Aerial Image Super-Resolution using Generative Adversarial Networks

TRISTAN FRIZZA

B.Eng (Hons) and B.Sci (Adv.)

THE UNIVERSITY OF SYDNEY

Supervisor: Dr. Donald Dansereau
Associate Supervisor: Dr. Nagita Mehr Seresht

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

School of AMME
Faculty of Engineering
The University of Sydney
Australia

13 January 2020

Executive Summary

Over the last couple of years, the rise of deep learning has enabled us to achieve previously unthinkable performance in computer vision. These highly complex models show great promise in learning to deeply model large and varied image datasets with a high degree of accuracy, often times achieving human-level performance on specific tasks such as image classification.

We look to these methods as a means for creating state-of-the art computer vision algorithms for the purposes of enhancing aerial imagery in a remote sensing business use-case. For our work we choose to focus specifically on the task of single image super-resolution which involves the estimation of a high-resolution image from its corresponding low-resolution counterpart. We see this as being particularly useful for remote sensing applications at Nearmap since very high resolution ortho-imagery and derived semantic segmentation masks are key features of their product. Being able to enhance this digitally is much more cost-effective than always having to continually develop custom image capture hardware.

In particular we look towards a specific genre of deep neural networks called Generative Adversarial Networks (GAN), a class of generative model which have shown immense promise in recent years. They allow us to learn the salient features of a dataset in an unsupervised fashion and furthermore exhibiting unrivalled performance in generating synthetic images that closely mimic the original dataset. We therefore apply GANs to our Convolutional Neural Network model for super-resolution in order to enable it to learn vastly more realistic up-sampled images that better capture semantic image details through a loss function designed to closely align with human perceptual evaluations of image quality. We extend this application to show how such a model can be used to optimise imagery for downstream semantic segmentation tasks, allowing us to take low-resolution aerial imagery and still achieve high segmentation accuracy by modifying the images rather than the segmentation model itself.

Declaration of Contribution

I hereby declare that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

The work I contributed to includes:

- Conducting background research and writing of the literature review.
- Developing an extensible Generative Adversarial Network framework and resulting super-resolution system for use in Nearmap's image processing pipeline.
- Designing and conducting experiments to test the performance of the model and its range of applicability.

The work not performed by me includes:

- Capture and initial pre-processing of the aerial imagery.
- Preparation and adaptive sampling of Nearmap's proprietary datasets such as Apollo.
- Development of the semantic segmentation model used for my image content loss functions.

.....	.....
Signature of the student	Signature of the supervisor
Tristan Frizza	Dr. Donal Dansereau
School of AMME,	Australian Centre for Field Robotics,
The University of Sydney	The University of Sydney
Date:	Date: .....

Abstract

We demonstrate the effectiveness of Generative Adversarial Networks (GAN) for intelligently up-sampling aerial imagery to greater than native resolution and compare their performance against traditional bilinear up-sampling as well as standalone Convolutional Neural Network (CNN) models. We firstly evaluate popular CNN super-resolution models and achieve a 3.32dB increase in PSNR and 0.1756 increase in SSIM on Nearmap's aerial image dataset. We then use GANs to generate high-resolution aerial images and apply this to our existing super-resolution framework to produce results that achieve much higher detail and semantic fidelity. Finally, we demonstrate the applicability of our super-resolution model as a means of optimising images for downstream semantic segmentation, increasing the IoU of derived masks when compared to a bilinear baseline by up to 0.34 for 4x and 0.62 for 32x up-sampling. These findings are significant because they highlight the weaknesses of common approaches to image loss metrics and propose an effective framework for learning content-based vision algorithms that more accurately capture image semantics. This is important for computer vision businesses like Nearmap to build more accurate image models that align with human perception.

Keywords: *Aerial Image Processing, Remote Sensing, Computer Vision, Machine Learning, Deep Learning, Deep Generative Modelling, Generative Adversarial Networks, Unsupervised Learning, Image Super-resolution, Semantic Image Segmentation, Representation Learning.*

Acknowledgements

Thank you to my thesis supervisors, Dr. Donald Dansereau and Nagita Mehr Seresht for their continual feedback and valued advice that has greatly helped guide my approach to this body of work. Furthermore I am grateful for the hospitality of Nearmap in making my work thoroughly enjoyable and give thanks to the fantastic AI team for their genuine friendliness and willingness to sit down and talk through the day-to-day challenges I've encountered. Many thanks must also be given to my parents, John and Margarita, who have been continually supportive of my work throughout this time and gone well out of their way to make the journey a lot smoother. Lastly I'd like to thank my other ESIPS comrades Filip, Zarif, Tom and Zac for being in this journey together with me. I'll deeply miss all the great banter and table tennis matches we had along the way.

Contents

Executive Summary ii

Declaration of Contribution iii

Abstract iv

Acknowledgements v

Contents vi

Chapter 1 Introduction 1

1.1 Motivation 2

1.2 Industry Context 4

1.2.1 Nearmap 4

1.3 Thesis Aims and Outcomes 5

1.4 Research Strategy 6

1.5 Thesis Structure 7

Chapter 2 Background 9

2.1 Semantic Image Segmentation 9

2.2 Machine Learning 11

2.2.1 Deep Learning 12

2.2.2 Convolutional Neural Networks 15

2.3 Generative Modelling 16

2.3.1 Variational Autoencoders (VAE) 18

2.3.2 Generative Adversarial Networks (GAN) 18

Chapter 3 Literature Review 21

3.1 Deep Learning in Remote Sensing 21

3.2 Image Synthesis 23
3.2.1 Deep Convolution GAN (DCGAN) 24
3.2.2 BigGAN 26
3.3 Image Super-resolution 27
3.3.1 Classical Super-resolution 28
3.3.2 Super-resolution using Deep Learning 28
3.3.3 Checkerboard-free Up-sampling 31
3.4 Super-resolution using GANs 34
3.4.1 Super-resolution GAN (SRGAN) 34
3.4.2 Enhanced Super-resolution GAN (ESRGAN) 37
3.5 State-of-the-art GAN Architectures 39
3.5.1 Self-Attention Generative Adversarial Network (SAGAN) 39
3.5.2 ProGAN 41
3.5.3 Multi-Scale Gradient GAN (MSG-GAN) 41
3.6 Achieving Stability in GAN Training 43
3.7 Summary 50

Chapter 4 Methodology **52**
4.1 Development Environment 52
4.2 Dataset 54
4.3 CNN Super-resolution 56
4.3.1 Baseline Model: SRResNet 57
4.3.2 EDSR 58
4.3.3 RRDN 59
4.3.4 Evaluation Metrics 61
4.3.5 Training 62
4.4 GAN Image Synthesis 63
4.4.1 Single-scale GANs 64
4.4.2 Multi-scale GANs 65
4.4.3 Loss Function 66
4.4.4 Evaluation Metrics 66

4.4.5 Training 67
4.5 GAN Super-resolution 70
4.5.1 Generator 70
4.5.2 Feature Loss 71
4.5.3 Discriminator 72
4.5.4 Improving Downstream Semantic Segmentation 75
4.5.5 Evaluation Metrics 75
4.5.6 Training 76

Chapter 5 Results **77**
5.1 CNN Super-resolution 77
5.1.1 SRResNet 78
5.1.2 EDSR 79
5.1.3 RRDN 80
5.1.4 Comparison between models 81
5.1.5 Performance at different up-sampling factors 83
5.2 GAN Aerial Image Synthesis 85
5.2.1 Single-scale Model 85
5.2.2 Multi-scale Model 89
5.3 GAN Super-resolution 91
5.3.1 Image Resolution Enhancement 91
5.3.2 Semantic Segmentation Enhancement 94

Chapter 6 Discussion **104**
6.1 Research Aims 104
6.2 CNN Super-resolution 105
6.3 GAN Aerial Image Synthesis 108
6.4 GAN Super-resolution 109
6.5 Summary 112

Chapter 7 Conclusion **114**
7.1 Implications 115

7.2 Future Work . . . 117

Appendix A Supplementary Results and Figures **120**

A1 GAN Super-resolution . . . 120

A1.1 Segmentation Accuracy Results . . . 120

A1.2 Segmentation Accuracy IoU Tables . . . 132

Appendix B Work, Health and Safety Report **134**

B1 WH&S Policy at Nearmap . . . 134

B2 WH&S during the ESIPS Placement . . . 136

B3 Safety and Security . . . 138

Appendix C Industry Case Studies **139**

C1 MECH4601: Professional Engineering II . . . 139

C1.1 Unit of study outline . . . 139

C1.2 Learning outcomes . . . 141

C1.3 Conclusion . . . 146

C2 AMME4710: Computer Vision and Image Processing . . . 147

C2.1 Unit of study outline . . . 147

C2.2 Learning outcomes . . . 148

C2.3 Conclusion . . . 157

Appendix D Practical Experience Logbook **158**

Appendix Bibliography **163**

CHAPTER 1

Introduction

Aerial imagery has undergone substantial change since its early uses in the World War I for reconnaissance and military surveillance. Going from primitive black and white film to today's high resolution digital imagery has enabled the field of remote sensing to expand its applications to anything from monitoring forest fires to urban planning and evaluating agricultural yields. Photographing from the altitude of light aircraft enables providers to capture extremely high quality ortho-rectified image maps with ground sampling distances (GSD) [1] on the scale of a few centimetres. Image post-processing techniques like Fourier Transforms have enabled immense resolution enhancements on top of continuous improvements in capture hardware. Most recently with the rise in effectiveness of deep learning we have seen it used extensively for tasks like semantic segmentation whereby key features like roads, roofs and trees can be extracted from existing photos. We look to the apply state-of-the-art in deep generative modelling to evaluate their advantages over more established convolution neural network classifiers. Generative Adversarial Networks (GANs) [1] have achieved unprecedented success in synthesising extremely realistic computer-generated images like human faces and landscapes by training two neural networks head-to-head.

Despite their impressive results, GANs have been slow to see real adoption in tasks outside of academia. This is in part due to them being so nascent and hence requiring deep theoretical expertise to design. Furthermore, most of the theory (or lack thereof) backing GANs is still on the whole poorly understood and they are notoriously unstable and computationally expensive to train, putting them out of reach for many real-world applications. In this paper we will apply these techniques to the task of aerial image super-resolution which involves

[1] Distance between image pixel centers as measured on the ground.

the intelligent up-sampling of images to above their native resolution, a task well-suited to remote sensing applications. We will be applying these methods to the imagery of Nearmap, a world-leading large-scale aerial imagery provider, with the intent of enhancing their current photo-imagery to sub-pixel resolution with the downstream benefit of improving the accuracy of their semantic segmentation predictions.

1.1 Motivation

"What I cannot create, I do not understand."

— Richard P. Feynman

Super-resolution of natural images is a highly challenging task because it is an ill-posed inverse problem, meaning several non-unique high-resolution solutions exist for the one low-resolution image. Furthermore, by effectively super-resolving an image we must go above simple interpolation strategies which simply scale up the existing information in the image and instead perform complex sub-pixel estimation, inferring new and useful information from the content and structure of the image. The reason we are able to reasonable achieve such a feat is because images often encode a surplus of information, and a lot of this missing pixel data such as edges and textures can be inferred from neighbouring pixels. This is where deep neural networks and GANs come into their own as they allow us to build a complex conditional model for these high-resolution images. Super-resolution of aerial imagery is particularly interesting because increased resolution is a highly sought after attribute of remotely captured imagery. Being able to have sharp, high-pixel density images would be invaluable in applications both in commercial photogrammetry and military surveillance.

Generative modelling has demonstrated tremendous potential for allowing us to build unprecedented complexity and semantic understanding into our models of the world. By learning how to effectively create new examples from a dataset, generative models are forced to learn extremely deep and meaningful representations of what they are tasked to model. Without needing any sort of explicit labels from humans, this approach can be extremely powerful for

learning high dimensional features of challenging datasets as an unsupervised representation learning task.

This differs substantially from the most popular and widespread uses of deep learning for computer vision which often train classification networks to predict labels of features in an image given a human-labelled ground truth to emulate. Generative models can be thought of as an extension to this framework to sidestep the need for human supervision. Although we still encode the dataset to learn more compact representations (like a classifier), we go a step further to use these learned features to believably rebuild our original data distribution, helping us refine and improve our general understanding by comparing our assumptions to reality. As an analogy, you would probably expect someone who has painted all their life to have a deeper appreciation of natural landscapes than one who has only ever seen them in art galleries. For these reasons I believe that generative modelling and unsupervised learning is already paving the way for many big breakthroughs in deep learning and computer vision and hence aim to explore their potential in this work.

FIGURE 1.1. The incredible representational power of StyleGAN to create artificial human faces indistinguishable from the reality [2].

We've already seen the proved effectiveness of generative models like GANs to create highly convincing imagery of anything from human faces to cars [2]. Modelling the manifold of real images is a very high dimensional task, something not previously possible to achieve.

1.2 Industry Context

1.2.1 Nearmap

Nearmap currently does an extensive amount of pre-processing on their imagery to improve colour contrast, sharpening and so on. However for instances where we have survey holes, i.e. certain areas of the ground have failed to be captured, they are simply filled in with low resolution overview photos taken by a secondary camera system. These filler photos are approximately 32x lower resolution and are primarily used to provide global context for feature matching. Hence in production when clients see these patches of low quality imagery stitched in alongside the high quality ones it often looks quite unsatisfactory and is functionally quite useless if a user is trying to understand the details of the image. Furthermore, we currently don't do anything specific to correct for motion blur or lens de-focusing issues and are currently working on non machine learning approaches to ISO noise reduction which are all areas where GAN enhancement could be highly beneficial.

FIGURE 1.2. Example of a survey hole (highlighted in red) where missing areas of the image are filled in with low resolution overview captures.

1.3 Thesis Aims and Outcomes

By researching this area of GANs for image super-resolution we aim to combine current best practices in the literature as well as propose our own improvements specific to our aerial image use-case. To overcome the relative difficulty in understanding and training GAN models, we also develop a self-contained GAN image processing library with the hopes that this makes use and adoption of these techniques more prevalent at Nearmap. This is very useful to Nearmap as their business relies on affordable image capture at a massive commercial scale. By performing cutting-edge super-resolution post-processing of images, they can be less reliant on purchasing or fabricating hugely expensive custom camera systems to achieve very high resolution imagery.

If successful, this GAN system could be used Nearmap's existing production pipeline to super-resolve (SR) aerial imagery and provide much more pleasing, high fidelity photos to customers. It would also more importantly enhance our dataset for when we run our semantic masking algorithms on them (since they don't cope well with blurry or pixelated images).

FIGURE 1.3. The effectiveness of deep learning for aerial super-resolution of aerial imagery. From left to right we compare the low-resolution nearest-neighbour up-sampled, bilinear up-sampled, GAN super-resolved (ours) and high-resolution original image.

This general GAN framework can then be fairly easily extended to fulfil a number of other image processing tasks, with the only main modification being just the dataset that it is trained on, as long as it reflects the image characteristics you are trying to reverse (e.g. pairs of clean and noisy images).

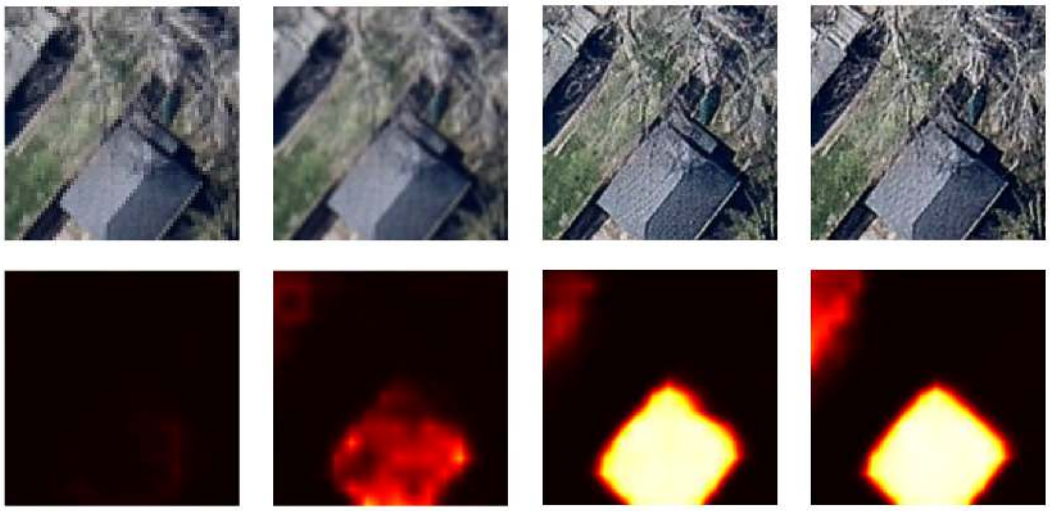


FIGURE 1.4. An example of how applying our super-resolution network to poor quality base imagery can greatly improve the downstream task of semantic image segmentation. From left to right the images are nearest neighbour, bilinear, GAN super-resolved and original. The bottom row depicts the semantic map prediction of each image for the class of shingle roofs and is clearly much more accurate post-super-resolution.

1.4 Research Strategy

This thesis tackles the research problem of achieving aerial image super-resolution. Succinctly, our primary research contributions are addressing the applicability of existing super-resolution methods to aerial imagery, designing modified generator and discriminator model architectures, engineering more semantically meaningful image loss functions fine-tuned for this domain, evaluating the improvement in perceptual image quality and segmentation accuracy and the overall practicality in training and deploying such models for an aerial imagery business. We achieve these goals in a few key steps as follows.

Firstly, we set out to review contemporary aerial image enhancement and super-resolution methods, both traditional methods as well as the more cutting-edge deep learning based ones. We then perform a deep dive into deep learning methods for single image super-resolution and evaluate many of the most popular approaches (not limited to just aerial the imagery domain). From this we summarise the best practical and most up to date techniques. We begin by building a straightforward convolution neural network model (SRResNet) that serves as our

baseline super-resolution model and apply this to our aerial dataset. We start by simply using a simple pixel-based loss objective, later augmenting it with an auxiliary feature based loss.

For the remainder of the thesis we build on top of this base model by employing a GAN training setup, whereby a second discriminator network is trained adversarially against our original convolution generator network. To get to grips with the GAN literature and the caveats of designing and training such models we also briefly experiment using GANs for image synthesis as a lead-in to applying them to the super-resolution task. For our super-resolution GAN we apply the most up to date network models, layers, regularisation techniques and so on to come up with a set of high-performing and stable super-resolution models, drawing inspiration from many different areas of GAN literature. We evaluate our results by super-resolving on a large number of images from our test set (unseen images) to validate that it does indeed generalise and works as expected. For this we compute various standard image metrics like PSNR and SSIM averaged over the test set to quantify the improvement our model gives. We finally fulfil our secondary use-case by passing all upsampled outputs of our model through Nearmap's proprietary semantic segmentation model and report the class-averaged IoU of traditional upsampling techniques (nearest-neighbour, bilinear) alongside our super-resolved results when compared against the high-resolution ground truth segmentation.

1.5 Thesis Structure

The following sections of this thesis cover the following topics in order:

- **Chapter 2:** Outlines the objectives of remote sensing and current applications of deep learning to related computer vision tasks. Gives a background rundown on the essentials of machine learning, deep learning and generative modelling to bring the reader up to speed.
- **Chapter 3:** Goes into depth on contemporary approaches to deep generative modelling focusing primarily on Generative Adversarial Networks. We critically explore the current state of the art architectures and their effectiveness when applied to image

modelling tasks, in particular to image super-resolution and touch on how we take inspiration from them in our own work.

- **Chapter 4:** Details the methodology behind our experimentation. It covers the dataset available, the framing of the super-resolution task in deep learning terms, the model architectures and their training procedures and parameters as well as methods for validating results.
- **Chapter 5:** Analyses the qualitative image results of our models as well as evaluating the quantitative metrics to benchmark performance.
- **Chapter 6:** Discusses the results in the context of our aims and desirable outcomes. Evaluates the effectiveness of our approach and critiques components of our model and approach.
- **Chapter 7:** Summarises and highlights the significance of the work conducted. Opens up the floor for future improvements and further research.
- **Appendix:** Contains additional results and experimentation as well as work, health and safety, industry case studies and practical experience logbook.

CHAPTER 2

Background

2.1 Semantic Image Segmentation

Semantic image segmentation relates to the generation of semantic masks to map out meaningful features of an input image. As an example, this is commonly seen in the vision systems of self-driving cars in order to detect and localise nearby objects like other cars, trees and pedestrians so that the navigation algorithm can meaningfully understand its surroundings.

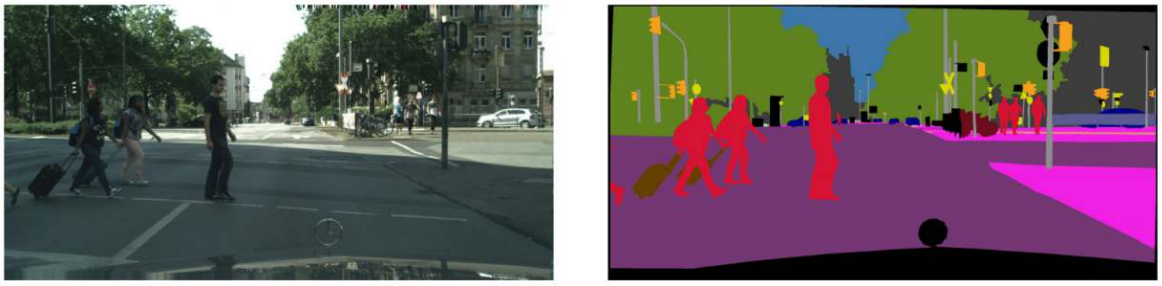


FIGURE 2.1. Semantic image segmentation of urban environments using DeepLab [3]. This allows semantically relevant regions of an image to be isolated and localised, such as recognising where pedestrians are in the field of view for autonomous vehicle perception.

In a similar fashion, semantic segmentation can be applied to aerial imagery in order to identify and predict the location of important features of the overhead imagery. This enables companies like Nearmap to accurately find pools, roofs, solar panels, roads, construction sites, bodies of water and so on as well as predict finer details like the shape of a roof and its material (shingle, metal etc.). The ability to automatically segment these allows for this to be done at a huge scale and with fairly high precision which gives an enormous advantage over a human operator who would not feasibly be able to label the shape of every roof in a

city. This information is invaluable to clients ranging from councils and governments to solar panel installers, telecommunications providers and property developers alike.

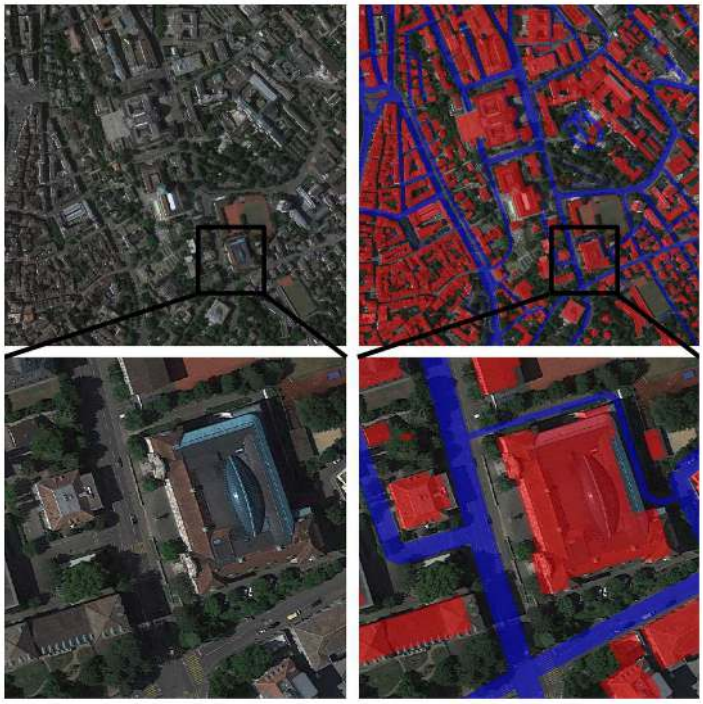


FIGURE 2.2. Semantic segmentation in the context of aerial orthoimagery can be used in order to extract buildings and roads [4]. This is useful to learn what geographical entities exist in an area, where they are located and perhaps their area or shape.

This task has proven to be challenging to automate, and is even often difficult for manual human labelling because of the ambiguity and high variance in conditions. It is common to have drastically different lighting conditions, occlusion from overhanging trees and shadows as well as very different roof shapes from city to city. This makes it exceptionally difficult to quickly and effectively perform this at scale, which is often a necessity for commercial purposes. It also becomes a challenge to come up with a well-defined taxonomy of classes to segment for and this usually depends heavily on the business use-case. To best handle this vast complexity, almost all successful approaches these days have turned to machine learning as their solution with some genuinely impressive results now being within the realm of possibility.

2.2 Machine Learning

Machine learning is a field closely tied with computer science and statistics. It takes a different approach to the conventional way in which we usually see computer science algorithms written. Rather than explicitly programming the rules of your algorithm, we give the algorithm the flexibility to learn its own rules based on how it best fits the underlying data. Based on a given update rule we can use errors in our predictions to perform optimisation on the parameters of the algorithm to achieve iteratively better results as we train the model. These sorts of algorithms have been widely used in applications from forecasting in financial markets to what we're using them for in computer vision tasks.

The 3 most notable approaches to machine learning have been supervised, unsupervised and reinforcement learning.

Supervised learning involves a scenario where your dataset has ground truth labels that have usually been annotated by a human. These labels allow us to easily define a loss function which is a quantitative measurement for how similar our model predictions were to the ground truth. This loss value can then be used to update accordingly the parameters of the model using strategies like maximum likelihood estimation (MLE) [1]. Supervised learning has seen by far the most real world adoption because of it's high performance and stable, monotonic training due to the strong signals given by the labels. This has been popular in many applications to do with fitting classification and regression models and is very effective for dealing with complex datasets with millions of data points.

Unsupervised learning on the other hand forgoes the need for explicit labels and can be thought of instead as clustering and pattern extraction from data. It is often used as a pre-processing step for supervised methods to find general features of the data and then run supervised classification over the top of that for faster and more accurate results. Clustering algorithms like K-means, DBSCAN [5] and dimensionality reduction algorithms like PCA [6], t-SNE [7] and UMAP [8] have also been instrumental for visualising extremely high

[1] MLE is a statistical method for estimating the parameters of a distribution/model by maximising the likelihood of the data under your distribution

dimensional data by projecting it into a low dimensional 2D or 3D space. This body of work will be primarily concerned with the notions of unsupervised learning as generative models typically are trained without the use of labels (although there are exceptions [9]). Even though unsupervised learning often does not yield results on the same tier as supervised approaches, it has an enormous advantage in that it requires no explicit labelling of data. This is an enormous advantage because the vast majority of real world data we encounter has not been nicely cleaned and labelled by someone. Therefore the holy grail of machine learning is to come up with powerful representation learners that need only minimal supervision (semi-supervised) or no supervision at all.

A sub-genre of unsupervised learning that is of interest is called self-supervised learning. This involves a scenario where the model uses its own self-generated supervision signal and hence has no external need for labels. Often this is achieved by performing an auxiliary task in which the model learns both how to label as well as how to predict on those self-made labels. This can be elegantly explained as "give a machine learning model a label and it will learn once, teach it how to label and it will always be learning".

The last category is reinforcement learning which uses no explicit dataset or explicit supervision signal, instead learning from experience based on interactions with its environment and maximising the rewards and minimising the penalties associated with those interactions. However this has only really seen some limited use in robotics and control applications and will not be elaborated on in this thesis.

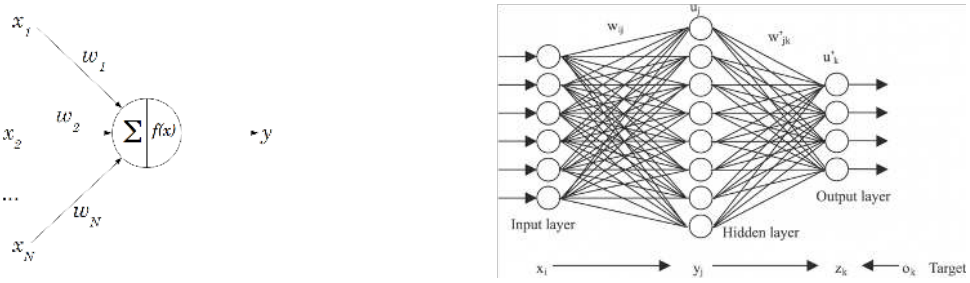
2.2.1 Deep Learning

Deep learning is a sub-field of machine learning primarily concerned with the use of Artificial Neural Networks (ANN) as differs to the logistic or tree-based regressor models you might see elsewhere. They are comprised of an interconnected feedforward network of perceptron units (also commonly referred to as neurons) [10]. On their own these neurons perform fairly simple affine transformations on their inputs followed by an activation function on the output, providing the non-linearity in the system. Here $\mathbf{x}$ is our input data vector, $\mathbf{w}$ is

the multiplicative weight vector, b is the additive bias scalar, f is some non-linear activation function and y is the final output prediction scalar.

$$y = f(\mathbf{w}^T\mathbf{x} + b) \tag{2.1}$$

However the overall complexity stems from the often dense interconnections between neurons in the network whose individual weightings can get into the millions of parameters. By stacking many of these layers one after the other we can compound the effects of the non-linear transformations, allowing us to model very complex functions. For this reason these networks are regarded as universal function approximators [11], meaning they can fully represent arbitrarily complex functions given sufficient parameters. This makes them great for fitting highly non-linear and high dimensional problems which is of great benefit in computer vision where our data dimension in pixels grows as the square of the image height/width.



(A) Graphical representation of a perceptron [2]. (B) Network topology of a neural net [3].

FIGURE 2.3. Artifical Neural Networks are a multi-layer feed-forward network of perceptron units.

Neural networks are able to learn by passing inputs through their network and computing a prediction output, this is known as forward-propagation or prediction. In a supervised setting, these predictions are compared to the ground truth label and the loss computed like before. However because we choose a differentiable loss function, we are able to work our way back through the network to compute local updates or gradients of the loss function with respect to each and every parameter. We then use an optimisation algorithm such as Stochastic Gradient Descent [12] to update our network weights in the direction of the gradient that minimises

[2] Accessed from: https://www.lucidarme.me/simplest-perceptron-update-rules-demonstration/

[3] Accessed from: https://www.extremetech.com/extreme/215170-artificial-neural-networks-are-changing-the-world-what-are-they

our loss. Eventually after many iterations the aim is to reach the minimum loss and hence achieve the most optimal model.

The recent spike in popularity and adoption of deep learning based methods has been enabled in no small part due to huge advances in hardware acceleration. The advent of GPU and ASIC TPU processors has allowed for extremely quick and highly parallelisable tensor (matrix) computations. Because deep learning can be essentially thought of as many, many matrix operations, this means we are now able to train deep networks with up to billions of parameters.

This has lead to massive breakthroughs in the computer vision field with the most notable examples of this being seen in results on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [13]. ILSVRC involves classifying images into 1000 categories from ImageNet [14], a labelled dataset of 14 million images, and has been seen as the long standing benchmark for vision algorithms. In 2012 a convolutional neural network (CNN) model named AlexNet [15] achieved a top-5 error rate of just 15.3%, breaking the previously held record by a whopping 10.8%. This was down to using this deep convolutional architecture and was made feasible by the use of GPUs in achieving this high processing throughput. Since then we have seen steady improvements in ImageNet scores down to below 3% error rate which is considered as good as accuracy of the human labellers themselves.

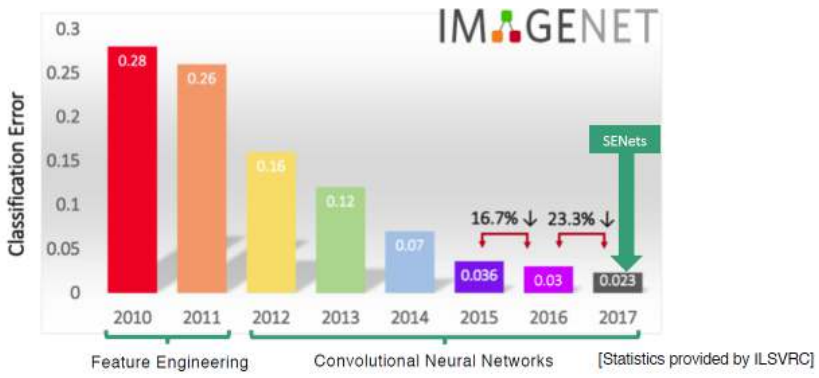


FIGURE 2.4. The vast increase in image classification performance on ILSVRC as a result of deep learning [13].

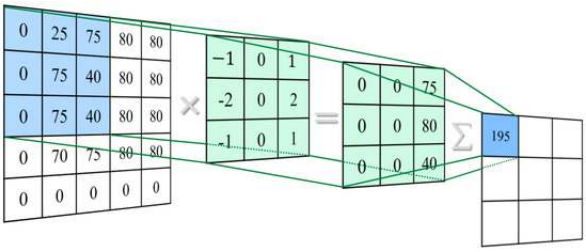


FIGURE 2.5. The concept of a convolutional kernel which works exceptionally well with image data [16].

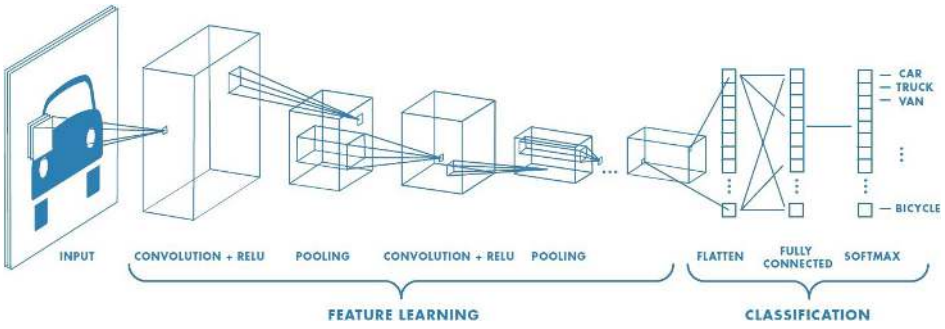


FIGURE 2.6. The network architecure of a typical CNN classifier [4].

2.2.2 Convolutional Neural Networks

Convolutional neural networks are a particular type of deep learning model that uses convolutional kernels as opposed to just fully connected perceptron layers. This form or regularisation allows the network to take advantage of the spatial information in the data as convolutions work by sliding a filter (e.g. a 3x3 kernel) along the data, meaning that every result in the output depends on multiple neighbouring input pixels. The parameters of each filter can then be learned via backpropagation as discussed earlier. By stacking these one after the other as is common in CNNs, we end up being able to learn hierarchical patterns as each layer pools the results of the previous layers into progressively higher abstractions.

It is for these reasons that CNNs have become a game-changer for computer vision because they are able to encode the spatial dependencies of neighbouring pixels well and also can learn hierarchical layers of meaning in images.

[4] Accessed from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

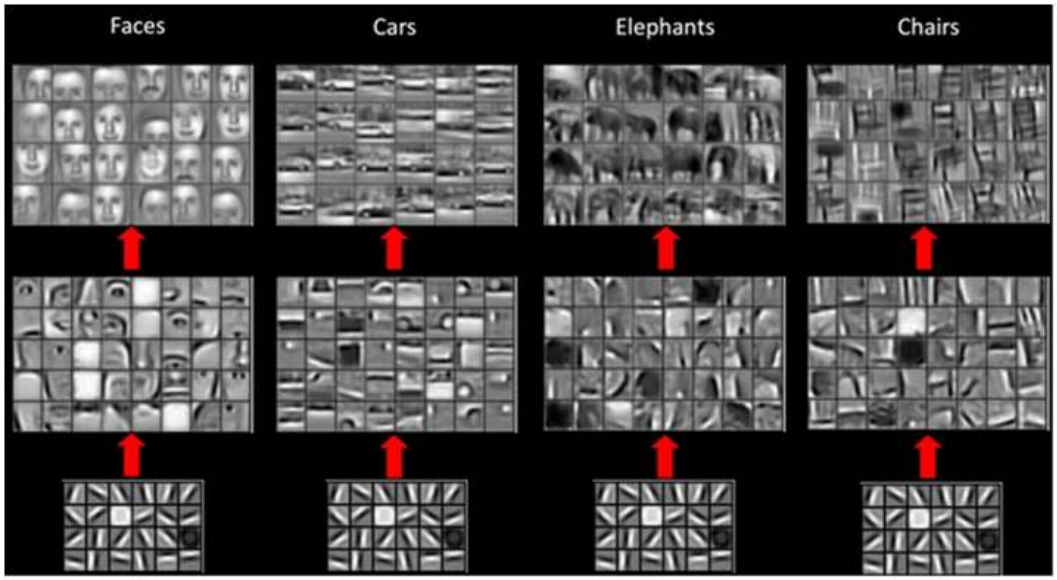


FIGURE 2.7. An illustration of how a CNN learns a hierarchy of features in successive layers [17].

This approach was inspired strongly by the way in which our visual cortex is thought to process information from our eyes. Neuroscientists have found strong evidence for learning hierarchies of visual information in the successive layers of the neocortex. This is a powerful approach as it allows lower layers to learn preliminary features like edges, then basic shapes and textures until the topmost layers can finally recognise complex features like faces due to this hierarchy of progressive complexity.

2.3 Generative Modelling

In machine learning and statistics there are two main approaches to classification. These are the generative and discriminative approaches [18]. The first attempts to learn a model of the joint probability distribution $P(X, Y)$, where X is our observable variable (our data) and Y is our target variable (our label). From this we are able to perform predictions by using the Bayes Theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

where $P(Y|X)$ represents the probability of a label given the data, thus allowing us to pick the most likely label y. On the other hand we can take the more commonly used and more

direct approach of discriminative classification which seeks to model the posterior distribution $P(Y|X)$ directly, i.e. learning a mapping straight from inputs X to class labels Y. Elegantly phrased by Vapnik [19], a compelling reason to use the discriminative approach is that "one should solve the (classification) problem directly and never solve a more general problem as an intermediate step (such as modelling $P(X|Y)$)". So it essentially boils down to whether you would like to optimise for the joint likelihood of inputs and labels or else the conditional likelihood of labels given inputs directly, the latter being more straightforward for pure classification.

However an interesting strength of generative models is that because even though they take the indirect approach to first model the joint probability distribution, this gives us a nice probabilistic model of our data allowing for more domain knowledge to be instilled. From this joint distribution we are also able able to, given this approximation of the true data distribution, be able to sample entirely new data points that follow the same general pattern as our original data but with some sensible variation. It is clear that if you want to generate a brand new image of a human face, you would need to have a really solid grasp of the semantics of what it means to be a face and the semantic features comprising it. This leads us to the crux of this thesis, to be able to learn deep and meaningful representations of image data as an alternative and hopefully more effective approach than that of discriminative classification for tasks like super-resolution.

However it is often intractable to learn the true distribution of our data $P(X)$ implicitly or explicitly. Hence we have to come up with the best possible approximations, which leads us to our discussion around deep neural networks. The use of neural networks to model these generative distributions is known as deep generative modelling and is mostly achieved as an unsupervised learning task. By far the two most common approaches are Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). Less popular models include autoregressive and flow-based models which both have their advantages and disadvantages.

2.3.1 Variational Autoencoders (VAE)

Variational Autoencoders [20] are a probabilistic approach to autoencoders, a type of neural network that employs and encoder-decoder setup to enforce an information bottleneck and effectively compress data into lower dimensional space. VAEs use bayesian variational inference to approximate the posterior with a pre-defined distribution $q(z|x)$, which is usually chosen to be Gaussian. They aim to do this by minimising the Kullback-Leibler divergence between our variational posterior $q(z|x)$ and the true posterior $p(z|x)$ to achieve a good approximation. However since as mentioned before the evidence or true distribution of the data $p(x)$ is intractable, we instead minimise the evidence lower bound (ELBO) which is an equivalent objective but is computationally tractable (by exploiting the fact the KL divergence is a distance and thus always positive).

These models use an encoder-decoder structure to model prior and posterior and can be trained unsupervised by comparing inputs to the encoder with the reconstruction on the output plus the KL divergence between our posterior and a pre-defined gaussian prior for the latent space (the hidden variables in the information bottleneck between encoder and decoder).

$$ELBO_i(\lambda) = \mathbb{E}q_\lambda\left(z|x_i\right)\left[\log p\left(x_i|z\right)\right] - \mathbb{KL}\left(q_\lambda\left(z|x_i\right)\|p(z)\right) \tag{2.2}$$

The advantage to VAEs is their general ease to train as well as their well defined as interpretable latent space z. However their weakness is that for a highly multi-modal data they tend to only be able to produce blurry images reconstructions as an average over the many modes of the data manifold which leads to less impressive results than that of GANs [21]. This is often a shortcoming of the loss functions used because L2 (least squares) image reconstruction loss encourages the learning of blurry averaged representations of the data.

2.3.2 Generative Adversarial Networks (GAN)

Generative adversarial networks are the other vastly popular branch of deep generative models that have garnered a lot of attention in the past few years. With their conception in 2014 by Ian J. Goodfellow [1] they are an implicit generative model, meaning that they

learn their deep neural network without explicit density estimation (like you would have in Autoregressive, Flow and Latent Variable models). Instead we train two networks head-to-head in an adversarial minimax game to try and find a Nash equilibrium between the two network objectives. The first network is commonly referred to as the 'Generator' (G) and the second the 'Discriminator' (D). The generator generates new samples (images in our case) and the discriminator is just a binary classifier that tries to predict which images are real (from our dataset) and which are fake (created by the generator).

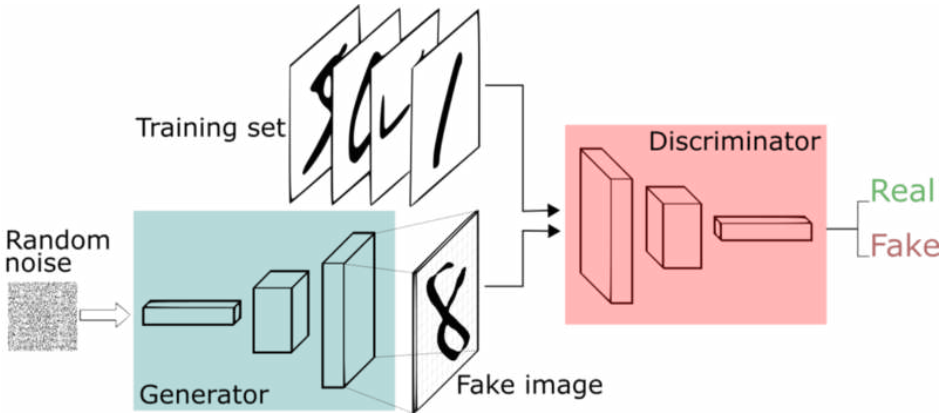


FIGURE 2.8. Generative Adversarial Network architecture diagram illustrating the role of the two networks in generating handwritten digits based on the MNIST dataset [5].

By having the discriminator maximise the log-likelihood for its binary classification task (classify samples as real/fake correctly) and the generator to minimise the log-probability of the samples it creates as being detected as fake by the discriminator (i.e. fooling it) we formulate the GAN training objective where both networks compete adversarially.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}(\boldsymbol{z})[\log(1 - D(G(\boldsymbol{z})))] \tag{2.3}$$

A good analogy as to the intuition of this approach is given in Goodfellow's original paper: "The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game

[5] Accessed from: https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles" [1].

GANs are universally renowned for creating some of the most realistic generated samples of any deep learning model. Their ability to have this extremely complex learned loss function allows for really amazing detail and hard-to-model dependencies in the data to be captured. However a huge downside to them is how long they take to train and worst of all how unstable this training can be. Because the GAN objective is not an explicit optimisation, this makes it really challenging to train. We will elaborate on the challenges of training and using GANs in a later section.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

FIGURE 2.9. General GAN training algorithm using stochastic gradient descent (SGD) [1].

CHAPTER 3

Literature Review

This work applies GANs to aerial image processing tasks where they are not used in the vast majority of aerial imagery cases, especially not in a production setting. Even though the concept is not entirely original [22] [23], I aim to make some fairly novel improvements on contemporary GAN and CNN techniques by leveraging recent research advancements. We anticipate some very positive outcomes because a lot of these state-of-the-art techniques have been proven to work very well individually or on other datasets (ImageNet type data primarily), and being able to pool a lot of these improvements together and fine-tune them to an aerial imagery-use case ought to achieve comparably impressive results.

3.1 Deep Learning in Remote Sensing

Deep learning has become an increasingly popular tool in the remote sensing community over the last few years. It is quite common to see convolutional neural network classifiers used for the detection of objects such as houses or even for things as abstract as flooding and earthquake damage for natural disaster areas. Similarly we see scene classification of images being popular to be able to classify images as forest or beaches for example.

Taking this a step further we see prominent use of CNNs for semantic image segmentation which performs classification on a pixel by pixel basis so that we can create masks over our image to localise important features such as cars, trees and swimming pools in aerial photographs. Businesses such as Nearmap use this extensively to automatically and at a huge scale be able to perform these segmentation tasks. The resulting masks are then extremely valuable when mapped back to geographical co-ordinates to be able to pinpoint where each

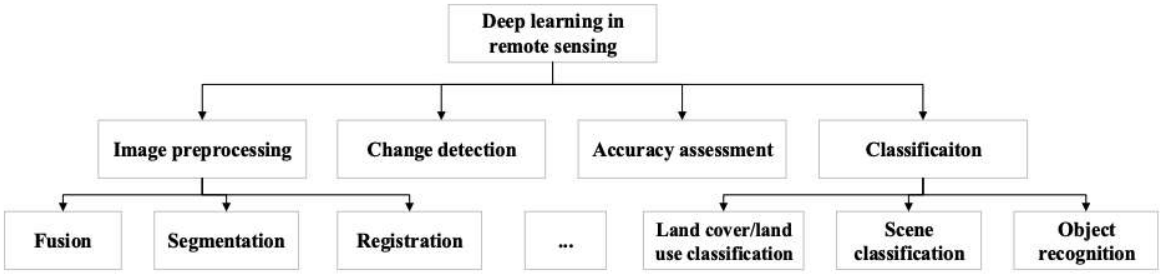


FIGURE 3.1. A diagram illustrating the major areas that deep learning has been applied to in remote sensing [24].

feature is in the world as well the amount or density of them. Similarly, given these masks one can perform change detection, which involves automatically tracking changes in urban and rural scenes. This is very useful for governments say to track construction projects and also monitor macro trends for things like the growing adoption of solar panels over time. These semantic segmentation deep learning models typically follow the encoder-decoder architecture whereby the original RGB image is sampled down into a lower dimensional feature space to compress and encode only the most salient features for the mask. This is then reconstructed and up-scaled in the decoder half of the network to produce the mask, which is then compared to the human labelled ground-truth to calculate the error in prediction. These models are fully convolutional, meaning every layer is a convolutional layer and we have no fully connected layers at all (because we are not performing classification and hence do not need to convert outputs to a scalar value).

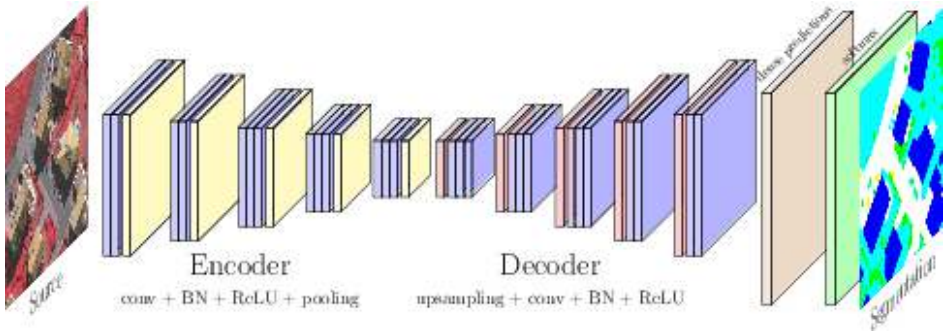


FIGURE 3.2. Fully convolutional network (FCN) architecture of SegNet [25] model for semantic image segmentation on ISPRS Vaihingen aerial imagery dataset [26].

Outside of the classification realm, deep learning is also notably used for image processing. A common application is for image fusion where we may have multiple different types of images like low-resolution multi-spectral (MS) but wish to combine this information with high-resolution panchromatic (PAN) imagery. This allows us to model a direct mapping between the low and high resolution patches in a method known as "pan-sharpening" [27] to achieve overall higher quality results than using either set of imagery standalone.

The task we will be focusing on the most is aerial image super-resolution which is an exciting branch of image processing to learn to reconstruct high resolution imagery from lower quality samples. This technique will be discussed further in its own section later on.

3.2 Image Synthesis

The most common task that GANs have been applied to (and succeeded at) is synthesising images from an input noise vector. Like a VAE model we aim to map our data to a continuous latent prior which allows for the sampling of new data points that also follow this same general distribution. The general setup as outlined in the diagram and algorithm above is to feed the generator an Gaussian noise vector of n variables. This is then up-sampled until we have an image on the encoder output. The discriminator is usually modelled as the inverse of this structure whereby we take an full scale image and use several layers to down-sample the image to output a binary prediction (i.e. it is just a standard binary classifier for images).

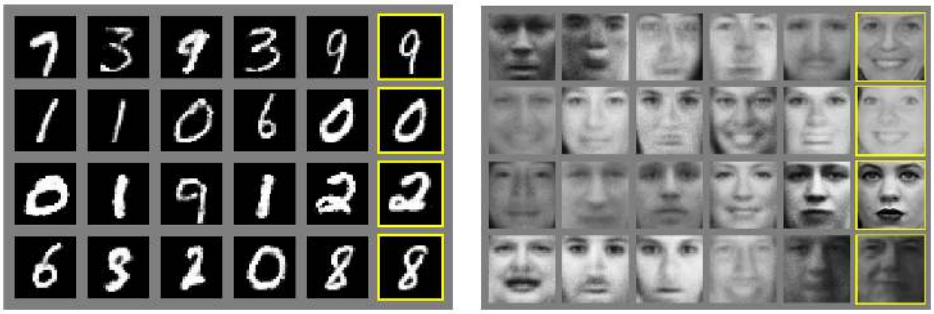


FIGURE 3.3. Image synthesis using GANs from the MNIST (left) and TFD (right) datasets. The images highlighted in yellow are the closest match in the original dataset, proving that GANs can learn new samples from outside the immediate dataset they are trained on [1].

Once trained, the GAN can be given any noise vector and produce a novel image from that part of its modelled distribution. The interesting thing is that our discrete dataset is now modelled as a continuous Gaussian vector space and hence we can predict on an interpolation of this latent space to produce a continuum of images that smoothly vary from one to the next.

FIGURE 3.4. Digits generated by linearly interpolating between co-ordinates in the latent space, z [1].

3.2.1 Deep Convolution GAN (DCGAN)

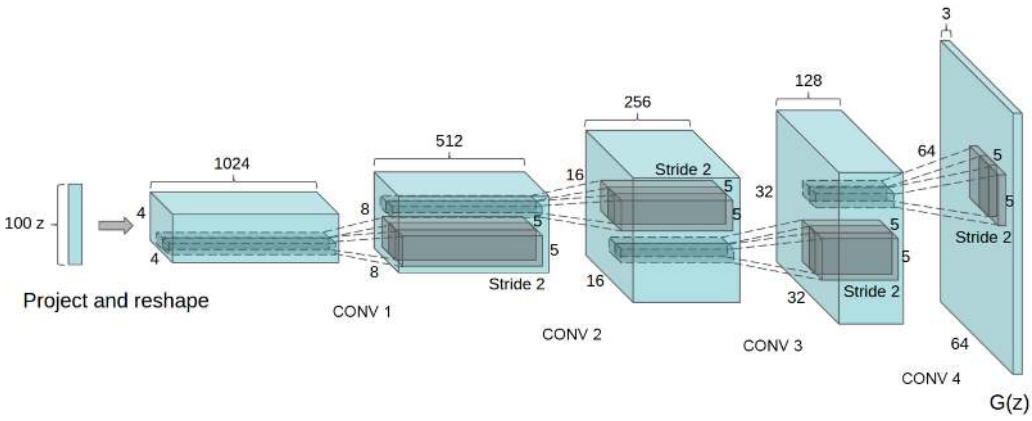


FIGURE 3.5. Deep Convolution GAN (DCGAN) architecture [28].

Deep Convolutional Generative Adversarial Networks (DCGAN) [28] came about in 2015 and although simple proved to be extremely effective and are often used as the baseline GAN architecture because they are known to work so well. The major step change was to use a fully convolutional set of layers rather than fully connected ones. Furthermore they replaced max pooling with stride 2 convolutions, included Batch Normalisation (BatchNorm) [29] and using leaky rectified linear unit (leaky ReLU) activations. Strided convolutions allowed for more parameters to be learned for down-sampling layers. Batch normalisation normalises input features of a layer to have mean 0 and variance 1 and is an essential component for avoiding mode collapse (explained in 4.4.5.1) in deeper networks. Leaky ReLU also greatly helped training because rather than rectify all negative values to 0, it allowed for a small flow of

gradient for negative signals and enabled stronger gradients to propagate from discriminator to generator.

Whereas many GANs before it struggled to create believable samples from anything other than simple MNIST digits, DCGAN was a big step up in image quality and complexity of learned features. A very interesting testament to this was the ability to perform vector arithmetic on the latent space vectors to add and subtract meaningful features of an image such as gender and eye-wear. This demonstrated that the latent features it had learned weren't arbitrary and actually had a lot to do with the semantic characteristics of the dataset, hence why Radford called DCGANs "unsupervised representation learners".

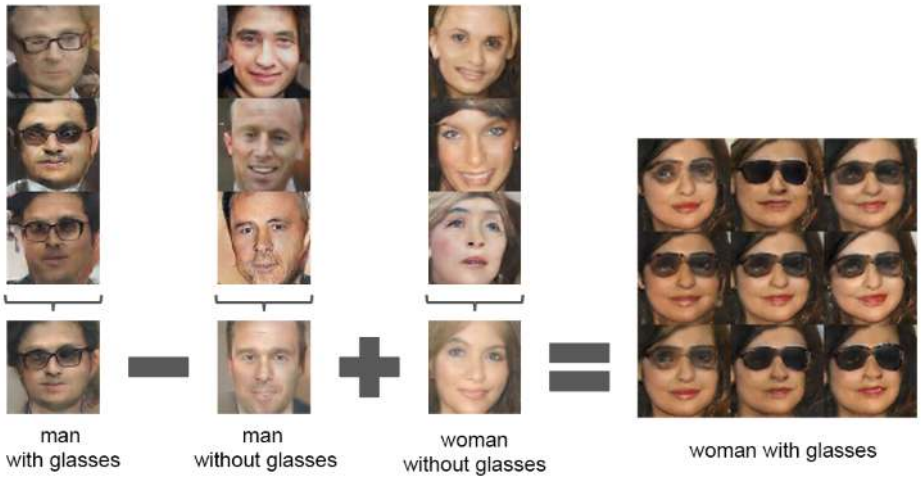


FIGURE 3.6. Vector arithmetic in the latent space of DCGAN illustrates the learning of important semantic features of the dataset [28].

We take great inspiration from this work and generally always use convolutional GAN layers with LeakyReLU and experiment with BatchNorm. However we also improve on this baseline with more sophisticated up-sampling that avoids the prevalent checkerboarding encountered in vanilla DCGANs (3.3.3). DCGANs also remain limited to small sized image generation and lack detailed output features. We enhance this by using different kernel and filter sizes as well as by learning features at more than one single scale (4.4.2).

3.2.2 BigGAN

Jumping forward a few years to 2018 we introduce the state-of-the art in large-scale image generation. BigGAN [30] scales up the number of parameters for a GAN model massively and also adds in some modern tricks to improve stability and image quality. They were successfully able to generate high-resolution, diverse samples from ImageNet which was previously impossible.

FIGURE 3.7. Conditional samples generated by BigGAN [30].

They found that increasing the batch size as much as possible helped a great deal since they postulated that each batch would cover more modes and provide better gradients for both networks. They also managed to achieve far better results by scaling up the number of channels in each layer but interesting found that increasing the depth did not really help. This is because the increased capacity helps it model the complexity of the dataset better. They also employed a somewhat similar setup to SAGAN [31] whereby they used Spectral Normalisation [32] (discussed in 3.5.1) and many of the same hyperparameters as well as using orthogonal initialisation and what they call the "truncation trick" (a trade-off between sample variance and how realistic they look). They also found that adding SpectralNorm and BatchNorm into the generator but only SpectralNorm in the discriminator avoided crippling training.

Likewise, we use these same combinations of normalisation techniques to improve stability and increase sample variety. We also increase the batch size and number of filters to improve the quality of images generated, however are not able to match the model sizes and batches used in this paper as they were achieved with very powerful and expensive hardware (Google TPUv3 Pod - over 100 petaflops of computing power).

3.3 Image Super-resolution

Image super-resolution (SR), in particular single image super-resolution is a class of techniques concerned with improving the inherent resolution of an imaging system. There are two major threads to this field of work, the first is optical SR which involves itself with transcending the diffraction limit of optical devices and the second is geometrical SR which is more to do with digital image processing and enhancement. Our work will be focusing on the second type as we aim to apply digital post-processing techniques on existing images without interfering with existing image capture hardware.

Super-resolution can further be applied to several enhancement tasks. These include single frame de-blurring to combat de-focus and aberrations typically by using spatial-frequency filtering. We also have sub-pixel localisation where image features at their native resolution may exist over multiple discrete pixels, however by measuring their relative intensities you might be able to compute their centroid and locate them within a sub-division of each pixel. This technique is the basis for applications such as super-resolution microscopy.

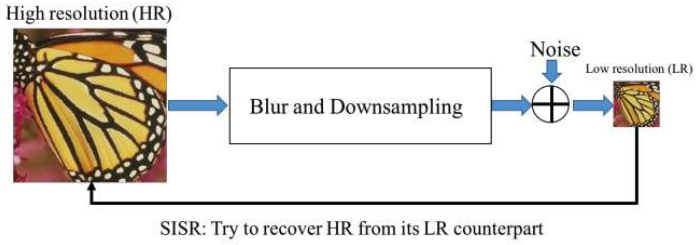


FIGURE 3.8. Diagram of the general setup for SISR problems [33]. From a base high-resolution image one can downsample and corrupt the image and then learn an algorithm to reverse that process.

In computer vision the task of recovering a high-resolution image from a single low-resolution image is known as single image super resolution (SISR). It is an inherently ill-posed problem due to the fact that for a single low resolution image there exists a multitude of possible high-resolution representations. This means it classifies as an undetermined inverse problem with non-unique solutions, making it particularly challenging.

3.3.1 Classical Super-resolution

The typical ways in which to resize images for the vast majority of cases is using a simplistic nearest neighbour (NN), bilinear or bicubic up-sampling function. These take a very general approach to inter-pixel interpolation and the outputs are always either too grainy (as in nearest neighbour) or overly smooth a lacking high frequency detail (as in bilinear and bicubic).

Approaches to circumvent the undetermined inverse issue are to impose a reasonable image prior to follow real image statistics or use something like a total variation prior for denoising. Another popular approach is example-based such as sparse coding (SC). Sparse coding (also known as sparse dictionary learning) is a representation learning method that tries to find a reconstruction of the input data using a linear combination of basic features. These basic features (called "atoms") are often an over-complete spanning set D (the dictionary) and their coefficients α that define the linear combination reconstruction. A sparsity constraint is also imposed on the co-efficients α. Sparse coding SR uses 2 dictionaries of image patches from both low-resolution and corresponding high-resolution images and given co-efficients of the low-resolution dictionary can learn a mapping to the high-resolution dictionary and thus recreate a high-resolution image [34].

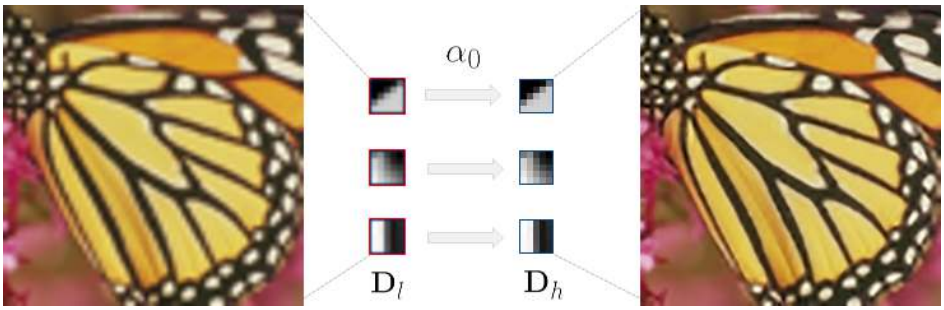


FIGURE 3.9. Sparse coding SR learns a mapping from a low-resolution to a high-resolution dictionary of image patches [34].

3.3.2 Super-resolution using Deep Learning

The first big breakthroughs in super-resolution came about by using deep convolutional networks which acted in a similar manner to sparse-coding approaches except rather than

explicitly learn the dictionaries, these features were implicitly learned in the hidden layers on the CNN. This made them end-to-end learners with little pre/post-processing [35].

3.3.2.1 Super-resolution Convolution Neural Network (SRCNN)

SRCNN [35] pioneered the effective use of convolutional networks to tackle the single image super-resolution problem. It achieved a significantly higher peak-signal-to-noise (PSNR) ratio than sparse coding, the classical metric for image quality and sharpness. Despite their success they still elicited some room for improvement. By performing bicubic up-sampling of input images to high-resolution space and then learning the convolutional features over the top of that it meant that all processing was done in a high-dimensional pixel space making it computationally inefficient. This bicubic up-sampling is also not a realistic approximation of the down-sampling kernel we aim to invert. Furthermore their network was only 3 layers deep and hence very shallow compared to current models. Finally by using a mean-squared-error (MSE) loss function between outputs and the ground truth this imposes an inherent smoothness on generated images that fails to fully capture the details of a true high-resolution image.

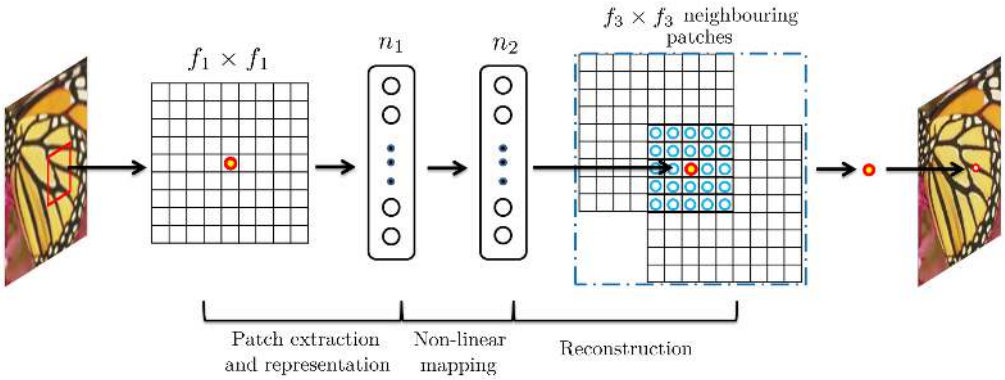


FIGURE 3.10. SRCNN learned convolutional feature maps can be viewed through the lens of the sparse coding framework [35].

3.3.2.2 Very Deep Super Resolution Network (VDSR)

Very deep super resolution networks such as VDSR [36] further improved the results of convolutional SR models. By extending the network to a full 20 learnable layers (inspired

largely by the VGG image classification model [37]) this hugely increased the representational capacity of the model as well as allowing it to learn deep, non-linear relationships. They also simplified the problem to only learn the residual from the interpolated low-resolution image to the high-resolution target.

3.3.2.3 Enhanced Deep Super-resolution Network (EDSR)

Enhanced Deep Residual Networks [38] took inspiration from both VDSR and SRResNet (which itself was more or less just a standard ResNet [39] re-purposed for SR). Residual networks (ResNet) were a groundbreaking model that introduced skip connections around groups of layers which allowed gradients to flow much more effectively in very deep models, leading to much faster training and hugely improved accuracy. EDSR took out and modified a few of the features of the SRResNet model that were not beneficial such as BatchNorm [29] which they found to limit the flexibility of the model by normalising features. This also decreased memory overhead by a considerable amount. They also found that using models with a large number of feature maps made the computations numerically unstable and hence introduced a residual scaling factor of 0.1 to the skip connections to improve stability. They

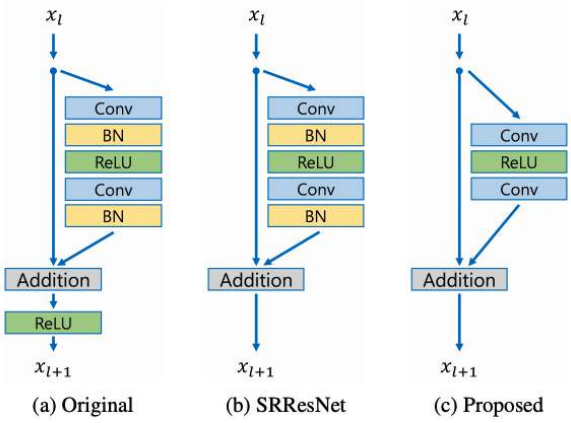


FIGURE 3.11. Comparison of the residual block structure in a.) ResNet, b.) SRResNet and c.) EDSR, the primary modification being the removal of BatchNorm layers [38].

also found that initialising model parameters with pre-trained weights from lower up-sampling factors (e.g. 4x up-sampling model initialised with 2x model parameters) greatly accelerated training and final performance. This is something we will absolutely look to use ourselves to

speed up training multi-resolution models. This model performed particularly well and went on to win the NTIRE Super-Resolution Challenge 2017 [40]. We use this residual approach in our own work and find that it helps greatly with sample quality and training speed. We similarly remove BatchNorm for super-resolution tasks and find that it improves results as they found.

3.3.3 Checkerboard-free Up-sampling

One technical difficulty in convolutional up-sampling layers is the presence of unwanted checkerboard artifacts. This is a core issue to tackle in our super-resolution models since we are always going from a smaller, lower resolution image to a higher one and hence we must perform some sort of up-sampling. Furthermore, it is a massive point of consideration when using GANs since a generator that produces artifact-ridden images can be trivially classified as fake by the discriminator, causing very little of use to be learned by either network.

FIGURE 3.12. An example of prevalent checkerboard effects in images, unfortunate side effect of the asymmetry of tranpose convolution (top row). This can be avoided by using more sophisticated upsampling techniques like resize-convolution (bottom row) [41].

The most simple and naive approach to deconvolution is often referred to as "transposed convolution". This involves every pixel in low resolution space being projected onto a larger output size (given by the size of your kernel) [42]. However a major issue with this occurs when the kernel size is not divisible by the stride and you run into uneven overlap when passing the convolutional filter over your image. This is even more pronounced for convolutions in 2D space than 1D. The overlap causes certain regularly spaced output pixels to be convolved

more times than their neighbours and hence producing the checkerboarding pattern. This gets compounded when we add successive layers of this type too [41].

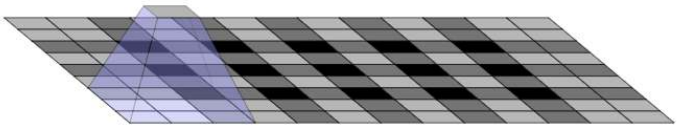


FIGURE 3.13. Uneven overlaps of a deconvolution kernel operation in 2-dimensions leads to perceivable checkerboarding in up-sampled images [41].

It might be expected then that the network can simply just learn to account for this unwanted overlap in it's weights. However this is difficult and usually not achieved. It is often the case that not only uneven overlaps have this problem but also even overlaps actually get learned by the kernel, producing similar artifacts. Having to learn this inherent imbalance also takes away useful capacity from your filters, making less available to actually learn what you want it to learn [41].

The solution proposed by Odena et al. is to use a kernel size that is divided by your stride which is equivalent to sub-pixel convolution [43]. This on its own does not fix all the issues though. They also use a "resize convolution" and involves performing a nearest-neighbour up-sample of the input image and then a 1x1 convolution to learn the high resolution image.

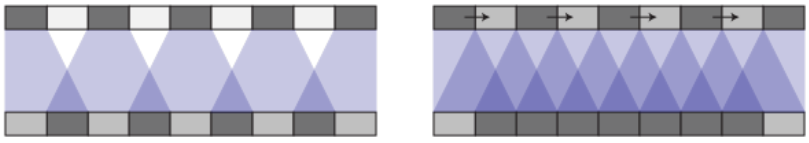


FIGURE 3.14. An illustration of uneven overlap in transpose convolution (left) compared to performing a NN-resize before convolution in a resize-convolution to avoid checkerboard overlaps (right) [41].

However one large drawback of this approach is the necessity to learn convolutional filters in the high-resolution space (i.e. post-resize). This requires more parameters and is slower to compute with more memory overhead. An alternative to this is to use sub-pixel convolution (also known as pixel-shuffle) [43] which learns to map the low resolution image to a series of sub-pixel filters which are then shuffled together in the final high-resolution image. One

caveat to this is that neighbouring pixels when shuffled together come from different filters and hence have different weight initialisations. This on its own creates a kind of checkerboard pattern. However a simple remedy is "Initialization to Convolution NN Resize" (ICNR) [44] which involves copying the initial weights of the first filter across to all other filters and hence maintain consistent weight initialisation between sub-pixels corresponding to the same low-resolution source pixel.

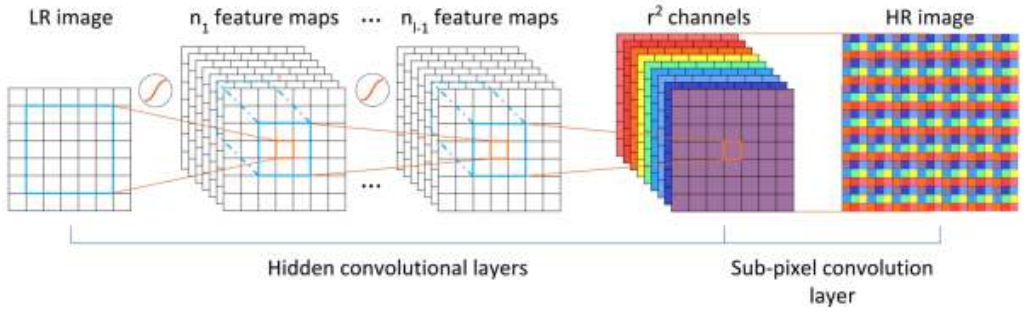


FIGURE 3.15. Sub-pixel convolution and shuffling where r is the up-sampling ratio [43]. Each pixel in the low-resolution image is expanded to r^2 sub-pixels via the intermediate feature maps which are then shuffled next to one another to form the corresponding high-resolution pixels.

One more recent paper [45] has claimed that pixel shuffle upsampling layers are still not perfectly checkerboard-free in their findings. They argue that pixel shuffle upsampling layers do not satisfy the condition of maintaining a steady state DC output value from a unit step response, instead producing a signal with period U where U is the upsampling factor. Hence they propose applying a zero-order hold H_0 with factor U to the output of the upsampling layer to smooth out the periodic signal and allow for a checkerboard-free output. We implement this by applying a Gaussian blur (achieved by using an AveragePool layer on the output with pool size U) to smooth over neighbouring pixels in the radius of U pixels. We find this to help remove subtle checkerboarding in some cases and hence leave it as an optional flag in our models.

The second major cause of checkerboard artifacts is due to asymmetric or uneven gradient updates, this time stemming from backward propagation rather than forward propagation (as for deconvolution artifacts). This is very common when using operations such as Max Pooling because at every step of backpropagation, gradients only flow through the single pixel

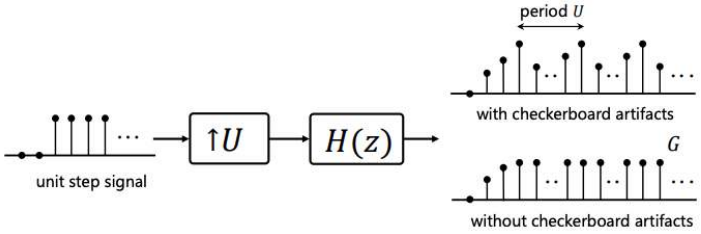


FIGURE 3.16. Diagram of the periodic nature of linear interpolators with upscaling factor U [45].

that is maximally activated in the convolutional filter's receptive field. The best way to avoid this is to simply avoid Max Pooling and instead opt for strided convolutions [28].

3.4 Super-resolution using GANs

Another major improvement in performance to the task of super-resolution came about with the application of Generative Adversarial Networks. Because the discriminator acts as a complex learned loss function, it allows these networks to model very fine-grained and highly photo-realistic detail not possible with pure CNN approaches.

3.4.1 Super-resolution GAN (SRGAN)

SRGAN [22] was a breakthrough paper that attempted to solve the overarching problem of a noticeable lack of finer texture details with deep CNNs for super-resolution. This work aimed at overcoming the shortcomings of over-simplified CNN pixel mean-squared-error loss functions. By introducing a much more sophisticated adversarial GAN loss this helps in pushing generated images onto the natural image manifold rather than dealing all the time with blurry averaged image representations.

Furthermore they integrated a content loss function (first proposed by Johnson et al. [46]) and were able to produce remarkably more photo-realistic and detailed super-resolved imagery than seen previously. Content loss is a very different approach to pixel-wise loss because instead of naïvely comparing individual pixel values, it leverages an auxiliary image classifier to compare image semantics. This is achieved by passing both the generated and ground-truth

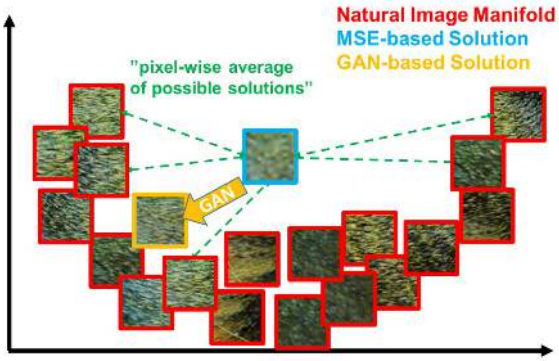


FIGURE 3.17. GANs (yellow) aim to sample from the natural image manifold (red) rather than producing an average image over the manifold like all MSE solutions (blue) which result in overly smooth images [22].

images through a network such as VGG-16 and taking the L2 difference between certain layer activations for either image.

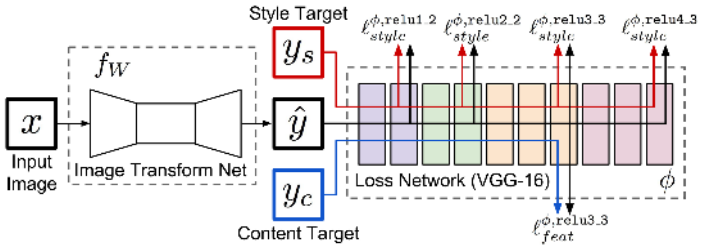


FIGURE 3.18. Diagram of how intermediate activations of a classifier network are utilised for content loss [46].

FIGURE 3.19. The advantage of content loss for resolving semantically meaningful features [46]. Important features like lashes and the iris are resolved to be far less blurry. Metrics listed in the brackets are PSNR/SSIM respectively.

We make extensive use of this content loss approach in our own work, however rather than using a VGG network trained on ImageNet, we employ a custom pre-trained aerial-image specific semantic segmentation model (courtesy of the Nearmap AI team). Furthermore we explore the use of Dense blocks for super-resolution tasks and compare them to simpler Residual blocks.

This semantic image loss has proven to achieve much more realistic looking images even though the PSNR is often lower than MSE pixel approaches, illustrating why pixel-based PSNR optimisation might not be the ideal metric for these image tasks. Often a common metric for the subjective evaluation of image quality is the Mean Opinion Score (MOS) which is simply an arithmetic mean of scores given by human evaluators on a predefined scale (usually 1-5).

We base a great deal of our work around this seminal paper, using it as inspiration to formulate the GAN approach to super-resolution. However, since this paper uses older and less effective architectures such as its SRResNet generator, we look to make a significant improvement on top of this by using more recent and sophisticated architectures (e.g. EDSR[38], RDN[47]) within this GAN framework.

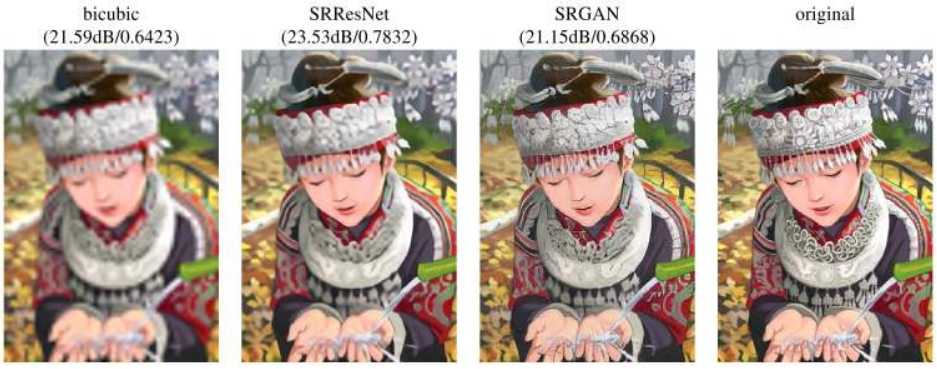


FIGURE 3.20. A comparison of super-resolution models highlighting the vast improvement in detail captured by SRGAN. PSNR and SSIM are listed in the brackets [22].

3.4.2 Enhanced Super-resolution GAN (ESRGAN)

Enhanced SRGAN (ESRGAN) took the aforementioned model a few steps further to deal with resolving finer detail as well as amending some of the unpleasant artifacts often accompanying hallucinated features. They modified the backbone residual block of the network to remove BatchNorm as well as add Residual-in-Residual Dense Blocks (RRDB) which were inspired by Residual Dense Networks (RDN) [47]. These RRDB blocks are densely-connected, sharing a lot of values between them via concatenation operations (as opposed to addition in residual connections) which has often been found to be advantageous in applications where high-fidelity reconstructions are vital [48]. This is because the inputs to earlier layers are essentially passed forward into the network and appended to deeper layers, preventing information from being lost in transit through the many deep layers of the network.

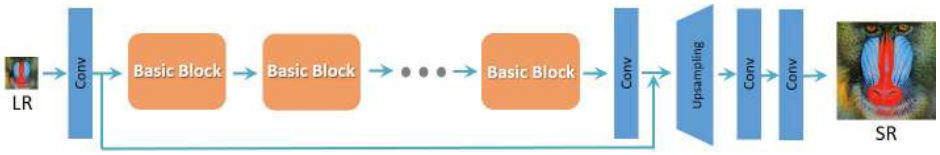


FIGURE 3.21. Overall generator architecture of their proposed RRDN network [23].

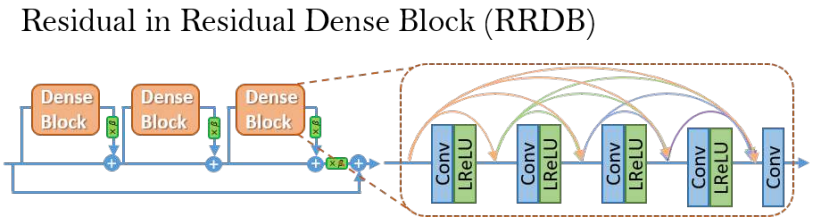


FIGURE 3.22. Connection structure of a single RRDB dense block [23].

Furthermore they also made it follow a relativistic GAN model [49], in other words to let the discriminator predict relative rather than absolute "realness". They also improved their perceptual loss function by evaluating the loss of layers before rather than after activations which they found to improve brightness consistency and texture recovery. We also use this pre-activation trick in our content loss function.

Lastly they also introduced an interesting mechanism to be able to trade off between direct PSNR pixel-wise optimistation and the more visually pleasing but also more hallucinated GAN

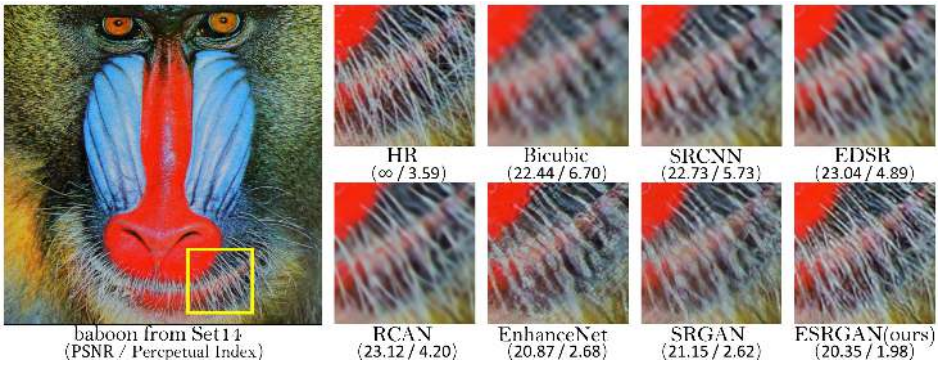


FIGURE 3.23. Comparison of various SR techniques, showing the almost perfect reconstruction of detail by ESRGAN [23].

objective. This involved interpolating the network weights between a PSNR trained objective and a GAN objective which produced much more perceptually consistent interpolation that doing so in pixel space. This is a potentially valuable technique that we can use in our own model to balance the GAN hallucination trade-off.

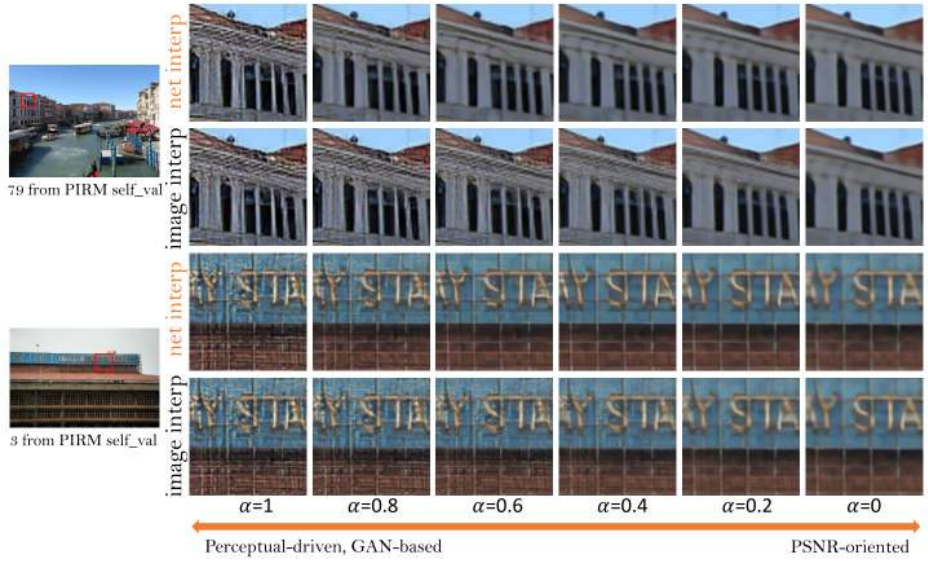


FIGURE 3.24. Comparison of interpolation between PSNR and GAN objectives for network and pixel-based interpolation [23].

3.5 State-of-the-art GAN Architectures

3.5.1 Self-Attention Generative Adversarial Network (SAGAN)

Self-Attention Generative Adversarial Network (SAGAN) [31] introduces the popular self-attention mechanism [50]. This allows SAGAN to learn much longer-ranged dependencies for image generation tasks and not just the dependencies of spatially local points. This allows it to ensure better global consistency of features across the image and allows convolutional GAN models to not be restricted necessarily by their limited receptive field. This was backed up by visualisations of the attention layers which showed that the generator indeed was leveraging regions that corresponded to object shapes rather than local regions of fixed shape.

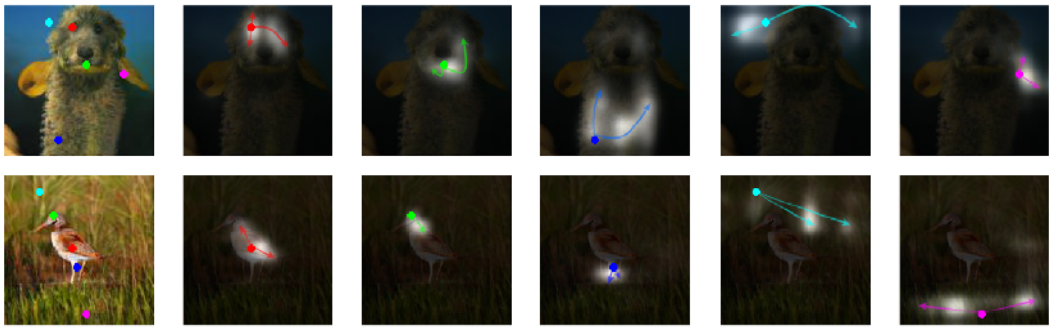


FIGURE 3.25. Visualisation of the attention maps for different features of an image. It is evident that semantically relevant portions of the same object are attended to rather than just a fixed receptive field region [31].

Another important inclusion made by this paper is to use Spectral Normalisation (SpectralNorm) [32] for the generator as well as the discriminator as it has been shown that generator conditioning affects GAN performance greatly [51]. Spectral Normalisation is a technique for weight normalisation that has demonstrated huge benefits in stabilising the training of discriminator networks. It aims to control the Lipschitz constant of the discriminator so that the gradients of the network remain bounded. This is important in adversarial training as large, unbounded gradient updates easily lead to massive training instability and hence divergence between the generator and discriminator.

Let's say our discriminator $D : I \to \mathbb{R}$ maps a real image I to a scalar binary classification output. For our discriminator to be K-Lipschitz continuous we must have:

$$\|D(x) - D(y)\| \leq K\|x - y\| \quad \forall x, y \in I \tag{3.1}$$

For linear functions such as the layers of a neural network $g(\mathbf{h}) = W\mathbf{h}$, the Lipschitz constant is the largest singular value or *spectral norm* of the weight matrix W [**?**]. The spectral norm (L_2 matrix norm) of a matrix A is this defined as:

$$\sigma(A) := \max_{h:h\neq 0} \frac{\|A\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = \max_{\|\boldsymbol{h}\|_2 \leq 1} \|A\boldsymbol{h}\|_2 \tag{3.2}$$

And hence our spectral normalisation process to ensure satisfaction of the Lipschitz constraint $\sigma(W) = 1$ is achieved as follows:

$$\overline{W}_{\mathrm{SN}}(W) := W/\sigma(W) \tag{3.3}$$

By satisfying that the discriminator is 1-Lipschitz continuous, we maintain a definitive bound on the gradients of the discriminator which prevents massive spikes in the update signal it feeds back to the discriminator, greatly improving training stability and convergence. In practice we estimate the spectral norm $\sigma(W)$ using power iteration since using Singular Value Decomposition (SVD) to compute it is computationally expensive and often unnecessary.

Their model managed to greatly improve the then best Inception Score (IS) from 36.8 to 52.52 and reduce the Fréchet Inception Distance (FID) from 27.62 to 18.65 on the challenging task of generating from ImageNet. These metrics will be discussed in a later chapter.

From this paper we ourselves integrate SpectralNorm and find it to be a key factor in achieving convergence in GAN training. We also recognise self-attention as a potential way to overcome the limitation of finite, local receptive fields in convolutional layers and capture longer range, global image features and dependencies. Unfortunately do not get the chance to implement it in our own work and it is left as an interesting future extension.

3.5.2 ProGAN

Getting a GAN to produce large, high-resolution images was for a long time a major challenge. The technique of progressively growing GAN images [52] involves building up the resolution of images handled by both generator and discriminator steadily. Typically one would go from extremely low resolution image synthesis on 4x4 images up to full high-resolution images of size 1024x1024. This approach produced the first GAN network to be able to create large scale, high quality imagery with good image variation. This approach was sensible because creating large scale images from scratch is an undeniably hard task. Based on the concept of curriculum learning (learning from progressively harder problems) it makes sense to first learn simpler representations that feed into larger, more complex ones. We use this paper as an inspiration for large scale image synthesis and employ many similar parameter choices and training tricks from it. However due to ProGAN not being an end-to-end training procedure and requiring tricky and sensitive hyper-parameter tuning, we choose to avoid the explicit progressive growing of GAN images in our own work.

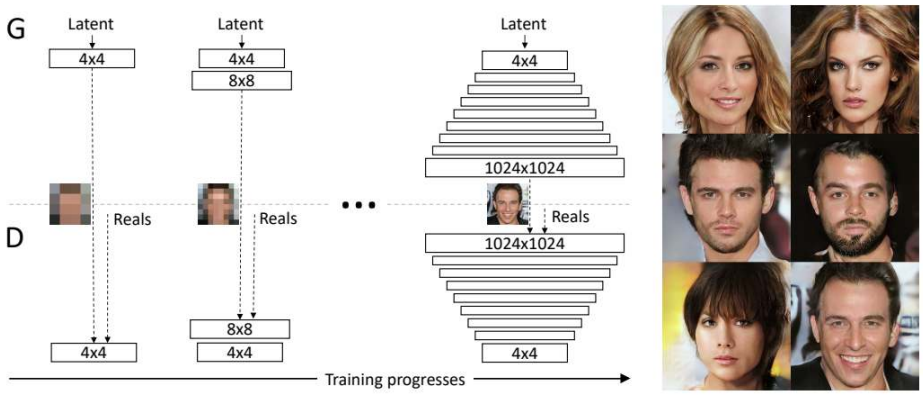


FIGURE 3.26. Diagram showing how ProGAN's progressively larger layers and outputs are blended into the model during training, allowing it to grow to full size HD image generation on the CelebA-HQ dataset [52].

3.5.3 Multi-Scale Gradient GAN (MSG-GAN)

Multi-Scale Gradient Generative Adversarial Network (MSG-GAN) was a further paper that looked towards improving high quality image synthesis from GANs. A commonly accepted

reason for the difficulty of image synthesis in GANs is due to uninformative gradients from the discriminator to the generator providing very little information causing a learning imbalance. They proposed a GAN model that learned representations at multiple resolutions and shared these different scales of outputs between generator and discriminator to provide more informative signals for feature learning.

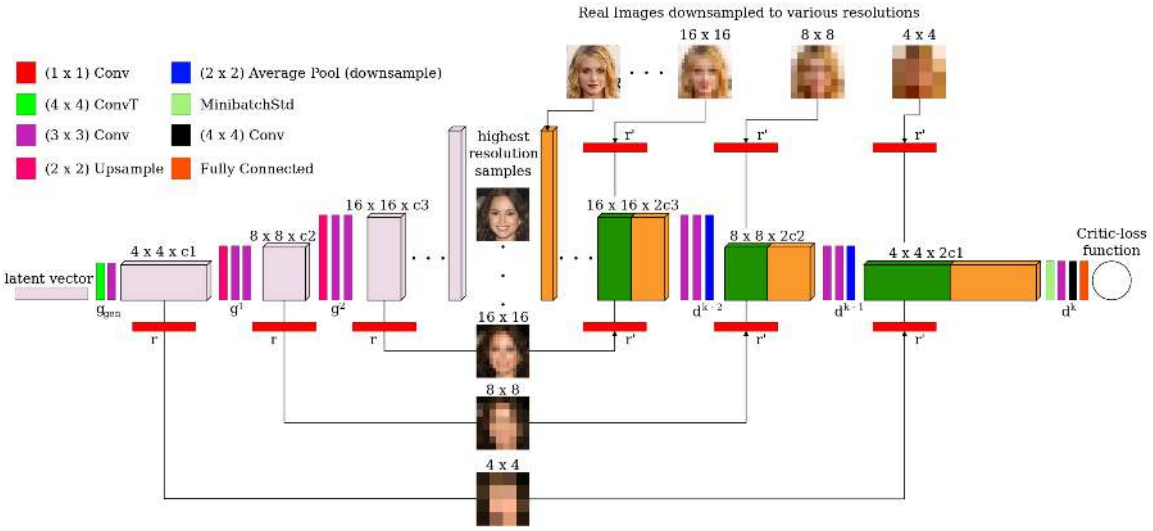


FIGURE 3.27. Network architecture of MSG-GAN showing how features are learned at multiple scales and parameters are shared for these resolutions between both sides of the network [53].

This process turns out to be quite effective in providing stable gradient signals between networks. Because it learns at different scales it is also noticeably fast to learn and is able to learn larger image generation in a similar way that ProGAN would by building up from smaller features. However this approach requires a lot less careful parameter tuning and is an end-to-end training process rather than requiring training in stages. Last of all the disadvantage in ProGAN is that by the time it learns the largest image generation, it has often squandered its earlier, low-resolution representations, whereas MSG-GAN retains this information across all scales throughout the training process.

We use this approach extensively as our baseline for generating high-quality GAN images as it is both fairly stable to train, faster to train due to more gradient sharing and in the end gives us quite pleasing results. We make our own alterations to this by modifying some of

the layers and their parameters as well as swapping out their equalised learning rate (which seems similar in effect to WeightNorm) for SpectralNorm as we find it to be more reliable. Applying this multi-scale learning to the super-resolution task rather than image synthesis is something that we have not seen much in the literature (although [38] approaches multi-scale super-resolution in a fairly different, non-GAN way in their MDSR network). We are hopeful that this approach could provide considerable benefits over current GAN architectures for super-resolution due to all the aforementioned strengths discussed.

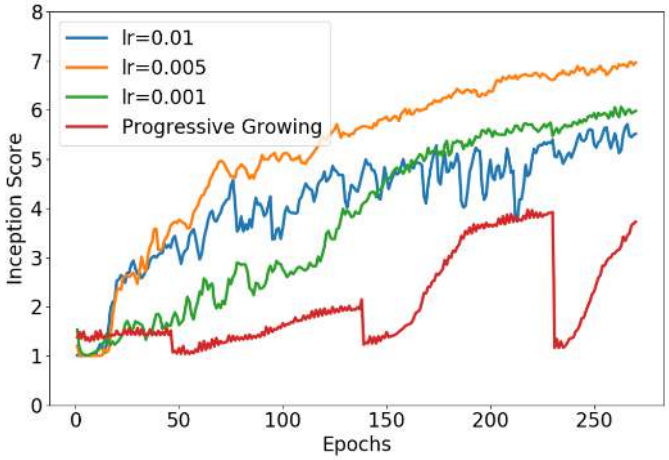


FIGURE 3.28. Comparison of Inception Score (IS) as a function of training epochs. MSG-GAN trains more stably and consistently than ProGAN and with better final scores [53].

3.6 Achieving Stability in GAN Training

A major consideration when training GAN models is that it is difficult to know for certain if your model is improving. This is because the loss is adversarial and essentially a zero-sum game between the generator and discriminator, hence the loss does not monotonically decrease like in typical neural networks. In fact in the ideal stable, case the loss would remain level and balanced for both networks during training.

Finding a balance between the strength of either network is often very elusive since it often occurs that the stronger network tends to overpower the weaker one in the long run and eventually one gets far too good for the other and then neither tend to learn anything and

training stagnates. We often refer to this as instability or a failure to converge in GANs where the networks cannot find a reliable Nash equilibrium. Arjovsky et al. [54] highlight the fact that GAN training instability is largely due to the passing of random, uninformative gradients from discriminator to generator when substantial overlap between the supports of the real and generated distributions does not exist.

There are a multitude of reasons why this often happens and here we will discuss in depth some of the crucial techniques that we discovered and tested from the literature to greatly improve these issues.

(1) **Input Normalisation**

Normalise inputs between -1 and 1 using $\tanh$ as your final layer activation function [55]. This helps universally with faster training and helps to avoid issues with vanishing and exploding gradients in deep multi-layer networks [56]. We found this to work well and hence used it as the universal normalisation scheme for our experimentation.

(2) **Modified GAN Minimax Loss Function**

In practice using $\max(\log(D))$ rather than $\min(\log(1-D))$ in the original GAN loss function greatly alleviates issues with vanishing gradients because it provides high gradient signals for the generator when it is poor whilst the discriminator is already quite good early in training. This is because when a sample is fake (label 0) we want the gradient signal from the discriminator to be high so that the generator learns quickly [1]. As we can see the gradient for the $\log(D)$ formulation is much steeper and provides a high gradient signal. See figure 3.29. We implemented our GAN loss function to follow this improved non-saturating GAN loss (NS-GAN).

(3) **Choice of Optimiser**

ADAM [57] with a $\beta_1 = 0.5$ (or even $\beta_1 = 0$) momentum term is typically the most commonly used option. The lower momentum is very helpful in preventing overly aggressive optimisation which is key due to GAN's inherent instabilities and back-and-forth training regime. We chose to use $\beta_1 = 0$ for the most stable training and used ADAM/RADAM (rectified ADAM) [58] as our optimiser.

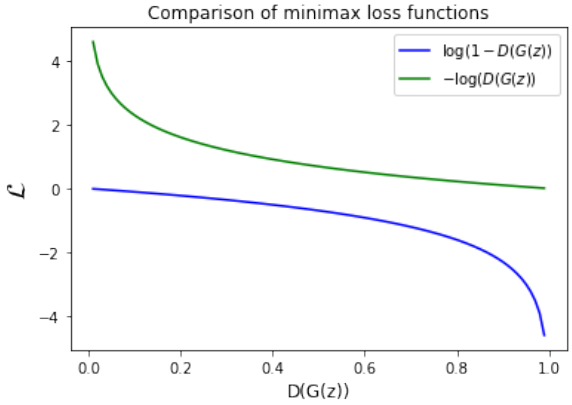


FIGURE 3.29. Comparison of minimax loss objectives for the generator. Blue indicates the original generator loss, which exhibits a shallow gradient for samples detected as fake (D=0). The non-saturating re-formulation in green has a much sharper gradient and hence provides much faster training early on.

(4) **Monitor Failures Early**

Knowing the failure modes and general trends of GAN training is important to be able to diagnose problems early. If the discriminator loss tends to zero then it is typically an unrecoverable failure. The ideal loss profile for the discriminator is to stay relatively flat (maybe even slightly decreasing) and most importantly with low variance. When the discriminator has high variance and regular spiking it is usually a sign that something is amiss. Lastly, a really useful trick is to monitor the gradient norms of either network. If the magnitude of the gradient starts to exceed 100 it is usually a certainty that the model is unstable and having difficulties in training [55].

(5) **Discriminator Noise**

Adding Gaussian noise to the input of the discriminator ("Instance Noise") helps it become more robust to variations in the input images and not overfit as hard [59]. Huszar et al. [60] hypothesise that the discriminator is very prone to overfitting and there also may exist a large set of near-optimal discriminators (i.e. non-unique) with losses close to the Bayes optimum. Since each of these near-optimal discriminators might provide very different gradients (for the same generative model q_θ and true data p but just based on different initialisations) this would mean the generator would not tend to receive many truly useful gradients. They argue that by crippling the discriminator, even though the Jensen Shannon (JS) divergence is constant locally in

θ, the variational lower bound doesn't have to be (and indeed is not tight when the discriminator is handicapped). Hence you may end up with a non-constant function of θ that may roughly guide you in the correct direction.

A more generalised view of crippling the discriminator is seen in how we don't training it until convergence (but rather update after one step). This is an extreme form of early stopping which acts as a regulariser, preventing overfitting. Some papers also add noise to generator [61], however we find this to be counterproductive as we typically encounter the situation of having an already weaker generator.

Similarly, using dropout [1] in your networks helps with crippling the performance, which is beneficial to balance the relative power of both networks as well as acting as a weak regulariser [62].

We found that this helped us in image synthesis tasks to prevent overfitting, however saw little benefit for super-resolution as it reduced the overall effectiveness of the discriminator (and training for super-resolution was already far more stable than image synthesis).

(6) **One-sided Label Smoothing**

Along a similar line of reasoning as the above point we also found that adding label smoothing and noise helped with regularisation of the discriminator. One-sided label smoothing involves replacing the original real label targets (label 1) with smoothed values chosen as a random number $y_{real} \in [0.7, 1.0]$ [63] [64]. This helps to lower the confidence of the discriminator and prevents it from falling into the situation of being absolutely confident about positive examples. To take this even further it is popular to also fully flip a small percentage of the labels to the opposite category [63]. Like disciminator noise, we used this technique to good effect in image synthesis to decrease overfitting, however did not deem it necessary for super-resolution.

(7) **Larger Kernels and more Filters**

Using the right size kernel is important. It should be large enough to capture sufficient

[1] A popular regularisation technique that involves randomly deactivating a subset of neurons every iteration to prevent strong local dependencies in network connectivity and reduce overfitting.

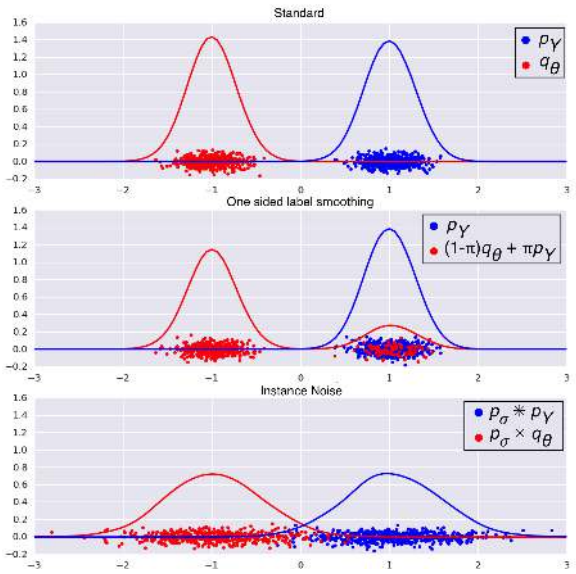


FIGURE 3.30. Comparison of the effects of one-sided label smoothing and instance noise for regularising the discriminator [60].

detail in its receptive field and be aware of global features (and not having tunnel-vision to just local features). However having overly large filters is also problematic and should be avoided.

Increasing the number of filters is usually beneficial to be able to capture more information relative to the complexity of the dataset to be modelled. Often this results in sharper edges and finer features. However it should be noted that if the model is flawed in other regards then increasing the filter size and just adding more parameters is almost certainly guaranteed to not fix those stability issues. We found that kernel sizes of 3 and 4 still seemed to work the best and it was also clear that for well-behaved models increasing the number of filters almost always yielded a small improvement in performance.

(8) **Avoid Sparse Gradients**

Using sparse gradients from layers like ReLU and MaxPool becomes very problematic for the stability of GANs. It is a much better alternative to use Leaky ReLU so that small negative gradients can pass back through and are not completely rectified. Furthermore it is preferable to use Average Pooling or Strided Convolutions for down-sampling operations and Pixel Shuffle [43] [44], Resize-convolutions [41] or

Fractionally-strided (a.k.a. Transpose) Convolutions for up-sampling. For all our models we used Leaky ReLU activations and made sure to avoid Max Pooling layers, opting instead for strided convolution layers to down-sample in the discriminator network (as pointed out in [28]).

(9) **PatchGAN Discriminator**

PatchGAN is an idea from Pix2Pix [62] that rather than letting the discriminator output a singular real/fake prediction for the entire image, it instead produces predictions on each patch of an $N \times N$ grid of that image. This means it essentially models an image as a Markov Random Field, where independence between pixels spaced greater than one patch apart is assumed. This is advantageous as a smaller PatchGAN has fewer parameters, runs faster and works for arbitrarily sized images. Furthermore because it acts on local "patches" of the image it ends up being a good critic of more local features, in particular style and texture. For our work it is straightforward to implement this by changing the output of the last layer of the discriminator to be convolutional (because convolutions output a 2D grid of values) rather than fully connected (usually outputting a single scalar). We tended to use a 4x4 grid of patches (i.e. patch size of $(image\_size/4, image\_size/4)$ in the hopes of capturing more local details and improving textures (which is important for super-resolution). The advantages we found to using this were that it allowed our network to work with any input size (since it had no dense fully connected layers) and it also was much more parameter efficient, since the number of parameters on the output layers no longer needed to be proportional to the input size.

FIGURE 3.31. Effect of patch size on image quality. 1x1 PixelGAN ignores spatial statistics while 286x286 ImageGAN lacks finer details in textures and style. The authors find a patch size of 70x70 to work best, which is a similar scale to what we ourselves choose [62].

(10) **Normalisation**

It is strongly recommended to normalise inputs as they pass through the network by using BatchNorm [55]. An important caveat to this is that BatchNorm should be applied to batches of real and fake images independently (i.e. no mixing between categories of samples). This helps keep the batch statistics consistent for the real and fake images. We find empirically that BatchNorm helps significantly with network training, and without it often the generator samples are featureless and not realistic at all. However we find it only really essential in the discriminator, and for memory and slight performance benefits we manage to avoid using it for our generator models.

Furthermore to improve the stability of training we find that using SpectralNorm [32] in our discriminator has a huge impact in ensuring that the networks don't rapidly diverge. We henceforth use SpectralNorm in almost all of our implementations as the massive improvement in stability is well worth the slight slowdown in training speed (due to weight updates being normalised). Certain recent papers [30] also promote the use of SpectralNorm in the generator network, however we find that this has a fairly minimal effect.

(11) **Two-time Update Rule (TTUR)**

To deal with the issue of the discriminator learning and converging much faster than the generator, Heusel et al. propose using individual learning rates for either network [65]. In particular they suggest using a higher learning rate for the discriminator than the generator. Although this may sound counter-intuitive at first, it is actually crucial to GAN stability that the discriminator learns faster than the generator. This is because the discriminator is responsible for the loss and hence gradient update signal for the generator, and without a strong discriminator to guide the generator it tends to not learn well at all. This has a similar effect to scheduling the discriminator to take 2 update steps for every step of the generator but overall requires less processing to be done as each network still only needs 1 step each. We find this to work quite well in practice, although increasing the learning rate of the discriminator too much relative to the generator does produce unstable training and poor sample generation.

For our purposes we typically adopt learning rates of $\ell_d = 0.0004$ or $\ell_d = 0.0002$ and $\ell_g = 0.0001$, which is quite commonplace.

3.7 Summary

In summary, we see that there is a clear advantage to using Generative Adversarial Network approaches over traditional up-sampling techniques and even standalone convolutional neural networks. The strength of using a GAN is that the extra discriminator network allows us to learn a fully customised loss function to capture complex elements of perceptual quality in images. This helps us in overcoming the major issue of blurriness and a gross lack of detail and texture when using models with an overly simplistic pixel-wise loss function. It is worth noting that common contemporary image quality metrics like PSNR and SSIM tend to fail to capture this improvement in perceptual quality, scoring higher for blurry pixel MSE optimised images.

Building from the work of notable papers like SRGAN and ESRGAN, we look to take this framework for training GAN networks for image super-resolution and make several improvements catering to the application of aerial image post-processing. Firstly, we swap out the generator SRResNet of SRGAN for the newer EDSR and RDN/RRDN architectures as they have shown to perform better when evaluated separately (i.e. without a GAN) mainly due to the removal of BatchNorm and in the latter the use of dense connections. We then look to apply better up-sampling methods like PixelShuffle to avoid the nasty checkerboard issues of deconvolutions, whilst also using less memory and computation than resize-convolutions. We also improve on SRGAN and ESRGAN by employing a patch-based model for a faster, more parameter-efficient discriminator that focuses more on important local texture detail. We then use the current best practices in GAN training, as outlined above, in particular adding spectral normalisation to allow for far more stable training.

Given that the loss function is critical in super-resolution, we tailor our network to the domain of aerial imagery by leveraging Nearmap's custom semantic segmentation network as our perceptual loss function, as it's trained specifically on aerial image data. Not only is using

a segmentation model rather than a classifier model as the perceptual loss a deviation from the literature, making use of a fully custom perceptual loss network rather than a generic ImageNet VGG model is certainly novel. We believe this is a crucial element in being able to instill our model with the necessary semantic information to super-resolve aerial images. Lastly, we observe that the use of GANs in aerial photogrammetry is quite rare, mainly remaining as a research exercise. We go beyond this to deliver a system that can both super-resolve aerial imagery as well as improve downstream semantic segmentation tasks (another new and unexplored application of super-resolution) and apply it in a production setting.

CHAPTER 4

Methodology

In our experimental approach we first frame the task as a typical single-image super-resolution problem and prepare a suitable dataset of low and high-resolution aerial image pairs. We then begin by implementing and training a baseline SRResNet model on 4x super-resolution, then moving to more sophisticated models like EDSR and RRDN. Meanwhile, we also compare the effectiveness of a pixel-based loss versus an auxiliary feature loss network which leads into our exploration of GANs as an important component in learning perceptually meaningful semantic detail. As a preface to our application of GANs to super-resolution, we briefly address the typical application of GANs to image synthesis (in our case of aerial imagery) in order to learn more about common GAN architectures and training techniques. We then apply this GAN knowledge back to the realm of super-resolution to intelligently up-sample aerial imagery at scales of 2-32x. Finally, we evaluate our performance qualitatively by comparing the perceptual characteristics of super-resolved imagery as well as quantitatively by measuring PSNR and SSIM image metrics as well as class-wise IoU of derived semantic segmentation maps.

4.1 Development Environment

To achieve our goals in this thesis, it was important to adopt good software engineering practices throughout the process. To manage our development environments we used Docker, a service for running containerised environments as a lightweight virtual machine (VM). This helped accelerate the software development cycle since it allowed for many software dependencies to handled consistently. This was useful because typically having to manage

python packages via Anaconda and CUDA GPU drivers across different machines can become very tedious and problematic.

Software testing is another essential component to any good software engineering project and hence we used CircleCI for continuous integration. This allowed us to test software packages continuously as we made changes and deployed them, meaning less bugs and quicker bugfix turnarounds. Integration testing is important in large software projects because there are often many moving parts and changing parts of the code in one place can fundamentally break functionality elsewhere.

For our actual writing of software we performed all our development using Python 3 in a Jupyter Lab notebook development environment. Jupyter is a key tool in the data science arsenal as it allows for users to execute code in standalone cells which greatly improves ones ability to quickly run small experiments and iterate on existing code. It also supports inline plotting, meaning results can be visualised from within the same panel and not needing external applications to be launched.

To build the neural networks that make up the backbone of our GAN pipeline, we use the popular deep learning framework Tensorflow (v2.0) from Google. We choose the very recently released 2.0 version over the more common 1.x predecessor due to the many improvements in the new release such as native Keras integration, eager execution and autograph as well as a major cleanup of the library API. Keras is a high-level API that sits on top of the Tensorflow back-end, greatly improving the speed and simplicity of creating custom network architectures. Eager execution allows for much more transparency into building Tensorflow models as it enables the user to inspect and debug the running of models and variables in a pythonic way which was previously not possible when models were only compiled to a computational graph.

On the hardware side, since deep learning is heavily computationally intensive we were lucky enough to be able to use NVIDIA GTX-1080 and V100 GPU processors from Nearmap's AI development compute cluster. This enabled us to run much more complicated models

and achieve far better results than if we only had access to a CPU, hence making this project feasible.

4.2 Dataset

To achieve the outcome of single-image super-resolution we need to create a custom dataset of images from different resolutions with the intent of learning the mapping from lower to higher resolution. Nearmap's imagery is stored as 256x256x3 image "tiles" which can be stitched together to create a full ortho-map. These tiles can be queried using their internal API which allows us to grab a large set of image tiles given a bounding box of lat-long co-ordinates and a date. Nearmap runs surveys every few months in most highly populated cities that they survey so it is important to consider the temporal aspect of this dataset. For our initial purposes however we will maintain a static dataset, using images all from the one survey to ensure consistency.

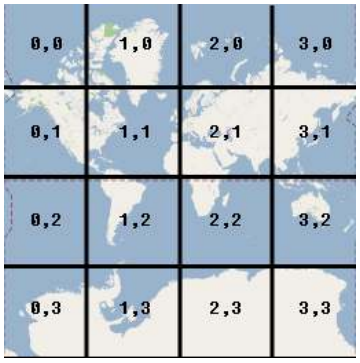


FIGURE 4.1. An illustration of the breaking up of a map projection into tiles for a given zoom level. Labelled numbers indicate tile co-ordinates [1].

Initially we created a dataset from the API by taking one super-tile (large contiguous block of land consisting of many tiles) and slicing it into many smaller component tiles to create a dataset of aerial images. We perform this operation for the same area at two different zoom levels to get a paired dataset of images from a lower and a complementary higher resolution. Initially we used zoom 16 and zoom 18 but found this to be too coarse and instead moved to zoom 19 and zoom 21 pairs to begin with. Each increase in zoom level

[1] Accessed from: https://developers.google.com/maps/documentation/javascript/coordinates

corresponds to a 2x increase in resolution, hence our zoom 19 and 21 pairs represented a 4x image up-scaling. The lower resolution image is typically derived from the higher one by means of down-sampling, except in the case of zoom 16 where images at this zoom level come from an entirely separate camera system (the "overview" camera). For down-sampling we used Tensorflow's `tf.image.resize` function with an anti-aliasing filter (hat/tent filter function with radius 1) applied.

(A) Zoom 19. (B) Zoom 21.

FIGURE 4.2. A typical example of an image pair used in our super-resolution dataset.

We also switched from pulling a super-tile [2] and using that for our training set and instead used Nearmap's "Apollo" dataset that is the AI team's standard training dataset for their segmentation models. This differs from the dataset we created because it uses adaptive sampling to collate many images from different non-neighbouring regions as well as oversampling some under-represented classes of images. We found this to be a much better dataset to train on because it was far more diverse (not just a block of adjacent tiles that all looked similar) and also cut down on the overwhelming number of grassy fields and trees that occur frequently in Nearmap's imagery.

For pre-processing we perform data augmentation to randomly rotate images and flip images. This helps with generalisation of the model. We also normalise image statistics from 0-255 represented as 8-bit integer tensors to 32-bit floats in the range -1 to 1. Normalisation is commonly known to help with improved gradient flow in neural networks due to the problem

[2] A large tile comprised of many smaller, neighbouring tiles.

of vanishing or exploding gradients in deep networks. Lastly to help speed up training we batch and pre-fetch our data so that computations can be performed in parallel. We use a generator to load data in batch-by-batch rather than trying to load in huge datasets all at once which more often than not would exceed the maximum available RAM. This data loading process was also implemented to be multi-threaded, leading to large speed-ups from the parallelised training process. Training in batches also allows us to use techniques like BatchNorm to normalise dataset statistics to a wider sample (than on an instance-wise basis) leading to more stable training. We choose a majority portion of our images to make up the training set and hold out a smaller subset for the evaluation set so that we can validate our results on unseen images.

4.3 CNN Super-resolution

As a precursor to Generative Adversarial Networks, we first focus on performing image super-resolution using standalone convolutional (CNN) models. This has a two-fold benefit as it first allows us to benchmark our later results against a conventional baseline and secondly will accelerate our GAN development as the generator half of the GAN network is still just a CNN model. Hence we can perfect our understanding of CNNs and create a sophisticated and high-performing CNN generator for super-resolution and then introduce a complementary discriminator and train the two adversarially to create our GAN. The added advantage to this is that since GAN models are inherently quite unstable, it has been common practice to first pre-train both the generator and discriminator models independently and then bring them together and train against one another. This circumvents the key issue of poor initial training whilst both networks are still very low-performing and hence cannot reliably offer the other network informative gradients. This pre-training approach has been shown to greatly improve training times and overall performance.

4.3.1 Baseline Model: SRResNet

We initially chose our CNN super-resolution model to follow the well established architecture of SRResNet [22]. This architecture uses multiple stacked residual blocks followed by several up-sampling blocks (the number of which depends on the scaling factor e.g. 4x requires 2 2x blocks) to create the final high-resolution output. Since this model is known to work fairly well, is fast to train and is fairly simple to implement, we decided this would be a good baseline to first build out.

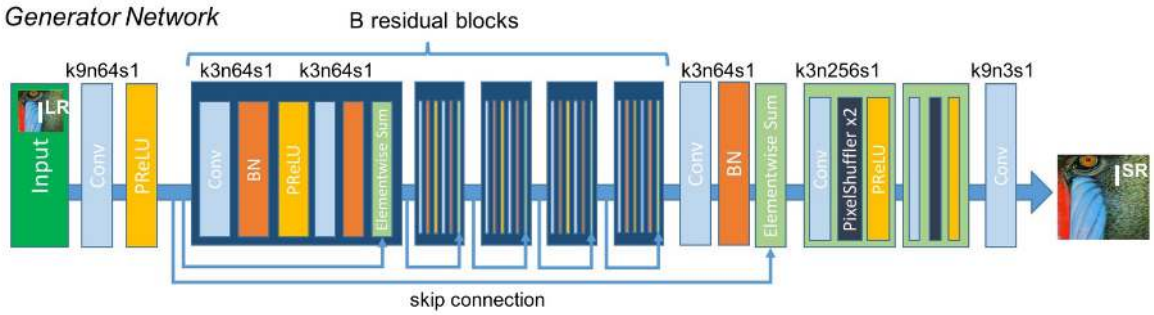


FIGURE 4.3. Generator network layer architecture of SRResNet [22]. The main backbone of the network is 16 stacks of residual blocks followed by 2 consecutive up-sampling layers to create the final 4x resolution image.

We train this model using a pixel-wise loss function. The two loss functions we investigate are Mean Absolute Error (L1) and Mean Squared Error (L2).

$$\mathcal{L}_{L1} = ||Y - G(X)||_1 = \frac{1}{HWC}\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{C}|y_{i,j,k} - G(x)_{i,j,k}| \tag{4.1}$$

$$\mathcal{L}_{L2} = ||Y - G(X)||_2 = \frac{1}{HWC}\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{C}(y_{i,j,k} - G(x)_{i,j,k})^2 \tag{4.2}$$

Here y is the high-resolution target image and x is the low-resolution input image. The indices i, j, k indicate the spatial coordinates of each pixel and colour channel respectively for an image of size (H, W, C). In our case for RGB colour images we have $C = 3$. Keep in mind these error metrics are often further averaged over a batch dimension N during training.

We train our SRResNet model on the aforementioned Apollo dataset, which for the purposes of our training we have configured to crop 256x256 sized aerial ortho-image pairs. We first

tackle the task of 4x super-resolution from zoom 19 to zoom 21 images as this is more challenging than the fairly simple 2x, but still provides a relatively difficult task. By using 4x as our baseline task, this allows us to compare performance across models fairly easily. Our network is trained using the ADAM optimiser with a initial learning rate of $lr_g = 2 \times 10^{-4}$ which is decayed by a factor of one half every 500 epochs. We have defined an epoch to be 1000 batches/optimiser steps, using a batch size of 12. We train the SRResNet model with 2 variations, the original model as it was presented in the original paper with BatchNorm, and another with BatchNorm removed.

For the up-sampling layers at the end we also use PixelShuffle as they recommend. This has the benefit (as discussed in 3.3.3) of avoiding the checkerboard artifacts of transpose convolutions and requires less memory and parameters than the checkerboard-free resize-convolution. However we also introduce ICNR kernel initialisation for these layers to alleviate the checkerboarding issues that affect vanilla PixelShuffle implementations in the early stages of training. Since we are pre-training our network anyway, we recognise that ICNR initialisation isn't hugely necessary since those early artifacts correct themselves as training progresses. ICNR is much more crucial if one skips any sort of pre-training and looks to train the GAN from scratch.

4.3.2 EDSR

Based on recommendations from the EDSR paper [38] we made a a few modifications to the SRResNet architecture. Firstly we removed the BatchNorm layers as they have been shown to reduce the representational power and introduce unwanted artifacts in super-resolution tasks. According to Lim et al. they "get rid of range flexibility from networks by normalizing the features" and by making this modification they reported a significant increase in performance as well as a roughly 40% saving in memory [38]. EDSR also changes the kernel sizes of SRResNet in its initial and final convolutional layers to be 3 rather than 9 and also uses ReLU rather than Parametric ReLU (PReLU) activations.

They also make the notable modification of forgoing activation functions outside of the residual blocks (i.e. removing them from the up-sampling layers). For our purposes we employ 16 residual blocks and 64 filters as stated in their smaller single-scale baseline model. Their full scale EDSR model is parameterised by 32 residual blocks and 256 filters which gives a slightly obscene 43 million parameters. Hence we stick to their smaller baseline model of 1.5M parameters for ease of training and speed, without sacrificing that much in the way of performance.

The authors also propose the concept of residual scaling for training deeper networks. This involves scaling the residual path of each residual block by a scalar e.g. 0.1, forcing the majority of the gradients to flow along the identity path. This helps greatly in stabilising training when using a large number of filters.

For comparability, we conduct the training of our EDSR model using the same optimiser and learning rate schedules as SRResNet, training also for 500 epochs at 4x super-resolution. We also similarly use 16 residual blocks and 64 filters. For our loss function we remain with L1 loss rather than L2 loss as this is commonly observed to yield slightly sharper results and exhibits greater training stability. We justify this choice of loss function later in our results section.

4.3.3 RRDN

As discussed in greater depth in 3.4.2, we also implement a third and final variant of our super-resolution network based on RRDN from the ESRGAN paper [23]. This architecture takes a slightly different approach by using densely connected blocks of convolutional filters which are then stringed together using residual connections. The advantage of this is being able to retain more information through the network by sharing connections in a concatenative manner, while still maintaining fast training times with local and global residual paths.

The ESRGAN paper also opts for Leaky ReLU activations, which are just like ReLU (Rectified Linear Unit) activations except they have a slight negative slope rather than being fully rectified. The authors recommend a negative slope of 0.2 which we ourselves use. Many

super-resolution papers also adopt this activation function over the PReLU (parametric ReLU) from SRResNet (which leaves the negative slope as a learnable parameter) as they find that there is marginal difference between them and PReLU is often slightly slower to train.

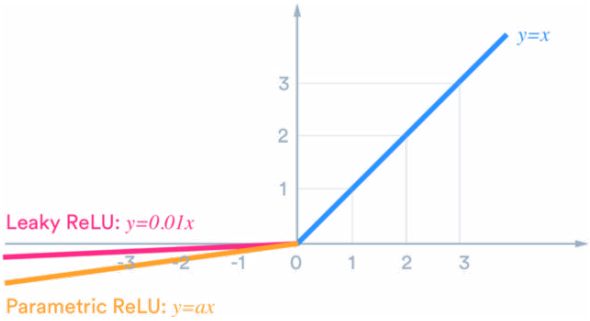


FIGURE 4.4. Comparison of different ReLU activation function variants: ReLU (blue), Leaky ReLU (pink) and PReLU (yellow) [3].

We base a large part of our implementation on the work done by [66] and hence carry over their naming scheme for the parameters of the RRDN architecture:

- T - the number of Residual in Residual Dense Blocks (RRDB)
- D - the number of Residual Dense Blocks (RDB) within each RRDB
- C - the number of consecutive convolutional layers inside each RDB
- G - the number of features per convolutional layer

We run a set of experiments using the same training setup as in SRResNet and EDSR to find the best trade-off in network parameters between training time and performance. We also test many of our own modifications such as Spectral Normalisation and output blur in the up-sampling layers to see their effect on the network. We also deviate from the ESRGAN paper to compare their use of resize convolution layers with pixel shuffle layers for the up-sampling part of the network. Lastly, we also use same the residual scaling parameter, β, and evaluate the performance of the network as it is recommended ($\beta = 0.2$).

[3] Source: https://medium.com/@chinesh4/why-relu-tips-for-using-relu-comparison-between-relu-leaky-relu-and-relu-6-969359e48310

4.3.4 Evaluation Metrics

Unfortunately, there is no one definitive best metric for the quality of the image. This is what makes optimising for image quality and resolution difficult. For the sake of simplicity we look at the PSNR, SSIM and MS-SSIM of images (as these are the traditional measure of image sharpness).

The Peak Signal to Noise Ratio (PSNR) defines ratio between the maximum power of a signal and the amount of corrupting noise commonly used in image and video compression analysis. It is a simplistic image quality metric that has been shown to correlate well with image quality when the encoding scheme/codec and image/video content are kept consistent, however is unreliable when this is not the case [67]. It has also been shown to not capture image quality as perceived by human observers well [68].

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{4.3}$$

$$= 20 \cdot \log_{10} (MAX_I) - 10 \cdot \log_{10}(MSE) \tag{4.4}$$

where MAX_I is the maximum possible pixel intensity (since we normalise pixel values to the range [0,1] this is then 1) and MSE is the pixel-wise mean squared error as seen in 4.2.

From the equation above it is clear that PSNR is optimised for when using MSE and MAE loss functions. However, we will see later on that despite this fact it still may not produce the most visually pleasing results (as judged by a human).

Our second metric is the Structural Similarity Index (SSIM) which aims to measure similarity between images. Like PSNR it is also a full-reference metric, meaning its estimation of image quality depends on an uncompressed ground truth reference image. SSIM was designed as a successor to PSNR and MSE image metrics that better captures perceptual quality. Rather than measuring absolute errors, it considers image degradation as the perceived change in structural information within an image [69].

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.5}$$

where x and y are the images to be compared, μ and σ are the mean and covariance respectively and c_1, c_2 are constants used to stabilise the division operation and are given in the original paper [69].

Multi-scale SSIM (MS-SSIM) is identical to the above SSIM metric that it is derived from, except it is a weighted combination of the SSIM evaluated at several resolutions. Each resolution is taken as a series of consecutive downsampling operations by a factor of 2.

For the implementations of each of these 3 image quality metrics we use the `tf.image` library. Any implementation details or parameters can be found in the respective documentation [4].

For a lot of our image quality comparisons we also tend to subjectively evaluate the images themselves based on their human perceptual quality. This helps to overcome the shortcomings of some of the above metrics and is necessary to track when unpleasant visual artifacts occur, however it is not entirely ideal. For the most part we rely on the PSNR and SSIM to indicate to us how well our network has achieved its MAE objective, however in further sections we will see how these metrics break down for maximising human-perceptual quality.

4.3.5 Training

For training of our CNN super-resolution networks we choose a batch size based on the size of images fed to the model to allow it to fit on RAM. For 256x256 pixel images on a GTX-1080 with 12GB RAM we found a batch size of 12 to fit on the GPU when using the larger RRDN models. Models often trained fairly quickly and good results could be seen within a matter of hours. The best models however were trained over a day or two. For the sake of consistency between model and parameter choices we trained each experiment for 500 epochs using the same dataset. As mentioned earlier we also used the ADAM optimiser for all tests, decaying

[4] https://www.tensorflow.org/api_docs/python/tf/image

the learning rate from 2×10^{-4} by a factor of one half every 200 epochs. These models served as our pre-trained base models from which we could transfer learn to newer models (e.g. for different resolution scales or using different objectives like a GAN), drastically cutting down training time by not needing to train from scratch every time.

We first performed individual experiments on each model architecture to find the optimal parameter and architecture settings for each, then took the best candidate models from each of SRResNet, EDSR and RRDN and trained them using identical settings (such as learning rate and batch size) so that the direct comparison between models remained fair. We then finally took the best performing model from all these architectures, which turned out to be the RRDN variant as it scored the highest performance without an excessive increase in model parameters, and trained it on all other resolution scales from 2-32x. We then compared the image metrics for this model across all these scales and saved these pre-trained models as the basis for transfer learning a GAN model for the resolutions in question.

4.4 GAN Image Synthesis

As a means to learn more about the fundamentals and later the specifics of GANs and how to train them effectively, we opt to first try our hand at the task of image synthesis. By first working on some simpler architectures and problems and gradually scaling the complexity we aim to eventually take what we have learned and directly apply it to our super-resolution GAN framework. The only major difference between image synthesis and super-resolution is that the latter is an image-to-image task and hence takes in an image input rather than an arbitrary Gaussian noise vector. This low-resolution image prior is much more informative than reconstructing images from scratch from an arbitrary vector of noise. Hence if we manage to get good image synthesis results then we should be able to directly apply those techniques to super-resolution with very good outcomes as the task is substantially simpler. We also choose to start on image synthesis as it is by far the most researched application of GANs and has a lot of useful literature to source inspiration from.

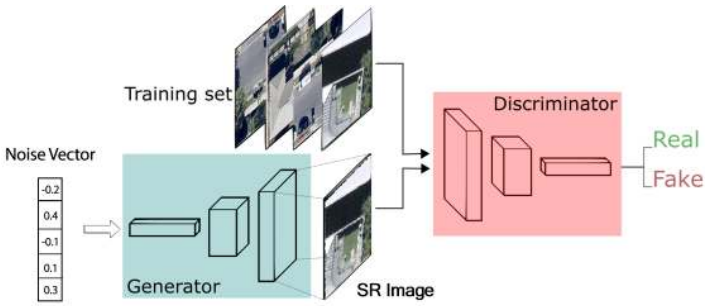
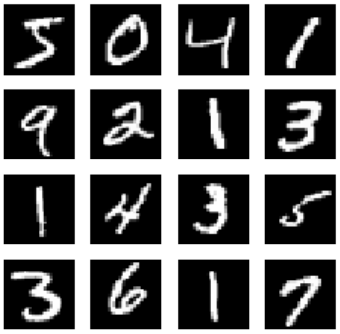


FIGURE 4.5. Diagram of the setup for image synthesis using GANs. We learn a generator G_θ with parameters θ mapping from a Gaussian latent space e.g. $z \in \mathbb{R}^{128}$ (indicated by the noise vector input) to real images $G_\theta(z) \in \mathbb{R}^{256\times256}$.

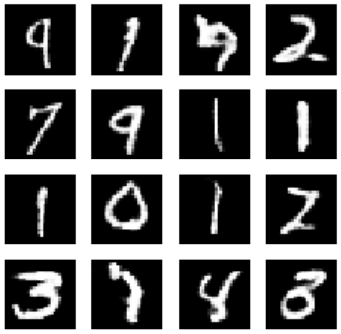
4.4.1 Single-scale GANs

As elaborated on in the literature review, we start with the canonical GAN architecture of DCGAN [28] due to it's relative simplicity and proved effectiveness.

As a quick prototype and in true machine learning fashion we start simple by building a DCGAN model (see 3.2.1) to generate digits from the MNIST dataset [70]. We follow the general approach outlined in the Tensorflow 2.0 image generation tutorials [5] which provides a fairly well-structured implementation of a basic DCGAN. Within about an hour of training we are already able to generate convincing novel handwritten digits from just random input noise, showing the remarkable capability of GANs to generate believable image samples.



(A) Real handwritten digit samples.



(B) Samples generated by DCGAN.

FIGURE 4.6. Comparison of the sample quality of MNIST [70] handwritten digits generated by DCGAN [28] with 1 hour of training to ground-truth images.

[5] https://www.tensorflow.org/beta/tutorials/generative/dcgan

We then adapt this general DCGAN framework for our own purposes so it can function with RGB aerial image data. This presents a much greater challenge than MNIST as we now are required to generate 3 times the channels as well as having to generate images that are substantially larger than 28x28 pixels. We scale up the model to output larger images and increase the number of model parameters to allow it to cope with the vastly greater complexity of modelling aerial ortho-imagery.

4.4.2 Multi-scale GANs

In much the same fashion as in the single-scale DCGAN, we train a multi-scale network based on MSG-GAN 3.5.3 (see 3.5.3. The major change here is that now the generator network learns a series of intermediate outputs for each level of the laplacian pyramid during upsampling. Each of these outputs is then fed directly into the matching resolution tier in the discriminator such that the image being created is both generated and discriminated at every single resolution. This approach provides a much higher gradient flow and allows the network to build meaningful representations of the image at each scale. The approach of scaling up you outputs has shown to perform very well in papers like ProGAN [52] and allows us to generate larger sized images (whereas DCGAN struggles to generate images larger than 64x64 without heavy regularisation).

Likewise, we task this model on our aerial image dataset in order to try and synthesise novel images that mimic our original data at a large size of up to 512x512 pixels. By doing so we can both prove that GANs build sufficiently complex and representative internal representations to model our aerial imagery as well as explore the approach of training a model to learn multiple scales. Both these concepts are sure to help our overall goal of applying GANs to the super-resolution of aerial imagery.

4.4.3 Loss Function

Other than architecture design, the other major thread of research in the GAN world is to do with designing better loss functions. There have been a multitude of different loss functions proposed, all claiming to outperform their predecessor.

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGANGP}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\nabla D(\alpha x + (1-\alpha\hat{x})\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x)-1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x}-1))^2]$
DRAGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{DRAGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\nabla D(\hat{x})\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x-\mathrm{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x}-\mathrm{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x}-\mathrm{AE}(\hat{x})\|\|_1]$

FIGURE 4.7. A table of common Discriminator and Generator loss functions proposed for GANs [71].

However, researchers from Google Brain [71] performed (to the best of their abilities) an unbiased comparative study of the most common loss functions. To keep results consistent across the board they maintained the same model architectures and fixed hyper-parameters and random seeds. They found astoundingly that given enough compute resources, all losses reached very similar levels of performance. They concluded that none of the tested losses consistently outperformed the non-saturating GAN (NS-GAN) from Goodfellow's original paper [1]. As mentioned above, we use this loss function for our own experiments due to it seemingly providing the best performance and stability in training as well as being fairly straightforward to implement.

4.4.4 Evaluation Metrics

GAN image synthesis is a task that is notorious for not lending itself well to conventional metrics. Since there exists no explicit ground truth image to compare to, this makes a lot of image metrics difficult to apply. PSNR is also not well-suited for this task as it tries to measure the sharpness of an image whereas in image synthesis our objective is to create

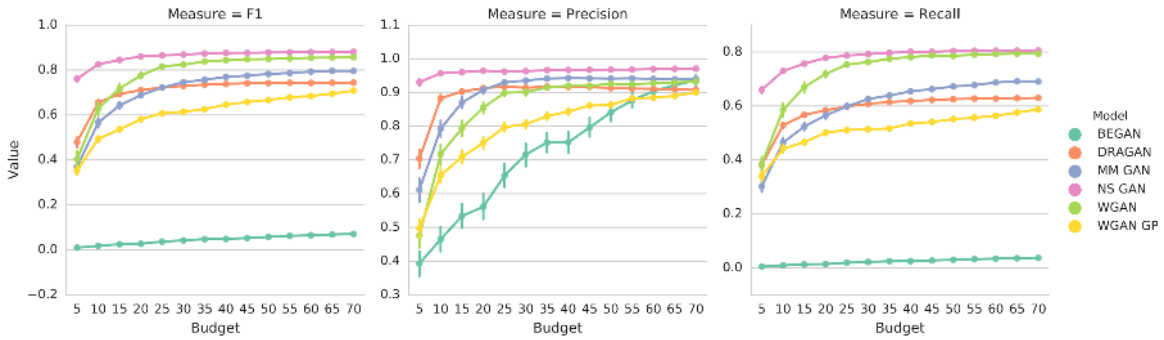


FIGURE 4.8. Plot of the distribution of the maximum F_1 achievable for a fixed budget with a confidence interval of 95%. Mean and confidence interval of the mean is estimated using 5000 bootstrap resamples from 100 runs [71].

"realistic" images. The most popular current metrics for GAN synthesis are the Fréchet Inception Distance (FID) [65] and Inception Score (IS) [63], however they rely exclusively on classification by a pre-trained Inceptionv3 ImageNet classifier and therefore likely won't translate well to our domain of aerial imagery. Therefore evaluations of GAN generated images for this task are largely done based on our subjective perception of image quality, in particular the ability to reconstruct difficult features like sharp roofs, straight roads and road markings for example. Since this section on image synthesis is not the direct goal of our thesis work, we do not dwell too much on metrics and obtaining the best quality results. Rather we use it as a means for building intuition towards GANs and applying this for the task of super-resolution in the next section.

4.4.5 Training

We trained our smaller 56x56 models with a batch size of 64, with bigger models training on larger 256x256 images only allowing for batch sizes of around 8. We found that GANs were far, far slower to train than pure CNN models. This was expected as their objective is not nearly as well defined and changes over time, as well as most networks requiring far more parameters to get good results. The fact that both networks must learn from one another elicits a very slow initial stage of training where neither network passes much useful information to the other. Outputs often looked completely unrecognisable during the first few hours of training. It is only over the span of a day to close to a week of training that one begins to see

imagery generated that resembles the input dataset. Models that had too few parameters also began to plateau in image quality and never managed to reproduce finer details, and hence were terminated and re-engineered.

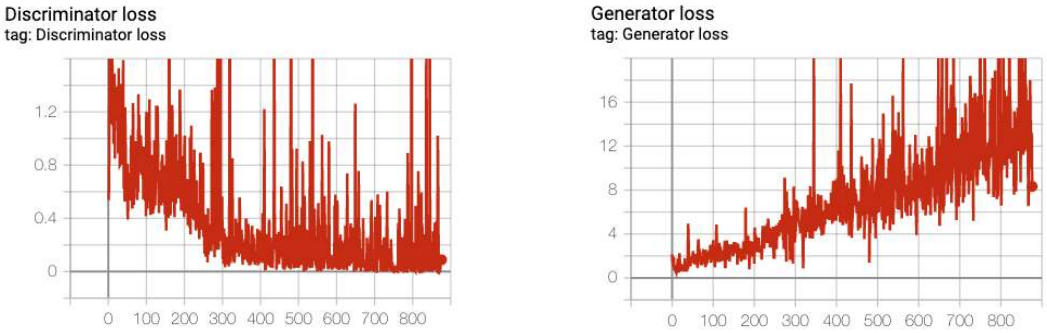
4.4.5.1 Mode Collapse

Training simply for longer with much more complex models allows for more refined features and sharper edges to be learned, however the fundamental issue of data diversity becomes hugely problematic. This leads to one of the major issues in GANs which is mode collapse. This is the case in which the output modes of the generator collapse, turning a multi-modal dataset into a fairly uni-modal one. This often means that the model will happily produce the same outputs regardless of different inputs fed to it. We can commonly identify this early by plotting random batches of outputs as we train to see if many of the images fall into a few distinct modes. This often occurs when the dataset is small and images are not very diverse, as well as having a poorly regularised discriminator that is not conditioned to strive for more varied outputs. We fix this issue by increasing the size of our dataset from 1000 to about 10,000 images. Another common technique to deal with this is Minibatch Discrimination [63] which involves additionally penalising similarity between generated images when fed to the discriminator, hence encouraging diverse outputs. However we do not end up using this technique in our own final model due to time and added complexity.

FIGURE 4.9. An example of mode collapse in GANs. For random, varied inputs the generator collapses to a unimodal representation and hence only outputs a very limited set of discrete output images.

4.4.5.2 Stabilising GAN models

A very common situation we encountered as we scaled up the complexity of my model was divergence of the 2 sides of the model. Again and again the discriminator would overpower the generator and fairly quickly drop to zero loss as it learned to classify the examples perfectly. By learning much faster than the generator, it became much more advanced and provided very few useful gradients with which the generator could learn to improve from. This often occurs when the generator outputs poor samples, which the discriminator can thus very easily classify. As the generator falls behind it begins creating more and more crazy samples in desperation to hopefully fool the discriminator, but just ends up becoming very unstable and outputting very unrealistic images. Hence the art of GAN training is to effectively balance the strength of both models so they learn at a similar pace and constantly challenge each other to meaningfully improve.



(A) Discriminator training curve. (B) Generator training curve.

FIGURE 4.10. An typical example of a GAN instability failure mode. Both halves of the model diverge, resulting in no useful gradients from the discriminator and poor samples generated by the generator.

To improve the considerable training instability and divergence issues we encountered, we implemented the many techniques and tricks as outlined in section 3.6. Tweaks such as discriminator noise and dropout as well as one-sided label smoothing helped significantly to balance the two networks and prevent gross overfitting of the discriminator. It also sped up training by providing more useful gradient flow between the two halves of the network. With this, we managed to fairly reliably train our GAN models without serious failure modes. However the task of balancing the relative network strengths became universally trickier the

larger we scaled the model, requiring more regularisation, more data and more ingenuity with the design of our network architecture.

4.5 GAN Super-resolution

One can think of this section as a culmination of the work and learnings of the previous two sections. We look to take the CNN image super-resolution model from 4.3 and augment it with the improved loss functions and adversarial training from 4.4. This aims to create a high-performance deep learning framework for image enhancement that better models the true qualities of visual perceptual in image quality.

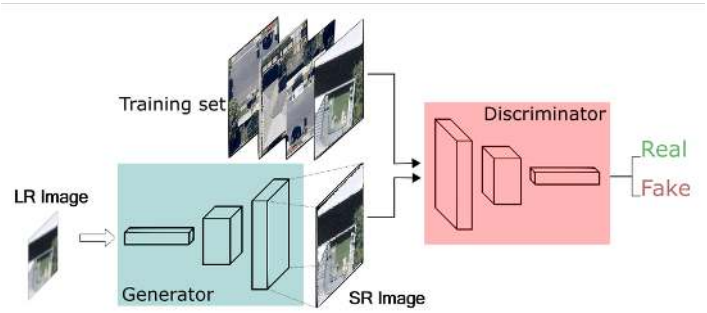


FIGURE 4.11. Diagram of the setup for image super-resolution using GANs. Note the similarity to the image synthesis task 4.5, except now instead of an arbitrary Gaussian latent vector as input we use our low resolution image for a much more informative prior.

4.5.1 Generator

We choose our generator model based on the results of pre-training in 4.3. Taking our best performing variants of each of SRResNet, EDSR and RRDN in terms of our aforementioned image quality metrics, we slot these models into our full GAN training model.

It is important to recognise the importance of pre-training, as with GAN image synthesis it was abundantly clear that both halves of the GAN perform extremely poorly when trained from scratch. By having our generator models already thoroughly pre-trained on the L1 pixel minimisation objective, we start GAN training with a generator model that already has most

of the knowledge needed to super-resolve an image. To reiterate, we know that from this conventional pixel-wise pretraining it may produce significantly sharper images, however still lacks a great amount of texture and detail information in its reconstructions. This is exactly the role which we employ our GAN to fill.

Since GANs are known to hallucinate details in order to try and produce more realistic images, there exists a concern that these details may not reflect reality and mislead the viewer. Hence, we also explore the idea proposed by ESRGAN to interpolate the network weights between a PSNR (pre-trained) objective and a final GAN objective. This allows us to have control over the level of detail synthesis.

4.5.2 Feature Loss

The careful choice of loss function(s) is crucial to the success of any deep learning model. In the task of image generation and thus super-resolution it is essential to accurately model a loss representative of image quality and resolution. Using pure pixel loss is a quick and simple way to compare images by finding the mean absolute error (L1) or mean squared error (L2) between corresponding pixels in either image. The unfortunate downside to only using pixel loss is a reconstructed image that is over-smoothed (especially for L2) and lacking much fine semantic detail. Empirically L1 loss gives sharper results than L2 but still leaves much to be desired.

Hence we look to augment our image loss with an image feature loss [46] term as explained previously in section 4.5.2. This is sometimes referred to as "perceptual loss" in the literature [46] [72], however we stick with the terminology of feature loss to avoid confusion with human perceptual evaluation. Unlike the typical VGG network used by [46] and many other implementations, we deem this as not being overly useful as it is trained on ImageNet and hence has little overlap with the aerial ortho-imagery we seek to model. Instead we use the Nearmap's own semantic segmentation model to be our auxiliary classifier network and compare the results of this both with using VGG feature loss. We therefore will pass both the generated and high-resolution ground truth images through this model and compare the

intermediate activations of the network to achieve a semantic comparison of the images. The intention of this is that if we are trying to super-resolve a tile roof, even though the individual pixel L1 loss might not give us much information, the content loss should pick up that it is a certain roof type and guide the generator towards producing a tiled pattern that matches semantically. Content loss can be expressed as the image normalised L2 loss between layer activations at layer l between source and target:

$$\ell_{\text{feat}}^{\phi,j}(\hat{x}, x) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{x}) - \phi_j(x)\|_2^2 \tag{4.6}$$

We also incorporate another loss term called the style loss. The style of an image can be described by the means and correlations over the different feature maps [73]. We compute a Grammian matrix to represent this quantity. It is formed by taking the outer product of the feature vector with itself at each location, and averaging it over all locations. For a particular layer:

$$G_{cd}^{l} = \frac{\sum_{ij} F_{ijc}^{l}(x) F_{ijd}^{l}(x)}{IJ} \tag{4.7}$$

This correlation operation essentially encodes the activations of the convolutional feature maps that co-occur. This throws out spatial information and instead is a way in which one can quantify the texture or style of an image (as patterns and textures are considered spatially invariant to their location in the image). By minimising the MSE of Gram matrices between input and generated images, Gatys et al. find that this allows us to meaningfully match the style of both images [73].

4.5.3 Discriminator

The final key component in our GAN framework is the discriminator network which detects if generated images are real or fake, thus acting as a fully learnable image loss function. The architecture of the discriminator is typically nothing more than a convolutional neural network with a binary classification output. However, we make some modifications from the original architecture outlined in SRGAN in order to achieve better results.

Firstly, we modify the discriminator to be more like the patch-based discriminator network of Pix2Pix [62]. This means that instead of outputting a scalar binary classification value [0,1] for the "realness" of the entire image, the network instead outputs a 2D matrix of such values with each entry corresponding to the "realness" classification for patches of the original image. We found that this helped the network be fully convolutional which meant it worked regardless of input image sizes, hence allowing it to work for large images without needing excessive parameters. For our discriminator we chose to output a 4x4 patch of discriminator values, aligning with the Pix2Pix paper. More detail on this can be found in bullet 9 of section 3.6.

Similarly to SRGAN we also used LeakyReLU activations as they are known to work better for GANs than ReLU function [55]. This is because having the leaky negative slope component helps with gradient flow and is not fully rectified. We apply Batch Normalisation with momentum of 0.8 in the same manner as SRGAN, to all convolutional layers but the input layer.

The big improvement we make is towards the regularisation of the discriminator network. By applying spectral normalisation to the weights of the discriminator, we effectively bound the gradients of the discriminator. This is crucial to the stability of the GAN as a whole since these same gradients are passed from the discriminator back into the generator, guiding its own learning.

We also employ a fairly unique discriminator approach in the domain of super-resolution inspired by work on conditional GANs [9] and in the relativistic GAN approach of ESRGAN [23]. We hypothesise that rather than having the discriminator determine the unconditional "realness" of any given image, it would be much more effective to have it predict instead the "realness" of the super-resolution operation itself. That is to say we aim to condition the discriminator on the low-resolution when it classifies the super-resolved image, i.e. changing it from $D(x_{sr})$ to $D(x_{sr}|x_{lr})$. We see this as having a two-fold advantage. First and foremost, by having the low resolution image as an input to the discriminator it is given a lot of information about the image before it is super-resolved and hence only really has to learn the slight texture and detail differences between the images rather than about the entire image

itself. This is almost like learning the residual in a residual network, lightening the load on what the network really needs to learn from the absolute realness to the relative realness with respect to the low resolution image. Secondly it circumvents the classic GAN fallacy of ignoring the input or latent space. By predicting whether the super-resolved image is a realistic up-sampling transform of the low-resolution image, this ensures that the transformation the generator applies to up-sample is meaningful given the input image, rather than simply learning to produce realistic images regardless of the input in order to fool the discriminator. In practice we implement this by concatenating a nearest-neighbour up-sampled copy of the low resolution image to the existing input of the high-resolution generated image to the discriminator.

For the loss function we choose the non-saturating GAN objective (as discussed in 4.4.3).

$$\mathcal{L}_{\mathcal{D}}{}^{NSGAN} = -\mathbb{E}_{x\sim p_d}[log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[log(1 - D(\hat{x}))] \tag{4.8}$$

$$\mathcal{L}_{\mathcal{G}}{}^{NSGAN} = -\mathbb{E}_{\hat{x}\sim p_g}[log(D(\hat{x}))] \tag{4.9}$$

Then as mentioned before we also augment this with the content loss function (see 4.6) to encourage the model to learn domain-specific semantic details from our segmentation network about features like roofs, cars, roads, trees and the like. Finally, we also decide to include the pixel-wise loss term on top of this as a means of regularising our model training to a safe PSNR-oriented guideline. This L1 pixel loss can still be handy in enforcing that edges are sharp and true to the low-resolution image so that the GAN doesn't stray too far from the original input. Hence, we combine all these loss terms together into a single weighted sum to form our final generator loss objective:

$$\mathcal{L}_{\mathcal{G}} = \alpha\mathcal{L}_{\mathcal{G}}{}^{NSGAN} + \beta\mathcal{L}_{feat} + \gamma\mathcal{L}_{L1} \tag{4.10}$$

For most of our tests we use $\alpha = 1 \times 10^{-3}, \beta = 5, \gamma = 1 \times 10^{-3}$. The rough rule of thumb we found was to keep the GAN and feature loss terms on roughly the same order of magnitude, which keeping the pixel-wise loss maybe an order of magnitude smaller to discourage the model to converge on blurry results after long periods of training. However we also discuss

the trade-off in changing the relative weighting of these parameters later on, depending on whether you want to steer your output images towards something more GAN hallucinated, something that maximises the semantic elements or stick with a more PSNR-centric output.

4.5.4 Improving Downstream Semantic Segmentation

The secondary use-case for our super-resolution model is for improving the accuracy of semantic segmentation predictions by using images that have been super-resolved. This is a worthwhile application of our image enhancement framework as it caters to the direct needs of Nearmap in their business of semantic segmentation of aerial imagery. Furthermore, our segmentation model is only ever trained for one particular zoom level, and so cannot cope well with any images from low resolutions. By optimising the images through our super-resolution pipeline we look to not only improve the resolution and sharpness, but also enhance these images using the aforementioned semantic feature loss to optimise them directly for the segmentation model.

The method for performing this evaluation is to run our segmentation prediction model across the same test set of images, one copy which has been bilinear up-sampled, one which has been super-resolved by our basic CNN, another with the GAN and one which is the original unmodified set of images. By doing so we can evaluate the accuracy of the segmentation masks generated for each up-sampling technique by comparing them to the segmentation on the original images. By taking the difference in Intersection over Union (IoU) of the semantic masks we can quantitatively measure the improvement in segmentation accuracy. Furthermore we can break these results down by prediction class to see whether super-resolution provides greater segmentation benefits for different types of features in our aerial imagery.

4.5.5 Evaluation Metrics

For the task of super-resolution we retain the original image quality metrics outlined in 4.3.4: PSNR, SSIM and MS-SSIM. Here we dig deeper into their limitations and their capability for modelling human perceptual quality of images. This leads us to explore other potential image

metrics, which we discuss but do not actively use because they are not standard practice in many super-resolution papers.

For the segmentation accuracy task we use Intersection over Union (Jaccard Index) to measure the similarity between two semantic segmentation masks. As the name suggests, it is defined as:

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.11}$$

Since the segmentation model outputs a probability value for every pixel for every class (channel dimension), we pick a threshold of $\sigma = 0.5$ to turn these probabilities into binary yes/no predictions.

4.5.6 Training

For training our super-resolution GAN model we stick with the ADAM optimiser once again, however we set the momentum term $\beta_1 = 0.9$ as suggested by ESRGAN [23]. Furthermore, we also back off the learning rate from pre-training to a less aggressive $lr_g = 1 \times 10^{-4}$ and $lr_d = 2 \times 10^{-4}$ using the two-time update rule as also mentioned in section 3.6. These learning rates are then halved in a piece-wise manner after 20000 steps of the optimiser to try and improve convergence later in training. For most models we train the super-resolution GAN between 10000 and 40000 iterations, often needing the greater number of steps for higher up-scaling factors. As a result the GAN part of training usually takes on the order of a few hours. Often training for longer than this does not achieve noticeably better results and in a lot of cases the GAN can often regress in quality as it tries to over-optimise its outputs. Like the pre-training, we also stick to a batch size of 16.

CHAPTER 5

Results

We first show the tangible results of CNN-based super-resolution to illustrate the ability of deep learning models to supersede traditional up-sampling approaches. We evaluate both the subjective image quality but also compare well-known image metrics like PSNR and SSIM (evaluated on the validation set) as a proxy for image quality. We then dive into the performance of the GAN-based approach in a similar manner to highlight its key differences. For our tests we aim to compare the results of different generator and discriminator architectures, choices of loss functions, regularisation techniques as well as how models perform at different up-sampling scales.

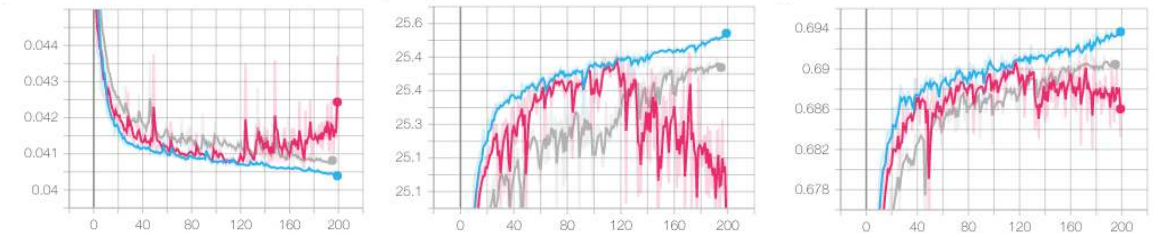
Lastly, we show results from evaluation of the candidate super-resolution models on improving downstream semantic segmentation. Here we list both examples of improved segmentation mask generation as well as an extensive quantitative comparison of IoU scores for each model across our different prediction classes. We also take this opportunity to display notable failure modes and edge cases that serve as a limitation to our model.

5.1 CNN Super-resolution

In order to evaluate different convolutional models we run experiments to asses parameter and architecture choices on our given metrics. Metrics are reported on the validation set and are taken to be the exponential moving average of the metric after the full 200 epochs (to smooth out local variations). We also report the number of parameters and time taken to train each model to get an idea of how practical these models might be.

5.1.1 SRResNet

To have a reliable baseline for super-resolution task we choose to evaluate the effectiveness of a simple SRResNet model on 4x super-resolution. As outlined in the methodology we keep all elements of training consistent in order to compare the effectiveness of candidate models. For SRResNet we test the original formulation as it is presented in [22] as well as a variant without Batch Normalisation. For the Batch Normalisation model we had to reduce the learning rate because it became unstable at higher learning rates.



(A) MAE Loss (B) PSNR (C) SSIM

FIGURE 5.1. Training curves of the original 4x SRResNet model (pink) to a variant with Batch Norm removed (blue). It is clear that removing Batch Norm helps significantly with stabilising the model and improving performance overall.

Model	Hyperparameters	# Parameters	MAE Loss	PSNR (dB)	SSIM	Time (hr,m)
SRResNet	BN (Original)	1.5M	0.04243	24.95	0.6860	19hr 33m
SRResNet	BN, $lr_i = 1 \times 10^{-4}$	1.5M	0.04074	25.42	0.6905	18hr 42m
SRResNet	no BN	1.5M	**0.04037**	**25.52**	**0.6937**	16h33m

TABLE 5.1. Comparison of performance between the 4x SRResNet model with and without Batch Norm. BN signifies Batch Normalisation and lr_i signifies the initial learning rate which unless specified is assumed to be as described in the methodology section.

We also tested the effect of ICNR initialisation in the Pixel Shuffle up-sampling layers. Passing a random noise input into a 4x up-sampling block that was completely untrained produced immediate checkerboarding due to the improper initialisation of the sub-pixels. When ICNR was added however we observed no such checkerboard artifact in the output image. This goes to show that if one wishes to avoid checkerboarding early in training then ICNR initialisation is a must for PixelShuffle layers.

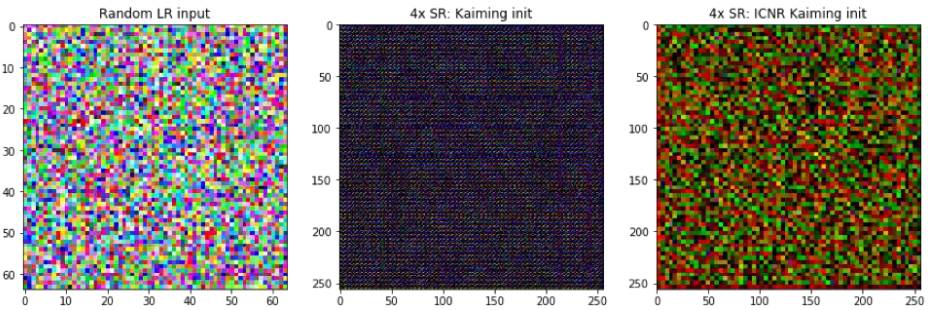


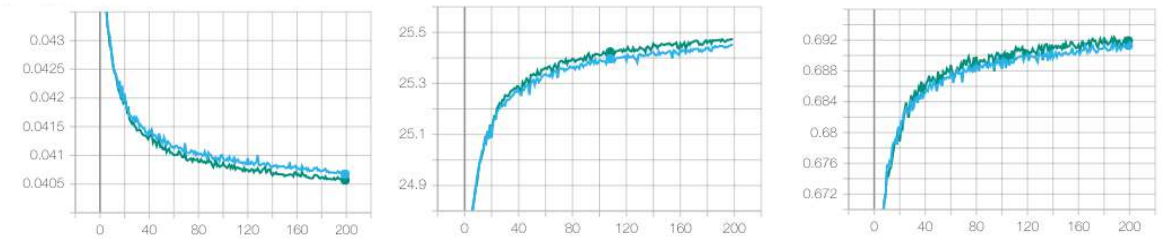
FIGURE 5.2. The effect of proper ICNR initialisation for sub-pixel convolution is very apparent during PixelShuffle up-sampling. In this example we feed in white noise and without proper initialisation a checkerboard pattern is immediately present even without training (middle), but with ICNR (right) our outputs actually still reflect the input noise pattern (sans gridding).

Model	Hyperparameters	# Parameters	MAE Loss	PSNR (dB)	SSIM	Time (hr,m)
EDSR	(Original)	1.5M	0.04067	25.45	0.6913	8hr 7m
EDSR	Upsampling activations	1.5M	**0.04057**	**25.47**	**0.6919**	8hr 27m

TABLE 5.2. Comparison of performance between original and modified 4x EDSR models. Up-sampling activations means we included activation functions in the up-sampling blocks (whereas the original had removed them).

5.1.2 EDSR

Much the same as for SRResNet, we tested both the original model from [38] as well as comparing it against a few variations of this same model. Below are once again plotted the training curves as well as a table of each model's evaluation metrics after training. We also found with this model that it did not cope well with very high initial learning rates of 4×10^{-4}, hence we relaxed these to 1×10^{-4} to make sure the models converged.



(A) MAE Loss (B) PSNR (C) SSIM

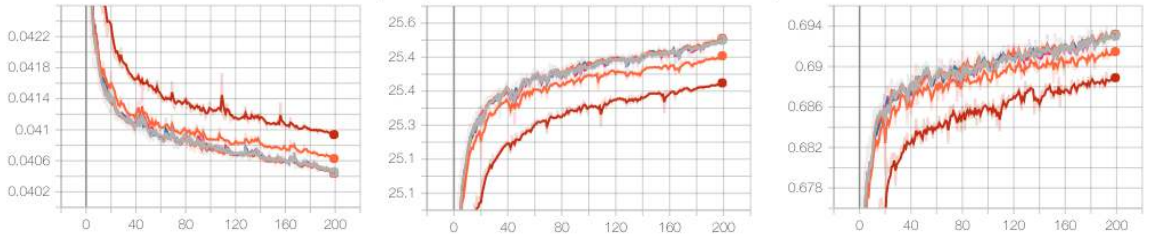
FIGURE 5.3. Training curves of the 4x EDSR model variants. The original model is shown in blue and the variant with upsampling activations in green.

Model	Hyperparameters	# Parameters	MAE Loss	PSNR (dB)	SSIM	Time (hr,m)
RRDN	RE (Original)	1.8M	0.04046	25.50	0.6930	11hr 41m
RRDN	PS	2.0M	0.04045	25.50	0.6931	11hr 4m
RRDN	PS, G=32	1.1M	0.04063	25.45	0.6915	9hr 42m
RRDN	PS, T=3	5.3M	**0.04039**	**25.52**	**0.6936**	19hr 27m
RRDN	PS, blur	2.0M	0.04044	25.50	0.6932	11hr 37m
RRDN	PS, SN	2.0M	0.04094	25.37	0.6889	11hr 41m
RRDN	PS, SU	2.3M	0.04044	25.51	0.6932	10hr 36m

TABLE 5.3. Comparison of 4x RRDN model variants. "RE" indicates the use of a resize convolution and "PS" indicates pixel shuffle for the up-sampling component of the network. "G" is the number of filters for each convolutional layer and "T" the number of RRDB blocks. "SN" indicates the use of spectral normalisation in convolutional layers, "blur" the use of a gaussian blur on the output of up-sampling layers and "SU" the use of a single 4x pixel shuffle up-sampling layer rather than stacking multiple 2x layers.

5.1.3 RRDN

Once again we repeated this process for the RRDN architecture. Since this architecture was inherently a lot more complex, we had a lot more parameters and tweaks to experiment with here. Below are the training curves of all the candidate models accompanied by a table of final quantitative results.



(A) MAE Loss (B) PSNR (C) SSIM

FIGURE 5.4. Training curves of the 4x RRDN model variants. Most models perform roughly as well as one another with the exception of the 32 filter mode (orange) and the spectrally normalised model (red).

Changing the loss function from MAE to MSE we also saw a very slight increase in blurriness of the final image and an almost unnoticeable change in PSNR and SSIM. This agreed with our expectations from the literature as MAE is nowadays the more popular choice due to

having marginally better sharpness of edges and better training convergence [74] [38] [23]. Hence we continued to use MAE loss for all experiments.

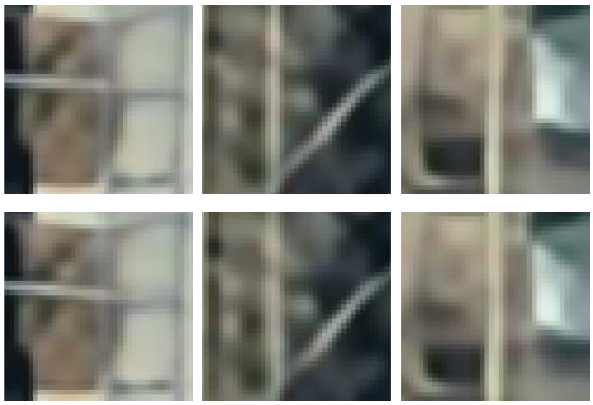


FIGURE 5.5. A close-up comparison of using a Mean Averaged Error (MAE) [top row] versus Mean Squared Error (MSE) [bottom row] as the loss function for the RRDN model. MAE provides ever so slightly sharper edges and boundaries.

5.1.4 Comparison between models

We then took the best models from each architecture and compared them head to head. We realised using individual learning rates for the different models as well as different batch sizes (due to different model sizes) made it the case that our model results could not be directly compared against one another. Hence, to compare our three different models against each other reliably, we chose the best model variant from each architecture and trained them all using the same training parameters e.g. learning rate and batch size. For this we used an initial learning rate of $lr_g = 2 \times 10^{-4}$ and decayed this by a factor half every 200 epochs. To accommodate for all model sizes we also used a batch size of 12 (since a batch size of 16 caused the larger RRDN models to run out of memory). We then trained all models for a total period of 500 epochs and compared their relative performances against the baseline, unmodified SRResNet model. For the purpose of these experiments we also included MS-SSIM to investigate the reconstruction of scale-dependent features.

Model	# Parameters	MAE Loss	PSNR (dB)	SSIM	MS-SSIM	Time (hr,m)
SRResNet (base)	1.5M	0.04066	25.44	0.6919	0.9296	39hr 56m
SRResNet	1.5M	0.04061	25.44	0.6926	0.9300	32hr 40m
EDSR	1.5M	0.04059	25.45	0.6930	0.9301	18hr 30m
RRDN	3.0M	**0.04035**	**25.51**	**0.6957**	**0.9308**	37hr 10m

TABLE 5.4. A comparison between the best instances of each of our 3 model architectures.

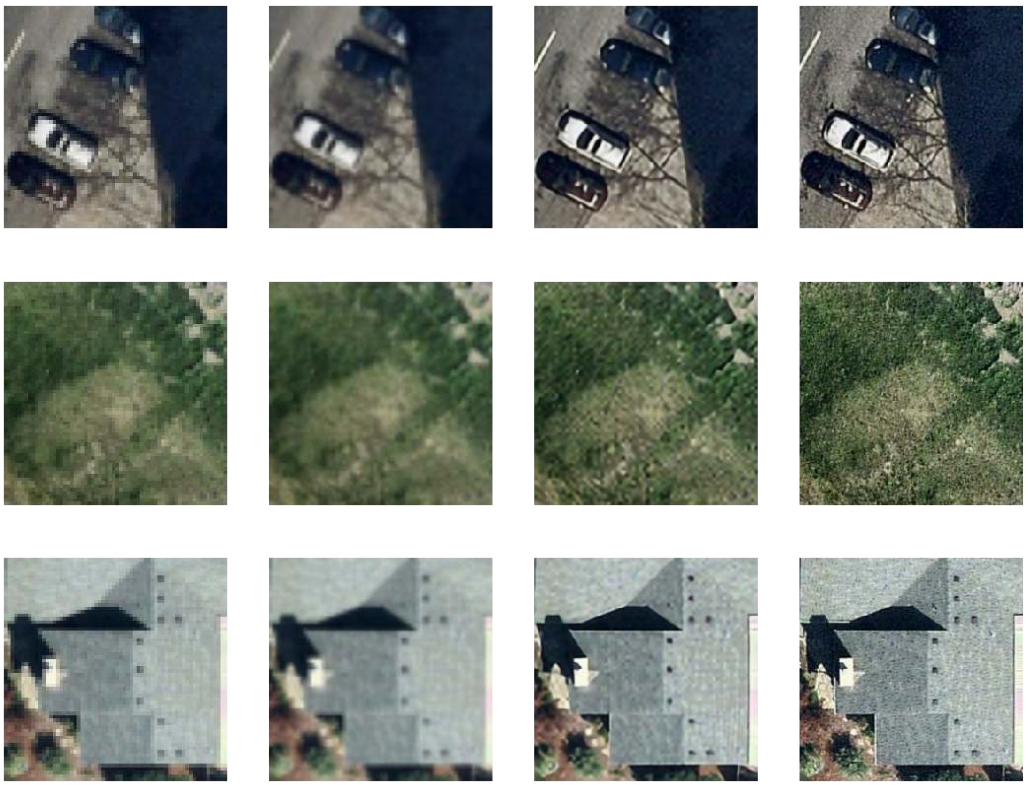


FIGURE 5.6. A set of sample outputs from the RRDN model. Although the super-resolution model manages to greatly enhance the general shape of objects, in particular excelling at sharpening lines, it struggles to capture fine detail and texture such as grass and roofing (rows 2 and 3). From left to right the columns portray a nearest-neighbour and bilinear up-sample of the low-resolution zoom 19 image, then the super-resolved equivalent using our RRDN model and finally the original high-resolution zoom 21 image on the right as the ground truth.

5.1.5 Performance at different up-sampling factors

From the above results it was therefore clear that the RRDN model was the best performer from the three in terms of super-resolution ability and training efficiency. Hence in the interest of time we decided to stick with this model only for future experiments. Here we investigate the full scope of super-resolution scaling factors, training the model not only on 4x but now also 2x, 8x, 16x, and 32x to see how it copes at each of these scales.

5.1.5.1 Transfer Learning across resolutions

We found that by transfer learning from the 4x model to any other resolution, we short-circuited the vast majority of the training process. By reusing the weights learned from a different resolution model, the new resolution model was able to perform well out of the box and required a much shorter amount of training to simply fine-tune to suit the new resolution. This worked because our model architecture remained almost identical between scaling factors, the only change being an extra up-sampling layer for each factor of 2x up-sampling our task required. This also hinted at the fact that the majority of features and relations learned by models at different scales were closely related.

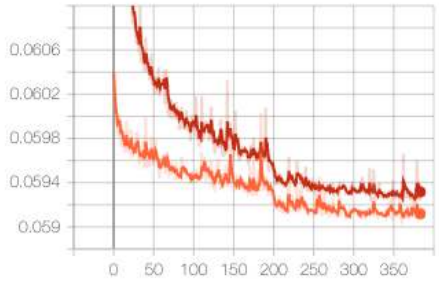


FIGURE 5.7. Validation loss curves for an 8x RRDN model. The red curve is an 8x model trained from scratch while the orange curve is an 8x model transfer learned from existing 4x model weights, clearly speeding up training and achieving better final performance.

By inspecting the results of our RRDN super-resolution model across multiple different up-sampling factors, it becomes quite evident that the deep learning approach to super-resolution greatly outperforms traditional means of up-sampling such as bilinear interpolation. Even more exciting is that for the higher scaling factors such as 32x, the advantage of the CNN

(A) 2x

(B) 4x

(C) 8x

(D) 16x

(E) 32x

FIGURE 5.8. Performance compared across different upsampling factors from 2x up to 32x in powers of 2. From left to right we show the nearest-neighbour and bilinear upsampled low-resolution images, followed by the equivalent super-resolution using RRDN and finally the high-resolution ground truth image.

Model	Scale	# Parameters	MAE Loss	PSNR (dB)	SSIM	MS-SSIM	Time (hr,m)
RRDN	2	3.0M	0.01395	34.98	0.959	0.9976	43hr 34m
RRDN	4	3.0M	0.04035	25.51	0.6957	0.9308	37hr 10m
RRDN	8	3.1M	0.05901	21.86	0.4651	0.7765	25hr 47m
RRDN	16	3.1M	0.07369	19.71	0.3607	0.6163	23hr 4m
RRDN	32	3.1M	0.09095	17.93	0.3111	0.4676	23hr 15m

TABLE 5.5. Performance at different super-resolution scales.

model becomes even more apparent as the network has more liberty to infer its own prediction for the scene as the low-resolution prior is extremely poor. It is quite incredible to see that in the 32x case the network manages to correctly infer the general shape and existence of 2 cars parked on the street, a footpath and a house roof with an associated shadow being cast. Judging by the low-resolution image and even the bilinear up-sampled image, this would be very difficult information to ascertain. This therefore gives credence to our aim of being able to up-sample zoom 16 imagery to zoom 21 (a 32x up-sampling operation) and hence being able to infer and potentially segment semantically significant elements of these low-resolution images.

5.2 GAN Aerial Image Synthesis

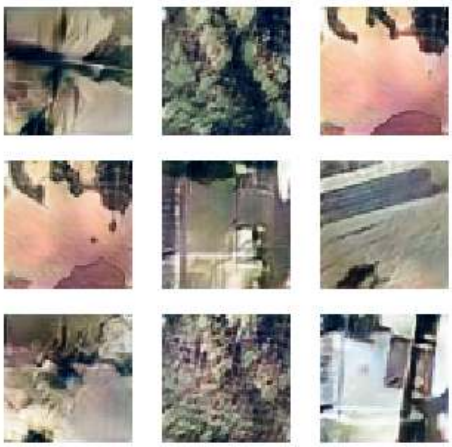
5.2.1 Single-scale Model

5.2.1.1 Original DCGAN

Using an implementation of DCGAN following the original paper [28], we tasked our network on learning small 56x56 patches of aerial imagery. Without any modifications the network seemed to create somewhat plausible patches of imagery. It was evident, however, that these shapes and textures were oversimplified and our generator was clearly underfitting our dataset (see 5.11 for an example). This concern was also reinforced by the fact that the discriminator had noticeably overfit and hence was not passing very informative gradients back to the discriminator. We found that this vicious cycle occurred when the generator produced poor

quality imagery, making it easy for the discriminator to do its job and not passing back many useful gradients, further causing the generator to not learn as it should.

(A) Real images sampled from Apollo.



(B) Images generated by DCGAN.

FIGURE 5.9. Comparison of generated versus ground truth images from our dataset. It is evident that DCGAN learns to replicate colours and low frequency features in vague blobby outlines as well as some simpler textures like tree foliage however struggles to create convincing building shapes and details.

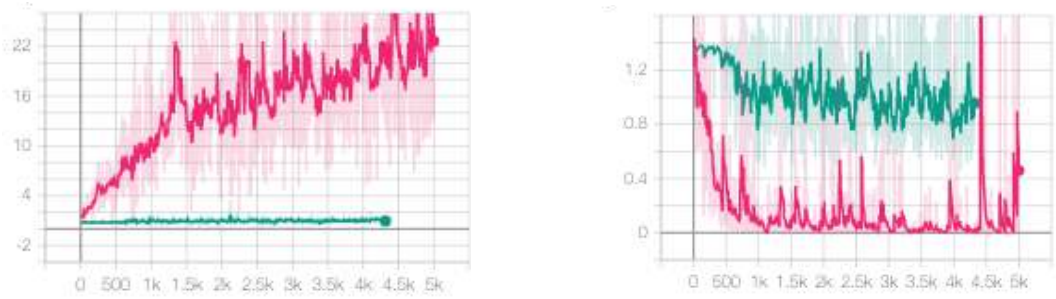
As well as the random samples plotted above, we also decided to perform a linear interpolation through the latent space z of the model and to plot these results. It is evident here that despite traversing a continuous Gaussian latent space, the output space of generated images seemed to hop in a more or less discrete manner between modes of the imagery.

FIGURE 5.10. Generated outputs from linearly interpolating the latent space vector z. It is clear that it jumps sharply from mode to mode in the data and is has not learned a continuous latent space.

5.2.1.2 Stabilising DCGAN training

To amend the unbalanced training situation of the above DCGAN, we implemented a number of the GAN stabilisation techniques outlined in 3.6. This change greatly improved the stability of the GAN model and aided its ability to train and learn to generate more meaningful imagery.

Below we have plotted the training curves of an untampered DCGAN model against a model with our stabilisation tweaks applied to illustrate the improvement in training.



(A) Generator loss. (B) Discriminator loss.

FIGURE 5.11. A comparison of the loss curves of the original un-modified DCGAN model (pink) and the DCGAN with our training stabilisation techniques added (green).

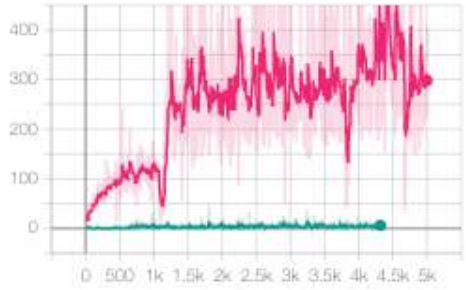
In order to debug when a model was showing poor stability characteristics we also added code to compute the global norm of all gradients flowing through the network. By inspecting the magnitude of the gradients in the network it was straightforward to pick up on a model that was unstable and diverging. A good rule of thumb was that the gradient norm in stable models always tended to stay below 100. The global norm here is defined as:

$$\text{global\_norm} = \sqrt{\sum_{l=0}^{L} ||g_l||_2^2} \tag{5.1}$$

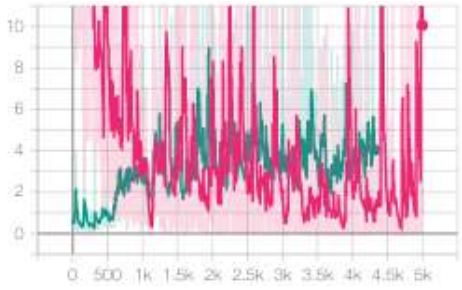
where g_l is the tensor of gradients for each parameter in layer l of the network, L is the total number of trainable layers in the network and $||.||_2$ indicates the Euclidean norm.

We used Tensorflow's `tf.linalg.global_norm` to compute this quantity. In our experimentation we found this to be an extremely useful metric to log in order to track model stabilisation issues and believe that it should see more adoption for GAN training generally in the field.

Below are shown the samples plotted from the more stable DCGAN model we constructed 7.1. In the same manner as before, we also plot a series of outputs illustrating a linear walk through the latent space of the GAN 5.14.



(A) Global norm of Generator gradients.



(B) Global norm of Discriminator gradients.

FIGURE 5.12. A comparison of the gradient magnitudes of the original unmodified DCGAN model (pink) and the DCGAN with our training stabilisation techniques added (green).

(A) Real image patches from Apollo.

(B) Generated images from our GAN.

FIGURE 5.13. A side-by-side comparison of 56x56 images synthesised from our improved DCGAN model to ground-truth images in the dataset.

FIGURE 5.14. Results of our model from performing a latent space interpolation of the stabilised DCGAN. We now notice a clear smooth transition between output modes.

5.2.2 Multi-scale Model

5.2.2.1 MSG-GAN

Training a multi-scale model based on MSG-GAN proved to yield a much more stable training procedure as gradients flowed across multiple layers of both the generator and discriminator, providing far stronger feedback for learning. Whereas the DCGAN struggled significantly to generate images any greater than 56x56 and had fairly coarse, unremarkable outputs, the multi-scale model produced far higher fidelity images because the model itself was more sophisticated and its stability allowed for larger images to be generated. From these images it is quite easy to recognise features like roofs, roads, trees and sometimes cars that the GAN has learned to synthesise. In a side-by-side comparison one can be sometimes be tricked into thinking some of the generated images are real, but usually closer scrutiny reveals certain wonky edges or nonsensical features.

(A) Real image patches from Apollo.

(B) Generated images from MSG-GAN.

FIGURE 5.15. A side-by-side comparison of 256x256 images synthesised from our version of MSG-GAN to ground-truth images in the dataset.

The real power of a multi-scale model comes in learning a hierarchy of scales. As we train the network we can see how it learns to build a consistent representation of the image for

each resolution level, and each of these lower resolution is gradually refined up until the full resolution final output. As the training starts, the feature maps at different scales of the network are wildly different, however as it trains one can see them all align in terms of colour and content.

FIGURE 5.16. Plot of the output images of the generator at every resolution tier of the network. The image sizes from left to right are in this case 4x4, 8x8, 16x16, 32x32, 64x64, 128x128 up to 512x512.

To demonstrate the power of this model to learn general and semantically relevant representations of the dataset we plot the output for the same latent vector at two different times during training. It is clear that even though the high level composition of the image is much the same, the model can tweak relevant semantic elements such as the shape and number of panels of the roof. This transformation is therefore evidently not just a simple operation on the global information of the image, but rather a complex alteration of the content of the image itself.

FIGURE 5.17. Output from our GAN model for the same latent vector at two different stages in the training process. The general structure of the image stays consistent but small details and features are tweaked by the GAN as it learns to generate more realistic imagery.

5.3 GAN Super-resolution

This portion of our results focuses primarily on the extension of our previously proposed CNN super-resolution framework (see section 5.1) by means of incorporating Generative Adversarial Networks. In this section we show both the remarkable ability of GANs to effectively synthesise detail in super-resolved images but also their strength in optimising imagery for semantic segmentation applications.

5.3.1 Image Resolution Enhancement

As a result of training our super-resolution model in a GAN setting, we were able to augment the L1 loss function of the generator with the GAN loss and the feature loss from the Nearmap segmentation model. Performing the super-resolution task once again at 4x after training yielded the following results.

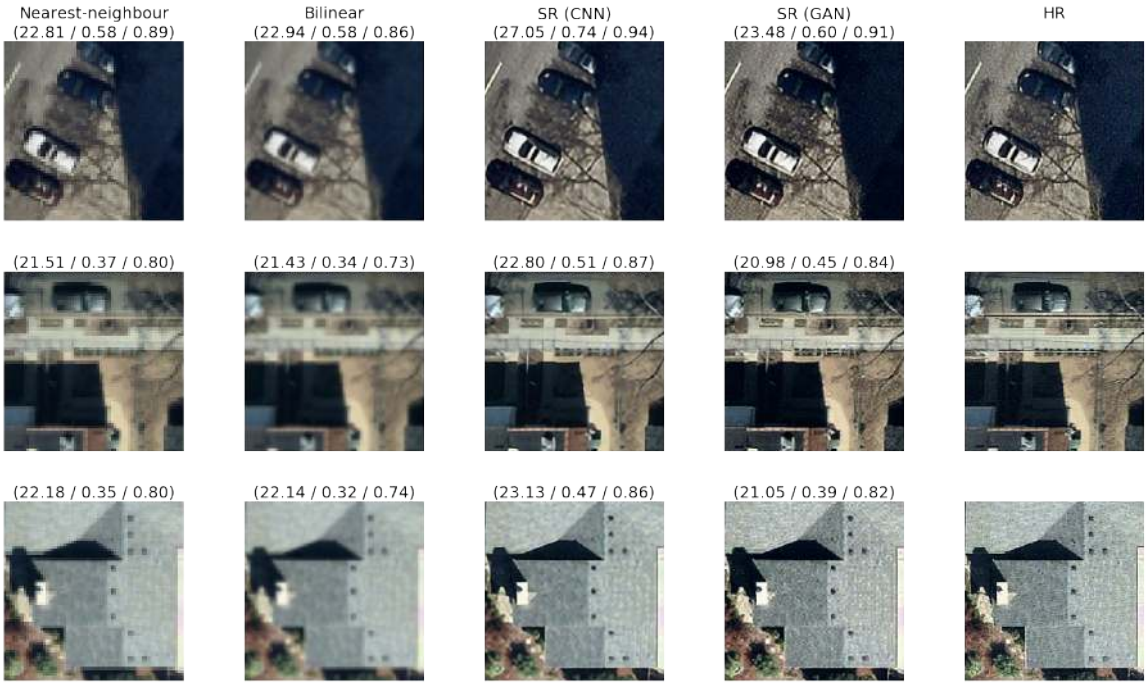


FIGURE 5.18. Samples from 4x super-resolution using the GAN network. The columns from left to right represent the bilinear upsampled, GAN super-resolved and high-resolution ground truth images. Metrics listed in the brackets are PSNR, SSIM and MS-SSIM respectively.

Factor	Bilinear			GAN			CNN		
	PSNR	SSIM	MS-SSIM	PSNR	SSIM	MS-SSIM	PSNR	SSIM	MS-SSIM
2x	27.06	0.7914	0.9559	31.26	0.9125	0.9915	35.08	0.9526	0.9971
4x	22.73	0.5262	0.8332	23.17	0.6077	0.9025	26.05	0.7018	0.9334
8x	20.37	0.3795	0.6585	20.49	0.3834	0.7332	22.41	0.4860	0.7903
16x	18.74	0.3287	0.5028	18.83	0.2998	0.5638	20.19	0.3823	0.6327
32x	17.30	0.3086	0.3801	15.34	0.1995	0.3401	18.34	0.3323	0.4811

TABLE 5.6. Quantitative comparison of image quality metrics on the test set for simple bilinear resizing, CNN super-resolution using a pixel-wise loss and finally GAN super-resolution. Using a pixel-wise objective in our CNN model yields the highest metric scores. Whilst the GAN performs worse than the CNN model in these metrics, it performs as well as or better than bilinear at all scales but 32x.

When compared with the bilinear interpolation baseline, it was clearly evident that our GAN model offered a sizeable advantage in terms of superior image quality and reconstruction detail, especially in a human-subjective evaluation. In the case of 4x super-resolution we found the super-resolved images produced by the GAN to replicate images very well, and without looking all too closely at the fine-grained details our generated images were difficult to differentiate the ground truth. In these results it was very apparent that the model was not only refining edges and outlines, but also being highly capable in synthesising believable textures for features like tree leaves and branches as well as tile and shingle roof patterns, to name a few.

(A) CNN super-resolution. (B) GAN super-resolution. (C) Original image.

FIGURE 5.19. A close-up detail comparison of an image patch, showing how the GAN, although not perfect, manages to perform much better in reconstructing the intricate detail of the tree leaves than without it.

In order to provide a fair comparison and justify the use of GAN models as a worthwhile successor to plain CNN models, we also compared our GAN results to those from the super-resolution model optimised with a pixel-wise loss. The results stemming from the GAN optimised model are evidently much more perceptually pleasing to a human viewer as they provide a much more realistic rendition of details in the image. Rather than visually flat and blurry surfaces, we have complex and semantically truthful detail being filled in.

Despite this, one major point of contention that might be raised with this model is its performance in our designated image quality metrics. In our tests of super-resolution, the GAN model did indeed typically score higher than traditional up-sampling techniques such as bilinear and bicubic in PSNR and SSIM. Our GAN model managed to increase PSNR by 0.44dB, SSIM by 0.0815 and MS-SSIM by 0.0693 when compared to bilinear up-sampling for 4x super-resolution. However when compared to the pixel-wise loss optimised model, it consistently scored lower in PSNR and SSIM metrics and by a substantial amount. Our CNN model optimised for pixel-wise loss instead increased PSNR by a greater 3.32dB, SSIM by 0.1756 and MS-SSIM by 0.1002 when compared to bilinear up-sampling for 4x super-resolution. This is a common characteristic identified extensively in the super-resolution literature whereby optimising for perceptual quality versus maximising PSNR are often considered to be orthogonal objectives and impossible to jointly maximise [72]. Since PSNR is directly optimised by minimising the MSE pixel error, this understandably produces the most optimal PSNR scores. However as identified before, visually all MSE/MAE optimised models score quite poorly in terms of human perceived visual quality. Hence this leads to the great debate that PSNR and SSIM are not fully representative of image quality and in particular image quality as perceived by a human observer.

We also test our GAN super-resolution models over the entire scope of up-sampling factors, up to 32x. We find as with the CNN model it clearly does very well on the lower scaling factor end where it has access to more informative low-resolution inputs and becoming less well defined towards the extreme 32x end. This however is where the GAN has more scope to imagine its own details, as opposed to bilinear up-sampling and to a certain extent typical CNN methods which remain fairly conservative in their estimations.

(A) Original (B) 4x (C) 8x (D) 16x (E) 32x

FIGURE 5.20. GAN super-resolution outputs at different scales. From left to right we have the original image, 4x, 8x, 16x and 32x. Above 4x images begin to lose finer details and by 32x the results tend to deviate noticeably from the ground truth target.

To be in control of the level of "hallucination" of the model we introduce an parameter t to allow us to interpolate the weights W of the network between a PSNR-oriented and GAN-oriented objective. This allows us to choose and balance the trade-off between conservative but over-smoothed versus hallucinated but perceptually sharp outputs.

$$W_{interp} = t * W_{GAN} + (1 - t) * W_{PSNR}$$

FIGURE 5.21. Results showing the successful interpolation between a PSNR (left) and GAN (right) objective, allowing us to control the degree of perceptual hallucination of the GAN.

5.3.2 Semantic Segmentation Enhancement

Furthermore, we applied our GAN super-resolution algorithm with the goal of increasing semantic segmentation accuracy on a swathe of classes relevant to aerial imagery such as roofing, roads, cars, solar panels and many others.

5.3.2.1 Evaluation of IoU

To make sure that our model worked well across the board and that we weren't simply identifying favourable examples, we evaluated the IoU accuracy of the model across a large test set of unseen images. For this we computed the intersection over union of masks generated by our model by comparing them to a ground truth of human-labelled masks. To show the improvement our model provides, we also repeat this for a simple bilinear resize model and take the difference in IoU between the results of both evaluations, giving us the relative improvement in IoU offered by our GAN model over a bilinear up-sampling model.

Looking at 4x super-resolution to begin with we can see that the GAN super-resolution models provides significant improvements in IoU prediction accuracy over a simpler bilinear regime. As expected, the GAN models sees the most improvement over the bilinear model in texture and detail rich categories such as tile, shingle and metal roofs. This is because the bilinear model heavily blurs the image and washes out any intricate patters, whereas the GAN is optimised to reconstruct the details of natural images. Likewise, it also seems to benefit some vegetation categories (that include finer leaf details) as well as important categories like trampolines, cars, swimming pools and solar panels. Another confirmation of the GANs attention to detail is seen in the marginal improvement in featureless categories such as shadows, roofs and hip roofs. For detection, these classes can be almost wholly segmented based on edge and shape information alone and hence the presence of texture detail is not terribly important for these categories. Thus, we see them only marginally better in the GAN that the bilinear model, not because the GAN does not detect them well but rather because they already segment well from bilinear up-sampling because they are low in detail. Lastly, some of the categories such as mansard, gambrel and turret roofs at the lower end of the plot are very rare classes and barely show up in our evaluation data, hence for the purposes of these experiments they can ignored on the basis of not having a statistically significant number of data points.

Hence, this shows that if we would want to predict on detail rich and specific categories like tile roof as opposed to just roofs in general, then the GAN model for SR would provide an enormous benefit to accuracy over a bilinear or even conventional CNN approach.

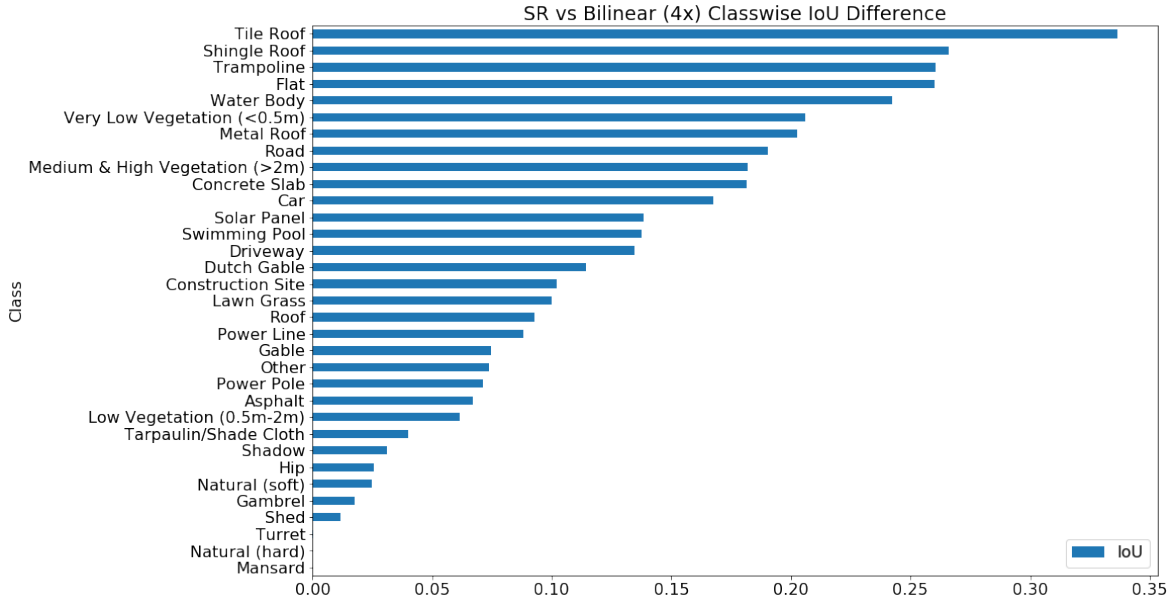


FIGURE 5.22. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and bilinear up-sampled images at a factor of 4x.

We go further to compare these results across other sets of resolutions up to 32x. For the larger up-sampling factors we see similar trends play out. Once again, the GAN excels in resolving roofing like shingle and tile roofs as well as detailed items like cars, trampolines and solar panels. We continue to see a net positive increase in the performance of every single category, except now at the higher up-sampling factors the upside gain in IoU becomes even higher, reaching a significant 0.3-0.6 IoU improvement in the top half of categories. We expect this since as we move to higher and higher up-sampling tasks, the information in the low-resolution image becomes more and more sparse. At 32x super-resolution one only has an 8x8 input image to work with, super-resolving that to 256x256. Hence a bilinear model will not adequately generate good predictions because it maintains the low information of the prior. However, the GAN model is extensively trained to perform inference on these low-resolution priors to spot patterns and fill in imagined detail when it can. We see that with

higher up-scaling factors the super-resolution task becomes more ambiguous and hence the GAN offers a greater level of creative freedom to generate likely details.

For our top performing categories we show that it is possible to increase the segmentation IoU score by 0.34 for 4x and 0.62 for 32x up-sampling (in these cases both of the top performing categories were the tile roof class).

We see that at higher resolutions, rather than just texture information lacking, shape and structure of features also tends to get noticeably distorted. This is therefore why we see at 32x super-resolution that the GAN model excels in broader categories such as detecting roofs generally, since it also acts to restore the general shape of features to ameliorate distortion from down-sampling.

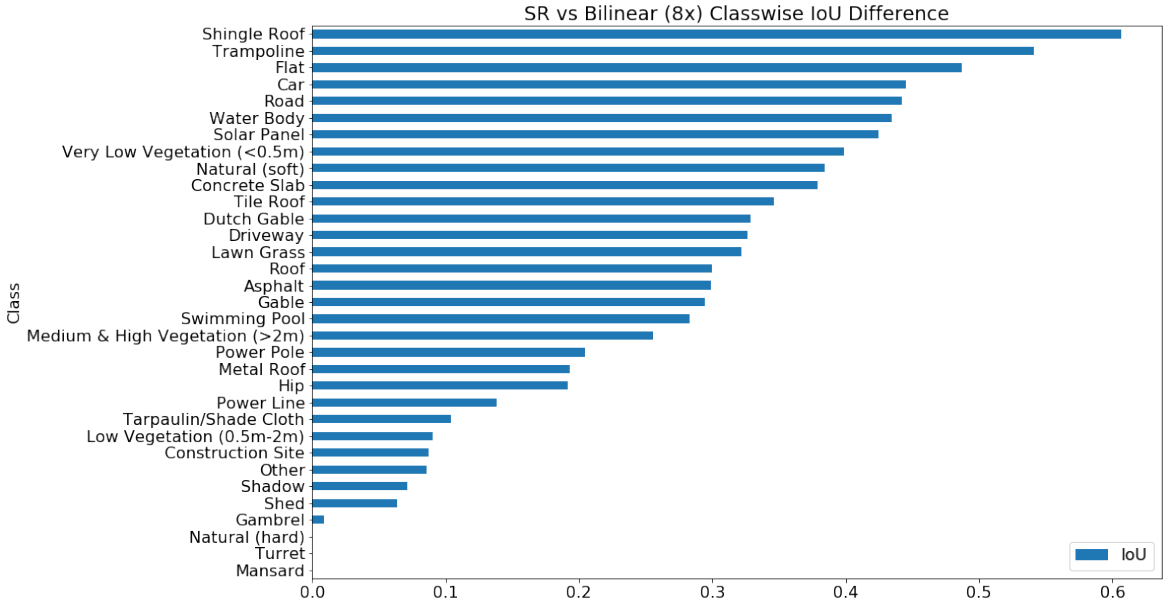


FIGURE 5.23. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and bilinear up-sampled images at a factor of 8x.

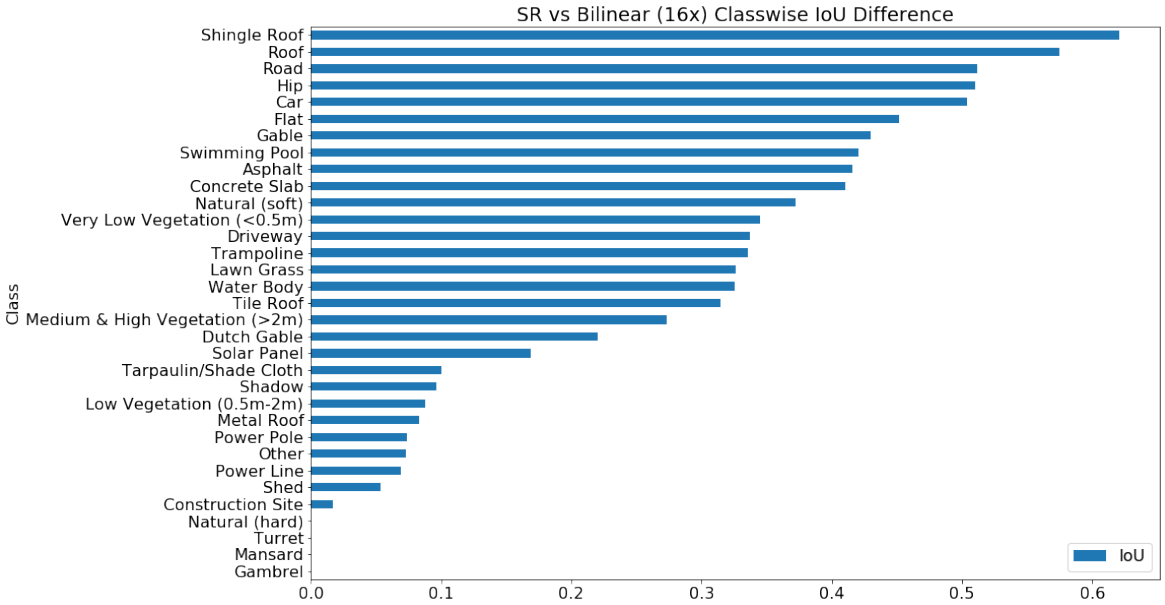


FIGURE 5.24. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and bilinear up-sampled images at a factor of 16x.

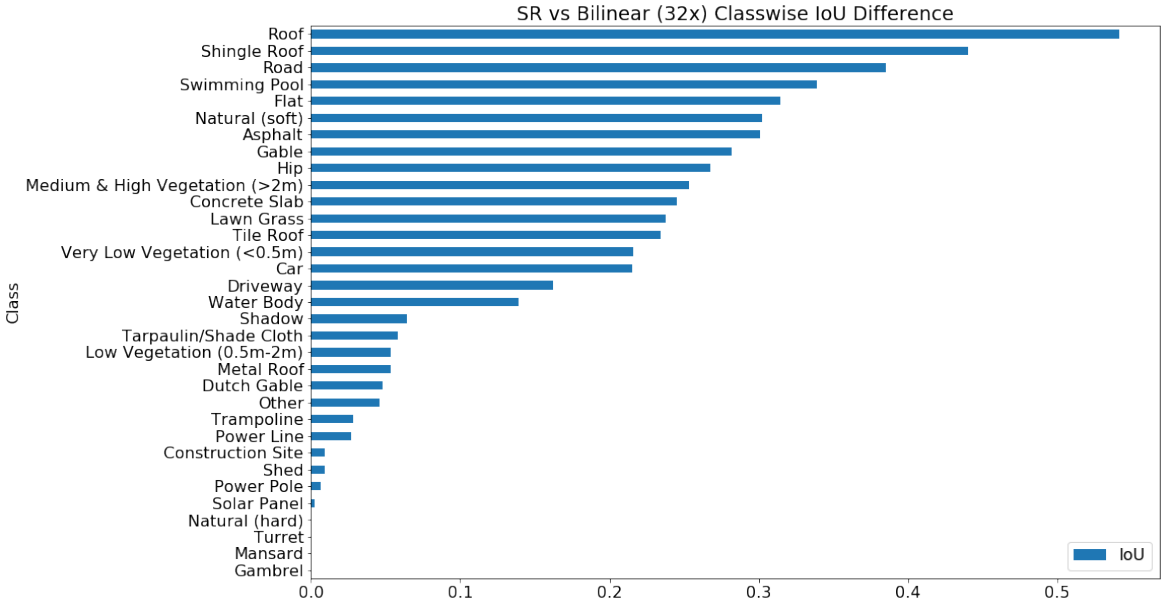


FIGURE 5.25. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and bilinear up-sampled images at a factor of 32x.

Furthermore, we test this further by comparing the improvement in segmentation IoU from the GAN model against the base convolutional model (i.e. without the GAN and feature loss). From this we can see that the advantage of the GAN training over just regular CNN super-resolution is that we can identify almost every single class (with the exception of a few fairly uncommon ones) with a significantly higher degree of accuracy. This is particularly evident when it comes to texture-rich classes such as roof types, succeeding in filling in important semantic detail like tile and shingle roof texture which would otherwise remain blurred in the L1 optimised model. The advantages are less significant over the L1 model for broader categories such as roofs, swimming pools and shadows as the L1 model is already quite proficient at resolving shapes, edges and outlines and for these classes the detail and texture is less important for detection.

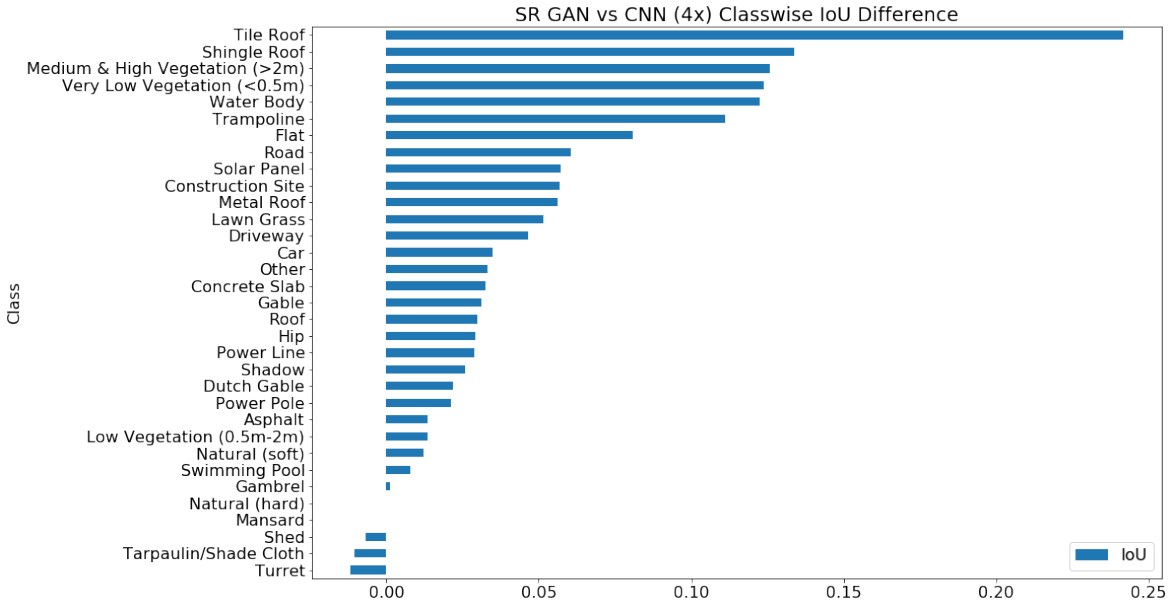


FIGURE 5.26. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and plain CNN super-resolved images at a factor of 4x.

For a full list of the increase in IoU of our GAN model over the L1 model across all other scales please visit section A1.2 the appendix.

5.3.2.2 Results of Semantic Mask Enhancement

In order to confirm the apparent strengths and weaknesses of our super-resolution model when it comes to enhancing segmentation masking, we also display a small sample of images from the test set and their masks. Here we focus on certain classes that the model sees the most advantage with as well as some cases that it struggles to improve.

It is evident from the following examples that the model provides a large advantage over bilinear and L1 optimised CNN models when texture and detail information acts as a crucial signal to the segmentation model. This is most evident in categories such as roof type, where the patterning of shingle and tile roofs is quintessential to proper detection. We also note by inspecting the derived masks that the L1 optimised convolutional model does often improve the quality of masks generated, however the optimisation of our GAN for realism and feature reconstruction gives it a substantial improvement in performance. In the majority of cases the GAN model fully recovers most if not all masked objects in the scene, whereas the bilinear and even purely CNN models tend to have much lower segmentation confidence and often fail to detect a number of key features.

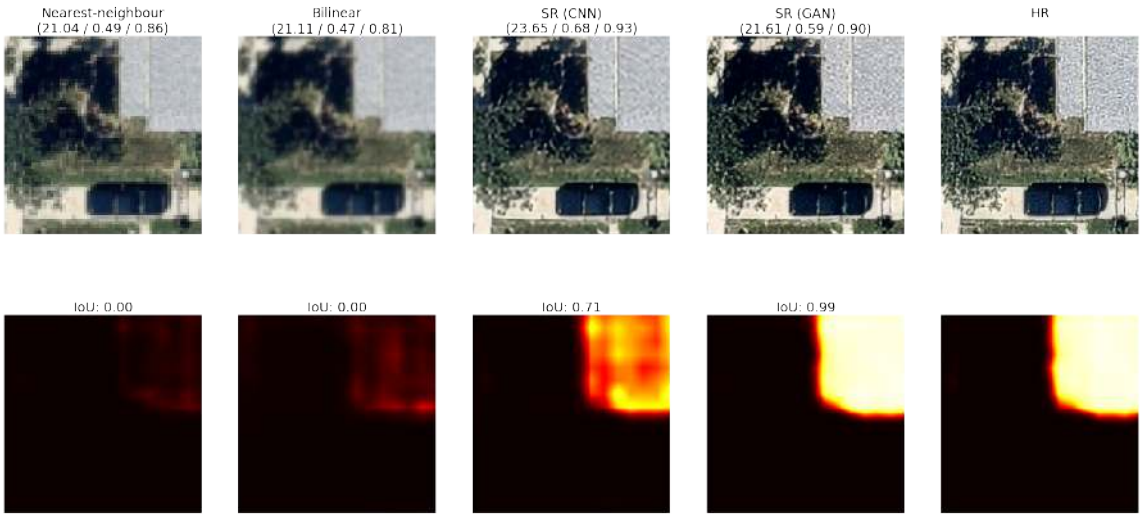


FIGURE 5.27. Comparison between the semantic segmentation masks derived from nearest-neighbour, bilinear, CNN super-resolution, GAN super-resolution and the ground truth image for the shingle roof category. Metrics in brackets are PSNR, SSIM and MS-SSIM respectively.

For the shingle roof category we can see that the sharpening from the CNN model elevates very low-confidence bilinear predictions up to an IoU of 0.71. However the GAN model trumps this significantly by boosting this to an incredible 0.99 intersection over union, in essence reconstructing the mask from the high-resolution image almost perfectly.

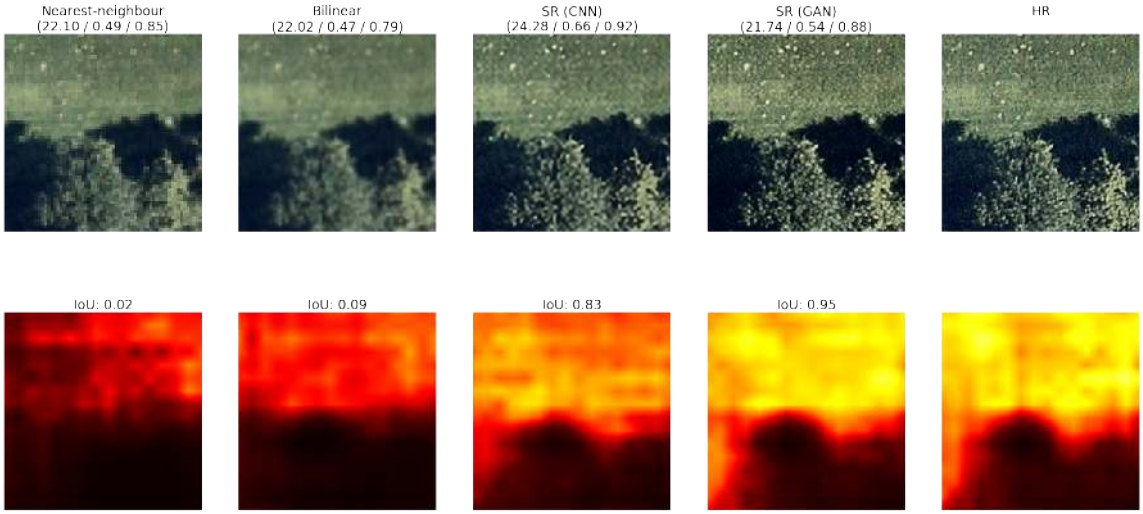


FIGURE 5.28. Comparison between the semantic segmentation masks derived from nearest-neighbour, bilinear, CNN super-resolution, GAN super-resolution and the ground truth image for the low vegetation category. Metrics in brackets are PSNR, SSIM and MS-SSIM respectively.

We show that this not only works well for roof type categories, but also shows great promise in super-resolving natural features like trees and grass. For the low vegetation category we see a 0.83 increase in IoU using the CNN model and a 0.95 increase using the GAN.

Furthermore, this approach helps us better detect smaller clusters of features such as a row of parked cars. In figure 5.29 we see that whilst the bilinear and CNN models only manage to detect 2 cars, the GAN model manages to pick up all four cars correctly in the scene, and therefore seeing a 0.77 increase in IoU.

We also can identify the situations where the GAN provides minimal increase over the CNN model. In figure 5.31 we segment for the road category and with just our CNN SR model we already achieve 0.99 IoU, meaning very little improvement can be made. This is because classes such as roads are primarily defined by their shape and edges and boosting the detail of the asphalt does little to increase segmentation detection.

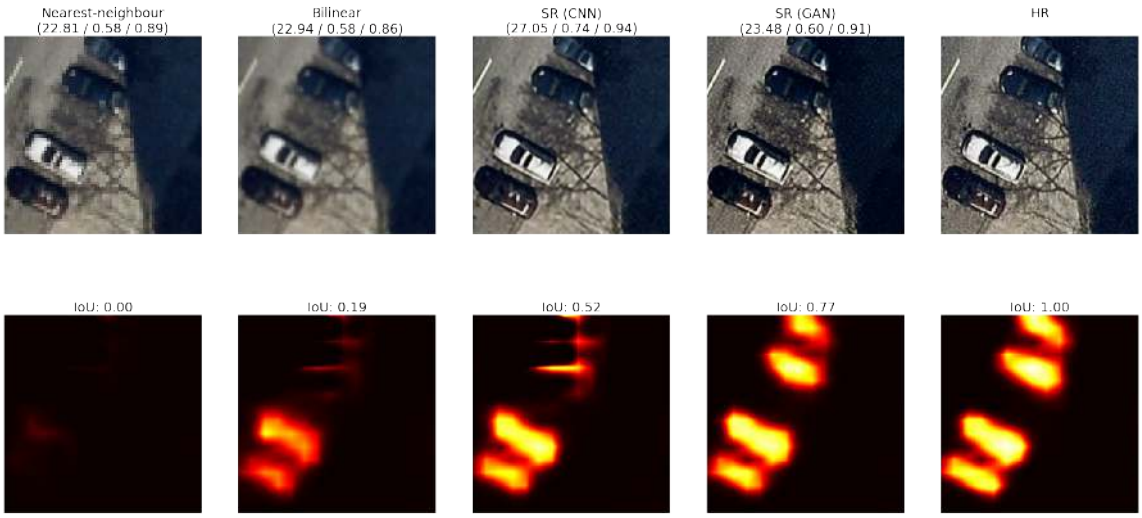


FIGURE 5.29. Comparison between the semantic segmentation masks derived from nearest-neighbour, bilinear, CNN super-resolution, GAN super-resolution and the ground truth image for the car category. Metrics in brackets are PSNR, SSIM and MS-SSIM respectively.

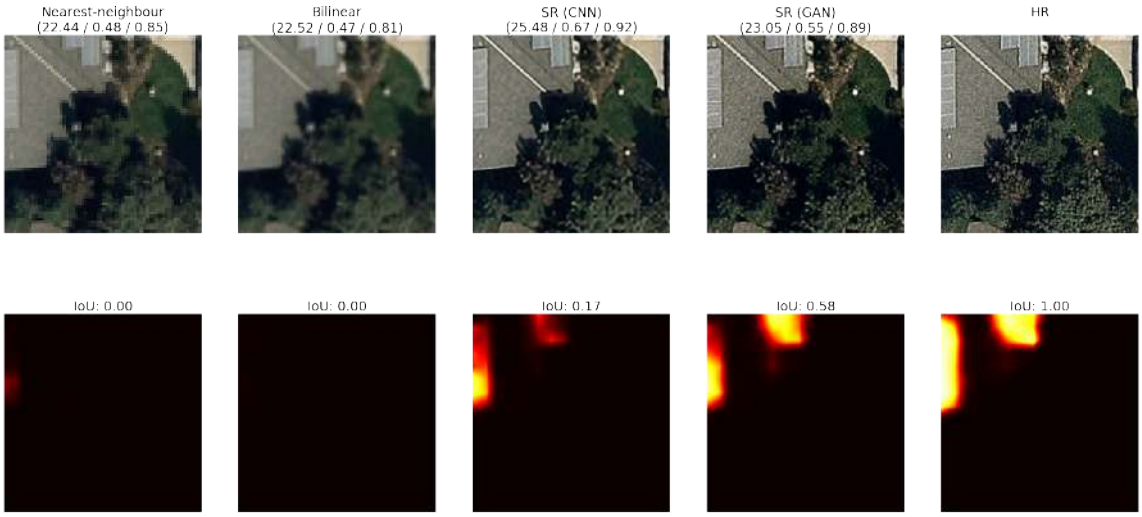


FIGURE 5.30. Comparison between the semantic segmentation masks derived from nearest-neighbour, bilinear, CNN super-resolution, GAN super-resolution and the ground truth image for the solar panel category. Metrics in brackets are PSNR, SSIM and MS-SSIM respectively.

For a full listing of segmentation enhancement examples from all resolution scales please visit section A1.1 in the appendix.

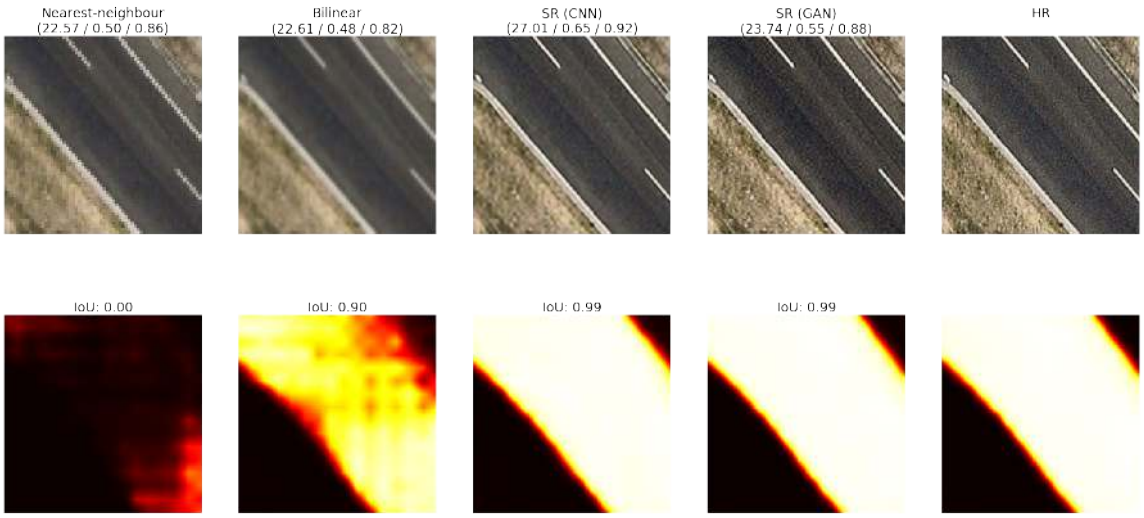


FIGURE 5.31. Comparison between the semantic segmentation masks derived from nearest-neighbour, bilinear, CNN super-resolution, GAN super-resolution and the ground truth image for the road category. Metrics in brackets are PSNR, SSIM and MS-SSIM respectively.

CHAPTER 6

Discussion

In this chapter we build on the results presented in the previous chapter, discussing them in the context of the literature as well as relating them to our overall thesis research objective. In this discussion we both elaborate on the strengths of this approach compared to predecessor techniques as well as highlighting the practical considerations and potential shortcomings.

6.1 Research Aims

Currently, the vast majority of approaches to enhancing aerial imagery, especially to increase the resolution of imagery, use fairly simplistic techniques such as inter-pixel interpolation. The real shortcoming of such an approach is that one does not use any knowledge or information about aerial images themselves to perform this up-sampling operation, rather treating it as simply a geometric averaging of pixel values ultimately leading to blurry and unappealing results. Given the immense success of deep learning to learn complex transforms in computer vision, in particular Convolutional Neural Networks, we investigated the use of these networks for super-resolving aerial imagery. However, as is evident in the literature, these networks although powerful, suffer from less than convincing outputs due to the use of overly simplified pixel-wise loss functions like pixel mean squared error. Thus, we focus on the use of Generative Adversarial Networks as an extension to the basic CNN single image super-resolution framework. We found that they provide a much stronger alternative in replicating human perceptual image quality by employing a fully-learned discriminator network to model image realism as a loss. By optimising aerial images for semantically relevant features like cars, roofing and solar panels for example, we have shown that we can also significantly

improve the detection and segmentation of these features post-super-resolution using typical semantic segmentation networks.

6.2 CNN Super-resolution

By testing all three of our candidate models we found that we could get surprisingly good results for 4x super-resolution using just a simple residual model with 2 up-sampling blocks append to the end (i.e. SRResNet). This approach already yielded results that were far, far superior to any traditional up-sampling methods like bilinear up-sampling, which was of course to be expected. Even simple convolutional super-resolution models were very adept at learning to sharpen edges and refining outlines and prominent features of the low-resolution images. This was in stark contrast to bilinear and bicubic up-sampling which heavily blurred any edges due to the nature of the interpolation.

From running extensive experiments to compare different variants of architectures and parameters we found that the most recent RRDN architecture did indeed perform best, as expected from the superior results of the ESRGAN paper [23]. Being able to add in dense layers alongside the residual connections allowed for high precision as well as relatively fast training times (although not quite as fast as the simpler EDSR model). However this increased performance came at the cost of roughly double the number of parameters, meaning it trained slower and used more memory. We found for RRDN that the depth was quite important and having more blocks in our network (say the original 16 rather than the 4 we ended up using) would have helped a bit in improving performance. However the full 16 block model of RRDN as proposed in the paper ended up at roughly 12M parameters and took on the order of a few days to pre-train, hence we decided this was largely unpractical for our purposes given the relatively small increase in PSNR and SSIM. The main takeaways in our generator design phase were that removing Batch Normalisation layers was an absolute must and hugely boosted both speed, memory efficient and stability and accuracy of training. We also found that pixel shuffle layers helped increase the speed by a noticeable amount and allowed us to use more parameters since we were still doing computation in low-resolution space. This

agrees with much of the literature on the consensus for up-sampling layers, and it offers a massive improvement over old fashioned deconvolution layers.

Despite most super-resolution papers only tending to work with 2x, 4x and on occasion 8x up-sampling tasks, we pushed this to its limit by attempting 32x super-resolution. As expected this was an unbelievably difficult estimation task for the network, however with some fine-tuning we were very pleased with the final results. From a highly-pixelated 8x8 source image, the network was more often than not able to resolve this 32x to resemble something reminiscent of an aerial image. It was often quite achievable to recognise features such as cars and roofs in these images, whereas any such inference on the low-resolution 8x8 image was quite literally impossible for a human. This highlighted the immense strength of deep learning networks to build large and sophisticated models of the world and infer a lot of detail and semantic information not only based on the immediate pixels, but the entire context of the image.

In accordance with many deep learning super-resolution papers we also found that using an L1 loss objective offered a very slight improvement in final results although the different was almost negligible once trained sufficiently. However, this brought us to our most important realisation and a large motivation for this thesis. Despite all the sophisticated network architecture changes and innovations, the output results of all of our CNN based super-resolution models lacked a serious level of detail. Other than the sharper edges, results were characterised by blurry reconstruction of textures throughout the image. This, as suggested extensively by the literature [46] [72], was first and foremost a symptom of the over-simplified and inexpressive pixel-wise loss functions commonly used in super-resolution implementations. Since these L1 and L2 loss functions performed a simple pixel by pixel error value, they largely ignored relationships on a macro scale and did not pay significant attention content of the image itself, rather doing something like sparse coding by simply learning a mapping from low-resolution patches of pixels to high-resolution ones. Since these loss functions also compute their loss averaged over all pixels in the image, this then encouraged the network to learn to generate images that were blurry and featureless as the error was lower on average. As identified in many papers, this results in images that are unappealing and

unrealistic to the human eye, hence motivating a search for more representative image loss functions.

We adhered to the common metrics used in super-resolution literature, primarily PSNR, SSIM and MS-SSIM. We found these to work suitably well in the case of optimising for an L1 or L2 objective since metrics like PSNR are directly optimised for by an L2 loss. Visually, results that scored higher in PSNR and SSIM also correlated fairly well with our visual evaluation of the quality of the image. It was clear that when PSNR and SSIM scored higher, lines were almost always sharper and shapes more well defined as one would expect in a good super-resolution function. Some unfortunate downsides to these metrics however were that they are full-reference image metrics, meaning that they require a ground truth image to compare against. This is not possible in the cases whereby you super-resolve an image above its native resolution and thus have no reference image available, making it difficult to quantitatively judge the quality of such an image.

Some other noteworthy shortcomings of this approach to super-resolution were that we found the algorithm to be very dependent on the method of down-sampling (to create the low-resolution image pairs). This is unfortunate because it means that in order to up-sample well, you would ideally want to have a good model for the down-sampling too in learning to reverse that operation. Since the majority of our results were based on the assumption of a bilinear down-sampling function, this meant that other down-sampling functions such as bicubic would cause a degradation in super-resolution performance. We also down-sampled using an anti-aliasing filter and noticed that if this was removed, the super-resolved images also suffered in quality since the algorithm had only been trained on anti-aliased imagery. Furthermore, this highlights the rather difficult fact in many super-resolution papers that bilinear or bicubic down-sampling functions don't reflect the down-sampling characteristics of real images exactly, meaning a lot of these algorithms when applied to low-resolution imagery in practice don't fare as well as their results might show. To overcome this, one would have to train the algorithm on true high and low resolution images, say taken at different focal lengths of the camera or else come up with a very good model of the down-sampling in a physical camera.

Finally, we recognised that it was a drawback to only be able to super-resolve at a single fixed scale for each of our various models. Hence for different up-sampling factors we would have to maintain multiple different models each trained individually. This invites scope for learning a multi-scale model as was explored briefly in our MSG-GAN experimentation. This is also explored in the EDSR paper, which proposed a multi-scale alternative called MDSR. The advantage of such a model would be to super-resolve imagery more readily without being limited to a single resolution that it was trained on. Furthermore, it is also advantageous in fact to train multi-scale models as they tend to learn more robust multi-scale feature representations [38]. We experienced this ourselves when we used our initial 4x super-resolution model as a starting point to transfer learn our other resolution models from. We found that not only did it train faster, needing only to fine tune an extra layer or two at the end of the network, but also after extensive training these transfer learned models always achieved a higher PSNR and SSIM than models trained from scratch. This was likely both from the additional training as well as the fact that features learned from other resolutions helped inform the super-resolution task at the resolution in question.

6.3 GAN Aerial Image Synthesis

In learning to use GANs to synthesise aerial imagery, we demonstrated their incredible ability to model the complex and diverse nature of our dataset. This fits in with our overall strategy of super-resolution as we seek to employ GANs as a means of building more informative image loss functions. Since our initial L1 and L2 loss functions are on the whole too simplistic for the task of super-resolution, we look to use this GAN loss as judged by an auxiliary discriminator network to point us in the direction of images that align more closely with human visual perception. Furthermore, by first learning to synthesis images using a GAN, it becomes simple to modify this framework to take in a low-resolution image prior rather than a random Gaussian latent vector as is commonly seen in image synthesis tasks.

This task was inherently challenging because our dataset was large and very diverse, containing any manner of vertical aerial imagery from across thousands of geographic locations. To

model this using a simple network was much more difficult than say learning to model human faces from a dataset such as CelebA, in which every image contains a similar set of features such as eyes, mouths, noses, hair and all images are in roughly the same orientation. Nevertheless, we constructed both a single scale DCGAN as well as a multi-scale MSG-GAN model. First and foremost, this introduced us to the vast difficulties associated with training GAN models as they are universally known as being unstable and very tricky to train correctly. To remedy this we scoured the literature for ways to improve this, finding the most inspiration from several well-known sources in the field [55] [63]. By implementing a number of training tricks, regularisation and normalisation techniques and modifications to both the training algorithm and model architecture, we managed to greatly improve the stability and final outputs of our GAN models. We found that the major improvements have come very recently, with Spectral Normalisation of GAN weights offering a significant boost to the stability to model training as well as discriminative learning rates. The biggest improvement however was using recent model architectures like MSG-GAN and other StyleGAN and ProGAN variants. The design of these networks involved a lot better flow of gradients between generator and discriminator networks and included many of the recent regularisation techniques to stop the discriminator from overfitting. This provided a significant improvement over older DCGAN models, allowing us to generate large scale aerial imagery up to 512x512px with a fairly convincing reconstruction of the typical features of an aerial image. We believe that even though GANs have been very much confined to the theoretical research domain, more stable and effective architectures, training and optimisation techniques are lowering the barriers to entry. It is clear that given their unbelievable skill in modelling such complex datasets that they will no doubt begin to see more and more mainstream use in the generation of synthetic data and unsupervised modelling of very high dimensional image (and other) datasets.

6.4 GAN Super-resolution

In light of the previous section, we applied our knowledge of GANs in the context of our overall super-resolution objective. Conveniently, we could make minor modifications to our GAN synthesis framework to accommodate for a low-resolution image instead as our input.

This low resolution image clearly allows for a much more informative prior for the GAN, making the super-resolution task significantly easier for a GAN than of image synthesis from an arbitrary vector. We believe this makes super-resolution a particularly interesting application of GAN networks because it is inherently much more information rich and allows typically unstable GAN models to be much more manageable to train. This, as well as the obvious advantages to being able to intelligently up-sample images, make it a very practical and useful application of GANs.

Integrating the GAN framework with our existing super-resolution models was fairly straightforward as it required no changes to our CNN super-resolution model. Rather, we wrapped this model in an adversarial training regime with a second, independent discriminator network. This independence of models made it overall very easy for us to first pre-train the generator network, elevating it to a high baseline, before training it adversarially. We found that this helped an inordinate amount with improving the stability of GAN training, as often training from scratch is slow and allows for the unstable GAN model to diverge.

We believe that this addition of more informative loss functions, especially functions that are customised for specific image-related tasks, is a major step in the right direction for achieving better performance in convolutional neural network models. As we saw in our own experiments, the extensive redesign of the network architecture only allowed us a marginal improvement in our metrics with the visual differences between our RRDN and SRResNet models being more or less unnoticeable to the casual observer. However, by concentrating instead on the loss functions used to optimise rather than the network design itself, we found that we could achieve remarkably more visually appealing results. As discussed extensively before, moving away from a simple L1/L2 loss function and towards semantic image loss functions like feature loss using auxiliary classifier/segmentation networks as well as GAN discriminator losses made a huge difference in the perceptual quality of our results. This allowed our model loss objectives to be far more expressive and capture a semantic understanding of the images we were super-resolving, as well as enforcing that images followed the expected, "realistic" nature of our ground truth dataset. This helped to

penalise models that previously were content with producing blurry and texture-less images as they overlapped poorly with the characteristics of real images from our dataset.

We establish then that the use of custom image loss functions provide a significant improvement over typical approaches to training convolutional neural networks, especially for tasks like super-resolution where detail is everything. We also agree with the literature that it is the combination of both the GAN and the feature loss function that generate the highest quality results [23]. In our work we show that not only can we use VGG style classifier networks trained on ImageNet as our feature loss (as is almost always the norm), but it is absolutely feasible to also employ your own custom trained semantic segmentation network. This improved method of training super-resolution networks shows considerable advantages over simpler up-sampling techniques such as bilinear and bicubic interpolation and even standalone deep learning networks like CNNs. We argue that although the use of additional loss functions requires more complexity and can be harder to design, tune and train, it affords much better results and is suffers no additional cost at inference time. This is because when we perform inference using our GAN super-resolution model, we are free to discard the discriminator network and hence incur no slowdown in speed or increase in model memory usage. One may liken this to simply engineering a better "teacher" network and objective function at train time, using large discriminator networks and complex losses to distill better representations for the existing generator model.

In this work, we came to realise that almost all the common image metrics are flawed in one way or another, especially with respect to their ability to encapsulate human subjective perception of image quality. We found that although our GAN results appeared far more realistic than the non-GAN outputs, they scored considerably lower in PSNR, SSIM and MS-SSIM. This therefore calls for image metrics that better reflect this quality. A number of attempts to achieve this have come about such as Ma's score [75] and NIQE [76], which are both non-referential image metrics. Likewise, Netflix has released their own full-reference metric VMAF for evaluating subjective video quality [77]. Unfortunately however, due to time constraints and the fact that these metrics not being standard practice in literature yet, in

addition to having their own downsides and failure cases, we decided not to include them in our thesis.

In terms of our overall thesis strategy and goal, we managed to firstly super-resolve images all the way up to an up-sampling factor of 32x. Smaller scale models at 2x, 4x and even arguably 8x managed to achieve super-resolution that was an very similar to the ground truth, making a very strong case for the perceptual quality that a GAN model can offer. We pushed the network to its absolute limits by also achieving super-resolution results at scales of 16 and 32x. Although these were not intended to afford much visual similarity to the high-resolution image (since there was simply not enough information to perform reconstructions from), we instead aimed to use these models for optimising semantic aerial image features. By up-sampling by a factor of 32x, we were mimicking the change in resolution observed when switching from Nearmap's zoom 16 overview camera system to its high definition zoom 21 system. By doing so, we wanted to prove that imagery at the low resolution of the overview camera could be optimised enough using our super-resolution algorithm to allow us to better detect semantic features such as roof outlines and roads from extremely low-resolution imagery. We proved through a number of experiments that super-resolution, especially GAN super-resolution, had a major effect in increasing the accuracy of our existing semantic segmentation model when applied downstream of our model. We believe the use of unsupervised networks like the one we have proposed as a pre-processing and data optimisation step for later networks and other computer vision tasks is an interesting avenue of exploration and it would be fascinating to see what other tasks it might enhance.

6.5 Summary

To summarise, the results we achieved through our design, experimentation and evaluation of contemporary approaches to single image super-resolution showed extraordinary promise for the use of Generative Adversarial Networks, as has been the general trend in the literature. This is in the way they allow us learn complex image loss functions and enhance the perceptual quality of regular Convolutional Neural Network models, despite common image metrics

failing to capture this. We successfully applied our models to super-resolve aerial imagery up to 32x, which not only served to improve the resolution of Nearmap imagery in a perceptually pleasing manner, but also demonstrated potential the for enhancing accuracy for their semantic segmentation network. We therefore view GAN models as being a powerful tool in general for image enhancement tasks as they offer a useful framework for modelling highly complex image data, demonstrated here in the case of vertical aerial imagery. Although often difficult to train and requiring sufficient expertise, we believe that these approaches will begin to see more widespread use in production systems as we have achieved due to newer architectures and training techniques as well as a greater understanding of the nature of optimisation in an adversarial setting.

CHAPTER 7

Conclusion

This thesis presents a powerful and extensible framework for enhancing aerial imagery through super-resolution by applying cutting-edge techniques related to Generative Adversarial Networks (GAN). To narrow the scope of our research, we honed in on the specific task of single image super-resolution and set out to tackle it in the domain of aerial imaging and remote sensing. In our work we set out to identify common techniques in image up-sampling, namely the widespread use of bilinear interpolation in practice, and investigate the advantages of deep learning to produce a learned model for super-resolution as indicated by the literature. We achieved a significant improvement over interpolation techniques using fairly standard Convolutional Neural Networks (CNN), showing that an internal model of the features of our dataset helped greatly when trying to infer edge and shape information, leading to far sharper results.

However in alignment with our expectations from the literature we recognised that no matter what sophisticated CNN architecture we used, the results always turned out smooth and relatively featureless. This hinted at the bottleneck being in typical super-resolution loss function design rather than generator architecture design. This motivated our extensive research into the recent and very successful application of GANs to image processing tasks like super-resolution. By leveraging the GAN framework for learning, we were able to straightforwardly extend our existing CNN super-resolution model to include an additional discriminator network. This allowed us to learn an entirely separate classifier model to judge the apparent realism of generated images, hence providing a useful loss signal to help tune our network to produce more believable super-resolved images. The GAN in combination with an additional perceptual loss function purposed from Nearmap's existing semantic aerial

imagery segmentation model, allowed us to successfully super-resolve images in a way that correlated strongly with human visual judgement of imagery.

We were able to up-sample Nearmap's imagery up to a factor of 32x, thus providing a way in which to intelligently fill in potential holes in survey regions that had only been captured by the low resolution overview camera system. Furthermore, we found that by super-resolving these low resolution images, we could directly optimise them for semantic segmentation later on, demonstrating a significant increase in the accuracy of derived semantic masks. We also demonstrated the applicability of such a model in production, as it is both fast and powerful and requires no more overhead than a standard CNN model due to all GAN modifications taking effect only during the training phase (and not during inference).

Hence, we strongly believe that Generative Adversarial Networks are a powerful method for modelling the intricacies of complex datasets. So far we see them being most applicable to computer vision tasks, especially tasks that require realism and detail to be believably modelled such as super-resolution. Although they are typically confined to the realm of research, we see them being a vastly practical tool for commercial applications such as we have demonstrated in aerial imaging, especially as the field matures and more robust training methodologies become available.

7.1 Implications

Despite the sizeable remote sensing industry and immense amount of image data it produces, the vast majority of applications fail to make the most of incredible technologies such as deep learning. With giants such as Google integrating deep learning on phones to enhance imagery and allow you to take incredible low-light photos for example, it is evident that learned post-processing techniques have massive potential. Since one of the fundamental limitations of any sort of remote imaging is typically the resolution at which photos can be taken, it is clear that being able to digitally enhance this in post-processing would be hugely valuable to create larger, sharper and more detailed photos. Deep learning is perfectly suited to this as it can learn generalisable characteristics over a huge set of data, performing highly

complex image transformations that are simply not possible using traditional techniques, especially not automated and at scale.

We see Generative Adversarial Networks as taking this amazing ability to the next level, as they augment conventional Convolutional Neural Networks with a much more informative and knowledge-rich visual objective function. This is essential for learning important semantic details about the scene, such as say the pattern of tiles on a roof which if not well modelled may look very noticeably artificial. This has the major advantage of allowing us to better model the subjective human perception of imagery, something which can not possibly be encapsulated by a simple pixel-wise loss function. This therefore has huge potential in all areas of deep learning for computer vision in encouraging more sophisticated image models to be learned. The counterargument to this however is that GANs permit the model with a greater degree of flexibility to "hallucinate" synthetic details, which although seemingly realistic, may not accurately depict the true nature of the ground truth image. It is therefore important to be aware of when synthetic data is generated, and be potentially able to modulate the level to which information is synthesised.

Finally, we see GANs as being an invaluable tool in the deep learning toolbox. This is especially true as we begin to see more and more the importance of large-scale unsupervised learning of patterns from unlabelled datasets as labels are very expensive and data is becoming ever cheaper. This is highly applicable for representation learning of key component features and semantics of massive datasets and as we showed it can be helpful as an initial optimisation step for later deep learning tasks. Although GANs are typically considered among the community to be generally very difficult to train and lacking practical applications, we demonstrate in our work that given the right architectures and with a few of the best practices for ensuring more stable training it is feasible to apply these models to production systems as we have done. Our hope is that we see a greater adoption of generative modelling techniques both in this industry and also generally in other deep learning applications.

7.2 Future Work

Generative Adversarial Networks provide a powerful yet flexible augmentation to more standard deep learning methods. As a result they are very versatile and can be readily applied to a number of additional tasks relating to aerial image processing.

Firstly, we believe the extension of our single image super-resolution pipeline to deal with multi image super-resolution would be a worthwhile endeavour. By being able to pool details and features across several different images this would greatly benefit our algorithm in being able to perform more accurate super-resolution. Not only would the signal for the model be much higher, but this would also help it to circumvent nasty cases where parts of the image are occluded or otherwise difficult to view. For example if an important feature laid in the shadow of a nearby building, single image super-resolution would be limited very poor quality data for its task. However if it was able instead to pull this information from the same location captured at a different date and potentially different sun angle, then super-resolving might become substantially easier. Likewise, being able to leverage Nearmap's accompanying oblique view imagery along with the vertical imagery we exclusively used would likely be very helpful in synthesising information sources, allowing us to have better intuition in our super-resolution even if features like light poles were not easily visible from a top-down perspective.

(A) Original image.

(B) Captured at a different date.

(C) Oblique view of image.

FIGURE 7.1. Multi image super-resolution might synthesise its input information from multiple different captures of the location or from oblique angle views.

Secondly, another interesting use-case that was pointed out was to use the GAN network to perform what is known as image in-painting. This involves masking a region of the image out and getting the model to predict the information that has been withheld. This allows us to create something analogous to Photoshop's content-aware fill, where missing parts of the image can be intelligently inferred from the surrounding pixels. This could be useful to paint out obstructing features such as shadows or tree overhang on roofing. GANs are especially suited to this task since they specialise in building a generative model for the dataset, hence allowing us to readily generate synthetic but believable data to paint into the image.

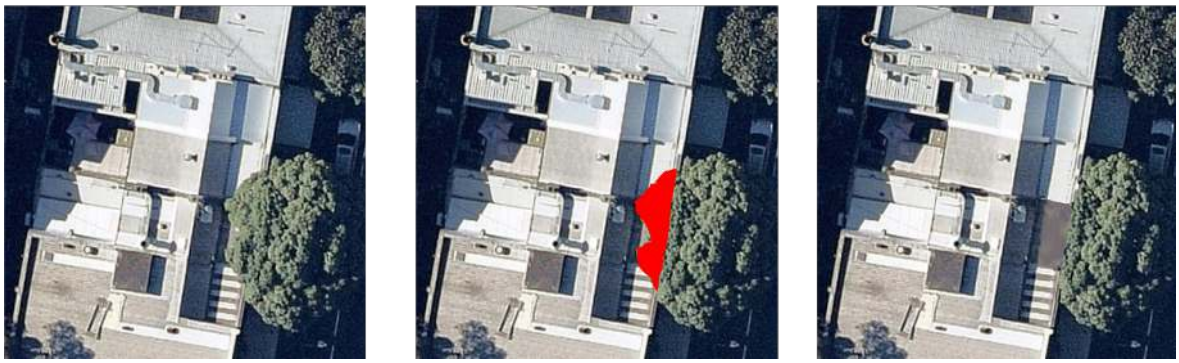


FIGURE 7.2. An illustration of the expected behaviour of the proposed tree overhang in-painting model. By using the tree overhang masks Nearmap already segments for, we can use this to automatically create masked and unmasked image pairs for the dataset on which our GAN model could be trained. This would allow it to fill in its interpretation of what might have existed below the tree overhang.

Finally, one highly practical application of using GANs for remote sensing applications would be conversely to leverage the discriminator network for anomaly detection to perform automated quality assurance (QA) on images. Since the discriminator models the realism or how real/fake an image is perceived to be, it acts as a good critic function for deciding whether any given image lays within our expected distribution of aerial imagery. For images that are blurred, have unusual artifacts or are corrupted, these should be readily flagged by the discriminator as anomalies. Furthermore, this approach can be flipped on its head and one can optimise the generator to generate the most similar possible image to the query image (inspired by AnoGAN[78]). Then by taking the difference between the optimised generator

image and the query image, one can localise the parts of the image that are unable to modelled by the generator and thus likely fall outside of the dataset and are anomalous.

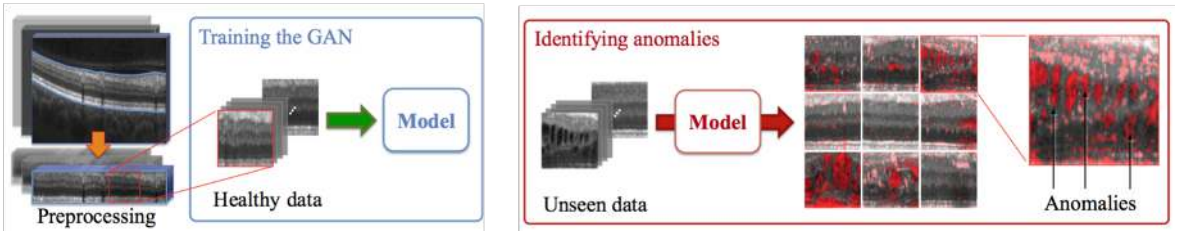


FIGURE 7.3. A diagram showing how a GAN can be used for the detection and localisation of visual anomalies in images [78]

APPENDIX A

Supplementary Results and Figures

This section contains additional findings and figures, omitted from the main body of the text for the sake of brevity.

A1 GAN Super-resolution

A1.1 Segmentation Accuracy Results

A1.1.1 2x Super-resolution

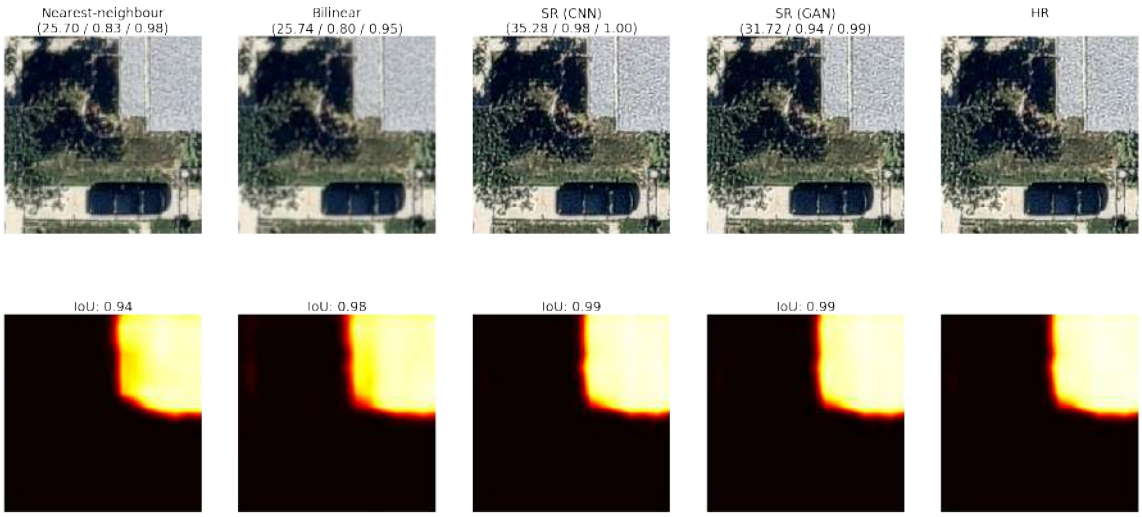


FIGURE A.1. Comparison of segmentation masks for the shingle roof class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

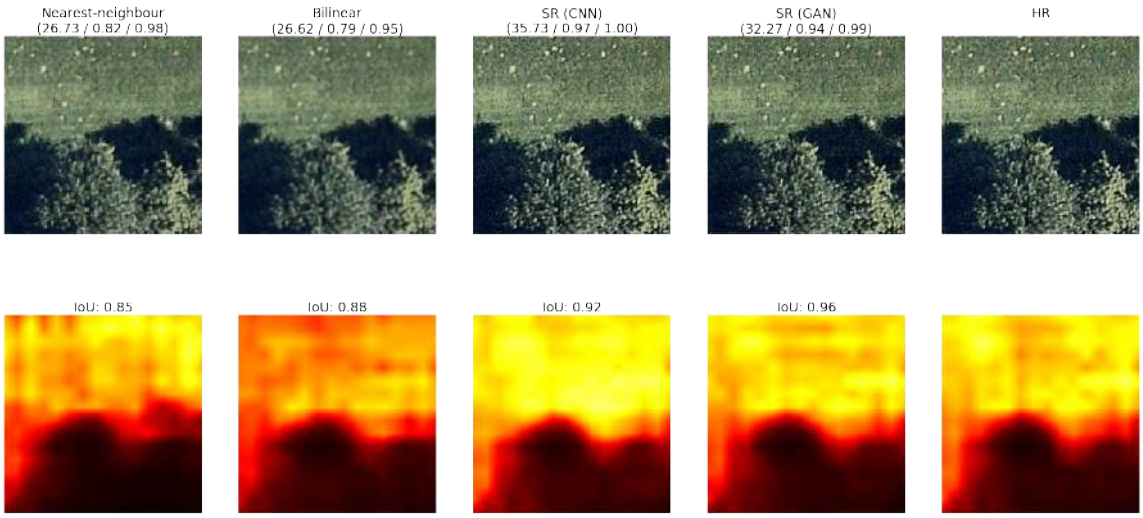


FIGURE A.2. Comparison of segmentation masks for the low vegetation class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

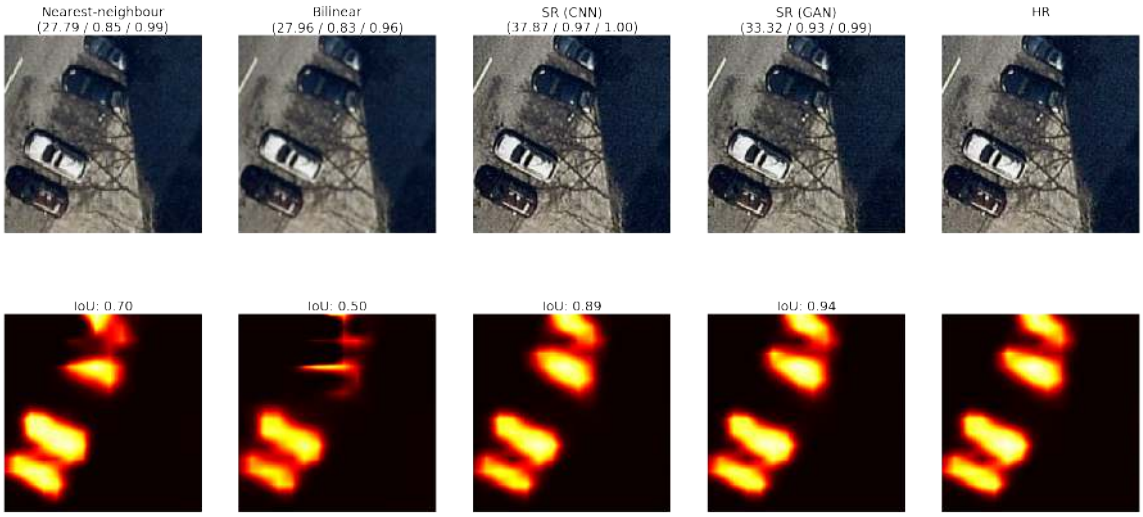


FIGURE A.3. Comparison of segmentation masks for the car class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

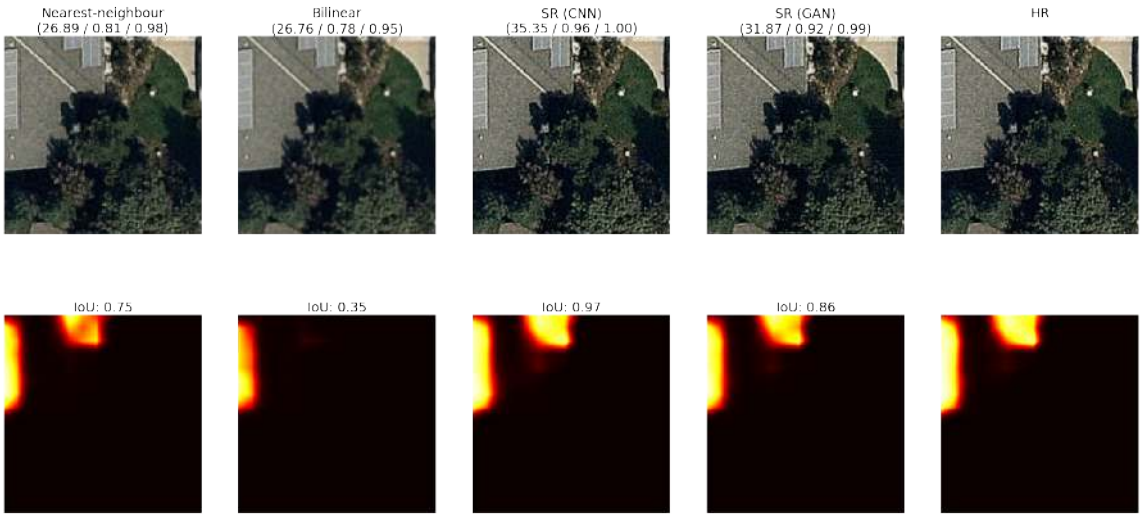


FIGURE A.4. Comparison of segmentation masks for the solar panel class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

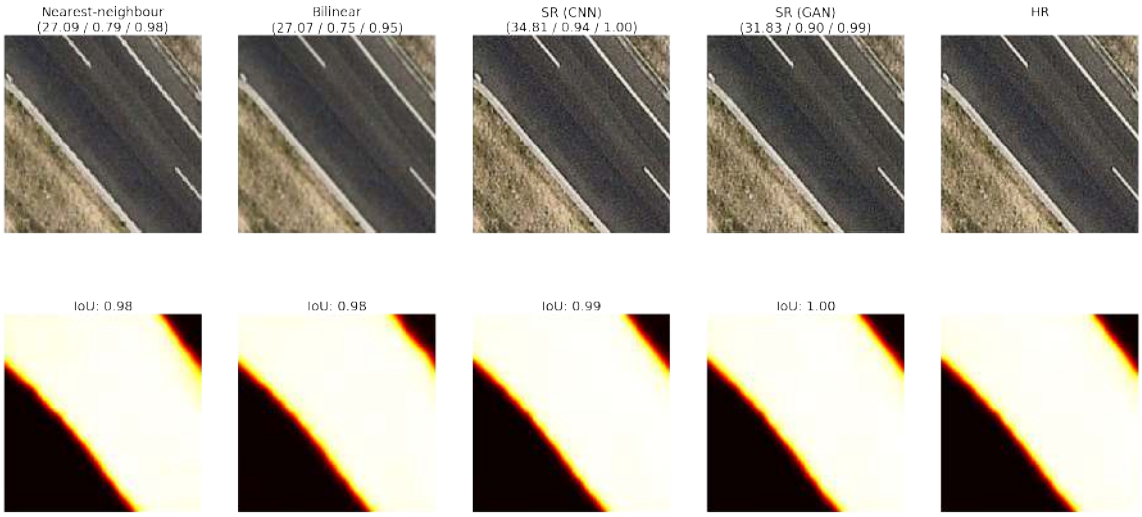


FIGURE A.5. Comparison of segmentation masks for the road class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

A1.1.2 4x Super-resolution

See results section 5.3.2.2.

A1.1.3 8x Super-resolution

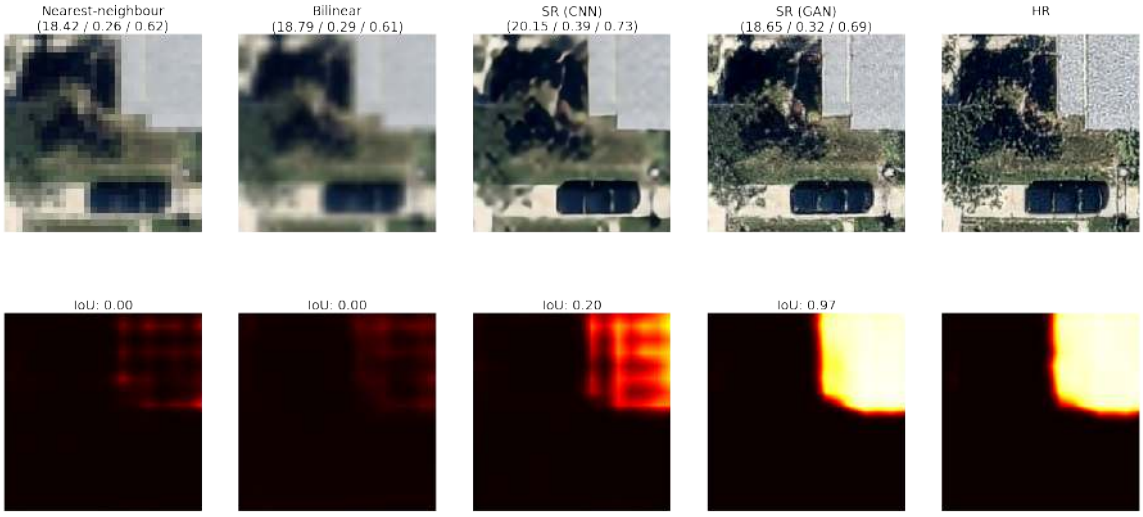


FIGURE A.6. Comparison of segmentation masks for the shingle roof class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

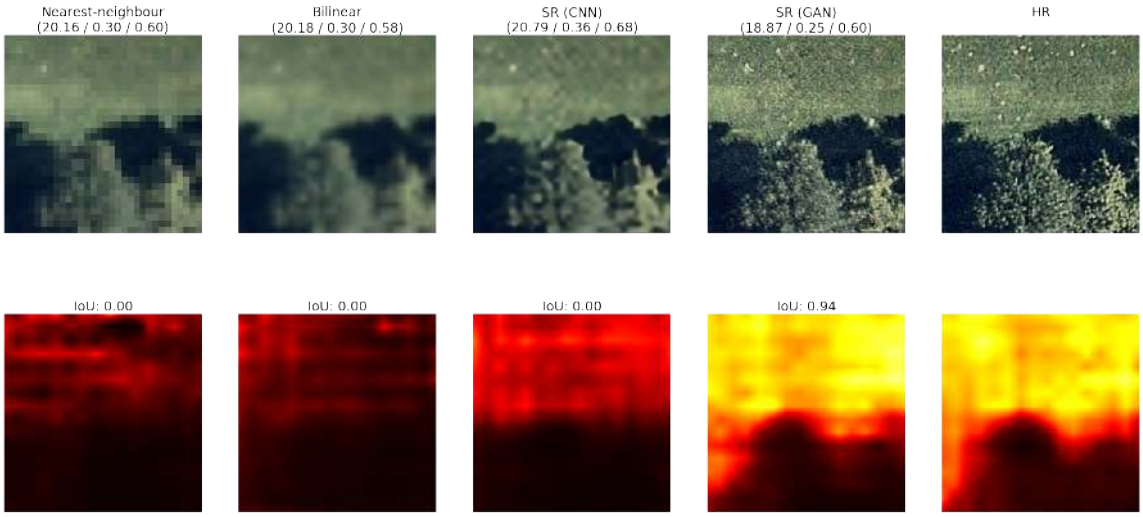


FIGURE A.7. Comparison of segmentation masks for the low vegetation class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

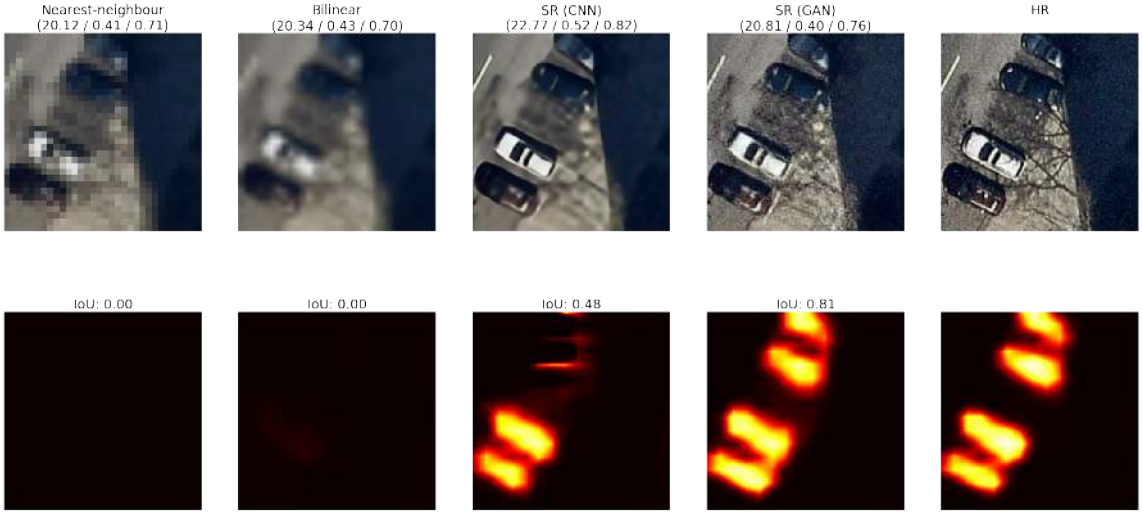


FIGURE A.8. Comparison of segmentation masks for the car class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

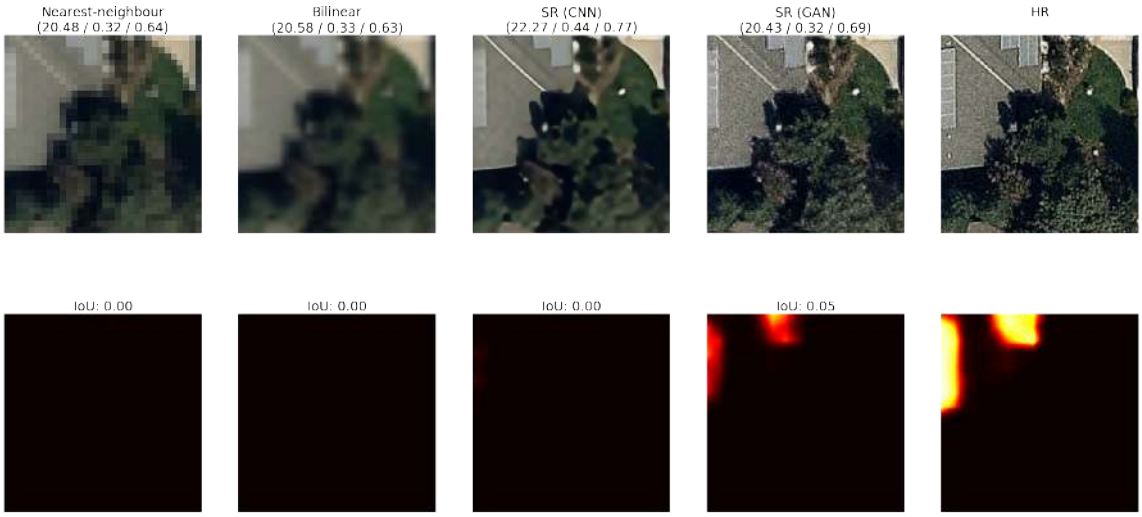


FIGURE A.9. Comparison of segmentation masks for the solar panel class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

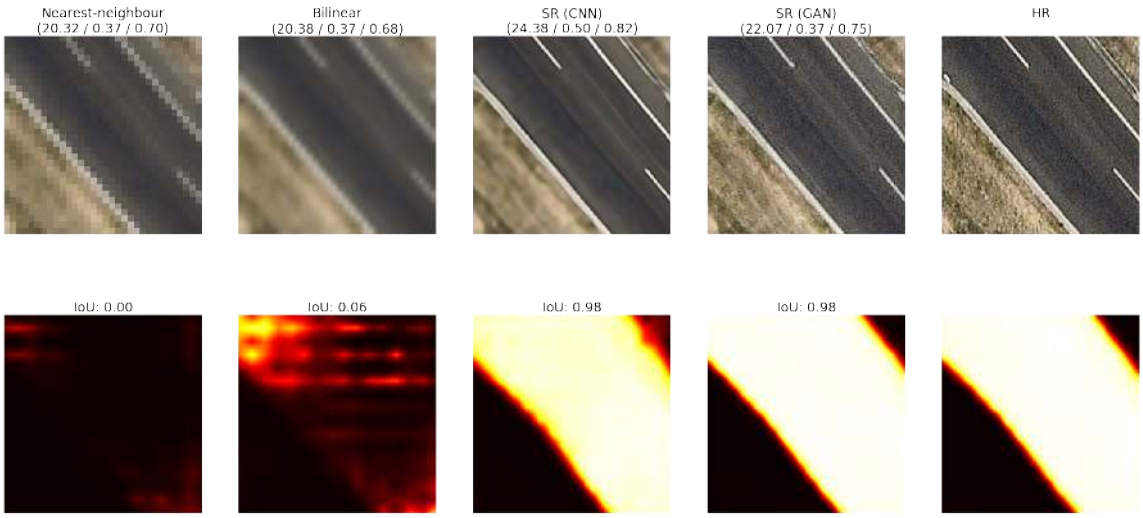


FIGURE A.10. Comparison of segmentation masks for the road class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

A1.1.4 16x Super-resolution

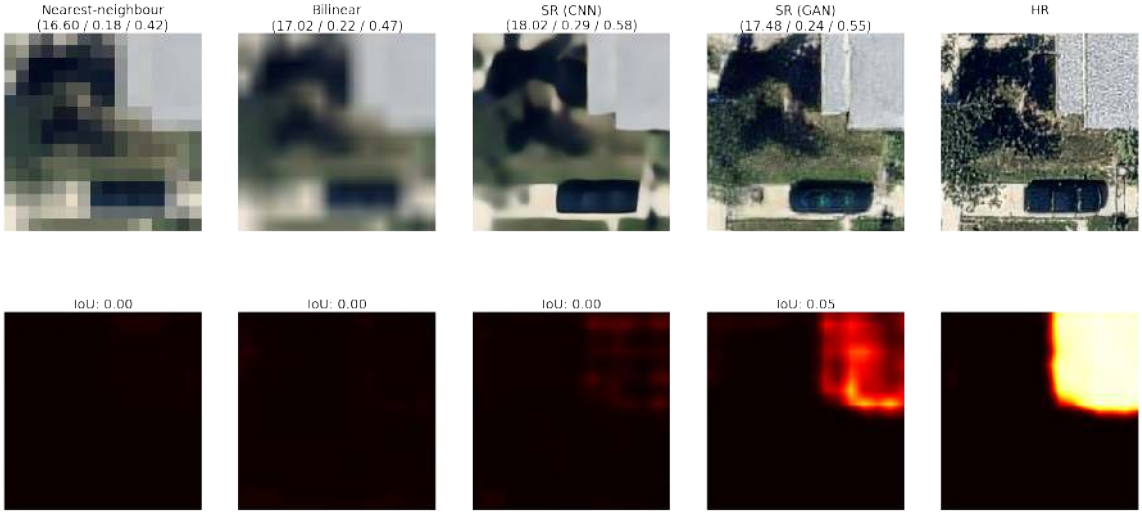


FIGURE A.11. Comparison of segmentation masks for the shingle roof class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

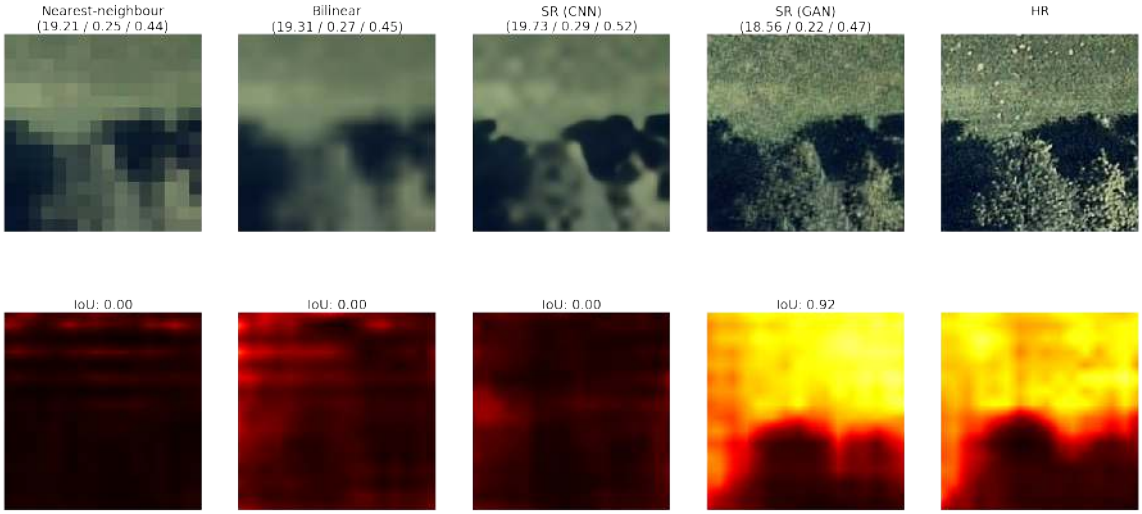


FIGURE A.12. Comparison of segmentation masks for the low vegetation class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

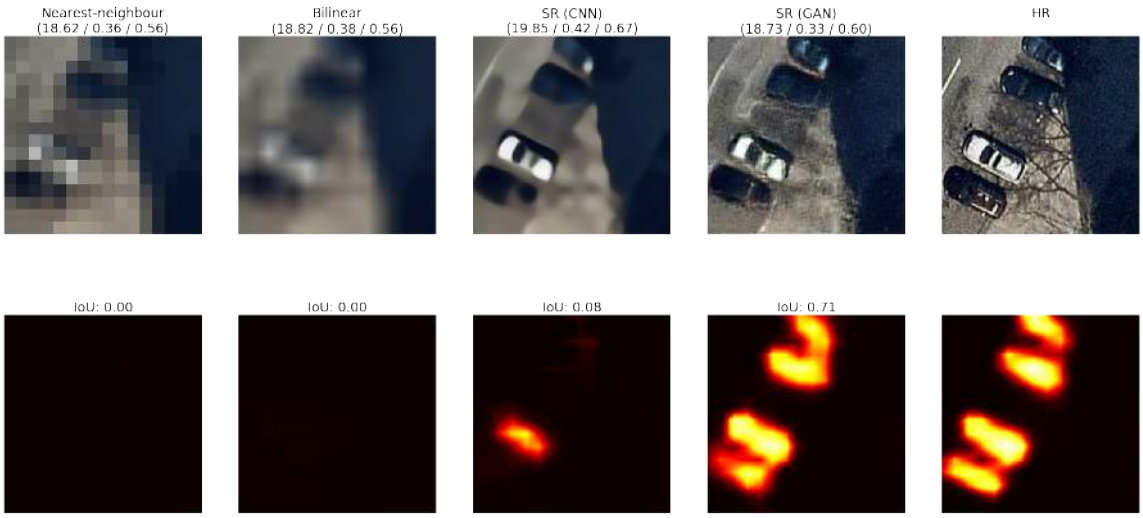


FIGURE A.13. Comparison of segmentation masks for the car class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

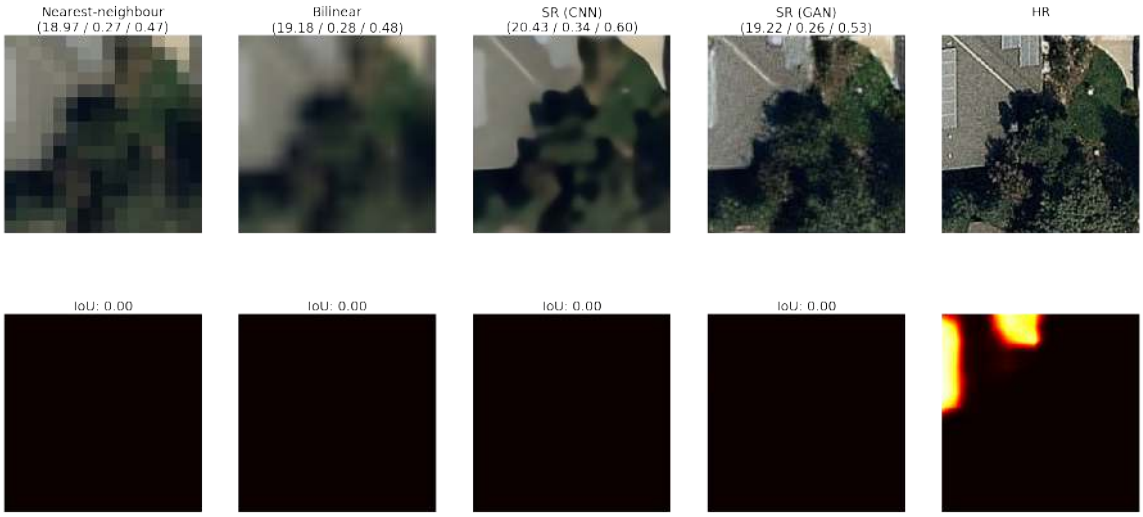


FIGURE A.14. Comparison of segmentation masks for the solar panel class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

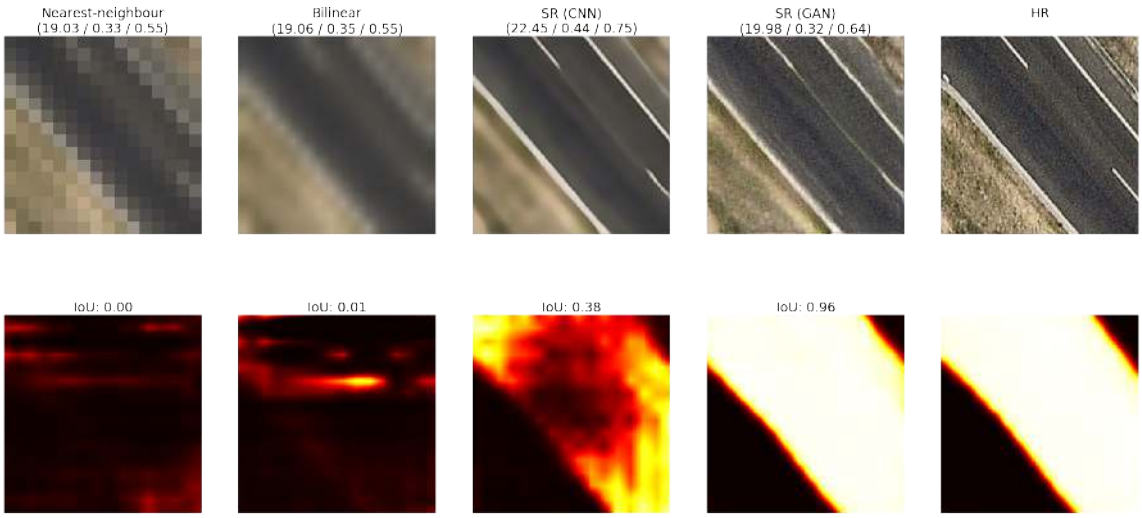


FIGURE A.15. Comparison of segmentation masks for the road class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

A1.1.5 32x Super-resolution

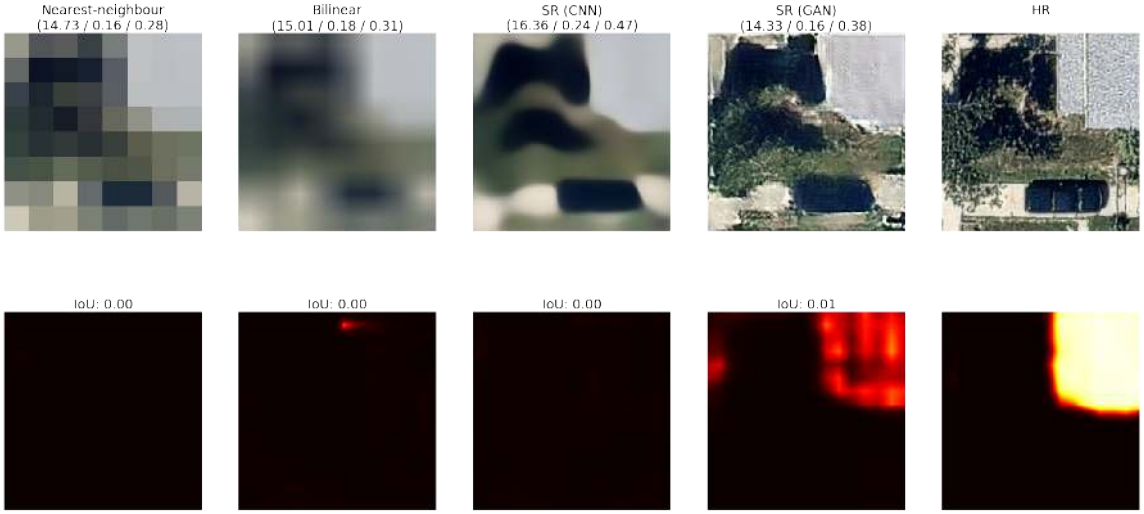


FIGURE A.16. Comparison of segmentation masks for the shingle roof class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

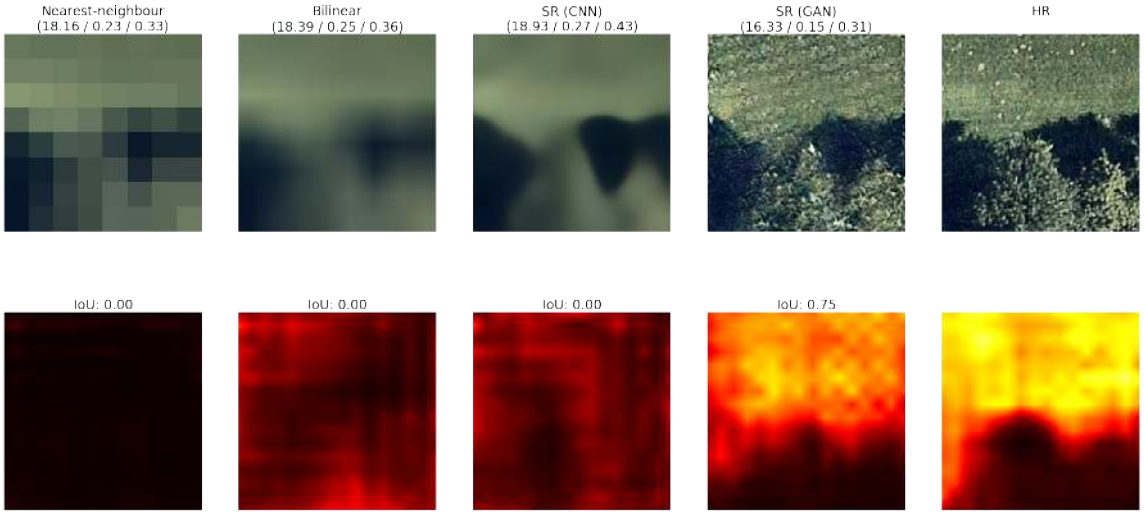


FIGURE A.17. Comparison of segmentation masks for the low vegetation class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

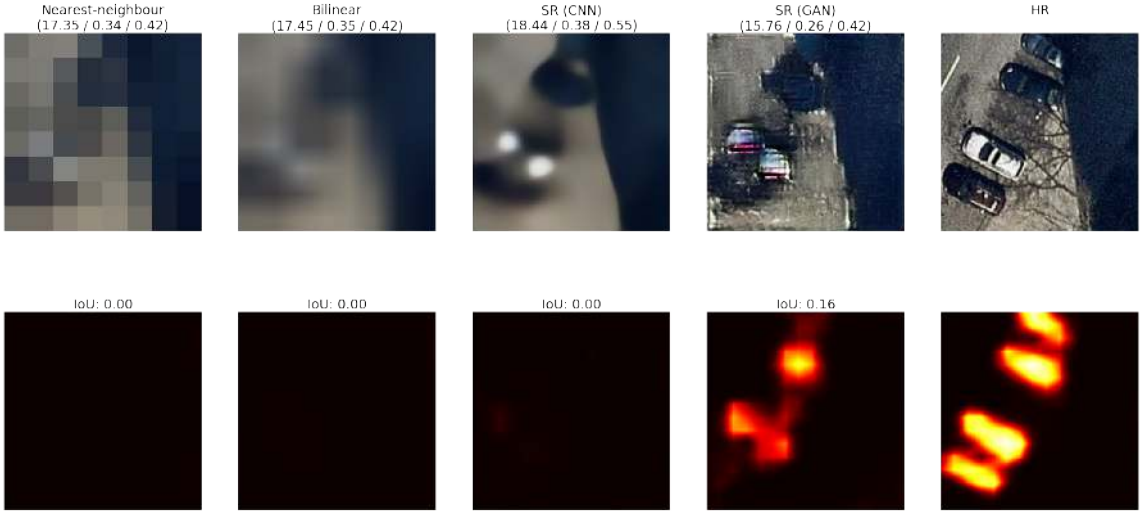


FIGURE A.18. Comparison of segmentation masks for the car class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

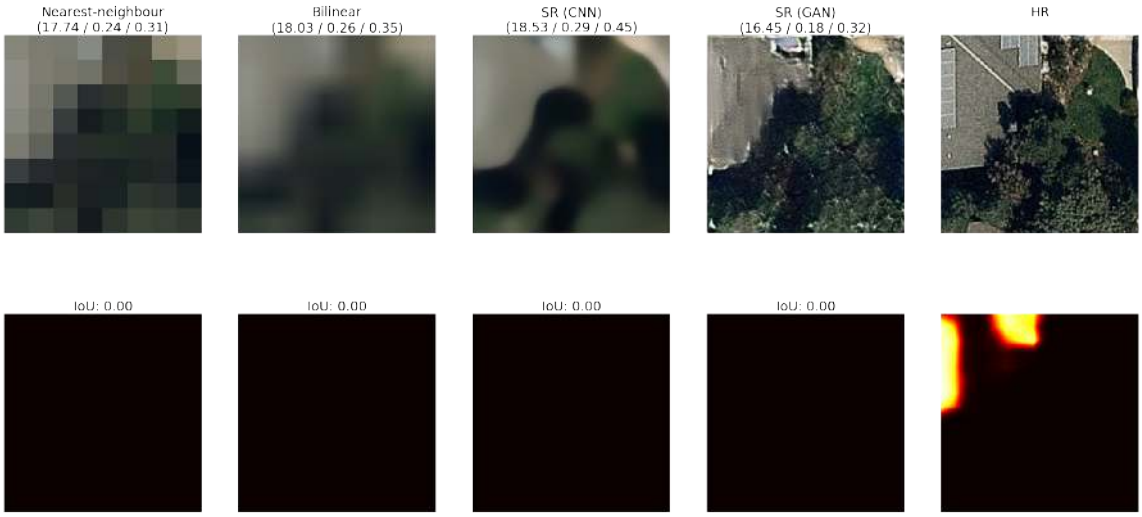


FIGURE A.19. Comparison of segmentation masks for the solar panel class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

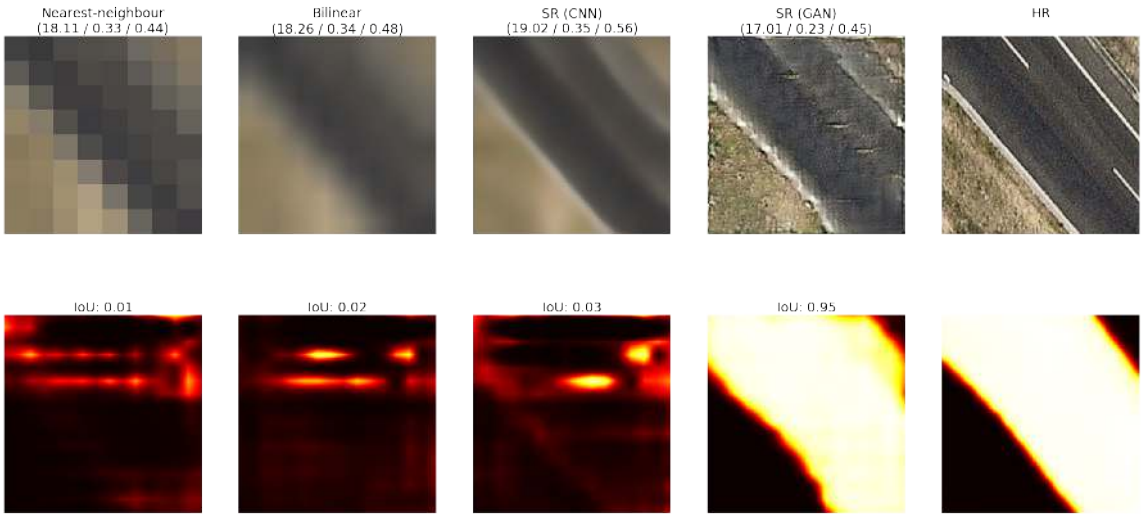


FIGURE A.20. Comparison of segmentation masks for the road class. Bracketed metrics are PSNR, SSIM and MS-SSIM respectively.

A1.2 Segmentation Accuracy IoU Tables

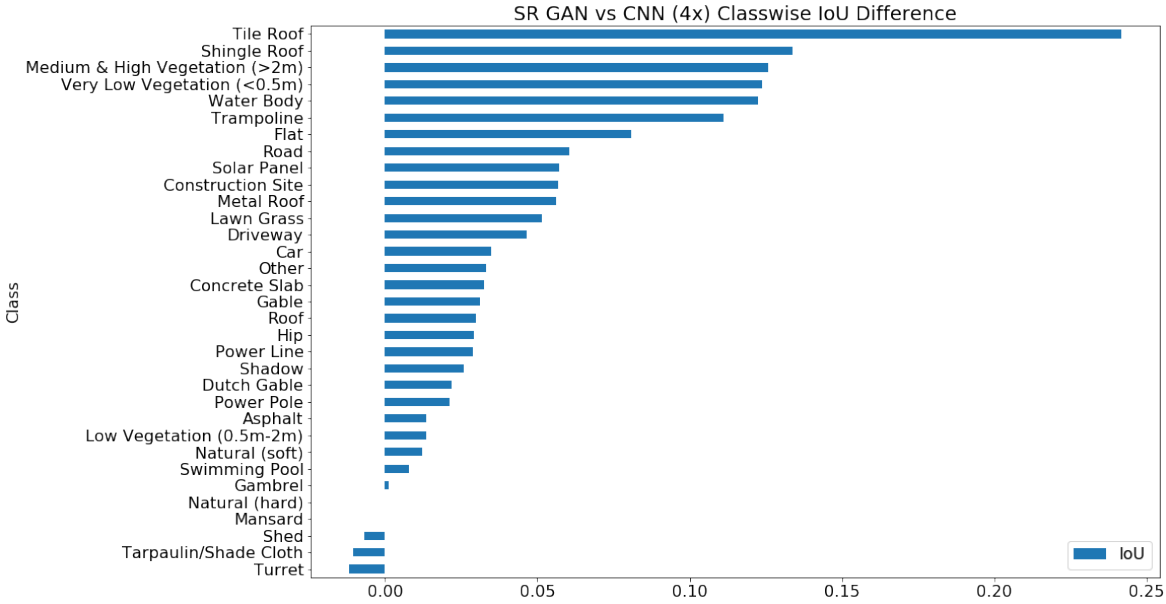


FIGURE A.21. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and plain CNN super-resolved images at a factor of 4x.

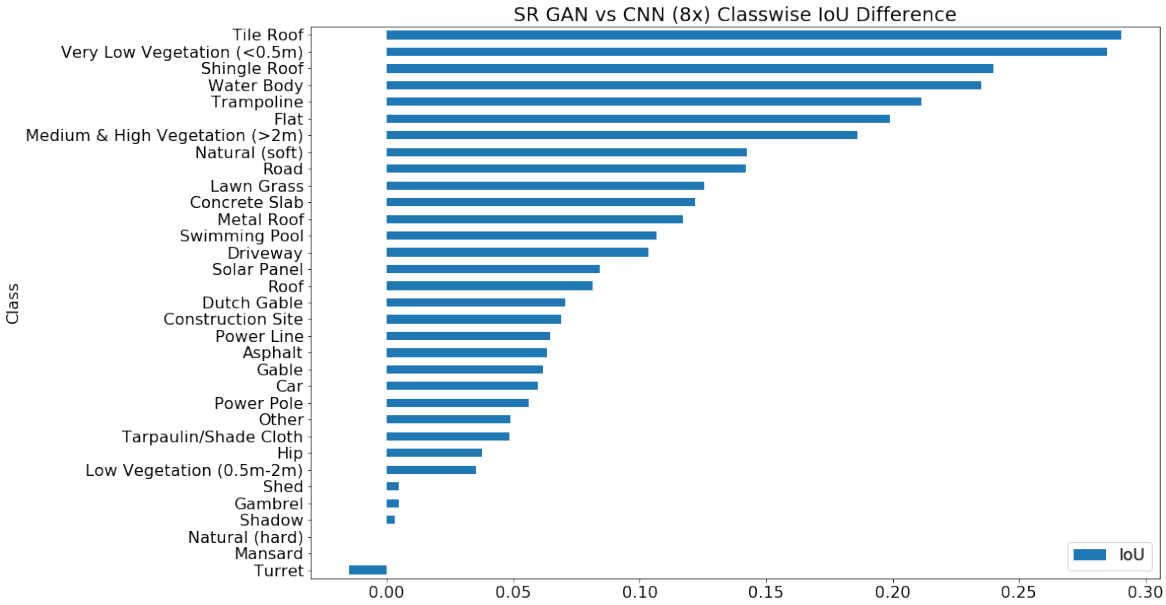


FIGURE A.22. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and plain CNN super-resolved images at a factor of 8x.

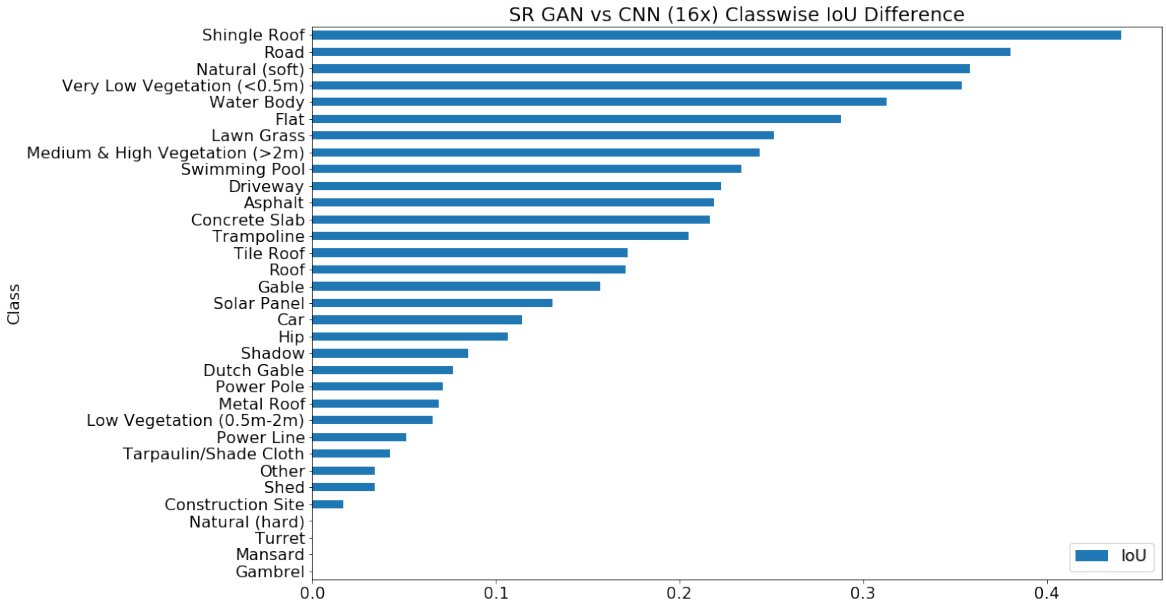


FIGURE A.23. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and plain CNN super-resolved images at a factor of 16x.

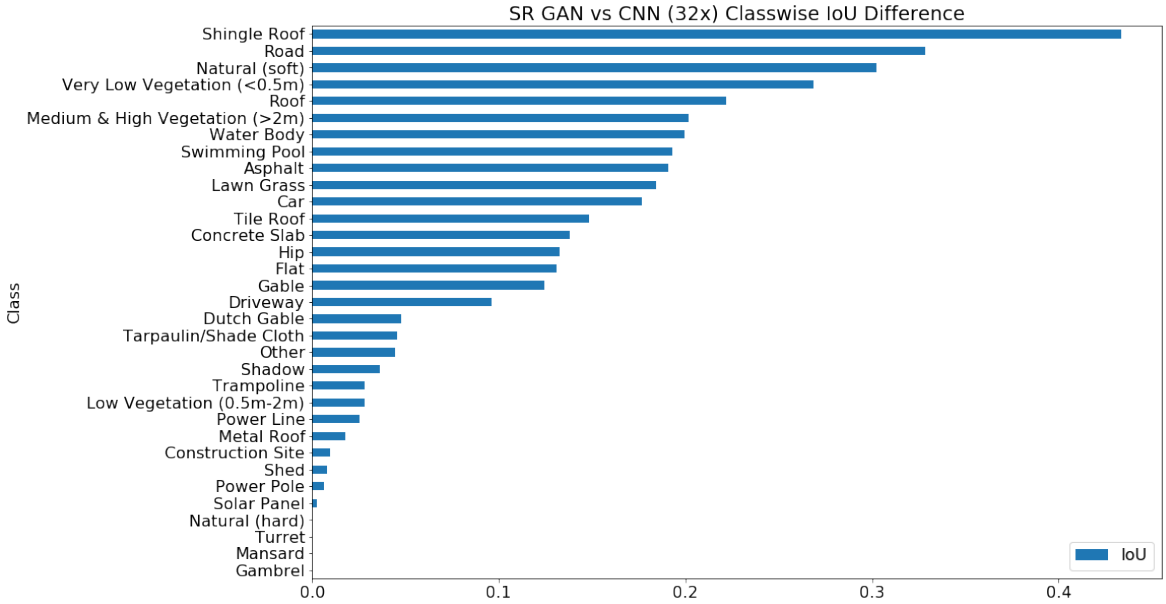


FIGURE A.24. Bar chart depicting the difference in IoU between masks derived from GAN super-resolved and plain CNN super-resolved images at a factor of 32x.

APPENDIX B

Work, Health and Safety Report

In this chapter we outline the Work, Health and Safety (WH&S) considerations and measures taken to minimise them in the context of our industry placement work undertaken at Nearmap.

B1 WH&S Policy at Nearmap

Nearmap is committed to "ensuring that employees and external visitors are provided with a safe and healthy working environment" as stated in their 2019 annual report. Nearmap manages work, health and safety according to its internal policy which is in compliance with NSW legislation (Work Health and Safety Regulation 2017) and federal legislation (Work Health and Safety Act 2011). It's policy includes several notable principles as listed below.

Among many others things, the Executive Management Team (EMT) are responsible for:

(1) Determining Health and Safety Policy for Nearmap's business.
(2) Evaluating all risks in Nearmap's business relating to health and safety at work, loss or damage to Nearmap property and risks to the public through Nearmap's activities.
(3) Ensuring that adequate safety training is carried out as required for Nearmap workers.
(4) Investigating all accidents and damage to property and recommend corrective action where necessary.

Likewise on top of existing commitments to productivity, quality and general supervision, Nearmap managers and supervisors are also responsible for some of the following WHS matters:

(1) Ensuring health and safety is considered in all planning and work activities in which they are involved or of which they have knowledge and that all workers work in a safe manner.
(2) Ensuring that workers are adequately trained and fully aware of any hazards in the workplace.
(3) Ensuring that all workers know the whereabouts of the first aid facilities and the nearest person trained in first aid.
(4) Investigating all accidents to determine their cause and eliminate re-occurrence and report the matter to Health and Safety Representative if appropriate.

Finally a non-exhaustive list of some of the key WHS responsibilities of all Nearmap employees are:

(1) Taking reasonable care for their own health and safety and that of their co-workers – ensuring their actions or omissions do not put others at risk.
(2) Adhering to all safe work practices, instructions and policies.
(3) Ensuring their work area is free of hazards.
(4) Immediately reporting any hazards, unsafe work conditions or equipment to management.
(5) Performing all work duties in a manner, which ensures individual health and safety and that of all other workers and visitors.

It is therefore the responsibility of all employees to be proactive in identifying and avoiding potential hazards and to report them to management, who's duty it is to monitor and put in place the necessary measures to minimise or totally avoid these risks. Incidents that may occur are reviewed by the Human Resources and WHS representatives at Nearmap in line with the company WHS policy and recommendations are acted upon to improve employee health and safety for future scenarios.

B2 WH&S during the ESIPS Placement

The work I undertook during my ESIPS placement was with the Artificial Intelligence (AI) Systems team at Nearmap. Although for the most part the nature of the work, primarily software development, as well as the work environment in the office were fairly low risk, there were still a number of WH&S considerations to be made.

As part of our company induction on our first day we also undertook an office tour and safety information session as part of on-boarding. This covered important WH&S considerations such as:

- Protocol for reporting hazards or incidents to management.
- Professional company code of conduct including Nearmap's company values.
- Fire evacuation procedures including siren signals, fire safety wardens and how to recognise them, emergency exit locations, routes to and locations of evacuation assembly points.
- First aid locations and first aid officers.

Other notable safety precautions that I took note of while working at Nearmap included:

- Maintaining a hygienic office, kitchen and bathroom, all of which were cleaned very regularly.
- Providing fresh food in the kitchen, especially with healthy options available.
- Keeping walking aisles and doorways clear of clutter and tripping hazards.
- Maintaining a well-lit work space to prevent eye-strain and tripping in unlit areas.

During my placement I upheld my responsibility to follow and enact safe and responsible work practices by:

- Being actively aware of potential workplace hazards to both myself and other employees around me
- Reporting such hazards to the relevant authorities.

- Being conscious of the ergonomics of my work station, such as requesting a more ergonomically-friendly keyboard that was happily supplied to me by the IT department.
- Preparing the orientation and height of my chair, desk and computer monitor such that I maintained good posture. Keeping my arms horizontal and level with the surface of the desk helped reduce arm strain, having the monitor at eye-level reduced eye and neck strain and keeping the back of my seat upright with knees at right angles improved my back posture.
- Making sure to take breaks at regular intervals to stretch and gaze away from the screen.
- Maintaining proper hygiene by showing up to work clean and properly dressed as well as making sure to wash my hands regularly during the day.
- Taking sufficient leave or working from home when sick in order to prevent germs from spreading to work colleagues.
- Taking the time to clean up after myself in the kitchen and at my desk, especially cleaning communal appliances such as plates, cutlery and the sandwich toaster.
- Keeping any wires and cables tied up and orderly under and around my desk to prevent tripping accidents or electrical hazards from damaged or exposed wiring.

In the second month of my work placement we also had a fire evacuation procedure enacted across our entire building. Once the alarm had sounded we all were guided by the fire safety wardens in an orderly manner to the emergency exit stairwell and made our way to the evacuation point at Barangaroo Reserve. Once the fire concerns were cleared we were free to return to the office. Towards the end of our placement period we also had ongoing construction in the confines of the office to carry out renovations. However all construction work was safely carried out and lots of barriers and temporary walls were erected to partition away those potentially hazardous areas and keep employees out of harms way.

By being cognizant of these potential hazards and acting upon them preemptively I consequently minimised potential cause for harm and improved safety procedures during my

work placement. Hence by following these steps the general day-to-day risk profile remained low for both me and my colleagues.

B3 Safety and Security

To ensure a safe workplace and work practices it was important to recognise the role of proper security measures, both physical and digital.

Physical security involved:

- Office security personnel on-site.
- Swipe cards to access the elevators and unlock doors to prevent unauthorised entrants.

Digital security was also paramount as Nearmap is very much a software and data-centric business. Measures involved:

- Two-factor authentication (2FA) measures for logging into secure websites and applications such as Git, Confluence and Jira. This greatly reduces the likelihood of unauthorised access to your accounts and logins by requiring a second verification step to a physical device on you.
- Mandatory high-strength passwords to work laptops and account logins. Passwords required at least 10 characters with at least 1 number and 1 non alpha-numeric symbol. Password length is the most important factor in ensuring that your password remains hard to compromise.
- Locking laptops when not in use.
- Using a secure, encrypted VPN service when working remotely. OpenVPN uses SSL/TLS for session authentication and IPSec ESP for secure tunnel transport over UDP.

APPENDIX C

Industry Case Studies

The Engineering Student Industry Placement Scholarship (ESIPS) program involved completing my thesis body of work whilst working in industry over a period of 6 months. From July to December 2019 I worked as a junior research intern at Nearmap Ltd. within the Artificial Intelligence (AI) Systems team under the managerial supervision of Dr. Nagita Mehr Seresht, a senior data scientist on our team. As a consequence of partaking in this work placement opportunity, I picked up many invaluable professional workplace skills as well as an opportunity to deepen my knowledge and experience in computer vision systems, especially with an industry focus. This chapter therefore details how my work at Nearmap adequately covered the breadth and depth of learning outcomes from both the MECH4601 (C1) and AMME4710 (C2) subjects I substituted for this unit of study.

C1 MECH4601: Professional Engineering II

This section goes into detail as to how my ESIPS placement at Nearmap achieved learning outcomes equivalent to those from MECH4601: Professional Engineering II and how the outcomes of this unit translate to an Australian Engineering workplace context.

C1.1 Unit of study outline

The goal of this unit is to give students an awareness of issues and challenges in the management of professional projects, to impart a knowledge of engineering and management practice with a global scope as well as to serve in improving both oral and written communication skills. It also serves the purpose, in conjunction with other subjects in the degree, to fulfil the

requirements of the Institution of Engineers, Australia in regards to teaching management and professional engineering skills. As is outlined in the unit of study document, upon completion of this unit students should be capable of:

- Plan small projects and contribute effectively to planning of larger projects.
- Work effectively in small teams
- understand their role and expected conduct in the management of engineering projects.
- Perform well in that role from the outset, with performance limited only by experience.
- Prepare an interesting and relevant presentation on aspects of their work for their peers or senior managers.
- Recognise the range of expertise they may need to call on in their role as an engineer working on a project (e.g. in project management, safety and environmental fields).
- Understand what the experts are saying, and be able to contribute effectively to that discussion.

During my time at Nearmap I directly gained experience and an understanding of all of the above criteria, allowing me to bootstrap the existing engineering, leadership and communication skills that I had developed during the course of my university career. In particular I worked on planning my overarching thesis project with my manager and head of the team to decide upon timelines and outcomes that would integrate well with the team mission at large. I worked on this project effectively amongst my team of 18 engineers and adhered to the expectations and conduct associated with my role. I worked diligently on the objectives set out in my project timeline and touched base regularly with my superiors to leverage their managerial expertise in helping to guide my research direction and strategising solutions to overcome obstacles and hurdles to my work. In the early stages of my project I presented a talk to the wider team on my research proposal and initial progress, following this up towards the latter stages of my placement to deliver a final company seminar presentation which was both interesting and engaging for the team, directly linking its potential advantages to the goals and purposes of the AI team. Furthermore, I displayed professional conduct in the

workplace and followed important guidelines regarding to work, health and safety. During meetings for both my team and the wider engineering cohort I paid close attention to the knowledge and experience of colleagues who were undoubtedly experts in their respective fields. When the chance presented itself and I had my own wisdom to contribute, I actively sought to participate in such discussions with other employees to drive certain engineering considerations.

C1.2 Learning outcomes

C1.2.1 Professional Effectiveness and Ethical Conduct

Awareness of ethical and other issues which can arise in the workplace.
Before commencing my placement I was made aware of the many responsibilities involved in the role via a multitude of statements and forms communicated to me via email. These related to my contractual obligations as a thesis placement student on the behalf of the university to follow the code of conduct and abide by company rules and regulations in the workplace. First and foremost there was an intellectual property form giving Nearmap the rights to the IP of my work. Intellectual property is a prominent matter of importance for engineering and technology companies as this knowledge often underpins the major value of the business. Furthermore, in my contract there were sections surrounding confidentiality of information and important trade secrets which is similarly important in preventing key business information from spreading outside of the company. I was also made to sign a share trading document which mandated that I was only to sell shares in certain trading windows, or otherwise to seek direct approval from the finance department if trading outside of these allocated times. It was highly important to be aware of this as it has been put in place to comply with bodies such as ASIC and maintain that violations such as trading on inside information, whether intentional or not, are avoided with a wide berth.

In addition to this, all new employees completed three days of compulsory on-boarding, training and education from the various departments in Nearmap such as Human Resources, Legal and Finance. Human Resources covered a lot of the important considerations around

company culture and core values of the business and how to be an ethical employee. This included workplace behaviour, interacting with colleagues and senior staff as well as more niche cases such as acceptable behaviour at work social events. They also covered areas such as acceptable behaviour on social media platforms such as LinkedIn and Facebook and how this reflects on Nearmap. They gave examples of unsatisfactory behaviour and conduct to make it abundantly clear to us the way in which we were expected to conduct ourselves whilst working at Nearmap. The legal team covered a lot of the regulations surrounding the business and legal considerations one might encounter. This for example included the use of the Nearmap logo and how employees should be aware that their conduct outside of work reflects directly on the perception and brand of the business. Finance as mentioned earlier covered a lot about acceptable share trading windows and how to get approval for funding and resources from your manager.

Understanding of what is required in the conduct and management of an engineering project.
In undertaking my 6 month thesis placement, this opportunity gave me a solid understanding of what would be involved in a long running engineering project. By using the company project management software Jira, an industry standard tool, I created my own project (called an "Epic") which contained a list of the various sub-tasks required to achieve the overall project goal. By including rough deadlines for the completion of these sub-tasks this allowed the software to generate a clear and concise roadmap for the completion of my thesis in the form of a Gantt chart. This Gantt chart was also included in both my thesis proposal and later updated in my thesis progress report. As was often the reality, not everything goes quite as planned and timelines I had budgeted took less or often more time than expected, meaning I had to readjust my priorities based on time and team-based constraints and considerations. Major decisions in terms of management of my project would often be run by my manager to validate their feasibility and make sure they still stayed in the expected scope of my project direction.

Ability to recognise the range of expertise you may need to call on in your role as an engineer working on a project (e.g. in the safety and environmental fields).
Throughout my placement it was very evident that my role required me to collaborate and

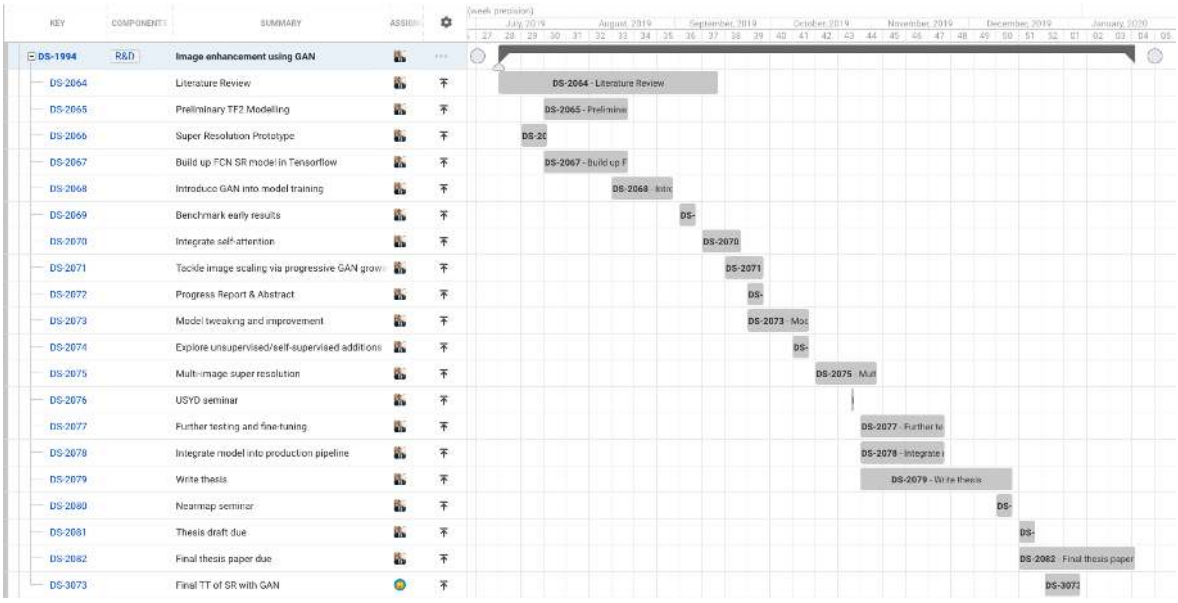


FIGURE C.1. Gantt chart on Jira covering the tasks and timelines across my overall thesis project.

seek the expertise of other members of the team and greater organisation. As a research-intensive and heavily engineering-centric project was the task at hand, I often organise meetings with more senior and experienced members of my team to pick their brain regarding challenges and ideas I was pondering. Many of the senior members in the team had a vast pool of experience in deep learning, computer vision, image processing, software engineering, systems architecture and cloud infrastructure between them. Given my relatively limited knowledge of a number of these areas I found it extremely important to draw on these external sources of knowledge to inform my own research and decision making throughout my project.

In particular, I met with my manager on a weekly cadence for an hour to deep dive into my most current stage of progress. Often times when I felt extremely stuck and seemingly running in circles, she was extremely adept in diagnosing the situation and suggesting insightful and very pragmatic approaches to testing and eventually overcoming these roadblocks. However it was not only my team that I was confined to seeking expertise from. I often reached out to engineers in the vision and sensor systems team to help guide my understanding of complex concepts such as signal processing, image quality metrics and analysis. It was unbelievably useful when I was trying to understand and strategise as to what image quality metrics I would

employ to best encapsulate human visual perceptual as is enhanced by GANs to talk to an engineer across the floor that had written a highly-cited paper on this exact topic.

C1.2.2 Project and Team Skills

Ability to work effectively in a small team to produce a technical report.
Throughout my 6 months at Nearmap I worked effectively within the modestly sized AI team. I collaborated particularly with team members working specifically on the deep learning research and model training front as opposed to the more software engineering and systems side of things. By working closely with the team I was able to get detailed feedback fairly often, allowing me to iterate quickly on my technical thesis. Furthermore by having the chance to occasionally present my ideas to the whole AI team at several weekly team meetings I was able to get really insightful input and comments on my work and suggestions on better or alternative ways to achieve my objectives. This helped significantly in informing the writing of my thesis as it allowed me to include a lot more potential experiments and discussion of ideas that I had not always considered. By writing drafts as my work progressed I also got suggestions at regular intervals from my company manager and academic supervisor on the context and style of delivery for my thesis paper which was very helpful in furthering the technical depth and more well considered discussion of these technical concepts.

To achieve my task of completing my thesis project it was vital to be able to plan and manage my project effectively. Rather than treating it as one massive daunting task, it was critical to segment the work into smaller chunks or sub-projects. Once again I used Jira to plan the timeline for these smaller components and organise my workflow around the major thesis deadlines. By dividing the workload and completing these sub projects one by one, this made the overall project much more manageable and allowed me to budget my time and resources a lot better. Furthermore, upon regular discussions with other members of my team and especially the more senior managers, we discussed in depth the alignment of smaller projects like mine into the grander strategy of the AI team as a whole. By participating in forums such as weekly team meetings this gave me the opportunity to have input into how larger projects across the team were planned and managed. During my last week at Nearmap we also had

a 2020 AI vision day where we spent several hours together as a group talking through the team strategy going into the next calendar year, covering current progress and the roadmap for expected future deliverables and potential deals with customers.

C1.2.3 Communication and Inquiry/ Research

Ability to prepare an interesting presentation on aspects of your work for your peers or senior managers.

During my placement I had the opportunity to present and communicate my work to a larger audience on a number of occasions. Towards the early stages of my time at Nearmap I presented my research proposal and early findings to the AI team I was working in. This was instrumental in bringing my peers up to speed with my work and allowed them to provide me with helpful feedback and wisdom that helped guide the early stages of my thesis ideation.

Later down the track I presented my thesis work at university to an audience of fellow thesis students and a variety of supervisors for thesis seminar day. The presentation was capped at 12 minutes and hence required me to explain a set of complex principles and findings in a short amount of time. I kept my presentation understandable and made a strong effort to keep it intriguing and relevant to the viewers to ensure that everyone remained engaged. Following the presentation I was very pleasantly surprised to heard a lot of kind and supportive feedback on my talk from both friends and random members of the audience alike, who said it was a really interesting topic and that my presentation of it had been fascinating.

My final presentation opportunity occurred during my last week at Nearmap when I gave my company seminar talk to not only my team but a wider audience from the engineering department. This I believe went down well and members of the audience seemed to be interested and enjoy my presentation and its implication for Nearmap's operations. I also received some excellent questions and feedback from senior engineers in the audience which were highly valuable in shaping some of my analysis towards the tail end of my thesis.

Ability to write a concise, technical engineering report.

Writing the thesis paper itself emulated the experience of writing a technical engineering

report. It required me to conduct extensive research and review of the literature, consequently summarising the key aspects and synergising ideas to identify potential gaps in the research field. It was important in such a technical report to elucidate the exact research design and methodology so another engineer would be able to confidently reproduce my findings. Furthermore, alongside this I conducted and presented my own unique experimental findings and discussed their results and significance in relation to the existing literature and to the field of computer vision and aerial imaging more broadly.

The content itself was of a deeply technical nature, delving into the guiding principles of deep learning and taking them to their most cutting edge state in research into deep generative models like GANs. This work not only covered high-level strategy and methodology but also detailed the various low-level implementation and training details that were instrumental in the success of the system as a whole. Despite this being a broad and wide-reaching body of work, covering a lot of areas from computer vision, deep learning, probabilistic generative modelling and general software engineering practices, I made a concerted effort with both my industry and academic supervisors to keep the work relevant and to the point. It was important to sufficiently cover the necessary background and theory to explain the importance of my work without going overboard with the overall scope.

C1.3 Conclusion

In conclusion, I believe my time at Nearmap more than sufficiently covered the key principles of professional engineering in the workplace. It gave me a fantastic opportunity to interact with skilled and diverse engineering teams and management, to communicate my work effectively in both written and oral format and taught me how to plan and contribute to engineering project management. This placement was exceptionally valuable in allowing me to work on real engineering problems in an industry setting, providing me with tangible first-hand experience. Hence, I deem this work to meet the requisite learning outcomes of MECH4601 as demonstrated by the many practical examples and learnings I outlined above.

C2 AMME4710: Computer Vision and Image Processing

This section goes into detail as to how my ESIPS placement at Nearmap achieved learning outcomes equivalent to those from AMME4710: Computer Vision and Image Processing and how the outcomes of this unit translate to an Australian Engineering workplace context.

C2.1 Unit of study outline

This unit of study aims to provide an introduction to many aspects of vision sensors, computer vision analysis and digital image processing. This course is tasked at covering several of the fundamental algorithms involved in computer vision and have exposure to the current state-of-the-art in computer vision research. Students will learn about the inherent difficulties around extracting information and reasoning from image data in an automated fashion. To that end they will therefore learn relevant skills and practical knowledge to come up with engineering solutions to problems involving vision.

In particular this covers the fundamental principles of:

- Image sensor architectures
- Radiometry
- Colour reconstruction
- Projective geometry

They will develop automated approaches to:

- Image analysis
- Segmentation,
- Object recognition,
- Classification
- Image enhancement

And also learn about approaches to:

- Radiometric and projective calibration of camera systems
- Stereo imaging
- Image-based navigation
- The estimation of 3D scene parameters from images

Although most of the practical work in this unit is undertaken in MATLAB, we use Python for the vast majority of the code written during our thesis and there exists quite a lot of similarity in programming paradigms between both languages.

C2.2 Learning outcomes

C2.2.1 Communication and Inquiry/Research

Develop skills in presenting a final design solution to a computer vision/image processing problem.

My overall thesis body of work was an embodiment of this learning outcome as the culmination of my work resulted in me presenting an extensive technical thesis proposing a solution to the challenging computer vision problem of single image super-resolution in the domain of aerial imagery. My thesis consisted of an extensive research proposal and the intuition behind why this problem was both challenging and interesting to solve in vision. It then followed that I conducted an extensive review of the contemporary literature, evaluating traditional techniques for up-sampling photos such as nearest-neighbour and bilinear interpolation as well as more sophisticated approaches such as sparse coding and convolutional neural networks. My work took these findings in literature as a solid grounding for what was commonly used in the real world to achieve image enhancement and super-resolution and proposed an even more cutting-edge solution inspired by much of the image enhancement and super-resolution work involving Generative Adversarial Networks. My final design solution was a product of numerous design iterations and extensive testing to come up with a robust and high-performing neural network model that demonstrated clear promise as a successor to simpler and more commonly used techniques. My methodology and research design detailed in great depth the various design choices and the notable trade-offs in speed, accuracy, memory, complexity,

ease of training and other factors that led to their choice as the final design. My results and testing validated these choices and gave tangible figures to back up the decisions I had made. These results took into consideration the most commonly used metrics for image sharpness and quality, furthermore discussing their apparent drawbacks in failing to fully encapsulate the scope of human visual perception. Finally, this solution was also presented on a number of occasions in oral format at both the university as well as at Nearmap, elaborating the computer vision problem I was aiming to solve and giving important details and knowledge as to how I reached this design solution. This presentation aimed to cover the requisite background knowledge for aerial image processing and image super-resolution whilst also leaving enough time to specifically elaborate on my personal innovations and approach in my body of work.

C2.2.2 Project and Team Skills

Develop skills in working on a design project within a team including communicating with team members, planning and managing tasks.

My project was embedded within Nearmap's AI Systems team and hence required close collaboration with many members of the team to achieve my final outcome of developing a practical and high-performing super-resolution algorithm. Without the many years of experience of the team I strongly believe that my task would have been substantially harder as I gleaned many excellent insights from them. In order to foster this collaboration, effective communication between me and my peers was of utmost importance. Since our entire team worked on the same floor and was situated in the same section of the office, it was easy enough to pop over to anyone you wanted to have a chat to due to the open floor plan. However, the common etiquette was rather to email or direct message the person if it was not an urgent matter as this prevented the annoyance of constant interruption and allowed people to reply in their own time (as many team members were often busy). For digital means of communication we used Outlook for email and calendar management as well as Slack for workplace instant messaging. This turned out to be very handy and easy to use, allowing myself to communicate to co-workers if in need of assistance or advice. If the matter was more complex than what could suffice in a few messages, it was straightforward to reserve one of the many meeting rooms to sit down and talk matters over face to face. The majority

of planning and management was handled via Jira, a project management software service that one could add tasks to and assign them to relevant people. This is primarily what I used to keep track of the progress and tasks to complete in my project. Since some knowledge in the team was also quite universal and oft accessed such as how to set up API keys and documentation on our machine learning models, it didn't make sense to have to bug people for this information all the time. Instead we used Confluence as an internal catalogue of tutorials, documentation and other relevant operational information which could be accessed easily and independently.

C2.2.3 Design

Design an engineering solution to a given image processing task by selecting and evaluating appropriate algorithms for a given image processing task.

In order to tackle our task of enhancing the resolution of aerial imagery we had to evaluate a number of appropriate computer vision algorithms to perform this. We first took the simple approach to using traditional up-sampling techniques in order to resize our images such as simple nearest-neighbour, bilinear and bicubic interpolation. We also investigated more sophisticated variants such as Lanczos up-sampling which instead uses a approximation to the sinc function to produce the theoretically best possible reconstruction of a bandwidth limited signal. However, in the super-resolution of digital images there are more clever approaches than simply trying to perform fancy interpolation and sub-pixel localisation based on the geometry of the image.

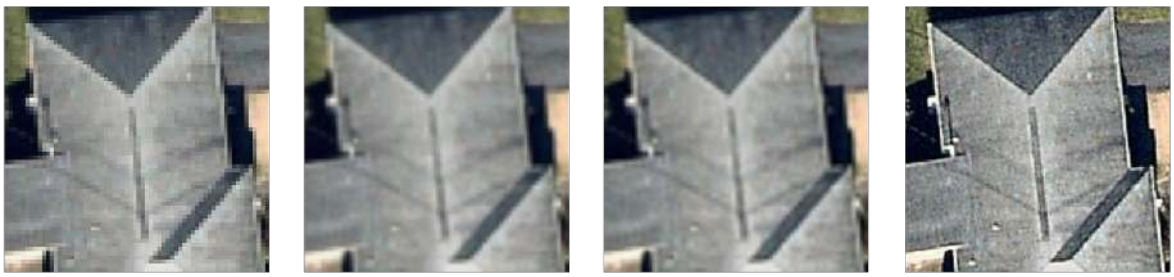


FIGURE C.2. Comparison of tradition image resizing methods. From left to right we have Nearest-neighbour, bilinear, bicubic interpolation and then the original high resolution image.

By leveraging Bayesian inference we can build a model that learns to recognise the features and semantic attributes of aerial images and generate informed predictions as to what the high-resolution image might appear to be when conditioned on the corresponding low-resolution image to be up-sampled. Since super-resolution is considered to be an ill-posed inverse problem whereby there exists a set of possible, non-unique solutions for any low-resolution image, modelling this as probabilistic helped greatly to capture the potential multi-modality in the details of the underlying imagery. As an initial foray into this approach we first looked at sparse coding which learns a dictionary mapping between patches of low and high resolution images. However according to the literature, convolutional neural networks performed this in essence but had a big upper hand in being able to learn deep and complex non-linear mappings (rather than purely linear) as well as being able to learn far more features within the convolutional layers. We henceforth trialled this approach, finding it to perform well at improving general sharpness and refining the shape and structure of objects when minimising a pixel-wise error objective. Unfortunately however, these networks failed to capture the semantic detail that becomes important in recreating the fine features of the image due the this over-simplified loss function.

Therefore, this led us to choose Generative Adversarial Networks as a worthwhile extension to plain convolutional networks to allow them to learn a fully customisable loss function, tailored specifically for gauging image realism. For us this was an important enhancement over prior techniques because it allowed for generated images to align far more strongly with human perceptual evaluation of super-resolved images. Furthermore, being able to use auxiliary feature losses from other deep neural classification or segmentation networks greatly aided in being able to instill our simpler convolutional model with a more sophisticated knowledge of objects in the scene such as houses, trees and cars. This was essential so details like leaves and roof tiles could be estimated based on a Bayesian understanding of the scene semantics rather than simply trying to refine geometric properties of the image.

FIGURE C.3. Comparison of different image up-sampling algorithms. From left to right we have CNN super-resolution using L1 minimisation, GAN super-resolution algorithm and the high-resolution ground truth. It is abundantly clear that the CNN is a significant improvement over the standard methods, however the GAN absolutely excels at producing the most semantically accurate images.

C2.2.4 Engineering/ IT Specialisation

To develop an understanding of the fundamental principles of how images are formed including the basics of image sensors, radiometry, colour and projective geometry.

Towards the start of my placement I was given a run down by the head of the Sensor Systems team at Nearmap as to how we performed image capture during our aerial surveys. We use an array of commercially available digital single lens reflex (DSLR) cameras to capture vertical and oblique imagery concurrently at a regular cadence. These cameras all use a complementary metal-oxide-semiconductor (CMOS) sensor. Every pixel on this sensor has an associated photo-diode that detects light and one or more active MOSFET transistors that are used to amplify the signal voltage of the photo-diode and reduce noise. The intensity of light for a pixel is hence understood from the voltage produced at each photodiode. Colour is typically captured by using an array of red, green and blue (RGB) colour filters arranged on the sensor in a Bayer mosaic pattern. Each of these filters is sensitive to their respective wavelengths of visible light and hence each of these RGB filters can capture their individual intensities of light for that colour. These colour signals are then typically then run through an demosaicing algorithm which can interpolate the full array of RGB image data.

These cameras are calibrated precisely by the manufacturer, however since we fly these systems in excess of 10,000ft altitude this introduces a plethora of new challenges to accurate capture of images. First, the fact that we fly so high means that extremely long focal lengths

upwards of 300mm must be used to ensure an adequate ground sampling distance (GSD) which is the distance between pixel centers in the physical world. By operating at such a distance, any vibrations or motion blur from the aircraft have a greatly amplified effect at extended radius. Furthermore, one has to account for the extreme environmental conditions flying at such an altitude as the sharp drop in temperature causes the glass of the lens to distort and hence calibrations need to be made for these conditions rather than for room temperature.

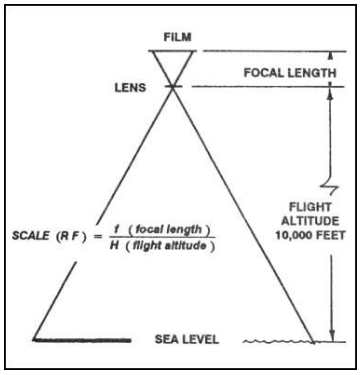


FIGURE C.4. Diagram illustrating the capture of vertical imagery and how the scale of features is calculated.

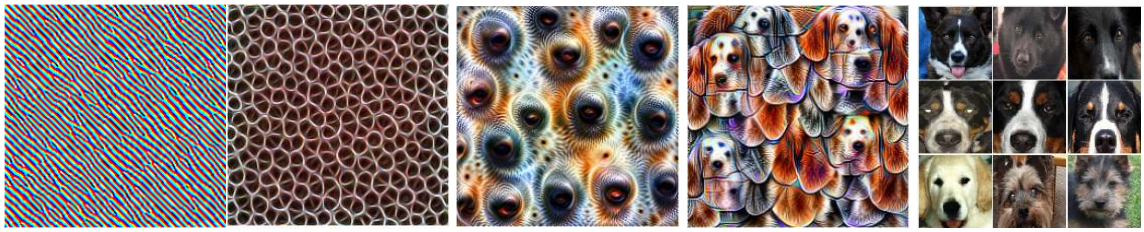
Projective geometry is also a vital consideration in aerial photogrammetry as we must consider how to accurately capture a view of the earth on a planar 2D image. As real life features on the ground are projected onto the sensor we have to take into account our focal length and altitude of flight to accurately calculate the real-life scale of features in our imagery. This is crucial to the business as one of our main selling points is being able to accurately measure the heights of buildings and areas of regions on the ground.

Apply basic techniques in image processing including the use of image filtering, features, edge detection, colour spaces/transforms and matching.

In the Nearmap image processing pipeline, raw images also undergo a number of filtering and post-processing steps. Often due to atmospheric haze and interference the images tend to come out a lot more flat and with a poorer dynamic range. By boosting the contrast it becomes easier to make out important features and edges. It is also important to be wary of losing information due to blowouts in the highlights of an image, hence images are often captured on the safe side of underexposed since the blacks can be boosted. This is especially prevalent

when flying over areas of water and catching nasty reflections of the sun off the water which blow out the image. To stitch together discrete images into vast ortho-maps, key features are detected and correlated between neighbouring images in order to find points on the images to align. By solving large sets of matrix equations to satisfy all these feature constraints, the images can be stitched together and aligned to form one enormous, continuous map of an entire city.

More specifically in my own work I was required to take these images and perform a resolution enhancement operation to super-resolve them. For the down-sampling operation it was important to add an anti-aliasing filter to the down-sampled image otherwise the result would have high-frequency artifacts as a result of the under-sampling of the image signal. In my image pipeline I also performed a channel-wise normalisation of the images to even out the image statistics, a trick that is common to aid machine learning models. Since the majority of my deep learning algorithm was based upon the use of convolutional kernels, this was the workhorse in automated feature learning in my aerial imagery. Rather than hand craft features or perform tedious template matching, the CNN builds up its own internal representation of features in its many filters. In a typical convolutional network, the layers loosely mimic the hierarchical layering of the visual cortex whereby early layers act primarily as edge detectors, abstracting these representations all the way up to high-level pattern and full object representations.



(A) Edge (conv2d0) (B) Texture (mixed3a) (C) Part (mixed4b/c) (D) Object (mixed4d/e) (E) Dataset Examples

FIGURE C.5. A visualisation of the hierarchy of features learned in GoogLeNet which has been trained on ImageNet. The network progressively builds up its understanding of images over deeper and deeper layer abstractions (layer numbers in parentheses) [79].

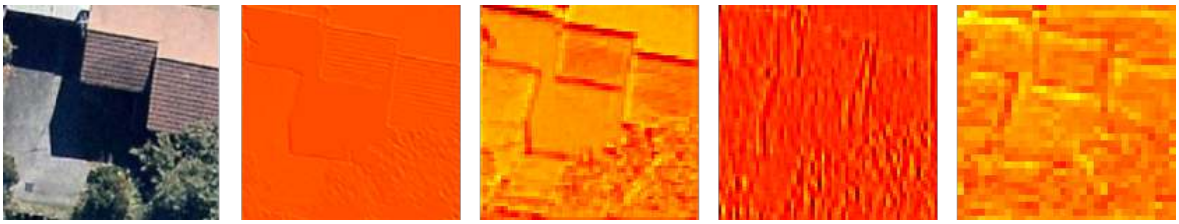


FIGURE C.6. Figure showing the RGB image on the left and some of the activations of the semantic segmentation network we used in the first four layers. We can see that these early layers focus on edge detection and image structure, getting less local and more general as the network deepens.

Apply advanced techniques in computer vision including stereo vision, 3D mapping, object detection, image classification and use of machine learning algorithms in vision.

At Nearmap we cover a lot of bases from regular photogrammetry and construction of orthoimagery maps to building 3D models of cities. 3D models are constructed using imagery captured at obliques angles, whereby images at different perspectives can be projected and correlated with one another to form a stereo observation of the scene and build a 3D model. However I did not directly contribute to this vein of work. My work centered primarily around image enhancement and semantic segmentation of the aerial imagery by using machine learning (more specifically deep learning). The primary business case of the AI team in which I worked is to design and deploy deep learning models that perform semantic segmentation on our imagery. Semantic segmentation is essentially the combination of object detection, localisation and classification. This is because the algorithm learns to classify on a pixel by pixel basis, allowing it to generate class-wise semantic masks of important features in the image. We might use this to detect and segment important features in aerial imagery such as roofs, vegetation, roads, cars, solar panels and so on.

My thesis body of work directly specialised in leveraging deep learning for single image super-resolution. I used convolutional neural network models to learn the mapping from low to high resolution images by first using a simple L1 image loss. It was evident from these experiments that neural network approaches were very promising as they allowed a rich set of complex features to be learned and as a result they achieved impressive results for super-resolution. I then extended this work to incorporate Generative Adversarial Networks,

another more recent style of neural network, to act as a perceptual loss and help the model learn convincing and semantically relevant features of the aerial imagery it was trained on.

Apply a wide range of image processing techniques to real world applications.
My entire time at Nearmap was focused around real-world applications, in particular for my case being able to take promising research areas of deep learning and apply them reliably in a production context. The semantic segmentation model we used serves a vast number of exciting business and real world use cases. For example, by detecting and segmenting for important categories of image features such as solar panels and pools we could detect the existence of almost every one of these in an area the size of Sydney or Chicago. This knowledge would be hugely valuable to other businesses such as solar panel sellers and installation businesses as they can instantly get an idea of their rough market size for any given geographical area. We also used this model extensively to predict for roof material such as shingle, metal and tile which could be an important source of information for businesses that service roofs or insurance companies to evaluate roof condition and potential damages. Another interesting use of this model was for change detection, whereby segmentation masks could be compared across different points in time to detect changes in the content of the scene, e.g. new construction and buildings being built. Specifically for my thesis project, the super-resolution infrastructure that I built was designed with production use in mind. I was conscious of inference times and avoiding large model sizes so it could easily be accessed behind a straightforward API call. Since it was directly used by Nearmap, this was solving the real world problem of how to intelligently enhance the resolution of their imagery and optimise these images for their segmentation masks.

Understand the type of algorithm required for a particular image processing task.
As elaborated on in previous sections, we performed extensive analysis of the literature to obtain a general idea as to what algorithms suited what tasks and their general ranking in performance. We compared traditional interpolation methods to using sparse coding and a more Bayesian approach via deep learning. It was quintessential to understand that convolutional neural networks offered a great advantage over regular fully-connected neural networks as they employed convolutional kernels to learn spatial relationships and image

feature hierarchies. Convolutional networks have been one of the breakthrough architectures in accelerating the performance of object classification, image segmentation and literally all other classes of computer vision tasks using deep learning. We then recognise the inherent strength of Generative Adversarial Networks to extend this CNN model, allowing us to model highly sophisticated elements of aerial imagery, aligning with human perception.

C2.3 Conclusion

In conclusion, my placement at Nearmap exposed me to a great number of computer vision principles and their usage in a highly practical setting. It was fascinating to see them be implemented and operate at the scale of surveying entire continents worth of imagery with world-class results. I was amazed not only by the hardware engineered to capture imagery at high altitude with challenging environmental conditions such as temperature and motion blur, but even more so the image post-processing and deep learning algorithms run over the top of such imagery to stitch together vast maps and extract semantic elements of the scene. In my own thesis work I specialised entirely in bringing the state of the art in deep learning such as convolutional neural networks and generative adversarial networks to help perform automated image enhancement, in particular focusing on the task of super-resolving aerial imagery. This work was both challenging and interesting, allowing me to dive deep into contemporary computer vision research as well as get hands on with practical implementation of such algorithms in a production setting. Hence, I believe that my thesis and case study cover the required knowledge and experience expected from the completion of AMME4710: Computer Vision and Image Processing.

APPENDIX D

Practical Experience Logbook

This section of the appendix gives a week by week breakdown of the nature of the work conducted during my ESIPS placement and is required for the completion of the ENG4000 practical experience module.

Week 1 (8/7/19-12/7/19)

Nearmap company onboarding. Meeting the member of the AI team and brainstorming thesis directions and applications.

Week 2 (15/7/19-19/7/19)

Getting familiar with the Nearmap business, their Mapbrowser product as well as the general codebase for my team. Configuring workspace with Docker and Jupyter as well as setting up remote ssh on GPU machines.

Week 3 (22/7/19-26/7/19)

Extensive literature research to get up to speed with deep learning and GAN state of the art approaches. Reading into style and domain transfer and begin experimenting with open source implementations.

Week 4 (29/7/19-2/8/19)

Decide definitively on addressing super-resolution as my primary focus after consulting with supervisors. Begin writing code for prototype model. Official new starter induction week, receive talks from all different departments in the business.

Week 5 (5/8/19-9/8/19)

Build out a simple SRResNet baseline model inspired by SRGAN, paying close attention to architecture choices in the up-sampling layers in order to avoid checkerboard artifacts. Spend time setting up Tensorflow 2.0 and configuring GPU drivers.

Week 6 (12/8/19-16/8/19)

Deliver preliminary informal team presentation on my proposed work and progress thus far, receive ample feedback from work colleagues. Finish writing code for the first iteration of the super-resolution model.

Week 7 (19/8/19-23/8/19)

Identified notable issues with generalisation, in particular severe model overfitting due to dataset issues. Expand the dataset to be not only larger but also diversely sampled. Begin experimenting with GAN models for image synthesis by using DCGAN.

Week 8 (26/8/19-30/8/19)

Addressing difficulties with DCGAN training, namely poor training stability characteristics and a tendency for the discriminator to overfit. Spend this time reading into training strategies in the literature and implementing them.

Week 9 (2/9/19-6/9/19)

Finish writing introduction and begin writing literature review. In the meantime, begin extending DCGAN model to a more stable multi-scale MSG-GAN architecture.

Week 10 (9/9/19-13/9/19)

Spend time optimising my data pipeline such that images can be loaded fast and in a highly parallelisable way. Use this to speed up training times by almost 10x which helps me prototype much faster.

Week 11 (16/9/19-20/9/19)

Encounter memory leak issues with new multi-threaded pipeline, spend several days bugfixing these issues. Continue writing literature review

Week 12 (23/9/19-27/9/19)

Continue to train more GAN and CNN models for super-resolution, trying to identify optimal hyper-parameters. Start on model evaluation and use this for writing the preliminary methodology results sections.

Week 13 (30/9/19-4/10/19)

Test GAN model on common datasets like CelebA to compare relative performance. Experiment with different up-sampling layer methods as well as exploring the effect of initialisation, namely trying Xavier, Unifrom, He and Orthogonal initialisation schemes.

Week 14 (7/10/19-11/10/19)

Manage to finally produce GAN aerial image synthesis results that appear quite stable and realistic to a human observer. Take this as a cue to switch back focus to the task of super-resolution and begin applying this GAN knowledge to my older CNN model.

Week 15 (14/10/19-18/10/19)

Begin to integrate feature loss into my super-resolution model. Learn how to use the Nearmap segmentation model and make some tweaks to allow it to integrate into my existing model.

Week 16 (21/10/19-25/10/19)

Spend time preparing for and eventually presenting at USyd thesis seminar day. Creating powerpoint slides and giving small internal talks to prepare.

Week 17 (28/10/19-1/11/19)

Back to training our SRGAN model, achieving very promising results with the new feature loss component and importantly starting to become able to learn texture detail into the model.

Start to look into EDSR generator as an alternative to SRResNet, find that it is much faster and accurate.

Week 18 (4/11/19-8/11/19)

Begin to expand super-resolution tasks from 4x to the harder 8x up-sampling which proves to be challenging. In the meantime start researching into full reference image quality metrics, both classical PSNR, SSIM, MS-SSIM as well as perceptual variants like VMAF.

Week 19 (11/11/19-15/11/19)

Testing the effect of extracting feature losses at different layers of the network and the results that produces. More tweaking of the GAN models such as pre-training, weight update schedules, regularisation and so forth to aid training stability.

Week 20 (18/11/19-22/11/19)

Focus on improving the discriminator side of the network. Change the network architecture to employ a patch-based GAN and fiddle more with the nature of the loss functions and their relative weightings to produce results that are most perceptually pleasing.

Week 21 (25/11/19-29/11/19)

Continue to scale up the difficulty of the super-resolution task, this time looking at achieving 16 and 32x super-resolution. Rewrite a large portion of my codebase and GAN super-resolution framework to make it more keras-like and production ready.

Week 22 (2/12/19-6/12/19)

Move my focus towards using super-resolution as a means for improving semantic segmentation mask IoU. Spend this time getting familiar with how Nearmap validates their segmentation models and use this as a basis for creating my own validation script to compute performance metrics across large swathes of the available data.

Week 23 (9/12/19-13/12/19)

Continue to optimise model hyper-parameters to achieve the best possible results we can. Particularly focus on getting the model to work well at high super-resolution factors such as 32x and collect results of this throughout for use in the thesis.

Week 24 (16/12/19-20/12/19)

Concentrate on wrapping up results and preparing models and codebase to hand over to the team. Write extensive documentation and set up a private repository for my work.

Week 25 (23/12/19-27/12/19)

Spent time preparing official Nearmap company presentation. Participate in the AI team 2020 vision day meeting. Finalise details of my project and give my manager a full run-down on my codebase and how to use the models.

Bibliography

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," 6 2014.

[2] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," 2018.

[3] L.-C. Chen, G. Papandreou, S. Member, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," tech. rep.

[4] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning Aerial Image Segmentation from Online Maps," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 11, pp. 6054–6068, 2017.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," tech. rep., 1996.

[6] K. Pearson, " LIII. On lines and planes of closest fit to systems of points in space ," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, pp. 559–572, 11 1901.

[7] L. Van Der Maaten and G. Hinton, "Visualizing Data using t-SNE," tech. rep., 2008.

[8] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," 2 2018.

[9] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," 11 2014.

[10] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 11 1958.

[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[12] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, pp. 462–466, 9 1952.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 9 2014.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," tech. rep.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," tech. rep.

[16] H. Chu, X. Liao, P. Dong, Z. Chen, X. Zhao, and J. Zou, "An Automatic Classification Method of Well Testing Plot Based on Convolutional Neural Network (CNN)," *Energies*, vol. 12, p. 2846, 7 2019.

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations," tech. rep.

[18] A. Y. Ng and M. I. Jordan, "On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes," tech. rep.

[19] V. Vapnik, *The nature of statistical learning theory*. Springer science \& business media, 2013.

[20] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," 12 2013.

[21] A. Dosovitskiy and T. Brox, "Generating Images with Perceptual Similarity Metrics based on Deep Networks," tech. rep.

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 105–114, 2017.

[23] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11133 LNCS, pp. 63–79, 9 2019.

[24] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," 6 2019.

[25] V. Badrinarayanan, A. Kendall, and R. Cipolla, "A2_segnet," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, 2017.

[26] N. Audebert, A. Boulch, H. Randrianarivo, B. Le Saux, M. Ferecatu, S. Lefevre, and R. Marlet, "Deep learning for urban remote sensing," in *2017 Joint Urban Remote Sensing Event, JURSE 2017*, Institute of Electrical and Electronics Engineers Inc., 5 2017.

[27] B. Huang, B. Zhao, and Y. Song, "Urban land-use mapping using a deep convolutional neural network with high spatial resolution multispectral remote sensing imagery," *Remote Sensing of Environment*, vol. 214, pp. 73–86, 9 2018.

[28] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," 11 2015.

[29] S. Ioffe and C. Szegedy, "Batch Normalization (InceptionV2)," 2015.

[30] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," 9 2018.

[31] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," 2018.

[32] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," 2 2018.

[33] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, "Deep Learning for Single Image Super-Resolution: A Brief Review," *IEEE Transactions on Multimedia*, pp. 1–1, 2019.

[34] J. Yang, S. Member, J. Wright, I. Thomas Huang, L. Fellow, Y. Ma, and S. Member, "Image Super-Resolution via Sparse Representation," tech. rep.

[35] D. C., L. C., H. K., and T. X., "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 295–307, 12 2016.

[36] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 1646–1654, 2016.

[37] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014.

[38] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, pp. 1132–1140, 2017.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, IEEE Computer Society, 12 2016.

[40] R. Timofte Eirikur, A. Luc, V. Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, X. Wang, Y. Tian, K. Yu, Y. Zhang, S. Wu, C. Dong, L. Lin, Y. Q. Chen, C. Loy, W. Bae, J. Yoo, Y. Han, J. C. Ye, J.-S. Choi, M. Kim, Y. Fan, J. Yu, W. Han, D. Liu, H. Yu, Z. Wang, H. Shi, X. Wang, T. S. Huang, Y. Chen, K. Zhang, W. Zuo, Z. Tang, L. Luo, S. Li, M. Fu, L. Cao, W. Heng, G. Bui, T. Le, Y. Duan, D. Tao, R. Wang, X. Lin, J. Pang, J. Xu, Y. Zhao, X. Xu, J. Pan, D. Sun, Y. Zhang, X. Song, Y. Dai, X. Qin, X.-P. Huynh, T. Guo, H. Seyed, M. Tiep, H. Vu, V. Monga, C. Cruz, K. Egiazarian, V. Katkovnik, R. Mehta, A. Kumar, J. Abhinav, A. Ch, S. Praveen, R. Zhou, H. Wen, C. Zhu, Z. Xia, Z. Wang, and Q. Guo, "NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results," tech. rep.

[41] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and Checkerboard Artifacts," *Distill*, 2016.

[42] V. Dumoulin, F. Visin, and G. E. P. Box, "A guide to convolution arithmetic for deep learning," tech. rep., 2018.

[43] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 1874–1883, 9 2016.

[44] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize," 2017.

[45] Y. Sugawara, S. Shiota, and H. Kiya, "Super-Resolution Using Convolutional Neural Networks Without Any Checkerboard Artifacts," in *Proceedings - International Conference on Image Processing, ICIP*, pp. 66–70, 2018.

[46] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," 3 2016.

[47] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual Dense Network for Image Super-Resolution," 2 2018.

[48] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, Institute of Electrical and Electronics Engineers Inc., 11 2017.

[49] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard GAN," 7 2018.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Transformer: Attention Is All You Need," *Behavioral and Brain Sciences*, no. Nips, pp. 1–101, 2017.

[51] A. Odena, J. Buckman, C. Olsson, T. B. Brown, C. Olah, C. Raffel, and I. Goodfellow, "Is Generator Conditioning Causally Related to GAN Performance?," 2 2018.

[52] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," 10 2017.

[53] A. Karnewar, O. Wang, and R. S. Iyengar, "MSG-GAN: Multi-Scale Gradient GAN for Stable Image Synthesis," 3 2019.

[54] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," 2017.

[55] S. Chintala, "How to Train a GAN," in *NIPS*, (Barcelona), 2016.

[56] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.

[57] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 12 2014.

[58] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the Variance of the Adaptive Learning Rate and Beyond," 8 2019.

[59] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, "Amortised MAP Inference for Image Super-resolution," 10 2016.

[60] F. Huszár and C. K. Sønderby, "Instance Noise: A trick for stabilising GAN training," 2016.

[61] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based Generative Adversarial Network," 9 2016.

[62] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5967–5976, 2017.

[63] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems*, pp. 2234–2242, 6 2016.

[64] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the Inception Architecture for Computer Vision," tech. rep.

[65] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," 6 2017.

[66] F. Cardinale, Z. John, and D. Tran, "ISR," 2018.

[67] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment," *Electronics Letters*, vol. 44, no. 13, pp. 800–801, 2008.

[68] Q. Huynh-Thu and M. Ghanbari, "The accuracy of PSNR in predicting video quality for different video scenes and frame rates," *Telecommunication Systems*, vol. 49, pp. 35–48, 1 2012.

[69] Z. Wang, A. C. Bovik, H. Rahim Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 13, no. 4, 2004.

[70] Y. Lecun, L. Eon Bottou, Y. Bengio, and P. H. Abstract|, "Gradient-Based Learning Applied to Document Recognition," tech. rep.

[71] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs Created Equal? A Large-Scale Study," 2017.

[72] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 1 2018.

[73] L. A. Gatys, A. S. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," 8 2015.

[74] Z. Wang, J. Chen, and S. C. H. Hoi, "Deep Learning for Image Super-resolution: A Survey," 2019.

[75] C. Ma, C.-Y. Yang, X. Yang, M.-H. Yang, Y. Yang, and M.-H. Yang, "Learning a No-Reference Quality Metric for Single-Image Super-Resolution," tech. rep., 2016.

[76] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a 'completely blind' image quality analyzer," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, 2013.

[77] M. Orduna, C. Díaz, L. Muñoz, P. Pérez, I. Benito, and N. García, "Video Multimethod Assessment Fusion (VMAF) on 360VR contents," 1 2019.

[78] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10265 LNCS, pp. 146–147, Springer Verlag, 2017.

[79] C. Olah, A. Mordvintsev, and L. Schubert, "Feature Visualization," *Distill*, vol. 2, 11 2017.