Super-Resolution of Satellite Imagery

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Masters of Science (Research)

at

The University of Waikato

by

Daniel Bull

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2021

Abstract

Can deep neural networks super-resolve satellite imagery to a high perceptual quality? This thesis explores the juxtaposition between the pixel accuracy and perceptual qualities of super-resolved imagery by comparing and combining a discriminative and a generative network. Rather than solving a theoretical problem, we tackle a real-world low-resolution scenario: Sentinel-2 imagery is super-resolved and evaluated against high-resolution aerial photos as ground truth; this is in contrast to super-resolving previously down-sampled data, which is the methodology used in most other studies. An existing feed-forward network architecture designed for super-resolution, called DeepSUM, is used to super-resolve multiple low-resolution images by a factor of four to obtain a single high-resolution image. DeepSUM is trained using a range of loss functions, to assess the effect on network accuracy. A novel loss function is created, called *variation loss*, to help better define edges and textures to create a sharper, perceptually better product. Using an SSIM (Structural Similarity Index) loss function gives the best result in terms of pixel-based performance. Running DeepSUM alone creates a superior output compared to bicubically up-sampling the input data, but the output is blurry and not photo-realistic. A probabilistic model from the literature, ESRGAN (Enhanced SRGAN), a Generative Adversarial Network, is trained against both raw Sentinel-2 data and the output of DeepSUM. Using ERSGAN for super-resolution, creates a perceptually better, more realistic looking output. However, the ESRGAN output is less accurate than the DeepSUM output, as measured using pixel-based metrics. Combining ESRGAN with DeepSUM is found to inherit some of the advantages of both approaches. In an end-to-end process, using ESRGAN with the output of DeepSUM trained using variation loss is found to super-resolve an image to better show boundaries, textures and detail.

Acknowledgements

The author thanks my supervisors Nick Lim and Eibe Frank for their amazing help.

Contents

Chapter 1

Introduction

1.1 Overview

Just as alchemists in the Middle Ages came up with more and more outlandish formulae for transmutation of base minerals into gold, so too toil data scientists in a quest to extract and enhance imagery from lower-quality dross. Super-resolution, by creating detail in images whence none existed before, is as its name implies, something of a mystical art, yet is ever more likely to be achieved in a practical and useful sense, given modern processor speeds and deep neural networks. Similar to gold in the ancient world, the value of super-resolution in the modern world is immense because of a plethora of potential applications. Unlike the transmutation of gold, it is not based on an elixir but a series of ever improving techniques, each built upon the previous iteration.

Super-resolution is the process whereby a higher-resolution image is obtained from a single or multiple lower-resolution image. This process is used in a wide variety of applications including medical imaging, imaging for security, and satellite remote sensing [Yang et al., 2019]. Video imaging super-resolution is an important and growing area, where multiple overlapping images obtained from video frames together provide the ability to create an enhanced picture. Modern cameras including the Pixel 3 from Google utilise this technique [Wired Magazine, 2019]. Higher-resolution images offer the benefit

of more detail which can be critical.

This study focuses on using super-resolution in remote sensing. The problem that super-resolution is trying to solve in this context is the issue of relatively low resolution of widely-available remote sensing imagery such as satellite imagery from Sentinel 2 satellites. Most areas of New Zealand are available as high-resolution aerial photography with a resolution of around 0.1m pixels or lower. However, due to the cost of capturing and processing the data, imagery is typically procured on a multi-year cycle so that available high-resolution data may be 4 years out of date. Aerial photography suffers from another issue in that it typically is taken during the summer months, during the middle of the day when shadows are minimal and cloud cover is reduced, and is not generally available over winter months. Typically, aerial photographs have three visual bands (such as RGB) and an NIR band.

Satellite imagery is generally cheap and has a high temporal return time, e.g., Sentinel 2 data is available weekly [European Space Agency, 2013], Planet data is available daily [Planet Labs Inc, 2021]. Satellite data often has multiple bands in the non-visual spectrum as well as RGB and NIR bands, but is limited in its usefulness due to its lower resolution. Typically, satellite data is used to identify large areas of landcover, but is less useful at identifying individual trees. For example, in remote sensing, a 10m pixel image will show forested areas, whereas a 1m pixel image will allow individual trees to be distinguished. As it is costly and sometimes impractical to improve the spatial resolution of images using enhanced hardware sensors, there is significant interest in software approaches to create high-resolution data from a lower resolution input.

This study investigates two super-resolution techniques using different approaches to the problem. DeepSUM is a feed-forward neural network which was the winner of the European Space Agency (ESA) PROBA-V Super Resolution competition winner in 2020 [Molini et al., 2019]. The approach used by this method is a more traditional deep neural network approach based on

discriminative training that attempts to minimise mean-squared error (MSE) in the predictions. In the present study, various iterations of this approach are investigated to replace the default loss MSE function with a function better suited to modelling super-resolution data. In contrast, *ESRGAN* is an advanced generative adversarial network (GAN), which super-resolves imagery to create a perceptually better output. This generative approach models the image data distribution and uses that to super-resolve. These approaches have contrasting strengths and weaknesses around accuracy and perceptual quality that are explored and analysed in this study. We will see that a combination of the two processes to some degree is able to use the strengths of each.

Rather than use a purpose-designed dataset such as used by the original DeepSUM paper [Molini et al., 2019], this study uses readily available satellite data and pre-processes the data to enable it to work in the algorithm. Unlike most other super-resolution studies, aerial imagery data is used as a ground truth, i.e., the high-resolution data and low-resolution data are from different sources.

Code used in this project is available from these repositories:

- https://github.com/danbull1/superresolution
- https://github.com/danbull1/deepsum
- https://github.com/danbull1/esrgan-tf2

This thesis contains the following chapters: Chapter 2 reviews previous work in this field, and looks at different loss functions and performance metrics used in super-resolution. Chapter 3 outlines the data and methodology used in this study. Results are presented in Chapter 4, and summarised and discussed in Chapter 5. Chapter 6 concludes the thesis and summarises the findings of this study.

Chapter 2

Literature Review

Super-resolution is an inherently ill-posed problem as a multiplicity of solutions exist for any given low-resolution pixel [Dong et al., 2015]. To overcome this issue, many complex super-resolution methodologies exist that attempt to exploit contextual information to infer missing high-resolution components. It should be recognised that while the majority of super-resolution studies focus on generic photo imagery, there is a substantial body of work focusing on super-resolving satellite imagery [Molini et al., 2019] which has its own unique issues and advantages over other imagery data (see Section 2.4.1). Looking beyond still imagery, studies such as [Xiao et al., 2020], [Irani and Peleg, 1991] focus on improving video quality by using multiple image frames together. Many studies address super-resolution in the wider context of image processing using techniques such as de-noising, compressing and imprinting imagery.

2.1 Background

Before the advent of modern machine learning, interpolation-based methods such as bicubic interpolation and Lanczos resampling were used [Yang et al., 2019]. Bicubic interpolation estimates the value of each pixel based on the surrounding pixels. Lanzcos resampling uses a low-pass filter to smoothly interpolate the value of a digital signal. Although these methods are fast and do not require any example data to work from, they suffer from low accuracy due to an

inherent lack of information about how to resolve the pixel values. This tends to blur high-frequency detail [Sun et al., 2008].

A generic category of super-resolution consists of so-called reconstruction based methods (also referred to as sparse coding methods). These involve dissecting different parts of an image into constituent parts before reassembling the image at a higher resolution. First, patches are taken from low-resolution (LR) images and preprocessed to normalise. The patches are sparsely coded and to create a dictionary that matches the patches from LR to high-resolution (HR) images [Zhang et al., 2020]. Various improvements to this basic idea have been proposed, for example, the use of a *gradient profile prior* or a parametric prior describing the shape and sharpness of the image gradients learned from a number of images [Sun et al., 2008]. It is noted by [Dong et al., 2015] that the process by which sparsely LR patches are matched to HR patches can be viewed as a convolutional neural network (CNN). However, the sparse coding solver is iterative rather than feed-forward, so is less computationally efficient.

In general, reconstruction based methods generate more defined detail than the interpolation-based methods, but suffer from disadvantages in that they require prior knowledge to restrict the possible solution space, and are usually time-consuming to run. The performance of reconstruction based methods degrades rapidly as the scale factor increase reducing their usefulness for larger increases in resolution[Yang et al., 2019].

Machine learning based methods including those using deep learning are the current focus of most research in super-resolution due to their relatively fast computation, and better performance than the above methods [Yang et al., 2019]. These methods learn mapping functions from LR to HR imagery. Note that several machine learning-based super-resolution methods exists that do not utilize deep learning. In [Li et al., 2019], a random-forest based algorithm is used to learn a dictionary mapping LR patches to HR patches in a method called feature augmented random forest (FARF). This method uses the dot

product of flattened feature vectors to differentiate feature patches when clustering them in the split-nodes of the random-forest. Several enhancements to this basic method are added, including adding gradient magnitudes to enhance the edge strength and create better image definition [Li et al., 2019].

2.2 Feed-Forward Networks

With the advent of deep convolutional neural networks (CNN), came the ability to learn pixel values by exploiting very high-level feature maps. These super-resolution methods can be divided in to single-image SR (SISR) and multi-image SR (MISR). As the names suggest, SISR infers HR data from a single image whereas MISR takes advantage of the information gain presented by multiple complimentary images of the same scene to better infer pixel values [Molini et al., 2019].

In one of the first SISR methods using deep CNN [Dong et al., 2015] implicitly learned patches via hidden layers, rather than explicitly learning a dictionary. In the model, called Super-Resolution Convolutional Neural Network (SRCNN), an LR image is first upscale to the desired size using bicubic interpolation. This image was then flattened and passed through three filters to generate an output that proved better than any of the previous SIRS methods at that time. Some of the learnings from this approach are that a wider network generates a better result, and that having a larger initial filter or convolution is advantageous. In both cases, performance is affected, and the authors suggest that there is a trade-off to be made. They also explore the idea that a deeper neural network with more layers may perform better. Although this turns out to be the case, the effect was not as strong as that for image classification. In general, the experimental results show that deeper networks, even when relatively shallow by modern standards, often did not converge, to a solution yielding high accuracy [Dong et al., 2015].

[Kim et al., 2016a] proposed a deep network based on increasing neural net-

work depth to significantly boost performance, inspired by the VGG network. Stacking many more filters, leads to filters that responded to a larger region of pixel space, resulting in an increased amount of contextual information that can be exploited to infer high-frequency detail. In [Kim et al., 2016a], rather than predict the ground truth (HR image), a residual image (r) is defined so that $r = y - x$ where y is the predicted image and x is the output image. The loss function used is the Euclidean difference between the reconstructed image (the sum of network input and output) and ground truth. The authors report that using a network that learns the residual image, converges much faster than simply learning the standard non-residual network, and shows superior performance at convergence in terms of a higher peak signal-to-noise ratio (PSNR) value obtained by the network. It is noteworthy that, additionally, gradient clipping is used to improve performance of the network. Gradient clipping is a process whereby gradients are confined to a certain range during network training in order to attain maximal speed of convergence. Finally, it is found that training the network using data taken at different scales, outperforms a network trained using a single scale even when tested against data from a single scale [Kim et al., 2016a].

Building on this work, [Kim et al., 2016b] proposed a method using a so-called recursive neural networks for super-resolution. The authors argue that it is not possible to get a performant network by simply stacking more layers on top of a network such as SRCNN, as this requires more data and makes the network prone to over-fitting. In their new method, called Deeply-Recursive Convolutional Network (DRCN), the network depth is increased by applying the same convolution 16 times to significantly boost performance. In order to overcome the problem of exploding or vanishing gradients, the network is modified to supervise all recursions. A second innovation proposed by the authors is to add skip connections, which are used to feed input data into the final stage of the network, the reconstruction stage. This serves to reduce network capacity required to store the input signal during recursions, and means that

an exact copy of input signal can be used during target prediction. The DRCN network is found to out-perform existing networks on super-resolving photos in [Kim et al., 2016b].

Several MISR techniques have been proposed in the context of video editing. Traditional video super-resolution (VSR) techniques take multiple LR frames, and output HR frames by taking in to account sub-pixel motions between neighbouring frames [Jo et al., 2018]. [Irani and Peleg, 1991] use the explicit horizontal, vertical, and rotational shifts to create a set of equations defining an image with respect to another image containing correction factors in a process called iterative back-projection. Registration of the second image with respect to the first image occurs by iteratively warping the second image towards the first image to gradually decrease the correction factors until they approach zero. Accurate knowledge of the relative displacements of scene regions is essential for this process. Super-resolution is carried out on down-sampled HR video frames in an iterative update process using a back-projection kernel. In this way each HR pixel is influenced by several of the surrounding LR pixels, in a fashion that is not too dissimilar to the process occurring in a modern CNN. Super-resolution is applied on colour images by converting the image to a YIQ representation. A YIQ representation is an alternative representation of colours where Y, is luminance, and IQ represent chrominance. Following the conversion, an average is taken of each of the chrominance components after registration, and the above technique is used on the IQ component of the image. The resulting super-resolved images are observed to be superior to the average of input images [Irani and Peleg, 1991].

In the two-step process described in [Irani and Peleg, 1991], the results rely heavily on the motion estimation and compensation step. This can lead to blurry frames. In the Dynamic Upsampling Filters (DUF) method, outlined in [Jo et al., 2018], motion information is implicitly utilised to generate dynamic filters. Rather than using a deep neural network to reconstruct an HR image in the feature space, the deep neural network learns the best upsampling

filters, which are then used to directly reconstruct HR frames from given LR frames. The authors compare the DUF process to other VSR methods, and find that their method was vastly superior to any comparable technique. The learnt dynamic filters show different activations for different motions indicating that they correctly exploit temporal information without explicit motion compensation. In general, the DUF method produced sharper images with better PSNR values [Jo et al., 2018].

Other video image super-resolution techniques concentrate on creating super-sampling algorithms tailored for real-time rendering. As video games get higher and higher resolution, and more photo-realistic effects, modern game engines often reduce the computational cost by rendering at a lower resolution, and up-sampling to the native resolution [Xiao et al., 2020]. There is an inherent problem in up-sampling this type of video in that point-sampled pixels are extremely aliased at edges and shadows, and the information at the target pixels is missing. The solution implemented in [Xiao et al., 2020] is a neural network that utilises information from previous frames to super-resolve the current frame. The network contains four main parts. A *Feature Extraction* module contains a 3-layer CNN that processes each frame individually and shares weights between frames. The output of the CNN is concatenated with the input data to get a 12-channel output. A *Temporal Reprojection* module projects the output of the previous module onto the current frame using the known motion vectors and warping the frames to fit the current frame. A *Feature Reweighing* module, masks out the aliasing by generating a pixel-wise weighting map for each previous frame. Finally, a *Reconstruction* module takes as input, the features of the previous module and outputs an HR image of the current frame. The loss function used in this network is a weighted sum of perceptual loss (see Section 2.6) and structural similarity index (SSIM). The authors find that their method significantly outperforms previous work including real-time temporal anti-aliasing, and other state-of-the-art image and video super-resolution techniques [Xiao et al., 2020].

2.3 Generative Models

While super-resolution studies initially used feed-forward networks, trained in a discriminative manner, using standard $\ell_1$and ℓ_2 loss functions, more recent approaches integrate perceptual loss and adversarial loss functions. These types of approach have been found to yield sharper images with better perceptual qualities [Lugmayr et al., 2020]. Another set of approaches to super-resolution is based on training generative models. Generative models are a class of algorithm that learns the probability distribution of data. Examples of these models include Generative Adversarial Networks (GANs), Variational Autoencoders, and Normalising Flow models. Learning the distribution of an image in image space, allows a network to generate photo-realistic images by finding natural HR images that correspond to the distribution in the LR space [Wang et al., 2018].

2.3.1 Generative Adversarial Networks

Several recent studies have looked at using Generative Adversarial Networks (GANs) to enhance the production of super-resolved data. These include [Hoque et al., 2019], [Jiang et al., 2019], [Ledig et al., 2017] and [Wang et al., 2018]. GANs employ adversarial training by using a Generator network and a Discriminator network. During training, the generator network strives to produce an HR image similar to the target HR image, whereas the discriminator acts as a judge and tries to find out whether the input is a fake or not. In this way, the discriminator guides the generator to produce steadily more realistic images, until the output image looks real. The discriminator network learns whether a sample is from the model distribution or the data distribution, and so pushes the generator to produce more photo-realistic images [Goodfellow et al., 2020].

GANs have been described as implicitly modelling the data distribution, as they do not directly select from distribution models, but learn to generate images from this distribution as a consequence of the adversarial model

[Liu et al., 2021]. In the context of super-resolution, this will likely produce an image that is not exactly the same as the ground truth, but has perceptual qualities that appear the same to the Human Visual System (HVS). GANs offer the possibility of photo-realistic images at large up-scaling factors [Ledig et al., 2017]. On the other hand, some of the fine detail created by the GAN may be visually pleasing, but may also be inconsistent with the ground truth [Jiang et al., 2019]. This effect is also commented on by [Blau and Michaeli, 2018] who described the trade-off between creating a high perceptual quality image, and an image free from any distortion (see Section 2.7).

While GANs can successfully generate photo-realistic images, their tendency to *dream* or hallucinate details can be a weakness particularly in the field of super-resolution where the ideal outcome is consistent with the ground truth. This "dreaming" occurs because their loss function is unsupervised and as long as the output fits in to the probability distribution, then the discriminator is satisfied [Goodfellow et al., 2020].

2.3.2 SRGAN

The first GAN to be used in the context of super-resolution is the Super-Resolution Generative Adversarial Network (SRGAN), created by [Ledig et al., 2017], which super-resolves photos by a factor of four. The generator part of the network consists of several blocks that use a deep residual network (ResNet)[He et al., 2016] with skip-connections. In residual neural network blocks (or ResNet) blocks, skip connections help bring the identity function to deeper layers. This means that network performance does not degrade with a deeper network.

In SRGAN, the generator part of the network consists of a number of residual blocks, each containing two convolutional layers with small 3×3 kernels and 64 feature maps followed by batch-normalization layers. The discriminator part of the network contains eight convolutional layers with an increasing number of 3 × 3 filter kernels; more specifically, they repeatedly increase by

a factor of 2 from 64 to 512. As in a VGG [Simonyan and Zisserman, 2014], strided convolutions are used to reduce the image resolution. LeakyReLU is used as the activation function through the network and max-pooling is avoided throughout. In the network, up-sampling through a dense layer, occurs after the ResNet blocks, as a final step.

SRGAN used a hybrid loss function consisting of a perceptual loss implemented as a pixel based MSE loss, and a VGG-based content loss function as described in 2.6. To this an adversarial loss function is added that encourages the network to create an image that looks real or is in the 'manifold of natural images' [Ledig et al., 2017]. Different VGG versions are considered for the perceptual loss function, and the VGG54 model is found to perform best, probably because it is the deepest network and can better capture fine detail. The authors report that deeper layers in the VGG network for the perceptual loss yield the most convincing results and speculate that using deeper layers allows the loss function to focus on content, leaving the adversarial aspect of the loss to focus on fine-level details. In comparison, when the network is trained with an MSE loss combined with an adversarial loss it achieves a higher PSNR value but creates an image that is perceptually too smooth and less convincing than the results achieved with a loss component more sensitive to visual perception [Ledig et al., 2017].

Unlike other super-resolution algorithms and GANs, SRGAN is not optimised for yielding perceptually realistic results.

2.3.3 Enhanced SRGAN

The Enhanced SRGAN (ESRGAN) is developed by [Wang et al., 2018] enhances SRGAN to achieve more realistic features and textures than the original GAN. The basic architecture of SRGAN carries out most of the computation in the LR feature space, before data is upsampled to HR. The stated aim of ESRGAN is to enhance visual quality of the output image in both sharpness and detail. This enhancement is achieved by changing several main compo-

nents of SRGAN. All batch normalisation is removed from the generator part of the network because it was found empirically that it tended to introduce artefacts and limit the generalisation ability of the network, especially when the statistics of the training and test datasets varied considerably. This had the added advantage of reducing memory usage because it makes training more stable. The second structural change made to the generator was to replace the basic residual blocks with a novel basic block called Residual-in-Residual-Dense-Block (RRDB), where residual learning is used at different levels in a way that is deeper and more complex than the SRGAN residual block. In the RRDB block all layers within the residual block are connected directly with each other. In ESRGAN, as in SRGAN, multi-level ResNet blocks are used, however the ResNet blocks used in ESRGAN use dense blocks as the main path so the network benefits from these dense connections. In both networks, a residual scaling parameter allows the network to be easily deepened by increasing the number of ResNet blocks.

The discriminator of ESRGAN, which estimates the probability that an image is real, is improved from the standard discriminator used by SRGAN. The enhanced discriminator is a "relativistic" discriminator that tries to predict the probability that a real image is relatively more realistic than a fake one. This takes advantage of a factor that the probability of real data being real decreases as the probability of fake data being real increases. If the discriminator is fed samples where half the data is fake, and half is real, it can use this knowledge that finding several fake samples increases the probability that the rest of the samples are real. Relativistic discriminators have been shown to be significantly more stable and generate higher quality data samples than their non-relativistic counterparts [Jolicoeur-Martineau, 2018]. Ablation studies in [Wang et al., 2018] show that the use of a relativistic discriminator is a key component in the improvement in performance of ESRGAN over SRGAN.

ESRGAN uses a perceptual loss function, as per SRGAN, but this is applied to features observed before activation rather than after activation. This is

found to improve performance as activated features are very sparse, which provides weak activation, and inconsistent brightness compared with ground truth.

The generator part of the network is first trained using a *PSNR-oriented network* using an ℓ_1 loss. This meant that undesirable local optima were avoided, and helps the discriminator focus on texture discrimination from the start [Wang et al., 2018]. The learning rate is progressively halved every 200,000 steps. Following the pretraining step, the full GAN is run. Training is carried out on down-sampled images using a LR patch size of 128 x 128, and a mini-batch size of 16. In the original paper, two settings are tested for the generator. Firstly 16 RRDB blocks are tested, and then 23 RRDB blocks are tested. It is found that a deeper network benefits from a larger patch but this costs more computing resource. Unsurprisingly, training with more data also helps the model achieve better results.

A precept is laid out in Section 2.7 of this study, where a GAN can be used to move along the perception-distortion curve [Blau and Michaeli, 2018]. In ESRGAN, network interpolation was proposed to balance perceptual quality and distortion. Network interpolation can be achieved by either interpolating weights or interpolating pixels in images. Weights are created based on an intermediate point between the PSNR weights from the pre-training stage and weights from adversarial training. This is achieved using this equation:

$$\theta_G^{INTERP} = (1 - \alpha)\theta_G^{PSNR} + \alpha\theta_G^{GAN}) \tag{2.1}$$

where α is the interpolation parameter used to create the balance. The interpolation means that the network only needs to be trained once, and these training weights can then be used to fine tune the the perception-distortion balance. In contrast, image interpolation is achieved at a per-pixel level, where the weighted average of PSNR and GAN outputs is used.

As a seminal super-resolution technique, ESRGAN has been previously used to super-resolve Sentinel 2 satellite data by [Salgueiro Romero et al., 2020] who used WorldView satellite as a ground truth. The down-sampled World-

View satellite data is used for training purposes, as insufficient matching pairs of WorldView and Sentinel data were available. Training with Sentinel 2 and WorldView pairs was only carried out at the adversarial stage of training. Sentinel 2 data was upsampled prior to using the algorithm to allow the data to be more easily registered and aligned, and the ESRGAN algorithm was altered to remove the built-in up-sampling step. Although the authors report results using standard metrics to measure accuracy that exceed other techniques, they also report some GAN generated artefacts. Using network interpolation with different alpha values in allowed them to minimise these artefacts.

2.3.4 Other Super Resolution GANs

[Hoque et al., 2019] compare the output of two GAN models to the output of two simple CNN SISR networks. The CNN-based networks performed vastly better in terms of PSNR, SSIM and other metrics compared to the GANs, however the GANs generate HR images that are visually sharp and very photo-realistic. Images generated by the GANs also produce artefacts that affect overall image quality. Interestingly, when the VGG network used by the GAN for perception loss, is trained using satellite data, rather than generic image data, image quality is boosted both visually and quantitatively [Hoque et al., 2019].

In a similar vein, [Jiang et al., 2019] create a GAN-based edge-enhancement network (EEGAN) for robust satellite image SR reconstruction. EEGAN has several interesting features that enhance its performance: The generator component of the network contains an Edge Extraction SubNetwork (EESN) that serves to extract edge features. The EESN also contains a mask branch to learn a noise mask. This enables the network to focus on the real edge information and remove noise and artefacts. A Laplacian operator is used to further refine and extract edges. The other main component of the generator is a Ultradense Subnetwork block that contains several dense blocks and a reconstruction block. The two blocks are combined to produce an image

with clean edges. The discriminator, using a VGG network, takes the output of the generator and encourages it to produce an image more consistent with the HR image. When tested on DigitalGlobe imagery, EEGAN exhibits the highest scores in all indicators, including PSNR, SSIM, and FSIM. Many fewer artefacts are created than using SRGAN [Jiang et al., 2019].

2.3.5 Other Generative Models

Rather than implicitly exploring the data distribution as a GAN will do, a variational autoencoder (VAE) explicitly explores the data distribution for modelling. [Liu et al., 2021] proposes a reference-based super-resolution model that learns patterns from a reference to guide the super-resolution process. To do this, the reference patterns are compressed into a latent space using Conditional Variational Inference (CVAE) to learn an explicit probability distribution, and then these patterns are re-sampled as a prior to super-resolve data. The model, RefVAE, can select from a range of perceptual or visual qualities using a perceptual loss function for training.

Another explicit modelling approach for probability distributions is based on using normalising flows. In normalising flow models, a function $f(x)$ is created that maps a data distribution $p_x(x)$ to a different distribution $p_z(z)$. Unlike GANs, normalising flows have a single loss function, the negative log-likelihood, and are relatively straight-forward to train. A normalising flow model has another advantage over a GAN in that it allows the exploration of a multitude of solutions to a super-resolution problem rather than a single solution, by altering guiding parameters.

A normalising flow model called SRFlow was created to super-resolve imagery by [Lugmayr et al., 2020]. In SRFlow, the network is trained using an HR-LR image pair to learn the the conditional HR-image distribution. The network is trained by directly minimizing the negative log-likelihood using standard stochastic gradient descent (SGD) techniques. The network consists of 23 Residual-in-Residual Dense Blocks (RRDB), used to find the underlying

representation of the LR data. Following this, an invertible conditional flow step is used to transform the data density to the conditional HR-image distribution. In inference, HR samples can then be generated from low resolution data by applying the inverse of the network. Rather than capturing a single image, SRFlow captures a range of possible outcomes, which can then be explored by using additional guiding information or randomly sampling outputs. The authors of SRFlow note that while perceptual information is superior to ESRGAN, fidelity is also preserved resulting in a high PSNR. Additionally, they note that the same techniques used to super-resolve data also allow for image manipulation techniques, for example transfer of image content from one image to another [Lugmayr et al., 2020].

2.4 Remote Sensing

Remote sensing is the use of electromagnetic energy to measure the physical properties of distant objects. The history of remote sensing can be traced back to World War I and World War II, when millions of aerial photographs were manually analysed for military purposes [Moore, 1979]. The development of remote sensing platforms progressed rapidly through the twentieth century. A key moment was the launch of the first Landsat satellites in 1972, the first dedicated Earth landcover imaging satellites. For the first time, repetitive images of the earth were easily available for analysis. The first Landsat satellites (1 and 2) carried a green and red sensor and two NIR sensors. More recent Landsat satellites (Landsat 8 and 9) can acquire data from 11 spectral bands at between 15 and 30m resolution, vastly increasing the amount of information that can be acquired [Wulder et al., 2019]. A parallel effort by the European Space Agency (ESA) created the SPOT satellites with a relatively high resolution of 2.5m RGB bands, and more recently the Sentinel series of satellites that collected the data used in this study. In the last decade, cube satellites, such as those launched by *Planet* provide a higher return rate with images

available up to two times daily [Planet Labs Inc, 2021].

With the plethora of data available from remote sensing, comes the problem of trying to make the information useful for humans. Traditionally, expert derived algorithms such as Normalized Difference Vegetation Index (NVDI) have been used, for example, to show whether landcover consists of green plants or not. These types of algorithms are useful for providing a broad overview of the image data, but cannot by themselves derive more complex information [nvd,]. Hence machine learning has become useful for carrying out tasks such as crop disease detection. New product creation, bias correction and code acceleration can require machine learning. Deep neural networks are often required to interpret the remote sensing images to extract information such as soil moisture distribution, vegetation, recognition of crop type regions among others [Lary et al., 2016]. More recently, object detection from satellite data has been used to detect farm information such as grass cover, or economically important information such as the fill level of oil storage vessels.

Satellite data is comparatively cheap compared to aerial photography, so it is used for tasks that require large area coverage or regular coverage over time. However, it does suffer from the issue that commercial satellite images generally have a lower resolution than aerial or drone imagery, so cannot always resolve features to a high enough level to be useful for a given task. In 2020, the highest-resolution commercially available satellite data was from the satellite WorldView-3 with 30cm ground sample distance (GSD). Other satellites that provide sub-meter imagery products of GSD 50cm include WorldView-2, GeoEye-1, Pleiades. This resolution is still not great enough for many tasks, e.g., traffic monitoring [Zhu et al., 2020]. Super-resolution aims to solve this problem by providing tools that can enhance data resolution, and open up further uses for satellite data.

2.4.1 Satellite Data

Satellite data suffers from an issue that other imagery datasets do not have: Cloud cover. In 12 years of observations by the Moderate Resolution Imaging Spectroradiometer (MODIS), it was found that 67% of the Earth's surface is covered by clouds on average. This is lower over land, with only 55% of the area covered in cloud, and cloud cover is much lower on average during late summer and early autumn [Meraner et al., 2020].

Other issues that are particularly prevalent in satellite data include noise, and blur. Noise from satellite data is caused partially by inaccuracies in the point spread function (PSF) of the satellite imaging system [Zhu et al., 2020]. Motion blur is caused by satellite movement sensor scanning. Satellite data is likely to be processed and resampled, which causes further blur [Zhu et al., 2020]. Atmospheric disturbance is generally accounted for in processing imagery but, this too causes an effect.

These issues are frequently overlooked in super-resolution studies, which often use down-sampled higher resolution data to generate low resolution training data. Using down-sampled data is a feature of many studies including [Hoque et al., 2019], [Johnson et al., 2016] and [Wang et al., 2018]. These types of model have a limitation in that the degradation model might not match reality [Molini et al., 2019]. These issues are elaborated on in [Zhu et al., 2020], where the authors created simulated LR data from aerial imagery data using a corresponding noise kernel to simulate noise, and varied the down-scaling factor within a small range to simulate the blur and aliasing variation. The simulated images were used to train a neural network, and the output of this was compared to a network trained with bi-cubically down sampled data. The Mean Opinion Score (MOS) from the result was significantly better when using the simulated data compared the bicubically down-sampled data [Zhu et al., 2020].

2.5 Evaluating Image Quality

For evaluating the accuracy and perceptual quality of super-resolved images, there are no standard metrics, and the relatively reliable Mean Opinion Score (MOS) of human observers is time-consuming and expensive to obtain [Wu et al., 2020]. In the absence of humans, Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM) are commonly used metrics to evaluate image quality.

Many studies including [Ledig et al., 2017] use MOS testing. Specifically, humans are asked to rate images using a numerical scale, and this is used to derive an overall score to rate an algorithm. However, as this quality is difficult to standardise, and not easy to use in a research setting, most studies focus on attempting to mathematically measure image quality.

PSNR is usually used for image processing as a quantity based on for MSE. In the following equation, L is the dynamic range of allowable image pixel intensities, e.g., an 8 bit image will have an L value of $8^2 - 1 = 255$. This allows the MSE to easily be used to compare images of different bit depths [Wang and Bovik, 2009]

$$PSNR = 10 \log_{10} \frac{L^2}{MSE} \tag{2.2}$$

A higher PSNR value indicates a higher degree of similarity, so that the PSNR value approaches infinity as the MSE approaches zero. PSNR is used in a range of context for the reasons described above, and because it makes it easy to compare to previous studies as it is considered the benchmark measure [Wang et al., 2004]. However, it is widely accepted that PSNR does not correlate well with a human's perception of image quality [Zhao et al., 2016]. For example, an image that has undergone small geometrical modifications could have a large MSE with respect to the original image yet appear identical to the human viewer. The reverse is also true, in that an image distorted by additive white Gaussian noise or blurring may have a small MSE with respect to the original image, yet appear very different [Wang and Bovik, 2009]. These issues highlight the fact that the Human Visual System (HVS) is highly adapted for

extracting structural information [Wang et al., 2004], whereas measures such as PSNR deal with information at a pixel level.

As a consequence of these issues with PSNR, a range of alternate measures have been explored that attempt to mimic the qualities found in the HVS. These qualities have been explored using psychological and physiological experiments. Many approaches attempt to estimate the visibility of errors, and weigh these according to some criteria to adjust the MSE and estimate the quality of an image. An example of this is an approach whereby an estimate of the threshold at which stimuli are apparent is made and this is used to to define visual error sensitivity [Wang et al., 2004]. One of the issues with studying these type of phenomena is that most psychophysical experiments are conducted using relatively simple patterns, and this does not necessarily generalise to the complexity of the real world. Also, it does not always follow that error visibility leads to loss of quality [Wang et al., 2004].

A different approach to the methodology of trying to mimic the HVS is measuring structural information change to provide an approximation of image distortion. One of the measures using this approach is the SSIM which evaluates the signal changes between two complex-structured signals directly. In SSIM, the luminance, contrast, and structure of the image signal are compared separately [Wang et al., 2004]. While PSNR considers the squared error between pixels, SSIM takes into account the idea that pixels have a strong interdependency especially when they are spatially close.

To achieve a measure of this interdependency, SSIM removes the contrast, and luminance components of an image, and then measures structural attributes. Where there are two aligned image patches x and y from the same location, luminance is defined based on a comparison of the average signal intensity for each image being compared. These values are defined as μ_x and μ_y. In this approach, the luminance is the product of the illumination and the reflectance. To calculate contrast, luminance is then removed from each signal by subtracting average signal intensity from each pixel value. Contrast

is defined as a comparison of the standard deviation of each signal, σ_x and σ_y. To estimate structural differences, the signals are first normalized by dividing by their own standard deviation, and the covariance, σ_{xy}, is calculated. The structural information therefore represents the structure independent of contrast and luminance.

Equations to define the concepts luminance, contrast and structure are:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2.3}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_1} \tag{2.4}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{2.5}$$

In these equations, several constants are introduced, namely C_1, C_2, C_3. These serve to avoid instability when standard deviation values are close to zero. For further definition of these constants see [Wang et al., 2004].

The entire SSIM is defined as:

$$SSIM(x,y) = [l(x,y)^\alpha] \cdot [c(x,y)^\beta] \cdot [s(x,y)^\gamma] \tag{2.6}$$

where α and β and γ are the weightings of the luminance, contrast and structure which is used to adjust the relative importance of each. In this study these were left as the default of 1.

As image distortions are likely to be spatially variant, i.e., image quality will be better in some areas than others, SSIM is calculated locally using local variance and mean statistics. To calculate SSIM for an entire image, a sliding window is used to move across the image, pixel by pixel. At each step the SSIM index is calculated resulting in an SSIM quality index map. This quality map is essentially a distortion matrix of the two images, so if one image is considered perfect, then the SSIM index shows where the images differ. The mean of this quality map is used to calculate the overall image quality. SSIM values range between 0 to 1 where a perfect reconstruction yields 1.

One issue with this method is what the authors call *blocking artifacts*. This is overcome by using a 11 x 11 circular-symmetric Gaussian weighing function

with standard deviation of 1.5 samples that is used to weigh statistics for each pixel calculation [Wang et al., 2004]. In this study, the default Gaussian and sigma values 11 and 1.5 are used for the SSIM calculation.

[Hore and Ziou, 2010] investigate the relationship between PSNR and SSIM in order to better understand when one metric is better than the other, and derive a simple analytical relationship between them. Using a variety of image distortions on images from the Kodak photo database, the authors showed that the luminance component of SSIM when comparing images f and g, $l(f,g) > 0.991$ ($\approx$1), for common and well known degradations such as Gaussian blur, additive Gaussian white noise, jpeg and jpeg2000 compression. In these situations, it can be mathematically proven that PSNR and SSIM are not independent and essentially have a linear relationship when $SSIM > 0.2$ and $SSIM < 0.8$.

Following up on the mathematical proof, [Hore and Ziou, 2010] experimentally measured the sensitivity of PSNR and SSIM to the different types of degradation applied to different images. Using a group of degraded images, a function called the F-score is used to calculate the variance of the set of mean values of the PSNR or SSIM in all groups over the mean value of the within-group variances. A low F-score indicates that the parameters do not have a huge effect on the quality measure. It is found that PSNR is very sensitive to Gaussian noise and slightly more sensitive than SSIM to Gaussian blur, whereas SSIM is slightly more sensitive than PSNR to the jpeg2000 quality compression parameter [Hore and Ziou, 2010].

Image quality details, depend on the sampling density of the image, the distance from the image plane to the observer, and how perceptive the observer is. SSIM has been extended to take into account the multi-scalar nature of the HVS. One example of this is MS-SSIM, a multi-scale version of SSIM that weighs SSIM computed at different scales according to the sensitivity of the HVS. MS-SSIM works by iteratively passing over the image, applying a low pass filter and downsampling the filtered image by a factor of two at each

pass. At each pass, the contrast and structural components of the image are measured as per SSIM. The luminance quality of the image is only measure at the final pass. As per SSIM, the final metric is an average of the measurement for each pixel, or each downsampled area. The different scales are weighed using constants that have been established by experimentation with the by HVS to obtain an overall measure. The authors find that the MS-SSIM out performs single scale SSIM measurements by better correlating with how a human perceives image quality [Wang et al., 2003]. Another variant of SSIM is the so called complex wavelet or CW-SSIM, which is more robust to small geometric distortions [Wang and Bovik, 2009].

Rather than specifically codify an image quality metric to try and estimate human judgement, [Zhang et al., 2018] find that internal activations of networks trained for high-level classification tasks correspond to human perceptual judgements fair better than mathematical formula such as SSIM. One of the issues with pixel based metrics such as SSIM is they rely on data matching up spatially, and cannot handle situations where spatial ambiguities are a factor.

The stronger a neural network performs at classification or detection, the more closely the model aligns to human perceptual behaviour. In this metric, the distance between two patches is calculated by normalising the activations and computing the ℓ_2 distance. A scaling vector is applied to the activations from each layer so that layers with more channels have the same weight as layers with fewer channels. Distance is then averaged across spatial dimensions and across layers. The metric, called Learned Perceptual Image Patch Similarity (LPIPS), is tested using distorted data with random noise, blurring, spatial shifts and corruptions. The authors used a Two Alternative Forced Choice (2AFC) test that asks which of two distortions is more similar to a reference. From this, it is found that LPIPs performs better than various low-level metrics in terms of replicating human perception. The different networks trialled included AlexNet, VGG, and SqueezeNet perform at a similar level to

each other [Zhang et al., 2018]. The mechanism used for LPIPs quality metric is the same mechanism leveraged by perception loss, which is explored in further detail in Section 2.7.

2.6 Loss Functions

For a long time, despite being the driver of a network's learning, the loss function attracted little attention within the image processing research community. In recent years, this has changed, and the loss function has become one of the hot spots of research, with much of the effort focused on improving or constructing loss functions for a specific algorithm or problem [Zhao et al., 2016].

Mean Squared Error (MSE) or ℓ_2 is the dominant loss function used regression tasks, that is used in a wide range of applications including super-resolution. The reasons that MSE is so popular are that it is convex and differentiable, and is often available pre-packaged in software applications that make it easy to use [Zhao et al., 2016]. In addition, MSE is simple and easy to understand, and relatively computationally cheap [Wang and Bovik, 2009]. In the context of image processing, MSE quantifies the difference between the original image and the distorted image. In super-resolution, it quantifies the error between the HR and upsampled LR images. If N is the number of pixels and $x = (x_i | i = 1, 2, \cdots, N)$ and $y = (y_i | i = 1, 2, \cdots, N)$, then this can be defined by the equation:

$$MSE(x, y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2 \tag{2.7}$$

The ℓ_1 loss is also widely used, and differs from ℓ_2 in that it does not over-penalize larger errors, and may therefore have different convergence properties [Zhao et al., 2016]. The ℓ_1 loss can be defined as:

$$\ell_1(x, y) = \frac{1}{N} \sum_{i=1}^{N} |x_i - y_i| \tag{2.8}$$

Pixel-wise loss functions such as MSE struggle to handle the uncertainty inherent in recovering lost high-frequency details. As discussed in detail by

[Ledig et al., 2017], pixel loss functions can appear overly smooth due to the pixel-wise average of possible solutions in the pixel space. By favouring an average over the plausible HR solutions, a significant reduction of high-frequency details occurs [Lugmayr et al., 2020]. Finding pixel-wise averages of plausible solutions can result in poor perceptual qualities and lack of high-frequency details at the edges [Rad et al., 2019]. This smoothing factor is known to become more extreme with higher upscaling factors such as x 4 [Wu et al., 2020]. This leads to the issue that although using an MSE loss may yield a high PSNR value, it may correlates poorly with image quality as perceived by a human observer.

Intriguingly, although SSIM and MS-SSIM are commonly cited functions to measure image distortion which is discussed in detail in Section 2.5, they are not commonly used as loss functions in super-resolution. This is despite both functions being differentiable, and exhibiting obvious advantages over using MSE in the context of image quality [Zhao et al., 2016].

In a comparison of different loss functions, [Zhao et al., 2016] test the ℓ_1, ℓ_2, SSIM and MS-SSIM loss functions with a simple neural network to super-resolve previously down-sampled images. It is shown that by themselves, SSIM and MS-SSIM performed slightly worse than the ℓ_1 and ℓ_2 loss. The performance of SSIM and MS-SSIM is strongly related to the sigma value used. A higher sigma value has the effect of blurring edges, as the calculation for a given pixel draws information from a large region. Both structural loss functions are not particularly sensitive to a uniform bias on a flat region, for example an area of bright sky in a dark image. Both measures preserve the contrast in high-frequency regions better than the other ℓ_1 and ℓ_2 losses. Of the single loss functions, ℓ_1 loss is shown to perform the best. The authors of [Zhao et al., 2016] speculate that this may be because ℓ_2 gets stuck more easily in a local minimum, and for ℓ_1 might more easily reach a better minimum. ℓ_2 performs well, but creates splotchy artefacts on flat regions of images. The best results are obtained using multiple loss functions. When the network is

trained with ℓ_1 then ℓ_2, it reaches a better minimum than using a single loss function. To capture both the contrast-preserving characteristics of MS-SSIM and the performance of the ℓ_1 loss, a loss function combining these two is also evaluated. This combined loss, called *mix* is found to perform better than any single loss function or the ℓ_1 and ℓ_2 combination [Zhao et al., 2016].

In a similar vein, [Yang et al., 2020] uses different loss functions to dehaze single images using a neural network called Y-net that is named for its structure. The authors report that when the network architecture was unchanged, the quality of the results improves significantly with more suitable loss functions. A novel loss function is proposed, extending SSIM by combining it with the discrete wavelet transform (DWT) called L_{W-SSIM}. This function is created by dividing images into many patches using DWT with various frequencies. The SSIM loss of each patch is calculated and the weights of each loss adjusted, to better preserve detail, and prevent halo artefacts. Results of the study show that for the problem of dehazing images, L_{SSIM} performed better than an ℓ_2 loss when evaluating the quality of the resulting images using SSIM, but slightly worse when using PSNR.

2.6.1 Perception Loss

The pixel-based loss approach has been used by various authors in work on super-resolution, colourisation, and other tasks, however, these methods do not capture stylistic differences between the output and ground-truth image. Ideally, in super-resolution fine details are inferred from visually ambiguous low resolution imagery [Johnson et al., 2016]. Both MSE and SSIM have been found to correlate poorly with human assessment of visual quality, as both capture low-level differences between pixels.

The *perceptual loss function* aims to capture differences based on high-level feature representations rather than pixel based differences enabling a form of style transfer. The aim of style transfer is to capture the content of a target content image y_c with the style of a target style image y_s. In this paradigm,

content is the larger spatial structures in the image whereas style refers to the colours and local structures of the image. The insight that allows this transfer is that higher layers (or the layers closer to the output) in the network capture the high-level content in terms of objects and their arrangement in the input image, but do not contain information about detailed pixel values. In contrast, the lower layers (or the layers closer to the input) of a network capture information about the style of an image, but not its global arrangement. The key finding here is that style and content representations are to some extent separable. When CNNs are trained on object recognition, they develop a representation of that image that is increasingly explicit, i.e., further along the network feature maps are increasingly about content rather than style [Gatys et al., 2015]. This insight has allowed the style transfer process to be used across a range of spheres in deep learning such as artistic creation, but also in super-resolution.

To apply style from one image to the content from another image, loss functions must be devised that allow this transfer. The method used by [Johnson et al., 2016] to aid super-resolution is to create a network with two components: an *image transfer network* and a *loss network*. As existing networks have already learnt to encode perceptual and semantic information, a network pre-trained for image classification is used as a fixed *loss network*. The network most commonly used, and that used by [Johnson et al., 2016], is VGG pre-trained on ImageNet, or the MS-COCO dataset.

The mechanism behind perception loss is as follows: A *Feature Reconstruction Loss*, also known as *Content Loss* is calculated that encourages pixels of the output image to have similar feature representations to the *target loss* by minimising the squared, normalized Euclidean distance between the feature map of the output shape from the image transfer network, and the feature map from the *target loss*. This uses the fact that the feature maps in the deeper convolutional layers of a network represent larger-scale features of the original image. When the image is reconstructed from higher layers, image

content and overall spatial structure are preserved but colour, texture, and exact shape may be different. The output image is perceptually similar, but does not match exactly.

The formula for Content Loss is as follows:

$$ContentLoss(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^{l}(g) - F_{ij}^{l}(c))^2 \tag{2.9}$$

Where $F_{ij}^{l}(g)$ refers to the ground truth feature map for layer l at jth position of the ith feature and $F_{ij}^{l}(c)$ refers to the predicted feature map for layer l at ith feature and jth position.

A second loss function called the *Style Reconstruction Loss* penalises differences in style, represented by colours, textures, and common patterns, when they deviate from the target. To mathematically calculate this deviation, a concept called the *dot product* is used. The dot product represents the length of a projected *vectora* on *vectorb*, times the length of *vectorb*. The larger this product is, the more similar two vectors are. The sum of all dot products of a set of flattened feature maps is called the Gram Matrix. This measures overall style as different feature maps capture different elements of style. A lesser Gram Matrix means that learnt features differ from target features. A greater Gram Matrix means that features in each set of feature maps occur together giving a measure of style similarity [Rupprecht, 2017]. Using a combination of deep and shallow convolutional layers allows the network to measure style similarity for different scales [Johnson et al., 2016].

The mathematical foundation of *Style Reconstruction Loss* is as follows:

$$StyleLoss(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (G_{ij}^{l}(g) - G_{ij}^{l}(c))^2 \tag{2.10}$$

Where $G_{ij}^{l}(g)$ refers to the target Gram Matrix for layer l at the jth position of the ith feature and $F_i j^{l}(c)$ refers to the predicted Gram Matrix for the jth position of the l layer. In this case, the Gram Matrix G can be defined as:

$$G_{ij}^{l} = \sum_{k} (F_{ik}^{l}(g) F_{jk}^{l}(c))^2 \tag{2.11}$$

or the dot product of the activations for layer l. If the feature maps of the ground truth and the output are similar, then the same neurons will fire in the equivalent activation layers, and the gram matrix of each feature will be similar. This similarity is measured by the MSE (although other similarity functions can be used).

In addition to the style and content losses, the authors of [Johnson et al., 2016] used a pixel loss function which is the normalised Euclidean distance between output and target, and a *Total Variation Regularization* which encourages spatial smoothness.

[Johnson et al., 2016] use perceptual loss to super-resolve photos by a factor of 4 and 8 using the relu2-2 layer from the VGG-16 network (shown in Figure 2.1) trained with MS-COCO data for the feature reconstruction loss (content loss). The purpose of the network is to allow transfer of semantic knowledge from the pre-trained loss network to the super-resolution network. Style loss is not used in the super-resolution process presumably because the existing image style is adequate for the task. A histogram match was performed as a post-processing step between the network output and the original LR data. Resulting images exhibit sharper edges, and a are much clearer than the comparative pixel based loss methods. However several grid-like artefacts are present in the images at the pixel level reduced the PSNR and SSIM values [Johnson et al., 2016].

In a twist on the concept use by [Johnson et al., 2016], [Rad et al., 2019] use a targeted loss function to favour more realistic textures for different areas. Images are divided up into different areas: background, boundaries and object. Then a perceptual loss is computed for each area using a different function each using a different layer of the VGG loss network. By targeting an area, the algorithm is able to focus on edges or textures depending on whether the object is background or foreground. Although this study did not exceed previous SSIM or PSNR measures for super-resolution, ablation surveys showed that the images produced are more pleasing to the eye.

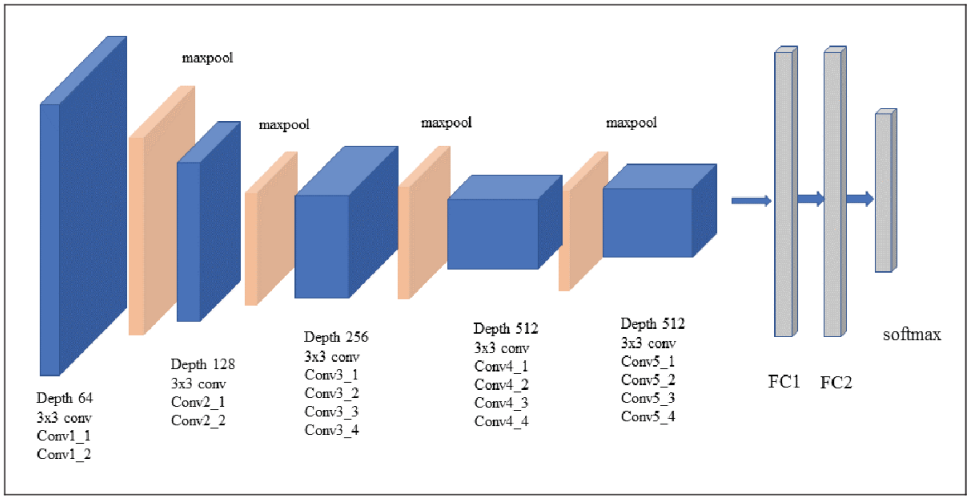


Fig. 3. VGG-19 network architecture

Figure 2.1: VGG-19 architecture: Typically the style feature maps are taken from the first convolutional block, whereas the content feature maps are from the fifth block

It is noteworthy that in perceptual loss functions, the VGG CNN is used, seemingly as this is the CNN used by [Johnson et al., 2016], and works well for most examples.

The VGG network used has been pre-trained on ImageNet [Simonyan and Zisserman, 2014] and it is instructive to consider example images from this dataset. Imagenet images are quite different in colour and texture to image patches created from satellite imagery. Hence, in this thesis, we also consider using a sparse autoencoder to capture feature maps to from the original HR satellite data in order to better capture feature maps relevant to the original data.

Autoencoders provide a way to learn features from unlabelled data in an unsupervised manner. In an autoencoder, data in passed through a neural network to compress the input in to a latent space, and this is then used to reconstruct the output. In other words, the network tries to reconstruct an approximation of the input. By creating a bottleneck such as limiting the number of hidden units, different structures in the data can be discovered [Ng et al., 2011].

In a so-called sparse autoencoder, this concept is taken further. A sparse

autoencoder is trained with a penalty, so that only a few nodes are encouraged to activate in each layer. The idea is that only the most useful structures relating to the data are discovered rather than redundant information. To this end, a regularisation term is added to the loss function that penalises the absolute value of the vector of activations for a layer [Ng et al., 2011].

Several other regularisation processes exist, including batch normalisation. Batch normalisation acts to standardise only the mean and variance of each unit in order to stabilise learning. Normalising the inputs can dramatically improve training time and performance by reducing the covariate shift or the change in distribution of network parameters during training.

In the perception loss function, Total Variation Loss is used to encourage image smoothness by reducing the amount of variation in the image. This concept is also used in image de-noising, where it is well-known that it possesses some properties such as which may be undesirable under some circumstances, such as staircasing and loss of texture [Chen et al., 2010]. Several algorithms exist to address this issue for example the total variation minimizing process of Rudin–Osher–Fatemi (ROF), where the model seeks to preserve image features such as edges [Rudin et al., 1992].

2.7 Perception-Distortion Trade-Off

[Blau and Michaeli, 2018] argue that there is an inherent trade-off between the perceptual qualities of an image, and the amount of distortion present when an image is recreated from noisy data or super-resolved. In this context, distortion refers to the dissimilarity between the reconstructed image $\hat{x}$ and the original image x. Perceptual quality refers to the visual quality of $\hat{x}$, regardless of its similarity to x. In other words, perceptual quality determines whether or not $\hat{x}$ looks like a valid natural image. x is a member of a set of natural images p_X, and $\hat{x}$ is a member of a set of derived images $p_{\hat{X}}$. Low perceptual differences occur when the distribution of reconstructed images approaches that of the

set of natural images or when $d(p_X, p_{\hat{X}})$ approaches 0, where the divergence function d corresponding to the HVS, has yet to be fully understood. In contrast, MSE estimates an average values over all possible explanations which in itself is not necessarily a valid image, and can be outside the manifold of natural images.

[Blau and Michaeli, 2018] find empirically and through mathematical proof that the relationship between distortion and perceptual qualities is a convex curve. To improve the perceptual qualities of an image to be very close to the ground truth implies distorting the image, and visa versa. From a practical point of view, there is an ideal place on the curve where distortion and perceptual validity are balanced depending on the use-case of the image.

The authors compare different super-resolution algorithms with regards to perceptual and distortion qualities of generated images. GANs are found on the perceptual side of the curve, whereas feed-forward networks with an MSE loss such as DeepSUM are found on the opposite side where distortion is minimised. This finding is echoed by the authors of ESRGAN [Wang et al., 2018]. In this continuum there is an unattainable region where both perceptual qualities are very good and distortion is minimised.

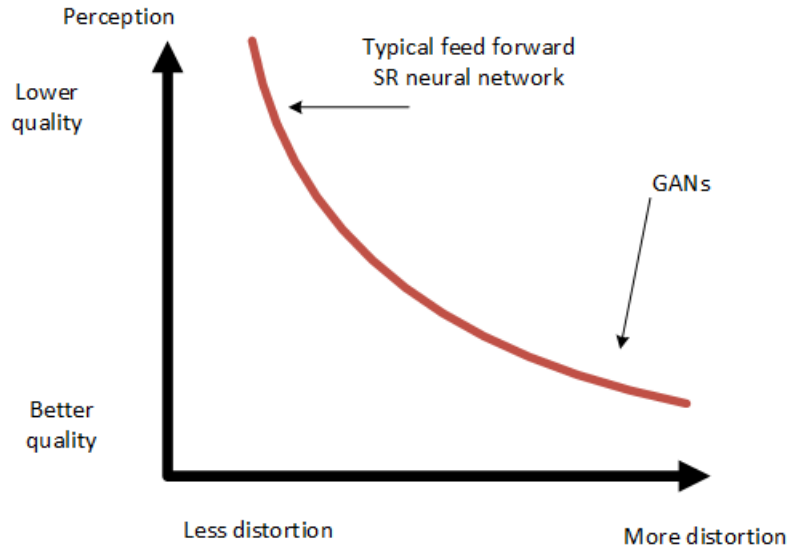


Figure 2.2: Perception-distortion tradeoff from [Blau and Michaeli, 2018] showing how improving an algorithm in terms of perception occurs only at the expense of increasing distortion and visa versa

According to the authors of [Blau and Michaeli, 2018], a Generative Adversarial Network (GAN)is the perfect place to explore this balance. As discussed further in Section 4, the loss function in a GAN can be composed of an adversarial loss and a distortion loss. This can be described as:

$$\ell_{GAN} = \ell_{distortion} + \ell_{adversarial} \tag{2.12}$$

where the distortion loss is typically an MSE loss, and the adversarial loss is the standard GAN adversarial loss which measures the deviation of the generated images from data distribution. By increasing the relative portion of MSE versus adversarial loss in a GAN, it is possible to move along the perception-distortion curve, and move from a blurry accurate image to a sharp but less accurate image.

It is important to note that based on the literature, the existence of the perception distortion trade-off is not universally accepted. [Lugmayr et al., 2020] explore normalising flows (see Other Generative Models), which may not have the same limitations as GANs. However, as this thesis compares feed-forward networks with the output of GANs, discussion of this trade-off is relevant.

Chapter 3

Methodology

In the experiments presented in this thesis, images cropped from Sentinel 2 satellite data are super-resolved using aerial photography as a ground truth. We consider three methodologies: Firstly, a feed-forward network called DeepSUM is trained and used to super-resolve multiple LR images to a single product. Different loss functions are tested to improve on the original process. Secondly, a GAN called ESRGAN is trained and used to super-resolve single Sentinel-2 cloud free images. Finally, the two methods, are combined whereby the product of DeepSUM is super-resolved and then passed through ESRGAN minus the upscale step, to improve the perceptual qualities of DeepSUM's output. These processes are shown diagrammatically in Figure 3.1. In each super-resolution process, images are super-resolved by a factor of four from 128 x 128 pixels to 512 x 512 pixels.

3.1 DeepSUM

The feed-forward network, DeepSUM is an example of discriminative learning applied to super-resolution. DeepSUM is a CNN developed by a team at *Politecnico di Torino* for super-resolving multiple unregistered temporal images to a single HR image using an upscale factor of three: In the original algorithm, the LR images are up-sampled from 128 x 128 pixels to 384 x 384 pixels. In this thesis, the algorithm has been adapted to up-sample imagery by a factor

of four to 512 x 512 pixels, among other changes.

Given the distortion-perception continuum explored in Section 2.7, DeepSUM is expected to produce an output that minimises distortion at the expense of perception. DeepSUM employs a supervised, discriminative deep learning approach, where the CNN learns the residual between a bicubic interpolation and the ground truth. Using multiple images, it aims to exploit the extra information provided by the temporal depth.

The method won the PROBA-V super-resolution challenge issued by the European Space Agency (ESA) [Molini et al., 2019]. In the PROBA-V challenge, teams are given multiple images from each of 78 Earth locations that need to be super-resolved against an HR image taken from the same satellite. The satellite data used by PROBA-V is Top-of-Atmosphere reflectances for the red and NIR spectral bands at 300m (LR) and 100m (HR) resolution. Each image comes with a quality map indicating pixels affected by cloud, shadow, ice, water etc. Each data point contains one HR image and several LR images recorded within 30 days of each other. This set of images is referred to as an *image set*. At each location, there are up to 19 different LR images. The data is not corrected to align with each other [Molini et al., 2019]. The unique feature of this dataset is that both the HR and LR images have been separately acquired by the same satellite, as opposed to using artificially downsampled data, i.e., data that has been previously downsampled from an HR image [Molini et al., 2019]. The competition expected LR images to be super-resolved from 128 x 128 pixels to a 384 x 384 pixel image.

3.1.1 DeepSUM Architecture

The DeepSUM process is a CNN with three main parts: an SISRNet block that performs several 2D convolutions followed by instance normalisation on each of the individual LR images; a RegNet block where the images are registered with respect to the best image in the group; and a fusion block called FusionNet where the multiple outputs are fused into a single image using 3D convolutions.

This fusion happens progressively so that the multiple images are gradually reduced down to a single image in a slow fusion that allows the network to learn relevant features from the feature space. Finally, a residual connection is added to the fusion output as a mean of the input images. DeepSUM is optimised in an end-to-end fashion so that the registration and fusion tasks leverage the learning capabilities of the SISRNet block. The output image is a single super-resolved image. The overall process is shown in Figure 3.2.

3.1.2 DeepSUM Unique Features

The algorithm has unique features that allow it to achieve a top result. The spatial registration task, which occurs in the RegNet block, occurs inside the algorithm itself. Registration filters are dynamically computed for each image, and the network makes use of the features to compute the optimal registration per image, rather than performing the registration in the pixel space. According to the authors, using the feature space is advantageous as it makes the algorithm robust to scene variations. The spatial registration step is trained concurrently as the image is super-resolved rather than doing this as a pre-processing step as occurs in the majority of other MISR methods. When the authors tested the network with and without the RegNet block, it was found to have a small but significant effect on the PSNR values obtained from passing test data through the network.

Another key aspect of the method is the use of what is called *mutual inpainting*, which is the process whereby unreliable areas as defined by the quality map are filled in with values from feature maps from other images, with more reliable values. This is important as cloud areas and other similar quality issues do not provide any useful information and if left in the dataset, create a lot of noise.

The loss function used in the algorithm is a modified MSE that uses only HR pixels clear of cloud, and image sets where at least one LR image is clear. Moreover, as each of the output images and the ground truth HR image can

have quite different brightness from each other, the modified loss function equalises the brightness intensities between the output SR image and the HR target. Also as each of the LR image could be expected to be shifted by a certain number of pixels, the output SR image is cropped by the maximum expected shift. Similarly, the evaluation function for the network, used to assess performance is a modified PSNR method. The authors note that this is also the method used by the ESA challenge to compared scores.

DeepSUM uses instance normalisation after each convolution in place of batch normalisation. This means the network can be trained independently of brightness differences between the images. Each layer in the network applies Leaky ReLU activations, except for the last layer, which uses the identity function.

3.1.3 DeepSUM Training and Metrics

The authors of DeepSUM state that the algorithm is difficult to train end-to-end from scratch due to several local minima, so they resort to pre-training the RegNet and SISRNet blocks. They train SISRNet (the initial block) by resolving single images against the corresponding HR image. This trains the block to find spatial correlations to generate the best feature maps for the SIRS task. They train the RegNet step to generate registration filters, i.e., a set of filters of size $K \times K$ that correspond to the set of possible shifts that an image would need to move.

The measure used by the authors of DeepSUM to rate is efficacy and compare the network to other similar networks is based on PSNR. A modified PSNR (mPSNR) is used, which only compares pixels where both the LR input images and HR ground truth image are cloudless.

From this, the authors find that the more images are used in an image set, the better the result, to a maximum of around 8 or 9 images, at which point, more images do not increase the accuracy. In the authors comparison of the DeepSUM algorithm and other state-of-the-art deep learn-

ing super-resolution algorithms such as IBP [Irani and Peleg, 1991] and DUF [Jo et al., 2018], DeepSUM was found to perform significantly better than bicubic interpolation, and better than other methods.

3.1.4 DeepSUM Algorithm Changes

For the experiments in this thesis, the DeepSUM algorithm, has been adapted to super-resolve Sentinel-2 data by a factor of 4 (rather than 3 as used in the original algorithm), using aerial imagery as ground truth HR images. This change is made to better match the data, as 10m pixels from Sentinel-2 resolve well to $4 \times 2.5m$ pixels with no rounding required. This change also means the up-sampling factor is the same as used in ESRGAN, which make these methods more comparable. Several changes are made to the algorithm to facilitate using these datasets with a different up-scaling factor.

Tweaks were required to the algorithm to work with a different size image set. In the *Fusion Net* sub-network, feature maps from each of the individual images are combined to create a single image. The original algorithm used the best 9 images in a set and these were reduced down to a single image using four $3 \times 3 \times 3$ 3D convolutional layers. A reduced image set size necessitated a minor architectural change. If c is the convolution size and t is the size of the tensor in the temporal dimension, each convolution step reduced the tensor in the temporal dimension by a factor of t - c + *1*. With a reduced set size, this required a lesser reduction. In place of the four $3 \times 3 \times 3$ 3D convolutions, three $2 \times 3 \times 3$ 3D convolutions were used followed by a single $3 \times 3 \times 3$ 3D convolution. These different configurations are shown in 3.3.

Other updates to the original algorithm include the creation of different loss functions, and addition of several accuracy metrics, notably SSIM and variation loss (see Section 4.4).

3.2 Sentinel-2 Imagery

The data used in this study is real LR data, not downsampled HR data. This represents both a challenge and an opportunity. A challenge in that extensive pre-processing has to occur to make the training data and the ground truth work within the network. The opportunity is to test assumptions made in other studies where synthetic data was used, i.e., LR data created by down-sampling the HR ground truth data, for example [Wang et al., 2018], [Hoque et al., 2019]. Synthetic data can run in to issues by potentially hiding shortcomings of the model used [Zhao et al., 2016].

The ESA designed the Copernicus Sentinel-2 mission to monitoring variability in land surface conditions to replace and provide continuity for the SPOT satellite data. Sentinel-2 has a wide swath of 290km and high revisit time of 5 to 10 days for each satellite. The mission comprises a constellation of two polar-orbiting satellites placed in the same sun-synchronous orbit, phased at 180° to each other. With these characteristics, under cloud-free conditions, mid-latitudes are imaged every 2 to 3 days by one of the two satellites [European Space Agency, 2013].

Sentinel-2 satellites have a range of sensors at different spatial resolutions resulting in 13 distinct bands. As shown by the figure above, the four bands comprising the RGB and near-infra-red (NIR) area of the spectrum have 10m spatial resolution. These bands were used in this study as they correspond to the bands that were available in the ground truth HR image used.

The other 9 bands available from Sentinel-2 satellites are mostly in the short-wave infrared (SWI) and NIR areas of the spectrum and have either 20m or 60m resolution. As no HR data is available in equivalent bands, there is no way of training these bands.

Sentinel 2 outputs are available as a range of products including uncompressed raw data, and radiometrically corrected radiance data. The Level-2A product used in this study provides orthorectified Bottom-Of-Atmosphere (BOA) reflectance with dub-pixel multi-spectral registration. This is a Level

1C product (top-of-atmosphere) that has been resampled with an atmospheric correction applied. A scene classification map (cloud, cloud shadows, vegetation, soils/deserts, water, snow, etc.) is included in the product. The Sentinel data is disseminated as tiled ortho-images, where each tile is $100km \times 100km$ in the UTM/WGS84 projection. The tiles are available to users in Sentinel-SAFE format from the Copernicus data hub[Tona et al., 2018]. Data older than a year is archived and kept in cold storage for users to access on request. Of the two satellite currently in orbit, Sentinel-2A was launched in 2015, whereas Sentinel-2B was only launched in March 2017, so no data is available from this satellite from before this time.

The HR data used in this study is from the Waikato Regional Aerial Photography Syndicate (WRAPS) dataset owned by the Waikato Regional Council. The data is freely available via their download service [LINZ, 2017]. A related data layer called the Waikato 0.3m Rural Aerial Photos Index Tiles contains metadata for each data tile including capture date. The dataset has 30cm pixel resolution GeoTiff data in the New Zealand Transverse Mercator (NZTM) map projection with a spatial accuracy of 0.5m. Data was collected over the course of the three years from 2016 to 2019 with the majority of data flown between December and March. As the data was generated from midday closely-spaced flight paths on selected days, it is almost entirely cloud free with minimal shadows. Included in the data is an auxiliary layer with flight dates. Using this layer means that data from a certain date period could be identified.

Data from Sentinel 2 was selected to show an intersection of the following qualities:

1. Sentinel data that is captured at the same time as WRAPs aerial photography.

2. Sentinel data that occurred after the launch of Sentinel-2B so as to take advantage of twice as much temporal data.

image name	image date	satellite
S2B_MSIL2A_20190116T221559_N0211_R129_T60HUC_20190116T235825	16 Jan	S2B
S2A_MSIL2A_20190121T221601_N0211_R129_T60HUC_20190121T234234	21 Jan	S2A
S2B_MSIL2A_20190208T222539_N0211_R029_T60HUC_20190209T000943.	8 Feb	S2B
S2A_MSIL2A_20190210T221601_N0211_R129_T60HUC_20190214T160249	10 Feb	S2A
S2A_MSIL2A_20190213T222531_N0211_R029_T60HUC_20190214T000858	13 Feb	S2A
S2B_MSIL2A_20190215T221309_N0211_R129_T60HUC_20190216T000238	15 Feb	S2B
S2B_MSIL2A_20190225T221559_N0211_R129_T60HUC_20190226T013517	25 Feb	S2B
S2A_MSIL2A_20190302T221601_N0211_R129_T60HUC_20190303T001135	2 Mar	S2A

Table 3.1: Table showing images used in study

3. Sentinel data that is as cloud free as possible. In the Waikato this means data from late summer to early autumn is likely to be the best.

4. Data that exhibited a wide range of different landscapes in order to better train.

5. Data with as few artefacts as possible in both the WRAPs and Sentinel datasets.

With these criteria, the area of interest selected was from February 2019 in the southern Waikato region of New Zealand. The time span used for the study was approximately six weeks, i.e. similar to the time span used by the authors of DeepSUM. The Sentinel images shown in Figure 3.1 were obtained from the Copernicus site.

3.3 Processing Environment

All data pre-processing occurred on a Windows PC with an Intel(R) Core(TM) i7-9750H CPU, 2.60GHz, 2592 Mhz, 6 Core(s), 12 Logical Processor(s) with a RTX 2060 GPU with 6GB on-board memory. Training occurred on a Linux Ubuntu server using a Nvidia 1080 GPU with 8 GB on-board memory.

Development of models and data processing was coded in Python 3.6 using Tensorflow and Keras. TensorFlow is an open source library for numerical

computation and large-scale machine learning. TensorFlow version 1.13.1 was used in this study for the majority of processing and training DeepSUM. TensorFlow version 2.1 was used for training and inferring with ESRGAN. Keras is a high-level neural networks library that runs on the top of TensorFlow. In this study Keras version 2.2.4 was used in conjunction with TensorFlow 1.13.1.

3.4 Image Processing

A significant series of preprocessing steps needs to occur before data can be used in DeepSUM and ESRGAN. These steps involve processing the LR, HR and cloud mask data, then cropping images from these datasets. Finally patches are taken from the image sets. Figure 3.1 shows these steps at a high level.

3.4.1 WRAPs Processing

As the WRAPS data and the Sentinel-2 images do not line up completely, the final data used is the intersection of these two image layers. This shown in Figure 3.5

WRAPS Index Tiles were merged by month to obtain a data layer. Figure 3.6 shows where large areas of aerial photography data is available from the same dates. Areas where tiles are sourced from multiple dates, e.g., Feb/Mar 2019 were excluded from the data layer. In this study imagery from February 2019 was used, as this presents an adequately sized area.

The WRAPS data was processed using ArcGIS Pro [Esri, 2021]. The following steps were carried out:

1. The WRAPs image tiles were added to a mosaic dataset, and overview tiles created. This made subsequent data processing more manageable and faster .

2. The resulting mosaic was reprojected from NZTM to a single image in WGS84 projection, so as to use the same projection as the Sentinel data.

3. The data was resampled down from 0.3m pixels to 2.5m pixels. This resampling meant that the WRAPS data was now exactly 4 times higher resolution that the Sentinel RGB images.

4. As part of the resampling, the WRAPS data was aligned with the Sentinel-2 images so that a single Sentinel-2 image pixel contained exactly 16 WRAPS pixels. WRAPs was aligned to Sentinel rather than the other way around, as the down-sampling step gave an opportunity for data alignment without further information loss.

A *mask area* raster layer was created in order to mask out areas where either WRAPs data or Sentinel data was not available or where the data was located in the sea.

3.4.2 Cloud Processing

As the WRAPs data is effectively cloud free, there is no need to account for cloud issues in this data. However, to make DeepSUM work without further adjusting tensor sizes, etc., a quality mask with no quality issues was created.

For the original DeepSUM algorithm, a quality map was helpfully supplied by the ESA. This quality map included cloud, cloud shadow and other ambiguous areas. In order to replicate the quality map for the Sentinel 2 data, the SNAP tool including the separate *IdePix* plugin from ESA [ESA, 2020] was used to process data. Using the .SAFE format, RGB imagery was exported from SNAP as a PNG file at full resolution for further processing. A quality map was created using the following process inside the SNAP tool for each image:

1. In order to create a cloud mask layer IdePix uses several bands other than RGB. The image was resampled using the *S2 Resampling Processor*. This serves to give the lower resolution bands in the image the same resolution (10m) as the RGB bands.

2. The IdePix plugin was used to create a cloud layer including cloud shadow with an eight pixel buffer.

3. A layer was created with any pixels that the IdePix algorithm marked as invalid, cloud, ambiguous cloud, sure cloud, cloud buffer, cloud shadow, cloud cirrus, cloud cirrus ambiguous, clustered cloud shadow.

4. The output of this layer was merged with the output of a combined boundary layer to screen out areas where no image was available for a particular Sentinel 2 photo.

5. The bit depth of the resulting layer was updated to 1bit in order to match that used in the original algorithm.

In creating a raster defining areas to mask, a balance was struck between removing too much cloud and useful imagery and removing too little. Inevitably, there are areas of wispy cirrus cloud around the edges of the more obvious cumulus clouds where IdePix does not always recognise all possible cloud areas. Adding a buffer to the clouded areas helped ensure the vast majority of cloud was removed. Figure 3.7 shows an example of a cloud mask.

3.4.3 Image Cropping

DeepSUM works on a single image band at a time, so processing needed to occur to create input products for each band. DeepSUM uses a data concept called an image set. An image set is a numbered directory containing a set of temporal LR images for each band, a corresponding HR image, and a cloud mask for each of the images in an image set, including the HR image. In the original DeepSUM, each LR image was 128 x 128 pixels, and each HR image was 384 x 384 pixels. As mentioned above, this data was supplied by the ESA.

In order to crop images to create image sets a script the following processes were run:

1. The area of the Sentinel-2 tile was randomly sampled, and a box of 128 x 128 pixels was drawn in the image

2. If all four corners of the box fell outside the *mask area*, then a clip was taken of all Sentinel-2 images (LR), each of the corresponding Quality Mask (cloud) layers, the single WRAPs image (HR), and the *fakecloud* layer. Note that the WRAPs data was clipped at 512 x 512 pixels, i.e., 16 times the size of the LR data.

3. When the *Quality Mask* image was clipped, the resulting raster was reversed so that cloud was 1 and non-cloud is 0 i.e. the opposite of the original cloud layer created by SNAP.

The result was a set of image sets, so that each image set contained 8 LR images for each of the three bands, 8 LR RGB images and 8 quality cloud mask images, each of 128 x 128 pixels. The ground truth in each image set was a single HR image of 512 x 512 pixels for each band, an HR RGB image, and a single *fakecloud* image. As per the original DeepSUM, the LR images and HR images used were all 8 bit, however the cloud mask image was 1 bit.

3.4.4 DeepSUM Data Preprocessing

Data pre-processing occurred to prepare the data for training and convert the images into a set of pickled numpy arrays [NumPy, 2021] containing image patches and metadata to feed in to the network. This is shown in figure 3.1 as the *Preprocessing Steps*. The amount of data generated is shown in Figure 3.2.

Each of the cropped LR images in an image set was first registered against the best LR image in the set. The best image was the most cloud-free image. A Fourier shift was performed on each of the images to move them by up to 4 pixels towards this image. Images where there was more than 70% cloud, or images which the Fourier shift algorithm tries to move by more than 4 pixels, were discarded. In the original DeepSUM, image sets with fewer than 9 images were discarded and when more than 9 images were present in a set, the clearest 9 images were used. In this study, as less temporal imagery was

Data	Description
Raw data	• 8 Sentinel-2 temporal images for a single area • 1 WRAPs mosaic
Image Sets	• 500 (training and validation). • 100 (test)
Pickled patches	• up to 80 000 patches for each band • equates to 20 patches per usable temporal image (training and validation) • up to 16 000 patches for each band (test)
Inference product	• around 90 images inferred (from 100 image sets) for each band

Table 3.2: The amount of processing output generated at each stage of the process

available, image sets with fewer than 6 images were discarded. An initial set size of 6 was chosen as this maximised the use of the available data, while still maintaining an adequate number of images. As discussed, the authors of DeepSUM found that a larger image set gave a better result up to 9 images.

The LR images and their corresponding cloud masks were bicubically upsampled by a factor of four. An arbitrary 20 patches of 96 x 96 pixels were taken from each image or cloud image in an image set. 20% of the data was randomly set aside as a validation set. Finally, the sampled images, masks, validation sets, and a shift matrix were loaded into blocks of numpy arrays

which were pickled. In this study, a further numpy array was created containing the number of the image sets that failed due to too much cloud, or too large of a Fourier shift. This was done to speed up the network processing time.

Several aspects of the original study were not implemented in this study. Some of this was a function of the data used. The data used in this study consisted of 8 bit images whereas the original DeepSUM PROBA-V data consisted of 14 bit images in 16 bit files. In the original DeepSUM, pixels with very high values (i.e., over 60000) were removed from the PROBA-V data. This step was not necessary in this study, as data artefacts such as very high pixel values were not a factor.

In the original DeepSUM, the preprocessing was done in multiple steps, however, in this study, these steps were combined in to a single step that also included data standardisation (discussed in the Section 3.4.5).

3.4.5 Data Standardisation

In the original DeepSUM algorithm, the entire dataset is standardised to have a mean of 0 and a standard deviation of 1 using the same adjustment for each image. This aided training as both unscaled input variables and an unscaled ground truth can result in a slow or unstable learning process. Also, gradient optimisation methods converge more rapidly with when features have zero mean and unit variance [Waldner and Diakogiannis, 2020]. Data scaling using fixed values, probably worked well in the original DeepSUM, as return values from the ProbaV satellite are more consistent than those from the Sentinel 2 satellites. In the modified DeepSUM process, standardisation is carried out as a pre-processing step, rather than as part of the algorithm. This is necessary as it allows mean and standard deviation values to be calculated for each of the individual photos taken and applied to each component of the image set. Figure 4.2 shows how the different images of the same area have different pixel values, so that it is better to calculate the image mean separately for each

image.

3.5 Data Augmentation

A second set of image sets was created where the image sets were augmented by adding downsampled HR image crops to each imageset. Three downsampled HR images were created in each imageset with either no modification, brightness increased by 20% from the existing HR image, or brightness decreased by 20% from the existing HR image.

Using the augmented data meant that the set size could be increased from 6 images per set to 8 images per set. The augmented image sets contained either 2 or 3 augmented images and 5 or 6 LR images depending on the number of LR images available (after some images could not be registered during the pre-processing step), to make a total of 8 images per image set. Memory capacity issues on the training server meant that 9 images per set could not be processed, so no further augmentation was tested.

3.6 DeepSUM Pre-training

In the original DeepSUM algorithm shown in Figure 3.2, the authors found that pre-training the SISR block and the RegNet block improved the ease of training of the entire network. In order to test that the weights obtained by pretraining DeepSUM on PROBA-V data would still work for the dataset used in this study, the pre-training SISR was run on Sentinel-2/WRAPs data using the Red band only, and compared to the original weights.

Results in Table 3.3 show that the weights from this study and the original DeepSUM weights are roughly equivalent, and so pre-training on the new data is not necessary. It was also found that pre-training separately for each band was not necessary, and using the generic weights was sufficient.

neuron	Our Weights (conv_up_8/W)	DeepSUM pretrain weights
1	0.10122343	0.09070993
2	-0.04995382	-0.04984398
3	0.12592246	0.12341627
4	0.27446136	0.2716387
5	0.03364012	0.02915093
6	-0.18003301	-0.17592782
7	0.01882109	0.01274273
8	0.06290359	0.05531104
9	-0.08259003	-0.08596298
10	0.31987825	0.3082205

Table 3.3: Table showing the *convup*8 weights from DeepSUM and weights from pretraining SIRS network on Sentinel-2/WRAPs data as an example. These weights are sufficiently similar that pre-training on our data was not necessary.

3.7 ESRGAN

The ESRGAN algorithm is the polar opposite of DeepSUM in that it is more perceptually focussed. The intuition and design of ESRGAN is described in 4.12. The original ESRGAN was created using pytorch, however the implementation used in this study was run using TensorFlow 2.1. This implementation of ESRGAN used 23 ResNet blocks. ESRGAN super-resolves RGB images (not a single band at a time).

To training ESRGAN, pairs of LR and HR data were serialised as tfrecords, which are sequences of binary strings optimised for fast data reading [Tfrecord, 2021]. As described in Section 4.12, pre-training was carried out, however only 20,000 steps were required, as the loss quickly converged. Following the pre-training step, the full GAN was run. As per the original paper, a super-resolution

factor of 4 was used so patches of 128 x 128 were super-resolved to 512 x 512. The LR image size, and the super-resolution factor are the same as was used in the implementation of DeepSUM. The exception to this was when ESRGAN was run against the output of DeepSUM, where data had already been super-resolved by a factor of 4. In this case, the up-sampling step was removed from the algorithm, as up-sampling was not required. As per the original implementation, a batch size of 16 was used for training, apart from when training against the DeepSUM output, where a batch size of 2 proved necessary to prevent out-of-memory errors.

3.7.1 ESRGAN Data Processing

When training against Sentinel-2 data, only cloud-free images were used, as ESRGAN has no built in method for imprinting cloud. RGB images were selected from the image sets created by image cropping (see Section 3.4.3). Processes carried out prior to the image cropping step were used to create LR and HR data inputs for ESRGAN (when super-resolving raw Sentinel images), as well as DeepSUM (see Figure 3.1). As ESRGAN responds well to a large number of training images, many more image sets were used to train ESRGAN than DeepSUM. Table 3.4 shows the amount of data used at each stage of the process.

3.8 Image Colour Adjustment

Initial experiments showed the results from DeepSUM and ESRGAN did not match the colours used in the original aerial imagery. Hence, in a final step, colours on test images were adjusted to match the original. Colour adjustment occurred in all cases after any training or inference steps. This adjustment occurred via the the process outlined in Figure 3.9

Data	Amount generated or used
Raw data	• 8 Sentinel-2 temporal images for a single area • 1 WRAPs mosaic
Image Sets	• 5000 (training and validation). Only rgb images were used to super-resolve data from raw Sentinel images • 100 rgb images used only (test)
DeepSUM outputs	• 5000 DeepSUM outputs used to train ESRGAN (process 3 in 3.1) • 100 DeepSUM outputs used as test data

Table 3.4: The amount of processing output used at each stage of the ESRGAN process

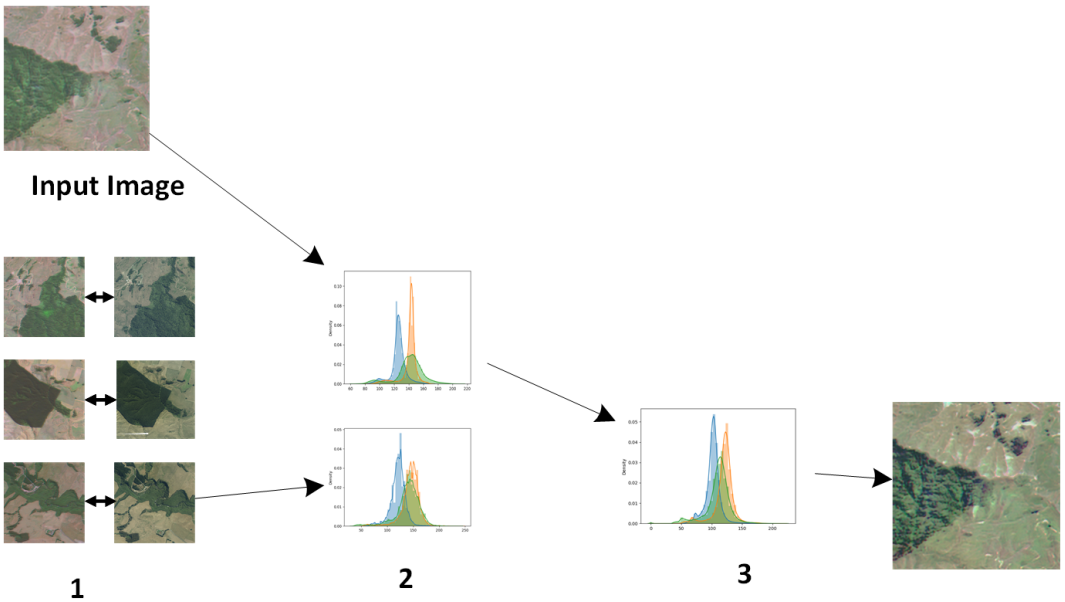


Figure 3.9: Colour adjustment process 1. A dictionary of 1000 images from the training data was created where each ground truth aerial image was matched to the corresponding DeepSUM output created from training data. 2. The histogram of each ground truth and output image was calculated. Note this calculation occurred as part of the match step as this was easier than storing

Using a histogram match would not be expected to give an exact colour match, but given 1000 samples, it would be expected that any given combination of colours would almost certainly be found in the dictionary. Once the dictionary was created, using the DeepSUM output to look-up colours from the dictionary meant that outputs other than DeepSUM for example GAN outputs could also be colour adjusted. This has the advantage that the adjustment can be performed in practice, when HR images are not available. [Johnson et al., 2016] appear to have used a similar technique, however, in their study, a match was created between output and a low resolution dictionary.

3.9 Inferring and Measuring Output

In each variation of DeepSUM and ESRGAN, 100 image sets or images were inferred from test data to create approximately 90 output images. In the case of DeepSUM, of the 100 image sets used, approximately 10 could not be used due to issues with excessive cloud cover over many images in a set, or data alignment as described in Section 3.4.4.

As described above, methods for assessing the accuracy of DeepSUM output are imperfect. In this study PSNR and SSIM values were used to assess overall accuracy of DeepSUM. Assessing the accuracy of each test run involved creating an output product from the preprocessed test data, and running a assessment algorithm over the output to calculate PSNR and SSIM values for both the output data, and bicubic upsamples of the raw Sentinel 2 imagery. When creating the bicubic upsamples, only completely cloud free images were used, and the bicubic output was averaged over each image set. LPIPs was used to measure perceptual qualities of the RGB images for each of the processes.

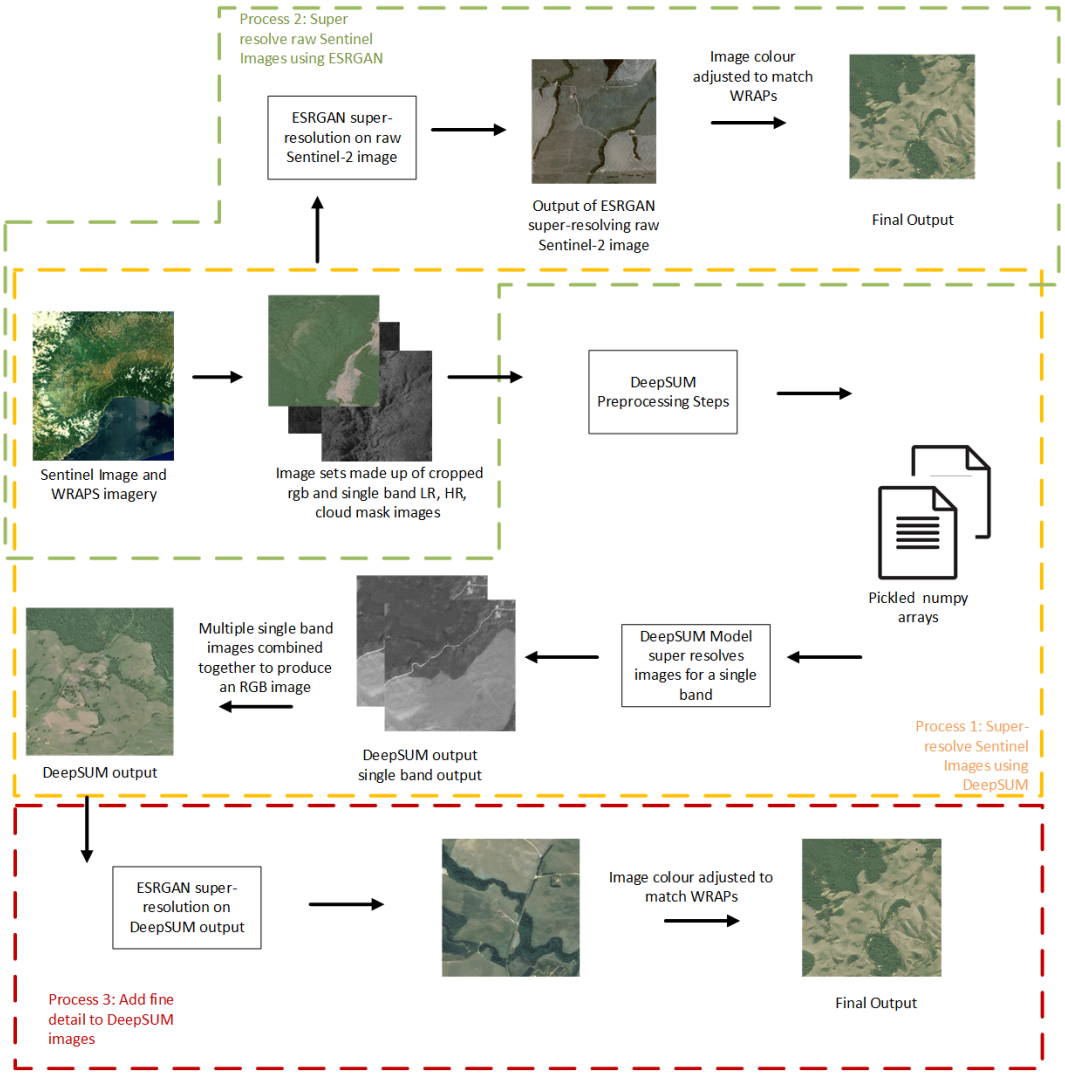


Figure 3.1: The overall DeepSUM and ESRGAN super-resolution process showing (1 - yellow) the adapted DeepSUM process, (2 - green) the ESRGAN process used on raw Sentinel-2 data, (3 - red) the ESRGAN process used on DeepSUM outputs.

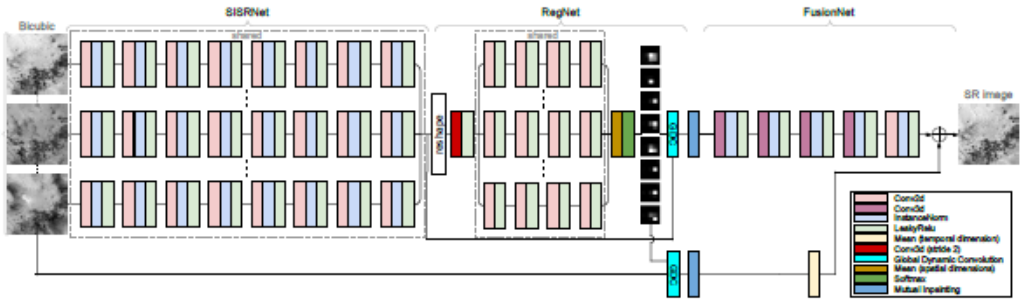


Figure 3.2: The DeepSUM network showing the three blocks taken from the original DeepSUM paper [Molini et al., 2019]

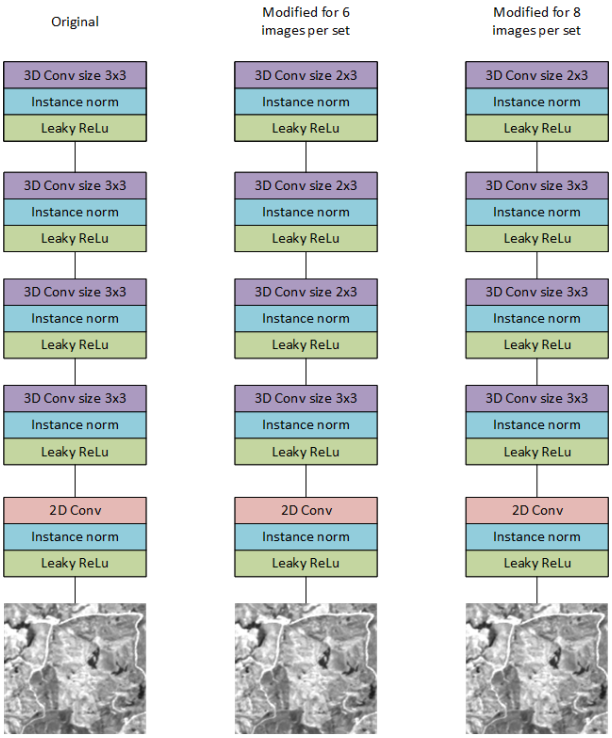


Figure 3.3: Fusion block modifications for different image set sizes

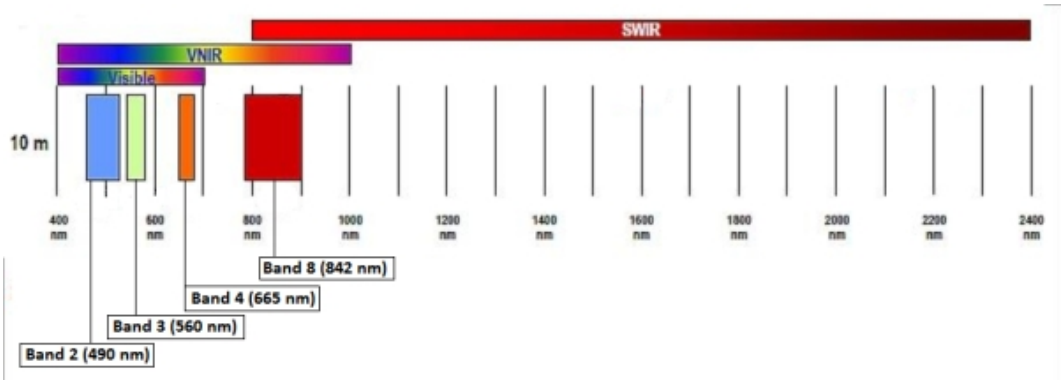


Figure 3.4: The Sentinel-2 satellite 10m spectral band [Tona et al., 2018]

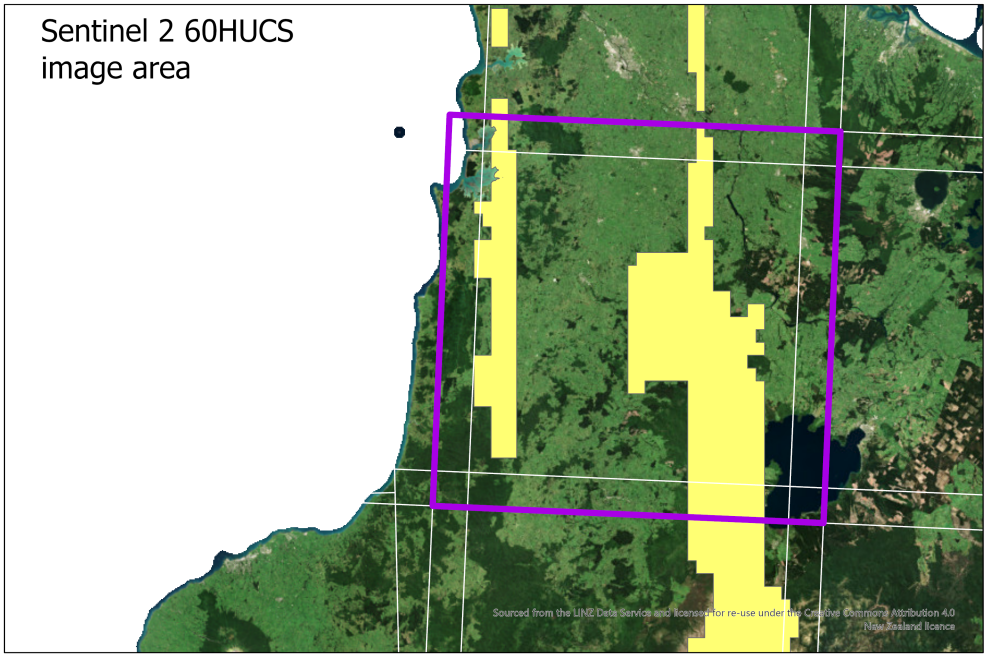


Figure 3.5: Intersection of Sentinel 2 and WRAPS data

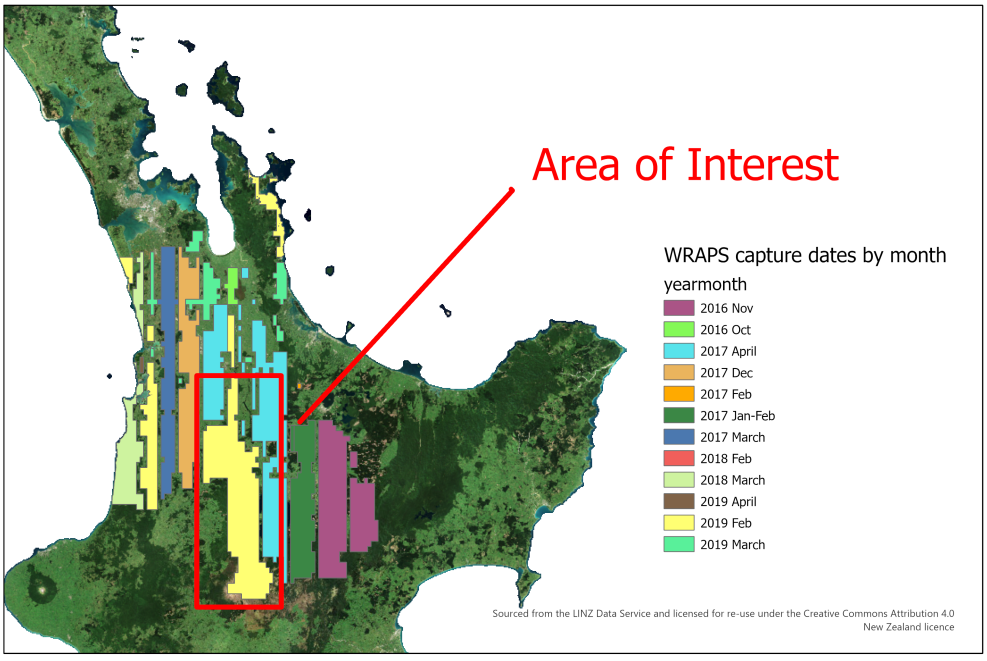
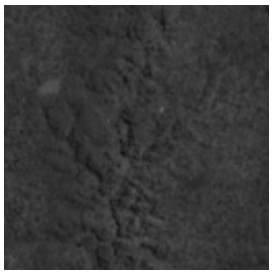
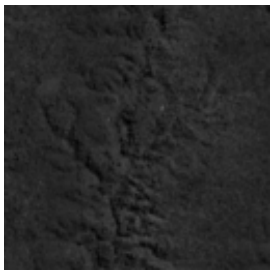


Figure 3.6: The WRAPS data captured with dates showing Area of Interest

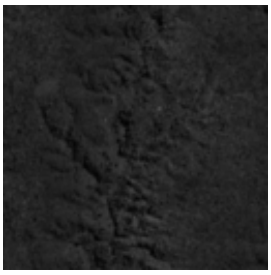
Figure 3.7: Example of cloud raster showing the area where part of an image has been removed. Cloud areas are black and non-clouded areas are white. In this case, the algorithm has missed some of the edges of wispy cloud, plausibly indicating that there could be some noise in the training data



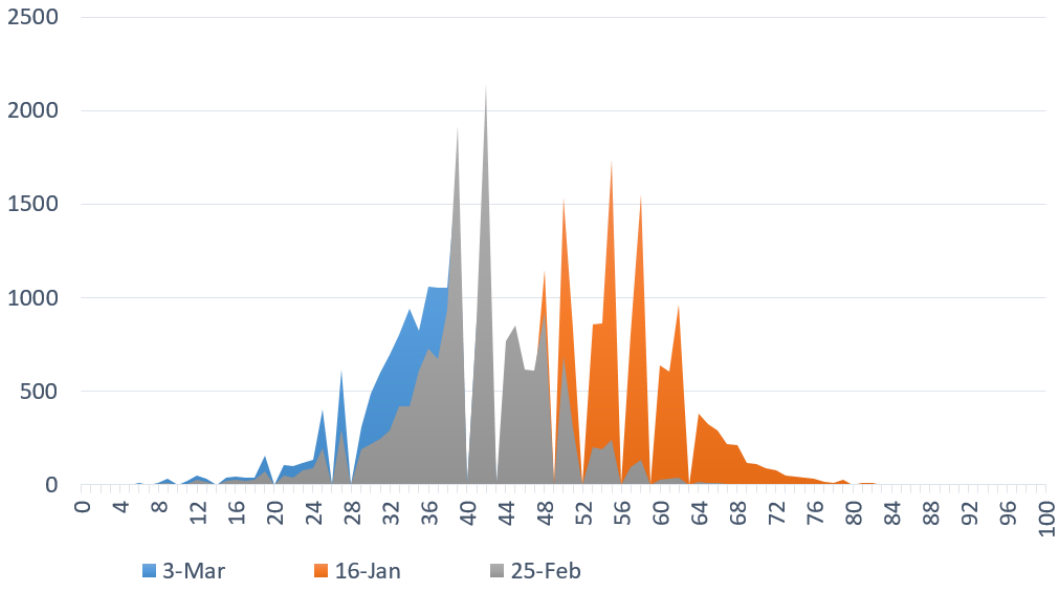
(a) patch 16 Jan



(b) patch 25 Feb



(c) patch 3 Mar



(d) histogram showing pixel spread for images

Figure 3.8: Images and pixel histogram: Three example images are shown of the red band from a single image set (out of a total of 8 possible images) to show how the images taken on different dates have a different spread of pixel values. The images shown are skewed towards to the lower end of the spectrum as they demonstrate mainly bush.

Chapter 4

Results

Experiments were carried out on the DeepSUM algorithm to find out whether using different data configurations, updating the loss function used, or making architectural changes to the algorithm improved the accuracy metrics of the model. Following this, an ESRGAN model was trained using both raw Sentinel data and DeepSUM outputs in an attempt to improve the perceptual quality of the output images.

Initial investigative work on DeepSUM was carried out using the red band of the imagery. Findings from the red band were applied to the blue and green bands.

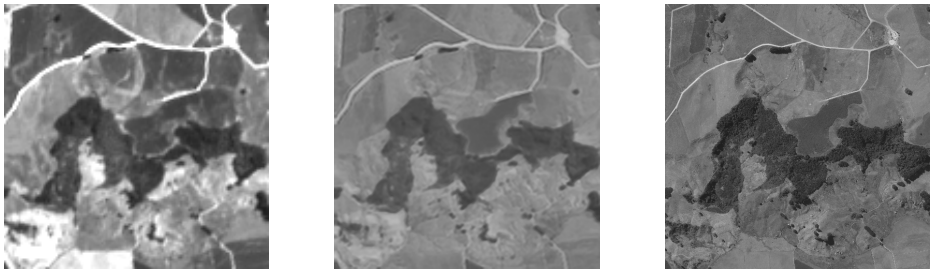
4.1 Data Split

Imagery was split into three categories to reflect the major land types found in the area as shown in Figure 4.1. These types are: farmland, bush, and mixed. The mixed land use class is any land where more than 25% and less than 75% of the area could be defined as bush.

(a) farmland (b) bush (c) mixed

Figure 4.1: Different land classes in study area

Results of running DeepSUM, shown in Table 4.1, show that the DeepSUM network performs vastly better than using a bicubic upsample to super-resolve data using both SSIM and PSNR metrics across all land types.



(a) original LR (b) network output (c) HR ground truth

Figure 4.2: Output of DeepSUM compared to ground truth and bicubically upsampled image using the red band

4.2 Data Transformation Effects

Various data transformation effects were tested to evaluate what works best in DeepSUM.

4.2.1 Image Standardisation

The effect of data standardisation can be seen in Table 4.1. While standardisation had a major effect on PSNR values, it did not affect SSIM to the same

	bicubic upsample		not standardised		standardised	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	15.1	0.27	17.1	0.32	**20.1**	**0.33**
farmland	13.5	0.38	18.1	**0.47**	**19.6**	0.46
bush	16.8	0.16	16.6	0.18	**20.8**	**0.2**
mixed	15.1	0.23	16.3	0.3	**19.8**	**0.31**

Table 4.1: Table showing effects of standardising the data on PSNR and SSIM values when compared to images upsampled using a bicubic upsampling

degree, indicating that contrary to what some other sources have reported, [Hore and Ziou, 2010], these metrics are to some degree uncoupled. It can also be seen that while bush has the best PSNR for both the bicubically upsampled images, and the images that have passed through the network, it also has a worse SSIM value. In a similar vein, the network has the least effect on improving PSNR and SSIM values for the bush land-use class compared to either farmland or mixed land-use.

The lower SSIM for the bush land-use class compared to farmland can be seen clearly illustrated in Figure 4.3, showing that bush has a lower SSIM in general than farmland.

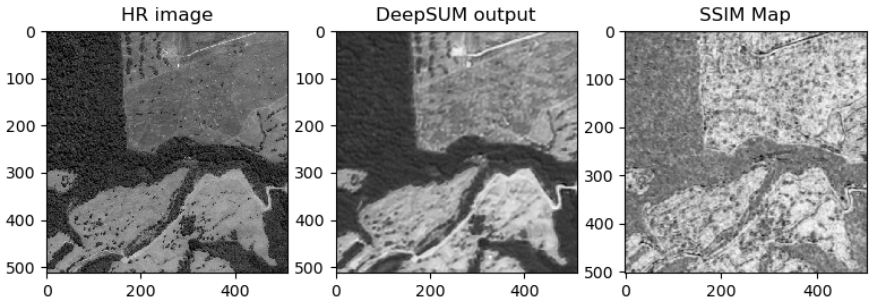


Figure 4.3: Difference in SSIM values for bush and farmland

Reasons for this are that bush has more pixel variation than farmland in the New Zealand context where open paddocks tend to be a uniform colour (see Section 4.4). A high level of pixel-by-pixel variation means that there is likely to be a low correlation between a pixel's value and its surrounding pixels. Adding to this, even two HR images taken at the same time could see differences in how bush appears as shadow effects, from the slight angle differences would cause pixel-wise differences in the image. This would not appear in farmland due to its more uniform nature.

4.2.2 Image Stretches

Various stretches were applied to the LR images to see if more information could be elucidated, as shown in Figure 4.4. It was hypothesized that expanding the data, particularly in the lower region of the spectrum, would allow the network to better show different patterns in bush or another of the land-use classes. In another experiment, a gamma correction of 0.5 was applied to the image. The gamma correction does not affect the black or white values in a raster dataset, but affects the contrast of the middle values of the data [McHugh, 2013]. Following each stretch, the data was standardised, as per the original dataset.

	standardised		stretch 1		stretch 2		gamma	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	**20.1**	**0.33**	18.8	0.26	19.8	0.33	15.0	0.27
farmland	**19.6**	**0.46**	18.0	0.38	19.6	0.45	11.3	0.35
bush	**20.8**	**0.2**	20.0	0.15	19.8	0.19	19.5	0.19
mixed	**19.8**	**0.31**	18.1	0.23	20.8	0.29	15.4	0.27

Table 4.2: Results of running DeepSUM using data stretched with different parameters

This hypothesize was disproven, as none of the stretched or gamma-corrected

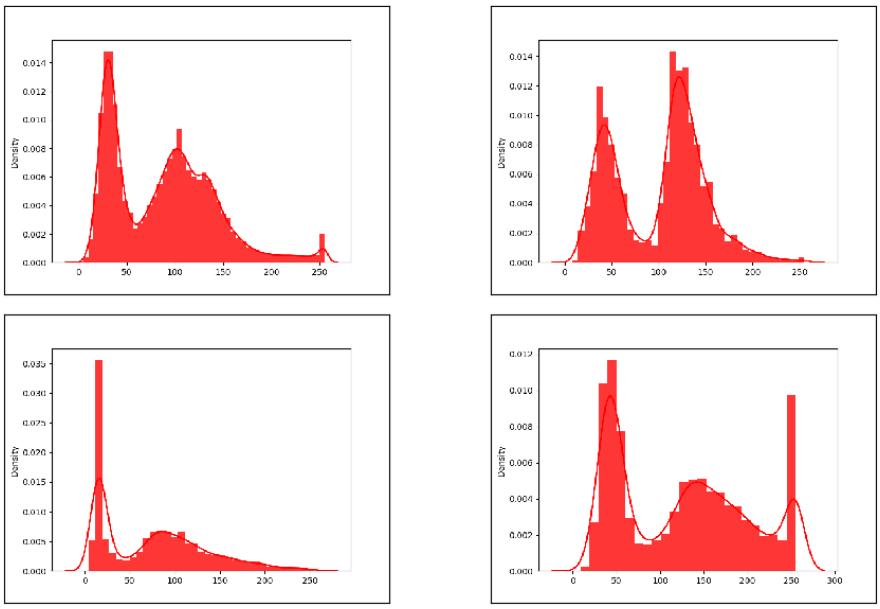


Figure 4.4: Histograms of different stretches applied to the red band from a single image from imageset 12. Left top: Original image, Right top: Stretch 1, Left bottom: Stretch 2, Left bottom: Gamma correction

data performed better than the standardised un-stretched data (see Table 4.2). In each instance, the un-stretched standardised data created a better model.

4.3 Data Augmentation

Use of augmented imagery can improve performance by increasing the size of the training set, and create variations of images that can improve the ability of the fit models to generalize. Imagery was augmented using the process described in Section 3.5.

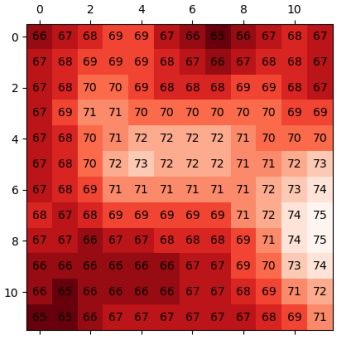
In this study, DeepSUM performed at a similar or slightly worse level when run with augmented imagery. In particular, the farmland land-use class did not super-resolve well. Reasons for this could be that the different texture and colours of the augmented HR source imagery caused the network to be less specific to mapping the Sentinel LR imagery to HR. With a different HR source, this mapping could be more generalised, and therefore weaker.

	6 images per set		8 images per set	
land use class	PSNR	SSIM	PSNR	SSIM
overall result	**20.1**	**0.33**	19.0	**0.33**
farmland	19.6	0.46	17.3	0.45
bush	**20.8**	**0.20**	20.8	0.19
mixed	19.8	**0.31**	19.6	**0.31**

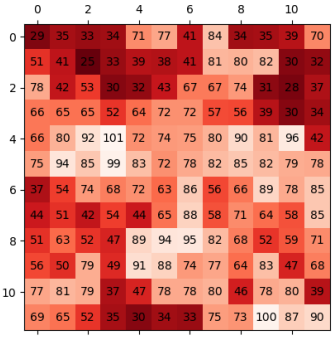
Table 4.3: Effect on DeepSUM of using two extra augmented HR images in the training data. The 8 image sets with 8 images per set included 2 or 3 augmented images.

4.4 Image Variation

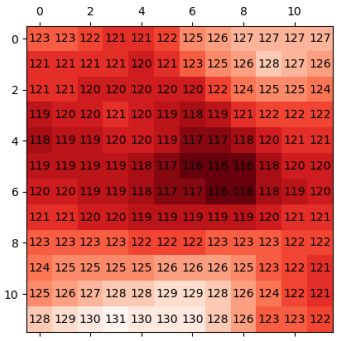
DeepSUM does not resolve the high level texture of the images well, particularly in the areas of bush. This effect is expected given the literature on pixel based loss functions (see Section 2.6). This can be seen in figure 4.5, where close inspection of an area of bush shows that the original HR image has a high level of contrast between adjacent pixels, where the texture of the photo could be described as doppled. In contrast, the LR output is a much smoother image and adjacent pixel values rarely vary by more than 2. This lack of variation will show up much more highly in the SSIM metric than PSNR, as SSIM takes in to account surrounding pixels when assigning a pixel value.



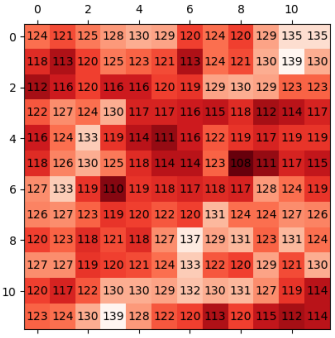
(a) DeepSUM output bush



(b) HR image bush



(c) DeepSUM output farmland



(d) HR image farmland

Figure 4.5: Pixel values for both ground truth HR image and DeepSUM output for equivalent areas of a selected area of bush and farmland. The bush shows greater variance than the farmland, with the result that DeepSUM performs better with regards to SSIM on farmland areas than on bush areas

.

To quantify this contrast, an algorithm was run which samples a 100 mini-patches of size 5 pixels by 5 pixels and 2 pixels by 2 pixels in each of an area of bush and an area of farmland from the same image patch. The sampling was carried out on both a ground truth HR image and the output of DeepSUM.

	High Res Image		Image output from DeepSUM (MSE)	
minipatch size	5 x 5 pixels	2 x 2 pixels	5 x 5 pixels	2 x 2 pixels
farmland	122.6	64.8	21.9	4.4
bush	184.7	91.4	15.8	2.5

Table 4.4: Variance differences between land-use types and ground truth and DeepSUM output

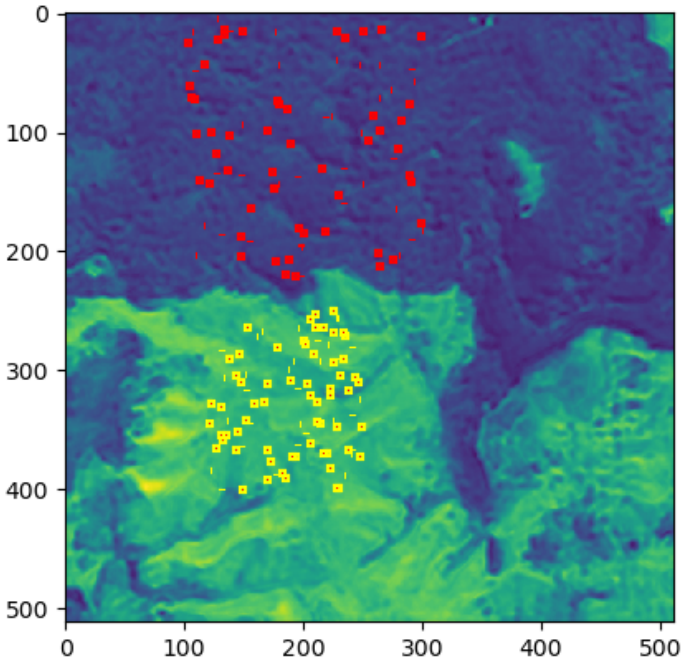


Figure 4.6: Location of mini-patches used to show differences in variance. The yellow dots represent a farmland land-use class mini-patch, whereas the red dots represent a bush land-use class patch

It can be seen in Table 4.4 that in both the larger 5 x 5 and smaller 2 x 2 mini-patches, bush have quantifiably more variance than farmland. The table also shows that DeepSUM does not successfully replicate the variance found in the original image in either land-use class, as in both cases the variance seen in the output image is far lower than the original HR image. However, contrary

to what is seen in the HR image, the DeepSUM output has a greater variance within farmland, and lower variance within the bush land-use class. In general, the algorithm models variance of farmland far better than the variance of bush. In fact the output image shows very little variance within the bush land-use class, i.e., a smooth unrealistic image is produced. This could go some way to explaining why in the derived output, the bush land-use class has a relatively high PSNR value, but a low SSIM compared to farmland.

It follows that DeepSUM struggles to replicate the variance of the ground truth. Apart from the use of the smoothing ℓ_1 or ℓ_2 loss functions, DeepSUM also employs a merge of the output of *SISR* blocks, in the fusion step, which averages the outputs. The nature of this mechanism ensures that the output data will converge on a mean value with less variation than the original image. SSIM and Multiscalar-SSIM loss takes information from surrounding pixels to estimate the value of a particular pixel. This too does not lend itself to replicating the variation of the ground truth image.

4.5 Effect of Merge Step

The final step that occurs in the DeepSUM algorithm is a merge of the output of the RegNet block, i.e., a set of single images, upscaled and passed through the network but not fused, and the output of the fusion step. It was hypothesized that this final step may negatively impact visual detail, by averaging detail between images in an imageset, to make the image less clear rather than improve detail.

Results listed in table 4.5 show that while using no merge function did not improve overall results, the results of the farmland land class were better when no merge function was used, whereas the bush and the mixed land-use class did not perform as well without the merge function.

	merge		no merge (ℓ_2)		no merge (SSIM)	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	**20.1**	**0.33**	19.1	**0.33**	18.8	**0.33**
farmland	19.6	0.46	20.3	0.45	**20.4**	**0.48**
bush	**20.8**	**0.20**	17.4	0.19	17.0	0.18
mixed	19.8	**0.31**	18.6	**0.31**	19.0	**0.31**

Table 4.5: Results of running DeepSUM with and with out the merge step using ℓ_2, and SSIM loss functions

4.6 Effect of Loss Functions

The original DeepSUM algorithm was trained using an MSE loss, however as noted previously, other loss functions that take in to account surrounding pixels could create an output better correlated with the HVS. In this section, the effect of pixel based loss functions including ℓ_1, ℓ_2, SSIM and MS-SSIM are explored, as well as two variants of a perceptual loss function.

4.6.1 Pixel Based Loss Functions

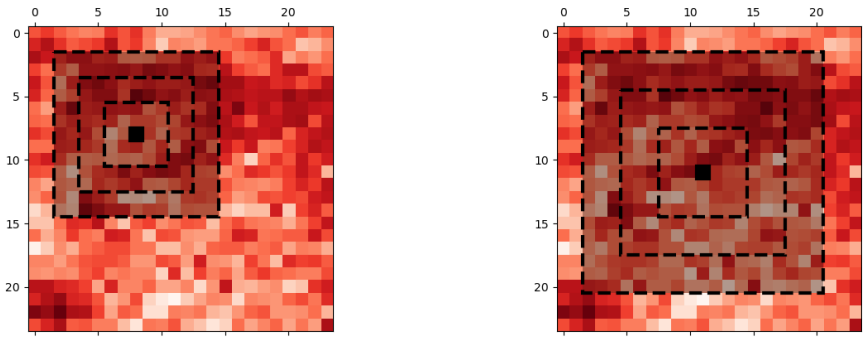
For each algorithm, training was carried out using the standardised dataset with 6 images per image set. A step size of 5×10^{-6} was used for six epochs in each case, as it was found that this was sufficient to converge. The SSIM function was run using the default parameters as used by [Wang et al., 2004] i.e., a filter of 11 and power factors of 0.0448, 0.2856, 0.3001, 0.2363, 0.1333. However, in the result below, a sigma value of 3 is used, as this was found to perform better than the default. The multiscalar-SSIM function was run using TensorFlow version 1.15, as this allows filter and sigma values to be updated from the defaults. When using multiscalar-SSIM, the image was downsampled three times. When this occurs on a 96 x 96 patch, the final scale level has too few pixels for the function to operate correctly using the default filter value of 11. Hence the multiscalar-SSIM was run with a filter size of 6 and sigma value

	MSE		L1		SSIM		MS-SSIM	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	20.1	0.33	20.1	0.26	**20.3**	**0.34**	20.1	0.33
farmland	19.6	0.46	**19.7**	0.45	**19.7**	**0.48**	**19.7**	0.46
bush	20.8	**0.20**	20.8	**0.20**	**20.9**	0.19	20.8	0.19
mixed	19.8	0.31	19.8	0.31	**21.0**	**0.32**	20.0	0.30

Table 4.6: Results from running DeepSUM using different loss functions

of 1.5.

Results in Table 4.6 show that there is some benefit from using different loss functions in DeepSUM, but this benefit is fairly minor when measured using pixel based measurements. The SSIM loss performs better than the other functions on all land-use classes with the exception of bush. There are subtle yet unexpected differences between the loss functions. One of these is the fact that the SSIM loss produces a worse SSIM metric on the bush land use type than the other loss functions. This is unexpected as the SSIM loss should be optimised for SSIM. It does however make sense when looking at the pixel values of the bush land-use class. As the SSIM metric is looking at the surrounding pixels, given a particular pixel, when the surrounding pixels have a high degree of variation, or change markedly in the spatial dimension, then this will potentially produce a smoother image that does not necessarily reflect the variance and texture seen in the original image. This idea is captured in Figure 4.7



(a) SSIM field of view of 2

(b) SSIM field of view of 3

Figure 4.7: Graphic showing the SSIM field of view with two different sigma values. Each line represents one standard deviation, so that pixels values 3 standard deviations away from a pixel still slightly affect that pixel

The SSIM loss function was run with a range of different sigma values, as it was hypothesized that using a different *field of view* for the SSIM function would have an influence on its performance. For example a smaller sigma value will only take account of pixels close to the pixel being looked at, whereas a larger value will take more account of pixels further away. The default value of 1.5 is often used as this worked well on the original data in the study by [Wang et al., 2004], however it is possible that different datasets could require different sigma values.

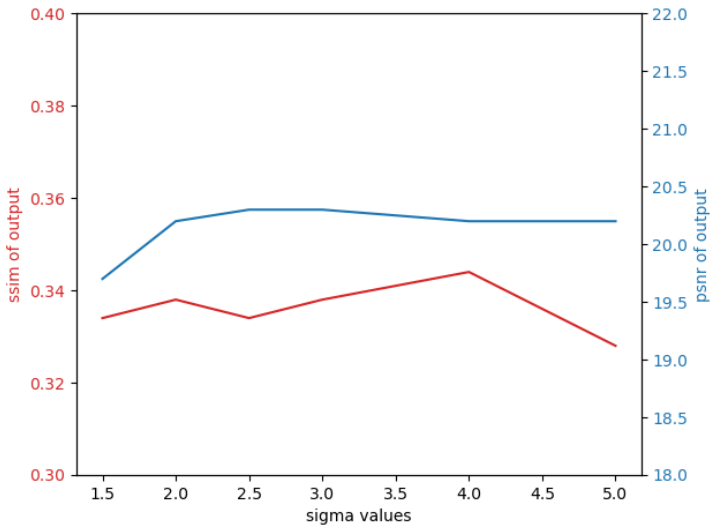


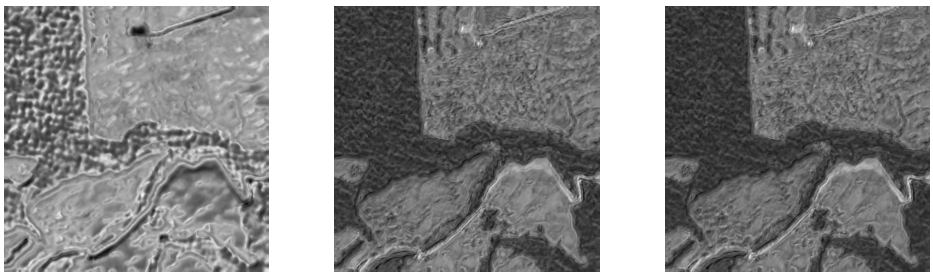
Figure 4.8: Graph showing performance of DeepSUM using an SSIM loss function with different sigma values

From 4.8, it is clear that a sigma value larger than 1.5 works better on the Sentinel 2 satellite data for SSIM.

4.7 Perception Loss Implementation

A perception loss function was created using VGG-19 and imagenet weights, and integrated into DeepSUM. The function compared the feature-map outputs of VGG-19 neural network for the DeepSUM output and ground truth, and from this returned a loss as described in Section 2.6.1.

As the data fed through DeepSUM was single-band imagery of 96 by 96 pixel patches, and the data expected by VGG-19 consists of 224 by 224 RGB images, as a preprocessing step, the loss function resized the input to fit VGG, and stacked three copies of the input data together to produce an RGB approximation. Deeper neural network layers were used for the content loss function, whereas layers from block 1 were used for the style loss function. The style loss function implemented a Gram Matrix equation. In the original perception loss function described by [Johnson et al., 2016], a total variation loss function is used as a smoothing function to reduce variation. In this study,



(a) block 5 conv 2 (b) block 4 conv 1 (c) block 5 conv 1,2

Figure 4.9: Output of DeepSUM using content loss from VGG19 with different convolutional layers as the loss layer

a total variational loss was not used, as spatial smoothness was not a desirable outcome.

As per the paper by [Johnson et al., 2016], a weighting parameter was used to weight style, content and total variational loss. In this study, a weighting factor of $W_content = 1 \times 10^5$), and $W_style = 1 \times 10^3$) were used. During training, this caused content loss to have a greater overall significance until content loss declined to the point at which style loss becomes more important.

Initially layers used in the perception loss function were the same as those used by [Johnson et al., 2016], i.e. content loss used block3conv3 and style loss used block1conv2, block2conv2, block3conv3, block4conv3 from VGG-19. Other combinations of content and style layers including a total variational loss were trialled, but output did not score well using PSNR and SSIM metrics nor was it perceptually very good.

Using content loss only, most of the layers higher in VGG19 in block 4 and 5, produced a strong structural output of the images. However, replicating the style of the original image was elusive using this methodology, as seen in Figure 4.9.

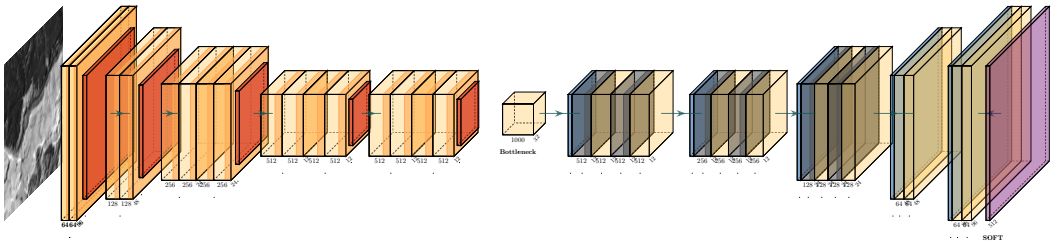


Figure 4.10: Architecture of the autoencoder used to create weights to use in a perception loss function. The autoencoder is an altered mirrored VGG network.

4.7.1 Perception Loss Using Weights from Aerial imagery

It was hypothesized that the perception loss function would work better when used with neural network weights trained on aerial imagery, specifically imagery similar to that found in the Waikato. A VGG network trained using imagenet data will contain feature maps with structures found in objects and photo scenes which are quite different to textures and content found in satellite imagery. Furthermore, networks trained on non-New Zealand satellite or aerial data will look for features found in cities or more built over countryside, rather than the bush and farmland as is found in the study area. With this in mind, an auto-encoder trained on WRAPs imagery, was created to generate feature-maps representative of those found in New Zealand satellite and aerial imagery.

The architecture of the auto-encoder used for the perceptual loss function was essentially a mirror of VGG-19, but, input was a single band image of 96 by 96 (see Figure 4.10). As in VGG, a pooling layer after each block reduced the size of each feature map by a factor of two, apart from block 4 where the feature map size was not reduced, and kept at 12 x 12. The bottleneck used for the autoencoder was a single dense layer with 1000 neurons. Training the autoencoder with the standard mirrored VGG configuration proved to be difficult, so regularisation was used to improve the training process and to

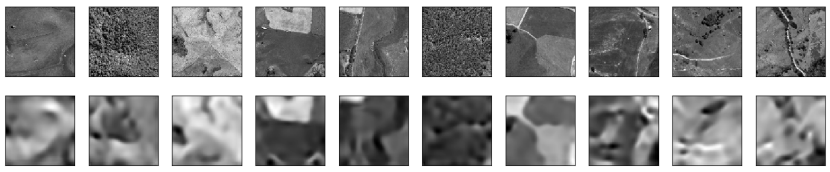


Figure 4.11: Sample of input data (top) and output data (bottom) of autoencoder showing how the basic structural information of the image is preserved. Loss of textural information appears to occur in the process.

help created stronger internal activations. Several regularisation factors were used in the dense layer in the autoencoder bottleneck including both ℓ_1 and ℓ_2 kernel regularisation with a regularisation factor of 0.01 and 0.1, bias and activity regularisation, both with a regularisation factor of 0.01. The standard relu activation function was used, with sigmoid activation used in the final layer. Potentially leaky relu would have helped improve training, however this functionality is not supported by keras at TensorFlow 1.13 (DeepSUM uses this TensorFlow version). Use of batch normalisation after each encode and each decode block was found to improve autoencoder performance, measure by reconstruction error. Batch normalisation was not used however, as it was found that feature maps produced in this way did not work as well in perception loss function. [Wang et al., 2018] too found that when batch normalisation was used, the network did not generalise as well. Input and output of the autoencoder is shown in 4.11.

The autoencoder was built in keras, and trained on 40,000 image patches taken from HR WRAPs aerial imagery of the study area, 4000 of which were used as a validation set. The network was trained for 9 epochs using a mini-batch size of 100.

In general, reducing the depth of the autoencoder, resulted in a better output, but, as the goal was to create a set of layer weights that could adequately encapsulate perceptual and structural information found in the aerial imagery, a wider, deeper autoencoder was used.

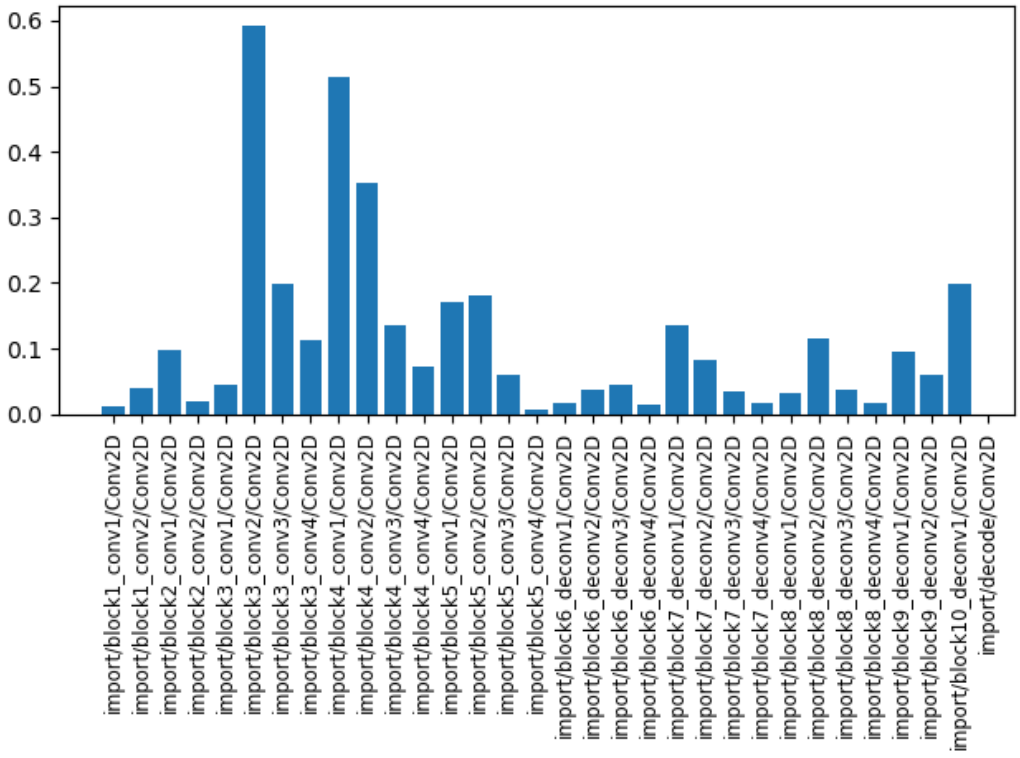
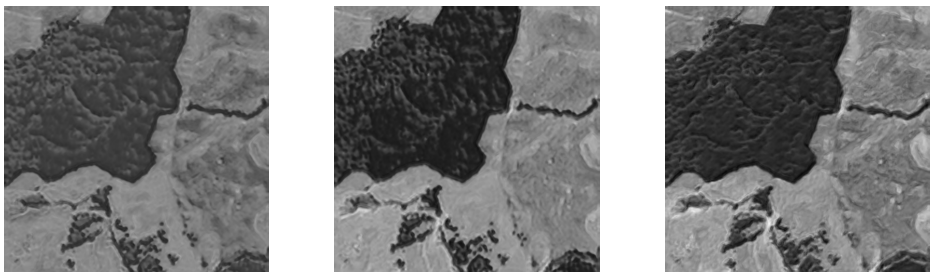


Figure 4.12: Graph showing correlation between the gram matrix of feature maps

In order to ascertain which layers best encapsulated the differences between styles, feature map outputs from bush imagery and from farmland imagery were compared. Twenty sample images of each of the bush land use class, and the farmland land use class were selected from from the autoencoder dataset. The gram matrix of outputs of each of the 20 pairs of bush and farmland imagery were compared using the Pearsons Correlation coefficient to find out which feature maps correlate closely between the land-use types indicating which layers respond to bush and farmland differently. A number close to 1 indicates a perfect degree of correlation, whereas a number close to 0 indicates that the layer responds differently to the different land-use types. This is shown in Figure 4.12.

From this it can be seen that useful layers to pick for use in the style loss function from the autoencoder were blocks 1 and 2. In these layers, correlation between the feature maps from different land-use classes is low. Conversely,



(a) Content Loss only

(b) Content and style loss A

(c) Content and style loss B

Figure 4.13: Output of DeepSUM using perceptual loss function with autoencoder weights. Different content and style layers were tested as the loss layers

layers in block 3 were found to correlate more strongly between land-use classes and so were less likely to encapsulate stylistic difference.

Several different combinations of style loss layers and content loss layers were tested. The combinations tested were:

1. Content Loss only: Using content layers block1-conv1, block1-conv2, block5-conv1, block5-conv2

2. Content and style loss A: Using content layers block1-conv1, block1-conv2, block5-conv1, block5-conv2, and style loss from layer block2-conv2, block4-conv4, block2-conv1

3. Content and style loss B: Using content layers block1-conv1, block1-conv2, block4-conv4, block4-conv3, and style loss from layer block1-conv1, block2-conv1

DeepSUM was trained using a perception loss function using each of the above layers from the autoencoder in the style and content loss functions. As per perception loss using the VGG network, a total variational loss function was not used, so as to maintain some of the texture of the imagery. Results are shown in Table 4.7 with example image outputs from DeepSUM in Figure 4.13.

	bicubic upsample		content loss only		content style loss A		content style loss B	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	15.1	0.27	**18.1**	**0.32**	17.7	**0.32**	17.7	0.31
farmland	13.5	0.38	**19.3**	**0.44**	18.5	**0.44**	18.7	0.43
bush	16.8	0.16	16.5	**0.17**	**16.6**	**0.17**	16.5	**0.17**
mixed	15.1	0.23	**18.1**	**0.30**	18.2	0.29	17.5	0.28

Table 4.7: Results of running DeepSUM with a perception loss function using different content and style layers from the autoencoder

Results on test data show that although a perception loss function performs better than bicubically upsampling images, it does not perform (using standard PSNR and SSIM metrics) to the same accuracy as pixel based loss methods. To see this compare results in Table 4.7 with Table 4.6. Perhaps unsurprisingly, the best performing loss function using these metrics did not include the style loss function. This follows the perception-distortion curve described in Chapter 2, where improving perception has a negative effected on distortion. In general the perception loss function created more sharply defined features and stronger colours and textures than the pixel based loss methods.

4.8 Variation Loss

As described above, the super-resolved images from DeepSUM using a range of loss functions often produced a smooth output as pixel values trend towards the mean. In particular bush appears as a monochrome, and lacks the texture of the ground truth. As discussed in 2.7, the best possible perceptual qualities occur when the output of an algorithm follow the natural distribution of an image.

In an attempt to better replicate the variation and, by proxy, the perceptual qualities of the ground truth, a variation-based loss function was created to

encourage pixel-by-pixel variance with the intuition that this would lead to an improved texture on the standard loss functions, and appearance more similar to the ground truth HR images. It should be noted that although variation type loss functions such as total variation loss are commonly used, these functions are designed to reduce variation, and smooth the image. In this study, the variation loss function attempts to preserve the variation found in an area, and as this is dependent on spatial differences in variation, this will vary throughout the image. A loss function encouraging variation in this way has not been found in the literature.

The mechanism by which variation loss was achieved was as follows: For each mini-batch of output imagery and ground truth, nine copies of the mini-batch were created, and each of those nine copies was moved by each of [-1,0,1] in the x-dimension and y-dimension. To account for border effects, the matrices were cropped at the border by a border parameter. This was the same parameter as used by DeepSUM to crop image borders. Following this, the off centre copies of the original mini-batch (and the images which were not moved or moved by (0,0)) were stacked together and a variance matrix created using only the stack dimension (see Figure 2.7). This effectively created a mini-batch of image variances. The output imagery and the prediction imagery were compared, and the difference calculated using both mean squared error and absolute error. The variance loss was combined with other loss functions using a *variance factor* hyper-parameter to weight its effect on the overall loss.

To define variance loss mathematically, if N is the number of samples and $x = (x_i | i = 1, 2, \cdots, N)$ is the predicted output and $y = (y_i | i = 1, 2, \cdots, N)$ is the ground truth, then V_x is the variance of output and V_y is the variance of the ground truth HR image. Variance loss can be defined by the equation:

$$L_{var}(x, y) = \frac{1}{N} \sum_{i=1}^{N} (V_{xi} - V_{yi})^2 \tag{4.1}$$

Using this definition, and using L_{MSE} loss as the pixel based loss function, and using α_V as the weighting of the variance loss or the *variance factor* the total

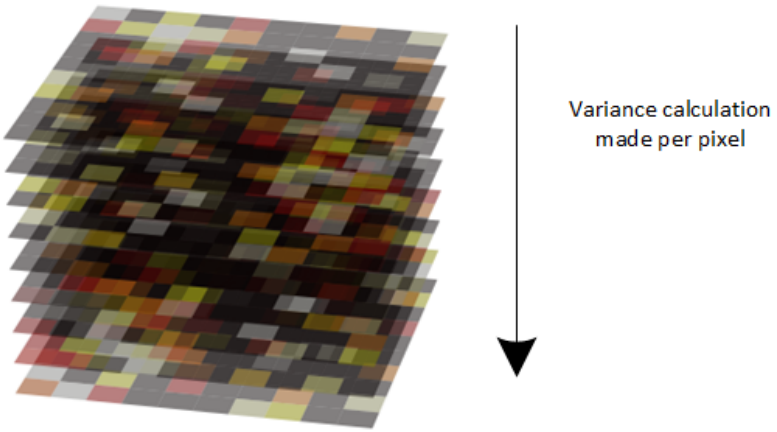


Figure 4.14: Variance is captured by stacking a copy of each image on top of each other, move each image by 1 or 2 pixels in an X and Y direction, and calculating the variance through the z axis.

loss, $L(x, y)$, is defined by the equation:

$$L(x, y) = L_{MSE}(x, y) + \alpha_V(L_{var}(x, y)) \tag{4.2}$$

As discussed, variance loss was calculated using an ℓ_1 as or an ℓ_2 loss. Naturally, different weightings were appropriate when used with the different loss functions. In a twist on the basic concept where the 8 pixels surrounding a pixel were used to calculate variance, a wider variance was calculated using the two pixels on either side of a particular pixel, i.e. calculating variance from the surrounding 24 pixels.

Results from using a variance loss component in the loss function showed an improvement on output quality in two ways. Land cover texture appeared more realistic and similar to the ground truth HR image, and less of a monolithic colour. This effect can simply be explained by the variance loss serving to somewhat recreate the pixel-by-pixel variation seen in the ground truth.

A second more surprising effect was that boundaries between features were

more detailed with finer texture and sharper edges when a variance loss component was used. This effect can be explained by the variance loss working to preserve high variance in boundary areas by forcing the high pixel values higher and the low pixel values lower, and so enhancing the edge effect, that is otherwise blurred by an MSE loss. The effect is the same as seen when a tractor drives over a muddy track, and existing ruts are deepened, but flat areas remain flat. In the HR ground truth image, boundary areas will naturally have a high variance as different features have different pixel intensities. Without the variance loss, pixels are more likely to fall in to a grey zone between the highs and lows creating a soft edge effect.

A low variance factor reducing the weighting of variance on the total loss seemed to work best when measured against pixel based measurements. When the α_V was increased to increase the weighting of variance loss, a less visually pleasing *Van Gogh* like texture appeared where bush in particular appeared slightly blotchy and less real. When DeepSUM with a high variance factor was trained using all three RGB bands the image had a very low LPIPs reading, making this perceptually almost as good as an ESRGAN output (See Section 4.14)

In general, adding a low amount of the variance loss factor markedly improved the image quality in a human visual sense (see Figure 4.15), however improved PSNR and SSIM values only slightly (see Figure 4.16). Using the MSE loss and a 9-pixel variance, the best variance weighting was 0.3. The lack of improvement in PSNR, and SSIM metrics is expected, as unlike human perception, these metrics do not favour replicating imagery texture and variance. Any improvement in image sharpness will not necessarily appear in these metrics either for the same reason.

It can be seen from the tensor board of the training run using MSE and variance loss that the variance loss makes up only a small portion of the overall loss function, i.e., between 2 to 10% of the overall loss when using the $variancefactor = 0.3$ option. While pixel-based losses declines quickly then

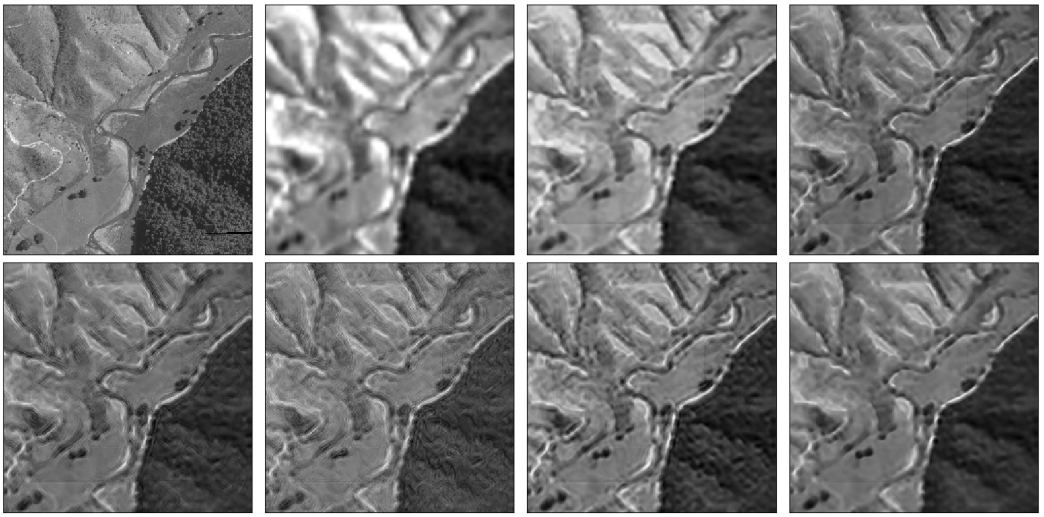


Figure 4.15: Images showing the effect of the variance loss on image quality. Zoomed in samples taken from top right of test image-52. Top-left to bottom-right: HR/ground-truth image, LR image bicubically upsampled, output using MSE loss only - no variance loss, output using MSE loss inc variance loss (0.3) BEST, output using MSE loss inc variance loss (0.6), output using MSE loss inc variance loss (1.0), output using MSE loss inc variance loss (0.3) with a 25 pixel sample size, output using L1 loss inc variance loss (0.1)

flattens off, the variance loss rises quickly, with the reduction in overall loss, then declines slowly after that, as see in Figure 4.17. From this, it can be surmised that using the variance loss function serves primarily to preserve the original variance of the imagery, which would otherwise be smoothed off by the MSE loss function.

Variance loss was tested as part of the overall loss component using both ℓ_1 and ℓ_2 losses with both a 9 pixel and 25 pixel sampling component. Overall, it had a relatively minor positive effect on PSNR and SSIM metrics, with the best result obtained using an ℓ_2 loss with a 9 pixel sampling component, and a variance factor of 0.3 (see Table 4.8). Of the three land classes, only bush performed worse when using an variance loss component, although this was visually more pleasing.

The effect of the different variance factors on pixel variance was explored

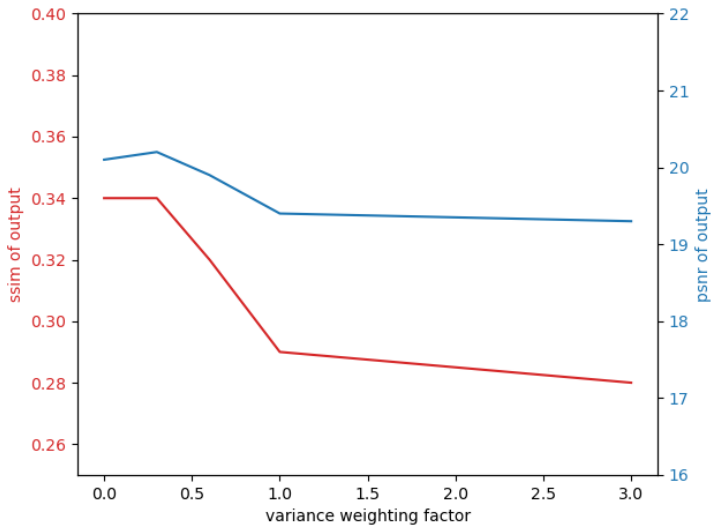


Figure 4.16: Graph showing the effect of variance factor weights on PSNR and SSIM of output image using an MSE loss. Variance was calculated using 9 pixels

(see Table 4.9). As expected, as the variance factor is increased, and the loss function is tilted towards increasing the pixel variance, the resulting output image has a higher variance. However, even using the highest variance factor, the output image still has a lower variance than the corresponding HR image.

DeepSUM outputs created using variance loss function with a high variance loss factor were used to train ESRGAN with the intention of producing super-resolved images with stronger edges and textures (see Section 4.13.1).

4.9 Ablation Studies

The SISRNet subnetwork consists of 8 blocks each containing a single 2D convolution, instance normalisation followed by leaky relu activation. Convolutional blocks were removed and added to the subnetwork to assess the effect of a deeper or shallower network on the final output. These changes were tested using the normalised red band data with an MSE loss. In order to test differences, in each instance, pretraining data had to be re-created as the network architecture changes meant that the original pre-training data could

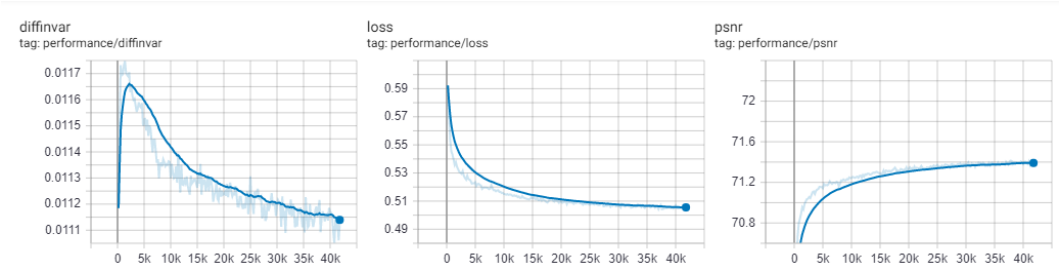


Figure 4.17: Tensorboard showing validation data performance on a training run. Loss shown are Variance Loss (diffinvar) and overall loss which includes MSE loss and variance loss. PSNR here is directly correlated to MSE loss. The graph shows both the smoothed curve (darker) and the un-smoother loss (lighter)

	L2 using 9px var loss		L2 using 25px var loss		L1 using 9px var loss		L1 using 25px var loss		Benchmark: MSE loss only	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	**20.2**	**0.34**	19.9	0.33	19.9	0.33	20.0	0.33	20.1	0.33
farmland	**19.7**	**0.47**	19.4	**0.47**	19.4	**0.47**	19.6	**0.47**	19.6	0.46
bush	20.7	0.18	20.3	0.17	20.3	0.17	20.5	0.17	**20.8**	**0.20**
mixed	**20.7**	**0.31**	20.5	0.30	20.5	0.30	20.7	0.30	19.8	**0.31**

Table 4.8: Effect of a variance loss component in an on PSNR and SSIM metric when ℓ_1 and ℓ_2 loss are used

not be used. To create the pre-train weights, the SISRNet subnetwork was run against ground truth data for 2 epochs prior to training the entire network.

In general, it was found that more convolutional blocks caused very little change to the accuracy of DeepSUM with a slight increase in PSNR with more blocks and a decrease in SSIM, as shown in Table 4.18.

Different Fusion Net block architectures of DeepSUM were also tested. In this subnetwork, feature-maps from each of the individual images are combined to create a single image via a series of 3D convolutions followed by instance normalisation. The existing structure was made deeper via more smaller convolutions, and shallower by using fewer larger convolutions. It was found that a

	no var factor	var factor 0.3	var factor 0.6	var factor 1.0	var factor 3.0	Benchmark: L2 loss no var loss
farmland	4.4	5.9	17.7	38.6	59.4	64.8
bush	2.5	2.8	20.9	38.4	72.0	91.4

Table 4.9: Effect of a variance loss component weighting (i.e. variance factor) on pixel variance in images

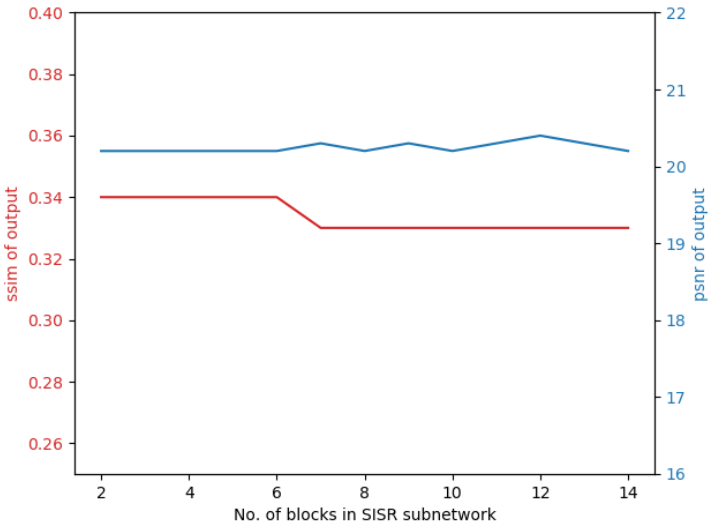


Figure 4.18: Effect of number of convolutional blocks on in SISR subnetwork on overall network

structure containing three convolution worked best, but the difference between the different architectures was minor, as shown in Table 4.10.

4.10 Other Colour Bands

Although Sentinel imagery contains 11 bands, only the visible bands (red, blue, green) were studied in this thesis, as good quality HR training data does not exist for other bands. DeepSUM was run using the blue and green band data. Pre-training data for SISR subnetwork was created specifically for each band, but it was found that this did not perform better than using pre-training data from the red band of the network.

Fusion architecture	PSNR	SSIM
2,2,2,2,2	20.3	0.33
2,2,2,3 (original)	20.2	0.33
3,3,2	20.3	0.34
4,3	20.2	0.33

Table 4.10: Effect of different fusion architectures on DeepSUM accuracy

4.10

	Red band		Blue band		Green band	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	19.1	0.27	23.0	0.41	19.9	0.27
farmland	17.7	0.45	20.9	0.51	20.4	0.47
bush	19.8	0.17	24.0	0.35	19.5	0.32
mixed	18.8	0.32	22.7	0.44	20.3	0.31

Table 4.11: Accuracy of different spectral bands trained with DeepSUM using an MSE loss

Results from the DeepSUM running the MSE loss function and the SSIM (best performing on the red band) were tested, as shown in Tables 4.11 and 4.12.

It was found that DeepSUM with SSIM performed better when using an SSIM loss, with the exception of PSNR values on the blue band.

4.11 Colour Adjustment

Histogram matching with a look-up dictionary of images was used to adjust the colour values of the outputs to match the aerial image ground truth using methodology outlined in Section 3.8. The output of DeepSUM already applies the mean and standard deviation from a configuration file to the data, but

	Red band		Blue band		Green band	
land use class	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
overall result	19.2	0.29	22.8	0.41	20.0	0.28
farmland	18.1	0.46	21.4	0.52	20.6	0.43
bush	20.0	0.18	23.9	0.34	19.7	0.20
mixed	18.8	0.34	22.3	0.43	19.5	0.29

Table 4.12: Accuracy of different spectral bands trained with DeepSUM using an SSIM loss

histogram matching perceptually improves on this result, and makes images more comparable to each other.

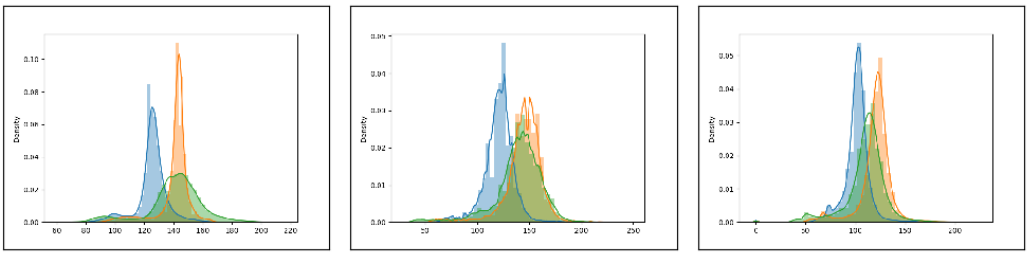


Figure 4.19: Example of colour adjustment showing Left: red, green and blue histograms of original output of DeepSUM, Mid: histogram of ground truth HR image, Right: Histogram of output after colour adjustment

It can be seen in Figure 4.20 how the colour adjustment process moves the spectral characteristics of an image towards that found in the HR ground truth, and so makes the image perceptually more similar.

4.12 ESRGAN

ESRGAN was run with the aim of comparing to the results of DeepSUM, and enhancing the results of DeepSUM. With this aim, an implementation of ESRGAN was used to super-resolved raw Sentinel data to compare this to the output of DeepSUM. In a second experiment, ESRGAN was used to enhance the output of DeepSUM to make it more photo-realistic, without further increasing the resolution. As described above, the output of DeepSUM lacks the pixel-by-pixel variation of the ground truth images, and some of the sharpness and detail is not recreated by the network particularly when an MSE loss is used.

4.12.1 Down-sampled WRAPs Data

As an initial proof of concept, 10,000 patches of WRAPs data (the HR data used to train DeepSUM) were down-sampled using nearest neighbour. A model was trained using $6x10^5$ steps, and inference carried out after each $2x10^5$ steps. The model successfully created the look and feel of the original image, but also changed the appearance of features and created features not present in the original image, as shown in Figure 4.21.

Figure 4.21: Output from ESRGAN on down-sampled HR image. Top left: Down-sampled image, Top right: GAN output after 200,000 steps, Bottom left: GAN output after 400,000 steps, Bottom right: ground truth. It can be seen that the model easily finds the high-level style of the data, but fails to replicate some of the fine level details, or incorrectly recreates detail. For example, the paddock boundary in the bottom left of the original image is square shaped, whereas the GAN recreates this as rounded, and fails to recreate some boundaries.

4.12.2 ESRGAN Trained Using Sentinel 2 Imagery

LR Sentinel 2 data was super-resolved in a similar fashion to the down-sampled HR data with the difference that each HR image was matched with up to 8

separate Sentinel 2 images, as each imageset used in DeepSUM contained 8 temporal images. Only cloud-free Sentinel 2 images were used (which excludes about a third of the available images), as unlike in DeepSUM there was no way of replacing clouded or shadowed areas with patches of image. Up to eight temporally different Sentinel 2 images were available from each image set, but not all of these were cloud free. Therefore this process yielded many more training patches than the DeepSUM output (around 30 000).

4.12.3 ESRGAN Trained Using DeepSUM Output

In a final step, output images from DeepSUM were paired with the original HR data to train an ESRGAN model. In the original ESRGAN network, an upsampling step occurs after a sequence of Resnet blocks that increases the resolution by a factor of four. As DeepSUM outputs images of 512 x 512 (the same size as the HR data), this upsampling was no longer necessary so was removed. To run samples of a higher resolution through the network, the batch size needed to be reduced from 16 to 2 to prevent out of memory errors. ESRGAN was run with 16 ResNet blocks for 2 000 000 steps, with the learning rate halved every 200 000 steps. This took approximately two weeks to train on a GTX 1080 GPU with 12 GB on board memory.

In order to create enough training data for ESRGAN, 6000 image sets were created to create approximately 5500 DeepSUM output images (using an MSE loss). These output images were used to train ESRGAN. It was originally assumed that the ESRGAN training weights from training using DeepSUM with an MSE loss would be sufficient to test other DeepSUM outputs. However, doing this was found to produced a blurred result, and it was necessary to generate DeepSUM outputs for each loss function and train ESRGAN separately. Due to the length of time required to both create large numbers of DeepSUM output images, and train ESRGAN, it was not possible to train ESRGAN on DeepSUM outputs with every different loss functions.

As per the DeepSUM output, colour adjustment was carried out on the

	bicubic upsample			DeepSUM only			ESRGAN only			DeepSUM then ESRGAN		
land use class	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
overall result	19.1	0.30	0.34	**20.1**	**0.31**	0.35	19.5	0.22	0.21	19.7	0.25	**0.20**
farmland	18.1	0.44	0.25	**20.1**	**0.46**	0.25	19.3	0.30	0.16	19.7	0.40	**0.15**
bush	19.8	0.23	0.38	**20.1**	**0.24**	0.40	19.6	0.18	0.24	19.6	0.19	**0.22**
mixed	18.1	0.38	0.30	**20.2**	**0.38**	0.30	19.8	0.27	0.17	19.8	0.30	**0.16**

Table 4.13: The effect of various steps on the on accuracy and perceptual metrics

ESRGAN output to align colour values with those from the ground truth.

Both low level metrics (PSNR and SSIM) and perceptual metrics (LPIPs) were used to compare output images. In the Table 4.13, all images have been colour adjusted to be similar to the ground truth.

Results show that as expected DeepSUM has the best PSNR and SSIM results, but using a combination of DeepSUM and ESRGAN gives the best LPIPS perceptual measure. The combination result is also better than ESRGAN alone for PSNR and SSIM values, with SSIM in particular showing much higher values than using ESRGAN alone.

Using ESRGAN alone created several obvious artefact. For example in the zoomed in views in Figure 4.23, the third image from the left shows a track, whereas the combination product and the HR image does not. The combination product is not immune to possible artefacts, as both the GAN output and combination output show the track disappearing (right image in 4.23), whereas the HR image does not.

4.13 Network Interpolation

In order to see the perception distortion trade-off, images and weights were interpolated. As described in Section 4.12 both intermediate points were found using a set of alpha values. Result can be seen in Figure 4.25 where the image is more or less perceptually driven.

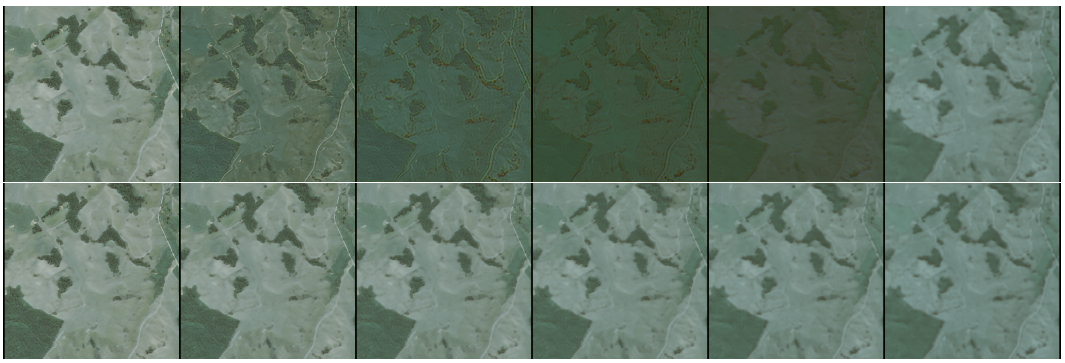


Figure 4.25: Top: Network interpolation, Bottom: Image interpolation. In both cases a single image was interpolated using alpha values of (1.0,0.8,0.6,0.4,0.2,0), so far left image is more perceptually driven and far right image is more PSNR oriented. Note that the images are shown before colour adjustment was used.

4.13.1 ESRGAN Trained Using DeepSUM Output with Variation Loss

It was hypothesised that using the DeepSUM output from the variation loss function as an input to ESRGAN would produce a better result than using DeepSUM with the MSE loss. The intuition behind this was that the variation loss function produces a DeepSUM result with enhanced contrasts at boundaries, and deeper if somewhat contrived textures. Passing this imagery through the GAN may allow the end to end process to show finer features, and elucidate enhanced textures. In this instance, a high variation loss factor of 3 was used, as though this created some artefacts and unnatural looking textures, it provided a higher level of contrast than a lower variation loss factor. It was hypothesised that ESRGAN would be able to overcome artefacts.

Initially DeepSUM with variation loss outputs were inferred using the ESRGAN model trained using the DeepSUM with MSE loss data. This approached failed to produce an adequate result, as the resulting images were were not an improvement on the process used in Section 4.12.3. To overcome this effect, an ESRGAN model was trained using DeepSUM with variation loss outputs.

Due to the time taken for the end-to-end process, only 4000 image pairs were used in training.

As can be seen in Figure 4.26 both the ESRGAN product and DeepSUM with variation loss produce cleaner lines and better definitions than using the same with the MSE loss only. This can be seen in the boundary between bush and paddock, and also in the definition of the track. In many examples of the ESRGAN output with the MSE loss roads and tracks become smoothed over, and do not re-appear in the ESRGAN output. This effect is greatly reduced when using DeepSUM with the variation loss. The boundary between bush and paddock is much cleaner and more natural in ESRGAN product trained with DeepSUM with variation loss output.

The DeepSUM output using variation loss (without passing through ESRGAN) has an LPIPs value almost as good as the GAN outputs (see Table 4.14). But, despite subjectively appearing perceptually better, the ESRGAN output from DeepSUM with variation loss does not produce better PSNR or SSIM or LPIPs metrics. Rather than suggesting that the DeepSUM with variation loss is perceptually inferior, it suggests that these metrics do not do adequately match human perception.

	bicubic upsample			DeepSUM only			DeepSUM then ESRGAN		
DeepSUM loss func	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
MSE loss	19.1	0.30	0.34	20.1	0.31	0.35	19.7	0.25	0.20
Var loss				19.0	0.23	0.23	19.0	0.21	0.21
SSIM loss*				**20.3**	**0.33**	0.33	18.1	0.19	0.26
Perception Loss				17.6	0.32	0.33	19.4	0.21	**0.19**

Table 4.14: The effect of loss function on DeepSUM output and ESRGAN output. In each case ESRGAN was trained on the DeepSUM output inferred with weights from the different loss function. *Note that ERSGAN was not specifically trained on the DeepSUM output from the SSIM loss, but rather used the training weights from training DeepSUM with an MSE loss. This could account for the lower SSIM and PSNR on ESRGAN output values when DeepSUM with SSIM loss is used

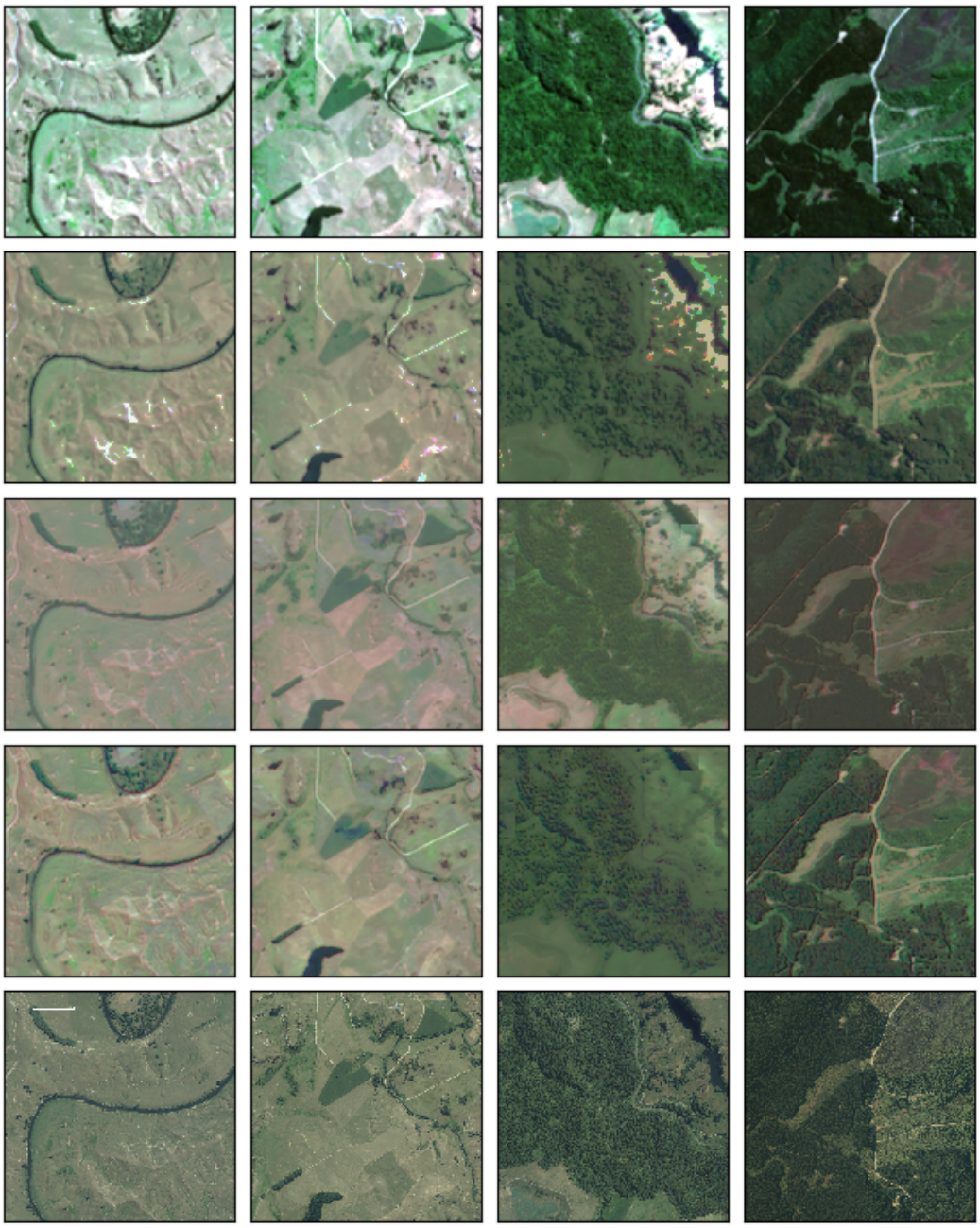


Figure 4.20: Output from four test samples showing colour adjust on bicubically upsampled LR data and DeepSUM output process. Top image: bicubic upsample with no adjustment, Second row images: bicubic upsample colour adjusted, Third row images: Output of DeepSUM (mse loss) with no adjustment, Fourth row images: Output of DeepSUM (mse loss) colour adjusted, Bottom row images: Ground truth HR data

Figure 4.22: Output from four test samples showing different stages of DeepSUM and ESRGAN process. Top image: Original Sentinel 2 image patch bicubically upsampled by a factor of 4 to be 512x512 pixels, Second row images: Output of ESRGAN working directly on a single Sentinel 2 image, Third row images: Output of DeepSUM (mse loss) using all three bands, Fourth row images: Output of DeepSUM passed through ESRGAN without upsampling, Bottom row images: Ground truth HR data

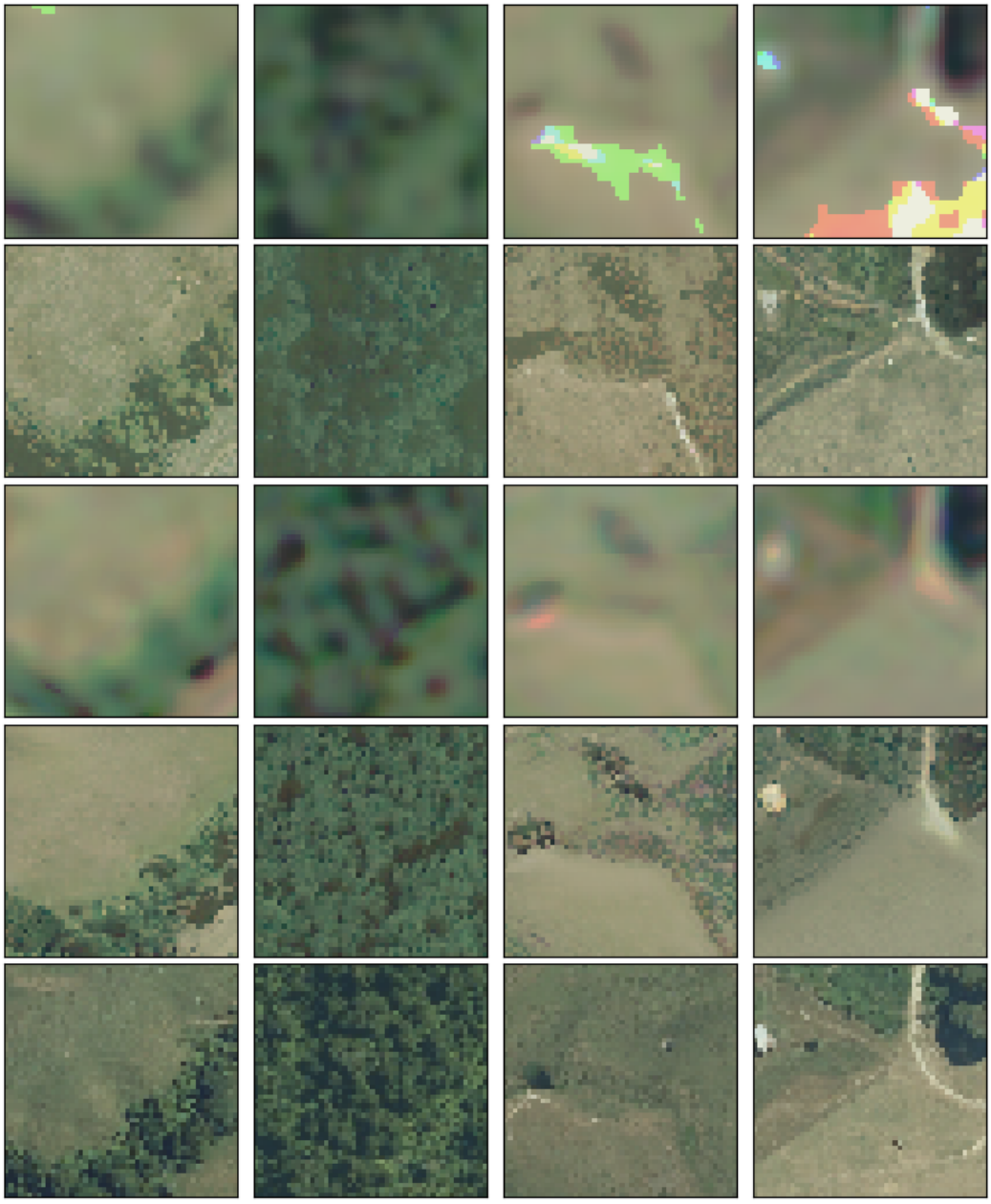


Figure 4.23: Zoomed in output from four test samples showing different stages of DeepSUM and ESRGAN processes. Each image is a 50 x 50 sample of the original 512 x 512 output. Top image: Original Sentinel 2 image patch bicubically upsampled by a factor of 4 to be 512x512 pixels, Second row images: Output of ESRGAN working directly on a single Sentinel 2 image, Third row images: Output of DeepSUM (mse loss) using all three bands, Fourth row images: Output of DeepSUM passed through ESRGAN without upsampling, Bottom row images: Ground truth HR data

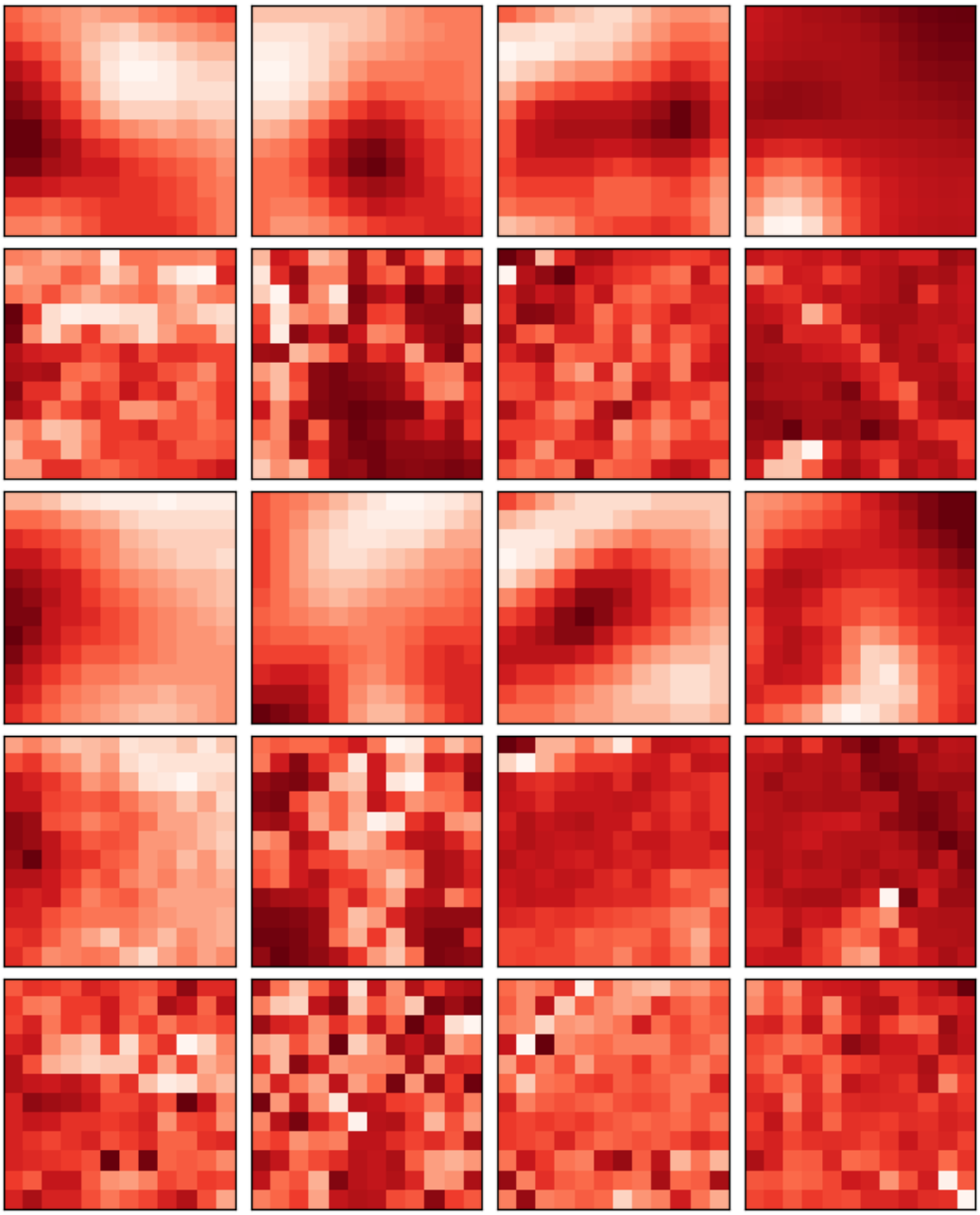


Figure 4.24: Zoomed in output from four test samples showing different stages of DeepSUM and ESRGAN processes. Each image is a 12 x 12 sample of the original 512 x 512 output showing only the red band where (darker is a stronger red colour). Top image: Original Sentinel 2 image patch bicubically upsampled by a factor of 4 to be 512x512 pixels, Second row images: Output of ESRGAN working directly on a single Sentinel 2 image, Third row images: Output of DeepSUM (mse loss) using all three bands, Fourth row images: Output of DeepSUM passed through ESRGAN without upsampling, Bottom row images: Ground truth HR data

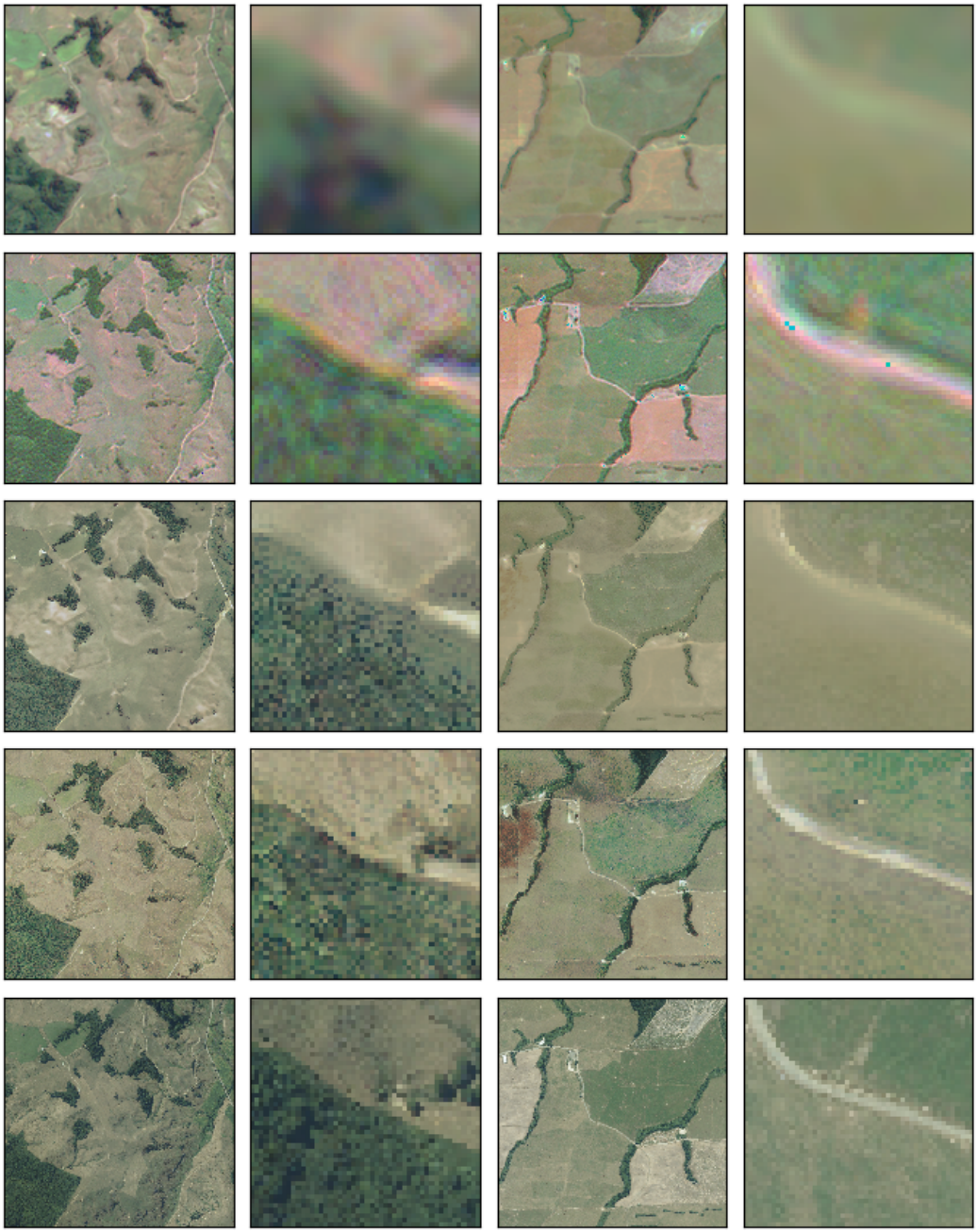


Figure 4.26: Output from two test samples showing effect of using the variation loss function on feature definition. Top row: Output of DeepSUM using an MSE loss with second and fourth column showing a zoomed in area of the first and third columns, Second row: Output of DeepSUM using the variation loss, Third row: ESRGAN product from DeepSUM (MSE loss), Fourth row: ESRGAN product from DeepSUM (variation loss), Bottom row images: Ground truth HR data

Chapter 5

Discussion

Satellite and aerial imagery have a myriad of purposes, some of which require absolute accuracy, for example identifying exact areas of landscape features such as buildings. Other uses are more fluid, for example, mapping vegetation types. Given these varied use cases, it is clear that both the perceptually enhanced, and visually more accurate images are valid for certain tasks.

5.1 An Image as a Representation of Reality

Reproducing the ground truth image will never occur exactly, as both the DeepSUM, and DeepSUM plus ESRGAN image are an amalgamation of several temporal images with images taken in different light, with different seasonal patterns. In this light, what is being created is a representation of reality, rather than reality. For example vegetation will show different shades of green and yellow depending on the length of grass and time of year. An image amalgamation will select the mid-point shade, while replicating the basic pattern. Only the ESRGAN output from a single raw image could reflect a reality, as this is a SISR process showing an image at a point in time. In a similar way, any SISR process is unlikely to exactly match the time when the aerial image (ground truth) was taken, so will too have pattern misalignments.

Because reality will never be achieved by an image derived from multiple sources, in some ways the perception-distortion dichotomy is an artificial

construct.

5.2 DeepSUM Algorithm

DeepSUM successfully super-resolved the Sentinel 2 satellite imagery using the aerial ground truth. The intuition that the information gain from using several different temporal images would aid super-resolution was not tested in this study. However, a superficially realistic image was produced with higher PSNR and SSIM values than using a bicubically upsampled image. Of the loss functions tested for use with DeepSUM, using an SSIM loss was shown to improve the accuracy of the original algorithm. Other loss functions produced enhanced results superior to that of the default MSE loss.

Using pixel based loss functions such as ℓ_1, ℓ_2, SSIM or MS-SSIM will always give a smooth unnatural looking result due to the averaging effect. In short, this effect is the blurring of hard lines, and smoothing of textures that can be seen in the results from DeepSUM. The cause of this effect is the mathematical imperative (in the case of an MSE loss) to find the least squared error. SSIM has a similar effect however this also takes into account surrounding pixels, but the end result is also a smoothed, compared to the ground truth.

As an example of how this works, in a raw image as used in this study, a single pixel of 10m x 10m could represent the edge of a road. If as is likely, this pixel does not fall exactly on the boundary of the road, then it will contain light return values from both the road and the vegetation on the side of the road. In an ideal world, this pixel will partially super-resolve to 16 pixels of 2.5m x 2.5m, some of which are road and some of which are vegetation. In the absence of exact information as to where the road boundary is, a mid-point pixel value is likely to be selected as having the least squared error compared to both the road and the surrounding vegetation.

This effect will be exacerbated by any minor misalignments of features.

Misalignment could occur by a number of mechanisms: As the ground truth imagery is photographed from an aeroplane (not a satellite), some effect of camera angle on feature location will be seen, so that even perfectly geo-referenced data may not always line up at a pixel level. Even imagery which has been correctly registered will have some minor misalignment with the ground truth. Similarly, as textures are being measured on a pixel-by-pixel basis, misalignment of patterns will mean that patterns are averaged, even if the basic pattern is the same. As DeepSUM is super-resolving multiple images, to some degree the averaging effect is possibly more pronounced than it would be for a single image. PSNR and SSIM type metrics reward this averaging effect. They do not favour the output of realistic textures or hard boundaries.

The perceptual loss function is theoretically able to define textures similar to the ground truth. As the style loss component is rewarded in finding similar textures to the ground truth, in theory this should work against the averaging effect, to attain a more perceptually accurate image. However, the content loss function which is used to define the overall structure of the image, uses an MSE loss on feature maps rather than on the image itself. This has a similar effect to the MSE pixel-based loss function. In this study, the perceptual loss function did not greatly enhance an image using either perceptual or pixel-based metrics.

The variation loss could be described as a pixel-based metric. As this works to preserve variation and contrast, the blurring effect is mitigated. However, when a low variation loss factor is used, this function has minimal effect on the image with some degree of enhanced contrasts, and textures. Using a high variation factor loss causes textures, and lines to appear artificial. Despite this, the DeepSUM output created using a high variation loss did exhibit a low LPIPs value, potentially indicating that LPIPs responds to the higher variation rather than specific textures of the variation.

Used alone, from this evidence and from the literature, it is clear that a feed-forward network such as DeepSUM will struggle to accurately render

textures that look correct to a human due to the smoothing effect of using a pixel based loss function. On the distortion-perception graph, its output is located far towards minimising distortion.

5.3 Effect of ESRGAN

Adding ESRGAN to the DeepSUM output was used to attempt to better render the contrasts and textures. Using the ESRGAN algorithm alone to super-resolve images produced a more photo-realistic image from Sentinel 2 data. This can be seen in texture of a zoomed in view of the ESRGAN products. However, the super-resolution process hallucinated details, and decreased pixel accuracy compared to a bicubically up-sampled image. On the dichotomy of the distortion-perception graph, this result is far towards the perception side.

The combination of DeepSUM and ESRGAN created an image with enhanced accuracy and better perceptual qualities compared to the bicubically upsampled images than using either DeepSUM or ESRGAN alone (see Figure 4.22). For example, in the zoomed-in image patches of the DeepSUM output, the fourth test image has a darker grassy area to the top right, which has been given the texture of bush. In contrast, when the same image is inferred using ESRGAN to process the DeepSUM output, the grassy area is given a less smoother texture more in fitting with the ground truth image. The 12×12 image patches shown in Figure 4.23, clearly illustrate how ESRGAN replicates the texture of the ground truth.

Using ESRGAN with the DeepSUM output, some minor detail was hallucinated, but far less than was created using the ESRGAN alone. Accuracy loss occurred when measured using PSNR and SSIM. Some of this loss in accuracy could be due to hallucination of detail, but a more significant cause is likely to be pixel misalignment of textures. As the detailed patterns produced by ESRGAN could not be expected to line up exactly against the patterns in the ground truth accuracy loss would be expected with a greater pixel-by-pixel

variation causing much larger differences in squared error.

The DeepSUM with variation loss and ESRGAN created the cleanest looking output as shown in Figure 4.26. ESRGAN appeared to respond to the greater contrasts produced by the variation loss by rendering crisper lines, and more perceptually accurate textures. Artefacts produced by the DeepSUM with variation loss disappeared after running images through ESRGAN.

This quality improvement was not quantifiable using any of the pixel or perceptually based metrics, as none of these metrics seem to favour sharp lines and boundaries. Interestingly, LPIPs does not improve markedly after running ESRGAN on the DeepSUM output with variation loss, despite the textures and lines becoming clearer.

5.4 Process Improvements

Other studies using a GAN such as [Hoque et al., 2019] and [Wang et al., 2018] used vastly more data to train the GAN than we have in this study. Using an MISR model as an input to the GAN, such as DeepSUM, meant that each output photo for the end-to-end process required 8 input images. [Wang et al., 2018] showed that a larger dataset and larger patch size leads to better results. They also found using a wider variety of data sources allowed different types of feature to be super-resolved better. In this study, we bumped up against the memory limits of the available servers, constraining the size of the patch we could use. However more data and a wider variety of data may have helped. In particular, the ESRGAN model used did not super-resolve buildings or roads well. This is probably because very few buildings were found in the training data. The model could be improved by adding more training data focused on buildings, and more built up areas.

Running ESRGAN from a DeepSUM output created with a variation loss factor of 3 was found to produce a sharp result. However, even with a high variation loss factor, its possible or likely this could be improved. Even the

variation loss of 3 did not create all the variation seen in the HR data. Potentially, a higher factor would create better data for the GAN to further refine.

As an end-to-end process, using DeepSUM as an example of a MISR process and ESRGAN together could potentially be integrated into a single process whereby multiple images are super-resolved with a combination adversarial loss and variation loss function.

5.5 Human Perception

There is some evidence from the literature that human perception is more complicated than it is often given credit for. In particular, the *Lines-as-Edges* hypothesis states that the human visual cortex includes cells that are responsive to edge patterns and fire strongly in response to lines and edges. Drawings simulate natural images because lines are drawn where edges often occur in natural images [Hertzmann, 2021]. It appears that if human are drawn to analyse patterns in edges and boundaries, when those boundaries are weaker, then this will affect a human's perception of this pattern. Enhancing edge patterns will produce a perceptually better image for the human viewer. When the purpose of image data is to identify features, areas, and boundaries, running DeepSUM with variation loss through the ESRGAN model is a clear winner, despite worse metrics than some of the other algorithms.

The metrics used in this study (PSNR, SSIM, LPIPs) do not favour creation of hard lines, and defined edges, in a way that is useful to a person. As LPIPs is a CNN, on some level it may respond to the presence of defined edges, but this does not show in the data from this study. Possibly, as defined lines are a small portion of any image, any effect they might have is dwarfed by the large areas of less defined vegetation or open land. In this light, the gap in knowledge here is less around the algorithm required to super-resolve an image, and more in field of defining a set of measurements which replicate the HVS.

5.6 Measurement improvements

The use-case of satellite and aerial imagery has some unique aspects that are not as important when using other imagery: Imagery is often used at different scales, and zoomed in closely. There is no background/foreground component, and an image does not reflect a scene people are used to seeing in every day life. Often images are not true to life, as for example cloud has been removed and images from different times or dates amalgamated.

For these reasons, it makes sense to create a metric to measure similarity and accuracy specifically for satellite and aerial imagery. An ideal measurement metric would favour defined clear boundaries and lines, correct textures and colours and depiction of detail. Should such a measurement be created, used as a loss function it would then favour the creation of images with these properties.

Given the artificial nature of the image being created, and its intended purpose, using a process that somewhat alters the image to improve its perceptual quality makes sense.

Chapter 6

Conclusion

This study explores different techniques for super-resolving satellite data, in particular the trade-off between optimising a result against pixel-based metrics such as SSIM and PSNR, and optimising for more perceptually-based metrics. Different loss functions were run, showing that the original CNN algorithm DeepSUM, could be improved. The addition of ESRGAN to the process showed how the data could be made much more photo-realistic. In particular, the GAN created much more realistic textures and fine detail; however, this came at some uncertainty as to accuracy of minor detail. Using the novel loss function *variation loss* with DeepSUM produced a much crisper final output with stronger edges and boundaries, and better textures. However, this result did not reflect in measurement metrics.

In the study, as each output image is an amalgamation of several temporarily different inputs, the result is not a true representation of any real image. In this sense, using a GAN to make the DeepSUM output appear more realistic-looking to a human is appropriate, as the output image is only ever a representation of reality. From running these processes, it is clear than none of the metrics used to measure output, namely PSNR, SSIM and LPIPs, successfully measure what is useful to a human. Better quantifying how humans perceive satellite data will advance this field.

References

[nvd,] Nvdi. https://www.usgs.gov/core-science-systems/nli/landsat/landsat-normalized-difference-vegetation-index?qt-science_support_page_related_con=0#qt-science_support_page_related_con, note = Accessed: 2021-03-03.

[Blau and Michaeli, 2018] Blau, Y. and Michaeli, T. (2018). The perception-distortion tradeoff. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6228–6237.

[Chen et al., 2010] Chen, Q., Montesinos, P., Sun, Q. S., Heng, P. A., et al. (2010). Adaptive total variation denoising based on difference curvature. *Image and vision computing*, 28(3):298–306.

[Dong et al., 2015] Dong, C., Loy, C. C., He, K., and Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307.

[ESA, 2020] ESA (2020). SNAP - ESA's SentiNel Application Platform. https://step.esa.int/main/download/snap-download/. Accessed: 2020-10-02.

[Esri, 2021] Esri (2021). ArcGIS Pro Overview. https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview. Accessed: 2020-10-02.

[European Space Agency, 2013] European Space Agency (2013). Sentinel 2. https://sentinel.esa.int/web/sentinel/missions/sentinel-2. Accessed: 2021-10-02.

[Gatys et al., 2015] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, pages 1646–1654.

[Goodfellow et al., 2020] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

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

[Hertzmann, 2021] Hertzmann, A. (2021). The role of edges in line drawing perception. *Perception*, 50(3):266–275.

[Hoque et al., 2019] Hoque, M. R. U., Burks, R., Kwan, C., and Li, J. (2019). Deep learning for remote sensing image super-resolution. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0286–0292. IEEE.

[Hore and Ziou, 2010] Hore, A. and Ziou, D. (2010). Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE.

[Irani and Peleg, 1991] Irani, M. and Peleg, S. (1991). Improving resolution by image registration. *CVGIP: Graphical models and image processing*, 53(3):231–239.

[Jiang et al., 2019] Jiang, K., Wang, Z., Yi, P., Wang, G., Lu, T., and Jiang, J. (2019). Edge-enhanced gan for remote sensing image superresolution. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5799–5812.

[Jo et al., 2018] Jo, Y., Oh, S. W., Kang, J., and Kim, S. J. (2018). Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3224–3232.

[Johnson et al., 2016] Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer.

[Jolicoeur-Martineau, 2018] Jolicoeur-Martineau, A. (2018). The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*.

[Kim et al., 2016a] Kim, J., Kwon Lee, J., and Mu Lee, K. (2016a). Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654.

[Kim et al., 2016b] Kim, J., Lee, J. K., and Lee, K. M. (2016b). Deeply-recursive convolutional network for image super-resolution. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pages 1637–1645.

[Lary et al., 2016] Lary, D. J., Alavi, A. H., Gandomi, A. H., and Walker, A. L. (2016). Machine learning in geosciences and remote sensing. *Geoscience Frontiers*, 7(1):3–10.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

[Li et al., 2019] Li, H., Lam, K.-M., and Wang, M. (2019). Image super-resolution via feature-augmented random forest. *Signal Processing: Image Communication*, 72:25–34.

[LINZ, 2017] LINZ (2017). Wraps aerial photography. `https://data.linz.govt.nz/layer/104600-waikato-03m-rural-aerial-photos-2016-2019/`. Accessed: 2020-10-02.

[Liu et al., 2021] Liu, Z.-S., Siu, W.-C., and Wang, L.-W. (2021). Variational autoencoder for reference based image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 516–525.

[Lugmayr et al., 2020] Lugmayr, A., Danelljan, M., Van Gool, L., and Timofte, R. (2020). Srflow: Learning the super-resolution space with normalizing flow. In *European Conference on Computer Vision*, pages 715–732. Springer.

[McHugh, 2013] McHugh, S. (2013). Gamma correction. `https://www.cambridgeincolour.com/tutorials/gamma-correction.htm/`. Accessed: 2021-03-03.

[Meraner et al., 2020] Meraner, A., Ebel, P., Zhu, X. X., and Schmitt, M. (2020). Cloud removal in sentinel-2 imagery using a deep residual neural network and sar-optical data fusion. *ISPRS Journal of Photogrammetry and Remote Sensing*, 166:333–346.

[Molini et al., 2019] Molini, A. B., Valsesia, D., Fracastoro, G., and Magli, E. (2019). DeepSUM: Deep neural network for super-resolution of unregistered multitemporal images. *IEEE Transactions on Geoscience and Remote Sensing*, 58(5):3644–3656.

[Moore, 1979] Moore, G. K. (1979). What is a picture worth? a history of remote sensing/quelle est la valeur d'une image? un tour d'horizon de télédétection. *Hydrological Sciences Bulletin*, 24(4):477–485.

[Ng et al., 2011] Ng, A. et al. (2011). Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19.

[NumPy, 2021] NumPy (2021). Numpy Home Page. `https://numpy.org`. Accessed: 2020-10-02.

[Planet Labs Inc, 2021] Planet Labs Inc (2021). Planet website. `https://www.planet.com`. Accessed: 2021-03-03.

[Rad et al., 2019] Rad, M. S., Bozorgtabar, B., Marti, U.-V., Basler, M., Ekenel, H. K., and Thiran, J.-P. (2019). Srobb: Targeted perceptual loss for single image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2710–2719.

[Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268.

[Rupprecht, 2017] Rupprecht, P. (2017). Style transfer. `https://gcamp6f.com/2017/12/05/understanding-style-transfer//`. Accessed: 2021-03-03.

[Salgueiro Romero et al., 2020] Salgueiro Romero, L., Marcello, J., and Vilaplana, V. (2020). Super-resolution of sentinel-2 imagery using generative adversarial networks. *Remote Sensing*, 12(15):2424.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Sun et al., 2008] Sun, J., Xu, Z., and Shum, H.-Y. (2008). Image super-resolution using gradient profile prior. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[Tfrecord, 2021] Tfrecord (2021). Tfrecord Tutorial. `https://www.tensorflow.org/tutorials/load_data/tfrecord`. Accessed: 2020-10-02.

[Tona et al., 2018] Tona, C., Bua, R., Arcari, C., Borgia, B., and Montieri, A. (2018). Open Source DataHubSystem: from Sentinels data access to an agnostic data distribution tool. In *42nd COSPAR Scientific Assembly*, volume 42, pages PSW.4–13–18.

[Waldner and Diakogiannis, 2020] Waldner, F. and Diakogiannis, F. I. (2020). Deep learning on edge: Extracting field boundaries from satellite images with a convolutional neural network. *Remote Sensing of Environment*, 245:111741.

[Wang et al., 2018] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., and Change Loy, C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

[Wang and Bovik, 2009] Wang, Z. and Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117.

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

[Wang et al., 2003] Wang, Z., Simoncelli, E. P., and Bovik, A. C. (2003). Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee.

[Wired Magazine, 2019] Wired Magazine (2019). How Google Pixel 3's Camera Works Wonders With Just One Rear Lens. `https://www.wired.com/story/google-pixel-3-camera-features/`. Accessed: 2020-10-02.

[Wu et al., 2020] Wu, Q., Fan, C., Li, Y., Li, Y., and Hu, J. (2020). A novel perceptual loss function for single image super-resolution. *Multimedia Tools and Applications*, 79(29):21265–21278.

[Wulder et al., 2019] Wulder, M. A., Loveland, T. R., Roy, D. P., Crawford, C. J., Masek, J. G., Woodcock, C. E., Allen, R. G., Anderson, M. C., Belward, A. S., Cohen, W. B., Dwyer, J., Erb, A., Gao, F., Griffiths, P., Helder, D., Hermosilla, T., Hipple, J. D., Hostert, P., Hughes, M. J., Huntington, J., Johnson, D. M., Kennedy, R., Kilic, A., Li, Z., Lymburner, L., McCorkel, J., Pahlevan, N., Scambos, T. A., Schaaf, C., Schott, J. R., Sheng, Y., Storey, J., Vermote, E., Vogelmann, J., White, J. C., Wynne, R. H., and Zhu, Z. (2019). Current status of landsat program, science, and applications. *Remote Sensing of Environment*, 225:127–147.

[Xiao et al., 2020] Xiao, L., Nouri, S., Chapman, M., Fix, A., Lanman, D., and Kaplanyan, A. (2020). Neural supersampling for real-time rendering. *ACM Transactions on Graphics (TOG)*, 39(4):142–1.

[Yang et al., 2020] Yang, H., Yang, C. H., and James Tsai, Y. (2020). Y-net: Multi-scale feature aggregation network with wavelet structure similarity loss function for single image dehazing. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2628–2632.

[Yang et al., 2019] Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.-H., and Liao, Q. (2019). Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121.

[Zhang et al., 2020] Zhang, J., Shao, M., Yu, L., and Li, Y. (2020). Image super-resolution reconstruction based on sparse representation and deep learning. *Signal Processing: Image Communication*, 87:115925.

[Zhang et al., 2018] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595.

[Zhao et al., 2016] Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2016). Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57.

[Zhu et al., 2020] Zhu, X., Talebi, H., Shi, X., Yang, F., and Milanfar, P. (2020). Super-resolving commercial satellite imagery using realistic training data. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 498–502. IEEE.