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

1st Istiaq Ahmad
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
istiaqahmadibnul@gmail.com

2nd Labib Sadman Azam
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
sadmanlabib77@gmail.com

3rd Moinul Hossain Bhuiyan
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
adibhossain0077@gmail.com

4th Abdullah Al Mamun
Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
abdullahdurjoy71@gmail.com

5th Dr. Muhammad Iqbal Hossain
Associate Professor, Computer Science & Engineering
BRAC University
Dhaka, Bangladesh
iqbal.hossain@bracu.ac.bd

***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 metric where the MSRGAN has outperformed the basic SRGAN and ESRGAN.**

***Index Terms*—ESRGAN, MSRGAN, SISR, SR, SRGAN**

I. Introduction

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.

A. 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.

B. Research Objective

In this research paper, the purpose was to build an efficient GAN-based model that can 4x upscale a low-resolution image on a customer-grade GPU. The following are the study goals for this paper:

- Explore various combinations and possible changes in the architecture for optimal solutions in different scenarios.
- Experimenting with the proposed model on versatile and single category-based datasets.
- Constructing a customizable GAN model that has unique specifications.
- Consumer-grade GPU friendly for both training and testing.
- Compare the models with multiple evaluation metrics.

II. Related Works

A. SRGAN

SRGAN is a type of GAN architecture that 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 [1]. The SRGAN model consists of a generator and a discriminator. The generator upscales the low-resolution images and the discriminator is a binary classifier that helps to differentiate between high-resolution output generated by the generator and original high-resolution images [2]. 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 [3]. 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. This helps improve the stability and performance of the training set of processes. In summary, SRGAN is a basic GAN architecture tool specially designed for image SR that can recover photo-realistic textures from heavily downsampled images.

B. 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 [4]. 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. 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 [5]. 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. Afterward, the discriminator in ESRGAN is designed to differentiate between the actual HR image and the generated super-resolved image [6]. 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 [7]. In conclusion, ESRGAN outperforms SRGAN by implementing a more sophisticated RRDB architecture, improving perceptual loss functions, and showcasing adaptability while managing various upscaling factors.

C. BSRNET

In the context of image SR, BSRNet might improve the quality of low-resolution contents. However, the actual functioning method of BSRNet in image SR may vary depending on the specific implementation and nature of the content data. The primary concept underlying SR 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. To use BSRNET in the context of image upscaling tasks, much computational data requires [8].

D. CNN

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 as shown in Fig. 1. 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 patterns, 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 [9]. 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 function effective [**?**]. The multidimensional output of the following layer in Convolutional Neural Networks (CNNs) is transformed into a one-dimensional array using a flatten 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 [10] [11].

E. 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

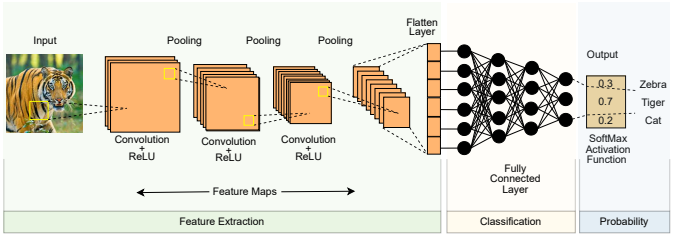

Fig. 1. The working mechanism of CNN.

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 [12]. 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 Fig. 2.

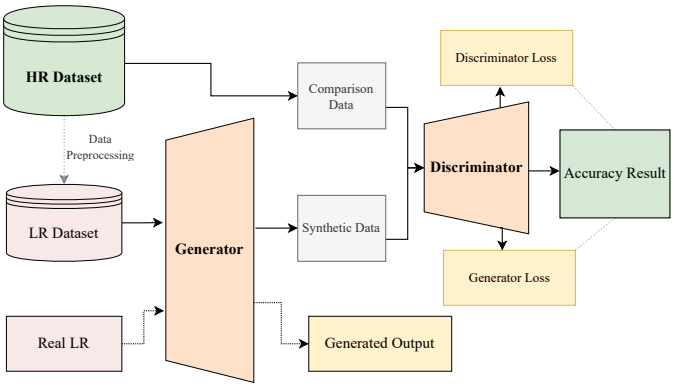

Fig. 2. The working mechanism of GAN.

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 [13]. 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.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1-D(G(z)))] \quad (1)$$

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, $\mathbb{E}_{x\sim Pdata(x)}[logD(x)]$. 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 $\mathbb{E}_{z\sim Pz(z)}[log(1-D(G(z)))]$ 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.

F. Activation function

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 [14]. There are many activation functions in the realm of neural networks. Some of them are described below.

1) 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 [15]. 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:

$$f(x) = max(0, x) \quad (2)$$

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 Fig. 3.

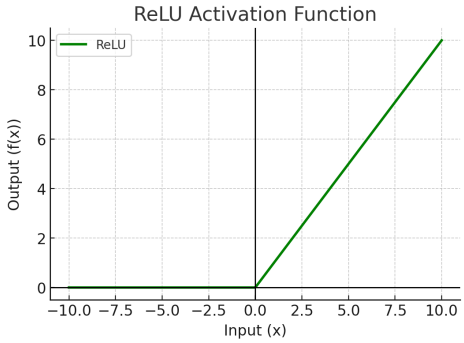

Fig. 3. Graph of a ReLU activation function.

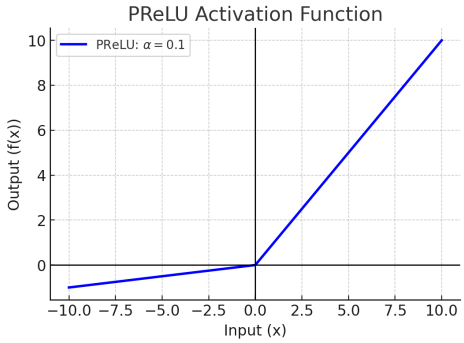

Fig. 4. Graph of a PReLU activation function.

2) 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 [16]

The LReLU is different from the standard ReLU activation function. Theoretically, it is more advanced than the standard ReLU activation function [17]. The standard equation for LReLU is:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{3}$$

Where x stands for the input. The main change in the function is the value of α. α 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 Fig. **??**. It helps prevent neurons from getting stuck.

3) 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 $\alpha = 0.01x$ for negative values whereas the slope in PReLU is a learnable parameter [18]. The equation for PReLU:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{4}$$

In Fig. 4 the slope α is adapted during training. Allowing each neuron to learn the most appropriate slope. This adaptability enhances the model's capacity to capture complex patterns better than ReLU and LReLU [18].

4) 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 [19]. 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. The equation is:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

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.

G. Loss function

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 [20]. 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.

1) MAE loss function: 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 :

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{6}$$

2) MSE loss function: 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 [20]. This method is comparatively straightforward and easy to use, which is why it is widely accepted. The mathematical term would be:

$$l_{MSE}^{SR} = \frac{1}{r^2WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} \left(I_{x,y}^{HR} - G_{\theta_G} \left(I^{LR} \right)_{x,y} \right)^2 \quad (7)$$

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.

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 ϕi,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 [2]. The equation would be:

$$l_{VGG/i,j}^{SR} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y} \right)^2 \quad (8)$$

4) 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) [21]. The formula of the BCE loss is:

$$\text{BCE}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right) \quad (9)$$

H. Evaluation metric

1) 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 [22]. 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:

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

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.

2) SSIM: Then comes the SSIM. It is another metric that is commonly and widely used. 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 [23]. 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:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (11)$$

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

I. 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 [24].

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp \left(-\frac{x^2 + y^2}{2\sigma^2} \right) \quad (12)$$

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

III. Dataset Extraction

A. 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 [25]. 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 Fig. 5. 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.

Again, from the massive dataset of 130000 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.

B. Data splitting

The whole dataset has been splitted into train and test. Here in Fig. 6 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.

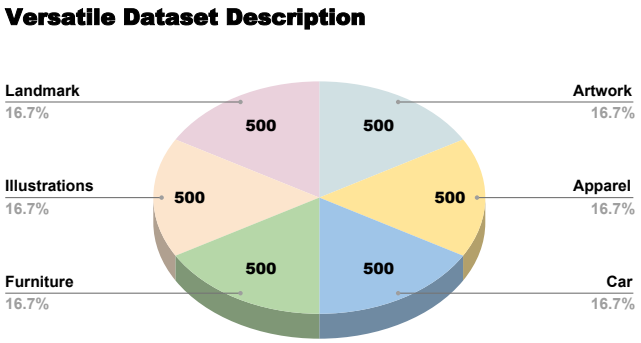

Fig. 5. Versatile category based dataset description.

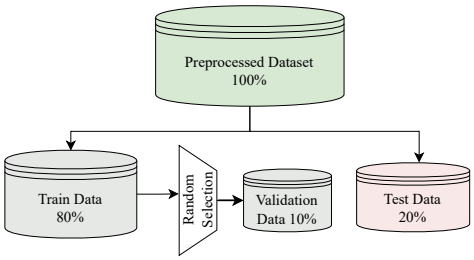

Fig. 6. The process of splitting the dataset.

C. Data pre-processing

1) 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 considered as the ground truth (GT). In order to train the model, a LR set of the data is required. The whole dataset has been downsampled into 4x lower resolution Fig. 7. The result was kept in a different folder from 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.

2) 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. Images are processed using Gaussian noise to mimic real-world uncertainties such as sensor noise and ambient conditions.

IV. Model Implementation

A. Hardware specification

The research focuses on developing a GAN model that can be trained on consumer-grade GPUs. Initially, the study began with the top-performing consumer GPU, the NVIDIA GeForce RTX 4090 (24 GB VRAM), to ensure optimal model development. After developing the model, it was optimized to run on more widely accessible GPUs, such as the NVIDIA GeForce RTX 3060 Ti (8 GB VRAM). Eventually, the model was tested

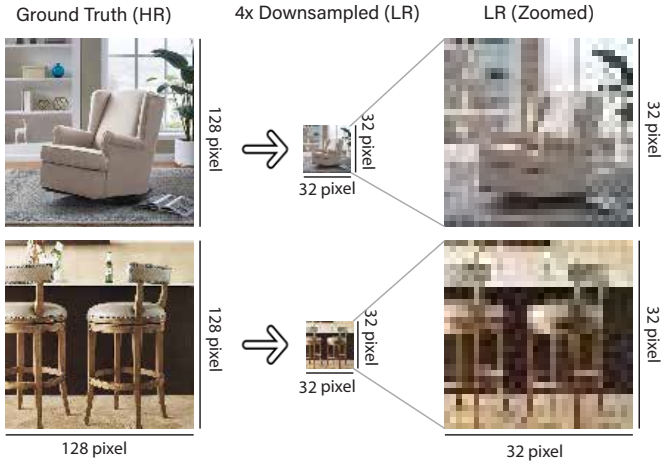

Fig. 7. Down-scaling the dataset to 4x LR.

on the NVIDIA GeForce GTX 1660 (6 GB VRAM), which, while slower, completed the training and testing successfully, fulfilling the research goal of balancing resolution and hardware limitations. Benchmark revealed that the RTX 3060 Ti was used as the baseline with 100% performance, while the RTX 4090 showed 370% performance and the GTX 1660 exhibited 68.3% performance.

B. Proposed Model: MSRGAN

The final goal of implementing the expected model was a modified version of SRGAN. 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 MSRGAN. 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 [26] Fig. 8. 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.

In the Fig. 8 the RRDB combined multilevel residual networks, where residual learning was used at different levels [27]. 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 [28].

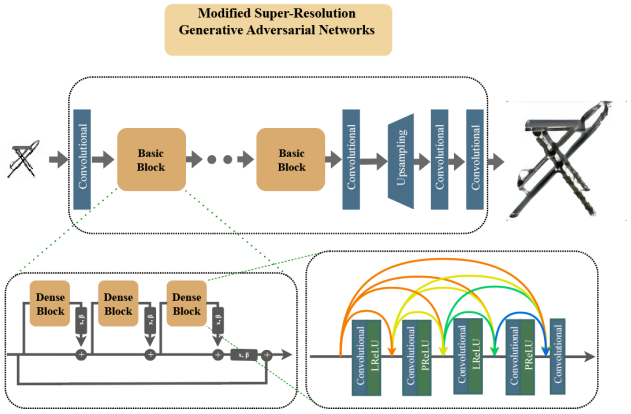

Fig. 8. MSRGAN with Modified RRDB activation functions.

The discriminator also used the combination of LReLU and PReLU which created a more robust and flexible process [26]. 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 [20].

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 [21].

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 [21].

C. SR for higher resolution inputs utilizing MSRGAN

The MSRGAN model, originally designed to upscale 32x32 pixel images to 128x128, was adapted to handle higher-resolution images using a splitting and upscaling technique. For example, a 512x512 pixel image was first downsampled to 128x128 as the LR input. This 128x128 image was then split into sixteen 32x32 pixel sections, which the model individually upscaled 4x to 128x128 pixels. Finally, the upscaled sections were merged back together to form the final 512x512 pixel output. Whole process presented on Fig. 9. This approach allows MSRGAN to upscale images of any resolution by processing smaller parts sequentially.

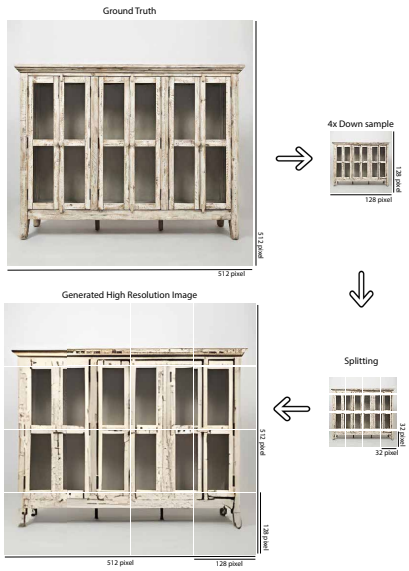

Fig. 9. SR for higher resolution images with MSRGAN.

D. SRGAN and ESRGAN implementation

The SRGAN's architecture of the generator includes five convolutional layers, each followed by ReLU activation. The first layer has 64 feature maps with a 9x9 kernel to extract low-level features, and the subsequent layers maintain 64 feature maps with 3x3 kernels. The final layer outputs 3 feature maps for the color channels. The discriminator consists of eight convolutional layers. Each convolutional layer is followed by ReLU activation. The output is flattened, passed through two fully connected layers, and finally reduced to a single scalar value predicting whether the input is real or fake. Perceptual Loss Function consists of content loss using MSE and adversarial loss using BCE with a sigmoid layer.

The ESRGAN's generator is based on the SRResNet architecture, with the BN layers removed. It uses RRDB instead of basic blocks. Each residual block in RRDB has 5 convolutional layers followed by leaky ReLU activation. The discriminator is based on a relativistic GAN. The RaD discriminator extracts features using a convolutional neural network, followed by fully connected layers that output a probability of authenticity. ESRGAN improves the perceptual loss by applying constraints on features before activation. A fine-tuned VGG19 network for material recognition is used to compute this loss.

V. Result Analysis

A. Inspecting loss curves

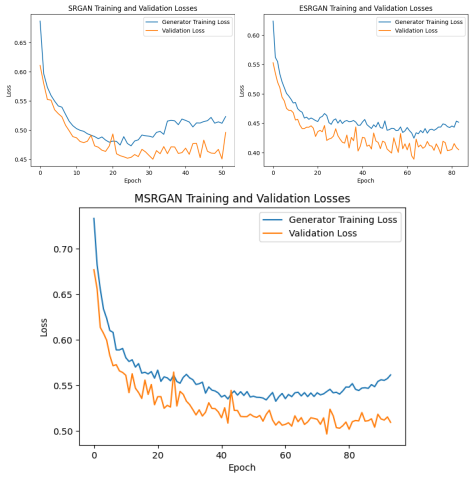

Fig. 10. Train and Validation curves on versatile dataset.

1) Result on versatile category dataset: The training curves in Fig. 10) 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.

2) Result 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 Fig. 11. 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

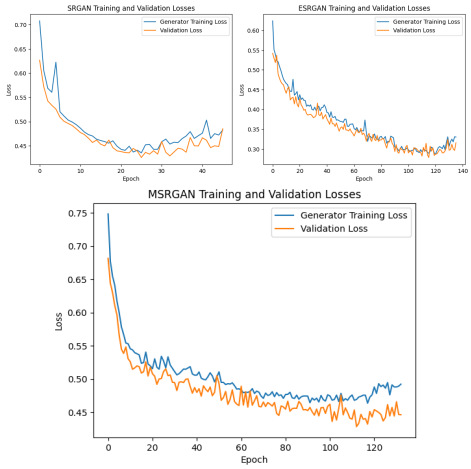

Fig. 11. Train and Validation curves on single category dataset.

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.

B. Evaluation Metric Result

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 Fig. 12 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 in Fig. 12) 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.

The metrics in Fig. 13 shows the fluctuated values of SSIM and PSNR for images from the test dataset. SRGAN generated an imbalance result with an average PSNR of 17.84. Then the model ESRGAN had a slightly better performance 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

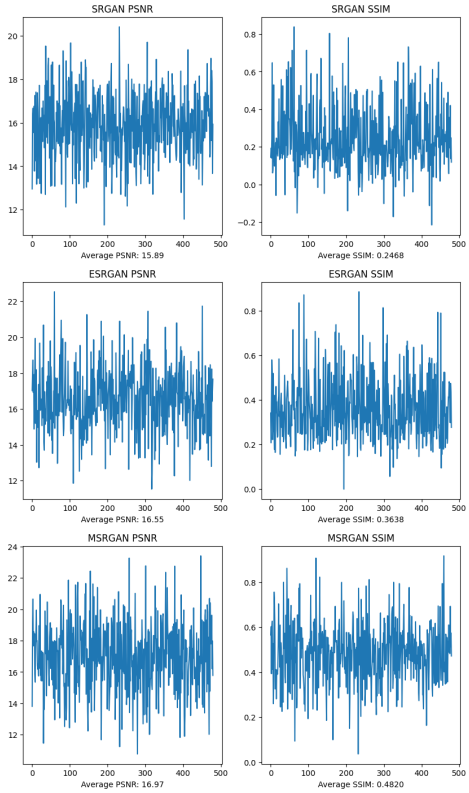

Fig. 12. PSNR and SSIM result on versatilele category dataset.

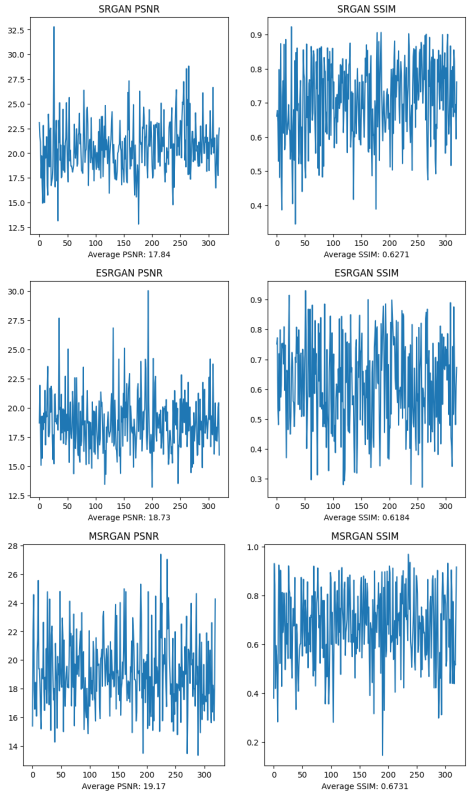

Fig. 13. PSNR and SSIM result on single category dataset.

results were really good where MSRGAN performed better in terms of both SSIM and PSNR metrics.

C. Generated image visual comparison

The result of the versatile category test dataset for all the models performed below average. For an example in the Fig. 14 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.

For single category some of the generated images were shown in the Fig. 15) to visualize the dissimilarity and variations between the models output. 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.

VI. Conclusion

A. Future works

This model we have developed has potential, but it needs some refining at this point. The paper suggests some pointers to improve the model, which would help it achieve far higher heights.

- Incorporate attention mechanisms to focus on certain portions of the image to use the model for a specific purpose.

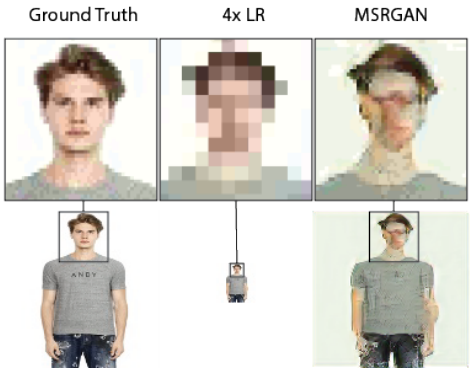

Fig. 14. Versatile category super resolution result example.

- Explore extending MSRGAN to multiscale super-resolution for more efficient and flexible upscaling.
- Use knowledge distillation to transfer insights from a larger pre-trained model to MSRGAN, cutting training time and processing needs.

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

B. Conclusion

In this thesis, a custom GAN model named MSRGAN was developed and evaluated against the established models

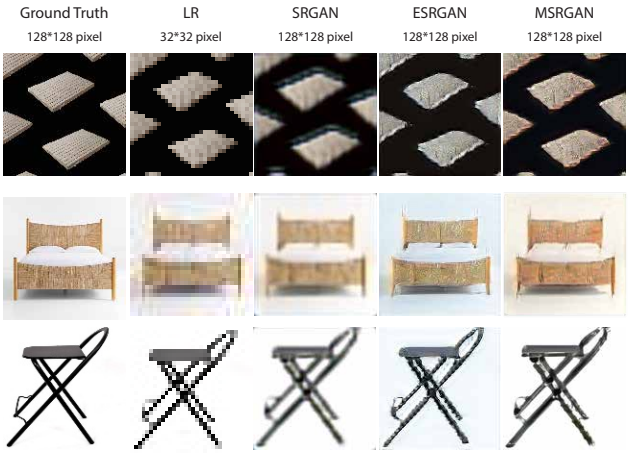

Fig. 15. Visualization of SRGAN, ESRGAN and MSRGAN results.

for image super-resolution under the condition of customer-grade GPUs. Compared to SRGAN and ESRGAN models the MSRGAN performed better in the context of image super-resolution. In this research, one of the major objectives was to develop a super-resolution model that is capable of running on a consumer-grade GPU. This model is not computationally 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 heavy computational power so a person with a consumer-grade GPU can also run to train and test the model. With the freedom of training their own dataset and implementing it. In conclusion, the MSRGAN model is a very capable image super-resolution model where it can surpass the basic SRGAN and ESRAN models. Thus, this model may serve as a stepping stone for the development in the field of image super-resolution.

REFERENCES

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

[2] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[3] 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*, pp. 187–205. Cham: Springer International Publishing, 2023.

[4] 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*, pp. 162–181, Springer, 2022.

[5] 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.

[6] 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*, pp. 16–27, Springer, 2023.

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

[8] 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.

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

[10] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[11] 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)*, pp. 401–404, 2015.

[12] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang, and B. Guo, "Styleswin: Transformer-based gan for high-resolution image generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11304–11314, June 2022.

[13] 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.

[14] T. Szandała, *Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks*, pp. 203–224. Singapore: Springer Singapore, 2021.

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

[16] Y. Guo, S. Li, and G. Lerman, "The effect of leaky relus on the training and generalization of overparameterized networks," 2024.

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

[18] 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, p. e621, 2021.

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

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

[21] 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.

[22] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369, 2010.

[23] 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.

[24] 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.

[25] R. Singh, "Google universal image embeddings (128x128)," 2020. Accessed: 2024-10-14.

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

[27] 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.

[28] 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.