Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Evaluating Neural Super Resolution Assisted by Depth Estimation

Author:
George Soteriou

Supervisor:
Dr. Amir Alansary

Second Marker:
Dr. Benjamin Hou

June 13, 2021

Abstract

The challenging problem of Super Resolution attempts to obtain a high resolution image output from its low resolution counterpart. This ill-posed problem has been a dream of most researchers up to recently with the introduction of Deep Learning. At the same time, self supervised learning approaches have made breakthroughs in monocular Depth Estimation. New networks now make it possible to predict the relative distance of an item using a single image.

In this work, the effect of adding Depth information to Super Resolution for the creation of photo realistic outputs has been evaluated. A new model, ESRGAN_D, has been proposed for this task following from the latest Generative Adversarial Network (GAN) architectures. Given the complexity of this model, hardware required to train it has been explored. In addition, given the subjective goal of photo-realism, the usefulness of existing PSNR and SSIM metrics have been discussed. Finally benchmarking of the proposed model is presented, showing improvements is depth heavy scenes, compared to previous state-of-the-art models.

Acknowledgements

I would like to thank my supervisor **Dr. Amir Alansary** for supporting me every week with new ideas and inspiration, and my second marker **Dr. Benjamin Hou** for providing great feedback for my interim report.

I would also like to thank my parents **Anthi** and **Nikos** and sister **Eleonora**, for always being there and caring for me on every step of my life. I would not be here without you!

Finally I would like to thank **Georgia** for her unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Super Resolution (SR) has been a topic of great interest for numerous researchers in both academia and industry over the past years. Companies like Nvidia, Facebook and Adobe have reported record-breaking revenue and advances in this field, by introducing methods like DLSS[1], SIGGRAPH[8] or plain Super Resolution[2] respectively. The idea of enhancing an image has been around for so long that even shows like CSI referenced it (Figure 1.1), but it is not as simple as it seems.

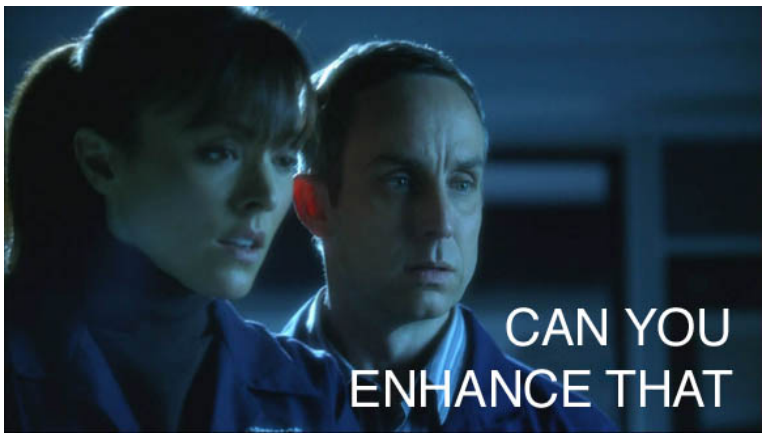


Figure 1.1: CSI Enhance[3]

SR is useful in so many fields. These include medical imaging[9], satellite imaging[10], surveillance[11] or more recently movie rendering[12], gaming citeXiao2020 or even potentially decreasing network usage when streaming videos or games[13]. Recently a large part of SR, specifically gaming and video, are focusing on scenes based on realistic looking environments. In most of these environments, depth is quite important as it can help the model understand a scene better as seen in Figure 1.2. This observation will be explored further in this project.

[1] https://tinyurl.com/nvidea-news-2021
[2] https://tinyurl.com/adobe-news-2021
[3] https://www.youtube.com/watch?v=I_8ZH1Ggjk0

As shown above the task of SR has been applied in many different ways. The most common usage is Single Image Super Resolution (SISR) which has been a low level vision challenge since the existence of digital images. From the first time machine learning was applied to Super Resolution in SRCNN by Dong et al.[14], many deeper methods have worked to improve this task. Deeper models can extract and enhance features in images effectively[15] whilst the most recent models use extra information to improve their predictions. For example, multi image SR or more specifically video SR use previous and next frames to help enhance the current frame[8].

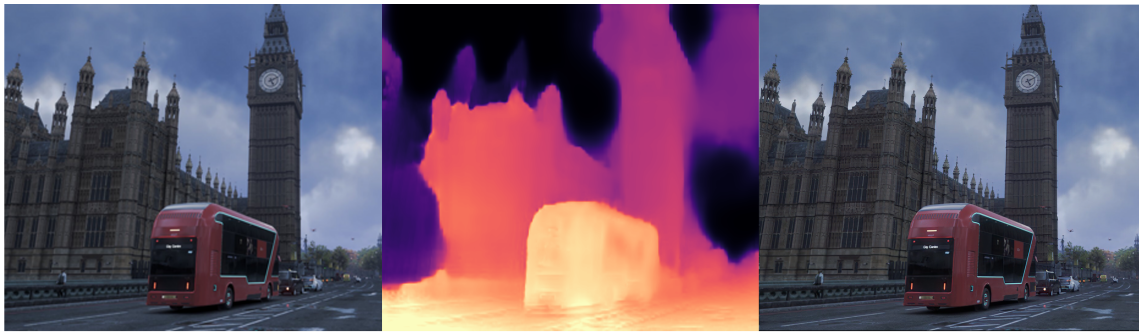


Figure 1.2: Example of a scene with variable depth enhanced with the model proposed

1.2 Objectives

Following previous observations, with the use of extra information about a picture, attempts will be made to enhance it further. In this project the extra information to be used is an estimate of the depth information of an image. As will be discussed in Chapter 2, Neural Networks are able to bring prior information into a problem and can therefore solve problems that could not be solved without them. A depth estimation model will be used to extract predicted depth information. This will then be used in addition to the image itself when applying Super Resolution.

A breakdown of the projects contributions are as follows:

- **Implement state of the art Depth Estimation models.** By looking at existing research in this field, state-of-the-art models are explored, understood and implemented to help extract depth maps from a RGB picture.
- **Develop and compare models for SR with the added depth maps.** By understanding the inner workings of SR, in Convolutional Neural Network, Residual Networks and Generative Adversarial Networks, existing models are implemented and them modified to accommodate for the extra depth information provided. A new ESRGAN_D has been proposed as a modification of ESRGAN.
- **Find suitable dataset for training these models.** Given the multi-modal nature of these new networks, training datasets have been picked to accommodate the texture and depth variations required to train them.

- **Evaluate the new models and compare with existing image-only SR models.** Evaluating the quality of a picture is a hard task in itself. This has been explored and explained in this project. In addition to that, a comparison has been made between the proposed ESRGAN_D and the original ESRGAN.

1.3 Challenges

This project proved to be more challenging than anticipated. The main issues that emerged during the project include:

- **Suitable Hardware availability.** Super Resolution and depth estimation are both highly memory demanding when using larger models and images. Most state-of-the-art models are trained on GPUs with at least 16 G of RAM and even then, for this use-case, GPUs are very slow for training, requiring hours or even days. Given the current GPU shortage in 2021, available hardware was limited and GPUs provided online via platforms like Google Colab[4] were not sufficient to train the model in a timely manner. This problem was overcome using Google's Cloud TPUs [5] as is discussed in Section 4.1.2.

- **Evaluation methods**. Knowing if an image "looks good" is undoubtedly subjective. When processing a $\times 4$ enlargement of an image, it is almost impossible to predict the exact original image thus many models focus on "realistic" image SR[5]. This however is much harder to measure as explained in Chapter 5.

[4] https://colab.research.google.com/
[5] https://cloud.google.com/tpu

Chapter 2

Background

Super Resolution (SR) has been a hot topic of research for the last decade[16, 17, 5]. Depth Estimation (DE), on the other hand, has been a recently emerging area of interest[18, 7] with the introduction of stereo camera systems for training. In this chapter, a literature review of recently published work that covers both SR and DE will be presented.

2.1 Single Image Super Resolution

At its core, Single Image Super Resolution (SISR) consists of taking a low resolution image of a scene and converting it to a high resolution image. As can be seen from Figure 2.1 the fundamental concepts underpinning SISR include getting a picture with a smaller pixel count, spreading the pixels out, and using an algorithm to fill in the gaps.

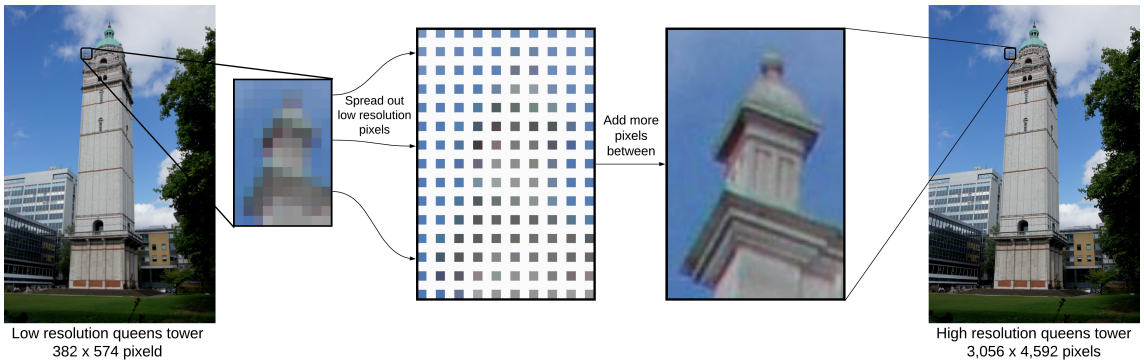


Figure 2.1: Basic premise for Super Resolution[1]

Principal approaches like Bicubic[19] or Lanczos[20] filtering have existed for many years. Despite these approaches being quite fast, the results are usually unremarkable. This is due to their simple algorithmic approach. As shown in Figure 2.2 the

[1]Image adapted under licensing with the Creative Commons Attribution 2.0 Generic license.

simplest algorithm, Nearest Neighbour, picks the closest colour around it. In contrast, Bilenar and Bicubic interpolate the space as shown in the diagram to choose the appropriate colours for the intermediate pixels. This tends to create very smooth textures and does not work well with sharp edges. Furthermore, solutions based on edge detection have also been tested in the past[21] with only slight improvements.

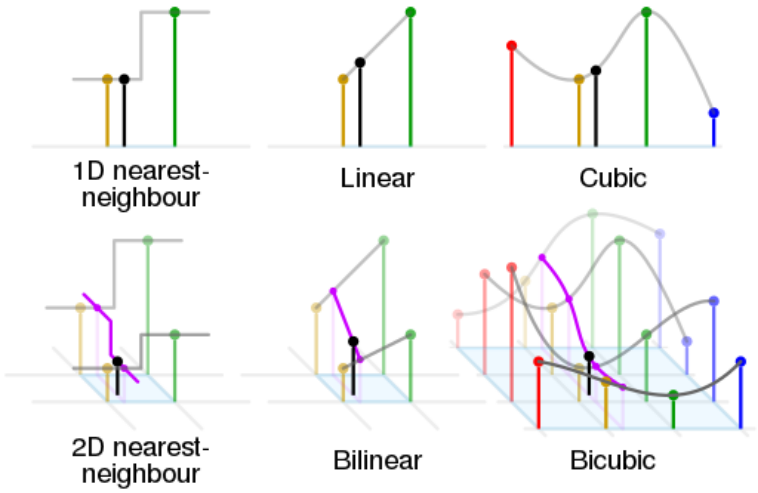


Figure 2.2: Comparison between Nearest neighbour, Bilinear and Bicubic interpolation[2]

In contrast to the classic algorithms, recent advances in the field have proposed complex mappings between the Low Resolution (LR) and High Resolution (HR) images using machine learning and training data to find an optimal function given a dataset. Deep learning approaches have quickly transformed this field by providing great breakthroughs in the quality of the images created.

Using quantitative metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) an attempt to correlate between the image quality and human perception can be made. Further details of these functions and how they are used will be explained in Chapter 5: Evaluation.

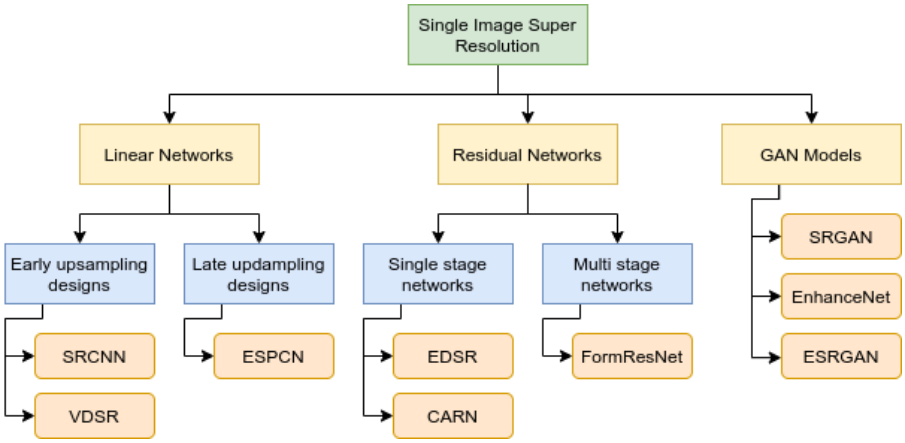


Figure 2.3: Well known Single image Super Resolution networks and models.

[2]Image under licensing with the Creative Commons Attribution-Share Alike 4.0 International

There are many approaches for SR[22], however, the models of interest for this project fall into 3 main categories as shown in Figure 2.3: Linear Networks, Residual Networks and Generative Adversarial Networks (GANs). Transformers have not been considered here as they do not integrate well with the existing Depth Estimation models presented later. This will be discussed further in Section 6.2.

2.1.1 Linear Networks

Linear Networks are simple in structure and consist of a single path without branches or skip connections. These models are split into two categories, the early upsampling and late upsampling designs. Early upsampling models upscale the image to match the expected output size using algorithms such as Bicubic interpolation, before passing it through the network. On the other hand, models with late upsampling, pass the low resolution image directly into the network and have layers close to the end of the model specifically for upsampling. The latter method drastically decreases the size of the models thus making them faster and less computationally expensive as explained in later models.

Super Resolution Convolutional Neural Network (SRCNN)

One of the first proposed models, SRCNN[17], shows that SR with machine learning is possible. This was an inspiration to all researchers. The model has a very simple design and follows the early upsampling method. It consists of just 3 convolution layers and 2 ReLU activation layers as seen in Figure 2.4.

Despite the simplicity of this model, it had rather significant results at the time, motivating the community to move forward with learning approaches. The CNN then uses 32×32 image patches to allow for faster training and Mean Squared Error (MSE) loss function to minimize the difference between the output and the expected HR image.

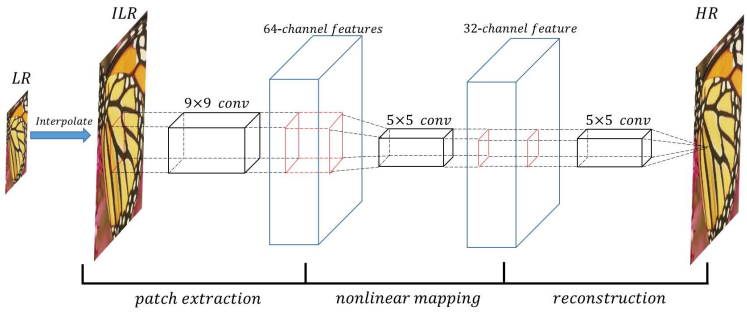


Figure 2.4: Sketch of the SRCNN architecture[1]

With the insight of current research and papers in deep learning[15], some drawbacks that are now understood about SRCNN include the unnecessarily large 9×9 and 5×5 kernels and the lack of depth in the model. With modern hardware, smaller kernels and deeper models, huge improvements can be observed.

Very Deep Super Resolution (VDSR)

An example of this improvement is observed in the VDSR model[2] that also utilises early upsampling. By employing smaller convolutions (3×3) to avoid slowdowns and by adding many more layers, this model achieves some considerable improvements over SRCNN.

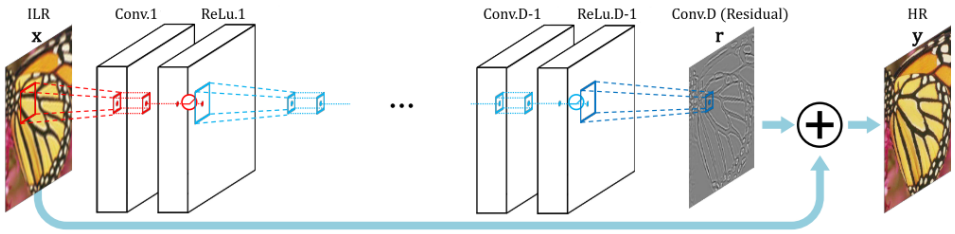


Figure 2.5: Sketch of the VDSR architecture[2]

Even though VDSR has a single skip connection, this network is mostly linear as can be seen in Figure 2.5. The maximum depth tested with this model had 20 layers. Despite these additional layers, subsequent models illustrated in future sections, will be considerably deeper and hence require more residual skip connections to amount to further improvements.

Efficient Sub Pixel Convolutional Neural Network (ESPCN)

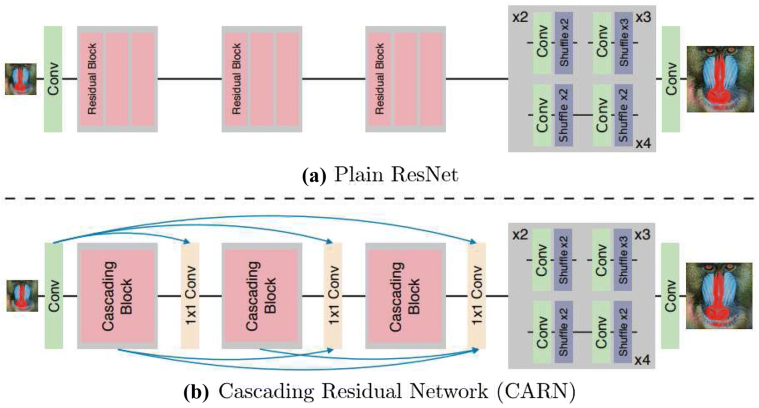
Before moving on to deeper models, an additional important step was made in the ESPCN model[23]. This model uses a late upsampling method, which allows the models to remain small and efficient until the last layers of the model. Because of this improvement, the model is able to achieve results equal to or better than SRCNN at only a fraction of the time. It has also allowed the model to be used for real-time upscaling of HR videos on a single GPU.

This breakthrough in computational complexity decrease, has allowed many deeper models to be trained in reasonable time to produce amazing results, even in real time. With this enhancement, as well as the work done with VDSR, it is now possible to proceed to the next stages of SR.

2.1.2 Residual Networks

Unlike Linear Networks, Residual Networks use newer deep learning techniques, such as repeated block architecture with skip connections to overcome gradient vanishing. The two subcategories for Residual Networks are single-stage and multi-stage nets. Single stage nets like CARN[3] and EDSR[4] use modified versions of ResNet[24] (Figure 2.6) and have achieved notable results. Unlike single stage nets, multi stage nets consist of multiple networks that are trained one after another and they help each other create the most optimal outcomes.

Cascading Residual Network (CARN)



(a) Plain ResNet

(b) Cascading Residual Network (CARN)

Figure 2.6: ResNet compared to CARN architecture[3]

In the case of the CARN model[3] the use of ResNet[24] blocks as both local and global modules is observed. This allows for more information to reach the subsequent layers of the network giving better predictions as the gradient is not lost in back propagation. The model and all the skip connections are shown in Figure 2.6. CARN-M is a variant of this model that follows the results of EDSR (explained below) and replaces the ResNet blocks with the more efficient blocks proposed, to produce a more light-weight SR model with comparable results.

Enhanced Deep Super Resolution (EDSR)

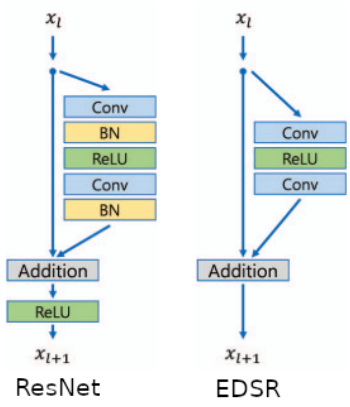


Figure 2.7: ResNet compared to EDSR block architecture[4]

The EDSR model[4] optimises ResNet[24] blocks. It removes the Batch normalization layers and some ReLU layers that are not required for the task of SR as seen in figure 2.7. As described in Nah et al.[25], batch normalization layers only reduce the range flexibility of a network, therefore it is better to remove them in order to increase image sharpness and performance. Furthermore, this decreases the computational complexity of the models allowing for up to 40% less memory usage during training. Once again this allows us to follow the previous pattern and create even larger models with further improved results as will be seen in ESRGAN below.

Form Residual Network (FormResNet)

To conclude Resudual Networks, FormResNet[26] is a multi-stage net. It is formed by two networks. The first network removes high frequency corruption in uniform areas while the second network adds structure to regions with sharp lines. The fascinating aspect of this type of training is that each network uses different types of loss functions resulting in each network being specialised to its task.

2.1.3 GAN Models

Generative Adversarial Networks, like SRGAN[6], use an approach where two networks, the generator and the discriminator, try to fool each other into producing better results. In this case, the generator, given a low resolution image, creates a SR version of that image while the discriminator is tasked to distinguish between the real HR image and the created SR image.

Super Resolution Generative Adversarial Network (SRGAN)

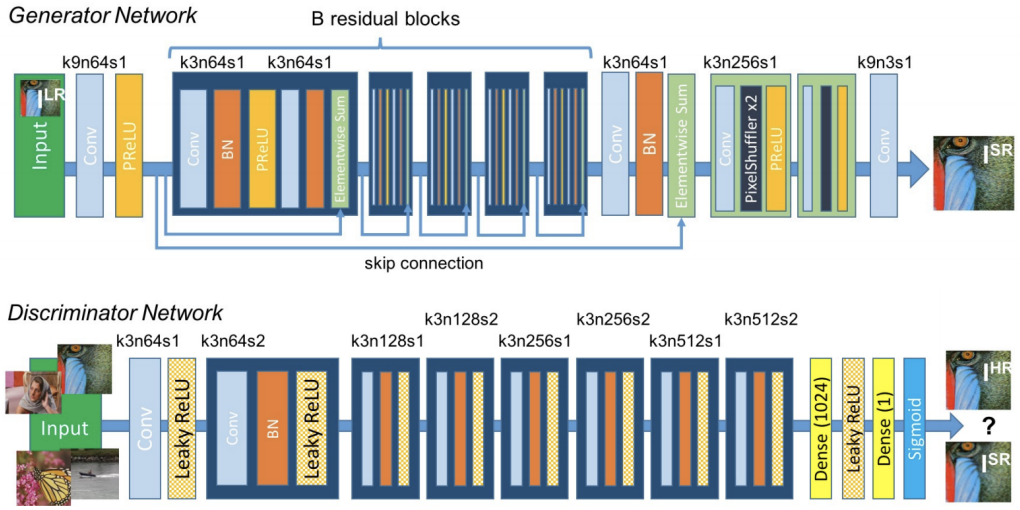


Figure 2.8: SRGAN architecture[5]

Ledig et al.[5] released one of the most important papers in SR in recent research. This publication contains SRResNet, a very popular net based on a modified ResNet[24] for SR, as well as SRGAN. They also introduced a new perceptual loss function while training.

SRGAN consists of SRResNet as the Generator and a VGG inspired discriminator network as seen in Figure 2.8. This network produces very realistic results that broke all records at the time.

The new loss function used in training consists of the content loss component, which is the Mean Squared Error (MSE), used by most previous SR models[17][23], but

is slightly modified as seen in equation 2.1. It also contains an adversarial loss component as is evident from equation 2.2.

The content loss uses the feature maps of a pre-trained 19-layer VGG net as described in Simonyan and Zisserman[27]. In the equation 2.1, $\Phi_{i,j}$ indicates the j^{th} convolution (after activation) before the i^{th} maxpooling layer within the VGG19 network. In addition, $I^{SR} = G(I^{LR})$, where G is the Generator and I^{LR} and I^{HR} are the low and high resolution images respectively, while $W_{i,j}$ and $H_{i,j}$ indicate the dimensions of the specific feature map.

$$l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{x=1}^{H_{i,j}} (\Phi_{i,j}(I^{HR})_{x,y} - \Phi_{i,j}(I^{SR})_{x,y})^2 \tag{2.1}$$

The adversarial loss, 2.2, encourages the GAN to keep solutions in the realm of natural images by trying to fool the discriminator network. In this equation the symbols above apply, as well as $D(x) = \sigma(C(x))$ where σ is the sigmoid function and $C(x)$ is the discriminator output.

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -log(D(I^{SR})) \tag{2.2}$$

Together these losses are combined so as to produce the overall loss of the network:

$$l^{SR} = l^{SR}_{VGG} + l^{SR}_{Gen} \tag{2.3}$$

Enhanced Super Resolution Generative Adversarial Network (ESRGAN)

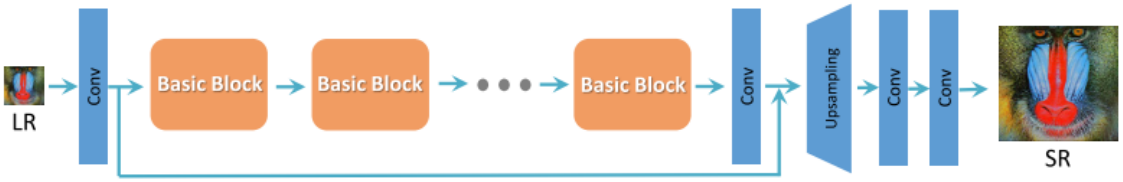


Figure 2.9: ESRGAN generator model[6]

Finally, the current state-of-the-art and the focus of this report, is the ESRGAN model[6]. It improves on SRGAN by removing many unnecessary layers like batch normalisation, whilst adding a few dense blocks and global connections. These changes are influenced by the EDSR and CARN models described in section 2.1.2. As before, this allows the model to learn faster and due to the decreased memory usage and more skip connections, we can make the model deeper still and and allow the use of larger patch sizes like 128×128 while training.

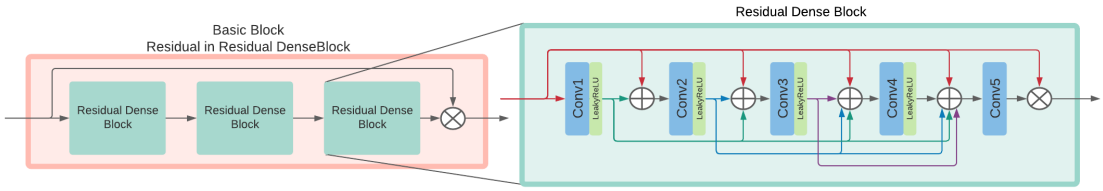


Figure 2.10: The contents of an ESRGAN Basic Block: Residual in Residual Dense Block (RRDB)

By replacing the Basic Blocks of the network with Residual in Residual Dense Block (RRDB) as seen in Figure 2.10, ESRGAN allows for a much deeper network with decreased effects of vanishing gradients. The full generator network can be seen in Figure 2.9 and has been tested in both 16 and 23 basic blocks length.

$$D\left(I^{HR}\right) = \sigma\left(C\left(I^{HR}\right)\right) \rightarrow 1 \text{ Real}$$
$$D\left(I^{SR}\right) = \sigma\left(C\left(I^{SR}\right)\right) \rightarrow 0 \text{ Fake}$$

a) Standard GAN

$$D_{Ra}\left(I^{HR}, I^{SR}\right) = \sigma\left(C\left(I^{HR}\right)\right) - \mathbb{E}\left[C\left(I^{SR}\right)\right] \rightarrow 1 \text{ More realistic than fake data}$$
$$D_{Ra}\left(I^{SR}, I^{HR}\right) = \sigma\left(C\left(I^{SR}\right)\right) - \mathbb{E}\left[C\left(I^{HR}\right)\right] \rightarrow 0 \text{ Less realistic than real data}$$

b) Relativistic GAN

Figure 2.11: Difference between standard discriminator used in SRGAN and relativistic discriminator used in ESRGAN

Wang et al. also made significant improvements to the discriminator network by following the Relativistic average Discriminator RaD[3] as shown in Figure 2.11. The symbols used here, include I^{HR} as the original real image, I^{SR} as the image generated when passing the low resolution I^{LR} image through the Generator, σ indicating the sigmoid function, $C(.)$ indicating the discriminator output and $\mathbb{E}[.]$ as the mean operation.

The original discriminator proposed with SRGAN had a direct real or fake output as shown in 2.11(a). Alternatively, RaD uses the average of the discriminator output to predict whether results are more realistic than the given fake data or less realistic than the given real data. This then defines the adversarial loss of both the generator and discriminator in equations 2.4 and 2.5 respectively.

$$l_G^{Ra} = -\mathbb{E}[\log(1 - D_{R_a}(I^{HR}, I^{SR}))] - \mathbb{E}_{I^{SR}}[log(D_{R_a}(I^{SR}, I^{HR}))] \tag{2.4}$$

$$l_D^{Ra} = -\mathbb{E}[\log(D_{R_a}(I^{HR}, I^{SR}))] - \mathbb{E}_{I^{SR}}[log(1 - D_{R_a}(I^{SR}, I^{HR}))] \tag{2.5}$$

Following the above change, this paper also innovated by updating the content loss $l_{VGG/i,j}^{SR}$ previously defined in SRGAN[5]. They are using the same VGG19 network but instead of using the activated features, they use the features before activation. This increases performance as the activations used previously where sparse and so provided weak supervision as well as inconsistent brightness.

The total loss of the network is then a weighted sum of the above:

$$l_G = l^{SR}_{VGG/i,j} + \lambda l^{Ra}_G + \eta L_1 \tag{2.6}$$

This network provides both sharper edges than all previous models and also better SR textures. An adaptation to this model is discussed in Section 3.2 and used to archive the goals of this project.

2.2 Depth Estimation

Depth Estimation from single images has always been a challenging task. Most approaches in the past have used supervised learning to achieve moderate results but recently as shown in Godard et al.[18, 28], significant improvements using self-supervised learning were achieved. It is also possible to further improve on these approaches as shown by Lyu et al. in HR-Depth[7] by implementing deeper and more connected networks as well as generating depth maps at different scale factors.

2.2.1 Self-Supervised Monocular Depth Estimation (Monodepth and Monodepth2)

Monocular Depth Estimation is the task of extracting the depth map from a single image. The usual issue with Monocular Depth Estimation is the lack of available training data, as extracting depth from a scene requires expensive hardware such as lasers. Even employing this costly equipment may not necessarily provide accurate data in the case of outdoors scenery. To solve this, researchers have used stereo cameras. One example dataset is the Kitty driving dataset[29] comprised of stereo images. With stereo images it is possible to calculate depth as shown in Figure 2.12.

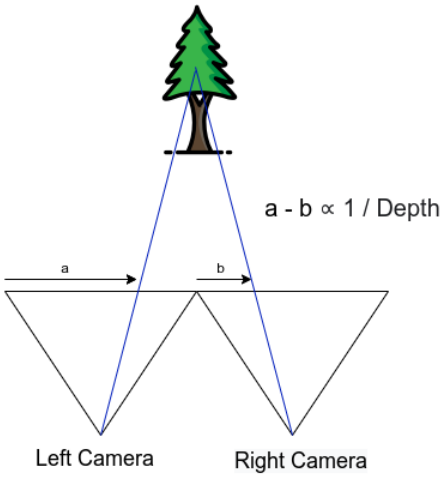


Figure 2.12: Explanation of how stereo is used to calculate depth.

To understand how these models work, a definition of a disparity map is required. A disparity map represents the pixel-wise difference between two stereo images.

Disparity maps are the real outputs of both monodepth models. In Figure 2.13 the right disparity map d^r can be applied with the left input image I^l producing the predicted right image $\tilde{I}^r$. The opposite applies to the right image I^r when applied with the left disparity map d^l, producing the predicted left image $\tilde{I}^l$. Both $\tilde{I}^r$ and $\tilde{I}^l$ can now be compared with I^r and I^l respectively to calculate the loss. It is also then trivial to compute the depth map from a disparity map.

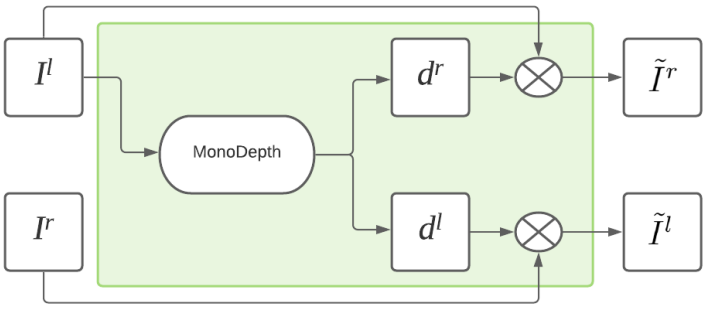


Figure 2.13: Explanation of how monodepth is trained.

This approach makes the training of these models significantly easier as they no longer need disparity or depth ground truth. This type of training is also called self supervised, as the output of the model is not immediately used to predict the loss.

2.2.2 High Resolution Self-Supervised Monocular Depth Estimation (HR-Depth)

In the HR-Depth[7] paper, Lyu expands on Monodepth in multiple ways. A deeper encode/decode model that can output multiple scales of disparity images was used. At the same time, an updated loss function was created for use during training.

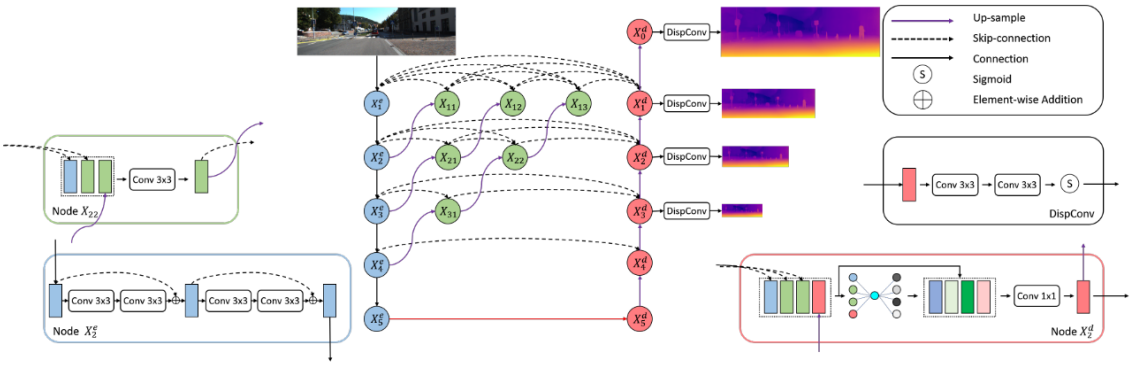


Figure 2.14: HR Depth Model[7]

As seen in Figure 2.14 the model has multiple nodes, each with a different function. Nodes labeled X_i^e are used for feature extraction and are generally residual blocks. Nodes labeled $X_{i,j}$ are used for feature fusion and only have 3x3 convolutions. Finally nodes labeled X_i^d are the fusion Squeeze-Excitation (fSE) blocks proposed in this paper using inspiration from U-Networks.

Additionally this network uses reprojection error $r(I, \tilde{I})$ as described in equation 2.7 where SSIM is Structural Similarity, described later, in Chapter 5, equation 5.7.

$$r(I, \tilde{I}) = \frac{\alpha}{2}(1 - SSIM(I, \tilde{I})) + (1 - \alpha) \left\| I - \tilde{I} \right\|_1 \tag{2.7}$$

This is then applied as a loss L_{re} shown in equation 2.8:

$$L_{re} = \min_{\tilde{I}} r(I, \tilde{I}) \tag{2.8}$$

Furthermore, in order to regularize the disparities, edge aware smoothing loss was used presented in equation 2.9. Here δ_x and δ_y refer to the partial differentiation of depth and RGB images, and the exponential operation of a matrix is a pixel-wise operation.

$$L_{smooth} = |\delta_x D_t| e^{|-\delta_x I|} + |\delta_y D_t| e^{|-\delta_y I|} \tag{2.9}$$

The final loss is, therefore, shown in 2.10 where s refers to the amount of different output scales and λ the weight of the smooth loss.

$$L_{final} = \frac{1}{s} \sum_{i}^{s} (L_{re}^{i} + \lambda L_{smooth}^{i}) \tag{2.10}$$

The HR-Depth model produces results that surpass all previous research. As explained in Chapter 3, this model will be used to extract the depth maps required for the project.

Chapter 3

Network Architecture

As described in the project objectives, state-of-the-art SR and DE models have been introduced above. Following the understanding of existing literature in Chapter 2, choices of models to be used for this project have been made. In this chapter, these choices, and their implementations will be described, as well as any modifications.

3.1 Depth Estimation Model

As the aim of this project is to use depth information to enhance SR, there was no requirement to build a new model for DE. Despite the potential for a new end-to-end DE-SR model, many unsolved difficulties would arise. These difficulties would be mostly based on the different types of training, self-supervised and supervised respectively. As such, this idea will be further addressed in Section 6.2: Future Work, considering possible workarounds for the difference in training methods. The remainder of this section explains how a pretrained model was utilised, and justifies why it was chosen.

Figure 3.1: An example of HRDepth output [7]

The network architecture selected, follows the implementation described by Lyu et al.[7], HR-Depth shown in Section 2.2.2. This model was chosen because of the unprecedented results it obtained. As pretrained models[1] were available for this architecture, a pre-processing computation was executed on all of the training and testing data to create the low resolution depth map of each picture. The maps created were single channel but for the reader's convenience, an RGB mapping has been used as shown in Figure 3.1.

[1] https://github.com/shawLyu/HR-Depth#pretrained-model

3.2 Super Resolution Models

In order to effectively make use of this new depth information, multiple ways of adapting the current state-of-the-art SR models were tested.

Initially, tests were conducted using SRCNN with an extra input layer. Changing SRCNN from 3 channels to 4, it was realised that the simplicity of the model could not learn much by only adding a depth map input without further editing the network.

A similar approach was used with SRGAN where the input number of layers was changed from 3 to 4 to allow for 3 RBG + 1 Depth channel. The only significant difference observed with this method was the increased smoothness of textures. This was caused by the fact that depth is a smooth map, resulting in this undesirable effect.

Finally, with inspiration from the evolution of SR over the years, it was decided to use the currently best GAN model for the task, ESRGAN [6]. Following from the conclusions of the prior models, a single convolution with the depth map was not sufficient to extract the information needed. Therefore the ESRGAN model was mc

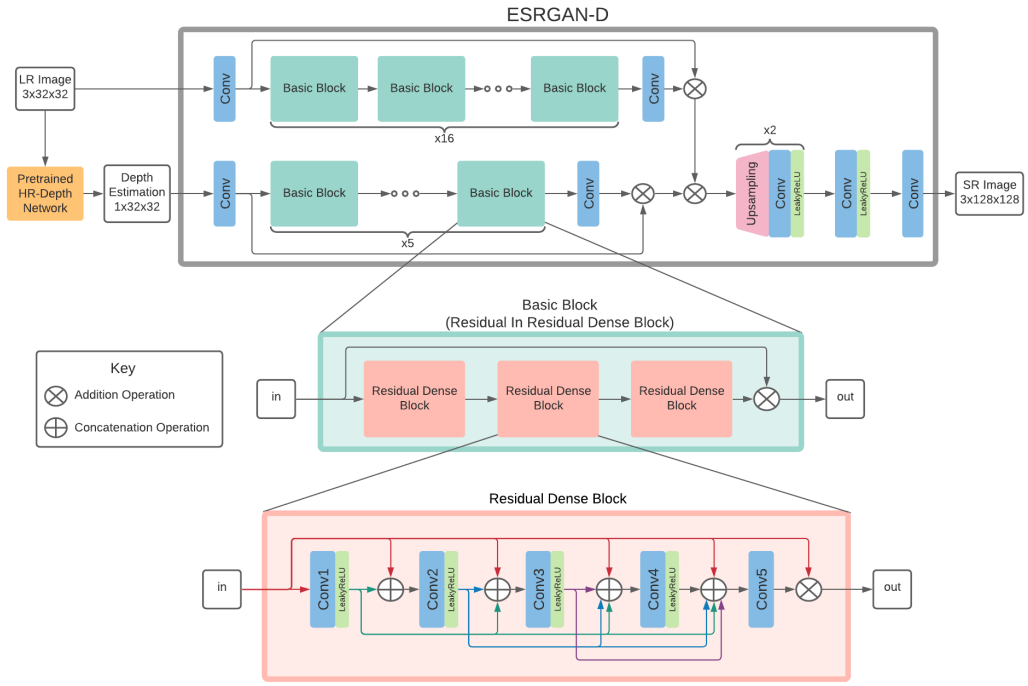


Figure 3.2: Proposed ESRGAN_D Generator Model

A new ESRGAN_D generator model was therefore created and proposed as depicted in Figure 3.2, to fill this requirement. Both the Low Resolution image and the depth map enter the network in parallel and each pass through separate feature extraction layers. They are, then, combined together before upsampling the image to its $\times 4$ larger size. The same Basic Blocks and Residual Dense Blocks are used from the ESRGAN paper. While the 16/23 repeating basic blocks are kept for the RGB path

of the network, only 5/6 layers are used for the depth path as the depth map is only a single layer rather than 3.

This generator model is then combined with the discriminator introduced in SRGAN/ESRGAN [5, 6] and described in Section 2.1.3.

$$l_G = l^{SR}_{VGG/i,j} + 0.001 \times l^{Ra}_G + 0.01 \times L_1 \tag{3.1}$$

$$l_D = l^{Ra}_G + l^{Ra}_G \tag{3.2}$$

The model generator, and discriminator losses are displayed in equations 3.1 and 3.2 respectively. The adversarial losses l^{Ra}_G and l^{Ra}_G follow from equations 2.4 and 2.5 respectively, while the content loss $l^{SR}_{VGG/i,j}$ follows from equation 2.1, with the modification described in the ESRGAN Section 2.1.3.

Chapter 4

Experiments

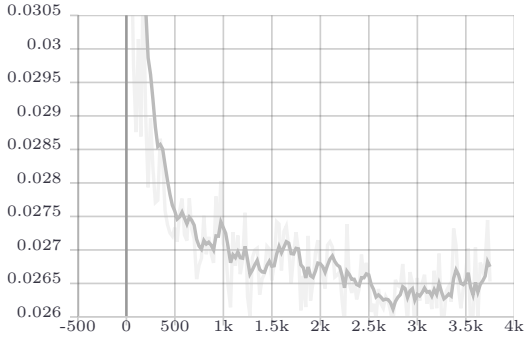
This chapter will cover how the models were trained, by documenting the software, libraries and data used, as well as the hardware difficulties faced. Before training the Super Resolution model, depth maps of all of our training data were created. As explained in Section 3.1, the Depth Estimation model used was pretrained and so will not be discussed further in this section.

4.1 Training Details

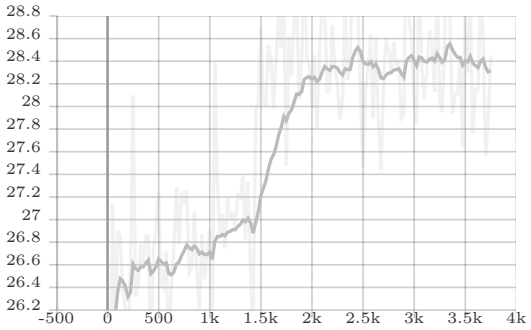
Following ESRGAN[6], training and testing were conducted on a scale factor of $\times 4$ between LR and HR images. LR images were created by down-sampling HR images using the Bicubic interpolation mode of torchvision's resize function. Once resized, LR images were passed through the HRDepth [7] model to create LR depth maps.

During training, random corresponding 128×128 HR and 32×32 LR & Depth patches were cropped. It was observed that the use of 128×128 size patches yield better results compared to smaller ones. This is due to them provide a large receptive field in deeper models, and therefore help capture more context. It is important to mention though, that these large patch sizes, along with the 32.0 million trainable parameters of the ESRGAN_D model require a lot if computing resources and time during training. More details about this problem will be presented in Section 4.1.2.

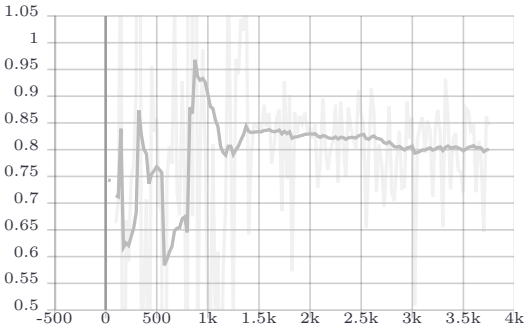
The new model was trained in two stages. First, the generator model was pre-trained without the discriminator, using only the L1 pixel loss for 150 epochs. During this part of the training, the Adam optimizer[30] was used with a learning rate of 0.0002. As has been shown by Wang et al.[6], by pertaining the model with L1 loss, GANs can obtain more visually pleasing results. This is due to the fact that initially they can avoid undesirable local minimums and results sent to the discriminator are not completely fake, allowing the focus on details earlier in the training process.



(a) Pre-training L1 Loss graph



(b) Pre-training validation PSNR graph



(c) Pre-training validation SSIM graph

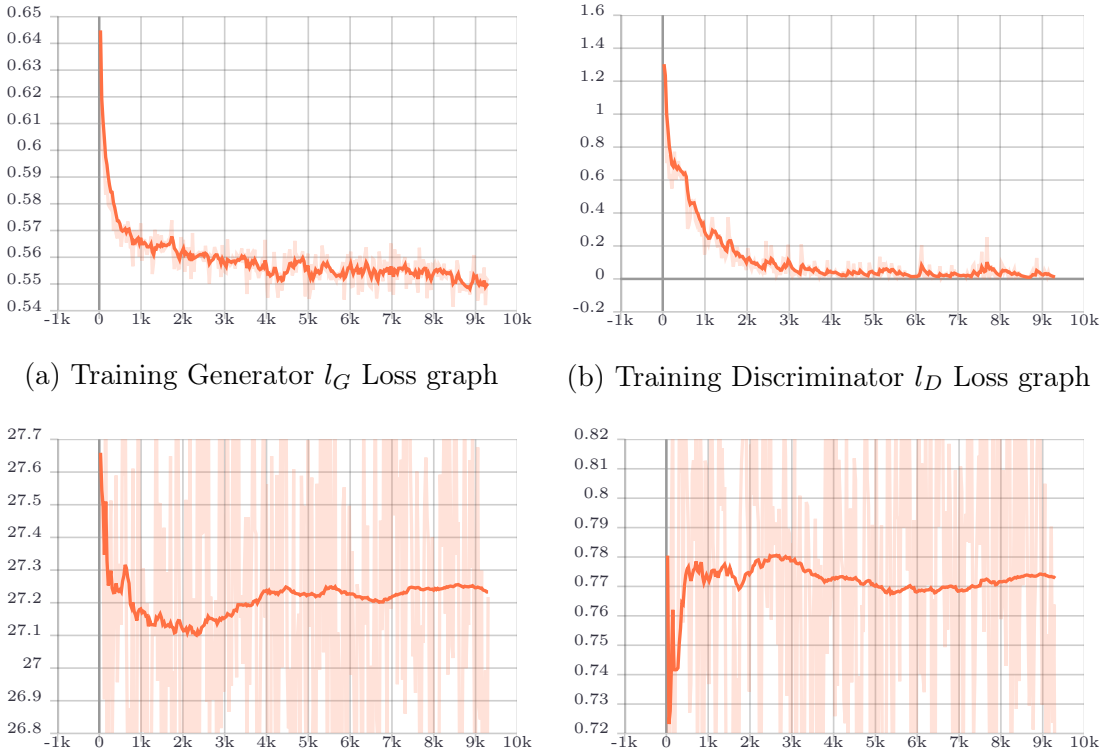
Figure 4.1: Pre-Training L1 Loss graph and PSNR/SSIM Validation metric graphs. Y axis: Loss metric, X axis: Iterations.

In Figure 4.1a the L1 training loss over 4k iterations corresponding to 150 epochs is presented. Similarly, in Figures 4.1b and 4.1c the PSNR and SSIM validation losses are shown. As expected, PSNR increases drastically over the training period, as it depends heavily on the L1 loss function. In contrast, SSIM seems to be very constant, especially later in the training period. This is once again expected as SSIM does not rely on L1, and is therefore not affected by this type of training.

After this pre-training phase, the generator is trained with the discriminator for 300 epochs. The loss functions used are presented in equations 3.1 and 3.2. The learning rate was set to 1×10^{-4} with the Adam optimizer and the β values of 0.9 and 0.999.

Following the same structure as above, figures 4.2a and 4.2b present the Generator and Discriminator Loss graphs over the 9k iterations corresponding to 300 epochs. Here the loss functions used are complex perceptual losses. As is discussed further in Section 5.1, PSNR and SSIM do not directly correlate with perceptual quality. This is apparent in figures 4.2c and 4.2d as during the whole training period both PSNR and SSIM are only changed by very small amounts.

All graphs presented in this section where created using the ESRGAN_D of length 23 & 6 for the image and depth branches respectively. Similar training has been conducted on the ESRGAN_D of length 16 & 5 although graphs for that model are not displayed, as they present only slightly worse results. All images presented in the Chapter 5 have been created using the 23 depth model.



(a) Training Generator l_G Loss graph

(b) Training Discriminator l_D Loss graph

(c) Training validation PSNR graph

(d) Training validation SSIM graph

Figure 4.2: Training Generator/Discriminator Loss graphs and PSNR/SSIM Validation metric graphs. Y axis: Loss metric, X axis: Iterations.

4.1.1 Software and Libraries

The models were implemented with the Pytorch framework[1]. The Pytorch-Lightning [31] framework was also used, to allow for conveniently logging results, easily switching between hardware devices, and overall faster research.

Pytorch-Lightning was chosen due to its ease of use. With a single variable change, the same model can be trained on CPUs, GPUs, TPUs, or even any distributed setup. Visualizations, logging and checkpointing happens automatically regardless of hardware, and if the model has any performance bottlenecks, helpful notifications show how to fix those issues. In combination with the torchmetrics project[2], that provides both SSIM and PSNR functions, training a network anywhere became quick and easy.

4.1.2 Hardware

As stated previously, finding appropriate hardware to train such a large model with big patch sizes and sufficiently large batch sizes in a reasonable amount of time is quite challenging. To obtain the best solution for this problem, an investigation

[1] https://pytorch.org
[2] https://torchmetrics.readthedocs.io/

was conducted. Due to the ease of running code on different hardware, provided by pytorch-lightning, an array of available devices were tested. Due to circumstances beyond our control, like the 2021 GPU shortage, the increasing contention for GPUs by students in DoC Labs and unreliable GPU provides like Paperspace[3], we had a limited range of devices available. Despite this, the investigation proved insightful as it provided information about the power and usability of available hardware in 2021 for training deep models.

Table 4.1: Hardware Investigation Training results and times

Device	CPU/RAM	Batch Size	Time/1 epoch	Time/200 epochs
Google Colab - Tesla V100 16G	4 threads/12G	1, 4, 16	Unexpected kernel shutdown	
Google Colab - Tesla P100 16G	4 threads/12G	1, 4, 16	Unexpected kernel shutdown	
DoC Labs - Titan X 12G	8 threads/32G	4	14 m 18 s	-
Paperspace - Quadro P6000 24G	16 threads/32G	"Setting up instance" for 6+ hours		
Paperspace - Tesla V100 32G	16 threads/32G	16	8 m 32 s	18 h 12 m
Google Colab - TPU 1 core	40 threads/32G	16	11 m 50 s	14 h 55 m
Google Colab - TPU 8 cores	40 threads/32G	16×8	9 m 01 s	10 h 17 m

As seen in Table 4.1 above, many GPUs were insufficient for any training with the proposed model. Both Tesla V100 and P100 with 16G of GPU memory from Google Colab forced an unexpected restart of the Jupiter notebook kernel for any batch size. The vms provided with these cards by Google had 4 CPU threads and 12 G of RAM so there bottleneck was probably the CPU/RAM.

Moving on, surprisingly the Titan X in DoC lab gpu machines was able to start training without initial memory issues regardless of its smaller, 12G GPU memory. As expected though, it was only able to train with batch size up to 4, or else it would run out of CUDA memory. It was also not possible to find a free Titan X to use in labs for a large continuous amount of time as any student can log in and start other computations, causing training to go slower or even stop due to lack of memory. As such, the 200 epochs could not be tested.

The next tests were performed on Paperspace using the DoC license given to students. From prior testing, the GPUs expected to work were those with more than 12G of GPU memory, therefore the Quadro P6000 (24 G) and the Tesla V100 (32G) were chosen. Setting up the Quadro P6000 was not possible as in 4 separate occasions, the instance was stuck at the setup stage. Therefore, no testing was conducted with this card.

After working through the many issues with paperspace, including accessing logs in tensorboard and getting a stable connection to the notebook, the Tesla V100 was tested successfully. As seen in the table, the Paperspace - Tesla V100 32G had the fastest first epoch time measured and this looked promising. Even though it further increased in speed per epoch later in training, reaching an average of 5.46 min, the total training time was 18 hours, which was not the best result.

[3] https://www.paperspace.com/

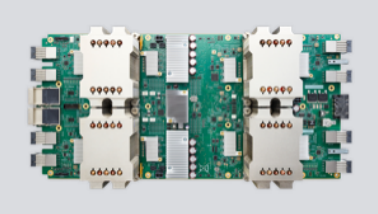


Figure 4.3: Cloud TPU v2[4]

Finally, the last devices tested are quite different to the previous ones. Google provides access to Cloud TPU v2, that are shown in Figure 4.3, through Google Colab[5]. These devices are custom made hardware that provide 180 teraflops of computational power and 64 GB High Bandwidth Memory (HBM). Articles like [32] have analyzed the differences between TPUs and GPUs with the general conclusion being that given large, computationally intensive workloads, and with limited tensors moved to the CPU for logging or other reasons, TPUs are generally faster. The AI domain specific architecture of TPUs allows matrix multiplication operations to happen extremely quickly and with less power.

Following the explanation in the Google Cloud blog post[6], when training a network, the model and parameters are initially compiled for the TPU and loaded in its memory. Then, the data is loaded, and as each multiplication is executed, the result can be passed onto the next multiplayer while also taking summations at the same time. This then continues for the next inputs. During this whole process, the benefit of TPUs is that no memory access is required at all, until the end of the computation.

As can be seen in the table, TPUs can be used with a single core or up to the 8 cores provided in Colab. Each core can calculate a batch at a time and can also seamlessly communicate with the other cores to make sure they all train a single model. For this reason, as seen in the table, the 8 core TPU had an effective batch size of 128. Both the single and multicore versions of the TPUs took a long time for the fist epoch, as everything needed to be loaded in the cores, but as seen in the 200 epoch test, they were able to outperform the Tesla V100.

To conclude the investigation, TPUs with 8 cores was the fastest and easiest to use solution. Therefore they were used for training the proposed model. In addition, as will be discussed in the Future work section, larger TPU pods can potentially provide even faster training, and could potentially be used in the future to train even deeper models.

[4] https://cloud.google.com/tpu
[5] https://colab.research.google.com/
[6] https://tinyurl.com/tpu-machine-learning

4.2 Data

4.2.1 Training

Following inspiration from previous SR papers, the DIV2K [33] dataset along with other well known image datasets were used. To further enhance the quality of the results, as proposed with ESRGAN, the datasets must have rich and diverse textures and objects. This was achieved by adding the Flickr2K [34] dataset while training. The images were also randomly flipped and rotated along with their depth counterpart. More details about the datasets below:

- **DIV2K**
  This dataset is used in the NITRE challenge. It contains 800 images of about 2K high-resolution. It has a mixture of textures, scenes and objects. Unfortunately the only having 800 images is a bit limiting on its own.

- **Flickr2K**
  The Flickr2K dataset consisting of 2650 2K high-resolution images collected from the Flickr website. These images are more diverse that the DIV2K dataset and can provide further texture detail.

4.2.2 Testing

For testing, the usual image datasets were used. These include Set5[35], Set14[36], BSD100[37], DIV2K[33], Urban100[38] and Manga109[39]. These datasets are widely used benchmarks and can be used to compare the results of this model with others.

Looking back at how the design of our model works, it is understood that the quality of the output is dependant on the quality of the depth estimation maps. Since the HRDepth model was itself pretrained with the Kitty dataset, it is required to understand what it can predict well. This is explored in detail by Tom van Dijk in[40]. From his conclusions it is evident that city scenes with cars, trees etc. are better reflected in depth images. Depth Estimation can detect objects not in the training set but it usually works better when the environment helps by displaying shadows or other details.

As such, in addition to the above testing datasets, it was decided to add a new dataset of cities, cars and roads. More specifically, cities, cars and roads in London. Unfortunately, taking these pictures during the covid pandemic and lockdown was not possible, and so, an alternative was found. A new WatchDogs10 dataset is proposed containing 10 HD images from the Watch Dogs: Legion[7] Video Game based in London. It is used to detect the effect of SR on images with good predicted depth maps. Thumbnails of the images are show in Figure 4.4.

[7] https://www.ubisoft.com/en-gb/game/watch-dogs/legion

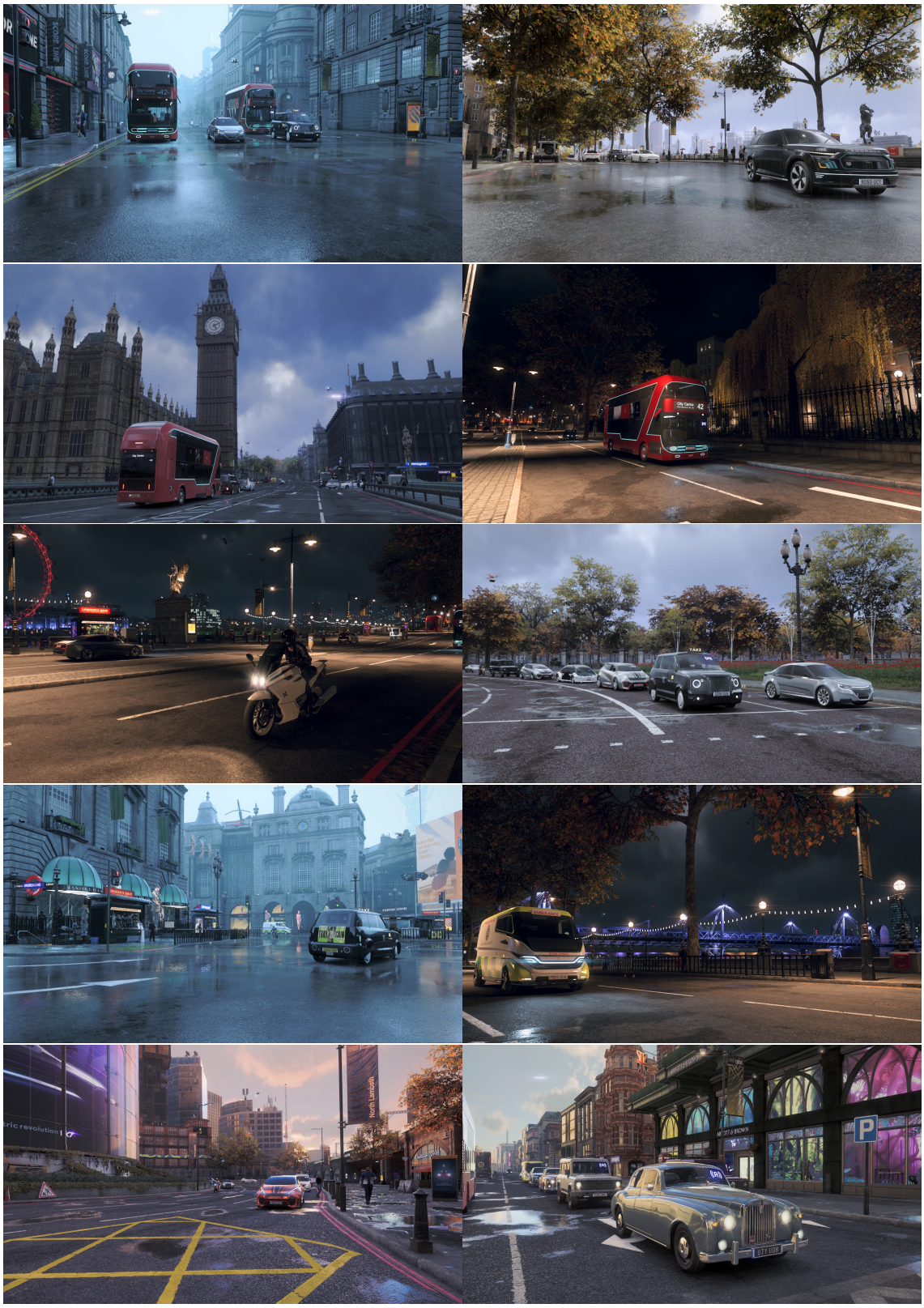


Figure 4.4: WatchDogs10 Dataset used for evaluation

Chapter 5

Evaluation

5.1 Metrics

Designing metrics to define the quality of an image is inherently an almost impossible task. In the field of Super Resolution two main algorithms are used. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM). This section will walk through each metric, and explain the benefits and limitations of each.

5.1.1 PSNR

Peak Signal-to-Noise Ratio is commonly used to measure the quality on a reconstructed lossy image. In simple terms it is the ratio of the maximum possible power of a signal (in images this is generally 255) and the pixel wise difference of the original image with the reconstructed image.

$$MSE(I, J) = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - J(i, j)]^2 \tag{5.1}$$

It is easily defined using the Mean Squared Error formula shown in 5.1 where I is the original image and J is the reconstructed image. Due to the nature of image pixels having a very wide dynamic range, PSNR is also defined as a logarithmic function with the units of the Decibel (dB) scale. It is therefore calculated as shown in equation 5.2.

$$\text{PSNR}(I, J) = 10 \times \log_{10} \left(\frac{\max(I)^2}{\text{MSE}(I, J)} \right) \tag{5.2}$$

In order to compare with results from other papers, and be consistent, PSNR will only be applied on the y-channel of images. Even though this metric is widely used,

Figure 5.1: From left to right: Bicubic interpolation, SRResNet output trained with MSE/PSNR metric, the proposed ESRGAN_D model output trained with perception loss described in equation 2.6, the original HR image. The numbers show PSNR and SSIM respectively. This is a $\times 4$ upsampling example.

the ability of MSE (and therefore PSNR) to capture perceptual information about textures is very limited as described in [41, 42, 43]. This can also be seen in Figure 5.1 where the highest PSNR value is shown for the SRResNet model even though the proposed ESRGAN_D model has a perceptually better result.

5.1.2 SSIM

The Structural Similarity Index Measure in contrast, attempts to perceive changes in structural information. As explained in [42], SSIM is calculated on $N \times N$ windows sizes where $N = 11$ in the default case. results are then reduced using an element-wise mean over the whole picture.

Before defining SSIM, three alternative functions need to be defined: luminance (l) 5.3, contrast (c) 5.4 and structure (s) 5.5. In all the following functions, the notation used is x and y referring to matrix window patches of size $N \times N$ of the source and target images respectively. Following that, μ_x and μ_y are referring to the average, while σ_x and σ_y are referring to the variance of x and y respectively. Finally c_1 and c_2 are constants derived using the dynamic range of the image and $c_3 = c_2/2$.

$$l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{5.3}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{5.4}$$

$$s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{5.5}$$

These formulas are then combined to form the SSIM measure using α, β and γ as weights:

$$\text{SSIM}(x, y) = \left[l(x, y)^{\alpha} \cdot c(x, y)^{\beta} \cdot s(x, y)^{\gamma}\right] \tag{5.6}$$

To conclude, the most common use of this equation is when α, β and γ are set to 1 and thus the formula is reduced to the one shown below:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.7}$$

Over the years improvements have been made such as multi-scale window sizes and more. However, even though these metrics have been used in papers and will be used here too, it has been shown by Ledig et al.[5] and others that they fundamentally disagree with the subjective evaluation of human observers. A clear example of this is again Figure 5.1 shown above. Therefore it is important to state that the results in the tables below are not directly indicative of the quality of the images and hence examples will be used so that the reader can observe the results.

5.1.3 Alternative Measures

Only recently, new measures have been introduced to attempt better detection of the perceptual quality of images. This is possible because of the better understanding of perception and distortion trade-offs explained in [44]. These new measures include Ma's [45] score and NIQE [46] and they have both been used in the PIRM-SR Challenge [47]. Unfortunately not many papers have used these metrics and therefore not many results are available for comparison. To add to that, the PIRM-SR challenge and dataset are no longer available online and so evaluating out model with it will not be possible. This would be an interesting direction for future work though.

5.2 Qualitative Results

Table 5.1: PSNR and SSIM comparisons of the Bicubic algorithm, ESRGAN model and the proposed ESRGAN_D model. Mean metric calculated per dataset

Dataset	PSNR			SSIM		
	Bicubic	ESRGAN	ESRGAN_D	Bicubic	ESRGAN	ESRGAN_D
Set5	28.644	30.465	30.725	0.8144	0.8518	0.8648
Set14	26.336	26.397	27.502	0.7116	0.7008	0.7437
DIV2K	28.213	28.166	28.823	0.7772	0.7758	0.7992
Urban100	23.219	24.338	24.993	0.6605	0.7326	0.7515
Manga109	25.350	28.678	29.075	0.7986	0.8654	0.8792
WD10	31.251	30.590	31.370	0.8502	0.8046	0.8363

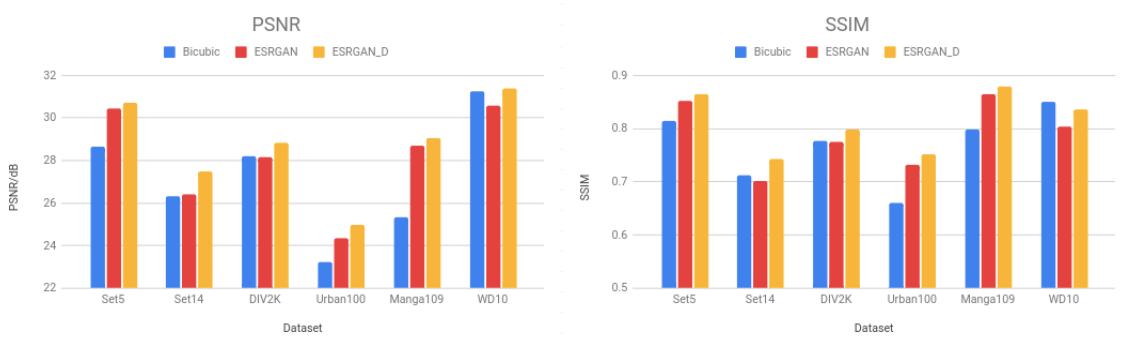


Figure 5.2: PSNR and SSIM comparisons of the Bicubic algorithm, ESRGAN model and the proposed ESRGAN_D model. Mean metric presented per dataset

In order to assess the success of this project both PSNR/SSIM metrics described above, and visual results will be presented. These methods will be used to compare the proposed ESRGAN_D model with its non-depth ESRGAN counterpart. As explained in Section 4.2.2, metrics alone do not necessarily mean better results.

Regardless, as shown in Table 5.1 and similarly in Figure 5.2, when considering PSNR and SSIM in the $\times 4$ enlargement challenge, the proposed ESRGAN_D model surpasses its no-depth counterpart in every dataset. As expected, Bicubic generally has the worst results, since all textures are smooth and no sharp edges are created. The difference between ESRGAN and ESRGAN_D is not as clear in the image results though.

Figure 5.3: Baboon and Monarch from Set14 Dataset. In order of top right to bottom left: HR, ESRGAN_D, Bicubic, ESRGAN, Depth map.

As shown in Figure 5.3, The depth estimation is not very helpful here. As such, ESRGAN takes the lead in these images, providing better texture reconstruction.

This is usually the case in images with limited depth information. This is expected, as the extra information provided here is not helpful, and only confuses the network.

In contrast, the images shown in Figure 5.4 belong to the new WD10 Dataset and have useful depth information. Here, the HR images are displayed along with the depth estimation maps, followed by zoomed in, cropped regions, enhanced from the LR counterparts using Bicubic, ESRGAN and ESRGAN_D methods. It can be observed that in regions with clear depth differences, ESRGAN_D results are sharper and clearer. At the same time though, in regions with flat depth estimations, textures are smoother than expected. Overall, since these scenes have a lot of well-estimated depth, it is possible to show that ESRGAN_D performs better.

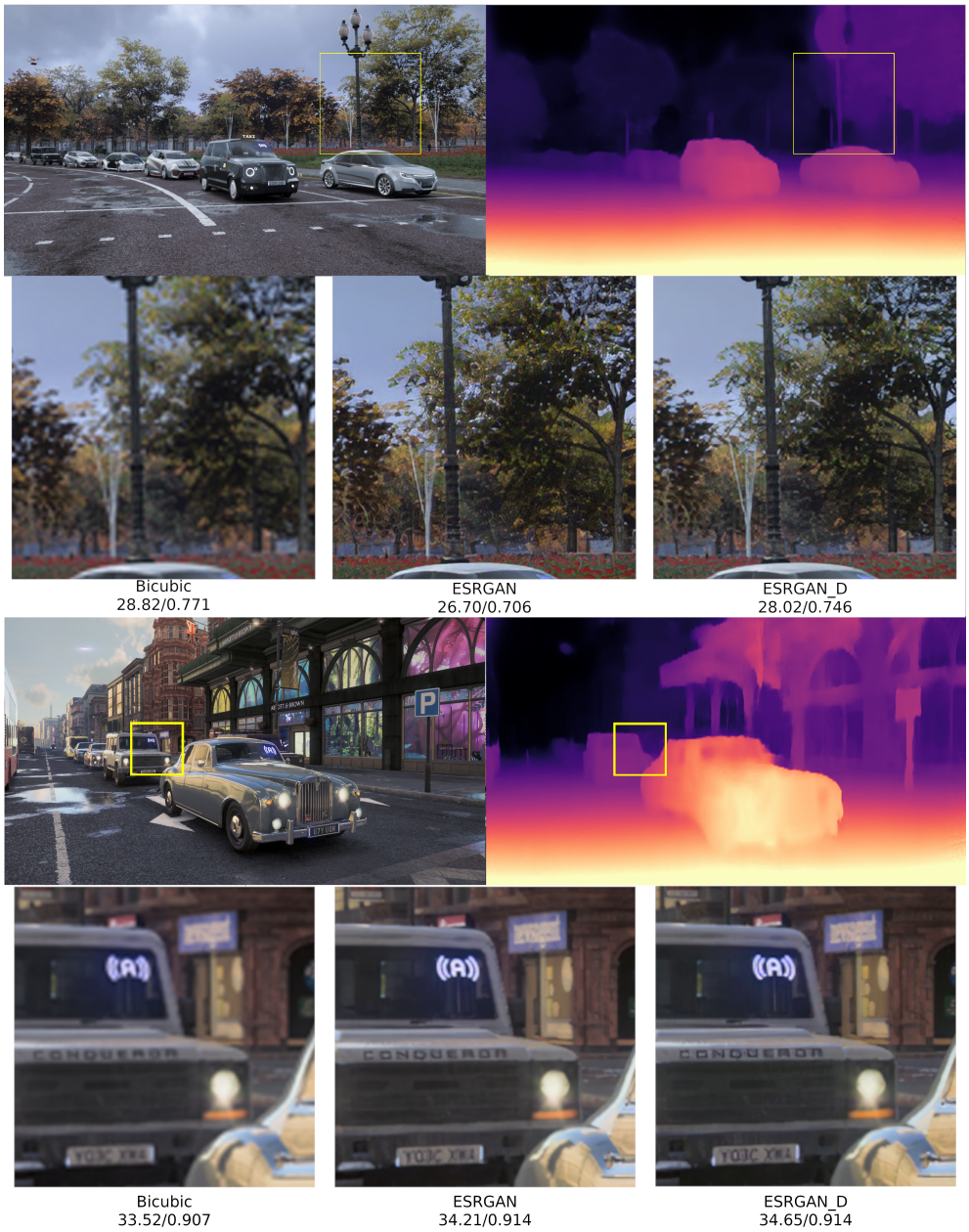


Figure 5.4: Comparison between Bicubic, ESRGAN and ESRGAN_D in sample WD10 dataset images with PSNR/SSIM metrics.

Finally an other example where DE helps SR in the DIV2K dataset can be seen in Figure 5.5. The passengers in this image have sharper outlines and the train hidden behind the street light is more clearly segmented compared to base ESRGAN.

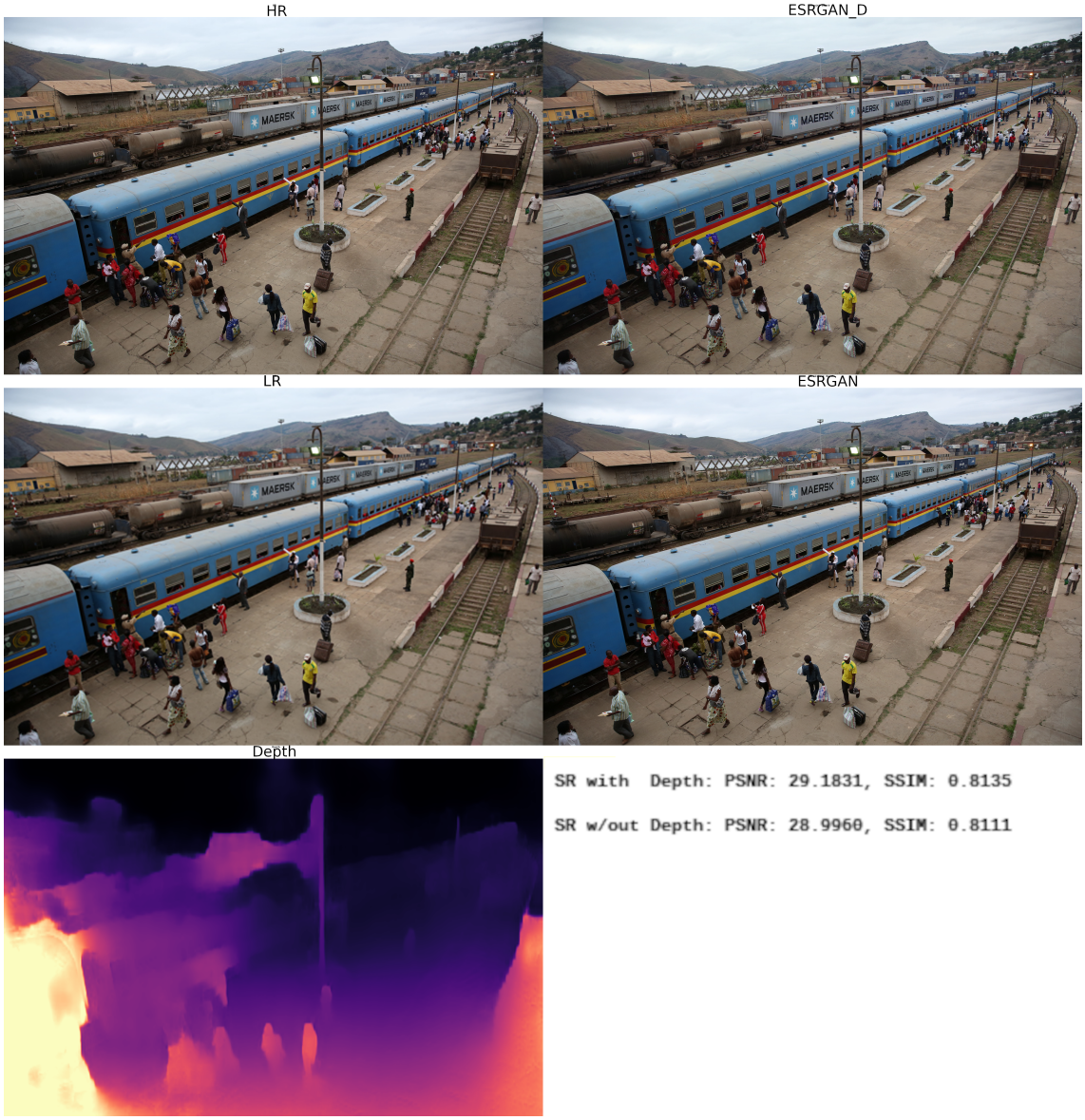


Figure 5.5: 0850 from the DIV2K test dataset with PSNR/SSIM metrics. In order of top right to bottom left: HR, ESRGAN_D, Bicubic, ESRGAN, Depth map.

The full results of this evaluation include 438 images that have been generated from all the test datasets mentioned above. An output of all these images can be found in the Results folder[1]. In addition, a comprehensive list of all metrics can also be found in Appendix A. In general the outcomes expressed above are the same for a majority of the generated images.

[1] https://www.doc.ic.ac.uk/~gs2617/results/

Chapter 6

Conclusion

6.1 Summary

Looking back at the project objectives, every goal outlined was met. Challenges found on the way have also been described and dealt with. Initially, state-of-the-art Depth Estimation models were recreated from the latest papers, providing useful depth maps for the training and testing datasets as shown in Section 3.1. Following that, after exploring many SR models, a new state-of-the-art ESRGAN_D model was proposed in section 3.2. It was then implemented, trained and tested with existing datasets, and also with a newly created one presented in Figure 4.4. Hardware issues have been explained and overcome using Google Cloud TPUs as shown in Section 4.1.2. Evaluation metric difficulties have also been discussed in Section 5.1 and many image examples of results have been provided to overcome the problem of perceptual quality. Finally, the results created have been evaluated and are comparable to existing models. In comparison, the ESRGAN_D proposed model benefits by having sharper edges on complex scenes that include variable depths, but also has some downsides explained in section 5.2.

6.2 Future Work

This project demonstrates that the use of depth data, even if it is estimated, can potentially improve Super Resolution. While working on the project, multiple avenues for further optimisation or research have become apparent. A discussion will be presented here showing these ideas.

6.2.1 TPU Pod accelerated training

As explained the Hardware section above, Google TPUs offer great improvements in the speed of training. In this project 8 cloud TPU v2 cores were used as they were provided in Google Colab. Alternatively, Google Cloud provides TPU v3 and by the end of the year will provide TPU v4 to paying customers. These cores have significant improvements over the previous models. For example the v2 cores used in this project can achieve a predicted 180 teraflops of computational power while the v3 can achieve 420 teraflops.

Figure 6.1: TPU v3 Pods [1]

To add to that, both v2 and v3 chips can be stacked and used as Pods of thousands of chips seamlessly communicating over a meshed network to provide over 11 and 100 petaflops respectively. TPU Pods v3 can be seen in Figure 6.1. These pods have also proven to be cheaper to use than normal GPUs as they finish training a lot faster and thus need less hours to be paid for. Using them in the future could allow for deeper models to be trained and be able to extract more features from multiple data types. This could possibly produce even better results.

6.2.2 Full end to end model with Transformers

As shown, currently two separate models are used for Depth Estimation and Super Resolution. These models could potentially be integrated together in the future into a single model with multiple losses. This would provide more difficulties though as stereo images are usually used in DE training while not used in SR. If this challenge is overcome, the DE model could continue training and improving while the SR modeled is trained too. Yet another expansion to the above would be the use of transformers, as attention could be used to understand the influence of depth even better.

6.2.3 Easy to use interactive interface to enhance images

It has been noticed that even though many models and papers exist for Super Resolution, there are no easy to use interfaces to try out these state-of-the-art models. To improve on this, a web interface with a back-end server to run these models

[1] https://cloud.google.com/tpu

would be useful to showcase them. In addition, a way to make this web interface deployable on any hardware would be useful and would allow many users to explore and understand these models better. An initial attempt to continue this work has started as an extension to this project and is available here: https://github.com/georgesoteriou/super-sampling-demo. Technologies used include Docker[48] for easy deployability, Flask[49] as backend and Vue Framework[2] as the front end environment.

6.3 Ethical issues

Due to the nature of this project, there are no users and therefore no personal information collection that needs to be discussed. Furthermore, all code and data that will be used for this project have sufficient licences which enables their use without any copyright issues.

[2] https://vuejs.org/

Bibliography

[1] Yang W, Zhang X, Tian Y, Wang W, Xue JH, Liao Q. Deep Learning for Single Image Super-Resolution: A Brief Review. IEEE Transactions on Multimedia. 2019;21(12):3106–3121.

[2] Kim J, Lee JK, Lee KM. Accurate Image Super-Resolution Using Very Deep Convolutional Networks; 2016.

[3] Ahn N, Kang B, Sohn KA. Fast, accurate, and lightweight super-resolution with cascading residual network. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2018;11214 LNCS:256–272.

[4] Lim B, Son S, Kim H, Nah S, Lee KM. Enhanced Deep Residual Networks for Single Image Super-Resolution. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. 2017 jul;2017-July:1132–1140. Available from: http://arxiv.org/abs/1707.02921.

[5] Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, et al.. Photo-realistic single image super-resolution using a generative adversarial network. Institute of Electrical and Electronics Engineers Inc.; 2017.

[6] Wang X, Yu K, Wu S, Gu J, Liu Y, Dong C, et al. ESRGAN: Enhanced super-resolution generative adversarial networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2019;11133 LNCS:63–79.

[7] Lyu X, Liu L, Wang M, Kong X, Liu L, Liu Y, et al. HR-Depth: High Resolution Self-Supervised Monocular Depth Estimation. 2020 dec. Available from: http://arxiv.org/abs/2012.07356.

[8] Xiao L. Neural Supersampling for Real-time Rendering - Facebook Research. Siggraph. 2020;39(4):1–12. Available from: https://research.fb.com/publications/neural-supersampling-for-real-time-rendering/.

[9] Alansary A, Rajchl M, McDonagh SG, Murgasova M, Damodaram M, Lloyd DFA, et al. PVR: Patch-to-Volume Reconstruction for Large Area Motion Correction of Fetal MRI. IEEE Transactions on Medical Imaging. 2017 oct;36(10):2031–2044.

[10] Müller MU, Ekhtiari N, Almeida RM, Rieke C. Super-Resolution of Multispectral Satellite Images Using Convolutional Neural Networks; 2020. 1.

[11] Aakerberg A, Nasrollahi K, Moeslund TB. Real-World Super-Resolution of Face-Images from Surveillance Cameras. 2021 feb. Available from: http://arxiv.org/abs/2102.03113.

[12] Vavilala V, Meyer M. Deep Learned Super Resolution for Feature Film Production. 2020. Available from: https://doi.org/10.1145/3388767.

[13] Christensen JH, Mogensen LV, Ravn O. Single Image Super-Resolution for Domain-Specific Ultra-Low Bandwidth Image Transmission; 2020.

[14] Dong C, Loy CC, He K, Tang X. Learning a Deep Convolutional Network for Image Super-Resolution. Springer Verlag; 2014. Available from: https://ieeexplore.ieee.org/document/7115171.

[15] Hayat K. Super-Resolution via Deep Learning. arXiv. 2017:1–33.

[16] Park SC, Park MK, Kang MG. Super-resolution image reconstruction: A technical overview. IEEE Signal Processing Magazine. 2003;20(3):21–36. Available from: https://ieeexplore.ieee.org/document/1203207.

[17] Dong C, Loy CC, He K, Tang X. Image Super-Resolution Using Deep Convolutional Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2016 feb;38(2):295–307.

[18] Godard C, Aodha OM, Firman M, Brostow G. Digging into self-supervised monocular depth estimation; 2019. Available from: www.github.com/nianticlabs/monodepth2.

[19] Keys R. Cubic convolution interpolation for digital image processing. IEEE Transactions on Acoustics, Speech, and Signal Processing. 1981;29(6):1153–1160.

[20] Duchon CE. Lanczos Filtering in One and Two Dimensions. Journal of Applied Meteorology and Climatology;18(8):1016–1022. Available from: https://journals.ametsoc.org/view/journals/apme/18/8/1520-0450_1979_018_1016_lfioat_2_0_co_2.xml.

[21] Li X, Orchard M. New edge-directed interpolation. Image Processing, IEEE Transactions on. 2001 11;10:1521 – 1527.

[22] Anwar S, Khan S, Barnes N. A Deep journey into super-resolution: a survey. arXiv. 2019:1–21.

[23] Shi W, Caballero J, Huszar F, Totz J, Aitken AP, Bishop R, et al. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network; 2016.

[24] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 2016-December. IEEE Computer Society; 2016. p. 770–778. Available from: http://image-net.org/challenges/LSVRC/2015/.

[25] Nah S, Kim TH, Lee KM. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. 2016 dec;2017-Janua:257–265. Available from: http://arxiv.org/abs/1612.02177.

[26] Jiao J, Tu WC, He S, Lau RWH. FormResNet: Formatted Residual Learning for Image Restoration. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. 2017;2017-July:1034–1042.

[27] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. International Conference on Learning Representations, ICLR; 2015. Available from: http://www.robots.ox.ac.uk/.

[28] Godard C, Mac Aodha O, Brostow GJ. Unsupervised monocular depth estimation with left-right consistency. In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. vol. 2017-Janua; 2017. p. 6602–6611. Available from: http://visual.cs.ucl.ac.uk/pubs/monoDepth/.

[29] Geiger A, Lenz P, Urtasun R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Conference on Computer Vision and Pattern Recognition (CVPR); 2012. .

[30] Kingma DP, Ba JL. Adam: A method for stochastic optimization; 2015.

[31] Falcon ea WA. PyTorch Lightning. 2019;3. Available from: https://github.com/PyTorchLightning/pytorch-lightning.

[32] Wang YE, Wei GY, Brooks D. Benchmarking TPU, GPU, and CPU Platforms for Deep Learning. 2019 jul. Available from: http://arxiv.org/abs/1907.10701.

[33] Agustsson E, Timofte R. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops; 2017. .

[34] Timofte R, Agustsson E, Van Gool L, Yang MH, Zhang L, Lim B, et al. NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops; 2017. .

[35] Hou J, Si Y, Yu X. A Novel and Effective Image Super-Resolution Reconstruction Technique via Fast Global and Local Residual Learning Model. Applied Sciences. 2020;10(5). Available from: https://www.mdpi.com/2076-3417/10/5/1856.

[36] Zeyde R, Elad M, Protter M. On Single Image Scale-Up Using Sparse-Representations. vol. 6920; 2010. p. 711–730.

[37] Martin D, Fowlkes C, Tal D, Malik J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. vol. 2; 2001. p. 416–423 vol.2.

[38] Huang JB, Singh A, Ahuja N. Single Image Super-Resolution From Transformed Self-Exemplars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. .

[39] Fujimoto A, Ogawa T, Yamamoto K, Matsui Y, Yamasaki T, Aizawa K. Manga109 Dataset and Creation of Metadata. In: Proceedings of the 1st International Workshop on CoMics ANalysis, Processing and Understanding. MANPU '16. New York, NY, USA: Association for Computing Machinery; 2016. Available from: https://doi.org/10.1145/3011549.3011551.

[40] Van DIjk T, De Croon G. How do neural networks see depth in single images? In: Proceedings of the IEEE International Conference on Computer Vision. vol. 2019-Octob; 2019. p. 2183–2191.

[41] Wang Z, Simoncelli EP, Bovik AC. Multi-scale structural similarity for image quality assessment; 2003.

[42] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: From error visibility to structural similarity; 2004. 4. Available from: https://ieeexplore.ieee.org/document/1284395.

[43] Torch | Training and investigating Residual Nets;. Available from: http://torch.ch/blog/2016/02/04/resnets.html.

[44] Blau Y, Michaeli T. The Perception-Distortion Tradeoff; 2018. Available from: https://ieeexplore.ieee.org/abstract/document/8578750.

[45] Ma C, Yang CY, Yang X, Yang MH. Learning a no-reference quality metric for single-image super-resolution; 2017.

[46] Mittal A, Soundararajan R, Bovik AC. Making a 'completely blind' image quality analyzer; 2013. 3. Available from: http://live.ece.utexas.edu/research/quality/niqe.

[47] Blau Y, Mechrez R, Timofte R, Michaeli T, Zelnik-Manor L. The 2018 PIRM challenge on perceptual image super-resolution; 2019.

[48] Merkel D. Docker: lightweight linux containers for consistent development and deployment. Linux journal. 2014;2014(239):2.

[49] Grinberg M. Flask web development: developing web applications with python. " O'Reilly Media, Inc."; 2018.

Appendix A

Raw Data

Table A.1: Set 5 Dataset Metrics

Image	psnr esrgan	ssim esrgan	psnr esrgan_d	ssim esrgan_d	psnr bicubic	ssim bicubic
baby.png	31.5163	0.8441	31.9262	0.8555	31.9645	0.8607
bird.png	33.6278	0.9259	33.2384	0.9237	30.4701	0.8780
butterfly.png	27.5757	0.9044	27.6673	0.9045	22.3786	0.7403
head.png	29.9376	0.6862	31.1757	0.7454	31.6188	0.7557
woman.png	29.6660	0.8986	29.6198	0.8948	26.7886	0.8375
Average	30.4647	0.8518	30.7255	0.8648	28.6441	0.8144
StDev	2.2569	0.0974	2.1526	0.0712	4.0592	0.0626

Table A.2: Set14 Dataset Metrics

Image	psnr esrgan	ssim esrgan	psnr esrgan_d	ssim esrgan_d	psnr bicubic	ssim bicubic
baboon.png	20.2684	0.4295	22.0372	0.4936	22.4679	0.4586
barbara.png	24.0716	0.6956	25.3753	0.7169	25.1695	0.6894
bridge.png	22.1930	0.5014	24.2250	0.5924	24.5837	0.5759
coastguard.png	22.6121	0.3678	24.8477	0.4933	25.6685	0.5207
comic.png	21.2081	0.6461	22.7439	0.6966	22.2267	0.6077
face.png	30.3089	0.7041	31.2389	0.7472	31.5811	0.7542
flowers.png	26.1326	0.7538	26.7737	0.7704	25.7562	0.7327
foreman.png	33.1137	0.8988	33.5831	0.9026	30.2198	0.8722
lenna.png	29.8190	0.7959	30.7167	0.8222	29.9158	0.8148
man.png	24.5844	0.6566	26.3523	0.7109	25.7623	0.6803
monarch.png	31.8918	0.9257	31.6314	0.9280	27.5687	0.8818
pepper.png	32.6397	0.8334	32.6955	0.8490	30.8079	0.8393
ppt3.png	25.6345	0.9409	26.1416	0.9368	22.3044	0.8228

zebra.png	25.0768	0.6611	26.6607	0.7514	24.6689	0.7121
Average	26.3967	0.7008	27.5017	0.7437	26.3358	0.7116
StDev	4.3753	0.1771	3.7675	0.1432	3.2028	0.1323

Table A.3: DIV2K Dataset Metrics

Image	psnr esrgan	ssim esrgan	psnr esrgan_d	ssim esrgan_d	psnr bicubic	ssim bicubic
0801.png	28.2113	0.8021	29.5074	0.8300	29.1122	0.8083
0802.png	33.5380	0.8381	34.2269	0.8674	35.0738	0.8964
0803.png	40.2640	0.9617	39.2773	0.9657	35.2399	0.9529
0804.png	27.7564	0.7803	27.5936	0.7758	27.0065	0.7397
0805.png	27.2993	0.6929	28.8772	0.7565	28.9198	0.7554
0806.png	27.7775	0.8301	28.8286	0.8472	28.2434	0.8337
0807.png	18.6667	0.4052	20.5071	0.4921	20.9579	0.4561
0808.png	24.8300	0.7356	26.5345	0.7820	26.8834	0.7868
0809.png	30.5392	0.7901	32.1170	0.8362	32.1788	0.8389
0810.png	24.8338	0.6923	26.8291	0.7570	27.0173	0.7497
0811.png	27.7048	0.7262	28.9100	0.7651	29.1788	0.7627
0812.png	25.8312	0.7533	27.4746	0.7813	26.9322	0.7573
0813.png	31.4840	0.8723	31.3615	0.8730	30.3392	0.8598
0814.png	27.8936	0.8883	27.9550	0.8890	28.0046	0.8744
0815.png	35.2791	0.9170	34.7618	0.9190	34.0283	0.9142
0816.png	30.3615	0.7815	31.3724	0.8094	31.1121	0.8181
0817.png	32.6019	0.8799	32.6954	0.8810	31.9319	0.8694
0818.png	28.9022	0.8889	29.3129	0.8918	27.9745	0.8626
0819.png	27.1040	0.7737	28.3303	0.8034	28.0924	0.7763
0820.png	23.6326	0.7551	24.4090	0.7700	23.1603	0.6934
0821.png	31.4520	0.9086	31.4626	0.9103	26.8364	0.8412
0822.png	30.1383	0.8464	30.6430	0.8530	29.4747	0.8130
0823.png	24.4914	0.7469	25.9016	0.7920	25.7788	0.7724
0824.png	27.9843	0.8710	28.6888	0.8803	26.8126	0.8448
0825.png	27.0157	0.7717	27.4521	0.7873	26.2560	0.7574
0826.png	23.7622	0.7381	24.2895	0.7451	24.1934	0.7045
0827.png	33.5908	0.8611	33.0257	0.8667	31.9187	0.8719
0828.png	22.0687	0.9246	21.6056	0.8929	17.4482	0.6789
0829.png	26.2286	0.5125	27.1194	0.5498	27.3065	0.5515
0830.png	21.9424	0.6130	24.2791	0.6970	25.1407	0.6973
0831.png	30.1599	0.8649	30.1945	0.8590	28.2986	0.8126
0832.png	28.4795	0.8249	30.5169	0.8619	29.4401	0.8437
0833.png	33.9609	0.8753	34.1770	0.8944	31.5666	0.8915
0834.png	23.9381	0.7269	25.2640	0.7547	25.9185	0.7445

0835.png	21.2471	0.4757	23.1595	0.5644	23.6023	0.5486
0836.png	25.6446	0.7751	26.2340	0.7882	24.5512	0.7201
0837.png	26.0596	0.8075	26.4752	0.8115	25.2328	0.7468
0838.png	36.7812	0.8763	36.6489	0.8964	38.7021	0.9221
0839.png	28.1947	0.7837	29.1340	0.8035	29.5378	0.8045
0840.png	28.2020	0.7803	28.7126	0.7958	28.4980	0.7823
0841.png	27.3347	0.8266	27.9584	0.8320	27.7027	0.8145
0842.png	28.4263	0.8085	29.8507	0.8384	29.6891	0.8346
0843.png	41.3282	0.9749	37.8080	0.9763	38.8742	0.9788
0844.png	45.2296	0.9878	41.9518	0.9855	40.2136	0.9843
0845.png	24.3120	0.7423	24.6591	0.7634	22.8412	0.6776
0846.png	25.4326	0.7901	25.9311	0.7946	23.5854	0.7185
0847.png	27.7915	0.8029	28.0468	0.8089	26.6750	0.7685
0848.png	27.0753	0.6884	27.6931	0.7215	26.4324	0.7121
0849.png	21.4003	0.7052	22.3651	0.7298	21.1302	0.6393
0850.png	28.9960	0.8111	29.1831	0.8135	28.1652	0.7915
0851.png	27.6722	0.8528	28.5868	0.8681	28.2796	0.8618
0852.png	28.5321	0.7855	29.3886	0.7995	30.0515	0.7989
0853.png	32.6273	0.9174	33.7339	0.9287	32.8700	0.9258
0854.png	22.8262	0.6358	24.6400	0.7035	24.7620	0.6882
0855.png	28.4358	0.6998	29.4623	0.7428	29.4725	0.7469
0856.png	24.3635	0.6354	25.1920	0.6549	26.5089	0.6660
0857.png	35.1425	0.7907	35.3440	0.8092	35.5197	0.8290
0858.png	27.4972	0.7332	28.3110	0.7534	28.3366	0.7463
0859.png	23.5065	0.5867	25.1638	0.6519	25.4111	0.6376
0860.png	18.9909	0.5243	20.9424	0.6046	21.0890	0.5460
0861.png	21.9062	0.7015	23.0666	0.7331	22.4450	0.6575
0862.png	31.4043	0.7710	33.3899	0.8193	33.8169	0.8358
0863.png	32.4030	0.8743	33.2163	0.8901	33.5397	0.8989
0864.png	29.2618	0.7315	29.5629	0.7441	29.9787	0.7584
0865.png	25.5620	0.6874	27.3008	0.7355	27.7241	0.7415
0866.png	23.5277	0.7586	24.6347	0.7775	25.8672	0.7742
0867.png	27.9980	0.8355	27.9749	0.8289	27.2461	0.7793
0868.png	32.3374	0.9515	32.8231	0.9538	31.0152	0.9385
0869.png	21.3670	0.7162	23.9961	0.7752	24.1038	0.7516
0870.png	28.4011	0.8202	28.9490	0.8352	28.4314	0.8109
0871.png	32.1239	0.8877	32.1419	0.8870	31.1062	0.8672
0872.png	23.2962	0.6186	24.0228	0.6518	23.7664	0.6329
0873.png	23.3860	0.6930	24.0991	0.7078	24.0593	0.6696
0874.png	29.2800	0.7386	29.8537	0.7552	29.6915	0.7632
0875.png	23.8231	0.6115	24.4436	0.6395	23.7341	0.5767

0876.png	20.6881	0.5084	22.0730	0.5563	22.5871	0.5173
0877.png	40.6411	0.9695	37.7871	0.9658	37.1151	0.9519
0878.png	31.9780	0.8717	32.3417	0.8783	30.4858	0.8582
0879.png	25.8066	0.8076	27.2196	0.8449	26.8162	0.8263
0880.png	30.7202	0.8681	31.9407	0.8867	32.0537	0.8825
0881.png	24.8388	0.7219	26.2397	0.7582	26.3996	0.7530
0882.png	33.3954	0.9414	33.6116	0.9483	33.0095	0.9468
0883.png	24.1002	0.6140	25.9441	0.6936	26.7901	0.7052
0884.png	25.8666	0.7170	26.5002	0.7331	26.0110	0.6854
0885.png	22.0658	0.5258	22.7152	0.5413	23.2744	0.5198
0886.png	36.2921	0.8943	36.3472	0.9151	35.5178	0.9243
0887.png	23.3213	0.6841	24.6060	0.7230	24.8823	0.7058
0888.png	30.9395	0.8706	30.9254	0.8748	29.2474	0.8471
0889.png	31.1801	0.8979	31.3694	0.9001	29.0479	0.8586
0890.png	23.3122	0.6692	25.0691	0.7160	25.9357	0.7101
0891.png	26.1284	0.7794	27.0759	0.8081	25.1940	0.7287
0892.png	31.3492	0.9225	30.6931	0.9153	26.7129	0.8058
0893.png	32.4024	0.7688	32.8309	0.7961	32.6685	0.8290
0894.png	31.2349	0.8667	31.6920	0.8751	30.8588	0.8665
0895.png	18.3983	0.4367	20.0216	0.5169	21.1143	0.5075
0896.png	38.9269	0.9407	38.1979	0.9417	35.1900	0.9451
0897.png	21.4364	0.6517	22.2894	0.6828	22.0826	0.6335
0898.png	34.0116	0.8866	33.8795	0.8927	33.2465	0.9107
0899.png	31.4314	0.8949	31.3864	0.8988	27.6622	0.8399
0900.png	27.2395	0.8264	27.9630	0.8411	25.3620	0.7831
Average	28.1656	0.7758	28.8227	0.7992	28.2130	0.7772
StDev	5.0861	0.1225	4.3954	0.1049	4.2515	0.1115

Table A.4: Urban100 Dataset Metrics

Image	psnr esrgan	ssim esrgan	psnr esrgan_d	ssim esrgan_d	psnr bicubic	ssim bicubic
img_001.png	26.6902	0.7118	26.9347	0.7335	25.1822	0.6705
img_002.png	25.3936	0.7785	26.0426	0.7909	24.2637	0.6816
img_003.png	21.6078	0.5886	22.9195	0.6441	22.0475	0.5535
img_004.png	22.0623	0.7960	22.7941	0.8094	21.1763	0.6859
img_005.png	26.9093	0.9399	27.0751	0.9354	23.4667	0.8362
img_006.png	21.8853	0.6009	22.1023	0.6080	20.5841	0.4966
img_007.png	27.9960	0.8232	28.1584	0.8240	26.3706	0.7769
img_008.png	19.2042	0.5703	20.4203	0.6216	19.7797	0.4972
img_009.png	30.9001	0.8910	33.2392	0.9086	30.9309	0.8556
img_010.png	26.7871	0.8631	26.8709	0.8627	23.6399	0.7880

img_011.png	18.3106	0.8199	17.7042	0.7854	16.2294	0.6165
img_012.png	22.6866	0.6993	23.3000	0.7086	22.5874	0.6182
img_013.png	28.4204	0.8130	28.6667	0.8127	26.5810	0.7290
img_014.png	20.3015	0.6282	21.7369	0.6677	21.3466	0.5812
img_015.png	25.4273	0.7001	25.5494	0.7119	23.0683	0.6083
img_016.png	27.9265	0.8716	28.9381	0.8875	25.5835	0.8247
img_017.png	23.8628	0.7859	24.7354	0.8024	23.4767	0.7163
img_018.png	23.1429	0.5754	24.4086	0.6355	24.9327	0.6221
img_019.png	20.3738	0.7552	20.9267	0.7696	19.1178	0.6557
img_020.png	19.3687	0.6273	20.5940	0.6721	20.5072	0.5990
img_021.png	28.4340	0.7280	28.7182	0.7476	27.9450	0.7257
img_022.png	23.6166	0.6958	24.9670	0.7278	25.0632	0.6935
img_023.png	29.2113	0.8904	28.7492	0.8812	26.7114	0.8125
img_024.png	18.7377	0.6182	19.0130	0.6066	18.2317	0.4618
img_025.png	31.3624	0.8679	30.9980	0.8740	26.3282	0.8008
img_026.png	26.5624	0.6234	27.9743	0.6955	25.5460	0.6163
img_027.png	26.1609	0.7243	27.3699	0.7581	26.5922	0.7229
img_028.png	29.3169	0.8342	30.5467	0.8544	28.5242	0.8128
img_029.png	25.8028	0.8382	26.2068	0.8373	22.8349	0.7097
img_030.png	21.5797	0.7462	23.0382	0.7595	21.6965	0.6267
img_031.png	20.8693	0.7234	23.0430	0.7563	22.8897	0.6913
img_032.png	26.4854	0.7983	28.0272	0.8241	26.0698	0.7529
img_033.png	26.1020	0.7777	26.6127	0.7831	24.2066	0.6683
img_034.png	19.6558	0.4815	21.9783	0.5514	21.4576	0.4845
img_035.png	28.1907	0.8529	28.3929	0.8555	24.8390	0.7282
img_036.png	27.4510	0.8551	28.4060	0.8619	25.3231	0.7877
img_037.png	23.7236	0.7118	24.2978	0.7323	24.4041	0.7112
img_038.png	24.2987	0.5515	25.9159	0.6214	24.6600	0.5302
img_039.png	22.2821	0.7433	22.7137	0.7365	20.8732	0.5793
img_040.png	27.6460	0.9413	26.3793	0.9271	19.5650	0.6982
img_041.png	25.6927	0.8726	24.9924	0.8513	21.1237	0.6730
img_042.png	27.6597	0.8751	28.4344	0.8776	24.9828	0.7724
img_043.png	32.0584	0.9081	30.2803	0.9065	22.3204	0.7897
img_044.png	29.7177	0.8426	30.5559	0.8550	27.1234	0.7300
img_045.png	21.9319	0.6384	23.7393	0.6915	21.0383	0.5297
img_046.png	22.3273	0.7369	22.8516	0.7282	22.6408	0.6736
img_047.png	19.9192	0.7271	21.2092	0.7398	20.0559	0.6331
img_048.png	18.7465	0.7822	18.6333	0.7627	18.2035	0.6732
img_049.png	21.8645	0.6920	22.5158	0.7100	20.5645	0.5968
img_050.png	22.0587	0.6704	23.4183	0.7106	23.8370	0.6730
img_051.png	24.6375	0.7613	26.2655	0.8090	24.9398	0.7447

img_052.png	28.1806	0.8911	28.5132	0.8872	23.8809	0.7398
img_053.png	21.0288	0.6785	21.6323	0.6938	20.3184	0.5934
img_054.png	18.1760	0.5210	20.1956	0.6139	19.9824	0.5320
img_055.png	31.7127	0.9199	31.0192	0.9167	25.2141	0.7832
img_056.png	22.7490	0.7217	23.8126	0.7491	22.2338	0.6279
img_057.png	30.1292	0.8824	29.7900	0.8718	26.2733	0.7949
img_058.png	23.9815	0.8330	24.7141	0.8320	22.4292	0.7192
img_059.png	21.8723	0.7602	21.7721	0.7437	20.4123	0.6546
img_060.png	21.1359	0.5213	22.0095	0.5491	21.1592	0.4448
img_061.png	21.9901	0.6997	23.4140	0.7205	22.9575	0.6280
img_062.png	20.2182	0.7896	21.2166	0.7861	19.8813	0.6494
img_063.png	20.8341	0.5672	21.6550	0.6028	18.0186	0.4332
img_064.png	22.7578	0.6782	24.8395	0.7274	24.9894	0.7057
img_065.png	21.3841	0.5701	24.3614	0.6755	23.9728	0.6143
img_066.png	19.7081	0.4984	21.6397	0.5983	21.3892	0.5604
img_067.png	19.1773	0.8649	18.4530	0.8382	17.0027	0.7028
img_068.png	29.9848	0.8816	29.5383	0.8635	27.0998	0.7563
img_069.png	23.3014	0.7197	23.9184	0.7272	22.9501	0.6332
img_070.png	20.0491	0.4977	21.2245	0.5500	21.5177	0.5287
img_071.png	25.4782	0.5259	26.8437	0.5999	25.7486	0.5577
img_072.png	19.8495	0.8209	19.8512	0.8067	16.7794	0.5647
img_073.png	17.9287	0.4618	19.2637	0.5073	19.4965	0.4431
img_074.png	21.3008	0.6864	22.4612	0.6751	22.1460	0.5518
img_075.png	31.3194	0.8728	31.1031	0.8717	27.1623	0.8030
img_076.png	21.4163	0.6866	21.5663	0.6765	21.5817	0.6299
img_077.png	20.6966	0.5943	21.8126	0.6412	22.0612	0.6040
img_078.png	26.4892	0.7349	26.5385	0.7409	25.7737	0.6843
img_079.png	22.7805	0.5802	23.9281	0.6323	23.8857	0.5971
img_080.png	34.9517	0.9395	34.8233	0.9369	32.7619	0.9172
img_081.png	35.5990	0.9620	34.3010	0.9552	28.9009	0.8958
img_082.png	31.7236	0.9206	31.8690	0.9202	27.9107	0.8758
img_083.png	19.9690	0.6021	21.4200	0.6577	20.5272	0.5570
img_084.png	26.9044	0.7584	27.6211	0.7767	26.5648	0.7202
img_085.png	24.9076	0.8599	26.9340	0.8778	25.9609	0.8374
img_086.png	28.6797	0.8565	29.0010	0.8597	28.0629	0.8327
img_087.png	26.8672	0.8624	26.9468	0.8612	22.8736	0.7383
img_088.png	16.4433	0.4165	18.5277	0.5154	18.7289	0.4243
img_089.png	24.4892	0.6601	25.1820	0.6709	25.5212	0.6266
img_090.png	37.6160	0.9694	34.8351	0.9612	30.5650	0.8880
img_091.png	21.2630	0.5651	22.4393	0.6063	22.2415	0.5312
img_092.png	18.7474	0.6387	18.5860	0.6345	16.5669	0.4368

img_093.png	28.6435	0.9182	27.8657	0.9069	23.8949	0.8078
img_094.png	24.6813	0.6936	26.0715	0.7428	25.3860	0.7045
img_095.png	17.6241	0.3556	19.1690	0.4179	18.7552	0.2835
img_096.png	24.9132	0.8647	23.4660	0.8300	21.3317	0.6855
img_097.png	22.2975	0.6784	23.5001	0.7075	23.0667	0.6441
img_098.png	18.6368	0.4572	19.7739	0.4981	19.6659	0.3936
img_099.png	23.7976	0.7394	24.3417	0.7517	22.4513	0.5886
img_100.png	26.0125	0.7280	26.1591	0.7391	22.4289	0.6035
Average	24.3376	0.7326	24.9928	0.7515	23.2195	0.6605
StDev	4.3373	0.1376	3.9237	0.1162	3.2276	0.1221

Table A.5: Manga109 Dataset Metrics

Images	psnr esrgan	ssim esrgan	psnr esrgan_d	ssim esrgan_d	psnr bicubic	ssim bicubic
ARMS.png	25.7425	0.7034	27.1885	0.7661	25.8844	0.7124
AisazuNiha....png	27.0091	0.6847	28.9814	0.7534	28.6250	0.7618
Akkera...png	25.1996	0.7522	26.6100	0.7935	23.6514	0.7051
Akuhamu.png	34.5748	0.9188	34.6461	0.9252	31.2867	0.9005
Aosugiru...png	38.0252	0.9664	34.6385	0.9630	31.6283	0.9246
Appare...png	27.8588	0.8631	28.4760	0.8767	25.4545	0.7917
Arisa.png	31.4957	0.9297	31.3445	0.9324	26.1284	0.8501
BEMADER_P.png	25.7348	0.8451	26.8129	0.8726	22.9957	0.7825
Bakuretsu....png	29.4857	0.9025	29.2884	0.8980	25.1242	0.7931
Belmondo.png	28.6044	0.8593	28.7333	0.8664	24.2135	0.7742
BokuHa...png	28.0383	0.9343	28.1783	0.9289	22.9627	0.7960
BurariTessen..png	25.8433	0.8044	26.1420	0.8102	23.5272	0.7212
ByebyeC-BOY.png	31.7322	0.9433	31.9247	0.9415	25.9447	0.8615
Count3De...png	29.9241	0.8246	30.6318	0.8591	25.1055	0.8075
DollGun.png	27.3187	0.8902	27.6541	0.8894	24.5019	0.7900
Donburakokko.png	26.0148	0.8774	26.2952	0.8733	23.8575	0.7644
DualJustice.png	29.2739	0.9446	29.4714	0.9392	23.9113	0.8094
EienNoWith.png	32.8961	0.9697	33.5525	0.9704	28.6613	0.9319
EvaLady.png	29.9810	0.8486	30.1142	0.8622	28.3618	0.8552
Everyday....png	30.8018	0.9665	31.6970	0.9683	26.5579	0.9106
GOOD KISS..png	29.3739	0.8894	29.6668	0.8935	26.0010	0.8111
GakuenNoise.png	29.7076	0.9020	29.8489	0.9026	25.3817	0.8216
Garakutaya..png	28.3691	0.8705	28.7506	0.8800	25.4909	0.7982
GinNoChimera.png	35.8771	0.9554	34.7947	0.9467	28.2549	0.9047
Hamlet.png	27.0678	0.9342	27.8580	0.9410	22.8597	0.8384
Hanzai..png	29.2693	0.9185	29.7433	0.9190	25.4457	0.8360
Haruichiban..png	36.3174	0.9640	34.7623	0.9605	28.7689	0.9109

HarukaRefrain.png	30.6091	0.9078	30.7467	0.9081	27.4412	0.8616
HealingPlanet.png	32.1423	0.9357	31.3088	0.9277	26.3937	0.8824
HeiseiJimen.png	29.7817	0.9077	29.7558	0.9121	26.8734	0.8481
Highsch...1.png	25.1783	0.9174	25.3886	0.9096	20.4854	0.7267
Highsch...20.png	25.3971	0.8932	25.9573	0.8956	21.5133	0.7262
Hinagiku..png	31.0394	0.8573	31.0133	0.8575	28.2104	0.7999
Hisoka..png	24.9086	0.5747	27.1902	0.6844	25.2808	0.6583
Jangiri..png	24.2136	0.8143	24.6481	0.8248	22.8258	0.7417
JijiBaba..png	29.7683	0.8624	29.7727	0.8706	25.8716	0.8099
Joouari.png	26.6005	0.8333	27.1002	0.8445	24.2492	0.7592
Jyovolley.png	33.6775	0.9621	33.0065	0.9581	26.5689	0.8745
Karappo..png	33.1032	0.9406	32.7299	0.9377	28.8275	0.8846
KimiHaBoku..png	26.1190	0.8359	27.0892	0.8704	23.0058	0.7001
KoukouNo..png	28.3517	0.7713	28.5520	0.7910	25.6415	0.7242
KuroidoGanka.png	30.5290	0.8669	30.8075	0.8699	26.3343	0.8040
Kyokugen..png	18.5281	0.6425	20.3030	0.7044	18.7510	0.5535
Lancelot..png	33.5754	0.9724	33.5295	0.9704	28.2753	0.9010
LoveHina.1.png	24.2816	0.9056	25.5070	0.9135	22.1687	0.8384
LoveHina.14.png	26.6536	0.8882	26.6829	0.8809	23.7927	0.7512
MAD STONE.png	32.9793	0.9289	32.5435	0.9257	29.6204	0.8658
MadouTaiga.png	24.6648	0.7780	26.1987	0.8212	22.9108	0.7321
MagicStar..png	27.2647	0.8959	26.9555	0.8864	24.1139	0.7656
Magician..png	24.3822	0.6326	26.3355	0.7301	24.1939	0.6699
MariaSama..png	28.7981	0.8921	29.3929	0.9145	24.4341	0.8416
MayaNoAkai..png	31.6798	0.9206	31.9489	0.9174	27.5576	0.8515
MemorySeijin.png	34.0752	0.8389	34.6199	0.8606	34.5672	0.8832
MeteoSan..png	34.2051	0.9504	33.4413	0.9456	28.4565	0.8797
MiraiSan.png	19.3879	0.5638	21.4523	0.6628	21.2184	0.5731
Misutenaide..png	30.5589	0.9319	31.5762	0.9452	26.9803	0.8906
MoeruOnisan.1.png	29.8125	0.9142	29.2183	0.9133	23.7972	0.8208
MoeruOnisan.19.png	30.4219	0.9221	29.9784	0.9180	25.3320	0.8013
Momoyama..png	22.1791	0.7386	23.2765	0.7566	22.3911	0.6433
Mukoukizu..png	31.2786	0.9520	32.1964	0.9569	26.7398	0.9070
MutekiBouken..png	30.3545	0.9285	30.1447	0.9236	24.9973	0.8214
Nekodama.png	29.5983	0.9377	29.7046	0.9323	24.2947	0.8099
NichijouSoup.png	33.5582	0.9521	33.2098	0.9475	27.4336	0.8761
Ningyoushi.png	32.1777	0.8829	32.8275	0.8938	29.1522	0.8545
OL_Lunch.png	24.5676	0.9185	25.3886	0.9277	21.4951	0.7824
OhWarera..png	30.5210	0.9148	30.4061	0.9123	25.8747	0.8394
PLANET7.png	31.0483	0.8743	30.8100	0.8785	26.0680	0.8084
Paraiso..png	30.0114	0.9056	29.7724	0.9064	25.6122	0.8091

Pikaru..png	26.9296	0.9034	27.0060	0.8972	23.4774	0.7597
Platinum..png	29.5006	0.9152	29.2532	0.9100	24.6752	0.8105
PrayerHa..png	27.0306	0.6844	27.2613	0.7011	26.1751	0.6437
PrismHeart.png	32.5380	0.9322	32.9624	0.9355	28.3189	0.8861
PsychoStaff.png	32.6471	0.9351	31.8159	0.9253	28.0069	0.8635
Raphael.png	26.0291	0.8236	26.8157	0.8445	23.7443	0.7722
ReveryEarth.png	27.2915	0.8477	28.2878	0.8650	25.3449	0.7683
RinToSite..png	28.8984	0.8426	29.2083	0.8740	25.6685	0.8763
RisingGirl.png	26.7136	0.7864	28.1469	0.8194	25.6620	0.7893
Saisoku.png	27.6794	0.7870	27.7822	0.8044	23.1976	0.7266
SaladDays.1.png	29.1187	0.9103	29.7436	0.9140	26.2889	0.8308
SaladDays.18.png	29.6208	0.9300	30.2043	0.9289	27.1565	0.8442
Samayoeru..png	27.7951	0.8294	28.6067	0.8428	25.3816	0.7425
Seisinki..png	30.1864	0.8625	30.1352	0.8655	26.9353	0.7985
Shimatte.1.png	30.0946	0.9413	29.5473	0.9358	24.1393	0.8428
Shimatte.26.png	25.7759	0.9108	26.1568	0.9167	21.5809	0.8113
SonokiDeABC.png	30.4168	0.9004	30.6324	0.8996	26.6256	0.8147
Syabondama..png	26.3235	0.8186	27.1229	0.8331	25.1495	0.7689
TaiyouNi..png	33.5007	0.8958	32.9340	0.9103	29.6261	0.8753
TapkunNo..png	32.0541	0.9350	31.8090	0.9355	27.0702	0.8748
Tasogare..png	28.8556	0.8700	29.4580	0.8900	25.3019	0.8268
Tennen..png	23.9195	0.7951	24.6533	0.8252	20.5708	0.6860
TensiNo..png	27.5056	0.8992	27.6170	0.8923	23.9624	0.7499
TetsuSan.png	26.3528	0.9453	27.1678	0.9399	21.3347	0.8413
Thats.0.png	18.3512	0.6337	20.3223	0.7054	18.6353	0.5409
Totteoki..png	32.2384	0.9289	32.5816	0.9338	26.9708	0.8703
Touta..png	24.8943	0.8352	25.3955	0.8459	21.8978	0.7386
Touyou..png	27.9182	0.8406	28.7286	0.8687	23.4442	0.7964
Tsubasa..png	30.0102	0.7851	30.9976	0.8227	29.4813	0.8154
UchiNo..png	28.2885	0.9598	28.6097	0.9574	22.7092	0.8459
UchuKigeki..png	24.6840	0.7045	24.6217	0.7083	23.5178	0.6155
UltraEleven.png	22.0018	0.8542	22.8988	0.8667	19.5407	0.6944
Unbalance..png	25.0354	0.8693	26.1814	0.8762	24.2792	0.7981
Wareware...png	15.6544	0.5011	21.2632	0.7557	20.6205	0.6300
YamatoNo..png	35.2190	0.9640	33.8721	0.9580	28.1395	0.9045
Yasasii..png	25.7735	0.8441	27.8177	0.8757	24.7216	0.8074
Youchien..png	31.8602	0.9708	31.9916	0.9684	23.6782	0.8293
Youma..png	28.0996	0.8504	28.0417	0.8495	25.0706	0.7518
YukiNo..png	35.6676	0.9118	34.7631	0.9096	31.5328	0.8932
YumeNo..png	28.6322	0.7689	29.4612	0.7988	28.2102	0.7806
Yumeiro..png	25.2879	0.8781	26.9772	0.8810	24.9653	0.7930

Average	28.6781	0.8654	29.0747	0.8792	25.3500	0.7986
StDev	3.8434	0.0934	3.1657	0.0687	2.7577	0.0798

Table A.6: Watch Dogs 10 Dataset Metrics

Image	psnr esrgan	ssim esrgan	psnr esrgan_d	ssim esrgan_d	psnr bicubic	ssim bicubic
watch_dogs_1.png	31.1146	0.7910	31.5027	0.8142	31.4949	0.8346
watch_dogs_10.png	34.6490	0.9145	34.2103	0.9141	33.5180	0.9068
watch_dogs_2.png	25.1810	0.6999	26.7378	0.7535	27.1716	0.7639
watch_dogs_3.png	35.7512	0.9250	35.2234	0.9271	34.7033	0.9255
watch_dogs_4.png	28.0655	0.7123	30.2209	0.7896	30.3665	0.8304
watch_dogs_5.png	30.1297	0.8041	31.5501	0.8529	31.1309	0.8774
watch_dogs_6.png	26.7023	0.7056	28.0183	0.7464	28.8241	0.7712
watch_dogs_7.png	31.4941	0.8230	32.1656	0.8452	31.8319	0.8571
watch_dogs_8.png	30.1005	0.7858	31.2004	0.8289	30.6911	0.8494
watch_dogs_9.png	32.7082	0.8846	32.8728	0.8911	32.7799	0.8854
Average	30.5896	0.8046	31.3702	0.8363	31.2512	0.8502
StDev	3.3315	0.0837	2.5798	0.0627	2.1968	0.0529