RHODES UNIVERSITY
Where leaders learn

Semantic Segmentation of Astronomical Radio Images: A Computer Vision Approach

A thesis submitted in fulfilment of the requirements for the degree of

MASTERS IN SCIENCE

in the Department of Physics and Electronics

AUTHOR:
Ramadimetse Sydil Kupa
ORCID Number: 0000-0003-0813-3177

SUPERVISOR:
Prof. O. Smirnov

CO-SUPERVISORS:
Dr. M. Atemkeng
Dr. K. Thorat

May 2021

Semantic Segmentation of Astronomical Radio Images: A Computer Vision Approach

Abstract

The new generation of radio telescopes, such as the MeerKAT, ASKAP (Australian Square Kilometre Array Pathfinder) and the future Square Kilometre Array (SKA), are expected to produce vast amounts of data and images in the petabyte region. Therefore, the amount of incoming data at a specific point in time will overwhelm any current traditional data analysis method being deployed. Deep learning architectures have been applied in many fields, such as, in computer vision, machine vision, natural language processing, social network filtering, speech recognition, machine translation, bioinformatics, medical image analysis, and board game programs. They have produced results which are comparable to human expert performance. Hence, it is appealing to apply it to radio astronomy data. Image segmentation is one such area where deep learning techniques are prominent. The images from the new generation of telescopes have a high density of radio sources, making it difficult to classify the sources in the image. Identifying and segmenting sources from radio images is a pre-processing step that occurs before sources are put into different classes. There is thus a need for automatic segmentation of the sources from the images before they can be classified. This work uses the Unet architecture (originally developed for biomedical image segmentation) to segment radio sources from radio astronomical images with 99.8% accuracy. After segmenting the sources we use OpenCV tools to detect the sources on the mask images, then the detection is translated to the original image where borders are drawn around each detected source. This process automates and simplifies the pre-processing of images for classification tools and any other post-processing tool that requires a specific source as an input.

Declaration

I, Ramadimetse Sydil Kupa, declare that this thesis titled, "Semantic Segmentation of Astronomical Radio Images: A Computer Vision Approach" and the work presented in it are my own. I confirm that:

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

Signed: R.S Kupa

Date: 12 November 2021

Contents

Acknowledgements

I wish to express my sincere appreciation to my supervisors, Dr. Marcellin Atemkeng, Dr. Kshitij Thorat; and Professor Oleg Smirnov: they convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without their persistent help, the goal of this project would not have been realized.

To my mentor and line manager of the science data processing team, Khutso Ngoasheng, I whole-heartedly appreciated his great advice which contributed enormously to the success of this study. I am also grateful to him for granting me time off work to focus on my thesis writing.

The physical and technical contribution of 'The South African Radio Astronomy Observatory' is truly appreciated. Without their support and funding, this project could not have reached its goal.

I wish to acknowledge the support and great love of my family, my mother, Kanyane Kupa; my father, George Kupa; my siblings, Phegelo, Lenaneo and Hyme; and my nephews, Tsamaisho and Oloketswe. They kept me going and this work would not have been possible without their emotional support.

I would like to also pay my special regards to my friends, Isabella Rammala and Portia Legodi, for listening to me talk about my research through the whole of the lockdown and assisting me where possible. To my best friend, Cheyeza Mabuza for counting the chickens before they hatch and already envisioning my masters graduation day. I truly appreciate the role each and everyone of you played towards this study.

Publications

The following is work that I was involved in during this research:

Conferences and Posters

- Neural Information Processing Systems (NeurIPS) Canada 2019 (NeurIPS)
- Deep Learning Indaba 2019 in Kenya (Deep Learning Indaba)
- Winner of best poster presentation at Deep Learning Indaba 2022 in Tunisia (Deep Learning Indaba)
- SARAO Postgraduate Bursary Conference 2019 and 2020

Journal publication

- W.D. Cotton, K. Thorat, J.J. Condo n, B.S. Frank, G.I.G Józsa, S.V. White, R. Deane, N. Oozeer, M. Atemkeng, L. Bester, B. Fanaroff, R.S Kupa, O.M. Smirnov, T. Mauch, V. Krishnan, F. Camilo, Hydrodynamical backflow in X-shaped radio galaxy PKS 201455, Monthly Notices of the Royal Astronomical Society, Volume 495, Issue 1, June 2020, Pages 1271–1283, `https://doi.org/10.1093/mnras/staa1240`

- R.S Kupa, et al, Semantic Segmentation of Astronomical Radio Images: A Computer Vision Approach - In preparation.

Source code

- The segmentation software developed for this project is released on GitHub via `https://github.com/sydil/Segmentor-and-Detector` and was inspired by `https://github.com/zhixuhao/unet`. See README for instructions.

Chapter 1

Introduction

1.1 Context to the study

The Square Kilometre Array (SKA) is a scientific project that aims to build the world's largest and most sensitive radio telescope. Under development by an international consortium, it is designed to answer questions about the nature of the Universe, the nature of gravity, the cosmic magnetic field, search for extraterrestrial intelligence, and reveal new mysteries about the Universe [Dewdney et al. (2009)]. The SKA is characterised by high sensitivity, high survey speed, high spectral and time resolutions, and a wide field of view. These characteristics ensure it can observe the many different objects in space to detect dim objects. The future SKA, its precursors MeerKAT [Jonas (2009)] and the Australian Square Kilometre Array Pathfinder (ASKAP) [Johnston et al. (2008)] are expected to produce vast amounts of data in the petabyte region [Sabater et al. (2017)]. MeerKAT is a South African radio telescope. It consists of 64 antennas with a minimum baseline of 28 m and a maximum baseline of 8 km. The antennas have a Gregorian configuration that provides optical performance and sensitivity. The frequency range of the L-band receiver is 856 MHz - 1712 MHz, and its angular resolution using the central frequency of 1284 MHz is 0.734 arcsec. ASKAP, on the other hand, is an Australian telescope with a maximum baseline of 6 km, a frequency range of 700 MHz - 1800 MHz and a field-of-view of 30 sq. degs. ASKAP aims to survey the visible sky to the root mean square (RMS) sensitivity of 10 μJy/beam. It is expected to produce a catalogue of about 70 million galaxies at 1100 MHz. Traditional data handling methods of transmitting, computing, analysing and archiving this data will not be adequate due to the advancement of the observation technology. Deployment of automatic techniques such as deep learning [LeCun, Bengio, and Hinton (2015)] is imperative for data analysis.

Deep learning is a subfield of machine learning based on ANNs with a significant trade-off of manual feature engineering. The standard format of deep learning consists of an input layer, one or more hidden layers, and an output layer, collectively referred to as a neural network. The first layers extract features in the data, and a decision is made in the two last layers. The layers that learn features are not directly designed by human engineers but are formed under the guidance of general-purpose learning procedures. This characteristic gives deep learning the advantage over classical model fitting. Radio telescopes produce visibility data, which are then rendered as images. Images from these telescopes contain significantly more sources and diffuse emissions than radio images from previous radio telescopes; as a result, it is challenging to use traditional methods to analyse them. This research, therefore, focuses on automating the process of source segmentation in such images through the use of deep learning techniques.

There are several types of architectures used for deep learning, e.g. (1) Artificial neural network (ANN) [Zupan (1994)], which consists of a group of neurons in each layer and is used to solve problems related to tabular, text, and image data. (2) Recurrent neural network (RNN) [Medsker and Jain (1999)] which is made up of a group of multiple neural networks with recurring connections in the hidden layers typically used for time-series, text, and audio data. (3) Convolutional neural networks (CNN) [O'Shea and Nash (2015)], which are primarily used to solve intricate image-driven pattern recognition tasks. Segmentation is a technique that gives us a far more granular understanding of the objects in the image. The objects in radio images are sources; this technique will help us understand the sources within the image and detect their precise location in the image.

1.2 Problem statement

Since the SKA will be the largest telescope of its kind in history, it will make serendipitous discoveries and contribute to making the most prominent data archive in the world [An (2019)]. Therefore, enormous challenges are expected with processing the SKA data. In recent years, CNN techniques have been applied in radio astronomy and have proved useful, e.g. (1) to detect system health for modern radio telescopes [Mesarcik et al. (2020)], (2) to detect radio frequency interference (RFI) [Mosiane et al. (2017)] and (3) Classification of Radio Sources [Aniyan and Thorat (2017)]. Aniyan and Thorat (2017), in the latter application, did a study on the classification of radio galaxies with AlexNet CNN based on galaxy morphology. This study looked at three classes of morphology, the Fanaroff-Riley (FR) classes I, II and bent-tailed galaxies. Their model performed acceptably well with an accuracy score greater than 80%. However, like the current frameworks in use, a user must perform the segmentation process manually as a pre-processing step before a classification framework step is conducted. This process is inefficient for images such as those expected from the future SKA telescope. The images have a much higher spatial density of sources, and far more complicated morphologies are revealed by sensitive observations. Moreover, it is time-consuming and can significantly affect the classification accuracy. It is evident that pre-processing the images is imperative in this field, and the work presented in this thesis addresses this problem. The techniques we are developing are expected to be particularly useful in this new era of radio telescope surveys, as the images show high concentrations of radio objects. A high precision, automatic source detection system is required to segment these radio sources from the clustered images.

1.3 Objective

In this research, we aim to automate the segmentation of sources from images using a Unet architecture [Ronneberger, Fischer, and Brox (2015)]. We use segmented images to detect the sources and draw borders around them. The Unet architecture is a universal function approximator with arbitrary accuracy which efficiently handles noisy inputs [Chen et al. (2016)]. It was initially developed for biomedical image segmentation, where it won a segmentation challenge by a large margin. Unet could be helpful, particularly for radio images, as it also deals with complex structures. Therefore, the study's objective is to investigate this data analysis method to solve the problem of analysing the vast amount of incoming data, and dealing with complexity in images, while minimising or avoiding time-consuming manual processes.

1.4 Contribution

Upon completion, the software produced by this study should assist astronomers in analysing and understanding radio images produced by telescopes. It should automatically segment the radio sources from the images with high precision. The output of this process is expected to be individual radio sources arranged in a catalogue. This tool will be invaluable to astronomers as it will avoid having to segment sources from images by hand before scientific investigations of the radio sources can occur.

1.5 Manuscript layout

This thesis is organised as follows:

Chapter 2 investigates basic interferometric radio imaging and source finding methods currently used in radio astronomy. We gain an understanding of how visibility data is imaged so that we can obtain the images to be used as input data for our study.

Chapter 3 investigates deep learning methods: what they are and where they are derived from in the greater scheme of machine learning. We take a look at the hyperparameters that are adjusted to optimise our work.

Chapter 4 investigates different segmentation techniques and their advantages and disadvantages. After carefully considering all the factors discussed in this chapter, Unet emerges as the most viable architecture that fits our purpose. It was initially designed for medical images; however, we will investigate its performance on astronomical images, and the findings will be detailed in the next chapter.

In Chapter 5, the Unet segmentation architecture is applied to radio images, and the ground truth images produced by pyBDSF and Unet are compared. Unet performs above average when applied to radio data with an accuracy of 99.8%.

Chapter 6 concludes and discusses future work.

Chapter 2

Interferometric radio imaging and source finding

2.1 Basics of radio interferometry

Karl Guthe Jansky (1932) was an engineer at Bell Laboratories who was tasked to perform an investigation to identify sources of static in overseas radio communications. He observed emission of unknown origin at 20.5 MHz (a wavelength of about 14.5 m). He eventually showed that this emission originates from the direction of Sagittarius, behind which lies the centre of our galaxy [Jansky (1933)]. However, the field of radio astronomy was truly born in 1940 when Grote Reber mapped the Milky Way galaxy at 160 MHz using a parabolic antenna [Reber (1940)]. Radio astronomy is limited to the frequency range of 15 MHz - 1.5 THz [Rohlfs and Wilson (2000)]. A wealth of scientific information can be recorded and studied at radio wavelengths. Radio telescopes are instruments which enable us to carry out observations of celestial objects at radio frequencies. Radio telescopes are constructed in different shapes and sizes from the specific type of radio waves they are designed to detect. Generally, every radio telescope has an antenna on top of a base and receiver equipment to capture and record the signals. An antenna is a device used to convert electromagnetic radiation into electrical currents. A radio telescope is one such example of a receiving antenna. The properties that follow are observed for a receiving antenna.

The radiation pattern (or primary beam pattern) of a receiving antenna defines the directional sensitivity of the antenna. It is not equally sensitive in all directions, as shown in Figure 2.1. This primary beam is composed of a main or major lobe. The size of the beam is quantified by the half-power beam width (HPBW) of the major lobe. The HPBW measures the angular extent of the beam between the points at which the major lobe falls to half its maximum value.

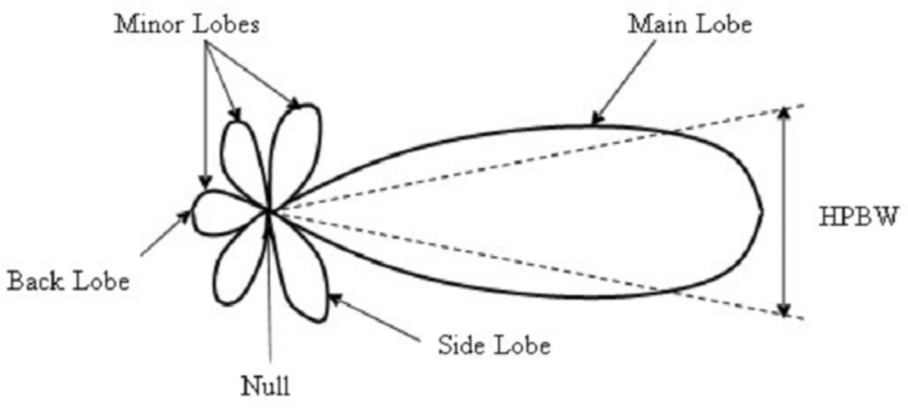


Figure 2.1: A radiation pattern of an antenna indicates the strength of the radio waves from sources. The main lobe represents radiation in the desired direction which HPBW measures, while the other lobes represent radiation in other direction [Abd Alaziz (2011).

For a telescope with a dish of diameter D, the attainable resolution R is:

$$R \propto \frac{\lambda}{D}, \tag{2.1}$$

where λ is the observing wavelength. To attain a high resolution image implies that the value of R should be very small. Since R and D are inversely proportional (as shown in Equation 2.1), increasing one parameter decreases the other parameter. To calculate the minimum angular separation between two light sources that may be resolved into distinct objects, we use $\lambda 1.22 \times R$, referred to as the Rayleigh criterion. The dish should be designed according to the desired resolution for a fixed observation wavelength. For example, the design can result in an impractically large dish size if we wish to archive a resolution similar to optical observations. Interferometry and aperture synthesis becomes imperative. Radio interferometry is a technique whereby several radio antennas are combined to simulate a single "aperture" (synthesised telescope) whose diameter corresponds to the largest distance between the antenna elements. Pairs of antennas form a baseline and the signal that each pair of antennas measures is recorded as a voltage. The voltage is synchronised through a correlator. The maximum resolution of the array is given by:

$$R_{max} \sim \frac{\lambda}{|\mathbf{u}|}, \tag{2.2}$$

where u is the maximum baseline vector and |.| is the Euclidean norm. An illustration of the possible maximum resolutions achievable by several dishes at a given frequency is shown in Table 2.1.

Maximum resolution (R_{max} in arcsec)				
λ / $|\mathbf{u}|$	28 m	100 m	8000 m	1000000 m
100 MHz	22099.8	6188	77.36	0.6187
1000 MHz	2209.98	618.8	7.736	0.06187
10000 MHz	220.9	61.88	0.7736	0.006187
100000 MHz	22.099	6.188	0.07736	0.0006187

Table 2.1: Different combinations of frequency and baselines achieve different resolutions (in arcsec). This table illustrates the possible achievable maximum resolution.

2.1.1 Two-element radio interferometer

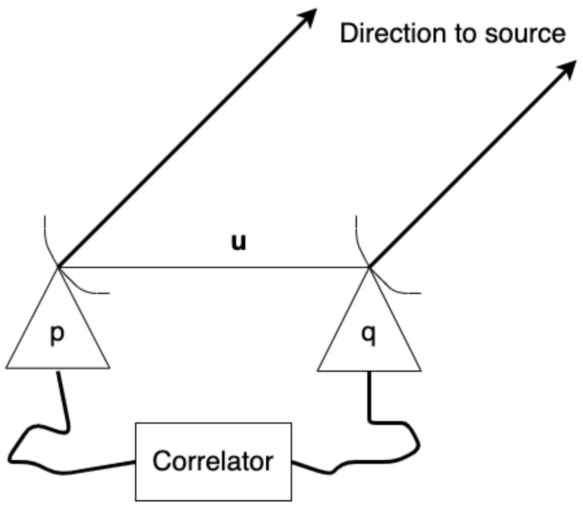


Figure 2.2: A two-element radio interferometer depicting two antennas receiving signals from the direction of the source. The signals are received by the antennas and then sent to the correlator.

Figure 2.2 illustrates a two-element radio interferometer. Suppose that p and q are two telescope antennas on the Earth's surface. For illustration, let us assume that the Earth is not rotating and that the source we are trying to measure is not directly above the antennas. An apparent baseline is seen by the source in such a case due to the projection of the baseline in the direction of the source. The wavefronts does not arrive at p and q simultaneously as indicated in Figure 2.2. This lag introduces a time delay that has to be incorporated to measure the signal from the same wavefront. In reality, when we make allowance for the Earth's rotation, the time delays and antenna direction need to be adjusted continuously. The plane's position relative to the Earth continuously changes as the Earth rotates and needs to be incorporated. This process is an illustration of how radio telescopes view the sky and the effects that are considered when viewing the sky.

2.1.2 Visibilities and imaging

Coherence is calculated by a visibility function [Rohlfs and Wilson (2000)]. This visibility function can be considered as a non-normalised measure of the electric field at pairs of points in a plane, adjusted according to the interferometer's characteristics.

We introduce the coordinate system for a practical application of a visibility function, such as imaging. The baseline vector $\mathbf{u}$ is described by component (u, v, w), which are measured in wavelength at the central frequency of the radio frequency signal band. The w-axis is the reference direction, usually chosen to contain the target radio source, oriented towards the phase centre of the observed field. The u- and v- axes point East and North in the (u, v) plane normal to the w-axis. The visibility function as described by Thompson, Moran, and Swenson (1991):

$$V(u,v,w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(l,m)e^{-2\pi i[ul+vm+w(\sqrt{1-l^2-m^2}-1)]}\frac{dldm}{\sqrt{1-l^2-m^2}}, \tag{2.3}$$

where $I(l, m)$ is the apparent sky in the direction of (l, m). That is the sky brightness distribution $I(l, m)$. The term $ul + vm + w(\sqrt{1-l^2-m^2}-1)$ describes the effect of the antenna locations on the earth, the rotation of the baseline while tracking at the source position (l, m). The term $\sqrt{1-l^2-m^2}-1$ comes from the compensating delay introduced by the correlator to ensure that $ul + vm + w(\sqrt{1-l^2-m^2}-1)$ is zero at the phase centre of the observation. Equation 2.3 is a not a Fourier transform, $\mathcal{F}$. In order to restrict it to a 2-dimensional Fourier transform, its either we must assume that the interferometer is tracking a small field in the sky (i.e. $\sqrt{1-l^2-m^2} \approx 1$) or, assuming a coplanar interferometer ($w = 0$):

$$\begin{aligned} V(u,v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(l,m)e^{-2\pi i(ul+vm)}dldm \\ &= \mathcal{F}\{I(l,m)\}. \end{aligned} \tag{2.4}$$

The continuous visibility in Equation 2.4 is sampled at each baseline. This equation is what an interferometer measures:

$$V^{meas}(u,v) = S(u,v)V(u,v), \tag{2.5}$$

where $S(u, v)$ is the sampling function. Inverting Equation 2.5 we obtain the dirty image:

$$\begin{aligned} I^D &= \mathcal{F}^{-1}\{V^{meas}(u,v)\} \\ &= \mathcal{F}^{-1}\{S(u,v)V(u,v)\} \\ &= \mathcal{F}^{-1}\{S(u,v)\} \circledast \mathcal{F}^{-1}\{V(u,v)\} \\ &= \mathcal{PSF} \circledast I(l,m), \end{aligned} \tag{2.6}$$

where $\mathcal{PSF} = \mathcal{F}^{-1}\{S(u, v)\}$ is the point spread function of the instrument, that is the response of the instrument to a point source. The dirty image from an interferometer is then expressed as the PSF's convolution and the sky's brightness. To get the sky brightness $I(l, m)$, we need to remove the PSF from the dirty image I^D using deconvolution, as shown in Figure 2.3. Deconvolution involves taking visibilities, removing the PSF and producing an undistorted image of the sky. There are different algorithms that can be used to deconvolve the dirty image, such as (1) Högbom's clean algorithm [Högbom (1974)], (2) maximum entropy (MEM) algorithm [Cornwell and

Evans (1985)], and (3) MS-clean [Cornwell (2008)]. Using these algorithms to clean the dirty image, we can approximate the observed visibilities to an image of the sky brightness. The image of the sky brightness is composed of radio sources and noise. In the next section, we shall learn more about our objects of interest: the radio sources within the images.

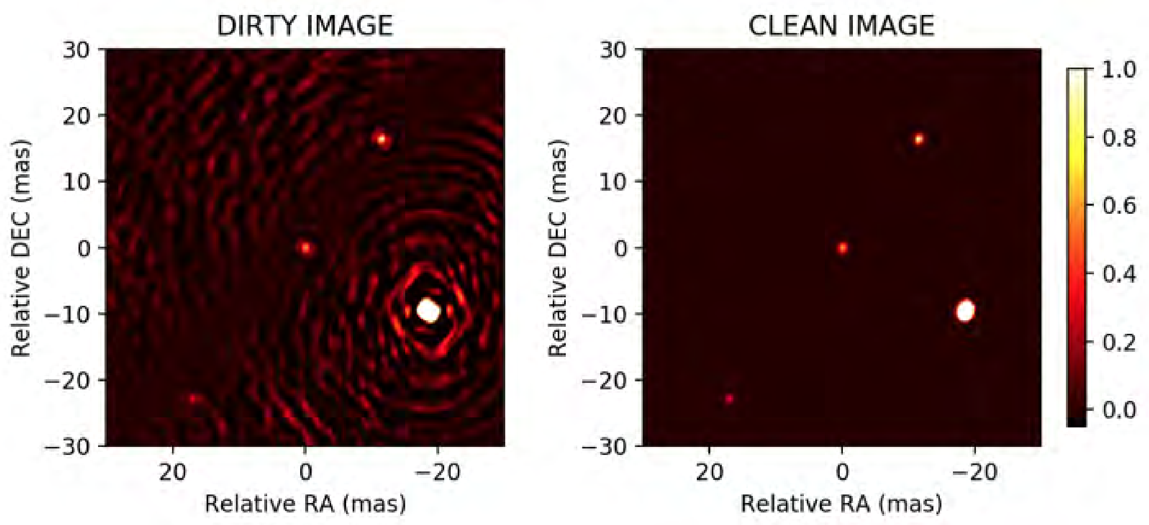


Figure 2.3: The right panel shows the dirty image, and the left plane shows the. deconvoluted image after 100 iterations of CLEANing [Zhao, An, and Lao (2019)].

2.2 Radio sources

2.2.1 What are continuum radio sources?

Objects in outer space that emit relatively large amounts of radio waves are called astronomical radio sources. Most astronomical objects emit radio frequency radiation, and radio galaxies, pulsars, quasars, and certain nebulas are among the objects that emit the strongest emission. The radio emission can either be a continuum or a line emission. Continuum emission is emitted by a source in a broad range of frequencies/wavelengths, while line emission is limited to a particular set of frequencies. This study will focus on continuum radiation. Extragalactic radio continuum surveys are a crucial tool for understanding the evolution of galaxies over cosmic time [Norris (2017)]. These surveys map large areas of the sky with high sensitivity in a uniform manner and are expected to detect objects over a wide range of distances. There are two dominant populations of radio continuum sources; namely, active galactic nuclei (AGNs) and star-forming galaxies (SFGs).

2.2.2 Active galactic nuclei (AGN)

A radio galaxy consists of dark matter, dust, gas, and stars with more light at radio than at visible wavelengths. There are compact radio galaxies and extended radio galaxies. Compact radio galaxies emit radio lobes not much larger than the galactic nucleus, while extended radio galaxies have lobes of radio emission extending millions of light-years from their nuclei. AGNs are powered by accretion onto supermassive black holes (SMBH) sitting in the nuclei of the galaxies and present unique observational signatures that cover a frequency range of more than twenty orders in magnitude on the full electromagnetic spectrum [Padovani et al. (2017)]. The properties of AGNs are:

1. They have very high luminosities of up to $\mathcal{L}_{bol} \approx 10^{48}\ erg \cdot s^{-1}$. This makes them the most powerful non-explosive sources in the Universe. Hence they are visible up to very high redshifts which is currently $z=$ 7.1 [Mortlock et al. (2011)].
2. In most bands, they have small emitting regions, of the order of a milliparsec, as deduced from their rapid variability [Ulrich, Maraschi, and Urry (1997)], giving them to have high energy densities.
3. Their luminosity functions have strong evolutions [Merloni and Heinz (2008)].
4. Their emission can be detected in all spectral bands.

There are two classes of AGNs:

1. **Radio-quiet AGN**
   These sources have radio emission that is too faint to be detected and associated with spiral hosts [Wilson and Colbert (1995)]. These are (1) Seyfert galaxies, (2) Radio-quiet quasars, and (3) low ionisation nuclear emission-line region galaxies (LINERs).

2. **Radio-loud AGN**
   These sources have radio lobes and jets with kinetic power, contributing a hefty portion of the total luminosity. Elliptical galaxies generally host them. These are (1) radio galaxies, (2) radio-loud quasars, and (3) blazars.

For the scope of this study, we shall focus on radio galaxies. Fanaroff and Riley (1974) classified radio galaxies into two morphological classes, namely Fanaroff and Riley class I (FRI), and class II (FRII). FRI sources have their brightest regions near the centre of the source, with darker edges as depicted in Figure 2.4. On the other hand, FRIIs are sources with the brightest regions close to the furthest extent of the emission, making the edges brighter as depicted in Figure 2.5. There is also a strong correlation between the morphological classes FRI and FRII and luminosity, with FRII sources being the most luminous.

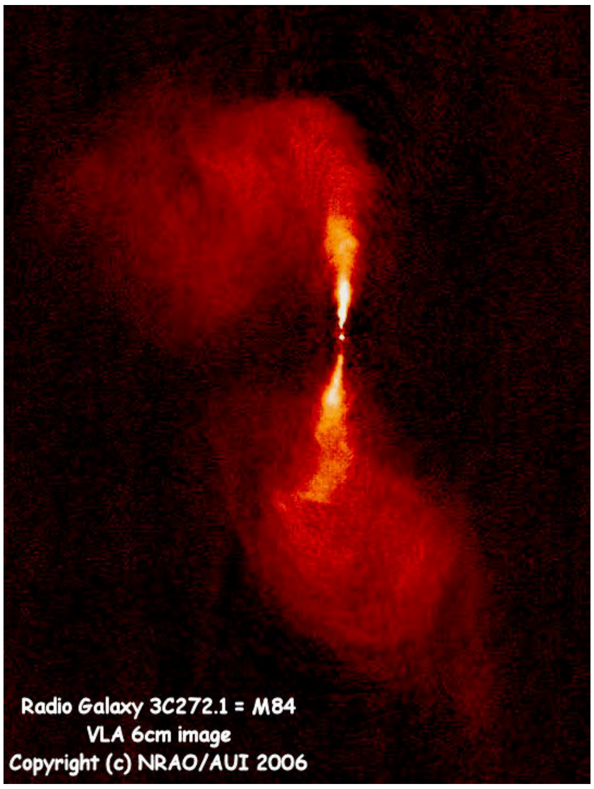


Figure 2.4: Messier 84 (M84) is an elliptical galaxy in the Virgo Cluster. This galaxy lies at an approximate distance of 60 million light years from our planet (Earth) and has an apparent magnitude of 10.1 V [*M84* (2006)].

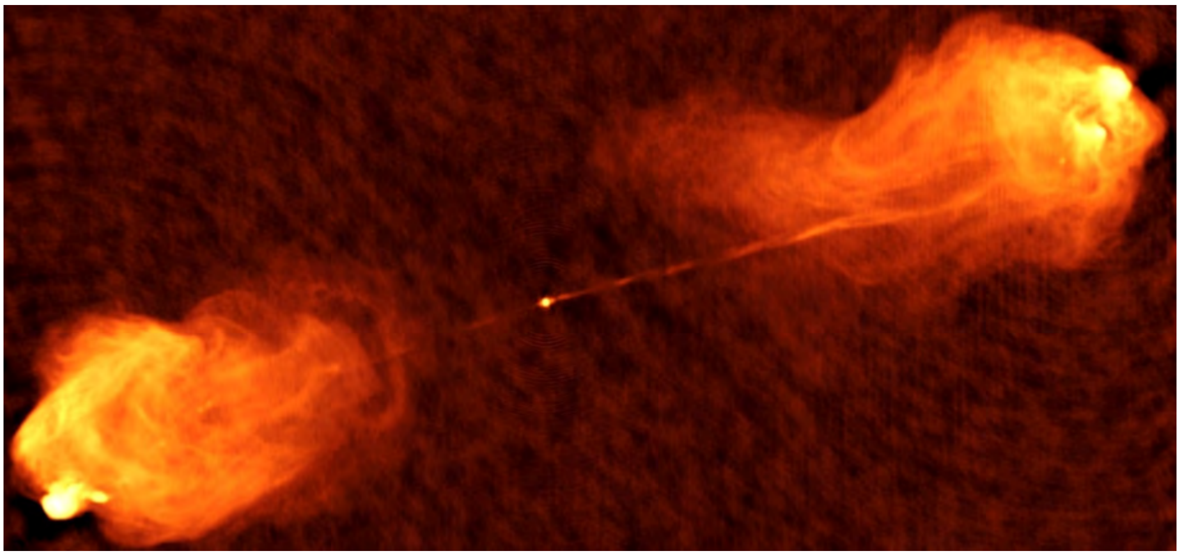


Figure 2.5: Cygnus A is one of the strongest radio sources in the sky. This galaxy lies approximately 600 million light-years from Earth and has an apparent magnitude of 16.22 V [*Cygnus A* (1984)].

2.2.3 Star-forming galaxies

Forming new stars is one of the important processes of a galaxy's life cycle. Chomiuk and Povich (2011) discovered that our own galaxy, the Milky Way, has a star-forming rate of only a few new solar masses per year. Firstly, the star-forming process involves the accumulation of molecular gas in large clouds, followed by a collapse under the influence of gravity. The collapse then births a new population of stars of various masses. These stars will range from dwarfs to highest-mass stars. They are largely responsible for the UV to optical emission observed in star-forming galaxies. As they age, these highest-mass stars emit fast outflows but eventually, they die in powerful supernova explosions. The preceding processes can significantly impact the galaxy's interstellar medium (ISM) by enriching, heating, and expelling the gas that could potentially form new stars. The evolution of stars being born and interacting with the galaxy's ISM is closely related to the evolution of the galaxy. A star-forming galaxy is shown in Figure 2.6.

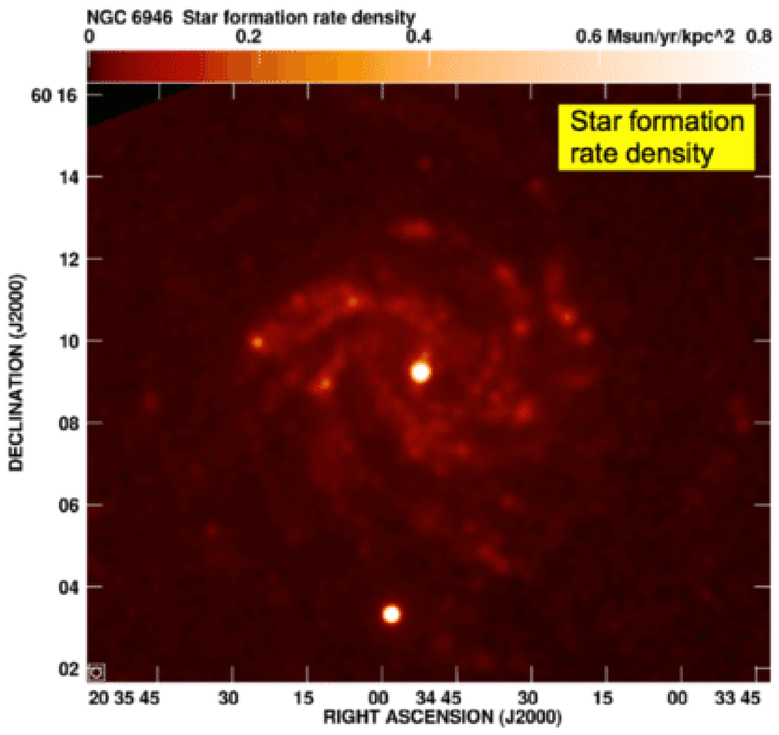


Figure 2.6: NGC 6946 is a star-forming galaxy with a small bright nucleus. This galaxy lies at an approximate distance of 22.5 million light years from our planet and has an apparent magnitude of 9.6 V [Source: Beswick et al. (2015)].

2.3 Source finders in radio images

Several tools are implemented for source finding in astronomical images. We classify them into two categories: the traditional approaches and the deep learning approaches.

2.3.1 Traditional source finders

ProFound is a source finding and image analysis package [Robotham et al. (2018)]. This algorithm detects sources in noisy images, generates segmentation maps, and measures the source statistics, such as flux, size, and ellipticity. Profound uses photometry to create the segmentation maps. The photometry technique measures the flux of light radiated by sources. It uses consecutive dilation and flux measurement operations. It is robust in its design but not super fast. Profound starts by making a segmentation map, followed by consecutive dilations and flux measurements on the segmentation map. Thereafter it calculates the convergent flux segment for each source. These combined make a final segmentation map. ProFound guarantees good flux convergence by preserving the sources' geometric features. ProFound's performance is satisfactory on real and simulated data. It uses the R data analysis platform[1], which is open-source and available on GitHub [2]. By default, it outputs a segmentation map, a sky map and a sky RMS. However, it has memory limitations for large images, which means that large surveys will be challenging to process. Its ability to assign flux to blended sources is rudimentary in comparison to other photometry packages. This source extractor has been successfully used at radio wavelengths [Hale et al. (2019)].

Python blob detector and source finder (pyBDSF) [Mohan and Rafferty (2015)] is another state-of-the-art package that extracts sources from radio interferometry images and makes the properties of the sources available. PyBDSF can be used in CASA[3] but can also be used in Python scripts, and it is open-source. This algorithm pre-processes an image by calculating its basic statistics, such as RMS and the mean of the image. After that, sets a threshold value (which could be fixed or calculated using a False Detection Rate formula) that will be used to separate sources from the background. The threshold is used to find the islands of source emission. Depending on the number of degrees of freedom, it will fit multiple Gaussians to the islands. The islands are then decomposed into shapelets. The shapelets and Gaussians are used to create residual images. Gaussians within a specific island are grouped into discrete sources. Then we subtract the residual image from the fitted Gaussians and generate images at various scales. Traditional source extraction software fits multiple Gaussians to source pixels which are adequate for point sources. The limitation of this algorithm is that it struggles to perform on extended sources [Mohan and Rafferty (2015)]. This source finder is available on GitHub [4].

Aegean [Hancock et al. (2012)] is a compact source extractor for radio images. Given a certain threshold, it seeks out islands of pixels above the threshold. It then uses the curvature within the image to determine how many Gaussian components would describe the island. Aegean works well for compact radio sources in images that do not have diffuse background emission. Hence pre-processing the image is required to remove diffuse emission. Aegean produces acceptable results for a range of applications

[1]https://cran.r-project.org/web/packages/R.oo/index.html
[2]https://github.com/asgr/ProFound
[3]https://casaguides.nrao.edu/index.php?title=Getting_Started_in_CASA
[4]https://github.com/lofar-astron/PyBDSF

but it is of limited use as it was designed for compact sources and does not cater for extended sources. This source finder is available on GitHub [5]. However, it struggles to segment single sources with multiple disconnected components, such as a giant radio galaxy whose lobes are completely separated accurately.

2.3.2 Deep learning source finders

ConvoSource [Lukic, De Gasperin, and Brüggen (2019)] is a novel deep learning model that identifies parts of the image that belongs to a source and that which belongs to the background. To check the performance of this model, the model is tested on artificial data that was created for the SKA challenge [Gupta (2021)]. Three classes were on an investigation: SFGs and two types of AGNs. The data has two frequencies (560 and 1400 MHz), three total integration times of 8 h, 100 h and 1000 h and three SNRs. The findings are pivoted on the SNR characteristic. At lower SNRs, Gaussian-fitting methods outperform ConvoSource in recovering AGN-type sources. However, ConvoSources outperformed them at recovering SFGs. At a higher SNR, ConvoSource outperforms the Gaussian-fitting method at the recovering AGN-type sources, whereas the Gaussian-fitting method outperforms Convosource in recovering the SFGs-type sources.

COSMODEEP is another extended extragalactic radio source finder that was developed by Gheller et al. [Gheller, Vazza, and Bonafede (2018)] using CNNs, for upcoming survey data. They have proven to be accurate in detecting very faint sources on simulated images. ClaRAN which means classifying radio sources automatically with neural networks [Wu et al. (2019)]) is a localization and recognition source finder trained on radio galaxy zoo data [Banfield et al. (2015)]. It produces accuracy that are greater than 90%. Deep Source [Sadr et al. (2019)] is another dynamic CNN point source finder on simulated images. This model requires tuning hyperparameters. These hyperparameters are the number and type of layers and the batch size, etc. This is in combination with the deep learning components such as the cost function and the gradient descent method.

2.3.3 Limitations of source finding

Due to correlated noise, radio astronomy images are difficult to segment because noise can mimic a real source (i.e. pixel to pixel noise is not a random, independent quantity). Using "standard" algorithms, one can pick up fake sources, especially near-the-noise thresholds, which is why radio astronomy segmentation generally uses Gaussian fitting and high thresholds. The other limitation is encountering a single source with multiple disconnected components, such as a giant radio galaxy whose lobes are entirely separated, which can be challenging to segment accurately. Classical segmentation approaches find it unfeasible to segment such radio sources.

[5]https://github.com/PaulHancock/Aegean

This chapter discussed the basics of radio interferometry and how radio telescopes collect radiation from sources as well as convert them into visibilities. We also look at how the visibilities are transformed mathematically to become images. This chapter helps us understand the astronomy behind the data we will use. In Chapter 3, we shall discuss a specific method of machine learning (deep learning) that this research makes use of to automate the segmentation of sources from radio images (both compact and extended) for the upcoming large surveys.

Chapter 3

Fundamentals of deep learning

A famous computer pioneer, Joseph Carl Robnett Licklider [Gupta (2021)] imagined a human-computer symbiosis where man and machine could interact closely. To set up the interaction process, a set of defined rules (or instructions) is defined and for which each actor (machine or human) is required to perform a task based on these set of rules. The latter was the birth of machine learning, which today has shown its potential to solve complex real-world problems in several application areas. This chapter discusses machine learning and several sub-fields of machine learning such as deep neural networks, convolutional neural networks.

3.1 Introduction

There is a limit to the speed and precision humans can attain when performing tasks beyond which external assistance is required. Machines can enhance human efforts, assisting them by performing tasks that require some intelligence, e.g. decision-making. The field of research to enable machines to act and think like humans is called artificial intelligence [Saleh (2019)]. A sub-field of artificial intelligence is machine learning. Guided by algorithms, machines learn to perform tasks independently without explicitly being programmed [Ławrynowicz and Tresp (2014)]. There are two main categories of machine learning: shallow learning algorithms, which require pre-processing to extract features, and deep learning algorithms, which do not require the assistance of feature extraction. In the big data regime, deep learning algorithms have outperformed shallow learning in terms of accuracy, as they enable automatic learning through the absorption of vast amounts of unstructured data such as text, images, or video [Mathew, Arul, and Sivakumari (2021)]. Novel developments in hardware and optimisation approaches have boosted the performance of training such large and robust networks. They provide a platform to learn data representations with multiple levels of abstraction using models made up of multiple processing layers. These methods have shown promising results in many fields such as object detection, speech recognition and biomedicine. However, the downside of deep learning is that it requires large amounts of data to perform better than other techniques. It is also computationally expensive to train due to the depth and complexity of the models. There is also no standard theory to guide on selection of tools within the deep learning model.

3.2 Learning methods

Machine learning is a tool used to give insight and turn it into knowledge. Learning is done by programming a computer to think like a human. The amount of data accumulated in the past 50 years is vast, and it is also helpful to explore the data and find any hidden patterns in the data. Machines are therefore programmed to automatically find value in the data that would be tedious for humans to perform. The findings in these data can then be used for predictive analysis and decision-making.

There are four types of machine learning methods: supervised, unsupervised, semi-supervised and reinforcement learning. Supervised learning algorithms learn from labelled data to predict outcomes of unseen data [Liu and Wu (2012)]. The labelling of the data entails tagging the correct class to it. It is similar to learning in the presence of a teacher or supervisor. The aim is to understand the mapping between inputs and outputs. For instance, we can try to understand the mapping between the weather forecast and visitors to the beach. Classification and regression are two techniques of supervised learning. Classification is used to classify similar data points into different classes and then find the rules to explain how to separate the different data points. Regression is another technique, and, differently from classification, regression outputs a value rather than a class. When we need to predict numerical values like stock market prices and the probability of an event happening, we use regression. Unsupervised learning algorithms allow the model to discover patterns and information on its own that were not previously discovered [Wani et al. (2020)]. It processes and learns from data that is not labelled. The aim is to discover hidden patterns in the data without any labels. Only the input data is given with no label, and we learn the hidden patterns. For instance, sorting out different colour coins into different colour groups. Clustering is a common technique that is used in this learning to create groups with differing characteristics without supervision. Dimensionality reduction is another technique that aims to discover the essential features such that the original features can be reduced. The reduced set will be smaller and more efficient but still contain important data. The other common technique is association. In this approach, rules are unpacked that are capable of describing the data.

Semi-supervised learning is a combination of both supervised and unsupervised learning. The learning process is not completely supervised, nor do we let the technique do its own thing. It is the middle ground. A common technique here is generative adversarial networks (GANs). GANs make use of two neural networks, a generator and a discriminator. The first generates output, and the latter critiques it. By competing and battling against each other, they both become skilled. Reinforcement learning algorithms solve a task through trial and error [Heidrich-Meisner et al. (2007)]. Essentially, an agent (one who takes decisions that will result in either a reward or punishment) is created to observe and analyse the environment in which it is put; therefore, it can take actions and interact with the environment. Intelligent machines are those that learn and make decisions like humans. The aim here is to use rewards to learn instead of labels. This algorithm is popular in robotics and in creating games.

Two types of neural networks used in deep learning are described in the following sections. These architectures have been applied in many fields, such as in computer vision [Cireşan, Meier, and Schmidhuber (2012)], machine vision, natural language processing, audio recognition, social network filtering, speech recognition, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs. They have produced results which are comparable to human expert performance [Schmidhuber (2015)].

3.3 Artificial neural network

The brain controls the entire human body. The smallest building block of the brain is a cell called a neuron. The interconnected neurons make up a "neural network" of the brain. Herculano-Houzel (2012) says there are approximately 100 billion neurons in the human brain which can learn as well as adapt to a new and changing environment [Walczak and Cerpa (2003)]. The human brain can analyse complex information and make decisions based on its analysis. The theory of artificial neural networks (ANN) is inspired by the basic neurological structure of the human brain. ANN's simplest form mimics the human brain, similarly connecting artificial neurons to biological neurons. Rosenblatt (1958), McCulloch and Pitts (1943) mathematically modelled the brain's neural activity. Hebb (1949) further explained learning in the brain using reinforcement-based mechanisms. After that, Rosenblatt (1958) developed a computational model of brain processing elements; this is where ANNs research started.

3.3.1 The biological neuron

A biological neuron [Vrahatis et al. (2000)] constitutes of a soma (i.e. cell body), an axon and a dendrite as shown in Figure 3.1. The soma constitutes a nucleus and other chemical structures necessary to supply the cell. The *axon* is responsible for transporting electrochemical signals from one neuron to another. In contrast, the *dendrite* receives signals from a neuron and transports them to the cell body. A *synapse* is a connection between two neurons or neuron dendrites to muscle cells. A neuron triggers its signal to be transferred to the subsequent neurons once the signal's strength reaches a particular threshold. Sending signals to the neurons to get triggered co-occurs on multiple neurons. An *axon hillock* located at the end of the cell body controls the neuron's firing. Many branches of *axon terminals* form at the end of the axon.

3.3.2 An artificial neuron

An artificial neuron imitates the functional structure of a biological neuron. Adapting a biological neuron to develop an artificial neuron is not exact. In a biological neuron, communication is done via the diffusion of positive and negative ions, while artificial neurons are mathematical constructs mimicking this operation. Figure 3.2 shows the building blocks of an artificial neuron. Figure 3.2 warrants detailed explanation:

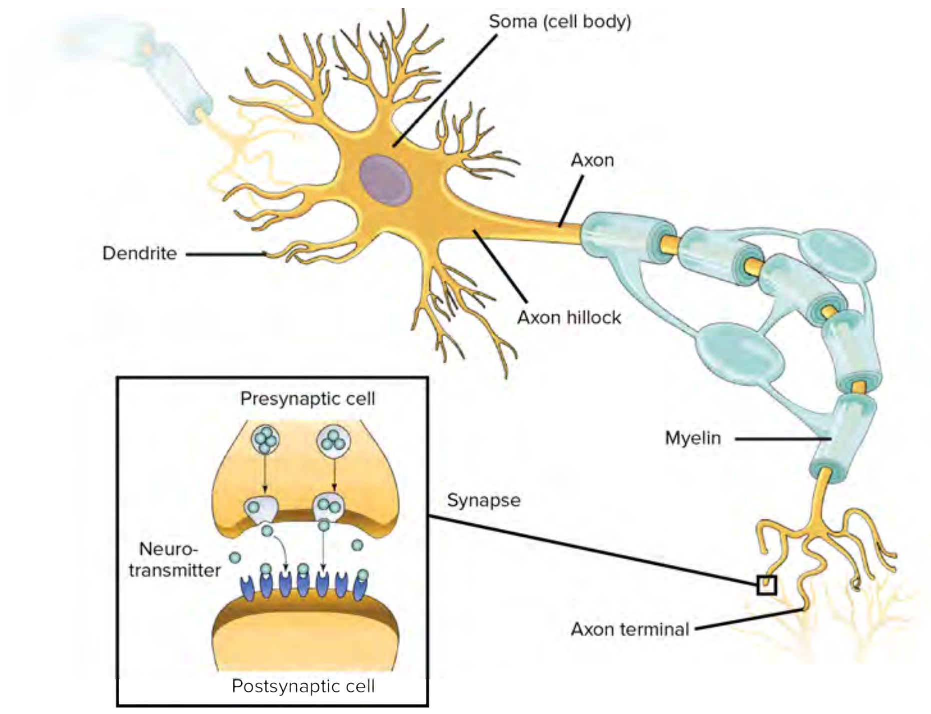


Figure 3.1: An image of a biological neuron which takes in input information through the dendrites, processes it and outputs the processed information through the axon terminal [*Biological neuron* (2010)].

assuming information or data in the artificial neuron, each data point x_j is scaled by a corresponding weight w_{ij}. The overall product is then summed and passed through an activation function that can help model the data's non-linearity. A bias can be added to the inputs to allow a shift in the activation function. A neuron is expressed mathematically as follows:

$$y_i = f(\sum_{j=0}^{n} w_{ij}x_j + b_j), \tag{3.1}$$

where f is the activation function, b_j is the bias and $i, j \in$ [0, n]

For instance, assume we have five neurons with no added bias in the neuron. Again, assume w_{i1} = w_{i4} = 1, w_{i2} = w_{i3} = -1 and w_{i5} = 0 then Equation 3.1 becomes,

$$y_i = f(x_1 - x_2 - x_3 + x_4). \tag{3.2}$$

Suppose now that the activation function selects the maximum between 0 and the strength of the signal then,

$$y_i = max(0, x_1 - x_2 - x_3 + x_4). \tag{3.3}$$

If we take two input examples such that $\mathbf{x} = (2, 4, 5, -1)$ and $\mathbf{x} = (3, -2, 4, 3)$ then $y_1 = max(0, -8) = 0$ and $y_2 = max(0, 4) = 4$. The value of the inputs towards

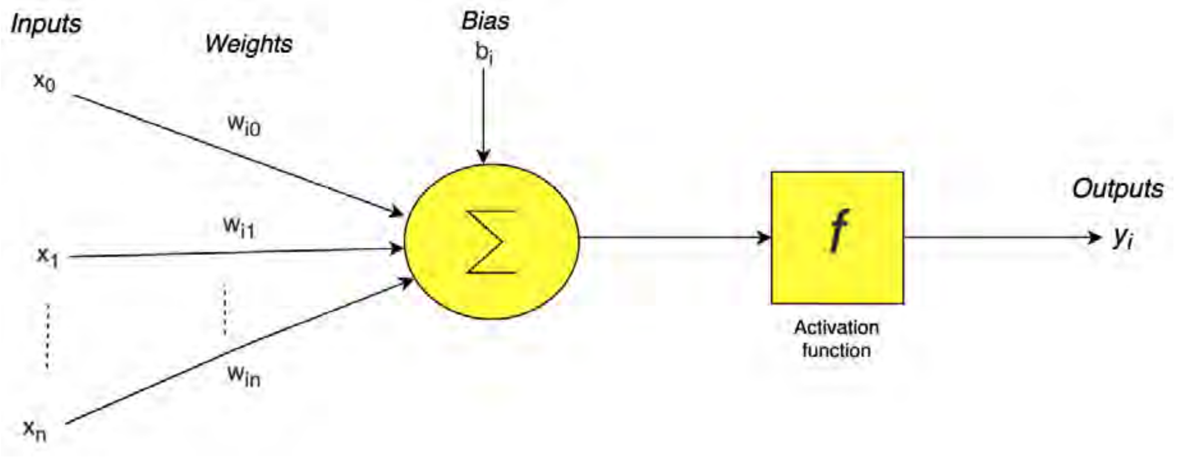


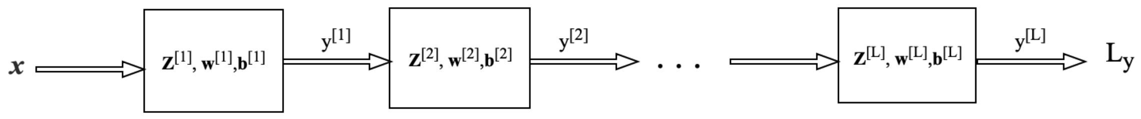
Figure 3.2: The structure of an artificial neuron that takes input information, processes it and outputs the processed information through the next layer.

the signal depends on the weight value put upon it. Weights can either be positive, negative or zero. Positive weights imply the signal is amplified; negative weights imply that the signal is reduced, while a weight of zero suggests there is no connection between two neurons. The weights can be adjusted to obtain the required output in a learning or training process discussed in Section 3.3.3.

An activation function is the weighted sum of the input that is transformed into an output. There is no rule on which activation function to choose on the various layers; however, they play a very impactful role in the performance and capability of the network [Guarascio, Manco, and Ritacco (2019)]. A hidden layer receives input from a previous layer and provides output to the following layer. These are the standard activation functions used, but they are not the exhaustive list. Rectified Linear Activation (ReLU) is easy to implement and effectively overcome vanishing gradients. A limitation that prevents training deep models, although It has a limitation of dead units. The Logistic (Sigmoid) function takes any real value as input and outputs values from 0 to 1. Larger input values tend to have output closer to 1, whereas the smaller input values tend to have output values closer to 0. Hyperbolic Tangent (Tanh) is similar to a sigmoid function even in shape. It takes any real value as input and outputs from -1 to 1. Larger input values tend to have output values closer to 1, whereas the smaller input values tend to have output values closer to -1. The output layer directly outputs a prediction of the input layer. All feed-forward neural network models must have an output layer. These are the standard activation functions used in the output layers. The linear activation function does not change the weighted sum of the input but returns the value as it is. Logistic (Sigmoid) as described above and softmax since it inputs a vector of real values and the output is a vector of the same length; moreover, the values sum up to 1 as in probabilities.

3.3.3 Learning

The learning process is done in two phases, the forward pass and the back pass. A forward pass structure is abstractly described as follows:



x
$Z^{[1]}, w^{[1]}, b^{[1]}$
$y^{[1]}$
$Z^{[2]}, w^{[2]}, b^{[2]}$
$y^{[2]}$
...
$Z^{[L]}, w^{[L]}, b^{[L]}$
$y^{[L]}$
L_y

Figure 3.3: The structure of a forward pass of a neural network. It combines the components of an ANN to achieve a prediction $y^{[L]}$. The input information contained in $\boldsymbol{x}$ is passed to the consecutive box, where it gets processed until it reaches the output stage.

The input layer $\mathbf{x} = (x_1, x_1, ..., x_n)$ is data that goes through the first layer denoted in the first block of Figure 3.3. It collects the weights associated with this layer $\mathbf{w}^{[1]}$ (which are the initialised weights) and bias $\mathbf{b}^{[1]}$ (which is the initialised bias) and uses these to calculate $\sum_{j=0}^{n} w_{ij}x_j + \mathbf{b}_j$ which we then call $\mathbf{Z}^{[1]}$. To move to the next layer, an activation function $f^{[1]}$ is applied to $\mathbf{Z}^{[1]}$, which becomes $\mathbf{y}^{[1]}$, the input vector to the next layer. In a similar process $\mathbf{w}^{[2]}$ and $\mathbf{b}^{[2]}$ are used to calculate $\mathbf{Z}^{[2]}$ to get the signal $\mathbf{y}^{[2]}$ for the next layers, and so on. Simultaneously, records of the weights and biases for all the layers up to the final layer L are kept. The output of layer L is $y^{[L]}$, which is the prediction of $\mathbf{x}$. Since the main parameters of the process weights and bias were randomly initialised, this might not be the most efficient approach to predict $\mathbf{x}$ as accurately as possible.

In the forward pass [Liu et al. (2015)], we made assumptions about the main parameters of the process and initialised the weights and biases to random numbers. We managed to get a prediction, but it is not necessarily the best prediction for the data. In this section, we review the weights and the biases to adjust them to better reflect the expected outcome from the data. Figure 3.4 shows the process of going back to make the adjustments through the back pass.
The output of layer L $y^{[L]}$ (which was obtained from the forward pass) is the probability of binary events occurring. Let the corresponding target be $\mathbf{t}$. We want to measure the discrepancy between $y^{[L]}$ and $\mathbf{t}$, to determine how well the network has predicted $y^{[L]}$ as close as possible to $\mathbf{t}$. The objective function is given as:

$$\mathbf{L}_y = E_p\big[l(\mathbf{x}, \mathbf{t}, m)\big], \tag{3.4}$$

where m is the model and $E_p[.]$ is the expectation of an unknown distribution p, $y^{[L]} = m(\mathbf{x})$. For a regression problem, $\mathbf{L}_y$ can be measured as quadratic:

$$\mathbf{L}_y = \frac{1}{n}\sum_i (m(\mathbf{x}) - t_i)^2 \tag{3.5}$$

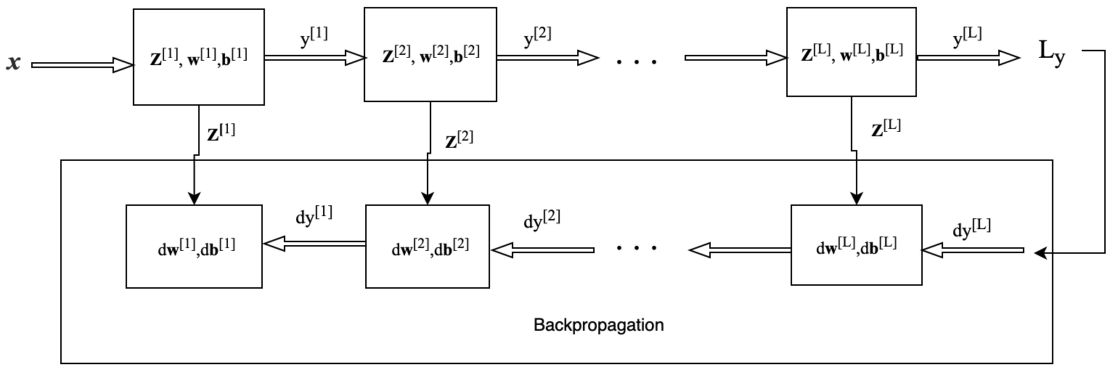


Figure 3.4: The structure of a back pass of a neural network. The border area shows how an error is propagated back through the layers to get the optimal weight and bias initialisation.

and for a classification problem from a cross-entropy perspective:

$$\mathbf{L}_y = -\sum_i t_i \log(m(\mathbf{x})). \tag{3.6}$$

If predictions deviate from the ground truth, the loss will be very high; otherwise, it will be very low. We use this function to adjust the network's weights and then continue performing iterations until the objective function is acceptably low [Ward and Joe (1963)]. Thus the objective function guides us to adjust the weights to be optimised. The forward pass is an activation flow, while the back pass is an error flow [Liu et al. (2015). We propagate from the loss function to the initial weights until we reach the optimal weights. This process is done by calculating derivatives to return to the initial weights. This process is elementary as the inputs are multiplied by the initial weight and operated on by a single activation function. Backward pass involves computing derivatives for each layer in the composite function and then passing them through the optimisation algorithms. Optimisation algorithms can be used to alter the attributes of the neural network to reduce the objective function by propagating back and adjusting the weights. A classical method to update the weights is the gradient descent optimisation flow as shown in Equation 3.7:

$$(\boldsymbol{w}^i)^{new} \leftarrow (\boldsymbol{w}^i)^{old} - (\alpha)\frac{(\partial \boldsymbol{L}_y)}{(\partial \boldsymbol{w}^i)^{old}}. \tag{3.7}$$

The learning rate is denoted as α in Equation 3.7, whereas the second term is the partial derivative of the loss function with respect to the weights. Each epoch produces the average loss of the training set, which is the difference between the predicted item and the ground truth item. To update a weight, we measure the rate of increase of $\mathbf{L}_y$ with respect to the change in $\mathbf{w}^i$. That is the derivative of the loss function (slope). An illustration of how the weights are updated is shown in Figure 3.5. At the start of the forward pass model, we initialised random weights and biases. This initialisation is similar to randomly placing a point on the curve, as shown in red. If the slope is positive, the error will increase if we increase the current weights. Therefore we should decrease the current weights. The error will decrease if the slope is negative as we increase the current weights. Therefore we should increase the current weights. Otherwise, if the slope is 0, do nothing, as this is the desired slope. This process will be repeated until convergence or when we find optimal weights that reduce error to an acceptable level. Moreover, the convergence depends on several factors, such as the quality of the training set, the initial parameters are chosen, the meta-parameters of the network, the optimisers and the learning rate.

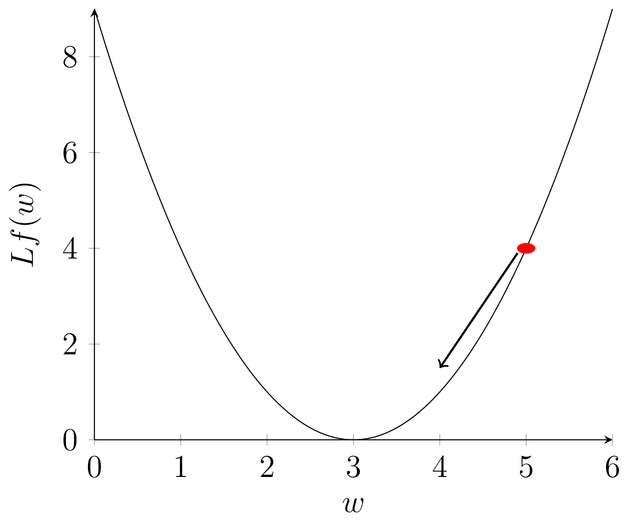


Figure 3.5: A quadratic function is used to optimise the weight initialisation starting from the point in red.

This section introduced ANNs and their relation to the biological neuron. The key attributes of an ANN, such as weights and bias initialisations, activation functions, and forward and back passes, are explained. We also look at the neural network's learning process and how we can optimise it. An ANN is computationally expensive since the different neurons of the model do not share weights. Therefore, it becomes dense, unlike convolutional neural networks, which share weights of neurons with sparse representations. Previously, the structure of ANN would be two layers deep as it was computationally extensive to build more than this. With computational advancements feasible, it has become common to build neural networks with ten or even 100 layers. Having these many layers assists the computer in seeing and learning complex situations. In the next section, we shall examine the convolutional neural network structure in detail.

3.4 Convolutional neural networks

Deep learning is commonly used in image processing and computer vision [LeCun, Bengio, and Hinton (2015)] applications. Convolutional neural networks (CNN) are often used to perform image processes such as recognition [Simonyan and Zisserman (2015)], classification [Krizhevsky, Sutskever, and Hinton (2012)], and segmentation [Kayalibay, Jensen, and Smagt (2017)]. More traditional algorithms that were used in the past to perform image recognition included object detection and semantic segmentation are linear classifiers [Skurichina and Duin (2002)], Bayesian classifiers [Cheeseman et al. (1988)], support vector machines [Suykens and Vandewalle (1999)] and decision trees [Apolloni, Zamponi, and Zanaboni (1998)]. These algorithms still outperform deep learning algorithms on small-scale data. However, with the advent of Big data, deep learning algorithms outperform traditional algorithms in terms of accuracy. A CNN uses a type of deep learning algorithm in the forward pass network to extract features from data automatically, unlike an ANN. CNN's have a sparse representation and are computationally cheaper for a given number of layers and neurons with fewer demands on memory when compared to an ANN. A CNN comprises two essential parts; the first is where knowledge is extracted from the data (feature extraction), and the second is classification. Features are extracted using a well-known process borrowed from signal processing; a convolution operation, followed by pooling, is applied to downsample the result of the convolution to reduce the representation. The output of the convolution and pooling is pasted into an activation function to form the feature map. The convolutional operation, the pooling and the activation function are organised into layers. The classification part occurs at the end of the feature extraction process when the data is clean and ready to be used by an ANN to make a decision. This decision-making step is also known as a fully connected layer. The local receptive field is responsible for identifying elementary visual features such as any contrasts in images, corners of objects, edges, and endpoints. These features are combined later to detect high-order features such as identifying objects or shapes. The weights are shared across the entire image: weights useful to identify the elementary visual features in one image segment have a high probability of being useful in the other parts of the image and are thus applied. It is also called a sparse model for this reason. These shared weights are distributed in planes to form a feature map. Each of these planes is constrained to perform the same operation on different parts of the image. A typical CNN model takes an input image and passes it through a series of convolution layers with filters, a pooling layer and a softmax function to class the object with probabilistic values between 0 and 1. We shall discuss the basic mathematical operations that make a CNN, how to operate on images, the basic building blocks of CNN, convolutional layers, padding, striding, and the pooling operation performed to merge information across adjacent regions.

3.5 Convolution

Convolutions are operations that are used to extract features from arrays. Mathematically, a convolution between two functions, say h and g, is defined as:

$$[h \circledast g](x) = \int h(l)g(x\text{-}l)dl. \tag{3.8}$$

This equation implies that we are measuring the overlap of functions h and g when they are shifted across the area of h. For discrete objects, a 2 dimensional convolution is given by:

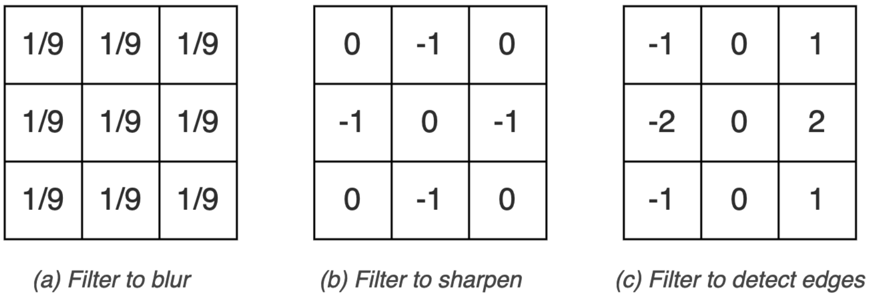
$$G(m,n) = \sum_{\sigma,k} g(\sigma,k)h(m\text{-}j,n\text{-}k), \tag{3.9}$$

where m and n are the indices of rows and columns of the convolved image respectively.

Usually, the size of the input image h is larger than the filter size of g since the filter will convolve the input image. Equation 3.9 implies an overlay of the filter array g on the input array h, and a convolution of the corresponding values from both arrays. We slide the convolution filter from the array's left to its right side, sliding at a specified number of pixels per step. At each step, the result is written to an output array. Each filter is a matrix of numbers representing a pattern the filter looks for in the image. Different filters assist the CNN in learning different features of the image during convolution, as shown in Figure 3.6. The learned features are then stacked up to form a feature map that acts as input to the next layer. For example, we have a neural network with one hidden layer and select a set of 8 filters for the input layer, with a set of 32 filters in the next (hidden) layer. The first convolutional layer will use the set of 8 filters to convolve the input image and stack the resulting values into a feature map consisting of 8 arrays. The feature map becomes the input to the first hidden layer. Since this layer has 32 filters, these 32 filters will be used to convolve each array in the feature map resulting in another feature map of size 32×8. An illustration of a single 2 dimensional ($2D$) convolution is shown in Figure 3.7. Other kinds of convolutions can be applied, such as dilated convolution, which creates spaces between pixels in the filter to increase the receptive field but at the same time does not add to the computational cost. However, it is prone to spatial loss error because we use a one-pixel value to represent an area of pixels.
A single pixel after convolution in an image in a CNN is expressed mathematically as follows:

$$y_i = \sum_{j=0}^{n} w_j x_{j+1} + b, \tag{3.10}$$

where b is some bias and n is the filter size. Equation 3.10 is depicted in Figure 3.8 with many neurons.



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(a) Filter to blur

0	-1	0
-1	0	-1
0	-1	0

(b) Filter to sharpen

-1	0	1
-2	0	2
-1	0	1

(c) Filter to detect edges

Figure 3.6: Different convolution filters can be used in a CNN to achieve preferred results.

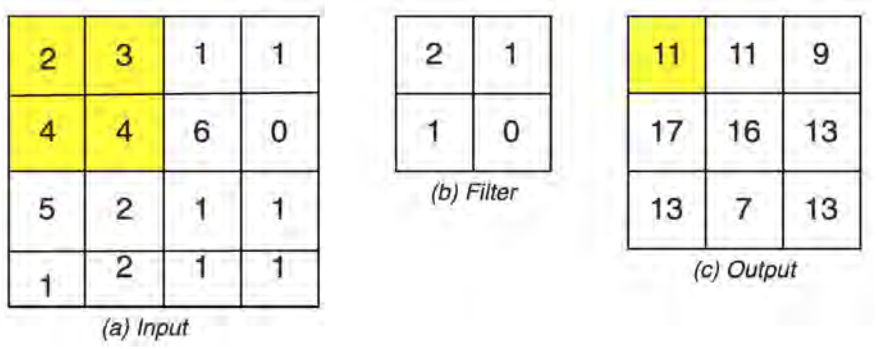


Figure 3.7: The convolutional filter cross-correlates the input array to produce the output array. This filter continues to convolve the whole input array until it is finished.

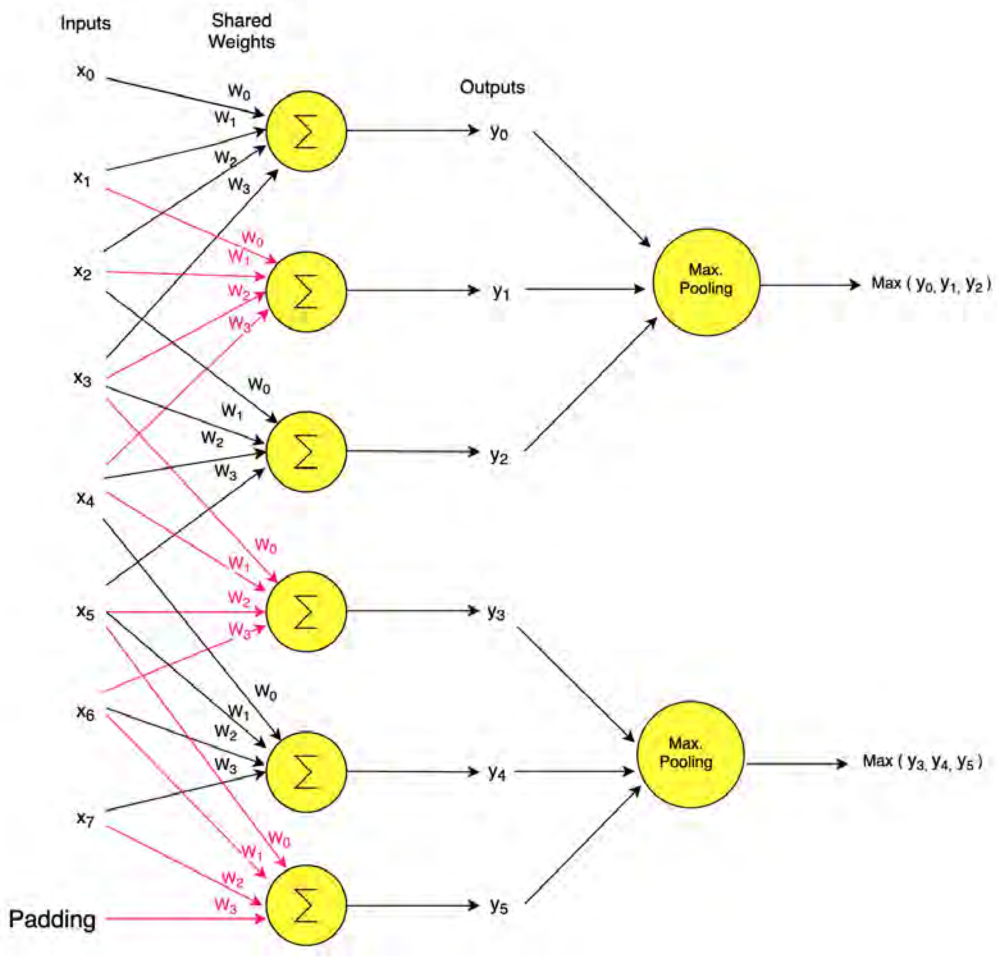
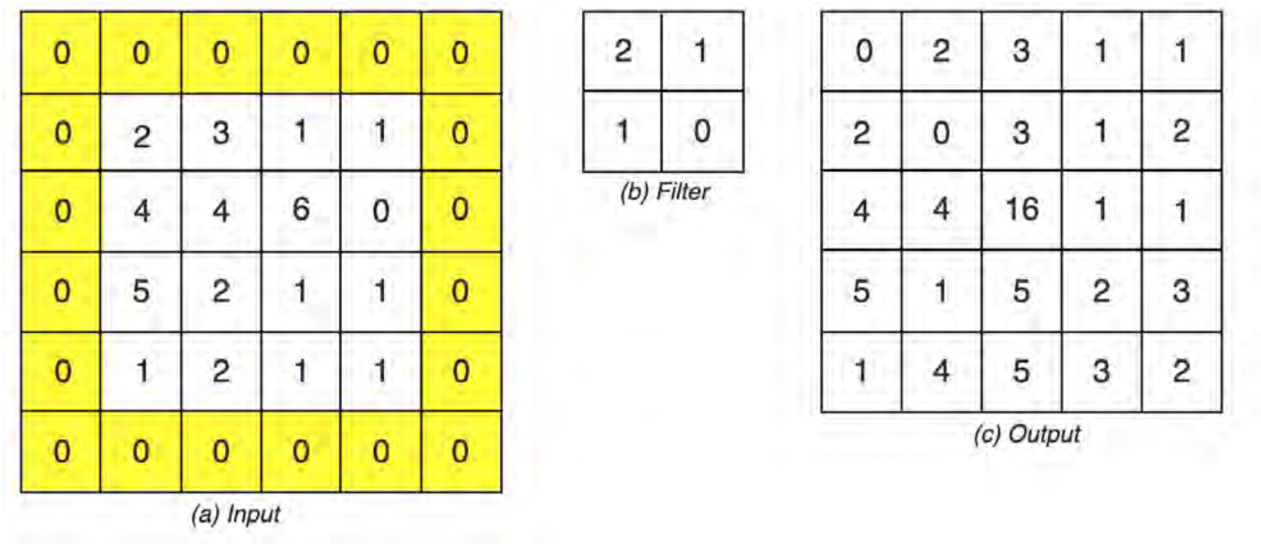


Figure 3.8: CNN neurons and how they share weights.

3.6 Padding

The dimensions of the input and output arrays in Figure 3.7 are not the same. The convolution operation has reduced the dimensions of the output array. The reduction may be a disadvantage when performing a process requiring the input and output arrays to be the same size. This problem is solved by padding or adding zeros to the edge of the input array so that the output array maintains the original input size after convolutions. Figure 3.7 illustrates that applying many filters to an image chop off a certain percentage of the original image. This operation causes a loss of useful information on the boundary of the image in some instances. Padding deals with the problem of border effects as illustrated in Figure 3.9 .



0	0	0	0	0	0
0	2	3	1	1	0
0	4	4	6	0	0
0	5	2	1	1	0
0	1	2	1	1	0
0	0	0	0	0	0

2	1
1	0

0	2	3	1	1
2	0	3	1	2
4	4	16	1	1
5	1	5	2	3
1	4	5	3	2

Figure 3.9: Padding an input image with zeros increases the input size, allowing for the output size to be equal to the original input size.

3.7 Striding

A filter's pixel distance from one convolution to the other is called a stride. The default value of a stride is 1, and from Figure 3.7 a stride involves sliding the filter one unit to the right. This operation reduces redundancy but down-samples the input image. It is used when the aim is to compress the feature map. The stride of 1 reduces the output dimension by one pixel. Moreover, a stride of more than one may be used to reduce the size of the output further, as illustrated in Figure 3.10 .

3.8 Pooling

A more robust way to achieve sub-sampling is by aggregating the sub-region of the input array. This aggregation is a solution to the sensitivity that feature maps have to record the precise location of the features. Pooling layers provide a way to downsample the feature map using average or maximum pixel pooling. It improves

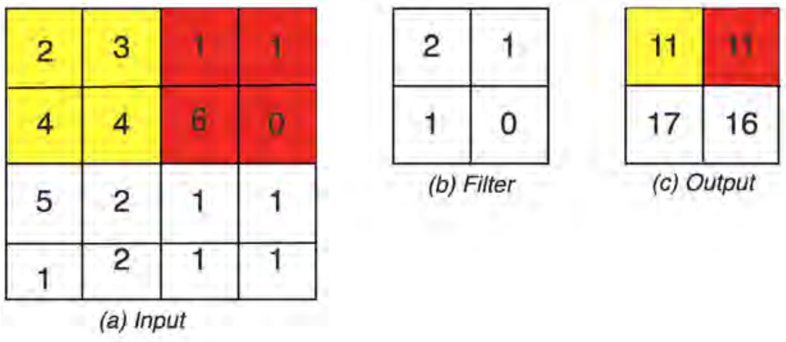


Figure 3.10: A filter is used to convolve an input image using a stride of two to obtain an output half the size of the input image. The cells in red illustrate how the convolution slides from yellow to red, skipping two cells.

computational efficiency as it reduces the size of each feature map. There are two types of pooling, average and maximum pooling. Average pooling calculates the average value in each patch of the feature map as shown in Figure 3.11, and maximum pooling takes the maximum value in each patch of the feature map as shown in Figure 3.12. Average pooling is useful when the image's background is darker than the pixels we are interested in. Maximum pooling adds translation invariance, which is helpful for cases where we care more about what is in the image than where in the image it lies.

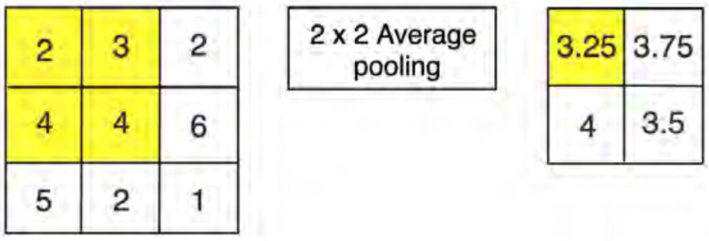


Figure 3.11: An average pooling operation which takes the average of the patch with values (2,3,4,4) and calculates its average to be 3.25.

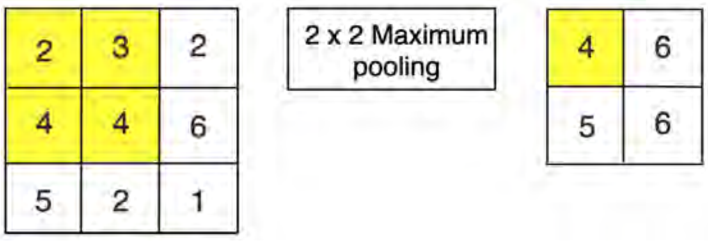


Figure 3.12: A maximum pooling operation which takes the maximum of the patch with values (2,3,4,4) and calculates its maximum to be 4.

In this chapter, we have learned about deep learning algorithms. We have looked at two types of deep learning algorithms that mimic the human brain. These two algorithms are ANNs and CNNs. We examined the basics of ANNs, their origin and how to construct such a network. After that, we investigated how the algorithm learns and how to optimise it to obtain the desired results more efficiently. ANNs have useful properties, although they also have some drawbacks. We then investigated CNNs and discovered that it compensates for the drawbacks of ANNs. The focus then shifted to CNNs, and we highlighted their advantages over ANNs, looking at the properties that make them exceptional. We looked at two deep learning algorithms and found one specialising in recognising, classifying and segmenting images. In Chapter 4, we shall focus on CNNs and segmentation.

Chapter 4

Segmentation using Unet

In this chapter, the Unet image segmentation algorithm is described in detail. Unet is an algorithm that was previously used in biomedical images. We explore how we can use it to automate the extraction of sources from dense radio images.

4.1 Introduction

Image segmentation is a process that partitions a digital image into multiple sets of pixels that share similar characteristics [Tan (2016)]. The aim is to simplify image analysis by subdividing it into meaningful areas. Image segmentation effectively assigns a label to every pixel in an image so that the pixels with the same label have a shared attribute. Typically, it can be used to detect objects and boundaries in images. The output of an image segmentation process consists of segments of at least two different classes that cover the whole image. The pixels in a region have one or more similar characteristics, such as (1) colour, (2) intensity, or (3) texture. Pixels in adjacent regions have different characteristics. There are two types of image segmentation methods. Instance segmentation labels every pixel in an image; the method identifies the "belonging instance" of the object. It identifies each distinct object of interest in an image. For example, each person in an image is segmented as an individual object, as shown in Figure 4.1 (left panel). Semantic segmentation is the second method, whereby each pixel in an image is classified according to a simple decision criterion. An example is when all the people in an image are segmented as one object and the background as another, as shown in Figure 4.1 (right panel). Let us explore applications of semantic segmentation on images.

4.2 The application of semantic segmentation

Semantic segmentation is applied in the automotive industry for autonomous vehicles [Anderson et al. (2014)]. Autonomous driving is a complex robotics task that needs perception, planning and execution. Safety is important in autonomous vehicles; thus, these tasks must be performed with the best possible precision. Semantic segmentation

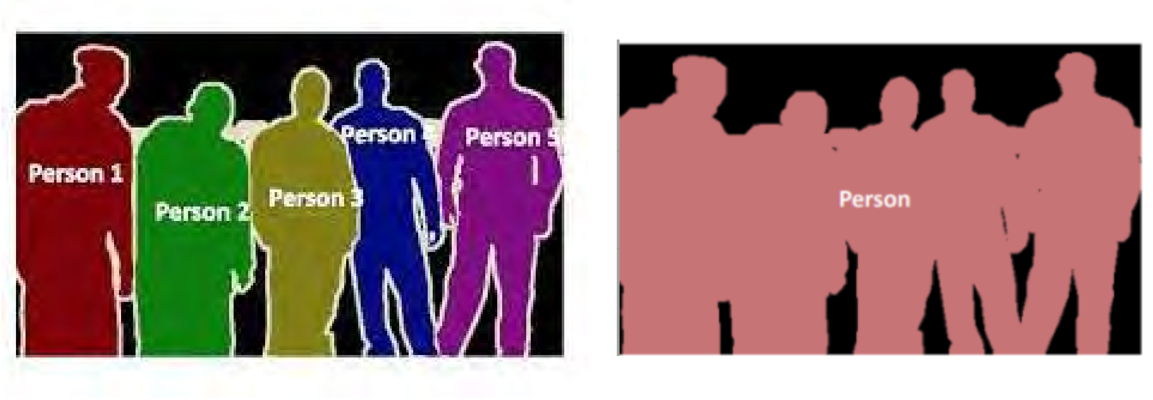


Figure 4.1: The image on the left panel is an instance segmentation method where all the persons are identified distinctly, and on the right panel is an image of a semantic segmentation method where the separation is either a person or background [*The binary notes* (2022)].

helps identify free spaces, lane markings and traffic signs [Anderson et al. (2014)]. Another application is in the medical sector for biomedical image diagnosis. Machines can be programmed to segment medical images like radiologists analyse an image. This process reduces the time required to run diagnostic tests [Novikov et al. (2018)]. Also, in the medical sector, machines can be programmed to segment neuronal structures in electron microscopic stacks [Ronneberger, Fischer, and Brox (2015)] - most of the work in this study is derived from this application. Geo-sensing is another sector where semantic segmentation is applied in monitoring areas of deforestation and urbanisation to classify land cover information. In aerial and satellite photographs, it is used to identify urban areas, agricultural areas, water, etc.. This can be seen as a multi-class semantic segmentation task [Kussul et al. (2017)]. In precision agriculture, semantic segmentation is used to help farmers to trigger weeding actions in real-time. The number of herbicides that are used on the crops is reduced as a result of precision farming [Kussul et al. (2017)].

Semantic segmentation has been helpful in improving and developing the lives of people around the world. Today, we enjoy the best healthcare due to more precise test results. Moreover, people feel safe driving in autonomous cars since the cars can detect and identify road signs and navigate themselves from one place to another with high precision. There are many new possibilities for semantic segmentation. Hence, we would like to develop semantic segmentation for astronomy.

4.3 Segmentation architectures

The applications mentioned above use different architectures to achieve their results. In this section, we shall examine different architectures in semantic segmentation. One popular architecture for performing semantic segmentation is a fully convolutional network (FCN), as shown in Figure 4.2. A series of convolutions and maximum pooling layers are used to compress an image to a smaller size and then predict each pixel's class at that granularity level. After that, it uses sampling and deconvolution layers to take the image to its original size. This basic structure is effective to segment objects from images [Zhang et al. (2017)]. The aim of downsampling the image to the level of granularity is to capture contextual information at that level, while the aim of upsampling the image to its original size is to recover its spatial information. The image size does not matter, and the final image will always be the same size as the original. A skip connection which bypasses one or more layers is used to recover spatial information lost in downsampling. Features from other resolution levels are added up to combine contextual with spatial information. The drawback of FCN is the uneven overlapping of the output of the deconvolution operation since it is not symmetrical and has poor resolution at boundaries, with a loss of information during downsampling.

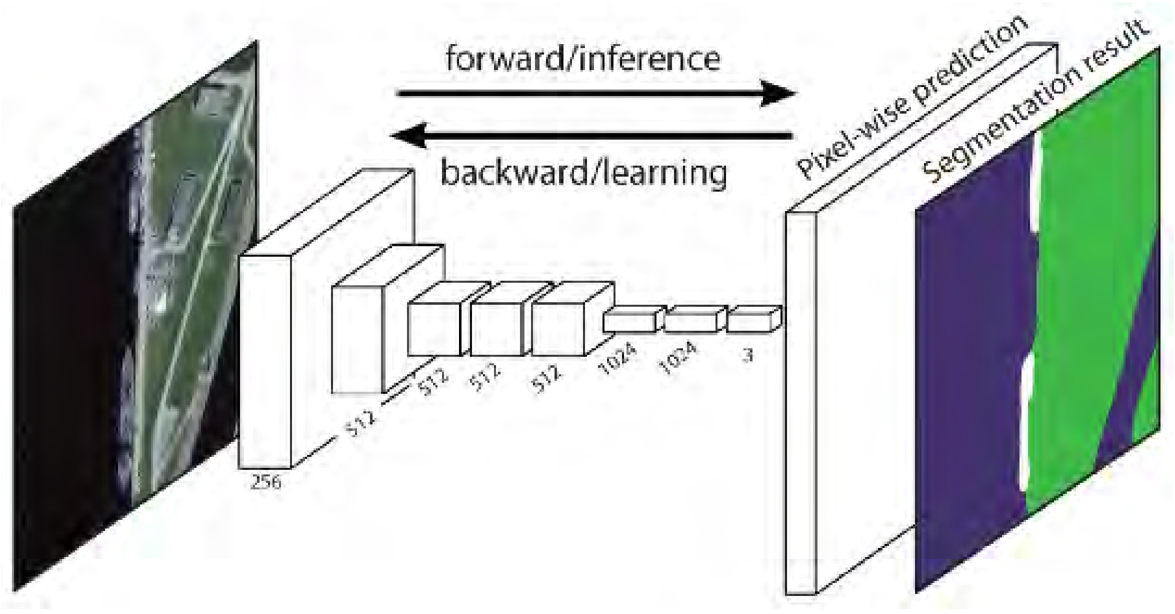


Figure 4.2: A FCN which takes in an input, passes it through a series of convolutions and max pooling layers before it predicts the pixel class to create a segmentation map [*FCN* (2015)].

Several variations on this basic architecture have been proposed. Unet is one of the architectures that build on FCN. It was developed mainly for medical purposes to help detect tumours in lungs and brains [Ronneberger, Fischer, and Brox (2015)]. The architecture of this neural network takes a 'U' shape, as shown in Figure 4.3, hence its name. The left side of the 'U' is a contracting path, while the right side is the expansive path. The contracting path is also known as the encoder, which is responsible for capturing the context of the image. It does this through a stack of convolutional and maximum pooling layers.

The expansive path is symmetric to the contracting path and is known as a decoder. This path is a unique part of the Unet architecture. It enables precise localisation and performs this through a stack of transpose convolutions. This architecture can also be seen as an end-to-end FCN because it contains convolutional layers but does not have dense layers; thus, it can accept an input image of any size. The most useful contribution of Unet comes from the shortcut connections, which recover or compensate for the loss of information we had with FCN. Unet suggests that to improve the loss of information problem, one can add information from every downsampling layer to the corresponding upsampling layer. This downsampling helps it to capture the finest details. We apply two consecutive convolutional layers in the contracting path using an input image file of 572 $\times$ 572. The first convolutional layer generates a 570 $\times$ 570 pixel image with 64 channels. The second convolutional layer generates a 568 $\times$ 568 image with 64 channels. The convolution process increases the image's depth, hence the 64 channels, while padding decreases the size of the image resulting in the decreased size of 570 $\times$ 570 and 568 $\times$ 568. In Figure 4.2, the small red arrows pointing down are the maximum pooling operators, which halve the image's size each time. This process is designed to capture the context of the image through different channels. The process is repeated three more times. When it reaches the bottom-most layer, two more consecutive convolutional layers are still applied, but this time with no maximum pooling. The image at this point has reduced to 28 $\times$ 28 $\times$ 1024.

The expansive path aims to upsize the reduced image to its original size while localising the images' objects. A transpose convolution is applied in this path to expand the size of the images. The small green arrows pointing upwards are the transpose convolutions (up-convolutions). The first transpose convolution upsizes the image from 28 $\times$ 28 $\times$ 1024 to 56 $\times$ 56 $\times$ 512. This image is then concatenated with the corresponding image from the contracting path to make the size 56 $\times$ 56 $\times$ 1024. These two images are mainly concatenated to get a more precise prediction. Two successive convolutions are applied. This process, just like before, is repeated three more times. When the uppermost level of the architecture is reached, the last step is to reshape the image to satisfy the prediction requirements. The reshaping is done by adding a convolutional layer of size 1 $\times$ 1, yielding the segmentation map as the output. The input image would be a radio image of size $m \times m$, and the output would be a segmentation map of size $m \times m$. The main drawback of this architecture is that it is computationally expensive.

Mask R-CNN is another segmentation architecture that combines two networks, faster R-CNN and FCN, as shown in Figure 4.4. Faster R-CNN is an architecture that is used to perform object detection. It has two stages, the first stage being where it forms object-bounding boxes. This process is also known as the region proposal network (RPN). In the second stage, it extracts features for each candidate box using region of Interest pooling (RoIPool) and uses them to classify and perform bounding-box regression. The output of mask R-CNN is the class label, the bounding box offset and the object mask. The mask is obtained by adding the second network, FCN. The total loss is calculated from the three outputs. The advantage is that it has three output products; however, the loss may increase as it accounts for the three products and inherits the drawbacks of FCN.

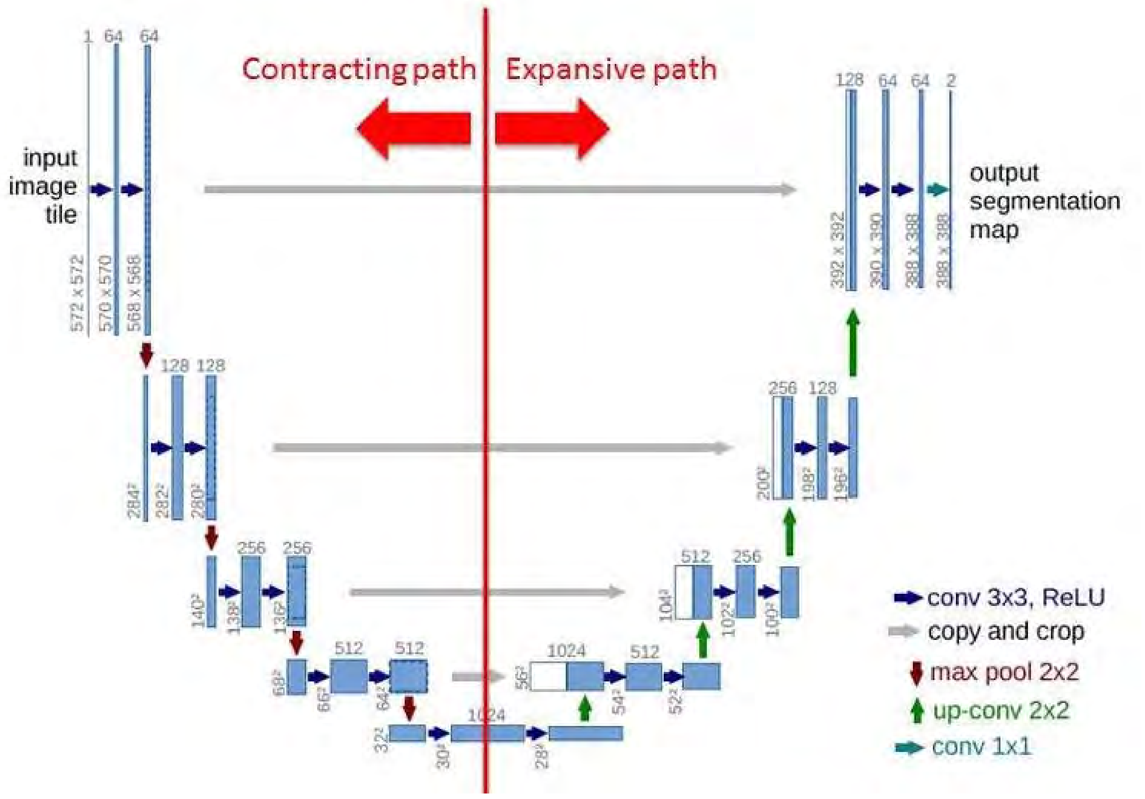


Figure 4.3: A Unet segmentation map showing the contracting and expansive paths that give it its symmetric shape. We show details of how the input image goes through a series of (1) convolutional layers, (2) maximum pooling layers, and (3) deconvolutional layers to get to the output. The dimensions of the arrays and the number of filters applied at each layer are indicated [Ronneberger, Fischer, and Brox (2015)].

4.4 Preferred segmentation architecture

Radio astronomical images contain information about celestial bodies that are not directly accessible; hence, they are important to astronomers. The quality of radio astronomical images is usually low relative to regular real-world images. The quality is mainly due to the way these images are acquired. These images have a low signal-to-noise ratio, making it tricky to use standard segmentation methods normally used on regular real-world images. Radio astronomy structures are seen at different resolutions. Traditional source finders may find it difficult to detect diffuse emission at a particular resolution, or rather they might find it difficult to associate it with the same source. Radio astronomical images are sensitive to different scales depending on the observational data. Therefore structures of a single source can be seen at best at a different resolution. For example, sharp features can be seen clearly at high resolutions, whereas extended features can be picked up better at low resolutions. We evaluate different models, looking at the complexity of the data, the output we want to obtain, and the accuracy of the models. Unet was the architecture chosen for further study due to its ability to (1) handle noisy data, (2) decrease the dimension of an image and then

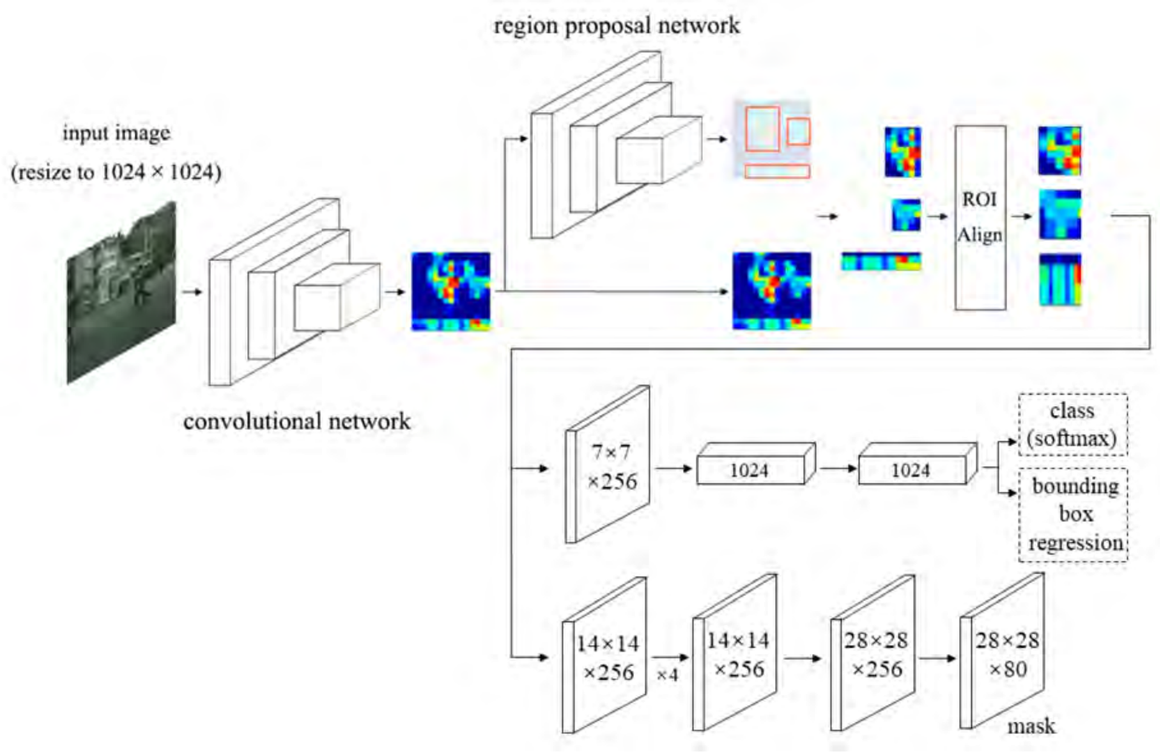


Figure 4.4: The mask R-CNN combines two architectures, faster R-CNN and FCN. These two architectures work together to produce three output products. The class, the bounding-box offset and the segmentation map [Zhang et al. (2020)].

increase it back (this is particularly favourable as astronomical images can have high resolution) and (3) it retains the image's original size at the end of the architecture. (4) Unet can effectively segment images with limited datasets as a starting point. Since we do not have much-labelled data, we can still segment our images due to this advantage. (5) It is more successful than conventional models in pixel-based image segmentation. The latter is particularly important as we want to automate the segmentation process with the best possible precision and accuracy. (6) Unet is also easy to implement, as it has ready-made libraries in Python, making it convenient. On the other hand, Unet has a major drawback because the architecture contains a several layers to downsample and to upsample which scale up the compuatational requirements to train the model. However, this thesis mainly focuses on the study of deep learning methods for radio source segmentation. We are not looking to search for the best deep learning methods in the literature to solve the radio segmentation problem, but rather as a first attempt providing baselines and testing if a deep learning method could even be used to solve the problem.

4.5 Evaluation techniques for segmentation architecture

Different evaluation techniques can be used to evaluate how well a segmentation architecture performs on unseen data. The Dice coefficient technique is often used in cases with a class imbalance. This method calculates the overlap between a *predicted* class and the *target* class as follows:

$$\text{Dice co-efficient}(D) = \frac{2 \cdot (\text{target} \cap \text{prediction})}{\text{target} + \text{prediction}}. \tag{4.1}$$

The objective is to maximise the dice co-efficient D; hence we minimise *(1-D)* where $0 < D < 1$. The second technique that can be used is the pixel-wise softmax with cross-entropy, as shown in Equation 3.6. In this architecture, the labels are represented in a one-hot encoded form. This technique is useful as it can be used to represent the target and evaluate cross-entropy. When the pixels are evaluated, they can belong to one of our target classes. Softmax pixel-wise is applied before applying cross-entropy. A third method is a focal loss, usually used in extreme class imbalance cases. Focal loss is a modification of the standard cross-entropy. Unlike cross-entropy, a focal loss does not penalise the model excessively when the class balance is known beforehand. The fourth technique we consider is "intersection over union" (IoU), which evaluates segmentation architecture performance as follows:

$$\text{IoU} = \frac{\text{target} \cap \text{prediction}}{\text{target} \cup \text{prediction}}. \tag{4.2}$$

The dice coefficient and the "intersection over union" coefficient give a quantitative measure of the overlap illustrated in Figure 4.5. Dice calculates the proportion of the total number of pixels in both images occupied by the area of overlap (multiplied by two because there are two images), whereas "Intersection over Union" calculates what proportion of the union of pixels in both images is occupied by their intersection.

4.6 Source Detection using OpenCV

In an effort to enrich the body of knowledge and simplify the process of programming computers, the Intel [1] team in 1999 bundled programming functions into an open source code in the Internet. This was the original goal of the team when they launched the open source computer vision library (OpenCV [2]). Programmers all over the world have used this library for academic and commercial purposes. Not only did they use the library, but they also helped optimize the library's algorithms. The major changes in the contribution took place in 2009.

[1] www.intel.com

[2] http://opencv.willowgarage.com/wiki/

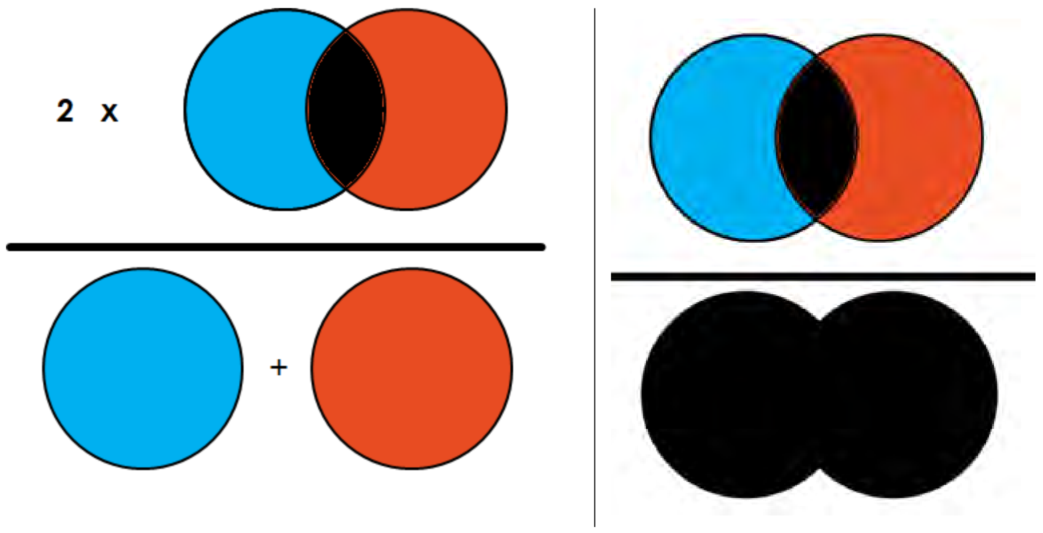


Figure 4.5: DICE vs "Interception over Union" evaluation techniques

4.6.1 Main algorithms and methods

There are common image processing techniques that are used in OpenCV. In Image analysis, there is a concept of image histograms that shows the number of pixels in an image having some given value in a table form. We perform a linear stretch to the range of intensities to enhance the image contrast. The caveat is that there is usually an imbalance in the intensities, where some are used more than others. Operating on the logic that a good image uses all the available intensities, we use a function that equalises the use of all intensities in the image. The distribution of an intensity gives a probability that some pixel has a specific value. This characteristic can detect specific image content by replacing pixel values with probabilities. This process, in turn, produces a probability map that can be used to detect an object through a mean shift algorithm [Comaniciu and Meer (2002)].

The operation of comparing pixel values in the input image with its neighbours and attaching a label on each pixel in the corresponding output image is called morphological operations. We can differentiate between dilated and eroded images. Erosion is when the output pixel value is given the minimum value of all the pixels in the input pixel's neighbourhood. Dilation, on the other hand, is when the output pixel value is given the maximum value of all the pixels in the neighbourhood of the input pixel. This filter is usually applied to binary images where we assume the object is white and the background is black. This operation is also sensitive to particular shapes in the input image. An erosion filter reduces the object size, and a dilatation increases the object size. These filters can also be used to detect lines and corners [Soille, Beucher, and Rivest (1993)].

We can also study the spatial frequencies of intensities. In this domain, intensities are grouped from lowest occurrence to highest frequency. This process is done by Fourier transform. Here, a filter either amplifies or removes certain bands of frequencies. A blur function calculates the average value in a rectangle neighbourhood to smoothen the image. This rectangle in the form of a mask is called a kernel which then can compute convolutions as described in section 3.5. The shape of the kernel can take other forms, such as a Gaussian. An image can be too small to represent high frequencies. Therefore we need to filter the high frequencies out. This process down samples the image; unfortunately, even if we upsample it, it will not restore the original image. However, we have created image pyramids which are structures that can be used for specific image analyses. For instance, we can perform object detection on the down-sampled image, followed by object detection on the up-sampled image and then use a median blur function to remove outliers. This filter replaces pixel values by the median of all considered pixels in the neighbourhood. This operation preserves the sharpness of edges, but at the same time, it distorts texture in uniform regions. In the above case, we have attenuated the high frequencies, but sometimes we want to amplify them. We will thus use directional or high pass linear filters where the image intensity gradient is calculated. These would be first or second-order derivatives. Laplacian filters can detect the change of sign in two neighbourhood pixels, then threshold the gradient of around zero, thus obtaining the edge map image. Another edge detector filter that uses two thresholds is Canny [Canny (1986)]. One threshold is low, and the other is high, yielding a low and high threshold edge map. We can then combine these two edge maps and yield good quality contours. The Hough filter is good for detecting lines [Galambos, Kittler, and Matas (2001)]. The input image is an edge pixel map usually applied after a Canny filter. An image structure is parameterised into two parameters. A 2-dimensional accumulator is constructed where each entry is a line. We search for all lines that pass through this point, incriminating the accumulator. If the line passes many points, the accumulator will be high, and thus this is detected as a line.

Feature point detectors help identify points of interest. A common point of interest is a corner. A corner detector [Harris and Stephens (1988)] identifies a corner as a junction of two edges as a 2-dimensional feature. Harris detector uses a corner detector definition to check the rate of intensity change around some point. It computes the intensity change for some direction; if the average change is high, we have a corner. FAST (Features from Accelerated Segment Test) algorithm [Rosten and Drummond (2006)] is a faster corner detector. It evaluates pixels in a circular form surrounding a point of interest. If all the surrounding pixels have a significantly different intensity than the point of interest, then we declare it a corner. SURF (Speeded Up Robust Features) algorithm is a key point detector [Bay et al. (2008)]. At different resolutions of the image, this filter calculates the derivatives. At some image resolution, the filter response will reach its maximum for some image points. Comparing this maximum with the minimum input value, if the maximum is at least higher, a key point is declared.

In this chapter, different segmentation architectures were discussed. Unet seemed to have the most suitable features for radio astronomical images; therefore, we concentrate on this architecture for further study.

Chapter 5

Astronomical radio image segmentation and object detection using Unet

In this chapter, we apply the Unet segmentation technique to radio astronomical images. As explained in Chapter 1, the aim is to find an automated process to segment sources from radio images as accurately as possible. This chapter describes all the processes to perform segmentation on radio images.

5.1 Data

The input data we will present to Unet consists of radio astronomical images. We make use of two datasets from different observatories. The first dataset of images was made using two mosaic radio images from the Australia Telescope Low-brightness Survey (ATLBS), observed with the Australian Telescope Compact Array (ATCA) [Thorat et al. (2013)]. These two high-resolution mosaics cover approximately 8 squared degrees of The Southern sky, and their radio source counts are in the flux density range estimate of $0.4 - 8.7$ mJy. Figures 5.1 and 5.2 depict the two mosaics. The second dataset is sourced from the MeerKAT telescope, which will be described in greater detail later. The reasoning behind using two datasets is that the ATCA data are hand-labelled and ready to use, while the second dataset was intended as the objective of this study.

5.2 Data pre-processing

Modern interferometric radio telescopes have high sensitivity and a large field of view; thus, images produced by them contain many sources. Individual sources may appear larger or smaller depending on their size and distance from us. This work aims to segment the individual sources using Unet. The two ATLBS mosaics have a catalogue of

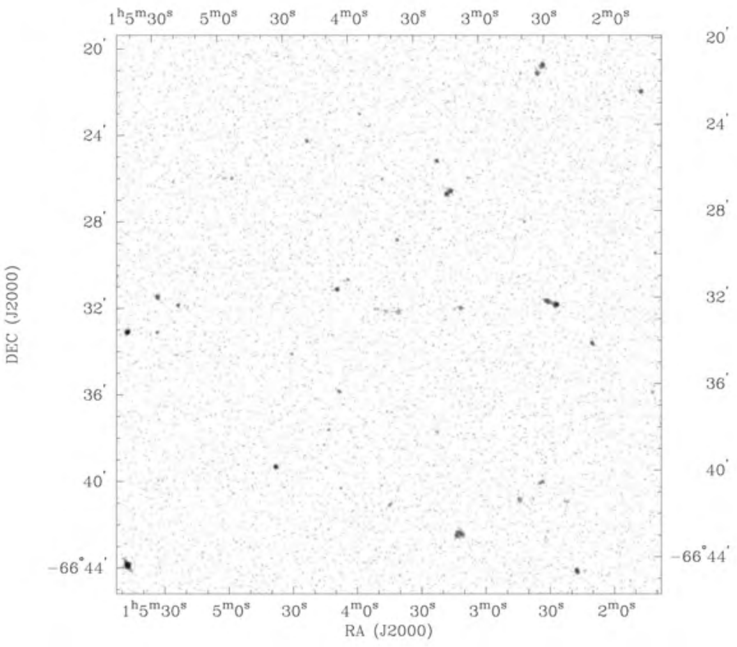


Figure 5.1: A grayscale rendering of a mosaic created by ATCA representing a region of the sky. [Thorat et al. (2013)].

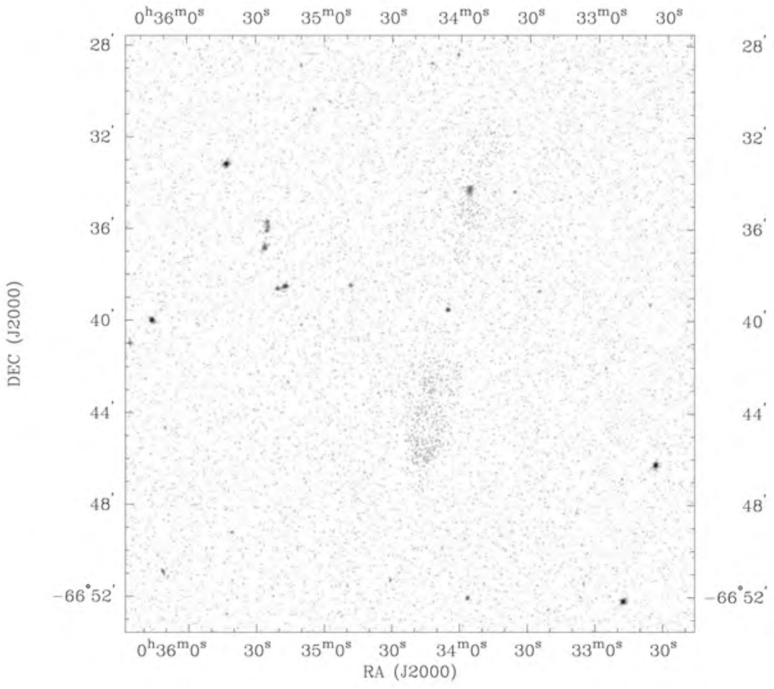


Figure 5.2: A grayscale rendering of a mosaic created by ATCA representing a region of the sky. [Thorat et al. (2013)].

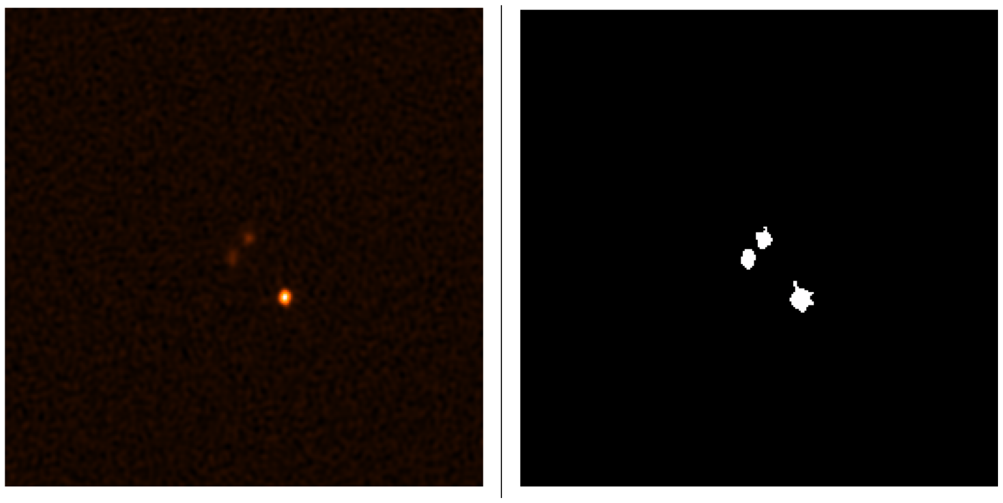


Figure 5.3: A cutout image (left panel) and its corresponding mask image (right panel) from ATBLS survey

known sources with corresponding ascension and declination values. To test the concept of Unet on radio images, we first test the model on this catalogue. The mosaics cover a large area of the sky, therefore capturing many astronomical objects in one image. For a granular analysis of these images, we create smaller images by cutting out images of size 256×256. These images were created by isolating the individual sources using their ascension and declination values and then adding background noise to them to make the images to be of size 256×256. We derived 119 images of size 256×256 using the known extended sources from the catalogue. The cut-out images that were created from Figures 5.1 and 5.2 make up the image dataset of 119. Figure 5.3 is an example of one of the cutout images (left panel) and its corresponding mask image (right panel) made from pyBDSF.

5.2.1 Creating a corresponding labelled dataset

In object detection, to train a model to predict objects, we need to have a dataset of the images together with a corresponding labelled dataset. To create a labelled dataset, we use pyBDSF to produce the corresponding images that mask the sources and background. In pyBDSF, to create the labelled dataset, we set the parameter, pixel threshold, to be 3 and the island threshold parameter to be 6 RMS noise. Pixel values on the image below these thresholds will be classed as belonging to the background class, and those above the thresholds belonging to the foreground or, instead, the source class. pyBDSF produces corresponding n mask images that will be used as the labelled dataset. We convert the images and mask images from FITS format to png format.

5.2.2 Training and Testing split

We aim to train the model and later test how it has learned from unseen data. We do not want to over-learn the model and encounter problems when the model is deployed in production. So we need to test how well it generalizes on unseen data. Train-Test split is a technique that is used to evaluate the performance of a machine learning model. This process requires splitting the dataset into two subsets. The first subset is the training dataset, a set of data that will be used to learn features of the data and fit a model. The second subset is used to create an unbiased evaluation of the model fitted to the training dataset. There are different splitting ratios that one can use. The common ones are 70:30 and 80:20 ratios of training and testing, respectively. We, therefore, split the n images and their corresponding masks into a training and testing set. We opt for a ratio of $70 : 30$.

5.2.3 Augmentation

One of the caveats of deep learning techniques, as mentioned in Section 3.1, is that they require more data to perform compared to traditional methods. However, in this work, we did not have enough labeled data to train and test the model. To augment the labeld data, the augmentation generator from Keras [1] is used. Keras is a software library that provides a Python interface for ANNs. This generator multiplies data by geometric manipulation of images. The different manipulation used are: (1) the rotation of a single image using four different angles; 50, 150, 90 degrees clockwise and 90 degrees counterclockwise, (2) translation of the image across the $X-$ axis and/or $Y-$ axis with an offset of $5, 20,$ and 40 percent of the image size respectively. (3) flipping of the images horizontally, adding different geometric manipulations contributes to the accuracy of the model up to a certain point. Data was increased by n frames and mask frames at $n \times 2$ in each dataset. Figure 5.4 shows some of the augmented images with their corresponding mask images using the ATBLS data.

The images on the left of Figure 5.5 were produced by MeerKAT. The image covers approximately 10.5 degrees squared of The Southern sky with an RMS sensitivity of 11.04×10^{-9} μJy/beam. To create the second dataset, we performed steps to create a corresponding mask image depicted on the right panel of Figure 5.5. The image size is greater than the FoV of the MeerKAT telescope to include bright sources in the sidelobes of the primary beam response of the antenna. The sidelobes have a lesser density of sources than the primary beam's main (central) lobe, and the sources in the sidelobes are often affected by artefacts. Therefore, we concentrate on the primary beam's main lobe, which corresponds to the central portion of the image. Our focus is on the parts of the image that contain astronomical objects. We made a cutout from the centre of the image of 4000×4000 pixels, covering approximately 1.96 degrees squared of the sky. Figure 5.6 illustrates the 4000×4000 pixels image and Figure 5.7 is the corresponding mask. These two images have been cutout to create smaller images. Figure 5.8 shows the cutout images and their corresponding mask image. The cutouts were augmented using the steps in the earlier section.

[1]https://keras.io/

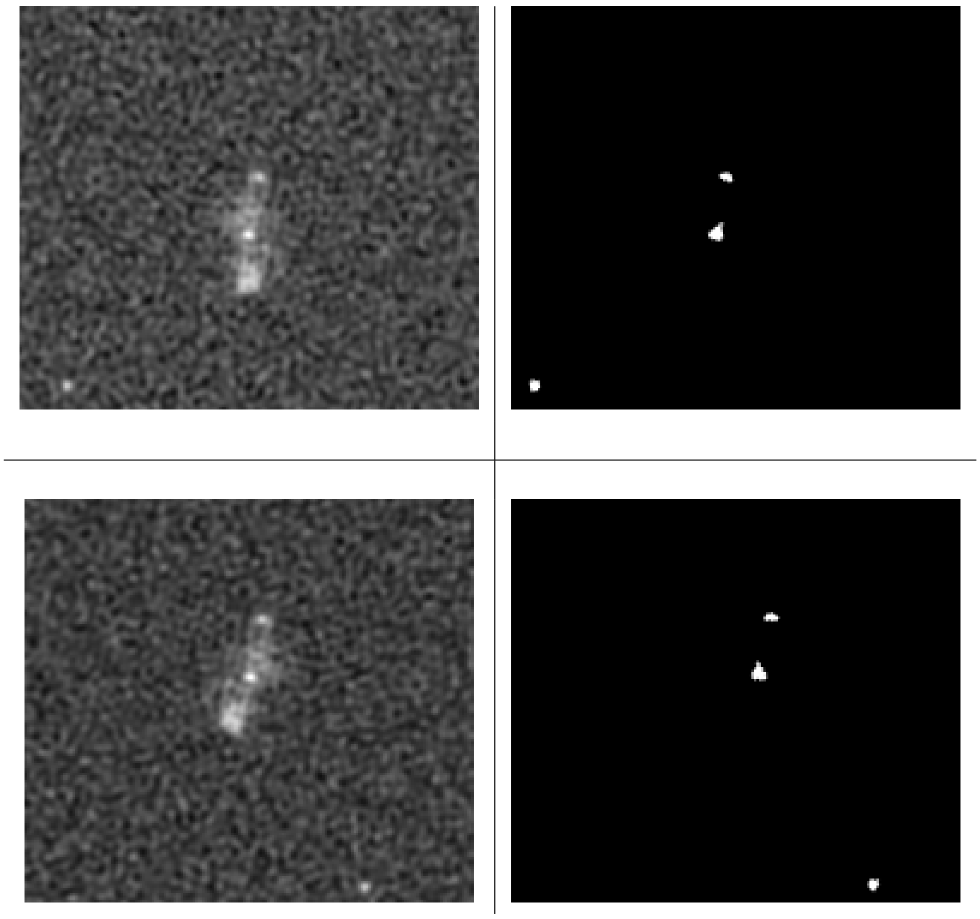


Figure 5.4: Cutout image (left panel) and its corresponding mask (right panel) from the dataset that has been augmented.

5.3 Segmentation using Unet

Unet, as explained in Chapter 4, labels images pixel by pixel to investigate objects in the image more satisfactorily. We have prepared radio images and want to investigate the radio sources in the images. The radio images are read as an array of pixel values of 256 $\times$ 256 (any other preferred size could be used). The pixel values range in brightness or grey scale from 0 to 255. The mask is consequently read similarly; however, the mask's pixel values are either 0's or 1's. A pixel of 0 is associated with the background label, and 1 is associated with a source label. This process constitutes a binary classification of pixels. The purpose of Unet is to take the image array of varying pixel values and produce a corresponding mask array of 0's and 1's. This process applies a series of mathematical operations to the image array to derive the desired mask array. These operations have been explained in detail in Chapter 3. The operations are readily available on Keras. We construct a Unet neural network consisting of a series of layers. Unet is a symmetric model, so deciding to reduce the image shape from 256 $\times$ 256 to 16 $\times$ 16 implies that we require 23 convolutional layers, 4 maximum pooling, 4 upsampling layers, 4 concatenating layers and 2 dropout layers. The hyperparameters used with these layers are detailed in Table 5.1. Table

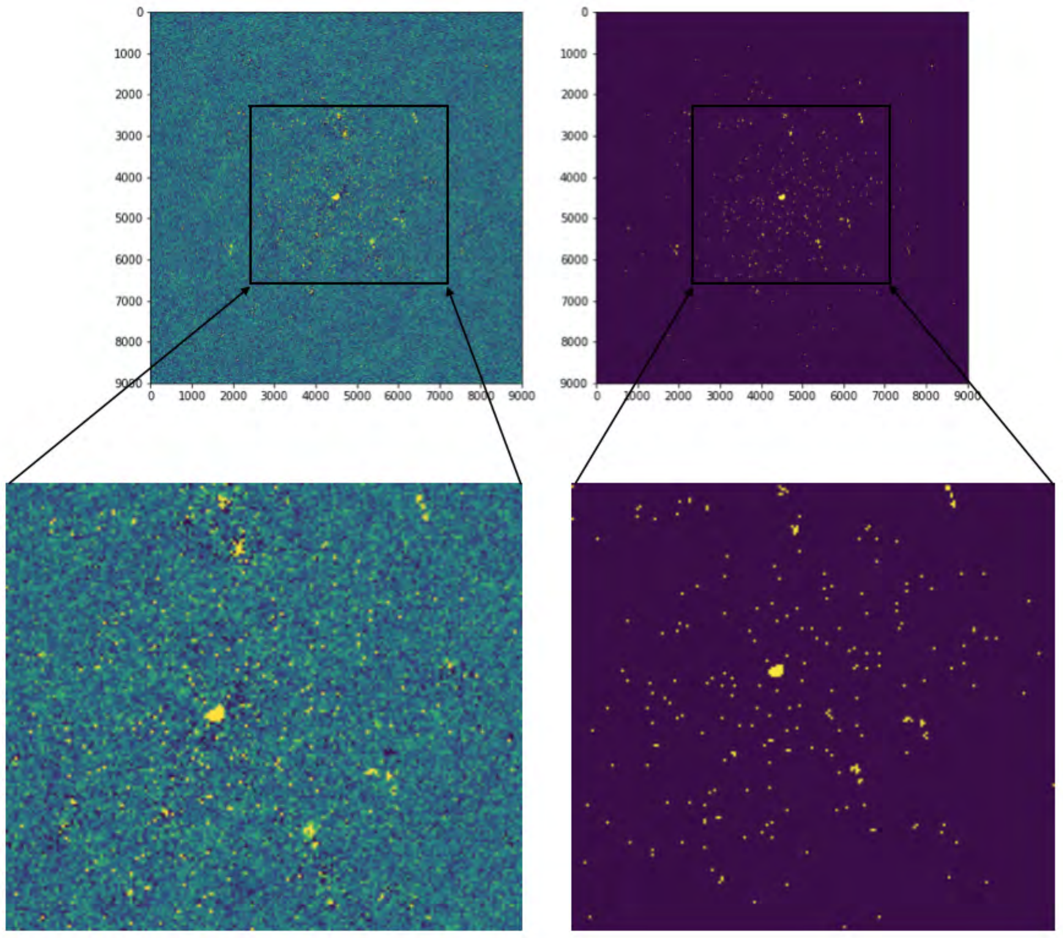


Figure 5.5: A 9000 × 9000 pixel image produced by MeerKAT and its corresponding mask made by pyBDSF with a subset image of 4000 × 4000 pixels and its corresponding mask.

5.2 illustrates how the layers have been put together to create a suitable Unet model for radio images. The training set was used to teach the constructed network how to create mask images. Different sets of layers and hyperparameters were tested. The layers we selected, combined with the hyperparameters from Table 5.1, yielded the best training accuracy of 99.8%.

5.4 Results and evaluation

We used the Unet model to train MeerKAT images, and it performed well, yielding a training accuracy of 99.8%. To test how well the model performs, we use data that

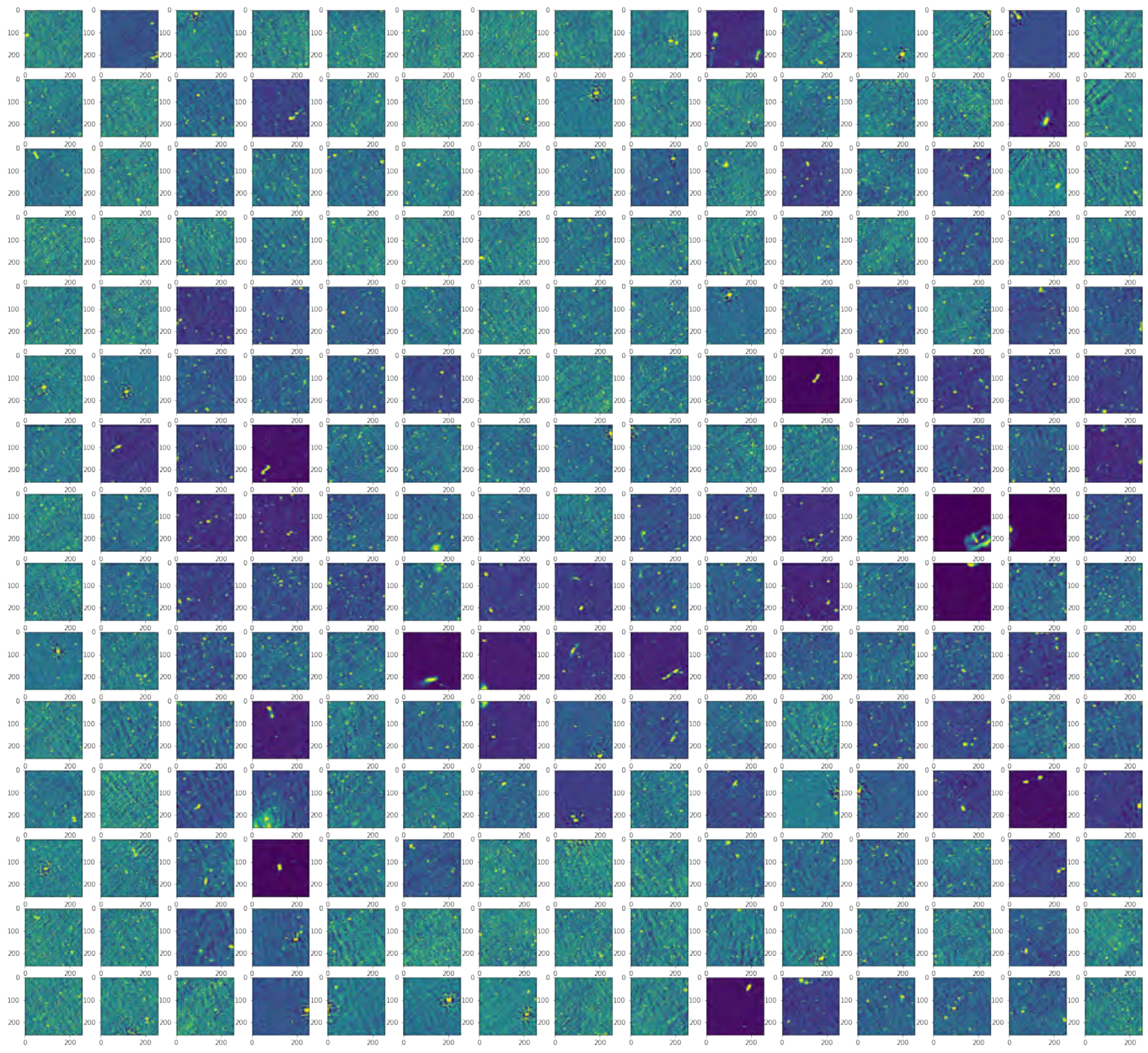


Figure 5.6: The image of size 4000 × 4000 which has been used to produce smaller cut-out images of size 256 × 256.

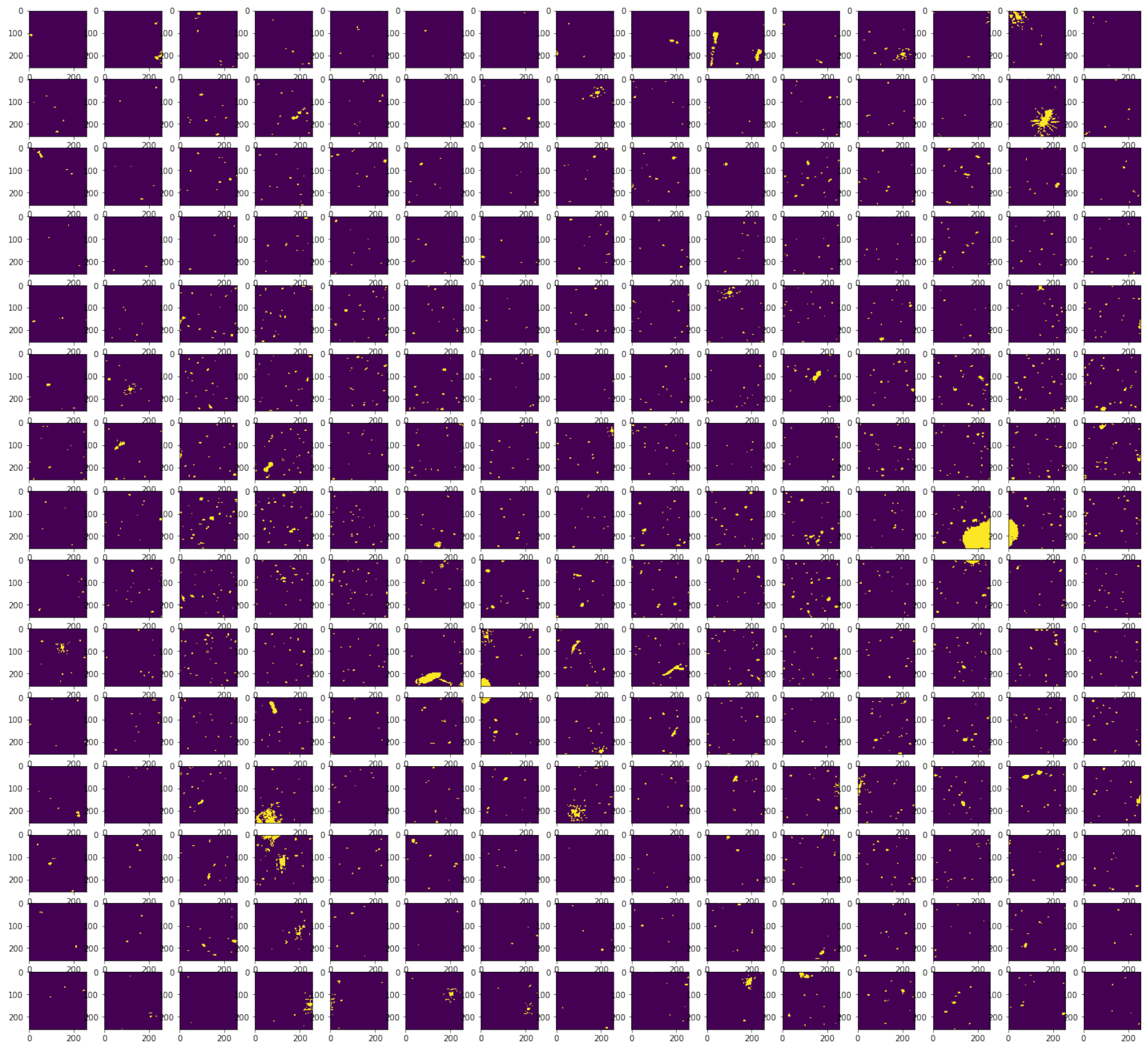


Figure 5.7: The mask image of size 4000 × 4000 which has been used to produce smaller cut-out of size 256 × 256.

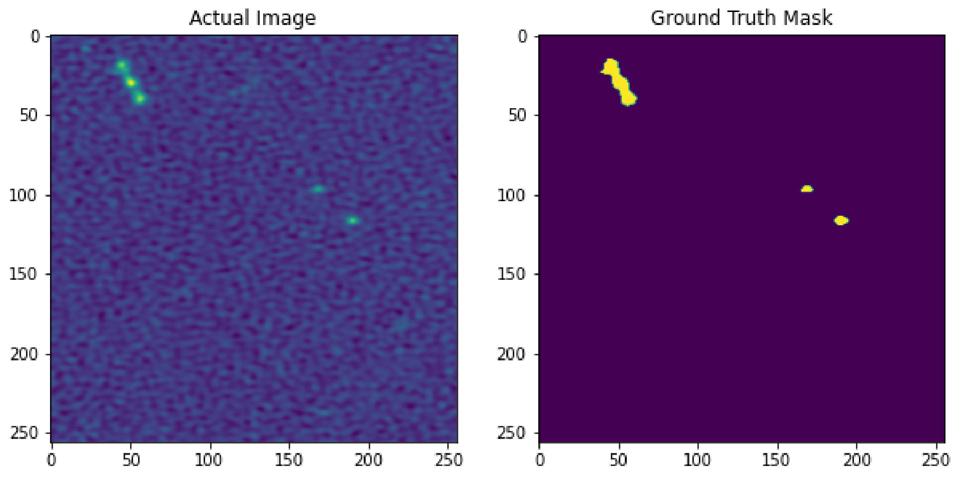
Hyperparameters	Value
Epochs	5
Steps per epoch	1000
Filter size	64
Kernel	3
Activation function	relu (Sigmoid)
Padding	Same
Optimiser	Adam
Learning rate	1e-4
Loss	Binary cross-entropy
Metrics	Accuracy
Kernel Initialiser	He normal

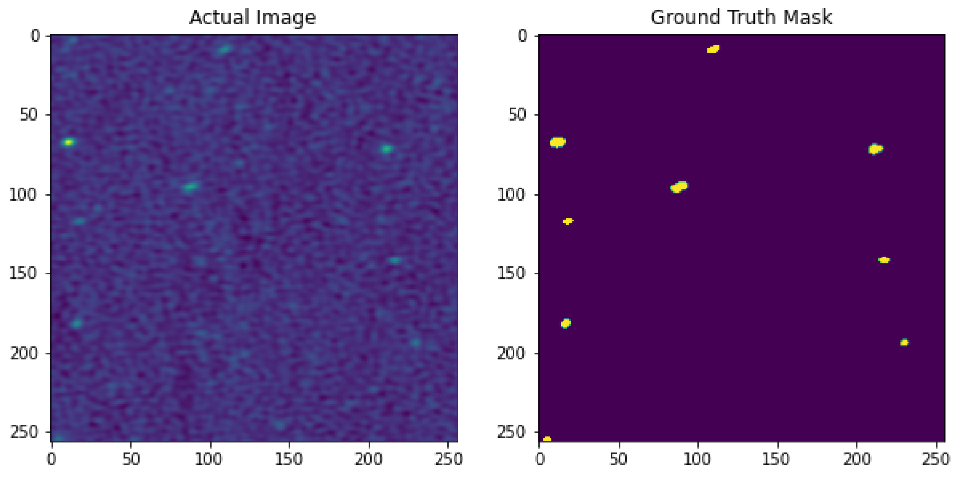
Table 5.1: The hyperparameters that produced the desired results.

it has not seen before. Before the training, we reserved data that will be used for testing how well the model will perform. Figure 5.9 shows examples of images from the testing set, how the model performs in segmenting the images, and an image of ground-truth predictions from pyBDSF. Visually, there seems to be a significant similarity between the prediction mask images and the ground-truth mask images. However, the prediction mask seems to detect fainter emissions as part of the source when the ground-truth image indicates it as noise. To evaluate and quantify how Unet has performed on the unseen radio images, we compare the overlap of the ground-truth masks on the masks produced by Unet. A perfectly predicted mask would overlap the ground truth without residuals. Figure 5.10 illustrates an overlay of the predicted images with the ground-truth visually, and Table 5.3 shows the quantitative performance of the model. The images in Figure 5.10 indicate that the two images are not perfectly overlapping, but there is spillover.

5.4.1 Source detection using OpenCV

OpenCV is used to detect all sources in the image, and a border is drawn around each detected source. OpenCV's inputs are the images and their corresponding masks. With a given threshold, the algorithm finds edges by defining join lines on all points along the image boundary that have the same intensity. When the source is found given the threshold, a bounding box function is then used to create an approximate rectangle of drops around the source, and the process repeats in another area of the image. Figure 5.11 shows the output of the results of the OpenCV algorithm which clearly shows that all sources segmented in the mask are detected in the image.





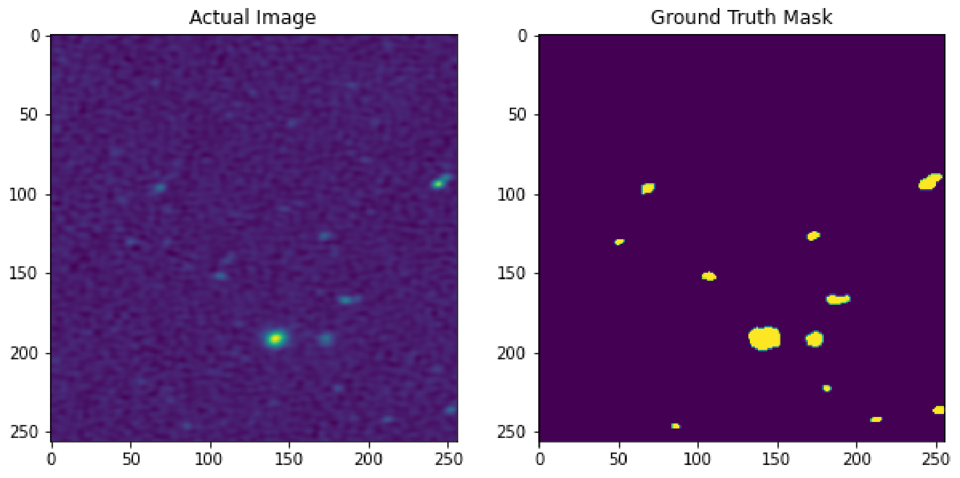
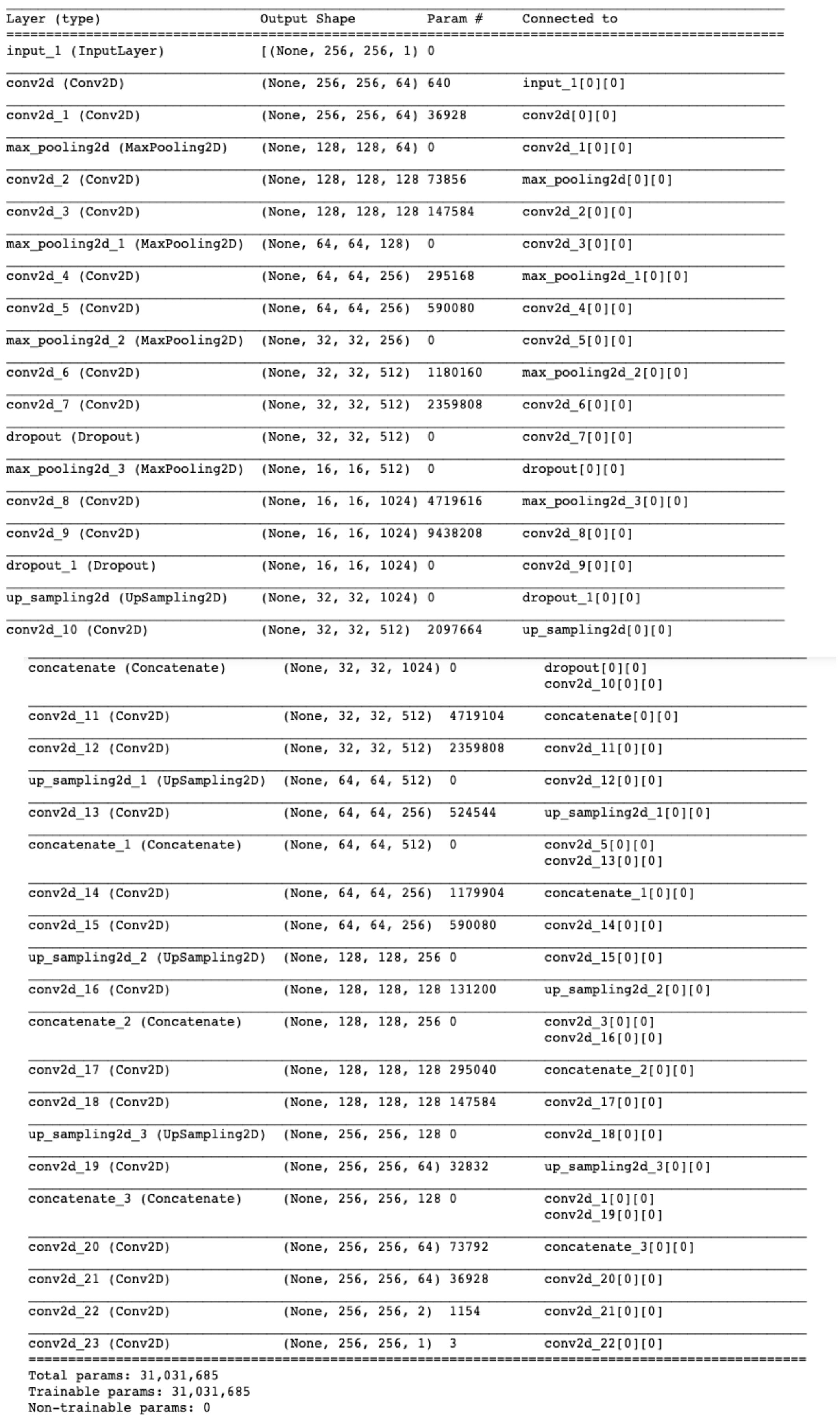


Figure 5.8: Cutout image (left plane) and its corresponding mask (right plane) from MeerKAT dataset.



```
____________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
============================================================================================
input_1 (InputLayer)            [(None, 256, 256, 1) 0
____________________________________________________________________________________________
conv2d (Conv2D)                 (None, 256, 256, 64) 640         input_1[0][0]
____________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 256, 256, 64) 36928       conv2d[0][0]
____________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 128, 128, 64) 0           conv2d_1[0][0]
____________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 128, 128, 128 73856       max_pooling2d[0][0]
____________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 128, 128, 128 147584      conv2d_2[0][0]
____________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 128)  0           conv2d_3[0][0]
____________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 64, 64, 256)  295168      max_pooling2d_1[0][0]
____________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 64, 64, 256)  590080      conv2d_4[0][0]
____________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 32, 32, 256)  0           conv2d_5[0][0]
____________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 32, 32, 512)  1180160     max_pooling2d_2[0][0]
____________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 32, 32, 512)  2359808     conv2d_6[0][0]
____________________________________________________________________________________________
dropout (Dropout)               (None, 32, 32, 512)  0           conv2d_7[0][0]
____________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 16, 16, 512)  0           dropout[0][0]
____________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 16, 16, 1024) 4719616     max_pooling2d_3[0][0]
____________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 16, 16, 1024) 9438208     conv2d_8[0][0]
____________________________________________________________________________________________
dropout_1 (Dropout)             (None, 16, 16, 1024) 0           conv2d_9[0][0]
____________________________________________________________________________________________
up_sampling2d (UpSampling2D)    (None, 32, 32, 1024) 0           dropout_1[0][0]
____________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 32, 32, 512)  2097664     up_sampling2d[0][0]
____________________________________________________________________________________________
concatenate (Concatenate)       (None, 32, 32, 1024) 0           dropout[0][0]
                                                                 conv2d_10[0][0]
____________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 32, 32, 512)  4719104     concatenate[0][0]
____________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 32, 32, 512)  2359808     conv2d_11[0][0]
____________________________________________________________________________________________
up_sampling2d_1 (UpSampling2D)  (None, 64, 64, 512)  0           conv2d_12[0][0]
____________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 64, 64, 256)  524544      up_sampling2d_1[0][0]
____________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 64, 64, 512)  0           conv2d_5[0][0]
                                                                 conv2d_13[0][0]
____________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 64, 64, 256)  1179904     concatenate_1[0][0]
____________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 64, 64, 256)  590080      conv2d_14[0][0]
____________________________________________________________________________________________
up_sampling2d_2 (UpSampling2D)  (None, 128, 128, 256 0           conv2d_15[0][0]
____________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 128, 128, 128 131200      up_sampling2d_2[0][0]
____________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 128, 128, 256 0           conv2d_3[0][0]
                                                                 conv2d_16[0][0]
____________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 128, 128, 128 295040      concatenate_2[0][0]
____________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 128, 128, 128 147584      conv2d_17[0][0]
____________________________________________________________________________________________
up_sampling2d_3 (UpSampling2D)  (None, 256, 256, 128 0           conv2d_18[0][0]
____________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 256, 256, 64) 32832       up_sampling2d_3[0][0]
____________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 256, 256, 128 0           conv2d_1[0][0]
                                                                 conv2d_19[0][0]
____________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 256, 256, 64) 73792       concatenate_3[0][0]
____________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 256, 256, 64) 36928       conv2d_20[0][0]
____________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 256, 256, 2)  1154        conv2d_21[0][0]
____________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 256, 256, 1)  3           conv2d_22[0][0]
============================================================================================
Total params: 31,031,685
Trainable params: 31,031,685
Non-trainable params: 0
```

Table 5.2: Unet neural network

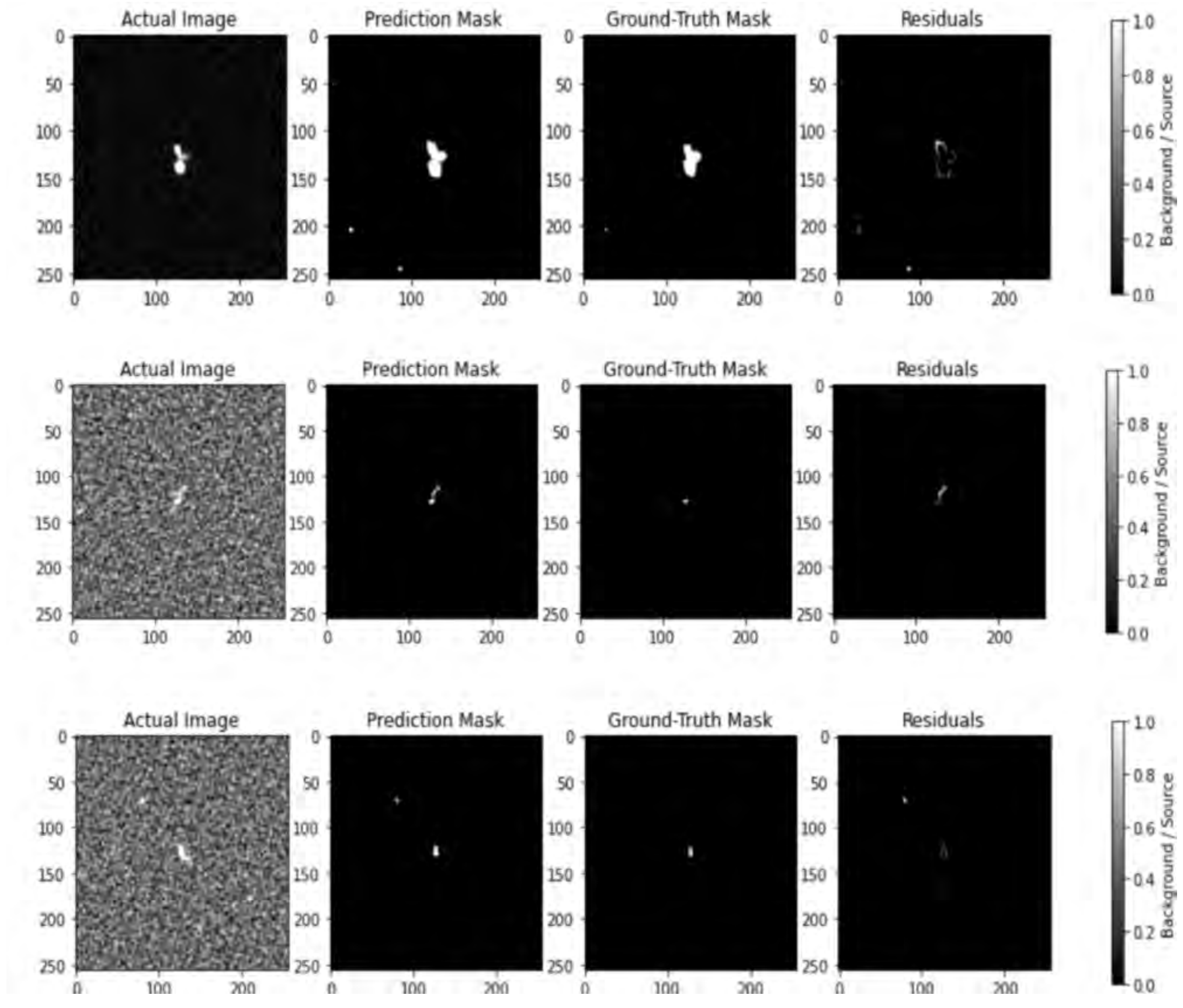


Figure 5.9: Example of actual images, how Unet has predicted the actual images versus how the ground-truth masks the image. The last column illustrates the residual when subtracting the prediction mask from the ground-truth masks.

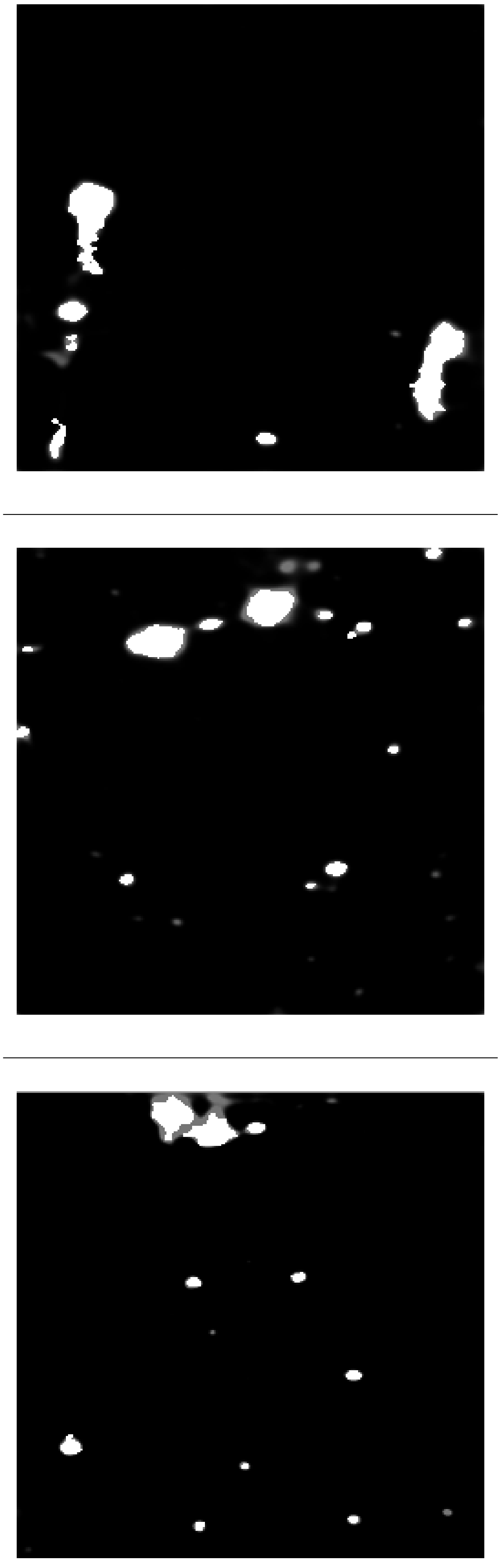


Figure 5.10: Overlap between ground-truth (White) and predicted mask images (grey).

Metric	Accuracy
DICE	0.998264
IoU	0.99826026

Table 5.3: An evaluation table showing the accuracy rates from the two evaluation techniques.

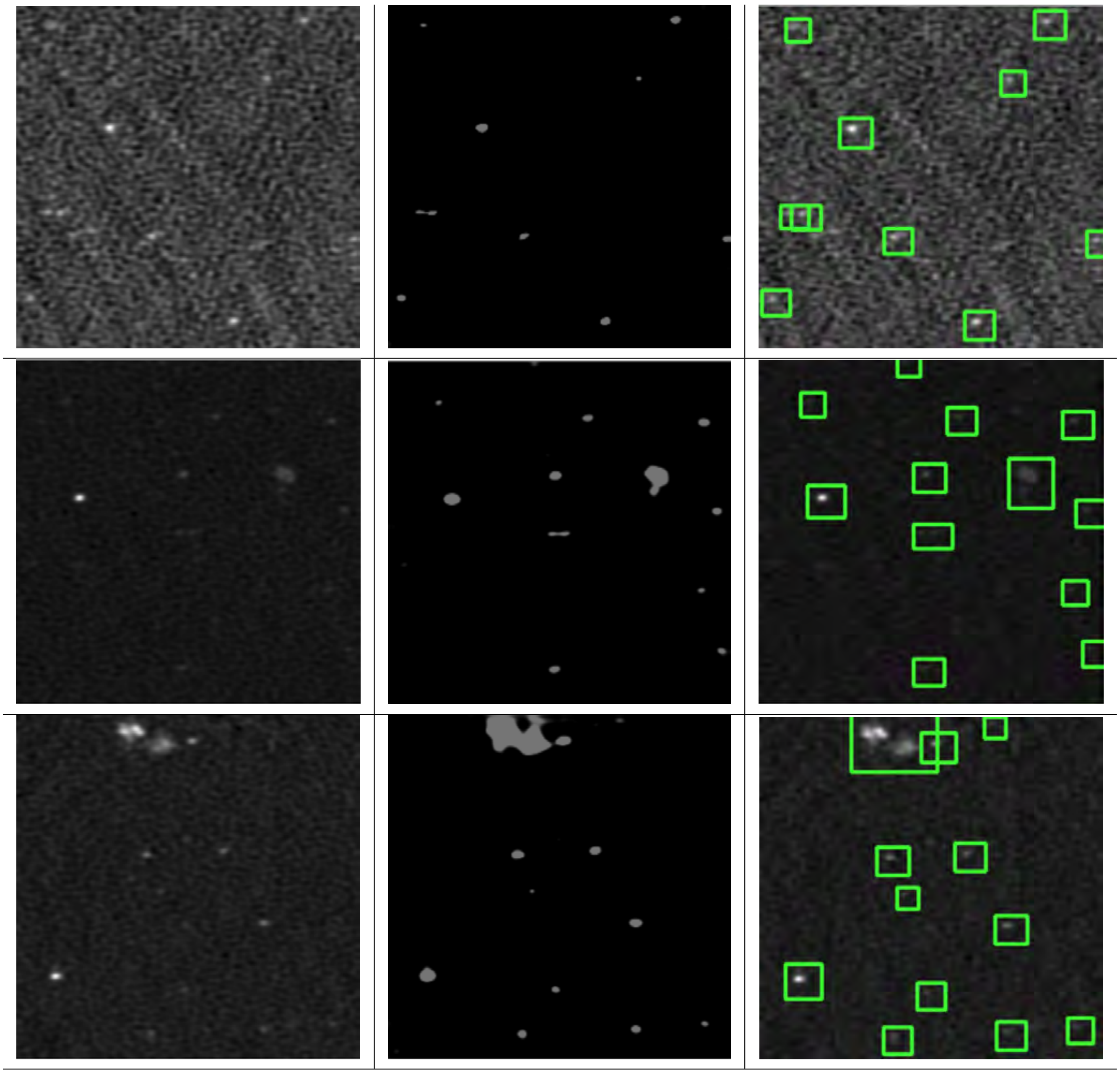


Figure 5.11: Image vs Predicted mask image using Unet vs The detected blobs using openCV

5.5 Aggregation of cutouts to create a big image

The 256 × 256 pixel cutout images were created so that we can train the Unet model and analyze the image in its granularity. We have achieved that goal by mapping out and segmenting the sources in the images using Unet. Ultimately we put the cutouts together and recreated the big image. Figure 5.12 illustrates 16 256 × 256 pixel cutout images that have been concatenated together to form a big image of size 1024 × 1024 pixel image. Figure 5.13 is the corresponding 16 256 × 256 pixel cutout mask image produced by Unet that has been concatenated together to form a big image of size 1024 × 1024 pixel image. Figure 5.14 is the concatenated image run through the OpenCV source detection algorithm. It shows all the detected blobs and has drawn a border around them.

5.6 Unet vs pyBDSF

5.6.1 Comparing parameters

The parameters of pyBDSF to process the images are given in Table 5.4: The frequency used is the central frequency, and the extent of the island used for fitting is three sigmas above the mean. Six sigma above the mean was used as the threshold to get the island peak. Table 5.1 shows the hyperparameters of Unet. Since these algorithms use different approaches, it is quite difficult to compare the hyperparameters of the two models; therefore, we give a summary of the parameters for each. We had to experiment with different values to obtain the optimal pixel and island thresholds for pyBDSF. We selected a pixel threshold of 3 and an island threshold of 6.

Parameter	Value
Frequency(Hz)	1.28E+09,
Threshold island	3
Threshold pixel	6

Table 5.4: The different parameters that were applied to process images in pyBDSF.

5.6.2 Performance accuracy

In testing out if we can use Unet, we first tried it on the ATBLS images and obtained an accuracy of 93%. This process means that on the testing dataset, the model managed to get 93% of the segmentation correctly compared to the ground truth. This accuracy level was adequate for us to go further and test it on images with sharp features produced by MeerKAT. We directly transferred it to the MeerKAT dataset on the Unet model, and it did not perform as expected. The initial performance was that the model could segment only 30% of the test data. So more tweaking was needed to adjust the performance. The hyperparameters shown in Table 5.1 have obtained the optimal accuracy of 99.8%. This accuracy means that the model can segment 99.8% of the images in the test set. The network is limited as it was trained on pyBDSF labels.

5.6.3 Runtime

We performed a time comparison between Unet and pyBDSF under similar conditions, as shown in Figure 5.15. Beforehand, we had prepared ten datasets consisting of 100 , 200 ,..,1000 radio images of size 256 x 256 pixels. We passed the images into the two models to produce mask images. We recorded the runtime for both models. The runtime of pyBDSF for the 100 images dataset is approximately 3.5 minutes to produce mask images. Much faster than pyBDSF, Unet takes a run time of approximately 16.5 seconds to produce mask images. The second dataset consists of 200 images; the third dataset consists of 300 images, and the number of images increases until the tenth dataset, which consists of 1000 images. As expected, the total time needed to produce the mask images in every dataset till the tenth dataset keeps increasing for both models. The rate of increase for pyBDSF is higher than the rate of increase for Unet. PyBDSF, on average, has a rate of increase of approximately 9 minutes runtime for every 100 images added, while Unet, on the other hand, has a rate of increase of approximately 45 seconds run time. It takes Unet 55 seconds less to run through a dataset of 1000 images than it takes pyBDSF to run through a dataset of 100 images.

The basis for learning is the pyBDSF; this limits the model to its baseline performance in terms of accuracy. Unet surpasses pyBDSF time-wise. Once trained, it is tremendously fast. There is no need to retrain the model all over again. We save and load the trained network parameters, and runtime is reduced. In this case study of 1000 images, it took approximately 145 seconds running time as opposed to 1 hour 22 minutes it took pyBDSF. The faster runtime is the unique selling point of this work. Unet is extremely quick to produce mask images. This, therefore, provides a solution to the previously mentioned problem of processing images with millions of sources. The point of this work is to demonstrate that Unet is potentially helpful as a source segmentation model in radio astronomy.

In this chapter, Unet segmentation architectures were discussed in detail. Unet has been applied to radio images, and the results were compared with the target images produced by pyBDSF. Unet has been successfully applied to radio astronomical data and has performed with a testing accuracy of 99.8% using both Dice and the "Intersection over Union" coefficient. The overlap between the target images produced by pyBDSF and those that Unet has produced shows that Unet maps out more emissions than pyBDSF. In Chapter 6, we shall have a conclusive discussion on how and why Unet produced more accurate results than pyBDSF when it learned to create masks using pyBDSF. We will also discuss future work that needs to be done to make Unet the standard tool to detect sources from images preserving the morphological structure of the objects.

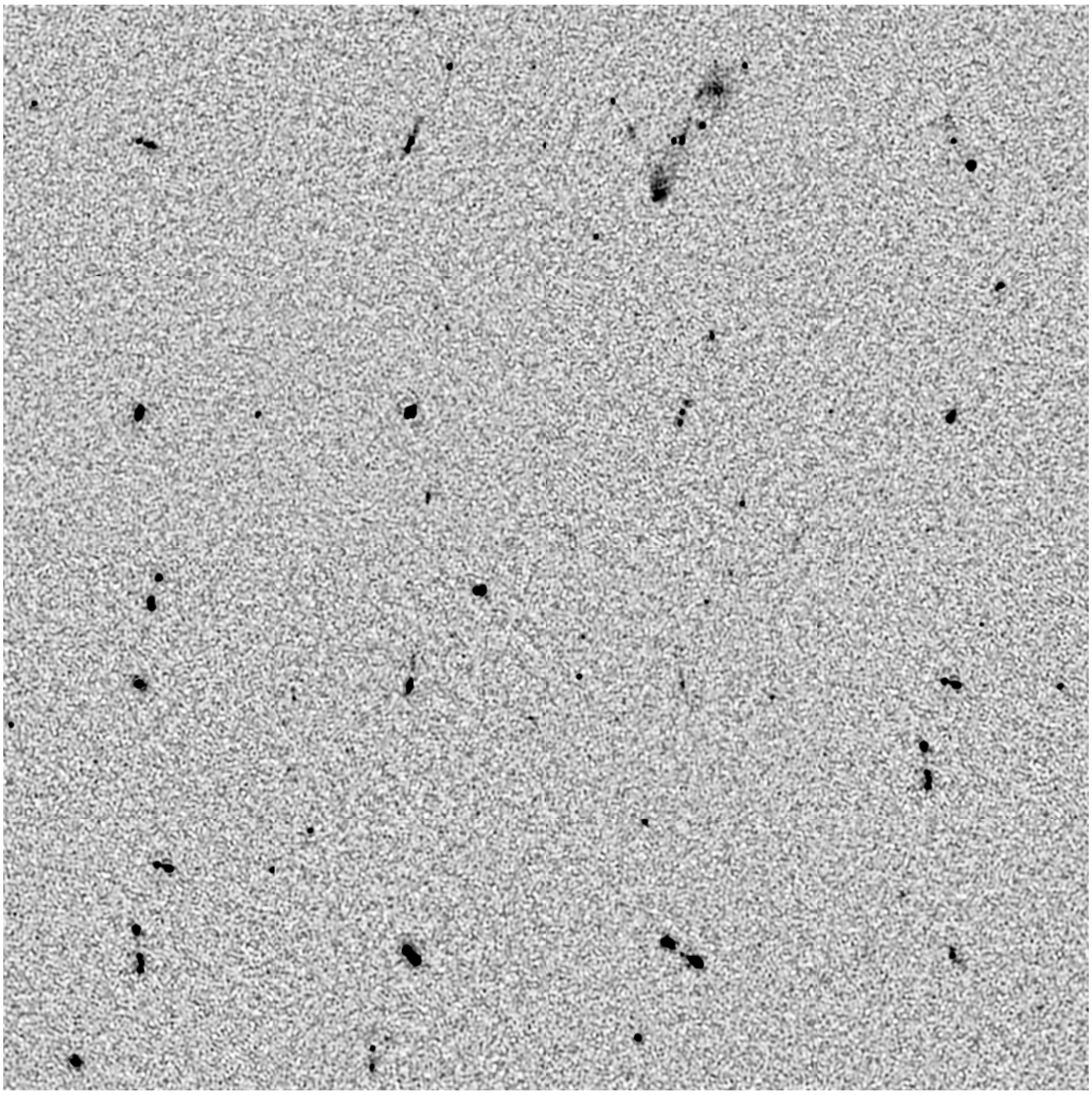


Figure 5.12: An aggregated image of 16 images of size 256×256 pixels produced by MeerKAT to make a big image of 1024×1024 pixels.

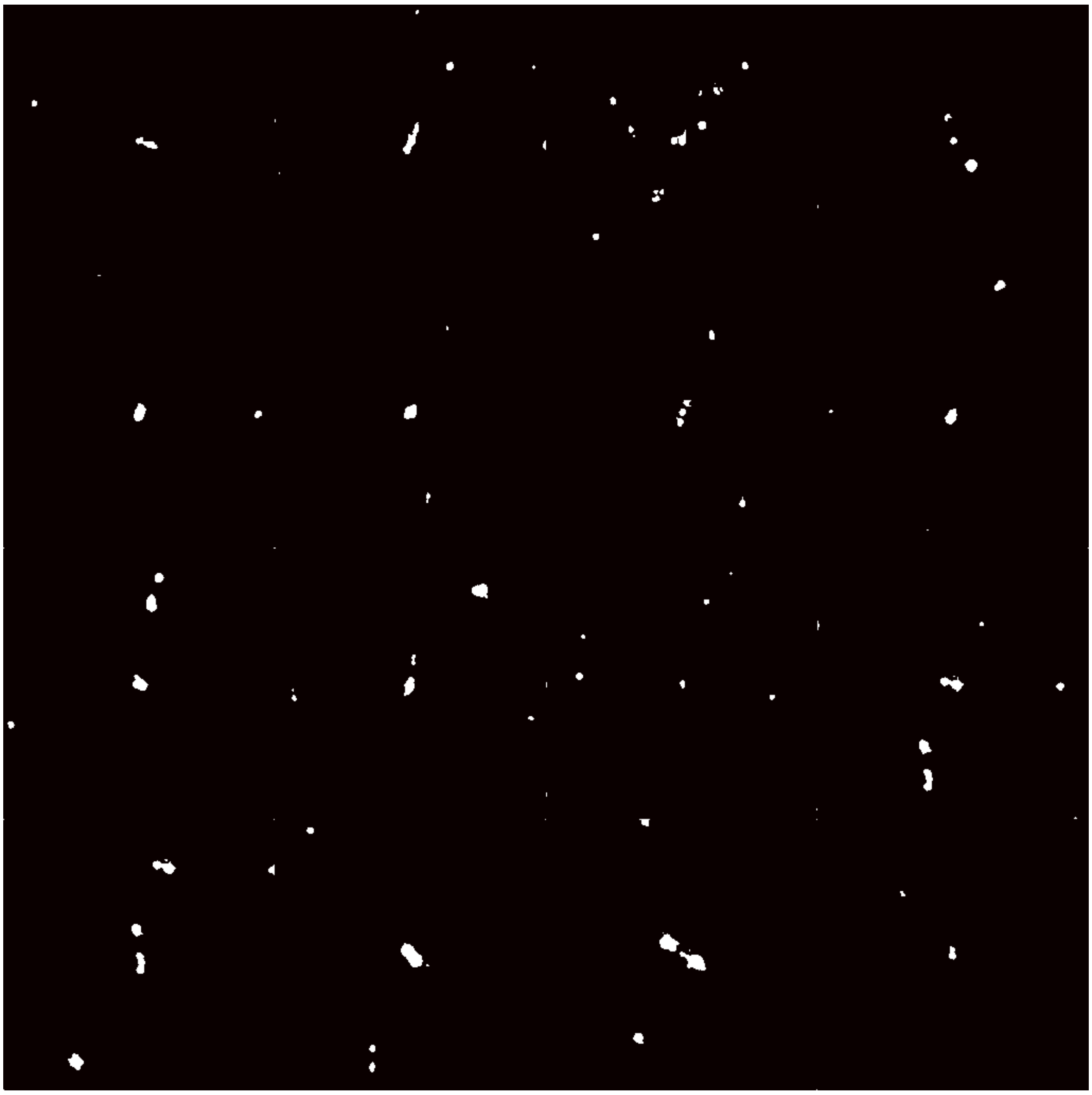


Figure 5.13: An aggregated image of 16 mask images of size 256 × 256 pixels produced by Unet to make a big image of 1024 × 1024 pixels.

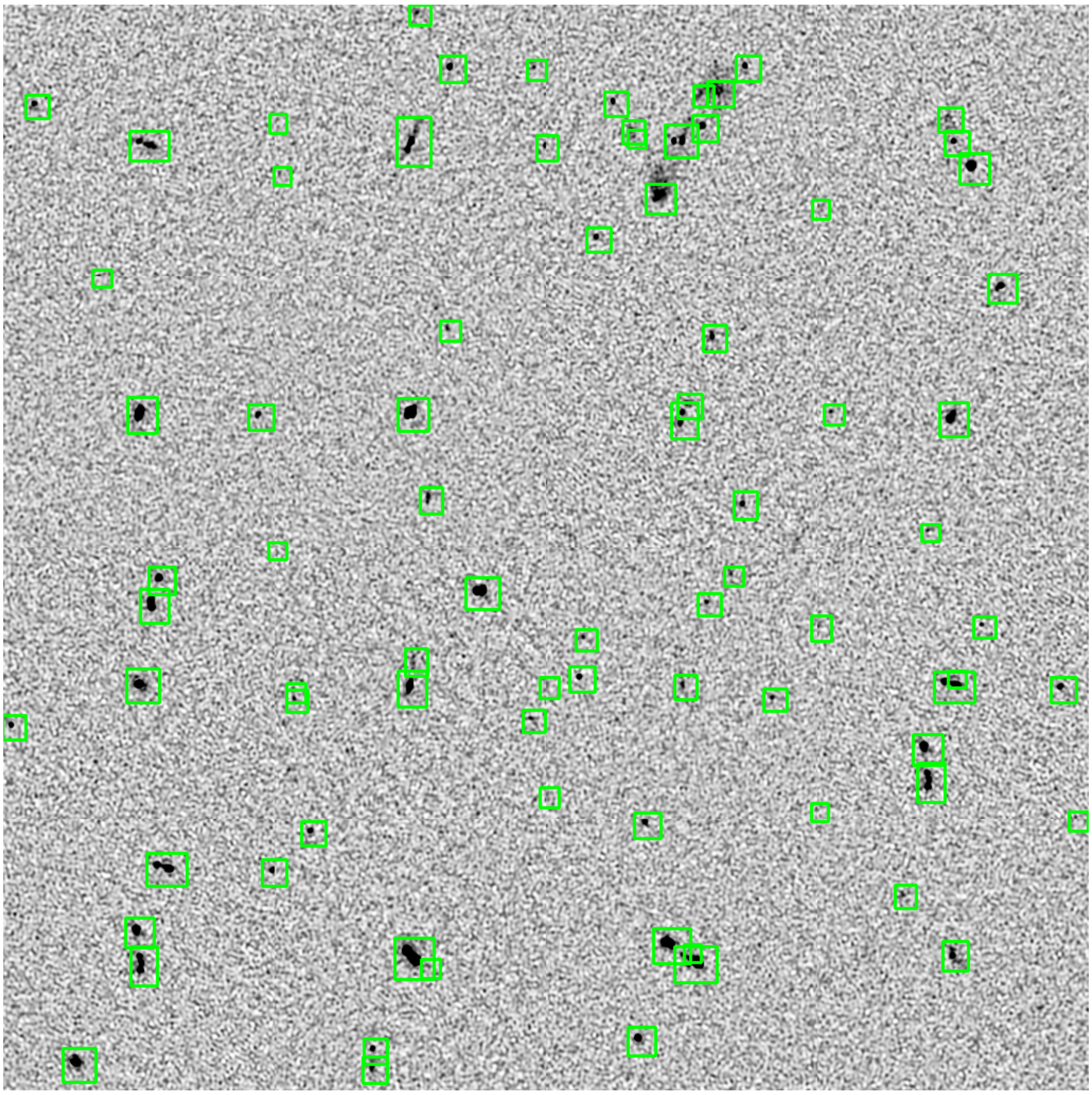


Figure 5.14: An aggregated image of 16 images of size 256 × 256 pixels that have run through the OpenCV detection algorithm to make a big image of 1024 × 1024 pixels.

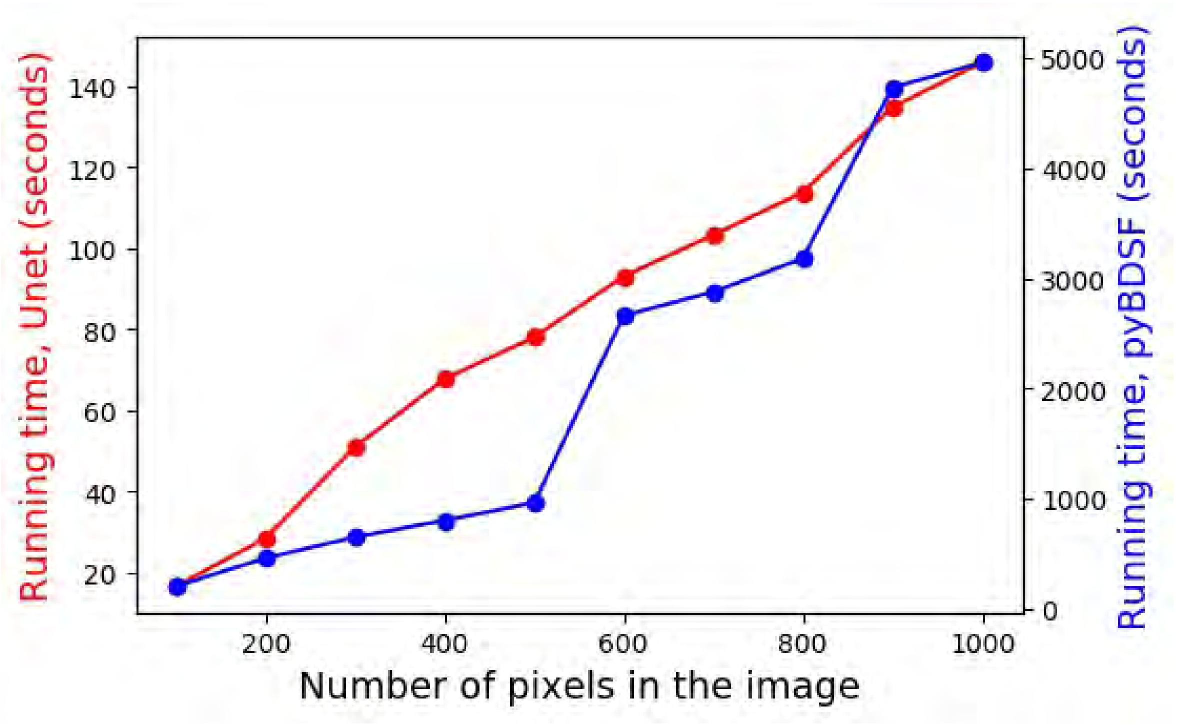


Figure 5.15: The above plot comparing the runtime for pyBDSF and Unet to produce mask images shows that that for the same number of pixels in the image, Unet is faster by a factor of 35 than pyBDSF.

Chapter 6

Conclusion

This study aims to create a tool that will automate source segmentation in radio images and detect the sources within them. The new generation of radio telescopes are designed to have a high resolution; consequently, the images they produce will have a high resolution and contain a high density of radio sources. It is tedious and time-consuming to separate the individual sources manually, and traditional machine learning methods fall short. They are unable to process large inputs of data efficiently and require human assistance to select variables. Therefore the tool has to be capable of dealing with the complexity of these radio images. The expected amounts of raw data from these upcoming radio telescope surveys are vast, estimated to be in the petabyte region. This data will consequently affect the imaging space, creating an apparent need for automation and the ability to handle large amounts of noisy data.

In the earlier chapters we touched on interferometry, the method that is used to produce the vast amount of raw data we expect. We learn that a diameter of a telescope is directly proportional to its resolution. Interferometry integrates an array of individual telescopes with the idea of having a combined collecting area which will be larger than any individual telescope in the array. The baselines of these individual telescopes creates a property (larger collecting area) that gives an interferometer a higher resolution than a single dish telescope. When electro-magnetic waves from distant sources transverse to earth, they land on the collecting area before they bounce off to a sub-reflector. They are then passed through a receiver where they are amplified and converted into voltage. Thereafter, they are digitized from analogue and then transported to a correlator. In the correlator, the data from individual telescopes will be correlated and different forms of delays on individual telescopes will be accounted for. Thereafter, the raw data is eventually processed into different data products. The different data products could be RDB files, Measurement sets, FITS files et cetera. This study is interested in the FITS files. The FITS files contain radio sources infused with noise. These radio sources can be continuum, AGN or star-forming galaxies. This study is interested mainly in radio galaxy morphology. Currently, there are source finders that are used in the field. We examine the details of these source finders, ConvoSource, ProFound, pyBDSF, COSMODEEP, ClaRAN, Deep source, and Aegean. We consider their advantages as well as their shortcomings. Unfortunately, most of the source finder's shortcomings are such that they cannot cope with the complexity of the images we are expecting, so; we need to find alternate ways of source finding in radio images that will deal with finding sources in noisy data as accurately and as efficiently as possible.

There is a certain level of speed and precision that humans can reach when performing tasks beyond which external assistance is required. Machine learning techniques can enhance human efforts by performing tasks in an automated way. Traditional shallow machine learning algorithms require the assistance of humans to extract features, while deep learning does not. Deep learning is fully exploited in other fields and has shown promising results compared to shallow machine learning models that tend to fail when dealing with vast amounts of data. Deep learning produces intelligent machines that can learn and make decisions like humans. A neural network, which is inspired by the neural connectivity of the human brain, provides the answer. We investigate two types of neural network architectures: ANN and CNN. Both these architectures consist of an input layer where data is presented, a hidden layer that processes the inputs and an output layer that produces the results. These layers consist of groups of multiple neurons with weighted connections. The difference between them is that an ANN is computationally expensive since the different neurons of the model do not share weights. Therefore, it becomes dense, unlike CNN, which shares weights of neurons yielding sparse representations. CNN is commonly used for computer vision and image segmentation.

To study the radio images, we partitioned the images to a granular state to perform pixel by pixel labelling. The intention is to separate all the sources from the background noise. The method of pixel by pixel labeling is called semantic segmentation. We investigated three popular architectures that perform semantic segmentation. (1) FCN, (2) Unet, and (3) mask R-CNN. FCN is the basic architecture for the other two architectures. So Unet and mask R-CNN are improvements of FCN. From these two improvements, Unet was chosen as the preferred architecture that will be applied to radio data. The choice was based on its ability to handle noisy, complex inputs and be more successful than conventional models in pixel-based image segmentation. Unet was applied to radio images. Firstly, it was applied to ATLBS data that has less resolution than the MeerKAT data that will ultimately be used for the study. The data was preprocessed for the Unet model. We explored different strategies to prepare the dataset. Eventually, after many trials, we managed to create a dataset of 256 x 256 pixel images using the ATLBS data. After that, we created a Unet network that consists of an encoder and a decoder phase with a series of convolutional layers, maximum pooling layers and dropout layers. We fed this model with the ATLBS dataset and their corresponding mask images that were created by pyBDSF. Initially, we obtained an accuracy performance that was less than a guess. The assumption we made about this poor performance was that the dataset of 119 images was not adequate. Therefore we augmented the data as a way of increasing the dataset. Different combinations of hyperparameters were tried out with the augmented data. The most optimal model we obtained achieved an accuracy of 93%. We decided to infer this model to higher resolution data produced by MeerKAT. However, as with the first dataset, we had to pre-process the data to get it to the desired shape and format. Feeding the dataset into the Unet model, we achieved a performance accuracy of 30%, which was poor. Alterations were made to the pre-processing steps. We produced a mask image before making cut-outs, and this significantly improved the optimized accuracy, where we obtained a performance accuracy of 99.8%. This model assisted us to separate radio sources from background noise. The output of this model is a segmentation map that shows sources in one colour and the background in another colour. The segmentation map becomes visually convenient for astronomers to see where in the image the sources are and where the background noise is.

A better visual tool for astronomers would involve detecting the individual sources and placing a border around them. We performed this process using open computer vision tools. The model expects an input image of any size. The image that we fed the model at first is the 256×256 image created by MeerKAT. The model creates contours around blobs inside the image; after that, it adds a few pixels to draw a box around the blob. The model can also extract the individual sources from the main image and put them in individual images in approximately 1.3 seconds. PyBDSF is used as its basis for learning, However; Unet managed to surpass the baseline performance. Unet, once trained, is tremendously fast. There is no need to retrain the model all over again. We save weights from the training process and load the trained network parameters and run-time is reduced when testing on unseen data. In this particular case study of 100 images, it took approximately 9 seconds. This is the unique selling point of this work. Not only does it replace pyBDSF with something as good but extremely quick. This thus provides a solution to the problem mentioned previously of processing images with millions of sources. Unet is useful as a source finder but due to it being trained only on pyBDSF labels, it will not outperform it. This work shows that Unet can be used for radio source segmentation and shows a lot of promise if given simulations on hand-labelled training data. Ultimately, We combine 16 256 x 256 pixel images to form a bigger image that shows mask images on a bigger image as well as detected blocks on larger images.

The ground-truth images are made with pyBDSF, contributing to what and how the model learns in order to create accurate masks. For future optimizations, handcrafting the ground-truth images (although tedious) would be helpful and more accurate predictions would be produced. The other drawback is that the model is using one specific image size to train and test the network. For future purposes, a modification allowing any preferred size would be an improvement and does not have to be an image that is divisible by 256 pixels. The final improvement that could be made is to create a catalogue of the individual radio sources and create information about the RA/DEC of the sources. This would be helpful in further studies, such as the classification of radio sources into morphological classes [Aniyan and Thorat (2017)].

Bibliography

Abd Alaziz, Wael (July 1, 2011). “Design and Development of Multi-Band Microstrip Rectangular Fractal Antenna for Wireless Applications”. PhD thesis. DOI: 10.13140/RG.2.2.21507.45606.

An, Tao (Jan. 23, 2019). “Science Opportunities and Challenges Associated with SKA Big Data”. In: *arXiv:1901.07756 [astro-ph]*. arXiv: 1901.07756. URL: http://arxiv.org/abs/1901.07756 (visited on 05/03/2021).

Anderson, James M. et al. (2014). *Autonomous vehicle technology: a guide for policy-makers*. In collab. with Rand Corporation and {and} Technology (Program) Rand Transportation Space. OCLC: ocn867840251. Santa Monica, CA: Rand Corporation. 185 pp. ISBN: 978-0-8330-8398-2.

Aniyan, Arun and Kshitij Thorat (June 13, 2017). “Classifying Radio Galaxies with Convolutional Neural Network”. In: *The Astrophysical Journal Supplement Series* 230.2, p. 20. ISSN: 1538-4365. DOI: 10.3847/1538-4365/aa7333. arXiv: 1705.03413. URL: http://arxiv.org/abs/1705.03413 (visited on 03/21/2021).

Apolloni, Bruno, Giacomo Zamponi, and Anna Maria Zanaboni (July 1998). “Learning fuzzy decision trees”. In: *Neural Networks: The Official Journal of the International Neural Network Society* 11.5, pp. 885–895. ISSN: 1879-2782. DOI: 10.1016/s0893-6080(98)00030-6.

Banfield, J. K. et al. (Nov. 1, 2015). “Radio Galaxy Zoo: host galaxies and radio morphologies derived from visual inspection”. In: *Monthly Notices of the Royal Astronomical Society* 453. ADS Bibcode: 2015MNRAS.453.2326B, pp. 2326–2340. ISSN: 0035-8711. DOI: 10.1093/mnras/stv1688. URL: https://ui.adsabs.harvard.edu/abs/2015MNRAS.453.2326B (visited on 03/10/2022).

Bay, Herbert et al. (2008). “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding* 110.3, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

Beswick, Rob et al. (May 29, 2015). “SKA studies of nearby galaxies: star-formation, accretion processes and molecular gas across all environments”. In: p. 070. DOI: 10.22323/1.215.0070.

Biological neuron (2010). URL: http://cnx.org/contents/GFy_h8cu@9.87:c9j4p0aj@3/Neurons-and-Glial-Cells (visited on 08/10/2022).

Canny, John (Nov. 1986). “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6, pp. 679–698. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851. URL: https://ieeexplore.ieee.org/document/4767851 (visited on 03/08/2022).

Cheeseman, Peter et al. (1988). “Bayesian Classification”. In: p. 5.

Chen, Liang-Chieh et al. (June 7, 2016). “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”. In: *arXiv:1412.7062 [cs]*. arXiv: 1412.7062. URL: http://arxiv.org/abs/1412.7062 (visited on 05/03/2021).

Chomiuk, Laura and Matthew S. Povich (Dec. 1, 2011). “Toward a Unification of Star Formation Rate Determinations in the Milky Way and Other Galaxies”. In: *The Astronomical Journal* 142.6, p. 197. ISSN: 0004-6256, 1538-3881. DOI: 10.1088/0004-6256/142/6/197. arXiv: 1110.4105. URL: http://arxiv.org/abs/1110.4105 (visited on 05/14/2021).

Cireşan, Dan, Ueli Meier, and Juergen Schmidhuber (Feb. 13, 2012). “Multi-column Deep Neural Networks for Image Classification”. In: *arXiv:1202.2745 [cs]*. arXiv: 1202.2745. URL: http://arxiv.org/abs/1202.2745 (visited on 03/15/2021).

Comaniciu, D. and P. Meer (May 2002). “Mean shift: a robust approach toward feature space analysis”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5, pp. 603–619. ISSN: 01628828. DOI: 10.1109/34.1000236. URL: http://ieeexplore.ieee.org/document/1000236/ (visited on 03/08/2022).

Cornwell, T. J. and K. F. Evans (Feb. 1, 1985). “A simple maximum entropy deconvolution algorithm”. In: *Astronomy and Astrophysics* 143, pp. 77–83. ISSN: 0004-6361. URL: http://adsabs.harvard.edu/abs/1985A%26A...143...77C (visited on 03/17/2021).

Cornwell, Tim (Nov. 1, 2008). “Multiscale CLEAN Deconvolution of Radio Synthesis Images”. In: *Selected Topics in Signal Processing, IEEE Journal of* 2, pp. 793–801. DOI: 10.1109/JSTSP.2008.2006388.

Cygnus A (1984). URL: https://www.nrao.edu/archives/items/show/33386/ (visited on 08/10/2022).

Dewdney, Peter E. et al. (Aug. 2009). “The Square Kilometre Array”. In: *Proceedings of the IEEE* 97.8. Conference Name: Proceedings of the IEEE, pp. 1482–1496. ISSN: 1558-2256. DOI: 10.1109/JPROC.2009.2021005.

Fanaroff, B. L. and J. M. Riley (Apr. 1, 1974). “The Morphology of Extragalactic Radio Sources of High and Low Luminosity”. In: *Monthly Notices of the Royal Astronomical Society* 167.1, 31P–36P. ISSN: 0035-8711. DOI: 10.1093/mnras/167.1.31P. URL: https://doi.org/10.1093/mnras/167.1.31P (visited on 03/21/2021).

FCN (2015). URL: https://nanonets.com/blog/how-to-do-semantic-segmentation-using-deep-learning/.

Galambos, C, J Kittler, and Jiri Matas (2001). “Gradient based progressive probabilistic Hough transform”. In: *Vision, Image and Signal Processing, IEE Proceedings -* 148, pp. 158 –165. DOI: 10.1049/ip-vis:20010354.

Gheller, Claudio, Franco Vazza, and Annalisa Bonafede (Nov. 1, 2018). “Deep Learning Based Detection of Cosmological Diffuse Radio Sources”. In: *Monthly Notices of the Royal Astronomical Society* 480.3, pp. 3749–3761. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/sty2102. arXiv: 1809.03315. URL: http://arxiv.org/abs/1809.03315 (visited on 03/10/2022).

Guarascio, Massimo, Giuseppe Manco, and Ettore Ritacco (Jan. 1, 2019). “Deep Learning”. In: *Encyclopedia of Bioinformatics and Computational Biology*. Ed. by Shoba Ranganathan et al. Oxford: Academic Press, pp. 634–647. ISBN: 978-0-12-811432-2. DOI: 10.1016/B978-0-12-809633-8.20352-X. URL: https://www.

sciencedirect.com/science/article/pii/B978012809633820352X (visited on 03/09/2022).

Gupta, Chirag (2021). "Modern Machine and Deep Learning Systems as a way to achieve Man-Computer Symbiosis". In: p. 8.

Hale, C L et al. (Aug. 11, 2019). "Radio source extraction with ProFound". In: *Monthly Notices of the Royal Astronomical Society* 487.3, pp. 3971–3989. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stz1462. URL: https://academic.oup.com/mnras/article/487/3/3971/5511783 (visited on 03/10/2022).

Hancock, P. J. et al. (Dec. 1, 2012). "Aegean: Compact source finding in radio images". In: *Astrophysics Source Code Library*, ascl:1212.009. URL: https://ui.adsabs.harvard.edu/abs/2012ascl.soft12009H (visited on 07/28/2021).

Harris, C. and M. Stephens (1988). "A Combined Corner and Edge Detector". In: *Procedings of the Alvey Vision Conference 1988*. Alvey Vision Conference 1988. Manchester: Alvey Vision Club, pp. 23.1–23.6. DOI: 10.5244/C.2.23. URL: http://www.bmva.org/bmvc/1988/avc-88-023.html (visited on 03/08/2022).

Hebb, Donald Olding (1949). "The organization of behavior; a neuropsycholocigal theory". In: *A Wiley Book in Clinical Psychology* 62, p. 78.

Heidrich-Meisner, Verena et al. (Jan. 1, 2007). "Reinforcement learning in a Nutshell". In: pp. 277–288.

Herculano-Houzel, Suzana (June 26, 2012). "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost". In: *Proceedings of the National Academy of Sciences* 109 (Supplement 1). Publisher: National Academy of Sciences Section: Colloquium Paper, pp. 10661–10668. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1201895109. URL: https://www.pnas.org/content/109/Supplement_1/10661 (visited on 03/13/2021).

Högbom, J. A. (June 1, 1974). "Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines". In: *Astronomy and Astrophysics Supplement Series* 15, p. 417. ISSN: 0365-0138. URL: http://adsabs.harvard.edu/abs/1974A%26AS...15..417H (visited on 03/17/2021).

Jansky, K. G. (Dec. 1932). "Directional Studies of Atmospherics at High Frequencies". In: *Proceedings of the Institute of Radio Engineers* 20.12. Conference Name: Proceedings of the Institute of Radio Engineers, pp. 1920–1932. ISSN: 2162-6626. DOI: 10.1109/JRPROC.1932.227477.

Jansky, Karl G. (July 1933). "Radio Waves from Outside the Solar System". In: *Nature* 132.3323. Number: 3323 Publisher: Nature Publishing Group, pp. 66–66. ISSN: 1476-4687. DOI: 10.1038/132066a0. URL: https://www.nature.com/articles/132066a0 (visited on 04/10/2021).

Johnston, S. et al. (Dec. 2008). "Science with ASKAP - the Australian Square Kilometre Array Pathfinder". In: *Experimental Astronomy* 22.3, pp. 151–273. ISSN: 0922-6435, 1572-9508. DOI: 10.1007/s10686-008-9124-7. arXiv: 0810.5187. URL: http://arxiv.org/abs/0810.5187 (visited on 09/21/2021).

Jonas, Justin (Sept. 1, 2009). "MeerKAT—The South African Array With Composite Dishes and Wide-Band Single Pixel Feeds". In: *Proceedings of the IEEE* 97, pp. 1522–1530. DOI: 10.1109/JPROC.2009.2020713.

Kayalibay, Baris, Grady Jensen, and Patrick van der Smagt (July 25, 2017). "CNN-based Segmentation of Medical Imaging Data". In: *arXiv:1701.03056 [cs]*. arXiv: 1701.03056. URL: http://arxiv.org/abs/1701.03056 (visited on 03/14/2021).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25, pp. 1097–1105.

Kussul, N. et al. (May 2017). "Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data". In: *IEEE Geoscience and Remote Sensing Letters* 14.5. Conference Name: IEEE Geoscience and Remote Sensing Letters, pp. 778–782. ISSN: 1558-0571. DOI: 10.1109/LGRS.2017.2681128.

LeCun, Yann, Y. Bengio, and Geoffrey Hinton (May 28, 2015). "Deep Learning". In: *Nature* 521, pp. 436–44. DOI: 10.1038/nature14539.

Liu, Qiong and Ying Wu (Jan. 1, 2012). "Supervised Learning". In: DOI: 10.1007/978-1-4419-1428-6_451.

Liu, Tianyi et al. (June 3, 2015). "Implementation of Training Convolutional Neural Networks". In: *arXiv:1506.01195 [cs]*. arXiv: 1506.01195. URL: http://arxiv.org/abs/1506.01195 (visited on 03/10/2022).

Lukic, V., F. De Gasperin, and M. Brüggen (Dec. 19, 2019). "ConvoSource: Radio-Astronomical Source-Finding with Convolutional Neural Networks". In: *arXiv:1910.03631 [astro-ph]*. arXiv: 1910.03631. URL: http://arxiv.org/abs/1910.03631 (visited on 03/10/2022).

M84 (2006). URL: https://www.cv.nrao.edu/~abridle/3c433.htm (visited on 08/10/2022).

Mathew, Amitha, Amudha Arul, and S. Sivakumari (Jan. 1, 2021). "Deep Learning Techniques: An Overview". In: pp. 599–608. ISBN: 9789811533822. DOI: 10.1007/978-981-15-3383-9_54.

McCulloch, Warren S. and Walter Pitts (Dec. 1, 1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259 (visited on 01/13/2021).

Medsker, Larry and Lakhmi C. Jain (Dec. 20, 1999). *Recurrent Neural Networks: Design and Applications*. Google-Books-ID: ME1SAkN0PyMC. CRC Press. 414 pp. ISBN: 978-1-4200-4917-6.

Merloni, Andrea and Sebastian Heinz (June 2008). "A synthesis model for AGN evolution: supermassive black holes growth and feedback modes". In: *Monthly Notices of the Royal Astronomical Society*, ???–??? ISSN: 00358711, 13652966. DOI: 10.1111/j.1365-2966.2008.13472.x. URL: https://academic.oup.com/mnras/article-lookup/doi/10.1111/j.1365-2966.2008.13472.x (visited on 03/19/2021).

Mesarcik, Michael et al. (Aug. 1, 2020). "Deep Learning Assisted Data Inspection for Radio Astronomy". In: *Monthly Notices of the Royal Astronomical Society* 496.2, pp. 1517–1529. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/staa1412. arXiv: 2005.13373. URL: http://arxiv.org/abs/2005.13373 (visited on 03/23/2021).

Mohan, Niruj and David Rafferty (Feb. 1, 2015). "PyBDSF: Python Blob Detection and Source Finder". In: *Astrophysics Source Code Library*, ascl:1502.007. URL: http://adsabs.harvard.edu/abs/2015ascl.soft02007M (visited on 05/10/2021).

Mortlock, Daniel J. et al. (June 29, 2011). “A luminous quasar at a redshift of z = 7.085”. In: *Nature* 474.7353, pp. 616–619. ISSN: 1476-4687. DOI: 10.1038/nature10159.

Mosiane, Olorato et al. (May 2017). “Radio Frequency Interference Detection using Machine Learning.” In: *IOP Conference Series: Materials Science and Engineering* 198, p. 012012. ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/198/1/012012. URL: https://iopscience.iop.org/article/10.1088/1757-899X/198/1/012012 (visited on 03/23/2021).

Norris, Ray P. (Oct. 2017). “Extragalactic Radio Continuum Surveys and the Transformation of Radio Astronomy”. In: *Nature Astronomy* 1.10, pp. 671–678. ISSN: 2397-3366. DOI: 10.1038/s41550-017-0233-y. arXiv: 1709.05064. URL: http://arxiv.org/abs/1709.05064 (visited on 04/11/2021).

Novikov, Alexey A. et al. (Feb. 13, 2018). “Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs”. In: *arXiv:1701.08816 [cs]*. arXiv: 1701.08816. URL: http://arxiv.org/abs/1701.08816 (visited on 03/29/2021).

O’Shea, Keiron and Ryan Nash (Dec. 2, 2015). “An Introduction to Convolutional Neural Networks”. In: *arXiv:1511.08458 [cs]*. arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458 (visited on 05/18/2021).

Padovani, P. et al. (Nov. 2017). “Active galactic nuclei: what’s in a name?” In: *The Astronomy and Astrophysics Review* 25.1, p. 2. ISSN: 0935-4956, 1432-0754. DOI: 10.1007/s00159-017-0102-9. URL: http://link.springer.com/10.1007/s00159-017-0102-9 (visited on 03/18/2021).

Reber, Grote (June 1, 1940). “Notes: Cosmic Static.” In: *The Astrophysical Journal* 91, pp. 621–624. ISSN: 0004-637X. DOI: 10.1086/144197. URL: http://adsabs.harvard.edu/abs/1940ApJ....91..621R (visited on 04/10/2021).

Robotham, A. S. G. et al. (May 21, 2018). “ProFound: Source Extraction and Application to Modern Survey Data”. In: *Monthly Notices of the Royal Astronomical Society* 476.3, pp. 3137–3159. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/sty440. arXiv: 1802.00937. URL: http://arxiv.org/abs/1802.00937 (visited on 03/10/2022).

Rohlfs, K. and T. Wilson (Jan. 1, 2000). *Tools of Radio Astronomy*. Journal Abbreviation: Tools of radio astronomy / K. Rohlfs, T.L. Wilson. New York : Springer, 2000. (Astronomy and astrophysics library,ISSN0941-7834) Publication Title: Tools of radio astronomy / K. Rohlfs, T.L. Wilson. New York : Springer, 2000. (Astronomy and astrophysics library,ISSN0941-7834). ISBN: 978-3-540-40387-6. DOI: 10.1007/978-3-662-05394-2.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (May 18, 2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *arXiv:1505.04597 [cs]*. arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 04/21/2021).

Rosenblatt, F. (1958). “The perceptron: A probabilistic model for information storage and organization in the brain”. In: *Psychological Review* 65.6. Place: US Publisher: American Psychological Association, pp. 386–408. ISSN: 1939-1471(Electronic),0033-295X(Print). DOI: 10.1037/h0042519.

Rosten, Edward and Tom Drummond (2006). “Machine Learning for High-Speed Corner Detection”. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Series Title: Lecture Notes in Computer Science.

Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 430–443. ISBN: 978-3-540-33832-1 978-3-540-33833-8. DOI: 10.1007/11744023_34. URL: http://link.springer.com/10.1007/11744023_34 (visited on 03/08/2022).

Sabater, J. et al. (Apr. 1, 2017). "Calibration of LOFAR data on the cloud". In: *Astronomy and Computing* 19, pp. 75–89. ISSN: 2213-1337. DOI: 10.1016/j.ascom.2017.04.001. URL: https://www.sciencedirect.com/science/article/pii/S2213133716301470 (visited on 05/18/2021).

Sadr, A. Vafaei et al. (Apr. 1, 2019). "DeepSource: Point Source Detection using Deep Learning". In: *Monthly Notices of the Royal Astronomical Society* 484.2, pp. 2793–2806. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stz131. arXiv: 1807.02701. URL: http://arxiv.org/abs/1807.02701 (visited on 03/10/2022).

Saleh, Ziyad (Apr. 9, 2019). "Artificial Intelligence Definition, Ethics and Standards". In.

Schmidhuber, Juergen (Jan. 2015). "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61, pp. 85–117. ISSN: 08936080. DOI: 10.1016/j.neunet.2014.09.003. arXiv: 1404.7828. URL: http://arxiv.org/abs/1404.7828 (visited on 03/15/2021).

Simonyan, Karen and Andrew Zisserman (Apr. 10, 2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]*. arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556 (visited on 03/14/2021).

Skurichina, M. and R. P. Duin (2002). "Bagging, Boosting and the Random Subspace Method for Linear Classifiers". In: *Pattern Analysis & Applications*. DOI: 10.1007/s100440200011.

Soille, Pierre, S Beucher, and JF Rivest (1993). "Morphological gradients". In: *Journal of Electronic Imaging* 2, p. 12.

Suykens, J.A.K. and J. Vandewalle (June 1, 1999). "Least Squares Support Vector Machine Classifiers". In: *Neural Processing Letters* 9.3, pp. 293–300. ISSN: 1573-773X. DOI: 10.1023/A:1018628609742. URL: https://doi.org/10.1023/A:1018628609742 (visited on 03/14/2021).

Tan, Y (2016). *GPU-based parallel implementation of swarm intelligence algorithms*. Morgan Kaufmann.

The binary notes (2022). URL: https://thebinarynotes.com/instance-segmentation-using-mask-r-cnn/ (visited on 08/10/2022).

Thompson, Anthony, James Moran, and Jr Swenson George (Jan. 1, 1991). *Interferometry and Synthesis in Radio Astronomy*. Vol. -1. Journal Abbreviation: New York, Wiley-Interscience, 1986, 554 p. Publication Title: New York, Wiley-Interscience, 1986, 554 p. ISBN: 978-3-319-44429-1. DOI: 10.1007/978-3-319-44431-4.

Thorat, K. et al. (Jan. 1, 2013). "HIGH-RESOLUTION IMAGING OF THE ATLBS REGIONS: THE RADIO SOURCE COUNTS". In: *The Astrophysical Journal* 762.1, p. 16. ISSN: 0004-637X, 1538-4357. DOI: 10.1088/0004-637X/762/1/16. URL: https://iopscience.iop.org/article/10.1088/0004-637X/762/1/16 (visited on 04/17/2021).

Ulrich, Marie-Helene, Laura Maraschi, and C. Megan Urry (Sept. 1997). "VARIABILITY OF ACTIVE GALACTIC NUCLEI". In: *Annual Review of Astronomy and Astrophysics* 35.1, pp. 445–502. ISSN: 0066-4146, 1545-4282. DOI: 10.1146/annurev.

astro.35.1.445. URL: http://www.annualreviews.org/doi/10.1146/annurev.astro.35.1.445 (visited on 03/19/2021).

Vrahatis, Michael et al. (2000). "Introduction to artificial neural network training and applications". In: DOI: 10.13140/2.1.1755.2322.

Walczak, Steven and Narciso Cerpa (Dec. 31, 2003). "Artificial Neural Networks". In: *Encyclopedia of Physical Science and Technology*. Journal Abbreviation: Encyclopedia of Physical Science and Technology, pp. 631–645. ISBN: 978-0-12-227410-7. DOI: 10.1016/B0-12-227410-5/00837-1.

Wani, M. et al. (Jan. 1, 2020). "Unsupervised Deep Learning Architectures". In: pp. 77–94. ISBN: 9789811367939. DOI: 10.1007/978-981-13-6794-6_5.

Ward, Jr and H Joe (1963). "Hierarchical grouping to optimize an objective function". In: *Journal of the American statistical association* 58, pp. 236–244.

Wilson, A. S. and E. J. M. Colbert (Jan. 1, 1995). "The difference between radio-loud and radio-quiet active galaxies". In: *The Astrophysical Journal* 438, pp. 62–71. ISSN: 0004-637X. DOI: 10.1086/175054. URL: http://adsabs.harvard.edu/abs/1995ApJ...438...62W (visited on 03/21/2021).

Wu, Chen et al. (Jan. 1, 2019). "Radio Galaxy Zoo: CLARAN - a deep learning classifier for radio morphologies". In: *Monthly Notices of the Royal Astronomical Society* 482. ADS Bibcode: 2019MNRAS.482.1211W, pp. 1211–1230. ISSN: 0035-8711. DOI: 10.1093/mnras/sty2646. URL: https://ui.adsabs.harvard.edu/abs/2019MNRAS.482.1211W (visited on 03/10/2022).

Zhang, Yiqing et al. (Jan. 2020). "Mask-Refined R-CNN: A Network for Refining Object Details in Instance Segmentation". In: *Sensors* 20.4. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 1010. DOI: 10.3390/s20041010. URL: https://www.mdpi.com/1424-8220/20/4/1010 (visited on 07/08/2021).

Zhang, Yuchen et al. (Apr. 20, 2017). "On the learnability of fully-connected neural networks". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, pp. 83–91. URL: http://proceedings.mlr.press/v54/zhang17a.html.

Zhao, Zhen, Tao An, and Baoqiang Lao (Oct. 31, 2019). "VLBI Network SIMulator: An Integrated Simulation Tool for Radio Astronomers". In: *Journal of The Korean Astronomical Society* 52.5, pp. 207–216. DOI: 10.5303/JKAS.2019.52.5.207. arXiv: 1808.06726. URL: http://arxiv.org/abs/1808.06726 (visited on 03/10/2022).

Zupan, Jure (Jan. 3, 1994). "Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them*." In: *Acta Chimica Slovenica*, p. 27.

Ławrynowicz, Agnieszka and Volker Tresp (Jan. 1, 2014). "Introducing Machine Learning". In: *Perspectives on Ontology Learning*. Vol. 18. Journal Abbreviation: Perspectives on Ontology Learning, pp. 35–50.