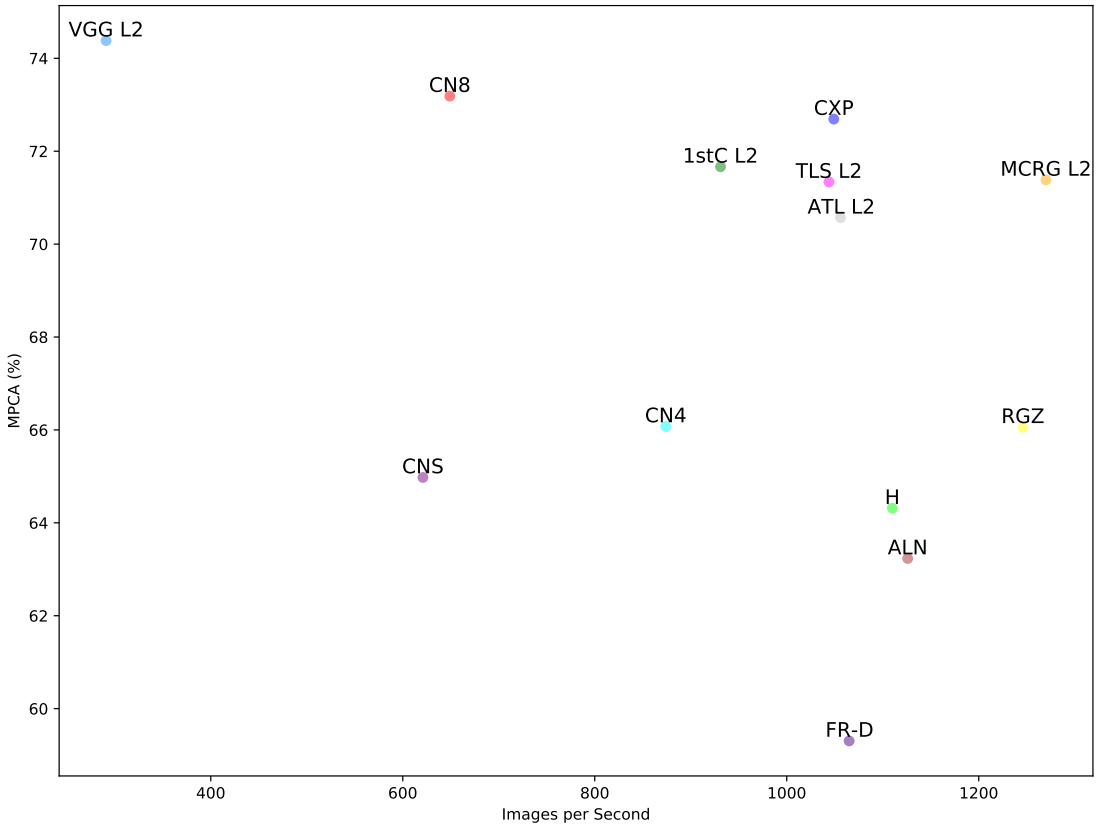

Figure 7.14: Images per Second vs MPCA

The classifier rank takes into account F1-score, precision and recall for all classes as well as mean per class accuracy. These values are given in Table 7.7 and 7.6. This is calculated by comparing the models of each architecture per run. For each of these metrics the architecture gains a point for having the higher value and loses one for having the lower value, draws result in no increase or decrease for either party.

Architectures on the right side of the figure outperform the classifiers to the left while the architectures to the top of the figure require less computational resources and have a higher classification rate compared to those near the bottom. Dividing Figure 7.15 into quadrants we see four groups of classifiers emerging: high computational rank with low classifier rank (top left), low computational rank with low classifier rank (bottom left), low computational rank with high classifier rank

Key	Images per Second	Inference Time (s)	STD
MCRG L2	1270	2.42	0.27
RGZ	1246	2.47	0.31
H	1110	2.77	0.30
FR-D	1065	2.88	0.24
ATL L2	1056	2.91	0.37
CXP	1049	2.93	0.22
ALN	1126	2.73	0.29
CNS	621	3.70	0.24
CN4	874	3.51	0.23
TLS L2	1044	2.94	0.22
1stC L2	931	3.30	0.26
CN8	649	4.73	0.25
VGG L2	291	10.53	0.34

Table 7.8: Inference Time and Images per Second

(bottom right) and high computational rank with high classifier rank (top right).

The trade-off in the loss of classifier rank for the gain in computational rank is quite significant when comparing the models in the top right quadrant with those in the bottom right. The three worst performing classifiers in terms of computational rank are in the bottom right quadrant, while the best performing architecture (MCRGNet L2) is in the top right. In terms of classifier rank the bottom right quadrant has the two best performing classifiers, with the top right having the 3rd and 6th best. MCRGNet L2, CXP and ATLAS L2 show a good trade-off between computational rank and classifier rank.

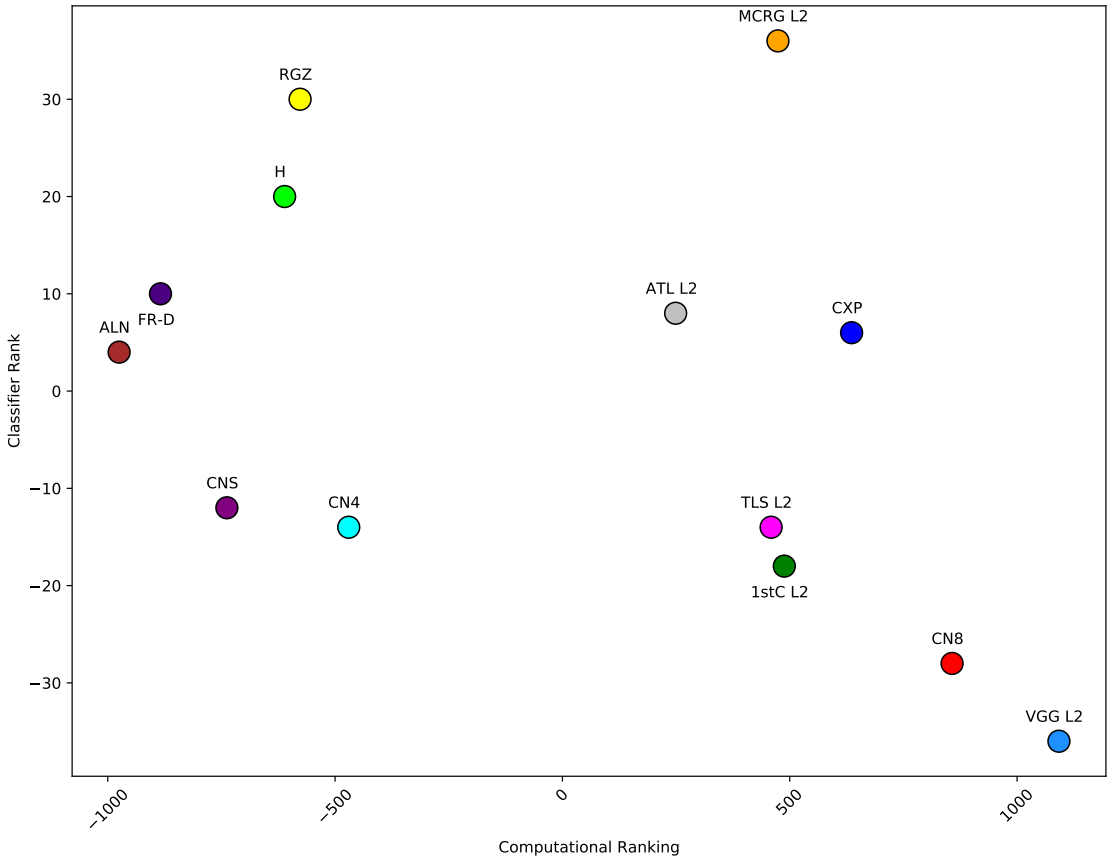

Figure 7.15: Class vs Computational Rank

Chapter 8

Conclusion

> *"Knowledge that is not being used for winning of further knowledge does not even remain – it decays and disappears."*
>
> — JD Bernal, *Science in History - Volume 3: The Natural Sciences in our Time, 1971*

In this chapter we offer some concluding remarks. In Section 8.1, we summarize the thesis. In Section 8.2, we restate the research objectives, discuss how these were addressed and conclude whether the objectives have been met. In Section 8.3, we give the general conclusions that can be drawn from this study. In Section 8.4, we discuss whether the overarching problem of developing an automatic classification pipeline for radio galaxy classification is now solved. We end the chapter with some final remarks in Section 8.5.

8.1 Summary of Thesis

Chapter 2 provides a brief background overview of radio astronomy. Chapter 3 discusses the background and history of CNNs. Chapter 4 outlines the dataset used for the experiments. Chapter 5 provides a literature review of CNNs used in radio astronomy. Chapter 6 discusses the experimental setup and experiments we perform. Chapter 7 gives the results of the experiments.

Three experiments were performed in this study. In the first experiment we assessed overfitting on the MURG dataset, the setup is discussed in Section 6.2 and the results in Section 7.1. In the second experiment we assessed the effectiveness of regularization techniques and select which regularized architectures will replace the original overfitting versions, the setup is discussed in Section 6.3 and the results in Section 7.2. In the third experiment we perform the architecture comparisons based, the setup is discussed in Section 6.4 and the results in Section 7.3.

8.2 Assessment of Objective Completion

In this section, we assess whether the objectives stated in Section 1.3 were completed. We briefly restate the objective of this study, discuss how this objective was addressed and critically evaluate each of the objectives.

8.2.1 Identifying proneness to overfitting

Objective

The architectures from the selected studies are retrained on the same dataset, we assess how prone the architectures are to overfitting relative to each other.

How was this objective addressed?

We used the loss and accuracy ratios of the validation and training sets as metrics to determine which architectures were more susceptible to overfitting. Both ratios are calculated by taking the corresponding metric from the training set and the validation set and dividing the values so obtained with one another. From this it follows that a lower accuracy ratio should indicate less overfitting (it is always larger than one), while a higher loss ratio indicates less overfitting (it is always smaller than one). If there is little to no overfitting present both ratios will be close to one. We plot these metrics against each other (accuracy ratio on the x-axis, loss ratio on the y-axis) and use a distance metric to determine the five architectures furthest from the (1,1) point (depicted in the Figure 7.1).

These are the metrics that were used to conclude which architectures show more significant signs of overfitting than other architectures. The identified five architectures from literature that overfit by employing a scatter plot of the accuracy and loss ratios (see Figure 7.1). The five architectures that were identified are: VGG, Toothless, ATLAS, MCRGNet and FIRST Class. Further results are discussed in depth in Section 7.1.

Has the objective been met?

The loss and accuracy ratios is most useful when comparisons between models trained on the same data are being made, rather than as standalone metrics to be assessed on its own to conclude overfitting. This is why we do not postulate a threshold value, but rather select the five furthest architectures from a predetermined point. The use of the loss and accuracy ratios to determine underfitting was not explored in depth and it could be useful in this regard. In conclusion, we have identified architectures that show more significant signs of overfitting than other architectures, we state that this objective has been met.

8.2.2 Assessing regularization intervention effectiveness

Objective

After identifying which architectures are more prone to overfitting, we test the various regularization interventions from the literature of selected studies on those that overfit. The most effective interventions are then applied to these overfitting prone architectures. The resulting architectures are then used to replace the original architectures in the recognition performance comparison study.

How was this objective addressed?

Two main regularization methods were used in the selected studies, namely convolutional dropout and L2 kernel regularization (see Table 5.15 for a complete breakdown per architecture). These two regularization strategies were then applied to the architectures identified in Experiment 1, we tested each method on its own as well as a combination of both. A baseline was also trained that uses no other regularization approach, other than dropout between the dense layers. The method with the best validation loss and accuracy across all of the overfitting architectures was then selected (and that showed reduced signs of overfitting). The five original architectures are then replaced in the comparison study with their regularized versions. These regularized versions use the aforementioned method with the best validation loss and accuracy. We found that applying convolutional dropout on its own yielded poor results. L2 kernel regularization gave the best results (and as such is the regularization strategy we would suggest to try first for the problem at hand). Using convolutional dropout and kernel regularization in conjunction usually resulted in a overregularized model.

Has the objective been met?

Further study of kernel regularization's effect when applied more broadly in the architecture is needed, since the kernel regularization we applied was limited to a single layer in all the networks. This was done to mimic the studies in literature who employed kernel regularization (see ConvNet8 as an example). We also do not explore the effect that the dropout rate has on combating overfitting since all of the selected studies used a similar dropout rate. We do not explore varying the number of dropout layers either. These additional experiments were excluded because the scope of this objective was of a more limited nature: to assess the effectiveness of the regularization methods present in the selected studies. As such, we rather focused on comparing convolutional dropout with L2 kernel regularization, as well as their combined effect. In conclusion, we were successful in assessing the

effectiveness of various regularization interventions and in selecting a method that provides adequate support in combating overfitting.

8.2.3 Assessing recognition performance

Objective

Classifier performance is assessed and compared with regards to precision, recall, F1-score and MPCA.

How was this objective addressed?

The architectures that were found to be overfitting in Experiment 1 were replaced with a regularized version from Experiment 2. The performance of the various architectures have been assessed on the same dataset, which allows for a comparison of the performance of the various architectures. This was done with multiple splits of the dataset. Ten models of each architecture was trained. The results associated with the various architectures are then compared. The top five architectures in this regard are VGG L2, ConvNet8, ConvXpress, FIRST Class and Toothless L2. MCRGNet L2 deserves to be higlighted since it comes quite close to Toothless L2 in terms of classification performance and has a slightly higher MPCA and utilizes few parameters.

Has the objective been met?

As stated above, the various architectures have been trained on ten splits to get the average performance of each architecture. We believe that ten runs are adequate to establish how the architecture would perform in general and although more could be useful, we had to limit ourselves to ten runs due to computational constraints. Furthermore, testing various other hyperparameter setups (batch size, learning rate, etc.) can possibly lead to better results but again to computational constraints we were unable to explore this. Using similar datasets from other domains for comparison would be a good way to validate the results, however since this thesis is focused on radio galaxy classification, this has been left for future work. In conclusion, the average performance of each architecture has been established.

8.2.4 Reporting computational requirements

Objective

Besides recognition performance, it is important to assess the computational requirements of various CNNs when considering which to use in practice. We assess

and report the computational requirements of each architecture in literature in this thesis.

How was this objective addressed?

The computational requirements of the various architectures have been assessed. We measure the GPU memory usage, floating point operations and classification rate of each architecture. Number of parameters are also assessed. The top five architectures in this regard are MCRGNet L2, Radio Galaxy Zoo, Hosenie, FR-Deep and ATLAS L2.

Has the objective been met?

Another computational metric that would be useful to consider would be average training time across various commonly used hardware setups and with varied batch sizes, since these metrics would be quite useful to select architectures that would have to be retrained often or in use cases that have tight computational constraints that has to be adhered to. Since the computational resources used for our experiments was shared, we were unable to establish a useful baseline for training time. In conclusion, we successfully report on the computational requirements of each architecture.

8.2.5 Ranking system

Objective

A ranking system has been developed to assess and compare CNNs that take into account the recognition performance and computational requirements of each architecture.

How was this objective addressed?

The assessments of recognition performance and computational requirements provided several metrics to compare the various architectures. However, a useful way to convey this information concisely is with a ranking system. The architectures that have both a positive classifier ranking and a positive computational rank is MCRGNet L2 (6th class. rank, 1st comp. rank), ATLAS L2 (7th class. rank, 5th comp. rank) and ConvXpress (3rd class. rank, 6th comp. rank). These are useful architectures to consider when computational resources are limited or a high classification speed is required. We also note that the architecture ranked the highest in the classification rank is the lowest in the computational rank (VGG16D L2).

Additional computational complexity in an architecture can aid recognition performance (see Figure 7.8) however this is not a guaranteed strategy nor a scalable strategy. This can possibly lead to overfitting unless the architecture is regularized. We discuss this further in Section 8.3. In comparison to MCRGNet L2, VGG16D L2 requires significantly more computational resources to perform better (and only marginally so in some cases, Figures 7.9 and 7.10). VGG16D L2 has a 3% higher MPCA than MCRGNet L2 but requires 3239 times more FLOPs and roughly 64 times more GPU memory. MCRGNet L2 is also 4 times faster at classifying images than VGG16D L2. Arguably, MCRGNet L2, ATLAS L2 or ConvXpress would be preferable architectures to select based on this information.

Has the objective been met?

A ranking system was developed that compares classifiers both in terms of computational requirements and classification capabilities. The classification ranking system does, however, only give a highlevel indication of an architecture's recognition capability due to the fact that any specific information about an architectures' performance in any given class is lost. Similarly, the computational ranking provides an aggregate metric and any specifics therefore are extirpated.

In conclusion, a ranking system has been created that reports on both computational and classification capabilities, while illustrating the trade-offs one would make when choosing an architecture.

8.3 General Conclusions

There are also a few other minor conclusions that can be drawn from the results obtained from the experiments we conducted in this thesis:

Dataset The MURG dataset has only been labelled by a single person, a dataset labelled by more subject matter experts would arguably be preferable. However, for the purposes of comparing architecture performance as well as identifying overfitting, we find that the MURG dataset was adequate since the alternative datasets were too small and noisy.

Why do the architectures overfit? VGG16D, FIRST Class and Toothless are likely overfitting because of the large number of parameters they have (201 million, 51 million and 71 million respectively). The original Toothless architecture shows less signs of overfitting than the other two mentioned before, likely due to the inclusion of batch normalization between the convolutional layers. The other architectures with high parameter counts, ConvNet4 and

ConvNet8, both have dropout between their convolutional layers as well as and L2 kernel regularization to counter overfitting.

MCRGNet and ATLAS show less signs than VGG16D and FIRST Class of overfitting and this is expected when looking at their parameter count. MCRGNet has the second least parameters at 213 thousand and ATLAS has 1.3 million parameters, both architectures have full dropout between their convolutional layers. The reason why these two smaller models overfit and others with similar capacity and regularization levels do not still eludes us at present. This phenomenon will be investigated as part of a future endeavor. The answer might lie in our metric in identifiying overfitting, i.e. the accuracy and loss ratio distance metric. Our selection of the five architectures with the largest distance does not guarantee that these architectures are overfitting, only that they present more signs of overfitting relative to the other architectures. Other small architectures with smaller distance metrics also show signs of underfitting as discussed earlier, which the accuracy and loss ratio distance metric does not highlight. Both of these factors are severe limitations when using a relative metric rather than an quantitative metric. However, what is clear is that L2 kernel regularization reduced the tendency to overfit in all of the above architectures. In that sense the metric is a useful tool to identify and reduce overfitting.

Intervention effectiveness Convolutional dropout and L2 kernel regularization overregularizes most architectures, convolutional dropout does not provide as much regularization for most architectures as L2 kernel regularization. L2 kernel regularization provides a good trade-off of providing regularization without significantly affecting training time.

Architecture A few design choices for CNN architectures can speed up model performance while driving down resource costs. A larger kernel's receptive field is equivalent to several smaller receptive fields when these are stacked without a pooling layer in between with the added bonus that more layers of non-linearity are added through more ReLU activation functions while driving down the number of parameters. This was originally used within the VGG architecture family developed by Simonyan and Zisserman (2015). Several of the architectures in Table 5.1 build on these design decisions, specifically ConvNet-8 (Lukic *et al.*, 2019*a*).

ConvXpress ConvXpress utilizes the aforementioned stacking strategy. It also uses a non-standard stride length which indirectly reduces its computational cost. This architecture performs well when compared to the other architectures in Table 5.1 (see Table 7.6).

Parameters Increasing your recognition performance by simply utilizing more and more trainable parameters is discouraged as it is a strategy that can lead to overfitting (increasing trainable parameters also does not scale well computationally). Furthermore, an increase in trainable parameters will increase computational complexity, training time and GPU memory usage. Overall Deep Learning models are viewed as inefficient in exploiting their full learning power (Muhammed *et al.*, 2017) given the large number of parameters they require relative to other machine learning approaches.

FLOPs Computational complexity (given as FLOPs) and recognition performance (approximated as MPCA) are weakly correlated (see Figure 7.8). Utilizing more computational resources is, therefore, likely to result in at least marginal increases in recognition performance. As we have hinted at previously, increasing your recognition performance by simply utilizing more and more computational resources is frowned upon as it is a strategy that does not scale well.

Classification Speed Generally, models with a higher MPCA, classify slower than those with a lower MPCA (see Figure 7.14). The trade-off between classification speed and recognition performance is evident as model MPCA decreases with an increase in images classified per second.

Ranking As alluded to earlier, CNNs can be ranked according to their recognition performance results and the computational resources that they require. The ranking we obtained doing just this is presented in Table 7.6. It is, however, important to realize that this study is not exhaustive enough to provide us with an absolute ranking of the architectures in Table 5.1. A more extensive study that considers all possible combinations of hyperparameters would be required for us to achieve the aforementioned goal. Such a study would, however, be computationally infeasible. This study does, however, provide us with a useful pragmatic ranking as the hyperparameters were chosen in accordance with excepted guidelines.

8.4 Influence on automatic classification of radio galaxies

As we have mentioned in the introduction to this work, the larger problem is that a method of automatic classification is needed that can classify large datasets reliably and fast. Our attempt is largely focused on creating a framework that provides the environment to compare architectures with regards to classification reliability

when trained on various different data splits and the speed at which this classification can be performed. We also contribute a novel architecture that provides a good trade-off between computational performance and classifier performance, i.e. it classifies images both fast and does so reliably for different models of the architecture. Arguably, newer architectures can do this even faster and more reliably. Architectures designed for other tasks, such as object detection or semantic segmentation might be better suited for the task at hand, given the preprocessing required to first locate and identify a radio galaxy before morphological classification can even begin. For an in-depth comparison of the more recent architectures that are being used in radio astronomy for image recognition and classification, please refer to the list of excluded architectures in Section 5.5. In conclusion, we do not adress the problem of providing an end to end calsification pipeline in this study. Our study can however be used to help realize this end goal. Which, according, to our literature review is still an ongoing endevour.

8.5 Final Remarks

Training CNNs for the purpose of image classification and specifically radio galaxy classification, has become a relatively easy task to set up given the computational resources available at present. But as Jitendra Malik, Arthur J. Chick Professor of Electrical Engineering and Computer Sciences at the University of California Berkeley and one of the seminal figures in computer vision, stated "There are many problems in [computer] vision where getting 50 percent of the solution you can get in one minute, getting to 90 percent can take you a day, getting to 99 percent may take you five years and 99.99 percent, maybe not happen in your lifetime" (Fridman and Malik, 2020).

The lack of large sets of annotated training data remains one of the greatest challenges in assessing and improving the general recognition performance of CNNs when they are employed for the morphological classification of radio galaxies (and all image classification algorithms). In addition, determining the computational resources a model requires is important to consider when selecting an architecture for deployment. The framework and experiments laid out in this study will hopefully be able to help aid the future of image recognition development within the field of radio astronomy.

List of References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
Available at: https://www.tensorflow.org/

Alger, M. (2016 12). *Learning to Identify Extragalactic Radio Sources*. Master's thesis, Australian National University, Canberra ACT 0200, Australia.

Alger, M.J., Banfield, J.K., Ong, C.S., Rudnick, L., Wong, O.I., Wolf, C., Andernach, H., Norris, R.P. and Shabala, S.S. (2018 August). Radio Galaxy Zoo: machine learning for radio source host galaxy cross-identification. *Monthly Notices of the Royal Astronomical Society*, vol. 478, no. 4, pp. 5547–5563. 1805.05540.

Alhassan, W., Taylor, A.R. and Vaccari, M. (2018 October). The FIRST Classifier: compact and extended radio galaxy classification using deep Convolutional Neural Networks. *Monthly Notices of the Royal Astronomical Society*, vol. 480, no. 2, pp. 2085–2093. 1807.10380.

Aniyan, A.K. and Thorat, K. (2017 June). Classifying Radio Galaxies with the Convolutional Neural Network. *The Astrophysical Journals*, vol. 230, no. 2, p. 20. 1705.03413.

Arslan, E.A. (2020). Radio galaxy morphology classification with mask R-CNN. In: *Proceedings of the 2020 4th International Conference on Vision, Image and Signal Processing*, ICVISP 2020, pp. 36–41. Association for Computing Machinery, New York, NY, USA. ISBN 9781450389532.
Available at: https://doi.org/10.1145/3448823.3448881

Baldi, R.D., Capetti, A. and Massaro, F. (2018 January). FR0CAT: a FIRST catalog of FR 0 radio galaxies. *Astronomy & Astrophysics*, vol. 609, p. A1. 1709.00015.

Becker, B. and Grobler, T. (2019). Classification of Fanaroff-Riley radio galaxies using conventional machine learning techniques. In: *Proceedings of the International Multidisciplinary Information Technology and Engineering Conference (IMITEC) 2019*, pp. 1–8. IEEE.

Becker, B., Vaccari, M., Prescott, M. and Grobler, T. (2021). CNN architecture comparison for radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, vol. 503, no. 2, pp. 1828–1846.

Becker, R.H., White, R.L. and Helfand, D.J. (1995). The first survey: faint images of the radio sky at twenty centimeters. *The Astrophysical Journal*, vol. 450, p. 559.

Best, P.N. and Heckman, T.M. (2012 April). On the fundamental dichotomy in the local radio - AGN population: accretion, evolution and host galaxy properties. *Monthly Notices of the Royal Astronomical Society*, vol. 421, no. 2, pp. 1569–1582. 1201.2397.

Bianco, S., Cadène, R., Celona, L. and Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE Access*, vol. 6, pp. 64270–64277.
Available at: https://doi.org/10.1109/ACCESS.2018.2877890

Bourke, T., Braun, R., Bonaldi, A., Garcia-Miro, C., Keane, E., Wagg, J. and SKAO Science Team (2018 January). Expected Science Performance of the Square Kilometre Array Phase 1 (SKA1). In: *American Astronomical Society Meeting Abstracts #231*, vol. 231 of *American Astronomical Society Meeting Abstracts*, p. 152.07.

Bowles, M., Scaife, A.M.M., Porter, F., Tang, H. and Bastien, D.J. (2020 12). Attention-gating for improved radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, vol. 501, no. 3, pp. 4579–4595. ISSN 0035-8711. https://academic.oup.com/mnras/article-pdf/501/3/4579/35920139/staa3946.pdf.
Available at: https://doi.org/10.1093/mnras/staa3946

Braun, R., Bourke, T., Green, J.A., Keane, E. and Wagg, J. (2015 April). Advancing Astrophysics with the Square Kilometre Array. In: *Advancing Astrophysics with the Square Kilometre Array (AASKA14)*, p. 174.

Capetti, A., Massaro, F. and Baldi, R.D. (2017 February*a*). FRICAT: A FIRST catalog of FR I radio galaxies. *Astronomy & Astrophysics*, vol. 598, p. 49. 1610.09376.

Capetti, A., Massaro, F. and Baldi, R.D. (2017 May*b*). FRIICAT: A FIRST catalog of FR II radio galaxies. *Astronomy & Astrophysics*, vol. 601, p. 81. 1703.03427.

Cheung, C.C. (2007 May). FIRST "Winged" and X-Shaped Radio Source Candidates. *The Astronomical Journal*, vol. 133, no. 5, pp. 2097–2121. astro-ph/0701278.

Chollet, F. *et al.* (2015). Keras. https://keras.io.

Cireşan, D.C., Meier, U., Gambardella, L.M. and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Comp.*, vol. 22, no. 12, pp. 3207–3220.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314.

Dahl, G.E., Sainath, T.N. and Hinton, G.E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In: *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8609–8613. IEEE.

Davis, M. (2002 06). The universal computer: The road from leibniz to turing. *The American Mathematical Monthly*, vol. 109.

de Vaucouleurs, G. (1959 January). Classification and Morphology of External Galaxies. *Handbuch der Physik*, vol. 53, p. 275.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee.

Dieleman, S., Willett, K.W. and Dambre, J. (2015 June). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, vol. 450, no. 2, pp. 1441–1459. 1503.07077.

Elmegreen, D.M. and Elmegreen, B.G. (1987 March). Arm Classifications for Spiral Galaxies. *The Astrophysical Journal*, vol. 314, p. 3.

Fanaroff, B.L. and Riley, J.M. (1974 May). The morphology of extragalactic radio sources of high and low luminosity. *Monthly Notices of the Royal Astronomical Society*, vol. 167, p. 31P.

Fielding, E., Nyirenda, C.N. and Vaccari, M. (2021). A comparison of deep learning architectures for optical galaxy morphology classification. *arXiv preprint arXiv:2111.04353*.

Fridman, L. and Malik, J. (2020 Jun). 110 - jitendra malik: Computer vision.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, vol. 36, no. 4, pp. 193–202.

Geman, S., Bienenstock, E. and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, vol. 4, no. 1, pp. 1–58.

Gendre, M.A., Best, P.N. and Wall, J.V. (2010 June). The Combined NVSS-FIRST Galaxies (CoNFIG) sample - II. Comparison of space densities in the Fanaroff-Riley dichotomy. *Monthly Notices of the Royal Astronomical Society*, vol. 404, no. 4, pp. 1719–1732. 1001.4514.

Gendre, M.A. and Wall, J.V. (2008 October). The Combined NVSS-FIRST Galaxies (CoNFIG) sample - I. Sample definition, classification and evolution. *Monthly Notices of the Royal Astronomical Society*, vol. 390, no. 2, pp. 819–828. 0808.0165.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media.

Gheller, C., Vazza, F. and Bonafede, A. (2018 Aug). Deep learning based detection of cosmological diffuse radio sources. *Monthly Notices of the Royal Astronomical Society*, vol. 480, no. 3, pp. 3749–3761. ISSN 1365-2966.
Available at: http://dx.doi.org/10.1093/mnras/sty2102

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning.* MIT Press. http://www.deeplearningbook.org.

He, K., Gkioxari, G., Dollár, P. and Girshick, R.B. (2017). Mask R-CNN. *CoRR*, vol. abs/1703.06870. 1703.06870.
Available at: http://arxiv.org/abs/1703.06870

Hinton, G., Srivastava, N. and Swersky, K. (2012*a*). Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*, p. 13.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R. (2012*b*). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.*

Hocking, A., Geach, J.E., Davey, N. and Sun, Y. (2015). Teaching a machine to see: unsupervised image segmentation and categorisation using growing neural gas and hierarchical clustering. *arXiv preprint arXiv:1507.01589.*

Hosenie, Z.B. (2018). *Source classification in deep radio surveys using machine learning techniques.* Master's thesis, North-West Univ., Potchefstroom.

Hubble, E.P. (1926 December). Extragalactic nebulae. *The Astrophysical Journal*, vol. 64, pp. 321–369.

Hubel, D.H. and Wiesel, T.N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The J. of Phys.*, vol. 195, no. 1, pp. 215–243.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning*, pp. 448–456. PMLR.

Johnston, S., Taylor, R., Bailes, M., Bartel, N., Baugh, C., Bietenholz, M., Blake, C., Braun, R., Brown, J., Chatterjee, S., Darling, J., Deller, A., Dodson, R., Edwards, P., Ekers, R., Ellingsen, S., Feain, I., Gaensler, B., Haverkorn, M., Hobbs, G., Hopkins,

A., Jackson, C., James, C., Joncas, G., Kaspi, V., Kilborn, V., Koribalski, B., Kothes, R., Landecker, T., Lenc, E., Lovell, J., Macquart, J.P., Manchester, R., Matthews, D., McClure-Griffiths, N., Norris, R., Pen, U.L., Phillips, C., Power, C., Protheroe, R., Sadler, E., Schmidt, B., Stairs, I., Staveley-Smith, L., Stil, J., Tingay, S., Tzioumis, A., Walker, M., Wall, J. and Wolleben, M. (2008 December). Science with ASKAP. The Australian square-kilometre-array pathfinder. *Exp. Astron.*, vol. 22, no. 3, pp. 151–273. 0810.5187.

Jonas, J. and MeerKAT Team (2016 January). The MeerKAT Radio Telescope. In: *MeerKAT Science: On the Pathway to the SKA*, p. 1.

Kharb, P., Stanley, E., Lister, M., Marshall, H., O'Dea, C. and Baum, S. (2015 March). Understanding jets from sources straddling the Fanaroff-Riley divide. In: Massaro, F., Cheung, C.C., Lopez, E. and Siemiginowska, A. (eds.), *Extragalactic Jets from Every Angle*, vol. 313, pp. 211–218. 1411.1534.

Kingma, D.P. and Ba, J. (2017). Adam: A method for stochastic optimization. 1412.6980.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, vol. 25, pp. 1097–1105.

Lacy, M., Baum, S.A., Chandler, C.J., Chatterjee, S., Clarke, T.E., Deustua, S., English, J., Farnes, J., Gaensler, B.M., Gugliucci, N., Hallinan, G., Kent, B.R., Kimball, A., Law, C.J., Lazio, T.J.W., Marvil, J., Mao, S.A., Medlin, D., Mooley, K., Murphy, E.J., Myers, S., Osten, R., Richards, G.T., Rosolowsky, E., Rudnick, L., Schinzel, F., Sivakoff, G.R., Sjouwerman, L.O., Taylor, R., White, R.L., Wrobel, J., Andernach, H., Beasley, A.J., Berger, E., Bhatnager, S., Birkinshaw, M., Bower, G.C., Brandt, W.N., Brown, S., Burke-Spolaor, S., Butler, B.J., Comerford, J., Demorest, P.B., Fu, H., Giacintucci, S., Golap, K., Güth, T., Hales, C.A., Hiriart, R., Hodge, J., Horesh, A., Ivezić, Ž., Jarvis, M.J., Kamble, A., Kassim, N., Liu, X., Loinard, L., Lyons, D.K., Masters, J., Mezcua, M., Moellenbrock, G.A., Mroczkowski, T., Nyland, K., O'Dea, C.P., O'Sullivan, S.P., Peters, W.M., Radford, K., Rao, U., Robnett, J., Salcido, J., Shen, Y., Sobotka, A., Witz, S., Vaccari, M., van Weeren, R.J., Vargas, A., Williams, P.K.G. and Yoon, I. (2020 March). The Karl G. Jansky Very Large Array Sky Survey (VLASS). Science Case and Survey Design. *Publications of the Astronomical Society of the Pacific*, vol. 132, no. 1009, p. 035001. 1907.01981.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comp.*, vol. 1, no. 4, pp. 541–551.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324.

Lettvin, J.Y., Maturana, H.R., McCulloch, W.S. and Pitts, W.H. (1959). What the frog's eye tells the frog's brain. *Proceedings of the IRE*, vol. 47, no. 11, pp. 1940–1951.

Li, X., Chen, S., Hu, X. and Yang, J. (2019). Understanding the disharmony between dropout and batch normalization by variance shift. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2682–2690.

Lukic, V., Brüggen, M., Banfield, J.K., Wong, O.I., Rudnick, L., Norris, R.P. and Simmons, B. (2018 May). Radio Galaxy Zoo: compact and extended radio source classification with deep learning. *Monthly Notices of the Royal Astronomical Society*, vol. 476, no. 1, pp. 246–260. 1801.04861.

Lukic, V., Brüggen, M., Mingo, B., Croston, J.H., Kasieczka, G. and Best, P.N. (2019 8*a*). Morphological classification of radio galaxies: capsule networks versus convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, vol. 487, no. 2, pp. 1729–1744. 1905.03274.

Lukic, V., de Gasperin, F. and Brüggen, M. (2019 12*b*). ConvoSource: Radio-Astronomical Source-Finding with Convolutional Neural Networks. *Galaxies*, vol. 8, no. 1, p. 3. 1910.03631.

Luo, W., Li, Y., Urtasun, R. and Zemel, R. (2017). Understanding the effective receptive field in deep convolutional neural networks. *arXiv preprint arXiv:1701.04128*.

Ma, Z., Xu, H., Zhu, J., Hu, D., Li, W., Shan, C., Zhu, Z., Gu, L., Li, J., Liu, C. and et al. (2019 Feb). A machine learning based morphological classification of 14,245 radio agns selected from the best-heckman sample. *The Astrophysical Journals*, vol. 240, no. 2, p. 34. ISSN 1538-4365.

Ma, Z., Zhu, J., Li, W. and Xu, H. (2018). Radio galaxy morphology generation using DNN autoencoder and gaussian mixture models. In: *2018 14th IEEE International Conference on Signal Processing (ICSP)*, pp. 522–526. IEEE.

Marcos, D., Volpi, M. and Tuia, D. (2016). Learning rotation invariant convolutional filters for texture classification. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2012–2017. IEEE.

Markoff, J. (2012 Nov). Seeking a better way to find web images.
Available at: https://www.nytimes.com/2012/11/20/science/for-web-images-creating-new-technology-to-seek-and-find.html

Maslej-Krešňáková, V., El Bouchefry, K. and Butka, P. (2021 05). Morphological classification of compact and extended radio galaxies using convolutional neural networks and data augmentation techniques. *Monthly Notices of the Royal Astronomical Society*, vol. 505, no. 1, pp. 1464–1475. ISSN 0035-8711. https://academic.oup.com/mnras/article-pdf/505/1/1464/38444912/stab1400.pdf.
Available at: https://doi.org/10.1093/mnras/stab1400

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133.

McGlynn, T., Scollick, K. and White, N. (1998). Skyview: The multi-wavelength sky on the interne. In: McLean, B.J., A, G.D., Hayes, J.J. and Payne, H.E. (eds.), *New Horizons from Multi-Wavelength Sky Surveys*, vol. 179 of *IAU Symp.*, p. 465. Cambridge University Press, Kluwer, Dordrecht.

Minsky, M. and Papert, S. (1969). An introduction to computational geometry. *Cambridge tiass., HIT*.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill Burr Ridge.

Mohan, N. and Rafferty, D. (2015). Pybdsm: Python blob detection and source measurement. *Astrophysics Source Code Library*.

Muhammed, M.A.E., Ahmed, A.A. and Khalid, T.A. (2017). Benchmark analysis of popular imagenet classification deep cnn architectures. In: *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pp. 902–907.

Mukherjee, S., Aviv, R. and Groopman, J. (2017 Mar). A.i. versus m.d.
Available at: https://www.newyorker.com/magazine/2017/04/03/ai-versus-md

Ng, A.Y. (2004). Feature selection, vs regularization, and rotational invariance. In: *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, p. 78. Association for Computing Machinery, New York, NY, USA. ISBN 1581138385.
Available at: https://doi.org/10.1145/1015330.1015435

Norris, R.P., Afonso, J., Bacon, D., Beck, R., Bell, M., Beswick, R.J., Best, P., Bhatnagar, S., Bonafede, A., Brunetti, G., Budavári, T., Cassano, R., Condon, J.J., Cress, C., Dabbech, A., Feain, I., Fender, R., Ferrari, C., Gaensler, B.M., Giovannini, G., Haverkorn, M., Heald, G., Van der Heyden, K., Hopkins, A.M., Jarvis, M., Johnston-Hollitt, M., Kothes, R., Van Langevelde, H., Lazio, J., Mao, M.Y., Martínez-Sansigre, A., Mary, D., Mcalpine, K., Middelberg, E., Murphy, E., Padovani, P., Paragi, Z., Prandoni, I., Raccanelli, A., Rigby, E., Roseboom, I.G., Röttgering, H., Sabater, J., Salvato, M., Scaife, A.M.M., Schilizzi, R., Seymour, N., Smith, D.J.B., Umana, G., Zhao, G.B. and Zinn, P.-C. (2013 March). Radio Continuum Surveys with Square Kilometre Array Pathfinders. *Publications of the Astronomical Society of Australia*, vol. 30, p. e020. 1210.7521.

Norris, R.P., Hopkins, A.M., Afonso, J., Brown, S., Condon, J.J., Dunne, L., Feain, I., Hollow, R., Jarvis, M., Johnston-Hollitt, M., Lenc, E., Middelberg, E., Padovani, P., Prandoni, I., Rudnick, L., Seymour, N., Umana, G., Andernach, H., Alexander, D.M., Appleton, P.N., Bacon, D., Banfield, J., Becker, W., Brown, M.J.I., Ciliegi, P., Jackson, C., Eales, S., Edge, A.C., Gaensler, B.M., Giovannini, G., Hales, C.A.,

Hancock, P., Huynh, M.T., Ibar, E., Ivison, R.J., Kennicutt, R., Kimball, A.E., Koekemoer, A.M., Koribalski, B.S., López-Sánchez, Á.R., Mao, M.Y., Murphy, T., Messias, H., Pimbblet, K.A., Raccanelli, A., Randall, K.E., Reiprich, T.H., Roseboom, I.G., Röttgering, H., Saikia, D.J., Sharp, R.G., Slee, O.B., Smail, I., Thompson, M.A., Urquhart, J.S., Wall, J.V. and Zhao, G.B. (2011 August). EMU: Evolutionary Map of the Universe. *Publications of the Astronomical Society of Australia*, vol. 28, no. 3, pp. 215–248. 1106.3219.

Ntwaetsile, K. and Geach, J.E. (2021 02). Rapid sorting of radio galaxy morphology using Haralick features. *Monthly Notices of the Royal Astronomical Society*, vol. 502, no. 3, pp. 3417–3425. ISSN 0035-8711. https://academic.oup.com/mnras/article-pdf/502/3/3417/38854272/stab271.pdf.
Available at: https://doi.org/10.1093/mnras/stab271

Polsterer, K.L., Gieseke, F., Igel, C., Doser, B. and Gianniotis, N. (2016). Parallelized rotation and flipping invariant kohonen maps (pink) on gpus.

Proctor, D.D. (2011 June). Morphological Annotations for Groups in the First Database. *The Astrophysical Journals*, vol. 194, no. 2, p. 31. 1104.3896.

Ralph, N.O., Norris, R.P., Fang, G., Park, L.A., Galvin, T.J., Alger, M.J., Andernach, H., Lintott, C., Rudnick, L., Shabala, S. *et al.* (2019). Radio galaxy zoo: Unsupervised clustering of convolutionally auto-encoded radio-astronomical images. *Publications of the Astronomical Society of the Pacific*, vol. 131, no. 1004, p. 108011.

Ren, S., He, K., Girshick, R.B. and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, vol. 28, pp. 91–99.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, vol. 65, no. 6, p. 386.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, vol. abs/1609.04747. 1609.04747.
Available at: http://arxiv.org/abs/1609.04747

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, vol. 323, no. 6088, pp. 533–536.

Sabour, S., Frosst, N. and Hinton, G.E. (2017). Dynamic routing between capsules. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, p. 3856.

Sadr, A.V., Vos, E.E., Bassett, B.A., Hosenie, Z., Oozeer, N. and Lochner, M. (2019 April). DEEPSOURCE: point source detection using deep learning. *Monthly Notices of the Royal Astronomical Society*, vol. 484, no. 2, pp. 2793–2806. 1807.02701.

Samudre, A., George, L.T., Bansal, M. and Wadadekar, Y. (2021 Oct). Data-efficient classification of radio galaxies. *Monthly Notices of the Royal Astronomical Society*. ISSN 1365-2966.
Available at: http://dx.doi.org/10.1093/mnras/stab3144

Sandage, A. (1961). *The Hubble atlas of galaxies*, vol. 618. Carnegie Institution of Washington Washington, DC.

Scaife, A.M. and Porter, F. (2021). Fanaroff–riley classification of radio galaxies using group-equivariant convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, vol. 503, no. 2, pp. 2369–2379.

Scaife, A.M.M. (2020). Big telescope, big data: towards exascale with the square kilometre array. *Phil. Trans. of the R. Soc. A: Math., Phys. and Eng. Sci.*, vol. 378, no. 2166, p. 20190060. https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2019.0060.
Available at: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2019.0060

Schlemper, J., Oktay, O., Chen, L., Matthew, J., Knight, C.L., Kainz, B., Glocker, B. and Rueckert, D. (2018). Attention-gated networks for improving ultrasound scan plane detection. *CoRR*, vol. abs/1804.05338. 1804.05338.
Available at: http://arxiv.org/abs/1804.05338

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014 January). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958. ISSN 1532-4435.

Tang, H., Scaife, A.M.M. and Leahy, J.P. (2019 September). Transfer learning for radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, vol. 488, no. 3, pp. 3358–3375. 1903.11921.

Tang, H., Scaife, A.M.M., Wong, O.I. and Shabala, S.S. (2021). Radio galaxy zoo: Giant radio galaxy classification using multi-domain deep learning. 2112.03564.

Wang, S. and Su, Z. (2020). Metamorphic object insertion for testing object detection systems. In: *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1053–1065.

Weiler, M. and Cesa, G. (2019). General E(2)-Equivariant Steerable CNNs. In: *Conference on Neural Information Processing Systems (NeurIPS)*.

Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

Werbos, P.J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560.

Werbos, P.J. (1994). *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, vol. 1. John Wiley & Sons.

Willett, K.W., Lintott, C.J., Bamford, S.P., Masters, K.L., Simmons, B.D., Casteels, K.R.V., Edmondson, E.M., Fortson, L.F., Kaviraj, S., Keel, W.C., Melvin, T., Nichol, R.C., Raddick, M.J., Schawinski, K., Simpson, R.J., Skibba, R.A., Smith, A.M. and Thomas, D. (2013 November). Galaxy Zoo 2: detailed morphological classifications for 304 122 galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, vol. 435, no. 4, pp. 2835–2860. 1308.3496.

Wu, C., Wong, O.I., Rudnick, L., Shabala, S.S., Alger, M.J., Banfield, J.K., Ong, C.S., White, S.V., Garon, A.F., Norris, R.P., Andernach, H., Tate, J., Lukic, V., Tang, H., Schawinski, K. and Diakogiannis, F.I. (2019 January). Radio Galaxy Zoo: CLARAN - a deep learning classifier for radio morphologies. *Monthly Notices of the Royal Astronomical Society*, vol. 482, no. 1, pp. 1211–1230. 1805.12008.