EXPLICIT USE OF FOURIER SPECTRUM IN GENERATIVE ADVERSARIAL NETWORKS

SOROUSH SHEIKH GARGAR

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF APPLIED SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

SEPTEMBER 2021

Abstract

Generative Adversarial Networks have got the researchers' attention due to their state-of-the-art performance in generating new images with only a dataset of the target distribution. It has been shown that there is a dissimilarity between the spectrum of authentic images and fake ones. Since the Fourier transform is a bijective mapping, saying that the model has a significant problem in learning the original distribution is a fair conclusion. In this work, we investigate the possible reasons for the mentioned drawback in the architecture and mathematical theory of the current GANs. Then we propose a new model reducing the discrepancies between the spectrum of the actual and fake images. To that end, we design a brand new architecture for the frequency domain using the blueprint of geometric deep learning. Then, we experimentally show that promising improvements in the quality of the generated images by considering the Fourier domain representation of the original data as a principal feature in the training process.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Hui Jiang, whose guidance was always a light throughout all the project sections. I wish to extend my special thanks to Mr. Behnam (Ben) Asadi, my great friend and colleague at Laboratory for Neural Computing and Machine Learning (NCML), who played a significant role in developing the ideas leading to this thesis. I would like to thank my roommate, Mr. Fazel (Alex) Arasteh from the data mining laboratory of York University, for his assistance in the implementation sections of this project and his constant valuable advice.

Table of Contents

List of Tables

List of Figures

Abbreviations

Table 1: Abbreviations list

Abbreviation	**Full form**
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
iFFT	Inverse Fast Fourier Transform
iDFT	Inverse Discrete Fourier Transform
AI	Artificial Intelligence
ANN	Artificial Neural Networks
NN	Neural Networks
GAN	Generative Adversarial Networks
CNN	Convolutional Neural Networks
MLP	Multi Layer Perceptron
RGB	Red Green Blue (image contex)

Continued on next page

Table 1 – continued from previous page

Abbreviation	Full form
FID	Fréchet Inception Distance
IS	Inception Score
KL-divergence	Kullback–Leibler divergence
EM-distance	Earth Mover's Distance
CVPR	Conference on Computer Vision and Pattern Recognition
DCT	Discrete Cosine Transform
CPU	Central Processing Unit
GPU	Graphical Processing Unit
GDL	Geometric Deep Learning
i.i.d	Independent and Identically Distributed
SVM	Support Vector Machines
FC	Fully Connected
BN	Batch Norm
ReLU	Rectified Linear Unit

Chapter 1

Introduction

Generative adversarial networks (GANs) proposed by [1] have had a significant effect on the computer vision literature. They have been used in a wide range of applications, such as generating realistic images [2]–[4], image-to-image mappings [5], text-to-image translation [6]–[9], transfer learning [10], photograph editing [11]–[14], 3D object generation [15], [16] and many more that are still coming up. Image synthesis state-of-the-art works successfully generated photo-realistic images almost indistinguishable for human eyes. This rapid progress gave birth to the applications such as deepfake [17] in which the central idea is to fool the human eyes into believing the machine-generated images as real ones. Naturally, it raised security concerns on the problem challenging photo-forensics experts on the subject [18]. On the flip side, recent works [19] have shown that having a "large" amount of data, one can detect fake images with relative ease using simple classifiers powered by neural networks. [20] attributes the machine's power for classifying to the dissimilarities between the authentic

and artificial images in their spectrum. It further discusses the roots of dissimilarities in the high-frequency residuals of the image synthesis in convolutional neural networks. It also discusses that the checkerboard effect, one of the most disputed phenomenons in the GANs, is one of the consequences of the frequency discrepancies.

The Frequency domain is a well-established tool to process and analyze the images in the signal processing literature. It represents images without losing any information due to the bijective property of the FFT. This representation gives us the advantage when dealing with the rate of change within the neighborhood of the pixels, intuitively making it related to the problem of the checkerboard patterns and any other harsh changes regarding the details of an image.

The main objective of the current research is to find a deep learning based method to reduce the existing gap between the frequency spectrum of the real images and the ones produced by the generative adversarial networks. We also ask the question of how we can find a way to incorporate frequency information of the images in the deep learning driven computer vision tasks. To be precise, to build a deep learning layer for processing frequency information along with spatial information.

Imitating human responses and behaviors has always been a topic of utmost interest for artificial intelligence researchers. Spatial frequency theory [21] is an acclaimed construct showing the weight of the spatial frequencies in the brain's neural response. Recent research [22] also reveals that the mechanism of the human primary visual cortex (V1) is highly related to the second-order spatial frequencies received from our visual sensory inputs. These

spatial frequencies stand out in the spectrum representation of an image. Useful filters such as the Gabor filter applied on the spectrum of the images are also considered to be similar to the actions of cells in the mammalian visual cortex for texture discrimination [23]–[25]. Knowing how vital the frequency domain is in the ultimate baseline of the AI, meaning humans, motivates us even more to use models consist of frequency-related elements.

To get an image realistic to the finest detail, generative models need to learn the distribution of the images accurately enough to imitate the spectrum of the generated images, which is not the case for the current approaches. Identifying this problem, in this work, we are trying to propose a new add-on to the current architectures of the CNN-based GANs in order to incorporate the frequency representation with more emphasis in the training process.

The rest of this work is organized as follows: In chapter 2 the necessary information for understanding the ideas are provided. Chapter 3 introduces the current major advances in the literature of generative adversarial networks and frequency-driven efforts in the area of neural networks. Next, in chapter 4 we discuss the idea of the geometric deep learning, which is the main building block for our contribution. Chapter 5 formalizes the problem at hand and gives theoretical methodologies for our contributions. Chapter 6 assess the performance of the idea in spatial and frequency domain as well as a stability analysis of the implementations. Finally, chapter 7 summarizes the contributions of the paper and discusses the future steps for this research.

Chapter 2

Backgrounds

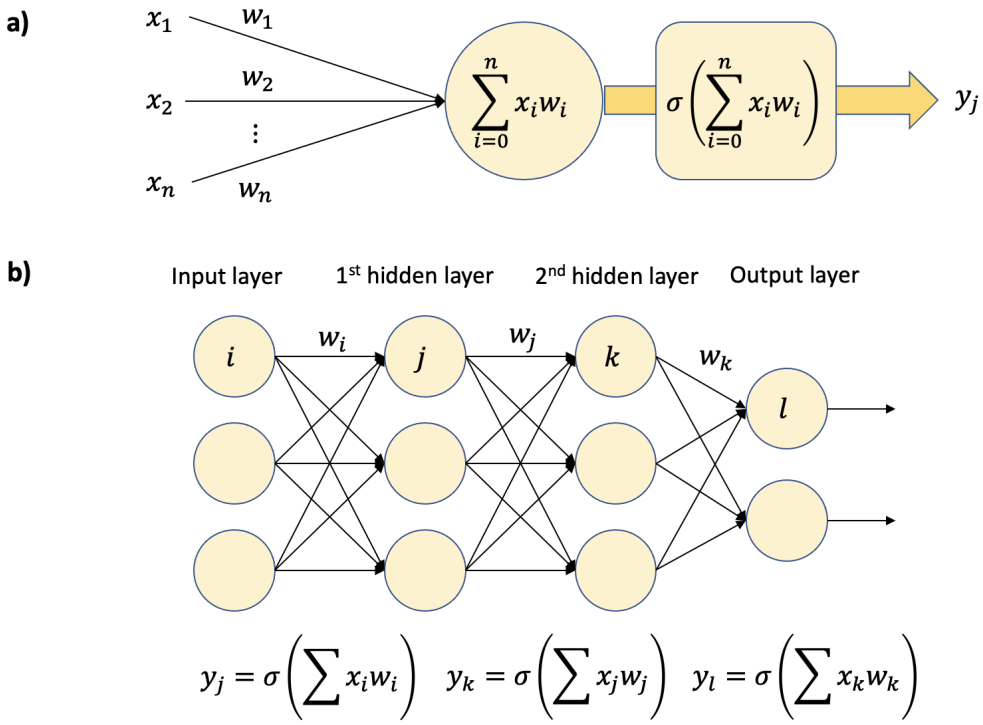
In this section, we first present the concept of neural networks, and with a simple example of MLP, we explain the procedure behind their ability to learn. Then we discuss Convolutional Neural Networks as the most successful neural network structure in vision tasks. Then we move to Generative Adversarial Networks, which are the main neural network architecture examined in the current work. We explain their structure, the basic mathematical idea behind them and intuitively explain how they are supposed to work. In section 2.3 we talk about the methods for GAN's quality evaluation in the spatial domain we used in our work. For evaluation in the frequency domain, we present three metrics later in 6.4. Section 2.4 gives a brief description of the frequency domain and its importance, plus some of its applications in natural language processing and computer vision. It also contains the fundamental mathematics of discrete Fourier transform in 1D and 2D.

2.1 Neural Networks

Neural Networks, also known as Artificial Neural Networks (ANNs), are a set of algorithms in machine learning. The recent deep learning revolution in computer science owes its existence to these frameworks of algorithms. They got their name on their design to mimic the structure of human neurons in the brain. The most simple structure of neural networks, known as Multi-Layer Perceptron (MLP), consists of a layer of nodes (neurons) as the input layer, one or more hidden layers, and an output layer connected with a designed architecture in a layer-wise manner. Values of the nodes in each layer can be seen as a vector and determined by three elements. The first element is the vector of the value of the previous layer. Second, a weight function linearly transforms the previous layer's outputs, represented as a matrix. Finally, a non-linear activation function adding non-linearity to the structure. Figure 2.1 shows the architecture a simple MLP with two hidden layers, and activation function σ.
NNs function upon training over a large amount of data, changing their wight matrices gradually with a process called backpropagation to improve their accuracy. The popularity of the NNs is owing to their quick performance and superb accuracy over data-driven tasks such as prediction, classification, and regression.

Convolutional Neural Networks (CNNs) CNNs are a subset of deep learning algorithms introduced in [26]. They are best for learning grid-like data such as Images and time series. They have achieved remarkable accuracy and became an inseparable part of image-related and video-related tasks such as object segmentation [27]–[29], image classification

Figure 2.1: (a) building blocks of the MLP with input vector $\mathbf{x}$, weights w and activation function σ, (b) an MLP structure with a two-neuron output layer and two hidden layers



[30]–[32], object detection [33]–[35] and image reconstruction [36]–[38]. Moreover, they have shown capabilities in natural language processing such as Sentiment Analysis and Topic Categorization tasks [39], [40], semantic clustering [41] and relation extraction [42]–[44]

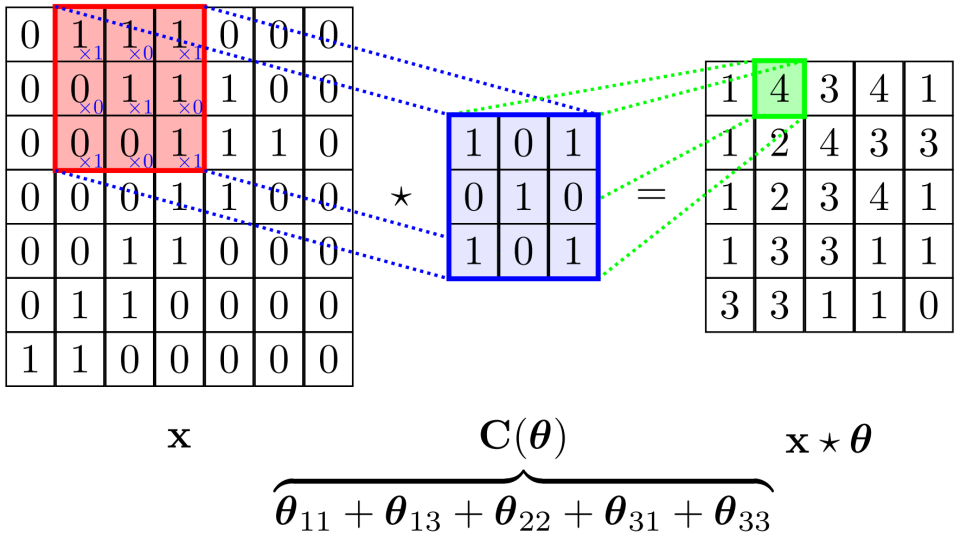
In a naive two-dimensional convolutional layer, the input should be in a grid form $\Omega = [H] \times [W]$ with vector $\mathbf{u} = (u_1, u_2)$ define over it. An image can be shown as signal in this space $\mathbf{x} \in \mathcal{X}(\Omega, \mathbb{R})$. A convolutional filter (whose weights are subject to learning) slides over the image. We can write the convolution step with filter size $H^f \times W^f$ using the linear combination of generators $\boldsymbol{\theta}_{1,1}, ...\boldsymbol{\theta}_{H^f,W^f}$. In here lets use the unit impulse function

$\boldsymbol{\theta}_{v,w}(u_1, u_2) = \delta(u_1 - v, u_2 - w)$.

$$\mathbf{F}(\mathbf{x}) = \sum_{v=1}^{H^f} \sum_{u=1}^{W^f} \alpha_{vw} \mathbf{C}(\boldsymbol{\theta}_{vw})\mathbf{x}$$

$\boldsymbol{\theta}_{vw}$ and $\mathbf{x}$ are 2D-matrices flattened into a vector here. Figure 2.2 shows the operation $\mathbf{C}(\boldsymbol{\theta}_{vw})\mathbf{x}$. In coordinates, it also corresponds to the standard notation of 2D-convolution

Figure 2.2: The convolution step with linear generators. (image credit for [45])



$$\mathbf{F}(\mathbf{x})_{uv} = \sum_{a=1}^{H^f} \sum_{b=1}^{W^f} \alpha_{ab} x_{u+a,v+b}$$

When we have more than one channel, e.g., RGB photos or in the later layers when we have several feature maps, a convolutional tensor is used instead of a convolutional filter. And, in coordinates, we have:

$$\mathbf{F}(\mathbf{x})_{uvj} = \sum_{a=1}^{H^f} \sum_{b=1}^{W^f} \sum_{c=1}^{M} \alpha_{jabc} x_{u+a,v+b,c}, \quad j \in [N]$$

where M and N are respectively the number of input and output channels.

After the convolutional step, the output, known as the feature map, is subjected to a pooling

operation which down-samples the feature map to reduce the redundancy and add robustness against over-fitting. Convolutional layers can be cascaded together to form a deep CNN, and they can also be a component of a neural network with linear layers haven their outputs flatten to a vector.

Deep-learning-powered computer vision has been thriving in virtue of the abilities CNNs unlock over the normal perceptron. The foremost power in CNNs is their ability to extract features such as patterns, texture, object edges, etc., automatically with convolutional kernels. Some other benefits of using CNNs are as follow: 1) avoiding parameter growth with increasing the number of inputs (let us say pixels for the case of images); this is on account of two facts, the weight sharing over layers, and having local connection which means each node is connected only to some local nodes in the previous layer not all of them. 2) Robustness against translation, in other words, when the input image is slightly shifted, CNN does not treat it as a completely different input. This characteristic, known as shift-equivariency, is explained in detail in chapter 4

2.2 Generative Adversarial Networks

Generative models In general, machine learning algorithms can be categorized into supervised and unsupervised learning. In the former, a labeled dataset is necessary. The goal is to predict an accurate function, mapping the features of the dataset to the target labels. Some examples of it are classification and regression. While in unsupervised learning, there is no labeled dataset; all the data are in the same category. Generative modeling is

an unsupervised task in machine learning aiming to learn the data distribution well enough so that the model can produce new data points with all the attributes of the input data. Intuitively, designing a generative model requires a deep understanding of the semantics of the data and the field. However, Generative Adversarial Networks (GANs) are a subcategory of generative models capturing the significant features of the dataset automatically.

Generative Adversarial Networks Goodfellow et al. [1] introduced the concept of Generative Adversarial Networks as an algorithmic architecture that uses two neural networks competing against each other. The goal is to generate new data points imitating the distribution of the original training dataset. The skeleton consists of two separate neural networks. A generator $\mathcal{G}$ that tries to capture training data distribution. Plus, a discriminator $\mathcal{D}$ whose goal is to predict the probability that its input is a generated fake sample or an original data point. The two networks will train simultaneously to satisfy the following two-player minimax game:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}(\mathbf{x})}[\log \mathcal{D}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{z}}(\mathbf{z})}[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z})))]. \tag{2.1}$$

$P_{\mathbf{z}}(\mathbf{z})$ is a defined prior over input noise variable $\mathbf{z}$, $\mathcal{G}(\mathbf{z}; \theta_g)$ is a mapping from noise latent space to the data space implemented with a neural network with parameters θ_g. $\mathcal{D}(\mathbf{z}; \theta_d)$ is another mapping from data space to a single scalar with parameters θ_d. $\mathcal{D}(\mathbf{x})$ shows the probability that $\mathbf{x}$ came from data distribution rather than generator. The first term of the right-hand side of the equation (2.1) is an expectation representing the quality of the model performance on detecting the authentic images. While the second term representing the

quality of the model on generated samples. Together they build up the loss function $V(\mathcal{D}, \mathcal{G})$, where the discriminator $\mathcal{D}$ tries to maximize and generator $\mathcal{G}$ tries to minimize. Training them simultaneously with precise hyperparameter tuning results in both networks getting better at their works in detecting or generating data points. At the final step, the stand-alone generator module will produce indistinguishable data points.

2.3 Methods for Measuring GAN's Quality

To measure the performance of the neural network models on supervised tasks, researchers use various metrics. For the classification, we have precision, recall, accuracy, F1-score, etc.; for regression tasks, we have mean square error, mean absolute deviation, etc. However, all of them need a labeled dataset out of hand for any unsupervised task, including GANs. Defining a metric for unsupervised tasks is more difficult. Clustering tasks, for example, are usually evaluated using the Silhouette Coefficient. It measures how similar an item in the cluster is to the other items in the same cluster. GANs case is different and even trickier; there is no label in hand, and there is no ground truth other than some samples from the real dataset. The loss for each neural network is not in any way an indicator of how successful the model is in generating a distribution close to the original sampled distribution. They only show how good is each module (Generator or Discriminator) in fooling the other one. Finding an excellent metric to evaluate the GANs performance has been a topic of controversy for a while now [46]. Nonetheless, for quantitative measurement of their success, there are two widely adopted metrics, the Inception Score (IS) [47] and Fréchet Inception Distance (FID)

[48].

Inception Score (IS) IS takes two main high-level concepts into account:

- each image decidedly belong to a class (image of a cat belongs to the cat class, and there is no confusion about putting it in another class)
- The images are diverse (the model can generate images of a wide variety of classes)

To mathematically talk about the concepts as mentioned earlier, using the notion of entropy is useful. The higher the entropy of the data, the less predictable its behaviors. Accordingly, if we get random variable x as the given images and y as the image class, to satisfy the first concept, we want the conditional probability $p(y|x)$ to have low entropy. In other words, predicting the class, given an image, should be a highly predictable task. For the second concept, we want the marginal probability $p(y)$ to have a high entropy; namely, we want the model to give us different classes as uniform as possible. Bringing together the two concepts, we have the final form of inception score using KL-divergence as the following:

$$\mathbf{IS}(\mathcal{G}) = \exp(\mathbb{E}_{\mathbf{x}\sim P_{\mathcal{G}}} D_{KL}(p(y|\mathbf{x})||p(y)) \tag{2.2}$$

For the computation of posterior distributions, the classifier, Inception Net, [49] is being used over the data (hence the name Inception Score). Then, the marginal probabilities are calculated by the following integral:

$$\int_{\mathbf{z}} p(y|\mathbf{x} = \mathcal{G}(\mathbf{z}))d\mathbf{z}$$

Fréchet Inception Distance (FID) FID introduced by [48] embeds a set of generated samples and also actual samples into a feature space. It uses an intermediate layer of Inception Net (usually after the third pooling layer). It deals with the embedded space like a multivariate Gaussian, and it estimates the mean vector and covariance matrix for both real and generated data distributions, μ_r, μ_g, Σ_r and Σ_g respectively. The Fréchet Inception Distance is then calculated between the two Gaussians.

$$\mathbf{FID}(r, g) = ||\mu_r - \mu_g||_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}}) \tag{2.3}$$

The trace function $Tr(.)$ is the sum of all the diagonal elements. Since it calculates how alike the authentic and generated distributions are, the lower the distance, the better the model; it is unlike the Inception Score, where the higher score is better. FID is usually a better metric than IS Since it is sensitive to GAN failures such as mode collapse. We explain failure modes in GAN's in section 6.2.

2.4 Frequency Domain

Decomposing an image into different size scales with a basis function is a prominent idea in mathematics and engineering. The idea is to write any spatial signal as a sum of orthogonal basis functions to get a new representation based on the different resolutions of the original signal. This representation is known as frequency representation. Most of the decomposition techniques, including the ones described in this thesis, are bijective mappings, i.e., there is a unique representation in the frequency domain for each spatial signal. Arguably frequency

domain's theoretical significance comes from the Convolution Theorem for standard Linear Time Invariant systems. The interactions of such systems with their input and output signals formulate by the convolution operator. The Fourier basis diagonalizes convolution, hence replace it with a simple element-wise multiplication in the frequency domain. The importance does not only summarize in theoretical works. The frequency-domain has opened up various tools for engineers. For audio signals applying filters to remove noise, detect different voices, or equalize the voices of different musical instruments are a small portion of many applications of frequency representations. In the case of the image spectrum, although the information is not apparent by the look of the spectrum, it contains valuable data related to the rate of change in moving to neighbor pixels, the lines and curves and edges of the objects, and noise in the images. It can also tell us how much information there is in a particular resolution, which is important when an image contains chaotic structures. There are helpful filters in noise reduction, edge detection, and many others operating on the images' frequency domain. A few series and transforms have been introduced to change the representation of spatial periodic and non-periodic signals to a frequency space, such as Fourier transforms and series, Laplace transforms, and Z-transform. In this work, we adopted the arguably most famous decomposition technique, Discrete Fourier Transform (DFT).

2.4.1 Power Spectrum

A power spectrum is, in essence, a measure of the strength of the different features at different resolutions. Basically, We decompose an image into different size scales, for example, a coarser

version like 2×2, and each new superpixel contains the dominant value of the corresponding pixels within the original image. If we subtract this dominant color from the original image, we are left with a version of the image without these largest structures. Repeating this procedure for less and less coarse versions of the original image give us a set of images (measurements on images) between the very coarse and the original image. Every image in between contains a measurement of how influential features in the original image are at that specific resolution, and the size scale associated with that resolution tells us what the size scale of those features is. For each resolution, the power spectrum is defined as the variance of the features on that resolution, i.e., how much total spread there is in values. A large spread means a strong signal and hence much variation on this scale.

To perform the size scale decomposition, we use DFT, which we will explain its detail in the next section. Basically, the time signal of each resolution corresponds to a sine and cosine pair with a period of $1/k$. We know k as the frequency. Since we are performing a 2D transform, vector $\mathbf{k}$ consists of two elements as well $\mathbf{k} = (k_x, k_y)$ and each of the elements has two amplitudes for cosine and sine. As we discussed earlier, the power spectrum is the variance of the features in size scales. However, since different combination of k elements can contribute to the same resolution (e.g. $\mathbf{k_1} = (k_x = 1, k_y = 2), \mathbf{k_2} = (k_x = 2, k_y = 1)$), the standard procedure is to bin them all together. The appropriate measure for the power for a specific $\mathbf{k}$ value is the average power within the corresponding $\mathbf{k}$ bin, multiplied with the volume of that bin in $\mathbf{k}$ space. Equation (2.4) shows the mathematical definition, and Fig 2.4 shows an example of the spectrum, power spectrum pair, indicated by two different $\mathbf{k}$ bins

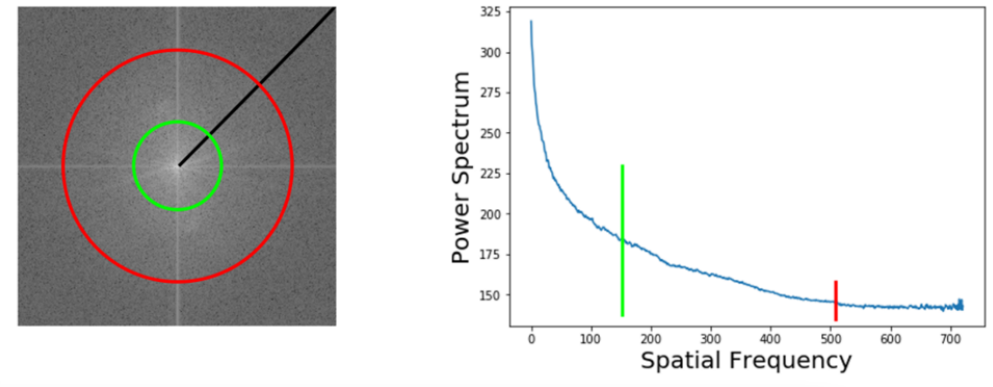
and their corresponding location on both spectrum and power spectrum.

$$PS(r) = \frac{1}{2\pi} \int_0^{2\pi} |F(r,\theta)| d\theta \tag{2.4}$$

where r and θ are coming from the standard change from Cartesian to polar coordinate change:

$$F(k,l) = F(r,\theta): \quad r = \sqrt{k^2 + l^2}, \quad \theta = \tan^{-1}(l,k)$$

Figure 2.3: left: spectrum of an image, right: power spectrum of the same image. yellow and red circles are equivalent resolutions which binned together in the power spectrum. Figure credit for [50]



2.4.2 Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT)

Fourier Transform presumably is the most common signal decomposition in the literature, and it is the foundation of the harmonic analysis. DFT is a sampled version of the Fourier transform, so it does not contain all the frequencies in the spatial function. however, it

contains enough frequency elements to describe one and only one spatial function without losing any information. 1D-DFT of the the function $f(x)$ with N points $N = 0, 1, ..., N-1$ is calculated as follow:

$$F(k) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} f(x) e^{-i2\pi \frac{kx}{N}} \tag{2.5}$$

To get back to the spatial domain the inverse DFT is as follow:

$$f(a) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{i2\pi \frac{ka}{N}} \tag{2.6}$$

The exponential terms are the basis function corresponding to each point in the Fourier domain representation $F(k)$. The value of each point in $F(k)$ is calculated by multiplying each point in the spatial domain with the corresponding Fourier basis and then summing them all up. In a more formal fashion, DFT expresses the spatial function $f(x)$ as a linear combination of orthogonal oscillating basis functions $e^{-i2\pi \frac{kx}{N}}$, indexed by their rate of oscillation (frequency). Dealing with the images, we need to extend the 1D-DFT to 2D-DFT. Given an $N \times N$ square image $f(i, j)$, we can extend the definition of DFT and iDFT to two dimensions as follow:

$$F(k, l) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) e^{-i2\pi (\frac{ki}{N} + \frac{lj}{N})} \tag{2.7}$$

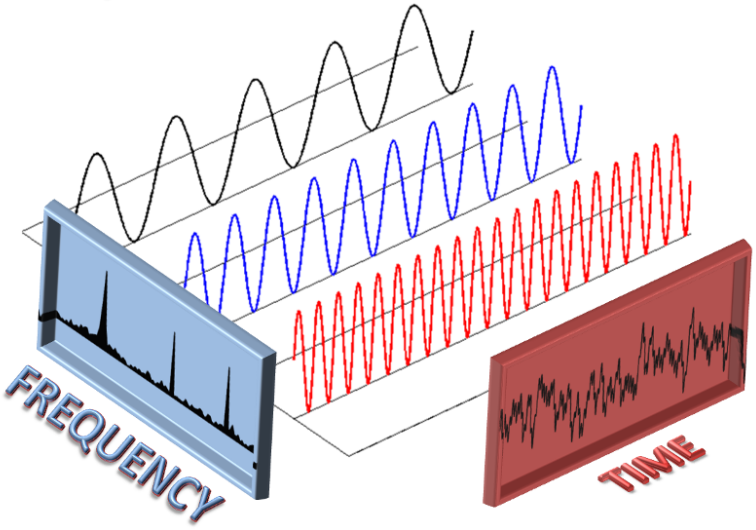
$$f(a, b) = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k, l) e^{i2\pi (\frac{ka}{N} + \frac{lb}{N})} \tag{2.8}$$

Note that the regularization constant $\frac{1}{N^2}$ only appears in the inverse function. The N is usually a power of 2, and when the original function does not satisfy the size, it is usually zero-padded to get to the desired size. DFT output is a complex function, and they can be shown with two different images, as *real* and *imaginary* or *magnitude* and *phase*. Neither

part can be ignored as both of them contain information about the original image.

The complexity of the DFT computation is $O(N^2)$; however, with a computational trick, it can be reduced to $O(N \log_2 N)$, the algorithms calculating the DFT with reduced complexity are known as Fast Fourier Transform (FFT). We use the terms DFT and FFT interchangeably in this document as the output to them is the same, only the procedure to get the output is different.

Figure 2.4: Demonstration of decomposing a signal into different cosine elements



Chapter 3

Literature Review

Since [1] introduced GANs in 2014, there has been a considerable body of work on improving and stabilizing this category of neural networks. [2] proposed a set of constraints, making GAN training with convolutional layers more stable. However, they could not overcome the mode collapse problem in training. They also showed the vector arithmetic properties of the latent input space of the generator similar to the known relations in the word embedding. Their other contribution was showing that adversarial network pairs in GANs can learn meaningful representation from objects to scenes. [51], [52] brought up a new distance to measure the difference between original data and generated data distributions, leading to the most noteworthy GAN success due to robustness to architectural choices. They were the first to address the mode collapse problem significantly. It is also worthy of mentioning that their new distance (estimated EM distance) provided meaningful learning curves for debugging and hyperparameter tuning. **sngan** applied spectral normalization over the weight

matrix, which not only helped the stability of the discriminator network but also, due to its significant computational benefits over the previous methods, made generating high-quality class conditional ImageNet [53] scale training possible. **sagan** used self-attention in both generator and discriminator, leading to a significant increase to the inception score of the GANs as well as delivering high-quality unconditional full-ImageNet samples. Their method involved a self-attention procedure allowing both generator and discriminator to have access to non-local regions of images when trying to detect or generate images. [54] continued the efforts to scale up the dataset with a new variant of orthogonal regularization and truncation trick. Their model allows increasing the batch size and model width, which not only allows for a significant scale-up but also increases the performance in the Inception Score. [55], [56] modified the discriminator by adding an extra path for the encoded generated images to decide not only based on the data space but also by another representation of the latent space. They introduced a novel representation learning approach to train GANs, which was one of our motivations in the current work. StyleGAN based GANs [3], [4] at the time of writing this document is holding the state-of-the-art in the literature. Their main contribution was the new architecture for the generator where it does not feed noise z directly to the generator. instead, it passes through an MLP to get a style vector w and feed in the generator architecture at different layers.

Although GANs have succeeded in delivering high-quality images almost impossible for human eyes to detect, recent studies [19], [20], [50], [57] showed that it is not an arduous task to detect GAN-generated images. In the core of the forensic analysis of [20], [50], [57], lies the

discrepancies between the spectrum of the generated images vs. the real ones. To further relate the frequency notion to the current state of neural network training in the literature [58] found evidence of a spectral bias on neural networks, following this idea [59] came up with band-limited training of the CNNs. Another frequency evidence lies in [60] which provided theoretical proof on having band-limited model as a condition for perfect learning in neural networks.

Learning in the frequency domain is not well-investigated in the literature. However, as we explained above, the clues are further pushing us to take the frequency domain More seriously. currently, the models are using fully connected [61] or CNN-based architectures[62] in the input. However, the recent CVPR paper [63] proposed a model based on DCT transformation, which aims to retain more original picture information through DCT transformation and reduce the communication bandwidth between CPU and GPU. In this work, we use geometric deep learning [45] to introduce a new model designed to perform specifically on the frequency domain of the image. We will explain the related literature review on geometric deep learning in more detail in the chapter 4.

Chapter 4

Geometric Deep Learning

In this chapter, we give a brief overview of geometric deep learning [45] and how it brings together a large class of neural networks working on different data types such as unstructured sets, grids, graphs, and manifolds to interpret under unified principles. Then we describe symmetries in the image data form and how geometric deep learning uses them to add inductive bias to the neural network framework, reducing the search space in the hypothesis class for more suitable functions. In section 5.4.2 we use the idea of equivariant functions to devise a suitable deep neural network architecture for the frequency domain. Our proposed architecture searches the frequency while keeps the essential property of shift invariancy of the image in mind. To the best of the author's knowledge, a systematic architectural design for the frequency domain has never been done in the literature. We consider it one of our main contributions.

4.1 Inductive bias

In machine learning theory, supervised learning is usually formalized by assuming that labels y are generated by an unknown function f, such that $y_i = f(x_i)$. Also, let us assume that the data points and labels come from an i.i.d sampling of the underlying data. $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where N here is the number of observations. The learning objective here is to estimate the function f as best as possible from a set of parameterized functions we have, known as a hypothesis class, $\mathcal{H} = \{f_{\theta \in \Theta}\}$. Neural network architecture can be seen as an instance of such parameterized hypothesis classes with parameters $\theta \in \Theta$ as the networks' weights. The ideal method to evaluate the chosen function from the hypothesis class is the expected loss defined as follow:

$$\mathcal{L}_P(f^*) := \mathbb{E}_P\{L(\tilde{f}(x), f(x)\}$$

$\tilde{f} \in \mathcal{H}$ is the chosen function, and P is the underlying distribution of the data. $L(.,.)$ is the suitable loss for our task; some examples can be square loss and binary cross-entropy. Thus, to have successful learning, one needs to consider a notion of regularity in choosing functions $\tilde{f}$. This idea of regularization is known as the inductive bias, which can be applied via various methods, explained further in this section.

To have a successful learning scheme, the other important and even more prominent property is that the hypothesis class should be dense enough to have the capacity to approximate functions over current large-scale, high-quality datasets. However, in the case of neural networks, it has never been a problem. Even a simple two-layer perceptron, $f(\mathbf{x}) = \mathbf{c}^\intercal \sigma(\mathbf{A}\mathbf{x} + \mathbf{b})$ shown to be dense in the space of continuous functions on $\mathbb{R}^d$.

To formalize the regularity notion into the learning scheme, we can get help from complexity measure $c : \mathcal{H} \to \mathbb{R}_+$, now the problem can be seen as:

$$\tilde{f} \in \arg\min_{g \in \mathcal{H}} c(g) \quad \text{s.t.} \quad g(x_i) = f(x_i) \quad \text{for} \quad i = 1, ..., N$$

The realizability assumption is considered for this formulation for convenience. In other words, now we are looking for the least complex (more regular) function among our hypothesis class. For the complexity measure, different norms of the working space are a well-known option used in learning approaches such as logistic regression or SVMs. In neural networks, network weights are a proper choice for complexity measurements. i.e. $c(f_\theta) = c(\theta)$. While explicitly applying regularities to the empirical loss is a common, well-defined approach known as "Structural Risk Minimization," the Implicit approaches are getting more and more attention of researchers [64], [65].

As the dimension of the space goes higher, finding suitable regular functions will get more challenging as well. In the case of the long-established regularity notion, 1-Lipschitz functions. The number of observations needed for the desired loss will grow exponentially. The good news is that real-world applications such as computer vision tend to work with functions that inherently possess some spatial structures in their physical domains. The geometric deep learning objective is to exploit structures as a notion of regularity further to reduce the search space for the most proper function.

4.2 Symmetries as Regularities

Symmetries on a domain or a signal are transformations that leave certain properties of the domain of the signal, or the signal itself, unchanged. Real-world applications are full of these operations. In computer vision's object detection tasks, shift or rotation are examples of transformations that leave the type of object invariant. If there is an image of a rhinoceros, the shifted or rotated rhinoceros should still be classified as a rhinoceros.

The symmetries of the domain Ω, and signal $\mathcal{X}(\Omega)$ operating on it present a constraint on the function f operating on those signals. It turns out that these symmetries are powerful prior to consider as inductive bias. They will reduce the search space $\mathcal{H}(\mathcal{X}(\Omega))$ to only the functions satisfying the invariancy. Here we explain the two groups of functions we use their symmetric properties in this thesis, *invariants* and *equivariants*.

Invariant functions: A function $f : \mathcal{X}(\Omega) \to \mathcal{Y}$ is invariant with respect to transformation $\mathfrak{g}$ if $f(\rho(\mathfrak{g})x) = f(x)$ and $x \in \mathcal{X}(\Omega)$, in other words, if its output is unchanged after the transformation.

Assuming the transformation $\mathfrak{g}$ is linear i.e $\mathfrak{g}.(\alpha x + \beta x') = \alpha(\mathfrak{g}.x) + \beta(\mathfrak{g}.x')$, $\rho(g)$ is the matrix equivalent of the the transformation $\mathfrak{g}$. The function for the rhinoceros classification example is categorized under shift invariant functions, which is very common in pattern recognition literature. Classic perceptron networks hypothesis' class consists of functions not having the shift invariant property. It is making them an unseemly choice for vision tasks where the object's position is not relevant in the final result. However, as we explain later in

section 4.3, applying a geometric prior for the shift would result in an architecture known as Convolutional Neural Networks (CNNs) (for further reading on CNNs please refer to section 2.1). That being said, shift geometric prior in the CNNs is not shift invariancy. In fact, it is shift equivariancy. That is to say, a particular amount of shift in the input would result in the same amount of shift in the output. Formally we can define an equivariant function as follow:

Equivariant functions: A function $f : \mathcal{X}(\Omega) \to \mathcal{Y}$ is equivariant with respect to transformation $\mathfrak{g}$ if $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$ and $x \in \mathcal{X}(\Omega)$. In other words, If the output react in the exact same way input reacted to the transformation.

A computer vision example of such tasks is image segmentation. If the segmented input moved, we expect the output to shift the same amount. Another noteworthy property of the invariant and equivariant function arises in their combinations. If $f_1(x)$ and $f_2(x)$ are equivariant and $g_1(x)$ and $g_2(x)$ are invariant, then by definition, $f_1 \circ f_2(x)$, $g_1 \circ g_2(x)$ and $f_1 \circ g_1(x)$ are equivariant, invariant and equivariant respectively. We also know the following properties for symmetric transformation, by the axioms of symmetric transformations.

Associativity: For all the functions belong to the same symmetry group $f_1 \circ f_2(.) = f_2 \circ f_1(.)$. Although invariants and equivariant are not from the same symmetry group, by definition they are also have this property together.

Inverse: for every function belong to the same symmetry group there is a unique inverse in the same symmetry group, e.g. $g_1^{-1} \circ g_1(x) = x$

Geometric deep learning provides a blueprint for constructing specific architectures satisfying

the explained conditions. For the specific case of equivariant functions. the blueprint's building blocks are as follow:

Linear equivariant layer B**:** $\mathcal{X}(\Omega) \to \mathcal{X}(\Omega)$

Non-linearity σ**:** applied element-wise as $(\sigma(x))(u) = \sigma(x(u))$.

Local pooling (coarsening) P**:** $\mathcal{X}(\Omega) \to \mathcal{X}(\Omega)$

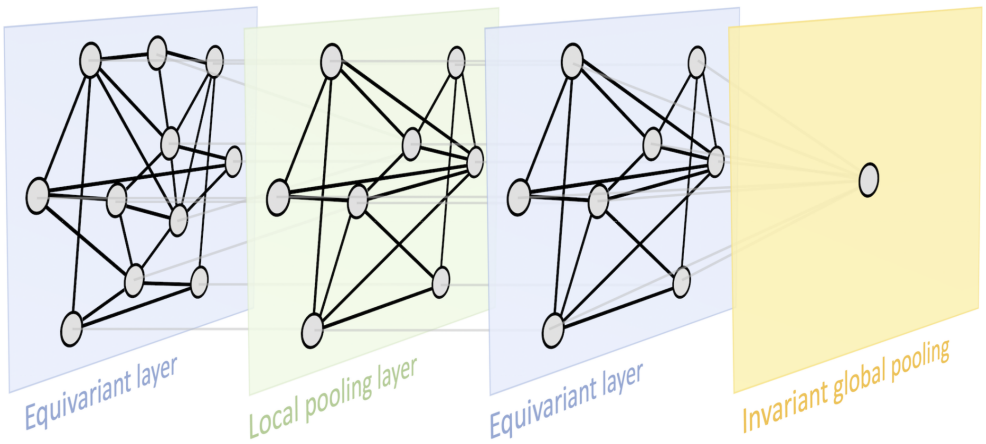
Invariant global pooling layer A**:** $\mathcal{X}(\Omega) \to \mathcal{Y}$

By using the mentioned building blocks in cascade, we can construct the equivariant function f as follow:

$$f = A \circ \sigma_J \circ B_J \circ P_{J-1} \circ ... \circ P_1 \circ \sigma_1 \circ B_1$$

The building blocks are designed such that the output of each block matches the input space of the next layer.

Figure 4.1: The blueprint for designing architectures using geometric priors (image credit for: [45])



4.3 Designing CNNs with GDL:

This section describes how we can apply the blueprint to a homogeneous grid space, where images are defined as signals over this space. Understanding this design is essential for the cause of this thesis since designing the Fourier layer for the discriminator of the GAN uses the same method with a similar approach.

Let's assume that we have a one dimensional grid for simplicity. in this grid each element $\mathbf{x}_u$ has a right and a left neighbor $\mathbf{x}_{u-1}, \mathbf{x}_{u+1}$. Using the blueprint we can design the equivariant function $\mathbf{F}$ with a local aggregation function ϕ operating over a grid element and its neighbors $\phi(\mathbf{x}_{u-1}, \mathbf{x}_u, \mathbf{x}_{u+1})$. Choosing $\phi(\mathbf{x}_{u-1}, \mathbf{x}_u, \mathbf{x}_{u+1}) = \theta_{-1}\mathbf{x}_{u-1} + \theta_0\mathbf{x}_u + \theta_1\mathbf{x}_{u+1}$ yields the function $\mathbf{F}$ as the matrix product:

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} \theta_0 & \theta_1 & & & \theta_{-1} \\ \theta_{-1} & \theta_0 & \theta_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \theta_{-1} & \theta_0 & \theta_1 \\ \theta_1 & & & \theta_{-1} & \theta_0 \end{bmatrix} \begin{bmatrix} -- & \mathbf{x}_0 & -- \\ -- & \mathbf{x}_1 & -- \\ & \vdots & \\ -- & \mathbf{x}_{n-1} & -- \\ -- & \mathbf{x}_n & -- \end{bmatrix}$$

In the machine learning, using this form of diagonal matrix is known as weight sharing. This matrix in an example of a circulant matrix, which is essentially a vector $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{n-1})$ append to itself after circular shifts. $\mathbf{C}(\boldsymbol{\theta}) = (\theta_{u-v \bmod n})$. product of a vector with a circulant matrix is equivalent with cyclic discrete convolution.

$$\mathbf{C}(\boldsymbol{\theta})\mathbf{x} = (\mathbf{x} * \boldsymbol{\theta})_u = \sum_{v=0}^{n-1} x_{v \bmod n}\, \theta_{u-v \bmod n}$$

In machine learning and signal processing, the above $\boldsymbol{\theta}$ is known as the filter, and its weights/values are subject to the learning process. Circulant matrices have commutativity property, which means their product is commutative, and it is also circulant. More mathematically $\mathbf{C}(\boldsymbol{\theta})\mathbf{C}(\boldsymbol{\eta}) = \mathbf{C}(\boldsymbol{\eta})\mathbf{C}(\boldsymbol{\theta})$ If we choose $\boldsymbol{\theta} = (0, 1, 0, \ldots, 0)^\intercal$ the resulted circulant matrix $\mathbf{S}$, shift vectors one position right (circularly) and we call it the shift matrix. From the properties of the convolution, and circulant matrices, we get the desired shift equivariance:

$$\mathbf{S}\mathbf{C}(\boldsymbol{\theta})\mathbf{x} = \mathbf{C}(\boldsymbol{\theta})\mathbf{S}\mathbf{x}$$

An interesting fact is that the other way around is also true. A matrix is circulant if it commutes with the shift. In other words, convolutions are the only linear operators keeping the equivariancy.

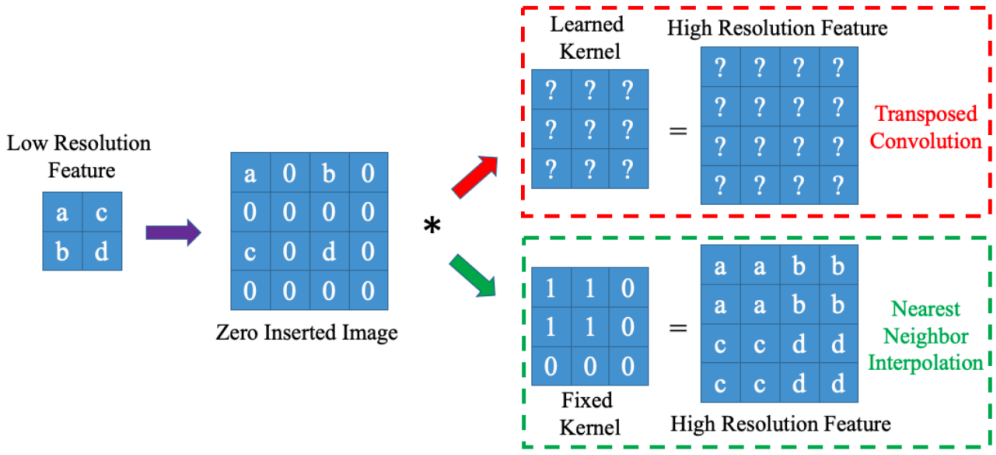
Chapter 5

Methodology

5.1 Frequency in the Neural Networks

Recent state-of-the-art GAN architectures used for vision tasks utilized the convolutional neural networks (CNN) structure for their generator and discriminator modules [3], [4], [54]. Regardless of the network architecture and the loss function difference, they all need to map from a low dimensional latent space to a high dimensional high-resolution image space. To this end different GANs used different upsampling approaches like bi-linear [3], [4], [54], transposed convolution [2], [66], [67] or nearest neighbor [68], [69]. In the simple case, when the up-sampler scales the image by a factor of $m = 2$; it inserts a zero between all the pixels in each row/column. Then it applies the convolution to interpolates the inserted zeros with suitable values. A simple formulation of the last two is shown in Figure 5.1. [20] first analyzed the frequency consequences of the GAN's up-sampling modules. Properties of the DFT

Figure 5.1: transposed convolution and nearest neighbour up-sampling (image credit for [20])



(Discrete Fourier Transform) reveal that zero-insertion in an image will affect the spectrum in a way that looks like multiple copies of the original spectrum attached together. The proof for the one-dimensional case is as follows, and it is easily extendable to the two-dimensional case. Given $x(n), n = 0, ..., N_1$ as the input signal to the up-sampler and $X(k), k = 0, ..., N-1$ as its DFT. By inserting the zeros we'll get $x'(n), n = 0, ..., 2N-1$ where $x(2n) = x(n)$ and $x(2n+1) = 0$ for $n = 0, ..., N-1$. Now the DFT of the $x'(n)$ will be $X'(k)$. For $k < N$,

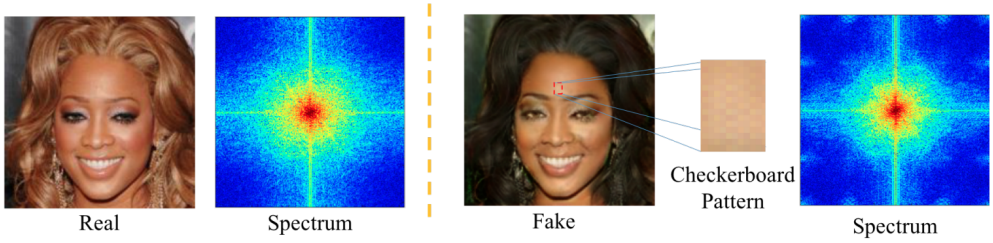
$$X'(k) = \sum_{n=0}^{2N-1} x'(n) exp(\frac{-i2\pi}{2N} kn) = \sum_{n=0}^{N-1} x(n) exp(\frac{-i2\pi}{2N} k(2n)) = X(k) \quad (5.1)$$

for $k \geq N$, let $k' = k - N$, thus $k' = 0, ..., N-1$ then,

$$X'(k) = \sum_{n=0}^{N-1} x(n) exp(\frac{-i2\pi}{2N} (k'+N)(2n)) = \sum_{n=0}^{N-1} (x(n) exp(\frac{-i2\pi}{N} nk' - i2n\pi)) = X(k') \quad (5.2)$$

The above shows that there will be two copies of the spectrum of the $X(k)$ in $X'(k)$ first at $[0, ..., N-1]$ and second at $[N, ..., 2N-1]$. This transformation introduces high-frequency components to the spectrum. [70] discussed that these high-frequency components play a significant role in a checkerboard artifact phenomenon seen in the outputs of generative convolutional networks. Figure 5.2 demonstrate the high-frequency components in the spectrum as well as the checkerboard artifacts. In the next step of the up-sampling, when the convolution occurs, we have neither a guarantee for the learned convolution kernel to be a suitable low-pass filter nor a sampling strategy to avoid such high-frequency artifact appear in the subsequent layer output. Thus, it can not fade out the newly introduced high-frequency components.

Figure 5.2: Left: spectrum of the real image; Right: spectrum of the generated image, the checkerboard artifacts are zoomed in, the extra high frequency components are visible in spectrum of the fake image. (image credit for [20])



5.2 DFT and Principles of Symmetry

Maybe the most famous property of the Fourier domain in signal processing arises from the convolution theorem. That is to say, the convolution operation in the spatial domain is

equivalent to element-wise multiplication in the Fourier domain. It is important since the interactions of linear time invariant (LTI) signals and systems can be described with the convolution operation. This property is derivable with properties of symmetry, and circulant matrices explained in section 4.3. To that end, we need to remember that diagonalizable matrices share common eigenvectors (with different eigenvalues) iff they mutually commute. Since circulant matrices satisfy these conditions, we can calculate the eigenvectors of one of them. Shift matrix $\mathbf{S}$ is a convenient choice here, whose eigenvectors luckily happens to be the DFT basis:

$$\boldsymbol{\phi}_k = \frac{1}{\sqrt{N}} \left(1, e^{\frac{2\pi i k}{N}}, e^{\frac{4\pi i k}{N}}, \ldots, e^{\frac{2\pi i (n-1) k}{N}}\right)^\intercal \quad , \qquad k = 0, 1, \ldots, n-1$$

Arranging the $\boldsymbol{\phi}$ vectors into a $n \times n$ matrix gives us $\boldsymbol{\Phi} = (\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{n-1})$. Matrix multiplication with $\boldsymbol{\Phi}$ yield the inverse DFT and with its complex conjugate $\boldsymbol{\Phi}^*$ the DFT. All circulant matrices share this eigenvectors, so the Fourier transform of the filter is the eigenvector matrix $\mathcal{F}(\boldsymbol{\theta}) = \boldsymbol{\Phi}^* \boldsymbol{\theta}$ for the circulant matrix $\boldsymbol{C}(\boldsymbol{\theta})$. Thus, we can write $\boldsymbol{C}(\boldsymbol{\theta})$ in diagonalized form to obtain the convolution theorem:

$$\boldsymbol{C}(\boldsymbol{\theta})\mathbf{x} = \boldsymbol{\Phi} \begin{bmatrix} \mathcal{F}(\theta_0) & & \\ & \ddots & \\ & & \mathcal{F}(\theta_{n-1}) \end{bmatrix} \boldsymbol{\Phi}^* \mathbf{x} = \boldsymbol{\Phi} \left(\mathcal{F}(\boldsymbol{\theta}) \odot \mathcal{F}(\mathbf{x})\right)$$

where $\odot$ is element-wise multiplication. we use the above mentioned characteristic of circulant matrices to provide a proof of work for our frequency domain architecture in the next section.

5.3 Problem Definition

[20] shed light on the dissimilarities between authentic image spectrum and the images generated by convolutional networks. These disparities mostly show themselves in the forms of high-frequency patches in the generated images spectrum, discussed by [70] as responsible for the checkerboard artifacts showed in them. Knowing that Fourier is a bijective mapping, any apparent difference in the spectrum demonstrates a flaw in learning the original distribution, which is the main objective of training GANs as a generative model. In this work, we are developing a new architecture for the discriminator to address the spectral difference where the discriminator gets the images in the Fourier and the spatial domains. We then experimentally demonstrate that our model can enhance the learned distribution, emphasizing the importance of the spectrum of images on the learning process.

5.4 Methodology

The most straightforward way for solving dissimilarities in the spectrum is to add frequency information to the discriminator. [5], [55], [56] have already proved that feeding information to the discriminator can boost the performance of the GANs for other applications. In this case, the network would better detect high-frequency components' differences in the generated images by providing the discriminator with frequency information. Thus, it pushes the generator to produce images matching the actual image distribution's spatial and frequency representation. Equation (5.3) shows the new loss function for the GAN incorporating the

Fourier transform $\mathcal{F}(.)$ of the images directly in the loss.

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}) =$$

$$\mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}(\mathbf{x})}[\log \mathcal{D}(\mathbf{x}, \mathcal{F}(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{z}}(\mathbf{z})}[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z}), \mathcal{F}(\mathcal{G}(\mathbf{z}))))]. \quad (5.3)$$

For preprocessing the Fourier information [50] discusses the following 1D representation of Fourier transform via azimuthal integration over radial frequencies, practical enough to point out the dissimilarities of the spectrum and spatial domain. Assuming that images are $M \times M$.

$$AI(\omega_k) = \int_0^{2\pi} \left\| \mathcal{F}(\omega_k \cdot \cos(\phi), \omega_k \cdot \sin(\phi)) \right\|^2 d\phi \quad (5.4)$$

$$\text{for} \qquad k = 0, ..., \frac{M}{2} - 1$$

Equation (5.4) can be seen as a coordinate conversion from Cartesian to polar and then getting the mean intensity over the radial distance. It is also used in [71] as the input to the frequency module. Despite its popularity due to more straightforward computation in 1D, performing average operation will cause loss of information in FFT amplitudes and completely disregard the phase information, which proved to be an essential part of the spectrum [72], [73]. In this work, we use both amplitude and phase of the image as the input to our frequency module to fully exploit the spectrum information.

At a high level, our work consists of two sections. First, we introduce a high-level architecture with unary losses (Figure 5.3) to incorporate frequency information into the GAN architecture in a guided manner. It will result in a discriminator more sensitive to

frequency discrepancies, which pushes the generator to produce images more realistic in the frequency domain. Second, we devise a Frequency module to use in our architecture. This module's task is to get the frequency spectrum and output a realness score. The frequency module uses a base layer we named *EV-Freq* which is designed based on the blueprint provided by geometric deep learning.

5.4.1 The High-Level Architecture

Importing the spectrum to the network has shown degradation in the performance of the model [71]. This phenomenon is a consequence of losing too much image information in the spectrum, hence, not having enough gradient for the adversarial training. In order to overcome this problem, we adopt the three unary terms proposed by [56] to guide the optimization.

Figure 5.3 shows a high-level view of our model's architecture. The generator and discriminator

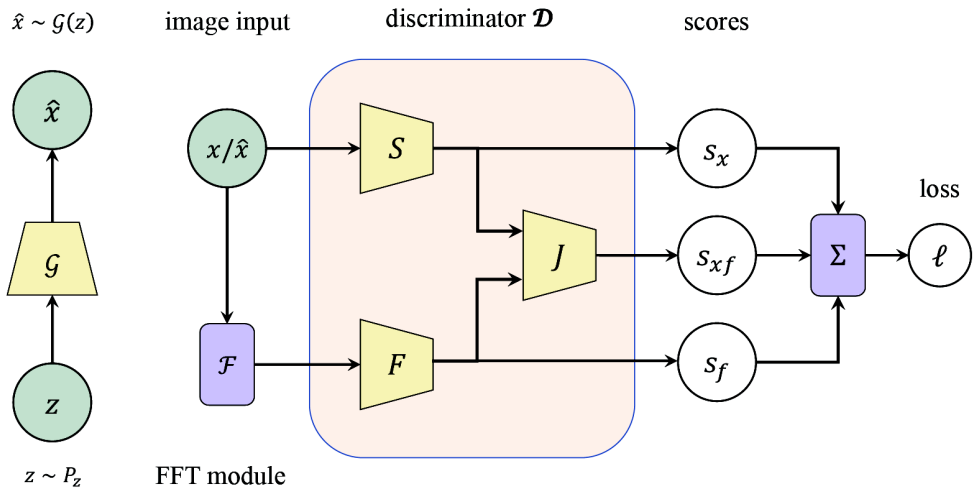
loss in our model can be described as follow:

$$
\begin{aligned}
s_{\mathbf{x}}(\mathbf{x}) &= \theta_{\mathbf{x}}^{\intercal} S_{\Theta}(\mathbf{x}) \\
s_{f}(\mathbf{x}) &= \theta_{f}^{\intercal} F_{\Theta}(\mathcal{F}(\mathbf{x})) \\
s_{\mathbf{x}f}(\mathbf{x}) &= \theta_{\mathbf{x}f}^{\intercal} J_{\Theta}(S_{\Theta}(\mathbf{x}), F_{\Theta}(\mathcal{F}(\mathbf{x}))) \\
\ell_{\mathcal{G}}(\mathbf{x}, y) &= y(s_{\mathbf{x}}(\mathbf{x}) + s_{f}(\mathbf{x}) + s_{\mathbf{x}f}(\mathbf{x})) \qquad y \in \{-1, +1\} \\
\mathcal{L}_{\mathcal{G}}(P_{\mathbf{x}}, P_{\mathbf{z}}) &= \mathbb{E}_{\mathbf{x}\sim P_{\mathbf{x}}}[\ell_{\mathcal{G}}(\mathbf{x}, +1)] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}, \hat{\mathbf{x}}\sim \mathcal{G}_{\Phi}(\mathbf{z})}[\ell_{\mathcal{G}}(\hat{\mathbf{x}}, -1)] \\
\ell_{\mathcal{D}}(\mathbf{x}, y) &= \ell(y s_{\mathbf{x}}(\mathbf{x})) + \ell(y s_{f}(\mathbf{x})) + \ell(y s_{\mathbf{x}f}(\mathbf{x})) \qquad y \in \{-1, +1\} \\
\mathcal{L}_{\mathcal{D}}(P_{\mathbf{x}}, P_{\mathbf{z}}) &= \mathbb{E}_{\mathbf{x}\sim P_{\mathbf{x}}}[\ell_{\mathcal{D}}(\mathbf{x}, +1)] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}, \hat{\mathbf{x}}\sim \mathcal{G}_{\Phi}(\mathbf{z})}[\ell_{\mathcal{D}}(\hat{\mathbf{x}}, -1)]
\end{aligned}
$$

ℓ in the sixth line refers to the loss we chose to apply on the instances. [56] chose hinge loss for this, and we intend to do the same here. The discriminator $\mathcal{D}$ has three modules. S takes the images in the data $\mathbf{x}$ only in the spatial domain while F's input is the data in the frequency domain $\mathcal{F}(\mathbf{x})$. With this model, we get two separate scores for the realness of the image's spectrum and images in the spatial domain. Each module calculates a realness scalar score based on its input. The scalar scores respectively are $s_{\mathbf{x}}$ and s_f calculated with their parameters $\theta_{\mathbf{x}}$ and θ_f. The desirable output should get a high score both in frequency and spatial domains. Thus, the last module J gets the output of the other two modules, $s_{\mathbf{x}}$ and s_f, as its inputs, and the output is the joint score $s_{\mathbf{x}f}$ which defines how real the image is concerning both discussed domains. The generator's $\mathcal{G}$ parameters Φ and discriminator's $\mathcal{D}$ are optimized to minimize the losses $\mathcal{L}_{\mathcal{G}}$ and $\mathcal{L}_{\mathcal{D}}$ respectively. The Discriminator loss $\mathcal{L}_{\mathcal{D}}$ trains the discriminator to identify the joint representations of its input coming from

real data, $(\mathbf{x}, \mathcal{F}(\mathbf{x}))$, or the generator, $(\mathcal{G}(\mathbf{z}), \mathcal{F}(\mathcal{G}(\mathbf{z})))$ with predicting, respectively, positive and negative values for them. The discriminator Loss consists of three losses coming from spatial, frequency, and joint modules. Each of them is responsible for 1/3 of the total loss. We recognize the equal weights for each module as a limitation for our work, and we planed to design an architecture with adaptive weights in our future works. The generator loss, $\mathcal{L}_{\mathcal{G}}$ optimize the generator to misguide the discriminator to predict incorrectly, stirring the generator to create images matching both representations (in spatial and frequency domain) of the real data.

Figure 5.3: Architecture for FreqGAN

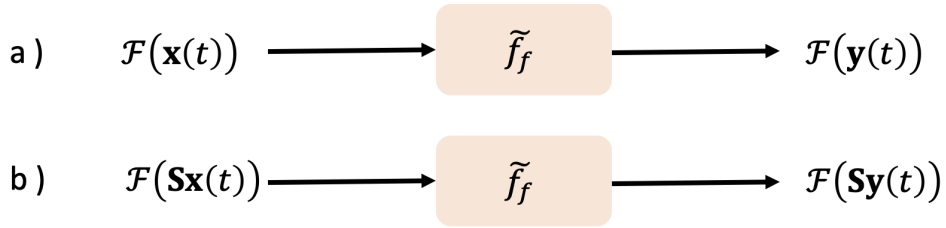


5.4.2 Frequency Module

Deep learning researchers developed various deep neural network architectures for different spaces and signals defined over them, such as sets, graphs, images, and text. However, there

is no particular design for the frequency spectrum. Current frequency-driven neural networks use fully connected or convolutional neural networks. The former results in a vast search space for optimization. Therefore, a slight chance to exploit the approximation function over frequency. The latter, without any modification, is designed for spatial feature extraction, which is by itself not a helpful feature in the spectrum data. Geometric deep learning establishes a framework for unifying some of the well-defined architectures. Moreover, it provides a blueprint for systematically design a new architecture for new domains and signals. In this section, by following the GDL blueprint, we propose a new architecture EV-Freq for frequency-domain signals. In section 4.3, shift equivariancy of the image's spatial domain provided the inductive bias needed for designing CNNs. We use the same idea here. Our functions in the frequency domain should not disturb the equivariancy of their corresponding spatial images. Let's denote out input image with $\mathbf{x}(t)$, our chosen function operating over frequency domain from our designed hypothesis class as $\tilde{f}_f \in \mathcal{H}_f$, and also $\mathcal{F}\left(\mathbf{y}(t)\right) = \tilde{f}_f\left(\mathcal{F}\left(\mathbf{x}(t)\right)\right)$. In other words, $\mathbf{y}(t)$ is the Fourier inverse of the $\tilde{f}_f$ output. Figure 5.4-a shows the relationship.

Figure 5.4: a) our designed function operating over and image b) desired behaviour of our designed input over shifted input



As we discussed in section 5.2, we can make the Fourier transfer of a function by matrix multiplication with matrix $\mathbf{\Phi}^*$. We use this notation in the following proof whenever we perform a Fourier transform on a signal. Aimed at keeping the equivariance in the spatial domain $\tilde{f}_f$ should satisfy the following equation: (Figure 5.4-b)

$$\tilde{f}_f\left(\mathbf{\Phi}^*\mathbf{S}\mathbf{x}(t)\right) = \mathbf{\Phi}^*\mathbf{S}\mathbf{y}(t) \tag{5.5}$$

In other words, the exact shift on input should be applied to the spatial domain image corresponding to the output Fourier spectrum. From figure 5.4-a we also have:

$$\tilde{f}_f\left(\mathbf{\Phi}^*\mathbf{x}(t)\right) = \mathbf{\Phi}^*\mathbf{y}(t) \tag{5.6}$$

Fourier matrix $\mathbf{\Phi}^*$ is Unitary orthogonal hence, $\mathbf{\Phi}\mathbf{\Phi}^* = \mathbf{\Phi}^*\mathbf{\Phi} = \mathbf{I}$, Thus from equation (5.6) we get:

$$\mathbf{y}(t) = \mathbf{\Phi}\tilde{f}_f\left(\mathbf{\Phi}^*\mathbf{x}(t)\right) \tag{5.7}$$

By putting equation (5.7) in equation (5.5):

$$\tilde{f}_f\left(\mathbf{\Phi}^*\mathbf{S}\mathbf{x}(t)\right) = \mathbf{\Phi}^*\mathbf{S}\mathbf{\Phi}\tilde{f}_f\left(\mathbf{\Phi}^*\mathbf{x}(t)\right) \tag{5.8}$$

From properties of circulant matrices discussed in section 5.2, $\mathbf{\Phi}^*$ contains the eigenvectors of $\mathbf{S}$ in each row. Hence, $\mathbf{\Phi}^*\mathbf{S} = \mathbf{\Theta}\mathbf{\Phi}^*$ whera $\mathbf{\Theta}$ is a diagonal matrix, consisting of $\mathbf{S}$'s eigenvalues on its diagonal. Knowing that we can rewrite equation (5.8) as:

$$\tilde{f}_f\left(\mathbf{\Theta}\mathbf{\Phi}^*\mathbf{x}(t)\right) = \mathbf{\Theta}\mathbf{\Phi}^*\mathbf{\Phi}\tilde{f}_f\left(\mathbf{\Phi}^*\mathbf{x}(t)\right)$$

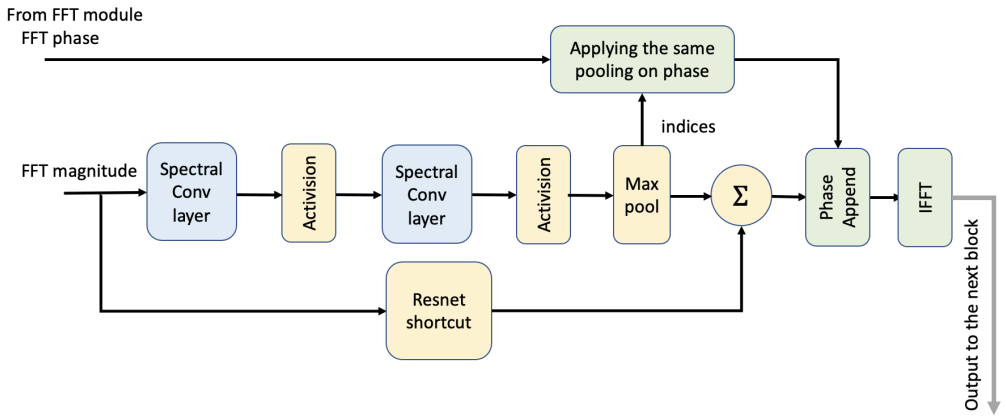
Finally we get to our desired constraint on the function $\tilde{f}_f$:

$$\tilde{f}_f\left(\mathbf{\Theta}\mathcal{F}\left(\mathbf{x}(t)\right)\right) = \mathbf{\Theta}\tilde{f}_f\left(\mathcal{F}\left(\mathbf{x}(t)\right)\right) \tag{5.9}$$

$\mathbf{S}$ is non-symmetric orthogonal, hence its eigenvalues are complex and roots of 1 with absolute value of 1 and they are located and the diagonal of $\mathbf{\Theta}$. Thus, if we define $\tilde{f}_f$ any function $\tilde{f}_f : \mathbb{R}^n \to \mathbb{R}^n$ that keeps the phase intact, the equation (5.9) will be satisfied.

Now that we found a set of functions operating on the spectrum, keeping the spatial equivarincy, our next step is to design an architecture satisfying the resulted constraint. To that end, we first perform the Fourier transform and separate amplitude and phase. While we keep the phase intact, we pass the amplitude from a residual block with spectrum normalization. The spectral normalization is an attempt to stabilize GAN training introduced by **sngan**. Performing max-pooling will change the dimension of the amplitude. We get the pooling indices and apply them to the phase to preserve the phase while sustaining its dimensions compatible with the amplitude. Since it is an element-wise operation over the phase matrix, it does not jeopardize the equivarincy. After all the computation on amplitude, we append the phase to get a complete spectrum representation. In the end, we perform a Fourier inverse to get back to the spatial domain and input to the next layer. Figure 5.5 shows the different elements of the architecture and their relationship together.

Figure 5.5: Architecture of one layer of the FFT module, based on resnet, containing two convolutional layers with spectral normalization, activation functions (ReLU in our case), maxpooling with phase implementation, phase append module, and Fourier inverse module



Chapter 6

Experiments

6.1 Experimental Setup

6.1.1 Systems and Baseline

The systems used for the experiments are provided by the Laboratory for Neural Computing for Machine Learning, the specifications of the systems are listed in the table 6.1.1

The experiments are all conducted with the python version 3.7, PyTorch version 1.7.1 utilized with CUDA 10.1, with torchvision version 0.8.2, and torchaudio version 0.7.2.

SNGAN **sngan** is the baseline chosen for the GAN experiments. It uses spectral normalization (normalization of the weight matrices by the largest singular value) in its convolutional layers. It is a fairly stable version of GANs, and it is used in other frequency-driven efforts for GAN improvements such as [71], so it makes our work comparable with others. The implementation is done based on *Mimicry* [74] which is a lightweight PyTorch library aimed towards the

System List			
System	**CPU**	**Memory**	**GPUs**
Speech 8	4-core CPU (Xeon W-2125)	32GB	1 × GTX 1080, 8GB memory
Text	6-core CPU (i7-5820K)	64GB	4 × TITAN X, 12GB memory
Video	6-core CPU (Xeon E5-1650)	64GB	3 × TITAN X, 12GB memory
Image	6-core CPU (i7-6800K)	128GB	4 × GTX 1080, 8GB memory

Table 6.1: List of systems used for the experiments

reproducibility of GAN research. It is introduced in the *CVPR* 2020 workshops to resolve (in their words): "(a) Standardized implementations of popular GANs that closely reproduce reported scores; (b) Baseline scores of GANs trained and evaluated under the same conditions; (c) A framework for researchers to focus on the implementation of GANs without rewriting most of GAN training boilerplate code, with support for multiple GAN evaluation metrics."

6.1.2 Datasets

We used two datasets in this work, CIFAR100 [75] and stl-10 [76]. The first one consists of 32×32 images in 100 classes developed by the University of Toronto. It has 600 images in each class, which makes it a total of 60000 images. There are 500 training and 100 testing images in each class. Since in the unsupervised problem of GAN, we do not need a separate testing class, we used all 600 images for the training purpose. This dataset provides us with images with different features due to its variety in classes. It is important for us since our objective

is to see the improvement of the GANs for the general problem of creating synthetic images, not as a class conditional problem. Some examples of the classes are animals, insects, fruits and vegetables, plants, household devices and furniture, people, plants, outdoor and indoor scenes with different lightings and vehicles. Stl-10 contains 10 classes of images: airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. in each 500 training and 800 testing images which we used both for training. The resolution of the original images is 96×96. However, we resized the images to 48×48 for memory-related and comparison reasons. The images in this dataset are acquired from rescaling of the ImageNet [53] dataset.

6.1.3 Optimization and GAN Hyperparameters Settings

We used Adam optimizer for both generator and discriminator with the setting: $(\beta_0, \beta_1) = (0.0, 0.9)$. The learning rate is set to 0.0002 with a linear decay throughout the training. The batch size is set to 64 and we adopt Xavier initialization for the all neural network weights. We update discriminator 5 times for each generator update to keep the training balance. The architecure for generator for each dataset is reported in the table 6.2 and 6.3, for the discriminator spatial path in table 6.4 and 6.5, for the discriminator frequency path in table 6.6 and 6.7 and finally for the last unitary module of the discriminator in tables 6.8 and 6.9. The FFT input to the frequency path consists of a double-sided FFT of the gray-scaled images. We chose to make our images monochrome since color frequencies possess less importance than brightness frequencies. We needed our module to focus on the brightness frequencies for this step. The addition of the color frequencies to the path is in our plans for future works.

Notations for reading the tables are: ResBlock: A residual block with a compatible shortcut whether it has down or up a sample or not, CONV: a convolutional layer with the written specifications, SNCONV: a convolutional layer with spectral normalization, K: kernel size, S: stride size, P: padding size, N: number of output channels, FC: fully connected layer with number of input and output neurons, Up: an upsampling procedure, PhaseAttach: procedure of attaching the phase which is pooled the same as amplitude in parallel, BN: batch normalization

Layer	**Input → Output Shape**
FC-(128,2048), Reshape	(128) → (256,4,4)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU, Up	(256,4,4) → (256,8,8)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU	(256,8,8) → (256,8,8)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU, Up	(256,8,8) → (256,16,16)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU	(256,16,16) → (256,16,16)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU, Up	(256,16,16) → (256,32,32)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU	(256,32,32) → (256,32,32)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU	(256,32,32) → (256,32,32)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU	(256,32,32) → (256,32,32)
BN, ReLU, CONV-(N3,K3,S1,P1), Tanh	(256,32,32) → (3,32,32)

Table 6.2: Architecture for CIFAR-100 generator $\mathcal{G}$

Layer	**Input → Output Shape**
FC-(128,18432), Reshape	(128)→(512,6,6)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU, Up	(512,6,6) → (256,12,12)
ResBlock: SNCONV-(N256,K3,S1,P1), BN, ReLU	(256,12,12) → (256,12,12)
ResBlock: SNCONV-(N128,K3,S1,P1), BN, ReLU, Up	(256,12,12) → (128,24,24)
ResBlock: SNCONV-(N128,K3,S1,P1), BN, ReLU	(128,24,24) → (128,24,24)
ResBlock: SNCONV-(N64,K3,S1,P1), BN, ReLU, Up	(128,24,24) → (64,48,48)
ResBlock: SNCONV-(N64,K3,S1,P1), BN, ReLU	(64,48,48) → (64,48,48)
ResBlock: SNCONV-(N64,K3,S1,P1), BN, ReLU	(64,48,48) → (64,48,48)
ResBlock: SNCONV-(N64,K3,S1,P1), BN, ReLU	(64,48,48) → (64,48,48)
BN, ReLU, CONV-(N3,K3,S1,P1), Tanh	(64,48,48) → (3,48,48)

Table 6.3: Architecture for stl-10 generator $\mathcal{G}$

Layer	**Input → Output Shape**
ResBlock: SNCONV-(N128,K3,S1,P1), ReLU	(3,32,32)→(128,32,32)
ResBlock: SNCONV-(N128,K3,S1,P1), ReLU, AvgPool-(K2,S2)	(128,32,32)→(128,16,16)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1)	(128,16,16)→(128,16,16)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1), AvgPool-(K2,S2)	(128,16,16)→(128,8,8)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1)	(128,8,8)→(128,8,8)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1)	(128,8,8)→(128,8,8)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1)	(128,8,8)→(128,8,8)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1)	(128,8,8)→(128,8,8)
ReLU, GlobalSumPool	(128,8,8) → (128)

Table 6.4: Architecture for CIFAR-100 Discriminator $\mathcal{D}$ spatial path

Layer	**Input → Output Shape**
ResBlock: SNCONV-(N64,K3,S1,P1), ReLU	(3,48,48)→(64,48,48)
ResBlock: SNCONV-(N64,K3,S1,P1), ReLU, AvgPool-(K2,S2)	(64,48,48)→(64,24,24)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1)	(64,24,24)→(128,24,24)
ReLU, ResBlock: SNCONV-(N128,K3,S1,P1), AvgPool-(K2,S2)	(128,24,24)→(128,12,12)
ReLU, ResBlock: SNCONV-(N256,K3,S1,P1)	(128,12,12)→(256,12,12)
ReLU, ResBlock: SNCONV-(N256,K3,S1,P1), AvgPool-(K2,S2)	(256,12,12)→(256,6,6)
ReLU, ResBlock: SNCONV-(N512,K3,S1,P1)	(256,6,6)→(512,6,6)
ReLU, ResBlock: SNCONV-(N512,K3,S1,P1), AvgPool-(K2,S2)	(512,6,6)→(512,3,3)
ReLU, ResBlock: SNCONV-(N1024,K3,S1,P1)	(512,3,3)→(1024,3,3)
ReLU, ResBlock: SNCONV-(N1024,K3,S1,P1)	(1024,3,3)→(1023,3,3)
ReLU, GlobalSumPool	(1024,3,3) → (1024)

Table 6.5: Architecture for stl-10 Discriminator $\mathcal{D}$ spatial path

Layer	**Input → Output Shape**
ResBlock: SNCONV-(N128,K3,S1,P1), ReLU	(1,32,32)→(128,32,32)
ResBlock: SNCONV-(N128,K3,S1,P1), ReLU, MaxPool-(K2,S2)	(128,32,32)→(128,16,16)
PhaseAttach, IFFT, GlobalSumPool	(128,16,16)→(128)

Table 6.6: Architecture for CIFAR-100 Discriminator $\mathcal{D}$ frequency path

Layer	**Input → Output Shape**
ResBlock: SNCONV-(N256,K3,S1,P1), ReLU	(1,48,48)→(256,48,48)
ResBlock: SNCONV-(N1024,K3,S1,P1), ReLU, MaxPool-(K2,S2)	(256,48,48)→(1024,24,24)
PhaseAttach, IFFT, GlobalSumPool	(1024,24,24)→(1024)

Table 6.7: Architecture for stl-10 Discriminator $\mathcal{D}$ frequency path

Layer	**Input → Output Shape**
FC-(128,1)	(128) *spatial* → (1) *s-score*
FC-(128,1)	(128) *frequency* → (1) *f-score*
Concat	(128) *spatial*, (128) *frequency* → (256)
FC-(256,1)	(256) → (1) *j-score*

Table 6.8: Architecture for CIFAR-100 Discriminator $\mathcal{D}$ unitary modules

Layer	**Input → Output Shape**
FC-(1024,1)	(1024) *spatial* → (1) *s-score*
FC-(1024,1)	(1024) *frequency* → (1) *f-score*
Concat	(1024) *spatial*, (1024) *frequency* → (2048)
FC-(2048,1)	(2048) → (1) *j-score*

Table 6.9: Architecture for stl-10 Discriminator $\mathcal{D}$ unitary modules

6.2 Stability of The Training

Before evaluating any result, we need to ensure that the model converges. Otherwise, the results can not serve as an indication of the superiority of our model. In GAN tasks, unlike conventional tasks, the objective in training is not improving variables such as decreasing a loss. We care about a gradual and continuous improvement in both agents involved in the process, meaning generator and discriminator. If any agents get too strong with respect to the other, the balance is disturbed, and the model will fail. This multi-agent scheme made GANs more susceptible to the instability of training. In fact, GANs are famous for being hard to train due to instability, and there have been several works to make them more stable. In this section, we provide empirical substantiations that our extension to the GAN framework will not jeopardize the stability of the baseline. We further discuss how each of the spatial and frequency modules acts separately convergence-wise and how they contribute to the convergence of the discriminator in total.

When networks are out of balance, we see a fast convergence in one of them, showing that the converged network has won the competition. It can happen when one network is inherently more potent. To address this kind of problem, we update the weaker network more frequently. In our cases, the baseline updates the discriminator 5 times for each generator update, and our extension uses the same ratio without instability. The other cause of failure in GANs is mode collapse. A mode collapse refers to a generator model that can generate one or a small subset of different outcomes or modes. It happens when the generator finds a way to produce plausible outcomes for the discriminator without imitating the distribution. Mode collapse is easily detectable by looking at the generated images since they come from a small set of images and are not diverse. In the learning curve, it shows itself as an oscillatory behavior in the generator loss (and, in some cases, discriminator loss).

Figures 6.1 and 6.2 show the Discriminator's total loss, sum of the discriminator's fake and real losses, and generator's loss together for CIFAR100 and stl10 datasets. The curves are in a balanced manner showing fair competition while converging to the same loss. There is no instability in the curves, whether a fast convergence or an oscillatory behavior. For a deeper look, each discriminator loss consists of 3 other losses coming from the spatial and frequency modules and one joint loss from the J module. Figures 6.3 and 6.4 show the derailed losses for the mentioned datasets. Frequency loss has a lower value compared to the spatial loss in CIFAR100 and greater in the stl10. It shows that the frequency domain is as useful as the spatial domain to detect the fake images for the discriminator. The joint loss trains to gives most of its attention to the more functional loss. The power of unitary modules emerges here.

While spatial and frequency losses equally contribute to the $\frac{1}{3}$ of the discriminator loss, the joint loss which is responsible for the last $\frac{1}{3}$ of the total loss, automatically learns to adopts its weights for better use of each domain.

Figures 6.5 and 6.6 show the raw scores used by discriminator for classification. The real scores are expected to be above zero and fake scores below zero. Although joint score mostly follows the spatial scores, frequency scores show fewer fluctuations and well above and below the 0 threshold, making it a valid feature for classification.

To recapitulate, frequency features showed to be a good representative for fake and real images in the discriminator and made the discriminator more powerful. The Generator also shows the capacity to be pushed along with the frequency extension in the discriminator without losing stability.

Figure 6.1: Discriminator's $\mathcal{D}$ total loss and Generator's $\mathcal{G}$ loss learning curve during the 100000 iterations for CIFAR100 dataset

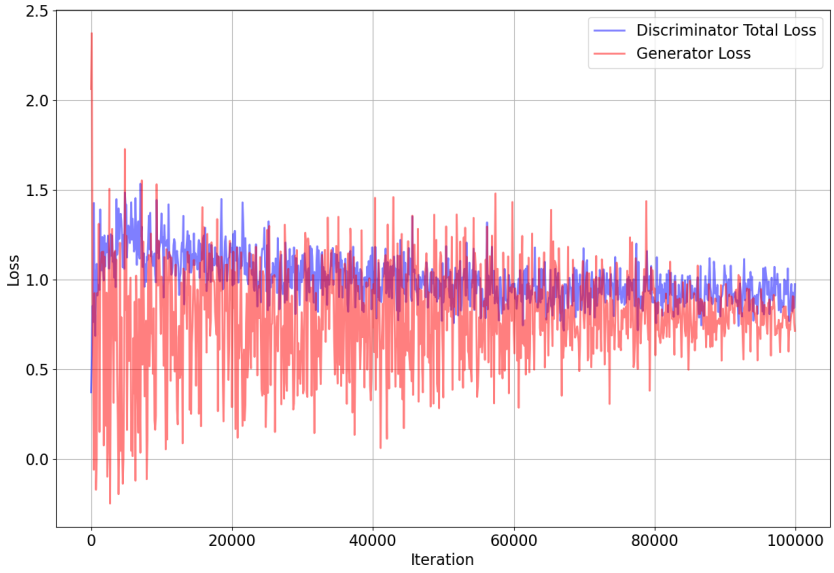


Figure 6.2: Discriminator's $\mathcal{D}$ total loss and Generator's $\mathcal{G}$ loss learning curve during the 100000 iterations for stl10 dataset

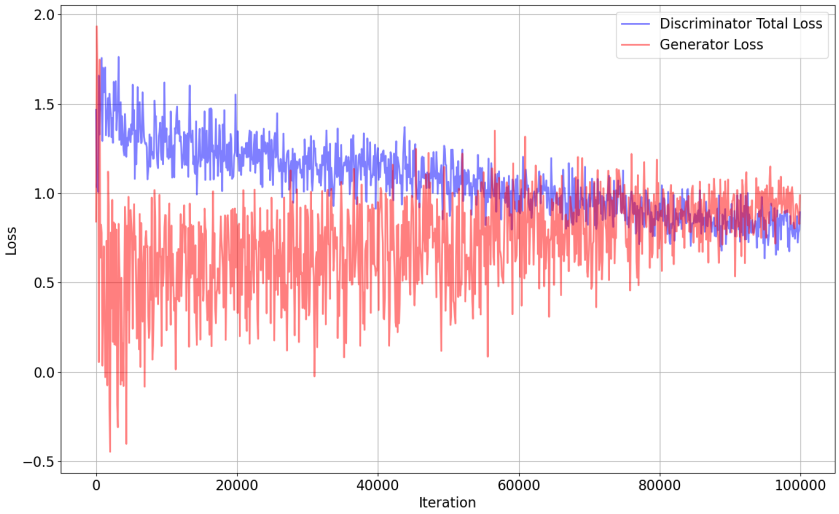


Figure 6.3: Detailed learning curves for loss of the 3 modules of the discriminator for CIFAR100 dataset.

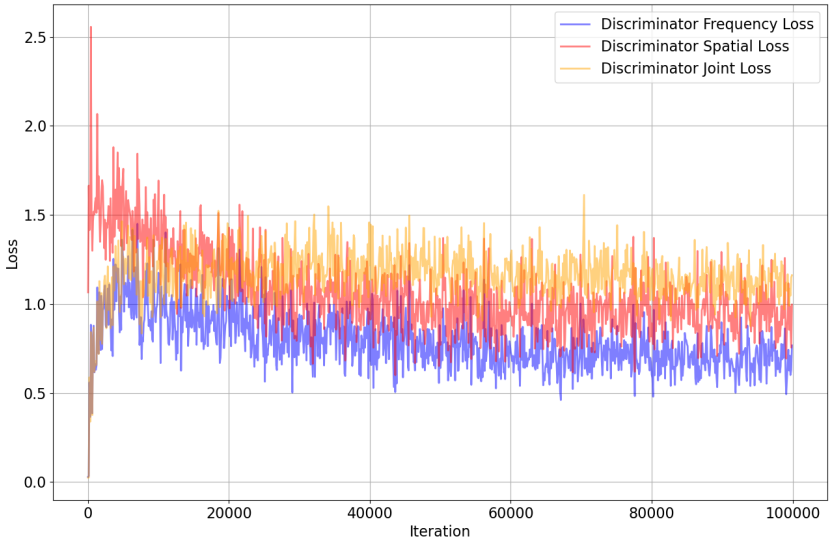


Figure 6.4: Detailed learning curves for loss of the 3 modules of the discriminator for stl10 dataset.

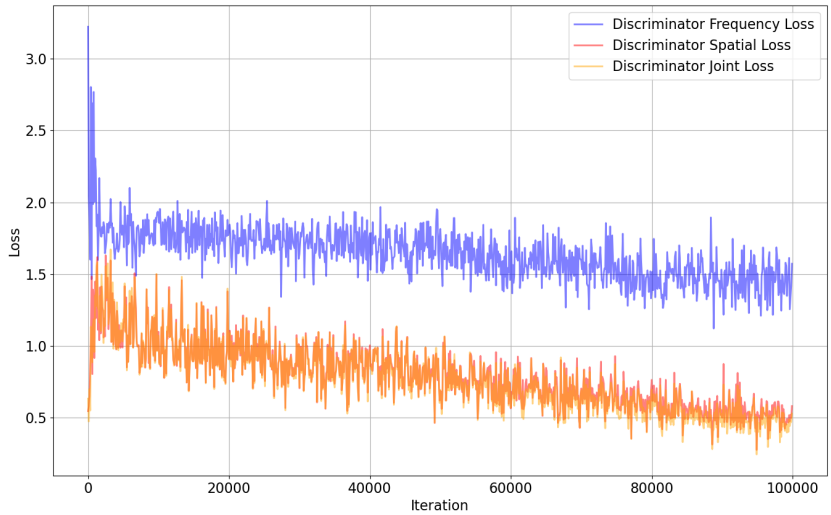


Figure 6.5: real and fake scores for each discriminator module during the training phase for CIFAR100 dataset.

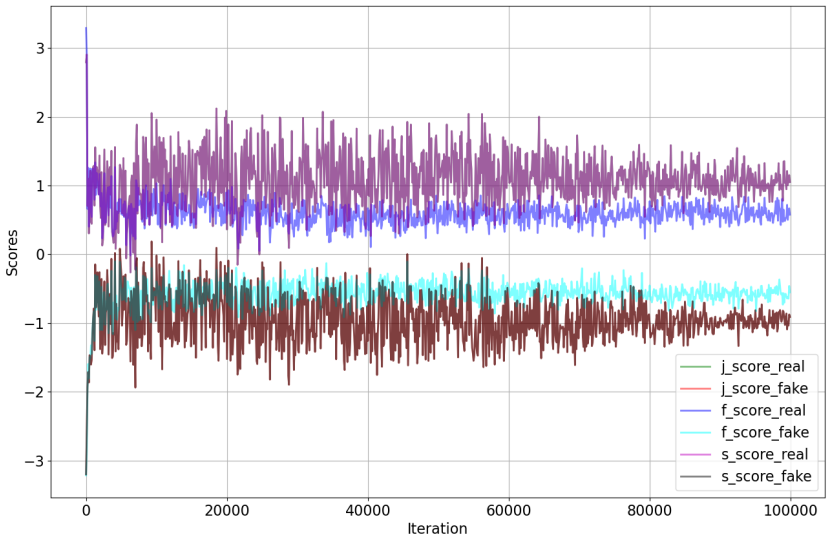
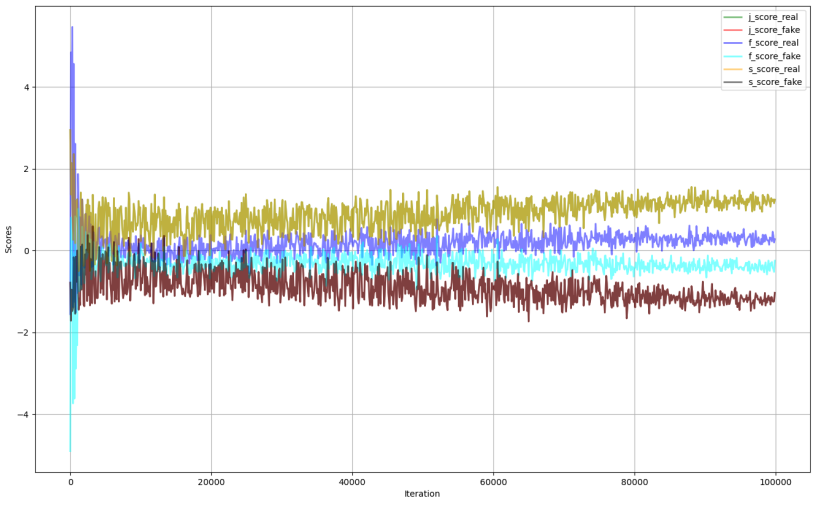


Figure 6.6: real and fake scores for each discriminator module during the training phase for stl10 dataset.



6.3 Evaluation in Spatial Domain

We use the conventional methods, Frechet Inception Distance and Inception Score explained in the section 2.3 for quantitative comparisons in the spatial domain. FID is calculated over 50000 samples of fake and real images. Inception score is computed over 50000 samples from the fake images with 10 splits. The evaluation is against the baseline (*SNGAN*) **sngan** and the SSDGAN [71] model. SSDGAN, the same as our model, uses frequency domain to enhance the discriminator. They use the azimuthal Integral of the amplitude as an input for their frequency module, which is a fully connected layer. Their model disregards the phase information of the FFT completely. While the amplitude of the FFT carries the intensity of each frequency, Phase information contains the location of these frequencies. To a human eyes phase information looks more important; hence we have not set them aside in our model. The other drawback of their model is using the azimuthal integration over FFT. By binning the same frequency resolutions, their model will lose the amplitude location information of the FFT. On the other hand, Ours takes the FFT as it is without any information loss as input.

table 6.3 and 6.3 demonstrate the quantitative comparison. Our model outperforms all other models except the inception score for stl-10, which is less but comparable to SSDGAN. However, SSDGAN has some major flaws, which we fully describe in the next section. Note that for the FID, the lower, the better, and for inception score, the higher, the better.

Figure 6.7 and 6.8 show some examples of the images created based on CIFAR100 and stl10 datasets respectively. We have not cherry picked the images and they are random images

Model	FID	Inception Score
SNGAN	22.61	7.57
SSDGAN	20.90	7.71
FreqGAN	**19.63**	**7.82**

Table 6.10: FID and inception score comparison between different models for CIFAR-100 dataset

Model	FID	Inception Score
SNGAN	39.56	8.04
SSDGAN	36.41	**8.47**
FreqGAN	**35.89**	8.31

Table 6.11: FID and inception score comparison between different models for stl-10 dataset generated by the generator of our model.

Figure 6.7: Random set of images generated based on cifar100 dataset

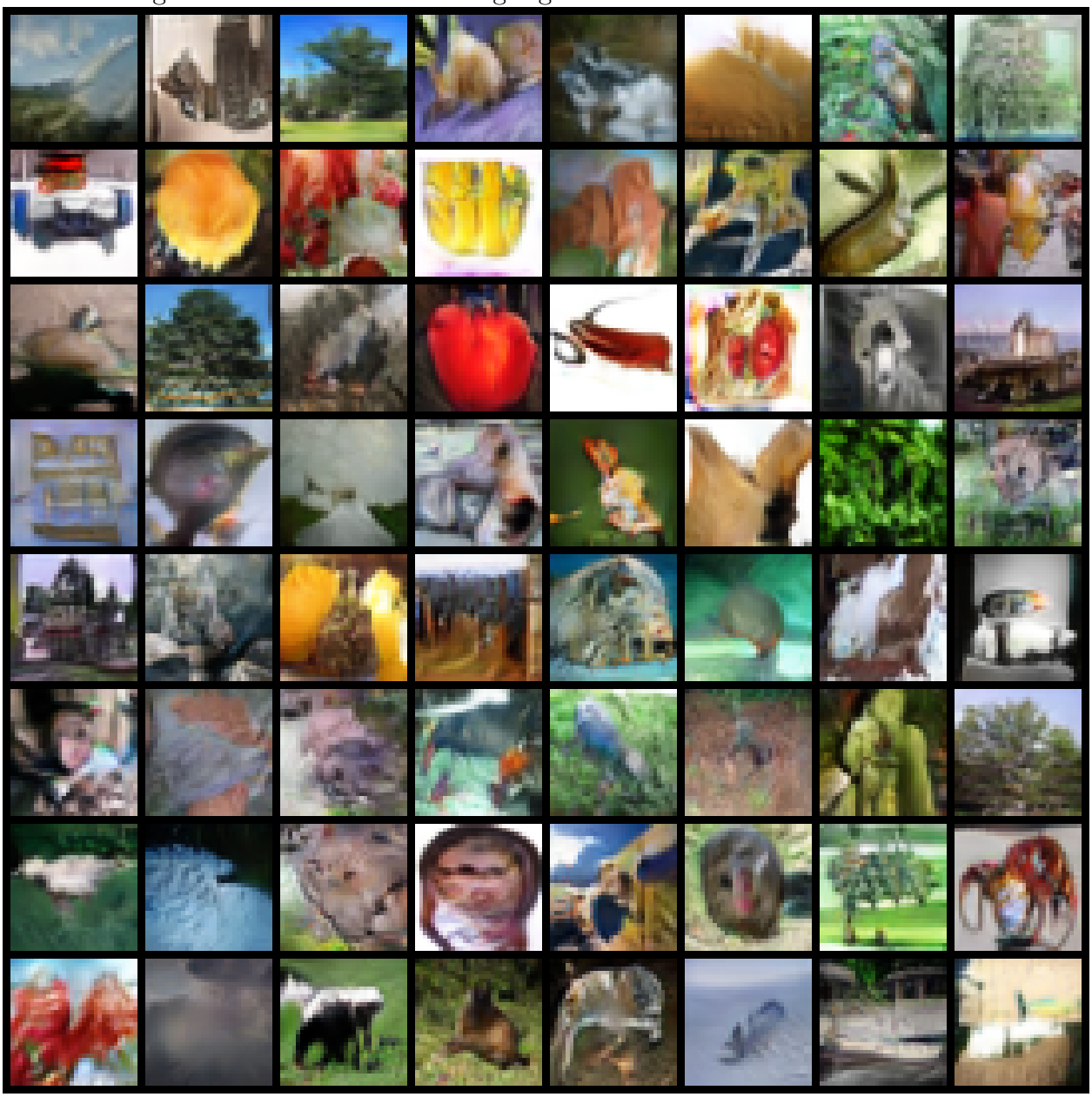
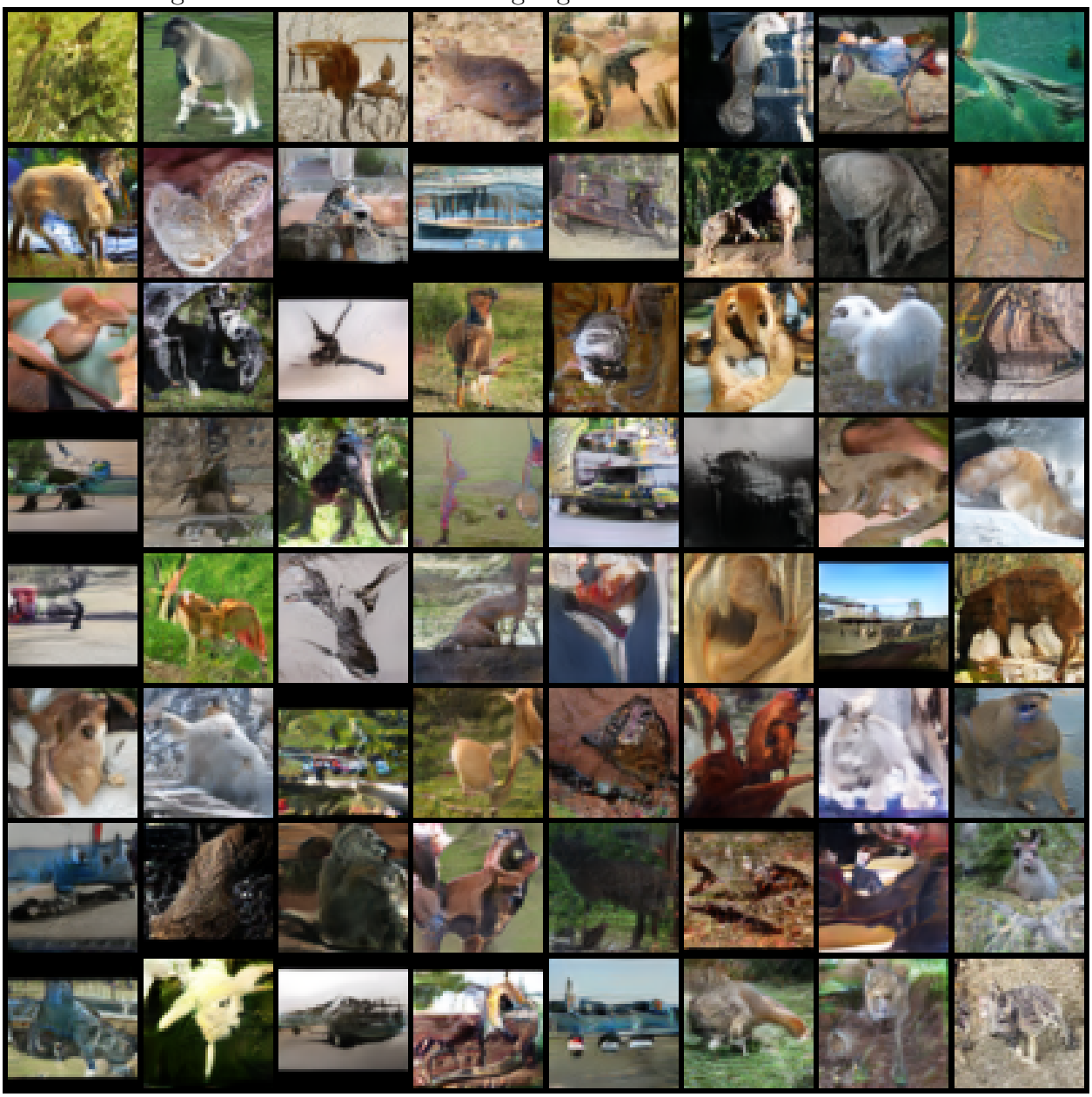


Figure 6.8: Random set of images generated based on stl10 dataset



6.4 Evaluation in Frequency Domain

Unlike the spatial domain, Evaluation in the frequency domain does not have a well-established method. To illustrate how generated images are similar to the real ones in the frequency-domain power spectrum is a good candidate. To further show the difference, we use the power spectrum distance between real images and those produced by the GAN models, normalized by the spectra of real images. (used in [77])

$$Power\ Spectrum\ Distance = \frac{|\mathbb{E}[PS(real))] - \mathbb{E}[PS(fake)]|}{\mathbb{E}[PS(real))]}$$

We also introduce the 2D frequency amplitude gap, which uses the average distance between the expected 2D-frequency gap between the amplitude of the authentic images and synthetic images normalized by those of authentic images.

$$2D\ Frequency\ Amplitude\ Gap = \frac{|\mathbb{E}[Amp(\mathcal{F}(real))] - \mathbb{E}[Amp(\mathcal{F}(fake)]|}{\mathbb{E}[Amp(\mathcal{F}(real))]}$$

Figure 6.9 shows the power spectrum of our model with real images and the other two models in the stl10 dataset. FreqGAN has the closest distance from the real image. It is more clear looking at the normalized power spectrum distance in figure 6.10. The ideal generator should have a constant zero distance throughout all the frequency bins. To our surprise, despite getting the azimuthal integration of the magnitude, which is essentially a variation of the power spectrum itself, SSDGAN has even poorer performance than the baseline SNGAN in the power spectrum. We believe this is owing to fully connected architecture. The fully connected neural networks have a dense hypothesis class in the frequency domain; therefore, estimating the best function even if the neurons are not many is hard for the neural network.

In addition, the lack of unitary modules will cause an uncontrolled effect of these not-well-trained elements in the training process. FreqGAN, contrastingly, shows an almost flat curve over all frequency bins. addressing the high-frequency discrepancies in the SNGAN.

Figure 6.11 measures the same idea in a more direct way with the normalized 2d frequency spectrum gap. The color map shows the difference; the darker the pixel, the more significant the difference between the fake and authentic images. FreqGAN here shows the best performance overall again, while SSDGAN shows inferior performance than the baseline.

For further discussion, we believe the difference of our work will show itself for higher resolution photos more significantly. The high-resolution photos suffer more from high-frequency discrepancies, and our model shows a spectacular result in addressing them. In our plans for future work, we have in mind to further analyze the relation of resolution with the performance of our model.

Figure 6.9: Power spectrum of the real images, FreqGAN, SNGAN and SSDGAN with respect to the frequency bin in different resolutions on the stl10 dataset.

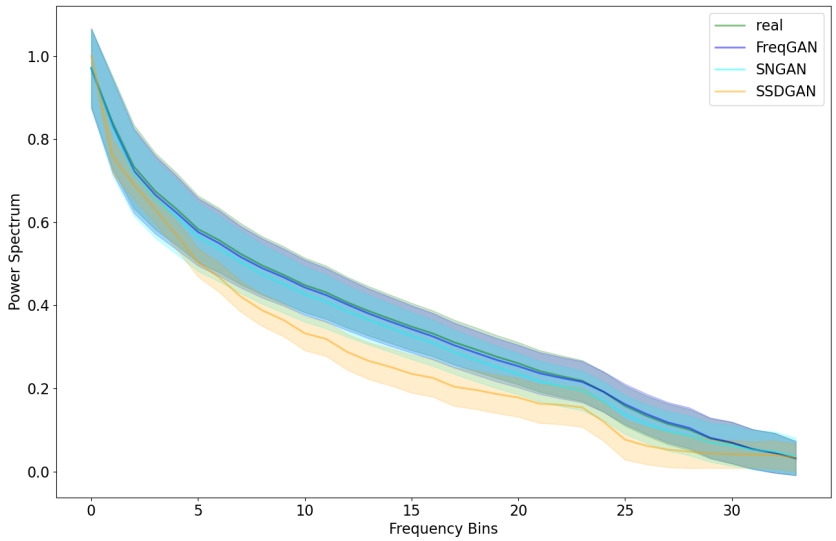


Figure 6.10: Normalized power spectrum distance of FreqGAN, SNGAN and SSDGAN with respect to the frequency bin in different resolutions on the stl10 dataset.

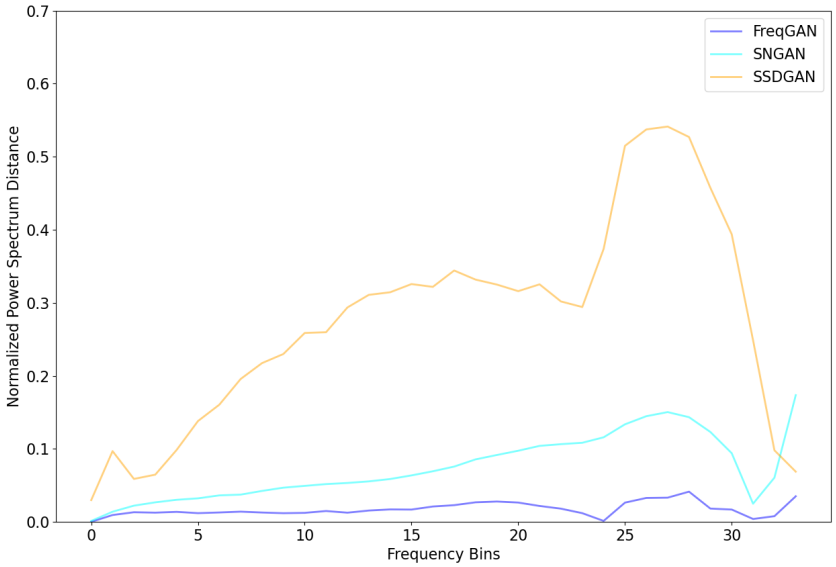
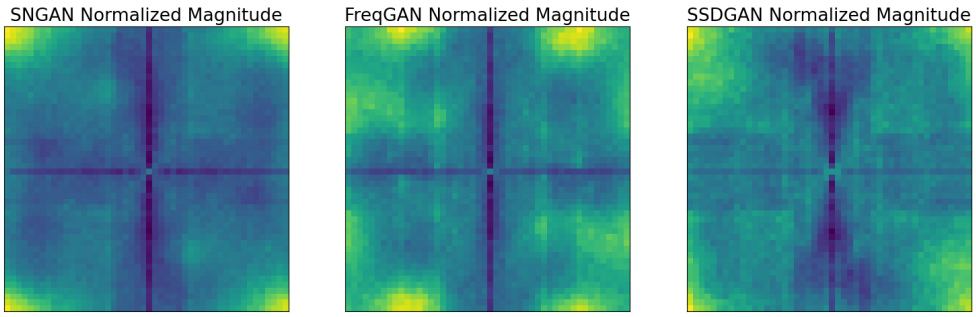


Figure 6.11: 2D frequency amplitude gap of FreqGAN, SNGAN and SSDGAN with respect to the frequency bin in different resolutions on the stl10 dataset.



6.5 Frequency Architecture Investigation

We used the powerful tool of geometric deep learning to design our architecture for the frequency domain. In this section, we discuss its performance over the fully connected architecture and regular CNN. We trained 3 trails of neural networks for each architecture. The fully connected version is the same as the work done by SSDGAN [71]. The CNN version is equipped with spectral normalization as well to keep the comparison fair. Tabel 6.5 shows that GDL based architecture outperforms other models by a significant margin.

Model	**FID**
SNGAN (Baseline)	22.61
Fully Connected	21.57
Unmodified CNN	22.01
GDL inspired CNN	**19.63**

Table 6.12: FID comparison between different architectures in the frequency domain for CIFAR100 dataset

Chapter 7

Conclusion

7.1 Conclusion

This thesis brought up current flaws in the CNN-based image generation techniques (such as generative adversarial networks), which made them easily distinguishable from authentic images. Then we investigate the possible justification for such flaws both mathematically and empirically. We found the main reason in high-frequency discrepancies between the spectrum of synthetic and authentic images. We showed that this disparity has its roots in the transconvolution layers of CNNs. This work proposed a solution for the GANs. An additional path for the discriminator, processing the spectrum of the images. Knowing the frequency representation of the images, the discriminator is equipped with direct access to detect problematic high-frequency elements. On the other side of the story, the generator adopts its weight to reduce the high-frequency elements, resulting in more realistic, hard to

distinguish images.

Different datasets exhibit different distributions in their spatial and frequency features. Some are easier to detect in frequency, some in spatial. The proposed unitary modules in this work adjust the amount of attention the model gives to frequency or spatial domains based on their importance in the data. This scheme in leading the optimization not only resulted in better outputs but also stabilized the training.

This work empirically investigates different architectural designs for the frequency path of the discriminator, including fully connected and CNN schemes, and comes up with a brand new architecture EV-Freq. EV-Freq has systematically designed based on the geometric deep learning blueprint. It utilizes the physical constraints of shift invariancy in the images as an inductive bias to reduce the search space and improve the neural network's estimation ability on the spectrum of the corresponding images. This thesis provides complete theoretical analysis and proofs for the newly design architecture.

We evaluate our extension to the GANs framework over two datasets, CIFAR100 and stl10, in spatial and frequency domains. In the conventional spatial domain, it outperformed the baseline by a significant margin. Our model also outperformed the other frequency-driven effort in improving GANs, SSDGAN. FreqGAN achieved an outstanding performance in the frequency domain, generating images almost identical to real images concerning the power spectrum. GANs are notorious for training instability. A section of this work focused on empirical substantiations that the proposed extension would not jeopardize the stability of the baseline.

7.2 Future Works

The future steps for the current research are listed as follow:

- This work improves the quality spectrum characteristics of the latent distribution of the generator by improving the discriminator power in frequency. Yet, it does not propose any direct change in the generator's architecture. Investigating new designs for the generator to facilitate adopting with the frequency features is one of our prominent future lines of research

- EV-freq introduced a shift equivariant function operating on frequency domain; however, these classes of functions keep the phase information intact, in the future steps, we continue our search to find functions operating on the phase of FFT spectrum to utilize them in the detection process of the discriminator.

- Current work uses the FFT of the grayscaled version of the images. In other words, it just takes advantage of the brightness frequency in the images. For future steps, we intend to extend our work to a colored version of the image utilizing the color frequency information in our model

- New PyTorch update introduces complex autograd. Since back-propagation is now available for complex numbers, we can readily work with complex losses. It promises a new architectural design with no need for IFFT at the end of the block, which can improve our performance in the spectral domain and improve the memory usage and computation complexity.

Bibliography

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[3] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[4] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. DOI: 10.1109/cvpr.2017.632. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.632.

[6] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017. DOI: 10.1109/iccv.2017.629. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.629.

[7] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, *Generative adversarial text to image synthesis*, 2016. arXiv: 1605.05396 [cs.NE].

[8] A. Dash, J. C. B. Gamboa, S. Ahmed, M. Liwicki, and M. Z. Afzal, *Tac-gan - text conditioned auxiliary classifier generative adversarial network*, 2017. arXiv: 1703.06412 [cs.CV].

[9] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, *Learning what and where to draw*, 2016. arXiv: 1610.02454 [cs.CV].

[10] S. Bartunov and D. Vetrov, "Few-shot generative modelling with generative matching networks," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2018, pp. 670–678.

[11] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 3943–3956, Nov. 2020, ISSN: 1558-2205. DOI: 10.1109/tcsvt.2019.2920407. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2019.2920407.

[12] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, *Neural photo editing with introspective adversarial networks*, 2016. arXiv: 1609.07093 [cs.LG].

[13] M.-Y. Liu and O. Tuzel, *Coupled generative adversarial networks*, 2016. arXiv: 1606.07536 [cs.CV].

[14] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, *Invertible conditional gans for image editing*, 2016. arXiv: 1611.06355 [cs.CV].

[15] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, *Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling*, 2016. arXiv: 1610.07584 [cs.CV].

[16] M. Gadelha, S. Maji, and R. Wang, "3d shape induction from 2d views of multiple objects," *2017 International Conference on 3D Vision (3DV)*, Oct. 2017. DOI: 10.1109/3dv.2017.00053. [Online]. Available: http://dx.doi.org/10.1109/3DV.2017.00053.

[17] M. Masood, M. Nawaz, K. M. Malik, A. Javed, and A. Irtaza, *Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward*, 2021. arXiv: 2103.00484 [cs.CR].

[18] H. Farid, *Photo Forensics*. The MIT Press, 2016, ISBN: 0262035340.

[19] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot... for now," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020. DOI: 10.1109/cvpr42600.2020.00872. [Online]. Available: http://dx.doi.org/10.1109/cvpr42600.2020.00872.

[20] X. Zhang, S. Karaman, and S.-F. Chang, "Detecting and simulating artifacts in gan fake images," *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 2019. DOI: 10.1109/wifs47025.2019.9035107. [Online]. Available: http://dx.doi.org/10.1109/WIFS47025.2019.9035107.

[21] L. Kauffmann, S. Ramanoël, and C. Peyrin, "The neural bases of spatial frequency processing during scene perception," *Frontiers in Integrative Neuroscience*, vol. 8, p. 37, 2014, ISSN: 1662-5145. DOI: 10.3389/fnint.2014.00037. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnint.2014.00037.

[22] L. E. Hallum, M. S. Landy, and D. J. Heeger, "Human primary visual cortex (v1) is selective for second-order spatial frequency," *Journal of Neurophysiology*, vol. 105, no. 5, pp. 2121–2131, 2011, PMID: 21346207. DOI: 10.1152/jn.01007.2010. eprint: https://doi.org/10.1152/jn.01007.2010. [Online]. Available: https://doi.org/10.1152/jn.01007.2010.

[23] S. Marĉelja, "Mathematical description of the responses of simple cortical cells," *J. Opt. Soc. Am.*, vol. 70, no. 11, pp. 1297–1300, Nov. 1980. DOI: 10.1364/JOSA.70.001297. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=josa-70-11-1297.

[24] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of the Optical Society of America A*, vol. 2, no. 7, pp. 1160–1169, Jul. 1985. DOI: 10.1364/JOSAA.2.001160.

[25] I. Fogel and D. Sagi, "Gabor filters as texture discriminator," *Biol. Cybern.*, vol. 61, no. 2, pp. 103–113, Jun. 1989, ISSN: 0340-1200. DOI: 10.1007/BF00204594. [Online]. Available: https://doi.org/10.1007/BF00204594.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[27] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," *Lecture Notes in Computer Science*, pp. 297–312, 2014, ISSN: 1611-3349. DOI: 10.1007/978-3-319-10584-0_20. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10584-0_20.

[28] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. DOI: 10.1109/TPAMI.2016.2644615.

[29] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017. DOI: 10.1109/iccv.2017.322. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.322.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

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

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016. DOI: 10.1109/cvpr.2016.90. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2016.90.

[33] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, *Overfeat: Integrated recognition, localization and detection using convolutional networks*, 2013. arXiv: 1312.6229 [cs.CV].

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, ISSN: 2160-9292. DOI: 10.1109/tpami.2016.2577031. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2016.2577031.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), Jun. 2016. DOI: 10.1109/cvpr.2016.91. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.91.

[36] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, *Pixel recurrent neural networks*, 2016. arXiv: 1601.06759 [cs.CV].

[37] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," *Lecture Notes in Computer Science*, pp. 89–105, 2018, ISSN: 1611-3349. DOI: 10.1007/978-3-030-01252-6_6. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-01252-6_6.

[38] J. Y. Cheng, F. Chen, M. T. Alley, J. M. Pauly, and S. S. Vasanawala, *Highly scalable image reconstruction using deep neural networks with bandpass filtering*, 2018. arXiv: 1805.03300 [cs.CV].

[39] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. DOI: 10.3115/v1/d14-1181. [Online]. Available: http://dx.doi.org/10.3115/v1/D14-1181.

[40] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014. DOI: 10.3115/v1/p14-1062. [Online]. Available: http://dx.doi.org/10.3115/v1/P14-1062.

[41] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic clustering and convolutional neural network for short text categorization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 352–357. DOI: 10.3115/v1/P15-2058. [Online]. Available: https://aclanthology.org/P15-2058.

[42] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 2335–2344. [Online]. Available: https://aclanthology.org/C14-1220.

[43] T. H. Nguyen and R. Grishman, "Relation extraction: Perspective from convolutional neural networks," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 39–48. DOI: 10.3115/v1/W15-1506. [Online]. Available: https://aclanthology.org/W15-1506.

[44] Y. Sun, L. Lin, D. Tang, N. Yang, Z. Ji, and X. Wang, "Modeling mention, context and entity with neural networks for entity disambiguation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15, Buenos Aires, Argentina: AAAI Press, 2015, pp. 1333–1339, ISBN: 9781577357384.

[45] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, *Geometric deep learning: Grids, groups, graphs, geodesics, and gauges*, 2021. arXiv: 2104.13478 [cs.LG].

[46] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, Feb. 2019, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2018.10.009. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2018.10.009.

[47] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, *Improved techniques for training gans*, 2016. arXiv: 1606.03498 [cs.LG].

[48] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium*, 2017. arXiv: 1706.08500 [cs.LG].

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016. DOI: 10.1109/cvpr.2016.308. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.308.

[50] R. Durall, M. Keuper, and J. Keuper, "Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020. DOI: 10.1109/cvpr42600.2020.00791. [Online]. Available: http://dx.doi.org/10.1109/CVPR42600.2020.00791.

[51] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein gan*, 2017. arXiv: 1701.07875 [stat.ML].

[52] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, *Improved training of wasserstein gans*, 2017. arXiv: 1704.00028 [cs.LG].

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[54] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[55] J. Donahue, P. Krähenbühl, and T. Darrell, *Adversarial feature learning*, 2016. arXiv: 1605.09782 [cs.LG].

[56] J. Donahue and K. Simonyan, *Large scale adversarial representation learning*, 2019. arXiv: 1907.02544 [cs.CV].

[57] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz, *Leveraging frequency analysis for deep fake image recognition*, 2020. arXiv: 2003.08685 [cs.CV].

[58] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, *On the spectral bias of neural networks*, 2018. arXiv: 1806.08734 [stat.ML].

[59] A. Dziedzic, J. Paparrizos, S. Krishnan, A. Elmore, and M. Franklin, "Band-limited training and inference for convolutional neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 1745–1754.

[60] H. Jiang, *A new perspective on machine learning: How to do perfect supervised learning*, 2019. arXiv: 1901.02046 [cs.LG].

[61] I. Sajedian and J. Rho, "Accurate and instant frequency estimation from noisy sinusoidal waves by deep learning," *Nano convergence*, vol. 6, no. 1, pp. 1–5, 2019.

[62] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "Fcnn: Fourier convolutional neural networks," in *Machine Learning and Knowledge Discovery in Databases*, M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, Eds., Cham: Springer International Publishing, 2017, pp. 786–798, ISBN: 978-3-319-71249-9.

[63] K. Xu, M. Qin, F. Sun, Y. Wang, Y.-K. Chen, and F. Ren, "Learning in the frequency domain," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020. DOI: 10.1109/cvpr42600.2020.00181. [Online]. Available: http://dx.doi.org/10.1109/cvpr42600.2020.00181.

[64] G. Blanc, N. Gupta, G. Valiant, and P. Valiant, *Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process*, 2019. arXiv: 1904.09080 [cs.LG].

[65] N. Razin and N. Cohen, *Implicit regularization in deep learning may not be explainable by norms*, 2020. arXiv: 2005.06398 [cs.LG].

[66] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," *arXiv preprint arXiv:1705.07215*, 2017.

[67] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794–2802.

[68] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[69] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019. DOI: 10.1109/cvpr.2019.00244. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.00244.

[70] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. DOI: 10.23915/distill.00003. [Online]. Available: http://distill.pub/2016/deconv-checkerboard.

[71] Y. Chen, G. Li, C. Jin, S. Liu, and T. Li, *Ssd-gan: Measuring the realness in the spatial and spectral domains*, 2020. arXiv: 2012.05535 [cs.CV].

[72] M. Kadiri, M. Djebbouri, and P. Carre, "Magnitude phase of the dual tree quaternionic wavelet transform for multispectral satellite image denoising," *EURASIP Journal on Image and Video Processing*, vol. 2014, Aug. 2014. DOI: 10.1186/1687-5281-2014-41.

[73] T. Huang, J. Burnett, and A. Deczky, "The importance of phase in image processing filters," English (US), *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 6, pp. 529–542, Dec. 1975, ISSN: 1053-587X. DOI: 10.1109/TASSP.1975.1162738.

[74] K. S. Lee and C. Town, "Mimicry: Towards the reproducibility of gan research," 2020.

[75] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[76] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 215–223. [Online]. Available: http://proceedings.mlr.press/v15/coates11a.html.

[77] R. Gal, D. Cohen, A. Bermano, and D. Cohen-Or, *Swagan: A style-based wavelet-driven generative model*, 2021. arXiv: 2102.06108 [cs.CV].