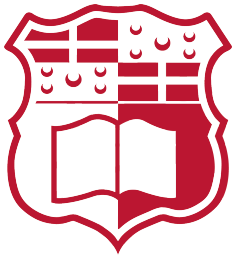

Deep Learning applied to Big Astronomical Data from SKA and its Precursors

Ph.D. Thesis

Daniel Magro

Supervised by Prof. Zarb Adami

Co-supervised by Dr DeMarco

Institute of Space Sciences and Astronomy

University of Malta

October, 2024

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

WWW.UM.EDU.MT

First edition, Monday 14th October, 2024

Acknowledgements

I am deeply grateful to my supervisors, Prof. Zarb Adami and Dr DeMarco, and advisors, Dr Riggi and Dr Sciacca, for their constant guidance, assistance, and encouragement throughout my PhD, without which this thesis surely would not have reached fruition.

I would also like to express my sincere gratitude to the Osservatorio Astrofisico di Catania (OACT) and Istituto Nazionale di Astrofisica (INAF) for their financial support.

Finally, I would like to thank my family for their support throughout this journey, as well as my friends and colleagues. Last but not least, I'd like to thank my cats for keeping me company during my late night writing sessions.

Abstract

The SKA, the completion of which is fast approaching, will be a revolutionary radio telescope, not only in terms of its sophisticated design as a telescope, but especially in terms of the ambitious scientific goals enabled by its observations. It will be made up of two complementary interferometers in different continents, with over 130,000 telescopes in Australia and almost 200 in South Africa, which is expected to produce images with unprecedented resolution, sensitivity, and field of view. This is expected to result in the generation of hundreds of petabytes of processed data annually.

Currently available solutions for processing this data, most of which rely on manual inspection by trained astronomers, would be intractable for the volume of the data expected. This has necessitated the development of automated tools which can handle these tasks, enabling astronomers to develop knowledge from the information extracted by these tools, rather than raw data from telescopes.

This thesis focuses on the application of various state-of-the-art methods to develop automated solutions for two select problems of relevance in astronomical instrument data. The first of these is the classification of gravitational lensing, for which a framework for dealing with classification tasks was developed. This framework contains functionality for loading and pre-processing astronomical data, as well as several CNN-based models for performing the classification task. This tool not only obtains very good results for the dataset used, improving upon the previously reported best performances in other works, but is also very adaptable to similar problems, where an object inside an image needs to be labelled as belonging to one of a number of classes.

The second problem is source finding applied to radio astronomy images, for which an Instance Segmentation solution was developed. This solution, based on Mask R-CNN, is capable of detecting, classifying, and producing a pixel-accurate mask for imaging sidelobes, point sources, and radio galaxies. While the performance achieved on sidelobes was less than desired, that for sources and galaxies was satisfactory. No previously existing solutions perform instance segmentation on astronomical data, so it is difficult to have a fair and direct comparison with other solutions. In fact, the dataset for training and evaluating this model was also collected and labelled as part of this work, and will eventually be published. Again, this tool can be very easily adapted to other astronomical problems of a similar nature.

Publications

1. **D. Magro**, K. Zarb Adami, A. DeMarco, S. Riggi, E. Sciacca, 'A comparative study of convolutional neural networks for the detection of strong gravitational lensing', Monthly Notices of the Royal Astronomical Society, Volume 505, Issue 4, August 2021, Pages 6155–6165, https://doi.org/10.1093/mnras/stab1635.

2. S. Riggi, C. Bordiu, F. Vitello, G. Tudisco, E. Sciacca, **D. Magro**, R. Sortino, C. Pino, M. Molinaro, M. Benedettini, S. Leurini, F. Bufano, M. Raciti, U. Becciani, 'Astronomical source finding services for the CIRASA visual analytic platform', Astronomy and Computing, Volume 37, 2021, 100506, ISSN 2213-1337, https://doi.org/10.1016/j.ascom.2021.100506.

3. S. Riggi, **D. Magro**, R. Sortino, A. De Marco, C. Bordiu, T. Cecconello, A.M. Hopkins, J. Marvil, G. Umana, E. Sciacca, F. Vitello, F. Bufano, A. Ingallinera, G. Fiameni, C. Spampinato, K. Zarb Adami, 'Astronomical source detection in radio continuum maps with deep neural networks', Astronomy and Computing, Volume 42, 2023, 100682, ISSN 2213-1337, https://doi.org/10.1016/j.ascom.2022.100682.

4. S. Riggi, C. Bordiu, **D. Magro**, R. Sortino, C. Pino, E. Sciacca, F. Bufano, T. Cecconello, G. Vizzari, F. Vitello, G. Tudisco, 'Radio source analysis services for the SKA and precursors', *Preprint*, January 2023, https://doi.org/10.48550/arXiv.2301.02804.

5. R. Sortino, **D. Magro**, G. Fiameni, E. Sciacca, S. Riggi, A. DeMarco, C. Spampinato, A. M. Hopkins, F. Bufano, F. Schillirò, C. Bordiu, C. Pino, 'Radio astronomical images object detection and segmentation: A benchmark on deep learning methods', Experimental Astronomy, May 2023. ISSN 1572-9508. https://doi.org/10.1007/s10686-023-09893-w.

6. R. Sortino, **D. Magro**, E. Sciacca, S. Riggi, G. Fiameni, 'Deep Neural Networks for Source Detection in Radio Astronomical Maps', Machine Learning for Astrophysics. ML4Astro 2022. Astrophysics and Space Science Proceedings, vol 60, 2023. Springer, Cham. https://doi.org/10.1007/978-3-031-34167-0_27.

7. R. Sortino, T. Cecconello, A. DeMarco, G. Fiameni, A. Pilzer, **D. Magro**, A. M. Hopkins, S. Riggi, E. Sciacca, A. Ingallinera, C. Bordiu, F. Bufano, C. Spampinato, 'RADiff: Controllable Diffusion Models for Radio Astronomical Maps Gen-

eration', IEEE Transactions on Artificial Intelligence, 2024, https://doi.org/10.1109/TAI.2024.3436538.

Contents

1 Introduction **1**
- 1.1 Problem Definition . . . 1
- 1.2 Motivation . . . 1
- 1.3 Approach . . . 2
- 1.4 Aims and Objectives . . . 2
- 1.5 Thesis Structure . . . 3

2 Background **5**
- 2.1 Radio Astronomy . . . 5
- 2.2 AI, Machine Learning, and Deep Learning . . . 12
  - 2.2.1 Neural Networks . . . 14
  - 2.2.2 Convolutional Neural Networks . . . 24
  - 2.2.3 Quantitative Performance Evaluation of NNs . . . 28
- 2.3 Conclusion . . . 31

3 Literature Review **33**
- 3.1 Classification Algorithms . . . 33
  - 3.1.1 Conventional Methods . . . 33
  - 3.1.2 Machine Learning Classifiers . . . 34
- 3.2 Semantic Segmentation Algorithms . . . 40
  - 3.2.1 U-Net . . . 40
  - 3.2.2 U-Net++ . . . 42
- 3.3 Instance Segmentation Algorithms . . . 43
  - 3.3.1 Review of Existing Methods and their Limitations . . . 44
  - 3.3.2 Mask R-CNN . . . 45

3.3.3 YOLO . . . 50
3.3.4 SOLOv2 . . . 51
3.3.5 Transformers . . . 53
3.4 Conclusions . . . 55

4 Detection of Strong Gravitational Lensing 57
4.1 The Datasets . . . 59
4.2 Image Augmentation . . . 60
4.3 Methodology . . . 62
4.4 Evaluation . . . 64
4.4.1 Metrics . . . 64
4.4.2 Results . . . 65
4.5 Interpreting CNN Features . . . 69
4.6 Conclusions . . . 73
4.6.1 Future Work . . . 74

5 Detection of Compact Sources, Extended Galaxies, and Sidelobes in Radio Astronomical Maps 75
5.1 The Dataset . . . 77
5.1.1 Observational Data . . . 77
5.1.2 Source Labelling Scheme . . . 78
5.1.3 Dataset Preparation . . . 79
5.2 Model Implementation . . . 81
5.2.1 Data Loading . . . 81
5.2.2 Model Architecture and Training . . . 82
5.2.3 Result Post-processing . . . 83
5.2.4 Run Options and Parallel Processing . . . 84
5.2.5 Data Outputs . . . 85
5.3 Evaluation Metrics . . . 86
5.4 Hyperparameter Optimisation . . . 88
5.4.1 Impact of backbone network . . . 88
5.4.2 Impact of weighted losses . . . 89
5.4.3 Impact of image pre-processing . . . 90
5.4.4 Impact of mask loss function . . . 90
5.4.5 Impact of RPN anchor size . . . 90
5.5 Results . . . 91
5.5.1 A Qualitative Look at the Model's Performance . . . 93

5.5.2 Performance against SNR 94
5.5.3 Performance for Different Radio Surveys 96
5.5.4 Comparison with Existing Solutions 97
5.6 Conclusions 99
5.6.1 Future Work 100

6 Conclusions **103**
6.1 Future Work 105

Appendix A Detailed Descriptions of CNNs implemented in LEXACTUM **107**

Appendix B Accepted LEXACTUM Command-Line Arguments and Default Values **111**

Appendix C Default Mask R-CNN Parameters and Accepted Command-Line Arguments **113**

References **119**

Index **135**

List of Figures

2.1 The Earth's atmosphere's electromagnetic opacity 8
2.2 Neuron from a biological brain 15
2.3 Neuron from an NN 16
2.4 Plot of the sigmoid, tanh, and ReLU activation functions 16
2.5 Overall structure (in terms of layers, the neurons within, and connections between them) of a simple NN with a single hidden layer 17

3.1 Graphical representation of the 'CAS Swinburne' model 35
3.2 Graphical representation of the 'lastro_epfl' model 36
3.3 Graphical representation of the two types of 'ResNet blocks' used by the 'CMU DeepLens' model 37
3.4 Graphical representation of the 'CMU DeepLens' model 38
3.5 Graphical representation of the 'WSI-Net' model 39
3.6 Graphical representation of the 'LensFlow' model 39
3.7 Graphical representation of the 'LensFinder' model 40
3.8 Comparison of Classification, Semantic Segmentation, and Instance Segmentation 41
3.9 Graphical representation of the 'U-Net' architecture 42
3.10 High-level graphical representation of the Mask R-CNN architecture 46
3.11 High-level graphical representation of the FPN architecture 48

4.1 Sample lensed and non-lensed images from the Space set 61
4.2 Sample lensed and non-lensed images from the Ground set 61
4.3 Input image used to visualise the features extracted by the CMU DeepLens model 69

4.4 Visualisation of a sample of features extracted by the first convolutional layer of a CMU DeepLens model 70
4.5 Visualisation of the features extracted by the last convolutional layer of the first '3 ResNet block' of a CMU DeepLens model 71
4.6 Visualisation of the features extracted by the last convolutional layer of the remaining '3 ResNet blocks' of a CMU DeepLens model 72

5.1 Sample dataset image at different stages of pre-processing 83
5.2 Sample image (2000×2000 pixels) from the SCORPIO ASKAP survey with sources extracted by ASGARD in parallel mode superimposed 85
5.3 Sample ASGARD image-format outputs, displaying segmentation masks and bounding boxes with classification labels and confidence scores for the detected objects, on three sample images from the dataset 87
5.4 Reliability-Completeness Curves per Class 92
5.5 Sample of instances where the trained model performs as expected, and correctly detects, classifies, and segments objects within the images 93
5.6 Sample of instances where the trained model does not perform as expected, and does not detect certain objects, or misclassifies them 94
5.7 Graph showing the Performance (F1-Score) of the trained model on each class for images at different SNR values 96

List of Tables

3.1 High-level feature comparison of existing source finders, based on deep learning models 50

4.1 The performances achieved for several metrics (Section 4.4.1) by 6 different models trained for a various number of epochs on the Space dataset 66
4.2 The performances achieved for several metrics (Section 4.4.1) by 6 different models trained for a various number of epochs on the Ground dataset 67

5.1 Number of images and objects per class in the produced dataset, results are also shown per data source 80
5.2 The Hyperparameter Combinations attempted in Section 5.4 91
5.3 The Performance achieved by the Hyperparameter Combinations described in Section 5.4 91
5.4 The Reliability, Completeness, F1-Score, and mAP50 for the best performing model 92
5.5 The model's performance on each object class for varying SNR ranges 95

B.1 The command line arguments which are accepted by LEXACTUM, including a short description, and accepted values 111

C.1 The default Mask R-CNN Hyperparameters and the values used for ASGARD113
C.2 The command line arguments which are accepted by ASGARD, including a short description, and accepted values 115

List of Abbreviations

OACT Osservatorio Astrofisico di Catania v
INAF Istituto Nazionale di Astrofisica v
NASA National Aeronautics and Space Administration 5
SKA Square Kilometre Array 1
IR Infrared 6
UV Ultraviolet 6
LIGO Laser Interferometer Gravitational-Wave Observatory 7
RFI Radio-Frequency Interference 9
HPC High Performance Computing 12
CSP Central Signal Processor 12
SDP Science Data Processor 12
SRC SKA Regional Centre 12
OECD Organisation for Economic Cooperation and Development 12
EU European Union 12
AI Artificial Intelligence 12
ML Machine Learning 13
NN Neural Network 2
DL Deep Learning 13
CNN Convolutional Neural Network 2
SVM Support-Vector Machine 34
HOG Histogram of Oriented Gradients 34
ReLU Rectified Linear Unit 14
ELU Exponential Linear Unit 26

SFG Star-Forming Galaxy 44
AGN Active Galactic Nucleus 44
SDC1 SKA Data Challenge I 44
TBD Thresholded Blob Detection 45
LocNet Localization Network 45
RecNet Recognition Network 45
FPN Feature Pyramid Network 46
ResNet Residual Network 21
RPN Region Proposal Network 47
ROI Region of Interest 45
FCN Fully Convolutional Network 46
NMS Non-Maximum Suppression 48
IoU Intersection over Union 31
YOLO You Only Look Once 50
PANet Path Aggregation Network 51
SOLO Segmenting Objects by LOcations 51
NLP Natural Language Processing 53
RNN Recurrent Neural Network 15
LSTM Long Short-Term Memory 53
GRU Gated Recurrent Unit 53
ViT Vision Transformer 54
DETR DEtection TRansformer 54
LEXACTUM Lens EXtrActor CaTania University of Malta 59
CASTLeS CfA-Arizona Space Telescope Lens Survey 58
CLASS Cosmic Lens All-Sky Survey 58
VLA Very Large Array 58
SLACS Sloan Lens ACS 58
ACS Advanced Camera for Surveys 58
HST Hubble Space Telescope 58
SL2S Strong Lensing Legacy Survey 58
CFHTLS Canada–France–Hawaii Telescope Legacy Survey 58
H-ATLAS Herschel Astrophysical Terahertz Large Area Survey 58
LSST Large Synoptic Survey Telescope 58
DES Dark Energy Survey 58

ESA European Space Agency . . . 58
KiDS Kilo-Degree Survey . . . 58
SDSS Sloan Digital Sky Survey
TPR True Positive Rate . . . 28
FPR False Positive Rate . . . 29
ROC Receiver Operating Characteristic . . . 30
AUROC Area Under the ROC . . . 30
AUC Area Under the Curve . . . 30
EMU Evolutionary Map of the Universe . . . 75
ASKAP Australian SKA Pathfinder . . . 75
ESP Early Science Project . . . 77
RMS Root Mean Square . . . 77
ATCA Australia Telescope Compact Array . . . 77
RGZ Radio Galaxy Zoo . . . 45
DR1 first Data Release . . . 45
FIRST Faint Images of the Radio Sky at Twenty cm . . . 45
WISE Wide-field Infrared Survey Explorer . . . 45
ATLAS Australia Telescope Large Area Survey . . . 78
FITS Flexible Image Transport System . . . 79
CAESAR Compact And Extended Source Automated Recognition . . . 80
WCS World Coordinate System . . . 80
DVC Data Version Control . . . 80
GPU Graphics Processing Unit . . . 64
HDF5 Hierarchical Data Format 5 . . . 85
PSF Point Spread Function . . . 9
Mask R-CNN Mask Region-based Convolutional Neural Network . . . 45
ASGARD Automated Source, Galaxy, and Artefact R-CNN Detector . . . 76
SNR Signal-to-Noise Ratio . . . 10
AP Average Precision . . . 31
mAP Mean Average Precision . . . 31
TP True Positives . . . 28
FP False Positives . . . 29
FN False Negatives . . . 29
TN True Negatives . . . 29

EOSC European Open Science Cloud 100
NEANIAS Novel EOSC Services for Emerging Atmosphere, Underwater & Space Challenges 100
CIRASA Collaborative and Integrated platform for Radio Astronomical Source Analysis 100

1

Introduction

1.1 | Problem Definition

The Square Kilometre Array (SKA) telescope will revolutionise radio astronomy, with its unprecedented sensitivity, resolution, and field of view ("∼nJy sensitivity, subarcsec spatial resolution, full frequency coverage from 50 MHz to 15 GHz" (Riggi et al., 2021a)). The SKA comprises two radio interferometers, SKA-Low in Australia with 131,072 telescopes, and SKA-Mid in South Africa with 197 telescopes (Dewdney et al., 2022; SKAO, c). The scale of processed data is expected to reach ∼710 PB per year, this volume of data is, again, unprecedented in the field of astronomy (SKAO, 2023).

The SKA telescope's launch is approaching, however, how this magnitude of data will be processed and converted into information is not yet completely known. This has necessitated the development of automated solutions, as no other manual solution could be tenable when considering the scale of data. The problem lies in that these solutions do not yet exist, or that most that do, are not mature enough to be realistically applied or produce the desired outputs reliably, if at all.

1.2 | Motivation

Astronomy has been studied for as long as humankind has been able to look up at the sky, with documented observations dating back to as early as ∼140 BC. The SKA will enable new discoveries, particularly the better understanding of, to name a few, the evolution of galaxies, dark energy, cosmology, gravity (using pulsars and black holes), cosmic magnetism, the dark ages, cosmic dawn, and epoch of reionization, and the cradle of life (with the search for signs of extraterrestrial life) (Dewdney et al., 2013;

Riggi et al., 2021a; SKAO, b). Radio Astronomy in particular, allows astronomers to view astronomical objects hidden behind others, by their radiation, as well as objects that do not give off visible light. The development of such automated tools will make it possible for astronomers to properly use and study the data, and then information, gathered by the SKA telescope, as well as its precursors.

1.3 | Approach

The general approach of this thesis will be to develop Neural Networks (NNs), particularly Convolutional Neural Networks (CNNs), that are capable of learning from available datasets, and applying the learnt knowledge on data which has not been seen before, in an automated and efficient manner. Tools and solutions will be built such that they initially solve a domain specific problem, however, should then be easily applied to other similar problems where a similar output for a similar input is expected, this will be elaborated upon in Section 1.4.

1.4 | Aims and Objectives

The first aim of this thesis is to develop a Classification model for astronomical objects. The goal is to have a model which can determine to which class an object shown in a given image belongs.

- In this thesis, several models will be implemented with the objective of determining whether a given image contains gravitational lensing, or not.

The second aim is to develop an Instance Segmentation model, which will detect, classify, and draw a segmentation mask, even differentiating between overlapping objects, for any given image - without the need for any additional user input or intervention.

- In this work, a tool will be developed with the objective of performing instance segmentation on images containing imaging sidelobes, point sources, and radio galaxies.

While these objectives mention specific use cases, it should be understood that the developed tools can very easily be trained with different data and labels/classes, to adapt the existing architectures to different problems very easily.

1.5 | Thesis Structure

Chapter 2 starts off by giving background information about Radio Astronomy, and a high level introduction to Artificial Intelligence and Machine Learning.

Chapter 3 goes on to review existing techniques, primarily Machine Learning-based ones, that have been applied, or make sense to apply, to Astronomy data, particularly for Classification and Instance Segmentation problems.

Chapter 4 then introduces Gravitational Lensing, along with the 'Gravitational Lens Finding Challenge 1.0' task and dataset. The framework built, LEXACTUM, is also described, mentioning the techniques used and making reference to the architectures discussed in Section 3.1.2 for the classification of the gravitational lenses. Finally, the performances achieved by LEXACTUM are presented and compared to those in previous works. An Analysis and Interpretation of the Features extracted by Convolutional Layers in one of the architectures is also presented to demonstrate the workings of CNNs.

In Chapter 5, the process of building a dataset from multiple different surveys and the collaborative process of manually labelling the final dataset is discussed. Next, the features of ASGARD, the implementation of Mask R-CNN in this work, are presented. The process of training the model and hyperparameter optimisation is also described. Finally, results and performances obtained are presented, also in the context of metrics such as the SNR of the data.

Finally, Chapter 6 concludes this thesis by highlighting the principal advancements made throughout this work. Furthermore, it also identifies areas for potential improvement and avenues for future research efforts that are believed to further enhance the performances achieved in this thesis.

2

Background

2.1 | Radio Astronomy

Astronomy is "the study of the stars and other celestial objects" (Unsöld and Baschek, 2013). More specifically, Observational Astronomy is the study of the observable, quantifiable, universe, particularly using scientific instruments, such as telescopes.

Optical Astronomy, or Visible-Light Astronomy, is most probably what one thinks of when they hear the word 'astronomy'. It is, in fact, the oldest form of astronomy, existing ever since people have looked up at the sky, with records of astronomical observations dating back as early as ∼140 BC, long before the advent of telescopes (Moore, 2012). To whom the invention of the telescope is attributed is a disputed topic, but it is generally accepted that Galileo Galilei was the person to truly refine the design, around 1609 (Moore, 2012). That being said, optical astronomy should not be thought of as a primitive technology of the past, on the contrary, the National Aeronautics and Space Administration (NASA) launched the James Webb Space Telescope in November 2021 to study other planetary systems and how stars and galaxies form (NASA).

While working as a radio engineer and studying the high-frequency radio waves emitted by lightning strikes during thunderstorms (atmospherics), Jansky (1933) noticed electromagnetic waves which did not match any known sources and could not be connected to the lightning strikes. After recording these signals over a year, the data showed how the direction of the sources changed by ∼360° in ∼24 hours, attributable to the Earth's daily rotation. Moreover, the Earth's rotation around the Sun is reflected in the point in time the waves reach their peak as well as their direction. These observations imply that these waves come from a source fixed in space, from outside the solar system, in this case, the Milky Way (Carroll and Ostlie, 2017; Jansky, 1933; Magro, 2013). Ultimately, this discovery lead to the understanding that the Milky Way, and celestial

objects in general, emit a substantial amount of radio waves, enough for them to also be observed and studied from the radio spectrum, and not just light which suffers from limitations that will be mentioned later in this section.

Maxwell (1890) is generally credited with predicting the existence of, and developing the theoretical framework around, electromagnetic waves, from his equations. The different components that make up the electromagnetic spectrum were then progressively discovered or demonstrated.

Young (1804) demonstrated that visible light is, in fact, an electromagnetic wave with his two-slit experiment, where light shone through a double slit displayed bright and dark fringes, implying the occurrence of constructive and destructive interference, and thus, that light has the properties of a wave.

Herschel (1800) experimented with measuring the temperature of the colour (visible light) spectrum from sunlight which had been passed through a prism, interestingly, when placing a thermometer just beyond the visible light on the red end of the spectrum, the temperature was even higher than that from the colour red, demonstrating the existence of Infrared (IR) radiation.

In 1801, Ritter demonstrated the existence of another region of radiation past the spectrum of visible light, this time beyond the violet end of the spectrum, Ultraviolet (UV). The existence of UV radiation was proved by measuring the speed at which silver chloride decomposed when exposed to different coloured light, finding that the fastest decomposition time was obtained when the substance was placed just beyond violet visible light (Davidson, 2015).

Röntgen (1896) is generally regarded as the first person to meticulously study X-rays, discovering the property of X-rays producing an image on imaging plates, even when passing through certain solid objects, thus realising the potential of X-rays for use in medical imaging (Busch, 2023).

Villard discovered a new type of highly penetrating radiation when studying radium emissions in 1900, with Rutherford naming them Gamma Rays in 1903 (Gerward, 1999).

In 1886, Hertz modified a pair of Knochenhauer spirals into an open resonant system. This was achieved by replacing one spiral with an induction coil that produces sparks and a straight wire, with a spark gap in between. The other spiral was replaced by a square- or circular-shaped (open, almost closed) loop, with a small gap at the ends of the copper wire. These two components correspond to an antenna and receiver setup, with which Hertz demonstrated resonance, as a spark in the first component (antenna) would generate radio waves, which would then be received by the copper loop (receiver), and produce a spark in the small gap, indicating the reception of the radio waves

(Hertz, 1893; Kraus, 1988). Hertz was also able to measure the wavelength of the produced radio waves, and knowing their frequency, could calculate their speed, which matched that of light. This feature, together with other experiments which showed (optical) properties such as reflection, refraction, and diffraction, the same properties present for visible light, placed radio waves on the electromagnetic spectrum (Hertz, 1893; Kraus, 1988).

Hertz is again attributed with laying the groundwork for the demonstration of microwaves (Bryant, 1988), however Bose is regarded as the first to generate microwaves and observe their optical properties, in the 1890s (Yadugiri, 2010).

The Earth's atmosphere reflects (back out), or absorbs, most of the electromagnetic radiation coming from space, thus preventing it from reaching the ground, namely due to ionospheric refraction and absorption by atmospheric molecules such as oxygen and water vapour. This particularly affects gamma rays, X-rays, UV, most IR radiation, and high-wavelength ($\lambda > \sim$30 m, $f < \sim$10 MHz) radio waves. This leaves visible light and radio waves as ideal candidates for observation from Earth (Condon and Ransom, 2016). This is shown graphically in Fig. 2.1.

One immediately obvious disadvantage of optical telescopes on Earth is that they can be easily hampered by cloudy weather. Radio telescopes, on the other hand, listen for radio waves, which have a longer wavelength, and are thus not as disrupted by poor visibility, and have a higher margin of error. The longer wavelength does, however, have its downsides, primarily, that for the same area to be captured, at the same quality, radio telescopes must be much larger. In order to overcome this limitation, radio telescopes can be physically placed in a telescope array, known as an interferometer (Jansen van Rensburg, 2012; SKAO, a).

For completeness' sake, there also exists Gravitational-Wave Astronomy. Gravitational waves were first proposed by Heaviside in 1893, and later by Poincaré in 1905, who drew a (since refuted) analogy to electromagnetic waves. Einstein (1916) is popularly attributed to having developed a more comprehensive understanding and framework for gravitational waves.

Gravitational waves make it possible to observe phenomena that do not emit electromagnetic radiation, and thus cannot be detected by e.g. optical or radio telescopes. Binary systems (two massive objects, e.g. black holes, orbiting each other) and supernovae are two such phenomena. When such massive objects move around, they create 'ripples' (gravitational waves) in spacetime, which propagate through space, and can ultimately be detected.

In 2015, around a hundred years after Einstein (1916) theorised gravitational waves, the Laser Interferometer Gravitational-Wave Observatory (LIGO) was the first to de-

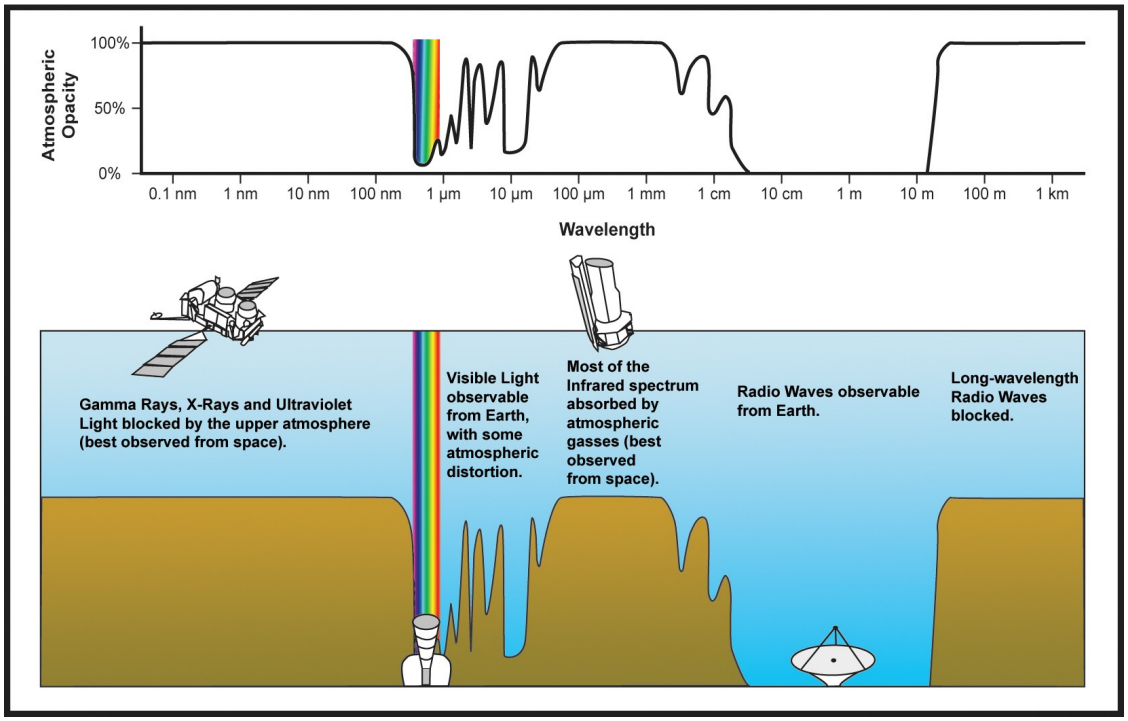

Figure 2.1: The Earth's atmosphere's electromagnetic opacity, as described in Section 2.1. Reproduced from NASA, released into the Public Domain (`https://commons.wikimedia.org/wiki/File:Atmospheric_electromagnetic_opacity.svg`).

tect gravitational waves. From this detection, the LIGO observed the final moments before two black holes merged (Abbott et al., 2016). More recently, in 2017, Abbott et al. observed the collision of two neutrons stars, which resulted in the emission of gravitational waves, followed by electromagnetic waves. This was the first instance of the same event being observed from both its gravitational and electromagnetic emissions, a first in multi-messenger astronomy (La Rana, 2023).

As explained in Thompson et al. (2017), a radio telescope is constantly exposed to radio waves, from astronomical sources in the sky, as well as interference (mainly generated by humans). The telescope's radio receiver monitors the power of the waves received, and converts it into a voltage. This received voltage is typically rather weak, thus it is passed to a Low-Noise Amplifier to amplify the signal without introducing excessive noise. Filters and Mixers may also be applied at this stage.

The boosted analog signal is then converted to digital using an Analog-to-Digital Converter, such that it can be processed by computers. When converted using a sufficiently high sampling rate, the Nyquist Rate, the digital signal is considered to not have lost any information. The Nyquist Theorem (Nyquist, 1928; Shannon, 1949) specifies

that the Nyquist Rate (sampling rate) should be at least double the highest frequency present in the analog signal. Converting to digital comes with advantages such as the practicality of transmission over long baselines (distances) without any distortion while stably maintaining any time delays, which is especially useful in the case of interferometers.

Next, the digital signal is converted from the time domain to the frequency domain by applying a Fourier Transform. This effectively extracts the frequency components making up the received signal. In single radio telescopes, this step is more relevant for spectral analysis, rather than the construction of the image. The Point Spread Function (PSF) of the telescope, which represents the difference between how an object actually looks and how it is detected by the telescope, can blur an imaged point source, typically at the centre. The PSF, also known as the 'dirty beam', will also have secondary peaks due to diffraction, known as sidelobes. In telescopes with a circular dish, the PSF typically resembles an Airy disk (Airy, 1835), i.e. a bright main lobe at the centre, with fainter sidelobes surrounding it. Sidelobes can generate imaging artefacts (false signals) within the image, especially around bright sources, which can be misinterpreted as legitimate sources. Algorithms such as CLEAN (Högbom, 1974) can mitigate this effect by deconvoluting the PSF from the telescope's detection.

When working with a single radio telescope, 'scanning' can be applied by repeating this process to measure other areas of the sky (e.g. following a raster pattern, i.e. left to right, top to bottom) and progressively building a 'map' of intensities, which is essentially an 'image' of the scanned area of the sky (Thompson et al., 2017).

The SKA telescope, currently under construction (as of 2024), consists of two complementary telescopes, SKA-Low ($\sim$50-350 MHz) in Australia and SKA-Mid ($\sim$350-3,050 MHz, even up to 14 GHz, with the ability to go up to 20 GHz) in South Africa. The specific locations of these telescopes were chosen, among other factors, due to their remoteness, with which come lower levels of ground-based Radio-Frequency Interference (RFI) (radio quiet zones). With 131,072 telescopes making up SKA-Low in Australia, and 197 making up SKA-Mid in South Africa, the SKA will be one of the largest telescopes, and the most sensitive. The maximum separation between telescopes in SKA-Low will be 65 km, and 150 km in SKA-Mid (Dewdney et al., 2022; SKAO, c).

This maximum distance, or baseline, between telescopes within an interferometer is key, since the greater the separation between telescopes, the better the angular resolution of the interferometer. This "general scaling relation" is implied by Equation 2.1, where greater values of D result in smaller values of θ, the angular resolution. Smaller values of θ imply better resolutions, since the dimensions of each 'pixel' are smaller, i.e. more pixels fit in the same area (Condon and Ransom, 2016).

$$\theta \approx \frac{\lambda}{D} \tag{2.1}$$

Where θ is the angular resolution of the telescope, λ is the wavelength of the waves being observed, and D, or B, refers to the maximum distance (baseline) between telescopes.

While Equation 2.1 provides a theoretical value for the angular resolution, in practice, there are other contributing factors. One such factor is how each individual baseline's contribution within an interferometer is weighted. The need for weighting baselines within arrays differently arises from arrays using different baselines for different sets of telescopes, e.g. the baselines being shorter at the core, and longer towards the outsides. Maintaining a uniform weighting in such scenarios would lead to degraded sensitivity. Natural weighting is one scheme that can be applied, which gives higher weight to shorter baselines, resulting in an increased sensitivity (the ability to detect fainter details), however penalising the resolution (Thompson et al., 2017).

As further explained in Thompson et al. (2017), interferometers share many of the same concepts described earlier for single radio telescopes. The imaging process similarly starts with the telescopes in the array each detecting radio waves and converting these into a voltage. These voltages are amplified, and converted to digital. Here, the accurate and precise recording of the timing of the received signals is paramount, together with properly synchronised clocks, as this becomes crucial for correlating and recombining the signals.

Since the telescopes in an array are all in different positions, the distance between the same astronomical source and each telescopes will vary, thus the telescopes will receive a given signal at different points in time. This phase offset due to the baselines is called "geometric delay". The signals from the telescopes in the array need to be synchronised, and this is typically accomplished by applying a delay or phase shift.

Next, cross-correlation is performed to ensure that each pair of telescopes (baseline) in the array is picking up the same astronomical object, e.g. by comparing the overlap of signals. If the detected signals are not sufficiently correlated (overlapping), their contribution to the result will be zeroed out when applying the weights at a later stage. This could happen if there was an issue in the aforementioned synchronisation/alignment step, or if, e.g. the object is too faint, or it has such a low Signal-to-Noise Ratio (SNR) that there is too much noise for the detections to be correlated. This step also produces the visibility function, including interference fringes, which are a visual representation of the correlation between the detections of the pair of telescopes in the baseline, where overlaps result in bright fringes (constructive interference), and misalignments

in dark fringes (destructive interference). This data provides further information about the astronomical objects being observed, such as the resulting amplitude indicating the brightness of the object, or the phase indicating the shape and position of the object. For example, if the phase for the same object varies for longer baselines, this indicates an extended morphology.

A Fourier Transform is then applied to each baseline's cross-correlated signal, i.e. the interferometric visibilities. As explained above for single telescopes, this decomposes the time domain signals into their constituent frequency components (the frequency domain). Each baseline's frequency data is then mapped onto the (u, v) plane. The (u, v) plane, conceptually, sits tangentially to the celestial sphere, centred at the astronomical object under observation. This plane essentially represents each baseline's reading (visibilities) on a grid according to their position. At this stage, the aforementioned weighting step is applied, which weights the contribution of each baseline according to some technique, e.g. uniform, natural, or Briggs (Briggs, 1995) weighting, to optimise both the interferometer's resolution and sensitivity.

The van Cittert–Zernike theorem (Born and Wolf, 2013; Zernike, 1938), in the context of interferometry, essentially states that the visibility function mentioned above is the Fourier Transform of the sky (Isella, 2011; Jansen van Rensburg, 2012; Vaporia). This effectively means that objects in the sky can be reconstructed by applying an Inverse Fourier Transform to the visibilities, or more precisely in this case, the weighted (u, v) plane. Thus, an Inverse Fourier Transform is applied, which converts the Frequency Domain data (from all the baselines) on the (u, v) plane into a reconstructed image of the sky's brightness.

Finally, the PSF deconvolution process is performed to minimise imaging artefacts. The process is the same as that described above for single telescopes, only in this case, the PSF will be that of the interferometer as a whole. Naturally, this 'combined PSF', or synthesized beam, will be more intricate than the PSF for a single telescope, which, as mentioned, typically resembles an Airy disk. This produces the interferometer's final image output.

Somewhat similarly to how a single telescope can scan the sky to build an image, interferometers can apply Mosaicking to image regions of the sky larger than a single pointing can achieve (the beam area). This is accomplished by repeating the above process to produce images of different, however adjacent or even overlapping, regions of the sky, with the produced beam-area images then stitched together. The resulting full-field image can then undergo an additional deconvolution step, to remove any artefacts potentially introduced from the stitching process (Thompson et al., 2017).

The sheer volume and velocity of data that will be collected once the SKA telescope

is operational is, by far, unprecedented in the field of astronomy, and introduces both a Big Data and High Performance Computing (HPC) element into the project. The SKAO (2023) estimate that SKA-Low and SKA-Mid will each output data at 7.2 Tb/s and 8.8 Tb/s, respectively. This raw data will then be transmitted to the respective locations' Central Signal Processors (CSPs) where, e.g. the aforementioned correlator process, among others, is carried out. This will produce another ∼5 Tb/s which is then transmitted to the locations' Science Data Processors (SDPs), supercomputers with a combined performance of 250 PFlops. These SDPs are responsible, among other things, for forming the final images of the sky from the CSPs' output data. This is expected to result in ∼710 PB/yr of data to be transmitted to, and stored at, SKA Regional Centres (SRCs), from where the final processed data can be accessed (Riggi et al., 2021a; SKAO, 2023).

After all this computation is done, the output produced is 'simply data', the insurmountable volume of the data makes it impossible for any scientist to extract anything meaningful from all of this data. The only solution to this problem is the automation of the process of converting the data to information, and subsequently knowledge and science, using Artificial Intelligence (AI) (Riggi et al., 2021a).

2.2 | AI, Machine Learning, and Deep Learning

While it is difficult to define AI, as it is difficult to define intelligence itself, Russell and Norvig (2020) refer to two main definitions, that of AI mimicking human intelligence in the sense of "performance", and intelligence in the sense of "rationality". For the context of this thesis, AI can simply be considered to be the former, i.e. the 'digitalisation' of human intelligence to perform a certain task. Particularly, to enable computers to perform tasks which could previously only be carried out by trained astronomers.

The Organisation for Economic Cooperation and Development (OECD) defines AI systems as machine-based systems that, using machine and/or human-based data as input, can make predictions, recommendations, or take decisions to accomplish specified objectives. These systems perceive the given environments, abstract their perceptions into models, and use the models in inference to formulate outcomes (OECD, 2019, 2022).

The Eurostat defines AI as systems that use a variety of techniques, such as "text mining, computer vision, speech recognition, natural language generation, machine learning, deep learning", to collect and/or apply data to "predict, recommend or decide" to achieve a specified objective (Montagnier and Ek, 2021).

In 2021, the European Commission proposed the regulation of AI in the European

Union (EU). Later that year, the EU Council published a first draft of the AI Act, which was a compromise of over three hundred submissions (Future of Life Institute, 2024). In this draft, 'AI system' was defined as software developed to generate outputs (such as "content, predictions, recommendations, or decisions") that relate to the environments they operate in, to accomplish a "human-defined" objective, making use of a number of specified techniques, some of which are Machine Learning (ML) (including Deep Learning (DL)), logic- and knowledge-based, and statistical approaches (European Commission, 2021). In 2024, the EU formally adopted this act (Future of Life Institute, 2024), and this definition has been updated slightly. This updated definition makes reference to AI systems having varying levels of autonomy, with the possibility for systems to adapt even after deployment, and for the input received, generate outputs ("predictions, content, recommendations, or decisions") as per specified objectives, that can affect the system's environment (European Parliament and Council of the European Union, 2024).

Lane and Williams (2023) extract the commonalities between the eight definitions Montagnier and Ek (2021) cite from several countries and Eurostat. Definitions commonly describe AI systems as intelligent, or specify cognitive tasks they are capable of. AI is often referred to as, or given the "form" of, a "system", "technology", "machine", or "software". Some definitions make reference to AI's environment, referring to its interaction with the world, even if only through observation and perception and not action. Definitions also refer to the level of AI's autonomy, suggesting possible limitations, and thus the possible need for human intervention for guidance or overriding (Lane and Williams, 2023).

The applications of AI are countless, robotics, Knowledge Representation and Reasoning, Natural Language Processing, and Computer Vision are a just few examples, with both existing and potential applications across virtually every industry, e.g. manufacturing, finance, medicine, and cybersecurity. The term AI is very broad, and encapsulates a plethora of different methods and techniques such as Genetic Algorithms, Fuzzy Logic, Expert Systems, and ML.

The principal focus of this thesis will be on ML-based methods. ML is the process of a computer observing copious amounts of data, and then building a model which best fits the data it has seen, with which it can make predictions on data which is new to it (Russell and Norvig, 2020). The reason this thesis focuses on ML-based methods is twofold. Firstly, it would be untenable, and likely impossible, to produce, say, an Expert System, capable of working with any data that is thrown at it. Secondly, it would be very hard to encapsulate the decision-making process of a trained astronomer into a series of rules, as they themselves would find it hard to explain how they reached a certain conclusion (Russell and Norvig, 2020). This is not to say that all other techniques are ir-

relevant or obsolete, on the contrary, the performance of ML systems can, potentially, be improved by collaborating the decision-making process with certain hard coded rules from experts.

Three different types of 'feedback' are used with ML algorithms. The first is 'Supervised Learning', where the input data is accompanied by an expected output, in some applications known as the 'label'. In this setup, the algorithm builds its model by first making a prediction for each input, and then adjusting the model based on whether its prediction was correct or not (when compared to the expected output). The second is 'Unsupervised Learning', where the algorithm extracts patterns from the data without any feedback, this is most commonly used with clustering tasks. Lastly, there is 'Reinforcement Learning', where the algorithm decides how to adjust the model based on a 'reward' or a 'punishment' for the result from a series of actions (Russell and Norvig, 2020).

2.2.1 | Neural Networks

NNs are one such ML algorithm. NNs are loosely based on the human brain, where brain neurons, made up of dendrites, somas (cell bodies), and axons (McCulloch and Pitts, 1943; Russell and Norvig, 2020; Woodruff), as shown in Fig. 2.2, are represented as inputs, non-linear activation functions, and connections to other neurons or outputs, as shown in Fig. 2.3, respectively. The 'net' of a node is calculated by multiplying each of the node's inputs by the weight associated to that input's connection, and summing these products. Additionally, a bias is also added, this can be represented as an input that is always set to 1 (e.g. $x_0 = 1$) with an adjustable weight (e.g. w_0). The output is then calculated by inputting the net into an activation function. Some widely used activation functions are the sigmoid (Equation 2.2), tanh (Equation 2.3), and Rectified Linear Unit (ReLU) (Equation 2.4), graphed in Fig. 2.4 (Russell and Norvig, 2020; Wasserman, 1989).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.3}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.4}$$

NNs are generally made up of layers, where each layer is made up of a number of neurons, or nodes. NNs typically consist of an input layer, to which the input is, in some way, connected, and an output layer, from which the output of the model is read. The

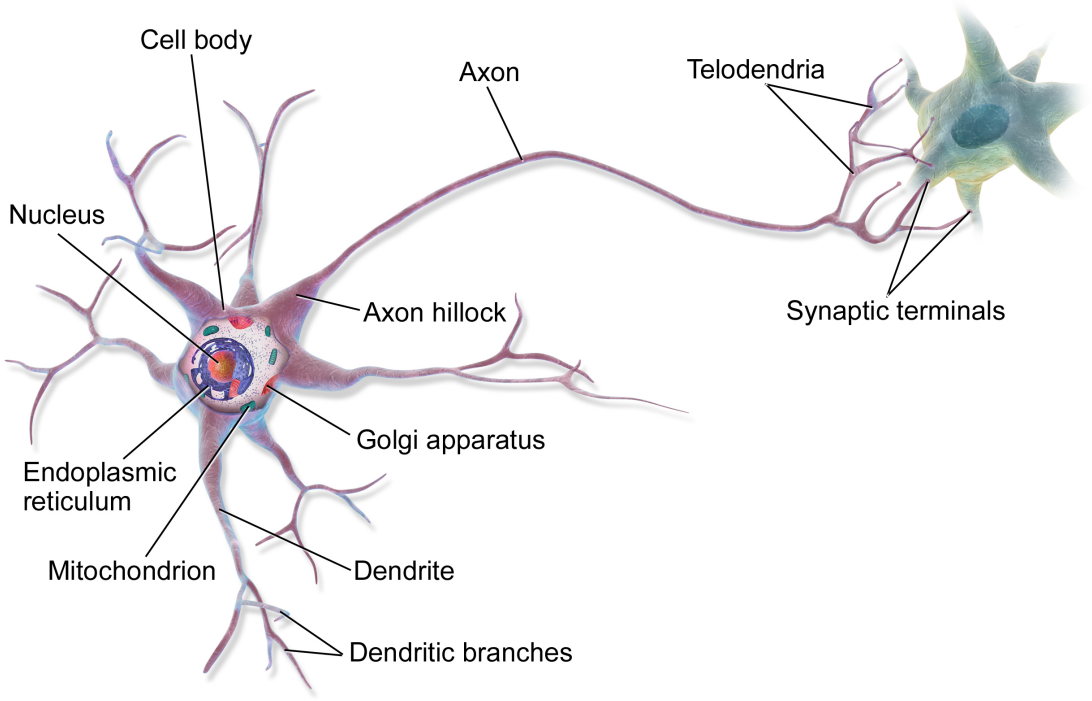

Figure 2.2: Neuron from a biological brain, as described in Section 2.2.1. Reproduced from Blausen.com staff (2014), released under the CC BY 3.0 Licence.

neurons in the input layer are connected to those in the output layer via connections. Most often, each neuron will have multiple connections from/to other neurons, and Fully-Connected Networks will have connections from/to every other neuron in the preceding and subsequent layer, respectively. DL further builds on this methodology by introducing 'hidden layers' and can be attributed to making NNs the most common and popular approach for the majority of applications (Russell and Norvig, 2020). The strength of DL is that, through hidden layers, which are simply additional layers of neurons in between the input and output layer, the NN is given much more representational power, which is effectively a precondition for dealing with real world data. A NN without hidden layers, in fact, has so little representational power, that it cannot represent something as simple as XOR, due to it being linearly inseparable (Wasserman, 1989). This architecture is presented graphically in Fig. 2.5.

In traditional, feed-forward NNs, the flow of data is from the input nodes (neurons), through any hidden layers (if present), to the output nodes (e.g. Recurrent Neural Networks (RNNs) are an exception to this, as certain layers feed their output back into themselves as inputs). During this forward propagation process, the network operates

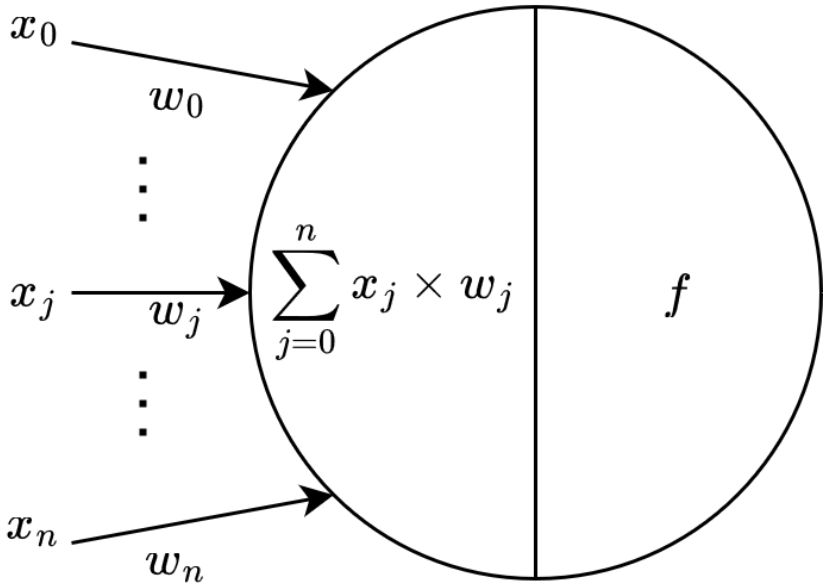

Figure 2.3: Neuron from an NN, as described in Section 2.2.1. Reproduced from CS231N Staff; Wasserman (1989).

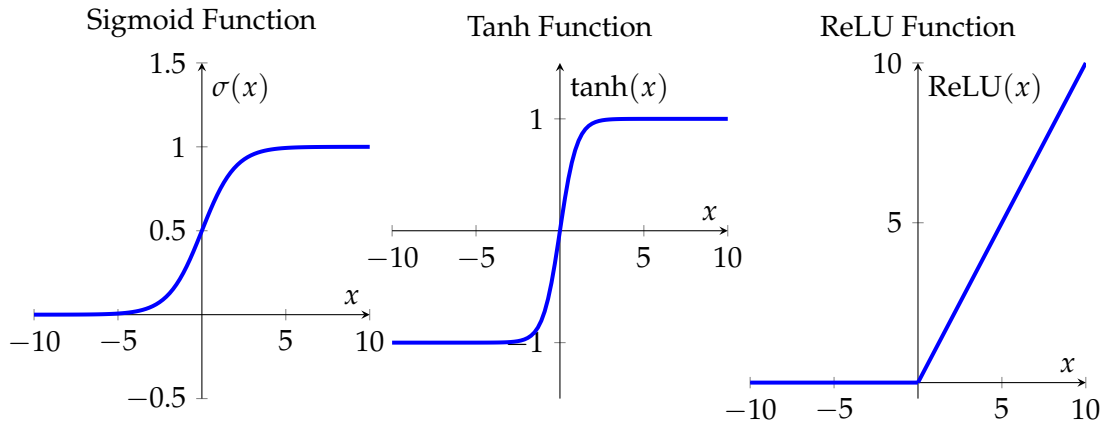

Figure 2.4: Plot of the sigmoid, tanh, and ReLU activation functions described in Section 2.2.1. Reproduced from Russell and Norvig (2020).

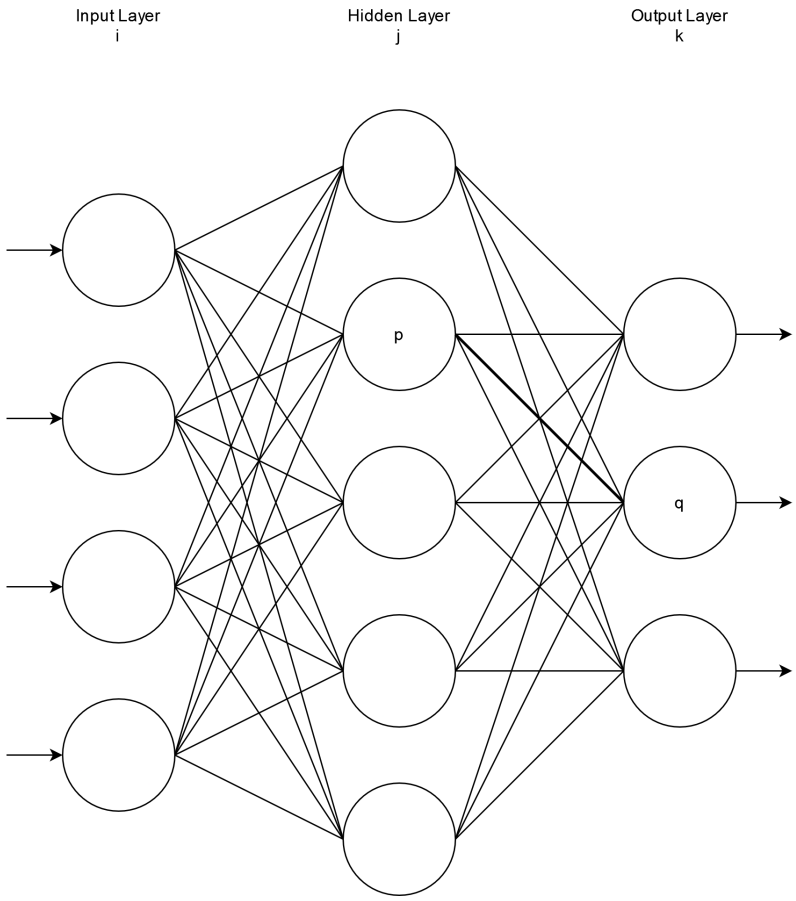

Figure 2.5: The overall structure (in terms of layers, the neurons within, and connections between them) of a simple NN with a single hidden layer, as described in Section 2.2.1. Reproduced from Wasserman (1989).

on a set of given inputs, and every layer is executed one after the other, using the output of the previous layer (or the input if it is the first layer), the weights, and biases to compute the output of every neuron in that layer, such that eventually the output layer will be executed, and an output is produced. Initially, the weights within the network will most likely be initialised randomly, so the network's output will not have any significance. These values are then progressively refined during training, through a process known as back propagation, which adjusts the weights of the connections within the network to produce the desired outputs for given inputs (Russell and Norvig, 2020).

The calculation of the net of the nodes in each layer is typically implemented as a matrix multiplication of an input vector (e.g. 1×5 matrix; where 5 is the number of nodes connected to the current layer) by a weight matrix (e.g. 5×4 matrix; where

4 is the number of nodes in the current layer, thus the number of nets to be output by that layer, and where every element in the matrix represents the weight between a node in the previous layer and a node in the current layer). In this example, the net will be a 1×4 matrix, representing a net value for each node in the current layer. Each value is plugged into an activation function to obtain the layer's output, the net's matrix shape is maintained here (Russell and Norvig, 2020; Wasserman, 1989). The use of these operations leads to the forward propagation process being a rather computationally efficient one.

When a network is being trained, each input in the training set first goes through the aforementioned forward pass so that the network's prediction (with its current set of weights) is computed. This predicted output is compared with the expected output from the training set ground truth, and a loss function is calculated, which represents the 'distance' between the predicted and expected outputs. This distance is quantified and calculated using a loss function, such as the squared loss function (L_2 loss), which is calculated as the square of the difference between the predicted and expected values. A key (although, technically, not absolutely necessary) feature of these functions is that they are differentiable, such that the gradient of the loss can be computed, therefore the weights can be adjusted in such a way that the loss is minimised using a gradient descent approach (Russell and Norvig, 2020).

The final stage of the training process is known as Back Propagation. During back propagation, the calculated loss metric is sent backwards through the network and used to update the weights, such that the network would produce a more desirable output in future runs, outputs which are nearer to the expected output. The process starts by calculating a per-neuron delta, δ, for the neurons in the output layer, which is the product of the neuron's error signal (Target $-$ Out) and the derivative of the activation function (e.g. for sigmoid, Out$(1 -$ Out$)$), as shown in Equation 2.5. This value is then plugged into Equation 2.6, which computes how the weight value for a connection between a given node in the preceding layer and the current node in the current layer should be updated. Finally, the weight value is updated by plugging the result of the previous calculation, $\Delta w_{pq,k}$, into Equation 2.7, which provides the value to be used in the next epoch (the concept of epochs will be described in a following paragraph) (Wasserman, 1989).

$$\delta = \text{Out}(1 - \text{Out})(\text{Target} - \text{Out}) \tag{2.5}$$

where Out is the neuron's output, and Target is the expected output (Wasserman, 1989).

$$\Delta w_{pq,k} = \eta \delta_{q,k} \text{Out}_{p,j} \tag{2.6}$$

where $w_{pq,k}$ is the weight between node p and node q in layer k, thus $\Delta w_{pq,k}$ is the change to be applied to that connection in the following epoch, η is the learning (or training) rate coefficient, $\delta_{q,k}$ is the δ value calculated in the previous step (using Equation 2.5 in the case of the output layer) for node q in layer k, and $\text{Out}_{p,j}$ is the output of node p in layer j (Wasserman, 1989). When this equation is being applied to the output layer, these variables and their indices match up with the sample NN shown in Fig. 2.5, for easier visualisation.

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \Delta w_{pq,k} \tag{2.7}$$

where $w_{pq,k}(n+1)$ is the weight to be used for the connection between node p and node q in layer k in the following epoch, $w_{pq,k}(n)$ is the weight of the connection between node p and node q in layer k in the current epoch, and $\Delta w_{pq,k}$ is the change to be applied, as calculated from Equation 2.6 (Wasserman, 1989).

For hidden layers, the overall process is identical, however the equation for calculating the δ is a bit more involving, as all the deltas and weights from the previous layer (previous here implies towards the output layer, since the direction is backwards, output to input) are required for the calculation, as shown in Equation 2.8 (Wasserman, 1989).

$$\delta_{p,j} = \text{Out}_{p,j}(1 - \text{Out}_{p,j}) \sum_{q} \delta_{q,k} w_{pq,k} \tag{2.8}$$

where $\delta_{p,j}$ is the delta for node p in (hidden) layer j, $\text{Out}_{p,j}$ is simply the output of node p in layer j, and $\sum_q \delta_{q,k} w_{pq,k}$ is the sum of the products of the deltas and weights for each connection from node p to all the nodes (q) in layer k (Wasserman, 1989).

Once the training set has been passed through the network, and the networks has updated its weights using back propagation, the network is applied in inference mode on the validation set, which, as mentioned, would be data the model will not train (update its weights) on, to gauge its performance on unseen data. The process described thus far, i.e. feeding the training set through the network in forward propagation, updating the weights using back propagation, and evaluating the model on the validation set, constitutes a single epoch. This entire process is repeated a number of times, for a number of epochs. When to stop can depend on a number of factors, such as reaching convergence, the network no longer showing improvement (lower losses) on the training set, or noticing the validation performance degrading, implying the network is

overfitting. Finally, once a model (network) is considered finalised, it is evaluated on the test set, which would be yet another set of data it would have not seen as yet, to determine how well it has learnt to perform (Russell and Norvig, 2020).

2.2.1.1 | The Vanishing Gradient Problem

While the back propagation process described above works very well, and is effectively the de facto solution for training NNs (updating the weights between all of the nodes within a model) (LeCun et al., 1998), this process can suffer from what is known as the Vanishing Gradient Problem. This problem is especially present in deeper NNs, i.e. those with more hidden layers (Glorot and Bengio, 2010). These vanishing gradients are caused by the algorithm multiplying more and more values of delta, δ, which are typically < 1 as the algorithm moves deeper into the network (from the output towards the input), leading the resulting value to progressively approach 0, i.e vanish (Nielsen, 2015; Russell and Norvig, 2020).

This problem is also exacerbated by activation functions which constrain the output up to 1 (i.e. close to 0), such as sigmoid or tanh, or even 0 in the case of ReLU (see Figure 2.4) (Russell and Norvig, 2020). On the other hand, when the output of an activation function such as sigmoid or tanh is close to 1 (or -1 in the case of tanh), given their saturation towards these limits, their gradient is instead close to 0, leading to the same issue.

Relating to this issue, poor NN weight initialisation can also result in vanishing gradients, as small weights will lead to near-zero outputs, whereas larger weights will output in the saturated areas of sigmoid or tanh, resulting in near-zero gradients (Glorot and Bengio, 2010). The issue that ultimately arises from this problem is that these exceedingly small values eventually become too negligible to effectively train the network (update the weights), resulting in very slow training/learning (Russell and Norvig, 2020).

As a final note, for completeness' sake, the Vanishing Gradient Problem forms part of the (more general) Unstable Gradient Problem, together with the Exploding Gradient Problem (Nielsen, 2015).

One commonly used technique to mitigate the vanishing gradient problem is the use of ReLU activation functions, which are unsaturated activation functions, instead of the aforementioned sigmoid or tanh activation functions (Glorot et al., 2011; Ioffe and Szegedy, 2015; Tan and Lim, 2019). Glorot and Bengio (2010) also propose a normalised weight initialisation technique for more stable weights during training, where the weights initialised are scaled per layer. Another widely used technique for the miti-

gation of the vanishing gradient problem is Batch Normalisation. Batch Normalisation will be discussed further in Section 2.2.1.2, however in this context, it can, for example, help sigmoid activation functions stay within their non-saturated range (Ioffe and Szegedy, 2015). Finally, due to their architecture, in particular the inclusion of a shortcut connection, Residual Networks (ResNets) (see Section 2.2.1.2) mitigate the vanishing gradient problem by providing a shorter path from the output to the earlier layers of the network, such that these deltas are multiplied by less fractions, and can thus have a higher impact on the weights (He et al., 2016; Magro et al., 2021).

2.2.1.2 | Further Components and Concepts in Neural Network Architectures

This section will introduce some components which are fundamental to most NN architectures, namely Pooling, Dropout, Batch Normalisation, ResNets, and Dataset Augmentation.

Pooling layers work similarly to Convolutional layers (described in Section 2.2.2), where a sliding window (a kernel) slides over the input to that layer, and perform a predefined operation. This 'fixed' operation is in contrast to convolutional layers, where the weights of the kernel change as the model is trained. The two most common operations are max pooling, where the output at each step of the kernel is the maximum of the selected pixels, and average pooling, where the output at each step is the average of the pixels selected by the kernel. For example, a 2×2 max pooling layer (with a stride of 2) will typically correspond to the input's dimensions being halved (every 2×2 group of pixels becomes a single pixel) (Russell and Norvig, 2020).

One advantage of pooling is that it makes the NN invariant to any small translations in the input. This is especially relevant when it is more important to know whether a feature is present, as opposed to precisely where it is. Furthermore, since pooling can reduce the spatial dimensions of a layer's output (and thus the next layer's input), this can reduce the computational load of the NN. Moreover, a pooling (max pooling in particular) layer, by preserving the most prevalent values, can in theory reduce noise (Goodfellow et al., 2016).

Dropout (Srivastava et al., 2014) is a computationally light regularisation method that has proven effective for mitigating overfitting. During training, for every (mini)batch, dropout randomly (according to a probability specified as a hyperparameter) 'deactivates' certain neurons (i.e. multiplies their output by 0), this can include input neurons. By randomly deactivating neurons, the possibility of strong co-adaptations forming, or the NN becoming overly dependent on some specific features, is effectively mitigated, as these neurons become unreliable on their own. This makes the model more robust

as it prevents the model from becoming dependent on a feature which might only be present in the training set, but not in unseen data (overfitting). During inference, the application of the NN to unseen data, dropout is not applied, and all the neurons are activated. To ensure that the total input to a neuron during training and during inference are on the same scale, "weight scaling" is applied, such that the weights during training are either scaled up, or scaled down during inference (Goodfellow et al., 2016; Russell and Norvig, 2020; Srivastava et al., 2014).

Batch Normalisation (Ioffe and Szegedy, 2015) is a technique for addressing "Internal Covariate Shift", which should, in turn, speed up the training process of NNs. Internal Covariate Shift refers to the changing of the distribution of network activations during training. Batch Normalisation addresses this by normalising the inputs to a layer, within every minibatch, such that they have a mean of 0 and variance of 1. Batch Normalisation allows for the use of higher learning rates, which in turn has allowed models to reach, or even surpass, their known performance in a fraction of training epochs. Batch Normalisation also simultaneously mitigates the Vanishing Gradient Problem and reduces the need for specific weight initialisation techniques by keeping activation functions within their non-saturated range (see Section 2.2.1.1). Finally, Batch Normalisation also regularises the model by preventing the noise in the training data from being learnt, which helps prevent overfitting (Ioffe and Szegedy, 2015; Russell and Norvig, 2020).

ResNets (He et al., 2016) are NN architectures that make use of "shortcut connections". They are a technique that can be applied to virtually any existing NN architecture, without introducing any additional parameters or computing complexity. These shortcut connection are applied from the output of one layer to the output of another layer deeper in the network. The output from the earlier layer is normally unaltered, and summed (element-wise addition) with the output of the deeper layer. The layers can either be fully-connected, or convolutional ones, in which case the feature maps are summed per channel. The significance of these shortcut connections is that they allow for networks to grow deeper, without running into the Vanishing Gradient Problem (see Section 2.2.1.1), as back propagation is given a 'shorter' route to 'reach' the earlier (shallower) layers within an architecture (He et al., 2016).

In He et al. (2016)'s work, two identical 34-layer models, one employing shortcut connections, the other not, were tested on a classification task, and the one making use of shortcut connections (the ResNet) reduced the error rate by 3.5 percentage points. Moreover, using another dataset, more layers were progressively added to a ResNet, and the performance incrementally improved until it peaked at 110 layers.

One of the most fundamental requirements for ML is data. Generally, with more

data, models can learn to generalise to unseen data better. The issue is that this data is very often limited, especially given that for most use cases, it needs to be labelled, most of the time manually. One strategy that has come about to mitigate this issue is to expand on the dataset by generating synthetic data (Goodfellow et al., 2016; Russell and Norvig, 2020).

This process is referred to as Dataset Augmentation, and has been in use since at least 1995, when Lecun et al. applied some affine transformations, such as shifting, scaling, rotation, and skewing, to the training set. Which augmentation techniques can be applied is highly dependent on the data and task. Classification tasks (see Section 3.1), for example, are typically more flexible with which augmentation techniques can be applied, as a given input (e.g. image) will be labelled with a single class, as is the case with the dataset in Section 4.1. Even so, in, for instance, a character recognition task, 180° rotations and flips cannot be applied, as characters such as '6' and '9' or 'b' and 'd' would no longer match their label (Goodfellow et al., 2016). This illustrates why special consideration must be taken for the specific data and task in question.

Apart from the more common affine transformations (i.e. straight lines remain straight, parallel lines remain parallel, etc.), namely translation/shifting, scaling, rotation, shearing/skewing, and flipping/reflection, Cubuk et al. (2019) employ other techniques, such as AutoContrast (normalise pixel value ranges), Invert (invert pixels), Equalise (equalises the image histogram), Solarise (invert pixels above a certain threshold), Posterise (reduce colour depth, i.e. blend similar colours together), Contrast, Colour, Brightness, Sharpness (modify the contrast, colour balance, brightness, or sharpness, respectively), Cutout (turn a random square of the image gray), and Sample Pairing (blend the image with another random image). Wang et al. (2024b) also provide a comprehensive list of data augmentation techniques, which are not limited to image data.

Semantic and Instance Segmentation tasks (see Sections 3.2 and 3.3, respectively) are normally more constrained with what augmentation techniques can be applied. While flips, rotations, and translations are typically safe (as applied in Section 5.2.1.1), as long as the same exact operations are applied to the images' mask labels, more 'aggressive' augmentations can lead to object shapes being distorted too far, such that the expected object detection or label no longer applies. This, paired with the limited variety introduced by some of these operations (such as translations theoretically not contributing significantly if the model includes pooling layers), has given rise to the application of generative models for the generation of synthetic data.

RADiff (Sortino et al., 2024), for example, is a latent diffusion-based conditional generative model, which, after processing the available data, can generate additional synthetic images, even allowing the user to specify the background "intensity and noise

pattern" (by providing an arbitrary astronomical image) and the number, specific locations, shapes, sizes, and types of objects (by providing an annotated segmentation mask, either from existing data, from another unconditional generative model trained to generate masks, or even manually designed by experts).

2.2.2 | Convolutional Neural Networks

CNNs have, over time, proven themselves as fundamentals when processing images, or videos, in DL, such as in detection and recognition tasks. A CNN is essentially a NN with at least one convolutional layer. The inspiration for CNNs, particularly the "convolution/subsampling combination" (LeCun et al., 1998) comes from Hubel and Wiesel's (1962) research into animal visual cortices (LeCun et al., 1998; Liu, 2018). In contemporary terminology, the "convolution/subsampling combination" refers to the combination of convolutional (feature extraction) and pooling (translation invariance) layers.

These findings then inspired Neocognitron (Fukushima and Miyake, 1982). Neocognitron is made up of alternating layers of 'simple' and 'complex' cells, which are reflected in modern day architectures with alternating convolutional (+ activation) and pooling layers.

Before LeCun et al.'s (1989) work, the status quo of CNNs was networks having manually specified parameters. LeCun et al. (1989) addressed this in their work on handwritten digit recognition, which is now referred to as LeNet-1. This work was the first to apply back propagation to CNNs, making it possible for these to be trained, and laying the groundwork for future developments.

LeNet-5 (LeCun et al., 1998) combines the advancements of their earlier work, LeNet-1, and Neocognitron, and produces a CNN that looks familiar to present day architectures, for the recognition (classification) of handwritten characters. LeNet-5 features alternating convolutional and pooling layers, followed by fully-connected layers to make a classification. It also uses back propagation for the learning of weights, and thus training.

The architecture described in Krizhevsky et al. (2012), now referred to as AlexNet, further advances on LeNet-5. AlexNet is a classifier trained on a subset of the ImageNet dataset, ILSVRC, on images from 1,000 classes. First, it increases the number of convolutional layers, consisting of 5 convolutional layers, compared to LeNet-5's 3. AlexNet also uses ReLU activation functions, in contrast to LeNet-5's sigmoids, which allow for deeper architectures before encountering the Vanishing Gradient Problem (see Sec-

tion 2.2.1.1). AlexNet also utilises Data Augmentation and Dropout (see Section 2.2.1.2) to prevent overfitting, allowing it to train for more epochs.

VGGNet (Simonyan and Zisserman, 2015) captures the general trend of the performance improving with CNNs growing deeper, as its variants feature 8-16 convolutional layers (11-19 'weight' layers when counting the fully-connected layers too), compared to AlexNet's 5 convolutional layers (8 'weight' layers in total). VGGNets also use 3×3 kernels throughout their architectures, in contrast to AlexNet using 11×11, 5×5, and 3×3 kernels at different parts of its architecture.

This correlation of deeper CNNs with better performance reaches its limits when networks start running into the Vanishing Gradient Problem (see Section 2.2.1.1). ResNets (He et al., 2016) mitigate this by introducing "shortcut connections", giving gradients a shorter path from the output back to the earlier layers (see Section 2.2.1.2), making it possible for networks to grow to even a hundred layers.

A CNN's architecture, like the NN architecture described in Section 2.2.1, starts with the input layer. In CNNs, however, this layer functions somewhat differently. Since the input is expected to have a grid-like shape (such as any normal image), as opposed to a flat tensor in the case of 'traditional' feed-forward, fully-connected NNs, the input layer simply stores the input tensor. This tensor, particularly in the context of computer vision, will typically be one of the following shapes:

- 2D ($H\times W$): single-channel (e.g. grayscale) image (alternatively ($H\times W\times1$));
- 3D ($H\times W\times C$): multi-channel (e.g. RGB), image;
- 4D ($T\times H\times W\times C$): e.g. RGB video;
- 4D ($H\times W\times D\times C$): e.g. volumetric image;
- Any of the above with an added dimension to store minibatch data.

Where H and W refer to height and width respectively, C to channels, T to time frames, and D to depth.

The first layer after the input in a CNN is typically a convolutional layer. Convolutional layers 'slide' a kernel, also referred to as a filter, over their input, and at each step compute the 'convolution' of the filter and the pixels it is currently 'over'. In the context of CNNs, convolution is computed as the sum of the element-wise product of the 2 matrices (tensors). Goodfellow et al. (2016); Russell and Norvig (2020) clarify that 'convolution' in the context of CNNs does not refer to the convolution operation as used in signal processing, rather, it more closely resembles the cross-correlation operation in

signal processing. Despite this, 'convolution' is still the term used to refer to this operation in the context of CNNs.

Each convolutional layer will have certain hyperparameters, such as the size of the kernel (for example 3×3), the stride (by how many pixels the kernel slides), the padding (whether to pad and keep the same dimensions as the input, or only use valid data), and how many different filters are applied to the same input (and thus how many feature maps are extracted). Each of these filters can be thought of as a pattern detector for one particular feature, and their values are learnt as the model trains. In CNNs, earlier convolutional layers generally extract lower-level geometric features, such as edges and corners, while deeper layers progressively extract higher-level, more sophisticated features, and become increasingly more adept at detecting objects, e.g. eyes or noses. The output of a convolutional layer will be (either one, or) a set of feature maps, according to the specified hyperparameter values (LeCun et al., 1989; Magro et al., 2021).

In CNN architectures, convolutional layers are very often followed by an activation function (non-linearity), typically ReLU, or one of its variations, such as Exponential Linear Unit (ELU). ReLU activation functions are ideal both due to their computational simplicity, as well as their ability to mitigate the Vanishing Gradient Problem (see Section 2.2.1.1), which allows for deeper network architectures.

Another vital component in CNNs is the Pooling layer. Pooling layers are very commonly placed after a convolutional layer + activation function pair, or a small group (short sequence) of such pairs. As discussed in Section 2.2.1.2, pooling layers 'slide' a window over their tensor input, and perform a given operation, commonly max pooling (output for a window is the highest value of a window) or average pooling (average of the values in the window). Pooling downsamples its input, as for, say, a 2×2 window with a stride of 2, only one value is output, effectively halving the input's spatial dimensions. This simultaneously improves the network's translation invariance, reduces the CNN's computational load, as well as, particularly with max pooling, reduces the noise by only keeping the most prominent values (Goodfellow et al., 2016; Russell and Norvig, 2020).

If the expected output of a CNN is not of the same format as its input, such as in regression or classification problems, fully-connected layers, such as those in Section 2.2.1, are used to convert the extracted high-level local features into a 'global' understanding of the image. After the CNN has extracted features from the input following the aforementioned convolutional, activation, and pooling layers, the output of the last layer can be flattened into a 1D tensor. This 1D tensor is subsequently connected to a fully-connected layer, which can in turn be connected to other fully-connected layers. These are then finally connected to an output layer, which may include a sigmoid or softmax

activation for a regression/binary classification or multi-class classification problem, respectively (Nielsen, 2015; Russell and Norvig, 2020).

The overall training process for CNNs is similar to that described in Section 2.2.1. During training, the network first processes all, or batches of, the training data and computes its output, or prediction, for each of them, based on the network's current state of kernels and weights. Next, a loss value is calculated according to a loss function, which measures how 'wrong' the network's prediction was, when compared to the ground truth (see Section 2.2.1). Back propagation is then applied to update the network's weights. If there are any fully-connected layers before the output layer, back propagation is first applied to them, as explained in Section 2.2.1. Any pooling layers or activation functions are not affected during this process, as they perform fixed operations which do not depend on any weights, however, they can affect the flow of back propagation, since ReLU can output 0, and max pooling essentially cancels out a portion of its input, so the gradient flow is altered accordingly. When the process reaches the convolutional layers, back propagation updates the weights of kernels by first computing each kernel element's contribution to the output, i.e. how much of the output, and thus the loss, is attributed to that weight. Gradient descent is then applied to update that weight value (LeCun et al., 1998). This process is then iterated for a number of epochs, until a specified stopping condition is met, as explained in Section 2.2.1.

In contrast to CNNs, NNs utilising only fully-connected layers, i.e. NNs without any convolutional layers, are not well suited for processing visual data. One main limitation is that, even for a 100×100 pixel image, one fully-connected layer would need $10,000$ neurons. Moreover, adding a subsequent fully-connected layer of the same size would result in 100 million weights to be learnt by the model. For context, the CNNs implemented in Chapter 4 have between 100 thousand and 6 million weights, for the entire network. Such a network would not only have an excessive number of parameters, which would require longer training and inference times, as well as a larger training set, but also be sensitive to translation, requiring objects to be centred within the image. On the other hand, for a convolutional layer, a CNN is only required to learn the parameters of a filter (or set of filters), which are then applied across the entire image, making it much more efficient in terms of learnt parameters (parameter sharing).

Another limitation when applying fully-connected layers to images is that the image needs to be flattened, meaning the spatial correlation between pixels is lost. "Local correlations" are vital when dealing with visual data, and enable NNs to form a hierarchy of extracted features, with earlier layers extracting low-level features (such as edgers and corners) and later layers progressively extracting objects. Due to their architecture, convolutional layers are in fact conducive to the formation of this hierarchy, as their

kernel only considers a small region of adjacent pixels at a time (Krizhevsky et al., 2012; LeCun et al., 1989, 1998; Magro et al., 2021; Nielsen, 2015; Russell and Norvig, 2020).

2.2.3 | Quantitative Performance Evaluation of NNs

During the development of ML solutions, it is imperative to also have a means to evaluate the completed solution, for instance to compare performance with that of other available solutions. In order to facilitate this evaluation in a fair and scientific manner, the data set is normally divided into 3 subsets, a training set, validation set, and a test set. The training set contains the data to be used exclusively to train a model during its training phase. The validation set, also known as the development set, is used during the training and development of the model, to detect overfitting (by applying the model trained in that epoch to unseen data) and fine-tune hyperparameters, respectively. The test set represents the final batch of unseen data, which is held out until the end of development to evaluate the performance of the completed model (Russell and Norvig, 2020). The training set typically consists of 60-80% of the dataset, whereas the validation and test sets usually contain 10-20% each. The splitting of data into these subsets is normally done randomly, however, in e.g. Sortino et al. (2023a) the split was checked and verified to contain an even spread of images from each survey, object instances, and SNRs across each set.

2.2.3.1 | Quantitative Evaluation Metrics

In classification problems, Positive, or P, refers to an instance which the model is being trained to detect. It can also be used to refer to the total number of factual positive instances, as per the ground truth, in a dataset. For example, in the context of Chapter 4, a positive instance is an image containing a lens. Conversely, Negative, or N, refers to instances which do not contain what the model is not trying to detect. Similarly, it can be used to refer to the total number of factually negative instances in a dataset. Again, in the context of Chapter 4, a negative instance is an image which does not contain a lens (Eusebi, 2013; Fawcett, 2006; Swets, 1988).

True Positives (TP) refers to the number of positive predictions made by a model, which are in fact positive. TP is a subset of P. The True Positive Rate (TPR) is the percentage (or rate) of instances a model has correctly predicted are positive (TP) out of the total number of instances which are factually (as per the ground truth) positive instances (P). For example, a TPR of 90% implies that from a dataset with 100 positive instances (P=100), a model detected 90 of them (TP=90). TPR can be easily calculated as

shown in Equation 2.9.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = \text{Recall} = \text{Completeness} \tag{2.9}$$

The TPR is also referred to as Recall or Sensitivity, and in astronomy contexts sometimes referred to as Completeness. Again using the context of Chapter 4, in simple terms, TPR measures how many lenses a model detects out of the total number of lenses in a dataset. Alternatively, TPR measures what percentage of lenses in the dataset the model detects (Eusebi, 2013; Fawcett, 2006; Magro et al., 2021; Metcalf, R. B. et al., 2019; Swets, 1988).

False Positives (FP) refers to the number of positive predictions made by a model, which are in fact negative. FP is a subset of N. The False Positive Rate (FPR) is the percentage (or rate) of instances a model has incorrectly predicted are positive, i.e. instances which should have been predicted as negative (FP), out of the total number of factually negative instances in the dataset (N). For example, a FPR of 20% implies that from a dataset with 100 negative instances (N=100), a model classified 20 of them as positive (FP=20). FPR can be easily calculated as shown in Equation 2.10.

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{2.10}$$

Again using the context of Chapter 4, in simple terms, FPR measures how many non-lenses the model will incorrectly classify as lenses. For the FPR, a lower value implies better performance (Eusebi, 2013; Fawcett, 2006; Magro et al., 2021; Metcalf, R. B. et al., 2019; Swets, 1988).

For completeness' sake, True Negatives (TN) refers to the number of instances which have been correctly predicted as negative by the model, whereas False Negatives (FN) refers to the number of instances which have incorrectly been predicted as negative, as they are in fact positive.

Accuracy is another widely used metric when dealing with classification problems, measuring what percentage of predictions made, whether positive or negative, were correct. Accuracy can be computed as shown in Equation 2.11 (Eusebi, 2013; Fawcett, 2006).

$$\text{Accuracy} = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FN + TN + FP} \tag{2.11}$$

Precision, also referred to as Reliability in astronomy contexts, is yet another very relevant and widely applied metric, measuring the percentage of instances a model has correctly predicted are positive (TP) out of the total number of positive predictions it has made (TP + FP). For example, a Precision of 90% implies that, if a model made 100 positive predictions (TP+FP=100), 90 of them were in fact positive instances (TP=90).

Precision is also straight forward to calculate, as shown in Equation 2.12.

$$\text{Precision} = \text{Reliability} = \frac{TP}{TP + FP} \tag{2.12}$$

Precision is typically presented together with Recall (Completeness or TPR), which was presented in this section. Referring again to the context of Chapter 4, in simple terms, Precision measures how many of the lenses a model detects are in fact lenses (as a percentage) (Eusebi, 2013; Fawcett, 2006).

From the Precision (Reliability) and Recall (Completeness) of a model, a further metric can be computed, known as the F1-Score, which is the harmonic mean of both values. The '1' in 'F1' represents the equal contribution (weighting) of the precision and recall to the result, and can be computed as shown in Equation 2.13 (Riggi et al., 2023; Sortino et al., 2023a).

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{Reliability \times Completeness}{Reliability + Completeness} \tag{2.13}$$

The Receiver Operating Characteristic (ROC) curve is a plot of the TPR on the y-axis against the FPR on the x-axis. Each point on the curve is calculated by evaluating a model's TPR and FPR at a range of thresholds, from 0 to 1 (0% to 100%), at regular intervals. Threshold here refers to what cut-off or boundary will be used to convert models' continuous confidence score output to a binary classification. For example, a threshold of 0.6 implies only instances a model is 60% confident are positive, or higher, are classified as positive. The ROC plot illustrates how a model's performance varies as the threshold is adjusted. This allows for visualisation of the trade-off between the TPR and FPR, enabling the fine-tuning of the threshold according to the desired balance between the metrics. A curve which resembles the TPR=FPR diagonal represents a model which is as effective as a coin flip (a random guess). On the other hand, a curve which sharply approaches TPR=1 (the top-left corner) indicates that the model achieves a high TPR while maintaining a low FPR. Moreover, the Area Under the ROC (AUROC), or the Area Under the Curve (AUC) in short, is a metric for quantitatively comparing ROCs, and thus model performances (Eusebi, 2013; Fawcett, 2006; Magro et al., 2021; Metcalf, R. B. et al., 2019; Swets, 1988).

Another widely used plot is the Precision-Recall (Reliability-Completeness) curve, which graphically depicts the tradeoff between a model's Precision (Reliability) and Recall (Completeness) as the confidence score threshold is adjusted over the range 0 (favouring completeness) to 1 (favouring reliability) (Fawcett, 2006). A plot with such curves is presented in Fig. 5.4. Precision-Recall curves can present more realistic views

of models' performance in scenarios where there is an imbalance in the number of instances of each class in the dataset, when compared to ROC curves (Davis and Goadrich, 2006).

In the case of object detection problems in images, where a model's predictions are not, say, a yes or a no, or a classification, but the location of an object in an image, the process of determining whether a prediction matches the ground truth is not as straightforward. This is normally determined by calculating the Intersection over Union (IoU) between the predicted region and the ground truth region, which, as the name suggests, is the ratio of the overlap (intersection) to the total area (prediction + ground truth - intersection). These aforementioned regions or areas could refer either to the bounding boxes, or the pixel-wise segmentation masks, depending on the problem. With this index, a correct prediction (TP) becomes an instance where the IoU of a predicted object and the ground truth object it is being compared to is greater than a preconfigured threshold (e.g. 0.5 or 0.6) (Riggi et al., 2023; Sortino et al., 2023a).

Finally, Mean Average Precision (mAP), and more specifically mAP50, are evaluation metrics, popularised by the Pascal VOC (Everingham et al., 2010) and COCO (Lin et al., 2014) challenges, commonly used in instance segmentation problems (as defined in Section 3.3), that jointly evaluate a model's localization as well as classification performance. The Average Precision (AP) for a particular object class can be computed by calculating the area under the (aforementioned) Precision-Recall curve for that class. The mAP can then be calculated as the mean of APs across all object classes. The mAP50 is therefore the mAP with APs computed using IoU thresholds of 0.5 (Sortino et al., 2023a).

2.3 | Conclusion

This chapter gives an introduction to the basic concepts which are used throughout this thesis, namely Radio Astronomy and AI. It starts with a brief history of astronomy paired with a high-level description of optical astronomy. This is then followed by a short account of the discovery of radio astronomy, together with a description of the electromagnetic spectrum, on which visible light (optical astronomy) and radio waves (radio astronomy) both exist. This leads into a description of the Earth's atmospheric opacity, the understanding of which makes it clear why optical and radio astronomy are the main viable solutions for surveys conducted from the ground. Next, some of the advantages of radio astronomy over optical astronomy are mentioned, alongside some of their disadvantages. A brief history and introduction to gravitational-wave astronomy is also presented.

Interferometers are then introduced, particularly for their ability to mitigate most of these disadvantages, namely the need for very large telescopes. Finally, the upcoming SKA telescope and the volumes of data at high resolutions it is expected to produce are mentioned, necessitating the application of AI to deal with this data.

The following section discusses definitions of AI and mentions several different techniques. Next, one of these techniques in particular, ML, is expanded upon since it will be the main solution applied to problems throughout this thesis. Again, one particular ML algorithm, NNs, are delved into given their prominence in this thesis, starting with their basis in the human brain and the modelling of artificial neurons from brain neurons. The general structure of NNs, that of neurons sorted into connected layers, is then discussed, and DL is introduced for the revolutionary representational power hidden layers give NNs. Next, the operation of NNs is described, namely the forward pass, loss functions, and the mathematics of weight adjustment during back propagation. Finally, the repetition of these processes for training a model on a given training dataset and validation set, together with its evaluation on an unseen test set, is described.

Next, an explanation of the vanishing gradient problem is given, together with commonly applied mitigations. Furthermore, key components and concepts in NNs are also presented, such as pooling, dropout, batch normalisation, ResNets, and dataset augmentation.

This introduction to AI, ML, and DL, is followed by a fundamental tool that makes NNs reasonably applicable to images, CNNs. Convolutional layers are capable of extracting features from images using a sliding kernel, all while keeping important spatial correlations in images, something which flat NN layers are not nearly as suitable for.

Finally, this chapter concludes by presenting a series of quantitative evaluation metrics which are used throughout this thesis to objectively compare solutions, such as TPR, FPR, Accuracy, Precision/Reliability, Recall/Completeness, F1-Score, ROC curves, AUC, Precision-Recall curves, and mAP.

3

Literature Review

3.1 | Classification Algorithms

Some of the content in this section, Section 3.1, incorporates concepts discussed in Magro et al. (2021).

Classification problems are tasks where the output will be one of a defined set of values, referred to as labels. In such problems, the objective of the solution is to assign, or predict, a label for the input data (Russell and Norvig, 2020). For example, in Chapter 4, the task is to classify a given image as containing, or not containing, gravitational lensing. This section presents a variety of classification algorithms, both conventional, and ML- or NN-/CNN-based.

3.1.1 | Conventional Methods

None of the conventional methods mentioned here are applied in this work, and are only described to broadly present what methods exist for tackling classification tasks.

3.1.1.1 | Visual Inspection

As the name suggests, visual inspection is the process of a person manually looking at, and labelling or classifying, images, one by one. Hartley et al. (2017) apply this method to 'Challenge 1.0' of the 'Gravitational Lens Finding Challenge' (Metcalf, R. B. et al., 2019). Hartley et al. (2017), using their purpose-built tool BIGEYE, which facilitates the process of displaying images in the dataset and allowing the user to label it, claim they can label up to 5,000 images an hour. This means labelling the 100,000 images in the challenge this was applied to would take upwards of 20 hours, whereas NN-based

solutions would complete the same task in around an hour. While the accuracy a trained astronomer should be able to achieve in theory beats that of a NN, in reality, it does not, as humans tend to make mistakes when viewing thousands of images, and in fact, in said challenge this solution fared relatively poorly when compared to other solutions (Metcalf, R. B. et al., 2019).

3.1.1.2 | Arc-finders

Arc-finders, such as ARCFINDER (Alard, 2006) and YATTALENSLITE (Sonnenfeld et al., 2018) attempt to solve the aforementioned 'Gravitational Lens Finding Challenge' (Metcalf, R. B. et al., 2019) by attempting to detect elongated structures indicative of lensing. In general, such techniques look for hard coded features which astronomers determine will help detect and properly classify the object in question. In the context of the 'Gravitational Lens Finding Challenge', these two Arc-finders performed worse than the Visual Inspection method (Metcalf, R. B. et al., 2019).

3.1.1.3 | Machine Learning (Pre-Selected Features)

These techniques consist of algorithms that extract a set of features for their inputs, from which an expert will determine which are most relevant. The final classification is then determined by a boundary placed on the feature space of the relevant features; the boundary is either specified by intuition or trial-and-error. MANCHESTER-SVM (Hartley et al., 2017) is a Support-Vector Machine (SVM) (Vapnik, 1979) based solution for the 'Gravitational Lens Finding Challenge'. ALL (Avestruz et al., 2019) is a similar solution, however instead uses a Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) based approach. In the 'Gravitational Lens Finding Challenge', these two techniques perform about as well as Visual Inspection, sometimes performing slightly better, and other times slightly worse (Metcalf, R. B. et al., 2019).

3.1.2 | Machine Learning Classifiers

This section will discuss CNNs that are capable of classification of images, in particular those which have previously been used on astronomy data. The majority of these models have been applied to 'Challenge 1.0' of the 'Gravitational Lens Finding Challenge' (Metcalf, R. B. et al., 2019). Virtually all the techniques mentioned in this section perform better than the Conventional Methods mentioned in Section 3.1.1.

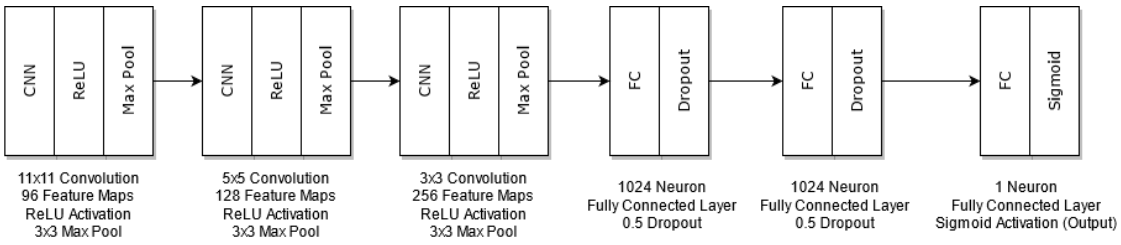

Figure 3.1: Graphical representation of the 'CAS Swinburne' model described in Section 3.1.2.1. Reproduced from Jacobs et al. (2017); Magro et al. (2021).

3.1.2.1 | CAS Swinburne

This architecture is based on AlexNet (Krizhevsky et al., 2012). The input image first undergoes three consecutive convolutional layers, with an activation function and a max pooling layer following each of the convolutional layers. The output of the last max pooling layer is passed to two successive fully-connected layers, each followed by a ReLU activation and a dropout layer. Finally, the last layer is fully-connected to a single neuron with a sigmoid activation, which represents the model's output (Jacobs et al., 2017; Metcalf, R. B. et al., 2019). This architecture is shown graphically in Fig. 3.1, with a more detailed textual description provided in Appendix A.1.

3.1.2.2 | LASTRO EPFL

This architecture resembles that described in Section 3.1.2.1, in that they are both made up of the same building blocks, however 'lastro_epfl' is a considerably larger model, with almost twice as many layers. The input image is first passed through 3 'blocks', each comprising a pair of convolutional layers with activation functions, a max pooling layer, and a batch normalisation layer. The final block is followed by a dropout layer, added to reduce the possibility of overfitting. This is then followed by a convolutional layer, a dropout layer, another convolutional layer, batch normalisation, and another dropout layer. The output from the last layer is then flattened, and connected to a triple of fully-connected layers and activation functions, with a dropout layer in between each pair. Batch normalisation is applied after the last of these fully-connected layers. Finally, the output is obtained from a final fully-connected layer, with a single neuron and a sigmoid activation function (Metcalf, R. B. et al., 2019; Schaefer et al., 2018). This architecture is shown graphically in Fig. 3.2, with a more detailed textual description provided in Appendix A.2.

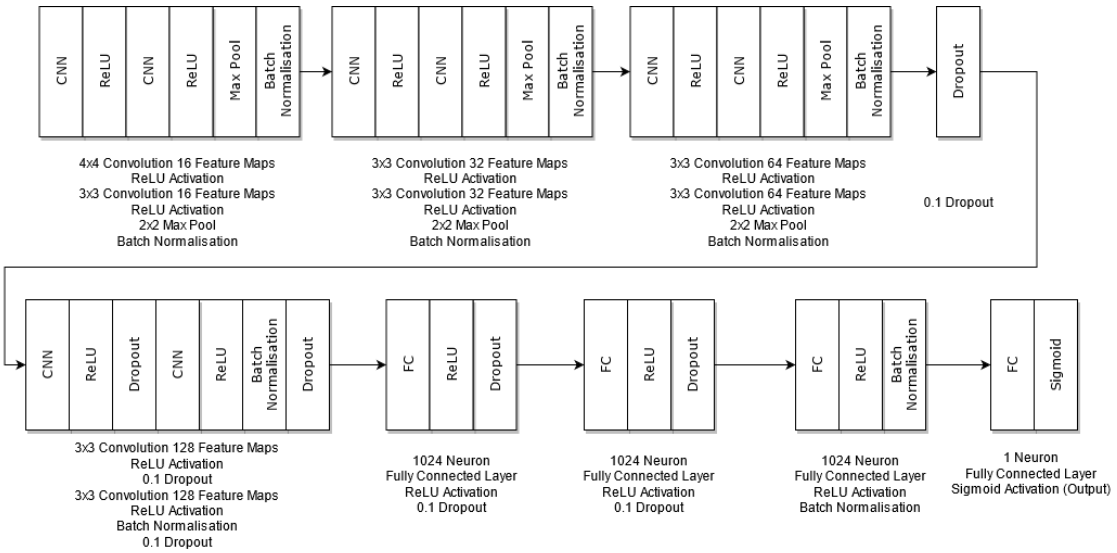

Figure 3.2: Graphical representation of the 'lastro_epfl' model described in Section 3.1.2.2. Reproduced from Magro et al. (2021); Schaefer et al. (2018).

3.1.2.3 | CMU DeepLens

When compared to the architectures described in Sections 3.1.2.1 and 3.1.2.2, CMU DeepLens shows similarities as it is also ultimately based on CNNs, however distinguishes itself as it makes use of ResNets (see Section 2.2.1.2). A ResNet is a network in which there exist "shortcut connections" from the input to the output of a series of convolutional layers. This local structure, i.e. the input to a series of convolutional layers, the convolutional layers themselves, and their output, will be referred to as a 'ResNet block'. The output of a 'ResNet block' is computed as the sum of the block's original input, and the output of the last convolutional layer within the block. One major advantage of ResNets is that, through these skip connections, the 'vanishing gradient problem' (see Section 2.2.1.1) is mitigated, as back propagation is given a 'shorter' route to 'reach' the earlier layers within an architecture.

CMU DeepLens makes use of two different architectures of ResNet blocks. The first of these maintains the original resolution of the image. This block first stores a copy of its input. Next, the input goes through a sequence of batch normalisation, activation functions, and a convolutional layer with another activation, three times. Finally, the output of the ResNet block is returned as the sum of the original input and the output of the convolutional layers within the block.

The second of these ResNet blocks downsamples the image. In this block, batch normalisation and an activation are first applied, before a copy of the tensor is stored. Then,

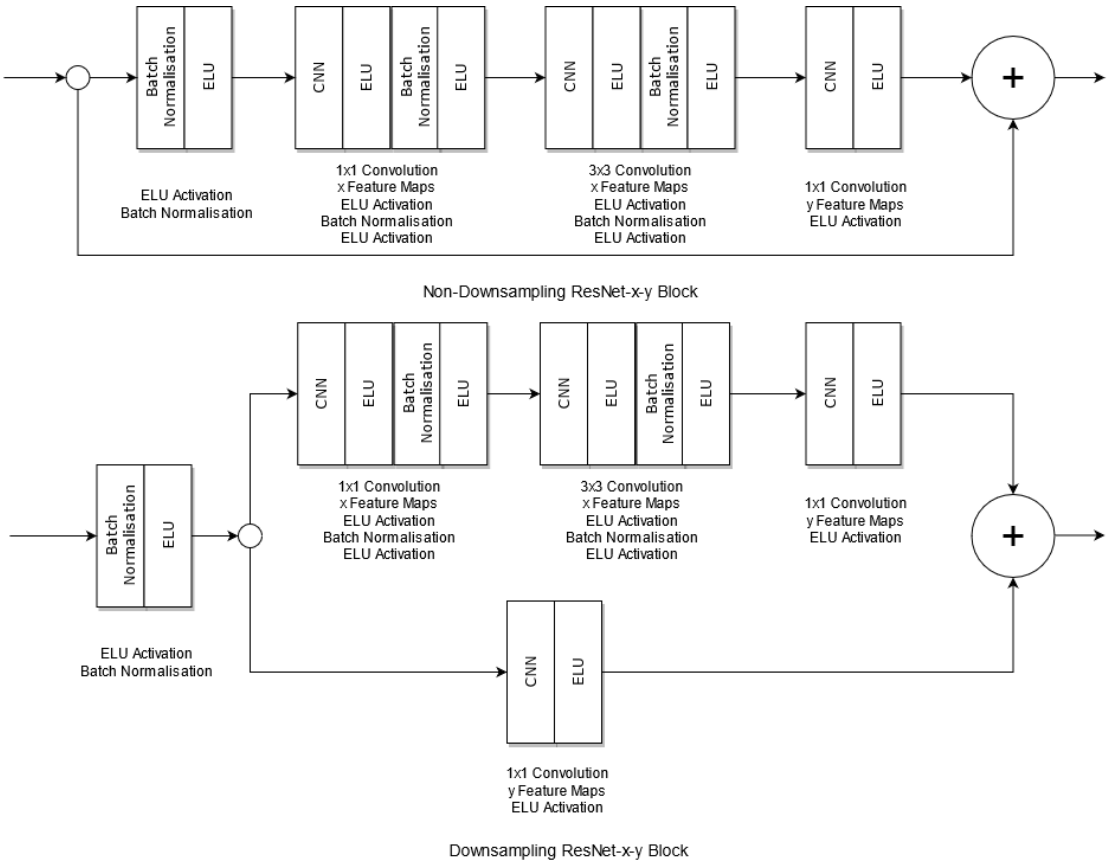

Figure 3.3: Graphical representation of the two types of 'ResNet blocks' used by the 'CMU DeepLens' model described in Section 3.1.2.3. Reproduced from Lanusse et al. (2018); Magro et al. (2021).

a convolutional layer with a stride of 2 is applied, which is responsible for the downsampling of the block's input. This is followed by a sequence of batch normalisation, activation, and a convolutional layer with another activation, twice. A convolutional layer with a stride of 2 is also applied to the aforementioned stored input, with which the dimensions of the input and the convolutional layers' output will match, after which they are summed, and returned as the block's output. The architectures of these ResNet blocks are shown graphically in Fig. 3.3, with a more detailed textual description provided in Appendix A.3.

The overarching architecture of the CMU DeepLens model is as follows. A convolutional layer with an activation is first applied to the input, followed by a batch normalisation layer. 3 'non-downsampling' ResNet blocks follow, after which a triple of ResNet blocks is applied 4 consecutive times. Each triple consists of a downsampling ResNet

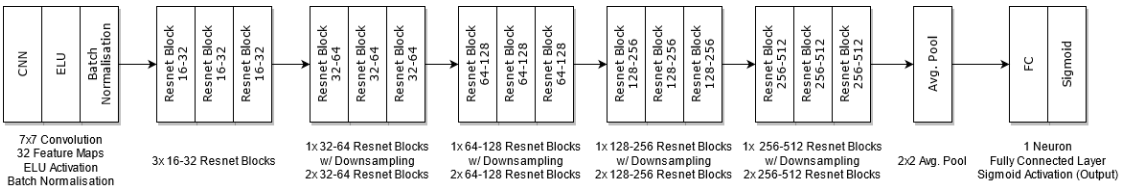

Figure 3.4: Graphical representation of the 'CMU DeepLens' model described in Section 3.1.2.3. Reproduced from Lanusse et al. (2018); Magro et al. (2021).

block and 2 'non-downsampling' ResNet blocks. The last ResNet block's output then goes through an average pooling layer, after which a fully-connected layer with a single neuron and a sigmoid activation function produces the model's prediction (Lanusse et al., 2018; Metcalf, R. B. et al., 2019). This architecture is shown in Fig. 3.4, with a more detailed textual description provided in Appendix A.3.

3.1.2.4 | WSI-Net

The WSI-Net model was not originally designed for astronomical applications, but to detect tumours in breast scans and classify them. The same architecture, up to the "classification branch", can be applied to classify an astronomical image (in this work, whether an image contains a lens). Its architecture resembles that of CMU DeepLens, described in Section 3.1.2.3, in that they are both based on ResNets. The original work does not specify hyperparameter values, and thus those mentioned here are what was found to produce the best results, empirically. In WSI-Net, the image is first passed through a convolutional layer with an activation function. This is followed by two ResNet blocks, as those described in Section 3.1.2.3, the second of which downsamples the image. Next, two blocks of convolutional layers, batch normalisation, and activation functions follow. This is followed by a max pooling layer, and then by a fully-connected layer. Finally, the model's prediction is calculated as the output of a fully-connected layer with a single neuron and a sigmoid activation (Ni et al., 2019). This architecture is represented graphically in Fig. 3.5, with a more detailed textual description provided in Appendix A.4.

3.1.2.5 | LensFlow

The architecture of LensFlow is relatively simplistic, and resembles the CAS Swinburne (Section 3.1.2.1) and LASTRO EPFL (Section 3.1.2.2) models in terms of the number and type of layers involved. The image is first passed through an average pooling layer. This is followed by 3 sets of convolutional layer + max pooling layer pairs. All of these con-

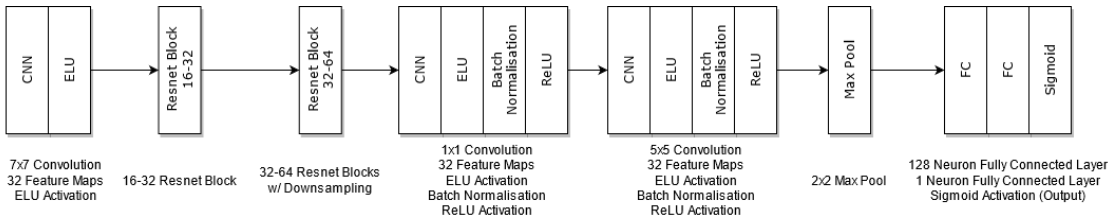

Figure 3.5: Graphical representation of the 'WSI-Net' model described in Section 3.1.2.4. Reproduced from Magro et al. (2021); Ni et al. (2019).

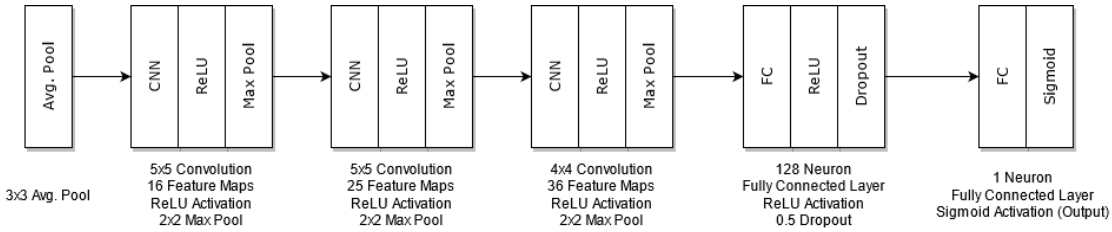

Figure 3.6: Graphical representation of the 'LensFlow' model described in Section 3.1.2.5. Reproduced from Magro et al. (2021); Pourrahmani et al. (2018).

volutional layers are followed by an activation. Next, the max pool's output is fed into a fully-connected layer and an activation function. This is also followed by a dropout layer, which is only active during the training of the model. A fully-connected layer with a single neuron and a sigmoid activation function produce the model's final output (Pourrahmani et al., 2018). This architecture is shown graphically in Fig. 3.6, with a more detailed textual description provided in Appendix A.5.

3.1.2.6 | LensFinder

Despite its very simplistic architecture, even when compared to CAS Swinburne and LensFlow (Sections 3.1.2.1 and 3.1.2.5), at effectively only 6 layers, LensFinder still manages to perform very respectably. The original work does not specify hyperparameter values, and thus those mentioned here are what was found to produce the best results, empirically. Furthermore, for its application in this work, i.e. a binary classification problem, the final layer's activation function was changed from a softmax to a sigmoid. The model starts with 2 convolutional layer + max pool pairs. These are followed by a fully-connected layer with an activation. Finally, the output is produced by a single neuron in a fully-connected layer with a sigmoid activation function (Pearson et al., 2018). This architecture is displayed graphically in Fig. 3.7, with a more detailed textual description provided in Appendix A.6.

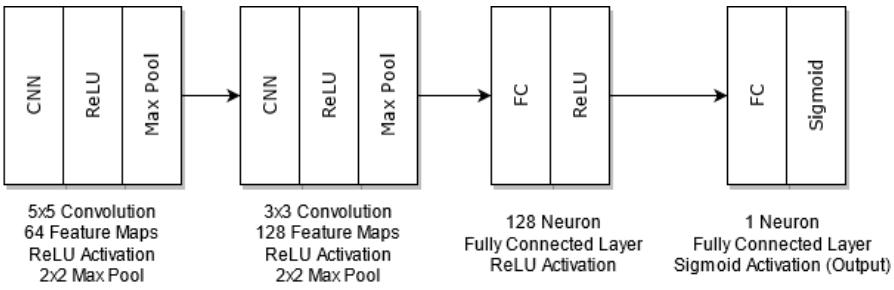

Figure 3.7: Graphical representation of the 'LensFinder' model described in Section 3.1.2.6. Reproduced from Magro et al. (2021); Pearson et al. (2018).

3.2 | Semantic Segmentation Algorithms

As described in Section 3.1, Classification Algorithms will assign a (singular) label to a given input. Semantic Segmentation algorithms, on the other hand, will classify every individual pixel in a given image, and classify which class they belong to, producing a pixel mask with the same dimensions of the original image (He et al., 2017; Sortino et al., 2023a). For instance, in the context of the Gravitational Lens Detection problem, classification algorithms will output whether a given image contains, or does not contain, lensing. Semantic segmentation algorithms will instead classify, or label, every pixel in the given image to denote which pixels include lensing, and which do not. Figure 3.8 shows this distinction graphically.

3.2.1 | U-Net

U-Net (Ronneberger et al., 2015) is a deep CNN model, originally developed for the segmentation of biomedical images. It gets its name from the model's 'u-shaped' architecture, as can be seen in Fig. 3.9. The model works in an end-to-end setting, meaning it receives raw images as input, and will output segmentation masks. Some advantages of this architecture include the small number of training examples required, in the original application, only 30 labelled images were used for training, however these were paired with image augmentation techniques. Training was also documented to take "10 hours", with inference taking around 1 second (Ronneberger et al., 2015). Another advantage of the architecture is that it can work with arbitrarily large image sizes, given its fully convolutional architecture.

The model starts out with its 'contraction' phase. During this phase, a series of convolution and max pooling steps take place. After each step, the dimensions (x, y) of the image are halved (assuming the standard 2x2 max pooling operation), while the fea-

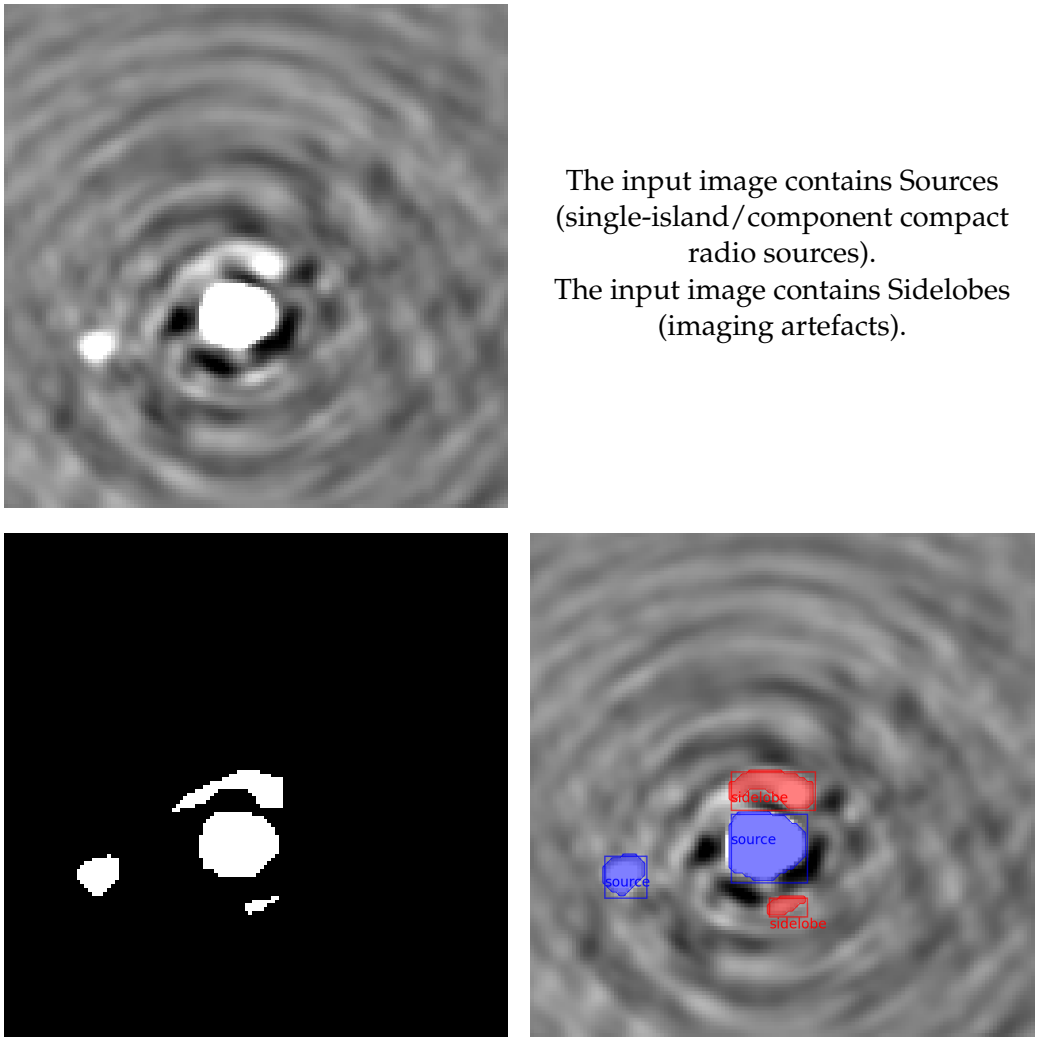

Figure 3.8: Comparison of Classification, Semantic Segmentation, and Instance Segmentation. The figure in the top-left shows an example input image (Reproduced from the dataset described in Section 5.1). The panel in the top-right shows the output from a Classifier, the bottom-left from Semantic Segmentation, and the bottom-right from Instance Segmentation.

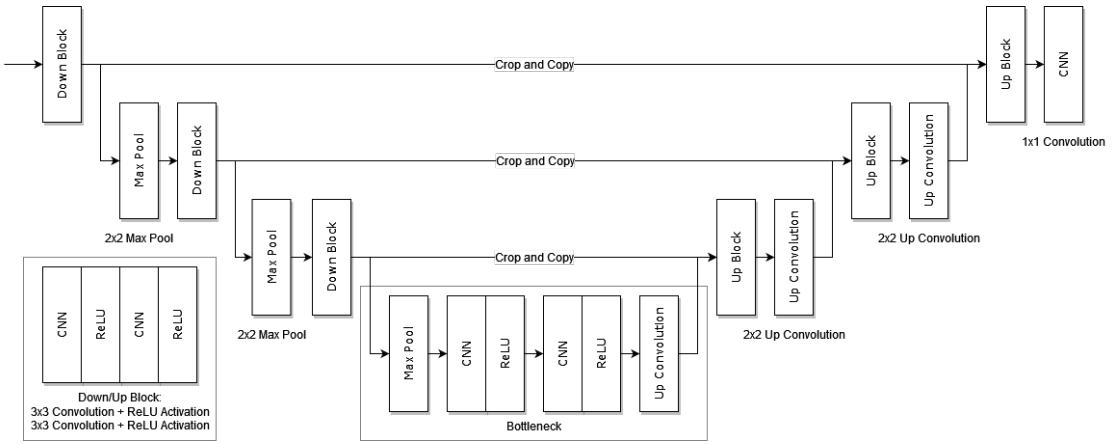

Figure 3.9: Graphical representation of the 'U-Net' architecture described in Section 3.2.1. Reproduced from Ronneberger et al. (2015).

tures double. With each step in the downwards contraction path, the model "increases the what" and "decreases the where".

After the end of the contraction phase, the bottleneck phase takes place, at the 'bottom' of the 'u shape'. This simply comprises 2 pairs of convolutions and ReLU activations, followed by an "up-convolution", which, contrary to the max pooling operation, doubles the dimensions (x, y) of the mask being produced.

Finally, the 'expansion' phase occurs, during which the model produces the final segmentation mask. Here, the model combines the feature maps extracted during the contraction phase, with the output of the up-convolution from the previous layer, and performs further convolutions and up-convolutions on the concatenated input. This phase will have as many steps as the contraction phase, and ultimately performs a 1x1 convolution, producing a segmentation mask for the original input image (Ronneberger et al., 2015; Sortino et al., 2023a). This architecture can be seen graphically in Fig. 3.9.

3.2.2 | U-Net++

U-Net++ (Zhou et al., 2018) is a variant of U-Net which further builds upon the U-Net architecture and attempts to improve its performance by augmenting the skip pathways with dense convolution blocks, and incorporating deep supervision.

As explained in the previous section, Section 3.2.1, U-Net features skip/shortcut connections, which combine (concatenate) the feature maps extracted during the contraction (downsampling) phase, with those from the corresponding layer in the expansion (upsampling) phase. U-Net++ enhances these skip pathways by adding a "dense

convolution block" (Zhou et al., 2018).

These dense convolution blocks consist of a series of convolutional layers, the number of which relates to the level of the u shape, with the shallower levels (closer to the top of the u shape) consisting of more layers. Each convolutional layer in these blocks performs a regular convolution, however taking the concatenated input of all the intermediate layer outputs up till that point in the block. Moreover, the input is also concatenated with the intermediate output of the convolutional layers in the dense convolution block from the skip pathway of the deeper layer. In theory, this bridges the "semantic gap", i.e. the varying levels of abstraction between the low-level downsampling features and high-level upsampling features. This in turn stabilises training by encoding the features more meaningfully and simplifying the gradient computation.

Furthermore, U-Net++ applies Deep Supervision (Lee et al., 2015), meaning it also assigns a loss value to the (intermediate) outputs of the convolutional layers within the skip pathways. This should help the model train better, as specific 'feedback' is being provided for these functions. This also technically allows for the model's architecture to be pruned for inference, which makes it more compatible with lower end hardware (Sortino et al., 2023a; Zhou et al., 2018).

3.3 | Instance Segmentation Algorithms

Some of the content in this section, Section 3.3, particularly Sections 3.3.1 and 3.3.2, incorporates concepts discussed in Riggi et al. (2023).

One of the aims of this work is to develop an automated and reliable instance segmentation method. Instance Segmentation is the combination of Object Detection and Semantic Segmentation. Object Detection is referring to the recognition of objects in an image, i.e. their classification/labelling, and the drawing of a bounding box around their position. Semantic Segmentation meaning the classification of the pixels belonging to objects of interest (He et al., 2017). In other words, the aim is to have a method which can be given an astronomical image, and no further user input, and will detect all the objects of interest within the image, assign them a label, draw a bounding box around their position, and highlight which pixels belong to that object, differentiating between overlapping objects. More precisely, in this context, the method should be able to recognise and consider multiple component galaxies as a singular galaxy.

3.3.1 | Review of Existing Methods and their Limitations

Before describing the architecture which was eventually developed and applied in Chapter 5, this subsection briefly introduces a number of methods and architectures which have been applied to the domain of astronomy, however, for varying reasons, do not satisfy the desired solution, i.e. to develop an all-in-one solution for instance segmentation of astronomical images.

3.3.1.1 | ConvoSource

ConvoSource (Lukic et al., 2020) is a CNN-based source finder for compact and extended Star-Forming Galaxies (SFGs) and Active Galactic Nuclei (AGNs) (both steep- and flat-spectrum). ConvoSource splits large images ($4,000 \times 4,000$ or $4,200 \times 4,200$ pixels) from the simulated SKA Data Challenge I (SDC1) (Bonaldi et al., 2020) dataset into 50×50 panels. The intended output is a panel of the same dimensions, with only a few pixels indicating the location of detected sources. Thus, its architecture is rather straightforward, comprising three convolutional layers and a fully-connected layer with a sigmoid activation to produce the output. The convolutional layers use a kernel size of 7, 5, and 3, and produce 16, 32, and 64 features, respectively, with 'same' padding to preserve the input dimensions. A dropout layer with a probability of 0.25 is also placed after the first layer. In one of its better performing scenarios, i.e. Band 1 1,000 h at SNR > 5 across all classes, ConvoSource achieves a precision of 0.73, a recall of 0.83, and an F1-Score of 0.78. While ConvoSource performs respectably, it does not satisfy the requirements of the desired solution, which is to perform instance segmentation, as it neither classifies the detected sources nor does it produce a per-pixel mask for detections (Lukic et al., 2020).

3.3.1.2 | DeepSource

DeepSource (Vafaei Sadr et al., 2019) is another CNN-based source finder, which, similarly to ConvoSource (Section 3.3.1.1), produces images of the same dimensions as the input, indicating the (predicted) true location of detected point sources with a single pixel. DeepSource is trained on a simulated dataset of 200×200 pixel noisy images of the radio sky, each image with a corresponding map with a single pixel representing the true location of a point source as the target/ground truth. The DeepSource architecture starts with a series of convolutional layers, each with a kernel size of 5, producing 12 features, and a ReLU activation (except for the last layer, which only produces 1 feature map). A shortcut connection and batch normalisation are also present in the network.

The goal of this part of the network is to enhance the image, and boost sources' SNR. This is followed by Thresholded Blob Detection (TBD), which dynamically (per image), progressively lowers a threshold until a pre-defined maximum number of blobs (groups of adjacent, connected pixels) are formed. This produces the initial list of potential point sources, which is then further refined by filtering out blobs with an area lower than a pre-defined value. Each blob's (predicted source's) centre is then calculated, and presented as the model's prediction. DeepSource is reported to achieve precision and recall of 0.45 and 0.85 on a SNR of 3, and 0.99 and 1.00 on SNRs greater than 4, respectively (Vafaei Sadr et al., 2019). Similarly to ConvoSource (Section 3.3.1.1), the performance is rather impressive, however is not a fit for the desired solution, as DeepSource detects and locates point sources, but does not distinguish between and classify objects, nor does it produce per-pixel masks for each object.

3.3.1.3 | ClaRAN

ClaRAN (Wu et al., 2018) is a radio source detector and morphology classifier based on Faster R-CNN (Ren et al., 2017). It combines radio and infrared data from the Faint Images of the Radio Sky at Twenty cm (FIRST) and Wide-field Infrared Survey Explorer (WISE) surveys, respectively, from the Radio Galaxy Zoo (RGZ) first Data Release (DR1) dataset, to create more comprehensive images with more than one channel of information. The ClaRAN architecture starts off with 13 convolutional layers, interspersed with 4 max pooling layers. The next 5 layers (layers 18-22) make up the Localization Network (LocNet), and are responsible for proposing regions, each thought to contain a radio source. The final 7 layers (layers 23-29), the Recognition Network (RecNet), are then responsible for determining which morphology each Region of Interest (ROI) belongs to, as well as coordinate adjustments for each ROI, assuming each morphology. The adjustment applied will be the one corresponding to the morphology determined by the first output. ClaRAN achieves a mAP of 0.77 to 0.84, depending on how the data is pre-processed. While ClaRAN does perform object detection (bounding boxes) and classification, it does not meet the criteria expected of the ideal solution, as it does not produce a per-pixel mask, and thus is not an instance segmentation algorithm (Wu et al., 2018).

3.3.2 | Mask R-CNN

Mask Region-based Convolutional Neural Network (Mask R-CNN) (He et al., 2017) was a state-of-the-art DL model when it was proposed by the Facebook AI Research team in 2017. Mask R-CNN is an instance segmentation algorithm, meaning it is an all-in-one

solution for performing object detection, classification, and semantic segmentation on images. It has been successfully applied to several domains, such as cancer detection and diagnosis in medical imaging (Cao et al., 2019), marine mammal identification in photogrammetry (Gray et al., 2019), as well as star and galaxy detection in (simulated) astronomical images (Burke et al., 2019).

Mask R-CNN is, at the time of writing, the latest in the line of R-CNN models (R-CNN, Fast R-CNN, and Faster R-CNN (Ren et al., 2017)). It builds upon, and shares a similar architecture to, its predecessor, Faster R-CNN (Ren et al., 2017). The advancements include a Feature Pyramid Network (FPN) as part of the backbone, replacing the ROI pooling step with RoIAlign, and the addition of a Fully Convolutional Network (FCN) for producing object masks. Fig. 3.10 depicts the high-level architecture of Mask R-CNN.

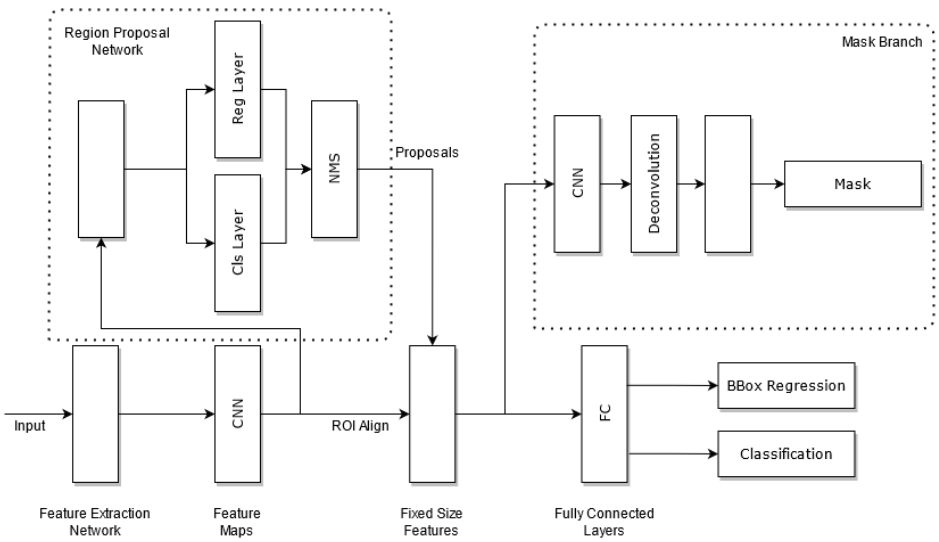

Figure 3.10: High-level graphical representation of the Mask R-CNN architecture described in Section 3.3.2, reproduced from Riggi et al. (2023); Yang et al. (2020).

3.3.2.1 | Feature Pyramid Network

The Mask R-CNN model starts off by passing the input image through the FPN, which functions as a feature extractor within the model. The structure of the FPN is based on two consecutive 'pyramids', the first of which is traversed bottom-up (referred to as the bottom-up pathway), while the second is traversed top-down (referred to as the top-down pathway).

The bottom-up pathway employs a ResNet for extracting features at multiple scales, typically a ResNet101 (He et al., 2016) in most implementations. The fact that multiple scales are used is vital, since objects will have different sizes in different images. As discussed in Section 3.1.2.3, ResNets allow for models to have even more layers (i.e. be 'deeper'), which is generally associated with higher performance, as their skip connections mitigate the vanishing gradient problem. At each stage of the bottom-up pyramid, the last ResNet block's output feature maps are laterally connected to the top-down pyramid at the same "stage". The last convolutional layer's output is used, not only because this is standard in ResNet-like structures, but also because the deepest layer at each stage is expected to have captured the most significant features. Note that this does not include the first convolution applied to the image, and this is generally not part of the symmetrical pyramid structures. As with most convolutional networks, the earlier layers capture lower-level features, whereas the later layers capture higher-level, "semantically stronger", features (He et al., 2017; Lin et al., 2017).

The pyramid in the top-down pathway will have an equal number of stages to the first (bottom-up pathway) pyramid. It will start by taking the semantically strong (however, spatially coarse) features extracted by the deepest layers of the first pyramid, and upsamples them by a factor of 2, using nearest neighbour upsampling. This can be thought of as the reverse process of a pooling operation, where, instead of having pairs of pixels compressed into one, one pixel is upsampled into two. Furthermore, the aforementioned lateral connections from the bottom-up pathway are passed through a 1×1 convolution and summed with the upsampled output, creating the next feature map which is passed to the next stage. This process is repeated for each stage in the pyramid. The power of these lateral connections is that they enable the model to combine semantically rich features from deeper layers, with spatially rich information from the earlier layers in the bottom-up pathway (He et al., 2017; Lin et al., 2017).

This is vital for Mask R-CNN, as it needs to be able to accurately detect and classify objects within the image, but also properly localise them spatially. This architecture is shown graphically in Fig. 3.11. The final feature maps are then passed to the Region Proposal Network (RPN) to determine the ROIs.

3.3.2.2 | Region Proposal Network

The RPN was introduced in Faster R-CNN (Ren et al., 2017) to replace the selective search algorithm in Fast R-CNN (Girshick, 2015). This allowed the network to be trained end-to-end, as it was built into the network, and not an external algorithm. As the name suggests, the purpose of the RPN is to propose regions which are likely to contain

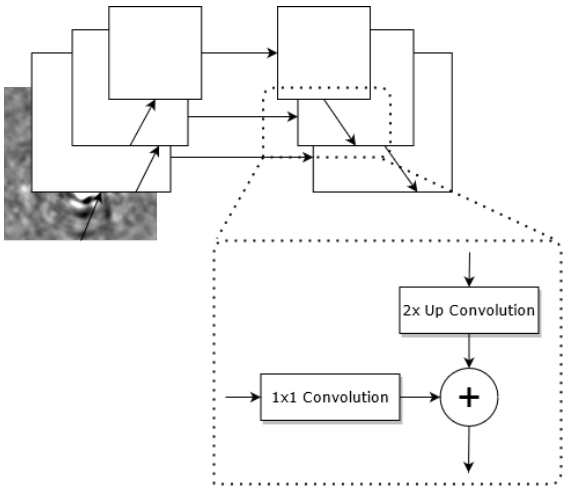

Figure 3.11: High-level graphical representation of the FPN architecture described in Section 3.3.2.1, reproduced from Lin et al. (2017).

an object of interest. It works by scanning over regions on the feature maps, called anchors, generated by the FPN. There will typically be thousands of anchors for any given feature map, of varying, predetermined, sizes (to cater for different object sizes) and aspect ratios (to cater for different object orientations). These are then processed in parallel, and evaluated by the RPN's classifier, which will determine whether they are foreground (contain an object of interest) or background, assigning an "objectness" score. Furthermore, the RPN will compute bounding box refinements, such that the proposed bounding boxes thought to contain objects will more accurately cover the object within them (He et al., 2017; Ren et al., 2017).

The anchors are ranked by the aforementioned objectness score, and only a specified number with the highest score are kept. Furthermore, the IoU of overlapping anchors are compared, and if the overlap is higher than a set Non-Maximum Suppression (NMS) parameter (typically set to 0.7), one is dropped, effectively eliminating duplicate object detection anchors. The remaining proposed regions are now considered ROIs, and passed on to the next step (He et al., 2017; Ren et al., 2017).

3.3.2.3 | RoIAlign

The regions proposed by the RPN will be of varying shapes and sizes, which poses a problem, as the architecture of the network, i.e. the subsequent fully-connected layers, requires a fixed-size input. ROI Pooling (Girshick, 2015) was originally introduced in Fast R-CNN to address this problem. There are, however, issues with this approach. The

variably-sized proposed region (from the RPN) is not guaranteed to neatly subdivide over the fixed-size grid, due to it not being able to make use of fractional pixels. This leads to rounding down, such that a whole number of pixels fit into each 'bin' within the fixed-size grid, resulting in loss of information as some pixels are ignored, and "harsh quantization" (He et al., 2017) issues. This leads to misalignment between the mask and the original image, which is tolerable in the case of detection and classification, however, is problematic when creating pixel-accurate masks.

To mitigate this, He et al. (2017) introduce RoIAlign in Mask R-CNN. RoIAlign solves the issues described with ROI pooling, as it uses bilinear interpolation instead of quantization. In effect, RoIAlign 'overlays' the fixed-size grid directly over the variably-sized proposed region, and for each 'cell' within the grid, samples four points. Each sample's value is calculated using bilinear interpolation, essentially a 'weighted average' based on the sample's spatial position and the neighbouring (surrounding) pixels' values. Note that 'pixels' here refer to values from a feature map, not the original image. The cell's final value is chosen as the maximum of the four samples (He et al., 2017; Sortino et al., 2023a). He et al. (2017) note that the use of RoIAlign over ROI pooling improves performance by up to 50%.

3.3.2.4 | Network Outputs

After the fixed-size regions have been created, Mask R-CNN splits into 2 branches, the first responsible for generating object classification and bounding box regression, and the second 'Mask Branch' allowing the model to perform instance segmentation and predict per-pixel masks for each detected object.

In the first branch, the fixed-size region feature maps are first flattened and passed through a series (normally 2 layers) of fully-connected layers. The fully-connected layers output a tensor of size $C + 1$ per ROI, where C is the number of classes, and the $+1$ accounts for the 'background' class. The first output, object classification, is computed as a softmax activation on this output tensor. This converts the tensor into a list of probabilities adding up to 1, where each value corresponds to one of the classes. The class of an object in a given ROI is determined as the class with the highest probability. The inclusion of the background class here is key as, if a region scores highest as a background, it is dropped and will not 'default' to any of the other classes. Furthermore, for each ROI, the fully-connected layers also output another 4 values, representing adjustments to the bounding box coordinates, so that the bounding box will fit better around the object. This output is referred to as bounding box regression, and is computed in parallel to the object classification output (He et al., 2017; Ren et al., 2017).

Table 3.1: High-level feature comparison of existing source finders, based on deep learning models, including ASGARD (Chapter 5), the implementation of Mask R-CNN in this work. See Section 3.3 for details.

Tool	Features			Works With			Trained With
	Detection	Classification	Segmentation	Sources	Galaxies	Sidelobes	
ASGARD	✓	✓	✓	✓	✓	✓	Real radio data from multiple surveys
ConvoSource	✓		✓	✓	✓		Simulated radio data from SKA Data Science Challenge 1
ClaRAN	✓	✓			✓		Real radio (FIRST survey, Radio Galaxy Zoo Data Release 1) and infrared data (WISE survey)
DeepSource	✓			✓			Simulated MeerKAT radio data

The second branch, known as the Mask Branch, runs in parallel to the first, and is responsible for generating the per-pixel mask for every detected object, which is what sets Mask R-CNN apart. The mask branch consists of a FCN (Long et al., 2015), chosen for its ability to retain spatial information, with a series of convolutional layers, each producing 256 features. Some of these convolutional layers upsample (deconvolution) the fixed-size ROI feature maps in order to increase their resolution, and thus the segmentation's performance. This FCN computes a per-pixel binary mask for every ROI, for each class, i.e. it will generate $C + 1$, or K, masks per ROI. Which mask is in fact used and presented as the model's prediction is determined by the aforementioned object classification output (He et al., 2017).

Table 3.1 shows a high-level comparison of the features of the models described in Sections 3.3.1 and 3.3.2, as well as how they were trained, and what objects they are trained to detect, segment, and classify.

3.3.3 | YOLO

You Only Look Once (YOLO) (Redmon et al., 2016) is a one-stage detector, meaning its architecture does not feature an RPN (see Section 3.3.2.2), and instead directly attempts to predict bounding box coordinates and their confidence score. This makes the model more computationally efficient and allows it to execute faster, however typically comes at a cost of performance (e.g. accuracy). This makes YOLO well suited for real-time applications. In this case, the poorer performance is in fact implied by YOLO's architecture.

YOLO starts by resizing the image to a fixed, pre-defined, size, and then dividing the image into an $S \times S$ grid, with S set to 7 in the original implementation. Here, each cell in the grid can only represent one class, which of course leads to inaccuracies if multiple objects are contained within the same cell. Furthermore, this can bias the model as

it might learn certain objects only in a specific context within a given cell. For each of these cells, the model predicts a number of bounding boxes, B (originally set to 2). Then, for each of these bounding boxes, the model predicts their coordinates, together with a confidence score, quantifying the product of how confident the model is that the bounding box contains an object and how well it thinks the box overlaps the object. As mentioned earlier, YOLO will also predict the class probabilities of each cell, for that entire cell, in the grid. Next, NMS is applied, which filters out bounding boxes with low confidence scores, as well as overlapping ones (NMS is also explained in Section 3.3.2.2). Finally, the remaining bounding boxes are considered the model's final output, and can be applied and overlaid onto the image. At a lower level, the YOLO architecture is made up of 24 sequential convolutional layers and 2 fully-connected layers (Redmon et al., 2016; Sortino et al., 2023a).

YOLO represents the first step in a, now, long series of versions and incremental improvements, with YOLO not being able to classify multiple objects within the same cell differently, or even detect them if there are more than B (two, originally), and only considering one aspect ratio. YOLO9000 (Redmon and Farhadi, 2017) is the first such increment, introducing Batch Normalisation (see Section 2.2.1.2) and anchors, allowing for it to detect objects at multiple scales and resolutions. YOLOv3 (Redmon and Farhadi, 2018) then incorporates a multi-scale feature extractor based off of FPNs (See Section 3.3.2.1). YOLOv4 (Bochkovskiy et al., 2020) replaces the FPN in YOLOv3 with a Path Aggregation Network (PANet) (Liu et al., 2018b) module. YOLOv7 (Wang et al., 2022) increments on this yet again by modifying the architecture to enable layer fusion, resulting in more efficient paths for the gradients during training, which ultimately improves the performance (Sortino et al., 2023a).

It is important to highlight that while all of these versions of YOLO detect objects within images, draw bounding boxes around them, and classify them, they do not perform segmentation, so they cannot be considered instance segmentation algorithms. YOLOv8 (Jocher et al., 2023) changes this by introducing functionality for the segmentation of detected objects. Finally, as of August 2024, YOLOv10 (Wang et al., 2024a) represents the latest development in the series of YOLO variants, removing NMS and instead making use of "Consistent Dual Assignment".

3.3.4 | SOLOv2

Segmenting Objects by LOcations (SOLO)v2 (Wang et al., 2020b) is a one-stage instance segmentation model. It is similar to YOLO (see Section 3.3.3) in that they are both one-stage models, however separates itself from both YOLO and Mask R-CNN (see Sec-

tion 3.3.2) as its architecture does not utilise bounding boxes, region proposals, or anchors, but instead directly classifies pixels as belonging to an object instance.

Similarly to YOLO, SOLOv2 starts by dividing the image into an $S \times S$ grid. Next, for feature extraction, SOLOv2 strongly resembles Mask R-CNN, as it utilises an FPN (see Section 3.3.2.1) to extract features at different scales. This is followed by one of the main innovations introduced by SOLOv2, "Dynamic Instance Segmentation".

In SOLO (v1) (Wang et al., 2020a), the architecture would predict masks using the same convolutional layer for all cells within the grid. SOLOv2 refines this process by employing "Dynamic Convolution". In SOLOv2, the features extracted by the FPN branch into the 'Kernel Branch' and 'Feature Branch'. The Kernel Branch receives the features extracted by the FPN at each scale, passes them through a series of convolutional layers, and generates a convolutional kernel to be used to detect a specific object. Additionally, information about a cell's location is encoded using CoordConv (Liu et al., 2018a). The Feature Branch receives the very same features from the FPN, however applies 'feature map fusion' (inspired by Kirillov et al., 2019) to merge them into a single feature map. 'Dynamic Convolution' is applied to combine the outputs of the 'Kernel Branch' and 'Feature Branch' and generate object masks. This involves convolving the specifically generated kernel from the kernel branch with the output of the feature branch, for each cell it believes contains an object. This is both more efficient in terms of system resources, as masks are only generated for cells where objects are detected, and results in better segmentation, as a kernel is generated specifically for that cell (Wang et al., 2020b).

Afterwards, another innovation of SOLOv2 is carried out, Matrix NMS, inspired by Soft-NMS (Bodla et al., 2017). The purpose is the same as with regular NMS (such as in Section 3.3.2.2), to discard any suboptimal or overlapping, likely duplicate, masks. Soft-NMS improves upon this by progressively decaying the confidence score of overlapping objects, rather than discarding them outright. Bodla et al. (2017) note how their method improves the performance for animals in a herd by at least 3%. This suggests that this technique may improve the detection performance on images with multiple, partially overlapping, astronomical objects. Matrix NMS accomplishes many of the same improvements achieved by Soft-NMS, however in a one-shot method that calculates, and then compares, the IoU of each mask pair using parallel matrix operations, rather than sequentially. Wang et al. (2020b) claim speeds of 500 masks in less than 1 ms.

Similarly to Mask R-CNN (see Section 3.3.2.4), SOLOv2's output is obtained from two parallel branches, the mask branch, explained above, and the classification branch, an FCN that outputs a probability per class for each cell in the grid (Wang et al., 2020a).

3.3.5 | Transformers

Transformers (Vaswani et al., 2017) were originally developed and applied for use in Natural Language Processing (NLP). Until their inception, RNNs were the state of the art for working with NLP use cases. RNNs were the logical solution to NLP, and other sequence based, applications as their structure takes into consideration previous inputs, or tokens, when running, which is necessary to capture the context when working with, say, sentences. Another advantage of RNNs is their ability to work with arbitrarily long sequences. Their sequential structure, however, is what leads to one of their greatest disadvantages, being their inability to be parallelised, making it intractable to work with growing datasets (Rumelhart et al., 1986). Yet another disadvantage is the vanishing gradient problem, leading to context built over a longer range to potentially be lost, however this is improved in variations of RNNs, such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014).

The architecture of Transformers differs completely to that of RNNs. For instance, the input is not passed through a NN token by token, where every step must wait for the previous to finish before it can be computed, instead, Transformers encode the input in one pass. The architecture of Transformers is based on 3 main pillars:

- Positional Encoding: this assigns an ordinal number to every tokenised element in the sequence, which allows transformers to retain information about the order of the data, despite the data being processed in one go.
- Attention: the model learns to choose which elements, e.g. words, of the input sequence to pay attention to, or attend to, at every step of producing the output.
- Self-Attention: this is potentially the most important advancement introduced by transformers, as the model applies attention on the input itself, to determine which elements are linked together, depend on each other, and refer to each other, to achieve a greater 'understanding' of it, and most of all to grasp the context. This mechanism theoretically outperforms that used by RNNs, as its memory is technically infinite, limited only by what can fit in computer memory, whereas RNNs will have little or no 'recollection' of input or output elements after a few steps, and while its variants improve on this issue, it is still present to some degree.

The architecture of Transformers is based on an Encoder-Decoder setup. The input sequence is first passed through an embedding layer, producing a vector representation of every element in the sequence, and then including positional encoding. The Encoder

then converts this embedding into a continuous representation, capturing the entirety of the input sequence, this includes self-attention, allowing the model to associate certain elements, e.g. words, with others in the sequence. The decoder then, at each step, takes the entire input (from the encoder's output) as well as the entire output sequence generated till that point (going through a similar embedding and encoding process), and sequentially generates the next element in the sequence, with the next element chosen by a linear classifier with a softmax activation.

3.3.5.1 | ViT

One of the pioneering works that successfully applied Transformers to images was Vision Transformer (ViT) (Dosovitskiy et al., 2021). ViT accomplishes image classification by chopping up input images into patches, then, using a linear projection, converts, or flattens, each patch into a vector representation, referred to as "patch embeddings" (Dosovitskiy et al., 2021). Through this process, images are effectively converted into tokens, and thus, the rest of the process then aligns with that described for transformers applied to NLP, a positional encoding is added to the embeddings, and the vectors are passed to an Encoder, the output of which, in the case of ViT, is fed into a classification layer.

According to Dosovitskiy et al. (2021), ViT outperforms the state of the art CNNs, at the time of writing. Broadly speaking, given their architectures and kernels, CNNs first look at images locally and progressively widen their field of view. Transformers have an advantage as, owing to their self-attention mechanism, can detect dependencies over the entire span of the image, meaning they consider a much larger field of view from the very first layer.

3.3.5.2 | DETR

DEtection TRansformer (DETR) (Carion et al., 2020) is yet another model that was among the first to apply Transformers to images. Similarly to ViT, DETR starts by converting the image into a series of flattened patches. More specifically, DETR starts with a CNN backbone, usually a ResNet (such as in Mask R-CNN), to extract features, after which positional encoding is added. These tokenised patches are then passed onto the transformer component of DETR.The encoder part of the transformer works normally, in that it applies self-attention, allowing for the model to learn the interactions and dependencies between different patches (parts of the input image).

The decoder shares a similar architecture to that of the encoder, however introduces 'object queries' as additional inputs, which represent objects the model is set to de-

tect. Furthermore, after applying self-attention, the decoder also applies cross-attention, which allows the model to 'link' the aforementioned object queries with the encoder's output, thus resulting in 'understanding' of the objects' classes and positions within the image. The model finally outputs a specified number of predictions, each consisting of a bounding box and class label, which can be 'no object'.

During training, loss is calculated as the sum of classification and bounding box loss. This is calculated using 'bipartite matching', which pairs every prediction with a bounding box from the ground truth, using the Hungarian algorithm to accomplish this pairing efficiently. This matching algorithm is ultimately what enables DETR to be trained in an end-to-end fashion (Carion et al., 2020; Sortino et al., 2023a).

DETR can be extended to also perform instance segmentation. This is achieved by including a CNN to generate a binary mask per object. When extended to perform instance segmentation, the calculation of the loss is also extended to include the mask loss.

3.4 | Conclusions

This chapter gives an overview of techniques that are applied to classification, semantic segmentation, and instance segmentation tasks, both in general and in astronomy contexts.

Section 3.1 starts with a look at solutions for classification problems, i.e. taking an image as input and classifying, or labelling, it. A set of conventional methods are presented first in Section 3.1.1, namely visual inspection, arc-finders, and ML with preselected features (such as SVMs or HOG).

Next, Section 3.1.2 presents a series of NN-based classifiers, particularly CNN-based ones. CAS Swinburne, LensFlow, LensFinder, and LASTRO EPFL are presented as CNN-based architectures primarily made up of convolutional layers, activation functions (mainly ReLU and ELU), and pooling layers, together with fully-connected layers, dropout, and a sigmoid (or softmax, in the case of multi-class classification) activation function for the output. CMU DeepLens and WSI-Net are also presented, which feature ResNets, i.e. these networks feature skip (or shortcut) connections which connect the output of a layer with that of another layer deeper within the network. This is particularly beneficial during back propagation, as they mitigate the vanishing gradient problem by providing a shorter route from the output to the earlier layers (i.e. those closer to the input).

In this thesis, namely in Chapter 4, all the architectures presented in Section 3.1.2

have been applied to the task of classifying a given image as containing gravitational lensing, or not. Contrastingly, none of the conventional techniques mentioned in Section 3.1.1 were applied. As mentioned in Section 3.1.2, all the CNN-based solutions outperformed the conventional techniques in the 'Gravitational Lens Finding Challenge' (Metcalf, R. B. et al., 2019). Furthermore, a technique such as visual inspection (Section 3.1.1.1) is clearly not suitable for the scale and volume of data that is expected from upcoming surveys.

Similarly, Section 3.2 presents semantic segmentation tasks, where the aim is to classify every pixel within a given image as belonging to a given class, or not. Fig. 3.8 graphically shows a typical expected output for such tasks. U-Net, a deep, fully convolutional, network, is presented as one potential solution to such tasks in Section 3.2.1. Similarly, Section 3.2.2 presents U-Net++, an upgraded version of U-Net.

Finally, Section 3.3 introduces instance segmentation tasks, where the goal is to detect objects of interest within a given image (and draw a bounding box around them), produce pixel-accurate masks for each of the detected objects, even differentiating between partially overlapping objects, and classifying/labelling each of the detected objects. A sample expected output for such tasks is shown graphically in Fig. 3.8.

Next, Section 3.3.1 goes over some of the existing solutions being applied to astronomy at the time, however none fully accomplish instance segmentation, as shown in Table 3.1. Section 3.3.2 then presents Mask R-CNN, an all-in-one solution that accomplishes all the requirements of instance segmentation and has shown success in other domains. Sections 3.3.3 and 3.3.4 present the YOLO and SOLOv2 architectures, respectively.

Section 3.3.5 gives a general overview of the architecture of transformers due to their recent development and rise in prominence in the state of the art for working with images. DETR is one such architecture that can be extended to accomplish instance segmentation, as described in Section 3.3.5.2.

4

Detection of Strong Gravitational Lensing

Some of the content in this chapter, Chapter 4, incorporates concepts discussed in Magro et al. (2021).

Einstein (1915) was reportedly the first to theorise, amongst other things, a set of equations that correctly calculated the effects gravity had on the path of light (Will, 2014). This development of how light bends was further expanded upon by Chwolson (1924) and Einstein (1936), to the extent that it might occur that a massive astronomical object would bend and focus the light coming from an object behind it, effectively functioning as a lens. This phenomenon was first confirmed when Walsh et al. (1979) observed a gravitational lens. A gravitational lens is a system in which an observer, a foreground astronomical object, and a background astronomical object, are aligned such that the gravitational field of the foreground object lenses the light from the background object towards the observer (Metcalf, R. B. et al., 2019).

Gravitational lenses can be categorised into several classes, some of which are strong lensing, weak lensing, and microlensing. While weak lensing and microlensing do have their applications, such as the potential for weak lensing to be used to build a three-dimensional map of matter in the universe, or for microlensing to detect planets with a mass equal to that of the Earth (Wambsganss, 1998) or to study the structure of quasars (Metcalf, R. B. et al., 2019), this thesis will focus on the detection of strong lensing. Strong lenses have been used as natural telescopes to, otherwise obscured, astronomical objects. Furthermore, they have been used to study the distribution of dark matter in galaxies and clusters, and to measure cosmological parameters, such as the Hubble constant (i.e. the rate at which the universe expands) (Metcalf, R. B. et al., 2019).

With the state of observational technology, and the current detection capabilities,

gravitational lensing is a rather rare phenomenon. In 1999, Kochanek et al. mention 47 known gravitational lenses. As of September 2023, the CfA-Arizona Space Telescope Lens Survey (CASTLeS) website lists 101 Multiply Imaged Systems (Kochanek et al.), 83 of which Kochanek et al. claim they are confident are lenses, and a further 17 Binary Quasars, 5 of which Kochanek et al. claim they are confident are lenses. Other surveys such as the Cosmic Lens All-Sky Survey (CLASS) (Myers et al., 2003) (which used the Very Large Array (VLA) and found 16 new gravitational lens systems), Sloan Lens ACS (SLACS) (Bolton et al., 2006) (which, initially, found 19 new gravitational lenses with the Advanced Camera for Surveys (ACS) on the Hubble Space Telescope (HST)), Strong Lensing Legacy Survey (SL2S) (Cabanac, R. A. et al., 2007) (which initially found 43 candidate strong lenses from the Canada–France–Hawaii Telescope Legacy Survey (CFHTLS)), and Herschel Astrophysical Terahertz Large Area Survey (H-ATLAS) (Negrello et al., 2016) (which found 80 candidate lensed galaxies, 20 of which have been confirmed to be strong lensed) have also contributed to the discovery of gravitational lenses. Metcalf, R. B. et al. (2019) state that across many datasets, less than a thousand lenses have been found.

This value is expected to increase by orders of magnitudes with the potentially discoverable lenses which are expected to be captured by experiments such as the SKA[1] (Blake et al., 2004), Large Synoptic Survey Telescope (LSST) (LSST Science Collaboration et al., 2009), Dark Energy Survey (DES)[2] (The Dark Energy Survey Collaboration, 2005), Kilo-Degree Survey (KiDS)[3] (de Jong et al., 2013), Euclid[4] (Laureijs et al., 2011) by the European Space Agency (ESA), and the Nancy Grace Roman Space Telescope[5] (Dressler et al., 2012) by NASA (Magro et al., 2021; Metcalf, R. B. et al., 2019). The LensPop[6] model predicts that the DES, LSST, and Euclid surveys could potentially discover 2,300, 120,000, and 280,000 lenses, respectively (Collett, 2015). The same model predicted that the Roman Space Telescope will be able to detect 16,778 strong lenses (Weiner et al., 2020). When considering these values, manual visual inspection by trained astronomers, as explained in Section 3.1.1.1, is clearly not a viable solution, and the necessity for efficient, automated, and accurate solutions becomes very evident. The purpose of this work is to address that need.

The 'Gravitational Lens Finding Challenge 1.0'[7], launched in 2019, studies the de-

[1] https://www.skatelescope.org/
[2] https://www.darkenergysurvey.org/
[3] http://www.astro-wise.org/projects/KIDS/
[4] http://sci.esa.int/euclid/
[5] https://roman.gsfc.nasa.gov/
[6] https://github.com/tcollett/LensPop
[7] Previously available at http://metcalf1.difa.unibo.it/blf-portal/gg_challenge.html (Last

tection efficiency of strongly lensed systems (Metcalf, R. B. et al., 2019). The challenge evaluates the capability of several techniques, many of which are ML-based, to detect whether a given image is a gravitational lens, or not. This work aims to further this study by introducing newer ML techniques, primarily CNNs, and comparing the performances achieved to those reported by Metcalf, R. B. et al. (2019).

Furthermore, this work aims to develop a framework which facilitates the application of NNs to the Gravitational Lensing problem. This will be achieved by providing the user with tools tailored for loading large astronomical datasets, applying pre-processing to them, applying image augmentation during training, easily training defined architectures, evaluating trained models, and peering into the inner workings of the trained models, ultimately allowing the user to focus on defining and developing the perfect architecture, or even applying a pre-defined one, to their problem. Moreover, this framework should be highly adaptable to classification problems on astronomical data in general, and not only the gravitational lensing problem.

The following section, Section 4.1 describes the structure and contents of the datasets provided for the 'Gravitational Lens Finding Challenge 1.0'. Section 4.2 goes on to discuss additional image augmentation techniques utilised to 'expand' on this dataset. Section 4.3 describes Lens EXtrActor CaTania University of Malta (LEXACTUM), the developed framework, and its features. The various neural network architectures implemented and included within LEXACTUM have been discussed in detail in Section 3.1.2. Section 4.4 explains the evaluation metrics produced by LEXACTUM, how the models were evaluated, presents the results achieved, and compares these to those obtained in other works. Finally, concluding remarks and possible future work and improvements are discussed in Section 4.6.

4.1 | The Datasets

The datasets used for this work were those provided for the 'Gravitational Lens Finding Challenge 1.0'[7 (on page 58)]. Each dataset is made up of 100,000 simulated images, 20,000 of which are meant for training the models, while the other 80,000 are meant for evaluating the performance of the trained models.

The 'Space' dataset is made up of simulated single channel images, mimicking data "from a satellite survey such as *Euclid*" (Metcalf, R. B. et al., 2019). The single channel of data was based on the band SDSS i (near infrared), corresponding to an effective wave-

Accessed in June 2021); currently offline, should be made accessible again at a new domain.

length (Schneider et al., 1983) of 7,625 Å (762.5 nm)[8] (Fukugita et al., 1996; Stoughton et al., 2002; York et al., 2000). The images have a resolution of 101×101 pixels, and each pixel has a size of 0.1 arcsec. 40% of the generated images feature gravitational lensing.

The second dataset is referred to as the 'Ground' dataset. 85% of its images were simulated to mimic those from a ground based survey, such as the KiDS survey (de Jong et al., 2013), 50% of which featuring lensing. The remaining 15% of the images were real images, obtained from a preliminary release of bright galaxies from the KiDS survey. In the case of the real images, lenses were added to 50% of the images. Each image contains 4 channels of data, based on the bands SDSS u, g, r, i. These correspond to ultraviolet, with an effective wavelength of 3,543 Å (354.3 nm), (blue-)green at 4,770 Å (477.0 nm), red at 6,231 Å (623.1 nm), and near infrared at 7,625 Å (762.5 nm)[8 (on page 60)] (Fukugita et al., 1996; Stoughton et al., 2002; York et al., 2000). These images have the same 101×101-pixel resolution, however the pixel size for this dataset is 0.2 arcsec (Metcalf, R. B. et al., 2019).

Fig. 4.1 shows an example of an image containing gravitational lensing, and another which doesn't, from the Space set. Similarly, Fig. 4.2 shows the two cases from the Ground set. ZScale Normalisation has been applied to these images.

4.2 | Image Augmentation

Image Augmentation is the process of performing transformations on images in the dataset before passing them on to the neural network to **train** on; Image Augmentation is only performed on the training set, during training. This adds diversity to the data, allowing for models to train for longer (a greater number of epochs) without overfitting. It also helps the model generalise better by showing the same object in different positions (by shifts/translations), orientations (by rotations), and sizes (by scaling). This is especially important in this scenario, as the 20,000 images provided for training the model are significant, but not overly abundant, especially when compared to the 80,000 images in the test/evaluation set that the model must learn to generalise for.

To this end, an Image Augmentation component was implemented in LEXACTUM, utilising the 'imgaug' library[9]. The following 9 transformations were defined, of which a random amount are picked to be applied to each image before it is given to the model during training (each image will have a different, random, set of transformations applied to it every epoch):

[8] Exact values cited are from `https://skyserver.sdss.org/dr1/en/proj/advanced/color/sdssfilters.asp`

[9] `https://pypi.org/project/imgaug/`

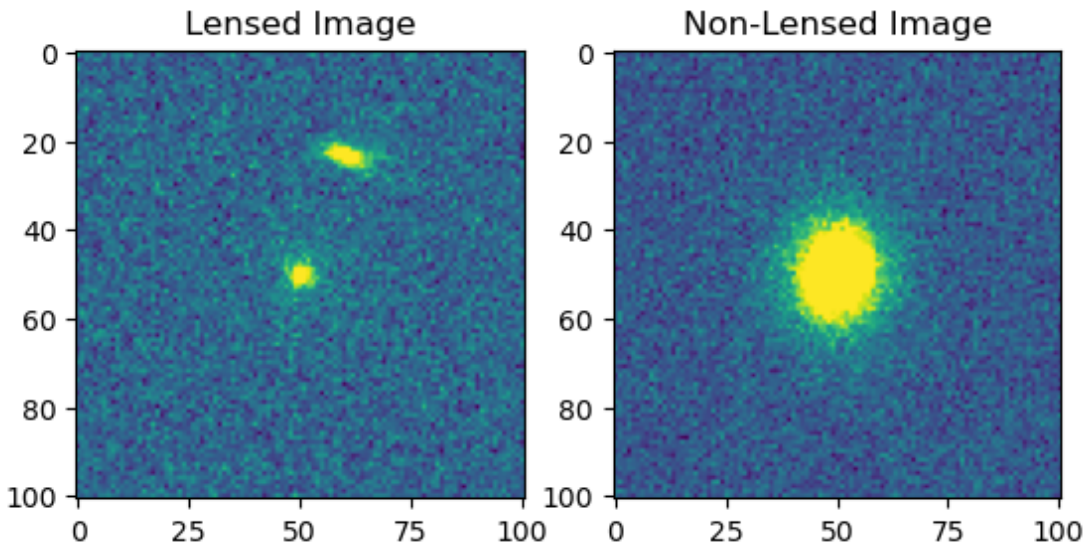

Figure 4.1: Sample lensed and non-lensed images from the Space set. The image on the left is a random lensed image from the Space set, whereas the image on the right does not contain lensing. ZScale Normalisation has been applied to both images. Reproduced from Magro et al. (2021); Metcalf, R. B. et al. (2019).

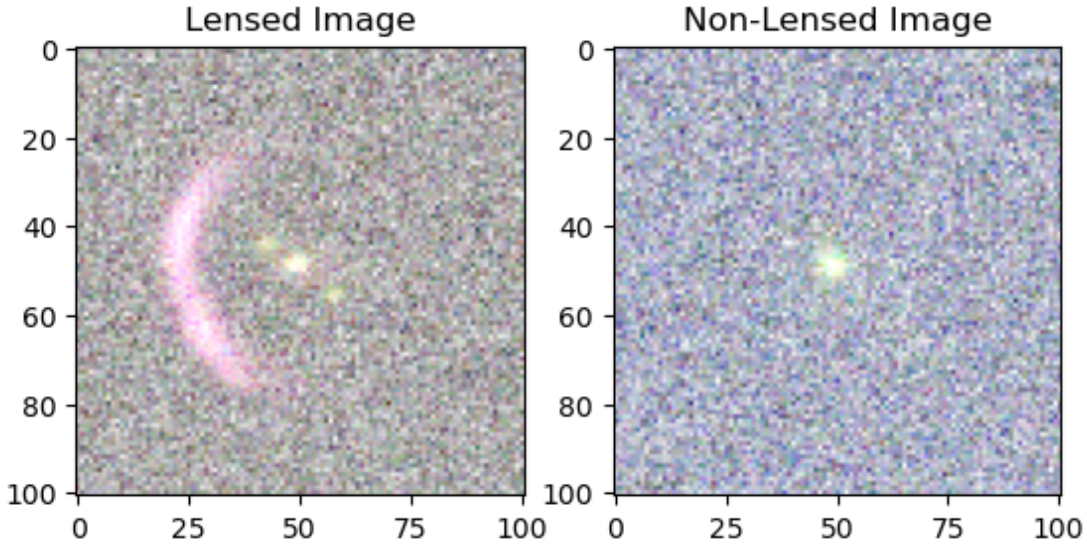

Figure 4.2: Sample lensed and non-lensed images from the Ground set. The image on the left is a random lensed image from the Ground set, whereas the image on the right does not contain lensing. ZScale Normalisation has been applied to both images. Reproduced from Magro et al. (2021); Metcalf, R. B. et al. (2019).

- A Vertical and/or Horizontal flipping of the image;
- A 90°, 180°, or 270° Rotation of the image;
- A Translation of [-10%, 10%] of the image along the X and/or Y axes;
- A Scaling of [0.75, 1] of the image along the X and/or Y axes;
- A Shearing of [-20%, 20%] of the image along the X and/or Y axes.

Some of the techniques described in Section 3.1.2, namely CAS Swinburne, LASTRO EPFL, CMU DeepLens, and WSI-Net, utilise image augmentation during training, however these are generally limited to flips and rotations (Jacobs et al., 2017; Lanusse et al., 2018; Ni et al., 2019; Schaefer et al., 2018). LEXACTUM's framework enables any architecture added to it to have the full list of transformations described readily available to it.

4.3 | Methodology

The framework developed in this work has been named LEXACTUM. All the code written for LEXACTUM is publicly available on a GitHub repository[10], under the GNU General Public License v3.0[11].

The first of its main features is the Image Augmentation component, which makes the techniques described in Section 4.2 readily available to all the models implemented within the LEXACTUM framework, including any models added by future users. This feature enables NNs to train for a greater number of epochs without overfitting, ultimately improving performance, as is presented in Section 4.4.2. This feature can also be toggled on or off.

Apart from image augmentation during training, all images are normalised using ZScale (NOAO, 1997). As with other components, the normalisation method can be easily swapped out for other techniques, or disabled altogether. LEXACTUM also uses a custom 'Data Generator', which allows the CPU to load and preprocess (including image augmentation during training) the next batch of images, while the GPU trains on the last batch of images. This makes the processing of large datasets, which would otherwise not fit in memory, possible, as well as providing a slight boost in performance (execution time) since operations are being carried out in parallel.

[10] `https://github.com/DanielMagro97/LEXACTUM`

[11] `https://www.gnu.org/licenses/gpl-3.0.html`, `https://github.com/DanielMagro97/LEXACTUM/blob/main/LICENSE`

Moreover, LEXACTUM saves trained models to disk, for use in future evaluation or inference. The functionality for loading these trained models from disk is also provided. Furthermore, LEXACTUM includes a 'results' package, which scores the trained models and calculates several metrics, described in detail in Section 4.4.1. This makes it very easy to quickly evaluate a newly developed model for comparison with other solutions, in a standardised way for reliable and repeatable evaluation.

Another feature is the modularity of the code, allowing for the rather easy development of new models, along with very easy integration into the framework. Other features include the ability to set parameters from the command line, which avoids the need for the user to delve into the code simply to change the configuration. Examples of such parameters are the dataset path (from which to train or evaluate the model), whether to train a model or load one from disk, the name of the model (to train or load), the number of epochs to train for, the batch size, and whether to use image augmentation during training or not. These are explained in further detail in Table B.1 in Appendix B.

Finally, the 'visualise features' component was created, which allows for the viewing of the feature maps at every convolutional layer that a trained model is 'looking at' during inference, further allowing the user to understand the inner workings of the model, something which can help in demystifying how the model is working, however is otherwise very often neglected, leading to NNs being thought of as a 'black box'.

The value of these features lies in the fact that their inclusion within the LEXACTUM framework makes them immediately available for any developers of new NN architectures. Firstly, having dataset loading (via a custom data generator), image preprocessing, image augmentation, trained model saving and loading, evaluation, and network visualisation makes it so that the user can focus all of their time and effort solely on developing the network architecture, and not 'boilerplate' code. In fact, it would be reasonable to state that as much as 75% of time spent on implementation would be dedicated towards the aforementioned features, whereas only 25% would be dedicated towards the implementation of the model. This is substantiated by the code written to implement LEXACTUM, its features, and models. The most complex model implemented (based on the number of convolutional layers), CMU DeepLens, having around 45 convolutional layers, took around 75 lines of code to implement. The aforementioned LEXACTUM features, on the other hand, took close to 400 lines of code to implement (both excluding empty lines or inline comments).

This time save allows researchers to instead devote more time and concentrate more on more scientifically impactful tasks, such as designing the architectures of new networks. The impact of one of these features in particular, image augmentation, will be

highlighted in Section 4.4.2.4. Additionally, removing the need for users to implement these features themselves greatly diminishes the possibility of any bugs being introduced, as they are provided out of the box in a framework which has been thoroughly tested and optimised. Moreover, some features which developers may often forgo, such as the neural network visualisation component, are made available, without the need for any further intervention. Finally, such a framework allows all its users to train and evaluate their models in a perfectly repeatable manner, which is key for the scientific comparison of one model to another.

Every architecture described in Section 3.1.2 was implemented in LEXACTUM. All of these models were then trained from scratch, i.e. transfer learning (models pre-trained from other datasets and fine-tuned with this dataset) was not employed, and weights were randomly initialised and learnt purely from the training set. As a preprocessing step, ZScale normalisation was applied to all images. Moreover, image augmentation was applied during training, as specified in Section 4.2. Initially, all models were trained for 5 or 10, 25 or 50, 100, and 250 epochs. If a model trained for 250 epochs performed well, and did not show signs of overfitting (judging by the trend of the model's loss and accuracy on the validation set over the final epochs), it was trained further until 500, or even 1,000, epochs. This process was repeated for both the Space and Ground datasets. All these models were trained on one of four Slurm[12] nodes, each of which was allocated 8 CPU cores, 48 GB of RAM, and an NVIDIA V100 Graphics Processing Unit (GPU) with 16 GB of VRAM. The weights files for these trained models have been made available on Zenodo[13] (Magro et al., 2020) and drUM[14] (Magro et al., 2024).

4.4 | Evaluation

4.4.1 | Metrics

Metcalf, R. B. et al. (2019) use the AUC, TPR_0, and TPR_{10}metrics to evaluate and compare models in their work.

The AUC has been defined in Section 2.2.3.1. Metcalf, R. B. et al. (2019) define the TPR_0 as the highest TPR a model is capable of achieving, while maintaining the FPR equal to 0. In other words, the computation of this metric starts with a very strict threshold for considering a confidence score for an image as a positive detection. This will make it such that only images the model is virtually certain are lenses, are considered

[12] `https://slurm.schedmd.com/overview.html`
[13] `https://doi.org/10.5281/zenodo.4299924`
[14] `https://drum.um.edu.mt/articles/dataset/LEXACTUM_trained_model_weights/26236664`

lenses. This will rule out false detections (FPR=0), however will also keep the TPR very low. The threshold is then progressively, at a very slow rate (by 0.001 in LEXACTUM's implementation), lowered, such that the classification of which instances are positive is less strict.Considering the difficulty of achieving a TPR_0 which is not 0, Metcalf, R. B. et al. (2019) define the TPR_{10}, which is similarly the highest TPR achievable, while not classifying more than 10 false positives (Metcalf, R. B. et al., 2019).

Finally, during evaluation, LEXACTUM also measures the total inference time for each trained model to classify the entire test set, i.e. how long each trained model took for inference on the entire test set is recorded and reported. The average inference time of each model, i.e. the average time needed for the model to make a prediction/classification on one image, was then calculated by dividing this length of time by the number of images in the test set. The inference times for the same model trained for different numbers of epochs were averaged out, as these are still the same architecture, meaning one inference time is reported for each model architecture. Furthermore, executions where the inference time was significantly different from others for the same architecture (outliers) were ignored, and not included in the average.

4.4.2 | Results

4.4.2.1 | Space Set Results

The architectures described in Section 3.1.2 were trained and evaluated using the LEXACTUM framework on the Space dataset, as outlined in Section 4.3. The models' performances, at each of the training epoch lengths, were presented in Table 4.1. The metrics reported are those detailed in Section 4.4.1. CMU DeepLens achieved the best TPR of 0.8738 when trained for just 25 epochs. Similarly, the lowest FPR was also achieved with a low number of epochs, that of 0.0042 by Lastro EPFL when trained for 5 epochs. CMU DeepLens was once again the best performer, this time attaining the best AUC of 0.9343, when trained for 500 epochs. In Metcalf, R. B. et al. (2019)'s paper, the best AUC for the Space set was 0.93 by LASTRO EPFL, whereas the implementation of CMU DeepLens scored 0.92. The highest TPR_0 was 0.2411, accomplished by CAS Swinburne when trained for 50 epochs. In Metcalf, R. B. et al. (2019)'s paper, the highest TPR_0 for the Space set was 0.22, by CMU DeepLens. WSI-Net accomplished the best TPR_{10} of 0.4211, when trained for 250 epochs. In Metcalf, R. B. et al. (2019)'s paper, the highest TPR_{10} for the Space set across all models was 0.36, by GAMOCLASS, another CNN-based solution. This is a very interesting finding, as a ResNet-based network which had not been included in the Metcalf, R. B. et al. (2019) paper, achieved a

Table 4.1: Table showing the TPR, FPR, AUC, TPR_0, TPR_{10}, and average execution time (Section 4.4.1) for 6 different models, as described in Section 3.1.2, trained for a various number of epochs on the Space dataset. Columns marked with an * indicate the score achieved by the model in Metcalf, R. B. et al. (2019). Values in these columns marked in green indicate better performance compared to the implementations in LEXACTUM, whereas values in red indicate worse performance. Reproduced from Magro et al. (2021).

Model Name	No. of Training Epochs	TPR	FPR	AUC	TPR_0	TPR_{10}	AUC*	TPR_0*	TPR_{10}*	Avg. Execution Time per Image (seconds)
CAS Swinburne	5	0.5250	0.0603	0.8489	0.1531	0.1861				
	10	0.5517	0.1077	0.8171	0.1054	0.1509				
	25	0.7221	0.1178	0.8870	0.0000	0.2705				
	50	0.6252	0.0461	0.8894	0.2411	0.3000	N/A			0.0124
	75	0.6503	0.0474	0.8963	0.0000	0.3221				
	100	0.6604	0.0591	0.8915	0.0000	0.3016				
	500	0.6551	0.0295	0.9086	0.0000	0.3602				
Lastro EPFL	5	0.3507	0.0042	0.8641	0.1539	0.2112				
	10	0.7302	0.3543	0.7825	0.1894	0.2455	0.93	0.00	0.08	0.0061
	50	0.6650	0.0287	0.9132	0.2107	0.3823				
	250	0.7937	0.0687	0.9322	0.0000	0.2268				
CMU Deeplens	5	0.6056	0.1539	0.7984	0.0000	0.1206				
	10	0.8268	0.2880	0.8710	0.0000	0.2309				
	25	0.8738	0.2726	0.9113	0.0000	0.0000				
	50	0.7570	0.0628	0.9243	0.0000	0.4073				
	100	0.8170	0.1321	0.9226	0.0000	0.0000	0.92	0.22	0.29	0.0061
	250	0.7592	0.0436	0.9291	0.0000	0.0000				
	500	0.7952	0.0626	0.9343	0.0000	0.0000				
	1000	0.8611	0.1634	0.9303	0.0000	0.0000				
WSI-Net	5	0.7132	0.2955	0.7935	0.0000	0.0000				
	10	0.5437	0.0187	0.8867	0.1799	0.2934				
	50	0.7888	0.1194	0.9115	0.0000	0.0000	N/A			0.0055
	100	0.7348	0.0624	0.9069	0.0000	0.3976				
	250	0.7255	0.0531	0.9083	0.0000	0.4211				
Lens Flow	5	0.6508	0.1520	0.8389	0.0728	0.1260				
	25	0.6431	0.0726	0.8799	0.1903	0.2704	N/A			0.0054
	100	0.6780	0.0636	0.8963	0.0000	0.3379				
	250	0.7384	0.0889	0.9046	0.0000	0.3632				
Lens Finder	5	0.4915	0.1001	0.8038	0.0885	0.1056				
	25	0.6203	0.0663	0.8739	0.2103	0.2395	N/A			0.0197
	100	0.6912	0.0855	0.8857	0.0000	0.2721				
	250	0.7651	0.1062	0.9056	0.0000	0.3739				

significantly higher score than that in the paper.

4.4.2.2 | Ground Set Results

Similarly to Section 4.4.2.1, the architectures described in Section 3.1.2 were trained and evaluated using the LEXACTUM framework, this time on the Ground dataset, as outlined in Section 4.3. The models' performances, at each of the training epoch lengths, were presented in Table 4.2. The metrics reported are those detailed in Section 4.4.1. CMU DeepLens yielded the best TPR once again, 0.9333 when trained for 100 epochs. The lowest FPR was achieved by CMU DeepLens, 0.0232 when trained for 25 epochs. The highest AUC was 0.9870, once again by CMU DeepLens, when trained for 150

Table 4.2: Table showing the TPR, FPR, AUC, TPR_0, TPR_{10}, and average execution time (Section 4.4.1) for 6 different models, as described in Section 3.1.2, trained for a various number of epochs on the Ground dataset. Columns marked with an * indicate the score achieved by the model in Metcalf, R. B. et al. (2019). Values in these columns marked in green indicate better performance compared to the implementations in LEXACTUM, whereas values in red indicate worse performance. Reproduced from Magro et al. (2021).

Model Name	No. of Training Epochs	TPR	FPR	AUC	TPR_0	TPR_{10}	AUC*	TPR_0*	TPR_{10}*	Avg. Execution Time per Image (seconds)
CAS Swinburne	10	0.8779	0.1077	0.9608	0.0000	0.0000				
	50	0.8995	0.0944	0.9720	0.0000	0.0000	0.96	0.02	0.08	0.0469
	100	0.8565	0.0406	0.9742	0.0000	0.0000				
	250	0.8726	0.0429	0.9758	0.0000	0.0000				
Lastro EPFL	50	0.9073	0.0536	0.9824	0.0000	0.5133				
	100	0.9110	0.0482	0.9844	0.0000	0.5504	0.97	0.07	0.11	0.0429
	250	0.9197	0.0489	0.9862	0.0000	0.0000				
CMU Deeplens	25	0.7733	0.0232	0.9588	0.0000	0.3840				
	50	0.9138	0.0568	0.9825	0.6046	0.6827				
	75	0.9026	0.0550	0.9804	0.0000	0.6536				
	100	0.9333	0.0660	0.9851	0.0000	0.6673	0.98	0.09	0.45	0.0594
	150	0.9205	0.0445	0.9870	0.0000	0.7042				
	250	0.8593	0.0858	0.9570	0.0000	0.0000				
WSI-Net	50	0.8560	0.0589	0.9620	0.0000	0.0000				
	100	0.8218	0.0301	0.9710	0.0000	0.5347	N/A			0.0231
	250	0.9127	0.0864	0.9742	0.0000	0.0000				
Lens Flow	50	0.8784	0.0744	0.9708	0.0000	0.5101				
	100	0.8831	0.0738	0.9726	0.0000	0.5648	N/A			0.0349
	250	0.9006	0.0733	0.9758	0.0000	0.0000				
Lens Finder	50	0.8556	0.0648	0.9665	0.0000	0.4442				
	100	0.8938	0.0805	0.9718	0.0000	0.5664	N/A			0.0293
	250	0.8997	0.0880	0.9671	0.0000	0.0000				

epochs. In Metcalf, R. B. et al. (2019)'s paper, the best AUC for the Ground set was 0.98, also by CMU DeepLens. The best TPR_0 was 0.6046, demonstrated by CMU DeepLens when trained for 50 epochs. In Metcalf, R. B. et al. (2019)'s paper, the best TPR_0 for the Ground set was 0.22, by Manchester SVM. CMU DeepLens achieved the highest TPR_{10}, 0.7042, when trained for 150 epochs. In Metcalf, R. B. et al. (2019)'s paper, the best TPR_{10} for the Ground set was 0.45, also by CMU DeepLens. This is an especially significant result, as it showcases how, with LEXACTUM, the very same network (as specified in Metcalf, R. B. et al. (2019)), registered a 56% improvement in the TPR_{10} metric over what is reported in Metcalf, R. B. et al. (2019)'s work.

This improvement is likely due to the slightly different image augmentation techniques applied, which allowed the same model to train for a higher number of epochs without overfitting. In the original work, this model was trained up to 120 epochs (Lanusse et al., 2018), whereas with LEXACTUM, it was trained until 250 epochs, which allowed for more optimal weights to be found at 150 epochs.

4.4.2.3 | Reproducibility of Results

In order to ensure the repeatability of this experiment, the robustness of the architectures, and the consistency of the results obtained, all of these models were trained and evaluated from scratch a second time, for both the Space and Ground dataset. The training, validation, and test set split used were also newly generated (same ratio, different selection). For the Space set, it resulted that between the two runs, the mean difference between the two runs of any model (with the same configuration and parameters) was 0.96%, with the greatest difference between any two runs being 2.97%. When comparing the two runs of the models on the Ground set, the mean difference was 0.39%, with the highest difference being 2.99%.

4.4.2.4 | The Importance of Image Augmentation

When trained for 250 epochs on the Space set, CMU DeepLens achieved an AUC of 0.9291, TPR of 0.7592, and an FPR of 0.0436, as shown in Table 4.1. This result was obtained with image augmentation during training, as specified in Section 4.2, enabled. CMU DeepLens was trained once again from scratch, using the exact same dataset, parameters, and configuration, only this time, image augmentation was disabled. The model from this training run scored an AUC of 0.8800, TPR of 0.7103, and an FPR of 0.1003. This result alone already showcases the impact of image augmentation on the performance of a model, with the AUC dropping by 5.28%, TPR dropping by 6.44%, and the FPR increasing by 130.05% (keeping in mind, lower is better for the FPR).

Moreover, the evolution of the accuracy of the two runs over the epochs during training was observed. For the model trained without image augmentation, the accuracy attained on the training data can be seen constantly improving epoch after epoch, even reaching 0.9996. However, the accuracy of that same model on the validation set only reaches 0.8156 after 250 epochs. This score is reached as early as the 15^{th} epoch, demonstrating how the model is failing to improve its accuracy on unseen data, i.e. overfitting. On the other hand, when image augmentation is applied during training, the model 'only' accomplishes an accuracy of 0.8989 on the training set. More importantly, however, is that by the 15^{th} epoch, this model has already achieved an accuracy of 0.8333 on the unseen validation set data, and even more impressively, manages to further improve on this, and continues to climb to an accuracy of 0.8813.

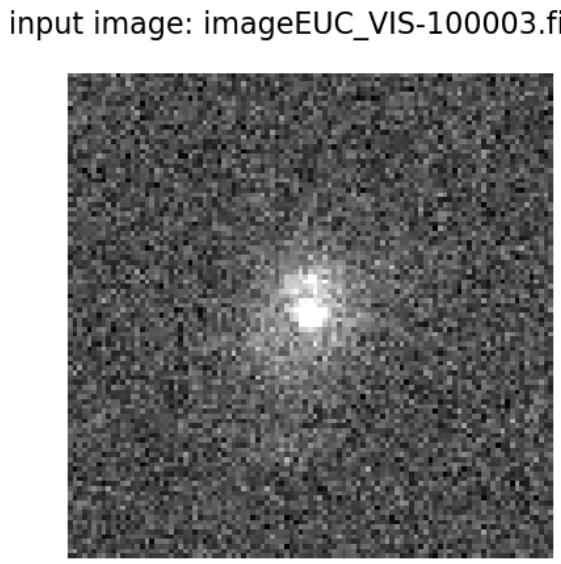

Figure 4.3: Input image, 'imageEUC_VIS-100003.fits' (Metcalf, R. B. et al., 2019), used to visualise the features extracted by the CMU DeepLens model which was trained for 500 epochs.

4.5 | Visualising and Interpreting Features Extracted by Convolutional Layers

As mentioned in Section 4.3, the 'visualise features' component was included in LEXACTUM, with the goal of demystifying the 'black box' that NNs are often viewed as, and revealing to the NN's developer, or any user for that matter, the inner workings of the model they have implemented, or are using. This component enables its users to examine any model's output for each and every convolutional layer within a CNN, for any given image. The fact that this feature is readily implemented in LEXACTUM means that such a feature, which developers may deem 'extra' to implement themselves, is automatically made available to them to apply to their model, without any further work or development required from their end.

In this section, 'space_cmu_deeplens_500epochs.h5' will be executed (since it obtained the highest AUC) on a random image from the Space dataset, and the features extracted by each convolutional layer will be visualised and interpreted. The random image used as input is shown in Fig. 4.3.

Fig. 4.4 presents a sample of features extracted by the first convolutional layer in the

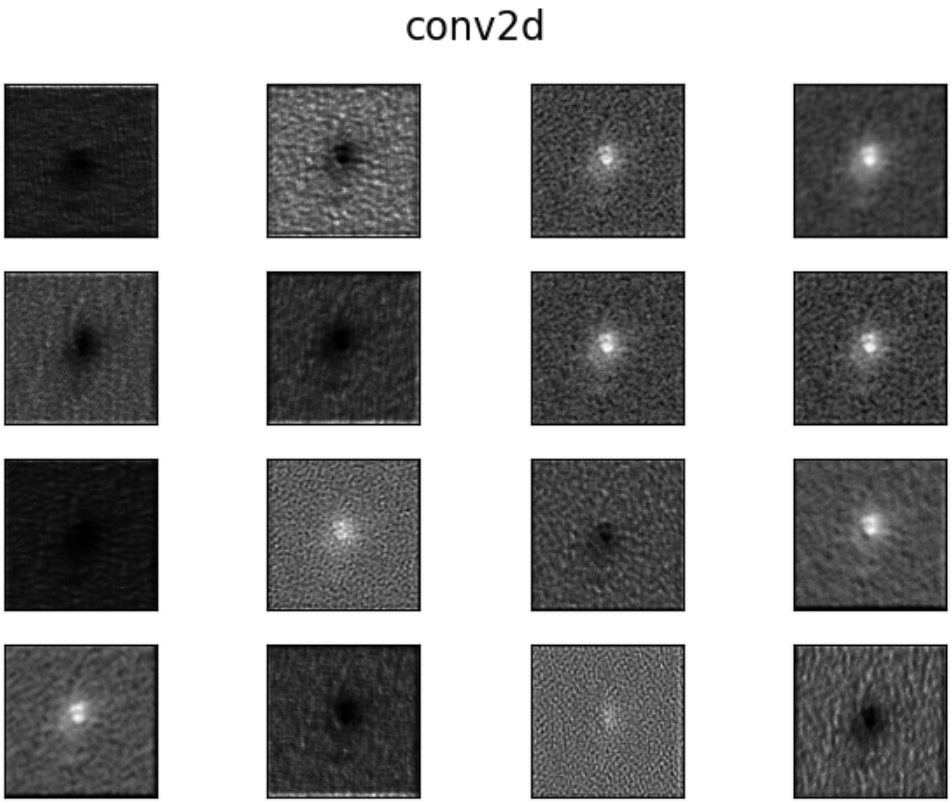

Figure 4.4: Visualisation of a sample of features extracted by the first convolutional layer of a CMU DeepLens model which was trained for 500 epochs. Reproduced from Magro et al. (2021).

CMU DeepLens model. In these feature maps, the original image is clearly still very prominent, as expected at such an early layer in the model. Here, the model is working towards highlighting low-level details within the input image that it has learnt are important and relevant. For instance, by adjusting the brightness, the contrast between the foreground object and the background, and the accentuation of the boundary between them, all of which is clearly evident in the features extracted by this layer.

Next, Fig. 4.5 displays a second sample of features, this time extracted by the final convolutional layer within the first '3 ResNet block'. As with the first convolutional layer, the input image can still be made out in the extracted features, although to a lesser extent. Again, it is a rather early stage in the model, and the model has not yet reached a stage where it is considering particularly sophisticated and abstract features and patterns.

Finally, Fig. 4.6 shows a sample of features extracted from the last convolutional

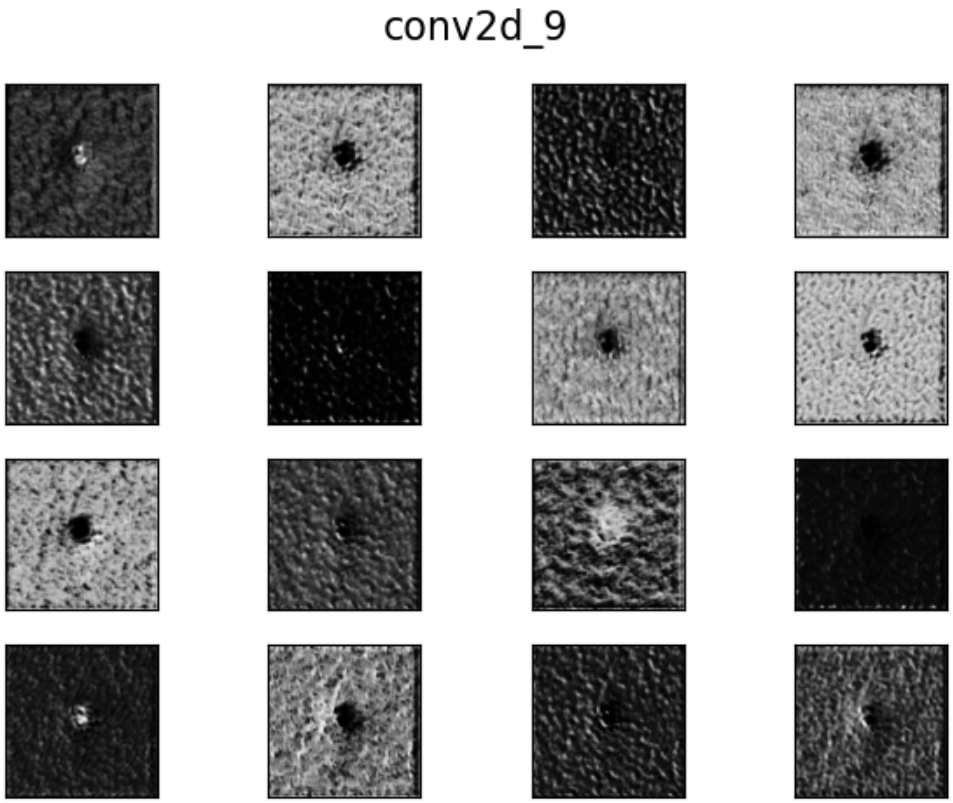

Figure 4.5: Visualisation of the features extracted by the last convolutional layer of the first '3 ResNet block' of a CMU DeepLens model which was trained for 500 epochs. Reproduced from Magro et al. (2021).

layer of each of the remaining '3 ResNet blocks'. Through every subsequent convolutional layer, the extracted feature maps become progressively harder to make out and interpret, and show less and less detail. According to Brownlee (2019), this is due to the features extracted by deeper layers being more abstract, showing "more general concepts", which in turn make it easier for the model to make a classification. This is the point at which it is typically no longer evident what the model is focusing on, and what patterns it is detecting. At this stage, the model is combining the basic features extracted by earlier layers to extract more sophisticated, higher-level, abstract features and patterns.

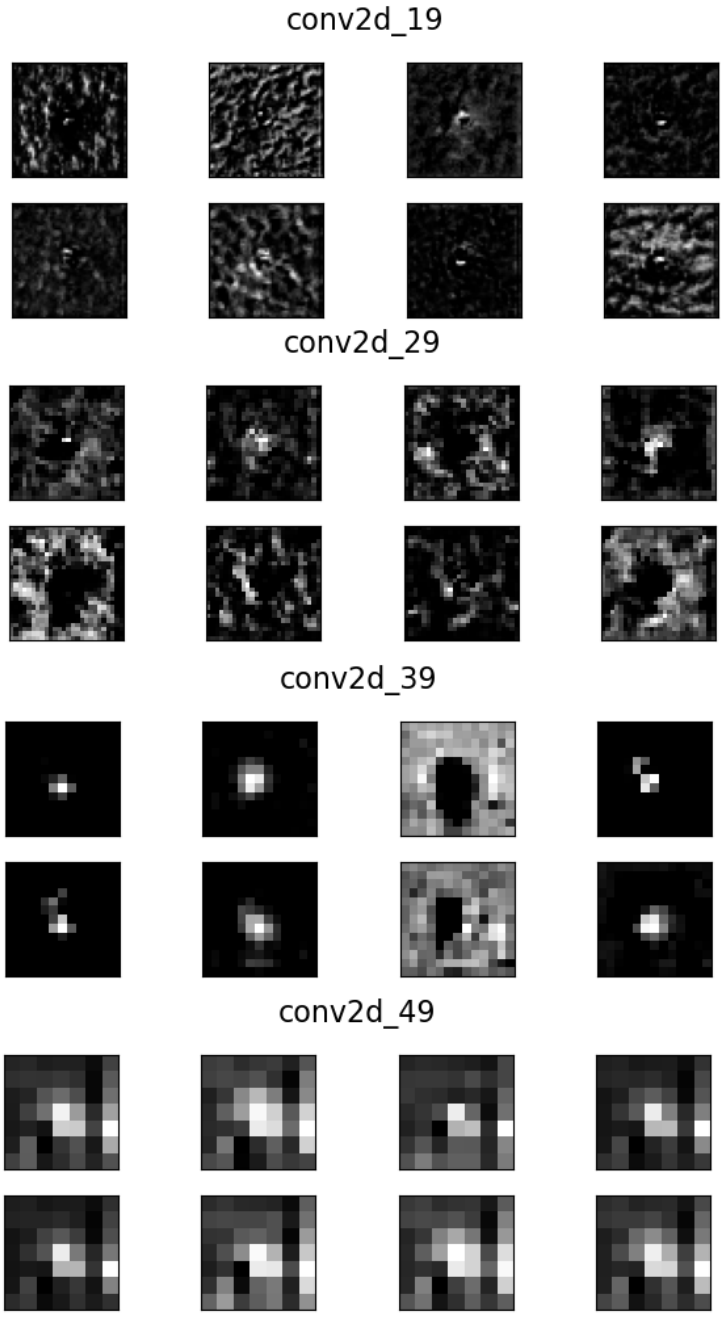

Figure 4.6: Visualisation of the features extracted by the last convolutional layer of the remaining '3 ResNet blocks' of a CMU DeepLens model which was trained for 500 epochs. Reproduced from Magro et al. (2021).

4.6 | Conclusions

The developed framework, LEXACTUM, evidently makes the development of new NN architectures, and application of existing ones, to the Gravitational Lensing problem an easier, more straightforward, and ultimately more successful endeavour. The readily available preprocessing options (image normalisation algorithms, as well as image augmentation techniques for training), trained model saving and loading features, and built-in evaluation suite which, being standardised, allows for perfect repeatability and consistency across all NNs evaluated, are automatically available to any NNs added to the framework. Moreover, the feature visualisation component described in Section 4.5 inherently enables users to delve into the inner workings of any model, out of the box, allowing for a better understanding of an implemented model.

Basing off the successful results showcased in Section 4.4.2, it is evident how NN architectures already implemented and evaluated in Metcalf, R. B. et al. (2019)'s work, when reimplemented in the LEXACTUM framework, in some cases achieve significantly better results. For instance, for CMU DeepLens on the Ground dataset, the TPR_{10} showed a 56% improvement over what the very same architecture accomplished in Metcalf, R. B. et al. (2019). Such a result affirms the impact of image augmentation, as outlined in Section 4.4.2.4, which is why its inclusion and ease of use within LEXACTUM is so distinguishing. Moreover, CMU DeepLens also registered a very impressive 175% improvement of TPR_0 over the winning solution for this metric in Metcalf, R. B. et al. (2019) for the Ground set. Furthermore, new architectures were implemented and applied, for instance WSI-Net, which achieved a TPR_{10} 17% higher than that of the winning solution in Metcalf, R. B. et al. (2019) for the Space dataset.

In essence, this work delves into data preprocessing techniques, particularly image augmentation techniques, which allow models to train for more epochs without overfitting the model to the training data. Moreover, techniques which had shown success in other domains were adapted for this problem, with promising results, reaffirming the adaptability and flexibility of CNNs. Finally, this work reaffirms the efficacy of CNN-based techniques applied to astronomical data, and ensures a promising way forward for dealing with upcoming big astronomical datasets which are otherwise intractable if manually annotated by astronomers.

As a final, crucial point, it is important to highlight how LEXACTUM, despite being applied to gravitational lensing here, can be applied, with all of its included ease-of-use features, to any classification problem in the astronomy domain.

4.6.1 | Future Work

The inclusion of an Elliptical Hough Transform as an optional preprocessing step could potentially lead to interesting results, as it might make features which models deem most relevant for determining whether an image is classified as a lens or not more prominent, thus augmenting models' discernment power. In Storkey et al. (2004), a similar concept is applied, however in their work it was noted that only larger features were detected. In view of this, the transform operation's output could alternatively be input to models as an additional channel alongside the original input image, rather than instead of the original input image.

Another feature with the potential to maximise the performance of models in LEXACTUM is the inclusion of a module which autonomously carries out hyperparameter optimisation. Such a process would be able to fine-tune the hyperparameters and remove the 'guesswork' or empirical process from the user, thus amplifying models' performance, without any user intervention. Examples of such hyperparameters can include the learning rate, number of epochs, dropout rates, activation functions, kernel sizes, number of features extracted, layers e.g. adding or removing a ResNet block, pre-processing algorithms, normalisation algorithms, etc.

Additional image augmentation techniques should also be explored, as these may hold the key for training models for an even greater number of epochs while mitigating overfitting.

Finally, more network architectures should be evaluated, ResNet-based networks, for example, have demonstrated very promising results with the classification of gravitational lenses. Furthermore, cutting-edge, state-of-the-art network architectures should be implemented and evaluated as they are released. One such example is ViT, described in Section 3.3.5.1. Since the publishing of this work in Magro et al. (2021), an implementation based on transformers, Lens Detector (Thuruthipilly et al., 2022), was applied to this same challenge. When comparing the principal metric of the challenge, the AUC, LEXACTUM (with CMU DeepLens) obtained a 0.9343, whereas Space Lens Detector scored 0.925 on the Space set. On the Ground set, LEXACTUM achieved a 0.9870 and Lens Detector 21 a 0.980 (Thuruthipilly et al., 2022). While these implementations did achieve very impressive results, at the time of writing, the CNN-based, particularly ResNet-based, architectures in LEXACTUM outperform them, however it is very possible, if not likely, that further advancements or different transformer-based architectures will lead to even better performances.

5

Detection of Compact Sources, Extended Galaxies, and Sidelobes in Radio Astronomical Maps

Some of the content in this chapter, Chapter 5, incorporates concepts discussed in Riggi et al. (2023).

As with upcoming surveys necessitating the automation of classification of gravitational lenses, upcoming radio continuum surveys will make manual source finding intractable, and likely render currently available automated solutions insufficient. One such upcoming survey is the Evolutionary Map of the Universe (EMU) (Norris et al., 2011), planned at the Australian SKA Pathfinder (ASKAP) telescope (Hotan et al., 2021; Johnston et al., 2008). The next-generation sensitivity, resolution, and field of view of such surveys is expected to make the detection of millions of sources possible. In order to automatically, and accurately, catalogue and extract information from such a vast and large dataset, a paradigm-shifting source finder will need to be developed.

In light of this, several existing source finders in the radio community, such as Carbone et al. (2018); Hancock et al. (2018); Riggi et al. (2019), have advanced towards this goal, and new methods such as Hale et al. (2019); Lucas et al. (2019); Robotham et al. (2018) have been developed. That being said, some essential algorithmic aspects, both for EMU and future SKA surveys, remain to be addressed, notably when dealing with data from the Galactic plane.

For instance, the recognition and filtering out of spurious sources is one such feature which is still lacking in the available solutions, so much so that many source finders report a false detection rate around 20%. This is particularly prevalent in observations with a lot of background noise, or sources with very extended morphologies. Some of the primary causes for these false detections are noisy backgrounds or imag-

ing artefacts, in particular, sidelobes around very bright sources, especially in high SNR images. False detections in multiple-island galaxies are, in part, due to islands which may have been incorrectly identified, however, deblending being carried out too aggressively may be the primary cause, such that these structures with multiple components are falsely identified as individual objects, in particular when dealing with structures with extended morphologies. In the aforementioned existing solutions, such spurious detections are not often correctly filtered out and discarded, and when they are, it is done using context-specific rules, as opposed to general rules, or through a manual process.

With existing source finders, another area which is poorly implemented is the identification of sources in the presence of groups of islands and their proper classification into predefined astronomical objects (e.g. radio galaxies). This is especially prevalent when detecting galactic objects within galactic plane surveys. In such surveys, extragalactic objects make up $\sim$90% of background sources. Despite the majority of these having a single-island morphology, radio galaxies[1] with extended morphologies (each island possibly associated to different components, e.g. core, lobe, or jet, of the object) have surpassed the number of previously known galactic objects in the same area. For example, the number of islands belonging to radio galaxies was $\sim$3 times larger than known, reported, galactic objects in the ASKAP Scorpio survey (Riggi et al., 2021b). This statistic emphasises the significance of a solution capable of automatically detecting and filtering out such objects, thus simplifying the search for galactic objects. As mentioned in Section 3.3.1, recent works (such as Lukic et al. (2018, 2020); Wu et al. (2018)) employing deep CNNs have already shown encouraging results for radio source detection and classification, particularly for radio galaxies in extragalactic fields.

This work will introduce a new source finding tool, Automated Source, Galaxy, and Artefact R-CNN Detector (ASGARD), with the goal of addressing and rectifying some of the aforementioned limitations and shortcomings in existing solutions. ASGARD is based on Mask R-CNN, an architecture designed to perform Instance Segmentation, i.e. the detection, classification, and drawing of per-pixel masks on detected objects, also differentiating between overlapping objects. It is trained to detect, and distinguish between, compact radio sources, radio galaxies with extended morphologies, and imaging artefacts (sidelobes).

The following section, Section 5.1, describes the composition of the dataset used, as well as the collaborative process followed to assemble the final dataset. Section 5.2 then outlines how the Mask R-CNN model was implemented in ASGARD, along with

[1] 'radio galaxies' is being used to refer to radio galaxies in general, without further categorisation

its features. Next, Section 5.3 describes the Metrics used to evaluate the performance of ASGARD. Section 5.4 outlines the process followed to optimise ASGARD's hyperparameters, to find the best performing architecture and hyperparameters. Then, Section 5.5 reports the results obtained by ASGARD, putting these in the context of the data's SNR, and presents qualitative examples of ASGARD's performance. Finally, conclusions and possible future improvements are outlined in Section 5.6.

5.1 | The Dataset

5.1.1 | Observational Data

5.1.1.1 | ASKAP Pilot Surveys

For the purpose of ensuring the ASKAP array's operation, observation strategy, and optimising the data reduction pipeline, the ASKAP EMU Early Science Project (ESP) was established. As part of this initiative, numerous pilot surveys were conducted on various areas of the sky deemed to be of interest, including the Galactic plane, yielding early scientific outcomes. The EMU Pilot Survey (Norris et al., 2021) was one of these surveys, with an area covering 270 deg^2, a background Root Mean Square (RMS) noise of 25 to 30 μJy beam^{-1}, spatial resolution of 11 to 18 arcsec, observed at 944 MHz, generating a catalogue of around 220,000 sources, 180,000 of which are single-component sources.

In 2018, as part of ASKAP's ESP, the SCORPIO field, with an area of around 40 deg^2, centred at galactic coordinates ($l = 343.5°$, $b = 0.75°$), was observed, making it the only field observed in the Galactic plane (in the ESP). This observation was first performed with 15 antennas, with an area of around 40 deg^2, an RMS of 541 μJy beam^{-1}, resolution of 24.1 $\times$ 21.1 arcsec^2, at 912 MHz (Umana et al., 2021). In this work, this survey will be referred to as ASKAP-15. This survey yielded a catalogue of around 4,000 compact sources (Riggi et al., 2021b). A subsequent observation, carried out on the same field, observed all three ASKAP bands (0.7 to 1.7 GHz) with 36 antennas. In this work, this survey will be referred to as ASKAP-36. This observation had an RMS of around 50 μJy beam^{-1}, resolution of approximately 9.4″$\times$7.7″, at 1,243 MHz (Ingallinera et al., 2022). As a result of the greatly enhanced sensitivity of this survey, the amount of detected compact sources increased threefold.

An earlier observation of the SCORPIO field, conducted with the Australia Telescope Compact Array (ATCA), with an RMS of 30 μJy beam^{-1}, resolution of around 10 arcsec, at 2.1 GHz (Umana et al., 2015), was repeated with ASKAP's improved field of view, allowing for Riggi et al. (2021b) to detect around 4,100 source components.

5.1.1.2 | The Radio Galaxy Zoo (RGZ) Dataset

RGZ[2] (Banfield et al., 2015) is a citizen science project, where the public can volunteer to label radio galaxies and their corresponding host galaxies, on radio and infrared images, through an online platform. DR1 (Wong et al., in preparation) is the first catalogue released from RGZ, from its first 2.75 years of operation, with over 12,000 citizen scientists contributing. Of the ∼100,000 sources in the DR1 catalogue, over 99% are from the FIRST survey (Becker et al., 1995) carried out at the VLA, with the remaining (less than) 1% of sources from the Australia Telescope Large Area Survey (ATLAS) (Norris et al., 2006). In this work, a subset[3] of the RGZ DR1 catalogue by Wu et al. (2018) was used, containing 10,744 images with 11,836 sources, consisting only of sources with a consensus of at least 60% and at most three peaks and three components. Radio galaxies in this catalogue are labelled according to their amount of components, C, and peaks, P. 68% of sources in the dataset fall under 1C-1P, 1C-2P, or 1C-3P, another 21% are represented by 2C-2P or 2C-3P, with the final 11% made up of 3C-3P.

5.1.2 | Source Labelling Scheme

Following the scientific use case detailed in the Introduction to Chapter 5, three object classes were selected for the source finder to detect and distinguish between:

1. "*galaxy*": including radio sources with an extended morphology, consisting of one or more extended (point-like or extended) islands (or components, if adhering to the RGZ terminology), each with one or more peaks (in brightness). For brevity, this class will be denoted as 'galaxy' hereafter, as their radio morphology resembles that of extended radio galaxies. As we are not requiring strict criteria (e.g. identification of the host galaxy) other than the morphology to define this class, these objects are to be treated as candidate radio galaxies. The goal is selecting these candidates with an automated procedure, for removal, as in Galactic science studies, or follow-up analysis, as in extragalactic science studies, which would then be carried out either manually, or automatically by another, more specialised tool;

2. "*source*": including isolated point- or slightly resolved compact radio sources, with single-island and single-component (or single-component and single-peak, as per RGZ terminology) morphology, typically resembling the synthesised beam shape

[2] `https://radio.galaxyzoo.org/`
[3] `https://cloudstor.aarnet.edu.au/plus/s/agKNekOJK87hOh0`

and fittable with a 2D Gaussian model. Throughout the work, 'source' will be used to refer to this class of objects. Please note that the majority of these are eventually radio galaxies or quasars, with only a small fraction (∼10%) being Galactic objects. This definition is, however, being based only on the radio morphology appearance, as the goal is selecting valid inputs for other classification tools in development (Riggi et al., 2021c), specialised for individual source classification;

3. "*sidelobe*": including artefacts in the image, primarily those produced by antennas during the imaging process. In this work, the 'sidelobe' label is restricted to sidelobes around bright compact sources, usually having a ring-like or elongated compact morphology. The detection of sidelobes (or imaging artefacts in general), and thus the ability to filter them out, is made even more impactful by the fact that they reduce source finders' reliability (or precision) on catalogues, even at high SNRs, as they are normally bright, even surpassing the 5σ significance threshold applied by traditional source finders.

Sample objects belonging to each of these classes can be seen in images from the dataset in Figure 5.3.

5.1.3 | Dataset Preparation

To assemble the dataset, objects from each of the three classes defined in Section 5.1.2 were searched for in the available radio data, specified in Section 5.1.1. When searching for "galaxy" class objects, labelled radio galaxies that may be visually misclassified by one or more isolated point-sources, belonging to the "source" object class were discarded from the RGZ DR1 subset (Wu et al., 2018). The resulting object counts per class and data source are presented in Table 5.1. The final, accumulated, dataset consists of ∼9,200 images, containing ∼19,000 sources, 1,290 sidelobes, and 3,200 galaxies. Finally, these 9,200 images were randomly split into a training and test set, containing 70% and 30% of the original dataset size, respectively (Sortino et al., 2023a,b).

Single-channel image cutouts were then extracted from this accumulated dataset in Flexible Image Transport System (FITS) format. The RGZ dataset used provided 132×132-pixel image cutouts with bounding boxes for radio galaxy objects, so the cutout size of 132×132 pixels was maintained and applied across the dataset. To make the training of the developed source detection framework possible, every image cutout in the dataset needs a corresponding series of masks, each mask representing a binary segmentation of an object in that cutout, i.e. a classification of every pixel in the cutout as belonging to the object (foreground) or not (background).

Table 5.1: Number of images (Column 2) and objects per class (Columns 3-5) in the produced dataset. Results are split by data source (Column 1, see Section 5.1.1 for details), with the totals in the last row. Reproduced from Riggi et al. (2023); Sortino et al. (2023a).

Data Source	Images	Object Counts		
		GALAXY	SOURCE	SIDELOBE
RGZ DR1	2974	2978	310	3
SCORPIO ASKAP-36	217	190	440	130
SCORPIO ASKAP-15	3723	5	13001	281
SCORPIO ATCA	2141	0	4698	21
ASKAP EMU Pilot	150	26	562	852
All	9205	3199	19011	1287

An initial object segmentation was generated for each cutout using the Compact And Extended Source Automated Recognition (CAESAR) source finder[4] (Riggi et al., 2016, 2019), configured with 'seedThr=5' and 'mergeThr=2.6'. In most cases, for point sources, these parameters were found to produce satisfactory results (masks), such that minimal manual refinement was required, if at all. For extended galaxies, however, this was not always the case, as their diffuse lobe emission in many instances does not meet the tool's standard 5σ threshold. Similarly, sidelobes around bright sources also required manual refinement in many cases, as these were frequently blended and misidentified as a singular object (Sortino et al., 2023a,b).

A data preparation team was assembled to manually inspect, review, and refine the preliminary pixel masks and object classifications produced by CAESAR, as well as to add any objects it may have failed to detect. This team was further subdivided into smaller groups of three or four people, each group assigned a subset of the dataset, such that each member's work would be reviewed by another member in the group. Before being shared with the team, the data and the corresponding masks were "anonymised", i.e. any World Coordinate System (WCS) or identifying metadata (e.g. references to observatories) were cleared from the file headers.

This data was then made available to the data preparation team through a git repository. This was used in conjunction with Data Version Control (DVC)[5], such that the git repository tracked the files in a connected cloud storage. This allowed for both the raw data (images) and any object segmentation masks (regions) or full image (cutout) labelled segmentation masks to be kept under version control throughout the process, from preliminary to final, in an environment ideal for collaboration (as is the case with typical git repositories). A process was set up so that any modified or refined object segmentation maps could automatically generate the updated image masks (Sortino et al.,

[4] `https://github.com/SKA-INAF/caesar`
[5] `https://dvc.org/`

2023a).

Any pre-processing performed by ASGARD before working with the images in the dataset is described in Section 5.2.1.1.

5.2 | Model Implementation

The source finding tool developed in this work, ASGARD, is based on the architecture of Mask R-CNN, which has been detailed in Section 3.3.2. ASGARD is built upon a Mask R-CNN implementation[6] written in Python and using the TensorFlow[7] (Abadi et al., 2015) and Keras[8] (Chollet et al., 2015) libraries. Naturally, several changes and additions to the aforementioned base implementation used were required, e.g. to accommodate for the different data format being handled (single-channel FITS files instead of RGB images), image preprocessing required (such as dealing with NaN pixels and data normalisation), image augmentation, alternative loss metrics (e.g. Dice Loss), result post-processing, and evaluation metrics, as will be covered in this chapter. All the code written for ASGARD is publicly available on a GitHub repository[9], under the GNU General Public License v3.0[10].

5.2.1 | Data Loading

The base Mask R-CNN implementation used, mentioned in Section 5.2, was developed to work with RGB images. Since radio continuum data is generally stored in FITS files, ASGARD's data loader was thus modified such that it could read 2D FITS files. Given that the base implementation's architecture expects a 3-channel (typically RGB) input, and that this dataset is composed of single-channel data, the data loader replicates the single grayscale channel over the 3 channels. While it was also possible (and in fact tested) to modify ASGARD's architecture to accept a single-channel input, a 3-channel input was maintained, as any weights for pre-trained models are provided for the standard architecture, i.e. that with a 3-channel input. Again, modifying the architecture is still a viable solution, as the weights for the first layer could be excluded, and randomly initialised, when loading the pre-trained weights, however, the former solution

[6] https://github.com/matterport/Mask_RCNN
[7] https://www.tensorflow.org/
[8] https://keras.io/
[9] https://github.com/SKA-INAF/mrcnn
[10] https://www.gnu.org/licenses/gpl-3.0.html, https://github.com/SKA-INAF/mrcnn/blob/master/LICENSE

was chosen for its relative simplicity. Of course, the model can still be trained from scratch (without using pre-trained weights).

The accepted dimensions of input images (cutouts) are not hard coded, and the architecture is flexible to datasets of different cutout sizes, however, the dimensions would need to be, at most, of similar magnitude to the Mask R-CNN `IMAGE_MAX_DIM` parameter (typical values are 256, 512, or 1024 pixels per side). Alternatively, larger images can also be handled with the parallel processing mode, as described in Section 5.2.4. Section 5.2.1.1 will elaborate on any data pre-processing implemented before the loaded data is passed to the model as input.

5.2.1.1 | Data Pre-processing

Before image (cutout) data is fed to the model, certain pre-processing operations are carried out on the data (in memory, i.e. not saved to disk) after it has been loaded from the dataset (from disk). The first of these is replacing 'NaN' pixel values with the image minimum, i.e. any pixels with a value of NaN are set to the smallest finite pixel value in the cutout. Next, pixel values are normalised using ZScale (NOAO, 1997) normalisation to enhance cutouts' contrast (of foreground objects to the background) and range of pixels values. Pixel values were then normalised yet again to span the more conventional range of 0 to 255. Finally, as mentioned in Section 5.2.1, the single grayscale channel is replicated over the model's 3 input channels (i.e. the input tensor is reshaped from $(x, y, 1)$ to $(x, y, 3)$ by setting every channel to the same values of the first). It is worth noting that background subtraction was not applied as one of these operations, as the model is being expected to learn how to function despite the noise naturally present in data (Sortino et al., 2023a,b).

Similarly to Section 4.2, image augmentation was employed during training to mitigate overfitting. Each image loaded undergoes a random number of augmentations (ranging from 0 to 2) before it is fed to the model for training. These image augmentation operations consist of flipping horizontally (left to right), flipping vertically (upside down), rotating 90° (clockwise or anti-clockwise), and translation (shifting). The amount and selection of operations applied to each image are randomised every epoch (Sortino et al., 2023a,b). This process is shown on a sample image from the dataset in Fig. 5.1.

5.2.2 | Model Architecture and Training

In order to make ASGARD more configurable and customisable with regard to its model architecture and training options, some new features were developed. Out of the box,

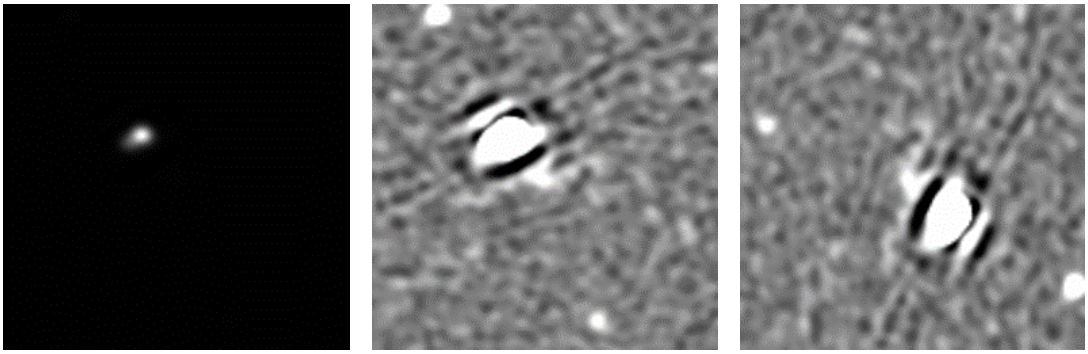

Figure 5.1: Sample dataset image at different stages of pre-processing. Left panel: a sample image from the dataset with no pre-processing applied (raw); Centre panel: the same sample image with ZScale normalisation applied; Right panel: the normalised sample image with random augmentation applied, in this case a rotation and a flip (this step is only applied during training).

Mask R-CNN comes with two backbones implemented, 'ResNet50' and 'ResNet101'. The first new feature was a custom backbone network, which is a very lightweight version of ResNet50. This 'simpler' and more shallow (less layers) backbone is in fact desirable as it reduces training times and, potentially, also decreases the chances of the model overfitting the training dataset. Results obtained with this modification are reported in Section 5.4.1.

During training, Mask R-CNN uses binary cross-entropy to calculate the 'mask loss'. Since binary cross-entropy calculates an average of pixel-by-pixel losses, this can result in a bias towards the background, especially if foreground objects are relatively small. Dice Loss (Milletari et al., 2016) instead calculates the 'overlap' between the ground truth and the prediction. Dice Loss was thus implemented in ASGARD, to provide an alternative mask loss function, which might be better suited. This will be evaluated in Section 5.4.4.

5.2.3 | Result Post-processing

When given an input image, the model generates a list of objects it has detected. Each of these objects will have an associated classification (label), confidence score (between 0 and 1), as well as a binary mask for which pixels in the image belong to the detected object. Once the model has generated this preliminary output for a given image, the following rules are applied to refine the model's initial list and produce ASGARD's final output (Sortino et al., 2023a,b):

1. Detected objects with a confidence score below a threshold (0.7 by default, configurable) are discarded;

2. Overlapping or adjoining objects are merged if they share the same classification label. If the labels are not identical, the detection with the lower confidence score is discarded.

5.2.4 | Run Options and Parallel Processing

ASGARD can be run in 3 different modes. The first of these is 'train', which, as the name suggests, indicates that the model will go through the part of the dataset dedicated to training to learn (or update, if starting from pre-trained weights) the weights to use for the Mask R-CNN model. On the other hand, 'test' is used to evaluate the performance (according to a set of metrics as described in Section 5.3) of an already trained model on data it has not seen, i.e. it was not trained on (the test set). Finally, 'detect', or the 'source finding mode', is used to apply a trained model to an image to detect any objects within it and obtain the model's prediction (the format of which is described in Section 5.2.5).

When compared to the base implementation of Mask R-CNN used, ASGARD's configurability has been enhanced significantly, as more than 60 unique command-line arguments were implemented to allow for users to specify not only the mode, but also the parameters to be used during execution. A list of all these command-line arguments, including a description, default value, and accepted values, is presented in Table C.2 in Appendix C.

With the goal of enabling ASGARD to run detection on larger images, an experimental implementation supporting parallel processing was prototyped, based on the *mpi4py*[11] (Dalcin and Fang, 2021) library. This implementation works by dividing the larger image into smaller cutouts, running detection on the individual cutouts (in parallel, when enabled), and reassembling the image (or rather, the detections), merging detections found at the edges of adjacent cutouts.

Further command-line arguments were implemented to allow for added configurability in this mode, such as for changing the cutout size and overlap. Fig. 5.2 presents the detections obtained by ASGARD in parallel mode, with n_{proc}=4 and tile (cutout) size=512 pixels, where n_{proc} is the amount of GPUs to be used simultaneously, on a 2000×2000 pixel image taken from the SCORPIO ASKAP survey.

[11] `https://pypi.org/project/mpi4py/`

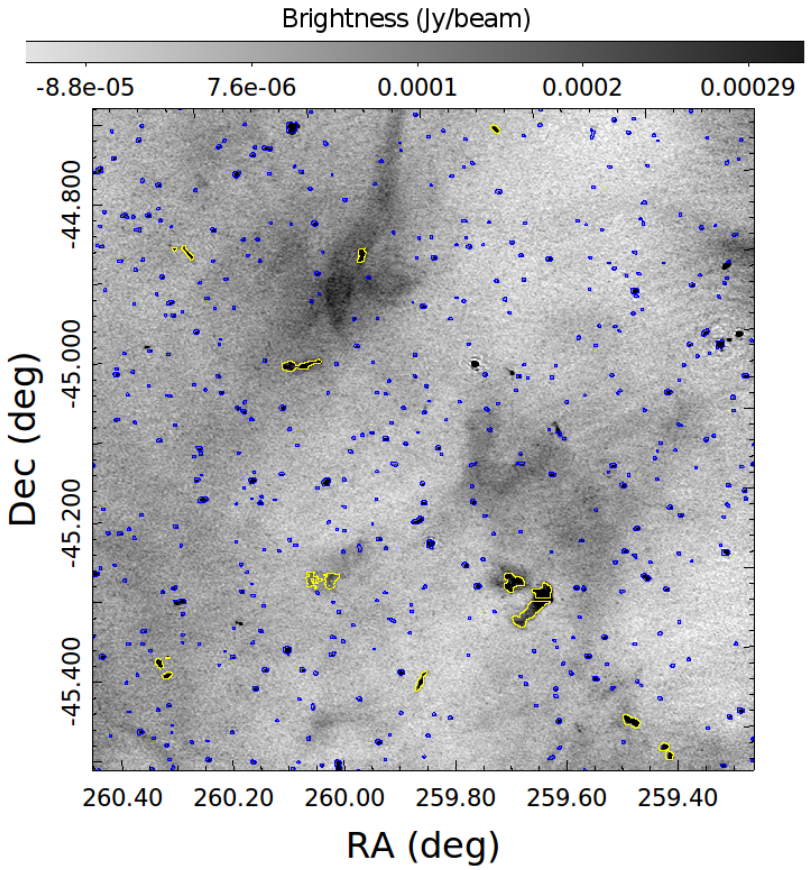

Figure 5.2: Sample image (2000×2000 pixels) from the SCORPIO ASKAP survey with sources extracted by ASGARD in parallel mode (see Section 5.2.4) superimposed (blue: compact sources, yellow: galaxies). Reproduced from Riggi et al. (2023).

5.2.5 | Data Outputs

The output ASGARD produces of course depends on the mode it is executed in. When run in 'train' mode, it will save the trained model weights at every epoch to disk in Hierarchical Data Format 5 (HDF5) format, together with the time spent on the epoch and the model's losses during that epoch. The 'test' mode calculates a set of metrics as defined in Section 5.3 for the quantitative evaluation of a trained model, along with the input images with the model's detections overlaid on them, which are useful both for qualitative evaluation and comparison, as well as diagnostic purposes, such that areas where the model is struggling can be observed visually. Finally, the 'detect' (source finding) mode produces the following list of outputs for any given input image:

1. a catalog of objects ASGARD has detected within the image in JSON format, the

structure of which was made to resemble that produced by traditional source finders (such as CAESAR[12] or AEGEAN[13]) for island sources, including the following details:

- object position information: centroid coordinates, bounding box coordinates, list of pixels belonging to the object, and outline of the detected object;
- classification: class label and confidence score;
- object flux information: integrated flux density, peak brightness, pixel brightness statistics (e.g. minimum, maximum, median, standard deviation, etc.), and flags (e.g. object located at image border, etc.);

2. DS9 region files for each detected object in polygon region format, with classification labels tagged and colours (associated with labels) applied;
3. the input image with the detected objects overlaid as colour-coded (according to their classification) pixel masks and bounding boxes, together with their classification and confidence score as text;

Detailed features or analyses, such as deblending or Gaussian fitting, of each detected object are intentionally not provided or performed by ASGARD, as other tools, such as CAESAR or AEGEAN already implement these features. The integration of these tools into ASGARD is planned for future development.

Fig. 5.3 presents sample image-format outputs of ASGARD, after the post-processing stage described in Section 5.2.3, on three different images, two of which contain a 2-component and 3-component radio galaxy, respectively, and one with a compact source and sidelobes. For each image, the detected objects are superimposed with their masks (colour coded by classification) and their classification and confidence score.

5.3 | Evaluation Metrics

In order to compare the performance of the best model with other existing CNN-based models, the following metrics were implemented. Terms such as TP, FP, FN, TN, IoU, Precision (Reliability), Recall (Completeness), F1-Score, Precision-Recall (Reliability-Completeness) curve, mAP, and mAP50 have been defined in Section 2.2.3.1.

In order to calculate all of these metrics, an analysis component was implemented in ASGARD. This goes through the test set, and for each image, runs the model to

[12] `https://caesar-doc.readthedocs.io/en/latest/usage/data_products.html`
[13] `https://github.com/PaulHancock/Aegean/wiki/Output-Formats`

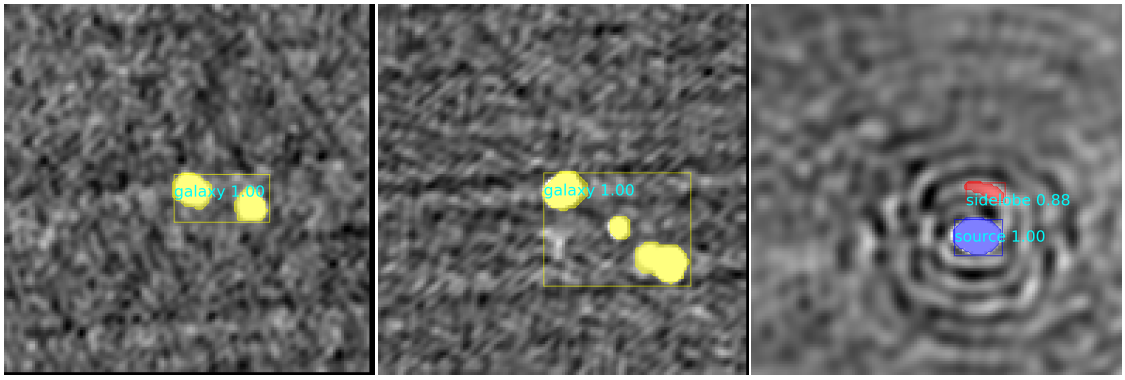

Figure 5.3: Sample ASGARD image-format outputs, displaying colour-coded segmentation masks and bounding boxes (according to their classification) with classification labels and confidence scores for the detected objects, on three sample images from the dataset. Left panel: a sample radio galaxy with 2 component islands (taken from RGZ data); Centre panel: a sample radio galaxy with 3 component islands (taken from RGZ data); Right panel: a sample source+sidelobe (taken from ASKAP-15 SCORPIO data).

detect objects, along with post-processing and score thresholding. From these outputs, a per class Reliability and Completeness is calculated. Furthermore, from these pairs of values, the class's F1-Score is also calculated, as shown in Equations 2.12, 2.9, and 2.13.

Additionally, the aforementioned analysis component stores the ground truth and the model's detections for each image in the test set in three specific formats, such that they can be used as input for three other open source, performance evaluation tools. The tools supported are *Object-Detection-Metrics*[14], *calculate_mean_ap*[15], and *metric-computation*[16] (which was developed as part of this work for Sortino et al. (2023a,b)). Each of these tools calculates the mAP overall, as well as for each class, for a given IoU threshold. Moreover, *calculate_mean_ap* was modified as necessary to produce a Reliability-Completeness curve for each class, and overall, such as those in Fig. 5.4.

Given that there is an imbalance in the number of instances of each class in the dataset, Reliability-Completeness curves were chosen over ROC curves as a metric, as they present a more realistic view of the model's performance in this scenario (Davis and Goadrich, 2006).

[14] https://github.com/rafaelpadilla/Object-Detection-Metrics
[15] https://gist.github.com/tarlen5/008809c3decf19313de216b9208f3734
[16] https://github.com/SKA-INAF/metric-computation

5.4 | Hyperparameter Optimisation

This section will detail the series of training runs carried out to tune the Mask R-CNN model hyperparameters, and quantify their effects on ASGARD's detection and classification performance. Measures were taken in order to make each run as comparable as possible and reduce the effects of any external factors as much as possible, thus isolating the effect of the tested hyper-parameter on performance. For instance, each parameter set was trained for 250 epochs, the train/test split used was always the same one, and all models were trained on the same GPU model on the same system with the same resources allocated. All these models were trained on the same system described in Chapter 4 (Section 4.3), where jobs were assigned to one of four Slurm nodes, each of which was allocated 8 CPU cores, 48 GB of RAM, and an NVIDIA V100 GPU with 16 GB of VRAM. Any performance metrics shown will be calculated on the weights after the 250th epoch. No overfitting is detected in all performed runs before reaching the stop epoch. All evaluation runs were carried out with a score threshold of 0.7 and an IoU threshold of 0.5.

5.4.1 | Impact of backbone network

The first hyperparameter tested was the architecture of the backbone network. The tested values were: 'resnet101' (1-1), 'pre-trained resnet101'[17] (1-2), 'resnet50' (1-3) and 'custom' (1-4). In each test case, all the remaining hyperparameters were kept at their default. All parameters and their defaults are shown in Table C.1 in Appendix C. The results for these runs are displayed in Table 5.3.

In these runs, the model with the ResNet50 backbone provided better results (in terms of overall F1-score) compared to the other backbones after the same number of training epochs (250). Using the custom backbone resulted in slightly worse detection and classification performances, with a $\sim$14% reduction in training times, when compared to ResNet101. For instance, training ResNet101 for 250 epochs takes 169.4 hours on average, whereas 146.5 hours were spent with the custom backbone, corresponding to a computing time reduction of 22.9 hours.

From this first set of runs, a significant imbalance between the models' segmentation losses ('rpn_bbox_loss', 'mrcnn_bbox_loss', and 'mrcnn_mask_loss') and classification losses ('rpn_class_loss' and 'mrcnn_class_loss') became evident, with the segmentation losses an order of magnitude higher than the classification losses. The segmentation

[17] Weights taken from `https://github.com/keras-team/keras-applications/releases/tag/resnet/resnet101_weights_tf_dim_ordering_tf_kernels.h5`

losses, as the names imply, measure (the inverse of) how accurately the model is segmenting the image, or more specifically, how accurately it is determining object bounding boxes and pixel masks. The classification losses, on the other hand, measure how accurately the detected objects are being labelled. In both cases, a greater value for loss indicates lesser performance. From a qualitative review of the results, particularly ASGARD's image-format diagnostic plots produced during evaluation (i.e. the input images with the detections overlaid, as specified in Section 5.2.5), it was clear that, despite the greater loss values, masks and bounding boxes seemed sufficiently accurate, whereas the classification labels were suboptimal.

With this in mind, and given that, by default, all of these losses have the same weighting when computing the models' total loss, poor performance on classification would have a fraction of the impact compared to what it should (owing to the aforementioned imbalance), such that it becomes very hard for it to improve during training. Therefore, all 4 of these backbones will still be considered until further results are in hand, and the following section, Section 5.4.2, will experiment with balancing the loss components.

5.4.2 | Impact of weighted losses

To balance the aforementioned classification (class) and segmentation (bbox and mask) losses, the models were trained again with a modified `LOSS_WEIGHTS` parameter, such that the weights for the aforementioned 'rpn_bbox_loss', 'mrcnn_bbox_loss', and 'mrcnn_mask_loss' were set to 0.1, with respect to 'rpn_class_loss' and 'mrcnn_class_loss' (the weights of which were kept at the default of 1.0). These updated values should make the smaller loss values of 'rpn_class_loss' and 'mrcnn_class_loss' more significant, and thus have a higher impact on the overall loss, balancing out the individual components' contribution to the total loss. The results for these runs are displayed in Table 5.3.

From these results, it was clear that, overall, the weighted losses resulted in a significant improvement over the equally weighted losses for all backbone choices. Thus, going forward, all parameter sets will use weighted losses. Since ResNet101 with weighted losses has performed the best so far, the following hyperparameters changes will be applied to it, and not to all parameter combinations. Similarly, the 'custom' backbone will also be 'brought forward' since it performed comparably, with shorter training times.

5.4.3 | Impact of image pre-processing

In all previous runs, the singular channel of each image (after normalising with ZScale transform) was being replicated over all 3 channels in order to conform with the 3-channel (RGB) data structure required as input by the Mask R-CNN network architecture, as explained in Section 5.2.1.1. The next set of runs were aimed to determine whether the model could learn better if a different ZScale contrast parameter was used for each channel. This was tested with the 'resnet101' and the 'custom' backbones, both with weighted losses and ZScale contrasts of 0.25, 0.5, 0.75 (3-1 and 3-2 respectively). The results for these runs are displayed in Table 5.3.

These parameter sets produced models with comparable performances to those reported in Section 5.4.2, however, the models in Section 5.4.2 perform slightly better, so are kept as the benchmark.

5.4.4 | Impact of mask loss function

As discussed in Section 5.2.2, Dice Loss was implemented as an alternative mask loss function to binary cross-entropy, as, in theory, it seemed like a more appropriate algorithm, given the typically smaller-sized objects of interest in the data. The effects of this on performance were again tested using the 'resnet101' and 'custom' backbones with weighted losses, with dice loss as the mask loss function (4-1 and 4-2 respectively). The results for these runs are displayed in Table 5.3.

Once again, with the new set of parameters, no improvement was registered over the performance achieved by the parameter sets in Section 5.4.2.

5.4.5 | Impact of RPN anchor size

When inspecting sample sources that were missed by the models tested in Section 5.4, it was observed that a fraction of them are very extended or elongated (e.g. with size comparable to the cutout size), or very small (segmentation mask of a few pixels). One potential solution to increase the detection chances for them is training the model with alternative values of the `RPN_ANCHOR_SCALES` and `BACKBONE_STRIDES` parameters, which are mostly controlling the object scale to which Mask R-CNN is sensitive to (Riggi et al., 2023).

Several tests were run using `RPN_ANCHOR_SCALES` and `BACKBONE_STRIDES` both set to [4,8,16,32,64] (the default value used so far), [2,4,8,16,32], [4,8,16,32,128], [4,8,16,64,128], [2,8,16,32,128], [2,8,16,64,128], or [2,4,8,16,32,64], with `RPN_TRAIN_ANCHORS_PER_IMAGE`

and `TRAIN_ROIS_PER_IMAGE` both set to 256 (the default value used so far) or 128. These runs were repeated for the 'resnet101' and 'custom' backbone.

From the results obtained, it was evident that the parameter sets tested in this section produced models that either performed abysmally, or were not working as intended due to a limitation with the base Mask R-CNN implementation used, thus, results for these tests are not reported.

Table 5.2: Table showing the Hyperparameter Combinations attempted in Section 5.4. The Performance achieved by each Parameter Set is shown in Table 5.3.

Parameter Set	Backbone	Pre-Trained Weights	Loss Weights	RPN Anchor Scales Backbone Strides	RPN Train Anchors Train ROIs	Channels	Mask Loss Function
1-1	ResNet101	No	Equal	4,8,16,32,64	256	Equal	B CE
1-2	ResNet101	Yes	Equal	4,8,16,32,64	256	Equal	B CE
1-3	ResNet50	No	Equal	4,8,16,32,64	256	Equal	B CE
1-4	Custom	No	Equal	4,8,16,32,64	256	Equal	B CE
2-1	ResNet101	No	Weighted	4,8,16,32,64	256	Equal	B CE
2-2	ResNet101	Yes	Weighted	4,8,16,32,64	256	Equal	B CE
2-3	ResNet50	No	Weighted	4,8,16,32,64	256	Equal	B CE
2-4	Custom	No	Weighted	4,8,16,32,64	256	Equal	B CE
3-1	ResNet101	No	Weighted	4,8,16,32,64	256	Contrasts	B CE
3-2	Custom	No	Weighted	4,8,16,32,64	256	Contrasts	B CE
4-1	ResNet101	No	Weighted	4,8,16,32,64	256	Equal	Dice
4-2	Custom	No	Weighted	4,8,16,32,64	256	Equal	Dice

Table 5.3: Table showing the Performance achieved by the Hyperparameter Combinations described in Section 5.4 and shown in Table 5.2.

Parameter Set	Reliability (Precision)			Completeness (Recall)			F1-Score				Avg. Epoch Time (s)
	Sidelobe	Source	Galaxy	Sidelobe	Source	Galaxy	Sidelobe	Source	Galaxy	All Classes	
1-1	0.15	0.50	0.85	0.44	0.88	0.95	0.22	0.63	0.89	**0.63**	2,450
1-2	0.22	0.40	0.97	0.44	0.92	0.89	0.29	0.56	0.93	**0.58**	2,460
1-3	0.34	0.57	0.95	0.45	0.90	0.94	0.39	0.70	0.94	**0.71**	2,220
1-4	0.13	0.44	0.96	0.44	0.91	0.91	0.20	0.59	0.93	**0.59**	2,150
2-1	0.47	0.66	0.91	0.23	0.86	0.90	0.31	0.75	0.91	**0.75**	2,400
2-2	0.12	0.53	0.82	0.42	0.84	0.92	0.18	0.65	0.87	**0.63**	2,450
2-3	0.30	0.59	0.95	0.27	0.86	0.91	0.29	0.70	0.93	**0.71**	2,190
2-4	0.22	0.59	0.94	0.24	0.91	0.85	0.23	0.71	0.89	**0.71**	2,070
3-1	0.20	0.61	0.94	0.42	0.88	0.89	0.27	0.72	0.91	**0.71**	2,370
3-2	0.23	0.55	0.91	0.35	0.90	0.88	0.28	0.68	0.90	**0.68**	1,990
4-1	0.35	0.56	0.95	0.12	0.91	0.81	0.18	0.69	0.87	**0.69**	2,400
4-2	0.20	0.40	0.89	0.28	0.88	0.85	0.23	0.55	0.87	**0.57**	2,020

5.5 | Results

The best performing parameter set according to overall F1 Score, was the model of Section 5.4.2 with parameter set 2-1 (see Table 5.3), achieving an F1-Score of 0.75, and the following metrics performances for sidelobes, sources, and galaxies respectively, as shown in Table 5.3: Reliability/Precision=(0.47, 0.66, 0.91), Completeness/Recall=(0.23,

Table 5.4: Table showing the Reliability, Completeness, F1-Score, and mAP50 for the best performing model.

	Sidelobe	Source	Galaxy
Reliability (Precision)	0.47	0.66	0.91
Completeness (Recall)	0.23	0.86	0.90
F1-Score	0.31	0.75	0.91
mAP50	0.21	0.83	0.80

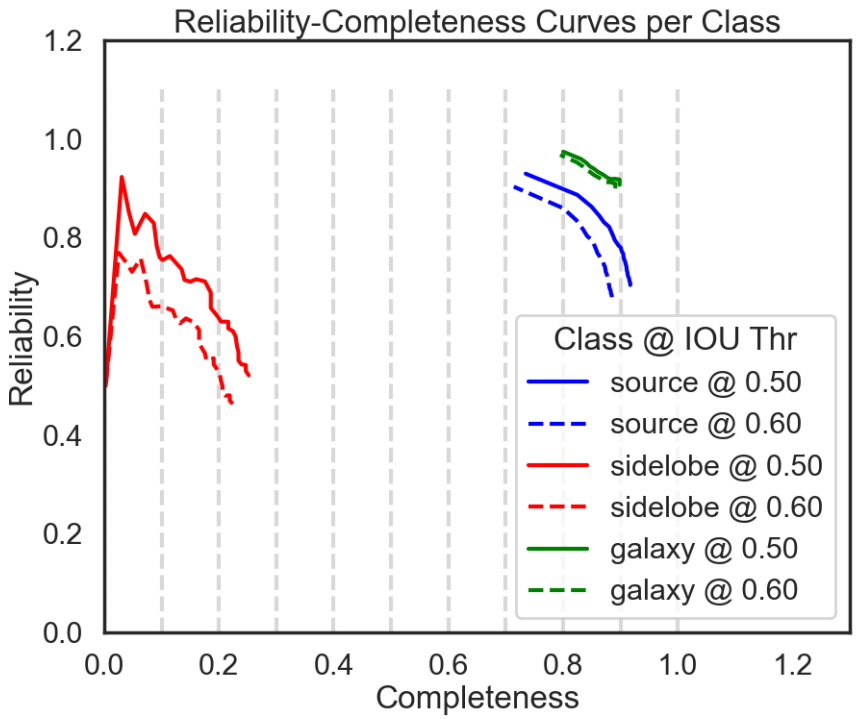

Figure 5.4: Reliability-Completeness Curves per Class at IoU Thresholds of 0.5 and 0.6.

0.86, 0.90), F1-Score=(0.31, 0.75, 0.91). With an IoU threshold of 0.5 (50%), the best model reached an mAP of 21% for sidelobes, 83% for sources, 80% for galaxies, and 76% overall, as shown in Table 5.4. Reliability-Completeness (Precision-Recall) Curves for each class (red: sidelobes, blue: sources, green: galaxies) at IoU Thresholds of 0.5 and 0.6 are shown in Fig. 5.4 with solid and dashed lines, respectively.

From the metrics presented, it is evident that the performance on sources and galaxies is rather promising, but below expectations for sidelobes. It can be argued that the poor performance achieved for sidelobes is only due to their relatively small representation in the dataset (1,280 sidelobes, compared to 19,000 sources). Class imbalance, how-

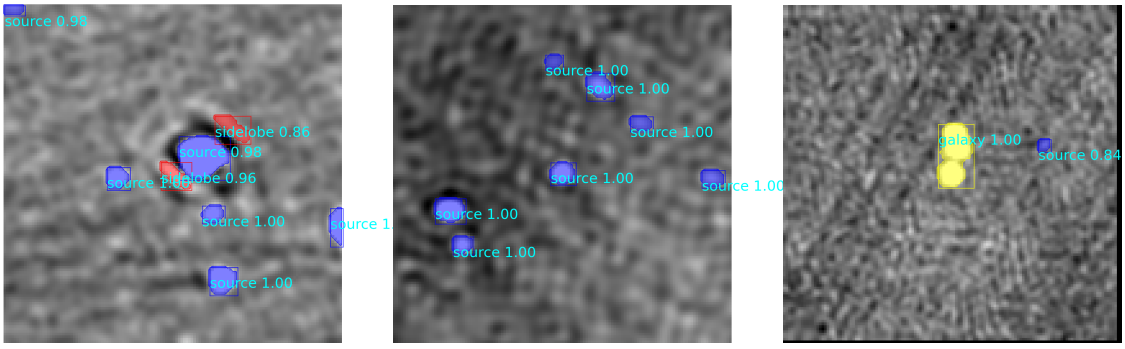

Figure 5.5: Sample of instances where the trained model performs as expected, and correctly detects, classifies, and segments objects within the images. Left panel: ASGARD's output on an image with both labelled compact sources and sidelobes. Centre panel: ASGARD's output on an image with only compact sources. Right panel: ASGARD's output on an image with both labelled galaxies and compact sources.

ever, does not seem to affect galaxies, for which there also are not as many instances available (3,200 galaxies). The results obtained on sidelobes are thus more likely due to their shape, as the Mask R-CNN architecture is known not to perform as well on thin and elongated shapes (Looi, 2019), "due to their thinness and curvature and therefore small area relative to the area of their associated ROIs" (Frei and Kruis, 2021). That being said, it is important to appreciate that no other automated solution attempts to detect sidelobes, let alone an 'all in one' solution that detects, classifies, and segments them, so this is an overall promising first step.

The following subsections delve deeper into the obtained metrics, highlighting the major achievements of the trained model, as well as the limitations.

5.5.1 | A Qualitative Look at the Model's Performance

In Fig. 5.5 a sample of images are shown where the model performs particularly well and detects, classifies, and segments the objects inside the images as expected. For example, in the left panel, all the objects inside the image are properly detected, segmented, and classified with a high score, including the two sidelobes and the very bright source at the image centre, as are the two sources partially cut at the image top-left and right-hand side border. The centre panel again shows a number of sources, all of which are correctly detected and classified, with a very good segmentation. Finally, the right panel shows a 2-component galaxy that is correctly classified, with the two masked blobs segmented as a singular object.

On the other hand, Fig. 5.6 shows instances where the model did not perform what

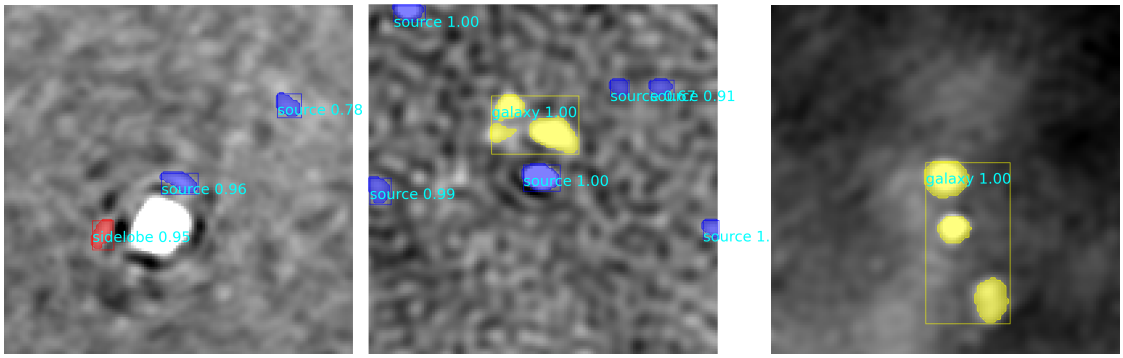

Figure 5.6: Sample of instances where the trained model does not perform as expected, and does not detect certain objects, or misclassifies them. Left panel: ASGARD's output on an image with both labelled compact sources and sidelobes. The bright source at the centre of the image is missed. Centre and Right panels: ASGARD's output on two sample images with only compact sources labelled. Some of them are misclassified as radio galaxies (shown in yellow).

was expected. The left panel shows a very bright source which was not detected, as well as the sidelobe above it incorrectly classified as a source. This image, and several others like it, confirm the metrics presented in Section 5.5, where the performance on sources and galaxies is respectable, however that on sidelobes is, so far, lacking. The centre and right panels both show 3 separate sources incorrectly classified as a galaxy with 3 component islands.

When inspecting the results, instances in which the model detected more real sources in the image than the ones actually labelled in the image as the ground truth were encountered. During evaluation, these extra detections are counted as spurious and slightly decrease the overall reliability of the model, whereas the cause is rather due to the dataset annotation. A further revision of the entire dataset is therefore planned for future releases of the model.

5.5.2 | Performance against SNR

The SNR of an image is defined as the ratio of pixels in an image that are considered information to background noise pixels, i.e. how many of the electrons received by the interferometer actually belong to an object (e.g. a source) and how many are noise. The higher the SNR value for an image, the better the quality of that image. SNRs less than 5 represent images with barely anything detected ($\sim$50% noise at SNR$\approx$2), SNRs between 5 and 10 images with detected objects ($\sim$20% noise at SNR$\approx$5), and SNRs greater than 10 images with reliable detections ($\sim$10% noise at SNR$\approx$10) (Hainaut, 2015). A compact

Table 5.5: Table showing the model's performance on each object class for varying SNR ranges. Note: Values for Sidelobes at SNRs < 5 and SNRs < 10 and Galaxies at SNRs < 5 are not present, as there were no instances of these in the dataset.

SNR	F1 Score		
	Sidelobe	Source	Galaxy
< 5	N/A	0.59	N/A
> 5	0.31	0.76	0.91
< 10	N/A	0.70	0.59
> 10	0.31	0.76	0.91
Overall	0.31	0.75	0.91

tool was developed for calculating the SNR of a given set of images, called CalculateAstroSNR[18], using the equation shown in Equation 5.1. Having the SNR values for the test set images in hand allows for the aforementioned performance metrics to be put in the context of the SNR of the data.

$$SNR = \frac{S_{peak}}{\sigma_{bkg}} \tag{5.1}$$

where S_{peak} is the source peak brightness (i.e. the highest brightness of the pixels found in the source segmentation mask), and σ_{bkg} is the standard deviation of the 3σ clipped distribution of background pixel brightness (i.e. pixels not included in any source segmentation mask for that ground truth image) (Sortino et al., 2023a).

Fig. 5.7 shows how the performance of the model (F1-Score) changes for each class across different SNR values. Some SNR bins do not include examples of certain classes, e.g. in this dataset, galaxies have SNRs > 5, whereas sidelobes are only found with SNRs > 200. As expected, the performance achieved for both sources and galaxies improves with the SNR, by ∼20% going from SNRs < 5 to SNRs well above 10. Fig. 5.7 also shows how the performance drops for SNRs > 200, which does not sound right at first, however one must consider that all of the sidelobes in this dataset exist in this range, and, as discussed in Section 5.5, the model struggles most with images containing sidelobes, so this explains the performance drop for this SNR range. F1-Scores are also reported in Table 5.5 for representative SNR ranges. The performance achieved on sources increases from 0.59 for SNRs < 5 to 0.76 for SNRs > 5. The same can be said for galaxies, which score 0.59 at SNRs < 10, and 0.91 at SNRs > 10.

[18]The code written for CalculateAstroSNR is publicly available on a GitHub repository (`https://github.com/SKA-INAF/CalculateAstroSNR`), under the GNU General Public License v3.0 (`https://www.gnu.org/licenses/gpl-3.0.html`, `https://github.com/SKA-INAF/CalculateAstroSNR/blob/main/LICENSE`).

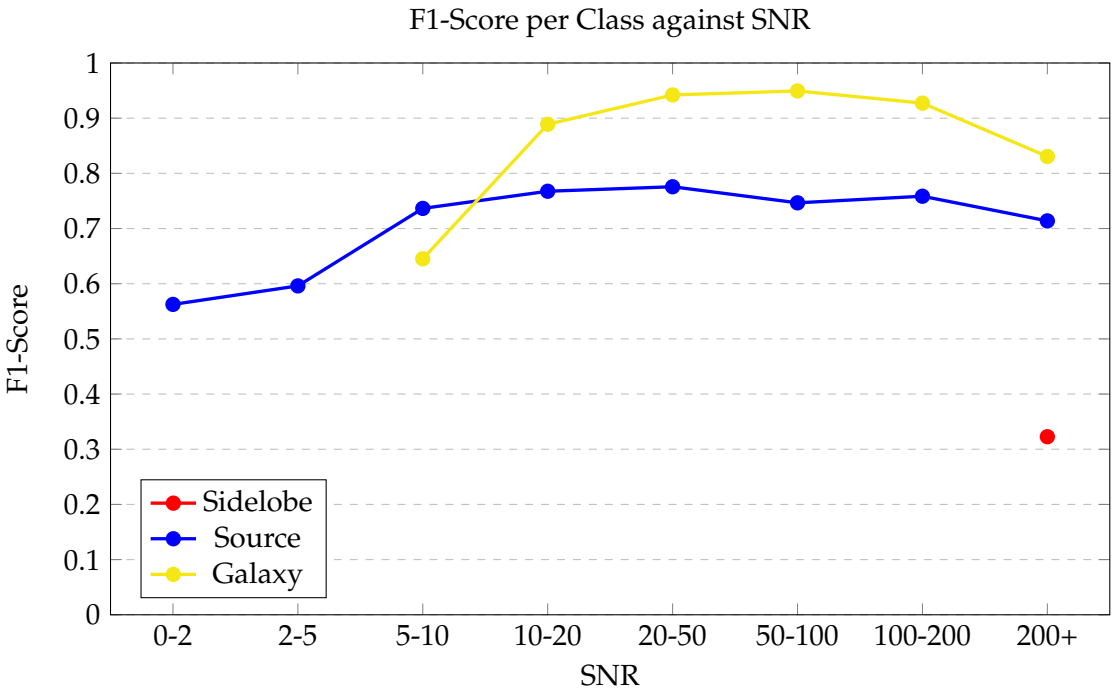

Figure 5.7: Graph showing the Performance (F1-Score) of the trained model on each class for images at different SNR values. Note: Values for Galaxies at SNRs < 5 and Sidelobes at SNRs < 200 are not present, as there were no instances of these in the dataset.

5.5.3 | Performance for Different Radio Surveys

As discussed in Section 5.1, particularly Section 5.1.1, the dataset used to train and evaluate ASGARD is made up of images from different surveys. This of course means that images, and the objects within them, within the dataset from different surveys will have different resolutions, frequency bands, PSFs and primary beams, background conditions and noise patterns, etc. It stands to reason that this variety of parameters will mitigate the possibility of the model overfitting to any one of these mentioned properties of a telescope or survey, and thus enhance its generalisation capabilities for new data from different sources. This does, however, require a rough balance of instances of each object from each survey to fully reap the aforementioned benefit. In its current state, the dataset does not achieve this balance, nor do others such as the RGZ DR1 dataset used as part of this dataset. Given the lack of an (as of yet) properly balanced dataset, such that the model cannot be expected to learn to properly generalise across images from distinct surveys, it is interesting to explore how the trained model's performance changes when evaluated on the individual surveys constituting the entire dataset.

Therefore, the model metrics were computed on ASKAP and RGZ FIRST images separately, which jointly represent ∼77% of the images making up the dataset (see Section 5.1.1). On ASKAP data, the model obtained an F1-Score of 0.5 for galaxies (0.33, 0.80, and 0.87 on 3-, 2-, and 1-component galaxies, respectively), whereas on RGZ data, the model scored 0.96 (0.97 and 0.94 on 2- and 3-component galaxies, respectively). The inferior performances on ASKAP data can likely be attributed to the limited number of samples available (132 ASKAP galaxies in the training set, only 10 of which with 3 components), compared to RGZ data (2,097 samples), which make up ∼93% of galaxy samples in the dataset. On the other hand, the F1-score for sources on RGZ data is ∼0.1, whereas for ASKAP it is ∼0.79 (greater than the performance achieved on the full test set). Again, this is likely due to the survey imbalance in the training dataset, in which ∼74% of the compact sources are currently provided by ASKAP. In part, lower performances on RGZ sources may also be due to the already noted limitations with relatively small objects, such as some labelled RGZ FIRST sources.

Taking this into consideration, it can be concluded that the survey data imbalance is indeed affecting the model performance. This issue is being addressed with the addition of new samples from other ASKAP pilot survey fields and from other SKA precursor surveys.

5.5.4 | Comparison with Existing Solutions

In order to understand the significance of the aforementioned performances and results achieved, this subsection will try to compare them with other existing and widely used solutions, to put the numbers into context. However, it is important to keep in mind that all of these solutions were trained and evaluated on different datasets, each with its own dataset size, SNRs, having real vs simulated data, etc., thus it is very difficult to draw any direct comparisons. Furthermore, all of these solutions have varying features, in terms of what types of objects they have been developed to detect and/or distinguish between, as well as the type of outputs they produce (e.g. classification vs semantic segmentation vs instance segmentation), and, hence, are evaluated differently. Table 3.1 in Section 3.3 highlights the main features and differences of the solutions compared.

5.5.4.1 | Comparison with *ConvoSource*

ConvoSource (Lukic et al., 2020) is a CNN-based solution for source-finding, described in more detail in Section 3.3.1.1. Unlike ASGARD which works on sidelobes, sources, and galaxies (without delving into specific subtypes), ConvoSource was trained to find compact and extended SFGs and AGNs (steep- and flat-spectrum), and was trained on the

SDC1 (Bonaldi et al., 2020) simulated dataset. Furthermore, as indicated, ConvoSource is a source finder, i.e. it finds the aforementioned objects in images, however does not perform any classification, nor instance segmentation as does ASGARD. Taking all of this into account, the ConvoSource model somewhat corresponds to an ASGARD model trained on a single class dataset, including both 'sources' and 'galaxies'. It is also important to understand that *ConvoSource* is trained and tested on simulated images, whereas ASGARD is tested on real images with realistic noise from the Galactic plane.

From the TP, FP, and FN rates reported by Lukic et al. (2020), the F1-Score can be computed. For the B1_8h, B1_1000h, B2_8h, and B2_1000h datasets, the F1-Scores at SNR=2 are 0.78, 0.69, 0.76, and 0.61, and at SNR=5 are 0.69, 0.78, 0.85, and 0.73, respectively. ASGARD achieves an F1-Score of 0.59 at SNR < 5, 0.74 for SNR 5-10, and 0.75 overall for sources, and 0.65 at SNR 5-10, 0.89 for SNR 10-20, and 0.91 overall for galaxies (there are no galaxies in the dataset at SNR < 5). However, it must also be taken into consideration that ASGARD also learns to detect, classify, and segment sidelobes, which especially hamper ASGARD's overall performance.

In order to obtain a fairer comparison, the best performing model was trained in the exact same way again, however this time without sidelobes, i.e. only with sources and galaxies. This resulted in a significant improvement of the F1-Score to 0.73 (SNR < 5), 0.82 (SNR 5-10), and 0.81 overall for sources. For galaxies the score achieved was 0.58 (SNR 5-10), 0.90 (SNR 10-20), and 0.93 overall for galaxies. The discrepancies observed may very well be a result of the methodologies used and, most importantly, due to the differences between the datasets (e.g. the homogeneous simulated data of SDC1 against real data mixed from different surveys), possibly penalising ASGARD more.

5.5.4.2 | Comparison with *DeepSource*

DeepSource (Vafaei Sadr et al., 2019) is another CNN-based source finder for point-sources, i.e. it does not classify objects, nor does it generate masks, as detailed in Section 3.3.1.2.

Vafaei Sadr et al. (2019) report a Purity (Reliability/Precision) of 0.45 and a Completeness (Recall) of 0.85 for an SNR of 3. They also include an additional metric, PC, which is the product of the Purity and Completeness, which comes out to 0.38 for SNR = 3. For SNR values of 4 and 5 the Purity, Completeness, and PC are all above 0.99. The general ASGARD model (which is also trained on sidelobes and galaxies) achieves a Purity (Reliability) of 0.66, a Completeness of 0.86, and a PC of 0.57 on sources. Similarly to what was done in the previous comparison, a model was trained purely on sources. This model achieves a Purity of 0.65, a Completeness of 0.92, and a PC of 0.60.

5.5.4.3 | Comparison with *ClaRAN*

ClaRAN (Wu et al., 2018) is a radio source detector and morphology classifier based on Faster R-CNN (Ren et al., 2017), as discussed in further detail in Section 3.3.1.3. ClaRAN locates objects in images and draws bounding boxes around them, and also classifies the detected objects' morphologies in terms of components and peaks. This is different from ASGARD in that ClaRAN does not generate a per-pixel mask, and that ClaRAN does not attempt to detect sidelobes.

Of the 5 pre-processing methods ClaRAN was tested with in Wu et al. (2018), the worst performing achieves an overall mAP of 77.4%, with the best scoring an mAP of 83.6% overall. ClaRAN here outperforms ASGARD's 76.0% overall mAP. However, it should be considered that ClaRAN was trained and evaluated only on the (modified) RGZ DR1 dataset (a subset of which is used in this work), which not only has an additional channel of data (infrared), but was also found to only contain 3 instances of sidelobes in 2,974 images, as shown in Table 5.1. This is in stark contrast to ASGARD, which only had access to one channel, radio, and was trained and evaluated on a dataset containing 1,287 sidelobes in 9,205 images.

5.6 | Conclusions

This work introduces a novel source finding tool, based on the state-of-the-art Mask R-CNN DL model, for detecting, segmenting, and classifying radio compact sources, extended galaxies, and imaging sidelobes from radio continuum maps. The developed source finder, ASGARD, was trained and evaluated on a dataset composed of images extracted from different radio surveys, including ASKAP EMU ESP and RGZ DR1. A hyperparameter optimisation process was also carried out to maximise the model's performance. Overall, promising reliability/completeness (precision/recall) performance metrics (above 80%) for compact sources and radio galaxies were achieved. Unfortunately, the performances achieved on sidelobes did not meet expectations, especially when compared to the other two classes. Regardless, ASGARD marks a significant breakthrough in the application of DL to radio astronomy data, and not only improves upon an entirely manual approach, which is still prevalent, but sets the stage for further state-of-the-art DL algorithms to be applied.

While Section 5.5.4 attempts to compare ASGARD to existing solutions quantitatively, this does not fully capture what was achieved, as no direct comparison could be performed, given that no existing solution has all of ASGARD's features and capabilities.

These promising results encourage further work to address some of the limitations mentioned, and to improve the model's performance. These are discussed in the following section, Section 5.6.1.

5.6.1 | Future Work

As concluded in Section 5.5.3, the dataset's class and survey imbalance is affecting the model's performance. In order to mitigate this imbalance, the size of the dataset is currently being increased with additional radio survey data from ASKAP and other SKA precursors. DL models for generating further synthetic data from data already in the dataset to balance out any underrepresented classes are also being explored, such as RADiff (Sortino et al., 2024). As discussed in previous sections, this is expected to lead to an improvement in source and galaxy detection capabilities.

One potential solution to addressing the unsatisfactory performance achieved on sidelobes is to explore other DL models, using the same dataset as used for ASGARD in this work. On paper, the architecture of Rotated Mask R-CNN[19] suggests that it could be well-suited and effective for the detection of sidelobes, as Frei and Kruis (2021); Looi (2019) claim it achieves better performance on objects with elongated morphologies (compared to a more vanilla implementation of Mask R-CNN). Another DL model, Tiramisu (Pino et al., 2021), achieved better performance on sidelobes when trained and evaluated on this dataset.

While ASGARD is capable of processing very large images, such as those produced by ASKAP pilot and future surveys, it is important to also mention that this is still a preliminary version of the feature, and is something that needs to be further refined.

Finally, work is being carried out to enable ASGARD to integrate into a larger pipeline (e.g. as the classifier), such that it can interface with other already existing source finders (e.g. perform classification on these tools' output catalogues), rather than being limited to function only in a standalone manner, as is the case in this work. A pipeline with this arsenal of tools complementing each other would be useful for the entire astronomical community, in all fields, and not only radio astronomy. Thus, it is intended for ASGARD to be incorporated into the upcoming European Open Science Cloud (EOSC) infrastructure, as a service. In a similar vein, the Novel EOSC Services for Emerging Atmosphere, Underwater & Space Challenges (NEANIAS) (Sciacca et al., 2020) and Collaborative and Integrated platform for Radio Astronomical Source Analysis (CIRASA) (Riggi et al., 2021a) are developing pilot systems for source finding and visualisation of astronomical data on the cloud. Such a cloud-based approach makes these tools scalable, i.e. they can

[19] `https://github.com/mrlooi/rotated_maskrcnn`

be executed on systems with considerably more computational power, such as those outlined for deployment in SKA Regional Centres. As detailed in Riggi et al. 2021a, ASGARD has already been implemented as one of the space services[20] as a supported application. However, as yet, its functionality is limited to smaller images or cutouts, but it is intended for these services to eventually include the entire pipeline.

[20] `https://github.com/SKA-INAF/caesar-rest`

6

Conclusions

In Chapter 4, the framework LEXACTUM was developed, which improves on the initial work done in Metcalf, R. B. et al. (2019), registering improvements over several of the reported metrics in the initial work. While this solution effectively 'solves' the Gravitational Lensing Classification problem, its achievements should not be limited to that, as the developed framework makes it very easy to adapt to other problems, and for the models to be retrained on any other domain's data, even allowing for the easy inclusion of other architectures if required.

Then, in Chapter 5, a dataset from different radio surveys was compiled, labelled, and revised collaboratively. The Instance Segmentation solution, ASGARD, was developed. The developed tool makes it very easy to input any image of any size, including very large ones such as those from ASKAP or sample SKA data such as that from SDC1, and in return get the same image with each object's pixels highlighted, with a classification made for the object, supporting sidelobes, sources, and galaxies - something which, so far, did not exist for astronomical data. The results obtained, while lacking for sidelobes, were very respectable for point sources and galaxies. Furthermore, being trained on a dataset made up from different surveys, this model should be much more capable of generalising to unseen data. Similarly to LEXACTUM, ASGARD is highly adaptable to other problems from different domains, and can be retrained, or fine-tuned, to recognise different objects, or to better familiarise itself with data from other surveys before being applied to them.

Moreover, the trained models in LEXACTUM and ASGARD serve as a very good starting point for datasets where no training set is available. If the dataset is collected from a different telescope, it is expected that the models will not perform as well as they did on the datasets in this work. As mentioned, ASGARD is already trained on data from a variety of surveys (see Section 5.1), so this alone helps it generalise further to

unseen data with different characteristics.

In fact, in an effort to quantify this, in Riggi et al. (2023), Mask R-CNN was retrained only on the data from the VLA telescope. This model of Mask R-CNN trained only on VLA data was then evaluated on the rest of the dataset, i.e. the data from ASKAP and ATCA. It resulted that this model scored, on average, 10% less on the unseen data from the different telescopes, when compared to the performance when a mixture of data from all the telescopes was used. This result not only proves that the mixture of data from different surveys and telescopes helped the model generalise, but also that when applying the model to data from completely different telescopes, the model still achieves comparable performance, albeit at a small, but expected, penalty.

While this shows the versatility of the trained models, it must be noted that the performance is ultimately degraded. This decline in performance is not limited to models trained on a specific telescope being applied to completely different telescopes, but can also occur when the characteristics of the images generated by a telescope change. Signals can be contaminated due to astrophysical foregrounds, the Earth's ionosphere, instrumental systematics, and RFI, to name a few, all of which create variations in the characteristics of the generated images.

Astrophysical foregrounds, such as cosmic dust emitting thermal radiation, bright sources, AGNs, and SFGs, can contaminate signals, particularly when capturing fainter signals. While these are considered mostly stable for localised regions over shorter spans of time, these can vary over time or for different regions of space. On the other hand, the Earth's ionosphere is a more variable factor, that can refract radio waves, particularly for lower frequencies. Instrumental systematics, such as calibration, are yet another factor that affects the images created by telescopes. Telescope calibration can degrade over time, and together with ageing components and variations in thermal noise, this can potentially hamper AI models' performance. Finally, RFI can also affect the images telescopes generate. The SKA, for example, was constructed in radio quiet zones to minimise human-made RFI. This, however, does not exclude it from other sources of RFI, such as thunderstorms or solar flares, which are sporadic over time (Mazumder et al., 2022).

Furthermore, works like Sortino et al. (2023a,b), which applied various models to a dataset very similar to ASGARD's, found significant improvement when starting training from pre-trained weights, rather than from scratch, particularly for transformers. Interestingly, the pre-trained weights used were from models trained on the ImageNet or COCO (Lin et al., 2014) datasets, which feature colourful, tangible, every-day objects, significantly different in nature from astronomical images. Thus, for a new survey, even if only a small training set is available, this can be used to simply fine-tune any

of the models in LEXACTUM, or ASGARD, that have been specifically pre-trained on astronomical data.

Additionally, models pre-trained on e.g. COCO expect RGB images, i.e. images for which the pixel value for any channel is an integer, typically in the range 0 to 255 (8 bits). Astronomical images, on the other hand, are typically stored in FITS files, which store significantly more precise values, up to 64-bit floating point numbers. This means that for a pre-trained model to properly accept astronomical data, the data's range must be compressed to the range of 0 to 255 integers. This results in adjacent pixels with subtle variations being mapped to the same integer value, losing precision and granularity. Ultimately, this harms models' performance, again highlighting the relevance and paramount importance of readily available models pre-trained on astronomical data.

6.1 | Future Work

For the solutions mentioned in this chapter, future work which can address the existing limitations have been discussed in Sections 4.6.1 and 5.6.1.

The dataset for training ASGARD in Chapter 5 will also be released and made publicly available. Furthermore, ASGARD should be retrained without normalising the input data to integers in the range 0 to 255 (as specified in Section 5.2.1.1). Originally, this had been done for compatibility with pre-trained weights, allowing for the experimentation and comparison of that option. The weights obtained from this process would be made publicly available, serving as a starting point for other datasets.

Different model options can also be provided. For example, models trained only on subsets of the dataset, such as limiting the training data to a specific telescope, or specific survey (see Table 5.1 in Section 5.1). This could potentially allow users to select the most appropriate, or similar, dataset for their specific dataset and use case.

One common advancement that can be made to both LEXACTUM for the classification of gravitational lensing (Chapter 4) and ASGARD for the instance segmentation of astronomical objects (Chapter 5) is the application of further state-of-the-art architectures.

Despite architectures such as Lens Detector (Thuruthipilly et al., 2022), a transformer-based architecture, being applied to lensing classification since the development of LEXACTUM, there is still room for e.g. ViT (Dosovitskiy et al., 2021) (see Section 3.3.5.1), a completely Transformer approach to be implemented and evaluated. Furthermore, OmniVec (Srivastava and Sharma, 2024), a state-of-the-art (as of 2024) classification model (Papers With Code, a), can also be implemented.

Since the development of ASGARD in Chapter 5, works such as Sortino et al. (2023a,b) have applied further architectures, including transformers, on a very similar dataset. Even so, there are still several models to be experimented with, along with ones that have been released since. Some examples include YOLOv8 (Jocher et al., 2023) (see Section 3.3.3) and SOLOv2 (Wang et al., 2020b) (see Section 3.3.4). Similarly, current state-of-the-art (as of 2024) instance segmentation models, such as EVA (Fang et al., 2023), InternImage(-H) (Wang et al., 2023), and PANet++ (Zamir et al., 2019) can be implemented (Papers With Code, b). It is noteworthy that two of these three state-of-the-art architectures, InternImage and PANet++, are CNN-based, and not transformers.

Another improvement that has already shown promise in LEXACTUM is the application of Dataset Augmentation, allowing models to train for longer without overfitting. Further techniques could be applied, such as those mentioned in Section 2.2.1.2 or Wang et al. (2024b). Moreover, generative models such as RADiff (Sortino et al., 2024) could provide a significant boost in performance by generating more varied, yet realistic, training data.

An additional, and as yet undiscussed, avenue for achieving better (e.g. classification) performances does not lie in the AI models applied, but in the data used. Modern telescopes, such as the SKA, are capable of capturing broadband continuum images. This refers to the capability of telescopes capturing a broad range of frequencies when imaging the sky, as opposed to a single (or narrowband) frequency. Observing the behaviour of emissions over the range of frequencies has proven to be a very strong discriminator between imaging artefacts (sidelobes) and true sky emissions.

Such broadband continuum data, together with the software to effectively leverage it, has the potential to notably boost models' precision by effectively differentiating between imaging artefacts and true sky emissions (Heywood et al., 2016). In Riggi et al. (2023), over 14% of 'spurious' sources (sidelobes) were misclassified as 'compact' sources, substantiating models' struggle with correctly classifying sidelobes.

A

Detailed Descriptions of CNNs implemented in LEXACTUM

Some of the content in this appendix, Appendix A, incorporates concepts discussed in Magro et al. (2021).

This appendix will describe the CNNs mentioned in Section 3.1.2 in greater detail.

A.1 | CAS Swinburne

This architecture is based on AlexNet (Krizhevsky et al., 2012). The input image first undergoes three consecutive convolutional layers, with kernel sizes of 11, 5, and 3, and 96, 128, and 256 feature maps, respectively. A ReLU activation function and a 3x3 max pooling layer follow each of the convolutional layers. The output of the last max pooling layer is passed to two successive fully-connected layers, each having 1,024 neurons, and each followed by a ReLU activation and a dropout layer with 0.5 probability. Finally, the last layer is fully-connected to a single neuron with a sigmoid activation, which represents the model's output (Jacobs et al., 2017; Metcalf, R. B. et al., 2019). This architecture is shown graphically in Fig. 3.1.

A.2 | LASTRO EPFL

This architecture resembles that described in Section 3.1.2.1, in that they are both made up of the same building blocks, however 'lastro_epfl' is a considerably larger model, with almost twice as many layers. The input image is first passed through 3 'blocks', each comprising a pair of convolutional layers with ReLU activations, a max pooling layer, and a batch normalisation layer. The convolutional layers in these 3 blocks all

utilise a kernel size of 3, save for the first convolution inside the first block, which uses a kernel size of 4. The convolutional layers inside the first block produce 16 features each, those in the second produce 32, and the third 64. In each block, the max pooling layer is a 2×2 one. The final block is followed by a dropout layer with 0.1 probability, added to reduce the possibility of overfitting. This is then followed by a convolutional layer (with a kernel size of 3, producing 128 features, and a ReLU activation), a dropout layer, another convolutional layer with the same configuration, batch normalisation, and another dropout layer. The output from the last layer is then flattened, and connected to a triple of fully-connected layers, each having 1,024 neurons and a ReLU activation, with a dropout layer in between each pair. Batch normalisation is applied after the last of these fully-connected layers. Finally, the output is obtained from a final fully-connected layer, with a single neuron and a sigmoid activation function (Metcalf, R. B. et al., 2019; Schaefer et al., 2018). This architecture is shown graphically in Fig. 3.2.

A.3 | CMU DeepLens

When compared to the architectures described in Sections 3.1.2.1 and 3.1.2.2, CMU DeepLens shows similarities as it is also ultimately based on CNNs, however distinguishes itself as it makes use of ResNets (see Section 2.2.1.2). A ResNet is a network in which there exist "shortcut connections" from the input to the output of a series of convolutional layers. This local structure, i.e. the input to a series of convolutional layers, the convolutional layers themselves, and their output, will be referred to as a 'ResNet block'. The output of a 'ResNet block' is computed as the sum of the block's original input, and the output of the last convolutional layer within the block. One major advantage of ResNets is that, through these skip connections, the 'vanishing gradient problem' (see Section 2.2.1.1) is mitigated, as back propagation is given a 'shorter' route to 'reach' the earlier layers within an architecture.

CMU DeepLens makes use of two different architectures of ResNet blocks. The first of these maintains the original resolution of the image. This block first stores a copy of its input. Next, the input goes through a sequence of batch normalisation, ELU activation, and a convolutional layer with another ELU activation, three times. Finally, the output of the ResNet block is returned as the sum of the original input and the output of the convolutional layers within the block. The second of these ResNet blocks downsamples the image by a factor of 2. In this block, batch normalisation and an ELU activation are first applied, before a copy of the tensor is stored. Then, a convolutional layer with a stride of 2 is applied, which is responsible for the downsampling of the block's input.

This is followed by a sequence of batch normalisation, ELU activation, and a convolutional layer with another ELU activation, twice. A convolutional layer with a stride of 2 is also applied to the aforementioned stored input, with which the dimensions of the input and the convolutional layers' output will match, after which they are summed, and returned as the block's output. The architectures of these ResNet blocks are shown graphically in Fig. 3.3.

The overarching architecture of the CMU DeepLens model is as follows. A convolutional layer with a kernel size of 7, and having 32 features, with an ELU activation, is first applied to the input, followed by a batch normalisation layer. 3 'non-downsampling' ResNet blocks, each with 32 features, follow, after which a triple of ResNet blocks is applied 4 consecutive times. Each triple consists of a downsampling ResNet block and 2 'non-downsampling' ResNet blocks. The ResNet blocks within each triple generate 64, 128, 256, and 512 features, respectively. The last ResNet block's output then goes through an average pooling layer, after which a fully-connected layer with a single neuron and a sigmoid activation function produces the model's prediction (Lanusse et al., 2018; Metcalf, R. B. et al., 2019). This architecture is shown in Fig. 3.4.

A.4 | WSI-Net

The WSI-Net model was not originally designed for astronomical applications, but to detect tumours in breast scans and classify them. The same architecture, up to the "classification branch", can be applied to classify an astronomical image (in this work, whether an image contains a lens). Its architecture resembles that of CMU DeepLens, described in Section 3.1.2.3, in that they are both based on ResNets. The original work does not specify hyperparameter values, and thus those mentioned here are what was found to produce the best results, empirically. In WSI-Net, the image is first passed through a convolutional layer with a kernel size of 7, generating 32 features, with an ELU activation function. This is followed by two ResNet blocks, as those described in Section 3.1.2.3, each producing 32 and 64 features, respectively, the second of which downsamples the image. Next, two blocks of convolutional layers, batch normalisation, and ReLU activation functions follow. The convolutional layers within the two blocks have kernel sizes of 1 and 5, respectively, and each generate 32 features, with an ELU activation. This is followed by a max pooling layer, and then by a fully-connected layer having 128 neurons. Finally, the model's prediction is calculated as the output of a fully-connected layer with a single neuron and a sigmoid activation (Ni et al., 2019). This architecture is represented graphically in Fig. 3.5.

A.5 | LensFlow

The architecture of LensFlow is relatively simplistic, and resembles the CAS Swinburne (Section 3.1.2.1) and LASTRO EPFL (Section 3.1.2.2) models in terms of the number and type of layers involved. The image is first passed through a 3×3 average pooling layer with a stride of 3. This is followed by 3 sets of convolutional layer + max pooling layer pairs. The convolutional layers have a kernel size of 5, 5, and 4, and generate 16, 25, and 36 features, respectively. All of these convolutional layers are followed by a ReLU activation, and all of these max pools are 2×2 with a stride of 2. Next, the max pool's output is fed into a fully-connected layer having 128 neurons and a ReLU activation function. This is also followed by a dropout layer with a probability of 0.5, which is only active during the training of the model. A fully-connected layer with a single neuron and a sigmoid activation function produce the model's final output (Pourrahmani et al., 2018). This architecture is shown graphically in Fig. 3.6.

A.6 | LensFinder

Despite its very simplistic architecture, even when compared to CAS Swinburne and LensFlow (Sections 3.1.2.1 and 3.1.2.5), at effectively only 6 layers, LensFinder still manages to perform very respectably. The original work does not specify hyperparameter values, and thus those mentioned here are what was found to produce the best results, empirically. Furthermore, for its application in this work, i.e. a binary classification problem, the final layer's activation function was changed from a softmax to a sigmoid. The model starts with 2 convolutional layer + max pool pairs. The convolutional layers have kernel sizes of 5 and 3, and produce 64 and 128 features, respectively, and have a ReLU activation. The max pooling layers are both 2×2. These are followed by a fully-connected layer with 128 neurons and a ReLU activation. Finally, the output is produced by a single neuron in a fully-connected layer with a sigmoid activation function (Pearson et al., 2018). This architecture is displayed graphically in Fig. 3.7.

B

Accepted LEXACTUM Command-Line Arguments and Default Values

Table B.1: Table listing all the command line arguments which are accepted by LEXACTUM, including a short description, and accepted values.

Flag	Description	Default Value	Accepted Values
Required Arguments			
dataset	Path to root/top directory of the dataset	Required	Path to Directory
Common Arguments			
train_or_load	Whether to train a model or load one from disk	train	train, load
model_name	Indicates which model architecture to train or the name of the trained model weights file to load	cmu_deeplens	cas_swinburne, lastro_epfl, cmu_deeplens, wsi_net, lens_flow, lens_finder, Trained model to load from disk
no_of_epochs	The number of epochs the model should train for	10	Integers
batch_size	The number of images per batch	8	Integers
augment_images	Whether to perform image augmentation on the training data during training	True	True, False

C

Default Mask R-CNN Parameters and Accepted Command-Line Arguments

Table C.1: Table showing the default Mask R-CNN Hyperparameters and the values used for ASGARD. Default Values from: `https://github.com/matterport/Mask_RCNN/blob/master/mrcnn/config.py`

Parameter	Default	ASGARD
BACKBONE	resnet101	resnet101
COMPUTE_BACKBONE_SHAPE	None	Default
BACKBONE_STRIDES	[4, 8, 16, 32, 64]	[4, 8, 16, 32, 64]
FPN_CLASSIF_FC_LAYERS_SIZE	1024	Default
TOP_DOWN_PYRAMID_SIZE	256	Default
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)	(4, 8, 16, 32, 64)
RPN_ANCHOR_RATIOS	[0.5, 1, 2]	Default
RPN_ANCHOR_STRIDE	1	Default
RPN_NMS_THRESHOLD	0.7	Default
RPN_TRAIN_ANCHORS_PER_IMAGE	256	256
PRE_NMS_LIMIT	6000	Default
POST_NMS_ROIS_TRAINING	2000	Default
POST_NMS_ROIS_INFERENCE	1000	Default
USE_MINI_MASK	True	False
MINI_MASK_SHAPE	(56, 56)	N/A
IMAGE_RESIZE_MODE	square	square
IMAGE_MIN_DIM	800	256

Table C.1 continued from previous page

Parameter	Default	ASGARD
IMAGE_MAX_DIM	1024	256
IMAGE_MIN_SCALE	0	Default
IMAGE_CHANNEL_COUNT	3	Default
MEAN_PIXEL	[123.7, 116.8, 103.9]	[0,0,0]
TRAIN_ROIS_PER_IMAGE	200	256
ROI_POSITIVE_RATIO	0.33	Default
POOL_SIZE	7	Default
MASK_POOL_SIZE	14	Default
MASK_SHAPE	[28, 28]	Default
MAX_GT_INSTANCES	100	100
RPN_BBOX_STD_DEV	[0.1, 0.1, 0.2, 0.2]	Default
BBOX_STD_DEV	[0.1, 0.1, 0.2, 0.2]	Default
DETECTION_MAX_INSTANCES	100	Default
DETECTION_MIN_CONFIDENCE	0.7	0
DETECTION_NMS_THRESHOLD	0.3	0.3
LEARNING_RATE	0.001	0.0005
OPTIMIZER	SGD	ADAM
LEARNING_MOMENTUM	0.9	N/A
WEIGHT_DECAY	0.0001	Default
LOSS_WEIGHTS	rpn_class_loss: 1.0, rpn_bbox_loss: 1.0, mrcnn_class_loss: 1.0, mrcnn_bbox_loss: 1.0, mrcnn_mask_loss: 1.0	rpn_class_loss: 1.0, rpn_bbox_loss: 0.1, mrcnn_class_loss: 1.0, mrcnn_bbox_loss: 0.1, mrcnn_mask_loss: 0.1
USE_RPN_ROIS	True	Default
TRAIN_BN	False	Default
GRADIENT_CLIP_NORM	5.0	Default

Table C.2: Table listing all the command line arguments which are accepted by ASGARD, including a short description, and accepted values.

Flag	Description	Default Value	Accepted Values
Required Arguments			
<command>	Indicates whether the model should train on the training set, evaluate performance of a trained model on the test set, or use a trained model to run detection on a provided image	Required	train/test/detect
Common Arguments			
–imgsize	Size the input image is resized to	256	Integers
–grayimg	Indicates that the image is in grayscale	False	N/A
–no_uint8	Indicates that values should not be converted to uint8	False	N/A
–no_zscale	Indicates that ZScale normalisation should not be applied	False	N/A
–zscale_contrasts	What ZScale contrast should be applied to each channel	0.25,0.25,0.25	0-1,0-1,0-1
–biascontrast	Indicates whether bias contrasting should be applied	False	N/A
–bias	Bias value to be applied (if –biascontrast is also present)	0.5	0-1
–contrast	Contrast value which should be applied	1.0	0-1
–no_norm_img	Indicates whether the input image should be normalised	False	N/A
–class_dict	What dictionary of classes to be used	{sidelobe: 1, source: 2, galaxy: 3}	Any
–classdict_model	What dictionary of classes should be used for the model	equal to class_dict	Any
–remap_classids	Indicates that classids should be remapped	False	N/A
–classid_remap_dict	What dictionary of classes should be used to remap classes from detections to the ground truth classes	equal to class_dict	Any

Table C.2 continued from previous page

–dataloader	What Dataloader type to use	datalist	datalist, datalist_json, datadir_json
–datalist	Path to the Dataset file list in the required format	<none>	Any path
–datalist_train	Path to the training set file list if the dataset has already been split	<none>	Any path
–datalist_val	Path to the validation file set if the dataset has already been split	<none>	Any path
–datadir	Path to the top directory of the dataset	<none>	Any path
–validation_ data_fract	What fraction of the dataset to dedicate to the validation set	0.1	0-1
–maxnimgs	The max number of images to consider in the dataset	-1 (all)	Integers
–weights	Pre-trained .h5 weights file to use if continuing training or running testing/evaluation or detection	<none>	Any path to a valid .h5 file
–logs	Where to store logs and weights files	logs/	Any path
–nthreads	Number of worker threads	1	Integers
Train Options			
–ngpu	Number of GPUs to use	1	Integers
–nimg_per_gpu	Number of images per GPU	1	Integers
–nepochs	Number of epochs to train the model for	1	Integers
–epoch_length	Number of data batches per epoch	None =>All samples	Integers
–nvalidation_ steps	Number of validation data batches per epoch	None =>All samples	Integers
–rpn_anchor_ scales	RPN Anchor Scales to use	4, 8, 16, 32, 64	Series of Integers
–max_gt_ instances	Max GT instances	300	Integers
–backbone	Backbone network to use	resnet101	resnet101, resnet50, custom

Table C.2 continued from previous page

–backbone_ strides	Backbone strides to use	4, 8, 16, 32, 64	Series of Integers
–rpn_nms_ threshold	RPN non-maximum-suppression threshold to use	0.7	0-1
–rpn_train_anchors_ per_image	Number of anchors to use per image	512	Integers
–train_rois_per_ image	Number of ROIs to feed to classifier per image	512	Integers
–rpn_anchor_ ratios	RPN Anchor Ratios to use	0.5, 1, 2	Series of Numbers
–rpn_class_loss_ weight	RPN Classification Loss weight modifier	1	Number
–rpn_bbox_loss_ weight	RPN Bounding Box Loss weight modifier	1	Number
–mrcnn_class_ loss_weight	Mask R-CNN Classification Loss weight modifier	1	Number
–mrcnn_bbox_ loss_weight	Mask R-CNN Bounding Box Loss weight modifier	1	Number
–mrcnn_mask_ loss_weight	Mask R-CNN Mask Loss weight modifier	1	Number
–(no_)rpn_class_ loss	Whether to use RPN Class Loss	True	N/A
–(no_)rpn_bbox_ loss	Whether to use RPN Bounding Box Loss	True	N/A
–(no_)mrcnn_ class_loss	Whether to use Mask R-CNN Class Loss	True	N/A
–(no_)mrcnn_ bbox_loss	Whether to use Mask R-CNN Bounding Box Loss	True	N/A
–(no_)mrcnn_ mask_loss	Whether to use Mask R-CNN Mask Loss	True	N/A
–mask_loss_ function	Which function to use to calculate Mask Loss	binary_ crossentropy	binary_ crossentropy, dice_coef_ loss
–weight_classes	Indicates that classes should be weighted	False	N/A

Table C.2 continued from previous page

Test Arguments			
–scoreThr	Object detection score threshold to be used during evaluation	0.7	0-1
–iouThr	IOU threshold used to match detections with true objects during testing/evaluation	0.6	0-1
–(no_)consider_sources_near_mixed_sidelobes	Indicates whether sources near or overlapping with sidelobes should be considered during evaluation	True	N/A
Detect Arguments			
–image	Image input on which detection should be run	<none>	Any path to an image
–xmin	From which x coordinate the image should be read	-1 (all)	Integers
–xmax	Up to which x coordinate the image should be read	-1 (all)	Integers
–ymin	From which y coordinate the image should be read	-1 (all)	Integers
–ymax	Up to which y coordinate the image should be read	-1 (all)	Integers
–detect_outfile	Filename the generated detection plot should be stored in	<none>	Any filename
–detect_outfile_json	Filename the json with detections should be stored in	<none>	Any filename
Parallel Processing Options			
–split_img_in_tiles	Indicates that the input image should be divided into tiles	False	N/A
–tile_xsize	Size of each tile (width)	512	Integers
–tile_ysize	Size of each tile (height)	512	Integers
–tile_xstep	How many 'tile_xstep' away the next tile/-cutout should be: 1 indicates no overlap, i.e. the next tile is adjacent to the current	1	Numbers
–tile_ystep	How many 'tile_ystep' away the next tile/-cutout should be	1	Numbers

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. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Abbott, B. P. et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116:061102, Feb 2016. doi: 10.1103/PhysRevLett.116.061102.

Abbott, B. P. et al. Gw170817: Observation of gravitational waves from a binary neutron star inspiral. *Phys. Rev. Lett.*, 119:161101, Oct 2017. doi: 10.1103/PhysRevLett.119.161101.

Airy, G. B. On the Diffraction of an Object-glass with Circular Aperture. *Transactions of the Cambridge Philosophical Society*, 5:283, January 1835.

Alard, C. Automated detection of gravitational arcs. *arXiv preprint astro-ph/0606757*, 2006.

Avestruz, C., Li, N., Zhu, H., Lightman, M., Collett, T. E., and Luo, W. Automated lensing learner: Automated strong lensing identification with a computer vision technique. *The Astrophysical Journal*, 877(1):58, May 2019. ISSN 1538-4357. doi: 10.3847/1538-4357/ab16d9.

Banfield, J. K., Wong, O. I., Willett, K. W., Norris, R. P., Rudnick, L., Shabala, S. S., Simmons, B. D., Snyder, C., Garon, A., Seymour, N., Middelberg, E., Andernach, H., Lintott, C. J., Jacob, K., Kapińska, A. D., Mao, M. Y., Masters, K. L., Jarvis, M. J., Schawinski, K., Paget, E., Simpson, R., Klöckner, H.-R., Bamford, S., Burchell, T., Chow, K. E., Cotter, G., Fortson, L., Heywood, I., Jones, T. W., Kaviraj, S., López-Sánchez, R., Maksym, W. P., Polsterer, K., Borden, K., Hollow, R. P., and Whyte, L. Radio Galaxy Zoo: host galaxies and radio morphologies derived from visual inspection. *Monthly Notices of the Royal Astronomical Society*, 453(3):2326–2340, 09 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv1688.

Becker, R. H., White, R. L., and Helfand, D. J. The FIRST Survey: Faint Images of the Radio Sky at Twenty Centimeters. , 450:559, September 1995. doi: 10.1086/176166.

Blake, C., Abdalla, F., Bridle, S., and Rawlings, S. Cosmology with the ska. *New Astronomy Reviews*, 48(11):1063–1077, 2004. ISSN 1387-6473. doi: https://doi.org/10.1016/j.newar.2004.09.045. Science with the Square Kilometre Array.

Blausen.com staff. Medical gallery of blausen medical 2014. *WikiJournal of Medicine*, 1(2), 2014. ISSN 2002-4436. doi: 10.15347/wjm/2014.010.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection, 2020. URL `https://arxiv.org/abs/2004.10934`.

Bodla, N., Singh, B., Chellappa, R., and Davis, L. S. Soft-nms – improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Bolton, A. S., Burles, S., Koopmans, L. V. E., Treu, T., and Moustakas, L. A. The sloan lens ACS survey. i. a large spectroscopically selected sample of massive early-type lens galaxies. *The Astrophysical Journal*, 638(2):703–724, feb 2006. doi: 10.1086/498884.

Bonaldi, A., An, T., Brüggen, M., Burkutean, S., Coelho, B., Goodarzi, H., Hartley, P., Sandhu, P. K., Wu, C., Yu, L., Zhoolideh Haghighi, M. H., Antón, S., Bagheri, Z., Barbosa, D., Barraca, J. P., Bartashevich, D., Bergano, M., Bonato, M., Brand, J., de Gasperin, F., Giannetti, A., Dodson, R., Jain, P., Jaiswal, S., Lao, B., Liu, B., Liuzzo, E., Lu, Y., Lukic, V., Maia, D., Marchili, N., Massardi, M., Mohan, P., Morgado, J. B., Panwar, M., Prabhakar, P., Ribeiro, V. A. R. M., Rygl, K. L. J., Sabz Ali, V., Saremi, E., Schisano, E., Sheikhnezami, S., Vafaei Sadr, A., Wong, A., and Wong, O. I. Square Kilometre Array Science Data Challenge 1: analysis and results. *Monthly Notices of the Royal Astronomical Society*, 500 (3):3821–3837, 10 2020. ISSN 0035-8711. doi: 10.1093/mnras/staa3023.

Born, M. and Wolf, E. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier, 2013.

Briggs, D. S. High fidelity deconvolution of moderately resolved sources. *Ph.D. Thesis*, 1995.

Brownlee, J. How to visualize filters and feature maps in convolutional neural networks, May 2019. URL `https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/`.

Bryant, J. The first century of microwaves-1886 to 1986. *IEEE Transactions on Microwave Theory and Techniques*, 36(5): 830–858, 1988. doi: 10.1109/22.3602.

Burke, C. J., Aleo, P. D., Chen, Y.-C., Liu, X., Peterson, J. R., Sembroski, G. H., and Lin, J. Y.-Y. Deblending and classifying astronomical sources with Mask R-CNN deep learning. *Monthly Notices of the Royal Astronomical Society*, 490(3):3952–3965, 10 2019. ISSN 0035-8711. doi: 10.1093/mnras/stz2845.

Busch, U. Claims of priority – the scientific path to the discovery of x-rays. *Zeitschrift für Medizinische Physik*, 33(2): 230–242, 2023. ISSN 0939-3889. doi: https://doi.org/10.1016/j.zemedi.2022.12.002.

Cabanac, R. A., Alard, C., Dantel-Fort, M., Fort, B., Gavazzi, R., Gomez, P., Kneib, J. P., Le Fèvre, O., Mellier, Y., Pello, R., Soucail, G., Sygnet, J. F., and Valls-Gabaud, D. The cfhtls strong lensing legacy survey - i. survey overview and t0002 release sample. *A&A*, 461(3):813–821, 2007. doi: 10.1051/0004-6361:20065810.

Cao, G., Song, W., and Zhao, Z. Gastric cancer diagnosis with mask r-cnn. In *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 60–63, 2019. doi: 10.1109/IHMSC.2019.00022.

Carbone, D., Garsden, H., Spreeuw, H., Swinbank, J., van der Horst, A., Rowlinson, A., Broderick, J., Rol, E., Law, C., Molenaar, G., and Wijers, R. Pyse: Software for extracting sources from radio images. *Astronomy and Computing*, 23: 92–102, 2018. ISSN 2213-1337. doi: https://doi.org/10.1016/j.ascom.2018.02.003.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8.

Carroll, B. W. and Ostlie, D. A. *An introduction to modern astrophysics*. Cambridge University Press, 2017.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

Chollet, F. et al. Keras. `https://keras.io`, 2015.

Chwolson, O. Über eine mögliche form fiktiver doppelsterne. *Astronomische Nachrichten*, 221(20):329–330, 1924. doi: https://doi.org/10.1002/asna.19242212003.

Collett, T. E. The population of galaxy–galaxy strong lenses in forthcoming optical imaging surveys. *The Astrophysical Journal*, 811(1):20, sep 2015. doi: 10.1088/0004-637X/811/1/20.

Condon, J. J. and Ransom, S. M. *Essential Radio Astronomy*. Princeton University Press, 2016.

CS231N Staff. Cs231n convolutional neural networks for visual recognition. URL `https://cs231n.github.io/neural-networks-1/`. Accessed on 11 January 2024.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

Dalcin, L. and Fang, Y.-L. L. mpi4py: Status update after 12 years of development. *Computing in Science Engineering*, 23 (4):47–54, 2021. doi: 10.1109/MCSE.2021.3083216.

Davidson, M. W. Johann Wilhelm Ritter, 2015. URL `https://micro.magnet.fsu.edu/optics/timeline/people/ritter.html`. Last Accessed on 14 February 2024.

Davis, J. and Goadrich, M. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 233–240, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143874. URL `https://doi.org/10.1145/1143844.1143874`.

de Jong, J. T. A., Verdoes Kleijn, G. A., Kuijken, K. H., and Valentijn, E. A. The Kilo-Degree Survey. *Experimental Astronomy*, 35(1-2):25–44, January 2013. doi: 10.1007/s10686-012-9306-1.

Dewdney, P., Turner, W., Millenaar, R., McCool, R., Lazio, J., and Cornwell, T. Ska1 system baseline design. *Document number SKA-TEL-SKO-DD-001 Revision*, 1(1), 2013.

Dewdney, P. et al. SKA1 design baseline description. Technical Report SKA-TEL-SKO-0001075, Revision 02, SKAO, January 2022. URL `https://www.dropbox.com/scl/fi/z75259ah3hlyrcnrdocyo/SKA-TEL-SKO-0001075-02_DesignBaselineDescription.pdf?rlkey=rku3q8w04zim47o1ywj4qccv6&dl=0`.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Dressler, A., Spergel, D., Mountain, M., Postman, M., Elliott, E., Bendek, E., Bennett, D., Dalcanton, J., Gaudi, S., Gehrels, N., Guyon, O., Hirata, C., Kalirai, J., Kasdin, N. J., Kruk, J., Macintosh, B., Malhotra, S., Penny, M., Perlmutter, S., Rieke, G., Riess, A., Rhoads, J., Shaklan, S., Somerville, R., Stern, D., Thompson, R., and Weinberg, D. Exploring the nro opportunity for a hubble-sized wide-field near-ir space telescope – new wfirst, 2012.

Einstein, A. Die feldgleichungen der gravitation. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften*, pages 844–847, 1915.

Einstein, A. Näherungsweise Integration der Feldgleichungen der Gravitation. *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften*, pages 688–696, January 1916.

Einstein, A. Lens-like action of a star by the deviation of light in the gravitational field. *Science*, 84(2188):506–507, 1936. doi: 10.1126/science.84.2188.506.

European Commission. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, 2021. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206`.

European Parliament and Council of the European Union. Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives

2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act)text with eea relevance., 2024. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689.

Eusebi, P. Diagnostic Accuracy Measures. *Cerebrovascular Diseases*, 36(4):267–272, 10 2013. ISSN 1015-9770. doi: 10.1159/000353863.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

Fang, Y., Wang, W., Xie, B., Sun, Q., Wu, L., Wang, X., Huang, T., Wang, X., and Cao, Y. EVA: Exploring the limits of masked visual representation learning at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19358–19369, June 2023.

Fawcett, T. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2005.10.010. ROC Analysis in Pattern Recognition.

Frei, M. and Kruis, F. Fiber-cnn: Expanding mask r-cnn to improve image-based fiber analysis. *Powder Technology*, 377:974–991, 2021. ISSN 0032-5910. doi: https://doi.org/10.1016/j.powtec.2020.08.034.

Fukugita, M., Ichikawa, T., Gunn, J. E., Doi, M., Shimasaku, K., and Schneider, D. P. The Sloan Digital Sky Survey Photometric System. , 111:1748, April 1996. doi: 10.1086/117915.

Fukushima, K. and Miyake, S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469, 1982. ISSN 0031-3203. doi: https://doi.org/10.1016/0031-3203(82)90024-3.

Future of Life Institute. Timeline of developments, 2024. URL https://artificialintelligenceact.eu/developments/. Last Accessed on 22 July 2024.

Gerward, L. Paul villard and his discovery of gamma rays. *Physics in Perspective*, 1:367–383, 1999. doi: 10.1007/s000160050028.

Girshick, R. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi: 10.1109/ICCV.2015.169.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/glorot11a.html.

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

Gray, P. C., Bierlich, K. C., Mantell, S. A., Friedlaender, A. S., Goldbogen, J. A., and Johnston, D. W. Drones and convolutional neural networks facilitate automated and accurate cetacean species identification and photogrammetry. *Methods in Ecology and Evolution*, 10(9):1490–1500, 2019. doi: https://doi.org/10.1111/2041-210X.13246.

Hainaut, O. Signal, noise and detection, Jun 2015. URL https://www.eso.org/~ohainaut/ccd/sn.html. Last Accessed in November 2023. Archived copy accessible from: https://web.archive.org/web/20231116030453/https://www.eso.org/~ohainaut/ccd/sn.html.

Hale, C. L., Robotham, A. S. G., Davies, L. J. M., Jarvis, M. J., Driver, S., and Heywood, I. Radio source extraction with ProFound. *Monthly Notices of the Royal Astronomical Society*, 487(3):3971–3989, 06 2019. ISSN 0035-8711. doi: 10.1093/mnras/stz1462.

Hancock, P. J., Trott, C. M., and Hurley-Walker, N. Source finding in the era of the ska (precursors): Aegean 2.0. *Publications of the Astronomical Society of Australia*, 35:e011, 2018. doi: 10.1017/pasa.2018.3.

Hartley, P., Flamary, R., Jackson, N., Tagore, A. S., and Metcalf, R. B. Support vector machine classification of strong gravitational lenses. , 471(3):3378–3397, November 2017. doi: 10.1093/mnras/stx1733.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322.

Heaviside, O. A gravitational and electromagnetic analogy. *The Electrician*, 31(Part I):281–282, 1893.

Herschel, W. Xiv. experiments on the refrangibility of the invisible rays of the sun. *Philosophical Transactions of the Royal Society of London*, (90):284–292, 1800. doi: 10.1098/rstl.1800.0015.

Hertz, H. *Electric waves: being researches on the propagation of electric action with finite velocity through space*. Dover Publications, 1893.

Heywood, I., Bannister, K. W., Marvil, J., Allison, J. R., Ball, L., Bell, M. E., Bock, D. C.-J., Brothers, M., Bunton, J. D., Chippendale, A. P., Cooray, F., Cornwell, T. J., De Boer, D., Edwards, P., Gough, R., Gupta, N., Harvey-Smith, L., Hay, S., Hotan, A. W., Indermuehle, B., Jacka, C., Jackson, C. A., Johnston, S., Kimball, A. E., Koribalski, B. S., Lenc, E., Macleod, A., McClure-Griffiths, N., McConnell, D., Mirtschin, P., Murphy, T., Neuhold, S., Norris, R. P., Pearce, S., Popping, A., Qiao, R. Y., Reynolds, J. E., Sadler, E. M., Sault, R. J., Schinckel, A. E. T., Serra, P., Shimwell, T. W., Stevens, J., Tuthill, J., Tzioumis, A., Voronkov, M. A., Westmeier, T., and Whiting, M. T. Wide-field broad-band radio imaging with phased array feeds: a pilot multi-epoch continuum survey with ASKAP-BETA. *Monthly Notices of the Royal Astronomical Society*, 457(4):4160–4178, 02 2016. ISSN 0035-8711. doi: 10.1093/mnras/stw186.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

Högbom, J. A. Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines. , 15:417, June 1974.

Hotan, A. W., Bunton, J. D., Chippendale, A. P., Whiting, M., Tuthill, J., Moss, V. A., McConnell, D., Amy, S. W., Huynh, M. T., Allison, J. R., and et al. Australian square kilometre array pathfinder: I. system description. *Publications of the Astronomical Society of Australia*, 38:e009, 2021. doi: 10.1017/pasa.2021.1.

Hubel, D. H. and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

Ingallinera, A., Cavallaro, F., Loru, S., Marvil, J., Umana, G., Trigilio, C., Breen, S., Bordiu, C., Buemi, C. S., Bufano, F., Collier, J., Etoka, S., Filipović, M. D., Goldman, S. R., Hopkins, A. M., Koribalski, B. S., Leto, P., Norris, R. P., Riggi, S., Schillirò, F., Tremblay, C., and van Loon, J. T. Evolutionary map of the Universe (EMU): 18-cm OH-maser discovery in ASKAP continuum images of the SCORPIO field. *Monthly Notices of the Royal Astronomical Society: Letters*, 512(1): L21–L26, 02 2022. ISSN 1745-3925. doi: 10.1093/mnrasl/slac017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/ioffe15.html`.

Isella, A. Interferometry Basics. `https://science.nrao.edu/opportunities/courses/casa-caltech-winter2012/Isella_Radio_Interferometry_Basics_Caltech2012.pdf`, 2011. Caltech CASA Radio Analysis Workshop, Pasadena, January 19, 2011.

Jacobs, C., Glazebrook, K., Collett, T., More, A., and McCarthy, C. Finding strong lenses in CFHTLS using convolutional neural networks. , 471(1):167–181, October 2017. doi: 10.1093/mnras/stx1492.

Jansen van Rensburg, J.-P. The design of a two-element correlation interferometer operating at l-band. Master's thesis, Stellenbosch: Stellenbosch University, 2012.

Jansky, K. Electrical disturbances apparently of extraterrestrial origin. *Proceedings of the Institute of Radio Engineers*, 21 (10):1387–1398, 1933. doi: 10.1109/JRPROC.1933.227458.

Jocher, G., Chaurasia, A., and Qiu, J. Ultralytics YOLO, January 2023. URL `https://github.com/ultralytics/ultralytics`.

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. Science with ASKAP. The Australian square-kilometre-array pathfinder. *Experimental Astronomy*, 22(3):151–273, December 2008. doi: 10.1007/s10686-008-9124-7.

Kirillov, A., Girshick, R., He, K., and Dollar, P. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Kochanek, C., Falco, E., Impey, C., Lehar, J., McLeod, B., and Rix, H.-W. Castles survey. URL `https://lweb.cfa.harvard.edu/castles/`. Accessed on 26 September 2023.

Kochanek, C., Falco, E., Impey, C., Lehár, J., McLeod, B., and Rix, H.-W. Results from the castles survey of gravitational lenses. In *AIP Conference Proceedings*, volume 470, pages 163–175. American Institute of Physics, 1999.

Kraus, J. Heinrich hertz-theorist and experimenter. *IEEE Transactions on Microwave Theory and Techniques*, 36(5):824–829, 1988. doi: 10.1109/22.3601.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

La Rana, A. *Gravitational Waves: An Historical Perspective*, pages 107–122. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-37387-9. doi: 10.1007/978-3-031-37387-9_8. URL `https://doi.org/10.1007/978-3-031-37387-9_8`.

Lane, M. and Williams, M. Defining and classifying ai in the workplace. (290), 2023. doi: https://doi.org/10.1787/59e89d7f-en.

Lanusse, F., Ma, Q., Li, N., Collett, T. E., Li, C.-L., Ravanbakhsh, S., Mandelbaum, R., and Póczos, B. CMU DeepLens: deep learning for automatic image-based galaxy-galaxy strong lens finding. , 473(3):3895–3906, January 2018. doi: 10.1093/mnras/stx1665.

Laureijs, R., Amiaux, J., Arduini, S., Augueres, J. L., Brinchmann, J., Cole, R., Cropper, M., Dabin, C., Duvet, L., Ealet, A., Garilli, B., Gondoin, P., Guzzo, L., Hoar, J., Hoekstra, H., Holmes, R., Kitching, T., Maciaszek, T., Mellier, Y., Pasian, F., Percival, W., Rhodes, J., Criado, G. S., Sauvage, M., Scaramella, R., Valenziano, L., Warren, S., Bender, R., Castander, F., Cimatti, A., Fèvre, O. L., Kurki-Suonio, H., Levi, M., Lilje, P., Meylan, G., Nichol, R., Pedersen, K., Popa, V., Lopez, R. R., Rix, H. W., Rottgering, H., Zeilinger, W., Grupp, F., Hudelot, P., Massey, R., Meneghetti, M., Miller, L., Paltani, S., Paulin-Henriksson, S., Pires, S., Saxton, C., Schrabback, T., Seidel, G., Walsh, J., Aghanim, N., Amendola, L., Bartlett, J., Baccigalupi, C., Beaulieu, J. P., Benabed, K., Cuby, J. G., Elbaz, D., Fosalba, P., Gavazzi, G., Helmi, A., Hook, I., Irwin, M., Kneib, J. P., Kunz, M., Mannucci, F., Moscardini, L., Tao, C., Teyssier, R., Weller, J., Zamorani, G., Osorio, M. R. Z., Boulade, O., Foumond, J. J., Giorgio, A. D., Guttridge, P., James, A., Kemp, M., Martignac, J., Spencer, A., Walton, D., Blümchen, T., Bonoli, C., Bortoletto, F., Cerna, C., Corcione, L., Fabron, C., Jahnke, K., Ligori, S., Madrid, F., Martin, L., Morgante, G., Pamplona, T., Prieto, E., Riva, M., Toledo, R., Trifoglio, M., Zerbi, F., Abdalla, F., Douspis, M., Grenet, C., Borgani, S., Bouwens, R., Courbin, F., Delouis, J. M., Dubath, P., Fontana, A., Frailis, M., Grazian, A., Koppenhöfer, J., Mansutti, O., Melchior, M., Mignoli, M., Mohr, J., Neissner, C.,

Noddle, K., Poncet, M., Scodeggio, M., Serrano, S., Shane, N., Starck, J. L., Surace, C., Taylor, A., Verdoes-Kleijn, G., Vuerli, C., Williams, O. R., Zacchei, A., Altieri, B., Sanz, I. E., Kohley, R., Oosterbroek, T., Astier, P., Bacon, D., Bardelli, S., Baugh, C., Bellagamba, F., Benoist, C., Bianchi, D., Biviano, A., Branchini, E., Carbone, C., Cardone, V., Clements, D., Colombi, S., Conselice, C., Cresci, G., Deacon, N., Dunlop, J., Fedeli, C., Fontanot, F., Franzetti, P., Giocoli, C., Garcia-Bellido, J., Gow, J., Heavens, A., Hewett, P., Heymans, C., Holland, A., Huang, Z., Ilbert, O., Joachimi, B., Jennins, E., Kerins, E., Kiessling, A., Kirk, D., Kotak, R., Krause, O., Lahav, O., van Leeuwen, F., Lesgourgues, J., Lombardi, M., Magliocchetti, M., Maguire, K., Majerotto, E., Maoli, R., Marulli, F., Maurogordato, S., McCracken, H., McLure, R., Melchiorri, A., Merson, A., Moresco, M., Nonino, M., Norberg, P., Peacock, J., Pello, R., Penny, M., Pettorino, V., Porto, C. D., Pozzetti, L., Quercellini, C., Radovich, M., Rassat, A., Roche, N., Ronayette, S., Rossetti, E., Sartoris, B., Schneider, P., Semboloni, E., Serjeant, S., Simpson, F., Skordis, C., Smadja, G., Smartt, S., Spano, P., Spiro, S., Sullivan, M., Tilquin, A., Trotta, R., Verde, L., Wang, Y., Williger, G., Zhao, G., Zoubian, J., and Zucca, E. Euclid definition study report, 2011.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2:396–404, 1989.

Lecun, Y., Jackel, L., Bottou, L., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., Simard, P., and Vapnik, V. *Learning algorithms for classification: A comparison on handwritten digit recognition*, pages 261–276. World Scientific, 1995.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. Deeply-Supervised Nets. In Lebanon, G. and Vishwanathan, S. V. N., editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 562–570, San Diego, California, USA, 09–12 May 2015. PMLR. URL `https://proceedings.mlr.press/v38/lee15a.html`.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. doi: 10.1109/CVPR.2017.106.

Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., and Yosinski, J. An intriguing failing of convolutional neural networks and the coordconv solution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018a. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/60106888f8977b71e1f15db7bc9a88d1-Paper.pdf`.

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018b.

Liu, Y. H. Feature extraction and image recognition with convolutional neural networks. In *Journal of Physics: Conference Series*, volume 1087, page 062032. IOP Publishing, 2018.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. doi: 10.1109/CVPR.2015.7298965.

Looi, S. Rotated mask r-cnn: From bounding boxes to rotated bounding boxes. `https://github.com/mrlooi/rotated_maskrcnn`, 2019.

LSST Science Collaboration, Abell, P. A., Allison, J., Anderson, S. F., Andrew, J. R., Angel, J. R. P., Armus, L., Arnett, D., Asztalos, S. J., Axelrod, T. S., Bailey, S., Ballantyne, D. R., Bankert, J. R., Barkhouse, W. A., Barr, J. D., Barrientos, L. F., Barth, A. J., Bartlett, J. G., Becker, A. C., Becla, J., Beers, T. C., Bernstein, J. P., Biswas, R., Blanton, M. R., Bloom,

J. S., Bochanski, J. J., Boeshaar, P., Borne, K. D., Bradac, M., Brandt, W. N., Bridge, C. R., Brown, M. E., Brunner, R. J., Bullock, J. S., Burgasser, A. J., Burge, J. H., Burke, D. L., Cargile, P. A., Chandrasekharan, S., Chartas, G., Chesley, S. R., Chu, Y.-H., Cinabro, D., Claire, M. W., Claver, C. F., Clowe, D., Connolly, A. J., Cook, K. H., Cooke, J., Cooray, A., Covey, K. R., Culliton, C. S., de Jong, R., de Vries, W. H., Debattista, V. P., Delgado, F., Dell'Antonio, I. P., Dhital, S., Stefano, R. D., Dickinson, M., Dilday, B., Djorgovski, S. G., Dobler, G., Donalek, C., Dubois-Felsmann, G., Durech, J., Eliasdottir, A., Eracleous, M., Eyer, L., Falco, E. E., Fan, X., Fassnacht, C. D., Ferguson, H. C., Fernandez, Y. R., Fields, B. D., Finkbeiner, D., Figueroa, E. E., Fox, D. B., Francke, H., Frank, J. S., Frieman, J., Fromenteau, S., Furqan, M., Galaz, G., Gal-Yam, A., Garnavich, P., Gawiser, E., Geary, J., Gee, P., Gibson, R. R., Gilmore, K., Grace, E. A., Green, R. F., Gressler, W. J., Grillmair, C. J., Habib, S., Haggerty, J. S., Hamuy, M., Harris, A. W., Hawley, S. L., Heavens, A. F., Hebb, L., Henry, T. J., Hileman, E., Hilton, E. J., Hoadley, K., Holberg, J. B., Holman, M. J., Howell, S. B., Infante, L., Ivezic, Z., Jacoby, S. H., Jain, B., R, Jedicke, Jee, M. J., Jernigan, J. G., Jha, S. W., Johnston, K. V., Jones, R. L., Juric, M., Kaasalainen, M., Styliani, Kafka, Kahn, S. M., Kaib, N. A., Kalirai, J., Kantor, J., Kasliwal, M. M., Keeton, C. R., Kessler, R., Knezevic, Z., Kowalski, A., Krabbendam, V. L., Krughoff, K. S., Kulkarni, S., Kuhlman, S., Lacy, M., Lepine, S., Liang, M., Lien, A., Lira, P., Long, K. S., Lorenz, S., Lotz, J. M., Lupton, R. H., Lutz, J., Macri, L. M., Mahabal, A. A., Mandelbaum, R., Marshall, P., May, M., McGehee, P. M., Meadows, B. T., Meert, A., Milani, A., Miller, C. J., Miller, M., Mills, D., Minniti, D., Monet, D., Mukadam, A. S., Nakar, E., Neill, D. R., Newman, J. A., Nikolaev, S., Nordby, M., O'Connor, P., Oguri, M., Oliver, J., Olivier, S. S., Olsen, J. K., Olsen, K., Olszewski, E. W., Oluseyi, H., Padilla, N. D., Parker, A., Pepper, J., Peterson, J. R., Petry, C., Pinto, P. A., Pizagno, J. L., Popescu, B., Prsa, A., Radcka, V., Raddick, M. J., Rasmussen, A., Rau, A., Rho, J., Rhoads, J. E., Richards, G. T., Ridgway, S. T., Robertson, B. E., Roskar, R., Saha, A., Sarajedini, A., Scannapieco, E., Schalk, T., Schindler, R., Schmidt, S., Schmidt, S., Schneider, D. P., Schumacher, G., Scranton, R., Sebag, J., Seppala, L. G., Shemmer, O., Simon, J. D., Sivertz, M., Smith, H. A., Smith, J. A., Smith, N., Spitz, A. H., Stanford, A., Stassun, K. G., Strader, J., Strauss, M. A., Stubbs, C. W., Sweeney, D. W., Szalay, A., Szkody, P., Takada, M., Thorman, P., Trilling, D. E., Trimble, V., Tyson, A., Berg, R. V., Berk, D. V., VanderPlas, J., Verde, L., Vrsnak, B., Walkowicz, L. M., Wandelt, B. D., Wang, S., Wang, Y., Warner, M., Wechsler, R. H., West, A. A., Wiecha, O., Williams, B. F., Willman, B., Wittman, D., Wolff, S. C., Wood-Vasey, W. M., Wozniak, P., Young, P., Zentner, A., and Zhan, H. Lsst science book, version 2.0, 2009.

Lucas, L., Staley, T., and Scaife, A. Efficient source finding for radio interferometric images. *Astronomy and Computing*, 27:96–110, 2019. ISSN 2213-1337. doi: https://doi.org/10.1016/j.ascom.2019.02.002.

Lukic, V., Brüggen, M., Banfield, J. K., Wong, O. I., Rudnick, L., Norris, R. P., and Simmons, B. Radio Galaxy Zoo: compact and extended radio source classification with deep learning. *Monthly Notices of the Royal Astronomical Society*, 476(1):246–260, 01 2018. ISSN 0035-8711. doi: 10.1093/mnras/sty163.

Lukic, V., de Gasperin, F., and Brüggen, M. Convosource: Radio-astronomical source-finding with convolutional neural networks. *Galaxies*, 8(1):3, 2020. ISSN 2075-4434. doi: 10.3390/galaxies8010003.

Magro, A. . A real-time, gpu-based, non-imaging back-end for radio telescopes, 2013.

Magro, D., Zarb Adami, K., DeMarco, A., Riggi, S., and Sciacca, E. Lexactum trained model weights, December 2020. URL `https://doi.org/10.5281/zenodo.4299924`. D. Magro benefited from a grant: Osservatorio Astrofisico di Catania - Istituto Nazionale di Astrofisica, MoU: "High performance computing in astronomy, astrophysics and particle physics".

Magro, D., Zarb Adami, K., DeMarco, A., Riggi, S., and Sciacca, E. A comparative study of convolutional neural networks for the detection of strong gravitational lensing. *Monthly Notices of the Royal Astronomical Society*, 505(4):6155–6165, 06 2021. ISSN 0035-8711. doi: 10.1093/mnras/stab1635.

Magro, D., Adami, K. Z., Marco, A. D., Riggi, S., and Sciacca, E. LEXACTUM trained model weights. 7 2024. doi: 10.60809/drum.26236664.v1.

Maxwell, J. C. A dynamical theory of the electromagnetic field (1865). *The Scientific Papers of James Clerk Maxwell*, 2, 1890.

Mazumder, A., Datta, A., Chakraborty, A., and Majumdar, S. Observing the reionization: effect of calibration and position errors on realistic observation conditions. *Monthly Notices of the Royal Astronomical Society*, 515(3):4020–4037, 07 2022. ISSN 0035-8711. doi: 10.1093/mnras/stac1994.

McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943. doi: 10.1007/BF02478259.

Metcalf, R. B., Meneghetti, M., Avestruz, C., Bellagamba, F., Bom, C. R., Bertin, E., Cabanac, R., Courbin, F., Davies, A., Decencière, E., Flamary, R., Gavazzi, R., Geiger, M., Hartley, P., Huertas-Company, M., Jackson, N., Jacobs, C., Jullo, E., Kneib, J.-P., Koopmans, L. V. E., Lanusse, F., Li, C.-L., Ma, Q., Makler, M., Li, N., Lightman, M., Petrillo, C. E., Serjeant, S., Schäfer, C., Sonnenfeld, A., Tagore, A., Tortora, C., Tuccillo, D., Valentín, M. B., Velasco-Forero, S., Verdoes Kleijn, G. A., and Vernardos, G. The strong gravitational lens finding challenge. *A&A*, 625:A119, 2019. doi: 10.1051/0004-6361/201832797.

Milletari, F., Navab, N., and Ahmadi, S.-A. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016. doi: 10.1109/3DV.2016.79.

Montagnier, P. and Ek, I. Ai measurement in ict usage surveys. (308), 2021. doi: https://doi.org/10.1787/72cce754-en.

Moore, P. *Eyes on the universe: the story of the telescope*. Springer Science & Business Media, 2012.

Myers, S. T., Jackson, N. J., Browne, I. W. A., de Bruyn, A. G., Pearson, T. J., Readhead, A. C. S., Wilkinson, P. N., Biggs, A. D., Blandford, R. D., Fassnacht, C. D., Koopmans, L. V. E., Marlow, D. R., McKean, J. P., Norbury, M. A., Phillips, P. M., Rusin, D., Shepherd, M. C., and Sykes, C. M. The Cosmic Lens All-Sky Survey - I. Source selection and observations. *Monthly Notices of the Royal Astronomical Society*, 341(1):1–12, 05 2003. ISSN 0035-8711. doi: 10.1046/j.1365-8711.2003.06256.x.

NASA. Science themes - james webb space telescope. URL `https://www.jwst.nasa.gov/content/science/index.html`. Accessed on 26 March 2022.

National Optical Astronomy Observatory. Iraf (image reduction and analysis facility) display help page, Mar 1997. URL `https://iraf.net/irafhelp.php?val=display`. Last Accessed in November 2023. Original link no longer works, archived copy accessible from: `https://web.archive.org/web/20220425231700/https://iraf.net/irafhelp.php?val=display`.

Negrello, M., Amber, S., Amvrosiadis, A., Cai, Z.-Y., Lapi, A., Gonzalez-Nuevo, J., De Zotti, G., Furlanetto, C., Maddox, S. J., Allen, M., Bakx, T., Bussmann, R. S., Cooray, A., Covone, G., Danese, L., Dannerbauer, H., Fu, H., Greenslade, J., Gurwell, M., Hopwood, R., Koopmans, L. V. E., Napolitano, N., Nayyeri, H., Omont, A., Petrillo, C. E., Riechers, D. A., Serjeant, S., Tortora, C., Valiante, E., Verdoes Kleijn, G., Vernardos, G., Wardlow, J. L., Baes, M., Baker, A. J., Bourne, N., Clements, D., Crawford, S. M., Dye, S., Dunne, L., Eales, S., Ivison, R. J., Marchetti, L., Michałowski, M. J., Smith, M. W. L., Vaccari, M., and van der Werf, P. The Herschel-ATLAS: a sample of 500 m-selected lensed galaxies over 600 deg2. *Monthly Notices of the Royal Astronomical Society*, 465(3):3558–3580, 11 2016. ISSN 0035-8711. doi: 10.1093/mnras/stw2911.

Ni, H., Liu, H., Wang, K., Wang, X., Zhou, X., and Qian, Y. Wsi-net: Branch-based and hierarchy-aware network for segmentation and classification of breast histopathological whole-slide images. In Suk, H.-I., Liu, M., Yan, P., and Lian, C., editors, *Machine Learning in Medical Imaging*, pages 36–44, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32692-0.

Nielsen, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015.

Norris, R. P., Afonso, J., Appleton, P. N., Boyle, B. J., Ciliegi, P., Croom, S. M., Huynh, M. T., Jackson, C. A., Koekemoer, A. M., Lonsdale, C. J., Middelberg, E., Mobasher, B., Oliver, S. J., Polletta, M., Siana, B. D., Smail, I., and Voronkov, M. A. Deep ATLAS radio observations of the chandra deep field?south/ispitzer/iwide?area infrared extragalactic field. *The Astronomical Journal*, 132(6):2409–2423, jan 2006. doi: 10.1086/508275.

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. EMU: Evolutionary Map of the Universe. , 28(3):215–248, August 2011. doi: 10.1071/AS11021.

Norris, R. P., Marvil, J., Collier, J. D., Kapińska, A. D., O'Brien, A. N., Rudnick, L., Andernach, H., Asorey, J., Brown, M. J. I., Brüggen, M., and et al. The evolutionary map of the universe pilot survey. *Publications of the Astronomical Society of Australia*, 38:e046, 2021. doi: 10.1017/pasa.2021.42.

Nyquist, H. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928. doi: 10.1109/T-AIEE.1928.5055024.

OECD. Scoping the oecd ai principles. (291), 2019. doi: https://doi.org/10.1787/d62f618a-en.

OECD. Oecd framework for the classification of ai systems. (323), 2022. doi: https://doi.org/10.1787/cb6d9eca-en.

Papers With Code. Image classification. `https://paperswithcode.com/task/image-classification`, a. Last Accessed in October 2024.

Papers With Code. Instance segmentation. `https://paperswithcode.com/task/instance-segmentation`, b. Last Accessed in October 2024.

Pearson, J., Pennock, C., and Robinson, T. Auto-detection of strong gravitational lenses using convolutional neural networks. *Emergent Scientist*, 2:1, 01 2018. doi: 10.1051/emsci/2017010.

Pino, C. et al. Proceedings of the vii international workshop on artificial intelligence and pattern recognition (iwaipr 2021). La Habana, Cuba, 2021.

Poincaré, H. Sur la dynamique de l'électron. In *Compte Rendu de l'Académie des Sciences*, volume 140, pages 1504–1508, 1905.

Pourrahmani, M., Nayyeri, H., and Cooray, A. Lensflow: A convolutional neural network in search of strong gravitational lenses. *The Astrophysical Journal*, 856(1):68, Mar 2018. ISSN 1538-4357. doi: 10.3847/1538-4357/aaae6a.

Redmon, J. and Farhadi, A. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Redmon, J. and Farhadi, A. Yolov3: An incremental improvement, 2018. URL `https://arxiv.org/abs/1804.02767`.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.

Riggi, S., Ingallinera, A., Leto, P., Cavallaro, F., Bufano, F., Schillirò, F., Trigilio, C., Umana, G., Buemi, C. S., and Norris, R. P. Automated detection of extended sources in radio maps: progress from the SCORPIO survey. *Monthly Notices of the Royal Astronomical Society*, 460(2):1486–1499, 04 2016. ISSN 0035-8711. doi: 10.1093/mnras/stw982.

Riggi, S., Vitello, F., Becciani, U., Buemi, C., Bufano, F., Calanducci, A., Cavallaro, F., Costa, A., Ingallinera, A., Leto, P., and et al. Caesar source finder: Recent developments and testing. *Publications of the Astronomical Society of Australia*, 36:e037, 2019. doi: 10.1017/pasa.2019.29.

Riggi, S., Bordiu, C., Vitello, F., Tudisco, G., Sciacca, E., Magro, D., Sortino, R., Pino, C., Molinaro, M., Benedettini, M., Leurini, S., Bufano, F., Raciti, M., and Becciani, U. Astronomical source finding services for the cirasa visual analytic platform. *Astronomy and Computing*, 37:100506, 2021a. ISSN 2213-1337. doi: https://doi.org/10.1016/j.ascom.2021.100506.

Riggi, S., Umana, G., Trigilio, C., Cavallaro, F., Ingallinera, A., Leto, P., Bufano, F., Norris, R. P., Hopkins, A. M., Filipović,

M. D., Andernach, H., van Loon, J. T., Michałowski, M. J., Bordiu, C., An, T., Buemi, C., Carretti, E., Collier, J. D., Joseph, T., Koribalski, B. S., Kothes, R., Loru, S., McConnell, D., Pommier, M., Sciacca, E., Schillirò, F., Vitello, F., Warhurst, K., and Whiting, M. Evolutionary map of the Universe (EMU): Compact radio sources in the scorpio field towards the galactic plane. *Monthly Notices of the Royal Astronomical Society*, 502(1):60–79, 01 2021b. ISSN 0035-8711. doi: 10.1093/mnras/stab028.

Riggi, S., Magro, D., Sortino, R., De Marco, A., Bordiu, C., Cecconello, T., Hopkins, A., Marvil, J., Umana, G., Sciacca, E., Vitello, F., Bufano, F., Ingallinera, A., Fiameni, G., Spampinato, C., and Zarb Adami, K. Astronomical source detection in radio continuum maps with deep neural networks. *Astronomy and Computing*, 42:100682, 2023. ISSN 2213-1337. doi: https://doi.org/10.1016/j.ascom.2022.100682.

Riggi, S. et al. Proceedings of the astronomical data analysis software and systems (adass) xxxi conference. Cape Town, South Africa, 2021c.

Robotham, A. S. G., Davies, L. J. M., Driver, S. P., Koushan, S., Taranu, D. S., Casura, S., and Liske, J. ProFound: Source Extraction and Application to Modern Survey Data. *Monthly Notices of the Royal Astronomical Society*, 476(3):3137–3159, 02 2018. ISSN 0035-8711. doi: 10.1093/mnras/sty440.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323 (6088):533–536, 1986. ISSN 1476-4687. doi: 10.1038/323533a0.

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Pearson, 2020.

Röntgen, W. C. On a new kind of rays. *Science*, 3(59):227–231, 1896. doi: 10.1126/science.3.59.227.

Schaefer, C., Geiger, M., Kuntzer, T., and Kneib, J. P. Deep convolutional neural networks as strong gravitational lens detectors. , 611:A2, March 2018. doi: 10.1051/0004-6361/201731201.

Schneider, D. P., Gunn, J. E., and Hoessel, J. G. CCD photometry of Abell clusters. I. Magnitudes and redshifts for 84 brightest cluster galaxies. , 264:337–355, January 1983. doi: 10.1086/160602.

Sciacca, E., Vitello, F., Becciani, U., Bordiu, C., Bufano, F., Calanducci, A., Costa, A., Raciti, M., and Riggi, S. Towards porting astrophysics visual analytics services in the european open science cloud. In Arai, K., Kapoor, S., and Bhatia, R., editors, *Intelligent Computing*, pages 598–606, Cham, 2020. Springer International Publishing. ISBN 978-3-030-52243-8.

Shannon, C. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949. doi: 10.1109/JRPROC.1949.232969.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition, 2015. URL `https://arxiv.org/abs/1409.1556`.

SKAO. What is radio astronomy?, a. URL `https://www.skao.int/en/resources/what-radio-astronomy`. Last Accessed in September 2024.

SKAO. Skao science goals, b. URL `https://www.skao.int/en/explore/science-goals`. Last Accessed in September 2024.

SKAO. Ska telescope specifications, c. URL `https://www.skao.int/en/science-users/118/ska-telescope-specifications`. Last Accessed in September 2024.

SKAO. SKAO Factsheets - Operational Model. `https://skao.canto.global/pdfviewer/viewer/viewer.html?v=MJ74FLU02G&portalType=v%2FMJ74FLU02G&column=document&id=2bsl6tbmrt2ntbta0v4ehjs21f&suffix=pdf&print=1`, March 2023.

Sonnenfeld, A., Chan, J. H. H., Shu, Y., More, A., Oguri, M., Suyu, S. H., Wong, K. C., Lee, C.-H., Coupon, J., Yonehara, A., Bolton, A. S., Jaelani, A. T., Tanaka, M., Miyazaki, S., and Komiyama, Y. Survey of Gravitationally-lensed Objects in HSC Imaging (SuGOHI). I. Automatic search for galaxy-scale strong lenses. , 70:S29, January 2018. doi: 10.1093/pasj/psx062.

Sortino, R., Magro, D., Fiameni, G., Sciacca, E., Riggi, S., DeMarco, A., Spampinato, C., Hopkins, A. M., Bufano, F., Schillirò, F., Bordiu, C., and Pino, C. Radio astronomical images object detection and segmentation: a benchmark on deep learning methods. *Experimental Astronomy*, May 2023a. ISSN 1572-9508. doi: 10.1007/s10686-023-09893-w.

Sortino, R., Magro, D., Sciacca, E., Riggi, S., and Fiameni, G. Deep neural networks for source detection in radio astronomical maps. In Bufano, F., Riggi, S., Sciacca, E., and Schilliro, F., editors, *Machine Learning for Astrophysics*, pages 135–139, Cham, 2023b. Springer International Publishing. ISBN 978-3-031-34167-0.

Sortino, R., Cecconello, T., DeMarco, A., Fiameni, G., Pilzer, A., Magro, D., Hopkins, A. M., Riggi, S., Sciacca, E., Ingallinera, A., Bordiu, C., Bufano, F., and Spampinato, C. Radiff: Controllable diffusion models for radio astronomical maps generation. *IEEE Transactions on Artificial Intelligence*, pages 1–12, 2024. doi: 10.1109/TAI.2024.3436538.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Srivastava, S. and Sharma, G. OmniVec: Learning robust representations with cross modal sharing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1236–1248, January 2024.

Storkey, A. J., Hambly, N. C., Williams, C. K. I., and Mann, R. G. Cleaning sky survey data bases using Hough transform and renewal string approaches. *Monthly Notices of the Royal Astronomical Society*, 347(1):36–51, 01 2004. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2004.07211.x.

Stoughton, C., Lupton, R. H., Bernardi, M., Blanton, M. R., Burles, S., Castander, F. J., Connolly, A. J., Eisenstein, D. J., Frieman, J. A., Hennessy, G. S., Hindsley, R. B., Željko Ivezić, Kent, S., Kunszt, P. Z., Lee, B. C., Meiksin, A., Munn, J. A., Newberg, H. J., Nichol, R. C., Nicinski, T., Pier, J. R., Richards, G. T., Richmond, M. W., Schlegel, D. J., Smith, J. A., Strauss, M. A., SubbaRao, M., Szalay, A. S., Thakar, A. R., Tucker, D. L., Berk, D. E. V., Yanny, B., Adelman, J. K., John E. Anderson, J., Anderson, S. F., Annis, J., Bahcall, N. A., Bakken, J. A., Bartelmann, M., Bastian, S., Bauer, A., Berman, E., Böhringer, H., Boroski, W. N., Bracker, S., Briegel, C., Briggs, J. W., Brinkmann, J., Brunner, R., Carey, L., Carr, M. A., Chen, B., Christian, D., Colestock, P. L., Crocker, J. H., Csabai, I., Czarapata, P. C., Dalcanton, J., Davidsen, A. F., Davis, J. E., Dehnen, W., Dodelson, S., Doi, M., Dombeck, T., Donahue, M., Ellman, N., Elms, B. R., Evans, M. L., Eyer, L., Fan, X., Federwitz, G. R., Friedman, S., Fukugita, M., Gal, R., Gillespie, B., Glazebrook, K., Gray, J., Grebel, E. K., Greenawalt, B., Greene, G., Gunn, J. E., de Haas, E., Haiman, Z., Haldeman, M., Hall, P. B., Hamabe, M., Hansen, B., Harris, F. H., Harris, H., Harvanek, M., Hawley, S. L., Hayes, J. J. E., Heckman, T. M., Helmi, A., Henden, A., Hogan, C. J., Hogg, D. W., Holmgren, D. J., Holtzman, J., Huang, C.-H., Hull, C., Ichikawa, S.-I., Ichikawa, T., Johnston, D. E., Kauffmann, G., Kim, R. S. J., Kimball, T., Kinney, E., Klaene, M., Kleinman, S. J., Klypin, A., Knapp, G. R., Korienek, J., Krolik, J., Kron, R. G., Krzesiński, J., Lamb, D. Q., Leger, R. F., Limmongkol, S., Lindenmeyer, C., Long, D. C., Loomis, C., Loveday, J., MacKinnon, B., Mannery, E. J., Mantsch, P. M., Margon, B., McGehee, P., McKay, T. A., McLean, B., Menou, K., Merelli, A., Mo, H. J., Monet, D. G., Nakamura, O., Narayanan, V. K., Nash, T., Eric H. Neilsen, J., Newman, P. R., Nitta, A., Odenkirchen, M., Okada, N., Okamura, S., Ostriker, J. P., Owen, R., Pauls, A. G., Peoples, J., Peterson, R. S., Petravick, D., Pope, A., Pordes, R., Postman, M., Prosapio, A., Quinn, T. R., Rechenmacher, R., Rivetta, C. H., Rix, H.-W., Rockosi, C. M., Rosner, R., Ruthmansdorfer, K., Sandford, D., Schneider, D. P., Scranton, R., Sekiguchi, M., Sergey, G., Sheth, R., Shimasaku, K., Smee, S., Snedden, S. A., Stebbins, A., Stubbs, C., Szapudi, I., Szkody, P., Szokoly, G. P., Tabachnik, S., Tsvetanov, Z., Uomoto, A., Vogeley, M. S., Voges, W., Waddell, P., Walterbos, R., i Wang, S., Watanabe, M., Weinberg, D. H., White, R. L., White, S. D. M., Wilhite, B., Wolfe, D., Yasuda, N., York, D. G., Zehavi, I., and Zheng, W. Sloan digital sky survey: Early data release. *The Astronomical Journal*, 123(1):485, jan 2002. doi: 10.1086/324741.

Swets, J. A. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988. doi: 10.1126/science.3287615.

Tan, H. H. and Lim, K. H. Vanishing gradient mitigation with deep learning neural network optimization. In *2019*

7th International Conference on Smart Computing Communications (ICSCC), pages 1–4, 2019. doi: 10.1109/ICSCC.2019.8843652.

The Dark Energy Survey Collaboration. The dark energy survey, 2005.

Thompson, A. R., Moran, J. M., and Swenson, G. W. *Interferometry and synthesis in radio astronomy*. Springer Nature, 2017.

Thuruthipilly, H., Zadrozny, A., Pollo, A., and Biesiada, M. Finding strong gravitational lenses through self-attention - study based on the bologna lens challenge. *AA*, 664:A4, 2022. doi: 10.1051/0004-6361/202142463.

Umana, G., Trigilio, C., Franzen, T. M. O., Norris, R. P., Leto, P., Ingallinera, A., Buemi, C. S., Agliozzo, C., Cavallaro, F., and Cerrigone, L. SCORPIO: a deep survey of radio emission from the stellar life-cycle. *Monthly Notices of the Royal Astronomical Society*, 454(1):902–912, 09 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv1976.

Umana, G., Trigilio, C., Ingallinera, A., Riggi, S., Cavallaro, F., Marvil, J., Norris, R. P., Hopkins, A. M., Buemi, C. S., Bufano, F., Leto, P., Loru, S., Bordiu, C., Bunton, J. D., Collier, J. D., Filipovic, M., Franzen, T. M. O., Thompson, M. A., Andernach, H., Carretti, E., Dai, S., Kapińska, A., Koribalski, B. S., Kothes, R., Leahy, D., Mcconnell, D., Tothill, N., and Michałowski, M. J. A first glimpse at the Galactic plane with the ASKAP: the SCORPIO field. *Monthly Notices of the Royal Astronomical Society*, 506(2):2232–2246, 05 2021. ISSN 0035-8711. doi: 10.1093/mnras/stab1279.

Unsöld, A. and Baschek, B. *The new cosmos: an introduction to astronomy and astrophysics*. Springer Science & Business Media, 2013.

Vafaei Sadr, A., Vos, E. E., Bassett, B. A., Hosenie, Z., Oozeer, N., and Lochner, M. DeepSource: point source detection using deep learning. *Monthly Notices of the Royal Astronomical Society*, 484(2):2793–2806, 02 2019. ISSN 0035-8711. doi: 10.1093/mnras/stz131.

Vapnik, V. Reconstruction of dependences from empirical data, 1979.

Vaporia. u-v plane. Vaporia: The Universe Astronomical Project. URL `http://astro.vaporia.com/start/uvplane.html`. Last Accessed in September 2024.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Walsh, D., Carswell, R. F., and Weymann, R. J. 0957 + 561 a, b: twin quasistellar objects or gravitational lens? *Nature*, 279(5712):381–384, 1979. doi: 10.1038/279381a0.

Wambsganss, J. Gravitational lensing in astronomy. *Living Reviews in Relativity*, 1:1–74, 1998.

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., and Ding, G. Yolov10: Real-time end-to-end object detection, 2024a. URL `https://arxiv.org/abs/2405.14458`.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022. URL `https://arxiv.org/abs/2207.02696`.

Wang, W., Dai, J., Chen, Z., Huang, Z., Li, Z., Zhu, X., Hu, X., Lu, T., Lu, L., Li, H., Wang, X., and Qiao, Y. InternImage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14408–14419, June 2023.

Wang, X., Kong, T., Shen, C., Jiang, Y., and Li, L. Solo: Segmenting objects by locations. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 649–665, Cham, 2020a. Springer International Publishing. ISBN 978-3-030-58523-5.

Wang, X., Zhang, R., Kong, T., Li, L., and Shen, C. Solov2: Dynamic and fast instance segmentation. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17721–17732. Curran Associates, Inc., 2020b. URL `https://proceedings.neurips.cc/paper_files/paper/`

2020/file/cd3afef9b8b89558cd56638c3631868a-Paper.pdf.

Wang, Z., Wang, P., Liu, K., Wang, P., Fu, Y., Lu, C.-T., Aggarwal, C. C., Pei, J., and Zhou, Y. A comprehensive survey on data augmentation, 2024b. URL https://arxiv.org/abs/2405.09591.

Wasserman, P. D. *Neural computing: theory and practice*. Van Nostrand Reinhold Co., 1989.

Weiner, C., Serjeant, S., and Sedgwick, C. Predictions for strong-lens detections with the nancy grace roman space telescope. *Research Notes of the AAS*, 4(10):190, oct 2020. doi: 10.3847/2515-5172/abc4ea.

Will, C. M. The confrontation between general relativity and experiment. *Living Reviews in Relativity*, 17:1–117, 2014.

Wong et al. Radio Galaxy Zoo, Data Release 1. in preparation.

Woodruff, A. What is a neuron? URL https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron. Accessed on 5 April 2022.

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. Radio Galaxy Zoo: Claran – a deep learning classifier for radio morphologies. *Monthly Notices of the Royal Astronomical Society*, 482(1):1211–1230, 10 2018. ISSN 0035-8711. doi: 10.1093/mnras/sty2646.

Yadugiri, V. T. Jagadish chandra bose. *Current Science*, 98(7):975, 2010.

Yang, Z., Dong, R., Xu, H., and Gu, J. Instance segmentation method based on improved mask r-cnn for the stacked electronic components. *Electronics*, 9(6), 2020. ISSN 2079-9292. doi: 10.3390/electronics9060886.

York, D. G., Adelman, J., John E. Anderson, J., Anderson, S. F., Annis, J., Bahcall, N. A., Bakken, J. A., Barkhouser, R., Bastian, S., Berman, E., Boroski, W. N., Bracker, S., Briegel, C., Briggs, J. W., Brinkmann, J., Brunner, R., Burles, S., Carey, L., Carr, M. A., Castander, F. J., Chen, B., Colestock, P. L., Connolly, A. J., Crocker, J. H., Csabai, I., Czarapata, P. C., Davis, J. E., Doi, M., Dombeck, T., Eisenstein, D., Ellman, N., Elms, B. R., Evans, M. L., Fan, X., Federwitz, G. R., Fiscelli, L., Friedman, S., Frieman, J. A., Fukugita, M., Gillespie, B., Gunn, J. E., Gurbani, V. K., de Haas, E., Haldeman, M., Harris, F. H., Hayes, J., Heckman, T. M., Hennessy, G. S., Hindsley, R. B., Holm, S., Holmgren, D. J., hao Huang, C., Hull, C., Husby, D., Ichikawa, S.-I., Ichikawa, T., Željko Ivezić, Kent, S., Kim, R. S. J., Kinney, E., Klaene, M., Kleinman, A. N., Kleinman, S., Knapp, G. R., Korienek, J., Kron, R. G., Kunszt, P. Z., Lamb, D. Q., Lee, B., Leger, R. F., Limmongkol, S., Lindenmeyer, C., Long, D. C., Loomis, C., Loveday, J., Lucinio, R., Lupton, R. H., MacKinnon, B., Mannery, E. J., Mantsch, P. M., Margon, B., McGehee, P., McKay, T. A., Meiksin, A., Merelli, A., Monet, D. G., Munn, J. A., Narayanan, V. K., Nash, T., Neilsen, E., Neswold, R., Newberg, H. J., Nichol, R. C., Nicinski, T., Nonino, M., Okada, N., Okamura, S., Ostriker, J. P., Owen, R., Pauls, A. G., Peoples, J., Peterson, R. L., Petravick, D., Pier, J. R., Pope, A., Pordes, R., Prosapio, A., Rechenmacher, R., Quinn, T. R., Richards, G. T., Richmond, M. W., Rivetta, C. H., Rockosi, C. M., Ruthmansdorfer, K., Sandford, D., Schlegel, D. J., Schneider, D. P., Sekiguchi, M., Sergey, G., Shimasaku, K., Siegmund, W. A., Smee, S., Smith, J. A., Snedden, S., Stone, R., Stoughton, C., Strauss, M. A., Stubbs, C., SubbaRao, M., Szalay, A. S., Szapudi, I., Szokoly, G. P., Thakar, A. R., Tremonti, C., Tucker, D. L., Uomoto, A., Berk, D. V., Vogeley, M. S., Waddell, P., i Wang, S., Watanabe, M., Weinberg, D. H., Yanny, B., and Yasuda, N. The sloan digital sky survey: Technical summary. *The Astronomical Journal*, 120(3):1579, sep 2000. doi: 10.1086/301513.

Young, T. I. the bakerian lecture. experiments and calculations relative to physical optics. *Philosophical transactions of the Royal Society of London*, (94):1–16, 1804. doi: 10.1098/rstl.1804.0001.

Zamir, S. W., Arora, A., Gupta, A., Khan, S., Sun, G., Khan, F. S., Zhu, F., Shao, L., Xia, G.-S., and Bai, X. iSAID: A large-scale dataset for instance segmentation in aerial images, 2019. URL https://arxiv.org/abs/1905.12886.

Zernike, F. The concept of degree of coherence and its application to optical problems. *Physica*, 5(8):785–795, 1938. ISSN 0031-8914. doi: https://doi.org/10.1016/S0031-8914(38)80203-2.

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., and Liang, J. Unet++: A nested u-net architecture for medical image segmentation. In Stoyanov, D., Taylor, Z., Carneiro, G., Syeda-Mahmood, T., Martel, A., Maier-Hein, L., Tavares, J. M. R., Bradley, A., Papa, J. P., Belagiannis, V., Nascimento, J. C., Lu, Z., Conjeti, S., Moradi, M., Greenspan, H., and

Madabhushi, A., editors, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00889-5.

Index

AI, Machine Learning, and Deep Learning, 12–31
  Convolutional Neural Networks, 24–28
  Neural Networks, 14–24
    ResNets, 22
    Batch Normalisation, 22
    Dataset Augmentation, 22–24
    Dropout, 21–22
    Pooling, 21
    Vanishing Gradient Problem, 20–21
  Quantitative Performance Evaluation of NNs, 28–31

Classification Algorithms, 33–39
  Conventional Methods, 33
    Arc-finders, 34
    Machine Learning (Pre-Selected Features), 34
    Visual Inspection, 33
  Machine Learning Classifiers, 34–39
    CAS Swinburne, 35
    CMU DeepLens, 36
    LASTRO EPFL, 35
    LensFinder, 39
    LensFlow, 38
    WSI-Net, 38

Gravitational Lensing, 57–74
  Datasets, 59
  Evaluation, 64
  Image Augmentation, 60
  Methodology, 62
  Results, 65
  Visualising and Interpreting Features Extracted by Convolutional Layers, 69

Instance Segmentation Algorithms, 43–55
  Mask R-CNN, 45
  SOLOv2, 51
  Transformers, 53
    DETR, 54
    ViT, 54
  YOLO, 50

Mask R-CNN, 75–101
  Conclusions, 99
  Dataset, 77

Evaluation Metrics, 86
Hyperparameter Optimisation, 88
Model Implementation, 81
Results, 91

Radio Astronomy, 5–12

Semantic Segmentation Algorithms, 40–43
U-Net, 40
U-Net++, 42