VIETNAM NATIONAL UNIVERSITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY
Faculty of Computer Science and Engineering

GRADUATION THESIS

ADVANCED GAN MODELS FOR SUPER-RESOLUTION PROBLEM

Major : Computer Science

Council : Computer Science 3 (English Program)
Instructor: Dr. Nguyen Duc Dung
Reviewer: Dr. Tran Tuan Anh
—o0o—
Students : Truong Minh Duy - 1652113
Nguyen Hoang Thuan - 1752054

Ho Chi Minh City, July 2021

Acknowledgement

We would love to show our deep and honest gratitude to our instructor Dr. Nguyen Duc Dung. His compassion and instructions are invaluable in this research.

We also sincerely thank all of the faculty's lecturers. Without their guidance, we cannot have ourselves equipped with enough knowledge to carry out this research. Not only that, they also taught us other important lessons such as research methodology and working manner which shapes us into who we are today.

Besides, we also feel grateful to our family and friends for being the moral support during all this time.

Finally, we wish them happiness, passion, and success in any path that they choose.

Students

Abstract

In recent years, it is undoubted that machine learning and its subset, deep learning, is a frontier in Artificial Intelligence research. Generative Adversarial Networks (GANs), an emergent subclass of deep learning, has attracted considerable public attention in the research area of unsupervised learning for its powerful data generation ability. This model can generate incredibly realistic images and obtain state-of-the-art results in many computer vision tasks. However, although the significant successes achieved to date, applying GANs in real-world problems is still challenging due to many reasons such as unstable training, the lack of reasonable evaluation metrics, the poor diversity of output image. Our research focuses on improving GAN models performance in a notoriously challenging ill-posed problem - single image super-resolution (SISR). Specifically, we inspect and analyze ESRGAN model, which is a seminal work in perceptual SISR field. During our research, we propose changes in both model architecture and learning strategy to further enhance the output in two directions: image quality and image diversity. At the end of this thesis, we obtain promising results in these two aspects.

Acronyms

ANN	Artificial Neural Network.
BRISQUE	Blind/Referenceless Image Spatial Quality Evaluator.
CNN	Convolutional Neural Network.
DCT	Discrete Cosine Transform.
DFT	Discrete Fourier Transform.
FFT	Fast Fourier Transform.
FID	Fréchet Inception Distance.
GANs	Generative Adversarial Networks.
HR	High-Resolution.
IQA	Image Quality Assessment.
LPIPS	Learned Perceptual Image Patch Similarity.
LR	Low-Resolution.
MOS	Mean Opinion Score.
MSE	Mean Square Error.
NIQE	Natural Image Quality Evaluator.
PSNR	Peak Signal-to-Noise Ratio.
ReLU	Rectified Linear Unit.
SISR	Single image super-resolution.
SR	Super-Resolution.
SSIM	Structural Similarity Index.
VAE	Variational Autoencoder.

Notations

$D_{\mathbf{KL}}(P\|Q)$	Kullerback-Leibler divergence of P and Q.
I^{HR}	A high-resolution image.
I^{LR}	A low-resolution image.
$[a, b]$	The real interval including a and b.
$\log(x)$	Natural logarithm of x.
$\mathbb{E}_{x\sim p}(f(x))$	Expectation of $f(x)$ with respect to the distribution p.
$p(x)$	A probability distribution over a random variable x, whose type has not been specified.
$x \sim p$	A random variable x follows a distribution p.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

One of the major research in the machine learning field is generative models. There are several reasons why this subject at the forefront of scientific research in recent times. First, generative models have a great capacity for handling and studying various kinds of data, especially with complex and unstructured data. Secondly, later advances in this area seem to be potential for generating synthesis data of higher quality and in greater quantities. The synthetic data, in short, is the artificial data containing the same statistical characteristics as its "real" counterpart. Because synthetic data are extremely valuable in many cases [112, 127, 94], many generative models have been developed to provide "nearly-real" data such as Variational Autoencoder (VAE) [64], deep auto-regressive network [83] or normalizing flow [65]. Among these techniques, GANs [38] are gaining more popularity because of their interesting idea and incredible results.

Inspired by the two-player game, in which one player (network) is the art forger who's trying to mimic pieces of art or realistic artworks; on the other hand, another player can be considered as an art inspector who is looking at a pile of real famous art and also this fake art that is forged by this art forger. Then, the second player tries to distinguish which ones are real and which ones are fake and send their feedback to the first player. Therefore, the quality of the image generated by the first network increase over time. Also, it is surprising that GANs can generate realistic images which easily fool even human as in Figure 1.1. The powerful data generation capacity allows GANs achieved state-of-the-art performance in many tasks, including computer vision [125], natural language processing [2], time series analysis [132], reinforcement learning [105], etc.

The incredible results of GANs do not mean that these networks do not suffer from their own problems. The two major issues are that training GANs is uneasy [29] and the results are difficult to evaluate [75]. Another possible problems are the imbalance between two player networks, underfitting, the diversity of the image generation, etc [125]. Although many variants of GAN has proposed, solving GAN's weaknesses is still an open research direction.

This research focuses on improving the performance of GANs. Instead of improving GANs in general, which requires heavy mathematics background, our approach is inspecting GAN-based approach for super-resolution problem. This problem is how to reproduce a higher-solution image from its low-resolution observation. The main rea-

Figure 1.1: The photo-realistic image generated by **GAN** [119]

son for our choice because super-resolution can be well-applied to some other computer vision tasks such as video enhancement, medical diagnosis, astronomical observation, etc [134].

By improving GAN-based approach for super-resolution, we hope to create a model that is versatile enough to apply for not only super-resolution but also other related fields. To achieve this target, we focus on two major problems of GAN model for image super-resolution: the quality [120] and the diversity of the output image [76]. Also, during our research, we try to apply some techniques to ease the training process, although it is not the main concentration.

1.2 Goal

The main objective of this research is to improve GANs for super-resolution and extend our results to other vision tasks such as image denoising. To achieve such goal, we plan to carry out the following tasks:

- Study GANs and their variants which are suitable for super-resolution problem.
- Train and evaluate existing GAN models.
- Based on the experiment results, devise and implement some development directions to improve the model performance
- Apply the model to other vision problems.

1.3 Scope

Although super-resolution is a wide and diverse field, in our thesis, we only concentrate on reconstructing photo-realistic images of natural scenes. The main reason is that working with photo-realistic images is very applicable in many real-life applications, and reconstructing photo-realistic image is gaining attention in the current research literature. Additionally, we do not consider all scale but only with the scale of **4x**.

In general, the super-resolution methods can be divided into two groups: single-image and multiple-image. However, we only consider single-image methods in this thesis as multiple images of the same high-resolution image are not always available.

1.4 Contributions

In this thesis, we propose two different ways for further improving the generated images of an existing GAN-based model. In particular, our contributions can be summarized as follows:

- On the first approach, we devise a novel learning strategy that consistently achieves better super-resolution image quality.
- By comprehensive experiments, we prove that using our loss enhances the learning ability not only in the spatial domain but also in the spectral domain.
- On the second approach, we design a diversify module which can be an add-on for any previous 1-to-1 super-resolution model to generate a distribution of fine-grained outputs
- Despite learning for super-resolution only, we show that our diversify model is potential for other vision tasks such as image denoising.

In the next section, we are investigating some SOTA[1] work on GAN-based solution for super-resolution problem and mentioned our target paper. In Chapter 3, we describe theoretical background required in this research. Then, we analyze some weaknesses of the baseline model as well as devise some ideas to further enhance the result quality. After that, the effectiveness of our method is proved by a series of experiments in Chapter 5. In the final chapter, we summarize our work and propose some future improvements.

[1] State-of-the-art

Chapter 2

Related Work

2.1 GAN-based approach for SISR

To the best of our knowledge, although there are many relevant surveys about GANs [125, 52, 40, 16, 44], and SISR [6, 58, 123, 28, 131], almost no surveys analyze GAN-based approach for SISR in detail. The previous GANs surveys mainly compare variant of GANs in some metrics: structure [125, 52, 16], loss function, application [125, 52, 40, 16, 44] and etc. Meanwhile, the majority of SISR surveys discuss not only the GAN-based approaches but together with various other types of networks such as linear networks, residual networks, recursive networks [6, 58, 123, 28, 131]. That is why we structure a taxonomy as in Figure 2.1. In summary, we categorize some recent results by three different main aspects: **target**, **approach** and **output diversity**.

Target-based classification: Almost GAN-based approaches for SISR aim to solve this problem end-to-end. In other words, the authors focus on super-resolving images in general rather than any specific types of image [69, 120, 84, 104, 78, 93, 9]. In general, those models can work with various type of images but their performance may vary depending on the kind of data used for training. On the other hand, there are other works only concentrating on a particular type of image. Taking DeepSEE [13] and Super-FAN [14] for examples, both papers only fit with super-resolving portrait

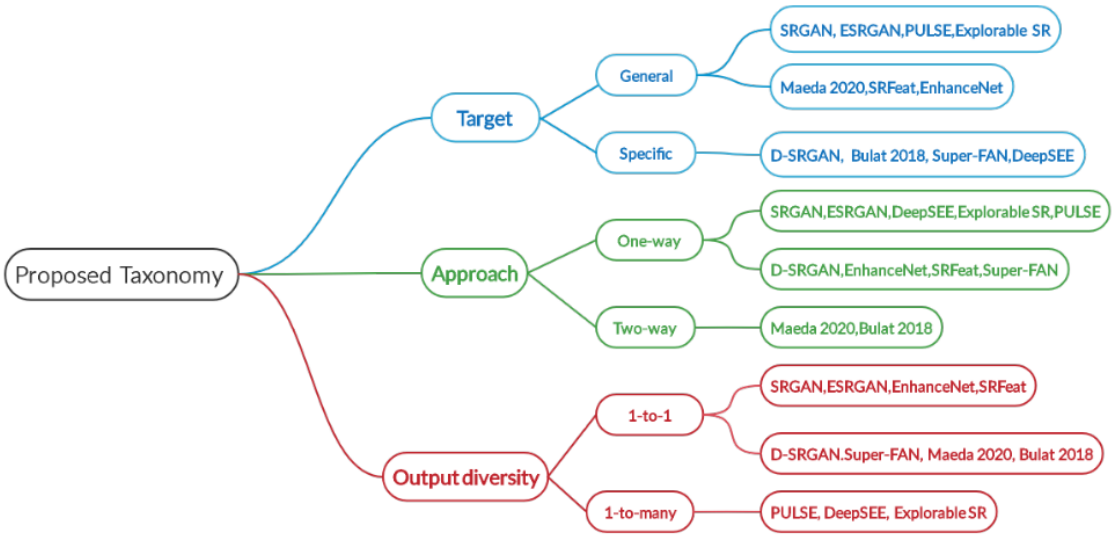


Figure 2.1: Our taxonomy for recent GAN-based approach for SISR

images, namely *face hallucination*. In detail, DeepSEE adds a semantic segmentation to guide their model provide a realistic image base on LR input, whereas Super-FAN applies a face alignment network alongside with the standard GAN network (based on SRGAN [69]). Furthermore, the work of Bulat et al. [15] concerned only face dataset in their experiments although they claim their technique can be extended to other kinds of images. Recently, Demiray et al. [24] have explored SRGAN for a new data type: digital elevations model (DEM).

Approach-based classification: The most common approach for SISR problem is solve this from one direction: low-to-high [69, 120, 13, 104, 93, 14, 9, 24], whereas an special work [84] try to do the opposite, learning from high-to-low. However, some authors claim this approach provides poor results when applying to real-world low-resolution images. The major drawback of those models is that they always pre-assume the simple downsampling operators (e.g bicubic, bilinear), which is rarely exist in real cases. To alleviate this problem, some recent research [15, 78] consider both low-to-high and high-to-low directions. These methods offer some advantages when applying in real images in which downsampling operation can be complicated or unknown. Moreover, they report good results in the case of unpaired data. However, these approaches trade-off more computing resources, as they normally use more than one pair of generator-discriminator to learn both directions.

Output diversity classification: As a matter of fact, SISR is 1-to-many given its ill-posed nature, i.e. many high-resolution images can be downscaled to the same low-resolution image. However, GANs is unstable and hard to train; as a result, most GAN-based approaches [69, 120, 104, 93, 14, 24] for SISR only try to learn a 1-to-1 mapping from one low-resolution image to one high-resolution output. There are some GAN-based research [13, 9, 84] apply their model for 1-to-many super-resolution. As for DeepSEE and ExplorableSR [13, 9], they introduce an output controllable module so that user can modify the output into many different high-resolution outputs per one low-resolution input; whereas, PULSE [84] follows an entirely different approach by searching in the latent space of a pre-trained GAN to produce the high-resolution image that is most consistent with the low-resolution reference image. In our perspective, DeepSEE and ExplorableSR seem to be a "work-around" solution as they allow the user to post-edit the output and do not learn a high-resolution distribution conditioned on the low-resolution image. Meanwhile, PULSE has a vital weakness that it highly depends on an external generative model.

2.2 Accuracy-driven models and perceptual-driven models

In this section, the model does not limit to GAN-based model but rather the "general model". Here, we review some recent single image super-resolution models and classify them based on quantitative metrics: accuracy-driven models and perceptual-driven.

To begin with, the evaluation metric can be divided into two groups: accuracy metrics and perceptual metrics. Accuracy metrics including two most common metrics for SISR: PSNR and SSIM which aim to computer the pixel-wise dissimilarity. Accuracy metrics normally are sensitive to distortion, but uncorrelated with human perception [11, 8]. On the other hand, perceptual metrics which is proved correlate better with

human opinion use deep neural networks to evaluate the score [11, 69]. We will cover more details and give example for each kind of metric in section 3.2.4.

Based on the above classification, in our thesis, we further classify super-resolution model into two categories:

- **Accuracy-driven models:** If the model only evaluates based on accuracy metrics, we consider this model as the accuracy-driven model. Most of the previous approaches is accuracy-driven methods. Publishing in 2014, SRCNN [18] proposed by Dong et al. use three convolution neural network layers to output the high-resolution image from its low-resolution counterparts. Later, some powerful architecture such as residual network [69], recursive network [62] or residual dense network [144] are applied to further improve SR performance. Recently, as a pioneer, Zhang et al. [143] combine attention mechanism and existing works to achieve a promising result. Following Zhang, other authors proposed more novel attention: holistic attention [89], second-order attention [23], two-stage attentive network [137], etc. Some other interesting approaches can be considered are: learn image downscaling [111], feedback framework [70], etc.

- **Perceptual-driven models:** If evaluation metrics of the model contain at least one perceptual metric, we consider this model as the perceptual-driven model. Due to the lack of powerful perceptual image quality assessment, the perceptual-driven approach receives much less attention for a long time. Currently, the majority of perceptual-driven methods is the GAN-based approach [69, 120, 93]. About GAN-based methods, many variants were proposed to further enhance quality results such as: use additional information from segmentation map [95], use U-Net based discriminator [55], use pre-train model [17], etc. About non-GAN approach, recently, Lugmayr design a novel architecture use normalizing flow [76] and obtain a comparable result with GAN-based models. On the one hand, perceptual-driven models can produce more pleasing pictures, especially in extreme super-resolution (larger than x8) [13, 17, 55]. On the other hand, images obtain by this type of model normally contain an unnatural noise.

2.3 Some recent noticeable results

In this section, we present some recent noticeable results which we will use in our experiments.

2.3.1 Recent IQA model selection

In our experiments, we consider two noticeable image-quality assessment models: LPIPS [141] and DISTS [26]. Summarized information of other models can be found in [60].

Learned Perceptual Image Patch Similarity (LPIPS) [141] compares the similarity of deep embedding of two images. At the beginning, the authors prove the deep features obtained by pass through images into the neural network correlate well with human opinion. Unlike the shallow feature in the traditional method only capture the whole image, deep representation can successfully capture the spatial and temporal dependencies in an image. We summarize the pipeline of LPIPS in section 3.2.4.5.

Figure 2.2: LPIPS [141] and DISTS [26] comparision. From left to right: **(a)** A grass image, **(b)** the same image, distorted by JPEG compression, **(c)** Resampling of the same grass as in (a). **Which image ((b) or (c)) is "closer" to image ((a))?** LPIPS choose (b) and DPIPS choose (c). Figure from DISTS [26]

DISTS [26] is a "deeper" version of LPIPS. DISTS tries to obtain a texture resampling tolerance ability. To achieve this goal, instead of only compute the spatial average of feature maps like LPIPS, they also compare the structural components. In other words, they replace the Euclidean distance in LPIPS with SSIM-like structure similarity measurements. To further improve performance, they propose a novel loss for the training process.

To illustrate the difference between LPIPS and DISTS, please see Figure 2.2. LPIPS predict image (b) is closer to referenced image (a), while DISTS chooses image (c).

2.3.2 Recent GANs

2.3.2.1 Relativistic GAN and its variants

In 2018, Martineau proposed a new type of discriminator: relativistic discriminator [57]. First, we take a glance at standard GAN [38]. In original GAN, the generator tries to increase the probability of generated data being real, while the discriminator evaluates received input (either fake or real data) is real or not. Martineau realizes the key missing property in standard GAN is that the generator only benefits from generated data. In relativistic GAN, both real and fake data equally take part in the generator and discriminator's learning procedure. Moreover, by mathematical formulation, they prove a standard GAN is just a specific case of a relativistic GAN [57].

In 2020, Martineau published another paper relate to relativistic GAN [56]. A new paper provides the mathematical foundations and devises more variants of Relativistic GAN. For convenience, the author concentrates on the critic score rather than the discriminator as in the previous paper. In short, the critic is the discriminator without the final activation layer. ($D(x) = a(C(x))$ where a is the activation layer, D is the discriminator, C is the critic). This notation is very similar to some prior works relate to Wasserstein GAN [7, 41]. We can interpret the critic as scoring the realistic rate of the input (instead of probability).

Although Martineau offers four variants of relativistic GAN in the later paper [56], their experiments show two most powerful model is Relativistic average GAN (RaGAN) and Relativistic centered GAN (RcGAN). As a result, we only focus on RaGAN and RcGAN in our experiments. Next, we introduce one definition and one theorem from [56], as those are crucial to build the formula of RaGAN and RcGAN.

Definition 2.1. [56] Let $\mathbb{P}$ and $\mathbb{Q}$ be probability distributions and S be the set of

all probability distributions with common support. A function $D : (S, S) \to \mathbb{R}_{>0}$ is a divergence if it respects the following two conditions:

$$D(\mathbb{P}, \mathbb{Q}) \geq 0$$

$$D(\mathbb{P}, \mathbb{Q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}$$

It is obvious from the formula that divergences are the gap between two probability distributions. During the training procedure, the distribution of real data is constant and the efficient critic/discriminator must reduce the divergences over time.

Theorem 2.1. [56] Let $f : \mathbb{R} \to \mathbb{R}$ be a concave function such that $f(0) = 0$, f is differentiable at 0, $f'(0) \neq 0$, $\sup_x(f(x)) = M > 0$ and $\arg(\sup_x(f(x))) > 0$. Let $\mathbb{P}$ and $\mathbb{Q}$ be probability distributions with support χ. Let $\mathbb{M} = \frac{1}{2}\mathbb{P} + \frac{1}{2}\mathbb{Q}$. Then, we have:

$$D_f^{Ra}(\mathbb{P}, \mathbb{Q}) = \sup_{C:\chi\to\mathbb{R}} \left(\mathop{\mathbb{E}}_{x\sim\mathbb{P}} \left[f\left(C(x) - \mathop{\mathbb{E}}_{y\sim\mathbb{Q}} (C(y)) \right) \right] + \mathop{\mathbb{E}}_{y\sim\mathbb{Q}} \left[f\left(\mathop{\mathbb{E}}_{x\sim\mathbb{P}} (C(x)) - C(y) \right) \right] \right)$$

$$D_f^{Rc}(\mathbb{P}, \mathbb{Q}) = \sup_{C:\chi\to\mathbb{R}} \left(\mathop{\mathbb{E}}_{x\sim\mathbb{P}} \left[f\left(C(x) - \mathop{\mathbb{E}}_{m\sim\mathbb{M}} (C(m)) \right) \right] + \mathop{\mathbb{E}}_{y\sim\mathbb{Q}} \left[f\left(\mathop{\mathbb{E}}_{m\sim\mathbb{M}} (C(m)) - C(y) \right) \right] \right)$$

are divergences

In theorem 2.1, $D_f^{Ra}(\mathbb{P}, \mathbb{Q})$ and $D_f^{Rc}(\mathbb{P}, \mathbb{Q})$ correspond to RaGAN and RcGAN, respectively. Also, **sup** stands for supremum or least upper bound. Further details and proofs can be found in [56].

Moreover, Martineau provides some examples about function f which is satisfied all conditions in theorem 2.1. All concave function f in Figure 2.3 is the appropriate choice for relativistic divergences. This is also the function used in original GAN [38], LSGAN [80] and HingeGAN [88] (note that SGAN mentioned in paper [57] is the original GAN [38]). The mathematical formula for three above function respectively are:

$$f_S(z) = \log(sigmoid(z)) + \log(2), \tag{2.1}$$

$$f_{LS}(z) = -(z-1)^2 + 1, \tag{2.2}$$

$$f_{Hinge}(z) = -\max(0, 1-z) + 1, \tag{2.3}$$

By combining theorem 2.1 with equations (2.1), (2.2), (2.3) and modifying little for suitable with super-resolution problem, we can obtain many variants of Relativistic GAN for our task.

In those equations below, I^{LR} denotes the low-resolution image and I^{HR} stands for the referenced high-resolution image. Next, $G(I^{LR})$ is the high-resolution image generated by the model and B is the batch size. Also, σ denotes the sigmoid function and $C(x)$ is the non-transformed discriminator output.

Combining $D_f^{Ra}(\mathbb{P}, \mathbb{Q})$ with equation (2.1) and follow the instruction from [57], we obtain the generator and discriminator loss for RaGAN:

$$\begin{aligned} L_G^{RaGAN} &= -\frac{1}{B}\sum_{b=1}^{B} \left(\log\left(1 - D_{RaGAN}(I^{HR}, G(I^{LR}))\right) + \log D_{RaGAN}(G(I^{LR}), I^{HR}) \right) \\ L_D^{RaGAN} &= -\frac{1}{B}\sum_{b=1}^{B} \left(\log D_{RaGAN}(I^{HR}, G(I^{LR})) + \log\left(1 - D_{RaGAN}(G(I^{LR}), I^{HR})\right) \right) \end{aligned} \tag{2.4}$$

where $D_{RaGAN}(x, y) = \sigma(C(x) - \frac{1}{B}\sum_{b=1}^{B} C(y))$.

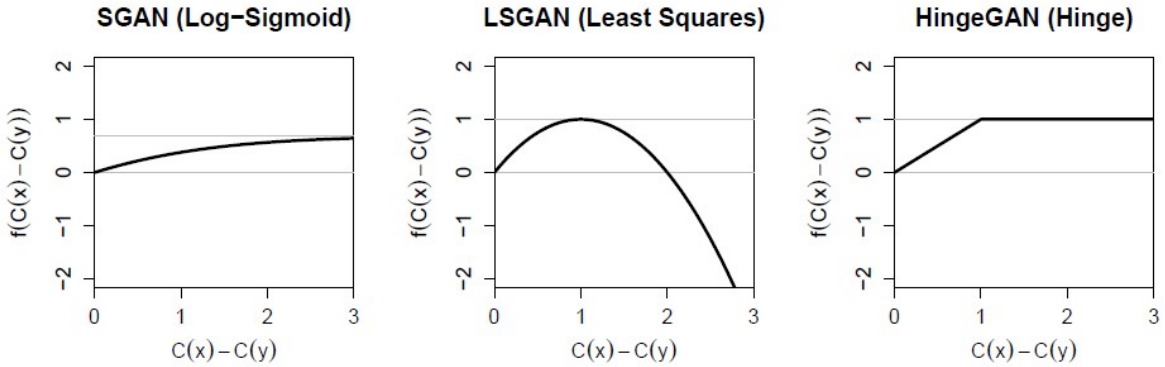


Figure 2.3: Subtypes of divergences. Plot of f with respect to the critic's difference (CD) using three appropriate choices of f for relativistic divergences. The bottom gray line represents $f(0) = 0$; the divergence is zero if all CDs are zero. The above gray line represents the maximum of f; the divergence is maximized if all CDs leads to that maximum. Figure from [56]

Combining $D_f^{Ra}(\mathbb{P}, \mathbb{Q})$ with equation (2.2) and follow the instruction from [57], we obtain the generator and discriminator loss for RaLS:

$$\begin{aligned} L_G^{RaLS} &= -\frac{1}{B}\sum_{b=1}^{B}\left(\left(D_{RaLS}(I^{HR}, G(I^{LR})) + 1\right)^2 + \left(D_{RaLS}(G(I^{LR}), I^{HR}) - 1\right)^2\right) \\ L_D^{RaLS} &= -\frac{1}{B}\sum_{b=1}^{B}\left(\left(D_{RaLS}(I^{HR}, G(I^{LR})) - 1\right)^2 + \left(D_{RaLS}(G(I^{LR}), I^{HR}) + 1\right)^2\right) \end{aligned} \tag{2.5}$$

where $D_{RaLS}(x, y) = C(x) - \frac{1}{B}\sum_{b=1}^{B} C(y)$.

Combining $D_f^{Ra}(\mathbb{P}, \mathbb{Q})$ with equation (2.3) and follow the instruction from [57], we obtain the generator and discriminator loss for RaHinge:

$$\begin{aligned} L_G^{RaHinge} = -\frac{1}{B}\sum_{b=1}^{B}\Big(&\max\left(0, 1 - D_{RaHinge}(I^{HR}, G(I^{LR}))\right) + \\ &\max\left(0, 1 + D_{RaHinge}(G(I^{LR}), I^{HR})\right)\Big) \\ L_D^{RaHinge} = -\frac{1}{B}\sum_{b=1}^{B}\Big(&\max\left(0, 1 + D_{RaHinge}(I^{HR}, G(I^{LR}))\right) + \\ &\max\left(0, 1 - D_{RaHinge}(G(I^{LR}), I^{HR})\right)\Big) \end{aligned} \tag{2.6}$$

where $D_{RaHinge}(x, y) = C(x) - \frac{1}{B}\sum_{b=1}^{B} C(y)$.

For relativistic centered GAN, we define $C_m(x, y) = \frac{1}{2B}\sum_{b=1}^{B}(C(x) + C(y))$. Combining $D_f^{Rc}(\mathbb{P}, \mathbb{Q})$ with equation (2.1) and follow the instruction from [57], we obtain the generator and discriminator loss for RcGAN:

$$\begin{aligned} L_G^{RcGAN} &= -\frac{1}{B}\sum_{b=1}^{B}\left(\log\left(1 - D_{RcGAN}(I^{HR}, G(I^{LR}))\right) + \log D_{RcGAN}(G(I^{LR}), I^{HR})\right) \\ L_D^{RcGAN} &= -\frac{1}{B}\sum_{b=1}^{B}\left(\log D_{RcGAN}(I^{HR}, G(I^{LR})) + \log\left(1 - D_{RcGAN}(G(I^{LR}), I^{HR})\right)\right) \end{aligned} \tag{2.7}$$

where $D_{RcGAN}(x, y) = \sigma(C(x) - C_m(x, y))$.

Combining $D_f^{Rc}(\mathbb{P}, \mathbb{Q})$ with equation (2.2) and follow the instruction from [57], we obtain the generator and discriminator loss for RcLS:

$$\begin{aligned} L_G^{RcLS} &= -\frac{1}{B}\sum_{b=1}^{B}\Big(\big(D_{RcLS}(I^{HR}, G(I^{LR})) + 1\big)^2 + \big(D_{RcLS}(G(I^{LR}), I^{HR}) - 1\big)^2\Big) \\ L_D^{RcLS} &= -\frac{1}{B}\sum_{b=1}^{B}\Big(\big(D_{RcLS}(I^{HR}, G(I^{LR})) - 1\big)^2 + \big(D_{RcLS}(G(I^{LR}), I^{HR}) + 1\big)^2\Big) \end{aligned} \tag{2.8}$$

where $D_{RcLS}(x, y) = C(x) - C_m(x, y)$.

Combining $D_f^{Rc}(\mathbb{P}, \mathbb{Q})$ with equation (2.3) and follow the instruction from [57], we obtain the generator and discriminator loss for RcHinge:

$$\begin{aligned} L_G^{RcHinge} = -\frac{1}{B}\sum_{b=1}^{B}\Big(&\max\big(0, 1 - D_{RcHinge}(I^{HR}, G(I^{LR}))\big) + \\ &\max\big(0, 1 + D_{RcHinge}(G(I^{LR}), I^{HR})\big)\Big) \\ L_D^{RcHinge} = -\frac{1}{B}\sum_{b=1}^{B}\Big(&\max\big(0, 1 + D_{RcHinge}(I^{HR}, G(I^{LR}))\big) + \\ &\max\big(0, 1 - D_{RcHinge}(G(I^{LR}), I^{HR})\big)\Big) \end{aligned} \tag{2.9}$$

where $D_{RcHinge}(x, y) = C(x) - C_m(x, y)$.

Two equations in (2.4) are the main functions in the adversarial loss in ESRGAN [120]. We also note that, in equations (2.1), (2.2) and (2.3), we have some constants such as log(2) or 1. However, as we use the minus operator, those constants are eliminated.

To sum up, we have already quoted some most crucial parts about the Relativistic GAN [57, 56]. Not only that, but we also applied it into the super-resolution problem. All variants of GANSs in this section will be covered in our experiments.

2.3.2.2 Wasserstein GAN and its variants

As observed from our earlier experiments in section 5.1.2.2, we find out that the ESRGAN's discriminator is sensitive to hyperparameters and hard to train, even though it uses RaGAN which is an innovative and powerful technique [57]. Hence, we discover some other discriminators to avoid this problem. Farnia and Ozdaglar [29] prove that GANs minimax games may not have any Nash equilibrium. They also show that recent Wasserstein GAN [7] can provide stable learning curves and better performance because WGAN can effectively reach a proximal equilibrium. However, it is a challenging task to approximate the K-Lipschitz constraint which is required by the Wasserstein-1 metric. Gulrajani et al. [41] sidestep this problem by introducing a new GAN loss, namely WGAN-GP. Both WGAN and WGAN-GP seem promising to enhance ESRGAN's performance. Similar with previous section, we will introduce some main points about WGAN and WGAN-GP.

Although we mainly use WGAN-GP [41], some concept from WGAN [7] is necessary to fully understand our work. In Wasserstein GAN, one of the core concepts is the

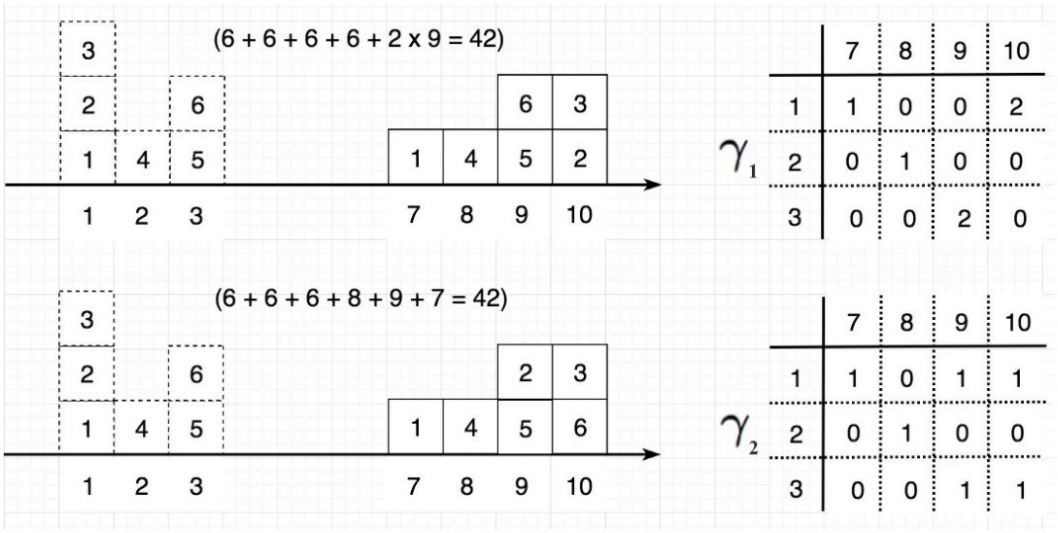


Figure 2.4: Wasserstein-1 distance illustration by moving box example. **Left image:** The position of box begin and after moving. **Right image:** Moving plan table. Figure from [50]

Earth-Mover distance or Wasserstein-1 metric:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi} \sum_{x,y} \|x - y\| \gamma(x, y) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} E_{(x,y)\sim\gamma} \|x - y\| \tag{2.10}$$

In equation (2.10), $\mathbb{P}_r$ and $\mathbb{P}_g$ are two probability distributions, where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. If we consider more than one random variable (e.g X and Y), the joint distribution of X and Y and the distribution of each random variable should not be confused. In fact, the marginal distribution of each random variable is the probability distribution of this random variable itself. To illustrate the Wasserstein-1 distance, we borrow the visual image from [50] (Figure 2.4)

We can think of the probability distribution as the heap of a certain amount of boxes. In Figure 2.4, Hui considers an example of moving boxes. Assume the original distribution is the box in columns 1, 2 and 3, how we can transfer all the original boxes to the new distribution (in columns 7, 8, 9 and 10). Let consider two different moving plans in Figure 2.4. The left image describes the initial and final position of all boxes, while the right image shows how the box is moved. For example, in plan γ_1, we have $\gamma_1(1, 10) = 2$ means that we moved 2 boxes from column 1 to column 10 (box 2 and box 3). To further simplify the calculation, if we move one box from column u to v, the transport cost is computed just by $|u - v|$. For example, the transport cost of moving box 1 from column 1 to column 7 is 6. In this example, two plans γ_1 and γ_2 share the same transport cost (the calculation inside the figure). However, in many cases, the cost can differ (example can be found in [50]). In general, the Wasserstein-1 distance is defined as the minimum possible transport cost in transforming the data distribution p to the data distribution q. In the mathematical formula, this task is denoted by **inf** operator (infimum - greatest lower bound) (as can be seen in equation (2.10)).

To apply Wasserstein distance in GAN, Arjovsky et al. [7] use the Kantorovich-Rubinstein duality to obtain more simple equation:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]\right) \tag{2.11}$$

where **sup** is the supremum or least upper bound. To understand about the function f, we consider another definition

Definition 2.2. [7] A real function $f : \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous if

$$|f(x_1) - f(x_2)| \leq K\,|x_1 - x_2|\,, \forall x_1, x_2 \in \mathbb{R}$$

In this case, with constant K, we denote $\|f\|_L \leq K$

If we rewrite the equation in the above definition as $\frac{|f(x_1)-f(x_2)|}{|x_1-x_2|} \leq K$, we can recognize Lipschitz constraint limits how fast the function change. Following this definition, the function f in equation (2.11) must be 1-Lipschitz function. To approximate this function, Arjovsky et al. removes the final activation layer and rename this to critic. Finally we achieve a new type of GAN: WGAN. Compare to original GAN [38], WGAN have two advantages: 1) reduce mode collape 2). reduce vanishing gradients - WGAN's generator can learn although critic is very powerful and vice versa. We borrow the Figure 2.5 to illustrate the second point. Further analysis and experiments can be found in [7].

However, enforce the Lipschitz constrant during the training procedure is an arduous task. In the original work, Arjovsky et al. [7] used weight clipping technique, but they admitted this technique is awful. In their experiments, WGAN still produce bad quality image and fail to converge in some cases. Gulrajani et al. [41] further improve the training of WGAN by using another technique - gradient penalty. To use this technique, they prove one theorem:

Theorem 2.2. [41] Let $\mathbb{P}_r$ and $\mathbb{P}_g$ are two probability distribution. Denote 1-Lipschitz function f^* is the optimal solution of : $\max_{\|f\|_L \leq 1}(\mathbb{E}_{y\sim\mathbb{P}_r}[f(y)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)])$. Then, f^* has gradient norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$.

Theorem 2.2 claims any points interpolated between real image and fake image should have a gradient norm is 1. We also note that above theorem is just a corollary of original theorem in [41]. By applying this theorem into WGAN, Gulrajani et al. proposed new algorithm: WGAN-GP (see algorithm 1).

Similar to WGAN, some experiments in WGAN-GP prove this loss enhances training stability [41, 50]. Although the image quality obtained from WGAN-GP slightly worse compared to the original GAN, it achieves more stable results during the training phase. In case the network architecture is not powerful, WGAN-GP can learn and produce acceptable outputs while original GAN fails to converge. Figure 2.6 demonstrates this point by comparing the inception score (higher is better) of four types of model: DCGAN([97]), WGAN-GP use RMSProp optimizer, WGAN-GP use Adam optimizer and WGAN (approximate function by weight clipping). As WGAN-GP helps the model easier to train, we can simply enhance it by using a deeper architecture such as residual network or dense network.

Applying WGAN-GP in super-resolution, we obtain a new equation

$$\begin{aligned} L_G^W &= -\frac{1}{B}\sum_{b=1}^{B} D_W(G(I^{LR})) \\ L_D^W &= \frac{1}{B}\sum_{b=1}^{B} D_W(G(I^{LR})) - D_W(I^{HR}) + \lambda_g(\left\|\nabla_{\hat{I}} D_W(\hat{I})\right\|_2 - 1)^2 \end{aligned} \tag{2.12}$$

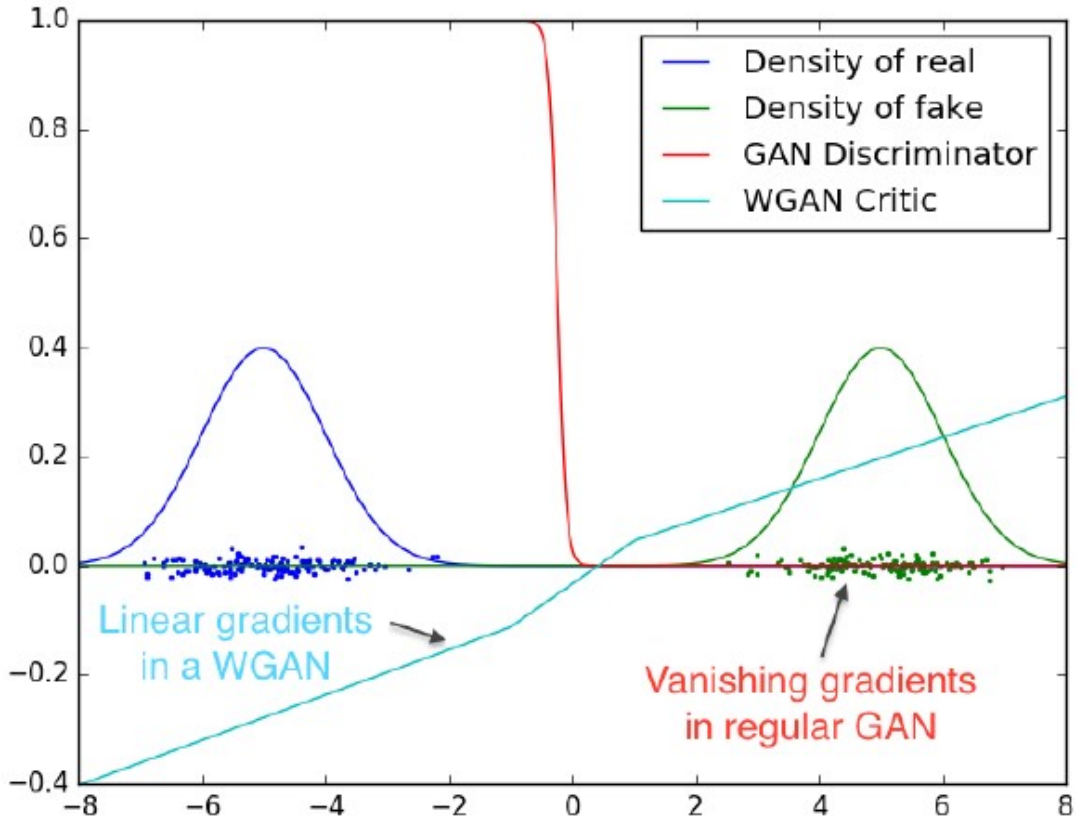


Figure 2.5: Gradient experiments between WGAN and normal GAN. Optimal discriminator and critic when learning to differentiate two Gaussians. Original GAN suffers from gradient vanishing, while WGAN can learn during the experiment. Figure from [7]

Algorithm 1 WGAN-GP: WGAN with gradient penalty

Require: The gradient penalty coefficient λ_g, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyper-parameters α, β_1 β_2.

Require: initial critic parameters ω_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{critic}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $x \sim \mathbb{P}_r$, latent variable $z \sim p(z)$, a random number $\varepsilon \sim U[0, 1]$.
      $\tilde{x} \leftarrow G_\theta(z)$
      $\hat{x} \leftarrow \varepsilon x + (1 - \varepsilon)\tilde{x}$
      $L^{(i)} \leftarrow D_\omega(\tilde{(x)}) - D_\omega(x) + \lambda_g(\|\nabla_{\hat{x}} D_\omega(\hat{x})\|_2 - 1)^2$
    end for
    $\omega \leftarrow Adam(\nabla_\omega \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, \omega, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$.
  $\theta \leftarrow Adam(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_\omega(G_\theta(z)), \theta, \alpha, \beta_1, \beta_2)$
end while

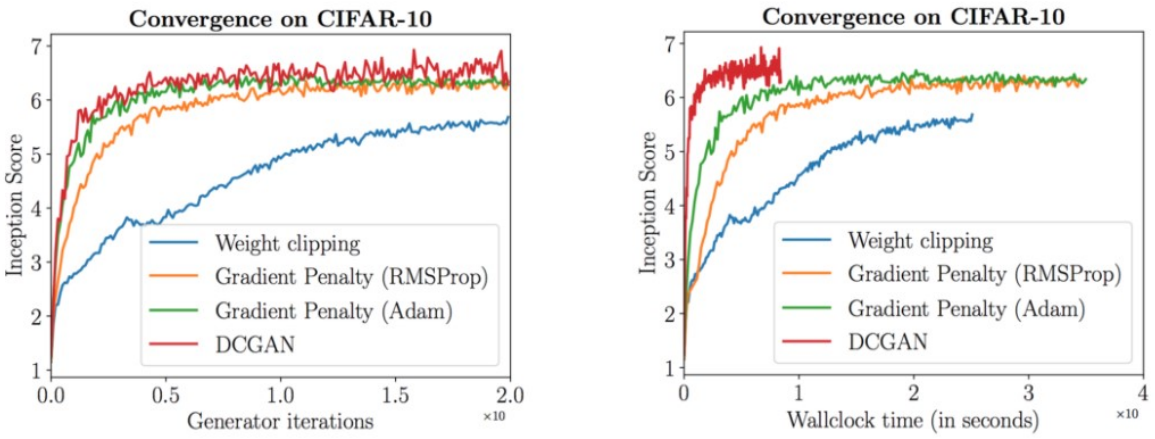


Figure 2.6: Compare inception score (higher is better) over training phase between four different models. **Blue:** WGAN, **Orange:** WGAN-GP uses RMSProp optimizer, **Green:** WGAN-GP uses RMSProp optimizer, **Red :** DCGAN. Figure from [41]

In equation (2.12), I^{LR}, I^{HR} and $G(I^{LR})$ stands for the low-resolution image, the referenced high-resolution image and the high-resolution image generated by the model, respectively. The image $\hat{I}$ sampled form I^{HR} and $G(I^{LR})$ with t uniformly random between 0 and 1 : $I = tG(I^{LR}) + (1-t)(I^{HR})$. Moreover, B is the batch size and λ_g is the new hyper-parameter: gradient penalty. We also notice that D_W represents the critic rather than the discriminator (same notation in WGAN-GP paper). In other words, with $C(x)$ is the non-transformed discriminator output, we have $D_W(x) = C(x)$

In summary, we have already covered three core concepts in Wasserstein GAN: Wasserstein-1 metric, K-Lipschitz constraint and the critic. Next, we took a glance at an enhanced version of Wasserstein GAN: WGAN-GP. Finally, we proposed a new WGAN-GP based loss for super-resolution in equation (2.12).

2.4 Frequency artifacts problem

2.4.1 Frequency artifacts

In the early stage, it is easier to separate between real images and images generated from GAN model. Recently, some state-of-the-art GAN-based models [119] can even trick human observers. However, some authors proved both generator and discriminator do not work well in the frequency domain [19, 32, 71]. Take StyleGAN for an example. As can be seen in the top images in Figure 2.7, although StyleGAN is SOTA in image generation task, the generator can produce high-quality images but suffers some artifact in the frequency domain. In bottom images, except the leftmost image when the authors change the spectrum over a large bandwidth, the discriminator scores in all other cases are similar. In other words, discriminator cannot clearly differentiate high-frequency components.

To investigate why GAN does not learn high-frequency features well, we need to understand the behavior of deep neural network. In 2018, by both theories and experiments, Xu [128] claims the F-principle:

Theorem 2.3. [128] **F-principle:** Deep neural networks often fit target functions from low to high frequencies during the training.

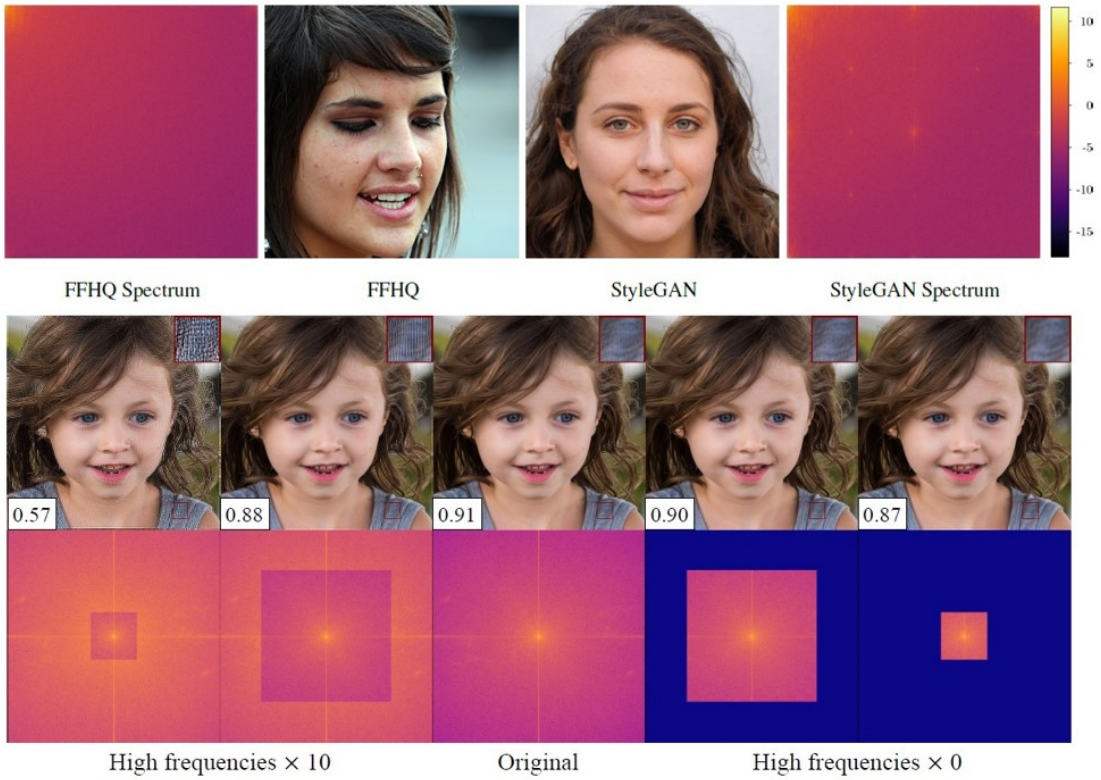


Figure 2.7: StyleGAN does not work well in frequency domain. **Top images:** The generator experiment. From left to right: the mean discrete cosine transform spectrum of real image, the real image, the generated image and this corresponding spectrum. **Bottom images:** The discriminator experiment. Images from the generator and scores from the discriminator with difference high-frequency components. Top images from [32] and bottom images from [19]

Theorem 2.3 means that although deep neural networks can approximate any kinds of function, they prefer low-frequency ones. Another analysis based on Fourier transform from Rahaman et al. [98] results in the similar phenomenon, which they called spectral bias. Further, in 2021, Ma et al. [77] extend this principle in non-gradient-descent-based training cases such as conjugate gradient or particle swarm optimization. Those results emphasize how generative models (not only GAN) work: they try to avoid high-frequency components and converge to a less optimal point.

From the above reason, the frequency gap normally exists between fake images and generated images [71]. There are several reasons why we should encourage the model to learn high-frequency details. First, large frequency gap means that the models fail to reproduce spectral distribution. The efficient model should produce realistic images in all aspects, not only in the spatial domain but also in the frequency (spectral) domain. Secondly, as the GAN model does not carefully consider high-frequency, their performance drops dramatically when they meet input contains dense high-frequency details [19]. Finally, some prior works try to guide the model focus more on high-frequency components and achieve superior results in various tasks: deep fake image recognition [32], image generation [19], inverse rendering and image regression [115], etc.

2.4.2 Related methods

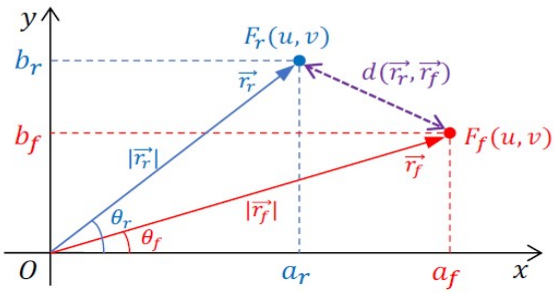
Many authors propose some methods to narrow the gap of generated images and real images. According to our survey, we can group those approaches into three main strategies (for illustration, please see Figure 2.8):

- **Regularize loss function:** By using a similar strategy as WGAN-GP, some authors add the constraint for the loss function. First, they use some kinds of transformation to convert both generated images and real images to the frequency domain. Next, they compute the dissimilarity between two received outputs. The more different between two outputs, the heavier penalty in the loss function. Following this strategy, Jiang et al. [54] develop a novel focal frequency loss inspired by Euclidean projection and focal loss. Meanwhile, Durall et al. [27] devise a spectral loss based on azimuthal integration over the spectrum and add this to the generator loss.

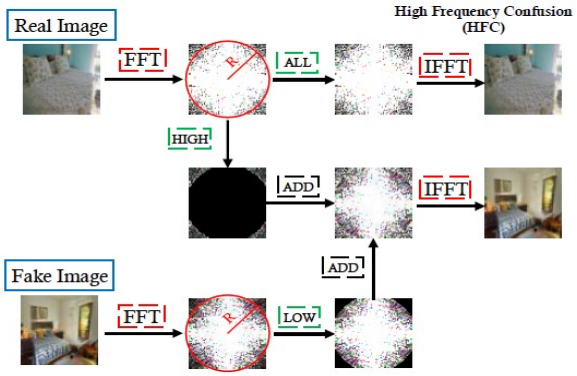
- **Change the inputs:** Sometimes, using the preprocessing methods is enough for improve the quality of generation. In 2021, Li et al. [71] introduce two new preprocessing methods: high frequency filters and high frequency confusion. The main idea is to swap the high-frequency components of real data with low-frequency components of fake data and vice versa. Also, Tancik et al. [115] map the input to Fourier features before passing them into the model. Another technique is frequency separation, which uses a simple linear filter to separate low-frequency and high-frequency details [33]. The authors force the discriminator to concentrate only on high-frequency components, while low-frequency counterpart is learned by pixel-wise loss.

- **Use additional module:** Avoiding overwhelm the discriminator by forcing them to learn in both the spatial and spectral domain, SSD-GAN [19] uses an extra spectral classifier to guide the generator learn in the spectral domain. Moreover, Zhou et al. [146] modify the frequency separation technique by using two discriminators including the high-frequency discriminator and the low-frequency discriminator instead of just use simple filter [33].

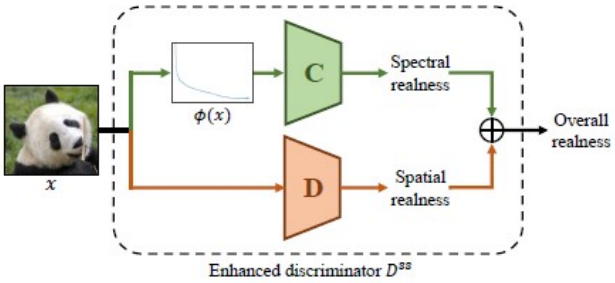
Each strategy has its own benefits and drawbacks. The first and second strategies do not require any change in network architecture. Therefore, they can apply in any generative models without putting any effort. However, as those strategies are simple, their performance cannot guarantee in general. Regularizing loss function normally adds the new hyperparameter: regularization hyperparameter. The suitable choice of this parameter to balance with original loss and develop an efficient regularized loss function are two most arduous tasks. Meanwhile, the second strategies largely depend on image's properties. Take high frequency confusion [71] for an example. While both image have similar texture (mainly low-frequency components) as in image (a) and (b) in Figure 2.9, this technique can provide plausible images. However, with image (e) mostly contains high-frequency details and image (f) mostly contains low-frequency details, output images usually are not realistic (same as image (h)). Finally, as the basic way to boost the model performance is increase the number of parameters, use additional module generally provide best results among three approaches. The most challenging part of this method is how to design a suitable module because no structure can work well in all cases. Especially in the hard-training model like GAN, append improper component can make the model diverge.



(a) Regularize loss function



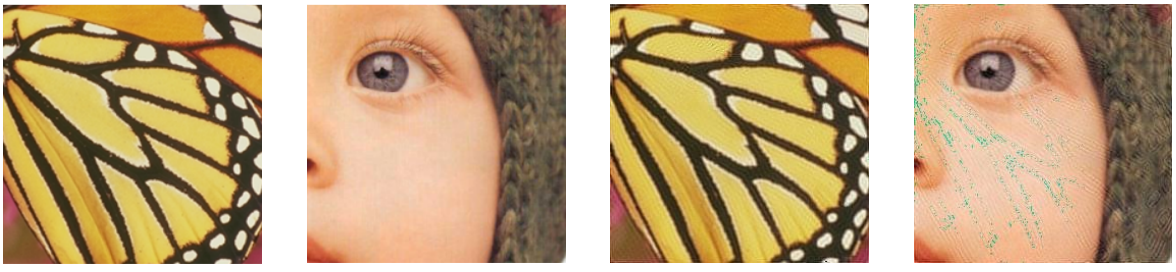
(b) Change the inputs



(c) Use additional module

Figure 2.8: Three strategies to alleviate frequency artifacts problem: **(a)** measure the difference of frequencies in focal frequency loss [54], **(b)** swap frequency components in [71], **(c)** the discriminator and the spectral classifier in SSD-GAN's architecture [19].

(a) Lemon image (b) Squirrel image (c) HFC(a,b) (d) HFC(b,a)



(e) Butterfly image (f) Baby image (g) HFC(e,f) (h) HFC(f,e)

Figure 2.9: High frequency confusion [71] experiment with different images. HFC(x,y) means the image generated by combine low-frequency components of image x with high-frequency components of image y

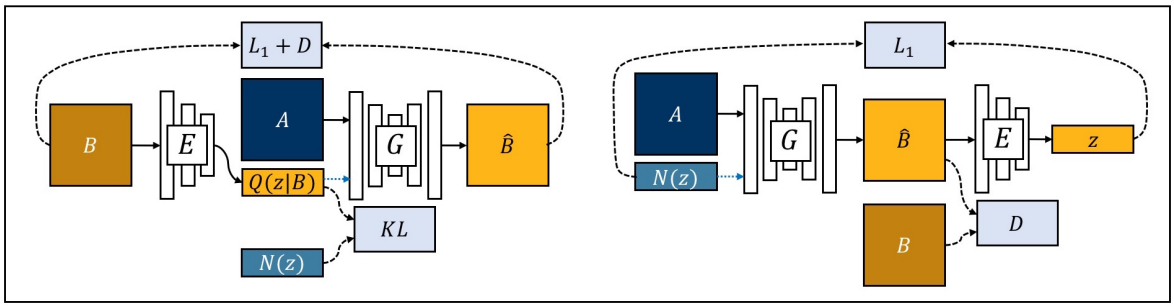


Figure 2.10: Architecture of BicycleGAN

2.5 Diversity-aware image generation

Aside from 1-to-1 image translation, many previous works also explore and introduce a variety of mechanism to enforce the generator to model a distribution of possible outputs in a conditional generative modeling setting. Nonetheless, the two most common methods of introducing diversity into deep generative models are by architecture design [148, 133] and additional loss [130, 79, 73].

2.5.1 Architecture design

The most notable example for the architecture design approach is arguably BicycleGAN [148] which train the generator and encoder in two cycles:

- On one hand, conditional Variational Autoencoder GAN (cVAE-GAN) use image reconstruction loss to force the encoder to learn meaningful encoded latent so that the generator can use the latent to generate natural outputs.
- On the other hand, conditional Latent Regressor GAN (cLR-GAN) constraint the generator to output an image that is consistent with the latent code by the latent reconstruction loss.

Built upon BicycleGAN, DMIT [133] utilizes not one but three additional encoders E_c, E_s, E_d to better disentangle the feature of the images. Although their architectures are identical to each other, these three encoders have different objectives by nature:

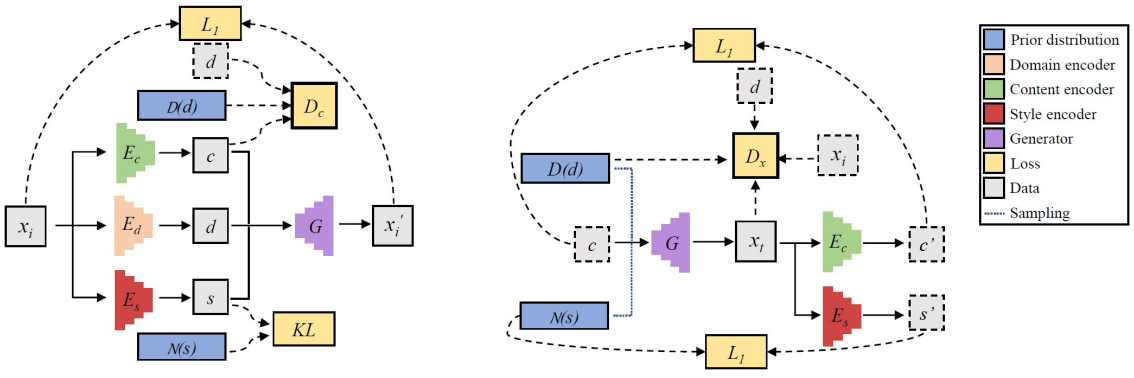


Figure 2.11: Architecture of DMIT

- E_c encodes the content and domain-specific features of the images.
- E_s extracts class-agnostic attributes of the images or, as the paper describes, the style of the output.
- E_d is used to output the class label, i.e domain that the generator is trained with.

On a downside, since there are no explicit regularization for diversity, these frameworks typically suffer from mode collapse and learn an overly simplified distribution where an input is always mapped to a single output regardless of variations in latent code.

2.5.2 Additional loss

Realizing the drawbacks of above-mentioned approach, some recent works have introduced an explicit regularizer to avoid the mode collapse problem of image-to-image translation models such as mode-seeking loss[79], diversity-sensitive loss[130] or contrastive loss [73].

As for the former two losses, they aim to maximize the ratio between the distance of two outputs and the corresponding latent codes like following:

$$\max_{\theta} \frac{d_I(G_\theta(z_1, c), G_\theta(z_2, c))}{d_z(z_1, z_2)} \tag{2.13}$$

Here, $d_I(\cdot, \cdot)$ can be any distance for images such as L_1 loss, perceptual loss or feature matching loss [130] and, similarly, $d_z(\cdot, \cdot)$ can be any distance for latent codes like L_1 loss or cosine similarity loss.

These losses works very well in many image translation tasks, however they tend to generate much diverse outputs even when the latent codes differ minimally. To combat this issues, the author of DivCo [73] encourages images generated from adjacent latent codes to be similar and push those generated from distinct latent codes far away by a contrastive loss:

$$\min \frac{\exp\left(\langle f, f^+ \rangle / \tau\right)}{\exp\left(\langle f, f^+ \rangle / \tau\right) + \sum_{i=1}^{N} \exp\left(\langle f, f_i^- \rangle / \tau\right)} \tag{2.14}$$

Here, f is vector feature extract from the output of the generator by an feature extractor, $\langle \cdot, \cdot \rangle$ is the inner product, f^+ is the vector features from the latent code adjacent to that for f and f_i^-s are the opposite.

At first glance, these losses seem to be different, however, they share the similar train of thought which is maximizing the distance of two or many outputs while keeping the distance between the corresponding latent codes small.

However, these explicit constraints might favor overly the diversity and eventually degrade the quality of the synthesized images. Therefore, when using these losses, extra care is necessary to balance between variety and image quality.

2.6 Baseline model selection

2.6.1 Overview

In this section, we only consider **SRGAN, EnhanceNet, ESRGAN and SRFeat** mentioned in many SISR surveys [6, 131, 123, 58] as we find them either promising

or interesting to study and suitable with our thesis scope. Moreover, many later models based on these models and make improvements in some aspects [78, 15, 14, 9]. Thus, we analyze these four models for further choosing the target model.

As for dataset, many models use DIV2K [3] for their training and validation process. This dataset comes from NTIRE challenge, contains 2K resolution quality and includes 800 images for training and 100 images each for validation and testing. Normally, this dataset is used on training and validation, as the test set is private.

2.6.2 SRGAN [69]

2.6.2.1 SRGAN overview

Although super-resolution works prior to SRGAN [69] get more accurate and achieve competitive performance in regard of the pixel-based metrics like PSNR, these models tend to produce perceptually unsatisfying images in the sense that they are smooth and often lacking high-frequency details expected at the higher resolution [69]. To combat such a problem, Ledig et al [69] introduce a first-ever GAN-based approach for super-resolution that performs poorly on PSNR and SSIM but produces visually pleasing high-resolution images. The main highlight of SRGAN is the use of multi-task loss formulation consisting of three different sub-losses:

- MSE (L_2) loss (2.15) that learns to pixel-wise similarity between the ground truth image and the output.

$$L^{SR}_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}\left(I^{HR}_{x,y} - G_{\theta_G}(I^{LR})_{x,y}\right)^2 \tag{2.15}$$

  where W, H and r are the width, height of the low-resolution input and the scaling factor respectively

- Perceptual loss (2.16) in terms of Euclidian distance of the ReLU activation layer of VGG [110] between the reconstructed image and reference image.

$$L^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}\left(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y}\right)^2 \tag{2.16}$$

  Where $W_{i,j}$ and $H_{i,j}$ describe the dimensions of the respective feature maps within the VGG network, while $\phi_{i,j}$ is the respective ReLU activation layer in VGG.

- Adversarial loss [85] that tries to encourage super-resolved outputs lying close to the manifold of natural images. Here the generator (2.17) and discrimnator (2.18) loss are defined as

$$L^{SR}_{Gen} = -\frac{1}{B}\sum_{b=1}^{B}\log D_{\theta_D}(G_{\theta_G}(I^{LR})) \tag{2.17}$$

$$l^{SR}_{Dis} = -\frac{1}{B}\sum_{b=1}^{B}\left(\log D_{\theta_D}(I^{HR}) + \sum_{b=1}^{B}\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR})))\right) \tag{2.18}$$

  where G, D and B are the generator, discriminator and batch size respectively.

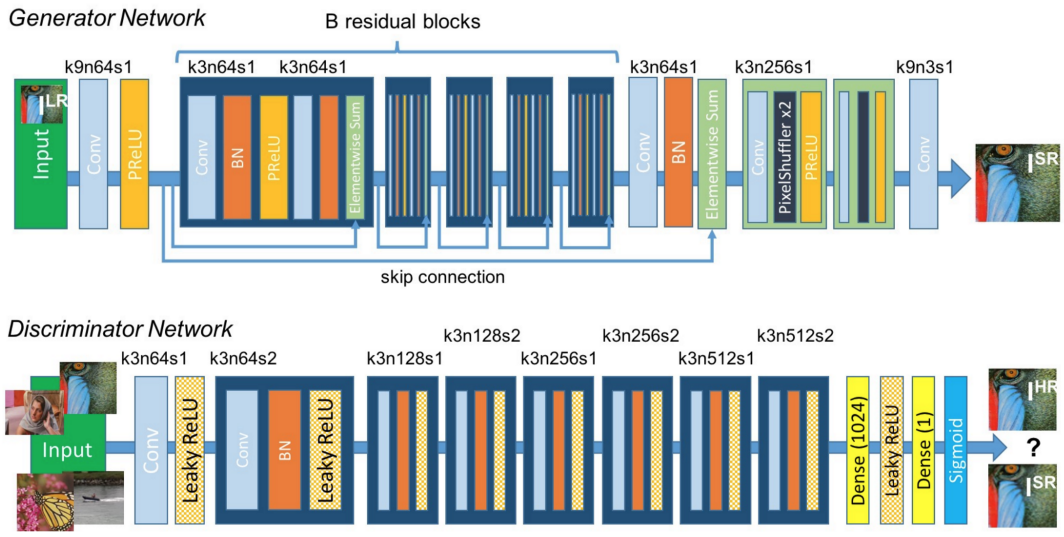


Figure 2.12: SRGAN architecture [69]

- To sum up, the total loss (2.19) that the generator uses is:

$$L^{SR}_{total} = \lambda_{MSE} \times L^{SR}_{MSE} + \lambda_{VGG} \times L^{SR}_{VGG} + \lambda_{Gen} \times L^{SR}_{Gen} \tag{2.19}$$

  where $\lambda_{MSE}, \lambda_{VGG}$ and λ_{Gen} are hyperparameters to balance between three losses.

Architecturally, SRGAN employs an deep residual network [39] with 16 residual blocks for the generator; on the other hand, for the discriminator, the design is inspired from Radford et al. [96] (Figure 2.12)

2.6.2.2 SRGAN analysis

On one hand, the highlights of SRGAN is the use of adversarial loss to enhance the perceptual quality of the image, which sets new state-of-the-art for 8x super-resolution at its time and indicates that GAN is one of the promising tools for single image super-resolution.

On the other hand, as mentioned above, there is a strange phenomenon that SRGAN achieves high perceptual quality but score low in pixel-wise metrics. There is an example to illustrate this issues in Figure 2.13 [69], where PSNR and SSIM's score of bicubic and SRResNet - a non-GAN approach for SISR - are both higher than the score of SRGAN but the generated image of SRGAN, to the human's eye, is more visually pleasing than the other two.

As described in [69], SRGAN's model contains many batch norm layers, which can explain for aforementioned phenomena. BN layers normalize the features using mean and variance in a batch during training and 'wash' out high-frequency features of the input images. When the statistics of training and testing datasets differ a lot, BN layers tend to introduce unpleasant artifacts and limit the generalization ability of the super-resolution module.

Moreover, another downside of SRGAN is that it uses feature **after** ReLU activation layer for the perceptual loss, which is sparse making the generator difficult to learn meaningful features for the task. Therefore, another work described in section 2.6.4 tries to address the above two stumbling blocks.

2.6.3 EnhanceNet [104]

2.6.3.1 EnhanceNet overview

Similar to SRGAN, Sajjadi et al. [104] proposes a novel super-resolving model EnhanceNet focusing on creating realistic textures rather than optimizing for a pixel-accurate reproduction of ground truth images during training with a combination of four different losses:

- MSE loss, which is exactly the same one used in SRGAN.
- Perceptual loss to encourage the network to produce images that have similar feature representations, which is also similar to the loss used in SRGAN.
- Adversarial loss which also the same loss used in SRGAN.
- Texture loss (2.20): unlike SRGAN that uses only perceptual loss to encourage visually pleasing super-resolution, EnhanceNet also makes use of a different loss inspired from Neural Style Transfer frameworks such as one in [35]. The texture loss is to match the texture of low and high-resolution images extracted from a pre-trained model (VGG) and is quantified as the L_1 loss between gram matrices computed from deep features

$$L^{SR}_{Text/i,j} = \frac{1}{H^2_{i,j}} \sum_{x=1}^{H_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\mathbb{G}(\phi_{i,j}(I^{HR}))_{x,y} - \mathbb{G}(\phi_{i,j}(G_{\theta_G}(I^{LR})))_{x,y} \right)^2 \tag{2.20}$$

  Where $H_{i,j}$ describe the height of the respective feature maps within the VGG network, while $\phi_{i,j}$ is the respective ReLU activation layer in VGG and the Gram matrix $\mathbb{G}(F) = FF^T \in R^{n \times n}$

As for the architecture (Figure 2.14), EnhanceNet utilizes a combination of fully convolutional network with ResNet inspired by Long et al. [74]. However, as Sajjadi et al. [104] replace the learnable upsampling module in the work of Long et al. [74] by nearest neighbour upsampling to reduce checkerboard artifacts [104]

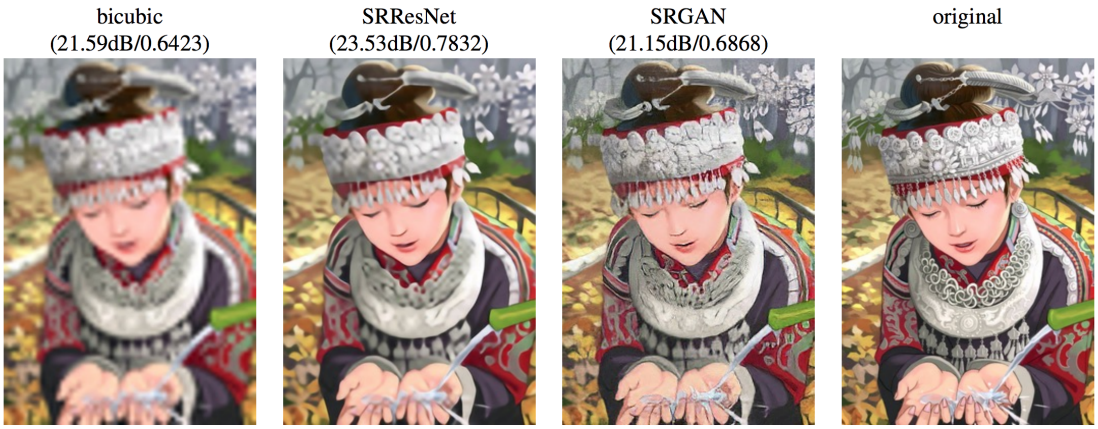


Figure 2.13: The two numbers in each picture of each methods are PSNR and SSIM respectively [69]

Output size	Layer
$w \times h \times c$	Input I_{LR}
$w \times h \times 64$	Conv, ReLU
	Residual: Conv, ReLU, Conv
	...
$2w \times 2h \times 64$	2x nearest neighbor upsampling Conv, ReLU
$4w \times 4h \times 64$	2x nearest neighbor upsampling Conv, ReLU
	Conv, ReLU
$4w \times 4h \times c$	Conv
	Residual image I_{res}
	Output $I_{\text{est}} = I_{\text{bicubic}} + I_{\text{res}}$

Figure 2.14: EhanceNet architecture [104]

2.6.3.2 EnhanceNet analysis

Similar to SRGAN, the use of additional loss besides the MSE loss really does improve the sharpness and fidelity of the generated images with a sacrifice of pixel-accuracy. In fact, EnhanceNet is reported to give the best performance for PSNR metric when only MSE loss is used but its output is overly-smooth and blurred, whereas the full model of EnhanceNet produces images with high perceptuality.

However, from the qualitative results reported in [104], we observe that images produced by EnhanceNet still have unwanted and strange artifacts, for example in the Figure 2.15. Unlike SRGAN, this cannot be the fault of the batch norm layers as they do not occur in EnhanceNet architecture. Our hypothesis is that EnhanceNet heavily uses perceptual and texture loss which depends on a pre-trained model for **classification**, not for **super-resolution**; therefore, optimizing both of these losses may confuse the model and introduce weird artifacts to the output images.

In addition, there is a vital disadvantage of using texture loss is the computational burden of Gram matrix. Especially with features maps that have large height but small width, we can see from the equation (2.20) that it can be extremely inefficient.

2.6.4 ESRGAN [120]

2.6.4.1 ESRGAN overview

ESRGAN builds upon SRGAN and currently one of the state-of-the-art models that produce realistic high-resolution images. ESRGAN's main aim is to improve the overall perceptual quality for super-resolution as well as combat some issues occurring in the work of SRGAN [120]

About network architecture, ESRGAN employs the basic architecture of SRResNet [69] (Figure 2.16), which learns a 1-to-1 mapping from low-resolution images to high-

resolution ones. Here, "basic blocks" could be selected or designed (e.g., residual block [42], dense block [46]) for better performance.

Compared to SRGAN, ESRGAN made two modifications to the structure of the generator (Figure 2.17):

- Remove all batch norm (BN) layers.
- Replace the original basic block with the proposed Residual-in-Residual Dense Block (RRDB), which combines multi-level residual network and dense connections.

Alongside re-structuring the generator G, ESRGAN also deploys some modifications for the loss:

- ESRGAN replaces L_2-norm loss used in SRGAN with L_1-norm loss to produce less smooth images [120] and preserves high-frequency features.
- Previously, perceptual loss used in SRGAN and other works prior to ESRGAN is based on minimizing the different after-activation features of the restored image and the ground truth. Contrary to the convention, ESRGAN uses features **before** the activation layer, which gives richer features representation.
- Remarkably, ESRGAN favors a recent innovative GAN model which is relavistic average GAN [57] rather than the standard GAN used in SRGAN, as shown in Figure 2.18.

Specifically, the generator loss and discriminator loss use in adversarial training are:

$$L_G^{RaGAN} = -\frac{1}{B}\sum_{b=1}^{B}\Big(\log\left(1 - D_{RaGAN}(I^{HR}, G(I^{LR}))\right) + \log D_{RaGAN}(G(I^{LR}), I^{HR})\Big)$$

$$L_D^{RaGAN} = -\frac{1}{B}\sum_{b=1}^{B}\Big(\log D_{RaGAN}(I^{HR}, G(I^{LR})) + \log\left(1 - D_{RaGAN}(G(I^{LR}), I^{HR})\right)\Big)$$

(2.4 revisited)

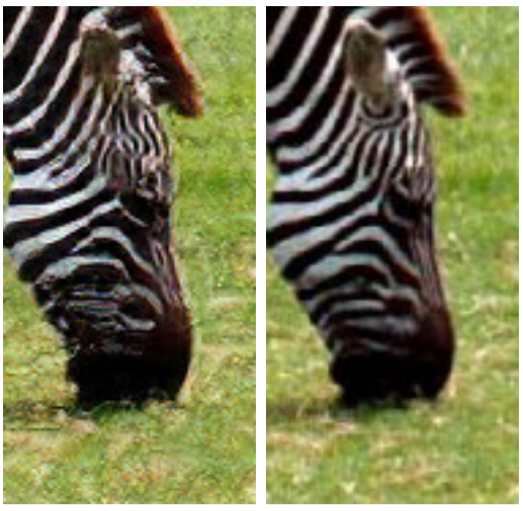


Figure 2.15: **Left**: Generated from the full model of EnhanceNet. **Right**: the ground truth [104]

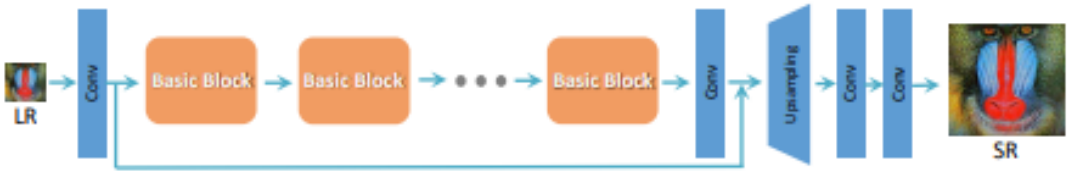


Figure 2.16: ESRGAN architecture [120]

where I^{LR}, I^{HR}, $G(I^{LR})$ and B stands for the low-resolution image, the referenced high-resolution image, the high-resolution image generated by the model and the batch size, respectively. Finally, $D_{RaGAN}(x, y) = \sigma(C(x) - \frac{1}{B}\sum_{b=1}^{B} C(y))$ where σ denotes the sigmoid function and $C(x)$ is the non-transformed discriminator output.

Not only that, ESRGAN also deploys a completely different training setting compared to SRGAN and EnhanceNet. Specifically, it is trained on a bigger dataset consisting of more than 3,450 images in total with some data augmentation techniques to avoid overfitting. Unlike SRGAN which trains the GAN from scratch, ESRGAN employs a pre-training phase with only L_1 loss first then train the full mode with all three loss. With these additional techniques, ESRGAN can learn more stably and capture richer texture as well as more high-frequency details of high-resolution images.

2.6.4.2 ESRGAN analysis

By using features before the activation layers for perceptual loss, ESRGAN overcomes two drawbacks of the SRGAN's design [120]:

- The features after the activation layers are sparse, especially after a very deep network, as deep learning models commonly use sparse activation functions such as ReLU. The sparse activation provides weak supervision and thus leads to inferior performance (Figure 2.19).
- By using features before activation, ESRGAN empirically produces more photo-realistic images with less inconsistent brightness compared with the ground-truth image.

The BN layer was empirically proven to be inefficient and computation-consuming in different PSNR-oriented tasks, as mentioned in 2.6.2.2. ESRGAN, therefore, removes BN layers for stable training and consistent performance. Furthermore, removing BN

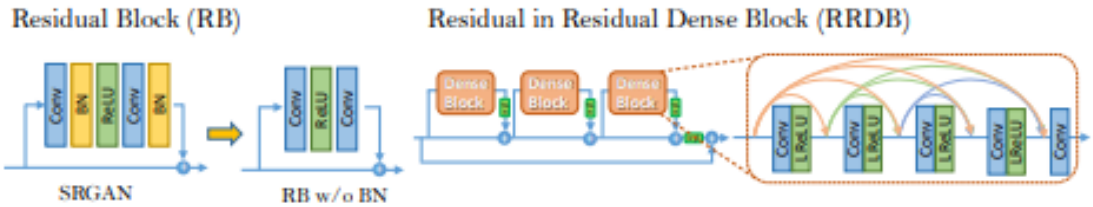


Figure 2.17: **Left**: ESRGAN removes the BN layers in residual block in SRGAN. **Right**: RRDB block is used in ESRGAN's model and β is the residual scaling parameter. [120]

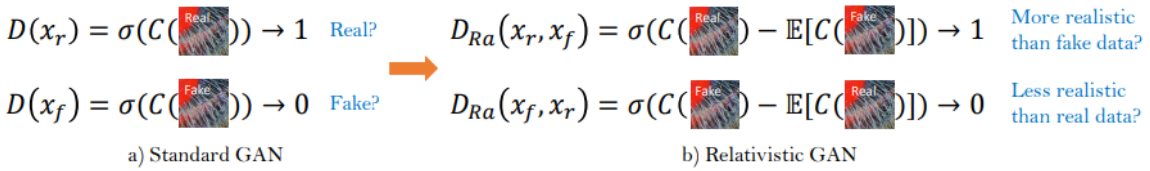


Figure 2.18: Difference between standard discriminator and relativistic discriminator. [120]

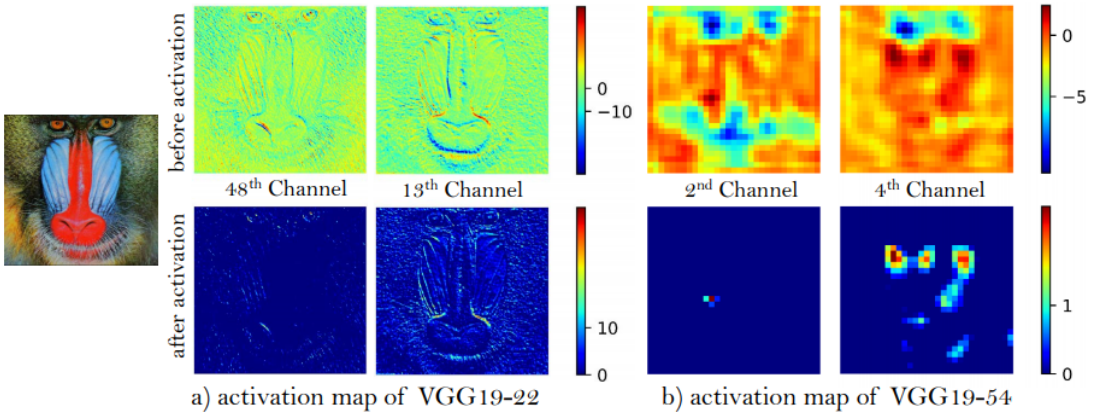


Figure 2.19: Representative feature maps **before** (first row) and **after** (second row) activation for a sampled image. [120]

layers helps to improve generalization ability and to reduce computational complexity and memory usage.

Additionally, it is observed that the adversarial loss for generator in (2.4) contains both I^{HR} and $G(I^{LR})$. Therefore, the generator benefits from the gradients from both generated data and real data in adversarial training, while in SRGAN only generated part takes effect. With these changes, not only is ESRGAN more stable than SRGAN [120] but also reported to achieve **both** high pixel-accuracy and perceptual quality [6].

On the downsides, ESRGAN's model is comparatively large with the numbers of learnable parameters up to nearly 40 million. Without decent computational resources,

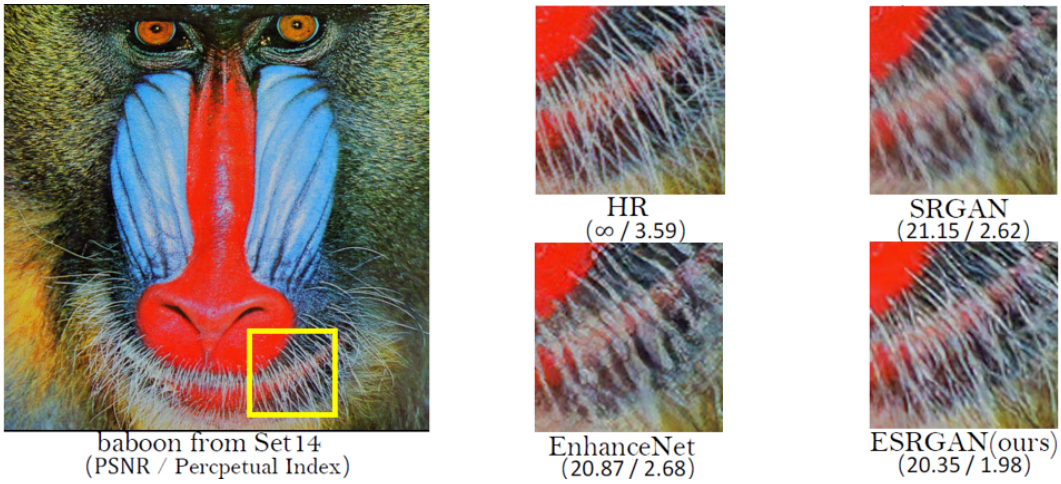


Figure 2.20: The comparison between SRGAN, ESRGAN and EnhanceNet [120]

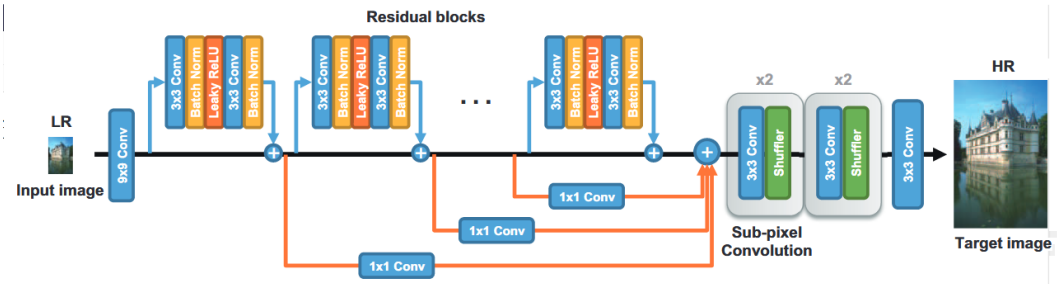


Figure 2.21: SRFeat architecture. [93]

one should not try ESRGAN but opt for other alternatives.

2.6.5 SRFeat [93]

2.6.5.1 SRFeat overview

Although previous GAN-based SISR to SRFeat [93] showed impressive results with photo-realistically textures and sharp images, they tend to include meanless high-frequency noise that is irrelevant to the input images. Hence, Park et al. [93] proposed a different approach to overcome the limits of prior work and maintain perceptual fidelity of the reference images by using an additional discriminator that works on features domain and guide the generator to generate high-frequency structural details rather than unwanted artifacts.

To achieve impressive results on SISR, SRFeat exploits a generator network with 16 residual blocks and multiple long-range skip connections inspired from the work of [120] and [69]. With longer skip connection and deeper network, the information in distant layers can be more effectively propagated. Then before the upsampling layers, SRFeat utilizes additional long-range skip connections to aggregate features from all previous residual blocks to further encourage back-propagation of gradients and give potentials to re-use intermediate features to improve the final feature, as shown in Figure 2.21.

In SRFeat, while the model is optimized with the three common loss used in almost all mentioned work in this chapters:

- MSE loss to encourage pixel-accuracy
- Perceptual loss to enforce perceptual quality
- Adversarial loss to make the model produce realistic images

Moreover, the novel idea of this work is the use of additional **feature adversarial loss**. Unlike the above-mentioned adversarial loss which works on **image** space, the feature adversarial loss tries to detect whether the **features** of the generated images is real or not. In detail, the feature GAN loss term L_G^f for the generator and the loss function L_D^f for the feature discriminator are defined as:

$$L_G^f = -\frac{1}{B}\sum_{B}^{b=1} \log D^f\left(\phi(G(I^{LR}))\right), \text{ and} \tag{2.21}$$

$$L_D^f = -\frac{1}{B}\sum_{B}^{b=1} \left(\log D^f\left(\phi(I^{HR})\right) + \log D^f\left(\phi(G(I^{LR}))\right)\right) \tag{2.22}$$

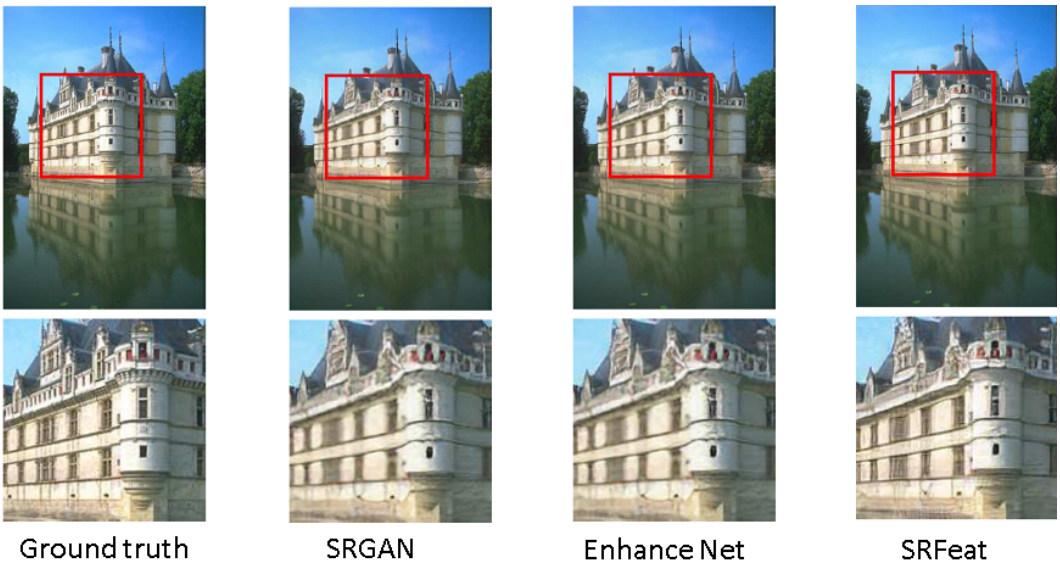


Figure 2.22: The qualitative comparison between SRGAN, EnhanceNet and SRFeat [93]

Here D^f is the feature discriminator and ϕ is the feature maps of a pre-trained model, which is VGG in SRFeat.

As for training procedure, though also using pre-training as ESRGAN and the same datasets as SRGAN (DIV2K [3]), SRFeat follows a slightly different setting. Particularly, SRFeat pre-trains the model with MSE loss on **ImageNet** [25] dataset first, then trains the model on DIV2K but **without** MSE loss

2.6.5.2 SRFeat analysis

The advantages of SRFeat mainly come from this additional discriminator. By distinguish between features of the synthetic image and the real image, the model output tend to include high-frequency structural features rather than random noisy artifacts as in the Figure 2.22.

However, adding additional discriminator means more training difficulties and computational resources (even more than ESRGAN). To sidestep this problem, as we mentioned above, the authors pre-train generator instead of training it from scratch.

2.6.6 Summary

Both **ESRGAN** and **SRFeat** present their outperformed results compared to both **SRGAN** and **EnhanceNet**. In summary, both ESRGAN and SRFeat are capable of generating more detailed structures (see Figure 2.22 and Figure 2.20) while two other methods either fail to produce enough details (SRGAN) or add undesired textures (EnhanceNet).

There is no original comparison from the authors between **SRFeat** and **ESRGAN**. According to Anward et al. [6], the PSNR and SSIM performance of **ESRGAN** for scaling factor 4 are better compared with many other models but they do not include **SRFeat** in their experimental results. Due to our limit computing resources, we target **ESRGAN** instead of the latter paper. The main reason is that **SRFeat** uses one

generator and two discriminators, whereas **ESRGAN** only uses one generator and one discriminator. In other words, **ESRGAN** uses less computing resources for training.

Chapter 3

Research background

3.1 Deep Learning

3.1.1 Artificial Neural Network [37]

The nervous system of both humans and animals is made up of processing units called neurons. The brain structure is composed of billions of neurons linked together, each neuron is capable of emitting information in the form of electrical signals. Inspired by this structure, in Artificial Neural Network (ANN), each neuron is a unit of computation whose input and output are scalar quantities. Each neural network value received is multiplied by the corresponding weight, adding all these products then again apply a nonlinear function to output the result (see Figure 3.1).

Artificial neural networks are built by connecting neurons together: the output of the preceding neurons will be the input of the following neurons. With precisely adjusted weights, the neural network can approximate many complex mathematical functions. Specifically, in Figure 3.2, each circle is one neuron, entering and leaving arrows are the inputs and outputs of that neuron respectively. The neurons are arranged as layers, representing the information flow through the network from the input layer to the output layer.

3.1.2 Activation function

Activation functions play a crucial role in ANN. Normally, an ANN consists of at least one non-linear activation layer. Non-linear means that the output cannot be

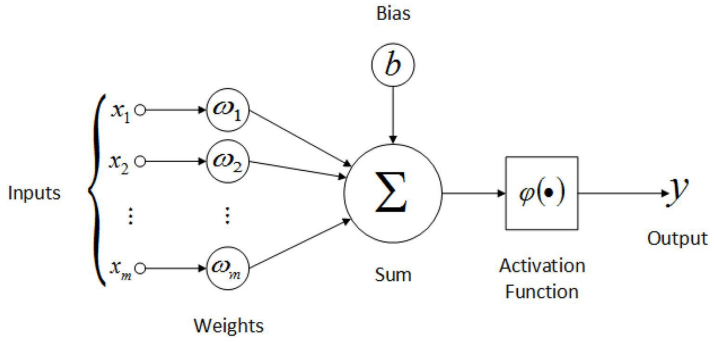


Figure 3.1: Inside a neuron in ANN [4]

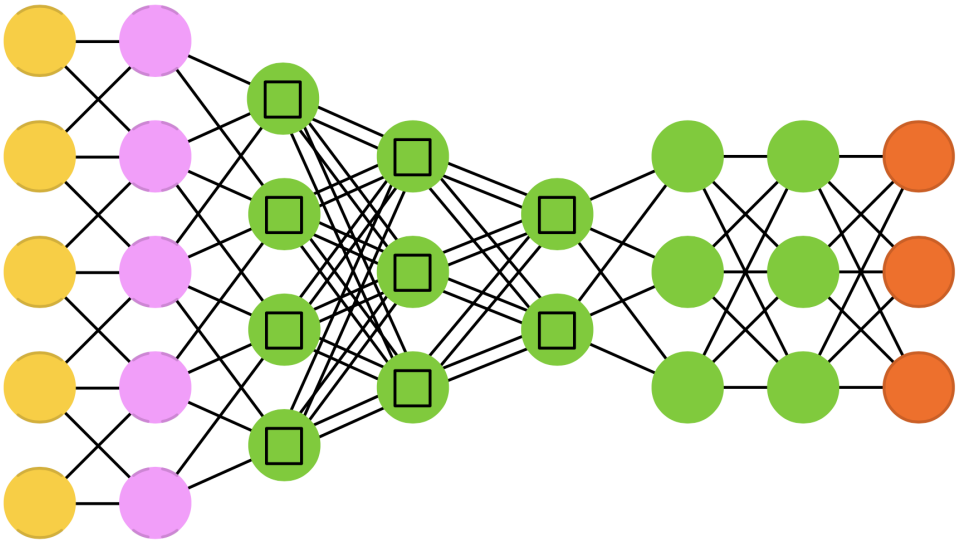


Figure 3.2: Artificial Neural Networks [118]

reproduced from any linear combination of the inputs. The composition of two or more linear functions is also a linear function. Hence, an ANN that only consists of linear functions cannot present the non-linear relationship and non-linear activation function must be used to avoid this linear relation. There are many non-linear functions that can be used in ANN. Currently, there is no theory about which non-linear function should be used in which case, so everything is much empirical. Among these functions, the following are most commonly used: sigmoid, Rectified Linear Unit (ReLU), Leaky ReLU.

3.1.2.1 Sigmoid function

Sigmoid activation function generates the output in the range between 0 and 1. Therefore, it is generally used for models where we have to estimate the probability as the output since the probability of any event is between the range of 0 and 1. Furthermore, this activation function is continuously differentiable. This property allows applying the gradient-based optimization method with this function easier to convergence.

However, there are some disadvantages:

- Gradient vanishing: If the input value is near 0 or 1, the derivative of this function goes to 0. Therefore, this function kills gradients and cannot learn.
- Limited output range: The output value only ranges from 0 to 1, which is not good in many complicated cases.

3.1.2.2 Rectified Linear Unit (ReLU)

It can be said that ReLU is the most popular used activation function in deep learning model, especially in convolutional neural networks. This function always generates non-negative value. In detail, ReLU returns 0 if the input is less than 0 otherwise

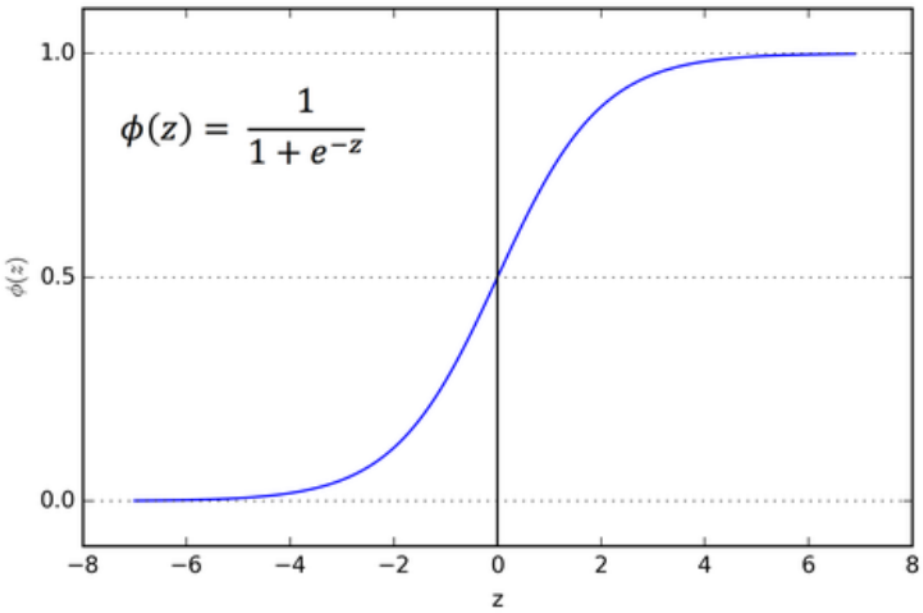


Figure 3.3: Sigmoid function [82]

ReLU returns the input itself. In many cases, the ReLU is the activation function that should be tried first because of its simple property (see Figure 3.4).

However, there are some drawbacks:

- Lost information: If the input value is negative, ReLU lost this input.
- Linear function: If all the input value is non-negative, ReLU becomes linear. However, this case rarely exists in real life, as the number of input is very large.

3.1.2.3 Leaky ReLU

To address the problem of original ReLU, we can use a Leaky ReLU activation function. Leaky RELU does not change the output value compared to ReLU if the input is non-negative. However, in the negative case, Leaky RELU produces a small slope of the input. This type of activation function is popular in tasks where we may suffer from sparse gradients, for example training generative adversarial networks. However, Leaky ReLU adds a new hyperparameter: the slope. In Figure 3.5 , the slope is 0.1, but it can any value (normally less than 1).

3.1.3 Convolutional Neural Network [68]

3.1.3.1 Overview

Convolutional Neural Network (CNN) is another family of neural networks that are widely applied in the field of computer vision as well as another field such as natural language processing, time series analysis,etc. Similar to ANN network, CNN networks also have weights and bias that can be learned by back-propagation algorithm, and also the process of minimizing an error function. Many optimization methods work well on ANN can also be applied for CNN and obtain good results. However, unlike ANN, the CNN neurons in the previous layer connect only a few neurons in the next layer, in

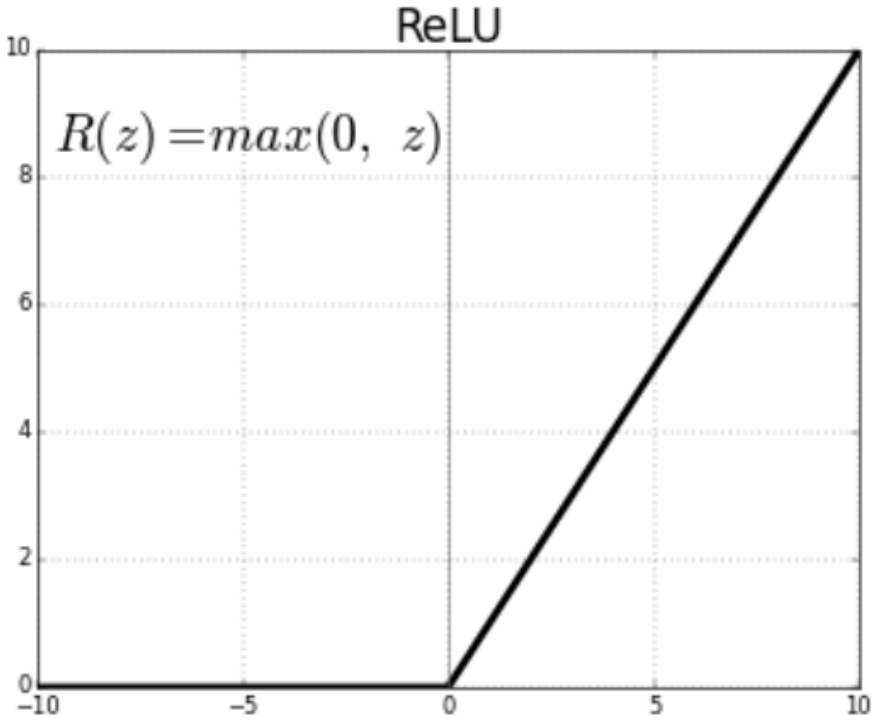


Figure 3.4: ReLU function [20]

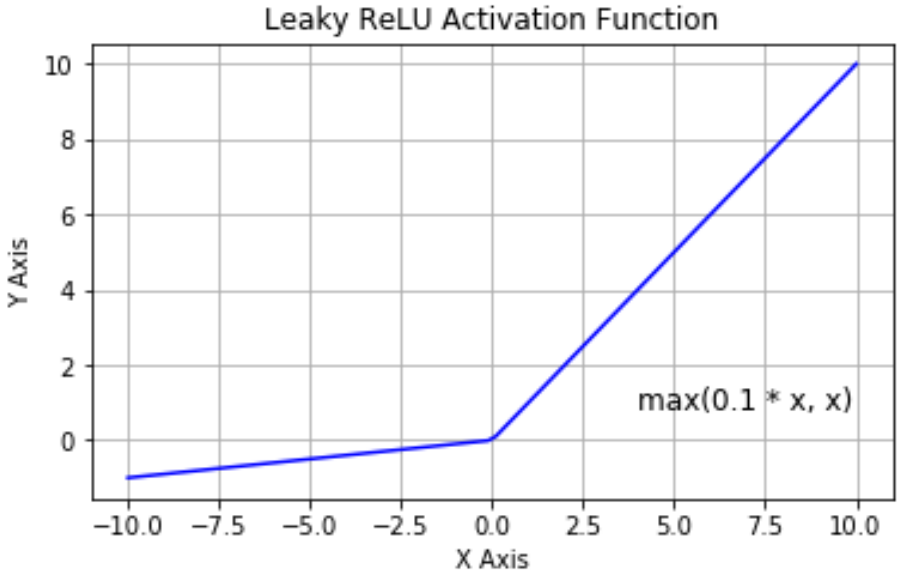


Figure 3.5: Leaky ReLU function (slope = 0.1) [102]

one area called the receptive field, instead of connecting with all. In fact, CNN takes advantage of the **parameter sharing** method for regularization [37]. It is obvious that digital images have some invariant property. For instance, if we remove one pixel of a picture of a dog, it still remains a picture of a dog. To effectively incorporate this property into network design, CNN share parameters across local regions of the image. This means that the same feature is calculated over different regions. In other words, we can find the dog with the same dog detector no matter where a dog appears in the picture (e.g column **i** or **i+1**).

The advantages of parameter sharing are that it reduces the number of model parameters significantly and enable CNN to increase network size without increasing the corresponding resources in training data. Besides parameter sharing, CNN also leverages two other important ideas, namely sparse interaction and equivariant representations, as we will discuss later.

3.1.3.2 Architecture

Similar to ANN, the CNN architecture is composed of many layers. The output of the previous layer is the input of the next layer. Normally, some below layers are used in CNN :

- Convolutional Layer
- ReLU Layer
- Pooling Layer
- Dense Layer (Fully connected layer)

Each layer can appear many times in this network and make the network very "deep". Usually, a batch of images is the input of CNN, where each image is considered as pixel matrix and each pixel has a dimension for its color (RGB) (see Figure 3.6).

3.1.3.3 Convolutional layer

As the name suggested, the convolution operation is the spirit of CNNs. Convolutional Neural Network can be understood as a network that uses convolutional operation in at least one of its layer. In its simplest form, the convolutional operation is the sum of the element-wise product of two matrices. The first matrix is called the input matrix, while another is called the filters (also known as the kernels). In the machine learning context, a kernel size is normally smaller than the input size and the kernel stores the weights learned to detect unique features. The kernel matrix strides on each part of the input matrix and performs convolution operation to get the final

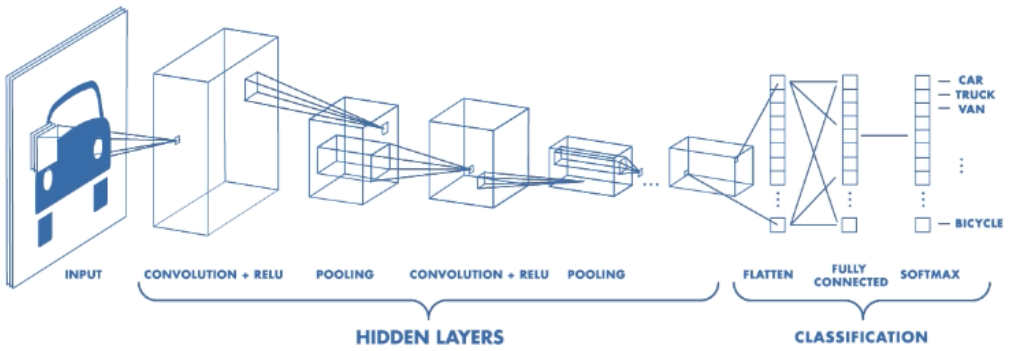


Figure 3.6: Classic Convolutional Neural Network architecture [103]

results. Figure 3.7 illustrates the 2-D convolution without flip the kernel (Goodfellow et al.[37] claim it is useful for writing proofs, but not usually necessary in a real implementation)

Convolution help improve the CNNs performance by leverage two important ideas (beside parameter sharing):

- **Spare interactions**: Unlike the classical ANNs, where every output of the previous layer interact with every input of the next layer. In CNNs, spare interactions are typically ensured by keeping the kernel size smaller than the input size. Spare interactions mean that this network requires both less storage to store the parameter and less operation to compute the output.
- **Equivariant representations**: This property is inherited by parameter sharing. In short, if the input changes, the output changes in a similar manner. For instance, if we move some objects in the input picture, its representation will change the same amount in the output. However, convolution is only naturally equivariant with translation operator. For another operator (e.g scale or rotate), we need to use some other mechanisms to force this constraint.

3.1.3.4 ReLU layer

For this layer, all negative values must be ignored and assigned to 0 by ReLU function. The input and output of this layer share the same size (see Figure 3.8). In many cases, ReLU is preferred because it speeds up the training time without dramatically decrease the accuracy.

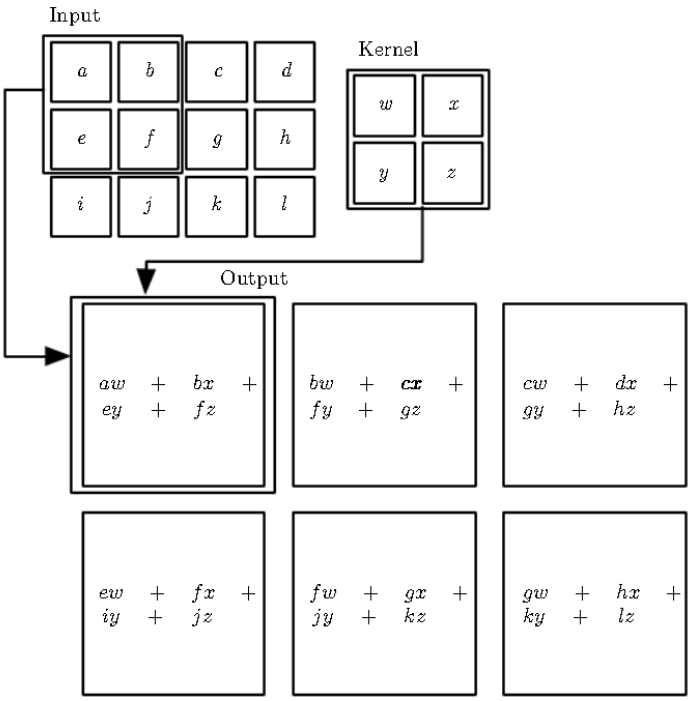


Figure 3.7: An example of 2-D convolution without kernel flipping [37]

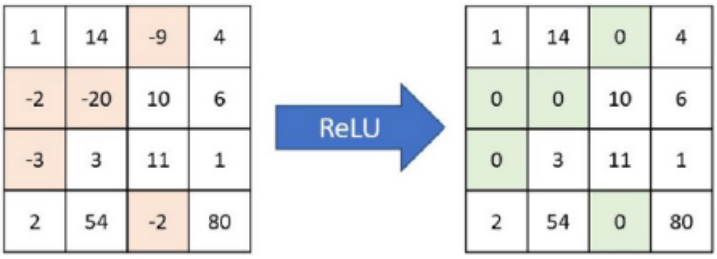


Figure 3.8: ReLU layer

3.1.3.5 Pooling layer

Pooling layers provide an approach to downsampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively.

3.1.3.6 Dense layer

This layer is mainly used for recognition and classification. The number of nodes in this layer refers to the total of classes need to be classified. In the dense layer, all inputs of the current layer interact with all outputs of the previous layer. The dense layer is usually used at the end of CNNs, because after being processed by three above-mentioned layers, the size of the data is not so large as the beginning. Therefore, using the dense layer does not require a large amount of computation resources.

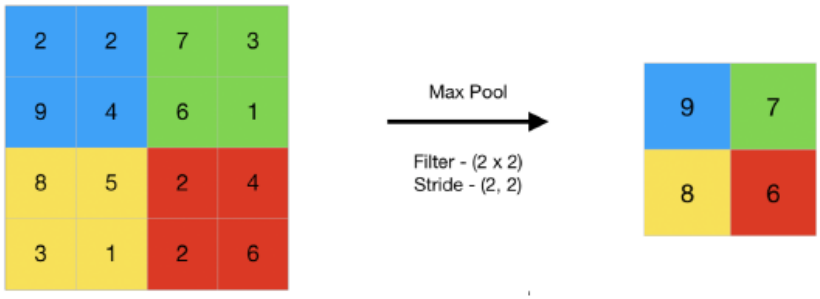


Figure 3.9: Pooling layer [61]

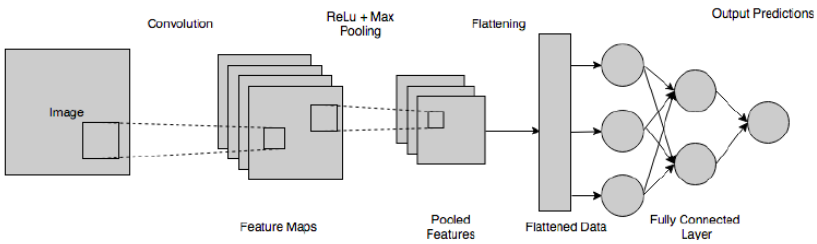


Figure 3.10: Dense layer (fully-connected layer) [109]

3.1.4 Generative Adversarial Networks

3.1.4.1 Overview

Generative Adversarial Networks [38] are systems based on a min-max strategy where two "players" are confronted: one generates data (the generator) and the other discriminates between fake and real data (the discriminator). Specifically, the generator's objective is to maximize the discriminator error while the discriminator wants to do the opposite. This is an iterative process that ends when the discriminator cannot distinguish between the real and the fake. We can think of a GAN as a "Cat and Mouse Game" between a cop and a money counter-feiter, where the counterfeiter (the generator) tries to fool the cop (the discriminator) creating an endless loop where both players keep improving themselves by pure competition.

Here, the loss for GAN (standard GAN [38]) is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_d}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D(G(z))\right)\right] \tag{3.1}$$

where

- p_d is the distribution of real data.
- p_z is the distribution of noise (usually a Gaussian distribution) from which we can generate a fake image.
- x and z are the real data and the input noise respectively.
- $\mathbb{E}[\cdot]$ represent the expectation operators.
- D outputs a real number ranged between 0 and 1 representing the probability for data being real (1) or fake (0). On the other hand, G outputs a generated sample.

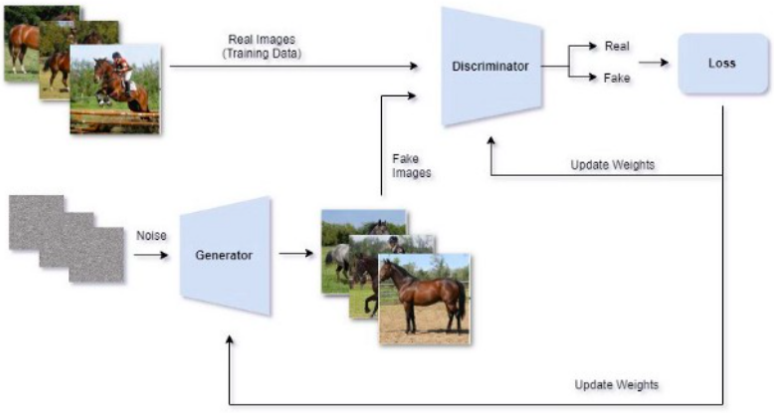


Figure 3.11: Overall architecture of GAN [99]

3.1.4.2 Nash equilibrium of GAN

As described, GAN is basically a zero-sum non-cooperative game. In short, in a zero-sum non-cooperative game, if one wins then others lose. In game theory, the GAN model converges when the discriminator and the generator reach what is called a Nash equilibrium.

In its most general form, the Nash equilibrium is a concept in game theory where an individual can receive no incremental benefit from changing actions, assuming other players remain constant in their strategies. In the case of GAN, Goodfellow et al. [38] has proven that the solution for equation (3.1), if exist, must satisfy the following relation:

$$V(D, G^*) \leq V(D^*, G^*) \leq V(D^*, G) \tag{3.2}$$

assuming D^* and G^* are the respective optimal point for D and G when its corresponding opponent is fixed, while $V(D, G)$ is the above loss (3.1).

Based on the relation (3.2) and some mathematical transformation [38], we can derive that the Nash equilibrium occurs when

$$D(x) = \frac{1}{2} \text{ and } p_d(x) = p_g(x) \ \forall x \tag{3.3}$$

where p_g is the distribution of the generated outputs of the generator.

Intuitively, solving the mini-max loss (3.1) means that we want the discriminator has only 50% chance of successfully detecting the real data because the distribution of the synthetic data is exactly the same with that of the real data.

3.1.4.3 Learning algorithm for GAN

In 3.1.4.2, we have shown that the equation (3.1) have a analytical solution, which depends on p_d (see (3.3)). However, because we do not know p_d beforehand as well as G and D are commonly parameterized as deep neural networks. Therefore, it is impossible to provide a closed-form solution for GAN. To train GAN, iterative methods such as Stochastic Gradient Descent are the only options. In the original GAN paper [38], Goodfellow et al. devise a simple learning algorithm.

Algorithm 2 Algorithm for training GAN

Require: discriminator weights θ_D, generator weight θ_G, input noise distribution p_z, learning rate α, hyperparameter k, batch size B, number of epochs E and dataset x

for epoch := 0 to E **do**

for k steps **do**

- Sample a minibatch of B noise samples $\{z^{(1)}, ..., z^{(B)}\}$ from noise prior p_z
- Sample a minibatch of B examples $\{x^{(1)}, ..., x^{(B)}\}$ from the dataset
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{B} \left[\log D(x^{(i)}) + \log\left(1 - D(G(z^{(i)}))\right)\right]$$

end for

- Sample minibatch of B noise samples $\{z^{(1)}, ..., z^{(B)}\}$ from noise prior p_z.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{B} \left[\log\left(1 - D(G(z^{(i)}))\right)\right]$$

end for

In each iteration, we first freeze the generator's weight and proceed to update the discriminator's weight for a few sub-steps using Gradient Ascent, then we freeze the discriminator's weight and update the generator's weight by using Gradient Descent.

3.1.4.4 Problems with GANs

Mini-max optimization is extremely difficult [101] and training GANs is no different. In fact, GANs suffer from many issues. Specifically, optimizing the GAN's mini-max loss or any other min-max function is extremely unstable and easily diverged. Taking the following min-max problem as a example:

$$\min_y \max_x V(x, y) = xy \tag{3.4}$$

Clearly, the Nash Equilibrium for equation (3.4) is $x = y = 0$; however, if we apply the learning algorithm in 3.1.4.3 with a large learning rate, the model can fluctuate unstably (Figure 3.12). Moreover, recently, MIT researchers [29] prove that GANs may not exist any Nash equilibrium in some cases. Without Nash equilibrium, we must change the loss function or use some other mechanism to help the networks easier to convergence.

The second vital issue of GANs is mode collapse. So what is "mode"? Real-life data distributions are multi-modal, i.e containing many modes. For example, in MNIST, there are 10 major modes from digit '0' to digit '9'. In practice, GANs often fail to capture a variety of modes in the real data but only produces some of the modes. To illustrate, in Figure 3.13, instead of learning to generate full 10 modes like the model in the top row, the model in the second row is only able to produce the images of digit **'6'**.

Besides from the above two issues, GANs also suffer from many other problems and addressing these problems remains active research in the field of AI:

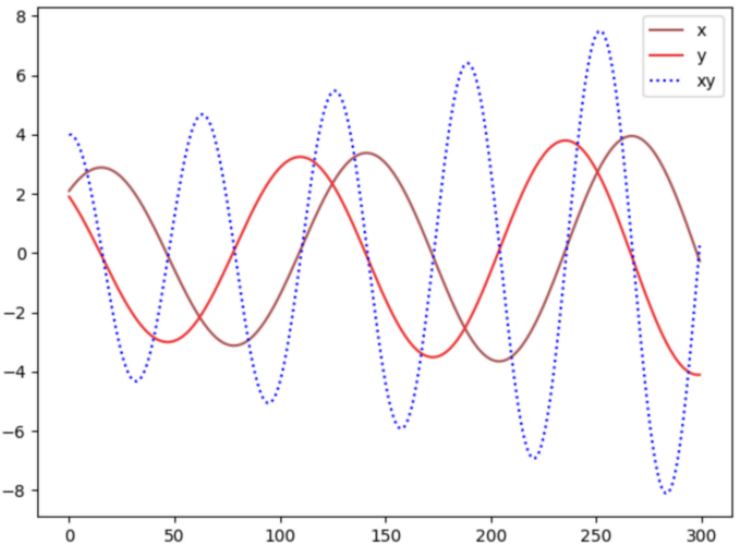


Figure 3.12: The divergence of the example function [49]

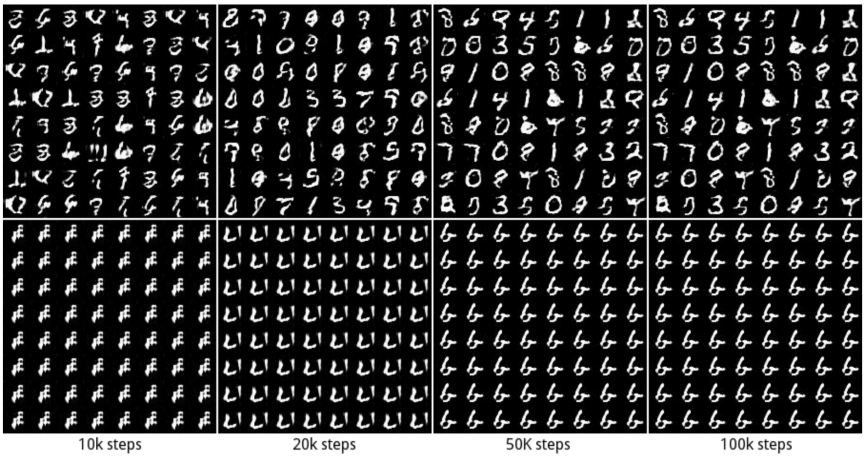


Figure 3.13: An example of mode collapsing [49]

- Diminished gradient: the discriminator gets too successful that the generator gradient vanishes and learns nothing.
- Unbalance between the generator and discriminator causing overfitting.
- Highly sensitive to the hyperparameter selections.

3.2 Single Image Super-Resolution

3.2.1 Overview

3.2.1.1 Pixel

Digital images, are spatially discrete and divided into (usually) rectangular picture elements, or pixels [31]. Specifically, a pixel, in the context of computer vision, is the numerical value of the scalar (gray scale or index) or vector (color or multi-spectral)

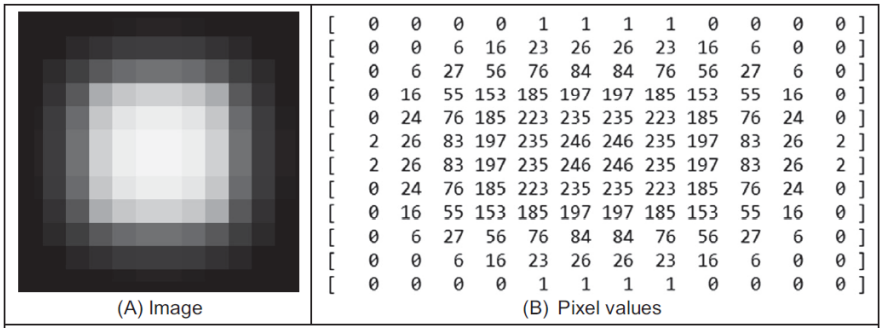


Figure 3.14: The sample image and its corresponding pixel matrix [90]

information at one point in a picture, or image. An image is typically represented as a 2D-array of pixels.

3.2.1.2 Resolution

By definition, image super-resolution refers to the image details [86]. In other words, the higher resolution, the more details are contained in the picture. There are various type of resolution, including pixel resolution, spatial resolution, spectral resolution, temporal resolution, intensity resolution, etc. In Super-Resolution (SR) context, we are mainly concerned about pixel resolution and spatial resolution [86].

- **Pixel resolution**: The term pixel resolution is often considered equivalent to pixel count in digital imaging. Therefore, we can define the pixel resolution using two number. The first number is the number of pixels across the image's width and the second number is the number of pixels the image's height.
- **Spatial resolution**: This term is a measure of the smallest discernible detail in an image. Quantitatively, several ways can be used to measure, such as line pairs per unit distance, or independent dots (pixels) per unit distance being common measures [36].

To illustrate the effect of pixel resolution and spatial resolution, see Figure 3.15

3.2.1.3 Super-Resolution

Super-Resolution (SR) is an important field of image processing techniques to enhance the resolution of images and videos in computer vision. In other words, it try to regenerate High-Resolution (HR) images from one or multiple observed Low-Resolution (LR) image(s) [86]. Super-resolution problem has various type of applications: satellite and aerial imaging, medical image processing, ultrasound imaging, automated mosaicing, infrared imaging, facial image improvement, etc [58]. Super-resolution can be

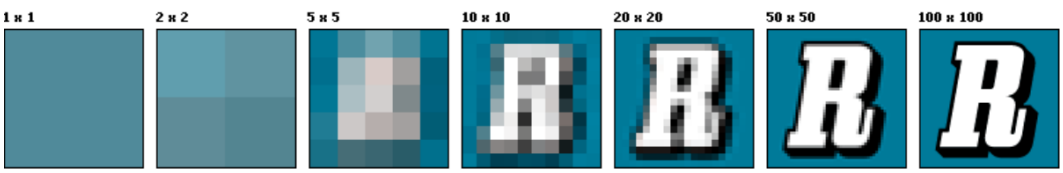


Figure 3.15: The effect of pixel resolution and spatial resolution [30]

confused with similar techniques, such as interpolation, restoration, or image rendering, but they are distinct [58].

There is no official classification of the SR method as it varies from authors to authors. The comprehensive survey [58] classifies SR method into two main groups of frequency domain and spatial domain, whereas a more common division is just the single-image method and multiple-image method [6, 123, 28, 131]. In general, the multiple-image strategy produces a higher accuracy result, as they combine more data extracted from the analyzed scene. However, it is not very practical as multiple images are not always available [28].

3.2.1.4 Single Image Super-Resolution

As mentioned above in chapter 1, in our assignment, we only focus on single-image super-resolution. SISR is a notoriously challenging ill-posed problem as many high-resolution images can downscale to one low-resolution image (see Figure 3.16). Another difficulty limiting the practical application of SISR reconstruction is this technique requires enormous computation resources because of a large number of unknown values, which require expensive matrix manipulations [86].

Recently, the majority of the methods for the SISR problem is example-based. In these methods, the model tries to learn how to map from the low-resolution image to the high-resolution image by using the supervised learning technique. Therefore, when applying to a new low-resolution image, the model can reconstruct its most likely high-resolution version. Although the variant models have been proposed, they share some similar characteristic such as model frameworks, upsampling methods, etc [123].

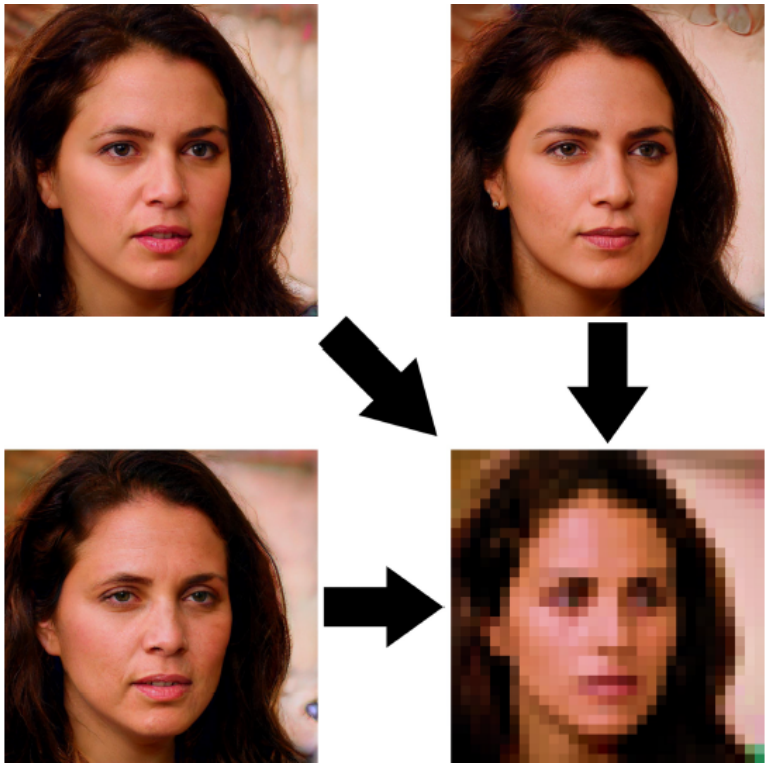


Figure 3.16: Many different HR images can all downscale (represented by the arrows) to the same LR image [84]

3.2.2 Model frameworks

3.2.2.1 Pre-upsampling

In this method, the low-resolution images are first interpolated to obtain a "coarse" high-resolution image. Now, CNNs [68] are used to learn an end-to-end mapping from the interpolated low-resolution images to the high-resolution images. The intuition was that it may be easier to first upsample the low-resolution images using traditional methods (such as bilinear interpolation) and then refine the resultant than learn a direct mapping from a low-dimensional space to a high-dimensional space [123].

The major weakness of this method as it suffers the side effects from a pre-defined upsampling method (e.g blurring). Moreover, the model almost works in high-dimensional space, so it requires large both computational cost and large training time [123].

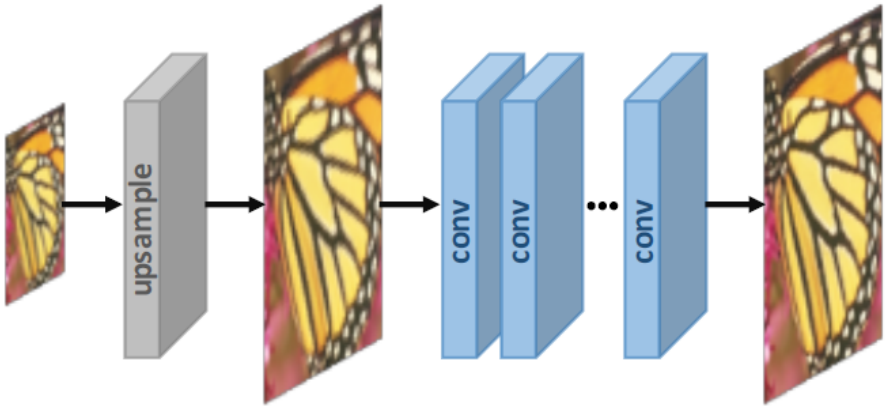


Figure 3.17: Pre-upsampling SR framework. The **gray** one defines pre-determined upsampling and the **blue** one denotes convolutional layer [123]

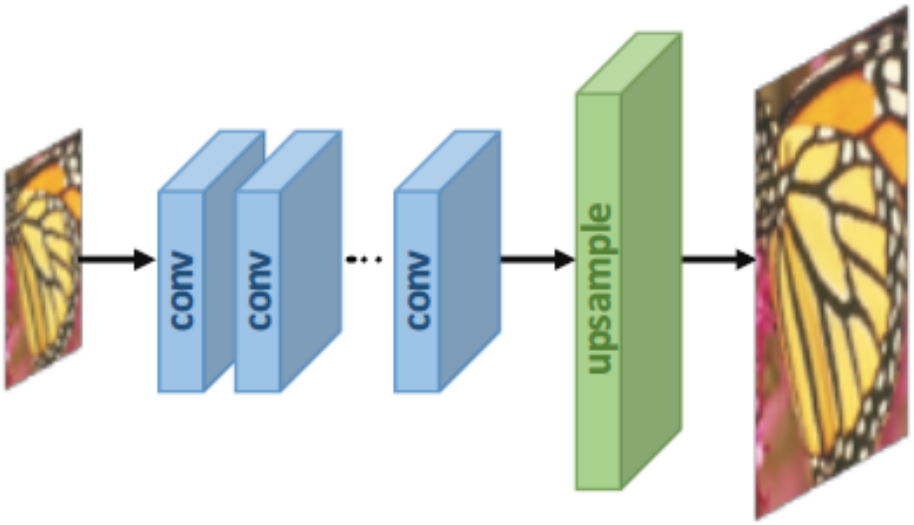


Figure 3.18: Post-upsampling SR framework. The **green** one defines learnable upsampling and the **blue** one denotes convolutional layer [123]

3.2.2.2 Post-upsampling

In this case, the low-resolution images are passed to the CNNs such that upsampling is performed in the last layer using a learnable layer. The advantage of this method is that feature extraction is performed in the lower dimensional space (before upsampling) and hence the computational complexity is reduced. Furthermore, by using a learnable upsampling layer, the model can be trained end-to-end [123].

Despite low computation and spatial complexity, this method still encounters some problems. First, the upsampling operator only applies at the end, which increases the difficulties in case of large scaling factors. In addition, this model requires train many models correspond with many scaling factors. These properties limit the use of this model in real-life application [123].

3.2.2.3 Progressive-upsampling

To address the post-upsampling drawback, a progressive upsampling framework was adopted by works such as Laplacian Pyramid SR Network (LapSRN) and Progressive SR (ProSR) [123]. The models, in this case, use a cascade of CNNs to progressively reconstruct high-resolution images at smaller scaling factors at each step. By decomposing a difficult task into simpler tasks, the learning difficulty is greatly reduced and better performance can be obtained.

This model sidesteps the multi-factor SR problem, but trade-off their model complexity and training stability.

3.2.2.4 Iterative up-and-down sampling

The model name is already self-explanatory. In short, it effectively alternating between the process of upsampling and downsampling. The models under this framework can better mine the deep relations between the LR-HR image pairs and thus provide higher quality reconstruction results. However, the design metric of the back-projection modules is still unclear and very potential to explore further [123].

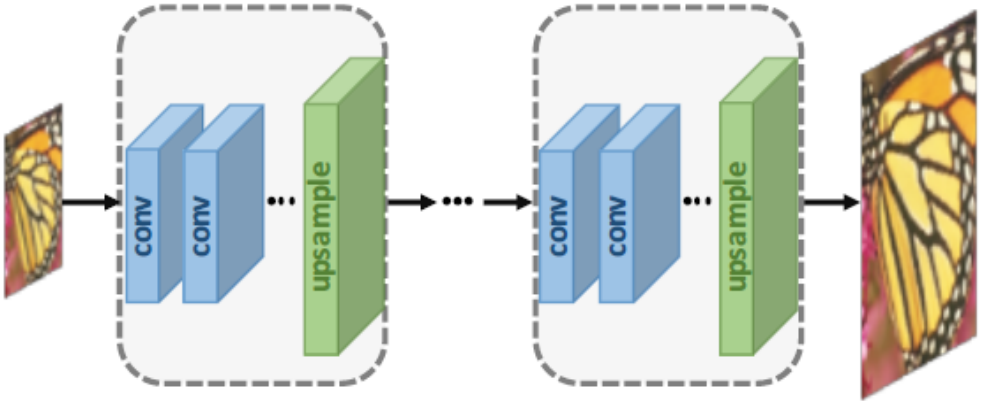


Figure 3.19: Progressive-upsampling SR framework. The **green** one defines learnable upsampling and the **blue** one denotes convolutional layer. And the blocks enclosed by dashed boxes represent stackable modules. [123]

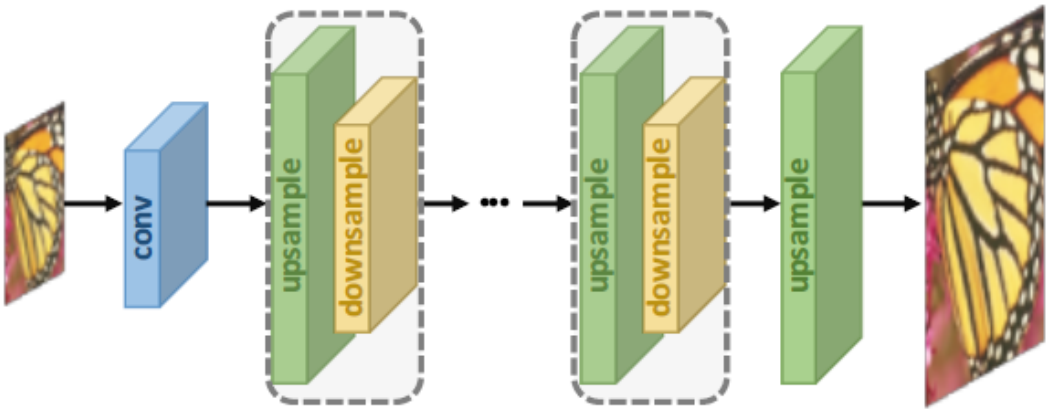


Figure 3.20: Iterative up-and-down sampling [123]

3.2.3 Upsampling methods

At the core of all super-resolution methods is the upsampling operators. Thus, knowing how to perform upsampling is of great importance. Despite many traditional upsampling methods [53, 116, 106], the current trend is to make use of CNN to learn how to end-to-end upsample. In this sub-section, we'll introduce some traditional interpolation-based algorithms and deep learning layers for upsampling.

3.2.3.1 Interpolation-based upsampling

Image interpolation is the traditional approach for image resizing and widely used by many vision applications. The three most popular traditional interpolation methods include nearest-neighbor interpolation, bilinear and bicubic interpolation. Since these methods are highly interpretable and easy to implement, they are still widely used in CNN-based SR models.

- **Nearest-neighbor interpolation** is a simple and intuitive algorithm. It only selects the value of the nearest pixel for each position to be interpolated regardless of any other pixels. Thus this method is very fast but usually produces "blocky" results of low quality.

- **Bilinear interpolation** first performs bilinear interpolation on the x-axis of the image then do the same on the y-axis, as Figure 3.21 shows. This method shows much better performance than nearest-neighbor interpolation while keeping relatively fast speed.

- **Bicubic interpolation** similarly performs bicubic interpolation on each of the two axises, as Figure 3.21 shows. Compared to bilinear interpolation, the bicubic interpolation takes 4×4 pixels into account and results in smoother results with fewer artifacts but much lower speed. In fact, this is the most commonly-used method in SR.

As a matter of fact, the interpolation-based upsampling methods improve the image resolution only based on its own image signals, without bringing any more information. Instead, they often introduce some unwanted artifacts, noise amplification, blurring results. Therefore, the current trend is to replace the interpolation-based methods with learning-based upsampling methods.

3.2.3.2 Learnable upsampling

In order to address the shortcomings of interpolation-based methods and learn upsampling in an end-to-end manner, there are many learning-based upsampling methods introduced into the SR field such as transposed convolution layer and sub-pixel layer.

- **Transposed convolution layer**, a.k.a. deconvolution layer [135] tries to perform transformation opposite a normal convolution, i.e., predicting the possible input based on feature maps sized like convolution output. Specifically, it increases the image resolution and expands the image size by inserting zeros and performing convolution (see Figure 3.22). However, this layer can easily cause "uneven overlapping" on each axis [92], and the multiplied results on both axes further create a checkerboard-like pattern of varying magnitudes and thus hurt the SR performance.

- **Sub-pixel layer** [108], another end-to-end layer, performs upsampling by generating a plurality of channels by convolution and then reshaping them, as Figure 3.23 shows. Within this layer, a convolution is firstly applied for producing outputs with s_2 times channels, where s is the scaling factor (Figure 3.23). Compared with transposed convolution layer, the sub-pixel layer has a larger receptive field, which provides more contextual information to help generate more realistic details. However, it may result in some artifacts near the boundaries of different blocks [34].

- **Meta upscale module**: The previous methods need to pre-define the scaling factors, i.e., training different upsampling modules for different factors, which is not efficient and not in line with real needs. Therefore, Hu et al. [45] propose a

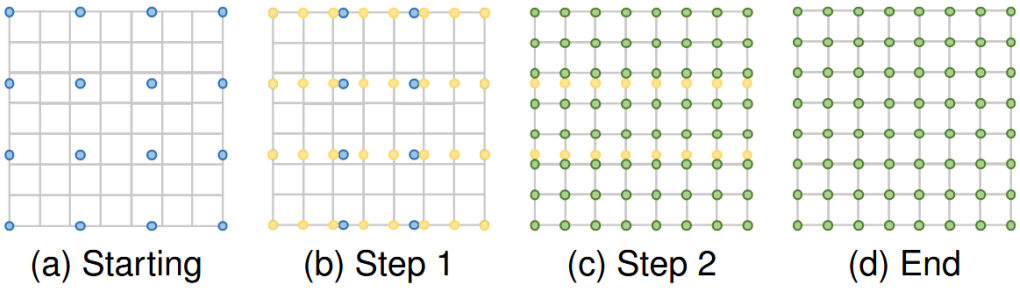


Figure 3.21: Interpolation-based upsampling [123]

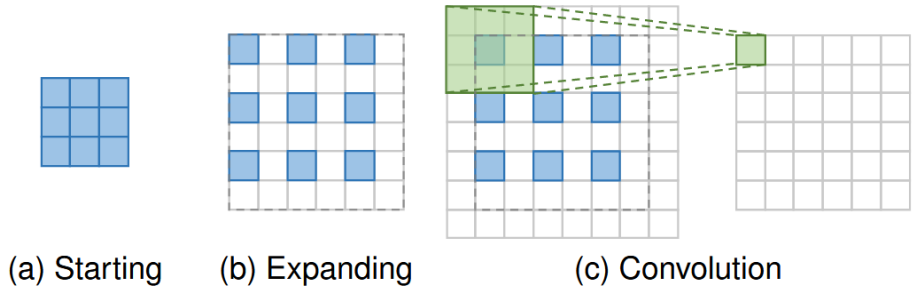


Figure 3.22: Transposed convolution layer [135]

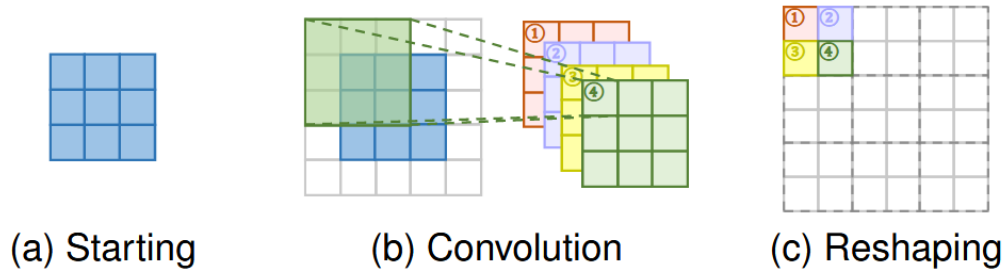


Figure 3.23: Sub-pixel layer [108]

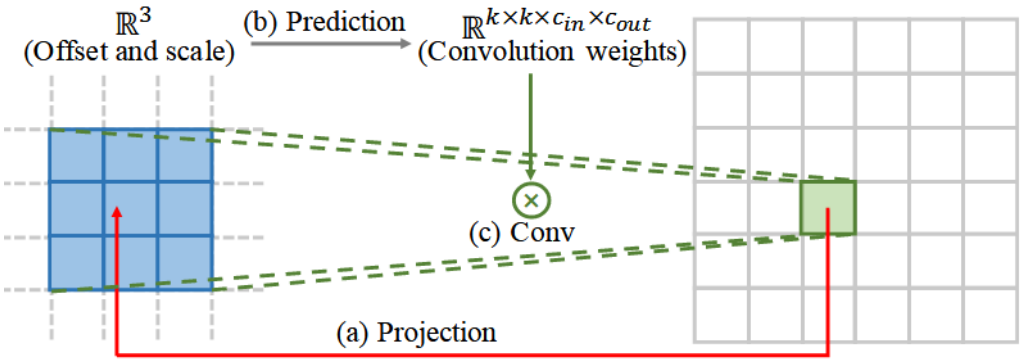


Figure 3.24: Meta-scale layer [45]

meta-upscale module (as Figure 3.24 shows), which firstly solves SR of arbitrary scaling factors based on meta-learning. However, this method needs to estimate a large number of convolution weights for each target pixel based on several values regardless of the image contents, so the prediction result may be unstable and inefficient for super-resolution with a large scale.

Nowadays, these learning-based layers have become the most widely used upsampling methods. These layers are usually used in the final upsampling phase for reconstructing HR images based on high-level features extracted in low-dimensional space, and thus achieve end-to-end SR while avoiding overwhelming operations in high-dimensional space.

3.2.4 Common metrics

3.2.4.1 Overview

In general, super-resolution quality assessment includes accuracy metrics and perceptual metrics. Specifically, accuracy metrics try to evaluate the reconstruction performance (i.e., compare the generated high-resolution image with the ground truth high-resolution image in pixel domain). On the other hand, perceptual metrics nor-

mally pass the image through a deep neural network and compute the distance between the feature vectors. The neural network can be considered as the non-deterministic approach which is similar to the nature of super-resolution. Some recent papers [69, 11] prove that the perceptual metrics more correlate with human opinion than the accuracy metrics. However, the accuracy metrics cannot be totally ignored, as these metrics measure the distortion rate of the image. Two recent perceptual super-resolution competitions [11, 138] used both two kinds of metrics (accuracy and perceptual) to evaluate the quality of submission model. Next, we'll introduce several metrics (both accuracy and perceptual metrics) and give a brief comparison in the final part.

3.2.4.2 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio (PSNR) is one of the most popular reconstruction quality measurement for many different vision applications (e.g., image denoising, image inpainting, image compression). For super-resolution, PSNR is computed via the maximum pixel value (denoted as L) and the Mean Square Error (MSE). Given the ground truth image I with N pixels and the reconstructed image $\hat{I}$, the PSNR between I and $\hat{I}$ can be defined as follows:

$$\text{PSNR} = 10 \log \left(\frac{L^2}{\frac{1}{N} \sum_{i=1}^{N} (I(I) - \hat{I}(i))^2} \right) \tag{3.5}$$

where l commonly equal to 255. Since PSNR only care about the differences between corresponding pixels instead of visual perception, it is not suitable for assessing fidelity of the reconstructed image. However, due to the necessity to compare with literature works and the lack of accurate perceptual-aware metrics, PSNR is still currently the most widely used evaluation criteria for SR models.

3.2.4.3 Structural Similarity Index

Considering that the human visual system (HVS) is highly adapted to extract image structures [122], the Structural Similarity Index (SSIM) [147] is proposed for evaluating the structural similarity between images, in terms of luminance, contrast and structures. For an image I with N pixels, the luminance μ_I and contrast σ_I are defined as the mean and standard deviation of the image intensity, respectively, i.e., $\mu_I = \frac{1}{N} \sum_{i=1}^{N} I(i)$ and $\sigma_I = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (I(i) - \mu_I)^2}$, where $I(i)$ represents the intensity of the i-th pixel of image I. And the comparisons on luminance and contrast, denoted as $C_l(I, \hat{I})$ and $C_c(I, \hat{I})$ respectively, are given by:

$$C_l(I, \hat{I}) = \frac{2\mu_I \mu_{\hat{I}} + C_1}{\mu_I^2 + \mu_{\hat{I}}^2 + C_1} \tag{3.6}$$

$$C_c(I, \hat{I}) = \frac{2\sigma_I \sigma_{\hat{I}} + C_2}{\sigma_I^2 + \sigma_{\hat{I}}^2 + C_2} \tag{3.7}$$

where $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$ are constants for avoiding zero-division, $k_1 \ll 1$ and $k_2 \ll 1$

Besides, the image structure comparison can be computed as follows:

$$\sigma_{I\hat{I}} = \frac{1}{N-1}\sum_{i=1}^{N}(I(i) - \mu_I)(\hat{I}(i) - \mu_{\hat{I}}), \tag{3.8}$$

$$C_s(I, \hat{I}) = \frac{\mu_{I\hat{I}} + C_3}{\mu_I \mu_{\hat{I}} + C_3} \tag{3.9}$$

where C_3 is a constant also for avoiding zero-division.

Finally, SSIM is given by:

$$\text{SSIM}(I, \hat{I}) = C_l(I, \hat{I})^{\alpha} C_c(I, \hat{I})^{\beta} C_s(I, \hat{I})^{\gamma} \tag{3.10}$$

where $\alpha, beta$ and γ are parameter to control the relative importance.

Since the SSIM evaluates the reconstruction quality from the perspective of the HVS, it is more suitable for perceptual assessment and is also widely used.

3.2.4.4 Mean Opinion Score

Mean Opinion Score (MOS) testing is a directed perceptual metric, where human raters are asked to assign perceptual quality scores to tested images. Typically, the scores are from 1(bad) to 5(good). Then, the final MOS is calculated as the arithmetic mean over all ratings. Although the MOS testing seems a faithful method, it has some inherent defects, such as non-linearly perceived scales, biases and variance of rating criteria. In reality, the MOS testing is the most reliable and commonly used method for accurately measuring the perceptual quality [104, 69, 123]

3.2.4.5 Learned Perceptual Image Patch Similarity

Learned Perceptual Image Patch Similarity (LPIPS) [141] consists of two networks correspond with two phases : the large network in the first phase and the small network in the remaining phase.

Figure 3.25 illustrate the architecture and pipeline of this perceptual metric. First, the image is divided into small patches in order to reduce heavy computation. The network receives two inputs: a patch from the generated image and a patch from the reference image and starts to calculate the distance between two above patches (the first phase). The feature vectors are extracted from the deep layers and then normalized across the channel. Next, they scale the channel (vector w in the figure) and compute the least square error (l_2 distance). The final step is get the average across all spatial dimensions and sum channel-wise.

In our thesis, we only use the first phase. The second phase is only used when we want to determine between two patches/images, which patch/image is closer to the

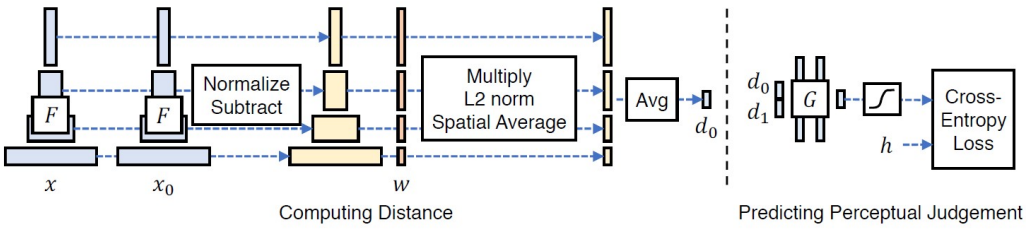


Figure 3.25: LPIPS network [141]

reference patch/image. In short, instead of directly comparing between two distances (two outputs in the first phase), the authors train a small network to map the distances to the perceptual judgement. Normally, in SISR, this stage is redundant. We also note that similar to mean-square error, LPIPS can be used as either metric (in evaluation) or loss function (in training).

3.2.4.6 Fréchet Inception Distance

In 2017, Heusel et al. proposed Fréchet Inception Distance (FID) [43] metric to evaluate the quality of GANs model. It soon became the common metric to determine the performance of GANs, including some recent SOTA such as StyleGAN2 [59]. This metric's idea can easily expand to access the quality of super-resolution model [13].

The main difference of FID metric compares to other metrics in this section is this considers the whole set of images rather than the single image. In this section, we denote the set of images generated by the SISR model is a super-resolution set and the set of the original high-resolution images is a high-resolution set. The authors assume two above sets follow the multidimensional Gaussian distribution. To formula this point, we assume the distribution of super-resolution set and high-resolution set are $\mathcal{N}(\mu_{SR}, \sum_{SR})$ and $\mathcal{N}(\mu_{HR}, \sum_{HR})$ respectively.

Similar to LPIPS, FID feeds the high-resolution set and super-resolution set into the deep network (normally Inception v3, but not compulsory). FID score is calculated based on Wasserstein distance:

$$\text{FID} = \|\mu_{SR} - \mu_{HR}\|_2^2 + \text{Tr}(\sum_{SR} + \sum_{HR} - 2(\sum_{SR}\sum_{HR})^{1/2}) \tag{3.11}$$

As can be seen from the formula, while other metrics direct deals with value in pixel domain or feature domain, FID deals with mean and covariance. The two most important properties of FID are correlate well with human perceptual judgment and sensitive to distortion [12]. To illustrate this point, please see Figure 3.26.

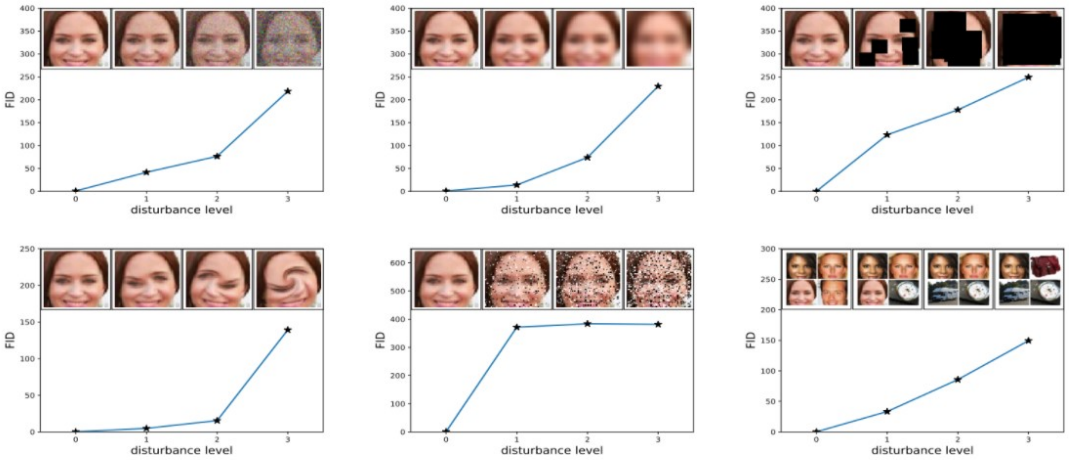


Figure 3.26: FID score correspond with different type of noise, including **upper left:** Gaussian noise, **upper middle:** Gaussian blur, **upper right:** implanted black rectangles, **lower left:** swirled images, **lower middle:** salt and pepper noise, and **lower right:** CelebA dataset contaminated by ImageNet images. Figure from [43]

3.2.4.7 Blind/Referenceless Image Spatial Quality Evaluator

In many computer vision tasks, it is difficult to have a reference image. For example, in real-world super-resolution or video-resolution, pictures are taken by the camera normally contain various distortion. Those distortions are normally caused by environmental factors, which means that they cannot totally avoid . The lack of completely-clean reference image in those situations rise the necessity of the image quality assessments without any ground truth information, namely no-reference IQA methods [147].

No-reference IQA methods can be categorized as opinion-aware (OA) methods and opinion-unaware methods (OU). BRISQUE [87] is the opinion-aware method as this metric uses human-rated distortion datasets for training (LIVE IQA). First, in this section, when we refer to the natural image, it does not strictly mean that the image contain only natural elements such as trees or animals. In fact, any image taken by the optical camera and is not suffered from any digital image processing can be understood at natural image.

The three key strategies in the BRISQUE metric including: natural scene statistic (NSS), mean subtracted contrast normalized (MSCN) and asymmetric generalized Gaussian distribution/ generalized Gaussian distribution (AGGD/GGD) model. First, they compute the locally normalized luminances and transform them use MSCN operator. This normalization procedure can significantly reduce the correlation between the pixel and 8-neighbor corresponding. Natural scene statistics based on the special

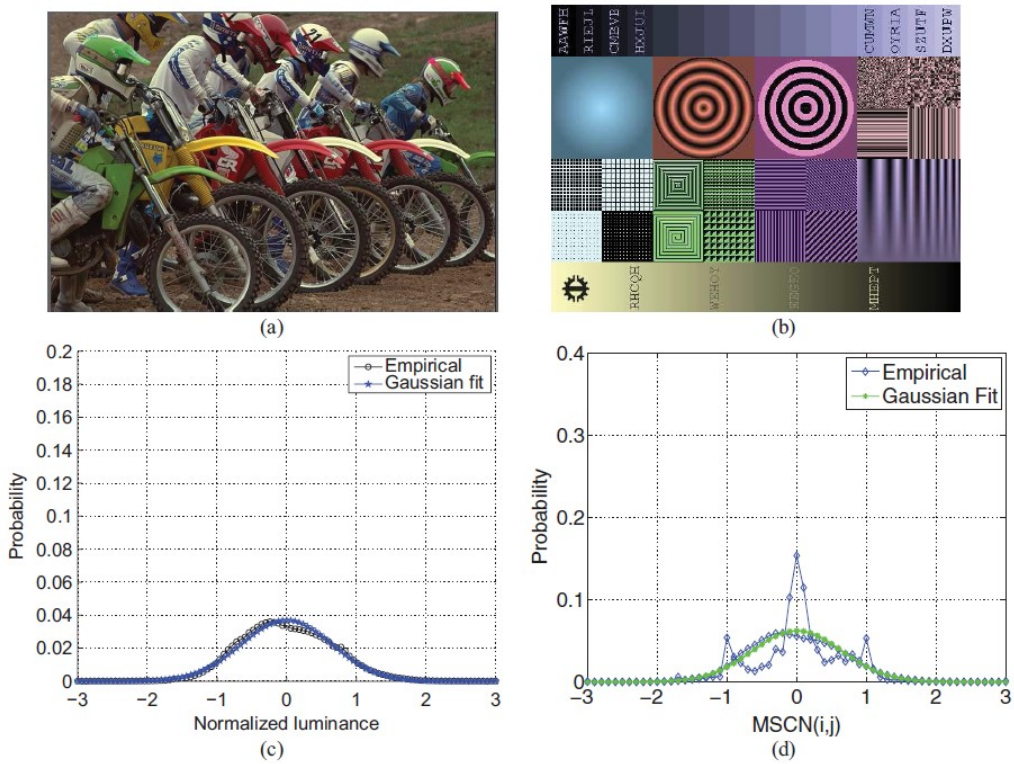


Figure 3.27: Natural scene statistic property **(a)** Natural image, **(b)** Artificial image, **(c)** Normalized luminance coefficients follow a nearly Gaussian distribution for the natural image (a). **(d)** This property does not true for the artificial image. Figure from [87]

statistical properties holds on the natural image. To illustrate this point, please see Figure 3.27.

Combining NSS properties and the principle of information theory, some previous models extract the NSS features in the wavelet domain or spatial domain to get the perceptual judgment [8]. BRISQUE shares a similar strategy with prior works, after extracted NSS features in form of MSCN coefficients and their pairwise products, they fit those values with GGD. Further, they also model the statistical properties between adjacent pixels using AGGD and extract features with two scale to make the result more reliable. The author also proved that NSS features form the pristine and distorted image are well-separated in GGD and AGGD space [87], allow those can be easily classified by a simple model. Therefore, BRISQUE use the support vector machine to match features to perceptual score.

3.2.4.8 Natural Image Quality Evaluator

Unlike BRISQUE in the above section, Natural Image Quality Evaluator (NIQE) [1] is an opinion-unaware (OU) method. At first, NIQE divides the image into small patches. Next, they use a similar strategy with BRISQUE including extract NSS features, model the statistical relationship by using GGD and AGGD, compute at two scales and four orientations. The major difference is that in lieu of using the obtained features in the above steps to train the regression model, they fit them to a multivariate Gaussian model. Finally, the NIQE score is computed by the distance from a universally learned multivariate Gaussian model of pristine natural images. The absence of human-rated distortion datasets in training procedure makes NIQE become a pioneer opinion-unaware metric [8].

3.2.4.9 Comparison

PSNR and SSIM is the traditional methods to access the quality of the model not only in super-resolution but also in various other image restoration tasks: image deblurring, image denoising, etc [60]. As those metrics evaluate the reconstruction accuracy directly in the pixel domain, they require less memory and computation compare to perceptual metrics. However, some papers prove that PSNR and SSIM do not correlate well with human-opinion score [69, 11]. Both Figures 2.13 and 3.28 provide the visual example for this point.

The accuracy-driven model has been studied and researched for a long time (NTIRE mainly focuses on PSNR/SSIM-driven model), however, perceptual models receive less attention. Therefore, the analysis of perceptual metrics is very limited. Recently, in PIRM 2018 challenge, Blau et al. conduct some experiments to analyze the correlation between some perceptual metrics and human opinion. The experiments include two regimes:

- **Normal perceptual quality regime:** The authors consider Spearman's correlation between mean-opinion scores and some common perceptual metrics.
- **High perceptual quality regime:** The authors zoom all the pictures in previous experiment into a larger scale and conduct the experiment again.

According to Figure 3.29, on a coarse scale, SSIM is anti-correlated with the human-opinion score. Also, LPIPS and NIQE show a high moderate correlation with human

Figure 3.28: Inconsistency between PSNR/SSIM values and perceptual quality. From left to right: nearest-neighbor (NN) interpolation, SRResNet [69] which aims for high PSNR, and SRGAN [69] which aims for high perceptual quality. The perceptual quality of SRGAN is far better than SRResNet. However, its PSNR/SSIM values are substantially lower than those of SRResNet, and even lower than those of NN interpolation. Figure from [11]

perception, while BRISQUE just moderate-correlated. However, on a finer scale, only NIQE is well-correlated. More experiments can be found in [11], but they provide similar trend with the above result.

We cannot find any prior comparison between FID and other perceptual metrics: LPIPS, BRISQUE and NIQE. As we mentioned above, previous work prove the well correlation between FID and human perception [43, 59]. Although both FID and NIQE are well-correlated, they are not perfect perceptual scores. FID assume the data distribution is Gaussian, which is not alway true. Further, in StyleGANv2 [59], Karras et al. shown that FID captures texture rather than shape. Consequently, FID cannot perfectly capture all aspect of image quality. About NIQE, a recent paper [5] shown counterexample of this metric, as can be seen in Figure 3.30. Those analysis highlights the urgent need for better perceptual quality metrics in future.

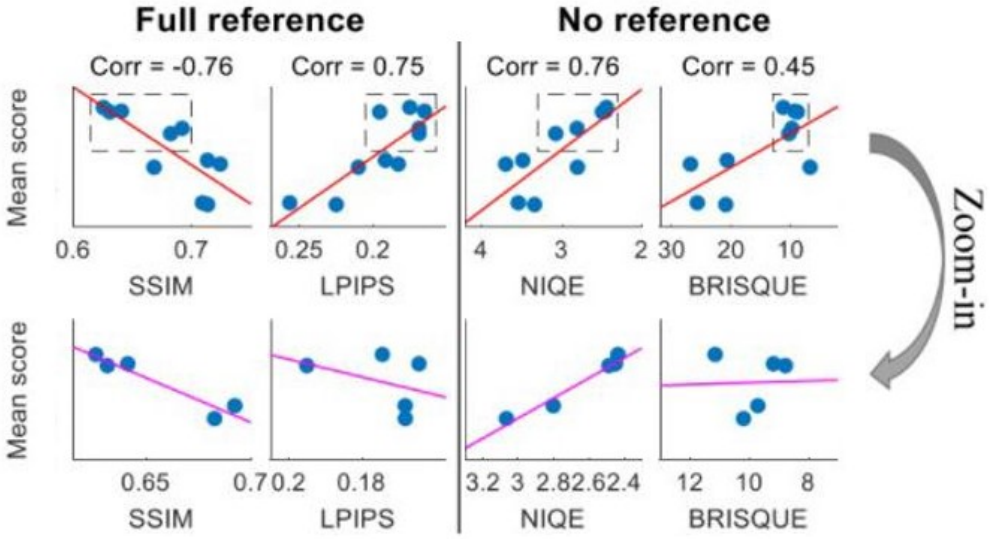


Figure 3.29: Analysis of image quality measures. First/second row : Normal/high perceptual quality regime. y-axis: mean-opinion score, x-axis: score from perceptual metric, the line is the corresponding least-squares linear fits in magenta. Figure from [11]

Figure 3.30: Inconsistency between NIQE score and perceptual quality. The NIQE for left picture is 3.24 and for right picture is 4.40. However, right picture have better visual quality. Figure from [5]

3.3 Frequency-domain processing

3.3.1 Frequency domain and spatial domain

While the spatial domain considers images in their normal two/three-dimensional space (gray-scale/RGB color images), the frequency domain mainly applies some kind of transformation (Fourier transformation, wavelet transformation, etc) and considers the sinusoidal relationships of the outputs. In other words, the spatial domain focus on the pixel value, whereas the frequency domain concentrates on the rate of change of pixel values. There are several reasons why we need to consider the frequency domain. First, several concepts in computer vision cannot be explained without the context of the frequency domain such as the sampling theorem or aliasing [36]. Secondly, some phenomenon is hard to observe in the spatial domain, even with the human observer as we can see in Figure 2.7. Some other examples can be found in [54, 32, 27]. Finally, several authors prove various applications of the frequency domain. Some noticeable results are: image compression, solving differential equations, 3D shape regression [115], etc.

The most essential characteristic is that we can transform the original image into the frequency domain and go back by inverse function without loss of any information. This property allows us flexible convert and work between two representations. In our thesis, we mainly focus on Fourier transform as it is a fundamental concept in frequency-domain processing.

3.3.2 Fourier transform

3.3.2.1 Overview

In the mathematical domain, Fourier analysis discovers how to decompose a complex function into multiple simpler trigonometric functions. The cornerstone of Fourier analysis is the Fourier series, which claims that any periodic real-value function (but with the finite area under the curve) can be equivalent express by the discrete weighted sums of sines/cosines functions. Later research named Fourier transform further generalize this property in the case of non-periodic function (see Figure 3.31). Currently, because of its wide applicability in many theoretical and practical disciplines, the vast majority of work in frequency-domain processing utilizes Fourier transform. In our thesis, we consider three important transform in digital image processing including discrete Fourier transform, fast Fourier transform and discrete cosine transform.

3.3.2.2 Discrete Fourier transform

The most important discrete transform in digital image processing is Discrete Fourier Transform (DFT). This is a complex-valued function transforms a discrete finite sequence of data points into another sequence of complex numbers. Note that in general, the input of DFT function can be complex. However, as digital image can be considered as a two-dimensional real-valued matrix, we can apply the formula of DFT without any modification. The 2D discrete Fourier transform follows the equation:

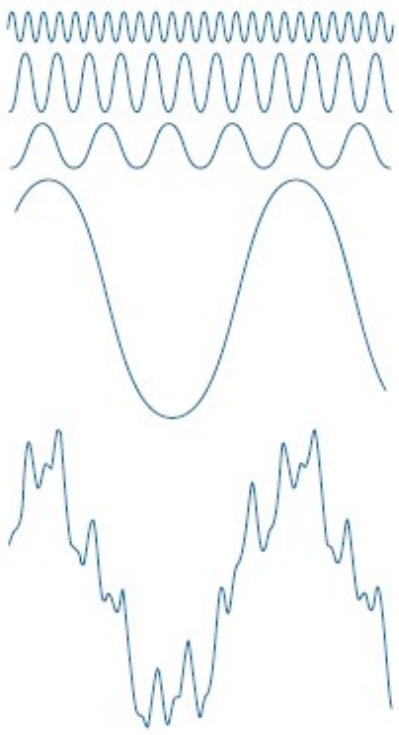


Figure 3.31: Fourier transform illustration. The function at the bottom is the sum of the four functions above it. Figure from [36]

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)e^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{3.12}$$

In equation (3.12), the image size is $M \times N$. $I(m,n)$ denotes the pixel value of this image, where m and n is the coordinate of this pixel. The sigma $\sum$ means that equation (3.12) must be computed from values of two variables u and v, where u from 0 to *M-1* and v from 0 to *N-1*. Note that we use (x,y) for spatial variables and (u,v) for frequency-domain variables. Finally, e and i are popularly known as Euler's number and the imaginary unit, respectively. For further analysis and how to yield this equation, please consider [36].

Furthermore, from the Euler 's formula on complex numbers:

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{3.13}$$

Combine two above equations, we obtain:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)\Big[\cos 2\pi\Big(\frac{ux}{M}+\frac{vy}{N}\Big) - i\sin 2\pi\Big(\frac{ux}{M}+\frac{vy}{N}\Big)\Big] \tag{3.14}$$

As can be seen from equation (3.14), the digital image is expressed as the weighted sum of cosine and sine functions, which correspond with the real and the imaginary components of frequency representation. We notice that frequency-domain variables (u,v) play a crucial role in this formula. For more details about the property of 2D DFT, please consider [36, 54, 27].

3.3.2.3 Fast Fourier transform

In practice, directly computing the equation (3.12) or (3.14) is relatively slow as the high computational requirements. In equation (3.12), we need approximately $M \times N$ multiplications and additions to convert the pixel value in spatial domains to frequency-domain value. With the image size $M \times N$, the total complexity of brute-force implementation is $O((MN)^2)$. The Fast Fourier Transform (FFT) provides a faster computation by reducing this complexity to $O(MN\log_2(MN))$, which increases the practical value of those above equations. There are many FFT algorithms, the most popular choice is the Cooley-Tukey algorithm. It factorizes the DFT matrix into a product of sparse elements and applies the recursion strategy to obtain the desired result. In our thesis, we do not focus on the detail of the Cooley-Tukey algorithm, as many built-in functions in PyTorch had already implemented this algorithm.

3.3.2.4 Discrete cosine transform

The Discrete Cosine Transform (DCT) performs a similar function with DFT, but it uses only real-value functions. The formula for DCT is:

$$D(u,v) = \frac{2}{\sqrt{MN}}C(u)C(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)\cos\left[\frac{\pi}{M}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{N}\left(y+\frac{1}{2}\right)v\right] \tag{3.15}$$

where $C(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if x} = 0, \\ 1 & \text{if } x > 0 \end{cases}$ and other notations are same as in 3.3.2.2.

We can easily realize the absence of an imaginary part in the equation (3.15). Further, the magnitude of the coefficient in the above equation represents how much the spatial pixel contributes to the whole frequency-domain matrix.

3.3.3 Power spectrum

In general, applying equation (3.14) to digital images result in complex arrays. In other words, we can rewrite $F(u, v)$ in equation (3.14) as:

$$F(u, v) = \mathrm{Re}(u, v) + \mathrm{Im}(u, v)i \tag{3.16}$$

Then, the magnitude of $F(u, v)$ is called the frequency spectrum:

$$|F(u, v)| = \sqrt{\mathrm{Re}^2(u, v) + \mathrm{Im}^2(u, v)} \tag{3.17}$$

and the phase angle/spectrum is defined by:

$$\phi(u, v) = \arctan\left(\frac{\mathrm{Im}(u, v)}{\mathrm{Re}(u, v)}\right) \tag{3.18}$$

Both the frequency spectrum and phase spectrum are required to reconstruct the original image from the frequency information. Figure 3.32 demonstrate how each spectrum contribute to image reconstruction.

Also, we further define the 2D power spectrum by using the formula:

$$P(u, v) = |F(u, v)|^2 \tag{3.19}$$

In above equations, $|F(u, v)|$, $\phi(u, v)$ and $P(u, v)$ are the 2D matrices with size $M \times N$. Furthermore, recent works [19, 27] highlight using the 1D power spectrum can reduce the dimension without significant effect the ability to emphasize the frequency difference between real and generated images. First, we need to convert the spectral variables from Cartesian coordinates to polar coordinates:

$$F(r, \theta) = F(u, v) : r = \sqrt{u^2 + v^2}, \theta = \arctan\left(\frac{v}{u}\right) \tag{3.20}$$

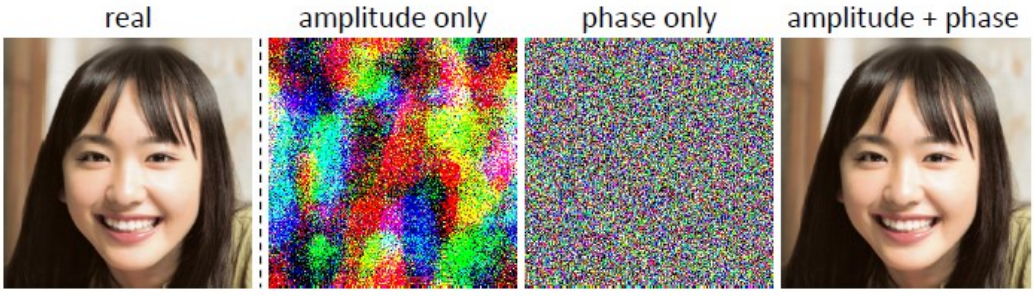


Figure 3.32: Reconstruct the image from frequency information. The necessity of both amplitude and phase information for a frequency distance verified by single-image reconstruction. Figure from [54]

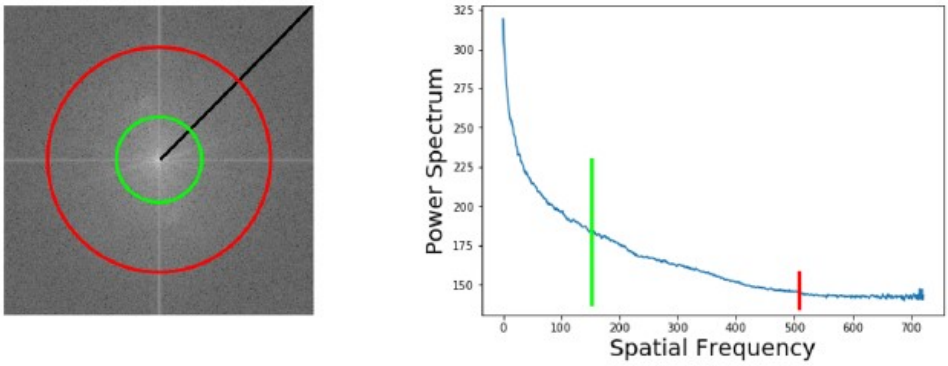


Figure 3.33: Example for the azimuthal integral (AI). **Left:** 2D Power Spectrum of an image. **Right:** 1D Power Spectrum: each frequency component is the radial integral over the 2D spectrum (red and green examples). Figure from [27]

From equations (3.20) and (3.19), it is clear that $P(r, \theta) = P(u, v)$. Finally, we achieve the reduced power spectrum by taking the integration in polar coordinates:

$$AI(r) = \int_0^{2\pi} P(r, \theta) d\theta \tag{3.21}$$

For illustration, Figure 3.33 [27] visualizes the processing step.

Chapter 4

Proposed Approach

The first goal of our thesis is to improve the image quality of ESRGAN. Not only that, we also aim to make the model diversity-aware by introducing additional latent input into the model. This chapter describes our method to obtain these above targets.

For a better illustration of ESRGAN's limitation, please follow our earlier experiments in section 5.1.

4.1 Analyzing and improving the image quality of ESRGAN

4.1.1 The visual quality result of ESRGAN

Publishing in 2018, ESRGAN achieved the best perceptual score in x4 perceptual super-resolution challenge [11]. Recently, in NTIRE 2020 challenge on perceptual extreme super-resolution (x16) [138], some variant works inspired by ESRGAN also obtained the highest-ranking perceptual score. Moreover, until now, according to "Papers with code" website[2], ESRGAN is still one of the most efficient perceptual SR models (top 3). Although the result of ESRGAN is very encouraging, as we mentioned above in chapter 2, the visual quality of GANs model suffers some problems. Of course, ESRGAN cannot bypass all problems, which is already proved in some previous work [93, 11]. Therefore, we hope our works can make ESRGAN produces more perceptually pleasing images. To obtain this goal, we analyze and redesign the learning strategy of ESRGAN. The details of our approach are discussed in the next section.

4.1.2 Proposed approach

4.1.2.1 Overview

Besides network architecture, the loss function plays a crucial role to provide high-quality results [123]. By defining a target to reach through training procedure, loss function represent learning strategies to guide model optimization. Without robust learning strategies, the network cannot reach its full potential. Some existing works already achieved the outstanding results (compared with baseline model) by using their

[2] https://paperswithcode.com/

novel loss without changing network architecture [41, 57, 55]. From those reasons, developing a more effective loss for ESRGAN is the challenging and demanding task.

4.1.2.2 Baseline ESRGAN loss

In this section, we briefly summarize the original loss function of ESRGAN. In general, ESRGAN loss can be divided into three overlap tasks:

- **Perceptual loss:** Encourage the generated image to have similar feature representations as the referenced image
- **Content/pixel loss:** Encourage the generated image to as same as the referenced image in pixel domain (as much as possible)
- **Adversarial loss:** Guide generator learn based on the feedback of the discriminator

Equation (4.1) shows the total generator loss of the baseline model [120]. We also note that, in our thesis, we usually change some notations for convenience. However, our modification does not affect the original formula.

$$L_G^{original} = L_{percep}^{VGG} + \lambda_G \times L_G^{RaGAN} + \eta \times L_1 \tag{4.1}$$

where L_{percep}^{VGG}, L_G^{Ra} and L_1 represent perceptual loss, adversarial loss for the generator and content/pixel loss, respectively. λ_G and η is the hyper-parameter to balance the loss component.

ESRGAN's authors defined the perceptual loss (L_{percep}^{VGG}) based on the distance between two features of the VGG network. We already analyze their novel loss in section 2.6.4.

L_1 loss is defined by pixel-wise error between the ground-truth image and the generated image:

$$L_1 = \frac{1}{B}\sum_{b=1}^{B} \left\| G(I^{LR}) - I^{HR} \right\|_1 \tag{4.2}$$

L_G^{RaGAN} and its discriminator counterpart L_D^{RaGAN} are determined based on the relativistic GAN [57]:

$$L_G^{RaGAN} = -\frac{1}{B}\sum_{b=1}^{B} \Big(\log\left(1 - D_{RaGAN}(I^{HR}, G(I^{LR}))\right) + \log D_{RaGAN}(G(I^{LR}), I^{HR}) \Big)$$

$$L_D^{RaGAN} = -\frac{1}{B}\sum_{b=1}^{B} \Big(\log D_{RaGAN}(I^{HR}, G(I^{LR})) + \log\left(1 - D_{RaGAN}(G(I^{LR}), I^{HR})\right) \Big)$$

(2.4 revisited)

In equation (4.2) and (2.4) , B denotes the batch size. I_{LR}, $G(I_{LR})$, I_{HR} is the low-resolution image, high-resolution image created by the model and the referenced high-resolution image, respectively. Finally, $D_{RaGAN}(x, y) = \sigma(C(x) - \frac{1}{B}\sum_{b=1}^{B} C(y))$ where σ stands for the sigmoid function and $C(x)$ is the non-transformed discriminator output.

To summarize, we have already quoted the mathematical formula (with little modification on notation) of ESRGAN loss for both discriminator and generator. Next, we will analyze each component of the original loss and develop a more effective version.

4.1.2.3 Pixel loss analysis

As mentioned above, the most common choices for pixel loss is L_1/L_2 loss [11]. Because both two loss aim to measure the pixel-wise similarity, they normally increase the reconstruction accuracy score (PSNR/SSIM). In our problem, as discussed earlier in section 3.2.4.9, the accuracy score should be balanced with the perceptual score. However, using L_1/L_2 loss has proved to reduce the model's ability to learn the texture [11]. As a consequence, many authors proposed various alternative options [11, 55] but their effectiveness is marginal.

In our opinion, although L_1/L_2 strategy has a bad influence on the model, we still use L_1 loss as the perceptual loss in our design (same as baseline ESRGAN). There are two main reasons for our decision. First, most state-of-the-art models use either L_1 or L_2 loss in their learning strategies [123], imply no current loss can outperform L_1/L_2 loss. Secondly, both ESRGAN [120] and SRFeat's [93] authors already proved the benefit of pixel-wise loss in training GAN. Specifically, pre-training with L_1 loss reduce the undesired artifacts and boost the model's performance [120]. To illustrate this point, please see our experiment in section 5.1.2.

4.1.2.4 Proposed perceptual loss

From equation (4.1), it is obvious that the perceptual loss is crucial (highest weight) for perceptual learning strategy. Both SRGAN [69] and ESRGAN [120] use VGG-perceptual loss and achieve the promising result. However, some experiments [55, 138] in a finer scale (x16) elucidate the weakness of VGG-based loss. Since a VGG model is trained for image classification problem, this model may not optimal for the super-resolution task. Jo et al. [55] report VGG network tend to produce hallucinated artifact and weakly capture exact details.

For all the reasons above, we try to find a proper perceptual loss. Recently, applying a full-reference Image Quality Assessment (IQA) general enhance the perceptual quality in various computer vision tasks [60]. Ding et al. [60] make a comparison between 11 different IQA models in image denoising, super-resolution, image deblurring and image compression. Their results point out that LPIPS [141] and DISTS [26] outweigh other IQA models in all above tasks. The example experiment is reported in Table 4.1. For more experiments and the summarized information for each IQA model, please refer [60]. About LPIPS and DISTS, we have already recapped some main points in section 2.3.1. Also, the main pipeline of LPIPS can be found in section 3.2.4.5.

Beside existing experiments in [60], we further conduct experiments by ourselves. The details of our experiments can be found in section 5.2.3.1. As a result from above experiments, we replace the VGG-perceptual loss by LPIPS-perceptual loss:

$$L_{percep}^{LPIPS} = \sum_{i=1}^{k} \tau^i \Big(\phi^i(G(I^{LR})) - \phi^i(I^{HR}) \Big) \tag{4.3}$$

In equation (4.3), ϕ denotes the feature extractor and τ denotes the transform operator which use to convert deep representation to LPIPS score. i is the layer i and k is the total number of layers in the network. We notice two extra important things about this loss. First, not only their network architecture and learning strategies, LPIPS also uses a training dataset with human perceptual judgements. Therefore, LPIPS loss proper reflects the human opinion preferences rather than VGG-based loss.

IQA Model	Denoising	Deblurring	Super-resolution	Compression
MAE [145]	0.527	0.164	0.309	0.455
MS-SSIM [124]	**0.564**	0.127	0.455	0.346
VIF [107]	0.273	0.600	0.418	0.018
CW-SSIM [126]	0.382	0.418	0.091	0.018
MAD [67]	0.418	0.455	0.346	0.382
FSIM [140]	0.236	0.054	0.091	0.127
GMSD [129]	0.091	0.018	0.127	0.127
VSI [139]	0.164	0.018	0.018	0.091
NLPD [66]	0.491	0.127	0.200	0.309
LPIPS [141]	**0.709**	**0.855**	**0.782**	**0.782**
DPIPS [26]	0.346	**0.891**	**0.782**	**0.855**

Table 4.1: Spearman's rank correlation coefficient of model raking score and mean opinion score. The bold value is two best results. Table from [60]

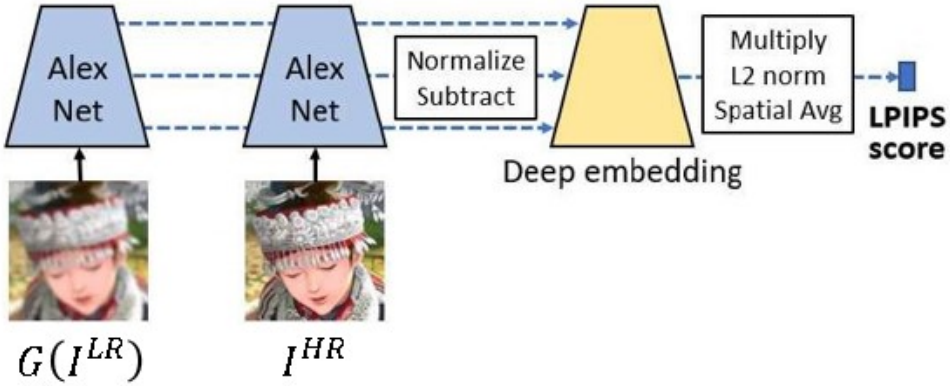


Figure 4.1: LPIPS loss illustration. Figure from [55], but has little modification.

Secondly, as LPIPS consists of many configurations, we use **AlexNet** for our thesis, following surveys and recommendations from the original paper. Figure 4.1 provides a better illustration for this loss.

4.1.2.5 Proposed adversarial loss

As we mentioned above, ESRGAN [120] uses the relativistic discriminator rather than the normal discriminator. They also proved using relativistic discriminator help the model learns finer details such as edges and textures. However, there are two reasons why we think the relativistic average GAN loss is not an optimal solution for our problem. First, ESRGAN uses RaGAN loss proposed by Martineau in 2018 [57]. Recently, in 2020, Martineau proposed some other variants for Relativistic GAN. However, the experiments of those variants are very limited. As a result, we raise the natural question "May any variants of relativistic GAN is the better choice for super-resolution ?". Secondly, as observed from our earlier experiments in section 5.1.2.2, the relativistic discriminator largely depends on the hyper-parameters. As a consequence, the model is hard to converge or extend.

In section 2.3.2, we proposed six alternative option for GAN loss inspired by Relativistic GAN and Wasserstein GAN (equation RaHinge (2.6), RaLS (2.5), RcGAN (2.7), RcHinge (2.9), RcLS (2.8) and WGAN-GP (2.12)). During our experiments, we

devises a new loss which can get more impressive results. This loss, which we name RaGP, get benefit from both high quality images of RaGAN and training stability from WGAN-GP. The mathematical formula of RaGP is:

$$\begin{aligned} L_G^{RaGP} &= L_G^{RaGAN} \\ L_D^{RaGP} &= L_D^{RaGAN} + \lambda_g(\left\|\nabla_{\hat{I}} D_{RaGP}(\hat{I})\right\|_2 - 1)^2 \end{aligned} \tag{4.4}$$

In above equation, the definition of L_G^{RaGAN} and L_D^{RaGAN} are as same as equation (2.4). The image $\hat{I}$ sampled form I^{HR} and $G(I^{LR})$ with t uniformly random between 0 and 1 : $I = tG(I^{LR}) + (1-t)(I^{HR})$. Moreover, we introduce new hyper-parameter λ_g and the special discriminator D_{RaGP}. On the one hand, D_{RaGP} behaves like the discriminator as in the original relativistic GAN [57], if D_{RaGP} receives two inputs. In other words, $D_{RaGP}(x, y) = D_{RaGAN}(x, y) \forall x, y$. On the other hand, D_{RaGP} behaves like the critic as in WGAN-GP [41], if D_{RaGP} receives only one inputs. In mathematical form, this means with $C(x)$ is the non-transformed discriminator output, we have $D_{RaGP}(x) = C(x)$. As can be seen from the equation (4.4), besides the GAN loss in ESRGAN, we add the gradient penalty to enforce the 1-Lipschitz constraint during the training phase.

4.1.2.6 Proposed frequency regularization

In section 2.4, we inspect the frequency artifact problem which normally occurs in the GAN-based models. Through our survey, there are three main strategies to alleviate this issue. In our opinion, regularizing loss function is a simple but effective method. Regardless of data or architecture, this approach can directly apply in any model without significant modification. Therefore, we develop a suitable frequency penalty loss function for the super-resolution task.

At the beginning, we convert both the real image I^{HR} and the generated image $G(I^{LR})$ to the frequency domain by using the equation (3.14). We assume after this step, we receive two outputs: $F(I^{HR})$ and $F(G(I^{LR}))$. We also notice that $F(I)$ means apply equation (3.14) for all pixels in an image I. Because we hope the frequency features of real and fake images are similar, the regularization loss should be the difference between those frequency features:

$$F_{diff} = F(G(I^{LR})) - F(I^{HR}) \tag{4.5}$$

However, as we mentioned in section 3.3.3, normally all components in equation 4.5 are complex tensors. Because most of the built-in functions in Pytorch only consider the real-value input, we need to modify the equation (4.5). To achieve this task, we define some operators on tensor T:

- $\sqrt{T}$ returns the new tensor which each elements is the square root of the corresponding elements in the original tensor T. Note that this operator domain includes complex number. In algebraic form, the square root [21] of the complex number $z = x + yi$ is: $\sqrt{z} = \sqrt{\frac{\sqrt{x^2+y^2}+x}{2}} + sign(y)\sqrt{\frac{\sqrt{x^2+y^2}-x}{2}}i$ where $sign$ is the sign function.
- T^H computes the element-wise conjugate of the tensor T [3]

[3] https://pytorch.org/docs/stable/generated/torch.conj.html

- $T \circ K$ denotes the element-wise product between tensor T and tensor K. T and K must have the same dimension
- $\sum T$ denotes the sum of all elements in tensor T
- $|T|$ returns the absolute value of all elements in T. With the complex number $z = x + yi$, we have $|z| = \sqrt{x^2 + y^2}$
- Combining all above operators, we define our norm of tensor T by the formula: $\|T\| = \left|\sum(\sqrt{T \circ T^H})\right|$

Finally, we construct the FFT loss:

$$L_{FFT} = \frac{\|F_{diff}\|}{\max(\|F(G(I^{LR}))\|, \|F(I^{HR})\|)} \tag{4.6}$$

In equation (4.6), the numerator is our modification from equation (4.5) to obtain real value. However, during our experiment, we found that only use the numerator leads the FFT loss to dominate the whole generator loss. Therefore, we add the denominator to reduce the magnitude of the FFT loss. In our experiment, our frequency penalty loss even works better than an alternative way name spectral loss in recent CVPR [4] paper [27]. First, about the super-resolution metric, our work achieves a better perceptual score. Secondly, using the same experiment in the spectral paper [27], our approach reaches a superior result.

4.1.2.7 Summary

In this section, we discover three potential weaknesses of ESRGAN, including: VGG-based perceptual loss, relativistic GAN and frequency artifacts. We also construct a new loss for both the generator and the discriminator. For the generator loss, the formula is:

$$L_G^{FFT} = L_{percep}^{LPIPS} + \lambda_G \times L_G^{RaGP} + \eta \times L_1 + \lambda_f \times L_{FFT} \tag{4.7}$$

And the corresponding discriminator loss:

$$L_D^{RaGP} = L_D^{RaGAN} + \lambda_g(\left\|\nabla_{\hat{I}} D_{RaGP}(\hat{I})\right\|_2 - 1)^2 \tag{4.8}$$

Our proposed loss introduces two new hyper-parameters: a gradient penalty coefficient λ_g and a frequency penalty coefficient λ_f. Both hyper-parameters weigh the influence of the regularization loss. For the details of other notations, please consider equations (4.3), (4.2), (4.4), (2.4) and (4.6). In the next chapter, we will prove the effectiveness of our approach.

4.2 Improve Diversity

Apart from perceptual quality enhancement, our project aims to diversify the baseline SR model by learning an additional multi-modal generator conditioned on the SR output of the pre-trained baseline, ESRGAN [120]. By doing so, the model is able to

[4] Conference on Computer Vision and Pattern Recognition

learn one-to-many mapping instead of deterministic reconstruction. However, without proper treatment, the model always tend to ignore latent input making the output less diverse. Therefore, to emphasize diversity, we explicitly use an diverse-sensitive loss (4.12). Also, since we are interested in a no-reference setting, we exclude any full-reference loss but instead use a ranking loss (4.10) based on an image quality assessment model.

4.2.1 Image-ranking loss

Unlike to prior works [120, 93, 104], we do not aim to reconstruct the high-resolution image from the input but we want the model to generate a variety of images, which makes our framework unsupersived by nature. Thus, our framework disregards the use of any full-supervised losses such as L_1 loss and perceptual-driven losses. Instead, we use no-reference score NIQE [1] as a mean to guide the generator to output aesthetic images. However, because this score is non-differentiable, we have to train another neural network to rank the quality of the output based on the score, as proposed in [142]. Particularly, given two SR outputs y_1, y_2, the Ranker R outputs have to satisfy:

$$\begin{cases} R(y_1) < R(y_2) \text{ if NIQE}(y_1) < \text{NIQE}(y_2) \\ R(y_1) > R(y_2) \text{ if NIQE}(y_1) > \text{NIQE}(y_2) \end{cases} \tag{4.9}$$

Then, from the Ranker, we can define an image-ranking loss as:

$$L^{SR}_{rank} = \mathbb{E}_{I^{LR}\sim p_i}\mathbb{E}_{z\sim p_z}\, \sigma(R(G(I^{LR}, z))) \tag{4.10}$$

where $R(.)$ is the ranking score of an image and σ is the sigmoid function to normalize the score from 0 to 1. A lower ranking score indicates better perceptual quality. In our work, we directly use the pre-trained Ranker from [142] as we empirically find it sufficient for our purpose.

4.2.2 Image hallucination

Aside from ranking loss, we use adversarial loss with random latent input z for image hallucination and add an diverse-sensitive loss to further encourage the diversity of the output:

- Relativistic least-square adversarial loss [57] that tries to encourage hallucinated outputs to be natural. Here the adversarial (4.11) loss is defined as

$$\begin{aligned} L^{SR}_{adv} = \mathbb{E}_{x_f\sim p_f, c\sim p_c}\left[(D(x_f, c) - \mathbb{E}_{x_r\sim p_r, c\sim p_c} D(x_r, c) - 1)^2\right] \\ + \mathbb{E}_{x_r\sim p_r, c\sim p_c}\left[\left(D(x_r, c) - \mathbb{E}_{x_f\sim p_f, c\sim p_c} D(x_f, c) + 1\right)^2\right] \end{aligned} \tag{4.11}$$

- Diverse loss maximizes the distance between outputs of two different latent codes

$$L^{SR}_{diverse} = \mathbb{E}_{I^{LR}\sim p_i}\mathbb{E}_{z_1, z_2\sim p_z} \max\left(0, m - ||G(I^{LR}, z_1) - G(I^{LR}, z_2)||\right) \tag{4.12}$$

Here, x_r, x_f are real, fake image, c is a conditioning described at the end of section 4.2.5 and m is a small margin to penalize small diversity only.

4.2.3 Low-resolution consistency

As pointed out in [76], GAN-based methods for super-resolution creates inconsistency with low-resolution inputs because of their tendency to hallucinate unnecessary details. To partially mitigate this issue, we also constraint the model's output close with the low-resolution image when being downsampled:

$$L_{lc}^{SR} = \mathbb{E}_{I^{LR}\sim p_i}\mathbb{E}_{z\sim p_z}\left\|I^{LR} - d_{\downarrow}(G(I^{LR}, z))\right\|_1 \tag{4.13}$$

where $d_{\downarrow}(\cdot)$ is the downscale operator used to obtain the low-resolution ground truth. Moreover, low consistency loss is a must-have in order for the model not to generate irrelevant outputs since we do not use supervised signal for the model during training.

4.2.4 Overall objective

Finally, the final loss for the generator that we use in this work comprises of three sub-losses L_{rank}^{SR}, L_{halluc}^{SR} and L_{lc}^{SR}:

$$L_{halluc}^{SR} = \alpha_{adv} \times L_{adv}^{SR} + \alpha_{diverse} \times L_{diverse}^{SR} \tag{4.14}$$

$$L_{G}^{SR} = L_{halluc}^{SR} + \alpha_{rank} \times L_{rank}^{SR} + \alpha_{lc} \times L_{lc}^{SR} \tag{4.15}$$

Whereas, the loss for the discriminator (4.16) is following:

$$\begin{aligned} L_{D}^{SR} = \mathbb{E}_{x_r\sim p_r, c\sim p_c}\left[\left(D(x_r, c) - \mathbb{E}_{x_f\sim p_f, c\sim p_c}D(x_f, c) - 1\right)^2\right] \\ + \mathbb{E}_{x_f\sim p_f, c\sim p_c}\left[\left(D(x_f, c) - \mathbb{E}_{x_r\sim p_r, c\sim p_c}D(x_r, c) + 1\right)^2\right] \end{aligned} \tag{4.16}$$

4.2.5 Architecture

As directly generating a distribution of fine-grained images from low-resolution input is notoriously hard, our model try to learn 1-to-many mapping from a random latent code conditioned on an SR image produced by a pre-trained SR model. Therefore, our framework consists of two generators, one for generating initial SR output and another to diversify that SR image. Furthermore, there is a discriminators that classify real/fake images.

In this work, while any pre-trained generator can be used, we only focus on the baseline ESRGAN [120]. Hence, the our model's global architecture is reported in the Figure 4.2. The architecture's first part is the pre-trained GAN which takes a LR input and output an SR image. Then, it is followed by a learnable high-pass filter to keep the low-frequency features. After that, the resulting features are diversified by our additional U-Net generator and finally combined with the high-frequency features from LR input through a Consistency Enforcing Module (CEM) similar to Bahat et al [9].

To generate a diverse range of outputs, our diversify module combines the input random latent vector with the output features at each scale, Figure 4.2, through a

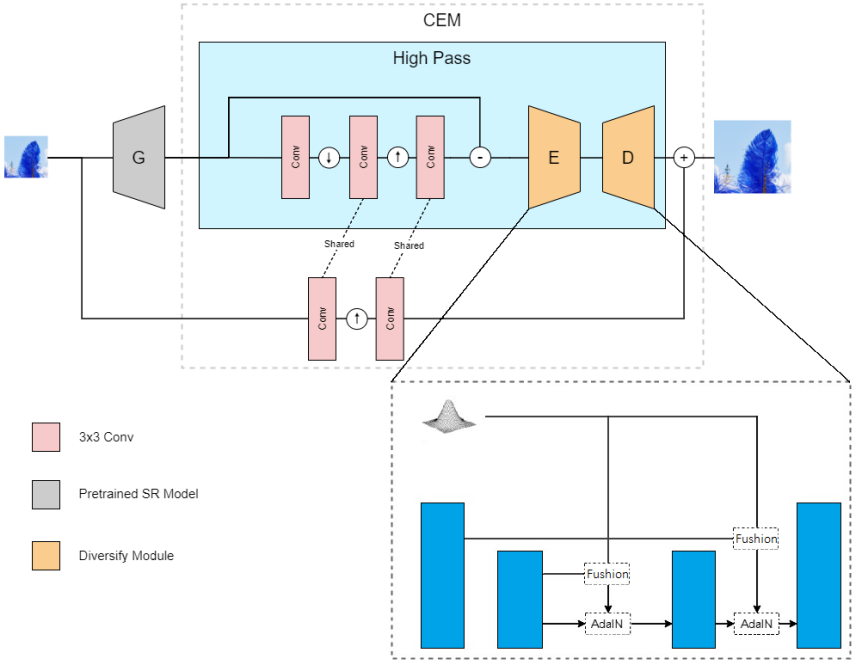


Figure 4.2: Generator architecture for diversity. Here ↓ is a 4x downsampling operator, whereas ↑ is a 4x upsampling operator

fushion layer to learn a better conditioning. Here, we simply choose fully-connected network as the fushion mechanism, however other methods can be used as well. Then the network use that conditioning to condition the features via AdaIN [48] which can be computed as following:

$$\text{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) \tag{4.17}$$

Where x, y, $\sigma(\cdot)$ and $\mu(\cdot)$ are the input tensor, the conditional vector, the mean and variance operator, respectively. For specific parameters used in the model, section 5.3.1 describes our architecture more in detail.

As shown in Figure 4.3, the discriminator used in our framework is built upon [91] where the architecture is inspired by PatchGAN [51]. Particularly, being conditioned on the low-resolution image, the discriminator consists of not only one but two identical branches of PatchGAN [51] described below. Finally, the output features from two branches are fused together and fed to the final fully-convolutional layer to output the real/fake score.

Here, we choose to condition the discriminator on the low-resolution input by using a conditional discriminator from [91] so that the generator can aware more about low-resolution image. Therefore, the discriminator receives two inputs, real/fake images and low-resolution condition:

$$\begin{cases} D(I^{HR}, I^{LR}) \text{ for real image} \\ D(G(I^{LR}, z), I^{LR}) \text{ for fake image} \end{cases} \tag{4.18}$$

4.2.6 Image restoration

In this section, we demonstrate the versatility of our super-resolution model by applying it for other image restoration tasks. Here, we use the **same** model trained for

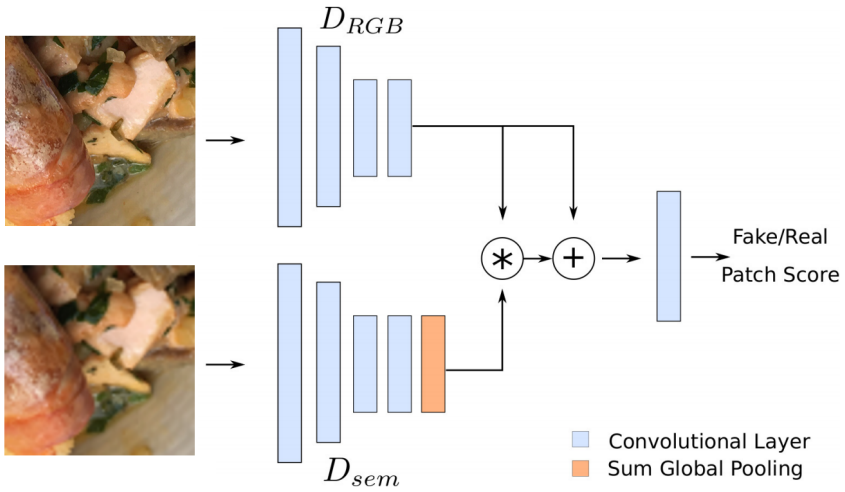


Figure 4.3: SESAME discriminator [91] architecture for diversity.

super-resolution and fine-tune it for the downstream tasks for approximately 4000 iterations. Particularly, we are interested in the performance of our model when restoring an degraded image by noise or compression artifacts. Let $\tilde{y}$ be a corrupted image and its low-resolution counterpart $x = d_{\downarrow}(\tilde{y})$ whose noise and other high-frequency degradations may be largely eliminate by down-sampling. Then, we can obtain a cleaner images y by simply feeding x to our model $G(x, z)$

As our model is trained to realize realistic and high-quality images, we can expect the output of our model to be cleaner. However, reconstructing the image with our model randomly and naively can be problematic because important image information can be lost during the down-sampling process and hard to be recovered. Therefore, instead of using a random latent code z, we opt to find z^* that minimize the image quality loss (4.10) and feed it together with the LR image to our model to output an clean and aesthetic image $G(I^{LR}, z^*)$.

Chapter 5

Experiments

During our research, we carry out a large number of experiments. Section 5.1 contains our earlier experiments in order to find out the limitations of ESRGAN. Also, we exploit some training tips to improve the training process of the baseline model. Two successive sections consist of many comparison tests to measure the effectiveness of our method.

5.1 Initial experiments

5.1.1 Training details

5.1.1.1 Dataset

For training and testing data, we mainly use the DIV2K dataset [3], which is a high-quality (2K resolution) dataset for image restoration tasks. Beyond the dataset of DIV2K that contains 800 images, we also use other datasets with rich and diverse textures for the training. To this end, we further use the Flickr2K dataset [117] consisting of 2650 2K high-resolution images collected on the Flickr website, and the OutdoorSceneTraining (OST) [121] dataset to enrich our training set. We train our models in RGB channels and augment the training dataset with random horizontal flips and 90-degree rotations.

5.1.1.2 Architecture and other information

As described in subsection 2.6.4, our base model, ESRGAN, deploys a variant architecture of residual network named SRResNet, which is an extremely deep and powerful model. In this section, we describe some important parameters and hyper-parameters used in the below experiments (Table 5.1)

As for computational resources, all the experiments are conducted on a machine with NVIDIA's 16GB Tesla V100 GPU and 28GB RAM CPU.

We split the dataset 80-20 for training and evaluation, respectively. To train the model, we follow the training procedure in [120], which uses 200 epochs in total and the batch size is 16. At the beginning, for evaluation, we use two metrics, namely PSNR and SSIM.

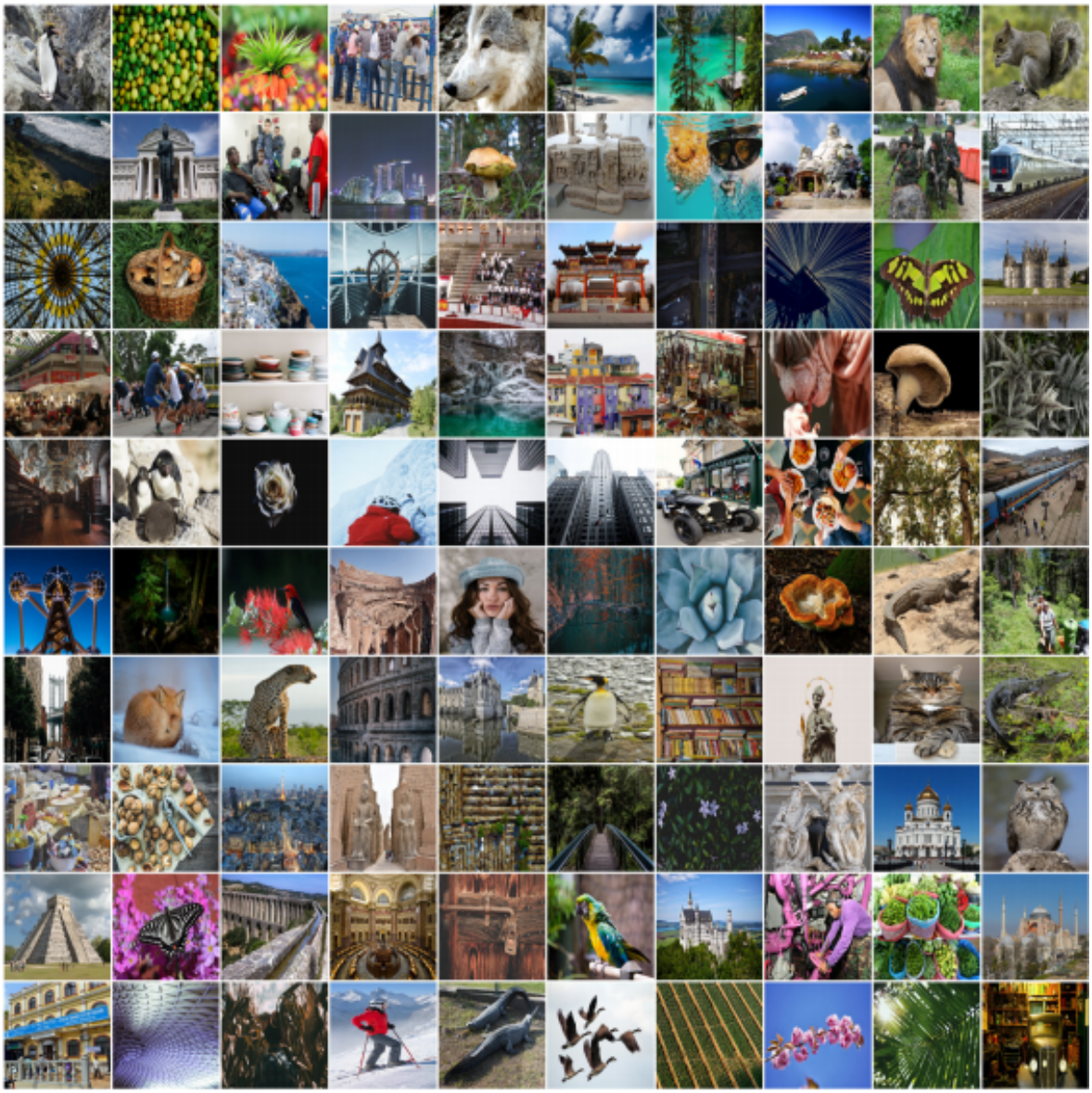


Figure 5.1: Some sample images of the dataset

5.1.1.3 Source code

As mentioned in section 2.6.6, we opt for ESRGAN as our target paper. However, because there is no official implementation for ESRGAN, we have to look for the most

Number of blocks in total (Depth)	115
Number of residual blocks	23
Number of filters	64
Number of learnable parameters	38,549,000
Input image size	32×32
Output image size	128×128
Optimizer	Adam

Table 5.1: Architecture and additional information

suitable implementation of **ESRGAN** that is available for us.[5]

After some initial training, the learning curves shows that the training process is unstable. Therefore, we further need to modify the code using some techniques (see section 5.1.2).

5.1.2 Improving the training process

5.1.2.1 Pre-training the model

In ESRGAN [120], the authors claim that although training the model with adversarial loss produces more perceptually pleasing images, pre-training the model with L_1 loss can lead to better and more stable performance. To verify if such claim is true, we conduct an experiment by training the model with and without pre-training. Specifically, this experiment is carried out in two different phases as follows:

- We firstly train the model within **65000** iterations in total and sample an example image once per **16250** iterations, Figure 5.2. As for the first half iterations, we use L_1 **loss only**; whereas, in the later half, we combine L_1 loss, perceptual and adversarial loss.
- Subsequently, we train another model with the same number of iterations without pre-training with L_1 loss but with **all three loss** at the **beginning**.

Finally, based on Figure 5.2, we can conclude that pre-training the model is essential for producing visually pleasing images and stable training.

5.1.2.2 Learning rate of the discriminator

During our initial experiment, we observed that the discriminator learns too fast and its loss is high compared to that of generator indicating that the generator's learning is slow and unstable. Therefore, we also conduct an experiment of training ESRGAN with lower discriminator's learning rates to study the impact of its learning rate on the stability of the training.

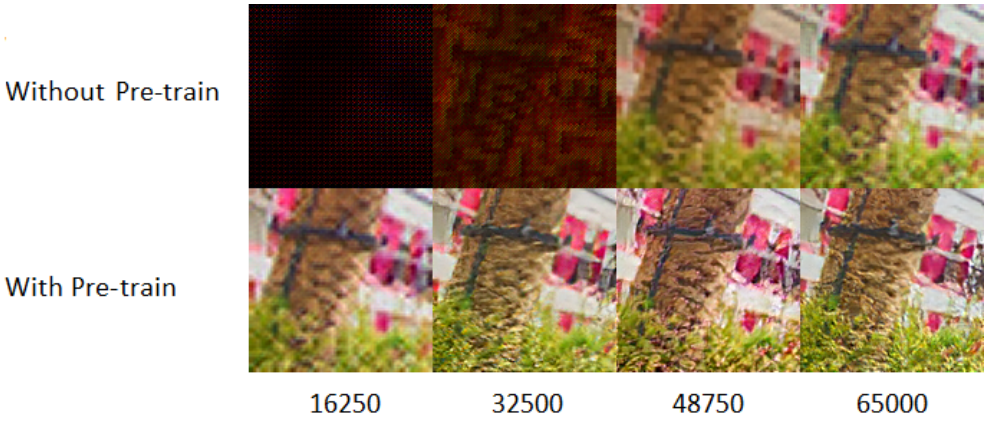


Figure 5.2: Comparison between two different training pipelines

[5] https://github.com/eriklindernoren/PyTorch-GAN

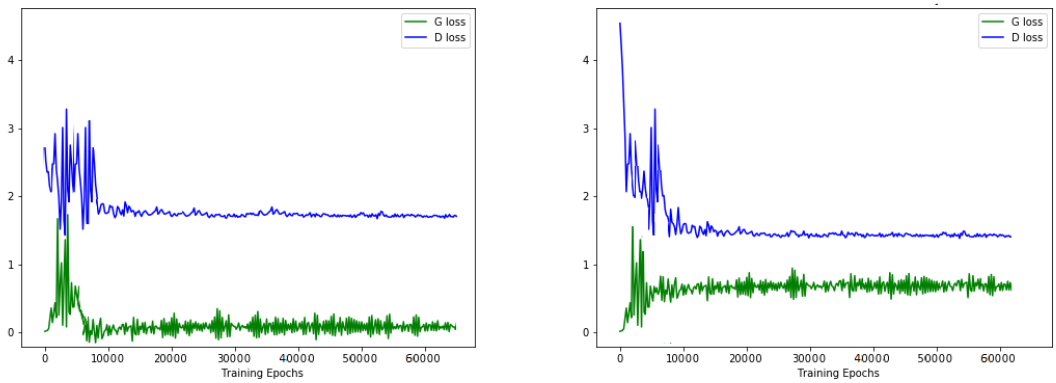


Figure 5.3: **Left**: Discriminator's learning rate is $2 * 10^{-4}$. **Right**: Discriminator's learning rate is $5 * 10^{-5}$.

In this experiment, we also only train the model within 65000 iterations with two different discriminator's learning rate: the default one used in ESRGAN [120] and a lower learning rate that we empirically choose for this experiment.

Figure 5.3 points out that with the default learning rate of ESRGAN, the generator loss is more unstable and learns more slowly. Whereas, with the lower learning rate, the learning is more stable. Therefore, we need to carefully choose the hyper-parameters for ESRGAN.

5.1.3 Initial qualitative and quantitative results

In this section, we simply report on qualitative of ESRGAN compared to ground truth and quantitative results of our final trained model compared to the current state-of-the-art model for super-resolution (SRFlow [76]).

Figure 5.4: Final qualitative results for the validation dataset

Metric	PSNR↑	SSIM↑
Ground Truth	∞	1
ESRGAN	26.22	0.75
SRFlow	27.09	0.76

Table 5.2: Final quantitative results for the validation dataset

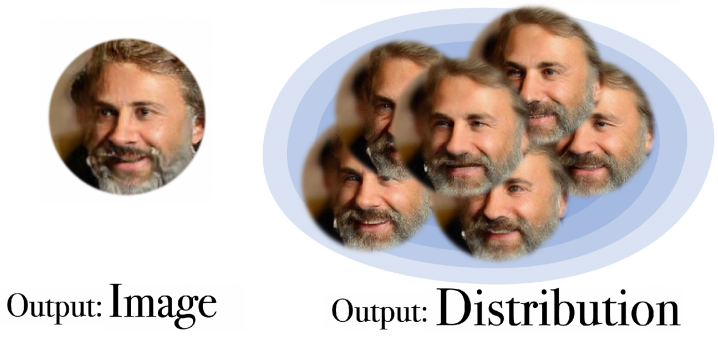


Figure 5.5: 1-to-1 vs 1-to-many [76]

5.1.4 Analysis

5.1.4.1 Curse of uniqueness

After experimenting and surveying the current literature, we observe that ESRGAN, as well as many other existing works, suffer from what we coin as the "curse of uniqueness". Specifically, they consider image super-resolution as a 1-to-1 mapping function and try to train the model to produce an **unique** high-resolution image for one low-resolution image. However, this is not true, given the ill-posed nature of super-resolution, because many high-resolution images can be downscaled into a single low-resolution image, which makes super-resolution 1-to-many rather than 1-to-1. Therefore, we plan to introduce a diversity-aware mechanism in our proposed model (Figure 5.5).

5.1.4.2 Low-resolution inconsistency

Throughout conducting some experiments, we observe that the generated high-resolution images is not consistent with the low-resolution, i.e when downscaling the generated images produces low-resolution images that is inconsistent with ground truth low-resolution images. Therefore, in this section, we report on both quantitative and qualitative results of low-resolution inconsistency

Figure 5.6 demonstrates the procedure we use for this experiment. In detail, we initially downsample the high-resolution ground truth by using bicubic downsampling to obtain a low-resolution image, which is super-resolved to produce a high-resolution image. Then, it is downscaled again and compared with the original low-resolution image to produce an error heatmap.

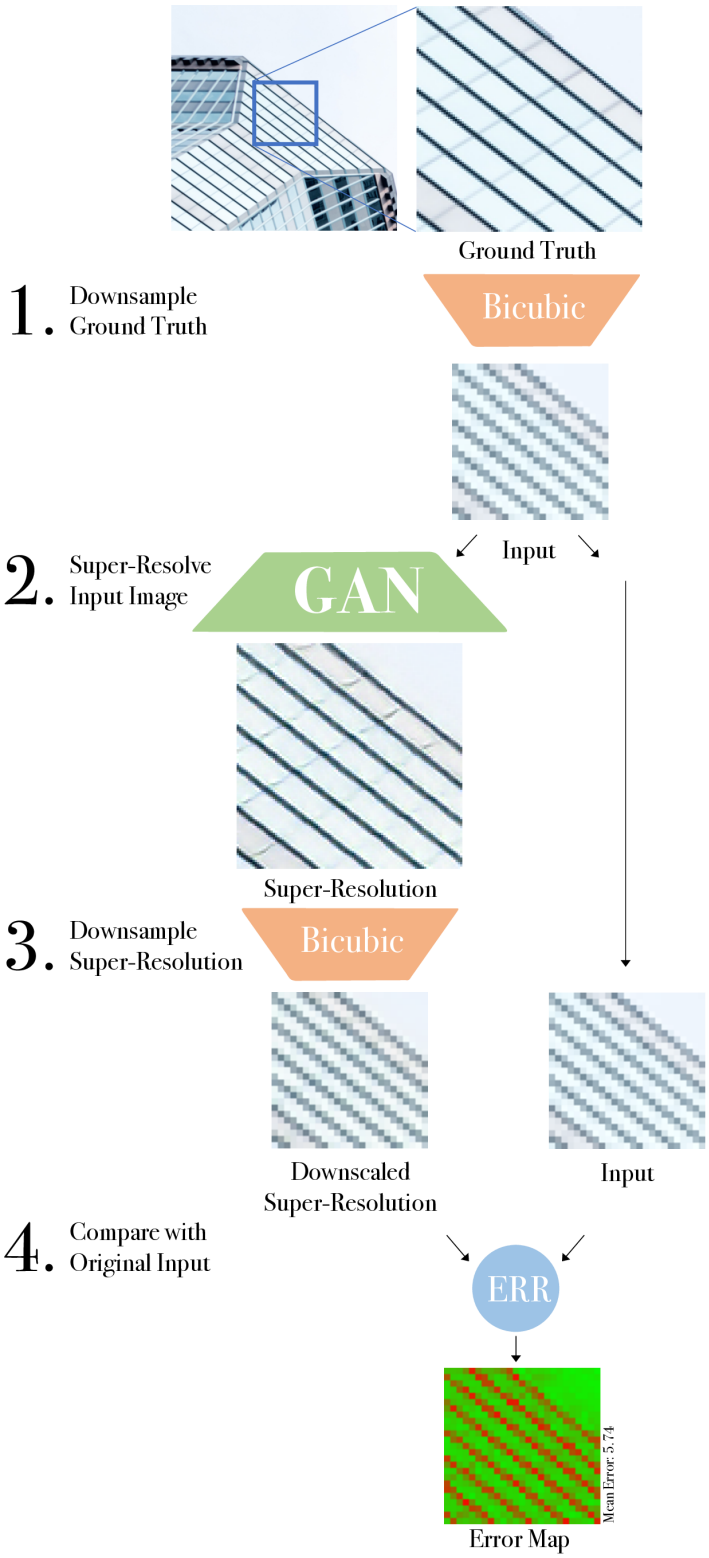


Figure 5.6: The experiment pipeline [149]

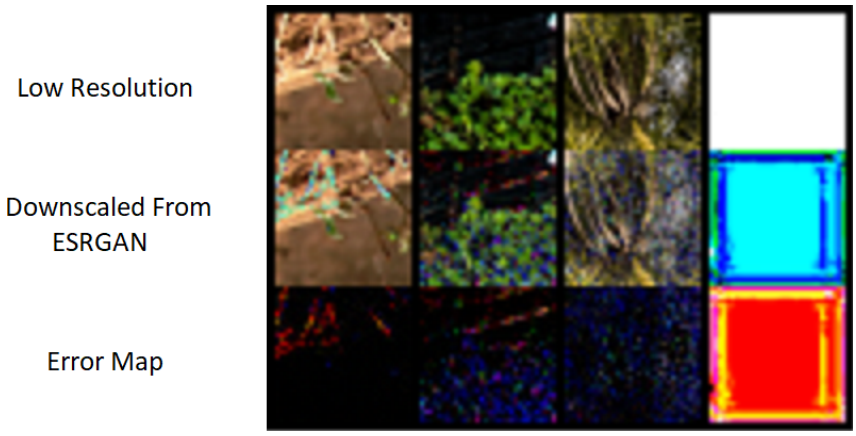


Figure 5.7: Qualitative results on the downscaled images of the model output for the validation dataset

Metric	LR-PSNR↑
Ground Truth	∞
ESRGAN	39.03
SRFlow	49.96

Table 5.3: Quantitative results on the downscaled images of the model output for the validation dataset

Based on Figure 5.7 and Table 5.3, we quantitatively and qualitatively show that ESRGAN for super-resolution creates unpleasing artifacts in the generated images which in turn leads to inconsistency with low-resolution images. Therefore, in our thesis, this is a critical issue that we plan to address.

5.2 Improving the image quality

5.2.1 Training details

5.2.1.1 Datasets

In general, using a large dataset increase the model's performance but requires a large amount of training time. Although we use a smaller dataset than the original paper, it took us more than 1 day for training with the baseline architecture. As a result, we have decided to break the main dataset down into three overlapped datasets. The vast majority of experiments are conducted on the smallest/default dataset. As it took us only approximately 4-5 hours for training, using this dataset allow us to try several loss functions and architecture. After our approach works well on various settings on the default dataset, we use the medium dataset to ensure our model can preserve its performance on a larger training set. Finally, we train our model on the large dataset to produce the visual image and compare the perceptual score with the pre-trained model. We briefly describe the content of each training set:

- **Small training set:** Including 800 images from DIV2K [3] training set.
- **Medium training set:** Including a total of 1600 images. We observe that both DIV2K [3] and Flickr2K [117] datasets mainly consist of landscape or human

pictures, which means that they lack some specific textures such as building or grass. Therefore, instead of combining 800 images from DIV2K with 800 images from Flickr2k, we mix 800 images from DIV2K with only 300 images from Flickr2K and shuffle them with 500 textures pictures for building and animal from OutdoorSceneTraining (OST) [121] dataset.

- **Large training set:** Including approximately 6000 images. The training set contains all images from DIV2K [3] and Flickr2K [117] datasets and more than 2000 other images from OST [121] datasets. However, as some images have the size smaller than the cropped size 128 × 128, we must remove it to avoid training errors.

About the validation set, the DIV2K validation set [3] is still our choice. In most experiments, we deal with a cropped validation set (1024 × 1024) rather than the original ones, as the official code[6] for FID metrics requires all observation to share the same dimensions. About the testing set, as we want to ensure the generalization of our model, we use four different datasets for testing, including Set5 [10], Set14 [136], BSD100 [81] and Urban100 [47]. These benchmark datasets are widely used in other SOTA models such as ESRGAN [120] or SRFlow [76]. However, we faced some difficulties in the testing phase. First, none of the above datasets provide low-resolution images. Secondly, some images such as an image 3096 from BSD100 has its width and height are not divisible by 4 (Ex: 481 × 321), which leads the generated image and the ground truth image to have different sizes. Thus, we randomly crop a picture to a pre-

Type of dataset	**Name**	**Number of images**	**Crop size**
Training set	Small training set	800	128 × 128
	Medium training set	1600	
	Large training set	6000	
Validation set	DIV2K - val	100	1024 × 1024 /None
Test set	Set5	5	128 × 128
	Set14	14	192 × 192
	BSD100	100	256 × 256
	Urban100	100	512 × 512

Table 5.4: Summary information about datasets use for image quality experiments

Configuration	**Description**
A	Baseline ESRGAN
B	Replace VGG-based loss by LPIPS-based loss in configuration A
C	Add gradient penalty to configuration B
D	Add frequency regularization loss to configuration C
X-medium	Config X train with medium training set
X-large	Config X train with large training set

Table 5.5: Some common configuration in our image quality experiments

[6] https://github.com/mseitzer/pytorch-fid

defined size and downscale to an LR image utilizing a built-in function in MATLAB. Table 5.4 summarize the relevant information about our datasets. It is clear that our experiments contain not only several datasets but also various sizes.

5.2.1.2 Configurations

During our experiments, there are some configurations we mention many times. For short, we denotes some notations as in the Table 5.5. Note that this is not all settings in our experiments, just some most common settings.

5.2.2 Evaluation metrics

In section 3.2.4, we had already analyzed the advantages and disadvantages of some metrics in the super-resolution task. Based on this analysis, we divided our metrics into four groups (sort by our priority from high to low):

- **Highest priority:** NIQE [1] and FID [43]
- **High priority:** LPIPS [141]
- **Medium priority:** PSNR and SSIM
- **Low priority:** BRISQUE [87]

In practice, it is hard to obtain a method that outperforms another method in all metrics. Therefore, the priority is necessary to choose a proper approach. In our thesis, we also need to measure the spectral distribution reconstruction accuracy. To this end, some previous research [27, 71] compute the reduced frequency representation in equation (3.21). However, they only plot mean and variance after azimuthal integration over the power spectrum of real and generated images. In our opinion, directly compare on the figure is hard and unreliable, especially when we deal with the complex curve. Therefore, we proposed our novel metric to quantify frequency domain difference, named frequency spectrum discrepancy (FSD), which is defined by the formula:

$$FSD(I^{fake}, I^{real}) = \sum_{r=1}^{\min(M,N)} \left| AI(I^{fake}, r) - AI(I^{real}, r) \right| \tag{5.1}$$

where the image size is $M \times N$, I^{fake} and I^{real} denote the generated and real images, respectively. $AI(I, r)$ expands the equation (3.21) by applying this equation with image I and radius r.

5.2.3 Quantitative results

We not only compare our final approaches with the baseline model but also investigate how each component contributes to this result. Therefore, we arrange our experiments with a similar structure as in section 4.1. Please notice in this section, without further annotation, all experiments are measured on validation set: **DIV2K-val**

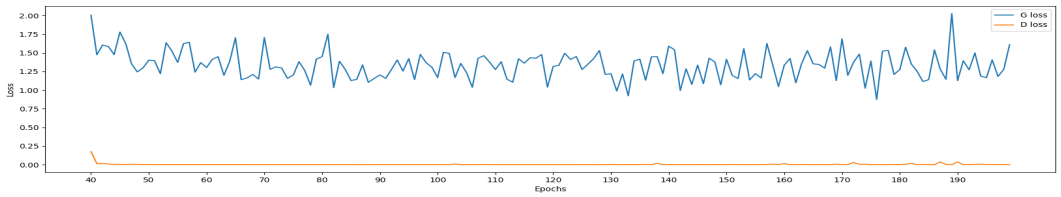
5.2.3.1 Perceptual loss

Table 5.6 demonstrates both the accuracy and perceptual score of the baseline model with three different options for perceptual loss. The top row is exactly the original work with a smaller training set (configuration A) [120], while the middle is our proposal (configuration B). In addition, we also evaluate an alternative way using DISTS [26] loss. Note that each type of perceptual loss has a different data pre-processing technique. For example, we transform data to have a mean and a standard deviation of ImageNet [25] in configuration A. About the case of LPIPS and DISTS, we scale a variable to have values in range $[-1, 1]$ and $[0, 1]$, respectively. Another important thing is our notations because we use them for later tables. Each metrics is located in the cells which is sorted from left to high based on their priority (left location has higher priority). Furthermore, an upward arrow means higher score is better, while a downward arrow shows the opposite.

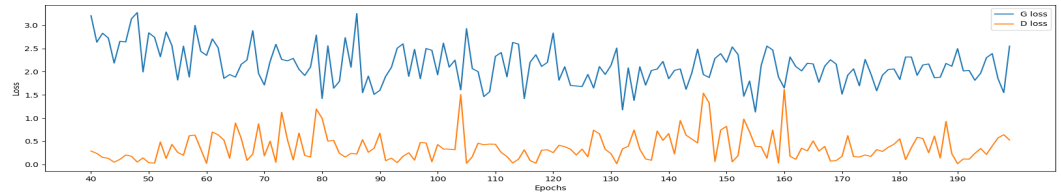
Clearly, two recent image quality assessment models achieve a better score compared

Configuration	**Metric score**					
Baseline model with	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
VGG loss (config A)	6.16	28.87	0.2590	24.00	0.63	66.17
LPIPS loss (config B)	4.01	**20.81**	**0.1488**	**24.74**	0.70	**27.58**
DISTS loss	**2.66**	22.11	0.1576	24.39	**0.71**	33.57

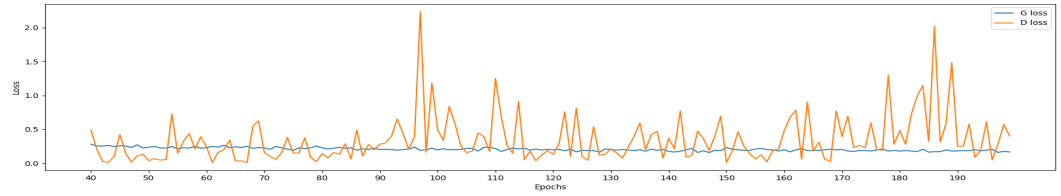
Table 5.6: The qualitative results of three different perceptual loss. **Notations:** Metrics in left cell has higher priority. ↑: higher is better, ↓: lower is better. A **bold** value for each column is the best score.



(a) Configuration A: VGG loss



(b) Configuration B: LPIPS loss



(c) DISTS loss

Figure 5.8: The learning curves of three different perceptual loss. **Blue:** Generator loss. **Orange:** Discriminator loss.

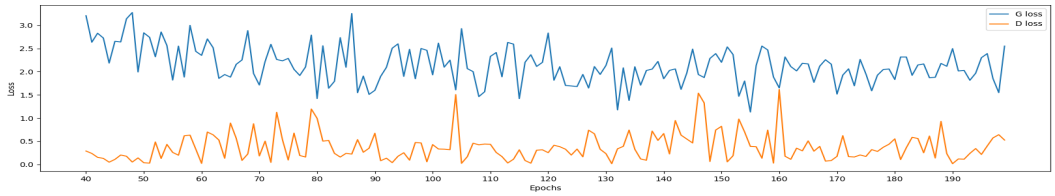
to the traditional choice. It is not a surprising result because some previous research obtains the same conclusion [60, 141, 26]. Moreover, in general, LPIPS outweighs DISTS in all metrics with the exception of NIQE. Due to its enormous architecture, DISTS also has a training time of more than eight times that for LPIPS. About learning curve, Figure 5.8 prove that each approach has its own weaknesses. All learning curves are drawn from fortieth epochs, as we pre-training generator with L_1 loss before this point. In our experiment, the discriminator in ESRGAN nearly cannot learn after a short time. Using the LPIPS loss can sidestep this problem, but it makes the loss of both the generator and the discriminator more fluctuate. Meanwhile, DPIPS loss allows the discriminator to learn but prevents the enhancement of the generator. From the reasons above, although LPIPS loss still has some limitations, it is safe to say that this is a highly satisfactory choice for perceptual loss.

5.2.3.2 Adversarial loss

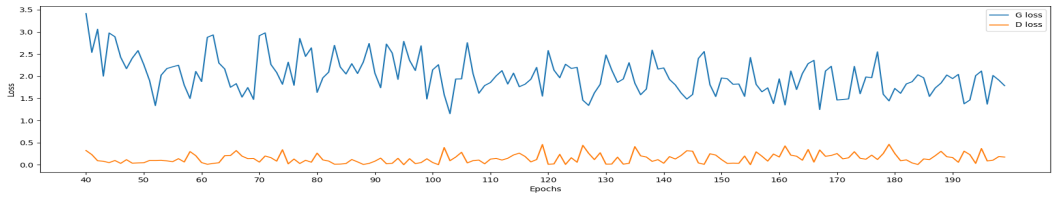
Beside the original work in ESRGAN (equation (2.4)), in section 4.1.2.5, we consider some variants of RaGAN [56] and Wasserstein GAN [7] including: RaHinge (equation (2.6)), RaLS (equation (2.5)), RcGAN (equation (2.7)), RcHinge (equation (2.9)), RcLS (equation (2.8)), WGANGP (equation (2.12)) and RaGP (equation (4.4)). Table 5.7 show these corresponding results. It can be seen that none of the versions of relativistic GAN deriving by Martineau [56] can cause a distinct difference. Since our aim is perceptual quality, we consider NIQE, FID and LPIPS are the primary metrics, in this term, Wasserstein GAN (the last two rows) is shown more efficient. Evaluation by our metric priority reveals that both the standard setting of RaGAN and WGANGP achieve inferior results compare to our combination loss named RaGP (please only consider the first row and the last two rows for clearer). Further, Figure 5.9 illustrates an additional benefit of our approach. By adding the gradient penalty, the generator and the discriminator can now learn more stable. We note that either WGANGP or RAGP requires an additional hyper-parameter: a gradient penalty λ_g. The numbers in Table 5.7 are their finest parameters in our experiments. Presented how to find their

Configuration	**Metric score**					
Config B with	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
RaGAN (original setting)	4.01	20.81	0.1488	<u>24.74</u>	**0.70**	<u>27.58</u>
RaHinge	**3.08**	23.37	0.1537	23.40	0.68	36.65
RaLS	3.71	21.92	0.1519	**24.78**	**0.70**	**22.78**
RcGAN	3.59	22.37	0.1565	24.20	<u>0.69</u>	28.69
RcHinge	3.40	20.61	0.1486	24.43	**0.70**	30.96
RcLS	3.49	22.51	0.1524	<u>24.74</u>	**0.70**	29.52
WGANGP with $\lambda_g = 10^{-3}$	3.72	**18.80**	<u>0.1442</u>	24.51	0.68	40.96
RaGP with $\lambda_g = 10^{-4}$ (config C)	<u>3.27</u>	<u>19.80</u>	**0.1427**	24.21	0.68	45.21

Table 5.7: The qualitative results of different adversarial loss. **Bold** values indicate the best performance and <u>underline</u> values represent the second



Configuration B: RaGAN



Configuration C: RaGP

Figure 5.9: The learning curves of two different adversarial loss. **Blue:** Generator loss. **Orange:** Discriminator loss.

value and further analysis are covered in appendix A.

5.2.3.3 Frequency regularization

To remove frequency artifact in GAN-based models while preserving high perceptual quality, we propose a novel and effective regularization loss - equation (4.6). Table 5.8 presents the comparison between with and without FFT loss in three different settings. Specifically, we consider the LPIPS-based perceptual loss combine with various types of adversarial loss, including RaGAN (configuration B), WGANGP and RaGP (configuration C). Note that the mix of LPIPS-based perceptual loss, gradient penalty and frequency regularization loss corresponds with configuration D in Table 5.5. It is obvious that applying the FFT loss enhance most metric scores regardless of which adversarial loss is used. These results also prove the high applicability of FFT loss because this loss can easily integrate into any loss function.

We also perform comprehensive experiments between our method with an alternative approach in the recent CVPR 2020 paper [27]. Similar to our strategy, Durall et al. propose a novel spectral regularization term based on the azimuthal integration - equation (3.21). They believe this term can compensate spectral distortions, so reduce the frequency gap between the generated images and real images. We borrow the Figure 5.10 from [27] to illustrate their procedure and quote their spectral loss formula:

Configuration				**Metric score**					
Config B with									
Adv loss	FFT	λ_g	λ_f	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
RaGAN	✗	✗	✗	4.01	20.81	0.1488	24.74	0.70	**27.58**
	✓		0.5	**3.05**	**18.02**	**0.1445**	**25.48**	**0.73**	37.01
WGANGP	✗	10^{-3}	✗	3.72	**18.80**	0.1442	24.51	0.68	40.96
	✓		0.3	**3.35**	19.19	**0.1391**	**25.58**	**0.72**	**26.46**
RaGP	✗	10^{-4}	✗	3.27	19.80	0.1427	24.21	0.68	45.21
	✓		0.3	**3.20**	**17.54**	**0.1410**	**25.67**	**0.74**	**37.99**

Table 5.8: The qualitative results between with and without FFT loss in three different settings. **Notations:** λ_g: a gradient penalty coefficient, λ_f: a frequency penalty coefficient. ✓ denotes using the FFT loss while ✗ indicates the opposite.

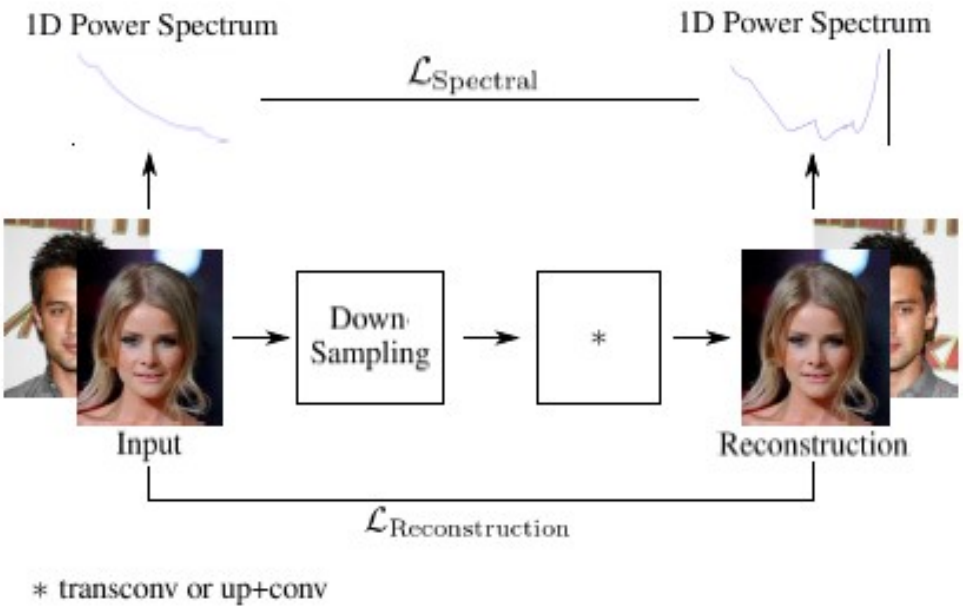


Figure 5.10: The schematic overview of spectral loss [27].

$$L_{spectral} = -\frac{1}{(M/2-1)}\Big(\sum_{r=0}^{M/2-1} AI(I^{real}, r).\log(AI(I^{fake}, r)) + (1 - AI(I^{real}, r)).(1 - \log(AI(I^{fake}, r)))\Big) \quad (5.2)$$

where M is the size of square images. Another notations are similar to equation (3.21). We note that we make a little change on the above equation to match with our notation, but do not affect the whole formula.

Equation (5.2) assume the square images, which means $M = N$ in our notation. As we random crop an image to a fixed size, we can directly apply equation (5.2) without any modification. By experiments, we conclude our FFT loss outperforms the spectral loss in two aspects:

- Our loss is the proper choice for the super-resolution task because our method produces better metric scores compared to the spectral loss.
- FFT loss is capable of reproducing more exactly spectral distribution.

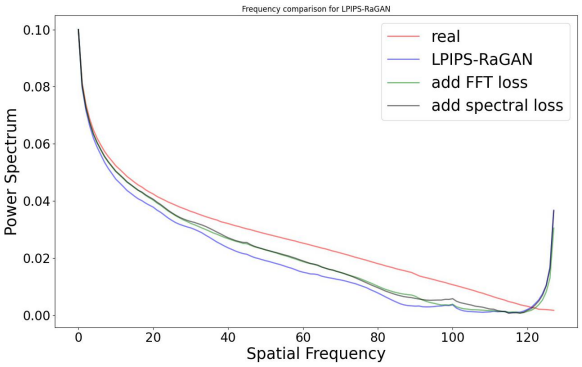
To validate our statements, we measure both the accuracy and perceptual score between FFT loss and spectral loss. Table 5.9 presents the metric scores when adding FFT/spectral loss to the combination of LPIPS-based loss with RaGP adversarial loss. In our notation, this setting equals configuration C. Other experiments in the case of RaGAN and WGANGP adversarial loss can be found in appendix B. For fair comparison, all experiments are conducted on two different hyper-parameters: 0.3 and 0.5. Compared to the spectral loss, FFT loss achieves significantly better scores in most reference metrics. Interestingly, even the model without using the frequency penalty term outperforms the spectral loss model in FID and LPIPS. These results reveal that the performance of spectral loss on the super-resolution task is very limited.

In order to measure the performance on spectral distribution reconstruction, we rely on the power spectrum test using in much previous research [27, 71]. The test is

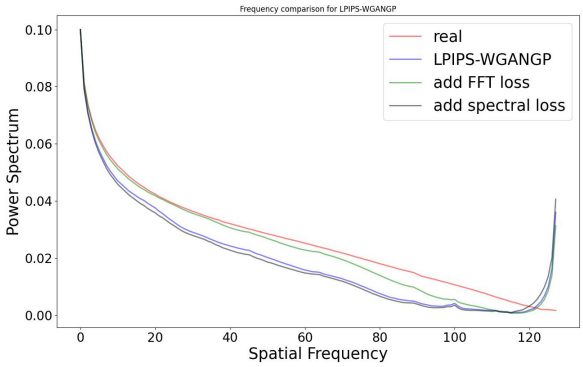
taken by observing the 1D power spectrum generating from real and fake images. To get the reduced spectral representation, we simply apply equation 3.21 with a large range of values of r. Of course, we hope the model can create pictures with the lack of frequency artifacts. In other words, the superior model should produce a small frequency gap. For a detailed comparison, please see Figure 5.11. Similar to previous experiments, we carry out our tests on multiple settings. Please focus on the green curve and the black curve for better comparison. In the case of RaGAN, adding the FFT or

Configuration **Config C with $\lambda_g = 10^{-4}$**		**Metric score**					
FFT/spectral loss	λ_f	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
✗	✗	3.27	19.80	0.1427	24.21	0.68	45.21
FFT loss	0.3	**3.20**	**17.54**	**0.1410**	**25.67**	**0.74**	**37.99**
spectral loss		3.44	20.38	0.1457	24.47	0.69	39.32
FFT loss	0.5	3.75	**17.73**	**0.1358**	**25.60**	**0.74**	**28.32**
spectral loss		**3.17**	20.12	0.1491	24.19	0.67	34.10

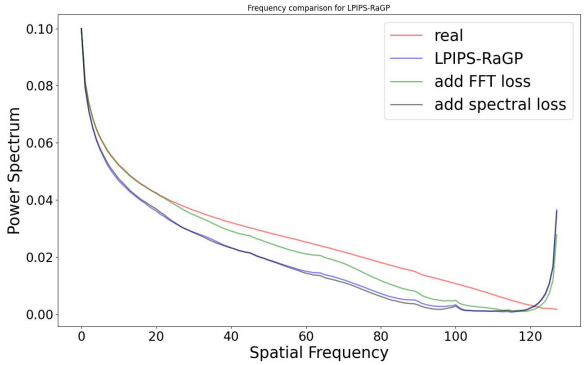
Table 5.9: The qualitative results between FFT loss and spectral loss using configuration C. **Notations:** λ_g: a gradient penalty coefficient, λ_f: a frequency penalty coefficient.



(a) Configuration B: RaGAN loss



(b) Configuration B but use WGANGP loss

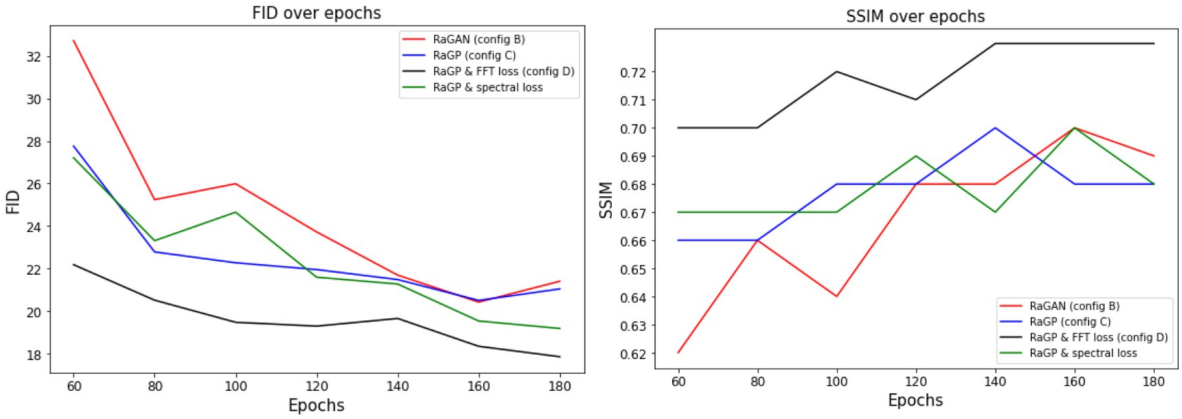


(c) Configuration C: RaGP loss

Figure 5.11: The effects of frequency regularization term on the frequency spectrum (azimuthal integral) of the output images. **Red curve:** real images, **Purple curve:** the standard model with LPIP-based perceptual loss and (a) RaGAN adversarial loss, (b) WGANGP adversarial loss, (c) RaGP adversarial loss, **Green curve:** add FFT loss to the standard model, **Black curve:** add spectral loss to the standard model.

Modification	Configuration		
	Config B with		
	RaGAN	WGANGP	RaGP
Original	0.00758	0.00736	0.00801
add FFT loss	0.00535	**0.00291**	**0.00374**
add spectral loss	**0.00521**	0.00840	0.00826

Table 5.10: The frequency spectrum discrepancy (FSD) of all experiments in Figure 5.11. Note that lower FSD is better.



(a) FID↓ score: lower is better

(b) SSIM↑ score: higher is better

Figure 5.12: The example of perceptual (FID) and accuracy (SSIM) score over training time. In both cases, our FFT loss variant (black line) is able to quickly achieve superior results. Recall that, before the fortieth epochs, only the generator is pre-training

spectral loss get the reduced power spectrum closer to the desired values. However, it is hard to conclude which loss is better because two curves are likely to represent the same function. When we consider an alternative option using WGANGP or RaGP, we observe the FFT loss obtains the best result among three models. Further, it is likely that applying spectral loss in those cases lead the model tend to produce more frequency artifacts. Now, we consider the quantitative value to check our observation. Table 5.10 quantify the result in Figure 5.11 by utilizing equation (5.1). As can be seen from this table, spectral loss slightly better than FFT loss when combining with RaGAN. In contrast, when we change the adversarial loss to WGANGP or RaGP, the performance of spectral regularization drops significantly. Again, only our FFT loss can work well on all settings.

The majority of our previous experiments consider only the final achievements, but how the impact of our modifications on the GAN training. To this end, after using the first forty epochs for pre-training, we observe the representation of perceptual and accuracy metrics once every twenty epochs. Figure 5.12 illustrates the FID and SSIM along the training epochs. We can observe that after 20 epochs, our FFT loss variant reach the middle point where unregularized GANs or spectral loss variant requires approximately 80-100 epochs to arrive. Furthermore, our proposed term is able to keep the finest scores during the training procedure.

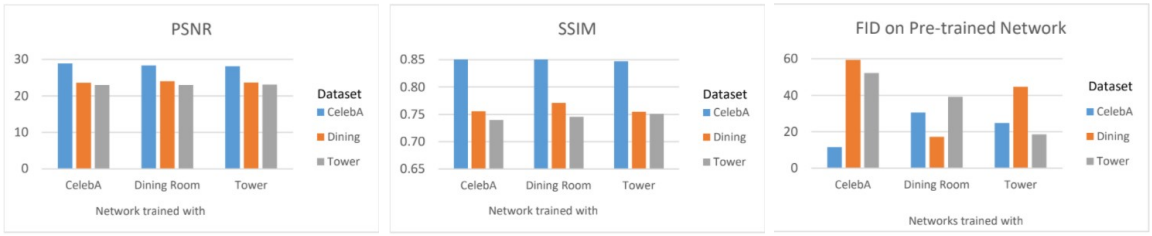
5.2.3.4 Benchmark evaluation

Until now, our ideas have been proved to work well on the validation set; however, it is uncertain that they would be preserved their performance on other datasets. As we mentioned above, we utilize four widely used datasets for testing: Set5 [10], Set14 [136], BSD100 [81], Urban100 [47]. Recall configuration A, B, C and D stand for VGG-RaGAN, LPIPS-RaGAN, LPIPS-RaGP, LPIPS-RaGP with FFT settings, respectively. The information of test sets can be found in Table 5.4 and the results are presented in Table 5.11. Compared to other options, our configuration with FFT loss stably gets better results in terms of highest priority metrics (NIQE and FID) and accuracy-driven metrics (PSNR and SSIM). About the BRISQUE score, the FFT loss variant put on an acceptable performance when it reaches the second-highest position in three over four cases. Unfortunately, although the FFT term satisfactorily works on LPIPS distance, the achievements are varied. However, this incident does not limit the effectiveness of our approach. This is mainly because in all datasets, FFT loss always surpasses other configurations when we consider the whole metric system rather than the single one.

In addition, we want to drive the attention to another phenomenon that occurs in our experiments. When we observe Table 5.11 and Table 5.9, the perceptual values in the former are extremely worse than those numbers in the latter, especially with FID.

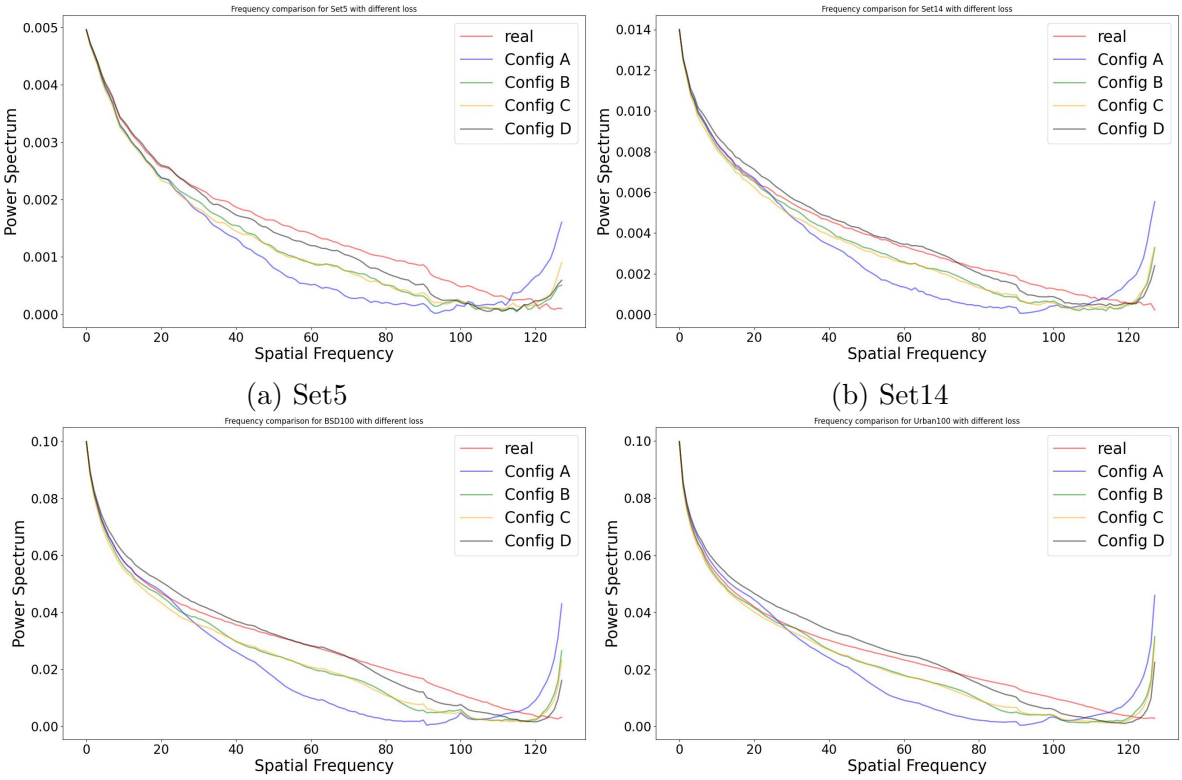
Test set	**Configuration**	**Metric score**					
		NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
Set5	A	**18.87**	168.94	0.1220	26.06	0.70	59.64
	B	**18.87**	192.47	**0.0598**	27.73	0.78	**19.65**
	C	**18.87**	169.55	0.0653	27.88	0.77	30.38
	D	**18.87**	**127.96**	0.0606	**29.65**	**0.83**	41.84
Set14	A	14.80	183.78	0.2254	23.51	0.62	69.00
	B	5.61	188.73	0.1309	24.09	0.66	**24.69**
	C	5.92	184.47	**0.1307**	24.16	0.64	45.15
	D	**5.28**	**169.55**	0.1313	**25.35**	**0.70**	37.17
BSD100	A	10.75	109.49	0.3106	23.02	0.59	60.06
	B	5.73	106.46	0.1841	23.22	0.62	**33.41**
	C	4.96	97.57	**0.1734**	22.76	0.58	52.46
	D	**4.69**	**94.88**	0.1739	**23.91**	**0.65**	41.82
Urban100	A	8.13	65.72	0.2922	20.97	0.58	68.33
	B	5.25	49.89	0.1777	21.28	0.63	**34.72**
	C	**4.77**	48.11	0.1698	21.06	0.62	49.11
	D	4.79	**44.13**	**0.1653**	**22.03**	**0.67**	40.10

Table 5.11: Quantitative evaluation results of different loss for SR on benchmark datasets. **Bold** values highlight the best performance and underline values indicates the second best.



(a) PSNR↑ (b) SSIM↑ (c) FID↓

Figure 5.13: With FID, Takano et al. [114] show that for the best result one should use the same image-type dataset that will be used at inference time. This phenomenon does not occur in case of PSNR and SSIM.



(a) Set5 (b) Set14

(c) BSD100 (d) Urban100

Figure 5.14: The frequency spectrum (azimuthal integral) of the output images of different loss for SR on benchmark datasets. **Configuration:** **A:** VGG-RaGAN, **B:** LPIPS-RaGAN, **C:** LPIPS-RaGP, **D:** LPIPS-RaGP with FFT loss.

Configuration	**Dataset**			
	Set5	Set14	BSD100	Urban100
A	0.00052	0.00114	0.00983	0.00808
B	0.00031	0.00056	0.00553	0.00434
C	0.00034	0.00066	0.00581	0.00457
D	**0.00017**	**0.00031**	**0.00253**	**0.00327**

Table 5.12: The frequency spectrum discrepancy (FSD) of all experiments in Figure 5.14. Note that lower FSD is better.

In order to explain this phenomenon, we recall the evaluation procedure. First, we train the GAN model to try to learn the distribution of the real dataset. We denote this target distribution by $\mathbb{P}_r$. In the testing phase, we use the generator to produce the fake data with the distribution $\mathbb{P}_g$. Then, we compute the Fréchet Inception Distance by applying equation (3.11). Table 5.9 considers an event where the generator receives the inputs from DIV2K training and validation sets. Because those inputs coming from the same source, it is safe to assume that they have a similar distribution. Therefore, the generator can reproduce the data have the distribution close to the ground truth (measured in Jensen-Shannon divergence). On the other hand, in Table 5.11, we enter the low-resolution images form various sources and they do not share any relationship with the training set in term of distribution. Thus, this lead to an increase of the measure distance. Further, each perceptual metric is the deep network train with their own training set, so they heavily rely on the training distribution. Thus, it does not surprising that their score is varied depending on the test set. Takano [114] et al. conduct a series of experiments to observe this phenomenon. Specifically, this matter is easy to spot in FID but does not occur in accuracy-driven metrics, like PSNR and SSIM. Please see a Figure 5.13 for a better illustration. For the above reason, for a fair comparison, we should not consider the absolute value of perceptual metrics from two different conditions. Instead, we should compare only the perceptual score conducting on the same constrain.

Furthermore, we conduct spectral distribution reconstruction experiments again. The illustrations are shown in Figure 5.14 and their numerical scores are presented in Table 5.12. It can be obviously seen that applying FFT loss gains considerable advantage in the frequency domain. Because the primary purpose of FFT loss is to minimize the frequency gap between real and generated images, these results emphasize our useful contribution.

5.2.3.5 Large network

Next, we investigate how our model operates if we train with a larger set of images. We therefore evaluate this aspect by reporting the numerical results in Table 5.13. In general, we observe similar trends with earlier experiments in both two cases.

Training set	**Configuration**	**Metric score**					
		NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
Medium	A	3.27	26.07	0.1632	23.45	0.66	40.96
	B	3.24	20.20	0.1384	24.60	0.70	37.09
	C	3.10	18.13	0.1374	24.62	0.70	40.72
	D	**3.06**	**16.70**	**0.1273**	**25.71**	**0.74**	**32.72**
Large	A	2.90	24.66	0.1862	23.67	0.67	43.89
	B	2.75	17.42	0.1315	24.80	0.71	43.06
	C	2.79	16.95	0.1227	25.36	0.72	**37.33**
	D	**2.65**	**16.01**	**0.1200**	**25.49**	**0.73**	47.20

Table 5.13: The quantitative results of different size of training datasets. **Bold** values highlight the best performance.

5.2.3.6 Final results

We have recognized and fixed some image quality issues in ESRGAN, enhancing the quality further without increasing the computation cost. Of course, our approach cannot totally remove model limitations but can alleviate them to some extent. Table 5.14 represents the quantitative scores between two different versions of loss. The standard version or configuration A deploys equation (4.1) for generator loss and equa-

Test set	**Configuration**	**Metric score**					
		NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
Set5	A-large	**18.87**	**107.53**	0.0687	27.83	0.78	**38.55**
	D-large	**18.87**	126.32	**0.0522**	**29.72**	**0.83**	47.80
	pre-trained	18.87	73.24	0.0495	30.56	0.84	38.55
Set14	A-large	5.30	153.30	0.1541	23.86	0.64	52.13
	D-large	**4.62**	**132.27**	**0.1149**	**25.33**	**0.69**	**48.08**
	pre-trained	5.32	123.76	0.1214	25.62	0.69	46.64
BSD100	A-large	4.79	88.31	0.1945	22.65	0.59	55.29
	D-large	**4.50**	**78.16**	**0.1514**	**23.66**	**0.64**	**48.38**
	pre-trained	4.89	67.55	0.1574	23.61	0.64	46.28
Urban100	A-large	5.23	45.71	0.1746	20.72	0.62	**59.51**
	D-large	**4.80**	**30.94**	**0.1294**	**22.10**	**0.68**	59.90
	pre-trained	5.25	28.22	0.1331	22.34	0.69	56.38

Table 5.14: Quantitative evaluation comparison between the baseline loss and our proposed loss in several datasets. Configuration A and D share the same architecture but different loss: **A**: VGG-RaGAN, **D**: LPIPS-RaGP with FFT loss. We also report the scores of pre-trained ESRGAN model for reference, but not include in our comparison

Configuration	**Metric score**				**Number of training**	
	NIQE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	images	iterations
Bicubic	5.20	0.409	26.70	0.77	None	None
EDSR [72]	4.46	0.270	28.98	0.83	800	> 300K
RRDB [120]	5.08	0.253	29.44	0.84	≈ 14K	≈ 1000K
RankSRGAN [142]	2.45	0.128	26.55	0.75	800	900K
ESRGAN-pretrain [120]	2.61	0.124	26.22	0.75	≈ 14K	≈ 1000K
SRFlow [76]	3.57	0.120	27.09	0.76	3450	400K
D-large (ours)	2.55	0.109	26.57	0.76	≈ 6K	74K

Table 5.15: General image SR results on the 100 validation images of the DIV2K dataset. We reuse the numerical results from SRFlow paper [76] and add the measurement of our case. Note that all batch sizes are 16 and 1K equals 1×10^{-3}

tion (2.4) for discriminator loss. On the other hand, two corresponding equation for configuration D are equation (4.7) and (4.8), respectively. Again, our setting enhances the overall performance, although we get undesired results in some metrics.

Due to lacking conditions for computation resources, we do not target to surpass the pre-trained model. However, to measure the performance, we evaluate our work with recent SOTA models in the general super-resolution task. Specifically, we utilize the previous experiment in SRFlow paper and measure our method follow their pipeline. Note that in this experiment, they use non-crop DIV2K validation set. Table 5.15 includes the detail values and contains some additional information about the number of images and iteration in the training procedure. It is clear that we cannot compete with other authors in terms of model size. However, compare to previous approaches, our results are very comparable. First, please consider the deterministic approach named bicubic and the accuracy-driven models such as EDSR and RRDB. Obviously, our methods outperform these options in the case of human-correlated scores (NIQE and LPIPS). Secondly, among perceptual-driven models, we get the best performance in SSIM and LPIPS values, and the second in other scores. Notably, our proposed loss exceeds the pre-trained version of ESRGAN in all kind of metrics.

Despite promising results in the validation set, our model cannot totally outperform the pre-trained model in the test set. To clarify this point, please consider the D-large and pre-trained row in Table 5.14. We can see that in Set5 set, the official version of ESRGAN heavily outweighs our setting. In other cases, although we still obtain the best performance in some key metrics such as NIQE and LPIPS, the pre-trained model is still better in some aspects like FID and PSNR. In theory, more data increases the neural network performance and generalization. Therefore, it is natural that the pre-trained model can properly perform in various data distribution. However, since we gain impressive achievements in specific cases, our results are encouraging and can be expanded in the future.

5.2.4 Qualitative results

In Figures 5.15 and 5.16, we choose four different examples to illustrate how each type of loss affects the visual quality. Each figure contains one example that concentrates on low-frequency components and the other focuses on high-frequency ones. There are two important notes for this experiment. First, the test sets do not ensure the high-resolution condition as in the training and validation set. Secondly, we pre-process the original image to ensure their size are divisible by 4 and downsample using MATLAB built-in function. In all observations, the bicubic method can preserve the structure but result in blurry images. Next, please concentrate on the first case in baboon image and baby image. Clearly, the baseline model suffers from undesired artifacts. Replacing VGG-based loss to LPIPS-based loss reduces these unnatural textures. However, our FFT loss further improves the visual quality with consistent details. A similar trend can be seen in the second patch of image baboon. Because this picture includes dense high-frequency components (baboon's whiskers), using only VGG loss or LPIPS loss contains unpleasing noise in their output. Meanwhile, combining with FFT loss produces sharper and more natural details; therefore, obtain a closer image with the referenced image. The final observation consider the case when the original image has a high-resolution such as img_091 from Urban100. In this situation, FFT setting preserve finer geometric structures compared with perceptual-driven options.

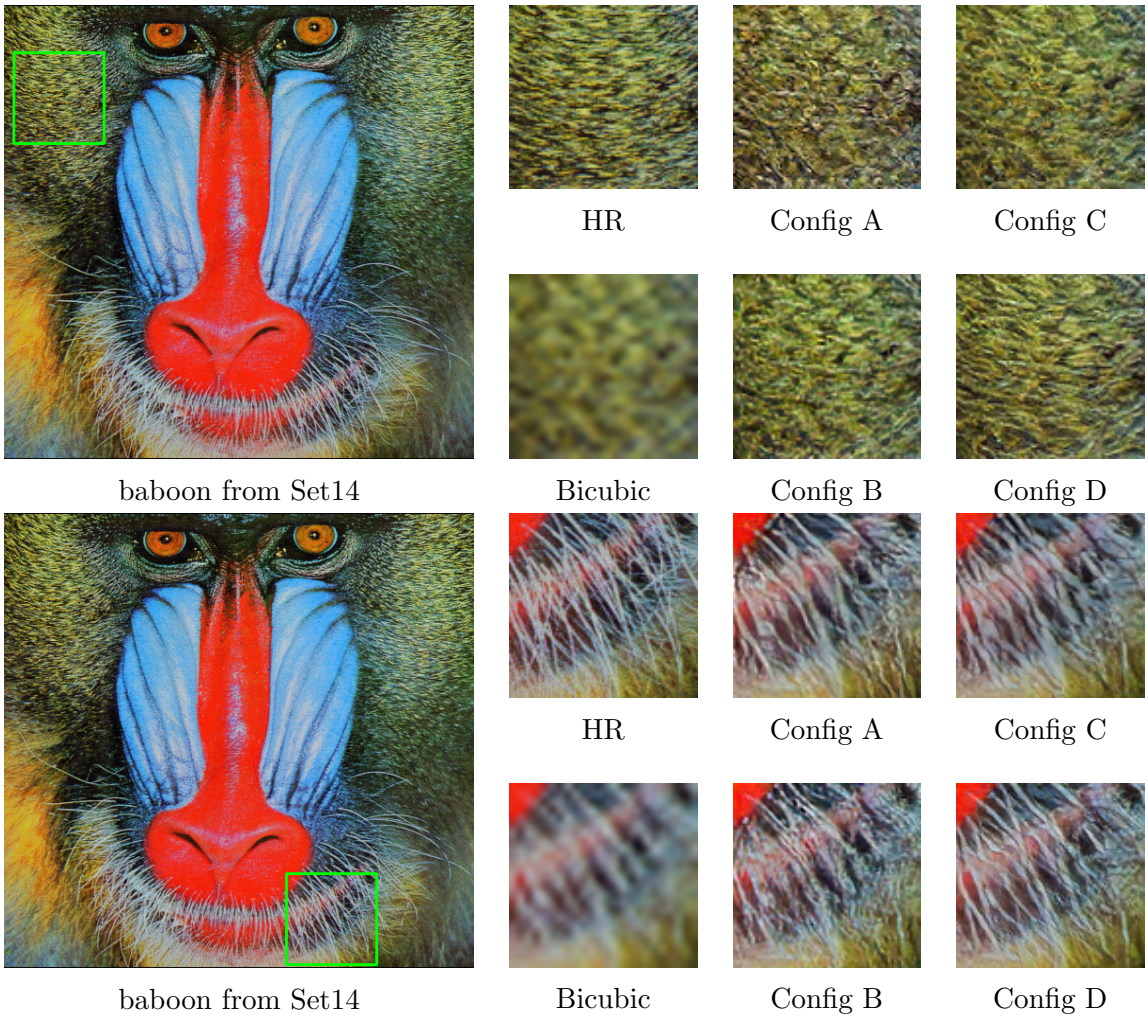


Figure 5.15: Visual comparison between different loss in image **baboon**. **A:** VGG-RaGAN, **B:** LPIPS-RaGAN, **C:** LPIPS-RaGP, **D:** LPIPS-RaGP with FFT loss. Other examples in the next page. Please zoom for better comparison.

Similar to quantitative comparison, we also conduct a visual comparison to the pre-trained ESRGAN model in Figure 5.17. For the first image, the pre-trained model can infer sharper details of the baboon's whiskers. However, in the second row, their output image suffers from some unnatural artifacts in paddle and bush textures. In this case, our method can provide a more pleasing picture.

For more comparison, including the side cases where the effect of our improvement is limited, please consider appendix C

5.3 Improving the image diversity

5.3.1 Training details

We use the same training dataset as in the initial experiment (section 5.1) and evaluate our model with the test set BSD100 [81]. However, instead of feeding to our model with 32×32 patches for training like traditional SR models, we use patches with size of 76×76. We use ADAM optimizer [63] to minimize the loss in (4.15)

Figure 5.16: Visual comparison between different loss in images **baby** and **img_091** . **A:** VGG-RaGAN, **B:** LPIPS-RaGAN, **C:** LPIPS-RaGP, **D:** LPIPS-RaGP with FFT loss. Please zoom for better comparison.

with $\alpha_{adv} = 0.005$, $\alpha_{rank} = 0.03$, $\alpha_{diverse} = 0.05$. Additionally, we set the batch size for training at 16 and train for around 80000 steps where the learning rates for both generator and discriminator are 2×10^{-4}. The size of the latent vector we used for training is 256×1.

As for our diversify module, we use a light-weight U-Net model which consists of an encoder and a decoder. Particularly, the encoder contains two blocks of a 3x3 convolution layer following by a $2\times$ downsample layer, whereas the decoder is the mirrored version of the encoder with 3x3 convolution and an $2\times$ upsample layer. The respective number of filters used in all the four convolution layers from left to right are $64, 128, 64$ and 32. Furthermore, the fushion block is actually a 2-layer full-connected network where the input dimension, hidden dimensions at each layer are 512 and 256, respectively.

5.3.2 Quantitative results

Here, we evaluate our model for general SR on the BSD100 test set. We compare our framework to ESRGAN [120], a variant of ESRGAN with noise input [9] and ExplorableSR [9]. Except for ESRGAN, which is a deterministic model, all methods

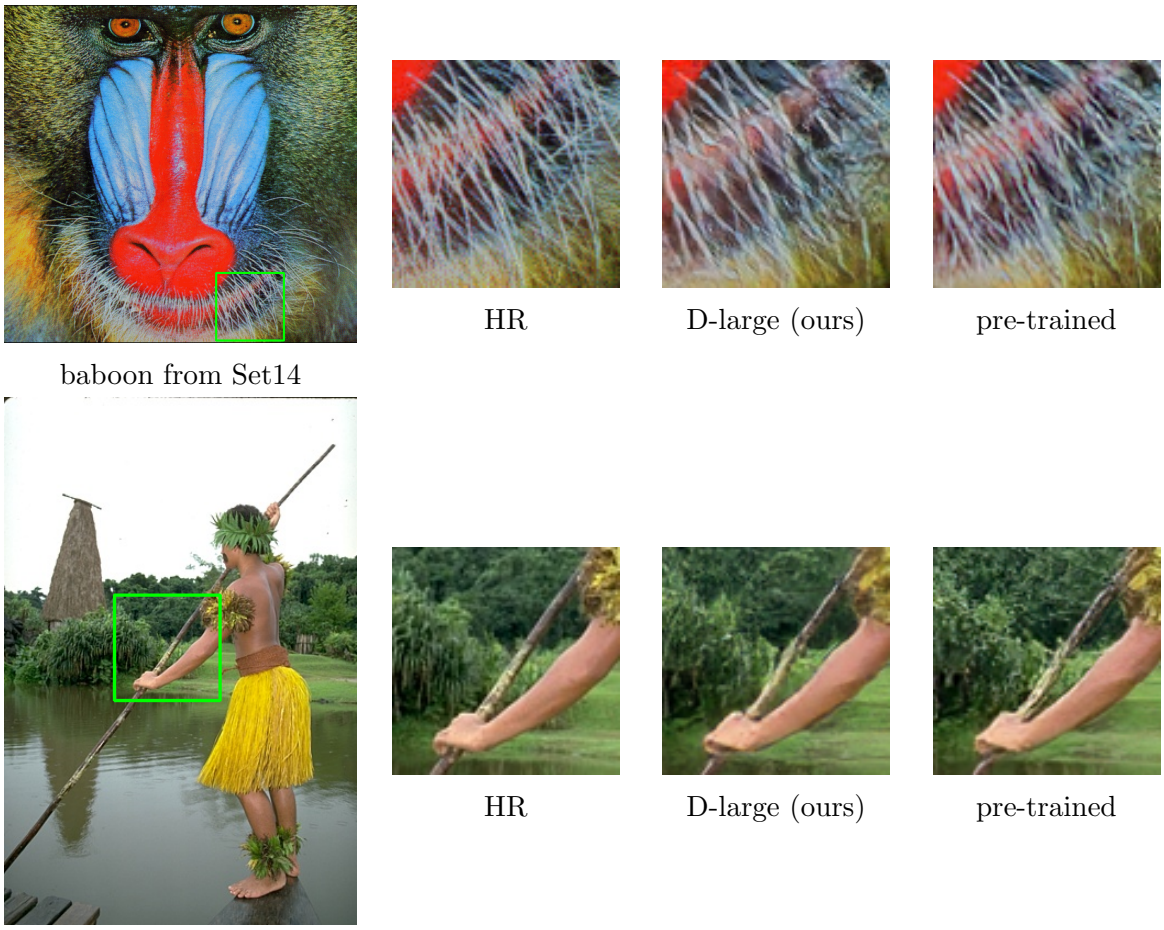


Figure 5.17: Visual comparison between our model and pre-trained ESRGAN model. Please zoom for better comparison.

are capable of generating many SR outputs. In Table 5.16, we directly use the same measurement and the provided quantitative results from [9] then compare with our model on three different aspects:

- **Diversity**: For each low-resolution input, we random 50 latent codes and use our generator to output 50 different SR images. Next, we compute the per-pixel variance.
- **Perceptual quality**: First, we compute the mean image from 50 SR outputs. Then, we use NIQE to score the visual quality.
- **Reconstruction error**: This is a low-priority criteria for this our work but we still include RMSE for fair comparison with [9] and gain some more insight about our model. Similar to above, we also use the mean image for measuring.

As a reconstructive model, ESRGAN clearly outperforms all other approaches in terms of RMSE and takes the second place in perceptual quality but its output has zero variation. Furthermore, equipping ESRGAN with random latent code helps the diveristy at the cost of worse perceptual quality. Interestingly, ExplorableSR [9] produces images that is the most diverse but far less visually pleasing than others. Conclusively,

	Diversity ($\sigma \uparrow$)	Perceptual quality (NIQE$\downarrow$)	Reconstruction error (RMSE$\downarrow$)
ESRGAN	0	3.5	**17.3**
ESRGAN with z	3.6	3.7	17.5
ExplorableSR	**7.2**	3.7	18.2
Ours	6.5	**3.2**	18.8

Table 5.16: Quality and diversity of SR results represented in average score.

this phenomena indicates that the model may learn to output diverse but degraded images.

Unlike above methods, our approach score the highest in terms of NIQE metric while being comparably diverse. Also, the high reconstruction error reported in our model is a sound evidence that our model does not overfit to the high-resolution ground truth.

5.3.3 Qualitative results

We provide visual examples for the experiments on BSD100 test set. An example for the variety of up-scaling factor 4× is shown in Figure 5.18. Our model accomplishes impressive perceptual quality, even though it is not trained with full-reference loss.

Observably, our model can generated diverse texture and structure of rock (first row of Figure 5.18) or animal fur details (second row of Figure 5.18). On the contrary, our model perform little diverse hallucination on blur background (the green square in the second row of Figure 5.18), which suggests that our model does not diversify blur regions but only the highly-detailed areas. However, the low-frequencies detail of faces, scene texts or symbols are not well hallucinated with our model (third row of Figure 5.18). Given that our models is trained on a dataset containing mostly natural sceneries, this phenomena is understandable. Furthermore, additional visualization can be found in appendix D

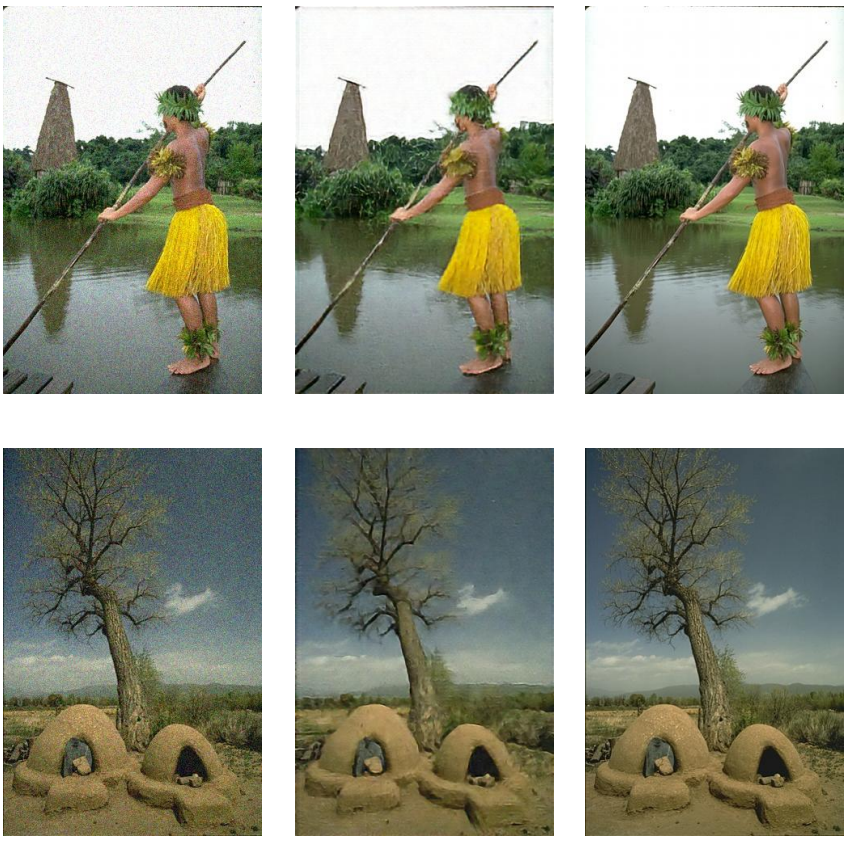
5.3.4 Image restoration

We provide additional qualitative results for image restoration, described in section 4.2.6. In this experiment, we add to the input image a white Gaussian noise with the standard deviation $\sigma = 0.05$, where the pixel scale is $[0, 1]$. Figure 5.19 reports some visuals with respect to the clean ground-truth for the original noisy image, when super-resolving the down-sampled image and use the optimal z^*, as described in section 4.2.6. Despite only being fine-tuned for only a few iterations, our approach provides promising results for image denoising.

5.3.5 Ablation study

Apart from the adversarial and low consistency loss, we additionally train our model without diverse (4.12) and/or image-ranking (4.10) loss, under the same training setting mentioned in section 5.3.1, to observe their impact quantitatively. Table 5.17 shows results on the BSD100 dataset. Without the diverse loss, the model tend to ignore the latent codes and suffers from mode collapse. Furthermore, when the quality rank loss is not used, the model performs much worse on perceptual quality metric.

Figure 5.18: Random SR samples generated by our model using LR images of BSD100 (4×) test set



Noisy Image SR Image Ground Truth

Figure 5.19: Visual result for image denoising.

Diverse Loss	Image-ranking Loss	Diversity ($\sigma \uparrow$)	Perceptual quality (NIQE$\downarrow$)	Reconstruction error (RMSE$\downarrow$)
✗	✗	2.2	3.6	18.4
✗	✓	2.4	**3.3**	**16.8**
✓	✗	**6.6**	4.9	19.9

Table 5.17: Quantitative impact of different combinations of losses.

EDSR

RankSRGAN

Figure 5.20: The effect of diversify module on different pre-trained models. **Top left:** Initial SR output. **Others:** Diversified SR outputs from **3** random z

Qualitatively, we observe the impact of pre-trained 1-to-1 model used to generate the initial SR output from which the final SR image is diversified through our additional module. In this experiment, we opt for two pre-trained models, namely RankSRGAN [142] and EDSR [72], for the pre-trained SR model in Figure 4.2 instead of the baseline model ESRGAN. From the Figure 5.20, it is clear that as long as the initial SR output is richly detailed, our model can diversify them well. Otherwise, our model struggles to hallucinate varied texture when the initial SR output is blur. Moreover, it should be noted that our diversify module can work well to some degree in spite of not being trained with these pre-trained models.

Chapter 6

Conclusion

In our research, we focus on improving GANs for computer vision applications. To achieve this, our approach is to enhance GANs for a specific well-applicable task (e.g super-resolution). Among a broad range of GAN-based methods for super-resolution, we inspected ESRGAN due to our limited resources. Through recent surveys and our initial experiments, we realized that even though ESRGAN has achieved impressive and competitive results in producing realistic high-resolution images, this method still has its own limitation. Thus, we made an effort to enhance the baseline model in two aspects: image quality and image diversity.

About image quality, we identified and fixed several issues in the original work. Following the recent research in image quality assessment, we improved the perceptual loss by using the LPIPS network, which offers stronger learning and creates more consistent details. In addition, we merged two noticeable results name relativistic GAN and Wasserstein GAN to produce a novel loss, called RaGP. The novel combination loss benefits not only quantitative scores but also training stability. Moreover, while previous works only enhance model ability in the spatial domain, we developed a frequency regularization term to guide the generator focus more in the frequency domain. By several experiments, our frequency solution surpasses many other alternative options in the same field.

About image diversity, we found out that most previous works including our baseline ESRGAN [120] is only capable of generating a single reconstruction. In our work, we introduced a framework for diverifying any 1-to-1 super-resolution models, which explicitly accounts for the ill-posed nature of the SR problem and creates a set of perceptually plausible SR images consistent with a given LR image. Particularly, we designed a multi-modal diversify architecture and utilized a differential non-reference loss so that the model does not overfit to the ground truth and learn a diverse set of outputs. Moreover, we evinced the adaptability of our model by exploiting it for other image manipulation tasks such as image denoising.

With just a small amount of data, the model showed promising performance within our scope. However, our achievements are restricted by two major obstacles. First, GAN's architecture use two networks, which makes the modification more challenging. Secondly, ESRGAN utilizes innovative and powerful architecture called residual in residual dense block. Thus, constructing a more effective architecture is an arduous task. Although we tried many different approaches like receptive field or attention mechanism, their improvements are extremely limited. However, in the future, there are several techniques that we expected to enhance our performance:

- Collect and train our model with more data and more iteration.
- Examine some latest SOTA methods such as transformer or diffusion model.

Appendices

A Hyper-parameters and learning curves

A.1 The gradient penalty coefficient

Table 5.7 report a qualitative scores of a broad range of adversarial loss. The experimental results shows that all variants of relativistic GAN get a similar results, which is worse than those values of Wasserstein-related method. However, there are several unclear question need to be answered. First, until now, we have not presented how we find the gradient penalty coefficient and how this hyper-parameter affects training stability. Secondly, it is unsure whether the different versions of relativistic GAN result in the same learning curve. To clarify this point, we carry out several experiments. Tables A.1 and A.2 present the details in the case of WGANGP and RaGP, respectively. Also, we show their corresponding learning curves in Figure A.1 and Figure A.2. Since we only modify the coefficient to balance the contribution of the regularization loss, the results are very similar. An identical trend can be observed when we consider the learning curves of relativistic GAN in Figure A.3. Although moving from relativistic average to relativistic centered makes the log loss more fluctuate, changing the divergence loss does not heavily impact the final curve.

Hyper-parameter	**Metric score**					
λ_g	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
$\mathbf{10^{-3}}$	3.72	**18.80**	**0.1442**	**24.51**	0.68	40.96
10^{-4}	3.48	20.68	0.1540	24.00	0.66	52.56
10^{-5}	**2.93**	21.49	0.1524	24.50	**0.70**	**32.19**

Table A.1: WGANGP hyper-parameter experiment. A **bold** value highlights the best performance

Hyper-parameter	**Metric score**					
λ_g	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
10^{-3}	4.31	21.49	**0.1358**	**24.59**	0.68	**35.21**
$\mathbf{10^{-4}}$	**3.27**	**19.80**	0.1540	24.21	0.68	45.21
10^{-5}	3.64	20.02	0.1566	24.48	**0.70**	42.00

Table A.2: RaGP hyper-parameter experiment. A **bold** value highlights the best performance

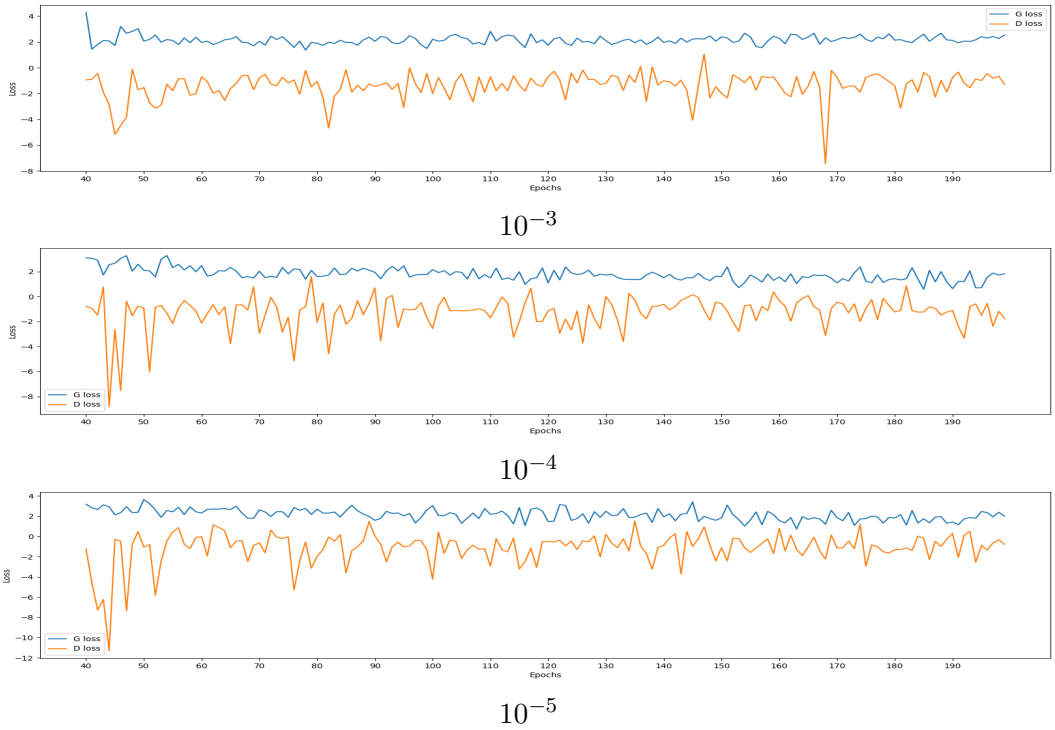


Figure A.1: WGANGP learning curve experiment. **Blue:** Generator loss. **Orange:** Discriminator loss.

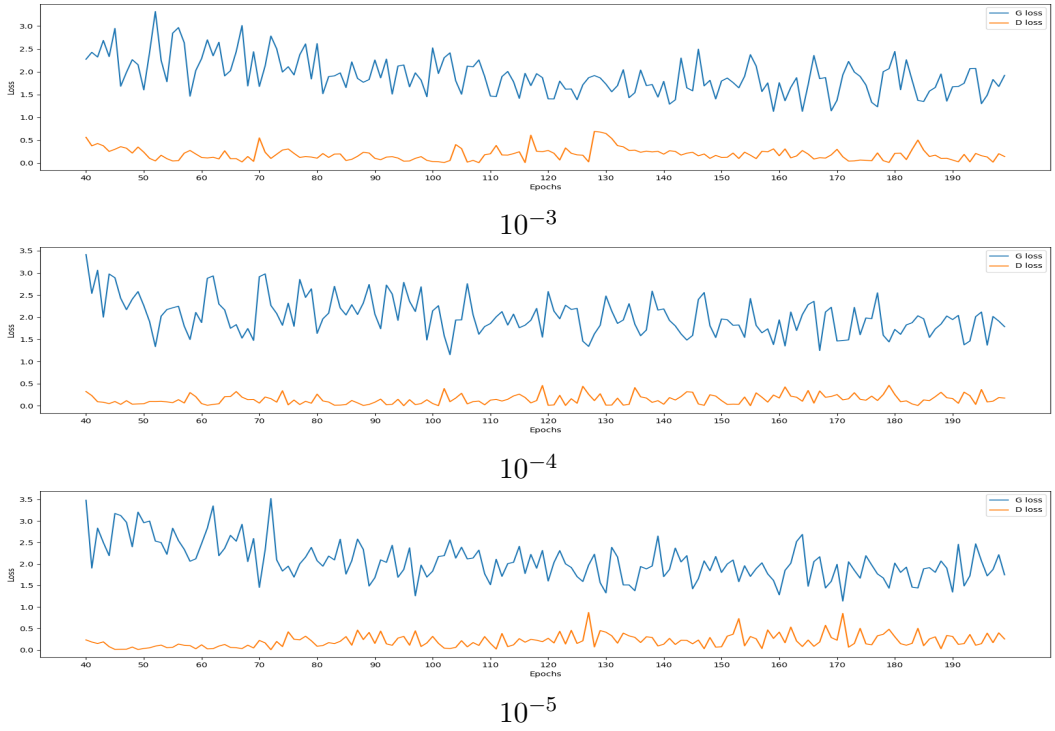
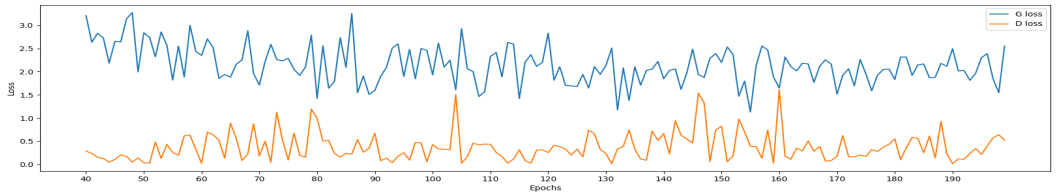


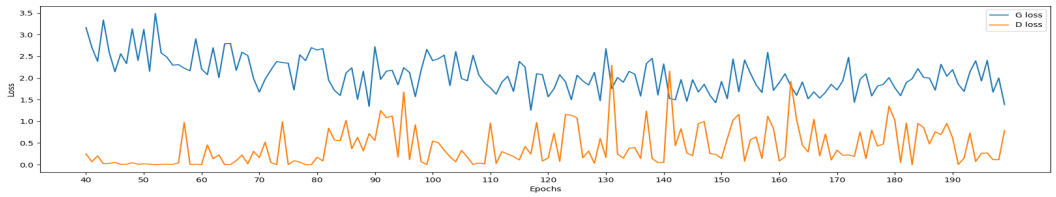
Figure A.2: RaGP learning curve experiment. **Blue:** Generator loss. **Orange:** Discriminator loss.

A.2 The frequency penalty coefficient

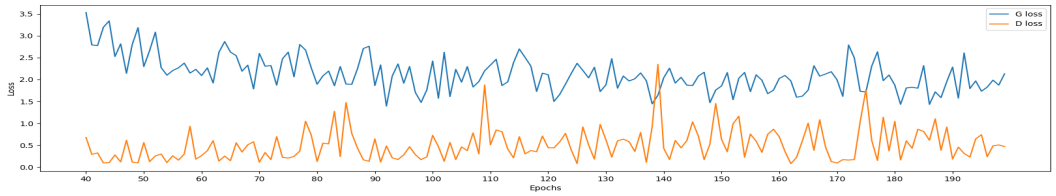
Similar to previous experiments, Table A.3 shows how we find a suitable frequency coefficient. We also note that, compare to related work [27], we choose a smaller value for this hyper-parameter. The main reason is that in equation 4.1, the primary



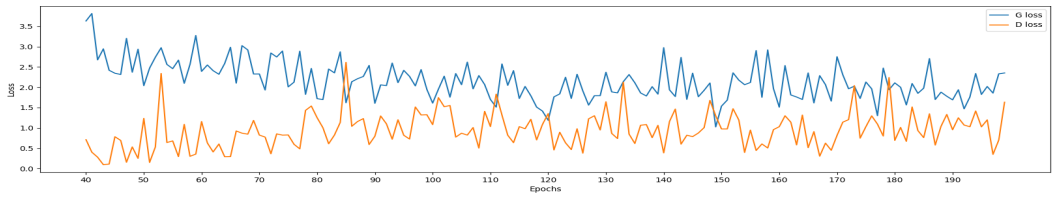
RaGAN



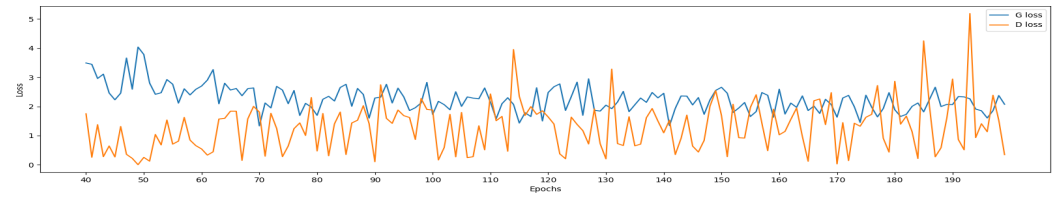
RaHinge



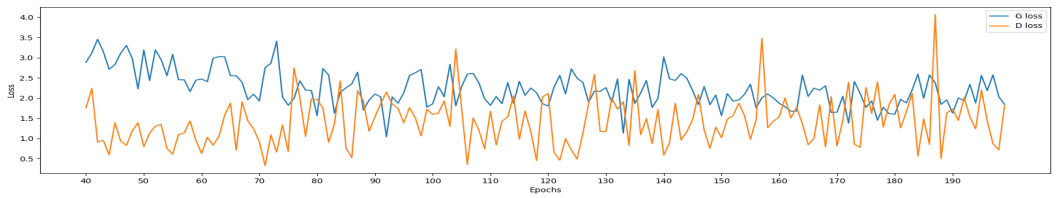
RaLS



RcGAN



RcHinge



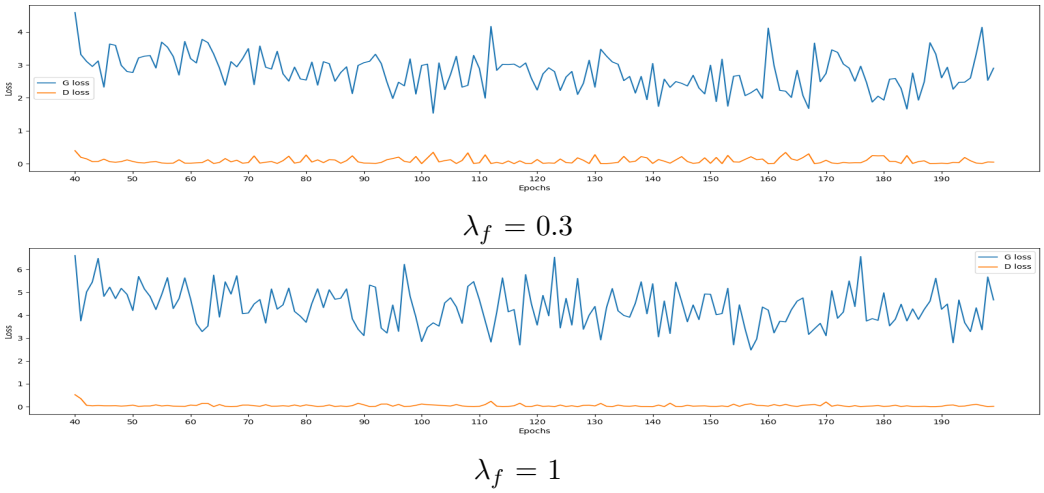
RcLS

Figure A.3: The learning curves of different relativistic adversarial loss for configuration B - LPIPS perceptual loss. **Blue:** Generator loss. **Orange:** Discriminator loss.

component is the perceptual loss rather than the adversarial loss. Therefore, we avoid to make the generator focus to much on frequency domains and neglect other information. Also, the empirical result in Figure A.4 indicates that using a very large value of λ_f can prevent the discriminator from learning.

Configuration Config B with			Metric score					
Adv loss	FFT	λ_f	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
RaGAN	✗	✗	4.01	20.81	0.1488	24.74	0.70	**27.58**
	✓	0.1	3.31	19.33	0.1496	24.93	0.72	29.48
	✓	0.3	**2.93**	20.19	0.1533	24.73	0.70	33.93
	✓	**0.5**	3.05	**18.02**	**0.1445**	**25.48**	**0.73**	37.01
WGANGP	✗	✗	3.72	**18.80**	0.1442	24.51	0.68	40.96
	✓	0.1	**3.42**	18.85	0.1410	24.75	0.70	35.22
	✓	**0.3**	**3.35**	19.19	**0.1391**	**25.58**	**0.72**	**26.46**
	✓	0.5	**3.35**	**17.62**	0.1404	24.97	0.71	36.91
RaGP	✗	✗	3.27	19.80	0.1427	24.21	0.68	45.21
	✓	0.1	3.47	19.00	0.1423	24.98	0.71	32.29
	✓	**0.3**	**3.20**	**17.54**	0.1410	**25.67**	**0.74**	37.99
	✓	0.5	3.75	17.73	**0.1358**	25.60	**0.74**	**28.32**

Table A.3: FFT hyper-parameter experiment. **Bold** values highlight the best performance.



$\lambda_f = 0.3$

$\lambda_f = 1$

Figure A.4: FFT learning curve experiment. **Blue:** Generator loss. **Orange:** Discriminator loss.

B More experiments on frequency regularization loss

B.1 Comparison with spectral loss

Table 5.9 proves the advantage of our frequency regularization term compare to a spectral loss from [27]. While this table includes only the experiment on RaGP settings, to generalize the conclusion, we do tests on two other configurations: RaGAN and WGANGP. The detailed score can be found in Tables A.4 and A.5. Again, our method achieves a superior result.

Configuration WGANGP with $\lambda_g = 10^{-3}$		Metric score					
FFT/spectral loss	λ_f	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
✗	✗	3.72	18.80	0.1442	24.51	0.68	40.96
FFT loss	0.3	**3.35**	**19.19**	**0.1391**	**25.58**	**0.72**	**26.46**
spectral loss		3.40	21.41	0.1477	23.98	0.69	39.71
FFT loss	0.5	**3.35**	**17.62**	**0.1404**	**24.97**	**0.71**	**36.91**
spectral loss		3.75	19.63	0.1427	24.35	0.68	44.14

Table A.4: The qualitative results between FFT loss and spectral loss on WGANGP setting in configuration B. **Bold** values highlight the best performance.

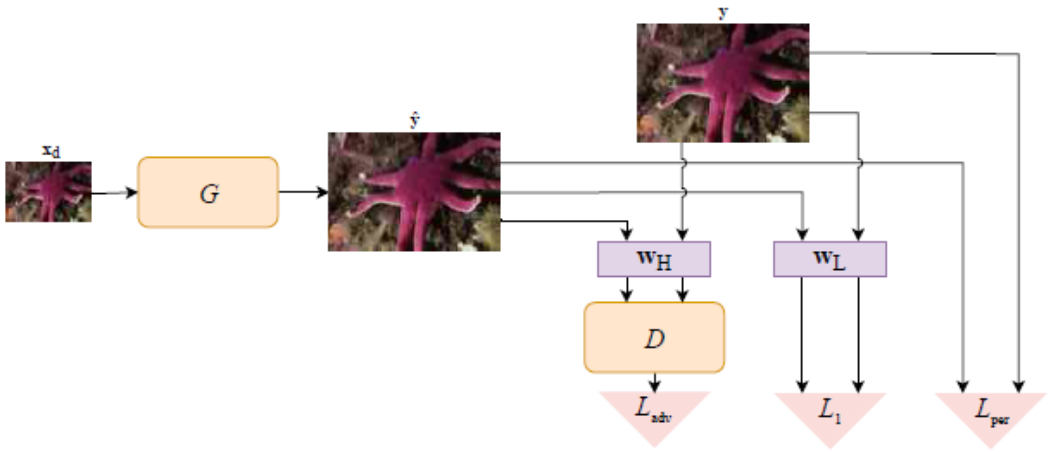
Configuration RaGAN (config B)		Metric score					
FFT/spectral loss	λ_f	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
✗	✗	4.01	20.81	0.1488	24.74	0.70	27.58
FFT loss	0.3	**2.93**	**20.19**	**0.1533**	**24.73**	**0.70**	33.93
spectral loss		3.53	22.14	0.1502	24.32	0.67	**29.54**
FFT loss	0.5	**3.05**	**18.02**	**0.1445**	**25.48**	**0.73**	37.01
spectral loss		3.58	20.36	0.1523	24.50	0.70	**30.58**

Table A.5: The qualitative results between FFT loss and spectral loss on RaGAN setting in configuration B. **Bold** values highlight the best performance.

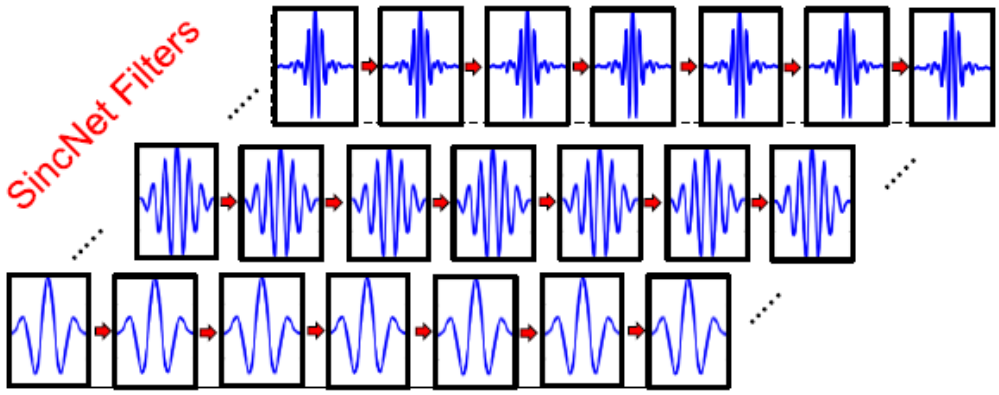
B.2 Comparison with other methods

We have already provided a comprehensive study between our FFT loss and spectral loss in section 5.2.3.3. To explore the results, we also examine some potential approaches from other authors. Jiang et al. [54] share the same strategy with our work when they develop a focal loss to correct a wrong spectral distribution. However, they construct their solution by geometry rather than algebra as we did. Instead of intervening in the loss function, some other choices try to modify the network input. With this aim, the HFC method [71] swaps the low-frequency and high-frequency components between real and fake images, while frequency separation [33] separates these details by a filter. Finally, we observe the technique named SSD-GAN [19], which integrates the spectral classifier to further provide frequency guides for the generator.

For implementation details, because most of the above works do not directly research on super-resolution, we re-implement their ideas in our code corresponding to configuration C in Table 5.5. However, we encounter difficulty in the frequency separation case, due to their filter description is very vague. Because a sinc filter is an ideal low-pass filter [113], we adopt SincNet [100] architecture in our experiment. The illustration figure for these approaches can be found in Figures 2.8 and A.5. The numerical results are presented in Table A.6. In short, some methods achieve a promising result in a specific case, but our work get the best performance in overall and in highest-priority metrics.



Frequency separation strategy [33]



SincNet [100]

Figure A.5: Frequency separation with SincNet filter illustration. **Purple block:** Low-pass filter and high-pass filter.

Configuration	**Metric score**					
Config C with	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
Original setting	3.27	19.80	0.1427	24.21	0.68	45.21
Focal loss ($\lambda_f = 0.003$)	3.30	21.04	0.1559	23.97	0.66	**35.60**
Focal loss ($\lambda_f = 0.005$)	3.46	19.68	0.1494	24.47	0.68	36.64
HFC (R = 16)	14.92	340.25	0.8827	7.77	0.11	144.87
HFC (R = 120)	34.55	385.81	0.9694	8.10	0.02	152.75
Sinc	6.93	30.53	**0.1316**	24.04	0.63	47.61
SSD	7.28	34.13	0.1368	24.40	0.66	54.86
FFT loss	**3.20**	**17.54**	0.1410	**25.67**	**0.74**	37.99

Table A.6: More experiments with FFT loss. **Notation:** λ_f: a frequency penalty coefficient and R: a frequency threshold radius (detail in [71])

B.3 The influence of different Fourier transform

In the early stage of this research, we uncertain about the suitable Fourier transform for the super-resolution task. While DFT shows weaknesses in large complexity, the comparison between FFT and DCT is highly ambiguous. In contrast to previous work

Configuration	Metric score					
Config C with	NIQE↓	FID↓	LPIPS↓	PSNR↑	SSIM↑	BRISQUE↓
FFT loss	**3.20**	**17.54**	**0.1410**	**25.67**	**0.74**	**37.99**
DCT loss	3.41	20.43	0.1471	24.11	0.68	43.10

Table A.7: The comparison between FFT loss and DCT loss. **Notation:** λ_f: a frequency penalty coefficient and R: a frequency threshold radius (detail in [71])

in [22], which utilized DCT to obtain the small translational shift insensitive, we found that FFT offers more particular information. Thus, the quantitative scores of FFT outweigh those of DCT in all super-resolution metrics (please see Table A.7).

C More qualitative comparison for image quality

Following our discussion in section 5.2.4, we provide more qualitative comparison in this section.

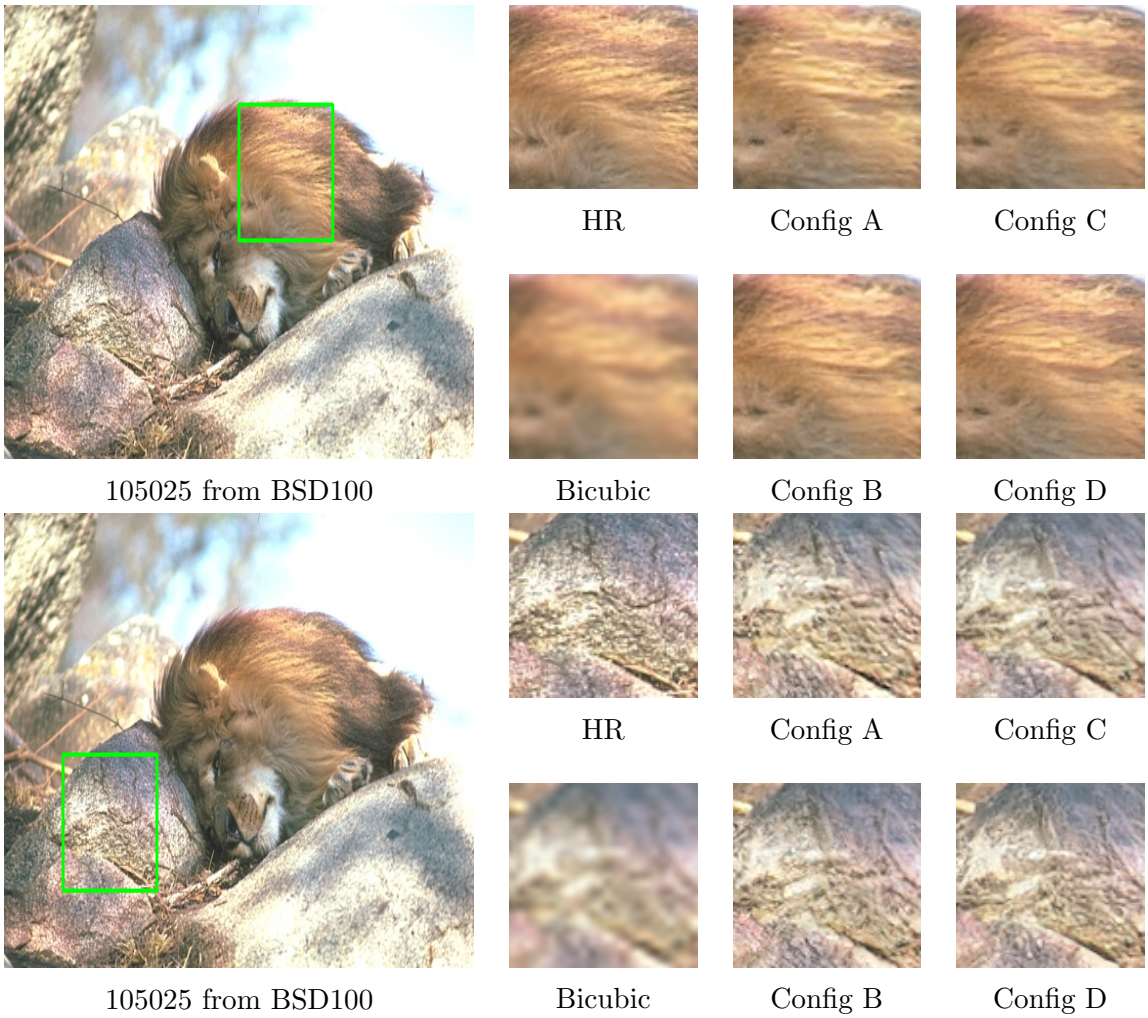


Figure A.6: Visual comparison between different loss in image **105025**. **A:** VGG-RaGAN, **B:** LPIPS-RaGAN, **C:** LPIPS-RaGP, **D:** LPIPS-RaGP with FFT loss. Please zoom for better comparison.

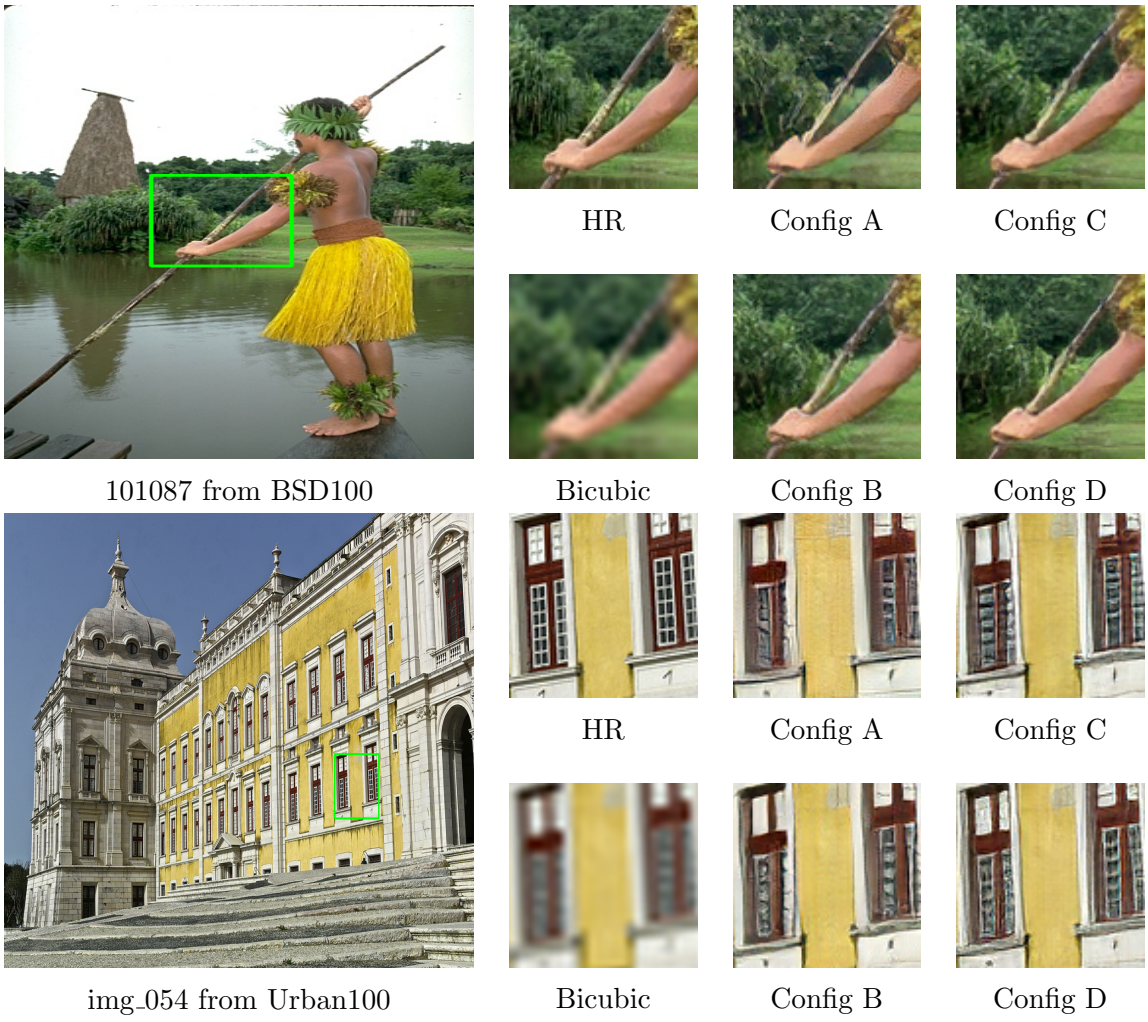


Figure A.7: Visual comparison between different loss in images **101087** and **img_054**. **A:** VGG-RaGAN, **B:** LPIPS-RaGAN, **C:** LPIPS-RaGP, **D:** LPIPS-RaGP with FFT loss. Please zoom for better comparison.

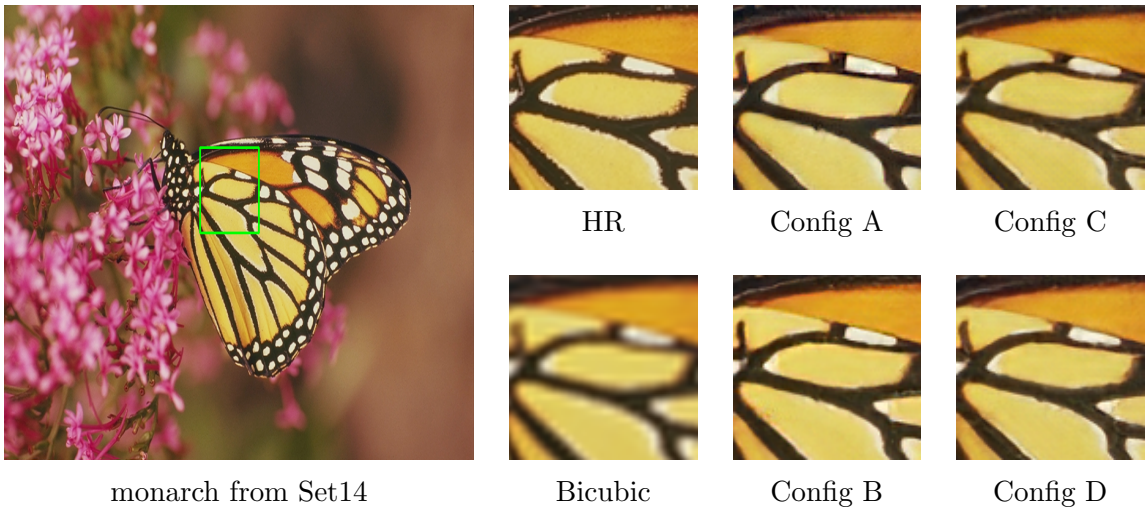


Figure A.8: Visual comparison between different loss in image **monarch**. All models provide a better result than the bicubic setting, but their outputs are nearly identical. The configuration details can be found in Table 5.5

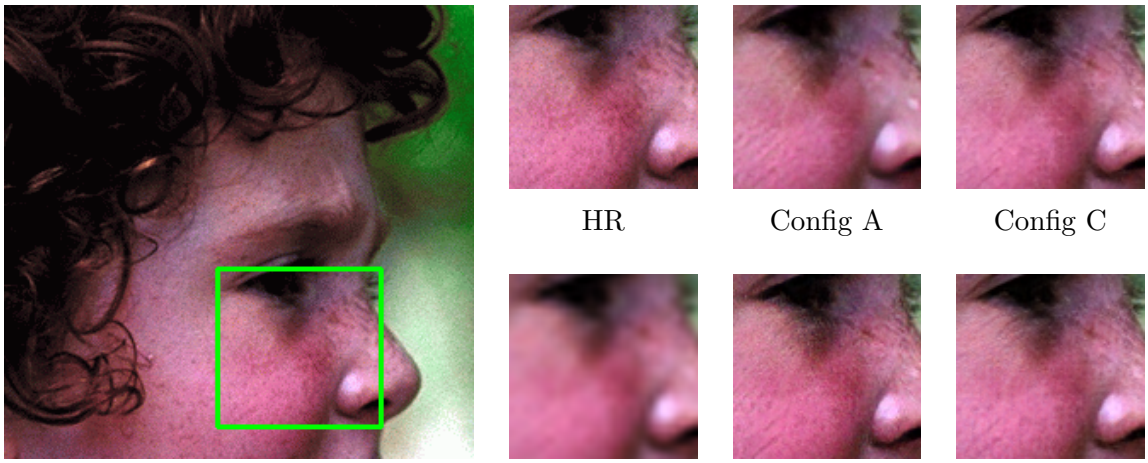


Figure A.9: Visual comparison between different loss in image **face**. All models provide a better result than the bicubic setting, but their outputs are nearly identical. The configuration details can be found in Table 5.5

D More qualitative comparison for image diversity

In this section, we provide more examples of our diversify outputs described in section 4.2.5

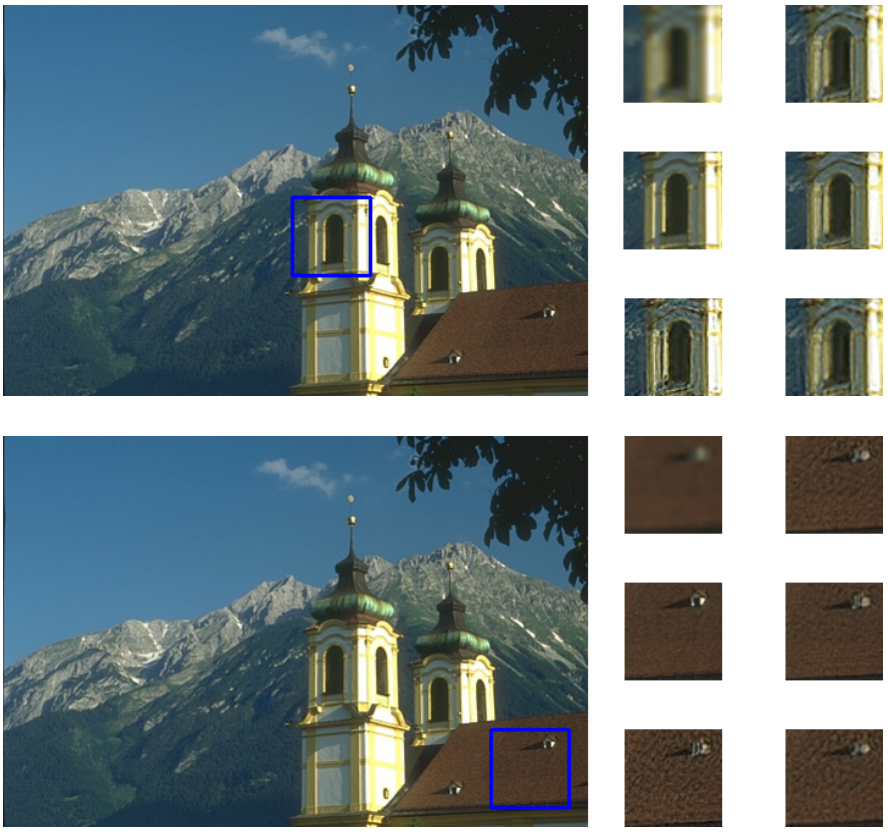


Figure A.10: **Top left**: Bicubic. **Middle left**: HR. **Others**: Random SR samples generated for image 126007 from BSD100

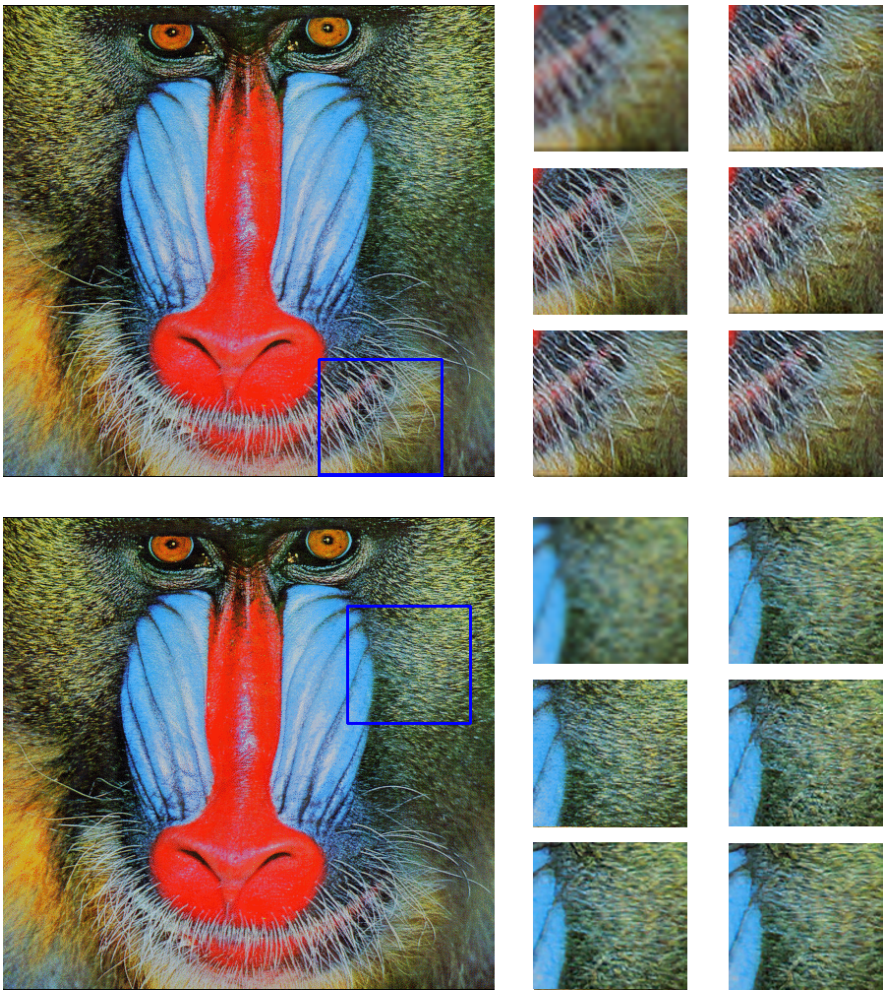


Figure A.11: **Top left**: Bicubic. **Middle left**: HR. **Others**: Random SR samples generated for image baboon from Set14

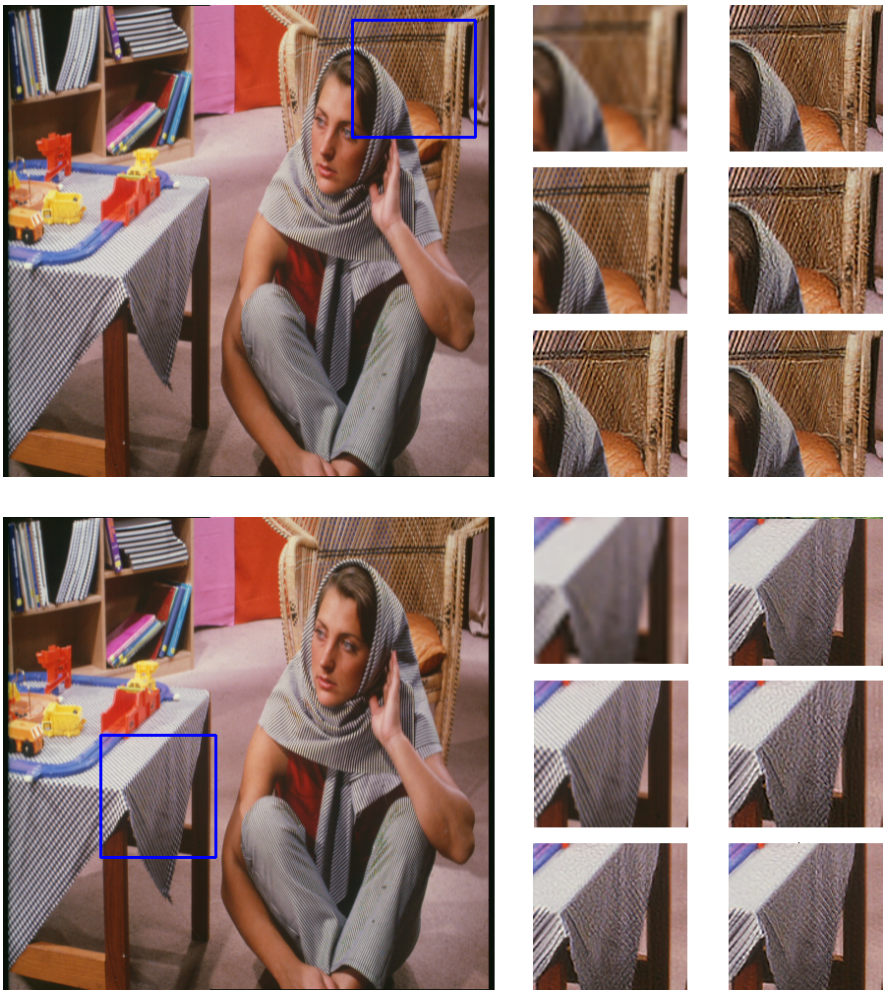


Figure A.12: **Top left**: Bicubic. **Middle left**: HR. **Others**: Random SR samples generated for image barbara from Set14

Bibliographies

[1] A. K. Moorthy A. Mittal and A. C. Bovik. "Making a 'completely blind' image quality analyzer". In: *IEEE Signal Processing Letters* 20.3 (2013), pp. 209–212.

[2] Jorge Agnese et al. "A survey and taxonomy of adversarial neural networks for text-to-image synthesis". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10 (2020).

[3] Eirikur Agustsson and Radu Timofte. "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2017.

[4] Jayesh Bapu Ahire. *The Artificial Neural Networks Handbook: Part 4*. 2018. URL: https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e (visited on 12/26/2020).

[5] Anastasia Antsiferova et al. "Barriers towards no-reference metrics application to compressed video quality analysis: on the example of no-reference metric NIQE". In: *CoRR* abs/1907.03842 (2019). URL: http://arxiv.org/abs/1907.03842.

[6] Saeed Anwar, Salman Khan, and Nick Barnes. "A Deep Journey into Super-Resolution: A Survey". In: *ACM Computing Surveys* 53 (May 2020). DOI: 10.1145/3390462.

[7] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Aug. 2017, pp. 214–223.

[8] Shahrukh Athar and Zhou Wang. "A Comprehensive Performance Evaluation of Image Quality Assessment Algorithms". In: *IEEE Access* 7 (2019), pp. 140030–140070. DOI: 10.1109/ACCESS.2019.2943319.

[9] Yuval Bahat and Tomer Michaeli. "Explorable Super Resolution". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2716–2725.

[10] Marco Bevilacqua et al. "Low-Complexity Single Image Super-Resolution Based on Nonnegative Neighbor Embedding". In: (Sept. 2012). DOI: 10.5244/C.26.135.

[11] Yochai Blau et al. "The 2018 PIRM Challenge on Perceptual Image Super-Resolution". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

[12] Ali Borji. "Pros and Cons of GAN Evaluation Measures". In: (2018). arXiv: 1802.03446 [cs.CV].

[13] Marcel C. Buhler, Andres Romero, and Radu Timofte. "DeepSEE: Deep Disentangled Semantic Explorative Extreme Super-Resolution". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Nov. 2020.

[14] A. Bulat and G. Tzimiropoulos. "Super-FAN: Integrated Facial Landmark Localization and Super-Resolution of Real-World Low Resolution Faces in Arbitrary Poses with GANs". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 109–117. DOI: 10.1109/CVPR.2018.00019.

[15] Adrian Bulat, Jing Yang, and Georgios Tzimiropoulos. "To learn image super-resolution, use a GAN to learn how to do image degradation first". In: Sept. 2018.

[16] Y. Cao et al. "Recent Advances of Generative Adversarial Networks in Computer Vision". In: *IEEE Access* 7 (2019), pp. 14985–15006. DOI: 10.1109/ACCESS.2018.2886814.

[17] Kelvin CK Chan et al. "GLEAN: Generative Latent Bank for Large-Factor Image Super-Resolution". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2021.

[18] Kaiming He Chao Dong Chen Change Loy and Xiaoou Tang. "Learning a Deep Convolutional Network for Image Super-Resolution". In: (2014).

[19] Yuanqi Chen et al. "SSD-GAN: Measuring the Realness in the Spatial and Spectral Domains". In: *AAAI*. 2021.

[20] Precious Chima. *Activation Functions: ReLU & Softmax*. 2020. URL: https://medium.com/@preshchima/activation-functions-relu-softmax-87145bf39288 (visited on 12/22/2020).

[21] Roger L. Cooke. *Classical Algebra: Its Nature, Origins, and Uses*. Wiley, 2008.

[22] S. Czolbe et al. "A Loss Function for Generative Neural Networks Based on Watson's Perceptual Model". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*. Advances in Neural Information Processing Systems. NeurIPS Proceedings, 2020.

[23] Tao Dai et al. "Second-Order Attention Network for Single Image Super-Resolution". In: (2019), pp. 11057–11066.

[24] Bekir Z Demiray, Muhammed Sit, and Ibrahim Demir. "D-SRGAN: DEM Super-Resolution with Generative Adversarial Networks". In: *arXiv e-prints* (Apr. 2020). eprint: 2004.04788.

[25] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[26] Keyan Ding et al. "Image Quality Assessment: Unifying Structure and Texture Similarity". In: *CoRR* abs/2004.07728 (2020). URL: https://arxiv.org/abs/2004.07728.

[27] Ricard Durall, Margret Keuper, and Janis Keuper. "Watch Your Up-Convolution: CNN Based Generative Deep Neural Networks Are Failing to Reproduce Spectral Distributions". In: (June 2020).

[28] Ruaa A. Al-falluji, A. A. Youssif, and S. Guirguis. "Single Image Super Resolution Algorithms: A Survey and Evaluation". In: *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 6 (2017).

[29] Farzan Farnia and Asuman Ozdaglar. "GANs May Have No Nash Equilibria". In: *ArXiv* arXiv:2002.09124 (Feb. 2020).

[30] Toadere Florin, Radu Arsinte, and Nikos E. Mastorakis. "Resolution Analysis of an Image Acquisition System". In: *Proceedings of the 5th European Conference on European Computing Conference*. 2011, pp. 382–391.

[31] David A. Forsyth and Jean Ponce. *Computer Vision: A modern approach, second edition*. Pearson, 2014.

[32] Joel Frank et al. "Leveraging Frequency Analysis for Deep Fake Image Recognition". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. June 2020, pp. 3247–3258.

[33] Manuel Fritsche, Shuhang Gu, and Radu Timofte. "Frequency Separation for Real-World Super-Resolution". In: (2019).

[34] H. Gao et al. "Pixel Transposed Convolutional Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.5 (2020), pp. 1218–1227. DOI: 10.1109/TPAMI.2019.2893965.

[35] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. *A Neural Algorithm of Artistic Style*. 2015. arXiv: 1508.06576 [cs.CV].

[36] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 2018.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[38] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014, pp. 2672–2680.

[39] Sam Gross and Michael Wilber. *Training and investigating Residual Nets*. 2016. URL: http://torch.ch/blog/2016/02/04/resnets.html (visited on 12/21/2020).

[40] Jie Gui et al. "A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications". In: *ArXiv* abs/2001.06937 (2020).

[41] Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf.

[42] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[43] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

[44] Yongjun Hong et al. “How Generative Adversarial Nets and its variants Work: An Overview of GAN”. In: *ACM Computing Surveys* 52 (2017), pp. 1–43.

[45] X. Hu et al. “Meta-SR: A Magnification-Arbitrary Network for Super-Resolution”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 1575–1584. DOI: 10.1109/CVPR.2019.00167.

[46] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[47] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. “Single image super-resolution from transformed self-exemplars”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5197–5206. DOI: 10.1109/CVPR.2015.7299156.

[48] Xun Huang and Serge Belongie. “Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization”. In: *ICCV*. 2017.

[49] Jonathan Hui. *GAN - Why is it so hard to train*. 2018. URL: https://medium.com/swlh/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b (visited on 12/26/2020).

[50] Jonathan Hui. *GAN — Wasserstein GAN and WGAN-GP*. 2018. URL: https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490 (visited on 05/03/2021).

[51] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[52] A. Jabbar, X. Li, and Bourahla Omar. “A Survey on Generative Adversarial Networks: Variants, Applications, and Training”. In: *ArXiv* abs/2006.05132 (June 2020).

[53] Jianchao Yang et al. “Image super-resolution as sparse representation of raw image patches”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587647.

[54] Liming Jiang et al. “Focal Frequency Loss for Image Reconstruction and Synthesis”. In: *arXiv preprint* arXiv:2012.12821 (2020).

[55] Younghyun Jo, Sejong Yang, and Seon Joo Kim. “Investigating Loss Functions for Extreme Super-Resolution”. In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (June 2020).

[56] Alexia Jolicoeur-Martineau. “On Relativistic f-Divergences”. In: *arXiv preprint arXiv:1901.02474* (2019).

[57] Alexia Jolicoeur-Martineau. “The relativistic discriminator: a key element missing from standard GAN”. In: *ArXiv* abs/1807.00734 (2018).

[58] Thomas B. Moeslund Kamal Nasrollahi. “Super-resolution: a comprehensive survey”. In: *Machine Vision and Applications* 25 (2014), pp. 1423–1468. DOI: https://doi.org/10.1007/s00138-014-0623-4.

[59] Tero Karras et al. “Analyzing and Improving the Image Quality of StyleGAN”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, pp. 8107–8116. DOI: 10.1109/CVPR42600.2020.00813. URL: https://doi.org/10.1109/CVPR42600.2020.00813.

[60] Shiqi Wang Keyan Ding Kede Ma and Eero P. Simoncelli. “Comparison of Full-Reference Image Quality Models for Optimization of Image Processing Systems”. In: *International Journal of Computer Vision* 129 (2021), pp. 1258–1281.

[61] Savya Khosla. *CNN - Introduction to Pooling Layer*. 2019. URL: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ (visited on 12/15/2020).

[62] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. “Deeply-Recursive Convolutional Network for Image Super-Resolution”. In: (2016), pp. 1637–1645. DOI: 10.1109/CVPR.2016.181.

[63] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[64] Diederik P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *CoRR* abs/1312.6114 (Dec. 2014).

[65] Durk P Kingma and Prafulla Dhariwal. “Glow: Generative Flow with Invertible 1x1 Convolutions”. In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018, pp. 10215–10224.

[66] Berardino A Laparra V Ballé J and Simoncelli. “Perceptual image quality assessment using a normalized Laplacian pyramid”. In: 6 (2016), pp. 1–6.

[67] Eric Cooper Larson and Damon Michael Chandler. “Most apparent distortion: full-reference image quality assessment and the role of strategy”. In: *Journal of Electronic Imaging* 19.1 (2010), pp. 1–21. DOI: 10.1117/1.3267105. URL: https://doi.org/10.1117/1.3267105.

[68] Yann Lecun and Yoshua Bengio. “Convolutional networks for images, speech, and time-series”. In: *The handbook of brain theory and neural networks*. MIT Press, 1995.

[69] C. Ledig et al. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. 2017. DOI: 10.1109/CVPR.2017.19.

[70] Zhen Li et al. “Feedback Network for Image Super-Resolution”. In: (June 2019).

[71] Ziqiang Li et al. “Are High-Frequency Components Beneficial for Training of Generative Adversarial Networks”. In: (Mar. 2021).

[72] Bee Lim et al. “Enhanced Deep Residual Networks for Single Image Super-Resolution”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. July 2017.

[73] Rui Liu et al. “DivCo: Diverse Conditional Image Synthesis via Contrastive Generative Adversarial Network”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021.

[74] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[75] Mario Lučić et al. "Are GANs Created Equal? A Large-Scale Study". In: *Advances in Neural Information Processing Systems*. 2018. URL: https://arxiv.org/pdf/1711.10337.pdf.

[76] Andreas Lugmayr et al. "SRFlow: Learning the Super-Resolution Space with Normalizing Flow". In: *ECCV*. 2020.

[77] Yuheng Ma, Zhi-Qin John Xu, and Jiwei Zhang. "Frequency Principle in Deep Learning Beyond Gradient-descent-based Training". In: *CoRR* (2021). URL: https://arxiv.org/abs/2101.00747.

[78] Shunta Maeda. *Unpaired Image Super-Resolution using Pseudo-Supervision*. 2020.

[79] Qi Mao et al. "Mode Seeking Generative Adversarial Networks for Diverse Image Synthesis". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[80] Xudong Mao et al. "Least Squares Generative Adversarial Networks". In: (2017), pp. 2813–2821. DOI: 10.1109/ICCV.2017.304.

[81] D. Martin et al. "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, 416–423 vol.2. DOI: 10.1109/ICCV.2001.937655.

[82] AI Master. *Sigmoid Function*. 2019. URL: https://ai-master.gitbooks.io/logistic-regression/content/sigmoid-function.html?q= (visited on 12/22/2020).

[83] Jacob Menick and Nal Kalchbrenner. "Generating High Fidelity Images with Subscale Pixel Networks and Multidimensional Upscaling". In: *ICLR* (Dec. 2018).

[84] Sachit Menon et al. "PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 2434–2442.

[85] Luke Metz et al. *Unrolled Generative Adversarial Networks*. 2017. arXiv: 1611.02163 [cs.LG].

[86] Peyman Milanfar. *Super-resolution imaging*. CRC Press, 2011.

[87] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. "No-Reference Image Quality Assessment in the Spatial Domain". In: *IEEE Transactions on Image Processing* 21.12 (2012), pp. 4695–4708. DOI: 10.1109/TIP.2012.2214050.

[88] Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: (2018). URL: https://openreview.net/forum?id=B1QRgziT-.

[89] Ben Niu et al. "Single Image Super-Resolution via a Holistic Attention Network". In: (2020).

[90] Mark S. Nixon and Alberto S. Aguado. *Feature Extraction and Image Processing for Computer Vision*. Academic Press, 2020.

[91] Evangelos Ntavelis et al. “SESAME: Semantic Editing of Scenes by Adding, Manipulating or Erasing Objects”. In: *Lecture Notes in Computer Science* (2020), pp. 394–411. ISSN: 1611-3349. DOI: 10.1007/978-3-030-58542-6_24. URL: http://dx.doi.org/10.1007/978-3-030-58542-6_24.

[92] Augustus Odena, Vincent Dumoulin, and Chris Olah. “Deconvolution and Checkerboard Artifacts”. In: *Distill* (2016). DOI: 10.23915/distill.00003. URL: http://distill.pub/2016/deconv-checkerboard.

[93] Seong-Jin Park et al. “SRFeat: Single Image Super-Resolution with Feature Discrimination”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[94] Srdan Popic et al. “Data generators: a short survey of techniques and use cases with focus on testing”. In: (2019), pp. 189–194. DOI: 10.1109/ICCE-Berlin47944.2019.8966202.

[95] Mohammad Saeed Rad et al. “SROBB: Targeted Perceptual Loss for Single Image Super-Resolution”. In: (Oct. 2019).

[96] A. Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *CoRR* abs/1511.06434 (2016).

[97] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *CoRR* abs/1511.06434 (2016).

[98] Nasim Rahaman et al. “On the Spectral Bias of Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 5301–5310.

[99] Fathy Rashad. *Generating Novel Content without Dataset - Rewriting the rules in GAN: Copy & paste features contextually*. 2020. URL: https://blog.crossminds.ai/post/generating-novel-content-without-dataset-rewriting-the-rules-in-gan-copy-paste-features-contextually (visited on 12/23/2020).

[100] Mirco Ravanelli and Yoshua Bengio. “Speaker Recognition from Raw Waveform with SincNet”. In: *2018 IEEE Spoken Language Technology Workshop (SLT)* (2018), pp. 1021–1028.

[101] M. Razaviyayn et al. “Nonconvex Min-Max Optimization: Applications, Challenges, and Recent Theoretical Advances”. In: *IEEE Signal Processing Magazine* 37.5 (2020), pp. 55–66. DOI: 10.1109/MSP.2020.3003851.

[102] Stefano Romanazzi. *BolognaNN - Photo Geolocation in Bologna with Convolutional Neural Networks*. 2018. URL: https://www.researchgate.net/figure/Plot-of-the-LeakyReLU-function_fig9_325226633 (visited on 12/20/2020).

[103] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 12/22/2020).

[104] Mehdi S. M. Sajjadi, Bernhard Schölkopf, and Michael Hirsch. "EnhanceNet: Single Image Super-Resolution through Automated Texture Synthesis". In: *CoRR* abs/1612.07919 (2016). arXiv: 1612.07919. URL: http://arxiv.org/abs/1612.07919.

[105] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. "RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[106] S. Schulter, C. Leistner, and H. Bischof. "Fast and accurate image upscaling with super-resolution forests". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3791–3799. DOI: 10.1109/CVPR.2015.7299003.

[107] Bovik AC Sheikh HR Wang Z and Cormack L. *Image and video quality assessment research at LIVE*. 2006. URL: http://live.ece.utexas.edu/research/quality/ (visited on 12/21/2020).

[108] W. Shi et al. "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1874–1883. DOI: 10.1109/CVPR.2016.207.

[109] Umit Mert Cakmak Sibanjan Das. *Hands-On Automated Machine Learning*. 2019. URL: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ (visited on 12/15/2020).

[110] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015.

[111] Wanjie Sun and Zhenzhong Chen. "Learned image downscaling for upscaling using content adaptive resampler". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 4027–4040.

[112] H. Surendra and .S Mohan.H. "A Review Of Synthetic Data Generation Methods For Privacy Preserving Data Publishing". In: *International Journal of Scientific & Technology Research* 6 (2017), pp. 95–101.

[113] Richard Szeliski. *Computer Vision: Algorithms and Applications, second edition*. 2020. URL: https://szeliski.org/Book/.

[114] Nao Takano and Gita Alaghband. "SRGAN: Training Dataset Matters". In: (Mar. 2019).

[115] Matthew Tancik et al. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *NeurIPS* (2020).

[116] Radu Timofte, Vincent De Smet, and Luc Van Gool. "A+: Adjusted Anchored Neighborhood Regression for Fast Super-Resolution". In: *Computer Vision - ACCV 2014 - 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part IV*. Ed. by Daniel Cremers et al. Vol. 9006. Lecture Notes in Computer Science. Springer, 2014, pp. 111–126. DOI: 10.1007/978-3-319-16817-3_8. URL: https://doi.org/10.1007/978-3-319-16817-3%5C_8.

[117] Radu Timofte et al. “NTIRE 2018 Challenge on Single Image Super-Resolution: Methods and Results”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018.

[118] Fjodor Van Veen. *The Neural Network Zoo*. 2016. URL: https://www.asimovinstitute.org/neural-network-zoo/ (visited on 12/23/2020).

[119] James Vincent. 2019. URL: https://www.theverge.com/tldr/2019/2/15/18226005/ai-generated-fake-people-portraits-thispersondoesnotexist-stylegan (visited on 12/26/2020).

[120] Xintao Wang et al. “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

[121] Xintao Wang et al. “Recovering Realistic Texture in Image Super-Resolution by Deep Spatial Feature Transform”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[122] Z. Wang, A. C. Bovik, and L. Lu. “Why is image quality assessment so difficult?” In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4. 2002, pp. IV-3313-IV–3316. DOI: 10.1109/ICASSP.2002.5745362.

[123] Z. Wang, J. Chen, and S. C. H. Hoi. “Deep Learning for Image Super-resolution: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. DOI: 10.1109/TPAMI.2020.2982166.

[124] Z. Wang, E.P. Simoncelli, and A.C. Bovik. “Multiscale structural similarity for image quality assessment”. In: 2 (2003), 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.

[125] Zhengwei Wang, Qi She, and T. Ward. “Generative Adversarial Networks: A Survey and Taxonomy”. In: *ArXiv* abs/1906.01529 (2019).

[126] Zhou Wang and E.P. Simoncelli. “Translation insensitive image similarity in complex wavelet domain”. In: 2 (2005), ii/573–ii/576 Vol. 2. DOI: 10.1109/ICASSP.2005.1415469.

[127] Xian Wu, Kun Xu, and Peter Hall. “A survey of image synthesis and editing with generative adversarial networks”. In: *Tsinghua Science and Technology* 22.6 (2017), pp. 660–674. DOI: 10.23919/TST.2017.8195348.

[128] Zhi-Qin John Xu. “Frequency Principle in Deep Learning with General Loss Functions and Its Potential Application”. In: (2018). arXiv: 1811.10146.

[129] Wufeng Xue et al. “Gradient Magnitude Similarity Deviation: A Highly Efficient Perceptual Image Quality Index”. In: *IEEE Transactions on Image Processing* 23.2 (2014), pp. 684–695. DOI: 10.1109/TIP.2013.2293423.

[130] Dingdong Yang et al. “Diversity-Sensitive Conditional Generative Adversarial Networks”. In: *Proceedings of the International Conference on Learning Representations*. 2019.

[131] W. Yang et al. “Deep Learning for Single Image Super-Resolution: A Brief Review”. In: *IEEE Transactions on Multimedia* 21.12 (2019), pp. 3106–3121. DOI: 10.1109/TMM.2019.2919431.

[132] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. "Time-series Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019, pp. 5508–5518. URL: https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.

[133] Xiaoming Yu et al. "Multi-mapping Image-to-Image Translation via Learning Disentanglement". In: *Advances in Neural Information Processing Systems*. 2019.

[134] Linwei Yue et al. "Image super-resolution: The techniques, applications, and future". In: *Signal Processing* 128 (2016), pp. 389–408. DOI: https://doi.org/10.1016/j.sigpro.2016.05.002.

[135] M. D. Zeiler et al. "Deconvolutional networks". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2528–2535. DOI: 10.1109/CVPR.2010.5539957.

[136] Roman Zeyde, Michael Elad, and Matan Protter. "On Single Image Scale-up Using Sparse-Representations". In: *Proceedings of the 7th International Conference on Curves and Surfaces*. 2010, pp. 711–730. DOI: 10.1007/978-3-642-27413-8_47. URL: https://doi.org/10.1007/978-3-642-27413-8_47.

[137] Jiqing Zhang et al. "A Two-Stage Attentive Network for Single Image Super-Resolution". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021).

[138] Kai Zhang, Shuhang Gu, and Radu Timofte. "NTIRE 2020 Challenge on Perceptual Extreme Super-Resolution: Methods and Results". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[139] Lin Zhang, Ying Shen, and Hongyu Li. "VSI: A Visual Saliency-Induced Index for Perceptual Image Quality Assessment". In: *IEEE Transactions on Image Processing* 23.10 (2014), pp. 4270–4281. DOI: 10.1109/TIP.2014.2346028.

[140] Lin Zhang et al. "FSIM: A Feature Similarity Index for Image Quality Assessment". In: *IEEE Transactions on Image Processing* 20.8 (2011), pp. 2378–2386. DOI: 10.1109/TIP.2011.2109730.

[141] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *CVPR*. 2018.

[142] Wenlong Zhang et al. *RankSRGAN: Generative Adversarial Networks with Ranker for Image Super-Resolution*. 2019. arXiv: 1908.06382 [cs.CV].

[143] Yulun Zhang et al. "Image Super-Resolution Using Very Deep Residual Channel Attention Networks". In: (Sept. 2018).

[144] Yulun Zhang et al. "Residual Dense Network for Image Super-Resolution". In: (2018), pp. 2472–2481. DOI: 10.1109/CVPR.2018.00262.

[145] Hang Zhao et al. "Loss Functions for Image Restoration With Neural Networks". In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 47–57. DOI: 10.1109/TCI.2016.2644865.

[146] Yuanbo Zhou et al. "Guided Frequency Separation Network for Real-World Super-Resolution". In: (June 2020).

[147] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[148] Jun-Yan Zhu et al. “Toward multimodal image-to-image translation”. In: *Advances in Neural Information Processing Systems*. 2017.

[149] Computer Vision Zurich. *Why I stopped using GAN*. 2020. URL: https://medium.com/swlh/why-i-stopped-using-gan-eccv2020-d2b20dcfe1d (visited on 12/26/2020).