Radio Interferometric Image Reconstruction for the Square Kilometer Array: A Deep Learning Approach

Tarek Allam Jr

September 12, 2016

Supervised by:

Dr. Jason McEwen & Dr. Denise Gorse

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science

of

University College London.

Department of Computer Science

University College London

Abstract

There are many algorithms that attempt to reconstruct radio interferometric images from raw visibilities acquired by radio telescopes. This project proposes a novel technique of applying deep learning using convolutional neural networks to solve the ill-posed problem of recovering images. The demand for new scalable algorithms is driven by advances in telescope technology and instrumentation; the amount of data these new telescopes will collect when in operation will not be able to be analysed using today's techniques.

Several experiments are conducted that explore different physical observational strategies to determine if a model can be developed that not only works well for a single telescope configuration but can be deployed for use with any radio telescope.

The results herein are compared to state-of-the-art radio imaging algorithms and show good results in comparison with regards response time and the quality of reconstruction, possibly paving the way for further research into this method.

Contents

List of Figures

Chapter 1

Introduction

Radio interferometric telescopes allow astronomers to make observations of the sky at otherwise inaccessible angular resolution. We are entering a new era of radio astronomy, with new radio interferometers under construction and design. A notable example is the Square Kilometre Array (SKA), whose science goals range from strong field tests of gravity to probing the era of reionization to astrobiology. However, the SKA will produce a deluge of data that existing imaging techniques will not be able to handle. This project presents novel solutions tailored to the problem of recovering images from the raw measurements acquired by radio telescopes. There exist many techniques today for signal reconstruction such as compressed sensing and deconvolution methods. These will be discussed further in Chapter 2 which will aim to provide an overview of the state of research today regarding image reconstruction.

Following a discussion of the radio imaging techniques of today, Chapter 2 also presents alternative areas of research in image processing that can be used for radio interferometric image reconstruction. This chapter will also present a brief overview of modern machine learning methods and how they relate to the problem at hand.

The design and methodology is described in Chapter 3. This chapter

will outline how data has been collected and pre-processed. It will give an overview of the choices that have been made in regards to the research pipeline noting that as a research project much of the development has been carried out in an iterative approach, the decisions as to the direction of research are accounted here.

Chapter 4 examines the results that have been gathered, including a full analysis of the results as well as interpretation what these results physically mean in the context of radio interferometry.

Finally, the project is concluded in Chapter 5 with a discussion of the results obtained. An outline of potential further work is expressed and where further work can be carried out.

An Appendix section follows this report which includes mathematical overviews for certain topics as well as larger code snippets that have been used for this project. The reader is encouraged to review the appendices to gain a full appreciation of the problems facing radio interferometric image reconstruction. The reader is also encouraged to visit `http://www.astro-informatics.org/wikis/radio_learning/`. Apart from two housekeeping posts all the material on this website has been produced in conjunction with this project.

Chapter 2

Background & Literature Review

2.1 Radio Astronomy & Interferometry

For thousands of years, man has looked to the sky to observe the heavens and wonder about the celestial objects above. The observation of celestial objects is called astronomy, and in recent times an explosion of activity has been seen due to the improvements in technology and instrumentation.

Earth's atmosphere limits what can be observed on the ground. Depending on the wavelength, λ, of the electromagnetic radiation, most is absorbed by the atmosphere. There exists a few windows where electromagnetic radiation can reach the instruments on the ground (Figure 2.1), the most important of which is the *optical window*, ranging from around 300nm to 800nm. This is considered to be the most important as this is range that corresponds to the sensitivity of the human eye [Karttunen et al., 2007]. Another large window exists at wavelengths 1mm to 20m, called the *radio window*. For a long time, the majority of research was conducted in only in the optical window, however, as technology has progressed with communications and signal processing, radio astronomy (as well as X-ray astronomy, infrared astronomy

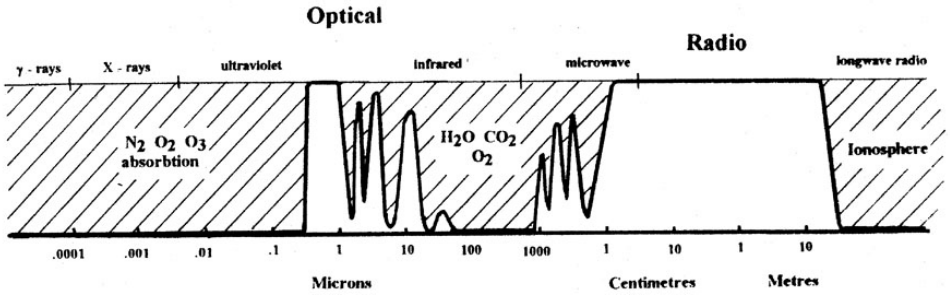


Figure 2.1: This figure shows which wavelengths of electromagnetic radiation are observable at ground level (Reproduced from [Burke and Graham-Smith, 2010])

etc.) is now as much of an active area as optical astronomy.

The radio spectrum is of great interest to astronomers and astrophysicists. Radio astronomy enables the study of the cosmic microwave background (CMB) which holds information about the beginnings of the universe. Radio astronomy is also the specialist tool for probing faraway exotic astrophysical objects such as neutron stars, black holes and quasars. It is possible to investigate the molecular components of the interstellar medium (ISM) with radio astronomy through analysis of emission processes such as synchrotron radiation, breaking radiation and dust emission using spectroscopy. The importance of radio astronomy can not be stressed enough but the above gives a very brief insight into the kind of physics that can be explored when making observations in the radio window.

However, when observing through radio astronomy telescopes, as with other kinds of telescopes, there is an upper limit for resolution. In simplified terms, the minimum angular distance one can observe at a given wavelength, λ, and with a diameter of of aperture, D, can be represented with the formula shown in Equation 2.1

$$\Theta = \frac{\lambda}{D} \tag{2.1}$$

where Θ is in radians.

This equation can be understood if one considers what can be observed through the human eye. The pupil of the eye has a diameter of the order of millimeters $\sim$ 2mm and observes light, which is electromagnetic radiation of wavelength $\sim 0.5\mu$m. Using Equation 2.1 the eye can resolve up to 50 arcsec of angular distance. Now consider the large optical telescope, Hubble[1]. Hubble has a much larger diameter of 2.4m but observes radiation of the same wavelength. This increase in collecting area improves the resolving power to 0.05 arcsecs which is desirable for astronomical objects that are very small. The largest radio telescope, the Arecibo Radio Telescope spans 305m in diameter, with an effective collecting area of $\sim$ 200m.[2] It observes wavelengths around 6cm which gives a resolving power of 60 arcsec, which is similar to the resolving power of the human eye. Considering this is the largest radio telescope that has been built and yet the resolving power is no better than the human eye, alternative techniques are required to improve resolution. To have single arcsec resolution at radio wavelengths one needs kilometer sized telescopes; to achieve this *synthesis* techniques are necessary using interferometric telescopes.

Interferometry is a technique that uses many small apertures to 'synthesise' a much larger one. One infers properties about the source from the received electric field created by aggregating electromagnetic radiation measurements from multiple telescopes. Current synthesis telescopes that exist today are the Very Large Array (VLA) which achieves resolution down to 1.4 arcsec from effective aperture size of 35km and the Atacama Large Millimeter Array (ALMA) which goes even further down to 0.005 arcsec resolution from effective aperture size of 15km. However, these kind of telescopes operate in a different way than normal single dish telescopes since only fragments of the aperture are really collected. What is actually mea-

[1] `http://hubblesite.org/`
[2] `http://www.naic.edu/`

sured is the *coherence* between pairs of antennas. The correlation between the signals observed at pairs of antennas is a complex *visibility* function and corresponds to a Fourier co-efficient, where the Fourier co-efficient depends on the configuration of the telescope i.e. how the pairs of antennas are arranged on the ground. Many telescopes, correlating all these signals gives many measurements in the Fourier plane. One might naively think to recover the image, simply do an inverse Fourier transform to go back to image space. However, when this is done, a *dirty image* is produced. It is called a dirty image because it is missing information due to only a finite number of measurements being collected in the Fourier plane. To completely recover the image using an inverse Fourier transform of the Fourier co-efficients, one would need to have sampled at an infinite number of points in the plane. This is the fundamental problem in radio astronomy and the core problem that is being addressed in this project; how can one best recover an image that has been observed through interferometric telescopes.

2.2 Image Reconstruction

The incomplete information that is gathered defines an ill-posed problem of image reconstruction. The problem can be represented in matrix form, shown in Equation 2.2, as the reconstruction of a signal $\boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^N$, from measured visibilities $\boldsymbol{y}$. The visibilities are of the form $\boldsymbol{y} \in \mathbb{C}^M$

$$\boldsymbol{y} = \boldsymbol{\Phi x} + n \tag{2.2}$$

where n is instrumentation noise and Φ is a matrix of the form $\Phi = \mathbf{MF}$. The matrix $\mathbf{M} \in \mathbb{R}^{\mathbf{M}\times\mathbf{N}}$ defines a binary mask that describes the configuration of the interferometer, and the matrix $\mathbf{F} \in \mathbb{C}^{\mathbf{N}\times\mathbf{N}}$ carries out a Fourier transform on the Fourier co-efficients collected. N is the number of discrete sampled points on a grid given by $N = \sqrt{N} \times \sqrt{N}$. To determine $\boldsymbol{x}$ one can

apply the adjoint matrix of Φ, $\Phi^\top$ to $\boldsymbol{y}$, where $\Phi^\top = F^\top M^\top$, resulting in the dirty image (See Equation 2.2.) The aim is to develop an algorithm that can recover the ground truth image (or as close as possible) from the dirty image that is measured.

$$\Phi^\top \boldsymbol{y} = F^\top M^\top \boldsymbol{y} = \boldsymbol{x} \tag{2.3}$$

2.2.1 Classical Algorithms

Algorithms currently exist that attempt this reconstruction with the most famous being CLEAN [Hogebom, 1974]. CLEAN, developed over 40 years ago, uses deconvolution techniques which attempt to decompose the image into a set of δ functions[3] with the assumption the object is a point source [Starck and Murtagh, 2007]. Further attempts at image reconstruction use the Maximum Entropy Method (MEM) [Skilling and Bryan, 1984] but CLEAN is still most commonly used today [Carrillo et al., 2012]. The algorithms that have been mentioned are plainly relatively old algorithms and recent research has been carried out over the last decade to develop new algorithms that can improve run time and can scale to the size of data that will be collected from telescopes in the future, such as the planned Square Kilometer Array (SKA).

The SKA specifications have been under development since 1997 [Burke and Graham-Smith, 2010] and the telescope is planned to observe first light in 2020. The SKA will exist in two locations, South Africa called MeerKAT and SKA pathfinder in Australia. The science goals of the SKA are truly profound but there exist many technological challenges that need to be overcome first. One of these challenges that is key to this project is how to deal with the deluge of data that will be collected from the telescope when

[3] `https://en.wikipedia.org/wiki/Dirac_delta_function`

in operation. The amount of data that will be gathered at the core will be multiple Terabits-per-second (Tb/s) with outlier points collecting 100Gb/s. When this is brought together at a central hub, the amount of data will be too large to be stored and processed off-line so real time in-situ processing will be required. Peta-operations-per-second as well as petabytes of memory will be needed in order to understand the science that will be implicit in the data [Barbosa et al., 2012]. It can be seen in [Carrillo et al., 2014] that classical algorithms are very slow with sub-optimal reconstructive performance. Therefore, due to the incredible data rates and requirements for in-situ processing new modern algorithms are required.

2.2.2 Modern Algorithms

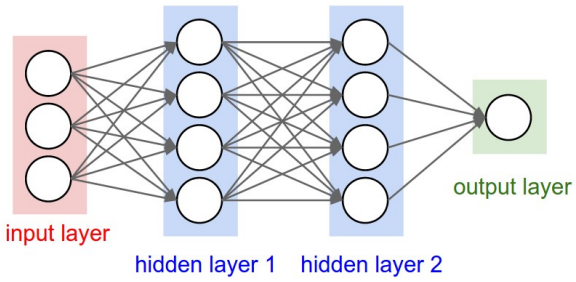
Although a lot of data will be collected, new algorithms attempt to exploit the sparsity of this data in relation to the Fourier co-efficients present using modern techniques of *compressed sensing*. Compressed sensing is a method that allows signal reconstruction whilst circumventing the Nyquist-Shannon theorem [Kutz, 2013]. Work found in [Carrillo et al., 2014], [Carrillo et al., 2012] and [Wiaux et al., 2009] show promising results regarding the quality of image reconstruction as well as algorithm run time using compressed sensing. Further discussion of similar algorithms that use compressed sensing techniques can be found in [Carrillo et al., 2014].

Another field that can be applied to the reconstruction of radio images is *super resolution*; this is the method that will be used in this work. Defined in [Starck and Murtagh, 2007], *"Super resolution consists of recovering object spatial frequency information outside the spatial bandwidth of the image formation system"* i.e. recovering information that is outside the realm of measurement. Work by [Gerchberg, 1974] and [Honma et al., 2014] show how super resolution methods can be applied to directly recover radio images. Further research in this field has been carried out by [Dong et al., 2014], in

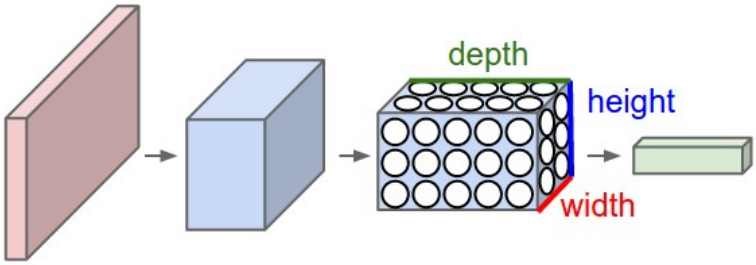
the context of natural images, showing how the general pipeline used in super resolution can be thought of as a simple convolutional neural network. These results are appealing on several levels, the first being that super resolution is already a field of research that has been applied to radio interferometric images with fair results. Also, the use of a convolutional neural network is attractive as it means that if one were able to develop an network that works for radio images, then one can benefit from the very quick test time once a network has been trained. Motivated by the results of [Dong et al., 2014] it has been decided to investigate here whether one can develop a suitable convolutional neural network that can achieve similar reconstructive performance to alternative state-of-the-art algorithms but at potentially a fraction of the test time of current methods. Details of the methodology used by [Dong et al., 2014] can be found in Section 3.1, but we shall first give in the next section an overview of what convolutional networks are and why they would be useful in the problem of radio image reconstruction.

2.3 Convolutional Neural Networks

Convolution neural networks (CNNs) have revolutionised the field of computer vision [LeCun et al., 1989]. However, it is only recently with the combined boost in computer processing power using Graphical Processing Units (GPUs) and an abundance of data that CNNs really have taken off [Krizhevsky et al., 2012]. In contrast to traditional neural networks, CNNs are not fully connected at each layer. A traditional neural network would have for each neuron a connection to each neuron in the previous layer. As an example, if one were to consider an image of $150 \times 150 \times 3$ the resulting weights produced equal to 67500, and considering that there will usually be several hidden layers of neurons, this can quickly get overwhelming. Instead, CNNs arrange neurons as a 3D volume, where the height and width are determined by the convolution operation and the depth de-



(a) Traditional Neural Network



(b) Convolutional Neural Network

Figure 2.2: Comparison traditional neural network neuron connectivity with that of a convolutional neural network (Reproduced from [Karpathy, 2016])

termined by how many feature maps desired by the user (See Figure 2.2 for a visual comparison of traditional neural networks and convolutional neural networks.)

The core computational operation of CNNs is the convolution operator. This is a mathematical operation the combines two functions to produce a third. It works in effect by sliding one function over another, multiplying it and adding together, see Equation 2.4 [Bengio and Courville, 2016, Karpathy, 2016].

$$h[x,y] = f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2] \quad (2.4)$$

where $h[x,y]$ is the new output function convolving $f[x,y]$ with $g[x,y]$. The output function is usually called the *feature map* or *activation map*, with

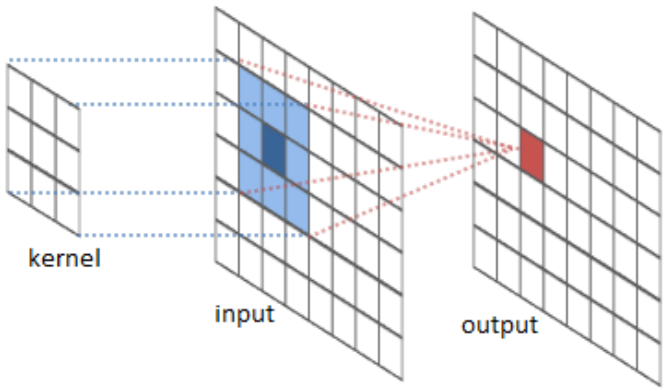


Figure 2.3: This figure shows the components of a convolution operation on a 2D input image with a kernel (Reproduced from [Olah, 2016])

$f[x, y]$ understood as the *input* and the $g[x, y]$ labelled as the *kernel*. This can be visualised with Figure 2.3

Much like the field of compressed sensing mentioned in Section 2.2.2 CNNs use prior knowledge of sparsity in systems to improve performance. CNNs leverage sparse interactions, parameters sharing and equivalent representations to boost performance of the model [Bengio and Courville, 2016]. To enjoy the benefits of sparsity, the kernel is often much smaller than the input. This helps control the number of parameters in the system, but the size of the kernel is a hyperparameter, which needs to be chosen by the practitioner.

The beauty of convolutions is that the output produced from convolving two functions together can itself be used as input for the next layer. This allows convolutional layers to be built that are using the previous layer's feature map as input into the next. Figure 2.4 is also helps one to visualise the layering process.

The hierarchical nature of these convolutional layers means features can be learned, without implicitly choosing them beforehand. This goes back to the original roots of artificial intelligence and neural networks in attempting to model the brain; it was shown in early work by [Hubel and Wiesel, 1959]

that hierarchical layers of feature maps in the visual cortex of the brain (in cats) are central to the processing of images.

2.3.1 Architectural Components

CNN architecture design is a very active area of research, with new architectures being proposed almost every month. However, there are common elements that form the building blocks of pretty much every CNN. The general components of a CNN often include the following:

1. Input layer.
2. Kernel or filter.
3. Feature map.
4. Activation function.
5. Pooling layer.
6. Fully connected layer.
7. Final output later.

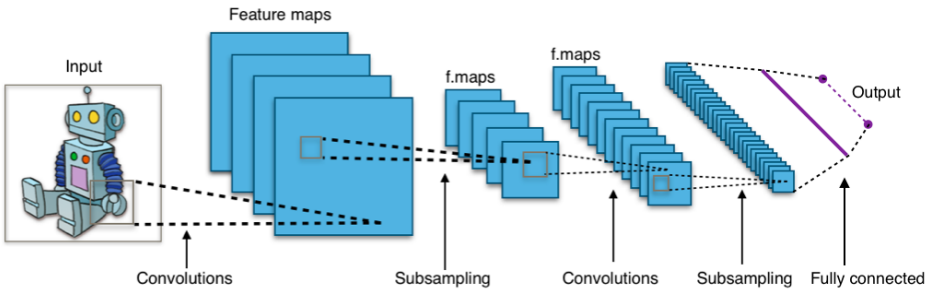


Figure 2.4: This figure shows the typical convolutional neural network architecture (Reproduced from [Wikipedia, the free encyclopedia, 2016])

It is typical to think of the combination of the convolution operation and non-linearity as a single *convolutional layer*. The activation function

is chosen by the machine learning practitioner. The most common choice today is a rectified linear unit (ReLU [Nair and Hinton, 2010]). There exist many other forms of activation functions such as sigmoid, tanh, or at a layerwise level maxout, to name a few but it has been shown that for deep learning, ReLU offers much better performance and faster convergence for CNNs [Krizhevsky et al., 2012]. The ReLU non-linearity can be represented by Equation 2.5.

$$f(x) = \max(0, x) \tag{2.5}$$

where x is the weighted sum of the inputs to a node.

If not using pooling, the output of the ReLU operation is then passed as input into the next layer. Depending on how many layers one wishes to implement, this process is repeated until the final layers, which are fully connected much like those of traditional neural networks. The output of the fully connected layers is used to compute a differentiable loss function whose gradient will be used for back propagation to update the parameters of the network. There is considerable choice as to which loss function one can use, however, depending on the type of problem one is trying to solve, there are some that are recommended.

If the desired output is a discrete value, the problem is defined as a classification problem [Prince, 2012]. In contrast, if one desires a continuous function, the problem is a regression problem. The problem of radio interferometric imaging is a continuous problem and so a suitable metric for regression is required. The most common loss function for regression problems is known as a Euclidean loss defined in Equation 2.6. This function is also known as the Residual Mean Squared Error [Karpathy, 2016] and calculates the difference in values from the data to the predictions. A discussion of some of the pitfalls relating to using Euclidean loss functions can be found in Chapter 5.

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2} \tag{2.6}$$

The loss function is then used to update the parameters of the network using a common algorithm called stochastic gradient descent (SGD) [LeCun et al., 1998, Ng, 2009]. Due to the time it would take to run the algorithm over a large training set, minibatch stochastic gradient descent is often used when training deep networks [Karpathy, 2016].

Chapter 3

Design & Methodology

This chapter describes the data that has been used in this project and outlines the computer resources and techniques used to conduct the analysis.

3.1 Super Resolution Neural Network: SRCNN

In Chapter 2, super resolution was discussed as an important technique in computer vision. More specifically, the paper by [Dong et al., 2014] presented a similar problem to that of radio interferometric image reconstruction where a low resolution image could be mapped to a high resolution image using a convolutional neural network. Chapter 2 also discussed the potential benefits CNNs offer regarding run time compared to current algorithms. The current work will explore this field of study further to investigate whether a CNN can be used to solve the problem of radio interferometric image reconstruction for use at the SKA.

The SRCNN pipeline is explained now in more detail along with how it might be used to build a network targeted at radio image reconstruction instead of natural images. As mentioned in Section 2.3.1 the general architecture of a CNN consists of common components: an input layer, a non-linearity function and a fully connected output layer. The SRCNN

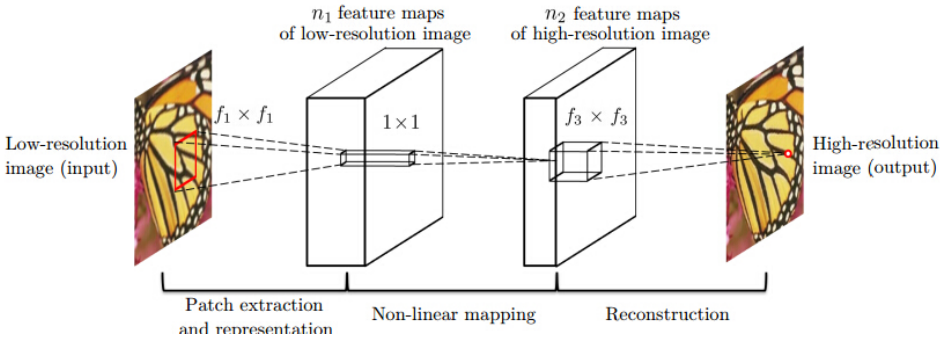


Figure 3.1: This figure shows the general architecture of the SRCNN (Reproduced from [Dong et al., 2014])

of [Dong et al., 2014] has this general structure with 3 main stages of formulation, patch extraction to create inputs, non-linear mapping via the convolution operator, and finally reconstruction via the output layer. In the first stage of patch extraction, overlapping patches are extracted from the low resolution image[1]. Each patch is then represented as a high dimensional vector which acts as the input to the network. This high dimensional vector is mapped to another high dimensional vector using the convolution operation discussed in the preceding chapter in Section 2.3. The aggregation of these vectors in the last stage produces a high resolution image as the output. The general SRCNN architecture can be seen in Figure 3.1.

The pipeline of computation can be understood with the following equations [Dong et al., 2014]. The paper denotes the low resolution image as the vector $\mathbf{Y}$ with the intention of outputting the continuous function, $F(\mathbf{Y})$, that is as close as one can get to the ground truth image, $\mathbf{X}$. A network architecture that has 3 layers is used, each with varying kernel sizes. The effect of the first layer can be expressed with the equation shown in Equation 3.1. This equation shows that the low resolution input image, $\mathbf{Y}$ is being convolved with a set of weights W_1 to produce a feature map, $F_1(\mathbf{Y})$. The following step can be expressed by Equation 3.2 which takes the re-

[1]Downscaling and then upscaling using bi-cubic interpolation creates a synthetic low-resolution image.

sulting feature map from stage one and convolves that with a different set of weights, W_2, to produce an output $F_2(\mathbf{Y})$. Finally, stage 3 repeats the process one more time, except without a non-linearity step to produce a continuous function, $F(\mathbf{Y})$ expressed in Equation 3.3

$$F_1(\mathbf{Y}) = \max(0, W_1 * \mathbf{Y} + B_1) \tag{3.1}$$

$$F_2(\mathbf{Y}) = \max(0, W_2 * F_1(\mathbf{Y}) + B_2) \tag{3.2}$$

$$F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3 \tag{3.3}$$

The continuous function $F(\mathbf{Y})$ is then compared to the ground truth image, $\mathbf{X}$, using the loss function, chosen to be a simple Euclidean loss $L(\Theta)$, with n signifying the number of training samples in a minibatch.

$$L(\Theta) = \frac{1}{n} \sum \|F(\mathbf{Y}_i, \Theta) - \mathbf{X}_i\|^2 \tag{3.4}$$

Following the comparison, the weights are updated using minibatch stochastic gradient descent with a momentum coefficient of 0.9 and a learning rate, η, the process is repeated until convergence.

$$\Delta_{i+1} = 0.9 \cdot \Delta_i - \eta \cdot \frac{\partial L}{\partial W_i^l} \tag{3.5}$$

The role of patch extraction helps with training the network through control of the number of parameters and control overfitting. (See Section 3.2.2 for a brief discussion on handling over fitting of data.) The patch extraction process creates a collection of 'sub-images' that are of much smaller size compared to the original input image. This controls the number of weights that are produced following the convolution with a kernel. This also allows one to create many 'sub-images' that can be used for training. It is known

that deep networks benefit from training with large datasets. The method of using patches to create a large dataset was used here but it will be discussed later in Chapter 5 how for further research one may want to consider using a full image as input rather than sub-images.

For this project, similar architectures will be be explored. The investigation hopes to discover whether CNNs have the potential to be used for radio interferometric image reconstruction. The sections that follow outlines the items used to investigate this problem.

3.2 Data

The data that has been used in this project has been gathered from two sources, ImageNet [Russakovsky et al., 2015] and the *zooinverse*[2] galaxy zoo project.

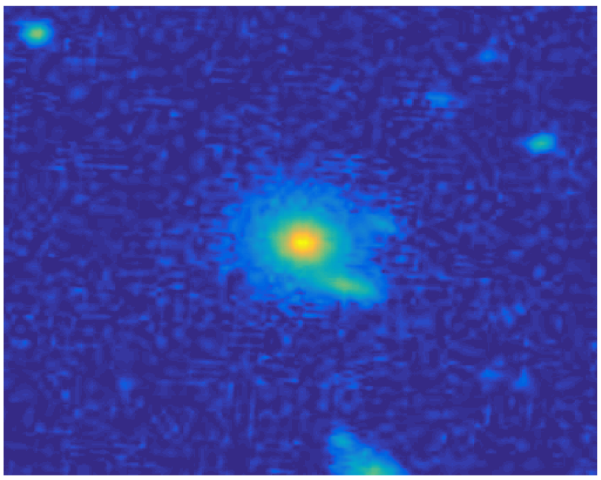
Ideally it would have been desirable to use a large dataset of astronomical images observed in the radio spectrum. However these were not available at the time of the work. Initial experiments used ImageNet data; it was later decided to use astronomical images observed in the optical spectrum as being somewhat closer to the visual structure expected in the radio images.[3]. However, in order to use optical images for training, some pre-processing is required.

3.2.1 Thresholding

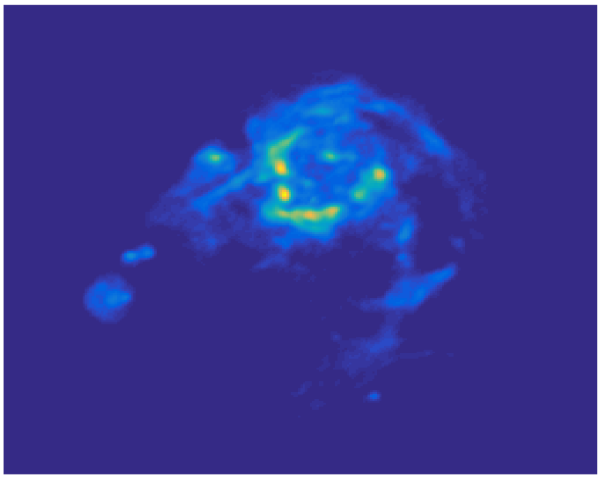
For the optical images to be more akin to radio images, one must perform a threshold of the originals to remove the excess *noise* that can be observed in the background. Shown in figure 3.2 is a comparison of an optical image

[2] `https://www.galaxyzoo.org/`

[3] The complete optical galaxy zoo dataset can be downloaded from `https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge/data`



(a) Randomly selected image from galaxy zoo dataset



(b) M31

Figure 3.2: Comparison of astronomical image observed in the optical domain (a) and with an astronomical image oberved in the radio domain (b)

chosen randomly from the dataset and a radio image observed at the VLA. It can be seen that the optical image contains more background information than that of the radio image.

If thresholding is not done, the model can learn the noise and as a result

does not handle real radio images well. Results from a model that has been trained on optical images that have not been pre-processed can be found in Chapter 4.

Thresholding the data is a sensitive task. Setting the threshold too high will result in too much information being set to zero, whereas setting the threshold too low results in a model that will learn the noise. With regard to choosing the correct level of threshold a domain expert, *Dr. Jason McEwen*, identified that a threshold in the range of 0.18 - 0.22 would be a suitable value to take.[4] To ensure this was appropriate preliminary tests were carried out to see how the signal-to-noise ratio (SNR), Equation 3.6 shown below, of dirty images and reconstructed images behaved when applying these thresholds.

$$SNR = 20 \log_{10} \left(\frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2} \right) \tag{3.6}$$

Above, $\boldsymbol{x}$ represents the original image whereas $\hat{\boldsymbol{x}}$ represents the reconstructed image. Shown in Figure 3.3 are the results of applying a threshold of 0.18, 0.20 and 0.22 to the dataset. It can be seen that the best performance with respect to SNR occurs with a threshold = 0.20.

3.2.2 Handling Overfitting and Underfitting

When using regression it is important to understand the subtleties of selecting the right model. The practitioner must be careful not to over fit the model to the training data. It might naively be thought that one can just train a model, and then just select the model which contains the parameters that give the lowest error. This is not a good idea since if one simply selects the model that gives the lowest error, there is a risk of selecting a model that overfits the data. In other words one will be selecting the model that best fits the training data, but would not be able to handle data that the model has not seen before.

[4] This was judged from results that can be found in Figure A.1 in Appendix A

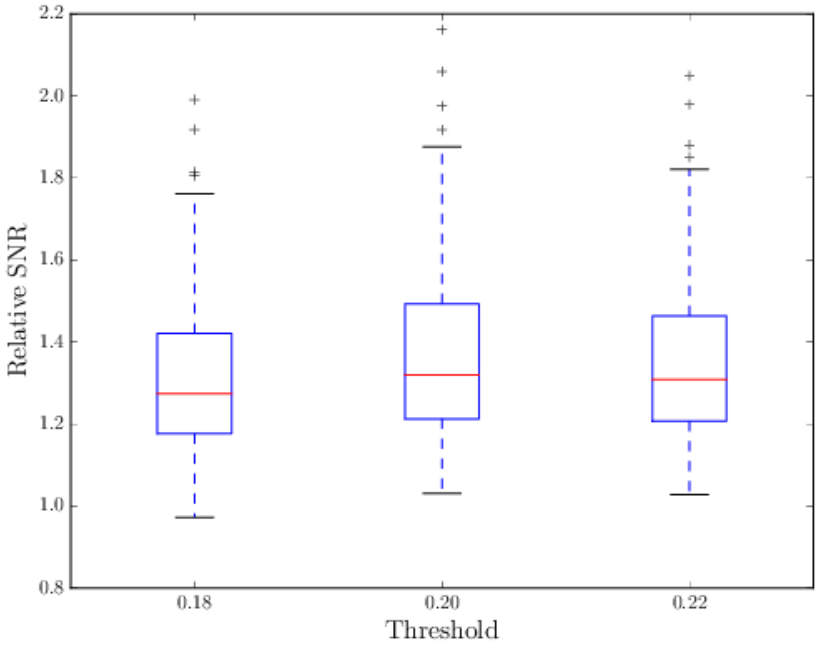


Figure 3.3: Relative SNR for the dirty image and the reconstructed image.

A technique that addresses this problem is *cross-validation* [Ng, 2009, Ivezić et al., 2014]. A simple approach to cross validation is to take the entire data set and to split it into 3 parts: the training set, the cross validation (or simply validation) set and the test set. It is usually the case that the training set should consist of 50-70% of the data, and the other two parts divided equally.

The *training set* determines the optimal weights for a given choice of architecture. The training set is then used to evaluate the training error between the input and the output by computing the Euclidean loss, Equation 3.4.

The *validation set* is used to evaluate the validation error of the model. Since the validation set was not used when fitting the model, it is a better representation of how well the model is doing. Thus, one would want to minimise the validation error in order to get a better model in general[Ivezić et al., 2014].

When the model has been chosen from the minimised validation error, the test error is evaluated using the *test set*. The results from this test error indicate how reliable the model is for the general case and for novel data. It is advisable to have both a validation set and test set because just as the weights can be learned from the training set, the hyperparameters can be learned from the validation set; because as we tweak them and look at the validation error, there is the risk that we are optimizing these parameters to suit the validation set.

3.3 Tools & Technology

3.3.1 Frameworks

Many frameworks exist today that abstract much of the complex implementation of deep learning algorithms away from the user, allowing the user to focus on applying the algorithm itself. It was felt appropriate to leverage the power of these mature software packages to conduct the experiments. Although the level of abstraction offered by frameworks aids development time, it was also desired to have a framework that can be manipulated at a low level should one be required to implement novel architectures and custom layers. To choose the most appropriate framework for the problem at hand, a deep learning framework review has been carried out. A screen shot of the review can be seen in Figure 3.4.

Following the review of the various frameworks that were available and after investigating the current literature around image reconstruction it was decided to use *Caffe* [Jia et al., 2014].

Caffe offers many pre-built layers but also allows custom layers to be built if required. Also, the relative maturity of the framework equates to a large community of developers and active forums. Another reason why Caffe was favoured over other frameworks is that it was the framework used for

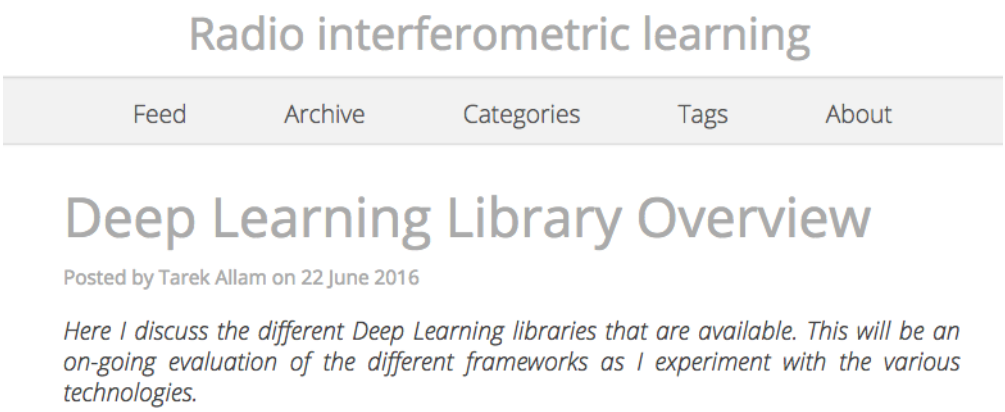


Figure 3.4: Screenshot of wiki post written on `www.astro-informatics.org`

developing SRCNN[5] by [Dong et al., 2014].

3.3.2 Computer Resources

A very important aspect to consider before running deep learning experiments is to see what computer resources one has available. It is understood that the use of a GPU can give considerable speed up when carrying out computationally expensive jobs [Krizhevsky et al., 2012]. To motivate the choice of a GPU over a CPU, a small test was carried out for simple, slightly modified, code provided by [Dong et al., 2014]. The results from this small experiment showed that for a network trained on a CPU for a maximum of 500 iterations, time to complete was 5 mins 35 seconds. In contrast, the same network trained on a GPU for the same number of iterations gave considerable speedup finishing in a time of 59 seconds[6]. With this in mind, it was decided to conduct the experiments here on GPUs. It was decided to configure a GPU Amazon instance to run the experiments (**g2.2xlarge** Amazon instance running on **Ubuntu** operating system has been used).[7]

[5] Code can be downloaded from `http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html`

[6] See `http://www.astro-informatics.org//wikis/radio_learning//posts/tutorials/2016/07/23/CPU-vs-GPU.html` for details of the test

[7] Configuration code and scripts can be found at `http://www.astro-informatics.org//wikis/radio_learning//posts/tutorials/2016/07/22/Caffe-on-AWS.html`

The specifications of the GPU used in conducting experiments is shown in the table below:

```
$ nvidia-smi -q | head


==============NVSMI LOG==============


Timestamp                           : Sat Jul 23 17:41:45 2016
Driver Version                      : 352.93


Attached GPUs                       : 1
GPU 0000:00:03.0
    Product Name                    : GRID K520
    Product Brand                   : Grid
```

```
$ uname -a
Linux ip-172-31-19-64 3.13.0-92-generic #139-Ubuntu \
SMP Tue Jun 28 20:42:26 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
```

3.4 Modeling Visibilities

As defined in Section 2.1 a visibility is a fundamental concept with regards to radio interferometry. A visibility describes the response of the telescope in relation to the observed image distribution on the sky. In order to model visibilities, the software package `PURIFY` has been used [Carrillo et al., 2014]. To model the measurement process of an interferometric telescope a Fourier transform of the ground truth image, $\mathbf{X}$, is done and then a binary mask is applied to the result. The combined Fourier transform and multiplication by a mask is represented by Φ in Equation 2.2. The observations of the telescope is defined by the mask, in a sense it describes where measurements are taken in the Fourier plane. In our simulations the mask is modelled

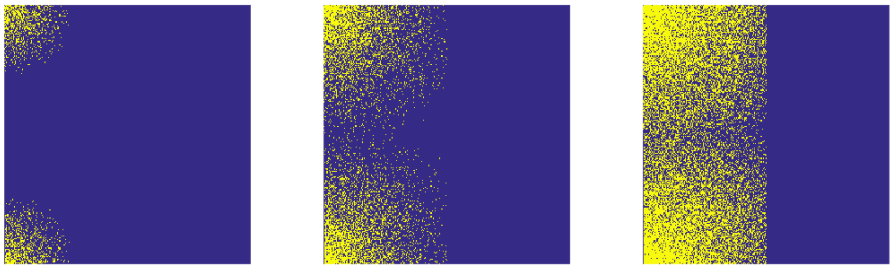


Figure 3.5: This figure gives a visual representation the matrix **M** with varying PI values which defines the telescope configuration. PI = 0.05 (left), PI = 0.20 (middle), PI = 0.50 (right). *Note, the origin x=0 and y=0 are in the top left of the image.*

using the statistical properties of a Gaussian, since it is expected that more measurements would be gather at lower frequencies than higher ones. Examples of binary mask 2D matrices can be seen in Figure 3.5 but it should be noted that because the signal that is received is real, there is a conjugate symmetry relationship in the Fourier space. The Fourier co-efficient for the $-x$ coordinate is the complex conjugate of the Fourier co-efficient for $+x$ coordinate, which means when positive x is measured, there is no need to measure negative x because it is effectively already measured.

A parameter that decides the type of visibilities one observes in relation to the mask is the proportion of image sampled value, or simply the PI-value. The PI-value is the proportion of mask samples one wants to use. For example, say a mask is set and applied to an image, the PI-value will use x proportion of those samples that have been taken. The combination of the mask and the PI-value uniquely identify the configuration of the telescope. It will be discussed later in Chapter 4 what are the significances of allowing these values to vary during training and validation.

`PURIFY` is itself an algorithm that aims to recover ground truth images from dirty images; however, for this project, only the code to simulate visibilities have been used. `PURIFY` uses SNR as the quality metric, so in keeping

with the literature, SNR is used here also. It is computed using the equation given earlier, Equation 3.6.

3.5 Network Architecture and Hyperparameters

With regards to the network structure, much of SRCNN remains the same as used by [Dong et al., 2014]. This was done so that a foundational test could be carried out to see whether the network proposed by [Dong et al., 2014] could work for radio images. Choices made include the kernel sizes, depth of feature maps, learning rate and learning policy (for example, whether the learning rate is decremented or weights made to decay). It was shown by [Dong et al., 2014] that increasing kernel sizes improved performance at the cost of training time. This issue will be investigated also with results and discussion in the following chapters.

3.6 Workflow

From the discussion in Section 3.2.2, it has been decided to use the hold-out cross validation method with a dataset of 100 images split into 60 training, 20 validation, and 20 test. Although this seems like a small dataset, due to the patch extraction process described in Section 3.1, the actual number of training and validation examples is $\sim$ 70,000 and $\sim$ 30,000 respectively. The model is trained using mini-batch SGD. Code from [Dong et al., 2015] has been used to formulate the patches and create HD5 files that can be used by Caffe.

Caffe uses *prototxt* files to define the network and to tell the network what hyperparameters to use. Developed by Google, prototxt files can be thought of as JSON files that are used to help serialise data. Within these files, users can define which datasets take part in which layers; this allows one to specify a training phase, which uses the training data, and validation

phase (called test phase in Caffe) that uses the validation dataset. The difference between the training phase and the validation phase is that the training phase performs a forward pass and backward pass of the network whereas the validation phase only performs a forward pass.

An example prototxt file that defines the SRCNN can be found in the Appendix C. Below is an example of the `solver.prototxt` which looks for a network definition and sets certain hyperparameters ready to be used by the Caffe binary to begin training.

```
# The train/test net protocol buffer definition
net: "RICNN/model/955-gz-optical-91-wt/RICNN_net.prototxt"
test_iter: 556
# Carry out testing every 500 training iterations.
test_interval: 500
# The base learning rate, momentum and the weight decay of the network.
base_lr: 0.00001
momentum: 0.9
weight_decay: 0
# The learning rate policy
lr_policy: "fixed"
# Display every 100 iterations
display: 100
# The maximum number of iterations
max_iter: 200000
# snapshot intermediate results
snapshot: 500
snapshot_prefix: "RICNN/model/955-gz-optical-91-wt/RICNN"
# solver mode: CPU or GPU
solver_mode: GPU
#solver_mode: CPU
```

Note the hyperparameters that have been defined; here we are specifying the path to where the network is defined:

`RICNN/model/955-gz-optical-91-wt/RICNN_net.prototxt`.

We then specify how often to validate the model. A base learning rate of 0.00001 is used along with a momentum of 0.9. We do not set a weight decay here but this can be set in relation to a different learning rate policy such as *step* or *inv*. Since this file relates purely to optimization of the model and tweaking hyperparameters, this will not be changed except for 'maximum iterations' of which defines how many forward and backward passes the network should be trained for.

When the `net.prototxt` file and `solver.prototxt` files are complete, these can be sent as arguments to the Caffe binary using the following command:

```
time ./build/tools/caffe train --solver \
RICNN/model/955-gz-optical-91-wt/RICNN_solver.prototxt -gpu 0 \
2>&1 | tee logs.log
```

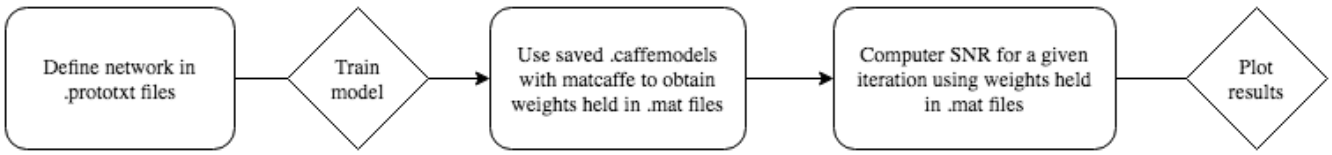
The above utilises the UNIX command `tee` which sends the standard output to a file as well as to the screen. The file that it is sent to can be processed using the file `parse_logs.py` that comes with the Caffe repository[8]. This produces two files, a training log and a validation log, which gives the values for the loss as a function of iteration.

The quantity that is of most interest to the user of the system is SNR; however this is not suitable as a training metric. A compromise is reached by monitoring SNR for the training and validation data with best on perfor-

[8] A bespoke `AWK` program has been written to do the same job, this is documented at `http://www.astro-informatics.org//wikis/radio_learning//posts/tutorials/2016/08/02/Caffe-cleanup.html`

mance on validation being defined on SNR. In order to do this, the weights for each iteration are needed.

In `solver.prototxt` is an option for *snapshotting* the model at a given iteration. This snapshot captures all information about the model for that iteration and saves this information to a `.caffemodel` binary file. To extract the information regarding the weights and biases for this particular model *matcaffe* has been used. Matcaffe is a `MATLAB` interface for Caffe. Using this interface one can obtain the weights and biases for a model corresponding to a particular iteration and save the results to `.mat` files. Having the weights and biases saved in .mat files allows one to compute the SNR for images in the validation set at a given iteration. The diagram below shows the typical workflow when constructing a new model.



Chapter 4

Results

4.1 Preliminary Results Using ImageNet

As discussed in Chapter 3, the initial experiments were heavily influenced by the work carried out in [Dong et al., 2014]. The SRCNN was trained on 'natural images' from the ImageNet dataset [Russakovsky et al., 2015]. These are everyday images such as animals, cars and other objects. SRCNN synthetically creates low resolution images by downsampling and then up-scaling via bi-cubic interpolation. In the context of radio imaging, the low resolution image is caused by the way the telescope captures information. Hence our version of the low resolution image is a 'dirty image' created to emulate the process of observation through a radio telescope (where the level of dirtiness is determined by the PI-value, see Figure 3.5 for details). As an initial test, it was of interest to see how a network that has been trained on natural images with synthetically rendered low resolution inputs via bi-cubic interpolation could handle natural images but in the above different form of low resolution.

Figure 4.1 shows a randomly selected image from the validation set from SRCNN, but with the 'dirty image' pre-processing method, applied using the algorithm from `PURIFY` in B. Further examples using the 'dirty image' pre-

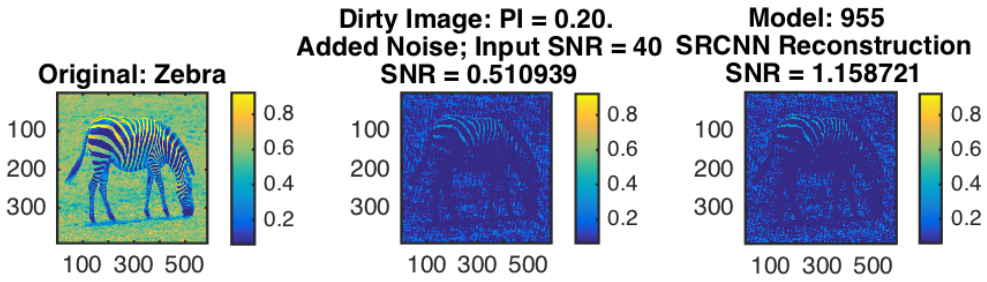


Figure 4.1: Image taken from validation set of SRCNN, but converted to low resolution of a dirty image instead of by bi-cubic interpolation.

processing method of other 'natural images' not in the SRCNN validation set can be seen in Figure 4.2.

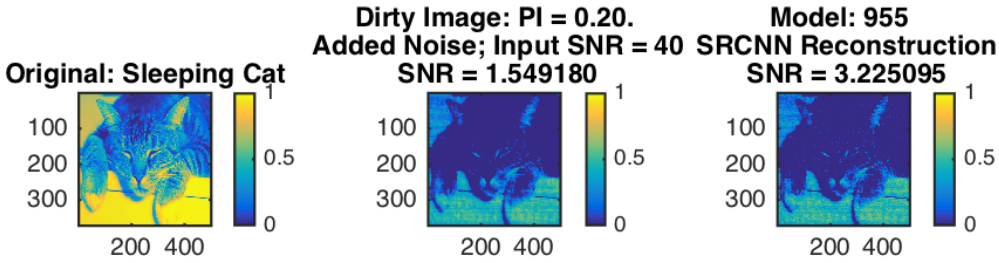
In Figure 4.3 are the results for inputting a dirty astronomical input into the SRCNN network which has been trained on ImageNet.

Although it may be visually difficult to see in the figures shown here, the network was able to to satisfactorily boost the SNR of this different pre-processed low resolution image, giving hope that similar improvement could be seen where astronomical images are provided as training data and as inputs.

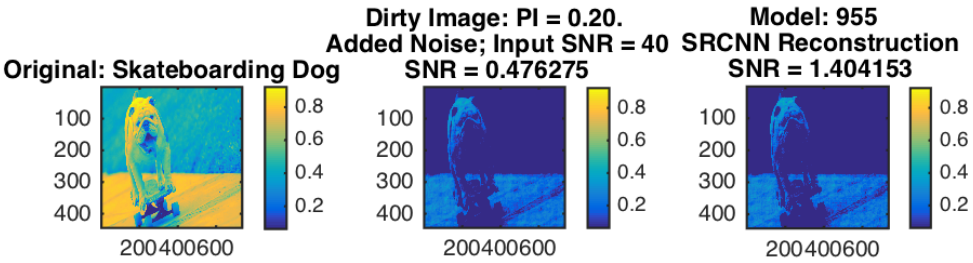
4.2 Astronomically Trained Networks

4.2.1 Single Telescope Observation

If PI-value and the mask are fixed throughout training, then this physically relates to a telescope of a certain configuration for a single observation. This is the easiest model to deal with, so initial experiments were conducted using this setting. As discussed in the previous chapter, the SRCNN was used as a



(a) A test image similar to ones found in ImageNet

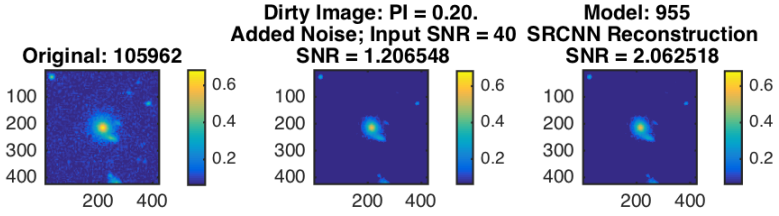


(b) Another image similar to those in ImageNet

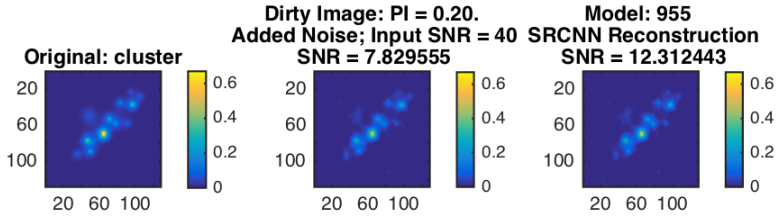
Figure 4.2: Randomly selected natural images used to test how SRCNN handles a different low resolution synthesising method.

starting point for trying to solve the ill-posed problem of radio interferometric image reconstruction.

Following the preliminary tests carried out in the previous section 4.1 it was decided to progress towards training the SRCNN on astronomical images using a pre-processing method akin to that which would be carried out when using radio telescopes. The SRCNN has been shown to have better performance with larger kernel sizes, more specifically the network architecture of 9-5-5 was shown the have the best results [Dong et al., 2014]. However when running convolutional neural networks, small changes in kernel size can have



(a) Random astronomical image taken from the *galaxyzoo* dataset that has been dirtied



(b) Cluster

Figure 4.3: Randomly selected astronomical images used to test how SRCNN handles a different low resolution synthesising method.

a drastic effect on training time. This is due to the increase in the number of parameters involved when one chooses to increase a kernel size. Therefore, it was decided to first train the network on the initial network architecture proposed in [Dong et al., 2014] which was 9-1-5. This would then later be compared to the 9-5-5 to see the difference in performance training time required. All experiments are carried out using the training and validation datasets described in Section 3.2.

In Figure 4.4, using the 9-1-5 architecture, it can be seen that there is a period of erratic training and validation error but that at around 40,000 iterations, a sudden drop in both training and validation error can be observed. At the same point, validation SNR rises sharply and maintains a

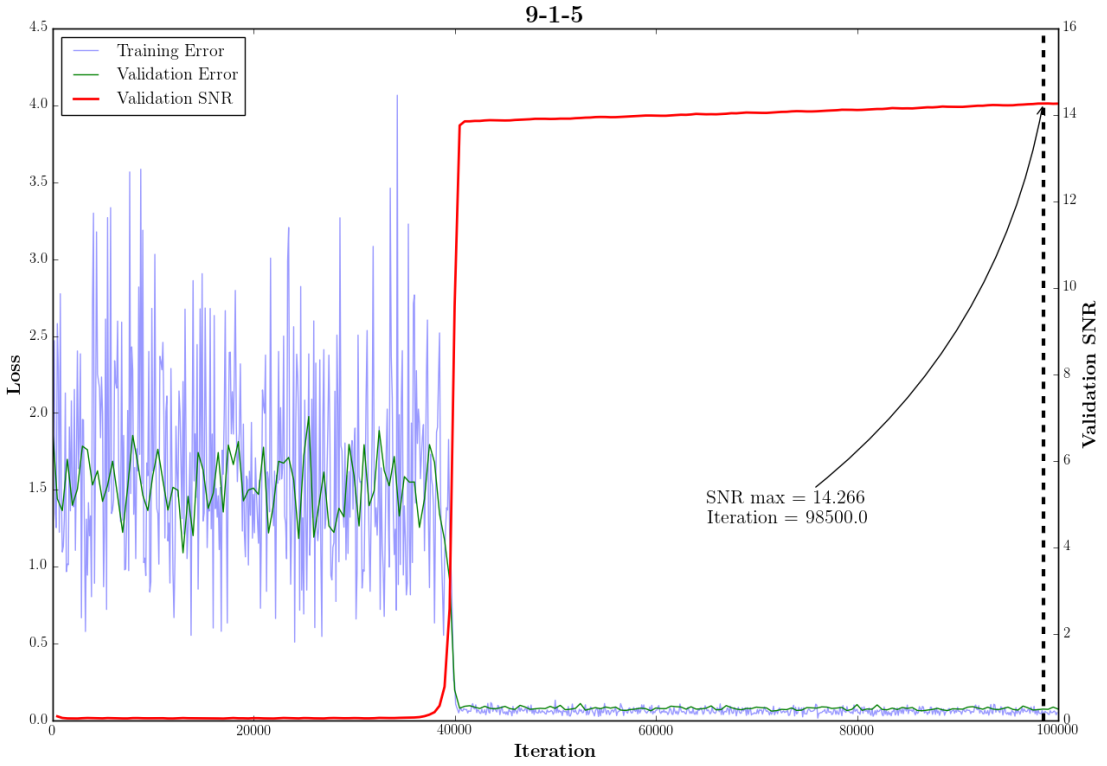


Figure 4.4: This figure shows how a 9-1-5 architecture trained with PI = 0.2 and mask set performs as a function of iterations, for 100,000 iterations.

steady increase in value. It can be seen that the maximum validation SNR, over this image set, is 14.266 at iteration 98500. In Figure 4.5, the same pattern emerges, but training is faster. A higher validation SNR value of 14.607 is also achieved, with the maximum SNR occurring at iteration 100,000, and with no apparent plateau, this suggests that further increases in SNR could be achieved if one were to continue training such an architecture for longer. This was indeed carried out for a further 200,000 iterations, taking the total to 300,000 iterations. The results of this can be seen in Figure 4.6

Figure 4.6 shows how the SNR did indeed improve with further iterations but seemed to plateau between 250,000 and 300,000. Because of this, it was decided to keep the maximum number of iterations constant at 300,000 for

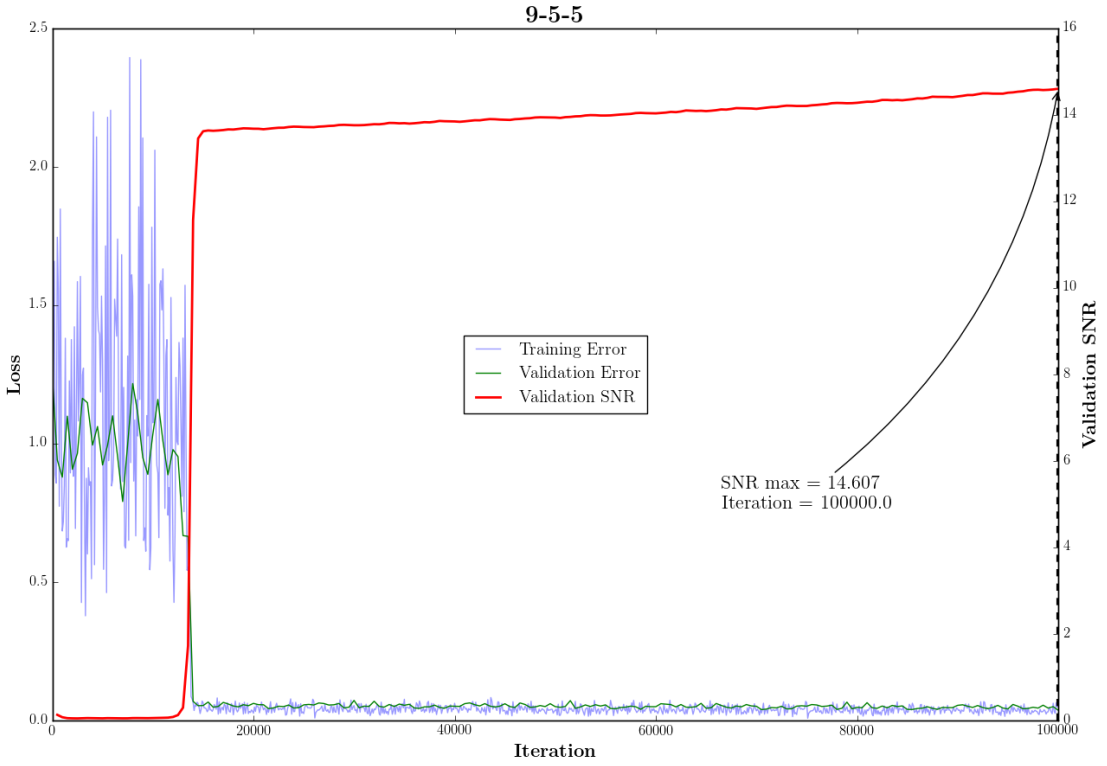


Figure 4.5: This figure shows how a 9-5-5 architecture performs as a function of iterations, for 100,000 iterations.

all further. The results of increased training time for the 9-5-5 architecture is shown in Figure 4.6. A further experiment, with architecture 9-9-5, was conducted but the time taken to train this model however was considerably slower due to the increase in parameters and achieve only a small increase in maximum SNR. As a result of this, it was decided to carry out investigations with the 9-5-5 architecture and not increase the kernel size further due to constraints on time (See Figure 4.7 for the results of training a 9-9-5 architecture).

The results presented here already show promising results suggesting that a CNN can be trained to reconstruct radio images using only dirty images as input and no other priors. The network learns the filters it needs to

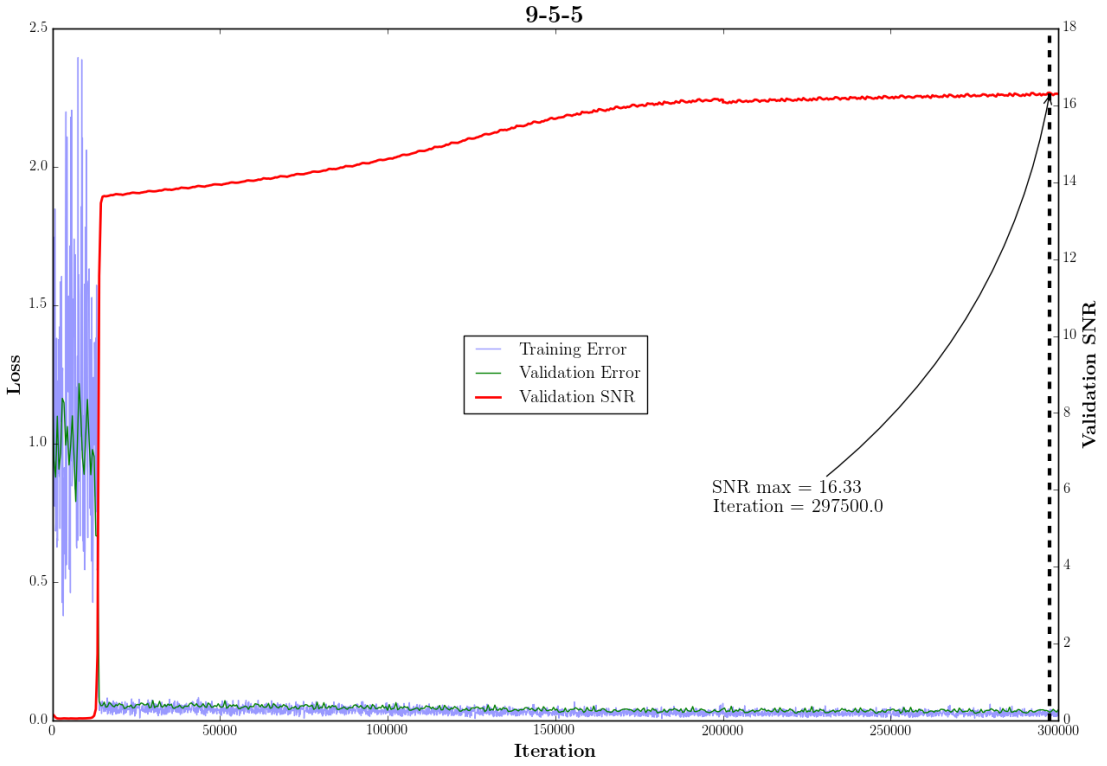


Figure 4.6: This figure shows how a 9-5-5 architecture trained with PI-value = 0.2 and a set mask throughout performs as a function of iterations, for 300,000 iterations.

produce the continuous function $F(\boldsymbol{Y})$. The setting used in these methods represent a single telescope configuration for a single observation. Hence we know at this point that fair results could be obtained using a model specifically trained for a particular telescope. Ideally one would like to create a model that allows the mask to vary as this is the more realistic setting of taking different observations with a single telescope. Obtaining such a model that is then trained for a specific telescope would be satisfactory but one wonders if there is a possibility one could train a network that is completely general in that it could be then deployed for any telescope configuration. As described in Chapter 2 the configuration of the telescope is set by the mask

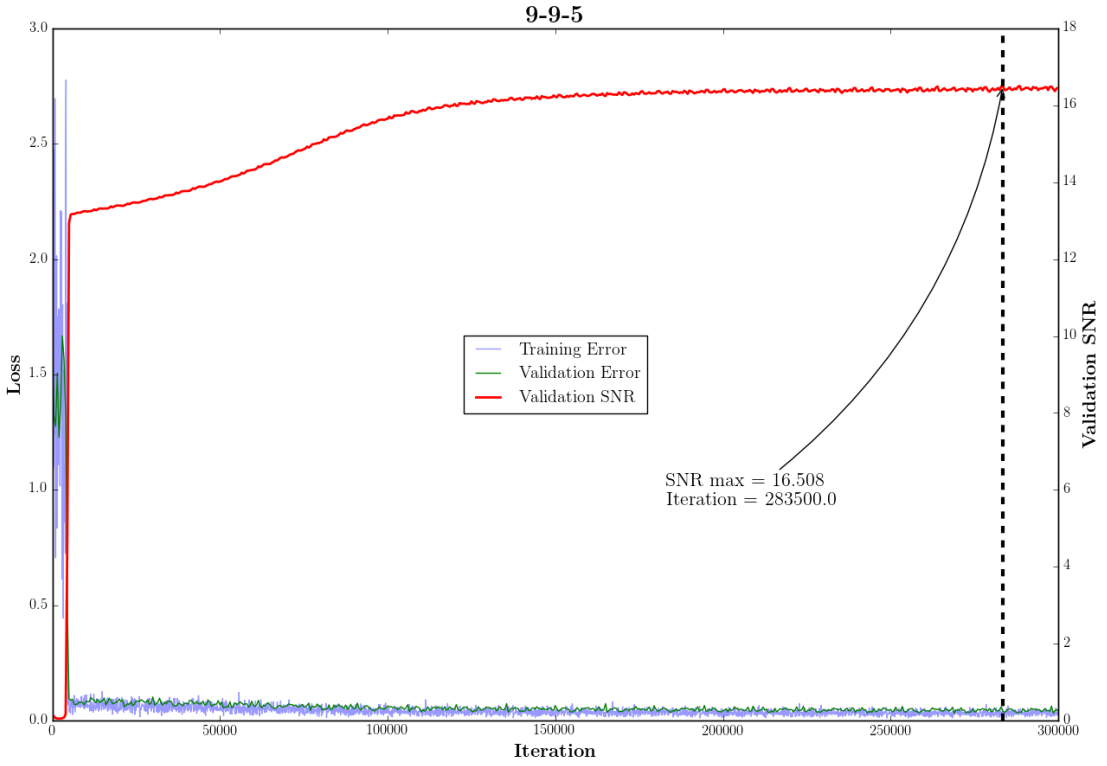


Figure 4.7: This figure shows how a 9-9-5 architecture performs as a function of iterations, for 300,000 iterations trained with a fixed value for PI = 0.2 and a set mask.

and the corresponding PI-value. Further investigations will be carried out, with results in Section 4.2.2, to see if a general network can indeed be trained that can then be deployed to telescopes where the mask and the PI-value are different.

4.2.2 Multiple Telescope Observations

As suggested in the previous section, if a training methodology could be devised such that the network's performance remains the same or even improves with different configurations of telescope, this would imply that one could train a single network that could be deployed for any telescope. To investi-

gate this, the network architecture of 9-5-5 was trained now with the mask varying throughout the training. The results of training such a model can be seen in Figure 4.8. Although noise is added, the setting used here better represents how telescopes physically make observations. Having a fixed value for PI-value but varying mask physically represents a single telescope configuration making different observations of the sky. Here a maximum SNR of 16.189 is recorded. This is very close to the maximum value observed with the same network but trained with a fixed mask throughout training. This is very interesting.

Continuing this investigation a final model was trained that allows both mask and PI-value to vary throughout training. This setting represents telescopes with different configurations making different observations i.e. the complete general case. The results can be seen in Figure 4.9. The model again produces a similar value to the model given in Figure 4.6 suggesting that a universal network could be produced.

4.3 User Deployment Investigations

It is of great interest to see how these models would cope should they be deployed. The following plots show for different models the relative and absolute SNR for validating on different PI-values and with varying mask.

Figure 4.10 shows the expected improvement in performances as PI-value increases since higher PI-value equates to more measurements being taken. There is no big drop off with PI = 0.15 and 0.3 performing well. Although this setting is not so useful as it is for a single telescope with a particular observation, it can be seen for a model could be trained for a single telescope configuration and still have fair performance upon deployment.

As we move progressively to a more realistic setting, Figure 4.11, shows the results for a single telescope configuration but for multiple observations. It can be seen that there is a small improvement but also less variability with

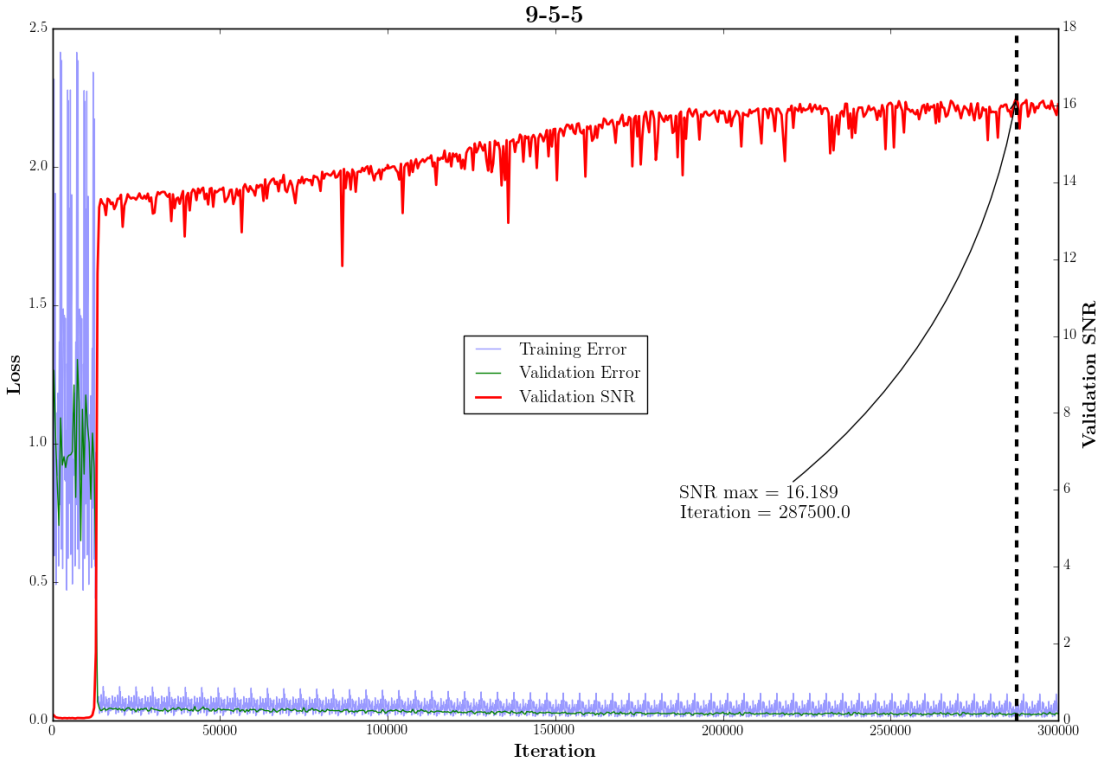


Figure 4.8: This figure shows how a 9-5-5 architecture trained with PI-value = 0.2 throughout but the mask being allowed to vary performs as a function of iterations, for 300,000 iterations.

each boxplot. The variability could be understood since we are validating on a model that we have trained on.

Figure 4.12 shows the general case that considers multiple telescope configurations making different observations. In this instance the performance is slightly worse, although for high PI values there is less variability in the plots. For this setting, we have made training much more difficult and complicated and actually seems better if we do not let PI to vary in the training, suggesting one could get better results training a network for a specific telescope configuration and then deploying this to other telescopes. It can be seen that good results are achieved for training on specific PI and then ap-

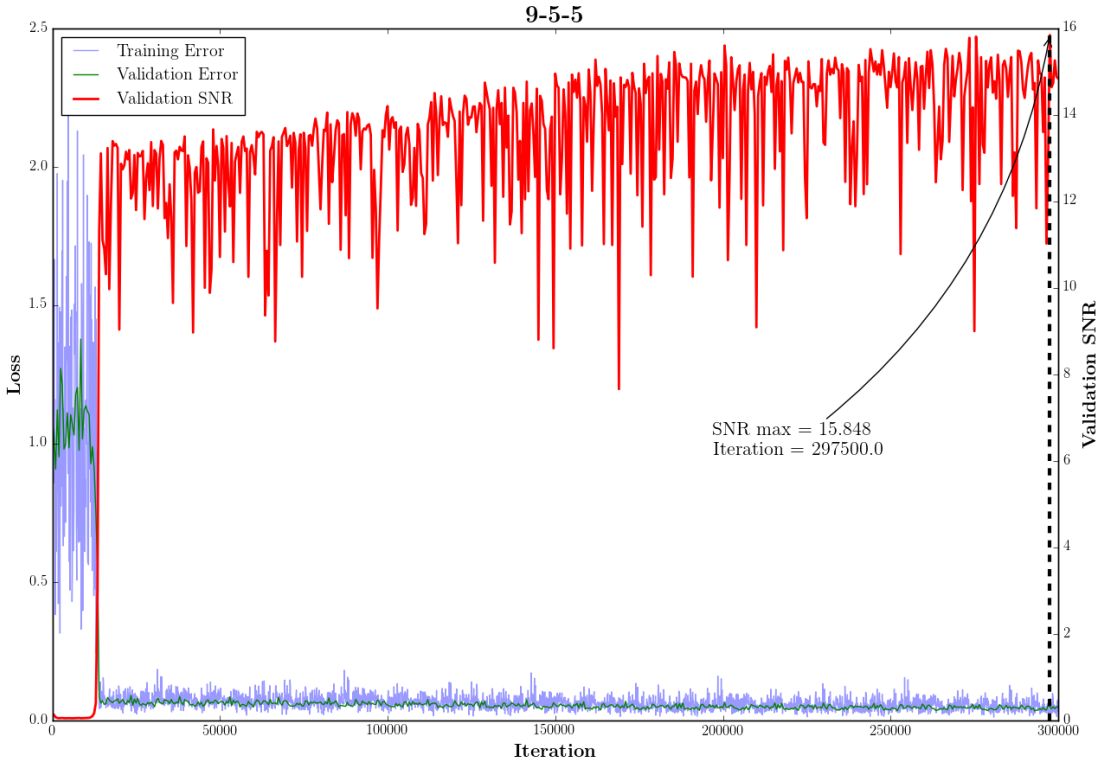
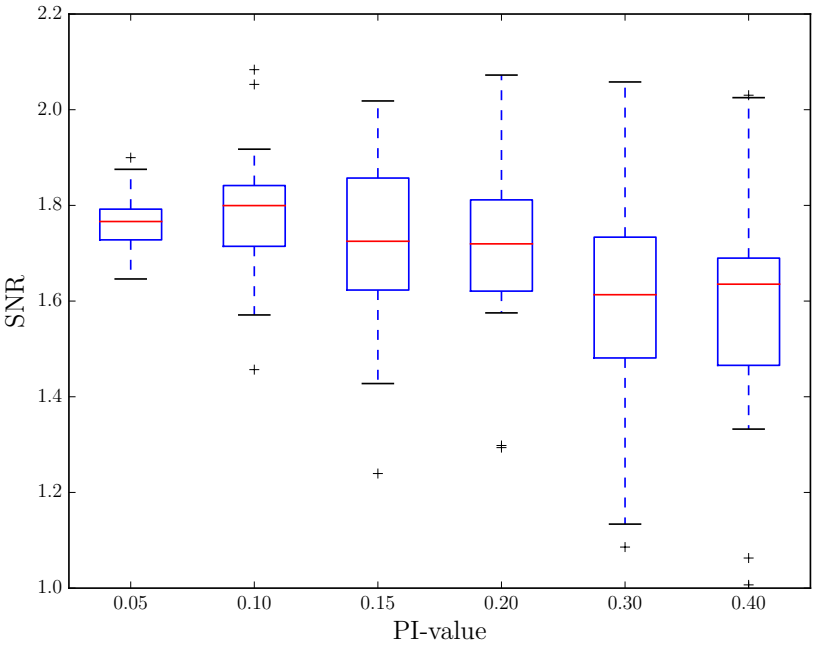


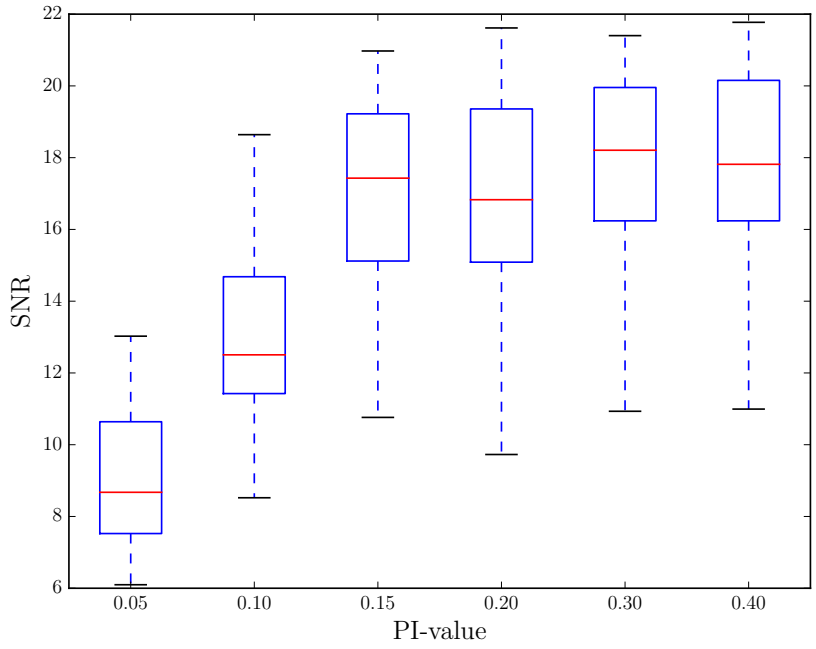
Figure 4.9: This figure shows how a 9-5-5 architecture trained with varying PI-value and varying mask performs as a function of iterations, for 300,000 iterations.

plying that model to other PI-values (other telescopes).

After exploring the various physical observing strategies it can be reasoned that for best results one might rather train for a single telescope configuration, however, promising results are still shown for the general case.

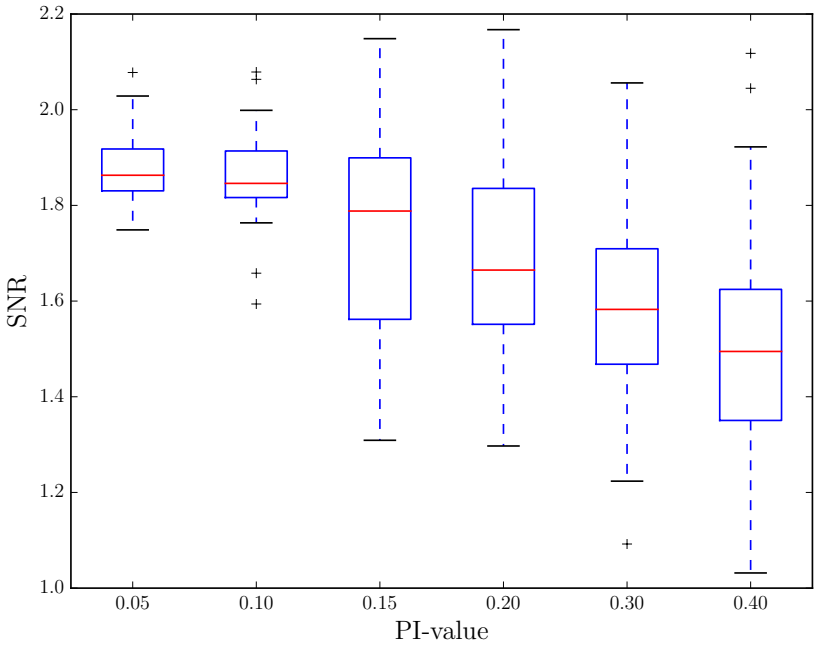


(a) Relative SNR

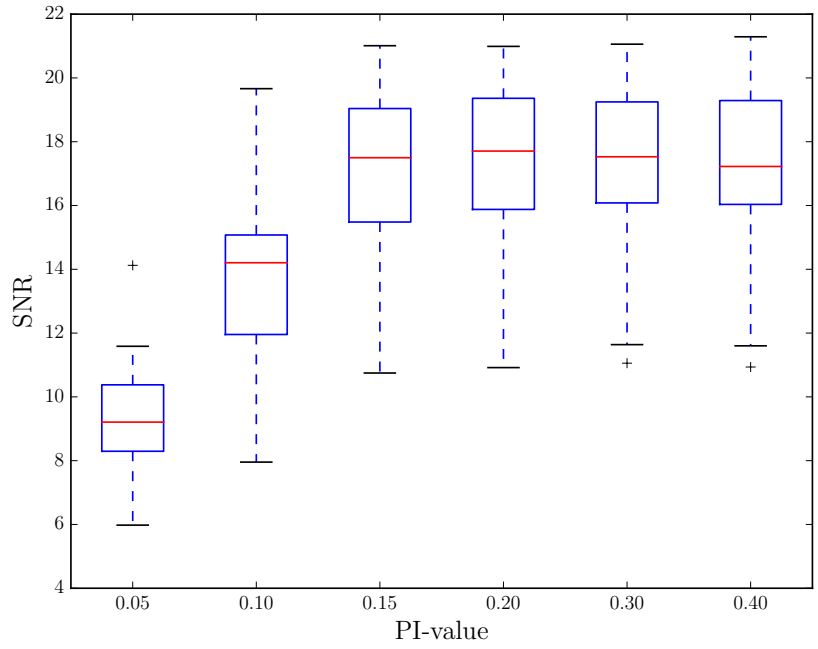


(b) Absolute SNR

Figure 4.10: Relative SNR and absolute SNR for a setting of fixed PI-value = 0.2 and set mask in training

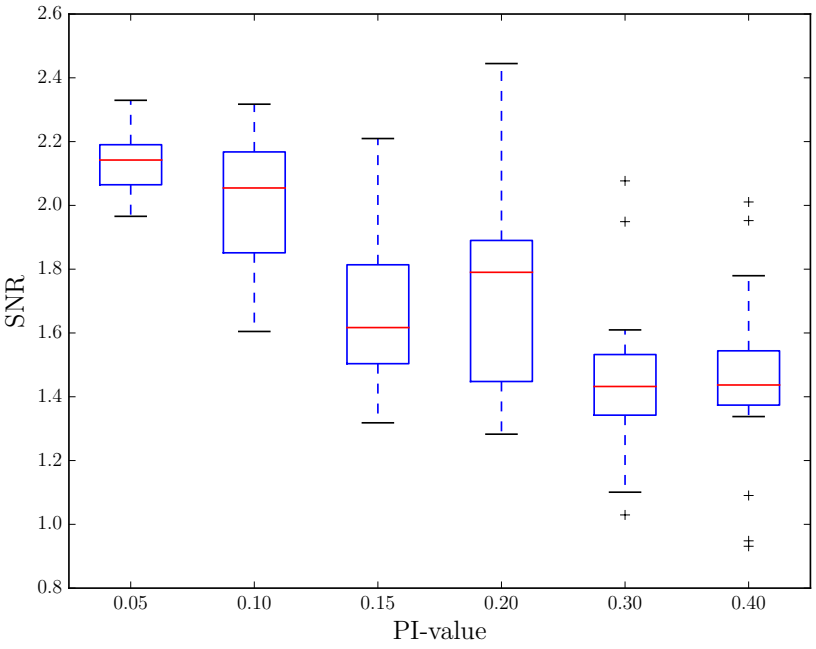


(a) Relative SNR

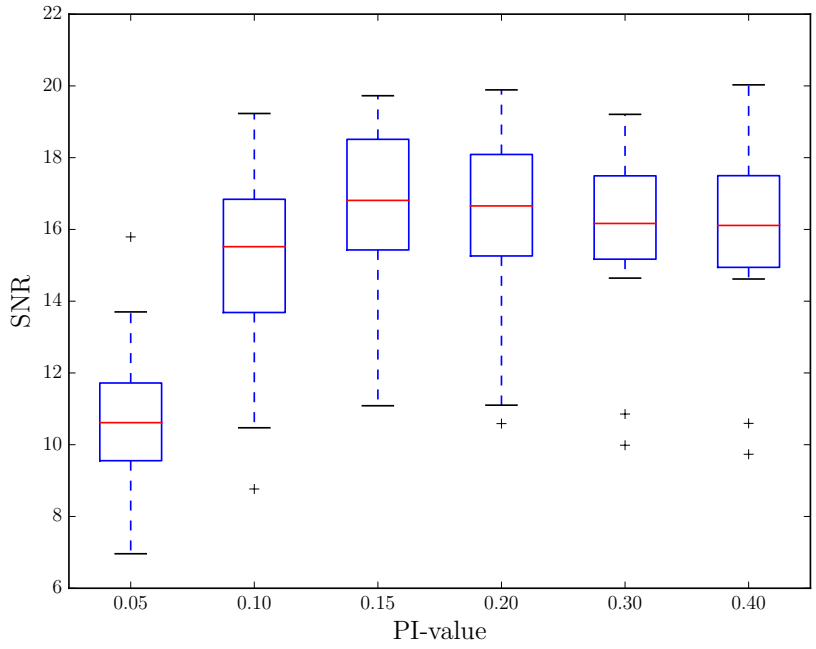


(b) Absolute SNR

Figure 4.11: Relative SNR and absolute SNR for a setting of fixed PI-value = 0.2 but mask allowed to vary in training



(a) Relative SNR



(b) Absolute SNR

Figure 4.12: Relative SNR and absolute SNR for a setting of both PI-value and the mask being allowed to vary in training

Chapter 5

Discussion

5.1 State of the Art Comparisons

The performances of the networks that have been trained here are close to those of the most often used algorithms of today. Refer to Figure 5.1 [Carrillo et al., 2014], a comparative study of modern radio interferometric image reconstruction algorithms. The figure shows the SNR for increasing PI-value (labelled as M/N in the figure). Although the SNR values for the CNN models do not match up to the best performing algorithms such as BPSA or SARA it is felt that with sufficient training time and with a training set of radio images, a CNN could be developed that not only outperforms the state-of-the-art regarding SNR but would also have a response time in usage un-matched by todays methods. The benefits of a CNN derive from the fact that a CNN is constant in response time, $\mathcal{O}(1)$ for a given input size. Compare this to the modern algorithms shown in Figure 5.1 which take considerably longer when the PI-value increases; a CNN would not be affected by this and would be able to respond up to 1000 times faster. This kind of speedup is exactly what is required for future radio interferometric telescopes such as the SKA mentioned in Chapter 2

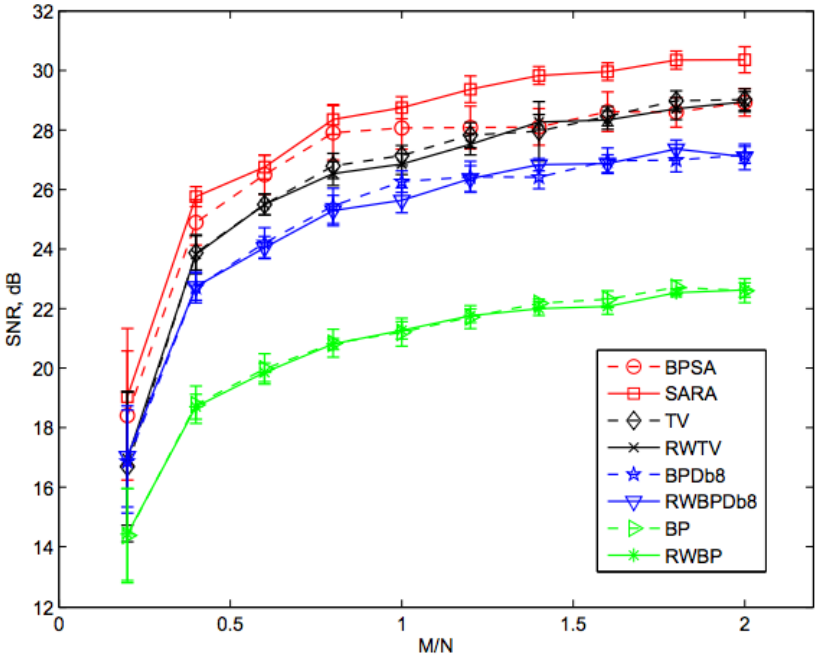


Figure 5.1: Comparison of various radio interferometric imaging algorithms for increasing PI-values, given as M/N here. (Reproduced from [Carrillo et al., 2014]).

5.2 Further Work

The work carried out in this project is by no means complete; due to the nature of research there are always further avenues one can explore. Although good results have been achieved using the simple architectures outlined by [Dong et al., 2014] it is felt that there are other architectures that could be used that might provide even better reconstructive performance. Due to the way radio telescopes take measurements, there is actually information about the entire image at each Fourier co-efficient. This would suggest that a more robust method would be to not use patches and sub-images but to instead use the entire image as input. The issues with this relate to an explosion of parameters but alternate architectures as discussed below could potentially control this problem.

5.2.1 Deconvolutional Layers

One such method that would allow one to use complete images would be to introduce a deconvolutional layer to the network. The deconvolutional layer effectively works like the convolutional layer but in reverse. The effect of such a layer is that instead of reducing the size of the image after a convolution with the kernel, the size now increases. This would allow one to have convolutional layers that shrink the size of the original radio image and then use deconvolutional layers to restore it again. Work conducted by [Xu et al., 2014] and by [Noh et al., 2015] shows how image reconstruction (pixel-wise prediction in the case of [Noh et al., 2015]) can be achieved using such networks. Furthermore, work by [Zeiler et al., 2010] shows that using deconvolutional techniques, the network can learn mid to high level features unsupervised, which is what is required for radio imaging.

5.2.2 Max Pooling and Unpooling

Another layer that could be introduced is a pooling layer. It was mentioned in Chapter 2.3.1 that a common feature of a CNN is a pooling layer. The purpose of such a layer is to introduce spatial invariance since only the maximum value is carried forward. Another effect of this is that it helps control overfitting by reducing the number of parameters. These two things would allow for the use of an entire image as an input and move away from patches and sub-images. Work carried out by [Zeiler et al., 2011] show promising results regarding reconstruction using a combination of 3D max pooling and unpooling.

Work by [Noh et al., 2015] looks at combining these methods of deconvolutions and max pooling and unpooling for image segmentation. The results presented in [Noh et al., 2015] show good results in relation to pixel-wise classification; this might then be able to be modified to suit the problem of radio interferometric image reconstruction. A visual representation of the

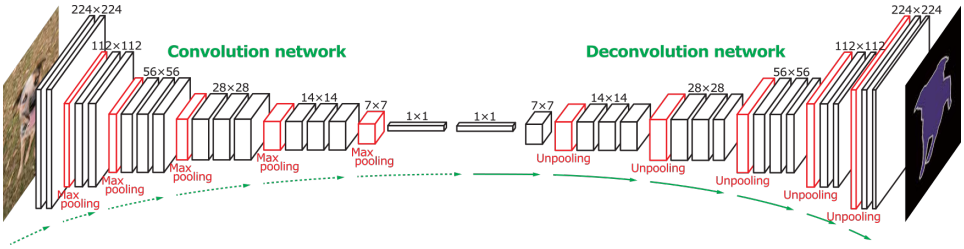


Figure 5.2: This figure shows the network architecture used that combines both deconvolutional layers and pooling layers (Reproduced from [Noh et al., 2015])

network can be seen in Figure 5.2. It should be noted that in order to be able to carry out these methods, one is required to have a large amount of data to prevent overfitting.

5.2.3 Bespoke Loss Function

The difference between an observed target and a predicted target in a regression analysis is known as the residual and is a measure of model accuracy. Throughout this project, Euclidean loss has been used as the residual for determining the weight updates. However as can be seen in Figure 5.3 using Euclidean loss can give surprising results; these images might be judged by a human observer to have different values while their Euclidean loss is the same. One might consider the development of a custom loss function or perhaps engineer a way to train on the SNR itself since this is the main quality metric that is being used when models are compared. This could for example by switching to a training method that did not require a differentiable metric, such as particle swarm optimisation (PSO).

5.3 Conclusions

From the investigations carried out for this project, it appears that constructing a convolutional neural network for radio interferometric image reconstruction is a very promising way forward. It has been shown that an

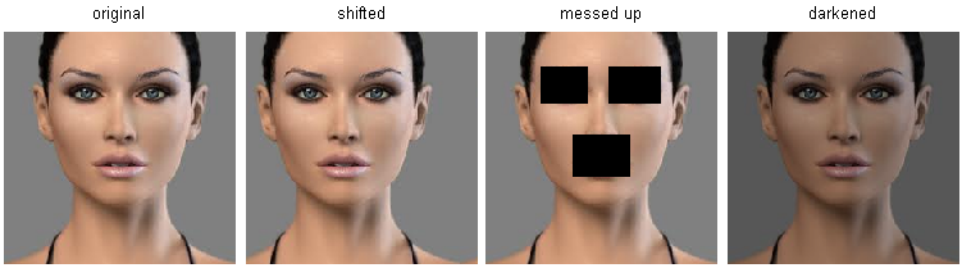


Figure 5.3: This figure compares different images that all have the same Euclidean loss shown in equation 3.4 (Reproduced from [Karpathy, 2016])

improvement of SNR is possible, albeit not to the same level as some of the state-of-the-art algorithms of today, but an improvement nonetheless. It is felt that with further research into the CNN technique, one could very well out-perform the current algorithms of today. One aspect where this technique already shows significant benefit over other techniques is the time taken to reconstruct an image, being greatly reduced by many orders of magnitude compared to other algorithms. It is also felt that one could in future explore architectures that move away from patch extraction and allow the model to be trained on the entire image. As explained in Chapter 2, the Fourier components hold information about all spatial elements of the image, therefore it can be reasoned that it would be to be beneficial to train a model with an input that is the entire image, instead of multiple small patches.

It has also been shown that one could possibly develop a single model that encompasses all configurations of telescopes. The implications for this would mean one could develop an algorithm that could rival CLEAN for longevity but completely overhaul it in terms of performance.

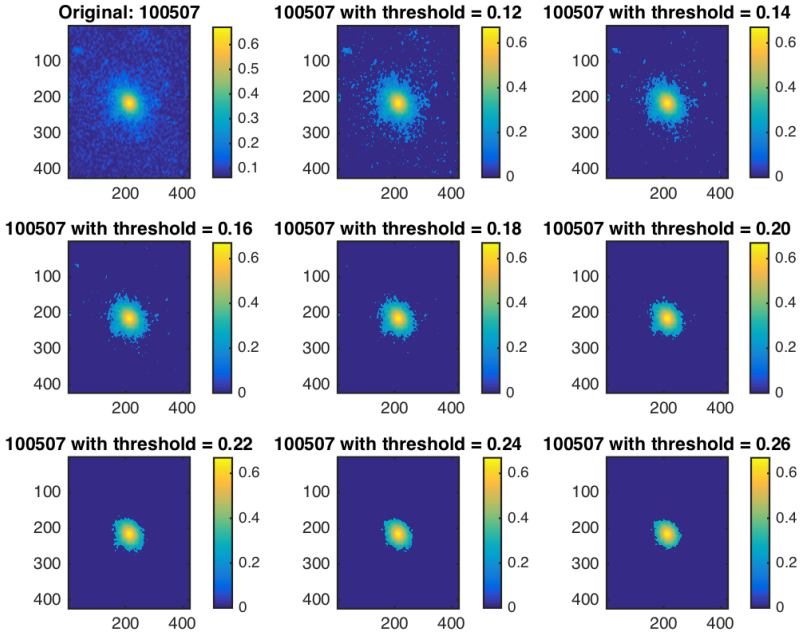
The future of this field of research is very exciting as a new era of astronomy beckons. The technological challenges that face us are unsurmountable with today's methods but with the novel techniques proposed here, it is felt that these challenges could be conquered very soon.

Appendix A

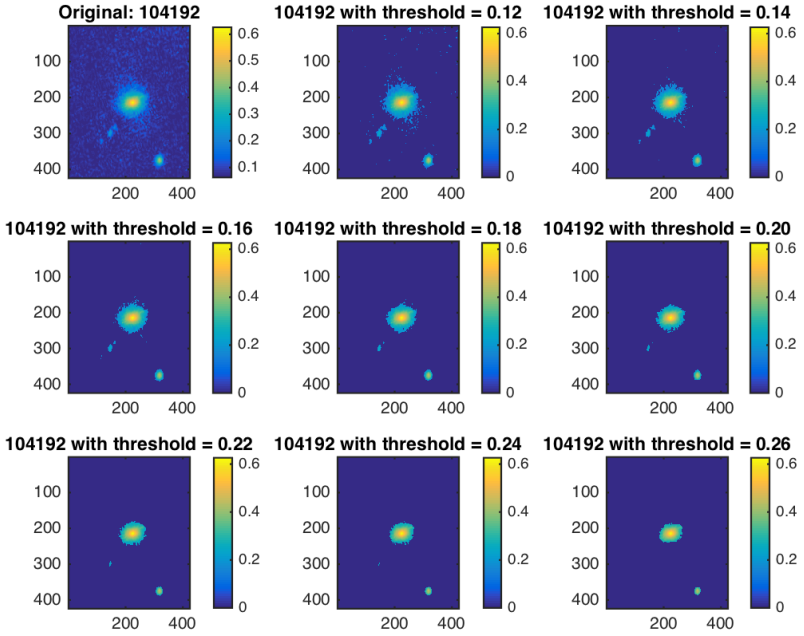
Thresholding

The Figure shown below is a comparison of astronomical images with varying thresholds.

For further examples, please see `http://www.astro-informatics.org/ /wikis/radio_learning//posts/tutorials/2016/08/04/Thresholding-data-II.html`



(a) Randomly selected image from galaxy zoo dataset



(b) Another randomly selected image from galaxy zoo dataset

Figure A.1: Comparison of threshold levels for astronomical images.

Appendix B

Visibilities

```
function im_d = create_dirty(im, p, input_snr)
% This function takes in an image and computes a dirty image with a specified
% level of sampling and added noise element. If 3 parameter not specified, no
% noise is added. This function is required to generate training and test .h5
% data files (see generate_train.m and generate_test.m)

switch nargin
    case 2

        rng_seed = 1;
        % Mask
        %mask = rand(size(im)) < p; ind = find(mask==1);
        mask=vdsmask(size(im,1),size(im,2),p,rng_seed);
        ind = find(mask==1);
        % Masking matrix (sparse matrix in matlab)
        Ma = sparse(1:numel(ind), ind, ones(numel(ind), 1), ...
        numel(ind), numel(im));

```

```
19          N = numel(im);
20          M = numel(ind);
21
22          A = \@(x) Ma*reshape(fft2(x)/sqrt(N), N, 1);
23          At = @(x) ifft2(reshape(Ma\'*x(:), size(im))*sqrt(N));
24
25          y = A(im); %% Use this for when not considering noise.
26
27      case 3
28
29          rng_seed = 1;
30          % Mask
31          %mask = rand(size(im)) < p; ind = find(mask==1);
32          mask=vdsmask(size(im,1),size(im,2),p,rng_seed);
33          ind = find(mask==1);
34          % Masking matrix (sparse matrix in matlab)
35          Ma = sparse(1:numel(ind), ind, ones(numel(ind), 1), ...
36          numel(ind), numel(im));
37
38          N = numel(im);
39          M = numel(ind);
40
41          A = @(x) Ma*reshape(fft2(x)/sqrt(N), N, 1);
42          At = @(x) ifft2(reshape(Ma'*x(:), size(im))*sqrt(N));
43
44          y0 = A(im);
45
46          % Add Gaussian i.i.d. noise
47          sigma_noise = 10^(-input_snr/20)*norm(im(:))/sqrt(N);
```

```
48          noise = (randn(size(y0)) + 1i*randn(size(y0)))*sigma_noise/sqrt(2);
49          y = y0 + noise;
50
51      otherwise
52
53          fprintf('Please provide correct number of arguments, at least
54          2 required\n');
55  end
56
57  %Dirty image
58  dirty = At(y);
59  im_d = 2*real(dirty);
```

Appendix C

Prototxt File

```
name: "RICNN"
layer {
  name: "data"
  type: "HDF5Data"
  top: "data"
  top: "label"
  hdf5_data_param {
    source: "RICNN/model/955-gz-optical-91-wt/train.txt"
    batch_size: 128
  }
  include: { phase: TRAIN }
}
layer {
  name: "data"
  type: "HDF5Data"
  top: "data"
  top: "label"
  hdf5_data_param {
```

```
    source: "RICNN/model/955-gz-optical-91-wt/test.txt"
    batch_size: 2
  }
  include: { phase: TEST }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 0.1
  }
  convolution_param {
    num_output: 64
    kernel_size: 9
    stride: 1
    pad: 0
    weight_filler {
      type: "gaussian"
      std: 0.001
    }
    bias_filler {
      type: "constant"
      value: 0
```

```
    }
  }
}

layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}

layer {
  name: "conv2"
  type: "Convolution"
  bottom: "conv1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 0.1
  }
  convolution_param {
    num_output: 32
    kernel_size: 5
    stride: 1
    pad: 0
    weight_filler {
      type: "gaussian"
```

```
      std: 0.001
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}

layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}

layer {
  name: "conv3"
  type: "Convolution"
  bottom: "conv2"
  top: "conv3"
  param {
    lr_mult: 0.1
  }
  param {
    lr_mult: 0.1
  }
  convolution_param {
    num_output: 1
```

```
    kernel_size: 5
    stride: 1
    pad: 0
    weight_filler {
      type: "gaussian"
      std: 0.001
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}

layer {
  name: "loss"
  type: "EuclideanLoss"
  bottom: "conv3"
  bottom: "label"
  top: "loss"
}
```

Bibliography

[Barbosa et al., 2012] Barbosa, D., Anton, S., Gurvits, L., and Maia, D. (2012). *The Square Kilometre Array: Paving the way for the new 21st century radio astronomy paradigm: Proceedings of Symposium 7 of JENAM 2010*. Springer Science & Business Media.

[Bengio and Courville, 2016] Bengio, I. G. Y. and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

[Burke and Graham-Smith, 2010] Burke, B. F. and Graham-Smith, F. (2010). *An introduction to radio astronomy*. Cambridge University Press.

[Carrillo et al., 2012] Carrillo, R., McEwen, J., and Wiaux, Y. (2012). Sparsity averaging reweighted analysis (sara): a novel algorithm for radio-interferometric imaging. *Monthly Notices of the Royal Astronomical Society*, 426(2):1223–1234.

[Carrillo et al., 2014] Carrillo, R. E., McEwen, J. D., and Wiaux, Y. (2014). Purify: a new approach to radio-interferometric imaging. *Monthly Notices of the Royal Astronomical Society*, 439(4):3591–3604.

[Dong et al., 2014] Dong, C., Loy, C. C., He, K., and Tang, X. (2014). Image super-resolution using deep convolutional networks. *arXiv preprint arXiv:1501.00092*.

[Dong et al., 2015] Dong, C., Loy, C. C., He, K., and Tang, X. (2015). Image super-resolution using deep convolutional networks. http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html [Online; accessed June 06, 2016].

[Gerchberg, 1974] Gerchberg, R. (1974). Super-resolution through error energy reduction. *Journal of Modern Optics*, 21(9):709–720.

[Hogebom, 1974] Hogebom, J. (1974). The clean algorithm. *Astrophysics & Astron. Science*, 15:417–429.

[Honma et al., 2014] Honma, M., Akiyama, K., Uemura, M., and Ikeda, S. (2014). Super-resolution imaging with radio interferometry using sparse modeling. *Publications of the Astronomical Society of Japan*, page psu070.

[Hubel and Wiesel, 1959] Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591.

[Ivezić et al., 2014] Ivezić, Ž., Connolly, A. J., VanderPlas, J. T., and Gray, A. (2014). *Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data.* Princeton University Press.

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[Karpathy, 2016] Karpathy, A. (2016). Stanford university CS231n: Convolutional neural networks for visual recognition. *Lecture Notes*.

[Karttunen et al., 2007] Karttunen, H., Pekka KrÃűger, H. O., and Markku Poutanen, K. J. D. (2007). *Fundamental Astronomy*. Springer Science & Business Media.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kutz, 2013] Kutz, J. N. (2013). *Data-driven modeling & scientific computation: methods for complex systems & big data*. OUP Oxford.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

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

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

[Ng, 2009] Ng, A. (2009). Stanford university CS229: Machine learning. *Lecture Notes*.

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

[Olah, 2016] Olah, C. (2016). Convolution operator. http://colah.github.io/ [Online; accessed September 09, 2016].

[Prince, 2012] Prince, S. J. (2012). *Computer vision: models, learning, and inference*. Cambridge University Press.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[Skilling and Bryan, 1984] Skilling, J. and Bryan, R. (1984). Maximum entropy image reconstruction: general algorithm. *Monthly notices of the royal astronomical society*, 211(1):111–124.

[Starck and Murtagh, 2007] Starck, J.-L. and Murtagh, F. (2007). *Astronomical image and data analysis*. Springer Science & Business Media.

[Wiaux et al., 2009] Wiaux, Y., Jacques, L., Puy, G., Scaife, A., and Vandergheynst, P. (2009). Compressed sensing imaging techniques for radio interferometry. *Monthly Notices of the Royal Astronomical Society*, 395(3):1733–1742.

[Wikipedia, the free encyclopedia, 2016] Wikipedia, the free encyclopedia (2016). Typical cnn architecture. https://en.wikipedia.org/wiki/Convolutional_neural_network [Online; accessed September 09, 2016].

[Xu et al., 2014] Xu, L., Ren, J. S., Liu, C., and Jia, J. (2014). Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, pages 1790–1798.

[Zeiler et al., 2010] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE.

[Zeiler et al., 2011] Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE.