# Generation of Super Resolution Images using Deep Neural Networks

*A Preliminary Report on the study of available methods for Super Resolution for fulfillment of the Summer Research Internship Project*

*by*

Abhimanyu Bhowmik


REGIONAL REMOTE SENSING CENTER, EAST

National Remote Sensing Centre (ISRO)

# Certificate

It is certified that the work contained in this thesis entitled **"Generation of Super Resolution Images using Deep Neural Networks"** by **Abhimanyu Bhowmik** has been carried out under my supervision and that it has not been submitted elsewhere for any purposes.

Dr. Arati Paul
Project Supervisor
Regional Remote Sensing Centre, East
National Remote Sensing Centre (ISRO)

# Declaration

This is to certify that the thesis titled **"Generation of Super Resolution Images using Deep Neural Networks"** has been authored by me. It presents the research conducted by me under the supervision of **Dr. Arati Paul**.

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for any purposes. Further, due credit has been attributed to the relevant state-of-the-art collaborations with appropriate citations and acknowledgements, in line with established norms and practices.

Abhimanyu Bhowmik
Summer Research Intern
Regional Remote Sensing Centre, East
National Remote Sensing Centre (ISRO)

# *Abstract*

Name of the author: **Abhimanyu Bhowmik**

Organization: **Regional Remote Sensing Centre - East, NRSC(ISRO)**

Title: **Generation of Super Resolution Images using Deep Neural Networks**

Name of the supervisor: **Dr. Arati Paul**

Month and year of submission: **April 2023**

Super Resolution Images are required to properly perceive the intricacies of any given image. Satellite imaging is one of such domains where details of an image must be preserved extremely carefully since image quality decreases drastically at high magnification. Due to developments in the disciplines of computer vision and deep learning, super-resolution which tries to increase picture resolution by computational means has advanced recently. Convolutional neural networks built on a range of architectures, such as autoencoders, generative adversarial networks, and residual networks, have been used to tackle the issue. Few studies concentrate on single or multi-band analytic satellite imaging, whereas the majority of research focuses on the processing of images with simply RGB colour channels. Super resolution is a highly important and significant operation that must be carried out carefully in the realm of remote sensing. Deep learning methods are used to satellite images in this work. It focuses particularly on the recently developed SR GAN, SR-ResNet, and EDSR models.All of the models' output is compared to both one another and the traditional super resolution benchmark bicubic interpolation. Results of the transfer learning before and after demonstrate a considerable gain in performance, with SR GAN outperforming the others.

# *Acknowledgements*

Words cannot express my gratitude to my professor and senior scientist Dr. Arati Paul for her invaluable patience and feedback. Without the assistance of Amity University Kolkata, which kindly gave me the knowledge and experience I needed to conduct study in this specific area, I would not have been able to go on this adventure. Additionally, this endeavour would not have been possible without the generous support from the Regional Remote Sensing Centre- East, NRSC, ISRO who provided me with this research opportunity.

I am also grateful to the professors of Amity University Kolkata, especially Dr. Semati Chakraborty and my classmates from my University, for their help, feedback sessions, and moral support. Lastly, I would be remiss in not mentioning my family, especially my parents. Their belief in me has kept my spirits and motivation high during this process.

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| **HR** | High Resolution |
| **LR** | Low Resolution |
| **SR** | Super Resolution |
| **CNN** | Convolutional Neural Network |
| **GAN** | Generative Adversarial Networks |
| **HVS** | Human Vision System |
| **VAE** | Variational Auto-Encoder |
| **EDSR** | Enhanced Deep Residual Networks |
| **PSNR** | Peak Signal-to-Noise Ratio |
| **SSIM** | Structural Similarity Index Measure |
| **RMSE** | Root Mean Squared Error |
| **SRGAN** | Super Resolution Generative Adversarial Networks |
| **SRVAE** | Super Resolution Variational Auto-Encoder |
| **VDVAE** | Very Deep Variational Auto-Encoder |
| **ResNet** | Residual Neural Network |
| **SRResNet** | Super Resolution Residual Neural Networks |

# Chapter 1

# Introduction

Super-resolution imaging is the process of up-scaling or enhancing the resolution of a lower resolution (LR) image to a higher resolution (HR) image. A high-resolution image has a higher pixel density and provides much more detailed information about the image. This is crucial for extracting minute details from satellite images and amplifying their zooming capacities with minimal distortion to the image. Images with high resolution are often desired in a vast number of digital image applications ranging from satellite imagery to medical imaging.

## 1.1 Image super-resolution

The resolution of an image may be raised in a number of ways. Hardware-wise, the easiest way to increase pixel sensors on a chipset is by shrinking the pixel size. Reduced pixel size results in shot noise and a reduction in photons hitting the sensor, deteriorating image quality. Therefore, the size of the image sensor can only be shrunk so far physically[11]. It would also be feasible to create bigger image sensors, but doing so would result in bulkier and costlier cameras. Further possibilities include expanding the sensor chip's size at the expense of raising the system's capacitance, leading to heavy and large hardware. Super-resolution involves the use of signal-processing techniques to construct HR images with better attributes than compared to a sampling grid[12]. This overcomes the constraints of sensor technology and imaging techniques. In order to eliminate signal loss in fused HR pictures, early research in the area of super-resolution was conducted in the 1980s in the frequency domain utilizing a number of under-sampled LR images[13].

## 1.2 Key Objectives

In this paper, several SR techniques have been examined and contrasted on the basis of numerous standard matrices. Deep learning techniques like GAN and CNN-based methods as well as conventional non-Deep learning approaches are examples of different methodologies. The project's primary goals are:

- Making model independent of the size of images and number of bands.
- Retain the geo-referencing information in the generated SR images.
- Training the general models in the specificity of satellite images and their attributes for improving the accuracy (i.e. Transfer Learning).
- Comparing various deep learning and other approaches with each other to infer the best suitable SR model for satellite imagery.

## 1.3 Applications

As Deep Learning has grown in popularity, upscaling techniques utilizing Deep Learning have begun to appear in recent years. In the context of satellite imagery, there are several applications that can be considered. For example, it allows objects to be easily identified, made detectable, and more evident for manual or automated object detection techniques.

By upscaling satellite images, we can monitor land, sea, forest, and many other graphical properties of the earth's surface in minute details. The specio-temporal information contained in satellite images is very helpful for urban planning, observing changes in land usage, transportation and environmental planning.

Satellite imagery data and inferences are essential in the areas of disaster management, defence, agriculture, telecommunications, and aviation. Enhancing the images used for these purposes will undoubtedly improve the inference's accuracy and make prediction much more efficient.

## 1.4 Challenges

Since satellite images are very different from normal day to day images we get, handling satellite data is quite challenging. A few challenges faced while handling satellite images are as follows[14].

- The greater dimension of multi spectral and hyper spectral data ranges from one to many bands. Each observation contains a significant amount of data due to the wide field of view and distance between the sensor and scene. Encapsulating and retaining most of the features is one of the challenging tasks to solve.
- Making models that can be used with both types of satellite photos, which might be single or multiband, is a challenging problem. Additionally, for multiband pictures, the suggested model may exhibit some degree of bias toward a particular band that has to be eliminated.
- The images are in *.tiff* format and are huge in size. Processing the entire image in an instance requires tremendous resources and high-capacity computing instances. To make it computationally less exhaustive, the images must be shredded down into smaller patches before super-resolution.
- The satellite images are encoded in multiple different formats, and handling the mismatch of the bit depth, such as, 8 bit encoded low resolution or high resolution image with a 16 bit encoded counterpart, is another challenge.
- Another major problem is adjusting the georeferencing of images after super-resolution imaging is performed on them. This adjustment of geo-referencing is crucial for map analysis, such as determining the exact coordinates of a point or image.
- A digital number (DN) is a numerical value assigned to each pixel in a satellite picture that records the intensity of electromagnetic radiation detected for the ground resolution cell represented by that pixel. The preservation of DN values during image super resolution is required since it is a critical aspect in remote sensing for image classification.

## 1.5 Organization of the Report

The structure of the report is as follows: Chapter II incorporates state-of-the-art solutions to Super resolution problem while Chapter III confers the theoretical background behind

the project. Chapter IV and Chapter V consists of the methodology and experimentation procedures whereas, Chapter VI showcases the results and outcome of the study.Finally, Chapter VII concludes the paper and highlights future research directions.

# Chapter 2

# Literature Review

The image reflects the overall perspective of image super-resolution and the common techniques used to achieve it. There are a number of techniques, including traditional classical techniques as well as machine and deep learning models. Deep learning techniques stand out among them as they have the It has the potential to deliver outcomes with consistently high accuracy. A few cutting-edge models are explained in further depth.

## 2.1 CNN-based model

The SRCNN[15], based of a 3-layered CNN framework[16] and employing a bicubic interpolated[17] low-resolution image as input to the model, is one of the earliest research that obtained decent results in the domain of superresolution. Since the Residual Neural Network(ResNet)[18] was developed, several studies have embraced it because it provides rapid training and improved performance for deep designs[19]. A few such state-of-the-art models like SRResNet[10], used in GAN implementation, and SRDenseNet[20] in which the size of the model is dilated for better results.EDSR(Enhanced Deep Residual Networks)[9] is another popular model for Single Image Super-Resolution.

## 2.2 GAN-based model

The Generative Adversarial Networks[21] for Super Resolution (SRGAN)[10] are entirely predicated on gated adversarial networks with the theory that the LR image is handed on to the generator network as input and the discriminator network attempts to make a

distinction between the original HR image and the SR output developed by the generator network[22]. Implementing game theory method, GANs[23] execute an implicit density estimation and learn to produce from genuine data distribution. GANs have historically been seen as very unstable and difficult to train, however once training is done correctly, they are capable of producing very excellent sample quality[24]. Among the top performing image superresolution algorithms are the ESRGAN[25], which is an enhanced version of the SRGAN and Enhanced Net (Single Image Super-Resolution Through Automated Texture Synthesis)[26].

## 2.3 VAE-based model

In order to produce new data, the VAEs[27] sample from a learned latent space. They simulate an explicit, difficult-to-optimize density function, so the lower bound of likelihood is optimised in its place. Most recently, research has shown a significant improvement by using more flexible approximations and giving the latent variables structure[28]. These VAE have historically been known for their poor sample quality. Even though a recent study found that these models performed fairly well, GAN models continue to be the superior option for creating SR images. The SR VAE[29] is a recent example of an image superresolution system that uses a VAE architecture. To create a richer prior distribution in this work, the authors use a bijective model corresponding to RealNVP[30]. Another recent model that produces noticeably good results is the VDVAE SR model[8], which is VAE-based (Image Super-Resolution With Deep Variational Autoencoders).

## 2.4 Transformer based model

Transformers and attention-based architectures have dominated the area of Natural Language Processing since their debut by Vaswani et al. 2017[31], easily surpassing other techniques employed at the time like LSTMs or RNNs in terms of performance. BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. 2019[32], GPT (Generative Pretrained Transformer) by Radford et al.[33], T5 (Text to Text Transfer Transformer), and other popular adaptations are among the most widely used (Raffel et al. 2020)[34]. Transformers entered the field of vision as a result of the impressive outcomes in other fields[35]. They are now used for a variety of tasks including classification problems, object recognition, and even superresolution, with modeling techniques like Yang et al.[36] achieving excellent outcomes in superresolution, more particularly in

pattern recovery. A Texture Transformer made up of a learnable pattern extractor, a relevance embedding module, a hardattention unit for feature transfer, and a softattention subsystem for feature synthesizing work together in the final model to accomplish this. Another feature of the model is that it uses 4 pictures as input to obtain further details about the low-resolution textures: the reduced image, the high-quality reference image, a 4x bicubic upsampled low-res image, and a reference image that has undergone consecutive down- and upsampling.

# Chapter 3

# Theoretical Background

This chapter covers all the theoretical groundwork required for understanding artificial intelligence, with a particular emphasis on deep learning and neural networks. The section **??** contains information on a number of different models, including ANN, CNN, and GAN, as well as the prerequisites needed to use them. For a better understanding of the data given in the next chapter, section 3.5 is also addressed in relation to satellite imaging in the context of remote sensing.

## 3.1 Artificial Neural Networks

In the domains of AI, machine learning, and deep learning, neural networks enable computer programs to identify patterns and address common issues. They are additionally referred to as simulated neural networks (SNNs). Their structure and nomenclature are modeled after the human brain, mirroring the communication between organic neurons.

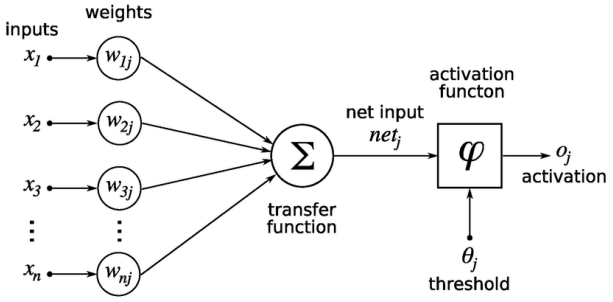


FIGURE 3.1: Architecture of a Single Neuron. [1]

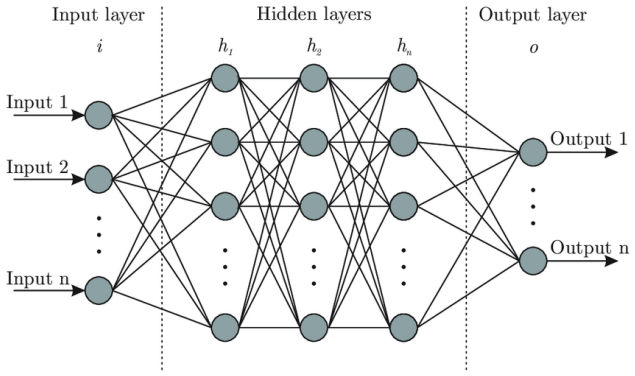


FIGURE 3.2: Figure of the architecture of a Multilayer Perceptron neural network. [2]

These networks mimic the function of the human brain. Deep learning techniques are based on neural networks, a branch of machine learning.

Combinations of fundamental neurons, also known as perceptrons (a fundamental unit depicted in the above figure 3.1), are placed in a network of layers to form a neural network (figure 3.2). An input layer, a processing layer, and an output layer are the three primary parts of a single neuron. Data is fed into the network through the input layer. The weighted summation of these inputs along with a bias ($\beta$) value are delivered to the transfer function as net input 3.1. The obtained output from the transfer function(3.2) is represented by the output layer (3.1.2).

$$y_{net} = \sum_i x_i w_i + \beta \tag{3.1}$$

$w$: Weight, $\beta$: Bias, $y_{net}$: Net input, $x$: Input

Equation 3.2, which includes a weighted summation and a non-linear activation function3.1.2 that resembles the threshold set by the cell body, may be used to describe an artificial neuron (known as ANN). Layers can be created by joining many neurons together.

$$y = f(\sum_i \omega_i x_i + \beta_i) \tag{3.2}$$

$x_i$: Network Input, $\omega_i$: Weight, $\beta_i$: Bias, $f$: Activation function,$y$: Network output

Simple logical processes can be carried out by interconnecting layers in sequence. The network can carry out increasingly complicated jobs as the layers get more sophisticated. A multilayer perceptron (MLP), which consists of multiple completely linked layers, is illustrated in Figure 3.2. Each neuron in a neural network is linked to every other neuron present in the previous layer.

### 3.1.1 Perceptron Learning Rule

After Rosenblatt (1958) originally put out this concept, a single-layer feed-forward network is referred to as a perceptron. The sensory unit (input unit), the associator unit (hidden unit), and the response unit (output unit) are the three components that make up the perceptron network. The associator units that connect to the sensory units have fixed weights that are randomly given values of 1, 0, or -1. Both the sensory unit and the associator unit employ the binary activation function. The activation of the response unit might be 1, 0, or -1. The associator is activated using a binary step with a predefined threshold $\theta$.

The output of a perceptron network is given by $y = f(y_{net})$ where $f(y - net)$ is describe in the equation 3.3.

$$f(y_{net}) = \begin{cases} 1, & \text{if } y_{net} > \theta \\ 0, & \text{if } -\theta \leq y_{net} \leq \theta \\ -1, & \text{if } y_{net} < -\theta \end{cases} \tag{3.3}$$

The weight updating between the associator unit and the response unit use the perceptron learning rule. The net output will compute the response for each training input and identify whether or not an error has occurred. In the instance of perceptron learning, the weight update is as displayed in the equations 3.4,3.5 .

$$w_{new} = \begin{cases} w_{old} + \alpha t x, & \text{if } y \neq t \\ w_{old}, & \text{otherwise} \end{cases} \tag{3.4}$$

$$b_{new} = \begin{cases} b_{old} + \alpha t, & \text{if } y \neq t \\ b_{old}, & \text{otherwise} \end{cases} \tag{3.5}$$

Here in equation 3.4 and 3.5 , $w$ and $b$ are the weights and biases of the perceptron network. $t$ is the target variable or the actual given label and $\alpha$ is the learning rate. A learning rate determines how fast or slow the algorithm converges to the optimum value. If the value of $\alpha$ is very high it takes very little time but it may not converge to the optimum solution, however a lower value of $\alpha$ increases the chance of convergence but takes more time to converge. The Learning Rate is a hyper-parameter which set by the user.

### 3.1.2 Activation Function

For neural networks, there are several activation functions that may be used. Their key characteristic is the non-linearity. Convolution is a linear process. This has the result that if several convolutions are applied successively to create an output, only one convolution can also provide the exact same result. As a result, a neural network with no non-linearities can essentially handle problems involving only linear regression. As a result, non-linearities are important in neural networks.

#### 3.1.2.1 Sigmoid Activation

A popular activation function with a distinctive "S" shaped graph is the sigmoid activation, often known as the logistic function. It is limited between 0 and 1 and has continuous derivatives. Thus, it is used for the mapping of arbitrary values to this range (as demonstrated in eq. 3.6). The sigmoid function, for instance, has the benefit of being easy to compute for its first derivative and may be employed as the final activation in binary classification tasks though never for categorical classification.

$$S_x = \frac{1}{1 + e^{-x}} \tag{3.6}$$

$$S'_x = S_x(1 - S_x)$$

where $x$: Network Input, $S_x$: Sigmoid of input x

#### 3.1.2.2 Softmax Activation

A sigmoid function adaptation that can handle inputs with multiple dimensions is the softmax activation function. Let z be the function's C-dimensional input vector. The softmax function returns C values in the range of zero to one such that all the values sum up to one. Equation 3.7 shows the mathematical representation of the function. The maximum value of the input vector gets reduced and ends up being the nearest to one.Because of this, categorical classifications where the network must "decide" which output value is most pertinent are perfect applications for the softmax function.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_j}} \qquad for\ j = 1, 2, ..., C \tag{3.7}$$

#### 3.1.2.3 ReLU Activation

As early as 1975[37], the rectifying linear unit (ReLU) activation was employed to simulate the activity of biological neurons. When their total input surpasses a specific threshold, biological neurons begin to fire; otherwise, they stay essentially dormant. This behaviour is mimicked by the ReLU process, which maps all negative inputs to zeros while allowing all positive inputs to pass through, yielding zero as a threshold value(as in eq. 3.8). It has been shown to increase the efficiency of model training[38]. The derivative of the ReLU function is the Heaviside-function, commonly referred to as the step-function.

$$ReLU = \max(0, x) \tag{3.8}$$

where $x$: Positive Network Input

#### 3.1.2.4 LeakyReLU and PReLU Activation

One of the factors in recent deep learning achievements has been ReLU. When used, it has shown better results than sigmoid. This is partly caused by the issue with vanishing gradients that arises with sigmoid activations. ReLU still has room for improvement, though. LeakyReLU [39] was invented, which does not zero off the negative inputs the way ReLU does. Instead, it leaves the positive input at its current value and multiplies the negative input by a small number (such as 0.02) [Figure: 3.3]. However, this has only slightly improved the accuracy of our models. By learning the small value (parameter) during training, we may enhance it and make our activation function more tolerant of the other factors (like weights and biases). Herein lies the role of PReLU [3] . With a little increase in training costs, backpropagation allows us to learn the slope parameter.

$$f(x_i) = max(0, x_i) + a_i min(0, x_i) \tag{3.9}$$

In the equation 3.9, $x_i$ is any input on the $i$th channel, and $a_i$ is the negative slope, a parameter that may be learned.

- If $a_i = 0$ , $f$ transform into ReLU.
- If $a_i > 0$, $f$ starts to leak. ReLU
- $f$ becomes PReLU if $a_i$ is a learnable parameter.

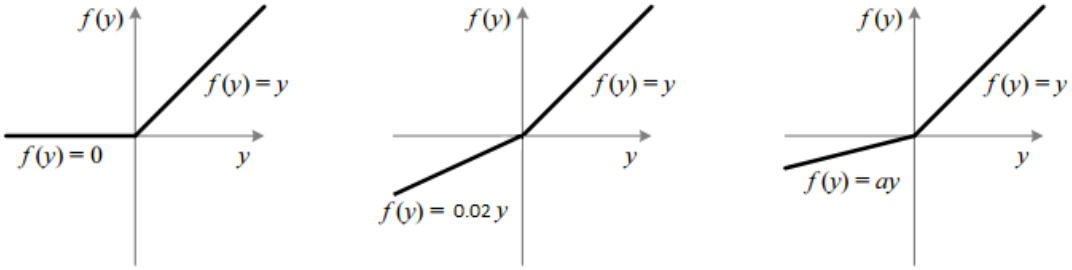


FIGURE 3.3: The figure shows the comperison graph of (Left) ReLU, (Middle), LeakyReLU and (Last) PReLU [3]

### 3.1.3 Optimizer

In order to train a neural network, we must adjust the weights for each epoch and reduce the loss function. An optimizer is a procedure or method that alters neural network properties like weights and learning rates. When mapping inputs to outputs, an optimization method determines the value of the parameters (weights) that minimizes the error. As a result, it aids in decreasing total loss and increasing precision. They also have a deep impact on the model's speed training. A deep learning model often has millions of parameters, making the process of selecting the proper weights for the model challenging. It highlights the importance of selecting an optimization algorithm that is appropriate for the model.

#### 3.1.3.1 Gradient Descent

By scaling the present location with a learning rate and deducting the resulting value from the current position to move on, the gradient or steepest descent algorithm continuously determines the very next point. It will deduct value from the objective function in order to minimise the function rather than the other way around. This process is carried out as follows(Eq. 3.10):

$$x_{n+1} = x_n - \eta \nabla f(x_n) \tag{3.10}$$

The gradient is scaled by a key parameter $\eta$, which in turn determines the step size. Performance is significantly impacted by it, which is known as the learning rate in machine learning.

- The gradient descent process will take significantly longer to optimise if the learning rate is very low, or it may approach its maximum number of iterations before discovering the best solution.
- If the learning rate is too high, the approach could bounce about the solution instead of converging, or it might even completely diverge.

#### 3.1.3.2 Stochastic Gradient Descent

In order to overcome the limitations of gradient descent, stochastic gradient descent (SGD) is an iterative approach for maximising an objective function. Since it estimates the gradient using a randomly chosen subset of the data rather than the true gradient, which is determined from the complete data set, it may be thought of as a stochastic approximation of gradient descent optimization. This minimises the extremely high computational complexity, especially in high-dimensional optimization problems, allowing for quicker iterations at the expense of a reduced convergence rate.

#### 3.1.3.3 Adam Optimizer

Along with its advantage of being simple, the SGD method has a number of drawbacks. One big issue is that it can be quite stagnant in some circumstances and become stuck in sub-optimal minima, like saddle points. The Adam Optimizer [40] was developed in order to address this issue along with many others using SGD. It maintains a very simple design and implementation while achieving faster convergence than the bulk of other optimizers in a variety of circumstances. The optimizer computes exponential moving averages of the first and second moments, or mean and uncentered variance of the gradients, to find update steps specific to each individual weight.

### 3.1.4 Loss Function

To reduce algorithmic error, neural networks employ optimizing techniques such stochastic gradient descent. We really use a Loss Function to calculate this mistake. It is used to express how well or poorly the model is working. Depending on the issue, there are several functions available to determine the loss based on the expected and actual value.

#### 3.1.4.1 Perceptual loss function

When evaluating two distinct photographs that appear similar, such as the identical photo that has been displaced by one pixel, perceptual loss functions are utilised. The tool compares important discrepancies between photographs, such as those in content and appearance. The performance of the generator network depends heavily on the specification of the perceptual loss function $l^{SR}$ utilised in this study. While $l^{SR}$ is frequently modelled using the MSE 3.12, the enhanced version of the loss discussed in Johnson et al. [41] and Bruna et al. [42] has been created as a loss function that evaluates a solution with regard to perceptually important properties. The weighted sum of a content loss ( $l_X^{SR}$ ) and an adversarial loss component is[10] and is used to define the perceptual loss as shown in Equation 3.11:

$$l^{SR} = \overbrace{\underbrace{l_X^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}^{\text{perceptual loss (for VGG based content losses)}} \tag{3.11}$$

#### 3.1.4.2 Content Loss

The pixel-wise MSE loss is describes as follows:

$$l_{MSE}^{SR} = \frac{1}{r^2 WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2 \tag{3.12}$$

The majority of cutting-edge methods for image SR depend on this as their primary optimization objective. The solutions to MSE (Eq. 3.12 )optimization problems frequently lack high-frequency content, resulting in perceptually unpleasant solutions with excessively smooth textures, even if they frequently achieve extremely high PSNR (Eq. 5.3). The content loss is built on the principles of Gatys et al. [43], Bruna et al. [42], and Johnson et al.[41] and employs a loss function that is more closely related to perceptual similarity rather than depending on pixel-wise losses. The pre-trained VGG 19 model described in Simonyan and Zisserman [44]'s description of the ReLU (Eq. 3.8) activation layers serves as the foundation for the VGG loss. With $\phi_{i,j}$ denoting the feature map produced by the $j$-th convolution (after activation) in the VGG19 network before the $i$-th maxpooling

layer. The euclidean distance between both the feature representations of a reconstructed image, $G_{\theta_G}(I^{LR})$, and the reference image, $I^{HR}$, is known as the VGG loss:

$$l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \tag{3.13}$$

Here, $W_{i,j}$ and $H_{i,j}$ outline the specific feature maps' dimensions within the VGG network.

#### 3.1.4.3 Adversarial Loss

Along with the previously described content losses, the perceptual loss in GAN base models like SRGAN also includes the generative GAN element. The discriminator network is encouraged to prefer responses that exist on the diversity of natural images by the attempt to deceive the network. The generative loss $l^{SR}_{Gen}$ is stated as follows based on the probabilities of the discriminator $D_{\theta_D}(G_{\theta_G}(I^{LR}))$ across all training instances:

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -log(D_{\theta_D}(G_{\theta_G}(I^{LR}))) \tag{3.14}$$

The likelihood that the reconstructed image,$G_{\theta_G}(I^{LR})$, is a genuine HR image is given by $D_{\theta_D}(G_{\theta_G}(I^{LR}))$. Rather than minimising $log[1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))]$ [21], we minimise $-log(D_{\theta_D}(G_{\theta_G}(I^{LR})))$ for improved gradient behaviour.

## 3.2 Convolutional Neural Network

An artificial neural network[3.1] and a convolutional neural network (CNN) vary primarily in a way that an ANN connects every neuron of a layer to every other neuron in the layer before it, but a CNN only allows input from just a restricted selection of units called the unit's receptive field.

CNN, as in figure 3.5, is performed using kernels(e.g. 1x1, 3x3, etc.) that filter out all but the horizontal and vertical variations and use the remaining information to generate features by performing element wise products. Moreover, a nonlinear activation function takes place over all the elements. This layer filters the remaining variations, resulting in a more distinct set of features than before. The kernels used have learning weights that change after each iteration for better performance.

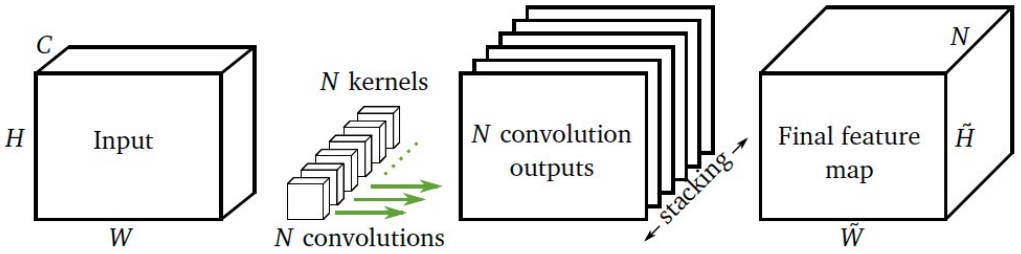


FIGURE 3.4: Figure capturing the working architecture of a CNN model [4].

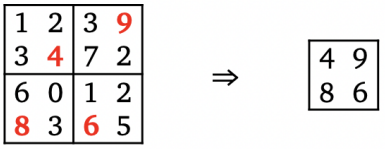


FIGURE 3.5: Figure showcasing the max-pooling operation in CNN [4].

By adjusting the size of a window that is slid over the input in such procedures, it is easy to vary the dimension of the data. The two most common pooling strategies are max and average pooling, where max pooling outputs each window at its maximum value. The result of the pooling procedure for each window in average pooling corresponds to determining the mean value.

## 3.3 Generative Adversarial Networks

A revolutionary generative technique in the field of deep learning is generative adversarial networks. GANs are two competing neural networks that were first developed by Goodfellow,et al. [21] in 2014.

A random input noise vector, z, is translated into a data space by one of the networks, called generator. The generator aims to map z to the actual data x as nearly as it can.The function G (z, g), where G is some modified version of multi-layer convolutional neural network and g specifies its parameters, does this. The discriminators' job in the second network is to calculate the likelihood that the x genuinely originated from the input rather than as an outcome of G. (z). The function D(x, d), where D is also a multi-layer convolutional neural network while d stands for its parameters [21], does this. The objective of training the discriminators is to ensure that they can distinguish between inputs that are created data (G) and genuine data (X) with accuracy (z) . When supplied with synthetic

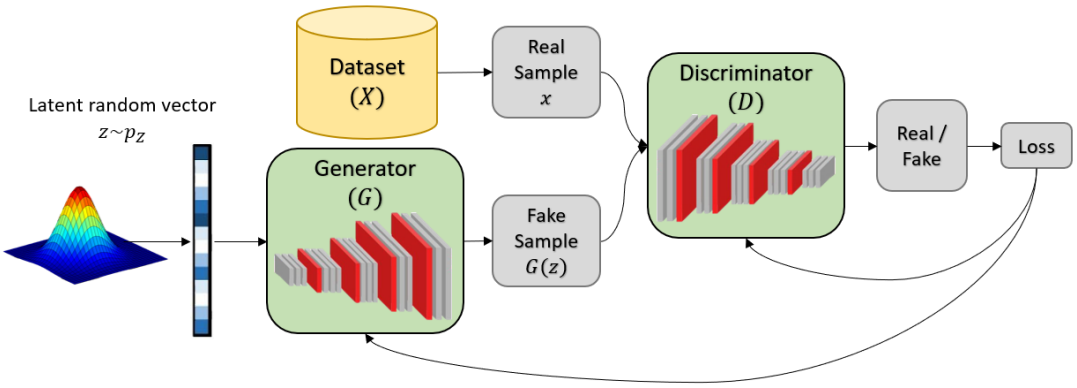


FIGURE 3.6: A schematic view of a GAN. The generator takes a noise vector z as input and maps this to G(z). The discriminator then receives input that are either x (real data) or G(z) (generated data). The discriminator then outputs a prediction if the input data is either fake (0) or real (1). These outputs are used to train the network.[5]

results, the generator is taught to minimise equation 3.15. To put it another way, the generator is taught to make the discriminator anticipate the created data as actual data.

$$log(1 - D(G(z))) \tag{3.15}$$

When two networks are used for this purpose, they engage in a minimax game as shown in equation 3.16 . Argumentative training is the name given to this type of instruction [45].

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(x)))] \tag{3.16}$$

By using loss function, the GAN's goal is to get the synthesized data distribution closer to the genuine distribution. The loss function in equation 3.17 might be determined to be equivalent to the Jensen-Shannon divergence which is stated in equation 3.17 when the discriminator operates at its best.

$$JS_{div}(P \parallel Q) = \frac{1}{2}K(P \parallel M) + \frac{1}{2}K(Q \parallel M) \tag{3.17}$$

where $K$ is the $KL$ divergence and $M = \frac{1}{2}(P + Q)$ [46]. The $KL$ divergence may be determined using the formula 3.18:

The maximum likelihood estimator, known as KL divergence, will approach infinite if the bases for the two distributions do not coincide. The logarithm of two is produced

$$K(P \parallel Q) = \sum_i P(i) log \left( \frac{P(i)}{Q(i)} \right) \tag{3.18}$$

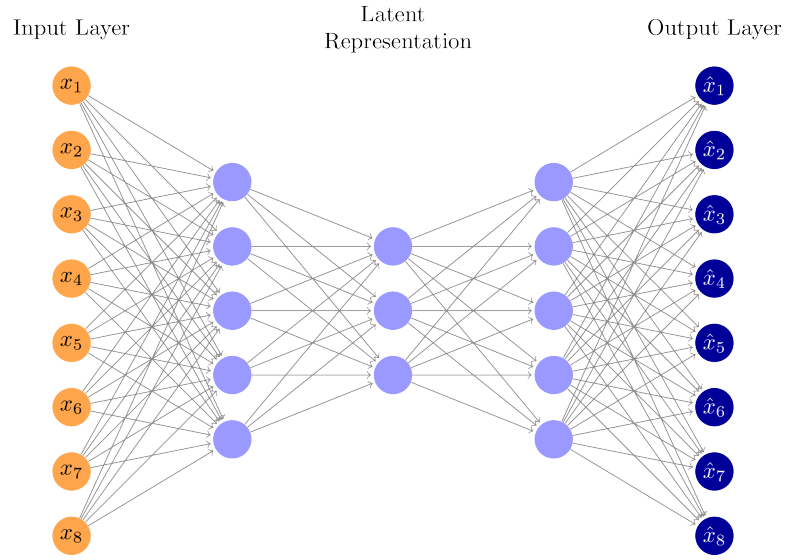


FIGURE 3.7: An architecture of Autoencoder [6]

while combining several KL divergences with the Jensen-Shannon divergence in the same situation with no overlap of the distributions. This results in certain difficulties during adversarial training with the loss function proposed in 3.17.

## 3.4 Autoencoder

An autoencoder (AE) is a method for unsupervised learning that identifies a mapping from high-dimensional inputs to low-dimensional representations, such as latent vectors [47]. The dimensions of this hidden or latent layer are less than the input data in order to avoid the autoencoder from learning the identity transform. Encoders (Enc) and decoders (Dec) are two symmetric neural networks that make up autoencoders. The latent space z, also known as the bottleneck layer because of its lower dimensions compared to the input data, is created when the encoder maps the input x into the latent space. Following that, the decoder will synthesise and reconstruct the input data x using this latent space. The aim of the reconstruction, also known as reconstruction loss, appears to be to minimise the difference between x and x. The reconstruction loss is described in the equation 3.19.

$$\min_{\theta,\phi} L(\theta,\phi) = \frac{1}{N}\sum_{i=1}^{N} ||x_i - Dec_\theta(Enc_\phi(x_i))||_2^2 \tag{3.19}$$

In order to guarantee that the reconstruction will be sufficiently comparable to the original data, the latent space must be taught the most significant feature changes of the original data. The whole network (encoder and decoder) is jointly trained to obtain a low reconstruction loss. In both the encoder and the decoder, autoencoders can contain a single layer, although two or more layers are more typical in these networks. The autoencoders in this situation are deep autoencoders 3.7.

Autoencoders may be divided into four categories: denoising autoencoders, contractive autoencoders, sparse autoencoders, and variational autoencoders .

Denoising autoencoders [48] attempt to denoise or recover a clear picture from a randomly partly distorted input, as their name indicates. These autoencoders fundamentally work on the principle of forcing the hidden layer to acquire robust features rather than merely the identity function, which would lead to yet another malformed picture.

Contractive autoencoders, often known as CAEs [49], are a little more complicated since they add a new component to their loss function to make the model more resistant to minute changes in the input values.

The hidden layer in sparse autoencoders [50] typically has dimensions greater than the input. Only permitting a few of the hidden neurons to be active at any given time solves the issue of avoiding the network to learn the identity function.

Lastly, variational autoencoders, or VAEs, have an autoencoder-like design (Fig. 3.7). Variational autoencoders use a variational strategy for latent representation learning, which is the primary distinction between them and other types of autoencoders.

## 3.5 Remote Sensing

Remote sensing is the acquisition of information from a distance. Space agencies observe Earth and other planetary bodies via remote sensors on satellites and aircraft that detect and record emitted and reflected energy. Remote sensors, which provide a global perspective and a wealth of data about Earth systems, enable data-informed decision making

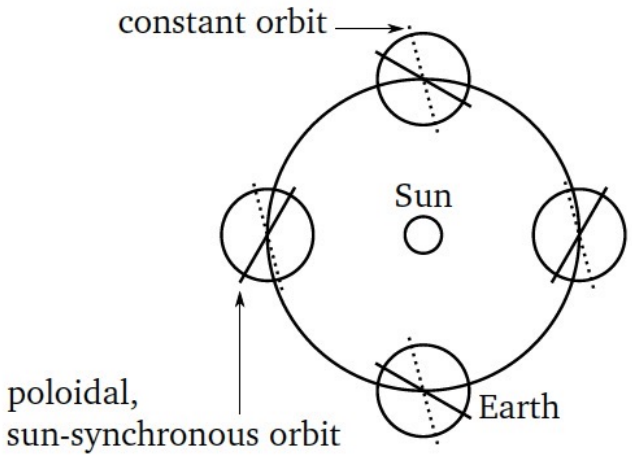


FIGURE 3.8: In the sun-synchronous orbit (solid), the earth's rotation plane revolves once a year. The non-spherical form of the earth imparts intrinsic angular momentum that propels this revolution. The rotational plane of an unaltered orbit, on the other hand, preserves its alignment [4].

based on the current and future state of our planet. Some common methods followed by space agencies for remote sensing are mentioned below:

- **Observing with the Electromagnetic Spectrum:** The oscillation of charged particles creates electromagnetic energy, which propagates as waves across the atmosphere and the emptiness of space. The amount of energy that each object on Earth reflects, absorbs, or transmits varies depending on its wavelength, and each object has a distinct spectral fingerprint.

- **Sensors:** To measure the energy which is bounced back, sensors or equipment on board satellites and aeroplanes either utilize the Sun as a light source or create their stream of illumination. Passive sensors are those that rely on solar energy from the Sun, and active sensors are those that have their internal energy source. Altimeters, scatterometers, and other radio detection and ranging (radar) sensors are examples of active sensors.

- **Resolution:** The utilisation of sensor data depends in part on resolution. Depending on the satellite's orbit and sensor configuration, resolution can change. For each dataset, there are four different forms of resolution to take into account: radiometric, spatial, spectral, and temporal.

- **Data Processing and Analysis:** Before the majority of researchers and consumers in applied science can make use of remote sensing data from instrumentation onboard

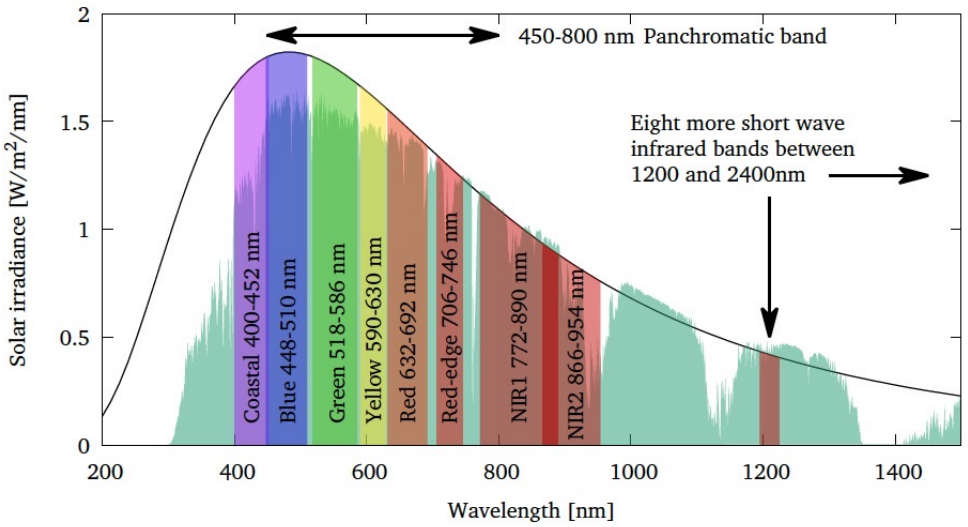


FIGURE 3.9: The wavelengths of the WorldView-3 imaging bands [3, 4], the solar radiation spectrum, and the spectral response of a pure black body at 5500 K [4].

satellites, the processing is necessary. They may be processed and used for several purposes, including agriculture, water resources, health, and air quality.

We will use satellite-sourced pictures for this project. Knowing the origins of the data is necessary in order to comprehend and analyse it. We will go into more detail about the difficulties in using the data source and discuss it in this part.

### 3.5.1 Satellite Imagery

There are several distinct remote sensing satellites in orbit now, each with a particular purpose. Mainly optical sensors will be the subject of our attention since they are the most accessible and relevant for deep learning image processing jobs. The Landsat satellites, for example, have been in orbit for several decades, providing the chance to analyse the temporal history of land cover. The majority of optical remote sensing satellites travel in poloidal, sun-synchronous lower earth orbit, with mean altitudes ranging from 450 km to 800 km.

Two key benefits of the sun-synchronous orbit 3.8 are as follows: Firstly, the satellite may be positioned in continuous sunshine, and secondly, with the exception of seasonal variations, a certain location on the planet is always shot in the same lighting circumstances. This makes captured photos more comparable. These satellites can return to each region

of interest within a few days or even once every day because the sun-synchronous orbital plane and the planet both spin slowly. Since the satellite's movement already covers one imaging dimension, many satellites scan the globe using line sensors, although two-dimensional sensors are also used. A zone of a specific width is captured by the satellite based on elevation, focal length, and sensor (the swath width) at ground level.

A specific range of wavelengths are used by each satellite to acquire photographs. The WorldView-3 satellite's imaging bands, which we shall employ subsequently, are summarised in Figure 3.9.

The near infrared, red, green, and blue bands are crucial for classifying land use pattern. Longer wavelengths are typically employed for geological surveys, military purposes, or fire detection. The quantity of vegetation covering the ground may be determined, in part, by comparing the red and red-edge or NIR bands.

# Chapter 4

# Methodology

In this section, we will take a closer look at the models employed in this work. The models used are EDSR, SRGAN and SR(PRE) or SRResNet. Because in the Deep Learning landscape, all of them are prominent state-of-the-art methods. Other very new methods, including VAE and transformers, are not performing well enough till now. Though very few recent studies show some promising results, they These models need robust testing and improvement to perform better consistently than GAN and CNN-based models. All of the models mentioned here were compared against each other and with Bi-cubic interpolation technique which is a classical SR method.

## 4.1 Bi-cubic Interpolation

Bicubic interpolation is a two dimensional approach for enhancing and expanding digital photographs using cubic splines as well as other polynomial techniques. When upscaling or resampling an image, retouchers and editors frequently employ it in computer image manipulation. The 2 interpolation methods that are often used are adaptive and non-adaptive. While non-adaptive approaches treat all pixels identically, adaptive techniques vary depending on what they will be interpolating. While non-adaptive algorithms include nearest neighbour, bilinear, bicubic, spline, etc., adaptive strategies are utilised in many proprietary approaches in specialist professional picture editing software. In general, Lagrange polynomials, cubic splines, or cubic convolution techniques can be used to do bicubic interpolation. Bicubic interpolation produces nicer interpolated surfaces than either bilinear interpolation or nearest-neighbor interpolation. Both Lagrange polynomials, cubic splines, or the cubic convolution technique can be used to do bicubic interpolation.

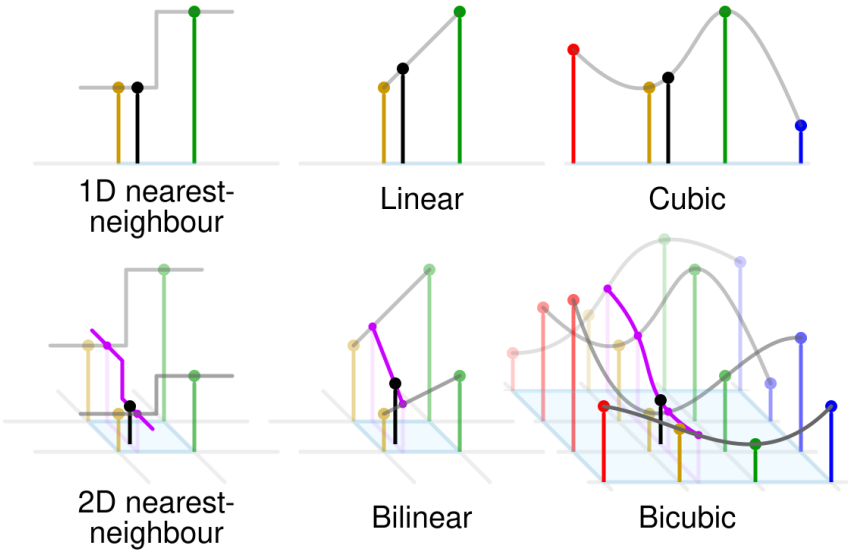


FIGURE 4.1: Bicubic interpolation is compared to a few 1- and 2-dimensional interpolations. Red, yellow, green, and blue dots represent the adjacent samples, while black dots represent the interpolated position. Their values are in line with their heights above the surface. [7]

When speed is not a concern, bicubic interpolation in image processing is frequently preferred to nearest-neighbor or bilinear interpolation in photo resampling. Bicubic interpolation considers 16 pixels (4 x 4) as opposed to bilinear interpolation's consideration of just 4 pixels (2 x 2). Bicubic interpolation produces smoother and less erratic images when resampling images. The Bicubic Interpolation methodology is applicable to both the upsampling and downsampling of pictures for various applications. It has been used in several works as a common benchmark to measure all other machine or deep learning models against because it is one of the most well-known classical super resolution techniques. It is also used in the same way in this research. In accordance with Gavade and Sane's 2013 [51], the bicubic interpolation is as follows. Assume that the values of the function f and its derivatives $f_x$, $f_y$, and $f_{xy}$ are known at the unit square's four corners (0, 0), (0, 1), (1, 0), and (1, 1). Consequently, the interpolated surface may be expressed as the Equation 4.1. The interpolation problem then consists of determining 16 coefficients

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j. \tag{4.1}$$

$a_{ij}$.

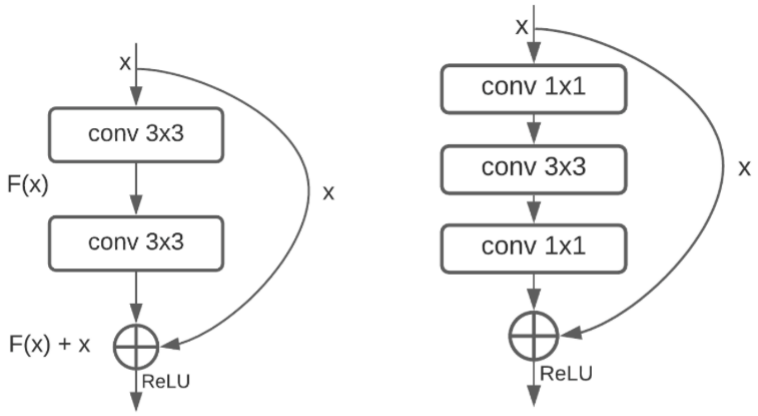


FIGURE 4.2: The ResNet architecture [8]

## 4.2 SR(PRE)

The SR-ResNet model is referred to as SR (PRE) in this study as it only uses and trains the Generator Network of the SRGAN model (described in section 4.4). A Resnet or residual network called SR-ResNet is utilised for Super Resolution tasks. In the subsequent section 4.2.1, the ResNet's architecture and workings are explained. The weights from this pre-trained generator, SR (PRE), are further employed as input weights in the SRGAN model for experimentation-related GAN fine-tuning.

### 4.2.1 ResNet

Residual networks, commonly known as ResNets, were initially developed in He et al. 2016 [18] to address the degradation problem in deep neural networks, in which the efficiency of network systems with several layers quickly declines just before the model is ready for convergence. As illustrated in fig. 4.2, which shows one residual block, the ResNet is made up of neural networks with skipping connections.

The outcome of the stacked layers is combined with identity mapping performed by the skip connections to produce $F(x) + x$. This manner, rather of learning via $F(x)$ in the usual situation without the skip connection, the identity mapping $x$ may easily be constructed in the scenario when it is the best solution by simply setting $F(x)$ to zero. 34 convolutional layers, 3x3 filters, and a ReLU activation function are the components of

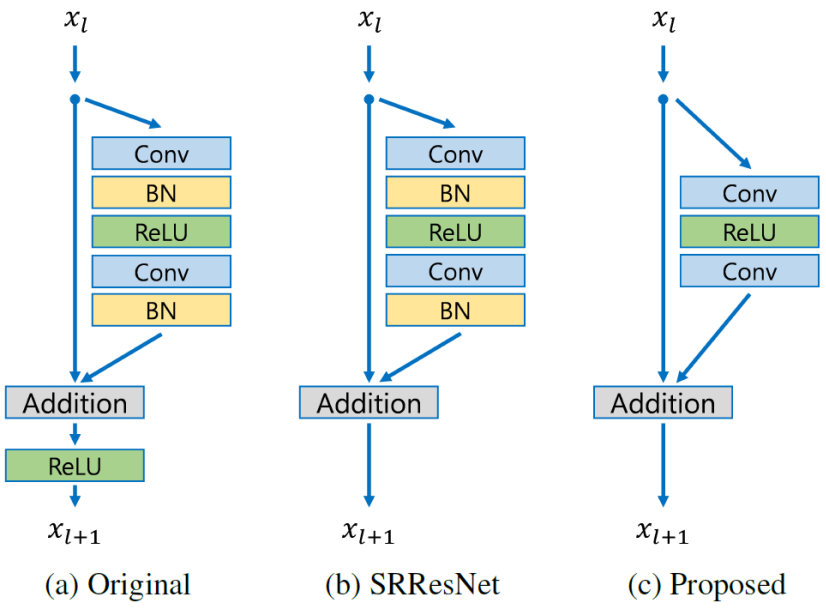


FIGURE 4.3: The EDSR architecture compared with SRResNet and Original ResNet structure [9]

the original ResNet(as shown in fig. 4.2.a). Another variation of the ResNet is the bottleneck architecture designed for deeper networks, which has a structural component that has three layers rather than two, the center one of which acts as a bottleneck with 3x3 convolutions, and the other two of which function as the rest of the network (as portrayed in fig. 4.2.b).

## 4.3 EDSR

EDSR, or Enhances Deep Super Resolution Model, is a deep neural network used for image superresolution. The network design of EDSR It was invented by Bee Lim et al. [9] in 2017. EDSR is based on the SRResNet architecture (4.2) and is simplified by analyzing and deleting unnecessary modules. Numerous super-resolution approaches have been compared to EDSR. SRResNet-like topologies are employed. Two layers of Convolution and one layer of ReLU activation function make up EDSR, which was heavily influenced by SRResnet. However, the batch normalisation of the residual block has been removed by the authors for increased effectiveness and performance. The architecture of residual block and single scale EDSR is given figure 4.3.

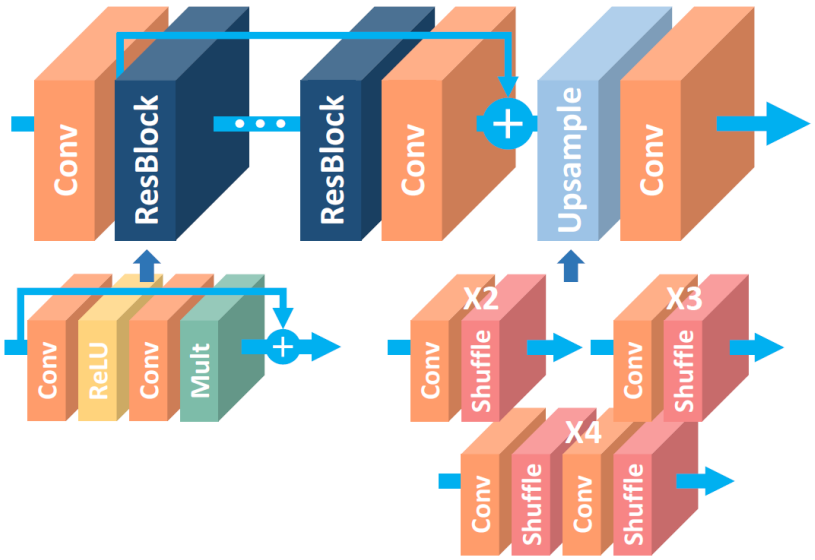


FIGURE 4.4: The architecture of single scale EDSR residual block [9]

### 4.3.1 Single-Scale EDSR Model Architecture

- The depth (number of layers) B and width (number of feature channels) F of a typical CNN architecture occupy around O(BF) memory with O(BF2) parameters.
- Each layer consists of a residual block and convolution layers. Outside of the residual blocks, ReLU activation layers don't exist.
- Thus, while taking into account restricted computational resources, raising F rather than B can optimize the model's capability.
- The learning becomes inconsistent even as number of feature maps rises, though.
- As recommended by Inception-v4 by Szegedy et.al [52], residual scaling with a factor of 0.1 is done at the residual path prior putting back to convolutional path. Constant scaling layers are positioned after the final convolution layers in each residual block. For improved computing efficiency, this layer can be combined with the prior convolution layer during the testing phase.
- Outside of the residual blocks, ReLU activation layers don't exist.
- Baseline model: Since each convolution layer only uses 64 feature maps, residual scaling layers really aren't needed.
- EDSR: By setting B = 32 and F = 256 with a scaling factor of 0.1, the basic model is extended.

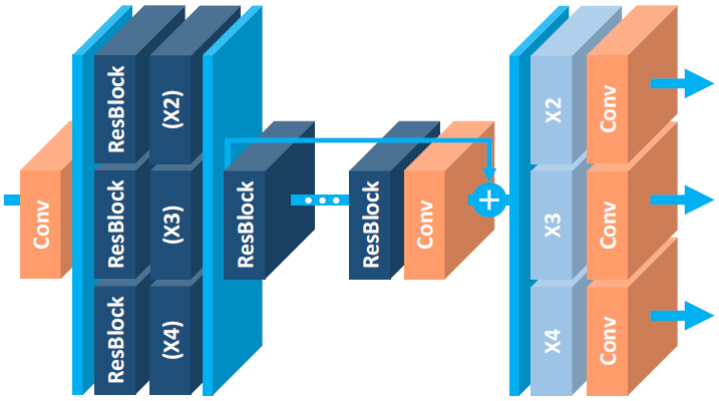


FIGURE 4.5: Multi-Scale EDSR Model [9]

### 4.3.2 Multi-Scale EDSR Model Architecture(MDSR)

- Super resolution at various scales is a challenge that is connected to others.
- Baseline Model: As seen in figure 4.5, the basic model is a single main branch with B = 16 residual blocks allows for the majority of the parameters to be shared across various sizes.
- In order to limit the variance resulting from input pictures of various scales, pre-processing modules are first placed at the head of networks. Two leftover blocks with 55 kernels make up each pre-processing module. The scale-specific portion can be shallow while the wider receptive field is covered in the early stages of networks by using larger kernels for pre-processing modules.
- Scale-specific upsampling components are placed in parallel at the multi-scale model's endpoint for performing multi-scale reconstruction.
- The basic multi-scale models has only 3.2M parameters, with comparable performance to the single-scale models, while the baseline single-scale models for the three distinct scales have around 1.5M parameters apiece, totalling 4.5M.
- As the scale-specific components are lighter than residual blocks components, the MDSR requires just 2.5 times as many parameters as the basic multi-scale model, with B = 80 and F = 64, or around 5 times greater depth.

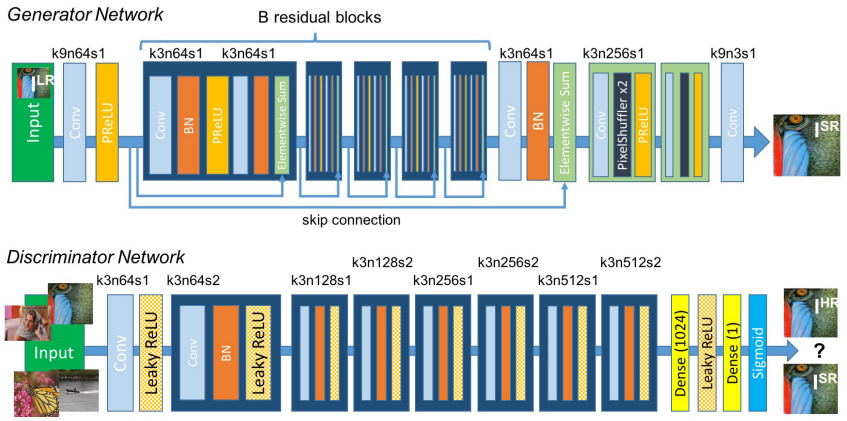


FIGURE 4.6: Architecture of the Generator and Discriminator Networks with the associated kernel size (k), number of feature mappings (n), and stride (s) for each convolutional layer. [10]

## 4.4 SRGAN

Based on ResNet architecture and tuned to create a new perpetual loss function that comprises of an adversarial loss and a content loss, SRGAN is a generative adversarial network for single picture super-resolution. Utilizing a discriminator network that has been trained to distinguish between the super-resolved pictures and the original photo-realistic images, the adversarial loss drives the output to the natural image manifold. The authors also employ a content loss that is driven by perceptual similarity rather than similarity in pixel space. It is a state-of-art network to estimate a super-resolved image from a low-resolution image with high upscaling factors(4x). The diagram 4.6 illustrates the architecture of the SRGAN Model for various feature maps, kernel sizes, and strides.

### 4.4.1 Generator Network

The Generator Network seeks to produce SR images that closely resemble the actual images, making it challenging for the Discriminator to categorise them. It basically comprises of an SR-ResNet generating network, where the low-resolution input is routed via a Parametric ReLU [section: 3.1.2.4 ]layer after first passing through a convolutional layer made up of 9x9 kernels and 64 feature maps. It is clear that the Parametric ReLU serves as the main activation function across the whole generator design. Since it is one of the finest non-linear functions for this specific purpose of translating low-resolution pictures to high-resolution images, the Parametric ReLU was chosen.

Several B residual blocks are used in the subsequent layer of the feed-forward fully convolutional SR-ResNet model. Following a batch normalisation layer, a Parametric ReLU [section: 3.1.2.4] activation function, a convolutional layer with batch normalisation, another convolutional layer with batch normalisation, and a final elementwise sum technique, each residual block has a convolutional layer with 3x3 kernels and 64 feature maps. The final result produced from the elementwise sum technique combines the feed-forward and skip connection outputs.

The remainder of the generator model is produced once the remaining blocks have been created, as seen in the picture 4.6. In this generator model architecture, the convolutional layer is 4x upsampled before being utilised to create the super-resolution pictures. The pixel shufflers insert values into the height and width dimensions using values from the channel dimension. With the help of two pre-trained sub-pixel convolution layers, the image's resolution is enhanced.

### 4.4.2 Discriminator Network

To differentiate between the produced SR pictures and the raw HR image, the discriminator architecture is utilised. It is designed to assist a standard GAN method [Fig:3.6] in the best possible way. The discriminator and the generator are concurrently getting better as they compete with one another. The generator tries to create realistic pictures so that it can avoid detection by the discriminator network while the discriminator network searches for the bogus images. Similar principles apply to how SRGANs operate, where the generative model G aims to deceive a differentiable discriminator D that has been taught to discern between genuine pictures and super-resolved images.

The discriminator architecture that has been created is highly logical and simple to comprehend. It employs a Leaky ReLU activation function after a first convolutional layer. For this architecture, the $\alpha$ value in Leaky ReLU's [section:3.1.2.4] is set at 0.2. The batch normalisation layer as well as the Leaky ReLU activation function are then followed by a number of convolutional layer recurrent blocks. After five of these repeating blocks, the dense layers and sigmoid activation function are used to carry out the classification activity. Keep in mind that the baseline convolutional size is 64 $\times$ 64, which is doubled by 2 after two full blocks of each until we reach the 8x upscaling factor of 512 x 512. The generator learns more efficiently and generates improved outcomes thanks to this discriminator model.

# Chapter 5

# Experimentation

This chapter discusses the experimental process, dataset, and data preparation with all the relevant details. The data description as well as georeferencing and meta-data information is mentioned in detail the section 5.1. As the dataset is contains georeferencing information, the data pre-processing is very different from regular images as mentioned in the section 5.2. The study uses the relevant pretrained deep learning models for super resolution and separates the data into training and testing data [Section: 5.3] for the transfer learning of the pretrained models in the subsequent stages of the investigation. The results of all pre-trained as well as trained models (transfer learning) were compared to a benchmark classical SR approach (bicubic interpolation) as well as with each other. The standard evaluation matrices which are used to measure the performance of the models with their theoretical formulation is shown in the section 5.4.

## 5.1 Dataset

Two satellite images were used for training and testing of the models. Among them, one is a High Resolution (HR) image and the other one is a Low Resolution (LR) image as portrayed in fig 5.2. The HR image in greyscale since it contains only 1 band. On the contrary, the LR image, having 3 bands, is in false colour RGB format. As both are satellite images, they also contain projection and Geo-Transformation information. The projection details of both the images are the same since they have been captured at the same location and time. The project details are given in table 5.1 and code snippet 5.1.

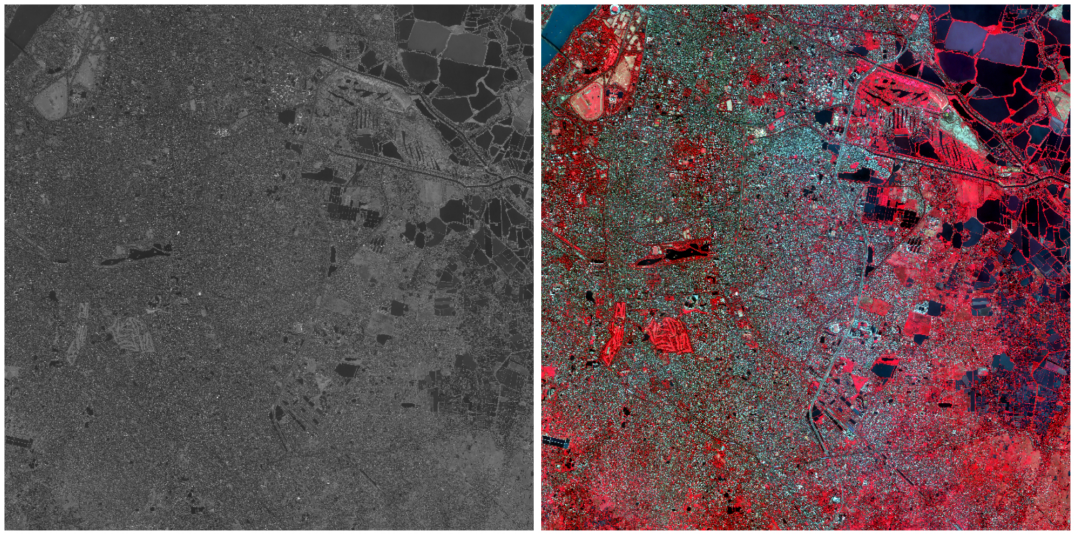

FIGURE 5.1: Data is displayed in the figure. The HR picture with one band is exhibited on the left, while the LR image with three bands is shown on the right.

```
Image Metadata (Coordinate System):

PROJCS["WGS 84 / UTM zone 45N",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",87],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
```

```
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32645"]]
Corner Coordinates:
Upper Left  (  635809.989, 2496067.008) ( 88d19'15.40"E, 22d33'57.23"N)
Lower Left  (  635809.989, 2482963.008) ( 88d19'11.36"E, 22d26'51.17"N)
Upper Right (  649945.089, 2496067.008) ( 88d27'30.23"E, 22d33'52.96"N)
Lower Right (  649945.089, 2482963.008) ( 88d27'25.76"E, 22d26'46.92"N)
Center      (  642877.539, 2489515.008) ( 88d23'20.69"E, 22d30'22.13"N)
```

TABLE 5.1: The Meta-Data information of the LR and HR images

| **Properties** | **Low Resolution** | **High Resolution** |
|---|---|---|
| Driver | GTiff | GTiff |
| Dtype | uint16 | uint16 |
| No data | None | None |
| Width | 5049 | 20193 |
| Height | 4681 | 18720 |
| Count | 3 | 1 |
| CRS | $CRS.from\_epsg(32645)$ | $CRS.from\_epsg(32645)$ |

## 5.2 Data Pre-processing

- At first, all the geo-transformation and the co-ordinate information are separated from the image and stored securely to be utilised during the data post-processing step. According to table 5.1 the Origin X & Y co-ordinates are used to identify the

TABLE 5.2: Geo Transformation information of the LR and HR images

| **Geo Transformation** | **Low Resolution** | **High Resolution** |
|---|---|---|
| Origin X Co-Ordinate | 635809.9893442297 | 635809.9893442297 |
| Pixel Width | 2.8 | 0.7 |
| X Pixel Rotation | 0 | 0 |
| Origin Y Co-Ordinate | 2496068.408324141 | 2496068.408324141 |
| X Pixel Rotation | 0 | 0 |
| Pixel Height | -2.8 | -0.7 |

top-left point in the image. Pixel width and height are the ratio of pixels to geo-coordinates in the X and Y directions. The rotation of X & Y coordinates signifies how much the satellite images are tilted from their actual geo-position.

- Before loading two input images for further processing, the images are divided into small patches using the patchify [53] library in Python to make them easier to work with. LR images are divided into 64x64 patches, whereas patches of HR images are of size 256x256. The above mentioned geo-transformation information is necessary to identify the geo-location of each patch and their respective coordinates on the earth's surface. This information is extracted using the GDAL[54] library in Python.

- LR image contains 3 bands, which must be separated before slicing into smaller patches and re-assembled before further preprocessing. Since the HR image contains only 1 band, the slicing can be done right away.

- All the image patches are converted to grayscale, i.e., in the range of (0, 255), as the network is trained to perform well on gray-scaled images.

- Also, all the images are converted into Uint8 from Uint16 as the networks are designed to work with Uint8 values.

- Model based Pre-processing: For prediction purposes, the proposed models can work with any dimensional images. However, in the training phase, the SRGAN model needs images of a specific size. To achieve this, all the LR training images are divided into 32x32 patches and the HR training images are chopped up into 92x92 patches.

## 5.3 Train-test Setup

As previously mentioned in section 5.2, LR images are divided into 32x32 training sets whereas HR images into 92x92 training sets before feeding to the SR models for Transfer Learning. A total of 37,560 images are produced in a training set with batch size = 3, which is further separated into train-validation sets as **Training:** 26292 and **Validation:** 7512 .

The train-validation split consists of (70% + 20%) of data, while the remaining 10% is manoeuvred for testing purposes. Training of the 3 models using transfer learning is performed in the train set with batch size = 3 and buffer size = AUTOTUNE. The models are trained for 1000 Epoches which takes approx *1.5 hour* for SRGAN and *1 hour* for both

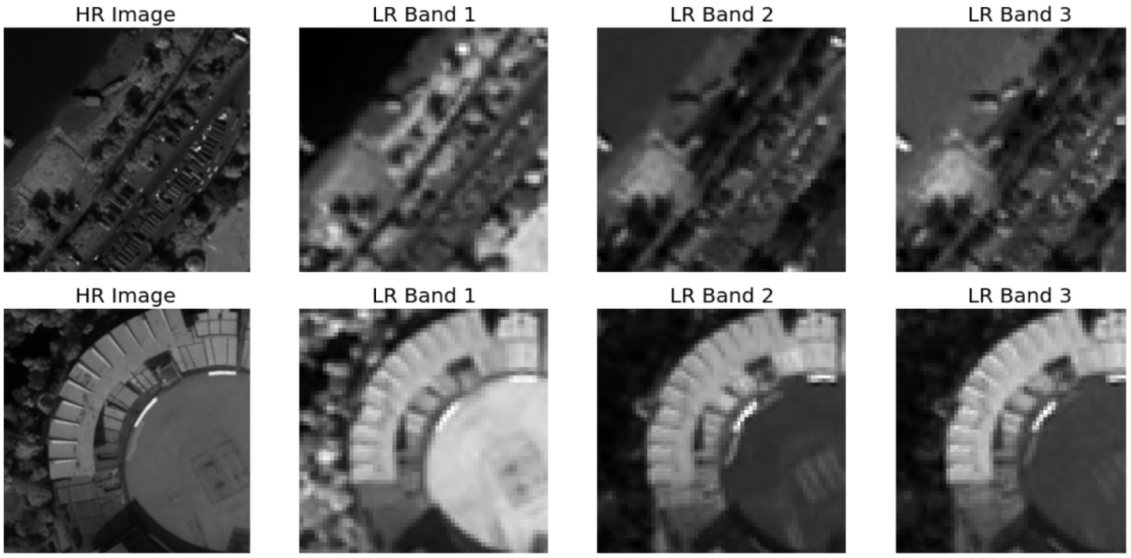


FIGURE 5.2: Sample patches with 3 LR and 1 HR band

EDSR and SR(PRE) each. For the bi-cubic interpolation, since it is a classic model, no further transfer learning or training has been performed over it.

All the models are tested with pre-trained weights which have been trained on the DIV2K dataset ( with PSNR on the DIV2K validation set = 28.89 dB (images 801 - 900, 6 + 4-pixel border included for EDSR and for SRGAN 1.55M parameters, trained with VGG54 content loss).

## 5.4 Evaluation

Subjective and objective approaches can be used to evaluate image quality [55]. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) are examples of objective approaches that compare quantitative parameters with an underlying data, in this instance the form of a reference picture [56]. Although PSNR and SSIM have evolved into common assessment techniques for computer vision applications, their outcomes can be rather deceptive when evaluating the quality enhancements of a picture [57]. When evaluating a picture, the Human Visual System (HVS) takes additional factors into consideration. When viewed by the HVS, the visual quality of image pairs with nearly equivalent PSNR value might vary substantially.

### 5.4.1 Root Mean Squared Error (RMSE)

As the square root of MSE(eq. 5.2), the Root Mean Square Error (RMSE) measures the amount of change per pixel caused by image processing. The most used estimator of an image quality measuring parameter is MSE. It is a complete reference metric, and the closer the number is to zero, the better the performance.

$$MSE = \frac{1}{mn}\sum_{x=0}^{m}\sum_{y=0}^{n}[f(x,y) - h(x,y)]^2 \tag{5.1}$$

$f(x,y)$: HR Image, $h(x,y)$:LR Image, $m,n$: No.of pixels in rows and columns

$$RMSE = \sqrt{MSE} \tag{5.2}$$

### 5.4.2 Peak Signal-to-Noise Ratio (PSNR)

The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed or reconstructed image. In order to calculate the PSNR, the MSE (mean square error) of the HR reference image, $f(x,y)$, and the processed image(LR Images) , $h(x,y)$, is used as in Equation 5.1. The MSE will indeed be near to 0 if the transformed picture resembles the reference image. As MSE approaches zero, the PSNR, which has the mathematical formula 5.3 will reach infinite. The similarity between $h$ and $f$ is strong when the PSNR value is high. A low

$$PSNR = 10 * \log_{10}\left(\frac{MAX_f^2}{MSE}\right) \tag{5.3}$$

PSNR value suggests that the pictures have numerical disparities. PSNR is measured in $dB$.(decibel).

### 5.4.3 The Structural Similarity Index (SSIM)

SSIM is a method for quantifying how similar two images are to one another. The SSIM index is a complete reference metric, meaning that the initial uncompressed or distortion free picture serves as the baseline for the measurement or prediction of image quality. A

transformed picture (LR Image), $h$, and a HR reference image, $f$, are both used by SSIM in its evaluation procedure. However, it evaluates resemblance of the photos using the known quality perception of the HVS rather than computing the difference between the two images. As a result, SSIM is the result of three assessments established between the two pictures.

The *luminosity* is the very first parameter represented by equation 5.4.

$$l(f,h) = \frac{(2\mu_f\mu_h + c_1)}{({\mu_f}^2 + {\mu_h}^2 + c_1)} \tag{5.4}$$

The *contrast* is the second parameter represented by equation 5.5.

$$c(f,h) = \frac{(2\sigma_f\sigma_h + c_2)}{({\sigma_f}^2 + {\sigma_h}^2 + c_2)} \tag{5.5}$$

The *structure* is the third parameter represented by equation 5.6.

$$s(f,h) = \frac{(2\sigma_{fh} + c_3)}{({\sigma_f}^2{\sigma_h}^2 + c_3)} \tag{5.6}$$

Here, the positive constants $c_1, c_2$, and $c_3$ are employed to prevent division by zero. The final equation 5.7 is obtained when $c_3 = c_2/2$:

$$SSIM(f,h) = l(f,h)c(f,h)s(f,h) = \frac{(2\mu_f\mu_h + c_1)(2\sigma_{fh} + c_2)}{({\mu_f}^2 + {\mu_h}^2 + c_1)({\sigma_f}^2 + {\sigma_h}^2 + c_2)} \tag{5.7}$$

here $\mu_f$ is the mean of $f$, $\mu_h$ is the mean of $h$, ${\sigma_f}^2$ is $f$'s variance, ${\sigma_h}^2$ is $h$'s variance, and $\sigma_{fh}$ is $f$ and $h$'s covariance. The Structural similarity value ranges from 1 to 1, where a value of 1 denotes complete identity between the pictures and a value of 0 denotes no structural similarity.

### 5.4.4 Visual inspection

After being subjected to super-resolution processing, certain features could surface that are challenging for objective assessment techniques to evaluate. When comparing before-and-after photographs, some subjective parameters such as visual inspection are required

because, in certain conditions, in spite of giving better performance in terms of objective measurement, the generated image is not quite a good representation of the actual image.

## 5.5 Data Post-processing

The satellite image Geo-transformation information is added after processing the images. The origins of all small patches are shifted according to the coordinate transformation. After the low-resolution images are turned into the high-resolution images, the pixel width and height information are transformed accordingly ( i.e. Pixel-width / Scaling-factor). After the small patches of Geo-referenced images are created, another large Super Resolution images is produced by assembling the small images. The Geo-referencing to the large image also attached after changing the scales accordingly.

# Chapter 6

# Results

In this chapter the pre-trained models, and models after transfer learning with the pre-trained weights, are compared against each other visually as well as through various metrics. Image patches from the original image were applied to the model for visualization before and after transfer learning. The performance of the models was compared using image quality metrics for quantitative measurement such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Root Mean Squared Error (RMSE).

## 6.1 Overview of Results

The brief overview of sample patches can be visually identified. The SR GAN performs significantly better than other models. Whereas SR (PRE) performs the worst among them. The output produced by the benchmark model after bicubic interpolation is more blurry. On the other hand, the output from the deep learning models consists of artifacts. It could be due to the different outcomes produced by each of the 3 layers, which couldn't be assembled properly.

Another important aspect to consider is that the images produced before transfer learning [fig:6.1] are more vibrant than the images that are produced after it [fig: 6.2] in the same false colour mapping. It can be portrait as a result of the change in the pixel values after super resolution. During training, grayscaled HR images can be the reason for this phenomenon, as the loss function tries to minimise the differences between produced and actual images, which in turn reduces the vibrancy of the images and makes them more inclined towards grayscale HR images.

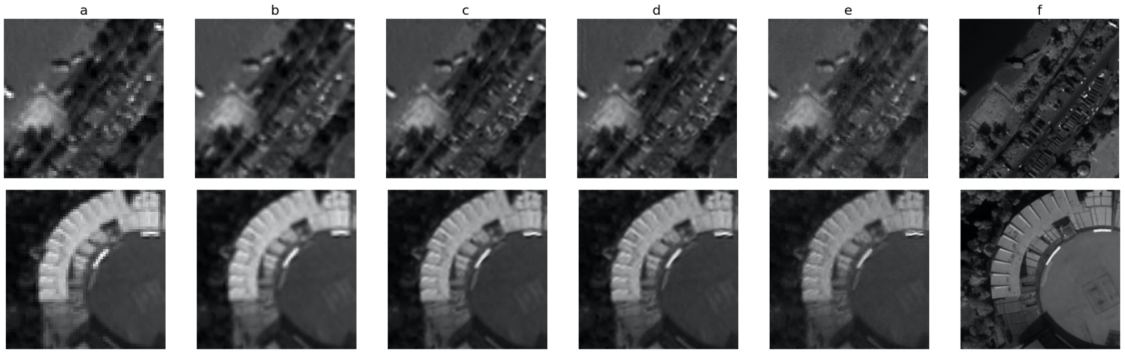


FIGURE 6.1: Sample patches of Image 3 and Image 5 before transfer learning (a) Input Image, (b) Bicubic Interpolation, (c) EDSR, (d) SR(PRE), (e) SR(GAN), Original Image

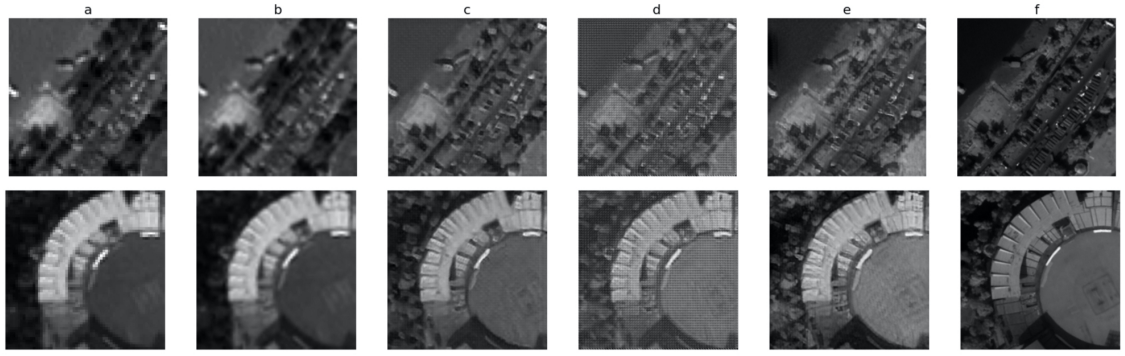


FIGURE 6.2: Sample patches of Image 3 and Image 5 after transfer learning (a) Input Image, (b) Bicubic Interpolation, (c) EDSR, (d) SR(PRE), (e) SR(GAN), Original Image

## 6.2 Quantitative results

For the quantitative estimation purposes of the results that have been generated through the CNN and the GAN networks, the PSNR, SSIM, and RMSE matrices are used for comparison. All of them are already discussed theoretically in the section 5.4. The results of the sample of five images were taken into account. Among them, the two images are shown in the figures 6.1 and 6.2. For visualisation purposes, each of the matrices is compared and plotted using a heatmap. Both the cases, i.e., before and after transfer learning, are studied this way in Figures 6.3 and 6.4 .

It is evident from the figure 6.3 that the PSNR values for image 1 are lower than their counterparts. However, Image 2 has the highest PSNR out of all of them. The outputs of Image 4 and the HR image, on the other hand, exhibit very little structural resemblance according to the SSIM findings, whereas Image 5 and its SR image exhibit significant structural similarity. The RMSE value, on the other hand, proposes that only Image 1

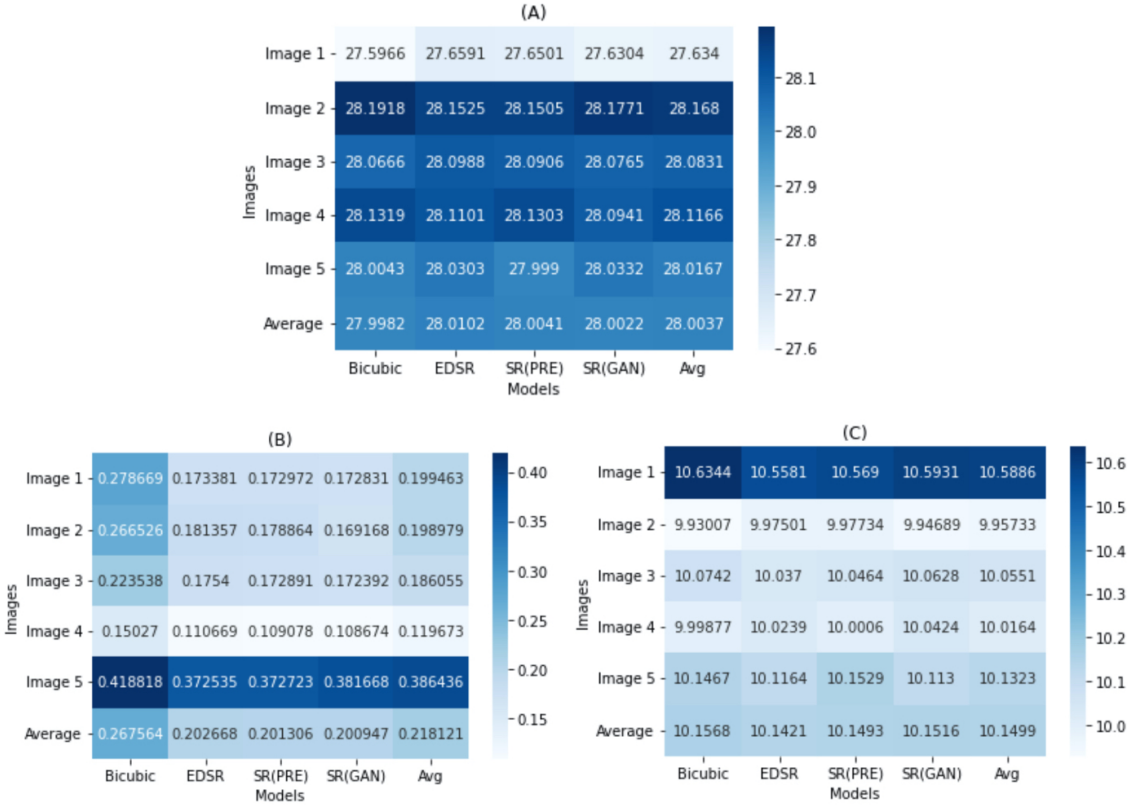


FIGURE 6.3: Comparison of the matrices before transfer learning (A) PSNR Values, (B) SSIM Values, (C) RMSE Values

has a significantly higher error than all other images, which is contradictory to the results produced by the PSNR metric.

The quantitative outcomes of SR pictures after transfer learning, however, reveal a totally different scenario. With the exception of one extract, the EDSR result from the fifth image, the PSNR readings are essentially the same. While the eye inspection contradicts this discovery, the Bicubic interpolated images appear to have a higher degree of structural resemblance. With the highest errors both before and after the training, Image 1 is consistently produced only by the RMSE.

The perceptual loss mentioned in the section 3.1.4.1 is taken into consideration when comparing the results. The results are taken before and after 1000 epochs of training of each of the 3 models. The figure 6.5 clearly shows the decrement of the perceptual loss after training. As perceptual loss is created to depict the human visual responses, it matches our visual intuition.

A significant boost in the performance of the models after transfer learning was anticipated, according to the human perception of images and perception loss. whereas all

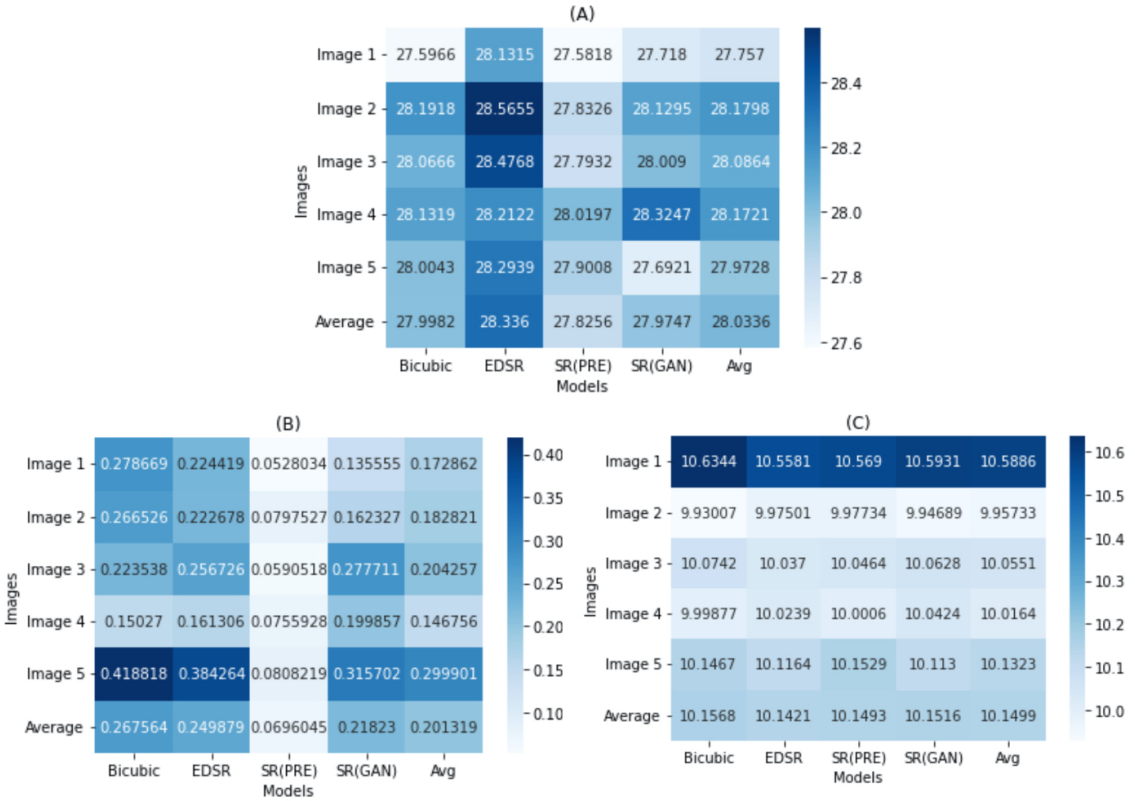


FIGURE 6.4: Comparison of the matrices after transfer learning (A) PSNR Values, (B) SSIM Values, (C) RMSE Values

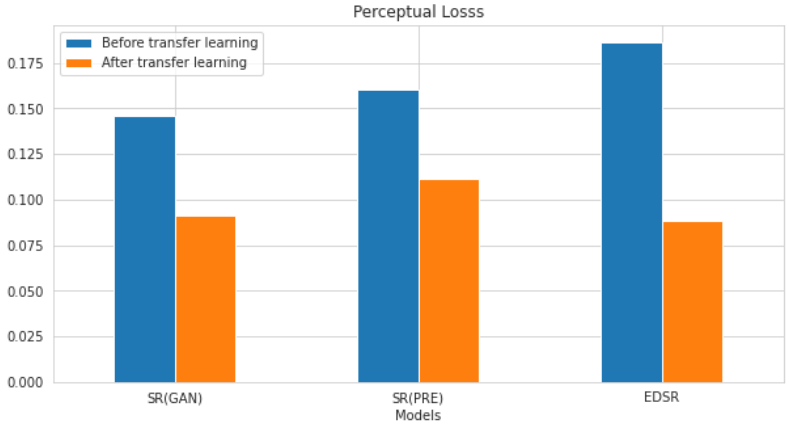


FIGURE 6.5: Comparison of the perceptual loss before and after transfer learning

other matrices are very conflicting and contradictory in nature. SSIM and PSNR change significantly, and RMSE remains almost the same before and after Transfer Learning.

The graph in the figure 6.6 shows a slight increase in the efficient performance of models after transfer learning. However, the dramatic improvement as shown by the pictures 6.1

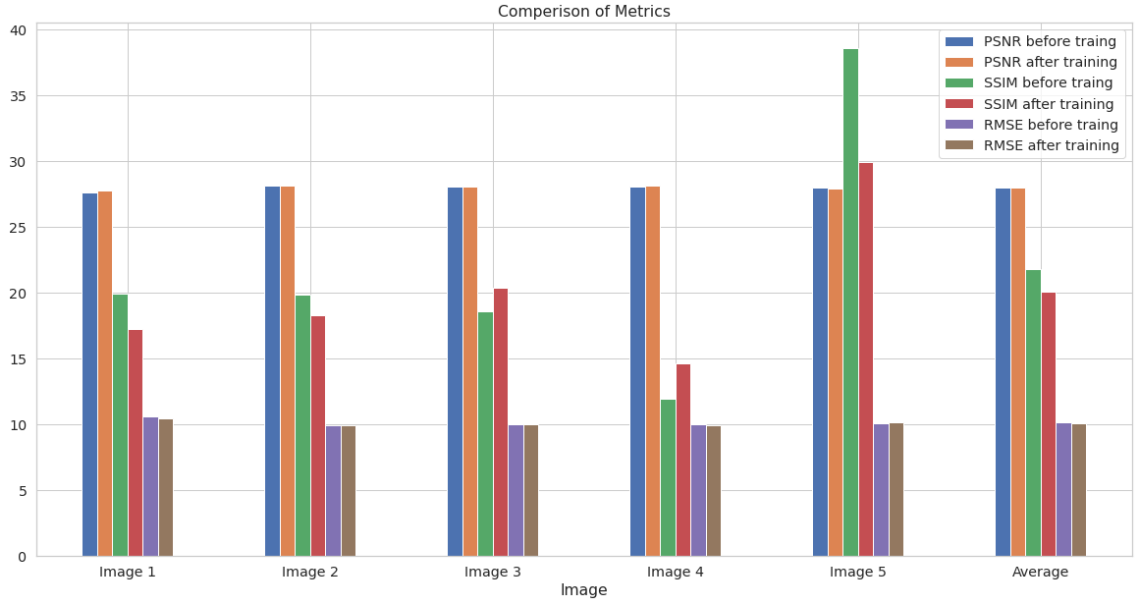


FIGURE 6.6: Comparison of all the matrices before and after transfer learning (Here SSIM is scaled 100x for visual comparison and Average signifies the average matrices of all the images in different models)

and 6.2 is not reflected in the figure 6.6. The new PSNR and new RMSE(after transfer learning) showcase a slight boost in performance. However, the old SSIM(before transfer learning) and the new SSIM(after transfer learning) exhibit unexpected results. In two instances, the new SSIM shows better results than the old SSIM. In the other 3 instances, the old SSIM shows similar or slightly better results than the new SSIM.

The poor performence with respect to the evaluation matrices can be because of the artifacts which are generated during SR processed after transfer learning of the discueed models. As the artifacts can destroy the image's structural and textural consistency.

# Chapter 7

# Final Remarks

Satellite image super resolution is a growing field of study which need to be addressed with specific requirements and attention to achieve appropriate outcome with respect to the domain of satellite imagery and signal processing. In this chapter the outcomes of the results are analysed with specific reason as well as the factors that can be improve also mentioned here (in section 7.1). The overall work and future aspect of this work is mention in the section 7.2 and 7.3.

## 7.1 Discussion

- SRGAN keeps performing better than other models. One of the major reasons is that SRGAN has larger filter kernels compared to EDSR (as portrayed in the figures 7.1 and 7.2), thus it can extract much more information prominently as compared to EDSR as illustrated in the feature maps of the respective models. We anticipate some improvements if the EDSR's filter kernel sizes are increased, but transfer learning won't be possible because any changes to the architecture will completely destroy the orientation and significance of weights and biases.
- Furthermore, SRGAN and SR(PRE) have the same CNN architecture. However, SRGAN uses the adversarial setting for training with images whereas SRPRE has a simple ResNet architecture. The weights and biases from SR(PRE) are taken as inputs by SRGAN and are treated in a better way in order to finetune the generator and provide better results.

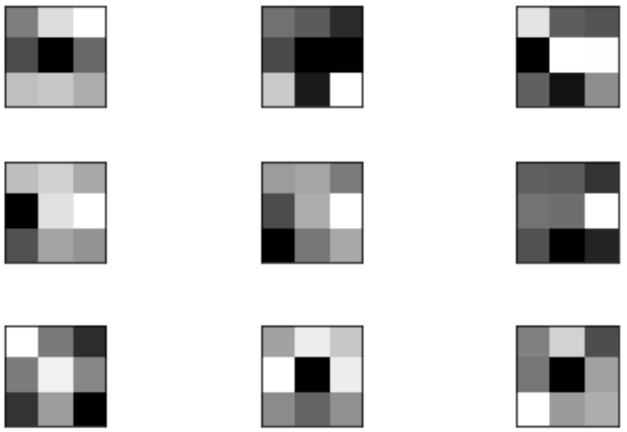

FIGURE 7.1: 3rd Layer EDSR Filter Kernels.

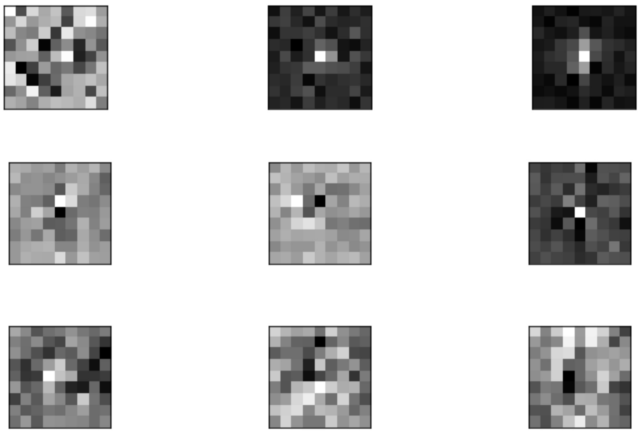

FIGURE 7.2: 2nd Layer SRGAN Filter Kernels.

- Most of the feature maps created by EDSR are dark, meaning that only a small amount of feature information can be gleaned from them.

  If somehow we can enhance the feature maps, we may think the results will improve. However, feature maps are the representation of the learnt weight and biases or the filter kernels, so we can't enhance the feature-maps without interfering with the filter kernels of the particular layer.

  In such a situation, we can think about dilated convolution, which will improve the information-capturing ability of the filter kernel. But, as we have discussed in the first point, it will change the architecture of EDSR itself, and we can't get the benefit of transfer learning.

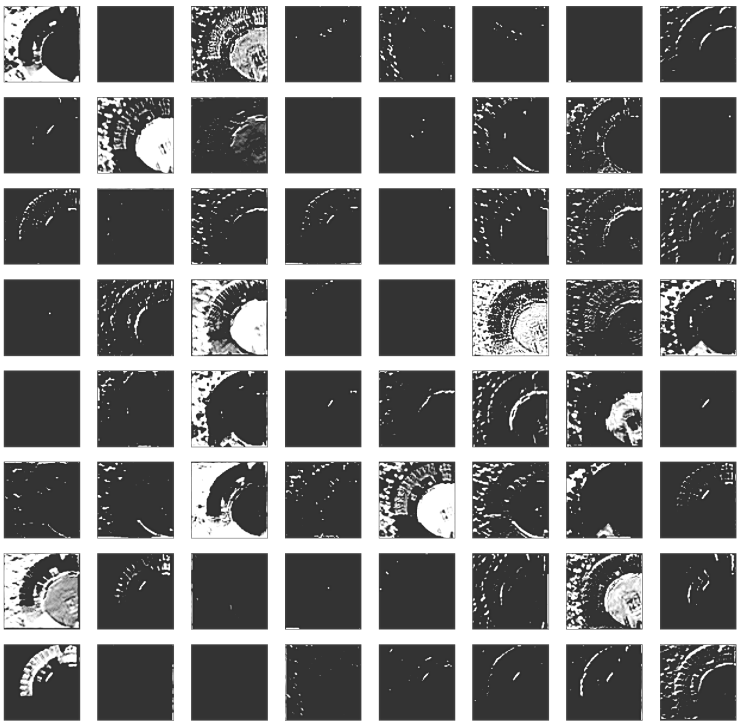

FIGURE 7.3: 3rd Layer Feature Map of EDSR.

Additionally, there is not much difference in the extracted features due to the similarity of the several resultant feature maps, as seen in figure 7.3. It can be the result of the model overfitting to the available data. In order to improve the model's efficiency and overall complexity, it may be addressed by adding Dropout layers to the EDSR model.

- Additionally, two bands in the LR image are uniformly darker, and the HR image sent to the network likewise has a band that is visibly dark. This may also be the cause of the feature map being really dark. Increased feature map quality can improve the quality of the SR image if certain image processing can be done without compromising the image's information.

- The feature maps of the SRGAN model extract various features. As seen in figure 7.4, the maps show different features with variations in contrast, sharpness, colour, brightness, noise, etc.

- One may also examine the artefacts that have been formed in the picture by zooming into the network-generated image as shown in Figure 7.5. It could be because the features that the three layers extracted have different properties, and when they

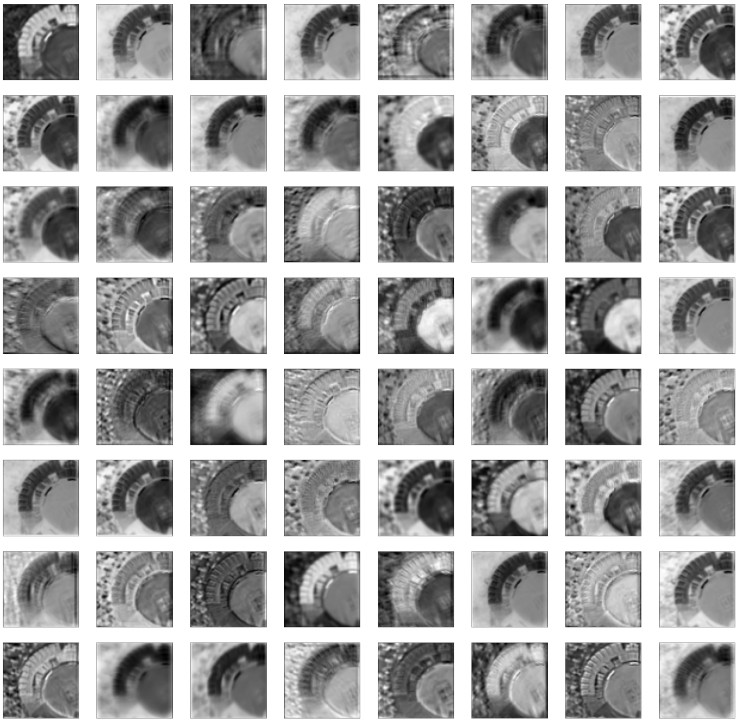

FIGURE 7.4: 2nd Layer Feature Map of SRGAN.

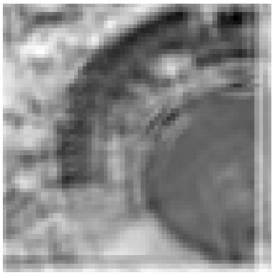

FIGURE 7.5: Network-generated image with artefacts

overlap, they produce RGB artefacts. One input layer instead of three can be used in the convolution layer to remedy this.

## 7.2 Conclusion

The current work discussed the use of Convolutional and Generative Adversarial networks for the purpose of generating SuperResolution of satellite images. Despite the fact that further research and development can be carried out in the future (given in the next section), we can conclude that the project and its outcomes based on the findings presented in chapter 6 are quite good but far away from being satisfactory. Our findings suggest

that GAN-based models can produce even better outcomes in the field of superresolution for satellite image processing in the future.

## 7.3 Future Works

Although current work is more or less satisfactory, further work can be conducted in the future to explore the prospect of producing better results.

1. EDSR shown during the report, even though PSNR and SSIM slightly increase over time, show clear signs of overfitting. The feature maps couldn't extract the relevant attributes and the model doesn't perform well with satellite images. In the future, a good experiment could be to introduce Dropout layers to that each and every feature map can learn to extract relevant features and prevent overfitting scenarios.

2. Due to the three bands' overlapping during the concatenation process in the post-processing stage and the various outputs they produced, all of the created SR images have artefacts. For better results, the model can be designed in such a way that the different channels are processed together while producing the final SR image. That could defy the possibility of overlapping and would refrain from producing artefacts.

3. Wide dynamic range may be found in many satellite images. To keep all the crucial details and qualities, efforts must be made to use the complete image in its full dynamic range. Multiple bands with the same or different resolutions may exist in certain satellite images; attempts may be made to analyze these types of images and use the high-resolution band to produce superior SR images.

4. On the basis of GAN, newly created VAE, or transformer architectures, several new models may be suggested that should be particularly trained on satellite imaging data and capable of extracting considerably more features within a huge dynamic range. Additionally, quicker training and testing times should be anticipated.

# Appendix A

The integrated development environment used for the coding of the model, pre and post transfer learning, is Kaggle with features as follows:

- CPU: Kaggle kernel
- GPU: 15.9 GB
- RAM: 13 GB
- Disk: 73.1 GB
- Site: https://www.kaggle.com

Google Colab has been used for the Geo-referencing purposes, with features as:

- CPU: Python3 Google Compute Engine backend
- GPU: 15.9 GB
- RAM: 12.68 GB
- Disk: 107.72 GB
- Site: http://colab.research.google.com

# Bibliography


[1] G. Keilbar, *Modelling Systemic Risk using Neural Network Quantile Regression*. PhD thesis, 08 2018.

[2] F. Bre, J. Gimenez, and V. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, 11 2017.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[4] M. U. Freudenberg, "Tree detection in remote sensing imagery baumerkennung in fernerkundungs- bildmaterial." https://bit.ly/3LpMUIx.

[5] M. BEN-YOSEF, *Multi-Modal Generative Adversarial Networks*. PhD thesis, Hebrew University of Jerusalem, 2018.

[6] J. Riebesell, "Autoencoder." https://tikz.net/autoencoder/.

[7] E. Dahlström, "Super-resolution using dynamic cameras," 2020.

[8] D. Chira, I. Haralampiev, O. Winther, A. Dittadi, and V. Liévin, "Image super-resolution with deep variational autoencoders," *arXiv preprint arXiv:2203.09445*, 2022.

[9] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144, 2017.

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

[11] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *IEEE signal processing magazine*, vol. 20, no. 3, pp. 21–36, 2003.

[12] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[13] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Advances and challenges in super-resolution," *International Journal of Imaging Systems and Technology*, vol. 14, no. 2, pp. 47–57, 2004.

[14] G. Tsagkatakis, A. Aidini, K. Fotiadou, M. Giannopoulos, A. Pentari, and P. Tsakalides, "Survey of deep-learning approaches for remote sensing observation enhancement," *Sensors*, vol. 19, no. 18, p. 3929, 2019.

[15] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[16] M. U. Müller, N. Ekhtiari, R. M. Almeida, and C. Rieke, "Super-resolution of multispectral satellite images using convolutional neural networks," *arXiv preprint arXiv:2002.00580*, 2020.

[17] M. A. Nuño-Maganda and M. O. Arias-Estrada, "Real-time fpga-based architecture for bicubic interpolation: an application for digital image scaling," in *2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig'05)*, pp. 8–pp, IEEE, 2005.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645, Springer, 2016.

[20] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proceedings of the IEEE international conference on computer vision*, pp. 4799–4807, 2017.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[22] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[23] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[24] D. Mahapatra, B. Bozorgtabar, and R. Garnavi, "Image super-resolution using progressive generative adversarial networks for medical image analysis," *Computerized Medical Imaging and Graphics*, vol. 71, pp. 30–39, 2019.

[25] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018.

[26] M. S. Sajjadi, B. Schölkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," *arXiv preprint arXiv:1612.07919*, 2016.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[28] Z.-S. Liu, W.-C. Siu, and Y.-L. Chan, "Photo-realistic image super-resolution via variational autoencoders," *IEEE Transactions on Circuits and Systems for video Technology*, vol. 31, no. 4, pp. 1351–1365, 2020.

[29] I. Gatopoulos, M. Stol, and J. M. Tomczak, "Super-resolution variational auto-encoders," *arXiv preprint arXiv:2006.05218*, 2020.

[30] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[33] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[34] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer.," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[35] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.

[36] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo, "Learning texture transformer network for image super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5791–5800, 2020.

[37] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological cybernetics*, vol. 20, no. 3, pp. 121–136, 1975.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[39] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, Citeseer, 2013.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, pp. 694–711, Springer, 2016.

[42] J. Bruna, P. Sprechmann, and Y. LeCun, "Super-resolution with deep convolutional sufficient statistics," *arXiv preprint arXiv:1511.05666*, 2015.

[43] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," *Advances in neural information processing systems*, vol. 28, 2015.

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

[45] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[46] M. Menéndez, J. Pardo, L. Pardo, and M. Pardo, "The jensen-shannon divergence," *Journal of the Franklin Institute*, vol. 334, no. 2, pp. 307–318, 1997.

[47] M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning," *arXiv preprint arXiv:1812.05069*, 2018.

[48] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.," *Journal of machine learning research*, vol. 11, no. 12, 2010.

[49] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot, "Higher order contractive auto-encoder," in *Joint European conference on machine learning and knowledge discovery in databases*, pp. 645–660, Springer, 2011.

[50] A. Makhzani and B. Frey, "K-sparse autoencoders," *arXiv preprint arXiv:1312.5663*, 2013.

[51] A. Gavade and P. Sane, "Super resolution image reconstruction by using bicubic interpolation," 10 2013.

[52] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[53] W. Wu, "patchify 0.2.3." https://pypi.org/project/patchify/.

[54] O. S. G. Foundation, "Gdal." https://gdal.org/.

[55] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*, pp. 2366–2369, IEEE, 2010.

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[57] K. Nelson, A. Bhatti, and S. Nahavandi, "Performance evaluation of multi-frame super-resolution algorithms," 12 2012.

[58] X. Liu, P. He, W. Chen, and J. Gao, "Improving multi-task deep neural networks via knowledge distillation for natural language understanding," *arXiv preprint arXiv:1904.09482*, 2019.

[59] NASA, "What is remote sensing?." https://www.earthdata.nasa.gov/learn/backgrounders/remote-sensing.