A GAN-Based Model for Single Image Super-Resolution on Consumer-Grade GPUs: Comprehensive Analysis and Development of MSRGAN

by

Istiaq Ahmad
20301056
Labib Sadman Azam
21301643
Moinul Hossain Bhuiyan
20301002
Abdullah Al Mamun
20301062

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2024

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]	[illegible]
Istiaq Ahmad	Labib Sadman Azam
20301056	21301643
[illegible]	[illegible]
Moinul Hossain Bhuiyan	Abdullah Al Mamun
20301002	20301062

Approval

The thesis titled “A GAN-Based Model for Single Image Super-Resolution on Consumer-Grade GPUs: Comprehensive Analysis and Development of MSRGAN” submitted by

1. Istiaq Ahmad (20301056)
2. Labib Sadman Azam (21301643)
3. Moinul Hossain Bhuiyan (20301002)
4. Abdullah Al Mamun (20301062)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 17th, 2024.

Examining Committee:

Supervisor:
(Member)

[illegible]

Dr. Muhammad Iqbal Hossain

Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

In a world where visual content plays a crucial role in anything imaginable, the need for sharper, more detailed images has never been more important. This research paper explores innovative approaches to improve the quality of single images through the application of Deep Learning techniques, specifically GAN architectures. The research focuses on developing an efficient and feasible model that can be trained in a consumer-grade GPU. Among various architectures, the research focused on the RRDB model for further development. With the modification of the RRDB model and the implementation of activation functions combination and proper loss functions, this research seeks to achieve enhanced performance and effectiveness. Finally, the proposed model MSRGAN was developed which was capable of training on a consumer-grade GPU with an average amount of video RAM. The model possesses the capability for 4x upscaling. For testing the performance, the research used PSNR and SSIM evaluation metrics where the MSRGAN has outperformed the basic SRGAN and ESRGAN.

Keywords: Single Image Super Resolution (SISR), Super Resolution Generative Adversarial Networks (SRGAN), Residual-in Residual Dense Enhanced Super Resolution (RRDB), Generative Adversarial Networks (ESRGAN), Modified Super Resolution Generative Adversarial Networks (MSRGAN).

Table of Contents

List of Figures

Nomenclature

The symbols & abbreviation used within this thesis paper is described in the list.

BSRNET	Blind Super-Resolution Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EGAN	Enhanced Generative Adversarial Network
ESRGAN	Enhanced Super-Resolution Generative Adversarial Network
GAN	Generative Adversarial Network
GT	Ground Truth
HR	High Resolution
LR	Low Resolution
LReLU	Leaky Rectified Linear Unit
MSRGAN	Modified Super-Resolution Generative Adversarial Network
NAD	Noise Aware Discriminator
PReLU	Parametric Rectified Linear Unit
PSNR	Peak Signal-to-Noise Ratio
RDB	Residual Dense Block
ReLU	Rectified Linear Unit
RRDB	Residual-in Residual Dense Block
SISR	Single Image Super Resolution
SR	Super Resolution
SRGAN	Super-Resolution Generative Adversarial Network
SSIM	Structural Similarity Index Measure
SSL	Structured Sparsity Learning

Chapter 1

Introduction

1.1 Background

In a world filled with visual content, the desire for crisper, more colorful high-resolution pictures has become a fundamental component of an individual's digital experience. The target of SISR is to enhance the LR images into HR images. Deep learning methods, especially through GAN architecture, have become more and more effective than traditional methods like bicubic interpolation which struggles to preserve the details. GAN-based SISR models have two parts, the generator creates better-quality images, and the discriminator verifies the difference between the generated and the original image. It helps the generator to improve over time. The SRGAN is a popular model. It focuses on making images look more realistic and natural, leaving the older methods to the dust. But the caveat is that these models can be very demanding on computer resources, needing industry-grade GPUs to work well. It makes it difficult for people who are using regular consumer-grade hardware.

1.2 Problem Statement

SISR models aim to reconstruct HR images from LR inputs. However, the computational complexity of state-of-the-art models poses significant challenges in real-world application deployment especially on consumer-grade hardware. Making it a challenge for the users who want to pursue this field of image super-resolution, without the industry-level equipment. Image upscaling enhances the quality and detail of LR images. Making them suitable for those tasks where HR images need to be the benchmark point. It also allows old, low-resolution images to be revitalized for modern use without losing clarity. All of the GAN-based models that are used are resource-intensive. To run these models the researchers need access to industry-level GPUs, thus a normal user can not develop these GAN-based models on their consumer-grade GPUs. The image-upscaling tools which are available on the internet, can not be modified according to one's need. This is an issue for most people aspiring to work in the field of image super-resolution.

1.3 Research Objectives

In this research paper, the purpose is to build an efficient GAN based model which can 4x upscale a low-resolution image on a customer grade GPU. This paper's main goal is to investigate and advance the application of deep learning techniques and architecture of the model, in the field of single image super-resolution. The following are the study goals for this paper:

- Investigate the effectiveness of different GAN models, where the resolution and quality of single images are enhanced.
- Evaluate the capabilities of selected models to improve the super-resolution.
- Examine the generator part, discriminator part, activation function and loss calculation methods used in those models.
- Explore various combinations and possible changes in the architecture for optimal solutions in different scenarios.
- Developing versatile and single category based datasets which are appropriate for the GAN models.
- Constructing a GAN model which has unique specifications and can perform better than existing models.
- Compare the models with multiple evaluation metrics

By addressing these goals, this paper will contribute to the evolution of technology that can improve the visual quality of photos, making them sharper, more detailed, and acceptable for a broad variety of real-world applications. Also, it will inspire the people who are interested in ML with limited resources to go through the experience of training his/her own dataset with this model.

1.4 Thesis Outline

- In chapter 2, this paper studies previous works related to deep learning based SISR methods that are being implemented over the years. Besides there is information about the traditional GAN architectures, convolutional neural networks, activation functions, loss functions, dataset fine tuning and measure metrics.
- In chapter 3, there is a detailed description of dataset extraction. It further explores how datasets have been split and how the datasets were preprocessed. This section also studies how datasets are being scaled down and how the images from datasets are being denoised.
- In chapter 4, the experimental framework and how the modified version of SR-GAN was implemented is discussed. Firstly there is a brief description about the types of configurations that were used to develop and test the model. Then which type of IDEs and what type libraries used in this MSRGAN context was

explained. Then this chapter further discusses the methodology of developing the MSRGAN, SRGAN and ESRGAN.

- In chapter 5, this thesis focuses on the result analysis of the newly implemented model called MSRGAN. This chapter consists of the loss curve inspections, the evaluation of the metric results and lastly the generated image visual comparisons.
- Chapter 6, includes the limitations of the existing SISR based GAN models. Then the chapter extends by including the future works with this newly developed MSRGAN model. Lastly this chapter concludes the whole thesis with general remarks.

Chapter 2

Background

2.1 Literature Review

In the paper titled “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network” a generative adversarial network (GAN) for image super-resolution (SR) called SRGAN is introduced. It tackles the problem of recovering finer texture information at large upscaling factors, which is still a big problem even with convolutional neural network developments[1]. The following model uses the PReLUn as its activation function.

This specific SRGAN includes a perceptual loss function which consists of two loss functions, an adversarial loss and a content loss[1]. This model is different from previous approaches that primarily concentrated on reducing mean squared reconstruction error.

Then “A Super Resolution Algorithm Based on Attention Mechanism and SRGAN Network” focuses on the GAN model and the SRGAN model[2]. The primary objective of the paper is to attain high-resolution images by inputting low-resolution images. It introduces three main changes. First of all, the author added a Ca (channel attentiveness) module to SRGAN to improve the expression of high-frequency features. To properly collect minute details, this makes the model even deeper. The authors of the paper removed the initial BN layer to improve network efficiency. Making the changes helps to maintain small details in the reconstructed image and probably lowers the chance of over-smoothing. For the activation function, the paper proposed the Leaky ReLU model. This model helps to attain better results and in the case of the loss function the paper proposed the combination of the L1 model and MSE model[2]. These changes helped to attain some good results where their PSNR and SSIM scores were better than others.

The paper is titled as “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks”. For single-image super-resolution, Enhanced SRGAN (ESRGAN) is superior to SRGAN[3]. The problem of unpleasant artifacts accompanying hallucinated details is addressed with ESRGAN. Three essential SRGAN components are examined in detail and improved. For network architecture, the use of the RRDB model makes most sense[3]. In this architecture model batch normalization is not being used and Leaky ReLU is used as its building block. This works as the activation function. For the loss function, two types of loss functions are used very commonly which are adversarial loss and perceptual loss. By deploying these things in a SR model, the chances of getting a better result and better performance increases drastically.

As different experiments are done with neural networking, deep neural networking "Activation functions" hold a quite significant role in the context of image SR models. There are various kinds of activation functions. The paper titled "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning" demonstrated the importance of the activation functions. It introduces non-linearity into neural networks and influences their performance. Due to these various performances, the experiment brings different outcomes. Although the functions that were used earlier such as tanh and sigmoid, had some barriers[4]. The introduction of the ReLU function has brought more optimization and convergence. But even so, ReLU has its drawbacks, which is the Dying ReLU issue and the domain of non-differentiability. Hence, many activation functions have been developed, including leaky ReLU, PReLU, ELU, GELU, SELU, Swish, Mish, and others. Amongst them, Leaky ReLU, SELU, Softplus, and LiSHT had greater effects[5].

There are many kinds and types of loss functions available to satisfy different scenarios. The loss function is commonly known as the cost function. It measures the difference between the generated output of the machine learning or fake HR outputs and the ground truth or real HR inputs. The loss functions determine how close the generated output is to the ground truth. The most commonly used loss functions are the Mean Absolute Errors (MAE), also known as the L1 error[6]. Then the Mean Squared Error (MSE), also known as the L2 error[6]. expectation loss, regularised expectation loss, Chebyshev loss, and log (cross-entropy) loss, etc. Common loss functions used in classification problems are hinge loss for support vector machines, softmax loss for multiclass classification using softmax activation, and cross-entropy loss for binary or multiclass classification. Mean absolute error (MAE) loss and mean squared error (MSE) loss are frequently employed in regression tasks. All things considered, loss functions are essential for directing machine learning models' learning process and helping them become more proficient at the task at hand.

The strategy to improve the quality and speed of single-image super-resolution using deep neural networks is always a challenging task. Current techniques upscale LR images before improving them, which is ineffective and resource hungry. The authors of the paper titled as "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network" provide a revolutionary CNN architecture that works directly with low-resolution data and employs a smart upscaling method[7]. In this paper, the authors focus on the task of SISR and they used ReLU as their activation function. The whole purpose of SISR is to estimate an HR image $I^{[SR]}$ given an LR image $I^{[LR]}$ downscaled from the corresponding original HR image IHR. The downsampling operation is deterministic and known to produce $I^{[LR]}$ from $I^{[HR]}$. Basically at first it convolves $I^{[HR]}$ using a Gaussian filter.The main idea of this research paper is to highlight the importance of SR in digital image processing, where the goal is to enhance the performance and resolution of LR photos. It highlights that SR has practical applications in numerous areas including HDTV, medical imaging, satellite imaging, face recognition, and surveillance. The research paper analyzes the problems of SR, including the loss of high-frequency information during the LR-to-HR transition and the inherent uncertainty in the representation from LR to HR space. It additionally provides two main kinds of SR methods: multi-image SR, which depends on numerous LR photographs of the same scene, and SISR, which intends to retrieve HR information from a single LR instance exploiting implicit redundancy discovered in natural data. Both methods have to deal with the ill-posed nature of the issue and need restrictions or previous information to guide the reconstruction process.

Moving on to the paper titled as "Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications" provides the importance of GANs as a potent framework for image synthesis, both unconditionally and with input conditions[8]. GANs have changed the development of high-resolution and photorealistic visuals, a task previously believed tough or unachievable. The emergence of GAN has prompted various unique applications in deep learning approaches. This paper presents an overview of GANs, highlighting their algorithms and applications in visual synthesis. It dives into critical strategies geared at stabilizing the famously hard GAN training process. Additionally, it covers GAN applications in image translation, image processing, and neural rendering. This paper addresses the fundamental concept of Generative Adversarial Networks (GANs) in the area of deep learning and its major impact on several aspects of visual content synthesis. GANs comprise a generator and discriminator network engaged in a competitive training process, resulting in the development of synthetic data that resembles actual data. GANs have successfully replaced hand-designed artworks in computer vision pipelines, especially for generation tasks. It derives objective functions from training data. However, GANs are tough to train because of the changing nature of the generator's output distribution, which demands careful control of training dynamics. Various ways have been proposed to stabilize GAN training over time. It is also important to differentiate between unconditional and conditional GAN frameworks where conditional GAN utilizes control signals for more precise generation tasks[8]. This major development has led us to various exciting applications in semantic picture creation, image-to-image translation, image processing, neural rendering etc. The overall topic is that GANs have become an essential tool in the area of computer vision, allowing numerous creative visual content-generating applications. The paper uses perceptual loss as the loss function. GANs comprise a generator and discriminator network engaged in a competitive training process, resulting in the development of synthetic data that resembles actual data. GANs have successfully replaced hand-designed components in computer vision pipelines, especially for generation tasks, by deriving objective functions from training data.

A new single-image upscaling method that improves picture quality and efficiency was studied in the work "Image and Video Upscaling from Local Self-examples". This approach emphasizes local self-similarity in the picture as opposed to other approaches that rely on external datasets. It reduces search time while maintaining the quality by extracting patches from small, localized portions of the input picture. This method is particularly effective for lesser scaling factors. It applies specialized filters for these small scalings, providing high-resolution outcomes compatible with the original picture. The algorithm is basic, efficient, and can be implemented in parallel on a GPU. It shows high-quality resolution enhancements, works perfectly with several image sequences, and is capable of real-time enhancement of LR images into HR images. It is important to address the difficulties of image upscaling as it is a fundamental image-enhancing procedure. It highlights the limits of traditional upscaling approaches, which frequently result in artifacts and image abnormalities. It presents an innovative method that uses local scale invariance in real images, concentrating on small, localized patches in order to improve the accuracy and efficiency of upscaling[9]. This creative method works better for small scaling factors and includes multiple upscaling steps employing specific filter banks to accomplish high-quality resolution enhancement. Additionally, it demonstrates the advantages of the proposed technique for image sequences, along with its efficient implementation on GPUs for real-time performance.

Besides, the research offers an enhanced version of SRGAN (Super-Resolution Generative Adversarial Network) by including Residual-in-Residual Dense (RRDB) blocks and eliminating batch normalization layers[9]. These modifications seek to enhance training efficiency and shorten the time necessary for super-resolution. These modifications seek to enhance training efficiency and shorten the time necessary for superresolution. The authors also underline the need to keep internal textures and improve visual quality in the super-resolved images. Additionally, the study describes adjustments made to the discriminator architecture, proposing the idea of a Realistic Average GAN to improve the comparison of visual and realistic quality in images. The suggested method is claimed to provide better textures and improve the frame generation rate compared to SRGAN. Furthermore, ESRGAN enhances realism by prioritizing the preservation of critical visual characteristics using VGGNet-19 for perceptual loss[9]. It generates super-resolution images that mimic real high-resolution content and appear sharper. ESRGAN considers various upscaling factors, increasing its versatility in handling different scaling requirements. It outperforms SRGAN by implementing a sophisticated RRDB architecture, improving perceptual loss functions, and showcasing adaptability while managing various upscaling factors. Moreover, It is important to consider the method's performance using multiple quality evaluation measures such as PSNR, and SSIM while noting that training GANs consumes significant time and suggests powerful hardware for quicker frame generation. SISR models can benefit significantly via modifications in neural networks architecture to generate high-quality and super-resolved images. It also analyzes the difference between SISR emphasizing the necessity to utilize temporal correlations between low-resolution images. Traditional SISR methods may often require computational costs that are higher and struggle with large scaling factors. ESRGAN considers various upscaling factors, increasing its versatility in handling different scaling requirements. It outperforms SRGAN by implementing a sophisticated RRDB architecture, improving perceptual loss functions, and showcasing adaptability while managing various up-scaling factors[10].

As this research paper stated before, the GAN is a framework that is set against the generator and discriminator against each other. In any kind of SR model, the generator is a neural network that creates HR images from LR inputs. Its objective is to produce images that are realistic and almost identical to actual HR images. This HR output is designed to fool the discriminator which often tries to differentiate between real and generated images. In the current GAN framework, the generator learns continuously through adversarial training. This continuous training then improves the quality of the generator to produce high-quality images. Then comes the concept of the discriminator. The discriminator is a neural network that checks and evaluates whether an image is real from the actual dataset or fake as generated by the Generator. It works just like a judge in a reality show. It pushes the generator to its limits to improve the overall image quality generated by the generator. Now there might be a situation where the generated image has too much noise. Then Noise Aware Discriminator or NAD is being deployed to handle noise from different stages of the image generator. This also helps the discriminator to be more robust across a variety of noise levels. The generator and the discriminator are always trained together in an adversarial manner.

The Introduction of a new model named the "Distillation Free One Step Diffusion model" is very crucial for image SISR methods. Its generator performs a single denoising step to recover the unused high-resolution image from low-resolution inputs. In the context of GAN architecture, this also introduces the very same NAD or Noise Aware Discriminator described before. It basically Enhances the authenticity of the generated images by leveraging information from a pre-trained Stable Diffusion UNet mode. NAD tries to process Noisy Intermediate unused features and helps improve the model's performance across various noise levels[11]. In this specific paper, in the generator part, the Distillation One Step Diffusion model uses the diffusion principle to progressively denoise and reconstruct HR images from LR inputs in one step. It also bypasses the need for multi-step diffusion sampling. Then the NAD discriminator helps maintain the realism and authenticity of the generated images. It ensures whether the picture that has been generated by the generator is satisfiable or not.

After analyzing the paper titled "Signal Processing: Image Communication", one can give valid reasoning for using LReLU as a Generator and Discriminator function. LReLU is often used in terms of image super-resolution. According to the paper by Masoumeh :ReLU was employed with a slope of 0.02 to handle issues like dying ReLU units[12]. Dying ReLU can occur during training datasets. It usually occurs in deeper layers. Leaky Relu allows a small and non-zero gradient which we will discuss in this paper further. The use of ReLU activations in the generators except for the output layer supports the learning of vast sets of HR images. Also, ReLU contributes in retaining positive activations while keeping the processes efficient. In image SR models, Leaky ReLU is employed in deeper layers to prevent the vanishing gradient problem. After tackling the vanishing gradient problems the LReLU ensures that the negative inputs will not lead to inactive neurons. These inactive neurons could hinder the generation of producing realistic HR images. LReLU could also be employed as a discriminator. As stated before, to implement SR models, the authors employed LReLU with a slope of 0.2. This is being applied in the discriminator network to enable it to effectively discriminate between real HR images and generated HR images. Also, it prevents the neurons from dying by allowing a small gradient when the input is negative. This maintains the learning, effectiveness, and enhancement of gradient flow. This is important to ensure the stability of adversarial training[12]. LeLU also helps the discriminator to refine its ability to inspect the quality of images generated by the generator network. The fact is that the use of Leaky ReLU is consistent with its widespread adoption in Various GAN architectures because LRelu helps improve the gradient flow and mitigates vanishing gradient issues. And Leaky ReLU also tackles the Dying ReLU problem. These all beneficiary factors help the generator to generate higher-quality images during the super-resolution process.

2.2 Neural Network (NN)

Neural networks are computational models inspired by biological neurons in the human brain, consisting of interconnected nodes organized into layers: input, hidden, and output. These layers process input data, learn complex patterns through weighted connections, and produce the final output.

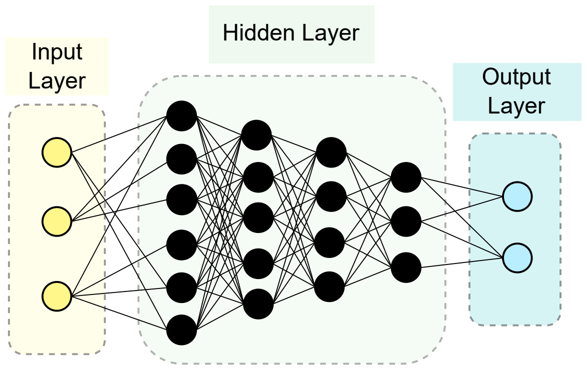

Figure 2.1: Working Mechanism of Neural Network.

Neural networks are computational models inspired by biological neurons in the human brain, consisting of interconnected nodes organized into layers: input, hidden, and output (2.1). These layers process input data, learn complex patterns through weighted connections, and produce the final output.

Deep learning is a subset of machine learning and it uses neural networks with multiple layers to model and solve complex problems. Deep learning automatically extracts useful information from photos and performs end-to-end learning, allowing networks to learn from raw data and tasks. It scales with data, unlike shallow learning, which converges when additional instances and training data are added. Deep learning's "deep" layers allow the network to learn hierarchical features and their representation, making it successful in natural language processing, reinforcement learning, and computer vision [13]. However, it requires substantial labeled data and computational resources, and the interpretability of complex models can be a challenge. Despite these limitations, neural networks remain the basic building blocks for deep learning, enabling more powerful and automated learning from trained data.

2.3 Generative Adversarial Network (GAN)

The discipline of generative modeling has seen a change because of the revolutionary class of machine learning models known as GAN. A new approach for producing fresh data instances that closely mimic an existing dataset is provided by GANs. The interaction between the generator and discriminator neural networks, which are involved in a competitive learning process, is the fundamental concept of GANs.

In this architecture, there are two sectors. One is the “Generator” and the other one is the “Discriminator”. In the GAN architecture, the ”Generator” acts as a creative element. The job of the “Generator” is to take in random noise and convert it into data instances that, as closely as possible, resemble the patterns found in the training data[14]. On the other hand, the ”Discriminator” undertakes the position of a critic. Its job is to differentiate between instances of actual data from the training set and artificial data generated by the generator (2.2).

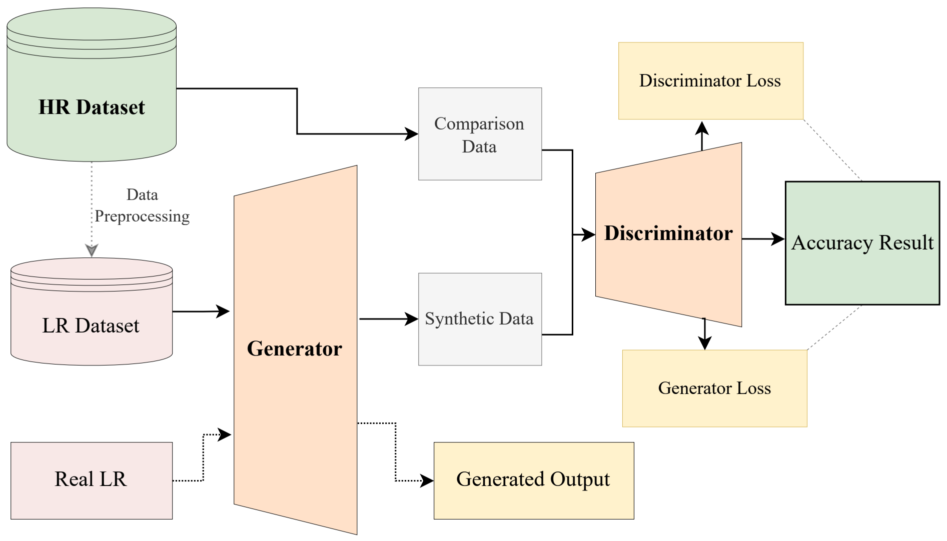

Figure 2.2: Working Mechanism of GAN.

The generator and discriminator engage continuously back-and-forth throughout the GAN training process. The ”Discriminator” wants to improve their ability to discriminate between actual and fake cases, while the ”Generator” wants to create data that is identical to real examples. Due to the competitive dynamic created by this adversarial training, both networks are constantly improving and challenging one another to achieve greater performance levels[15]. The generator aims to maximize the likelihood that its generated samples would be mistakenly classified as real by the discriminator. On the other hand, the discriminator aims to maximize the accuracy of its distinction between authentic and fraudulent data. This antagonistic goal creates a delicate balance, and careful parametric adjustment is frequently necessary for GAN training to be effective.

$$min_G max_D V(D,G) = E_{x \sim Pdata(x)}[logD(x)] + E_{z \sim Pz(z)}[log(1 - D(G(z)))]$$

G is a neural network that generates synthetic data, attempting to mimic the real data distribution. D is another neural network that acts as a binary classifier. It takes input data and tries to distinguish between real data (x) and fake/generated data ($G(z)$). P(Sub)Data(x) represents the distribution set of real data. The generator tries to generate data that is similar to this distribution. $Pz(z)$ represents the distribution of the noise that is fed into the generator. It's a source of randomness that allows the generator to produce diverse outputs. The GAN's objective is formulated as a minimax game between the generator and the discriminator. The goal is to minimize the generator's loss while simultaneously maximizing the discriminator's loss.

The expected value of the log probability that the discriminator assigns to the produced data is represented by the second term, $E_{x \sim Pdata(x)}[logD(x)]$. In order to increase the likelihood that created data will be seen as authentic by the discriminator, the generator aims to reduce this.

The second term $E_{z \sim Pz(z)}[log(1 - D(G(z)))]$ represents the expected value of the log probability that the discriminator assigns to the generated data. The generator wants to minimize this, making generated data more likely to be classified as real by the discriminator.

2.4 Convolutional Neural Network (CNN)

The artificial intelligence technique known as a convolutional neural network (CNN) is made specifically for tasks involving images and visual input. It can be viewed as an intelligent system with the ability to see and comprehend images.

For making a computer to identify an object in a picture this CNN architecture is important. A CNN consists of layers, and each layer has a distinct function (2.3). It is possible for the first layer to identify basic objects like edges and colors. Deeper layers integrate these basic characteristics to comprehend increasingly intricate pat- terns, such as textures and forms. Each layer gives a unique value. These values of each segment add up and in the end, tell a computer what it is. The concept of applying filters or tiny grids that move over the image is where the word "convolutional" originates[16]. These filters assist the network in learning relevant data and concentrating on particular details. CNNs use a technique called pooling, which minimizes the amount of data they must evaluate while maintaining crucial features, to make this func- tion effective[17]. The multidimensional output of the following layer in Convolutional Neural Networks (CNNs) is transformed into a one-dimensional array using a flat– ten layer. Then using the fully connected layer the model gives the probability of an object.CNN can identify brand-new, unidentified photographs after it has been trained[18][19].

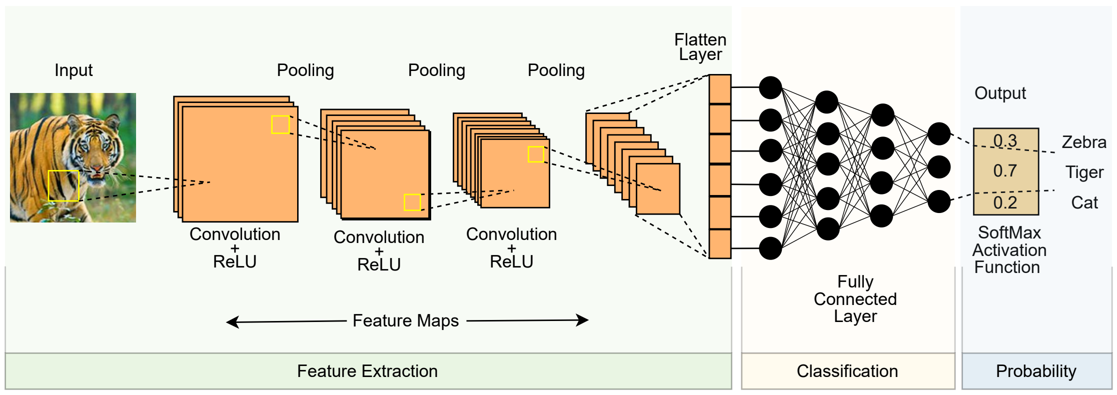

Figure 2.3: The working mechanism of CNN

To sum up, CNNs are similar to visual investigators. They deconstruct images into their more basic components, develop their ability to spot patterns and apply these skills to comprehend and categorize new images.

ResNet

A kind of convolutional neural network (CNN) used for tasks involving images is called a residual network, or ResNet. The usage of residual blocks, which are intended to aid in the training of extremely deep neural networks, is one of its distinguishing features. A quick connection is used in a residual block to merge the input and output of a layer. The network can concentrate on acquiring residual information, that is, the variation between a layer's input and output thanks to this connection. The goal is to facilitate the network's ability to recognize and maximize an image's essential characteristics. These shortcut connections solve a common problem in deep networks that is referred to as vanishing gradients. Gradients in deep architectures may decrease during backpropagation as the network gets more complicated, which makes learning more difficult. By giving gradient flow a direct path via the residual connections, training very deep networks is possible without losing crucial data. In ResNet, every residual block picks up unique picture features[20]. The network can recognize and comprehend progressively complex patterns in the data by stacking these blocks. As the network gets deeper, performance deterioration is avoided because of this architecture, which encourages information to go through the network smoothly[21]. In image identification tasks, ResNet has proven remarkably successful, attaining state-of-the-art performance on multiple benchmarks.

2.5 Activation Functions

Every neural network model has criteria to understand whether the neural networks should be activated or not. The function that determines the activation of a network is called an activation function. By doing this, it introduces non-linearity to the model, enabling the network to learn complex patterns[22]. There are many activation functions in the realm of neural networks. Some of them are described below.

ReLU

ReLU stands for rectified Linear Unit. It is generally used as an activation function in neural networks. This activation function transforms the weighted input from a node of the neural network into the node's output. It is a piecewise linear function that outputs the input directly if it's positive and if not, it gives output zero. ReLU introduces non-linearity in existing neural network activation functions. ReLU overcomes the known Vanishing Gradient Problem which exists on other activation functions such as Hyperbolic and Sigmoid tangent functions[23]. ReLU also enables faster learning and better performance.

ReLU increases non-linearity in images which is crucial for this research as images inherently contain non-linear features (e.g. pixel transitions, borders, colors). The ReLU activation function is defined as:

$$f(x) = max(0, x)$$

This function takes an input value denoting x and outputs the maximum of 0 and x. When the input value is positive the activation function outputs x otherwise it gives 0 as an output. Which can be seen as well in the figure (2.4).

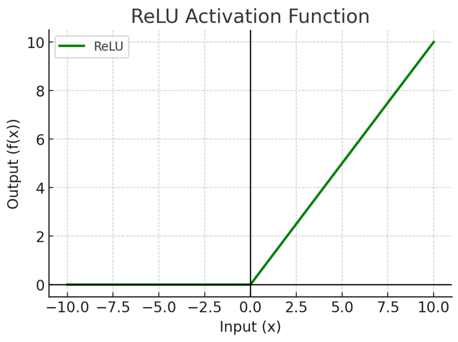

Figure 2.4: Graph of a ReLU activation function.

Leaky ReLU

In the standard ReLU activation function, the training of large data suffers from a problem that is addressed as "Dying ReLU". In this particular problem, neurons can get stuck with a “0” Output. To counter this “Dying ReLU” problem, a new activation function is being used which is called Leaky ReLU. Leaky ReLU introduces a small slope for negative inputs[24].

The LReLU is different from the standard ReLU activation function. Theoretically, it is more advanced than the standard ReLU activation function[25]. The standard equation for LReLU is:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$$

Where x stands for the input. The main change in the function is the value of . is a small positive constant which is usually around 0.01. In the context of LReLU, when x is negative, unlike standard ReLU, Leaky allows a small gradient to flow which can be seen in the figure (2.5). It helps prevent neurons from getting stuck.

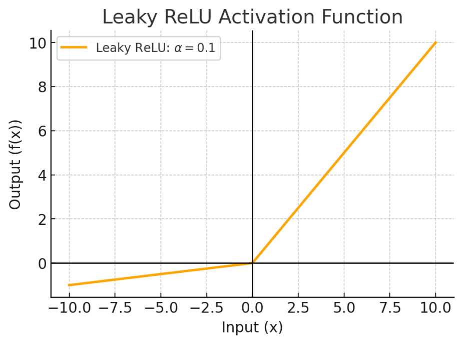

Figure 2.5: Graph of a LReLU activation function.

PReLU

Now there is an advanced version of Leaky ReLU currently available to be explored. It is called PReLU.It is a more advanced variation of LReLU. LReLU uses a fixed predefined slope = $0.01x$ for negative values whereas the slope in PReLU is a learnable parameter[26]. The equation for PReLU:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$$

Here (2.6) the slope is adapted during training. Allowing each neuron to learn the most appropriate slope. This adaptability enhances the model’s capacity to capture complex patterns.

Basically, PReLU is a more powerful and advanced activation function than both of the standard ReLU and Leaky ReLU[26]. PReLU adapts data and helps improve image upscaling tasks by enhancing feature representations and reducing dead neurons.

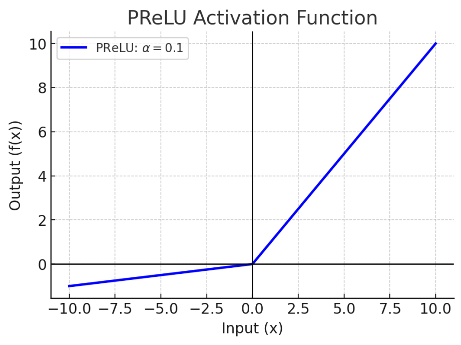

Figure 2.6: Graph of a PReLU activation function.

Sigmoid

A sigmoid function is a mathematical function with an S-shaped curve. It is commonly used in various fields of artificial neural networks. One well-known function that denotes the sigmoid function is called "The Logistic Function". It maps real numbers to a range between 0 and 1. It's commonly used in binary classification tasks where we need to predict probabilities. However, it tends to saturate (approach 0 or 1) for large or small inputs leading to a vanishing point gradient during training[27] (2.7). Previously we focused on the ReLU, Leaky ReLU, and PReLU functions. Those functions are for hidden layers due to their efficiency and gradient propagation whereas sigmoid remains relevant for output layers in binary classification.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

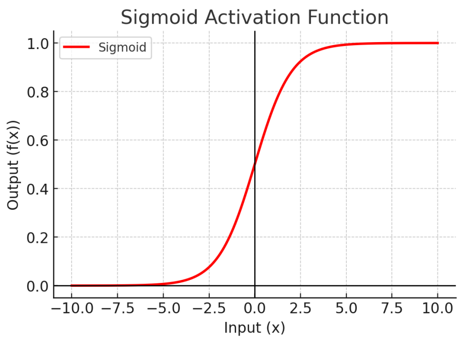

Figure 2.7: Graph of a Sigmoid activation function.

In the Neural Network, sigmoid functions serve as activation functions for artificial neurons. Allowing them to model complex architectures by doing binary classification, while ReLU excels in efficiency and scalability for deep learning tasks.

2.6 Loss Functions

A loss function in machine learning quantifies the difference between the predicted output of a model and the actual target value. The purpose of the loss function is to serve the model as a guide for the optimization process, helping the mode to adjust its weights during training[28]. There are many loss functions available. Among those most common loss functions include Mean Absolute Error (MAE), Mean Squared Error (MSE), Perceptual loss, and so on. The lesser the loss value indicates better model performance. The choice of loss function depends on the specific problem and the model architecture.

MAE Loss Functions

In the field of image super-resolution, the MAE loss function is most commonly used, which can be seen in some GAN architecture. This function is used to measure the difference between the produced high-quality image and the ground truth. The MAE loss function is also known as the L1 loss function. It helps to compute and average the absolute gap/differences between the pixels generated by the model and the pixels in the ground truth pictures. The mathematical equation is :

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.2}$$

MSE Loss Functions

In SRGAN the Mean Squared Error (MSE) is most commonly used. It is used in the generator's loss function comparing the truth image and the high resolutions output pixel by pixel. Because of the process where it is squaring, it discourages significant errors more severely, which incentivizes the generator to produce images with precise pixel values[28]. This method is comparatively straightforward and easy to use, which is why it is widely accepted. The mathematical term would be:

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

The MAE and MSE, both loss functions are quite similar. It depends on the user which one to use. If the user wants a loss function that handles all the errors equally, is easy to interpret and use, also which would be robust to outliers then the MAE loss function would be useful.

But if the user desires a loss function that would be differentiable everywhere, converges more quickly during training, and stresses big errors, then MSE would be the way to go.

Perceptual Loss

Perceptual loss is a type of loss function that is used to measure the gap between the ground truth and the image that is generated. It compares the high-level features taken from models trained by CNNs.

These loss functions intend to capture variations in perception that may not be visible at the pixel level. The produced picture is urged to match the style of a reference image and retain the content of the target image by minimizing perceptual loss[3].

VGG19

Introduced in the Perceptual Losses for Super-Resolution. VGG Loss is a sort of content loss. VGG Loss is an alternative to pixel-wise losses. The goal is to approximate perceptual similarity more closely. The pre-trained 19-layer VGG network's ReLU activation layers work as the foundation for the VGG loss. With i,j we indicate the feature map obtained by the j-th convolution (after activation) before the i-th maxpooling layer within the VGG19 network, which is considered given[1]. The equation would be:

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

Binary Cross-Entropy (BCE)

The Binary Cross-Entropy (BCE) is a type of loss function that is commonly used for binary classification tasks. Its job is to measure the difference between two probability distributions: the predicted probability (output of the model) and the actual binary label (0 or 1)[29]. The formula of the BCE loss is:

$$\text{BCE}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

2.7 Fine Tuning

The dataset we have selected needs to be pre-processed. By pre-processing the whole dataset we will be able to feed the research model with the appropriate information and try to bring out the most optimal output.

Optimizer

Adam Optimizer stands for Adaptive Moment Estimation. An optimizer is an algorithm used to adjust the weights of a neural network to minimize the loss function. It is relatively better than other Optimizers like SGD, AdaGrad, and RMSProp[30]. Amongst them all, Adam is often preferred for SISR-related tasks due to its fast convergence and robust performance with minimal tuning. Adam optimizers handle noisy gradients well [31]. Unlike SGD which is slower. SGD sometimes might offer better Generalization but it is slower compared to ADAM optimizer. Then AdaGRAD can be a useful choice in SISR tasks with its sparse gradient update capability but it may become inefficient for longer training runs. Lastly RMSProp can be a better alternative than SGD and ADAGrad but it is sensitive to initial learning rate[30]. Adam optimizer helps by efficiently navigating the parameter

space over several iterations or epochs to find the optimal values that minimize the loss function. As training progresses through multiple iterations or epochs, Adam optimizer automatically adjusts the learning rates based on how the gradients are changing[31]. Allowing the model to settle into more optimal weights. Furthermore it provides stable convergence specially in deeper neural networks where traditional optimizers might oscillate or get stuck in local minima.

Gaussian Blur

By adding random changes to data, Gaussian Blur noise produces a realistic unpredictability that resembles uncertainty found in the actual world. It has a standard deviation and mean, which are often zero. It also has a Gaussian distribution. Images are processed using Gaussian noise to mimic real-world uncertainties such as sensor noise and ambient conditions. Image processing methods are tested for robustness, ensuring performance in a range of situations, by adding this mathematically well-defined noise[32]. It helps assess and improve noise reduction methods, which are essential for improving image quality.

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

The "Gaussian noise" facilitates standardized statistical analysis, which aids in the creation of efficient image-processing techniques for computer vision applications.

Early Stopping

Early stopping is a technique in machine learning where it is used for regularization. It is designed to prevent both overfitting and underfitting during the model training process. The model's initial job is to find the underlying patterns in the test dataset. The reason for overfitting is when a model captures not only the underlying patterns in the training data but also the noise and minor variations as well[33].This results in excellent training performance but poor generalization to unseen data. On the other hand, underfitting happens when the model is too simplistic or hasn't learned enough that causes it to perform poorly on both training and validation sets. Early stopping helps to eliminate these issues by monitoring the model's performance. It does that on a separate validation set during the dataset training[34]. Once the performance on the validation set ceases to improve for a predetermined number of iterations which is also known as ”patience”, training is halted. This step does not let the model learn unnecessary details from the training data while also ensuring it has been trained sufficiently to capture the essential patterns.

2.8 Evaluation Metrics

In the realm of image super-resolution, different metrics functions are used to evaluate the quality of the high-resolution images that are generated compared to the ground truths. These functions help the user quantify the performance of the super-resolution models by measuring the similarities or differences between the predicted image and the actual high-resolution image. To measure the outcome the most common metrics are PSNR and SSIM.

Peak Signal-to-Noise Ratio (PSNR)

Starting with the PSNR, it stands for Peak Signal to Noise Ratio. PSNR is one of the widely used metrics in the field of signal processing, specifically in the field of Image and comparison. It serves as a quantitative measure where it compares the signal of a compressed signal quality against its original quality[35]. The PSNR is computed by dividing the maximum strength of a signal by the power of noise that tampers with the signal's ability to be represented accurately. The formula for PSNR is defined as follows and is commonly represented in a logarithmic scale:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right)$$

Typically decibel (dB) is used to express the resulting PSNR value. Higher PSNR values indicate the fidelity is greater or that it shows the generated signal reflects the original with little noise or distortion.

Structural Similarity Index (SSIM)

Then comes the SSIM. It is another metric that is commonly and widely used. The PSNR concentrates on pixel by pixel. But for SSIM it takes a different approach. The SSIM concentrates on the perceived structure information as well as the luminance comparison between the generated image and the ground truth. The SSIM is more closely designed to align with human vision[36]. SSIM has three components.

The luminance similarity, the contrast similarity, and the structure similarity. ALL three components are combined to produce an overall SSIM index. It ranges between -1 and 1, where 1 indicates perfect similarity. The formula for SSIM is:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

The SSIM has advantages over PSNR, as SSIM can come close to the human vision system.

2.9 Some Existing Models

SRGAN

SRGAN is a type of GAN architecture which is one of the first and most efficient techniques that allows the model to achieve an upscaling factor of almost 4x for most image visuals[37]. The SRGAN model consists of two main components. One is a generator and the other is a discriminator. The generator upscales the low-resolution images while the discriminator tries to differentiate between the super-resolved images generated by the generator and the original high-resolution images. The discriminator in SRGAN is a binary classifier that helps to differentiate between high-resolution output generated by the generator and original high-resolution images[1]. This model uses a perceptual loss function which consists of an adversarial loss and a content loss. The adversarial loss pushes the solution to the natural image variations, making the super-resolved images more similar to the natural images. The content loss is motivated by perceptual similarity instead of pixel space similarity[38]. Actually, the perceptual loss in SRGAN is a key component of the SISR model that helps it generate high-resolution images that are perceptually similar to the original images.
Afterward, the activation function PReLU is used between the convolutional layers. The data is often normalized to ensure that the pixel values fall within a certain range, Typically between 0 and 1. This helps improve the stability and performance of the training set of processes. Lastly, the data is divided into batches which are used to train the model. Using batches of data instead of individual samples can make the training process more efficient[39].

ESRGAN

ESRGAN goes beyond SRGAN's introduction of the idea of perceptual loss to improve realism. ESRGAN improves perceptual loss functions by prioritizing the preservation of critical visual characteristics more strongly[10]. For the perceptual loss, VGGNet-19 is used. ESRGAN generates super-resolved images that closely mimic real high-resolution content while also appearing sharper thanks to its more efficient use of perceptual loss, which results in a more organic and beautiful final product.

ESRGAN works by estimating and generating a high-resolution image from a low-resolution image. It uses deep neural networks to do this. ESRGAN consists of content loss and adversarial loss. The content loss ensures that the upscaled image is similar to the original HR image[40]. Meanwhile, the adversarial loss ensures the fact that the upscaled images are indistinguishable from real HR images.

The generator in ESRGAN is designed to create high-resolution images from low-resolution inputs. ESRGAN uses RRDB as a generator which combines multi-level residual network and dense connections. RRDB's main focus is to introduce model modification such that the training is efficient and less complex. Afterward, the discriminator in ESRGAN is designed to differentiate between the actual HR image and the generated super-resolved image. It cross-checks whether the upscaled images are real or fake[41]. The discriminator or ESRGAN is a multi-scale attention U-net architecture. It tries to predict the probability that a real image is relatively more realistic than a fake one. These two components work together to create an efficient ESRGAN architecture[42]. One of the benefit ESRGAN's has that it can be trained

with a large number of data and it results in proper output. During training images are artificially corrupted to emulate real-world degradation[40]. The ESRGAN model is then trained to recover the original image or upscale the image. In conclusion, ESRGAN outperforms SRGAN by implementing a more sophisticated RRDB architecture, improving perceptual loss functions, and showcasing adaptability while managing various upscaling factors.

BSRNET

Then comes BSRNET. It stands for Blind Super-Resolution Network and it is a model used for image super-resolution tasks. In the context of image super- resolution, BSRNet might improve the quality of low-resolution contents. However, the actual functioning method of BSRNet in image super-resolution may vary depending on the specific implementation and nature of the content data. The primary concept underlying super-resolution models such as BSRNet is to develop a mapping from low-resolution inputs to high-resolution outputs. This is often accomplished using a deep learning framework that employs vast quantities of training data to learn this mapping. The trained model may then be applied to improve the resolution of previously unnoticed low-quality photos. However, the main problem of implementing BSRNET in video upscaling tasks is, that BSRNET is best suited to image upscaling tasks. To use BSRNET in the context of image upscaling tasks, much computational data may be needed[43].

Chapter 3

Dataset Extraction

3.1 Dataset Description

The dataset "130k Images (128x128) - Universal Image Embeddings" by Rohit Singh, one of the lead engineers at Samsung R&D has been selected for the thesis. The usability rating of the dataset is 8.24 by Kaggle. The database consists of 11 categories with a total of 130k images. From where 6 categories were selected to generate a versatile dataset, the details shown in the chart (3.1). For the training purpose $2,400$ images were selected and for the testing purpose to evaluate the model 600 images were selected. So, in total the dataset stands for $3,000$ images.

The artwork has been selected because this category contains human paintings and designs. In this category, the research papers published on different GAN models have got some satisfactory results. These artworks would help the research to understand how much of an improvement the model can provide in terms of sharpness and details. The apparel category contains pictures of humans and clothing. This will help the research to determine how the model performs on human faces, and how much of an improvement it can provide by upscaling these sorts of images. The clothing in this category can also help to understand how the model may perform in multi-colored objects or clothes. A car is an object that can be easily identified by different generative models. As it has a specific structure it would be a bit easier to feed the model and try to get a more effective upscaling image. The furniture category has different types of chairs, tables, beds, etc. These are selected because the furniture has sharp edges. It would be easier for the model to identify the edges and upscale it. Thus, it would provide refined output. That is the reason why this category was picked. In the case of illustration, it contains different images of cartoons, anime characters, and graffiti. In the papers that were found on GAN architecture, different types of cartoon images were used in the dataset and those researchers got quite favorable results. That is why this category has been chosen. The landmark category was chosen to facilitate more practical real-world implementations. People often take photos of nature. Those pictures may come out blurry or in low-end mobile phones may not take good photos. To address this problem this model has chosen a landmark category which is filled with different famous places and scenarios. By training with this many places pictures the model of this paper would be able to provide some improved pictures. At first, the thesis focused on running the

GAN models on a versatile dataset of three thousand images. Amongst them, for training, 2400 images were used and the model was tested using the remaining six hundred images. Here, after each epoch for validation randomly selected ten percent images from the training images were used which were three hundred of randomly selected images.

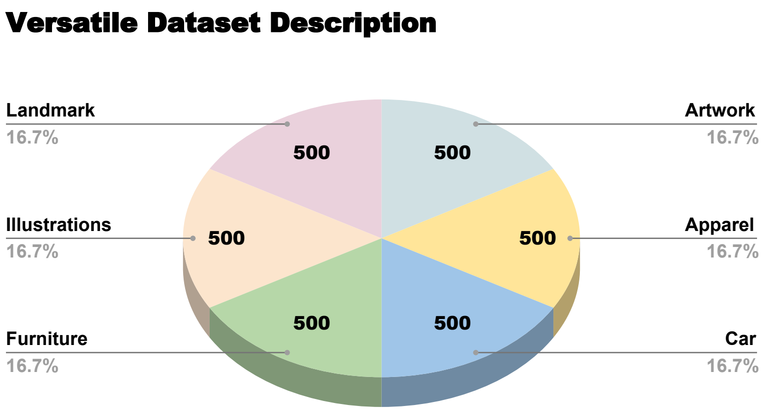

Figure 3.1: Versatile category based dataset description.

From the massive dataset of 130, 000 pictures, that has been taken from Kaggle. Among 10488 images of furniture only 2000 images have been taken. This time the model is working on a single category. The purpose of working with a single category is to test the model and to determine how well it can perform on a dataset of just a single category. The 2000 images that have been selected are of types of furniture. The training dataset contains 1600 pictures from the selective dataset. After every

epoch, the model calculates generation loss, discrimination loss, and validation loss. For the validation, the model is using 10% of the training data, which is on 160 images. These 160 images that are being used for validation are randomly selected after every epoch. And, finally, for testing purposes, the research paper has used 400 of the images in this category. These 400 images have been kept blind from the model. In order to get proper results.

3.2 Dataset Splitting

The whole dataset has been splitted into train and test. Here (3.2) the ratio for this has been 80:20. For training the model 80% dataset was selected and for testing the rest 20%. The 20% for testing the dataset was kept blind from the model while training. Moreover, for validation, 10% of the test dataset was selected randomly.

We have scrambled the data, which suggests that we have mixed up the sequence in each batch, for our model to understand the pattern. This is done to prevent bias in the trained model. Because the model may begin to memorize the same labeled image and fail to train properly if it is shown repeatedly for an extended length of time. When an epoch has ended, the task of the model is to validate. It is essential for calculating the accuracy. So, to put it in numbers where we had 3000 images in total. From that 2400 of them had been used for the training section. Again from the 2400 images randomly selected 240 images were used for validation. The remaining 600 images were used for testing the model and finally evaluating the results. And for the single-category dataset, 2000 images of furniture were selected. This dataset was also split into an 80:20 ratio. So, for testing the amount of images was 1600. Among them, randomly selected 200 images were used for validation. Finally, 400 images were used in the testing part. And the performance of the model was evaluated by these 400 images where the model has learned about only a specified category.

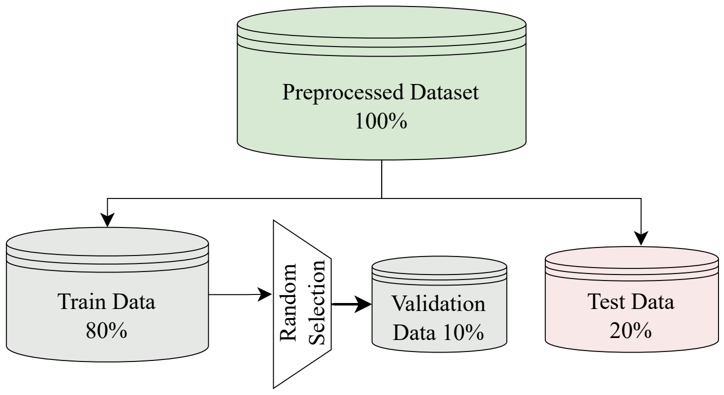

Figure 3.2: The process of splitting the dataset.

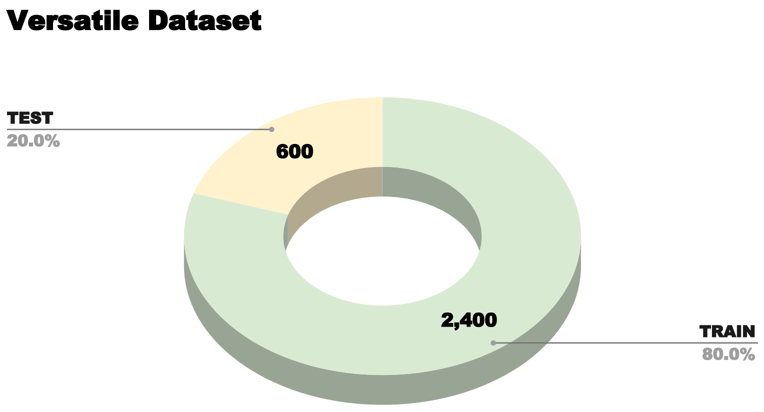

Figure 3.3: Versatile category dataset splitting for train & test.

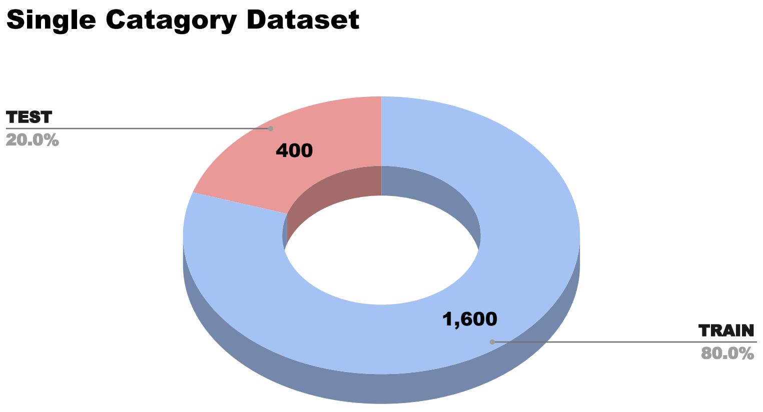

Figure 3.4: Single category dataset splitting for train & test.

3.3 Dataset Preprocessing

Down-scaling the Dataset

Deep learning models require a set of pair images for training purposes. The first dataset is made up of HR pictures, which are able to be regarded as the ground truth (GT). In order to train the model, a LR set of the data is required. To achieve the LR images OpenCV library was used to convert the main dataset into a 4× LR dataset (3.5). In this process, the whole dataset was taken in a single folder to take as the input path for the code used in downscaling. The downscaler code works based on OpenCV and OS library where it resized the each data into a 4× lower resolution. The result is kept in a different folder as the training LR where each data was named exactly the same as its input name. So, this process produced exact same pair of datasets among which one was the base HR set and another was the 4× LR dataset.

Image Denoising

There are many Image Denoising methods. But for this research purpose, we have chosen the "Gaussian Noise" method. By adding random changes to data, Gaussian noise produces a realistic unpredictability that resembles uncertainty found in the actual world. It has a standard deviation and mean, which are often zero, and it has a Gaussian distribution. Images are processed using Gaussian noise to mimic real-world uncertainties such as sensor noise and ambient conditions. Image processing methods are tested for robustness, ensuring performance in a range of situations, by adding this mathematically well-defined noise. It helps assess and improve noise reduction in images. It is essential for improving the image quality. The "Gaussian noise" facilitates standardized statistical analysis, which aids in the creation of efficient image-processing techniques for computer vision applications.

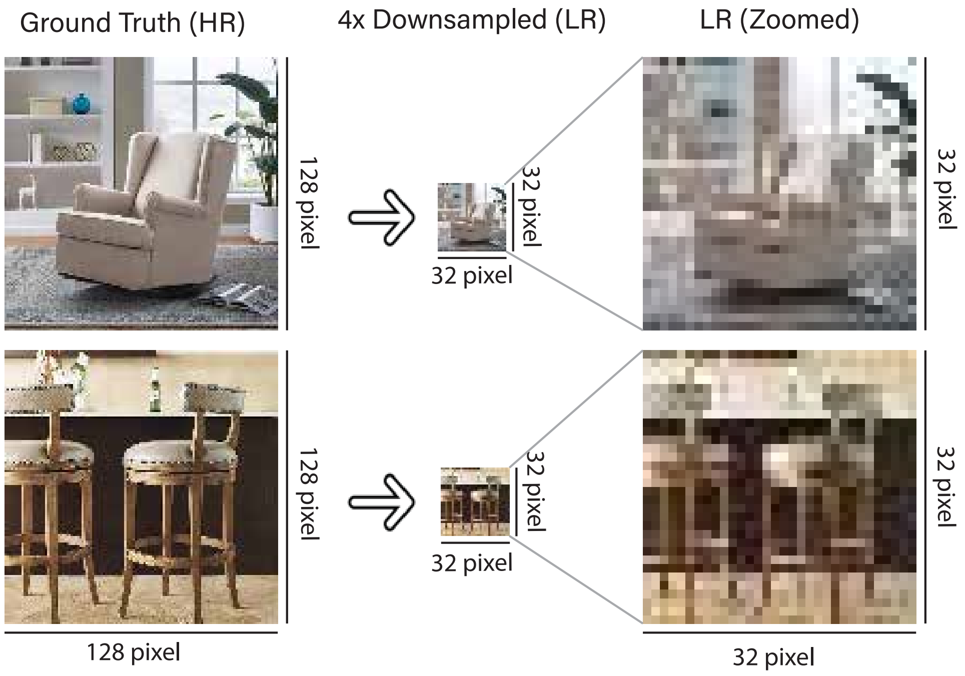

Figure 3.5: Down-scaling the dataset to 4x LR.

Chapter 4

Model Implementation

4.1 Experimental Framework

Hardware Specification

The research is based on a model which is applicable for training on consumer grade GPUs. For this purpose the investigation started from the top level of consumer grade GPUs and eventually by optimization, the model was able to run on an average consumer grade GPU. Until October 2024 the best performing consumer grade GPUs that were available in the market was the NVIDIA GeForce RTX4090 which has 24 GB of vram. As a result the initial research started with the best performing GPU to make sure the proper development of the GAN models. Also to avoid bottleneck issues the processor's computational power was also considered. By using Intel Core i9 13900K, the bottleneck issues will be solved. Also the system had 64 GB ram configured in dual channel mode. To combine the whole setup Gigabyte Z790 UD AX DDR5 motherboard was used with sufficient power supply. After successful generation and results the models were optimized to shift on an average to high level GPU named the NVIDIA GeForce RTX3060 Ti. This GPU had a 8 GB vram variant. Eventually the models were fitted to train on this configuration. Lastly an average GPU NVIDIA GeForce GTX 1660 was used to test which had a 6 GB of vram. On this GPU the model took a comparatively higher time but was able to complete the training and testing which was the primary objective of this thesis. It had to develop a balance trade off with the resolution and the hardware limitation. The GPU used in the research was examined properly and the benchmark tested from trustable websites. The result of the benchmark test is given below (4.1). Benchmark testing is based on some factors like the computational power, synthetic or 3D rendering, probable frame per second generation capability etc. The current baseline of the benchmark test is RTX 3060 Ti which is being considered as the 100% on the average performance. Where RTX 4090 has a comparative 370% average performance and GTX 1660 has a 68.3% average performance.

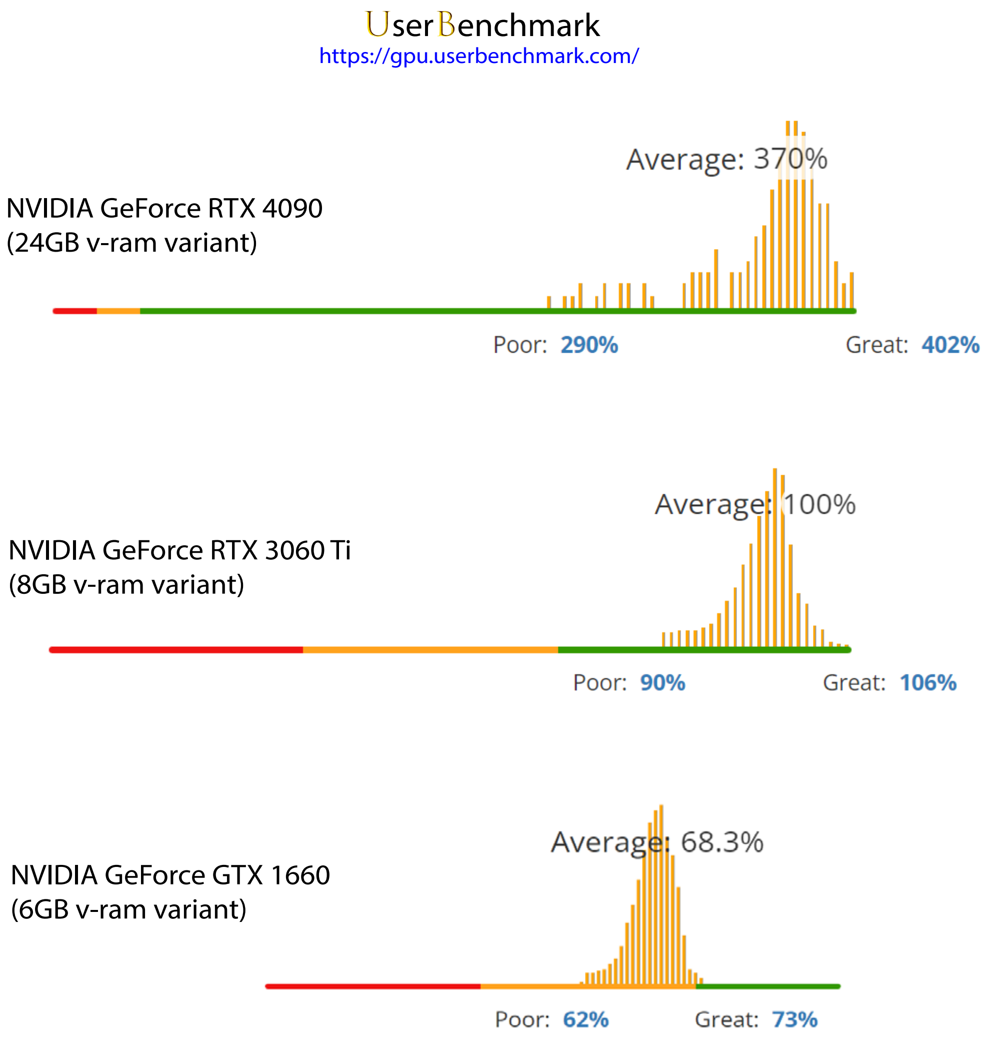

Figure 4.1: UserBenchmark testing result for GPUs utilized on the thesis.

Software Environment Setup

To run the models, Jupyter Notebook was chosen as the development environment. Jupyter Notebook is a flexible and user-friendly interface, at the same time it is good for deep learning training and testing. The models were developed in Python language where the research used Python version 3.12.4 and the software Jupyter Notebook version was 7.0.8. The language Python has rich deep-learning libraries and frameworks. At the same time, it has an active community and the developing team is updating the language regularly.
There are several benefits of using the Jupyter Notebook. It provides both coding and documentation at the same time on its platform. As a result, the codes can be marked down alongside the real-time code execution. Moreover, the cell-based execution system of Jupyter Notebook gives a huge advantage compared to the traditional IDEs like Visual Studio Code, PyCharm, etc. Cell-based execution systems enable the execution of specific parts of the code whenever needed from a single .ipynb file. Furthermore, this software is highly compatible to combine the GPU-powered environments like Google Colab, Kaggle. But it runs on an offline environment enabling specific hardware using opportunity. Finally the visualization and analysis ability through different libraries like matplotlib integrates seamlessly. This helps to have a proper visualization of training curves and results.

Library Overview and Functional Roles

The main training process was based on the PyTorch library where from handling the tensors to optimizing the result different libraries were used. To have a depth view the libraries and their functionalities on the MSRGAN are explained in this part.At the very beginning the existence of the computational device was tested by torch.device library. Where by using the torch.cuda.is_available the detected device(GPU's) availability was ensured. To develop the neural network the torch.nn library was utilized. Where nn.Module was the base class of neural network modules. The 2D convolutional layers were built by nn.Conv2d. For activation functions nn.LeakyReLU, nn.PReLU was introduced. Moreover the loss functions like nn.BCEWithLogitsLoss for binary classification were also part of the torch.nn library. Furthermore nn.Sequential was used for combining layers to modify the ResNet. Then for optimizing the parameters during the training optim.Adam function was used to utilize Adam optimizer. To load the dataset customly torch,utils.data library was used. Also, the os library helped to combine the paths or directory of the dataset. For applying the downsampling, a library named open CV was being used to trace the images as read and save them as required format. Resizing and adding Gaussian blur was also part of this library. For randomizing the validation dataset the random function was imported. To evaluate the results skimage.metrics were used for PSNR and SSIM evaluation metric calculations. Also for representing the execution progress tqdm library used. Lastly, besides some basic functions like numpy, the matplotlib.pyplot was utilized to observe the training curves.

4.2 Proposed Model: MSRGAN

The final goal of implementing the expected model was a modified version of SR-GAN. As a result the thesis focused on a step by step approach to initially develop the SRGAN then the RRDB model based Enhanced-SRGAN and finally the MSR-GAN. The process would help the research to have a balanced development of these GAN models and the result analysis could compare the accuracy with these models.

The Generator class consists of a residual dense block, which was the building block of the RRDB module. It consisted of five convolutional layers with LeakyReLU and PReLU activations[44] (4.2). These convolutional layers were designed to extract features from the input image. The Leaky ReLU activation function applied non-linearity to the output. By using LeakyReLU and PReLU in an alternating manner, a hierarchical feature extraction process was created. The concatenation of features from different layers allowed for feature fusion, which combined the strengths of both activation functions.

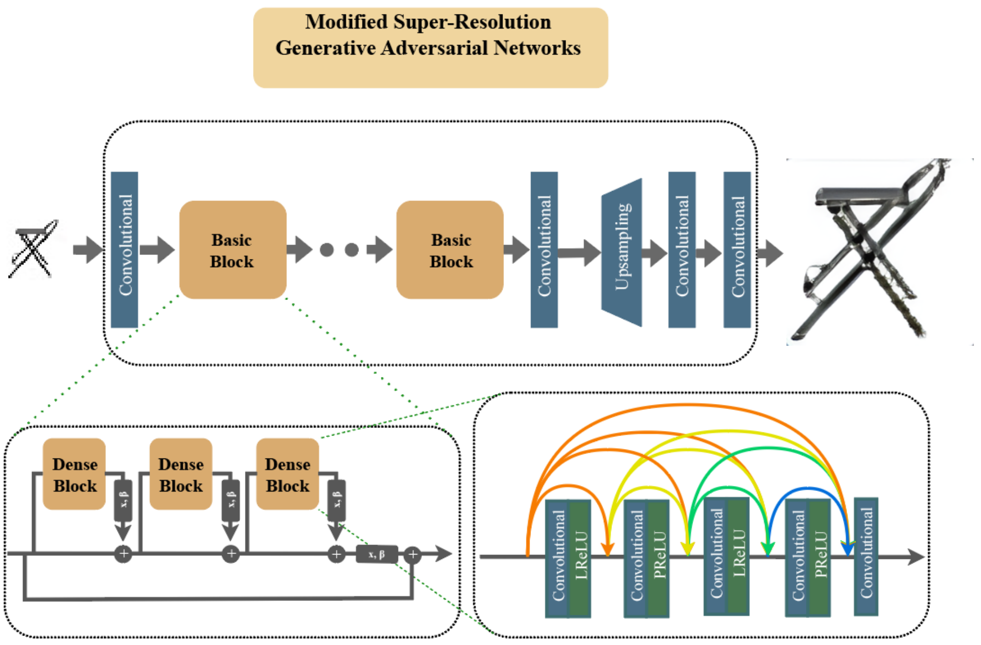

Figure 4.2: MSRGAN with Modified RRDB activation functions.

In the figure (4.2) the RRDB combined multilevel residual networks, where residual learning was used at different levels[45]. Dense blocks were also used in the main path for increasing the network's capacity. The RRDB module featured a residual-in-residual architecture, where each residual dense block was encapsulated within an additional residual connection. It was designed to learn residual functions that can be added to the input of the block to produce the output. The output of each residual block was linked with the input of the next residual block, creating a dense connection between the blocks[46].

The discriminator also used the combination of LReLU and PReLU which created a more robust and flexible process[44]. It consisted of eight convolutional layers with LeakyReLU and PReLU activations followed by two fully connected layers with PReLU and lastly Sigmoid activation functions for output. The convolutional layers extracted features and downsampled the image through stride-2 convolutions. The data was prepared for the fully connected layers as the output feature maps were flattened into a single vector. The fully connected layers produced a probability value indicating the authenticity of the input image. The LeakyReLU activation function introduced nonlinearity and the PReLU activation function was used in the first fully connected layer to adaptively learn the activation function. The sigmoid activation function was used to produce a probability value between 0 and 1.

The perceptual similarity between the generated HR image and the GT image was ensured by content loss. Rather than pixel wise comparison this model compared features extracted from a pretrained Resnet-18 model. Both the generated image and HR image are passed through the resnet model for feature extraction from the layers before the fully connected layer. The L1 loss computed the absolute contrast between these feature maps to ensure quality in a perceptual sense[28].

The adversarial loss encouraged the generator to produce authentic, high-resolution images that can trick the discriminator. This loss enabled the model to produce images with greater details and better quality. The derived HR image is sent to the discriminator, which returns a probability indicating whether it was real or not. The adversarial loss was estimated using binary cross-entropy with logits[29].

The total generator loss included both the content loss and adversarial loss. By balancing these two aspects, the generator could create high-resolution images that were both visually appealing and structurally and detail-wise similar to the GT images in terms of structure and details. The total loss for the generator was a weighted summation of the content loss and the adversarial loss. In this case, the adversarial loss is weighted by 0.01, emphasizing the content loss.

The discriminator's loss was a combination of both real and fake prediction losses. The discriminator's task was to differentiate between real high-resolution images and the generated fake images. Its loss was computed by comparing the real and fake predictions with the corresponding GT. For real images, the discriminator predicts probabilities using binary cross-entropy, with real images labeled as 1. For fake images generated by the generator, the target is 0[29].

SR for higher resolution inputs utilizing MSRGAN

The model MSRGAN initially capable of upsampling a 32x32 pixel image into a 128x128 pixel image. But in real life almost all images have higher resolutions. So, to upscale a higher resolution image a splitting and part by part upscaling trick was tested on MSRGAN. For example a 512x512 pixel image was first downsampled into a 128x128 pixel image as the initial LR input against the 512x512 GT. Then that 128x128 pixel image was splitted into 16 smaller 32x32 pixel images. Each of the images were upsampled 4x to 128x128 pixel images by the model. Lastly these 16 generated HR images were merged together sequentially to have the final

512x512 pixel output. This technique of splitting and separately upscaling can give the model the capability to upscale any resolution of images.

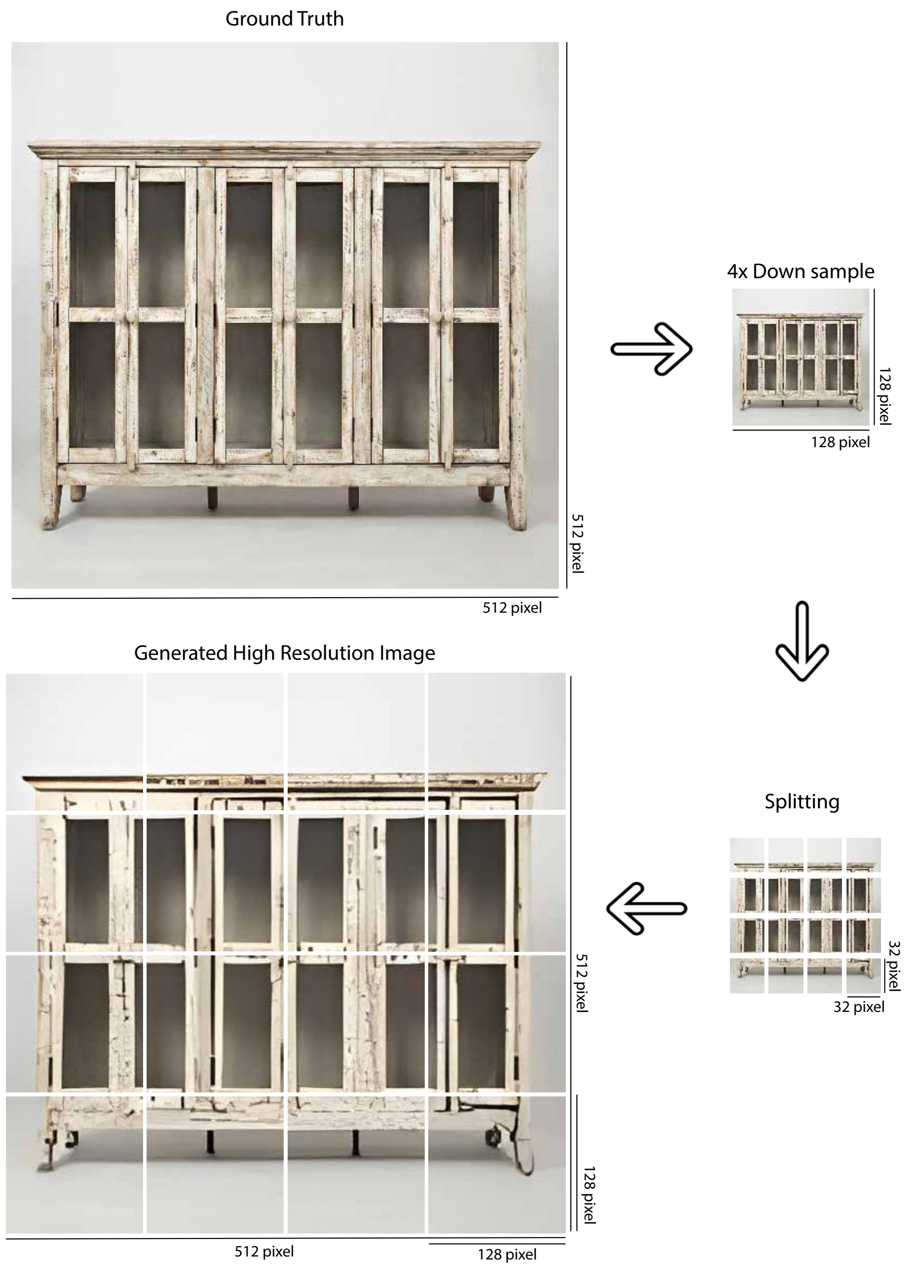

Figure 4.3: SR for higher resolution images with MSRGAN

4.3 Development of SRGAN & ESRGAN

To compare the result of MSRGAN with the existing models the research needed to implement some existing models for having a proper evaluation. Here the initial SR model of SRGAN and the RRDB based and upgraded version ESRGAN was implemented.

SRGAN

The purpose of SRGAN was to enhance the resolution of LR images by utilizing deep neural networks. The SRGAN model consisted of two main segments. The first part was the generator and the second was the discriminator. LR images were upscaled in generator part while the discriminator tried to differentiate between the generated fake HR images and the ground truth which was the original HR images[1]

The Generator class for this model had five convolutional layers. These layers were followed by the ReLU activation function. The initial convolutional layer started with 64 feature maps, a kernel size of 9x9. This layer extracted low-level features from the images. Subsequent convolutional layers maintained the equal number of feature maps (64) and kernel size 3x3. The final layer had 3 feature maps representing the number of color channels, with a kernel size of 9x9. After each convolutional layer, A ReLU activation function (self.relu) is applied element wise to introduce nonlinearity into the network. The forward method passed forward through the network. It was sequentially applied in each layer for passing the result through ReLU activations.

The primary role of the discriminator was to differentiate between HR images and those generated fake images by the generator. The architecture consisted of a series of convolutional layers followed by fully linked layers, and it used nonlinear activations to learn complicated discriminative features. The discriminator class consisted of eight convolutional layers followed by ReLU activation functions. The initial convolutional layer took a 3-channel input image (RGB) including 64 feature maps and a kernel size of 3x3. From the second to eight convolutional layers the depth of feature maps was increased while the spatial dimensions were reduced through a combination of stride 2 and padding 1. Each of these 8 convolutional layers was followed by a ReLU activation for addressing the vanishing gradient problem and introduces non-linearity. After that, the data was prepared for the fully connected layers as the output feature maps are flattened into a single vector. Then the flattened feature vector was transformed into a 1024 dimensional representation by the first fully connected layer again followed by a ReLU activation. Lastly, the final fully connected layer was reduced dimensionality to a single scalar representing the input image is real. In the forward method the input image wass sequentially passed through every convolutional layer, followed by ReLU activations and culminating in a single output value.

The perceptual loss function was critical for evaluating the performance of the generator network. Mainly, the perpetual loss function contains content loss and adversarial loss. Here, mean squared error loss was used to calculate content loss.

MSELoss was mainly utilized for regression tasks where the network output should be consistently similar to the goal values. It computed the average of the squared differences between the projected outcomes and the genuine target value. Again, Binary Cross-Entropy Loss and the sigmoid layer were combined for calculating adversarial loss. By making a combination of these operations into a single layer, the model had the advantage of the log-sum-exp for numerical stability[1].

ESRGAN

The methodology for ESRGAN comprises a modified network architecture, a relativistic discriminator, and a perceptual loss function. For the generator, the network architecture was based on SRResNet where the majority of the computations were done in low-resolution feature space. The generator was constructed by striking out the BN layers and replacing the basic block with RRDB of the SRGAN model. The BN layers were removed because they introduced undesirable artifacts and compromised the generation ability when trained under a GAN network. This action improved the generalization ability and reduced memory utilization and computational complexity. The RRDB combines multilevel residual networks in which residual learning is implemented at different levels . Dense blocks were also employed along the main path to increase the network's capacity[3]

Each residual block in the RRDB consists of 5 convolutional layers. These convolutional layers were constructed to derive features from the input image, and they are followed by activation functions ReLU. The activation function will apply leaky ReLU non-linearity to the output of each convolutional layer, with a negative slope of 0.2. The RRDB module was designed to have a residual-in-residual structure. RRDB's each residual dense block was nested inside another residual connection. It was designed to learn residual functions, which are functions that can be added to the input of the block to produce the output. The output of each residual block was linked with the input of the next residual block, creating a dense connection between the blocks[3].

The discriminator was enhanced based on the Relativistic GAN, which predicted the probability of relativity between a real image and a fake one[47]. This approach encouraged the generator to retrieve more realistic texture details. The RaD discriminator works by taking the generated image and the real image as inputs. A convolutional neural network was used to extract features from the input images. Then the extracted features were passed through a series of fully connected layers to produce a probability distribution over the possible outputs. The discriminator's output was a probability value that indicated the likelihood of the generated image being more realistic than the real image.

The perceptual loss function was developed more effectively by applying constraints on features before activation instead of after, as commonly practiced in SRGAN. This approach overcame sparse activation and inconsistent reconstructed brightness of the original design. A variant of perceptual loss using a fine-tuned VGG19 network for material recognition was used[3].

Chapter 5

Result Analysis

5.1 Inspecting Loss Curves

Result on Versatile Category Dataset

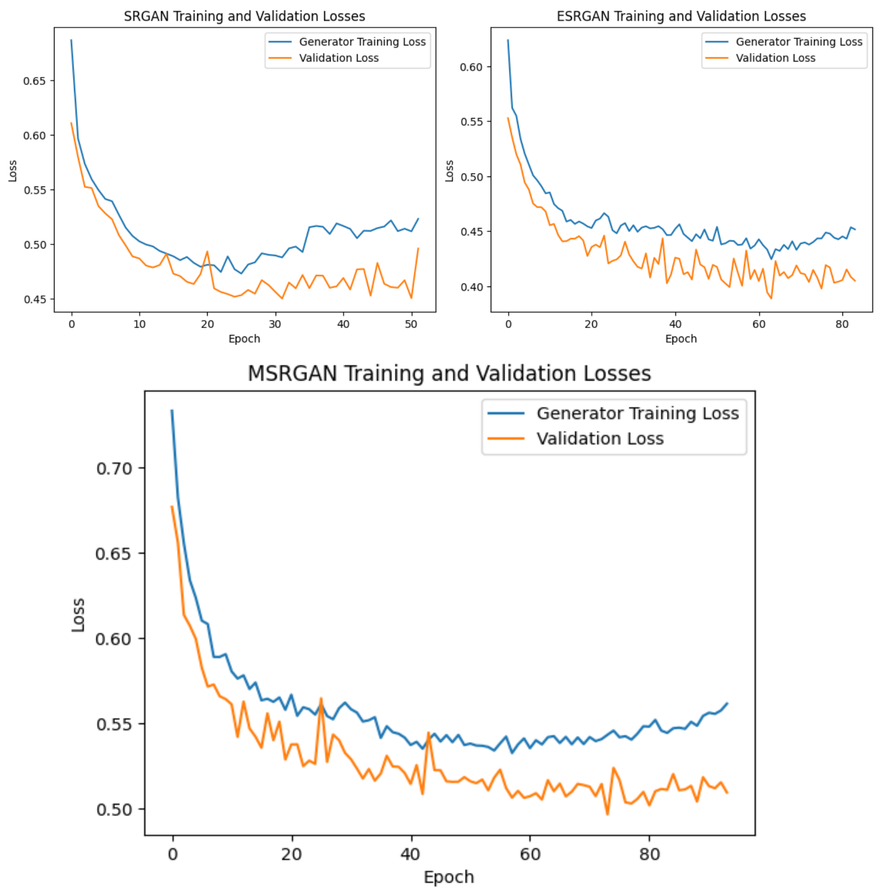

Figure 5.1: Train and Validation curves on versatile dataset.

The training curves (5.1) in the versatile dataset for the models ESRGAN and MSRGAN shows a balanced learning process. Where SRGAN trained for 25 epoch, ESRGAN had 115 epoch and MSRGAN runned for 197 epoch. For both the model early stopping was applied to by utilizing a patience of 5 epoch. So, when the learning accuracy was a continuous downgrade for more than five epochs ythe model would stop and consider the best value. At the same time, the patience level would help the models to overcome local minima to ensure a balance of learning and avoiding under-fitting. As a result the research could have proper models which were free from the under-fitting or over-fitting issue.

Result on Single Category Dataset

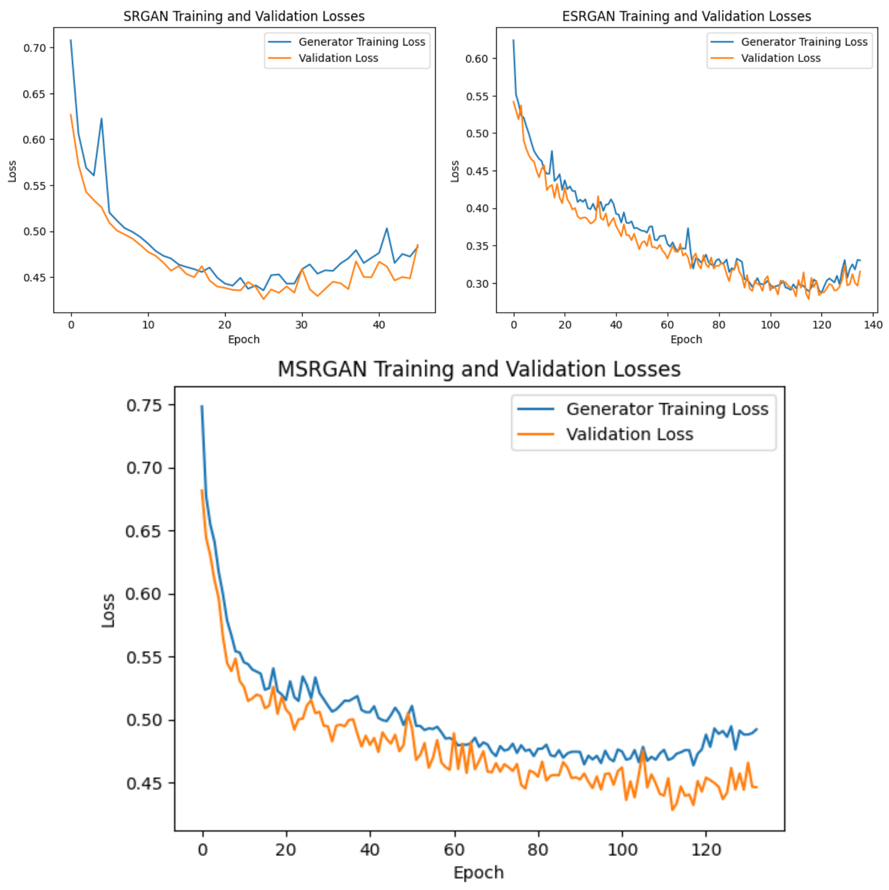

Figure 5.2: Train and Validation curves on single category dataset.

The training progress for the models on the single category dataset was better than the versatile dataset. A total of 13 epochs were spent training SRGAN, 152 epochs were spent training ESRGAN, and 123 epochs were spent training MSRGAN (5.2). The patience was 5 for early stopping to avoid the over fitting issue. Here SRGAN learned faster as it was a basic model and complexity was less as well as the feature extraction from both ESRGAN and MSRGAN. The model MSRGAN and ESRGAN might have local minima problems. So, for the better result and for avoiding any local minima the patience count was increased from 5 to 20 for both ESRGAN and MSRGAN. So, the under-fitting or overfitting was handled carefully. Also, the complex models of ESRGAN and MSRGAN was optimized with Adam for faster learning. Over all, according to the loss curve, the models were trained in a proper way.

5.2 Evaluation Metric Result

Result on Versatile Category Dataset

The testing results of the model were compared by two evaluation metrics. The metrics were PSNR and SSIM. At first for the versatile dataset the test dataset gave a result which was almost similar to each other. As we can see in the figure (5.3) the PSNR result for SRGAN was 15.85, for ESRGAN was 16.55 and for MSRGAN was 16.97. So, the proposed model MSRGAN performed slightly better than both ESRGAN and SRGAN. On the other side the result matrix (5.3) shows the SSIM where SRGAN was 0.2468, ESRGAN was 0.3638 and MSRGAN was 0.4820. So in the SSIM the proposed model also performed better. Here MSRGAN's generated images has a 48.2% similarity with the ground truth where both the SRGAN and ESRGAN has less than 40% similarity. But the result was not good enough to satisfy human eye as a super resolved image.

Result on Single Category Dataset

The metrics (5.4) showed the fluctuated values of SSIM and PSNR for images from the test dataset. Here for SRGAN the max PSNR crossed 32.5 which was really good but the lowest result crossed 12.5 which was not optimal. As a result the model SRGAN generated an imbalance result with an average PSNR of 17.84. Then the model ESRGAN had a slightly better performance with highest PSNR of 30 and lowesr PSNR of 13 and an average of 18.73. Finally the model developed in the thesis performed an optimal and balanced result compared with the SRGAN and ESRGAN with an average result of 19.17. Where the highest PSNR was near 29 and lowest PSNR was around 14. Next in the SSIM metric, MSRGAN also outperformed the ESRGAN and SRGAN model with the average similarity index of 0.6731 where SRGAN had 0.6271 and ESRGAN had 0.6184. So, the generated images in MSRGAN had a 67.31% similarity with the ground truth. As the models are based on very small sized images the models struggle to generate better outcomes. Without any additional enhancing libraries or features the results were really good where MSRGAN performed better in terms of both SSIM and PSNR metrics.

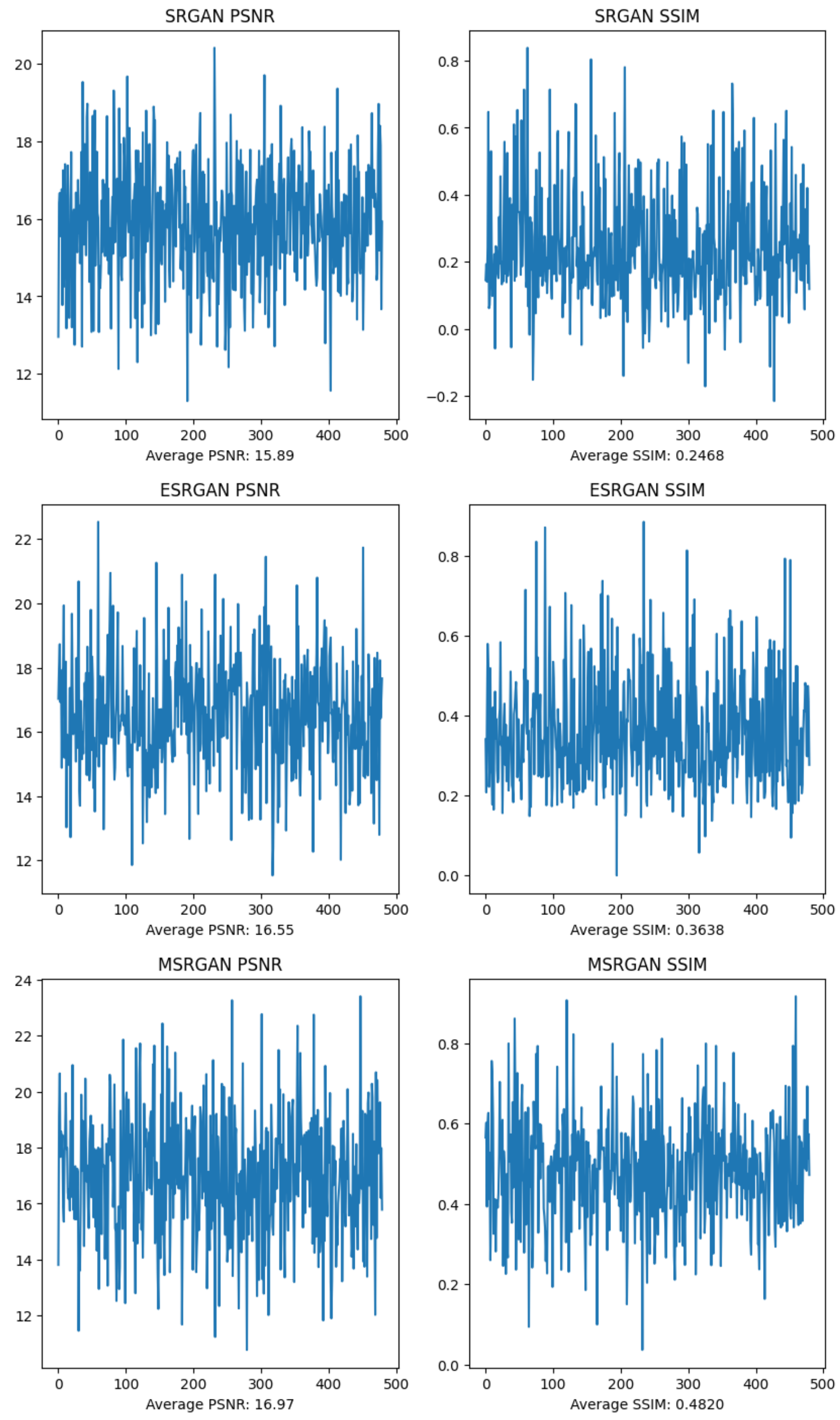

Figure 5.3: PSNR and SSIM result on versatilele category dataset.

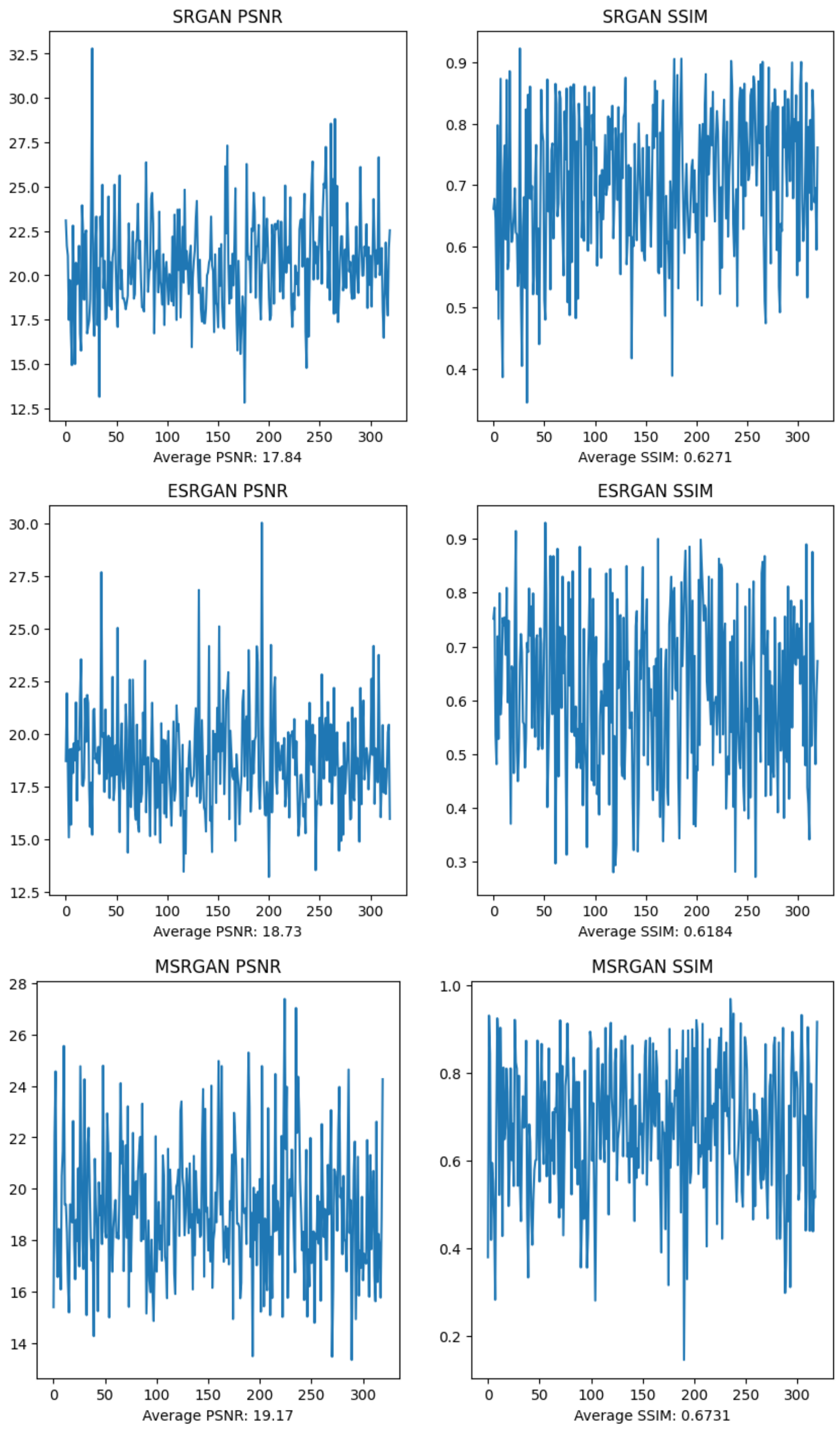

Figure 5.4: PSNR and SSIM result on single category dataset.

5.3 Generated Image Visual Comparison

Result on Versatile Category Dataset

The result of the versatile category test dataset for all the models performed below average. The main reason behind that was the size of the image and the category variation. For an example in the figure (5.5) shown image was one of the images from the apparel category where the LR was barely understandable as a human face. The super resolved image developed the outer shape but was unable to detect the details as the LR was too pixelated. So, the versatile category result was less than expected and the thesis next experimented on the single category dataset.

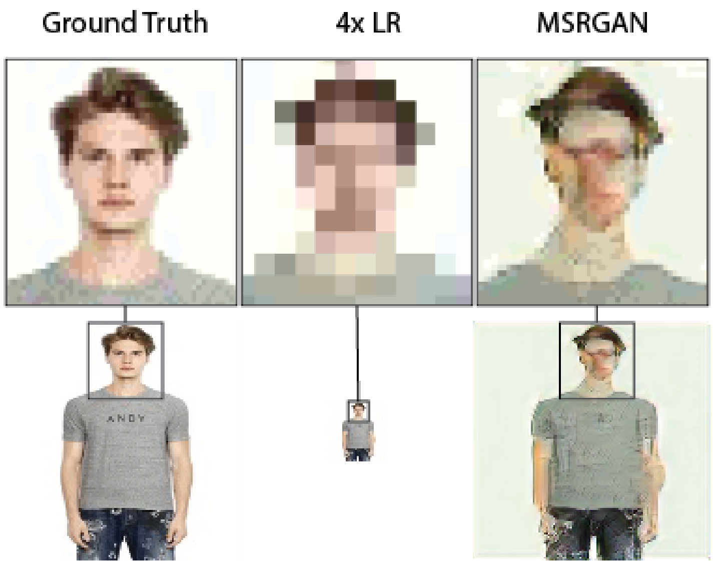

Figure 5.5: Versatile category super resolution result example.

Result on Single Category Dataset

After training the model from the test dataset some of the generated images were shown in the figure (5.6) to visualize the dissimilarity and variations between the models output. The ground truth was the main dataset's HR images of 128x128 pixels. Those ground truth images were downsampled 4x and converted into 32x32 pixels which were used as the low resolution input for the generator of the models. From each model that is being trained, the generator generated fake HR images which were the output of the model. Below some of the generated images with their HR and LR has been shown. From the comparison the MSRGAN also outperformed the ESRGAN's and SRGAN's images with better visualization experience. The output of the SRGAN was blurry and noisy, where ESRGAN was sharper than natural and lastly the MSRGAN's generated image was better in terms of both the color balance and accuracy.

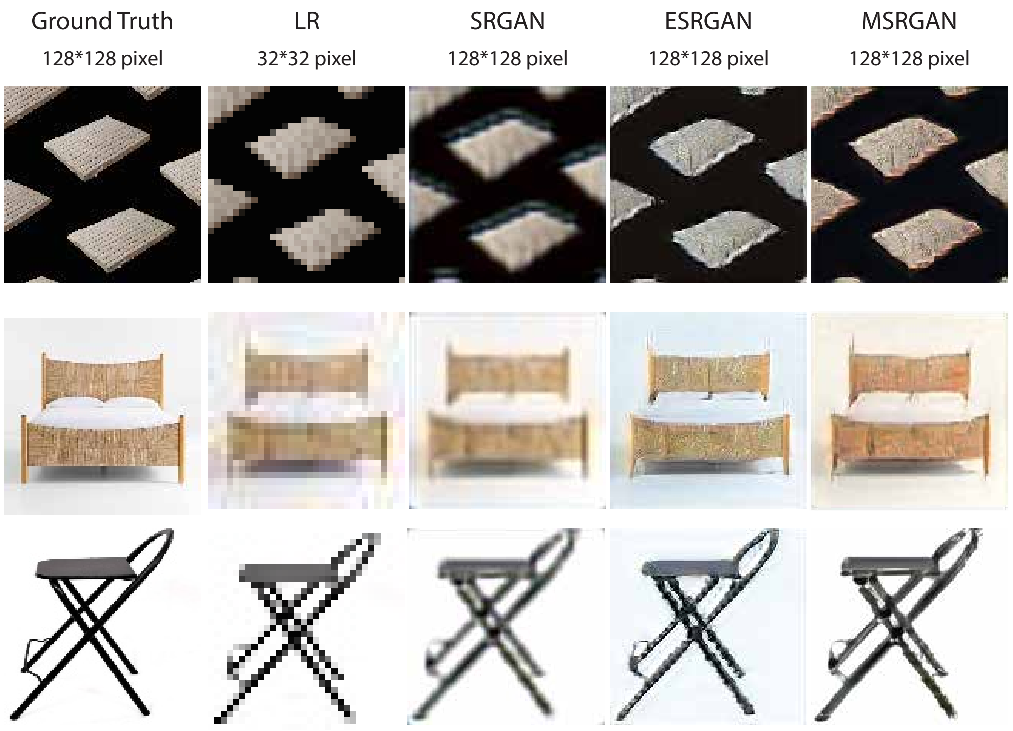

Figure 5.6: Visualization of SRGAN, ESRGAN and MSRGAN results.

Chapter 6

Conclusion

6.1 Limitation

For the research, the model uses a consumer-grade GPU. Because of this, the ground truth images we have selected were 128x128 pixels. The average user who has a minimum 6GB Visual RAM, can train and test the model with this particular resolution. This model is capable of 4x upscaling. So, a person can provide a 32*32 pixel low-resolution image and will have a 128*128 enhanced high-resolution image. Based on these 128x128 pixels images we are getting this sort of output. If we want to get an enhanced output we have to deliver a better input set. Which is not possible with the current state-of-the-art consumer-grade GPU. So as a result, we have to use professional-grade GPUs such as NVIDIA RTX A6000, NVIDIA Quadro RTX 8000, etc. If those kinds of GPUs are used the model would be able to work with 1080p-4K images.

The consumer grade GPUs selected to train and test the models were chosen in a way that an average person would be capable of going through the experience. At first the research used the highest consumer grade GPU till October 2024 RTX 4090 (24 GB variant) where the model performed really well and the time to train the model was comparatively much faster. At the same time the model had the capability of handling a comparative higher pixel LR as the usable visual memory was 24 GB. But the same model with the same type of LR inputs failed on a RTX 3060 (8GB variant) due to the lack of visual-ram. As a result the further research targeted a low-resolution dataset as ground truth. After many attempts and combinations the research developed a way to make the model executable on an average consumer grade GPU with a minimum 4 to 6 GB visual ram. At the same time the method of splitting and generating overcomes this problem in a simpler way.

In the case of the model used in this research, incorporating additional internal blocks or layers significantly increases the time required to generate the output. The extended processing time, relative to the quality of the output, diminishes the model's comprehensive efficiency. It is crucial to strike a balance between the number of layers and the depth of feature extraction as it will help to enhance the time efficiency. It will ensure that the model remains both computationally feasible and effective in producing the desired results.

6.2 Future Works

This model we have developed has potential. But at this point, the model needs some refining. In order to make the model better the paper suggests some pointers. It would help the model to achieve far higher heights.

- Incorporate attention mechanisms to focus on certain portions of the image, which could lead to greater performance and lower processing requirements.
- Investigate the feasibility of extending the MSRGAN model to include multi-scale super-resolution, for more flexible and efficient image upscaling.
- Explore the use of knowledge distillation techniques to transfer knowledge from a more extensive pre-trained model to the MSRGAN model, for reducing training duration and processing requirements.

By implementing these things the model would be better. At the moment it can not be implemented due to resource and time constraints. But the model has merit where it can help in the field of image super resolution.

6.3 Conclusion

In this thesis a custom GAN model named MSRGAN was developed and evaluated against the established models for image super resolution under the condition of customer Grade GPUs. These models included the basic SRGAN and ESRGAN models. Compared to those models the proposed model does perform superior. In the context of image super-resolution, the MSRGAN model has done a great job. In this research one of the major objectives was to develop a super-resolution model that is capable of running on the consumer grade GPU. This model is not computational power hungry as the typical model's require. Yet it has produced a better performance. As this model works with a specific pixel based 4x upscaling it does not require a heavy computational power so a person with a consumer grade GPU can also run to train and test the model. One can even customize the model as he/she likes. For young students and researchers interested in computer vision, this creates new opportunities without requiring industry grade hardwares. The hurdle which was so high to leap once, can be easily surpassed.

So in conclusion the MSRGAN model is a very capable image super-resolution model where it can surpass the basic SRGAN, ESRAN models. On top of that it is more computational power efficient compared to other existing models. Thus, this model may serve as a stepping stone for the development in the field of image super-resolution.

Bibliography

[1] C. Ledig, L. Theis, F. Huszar, *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 (CVPR)*, Jul. 2017.

[2] B. Liu and J. Chen, "A super resolution algorithm based on attention mechanism and srgan network," *IEEE Access*, vol. 9, pp. 139 138–139 145, 2021. DOI: 10.1109/ACCESS.2021.3100069.

[3] X. Wang, K. Yu, S. Wu, *et al.*, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Sep. 2018.

[4] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, *Activation functions: Comparison of trends in practice and research for deep learning*, 2018. arXiv: 1811.03378 `[cs.LG]`. [Online]. Available: https://arxiv.org/abs/1811.03378.

[5] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, *Activation functions in deep learning: A comprehensive survey and benchmark*, 2022. arXiv: 2109.14545 `[cs.LG]`. [Online]. Available: https://arxiv.org/abs/2109.14545.

[6] K. Janocha and W. M. Czarnecki, *On loss functions for deep neural networks in classification*, 2017. arXiv: 1702.05659 `[cs.LG]`. [Online]. Available: https://arxiv.org/abs/1702.05659.

[7] W. Shi, J. Caballero, F. Huszar, *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[8] M.-Y. Liu, X. Huang, J. Yu, T.-C. Wang, and A. Mallya, "Generative adversarial networks for image and video synthesis: Algorithms and applications," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 839–862, 2021. DOI: 10.1109/JPROC.2021.3049196.

[9] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Trans. Graph.*, vol. 30, no. 2, Apr. 2011, ISSN: 0730-0301. DOI: 10.1145/1944846.1944852. [Online]. Available: https://doi.org/10.1145/1944846.1944852.

[10] X. Xiang, Y. Tian, V. Rengarajan, L. D. Young, B. Zhu, and R. Ranjan, "Learning spatio-temporal downsampling for effective video upscaling," in *European Conference on Computer Vision*, Springer, 2022, pp. 162–181.

[11] J. Li, J. Cao, Z. Zou, *et al.*, *Distillation-free one-step diffusion for real-world image super-resolution*, 2024. arXiv: 2410.04224 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/2410.04224.

[12] M. Zareapoor, M. E. Celebi, and J. Yang, "Diverse adversarial network for image super-resolution," *Signal Processing: Image Communication*, vol. 74, pp. 191–200, 2019, ISSN: 0923-5965. DOI: 10.1016/j.image.2019.02.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596518309937.

[13] A. Dongare, R. Kharde, A. D. Kachare, *et al.*, "Introduction to artificial neural network," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 1, pp. 189–194, 2012.

[14] B. Zhang, S. Gu, B. Zhang, *et al.*, "Styleswin: Transformer-based gan for high-resolution image generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 11 304–11 314.

[15] T. Arora and R. Soni, "A review of techniques to detect the gan-generated fake images," *Generative Adversarial Networks for Image-to-Image Translation*, pp. 125–159, 2021.

[16] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.

[17] DataDrivenInvestor, *Introduction to how cnns work*, https://medium.datadriveninvestor.com/introduction-to-how-cnns-work-77e0e4cde99b, Accessed: 2024-10-14, 2019.

[18] L. Alzubaidi, J. Zhang, A. J. Humaidi, *et al.*, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[19] Á. Zarándy, C. Rekeczky, P. Szolgay, and L. O. Chua, "Overview of cnn research: 25 years history and the current trends," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 401–404. DOI: 10.1109/ISCAS.2015.7168655.

[20] C. Zhang, P. Benz, D. M. Argaw, *et al.*, "Resnet or densenet? introducing dense shortcuts to resnet," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2021, pp. 3550–3559.

[21] J. Liang, "Image classification based on resnet," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1634, 2020, p. 012 110.

[22] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," in *Bio-inspired Neurocomputing*, A. K. Bhoi, P. K. Mallick, C.-M. Liu, and V. E. Balas, Eds. Singapore: Springer Singapore, 2021, pp. 203–224, ISBN: 978-981-15-5495-7. DOI: 10.1007/978-981-15-5495-7_11. [Online]. Available: https://doi.org/10.1007/978-981-15-5495-7_11.

[23] J. Brownlee, *Rectified linear activation function for deep learning neural networks*, https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/, Accessed: 2024-10-14, 2019.

[24] Y. Guo, S. Li, and G. Lerman, *The effect of leaky relus on the training and generalization of overparameterized networks*, 2024. arXiv: 2402.11942 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2402.11942.

[25] T. D. Science, *Comparison of activation functions for deep neural networks*, https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a, Accessed: 2024-10-14, 2019.

[26] S. M. A. Bashir, Y. Wang, M. Khan, and Y. Niu, “A comprehensive review of deep learning-based single image super-resolution,” *PeerJ Computer Science*, vol. 7, e621, 2021.

[27] S. Sharma, S. Sharma, and A. Athaiya, “Activation functions in neural networks,” *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[28] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss functions for neural networks for image processing,” *arXiv preprint arXiv:1511.08861*, 2015.

[29] U. Ruby and V. Yendapalli, “Binary cross entropy with deep learning technique for image classification,” *Int. J. Adv. Trends Comput. Sci. Eng*, vol. 9, no. 10, 2020.

[30] T. Lin and C. Lin, “Single hyperspectral image super-resolution using admm-adam theory,” English, in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*, ser. International Geoscience and Remote Sensing Symposium (IGARSS), Funding Information: This study was supported partly by the Einstein Program (Young Scholar Fellowship Program) of Ministry of Science and Technology (MOST), Taiwan, under Grant MOST 110-2636-E-006-026; and partly by the Higher Education Sprout Project of Ministry of Education (MOE) to the Headquarters of University Advancement at National Cheng Kung University (NCKU). Publisher Copyright: © 2022 IEEE.; 2022 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2022 ; Conference date: 17-07-2022 Through 22-07-2022, United States: Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1756–1759. DOI: 10.1109/IGARSS46834.2022.9883334.

[31] M. Lee and J.-P. Heo, *Noise-free optimization in early training steps for image super-resolution*, 2023. arXiv: 2312.17526 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/2312.17526.

[32] J. Flusser, S. Farokhi, C. Höschl, T. Suk, B. Zitová, and M. Pedone, “Recognition of images degraded by gaussian blur,” *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 790–806, 2016. DOI: 10.1109/TIP.2015.2512108.

[33] J. Bai and L. Shi, “Truncated kernel stochastic gradient descent on spheres,” *arXiv preprint arXiv:2410.01570*, 2024.

[34] L. Prechelt, “Early stopping - but when?” In *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 55–69, ISBN: 978-3-540-49430-0. DOI: 10.1007/3-540-49430-8_3. [Online]. Available: https://doi.org/10.1007/3-540-49430-8_3.

[35] A. Horé and D. Ziou, “Image quality metrics: Psnr vs. ssim,” in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[36] D. R. I. M. Setiadi, “Psnr vs ssim: Imperceptibility quality assessment for image steganography,” *Multimedia Tools and Applications*, vol. 80, no. 6, pp. 8423–8444, 2021.

[37] DigitalOcean, *Super resolution generative adversarial networks*, https://www.digitalocean.com/community/tutorials/super-resolution-generative-adversarial-networks, Accessed: 2024-10-14, 2021.

[38] M. Vasamsetti, P. Kaja, S. Putta, and R. Kumar, "Combining super-resolution gan and dc gan for enhancing medical image generation: A study on improving cnn model performance," in *GANs for Data Augmentation in Healthcare*, A. Solanki and M. Naved, Eds. Cham: Springer International Publishing, 2023, pp. 187–205, ISBN: 978-3-031-43205-7. DOI: 10.1007/978-3-031-43205-7_11. [Online]. Available: https://doi.org/10.1007/978-3-031-43205-7_11.

[39] GeeksforGeeks, *Super resolution gan (srgan)*, https://www.geeksforgeeks.org/super-resolution-gan-srgan/, Accessed: 2024-10-14, 2021.

[40] Y. Yao, Z. Cui, D. Wang, and M. Zhang, "Efrg-srgan: Combining augmented features for real-world super-resolution," *Signal, Image and Video Processing*, pp. 1–15, 2024.

[41] Z. Wei, Y. Huang, Y. Chen, C. Zheng, and J. Gao, "A-esrgan: Training real-world blind super-resolution with attention u-net discriminators," in *Pacific Rim International Conference on Artificial Intelligence*, Springer, 2023, pp. 16–27.

[42] E. Contributors, *Enhanced super resolution generative adversarial networks (esrgan)*, https://esrgan.readthedocs.io/en/latest/pages/esrgan.html, Accessed: 2024-10-14, 2021.

[43] S. Y. Kim, J. Lim, T. Na, and M. Kim, "3dsrnet: Video super-resolution using 3d convolutional neural networks," *arXiv preprint arXiv:1812.09079*, 2018.

[44] F. Manessi and A. Rozza, "Learning combinations of activation functions," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 61–66. DOI: 10.1109/ICPR.2018.8545362.

[45] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *CoRR*, vol. abs/1608.02908, 2016. arXiv: 1608.02908. [Online]. Available: http://arxiv.org/abs/1608.02908.

[46] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," *CoRR*, vol. abs/1707.02921, 2017. arXiv: 1707.02921. [Online]. Available: http://arxiv.org/abs/1707.02921.

[47] A. Jolicoeur-Martineau, "The relativistic discriminator: A key element missing from standard GAN," *CoRR*, vol. abs/1807.00734, 2018. arXiv: 1807.00734. [Online]. Available: http://arxiv.org/abs/1807.00734.