ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Training Stability of Multi-modal Unsupervised Image-to-Image Translation for Low Image Resolution Quality

A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment of Master of Science in Computer Engineering

By: Yonas Desta Haileyesus

Advisor: Dr. Bisrat Derbesa Dufera

May 25, 2023
Addis Ababa, Ethiopia

Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering

The undersigned have examined the thesis titled:

Training Stability of Multi-modal Unsupervised Image-to-Image Translation for Low Image Resolution Quality

Approval by Boards of Examiners

Dr. Bisrat Derbesa Dufera Chairman, Dept. Graduate Committee	Date	Signature
Dr. Bisrat Derbesa Dufera Advisor	Date	Signature
Internal Examiner	Date	Signature
External Examiner	Date	Signature

DECLARATION

I declare that this thesis is my original work and has not been presented for a degree in any other university, and that all sources of materials used for the thesis have been duly acknowledged.

Student Name: Yonas Desta Haileyesus

Signature:_____

Date:_____

ACKNOWLEDGEMENTS

First and foremost, I would like to praise the Almighty God for giving me the strength to complete the thesis, which has been a protected and challenging journey.

I want to thank my advisor, Dr. Bisrat Derbesa, for his assistance, direction, advice, and support during the thesis process. He has encouraged me to work on impactful and challenging research topics and has taught me how to find good research problems.

Furthermore, I would like to thank my family, friends, all Msc computer engineering students, Imo group friends, and colleagues who supported me throughout the thesis by encouraging, sharing ideas, and critical feedback. This thesis would not have been feasible without their assistance.

Finally, I give my special appreciation to Meseret Haileyesus(CEO of CCFWE) and Yidinakach Gemeda for the companionship, support, and trust they gave me during the most challenging times.

ABSTRACT

The ultimate objective of the unsupervised image-to-image translation is to find the relationship between two distinct visual domains. The major drawback of this task is several alternative outputs from a single input image. In a Multi-modal unsupervised image-to-image translation model, There exist common latent space representations across images from many domains. The model showed one-to-many mapping and its ability to produce several outputs from a particular image source. One of the challenges with the Multi-modal Unsupervised Image-to-Image Translation model is training instability, which occurs when the model is training using a data set with low-quality images, such as 128x128. During the training instability, the generator loss reduces slowly because the generator is too hard trying to find a new equilibrium. To address this limitation, We propose spectral normalization as a method for weight normalization, which would limit the fitting ability of the network to stabilize the training of the discriminator in networks. The Lipschitz constant was a single hyperparameter that was adjusted. Our experiments used two different datasets. The first dataset contains 5000 images, and we conducted two separate experiments using data set with 5 and 10 epochs. In 5 epochs, our proposed method has achieved overall training loss generator losses reduced by 5.049 % on average and discriminator losses reduced by 2.882 % on average. In addition, in 10 epochs, total training loss generator losses of 5.032% and discriminator losses of 2.864% decreased on average. The second data-set contains 20000 images, and we used datasets with 5 and 10 epochs in two different experiments. Over 5 epochs, our proposed method reduced overall training loss generator losses by 4.745 % on average and discriminator losses by 2.787 % on average. Furthermore, in 10 epochs, the average total training loss was reduced, with generator losses of 3.092 % and discriminator losses of 2.497%. In addition, During the transition, our approach produces output images that are more realistic than multi modal unsupervised image-to-image translation.

Keywords: Generative Adversarial Networks, Image-to-Image translation, Style Transfer

Contents

List of Figures

List of Tables

Symbols

$\approx$	Approximately equal to
$\sim$	Equivalence relations
$\mathbb{E}$	Expected Value
$\sim$	distributed as
$\vert$	conditional or given
$\mathbb{R}$	Real Value
$\in$	Set membership
$\subseteq$	Subset
$\sum$	Summation
$\boldsymbol{\theta}$	Thetha
$\boldsymbol{\phi}$	Phi

ACRONYMS

Chapter 1

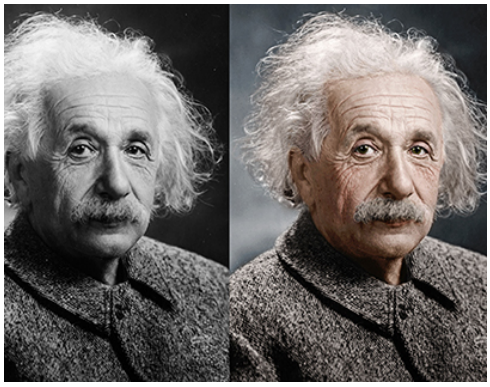
Introduction

1.1 Motivation Of Papers

Today, several images have become freely available because of the fast growth of the Internet and digital capture technologies. There is now a considerable need for picture syntheses and editing activities, such as eliminating undesired elements from marriage photos, changing the hues of landscape views, and transforming image sketches or the other way around. There are several approaches proposed, including intelligent image synthesis and editing. It is traditional approach based on pixels, patches, and low-level image features (characteristics of image content such as color, texture, and shape). These days, deep learning techniques provide an alternative; deep neural networks perform significantly better when network-trained utilizing large-scale data.

The Image-to-Image Translation (I2I) is an effective way to handle various kinds of problems in computer vision and graphics applications [1–3]. The aim is to learn how to translate an image from the **X** to an exact target in the **Y** image [2, 3]. For example, Image colorization [4] translates Gray-scale photos to Red,Green,and Blue (RGB), as can be observed in Fig 1.1(a); Style transfer [5, 6] concept involves converting an image from one form style to another, as shown in 1.1(b), which depicts a photo of the Eiffel Tower style transition from day to night. This category also includes other tasks, such as semantic segmentation [7–9] super-resolution [10, 11], cartoon generation [12–14], Image manipulation [2], etc.

Before understanding how to translate an image between domains, it is crucial to understand the fundamental characteristics that these representations share, which could be either domain-independent or domain-specific. Domain-independent characteristics denote the basic spatial structure pre-



(a) *Gray to RGB*

(b) *day to night*

Figure 1.1: *As an example, consider image-to-image translation (I2I).*

served during translation [15, 16]. Domain-specific aspects are spatial structure rendering and modification during translation [17].

1.2 Problem of Statement

The image-to-image translation falls into categories according to the availability of training data from the two domains of **X** and **Y**. The first one is Supervised image-to-image translation, training data image pairs (x, y), and in many situations, such image pairs do not exist. Unsupervised image-to-image translation, or translating from **X** to **Y** without necessary input-output samples, is another approach to solving these issues. Due to its simplicity of usage and flexibility, unsupervised image translation has a lot of potential. However, it is more complex to utilize because there may be infinite ways of mappings **X** and **Y** domains. A key issue, in terms of probability expressed, is to learn about the joint distribution of pictures that exist in two domains. Therefore, more assumptions and constraints are necessary to understand how to correct the way a map from source to target domains. The coupling theory [18] implies that a limitless number of joint distributions might lead to two marginal distributions in distinct domains.

Many existing works addressing this problem make assumptions about the relationships between the two domains. The Unsupervised Image-to-Image Translation Networks Unsupervised Image-to-Image Translation Networks (UNIT) [19] method assumption is shared latent space, i.e., images can map to fully shared latent space representations even though they come from two different domains. However, the translation method lacks diversity in the trans-

lated outputs result. Multimodal Unsupervised Image-to-image Translation (MUNIT) addresses the UNIT problem of a lack of diverse translation outputs. The MUNIT method assumption is disentanglement representation, i.e., two images from various domains can map shared common latent space representations [15].

Most of the success of the above-mentioned I2Imethods [1], [15, 16] and [19] has come from transferring style image elements with resolution quality 256x256 or greater than pixel size 256x256. Additionally, those techniques required a large amount of computational memory. They employed eight **NVIDIA TITAN V** with 12GB GPUs for their job. Due to limited computational computer memory and GPU resource, running the MUNIT model on a general-purpose computer with a consumer-grade GPU takes a lengthy time. For addressing the computational memory issue, the most common way is reducing the image resolution quality. The MUNIT framework trained with low image resolution quality, which causes the generator loss slowly decrease, and the discriminator prevents them from learning new information that would help them perform better in loss reduction. It was hard to get an equilibrium for the training model. Because discriminators have trained with original images, the generator cannot learn new information. As a result, at lower image resolutions, The MUNIT model shows training instability between the two neural networks, Generator and Discriminator.

1.2.1 Research Questions

At the end of the research, we will answer the following questions.

Q1: What does the effect of Spectral normalization apply on Multi-modal Unsupervised image to image translation using low image resolution ?

1.3 Objectives

1.3.1 General objective

The main objective of this research is to resolve the training instability that occurs when the Multimodal unsupervised image-to-image translation

(MUNIT) model trains with low image resolution quality.

1.3.2 Specific objective

- To implement spectral normalization in multi-modal unsupervised image-to-image translation.
- To investigate Spectral normalization on training instability in MUNIT
- To evaluate and compare our training loss generator and discriminator to the MUNIT model.

1.4 Methodology

The first task of this study was to start a literature review. Throughout the investigation, several types of literature review areas of image translation based on the Generative Adversarial Generating Networks (GANs) way to discover gaps in previous work and develop the problem statement. When an issue becomes a postulate, a technique to remedy it is developed and put into effect. Following the implementation of the suggested technique, edge-to-shoe image datasets are gathered and correctly trained using the proposed system. Finally, we proposed a method of training loss compared to the original MUNIT.

1.5 Contributions

Here, we present generative model frameworks based on deep learning for learning image-to-image translation for unpaired-domain with low image resolution quality. We are introducing spectral normalization for learning image-to-image translation for unpaired-domain. The main contributions of spectral normalization are as follows:

- It is the first general-purpose deep neural network designed to learn Translation between unpaired images.
- It is the first translation network to perform style transfers between low image resolution quality without changing the network's architecture or

one of its hyper-parameters. Owing to the over-complete, multiscale representations it learns, Spectral normalization adapts its feature preservation a weight normalization. It is training that stabilizes between the discriminator and the Generator networks.

1.6 Scope and Limitations

We are focusing on research to develop an image-to-image translation between two domains using the MUNIT framework that can handle training instability with transitional from edges-to-shoes datasets with 128x128 resolution image quality. Large image resolution includes datasets like synthetic to real and summer to winter High Definition (HD) images, which have a resolution of more than 512x512, and animal 2-Dimensional image (2D) image form translation, which need an NVIDIA Tesla P100 or V100 Graphics Processing Unit (GPU) with 16GB + GPU memory.

1.7 Thesis organization

The rest part of the paper is organized as follows: The following Chapter (chapter 2) goes into the theoretical background of research. In chapter 3, for preliminary discussion, we will discuss the areas of literature related to our area of thesis. In chapter 4, We will provide our recommended technique. chapter 5 shows our Implementation Details. Finally, in chapter 6, we will present our findings and suggest potential areas for further investigation.

Chapter 2

Theoretical Background of Research

In the current chapter, we'll present and review core techniques that makeup to understand the backbone of our thesis. We begin with an overview of Generative Modeling and the classification of Generative Models. We will quickly show two of the most extensively utilized strategies in deep generative model domains for image-to-image translation challenges. They are variational auto-encoders and generative adversarial networks. Furthermore, we review a MUNIT. In conclusion, we discuss the theory of image-to-image translation and the basic terms applied throughout the thesis paper has given to understand.

2.1 Generative Model

A generative model describes generating new data in terms that come from the original distribution of data. For example, we have horse image data, and we are using a model that generates a new horse image that is similar and seems to be actual. However, specifically different from our existing horse data images. It is one type of problem that could address by generative modeling. Figure 2.1 represents a high-level overview of a typical Generative modeling process.

The I2I objective seeks to understand the mappings of two distinct domains and how to express these mappings to get the necessary outcomes [21]. The generative model [22, 23] enables the generation of data rather than just the distinction of data sets. Deep generative models, for example, have demonstrated significant performance increases in formulating predictions [24], forecasting missing data [25], compressing datasets [26], and producing unseen

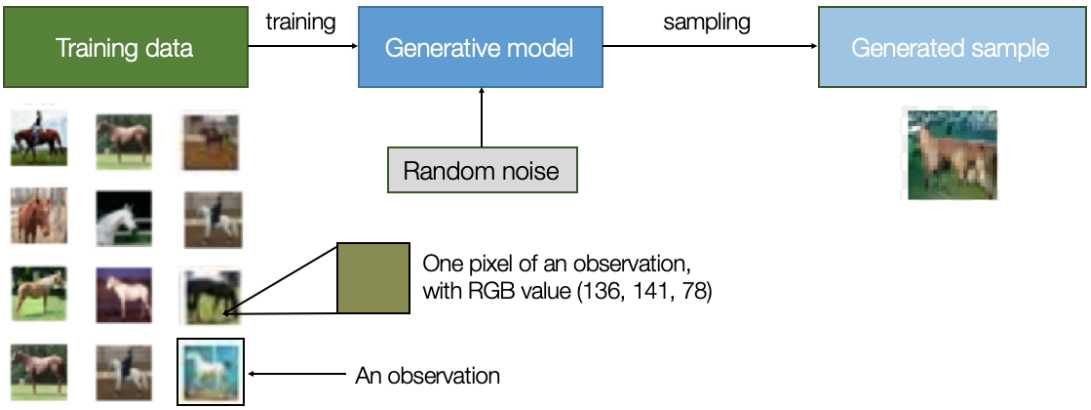


Figure 2.1: *The approach for generative modeling* [20].

data. In an I2I objective, a generative model may mimic the distribution of the desired domain by supplying believable false data, mostly translated pictures that look to have come from the distribution of the desired domain.

There are two types of generative models: traditional and deep. Traditional generative models employ machine learning procedures, whereas deep learning models use deep learning methods.

Traditional generative models estimate the distribution using various kinds of the Probability Density Function (PDF) [27] and do not perform well on complicated distributions [27]. The Hidden Naive Bayes Model (NBM) [28], and Hidden-Markov Model (HMM) [29] are examples of such models.

Deep generative models have generated new samples from large datasets using techniques such as; stochastic back-propagation, deep neural networks, and approximate Bayesian inference [30]. Examples of such models include Deep Boltzmann Machines (DBMs) [31], Deep Belief Networks (DBNs) [32], Variational Auto-Encoders (VAEs) [33], and Generative Adversarial Generating Networks (GANs) [34]. In recent years, the most popular and efficient deep generative models have been Variational Auto-Encoder (VAE) and Generative Adversarial Generating Network (GAN) [30].

We will briefly introduce two deep generative models. section 2.2 in section 2.3, and section 2.3, we discuss the Generative Adversarial Network (GAN), a highly frequently applied and effective technique on image-to-image translation issues. Both models attempt to build an exact copy $x = g(z)$ to generate the needed samples X from the latent variable z. However, their techniques differ [21]. AVAE model approximates data distribution by enhancing the lower

limit of the data log-likelihood, which is the GAN approaches to finding the Nash equilibrium between two networks of generator and discriminator [21].

2.2 Variational Auto-Encoder

In 2013 introduced a new type of neural network called the Variational Auto-Encoder [33]. They have used deep learning architectures for generative modeling. A Variational Auto-Encoder understands the fundamental probability distribution [30]. It prepares another sample dependent on Bayesian inference by optimizing the expected lower limit of the data log-likelihood [30].

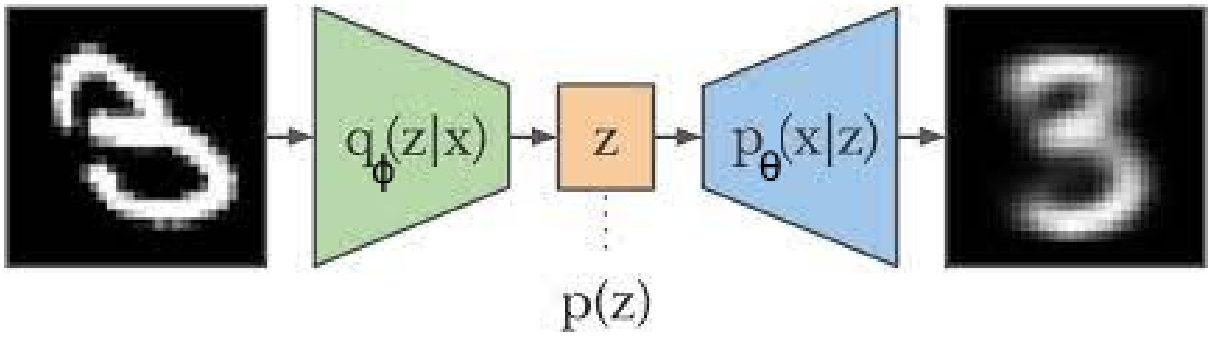


Figure 2.2: *The structure of a VAE*

A VAE VAE uses both an encoder and a decoder approach. the encoder is the recognition approach $q_\phi(z|x)$ to estimate the posterior distribution $p(z|x)$, and the decoder is the generative model $p_\theta(x|z)$ to map the latent variable z to the data x. By maximizing the log-likelihood function $\log p_\theta(x)$, the generative model of a VAE has trained to identify a distribution $p(x)$ that is close to the provided data x [21].

$$\begin{aligned} \log p_\theta(x) &= \sum_{i=1}^{N} \log p_\theta(x_i), \\ \log p_\theta(x_i) &= \log \int p_\theta(x_i|z)p(z)dz \end{aligned} \tag{2.1}$$

To determine the best solution expressed in the Equation of Stochastic gradient descent (SGD) merged with the naïve Monte Carlo gradient estimator (MCGE) could utilize in Equation 2.1. Despite this, it frequently fails due to the exponentially skewed samples $p_\theta(x|z)$ [21]. That has a lot of variation. A VAE presents the model $q_\phi(z|x)$ recognition as a multivariate Gaussian distribution with a diagonal covariance structure [21].

$$q_\phi(z|x) = N(z|\mu_z(x,\phi), \sigma_z^2(x,\phi)I) \tag{2.2}$$

$$\log p_\theta(x_i) = L(x_i, \theta, \phi) + D_{KL}[q_\phi(z|x_i)||p_\theta(x_i|z)] \tag{2.3}$$

The D_{KL} refers to the positive KL divergence. In addition, θ and ϕ are parameters. We could set a variational lower bound on the log-likelihood [21].

$$\log p_\theta(x_i) \geq L(x_i, \theta, \phi) \tag{2.4}$$

As a result, a VAE differentiates and optimizes the lower limit $L(x_i, \theta, \phi)$ rather than $\log p_\theta(x_i)$. The ultimate goal function of a VAE can be shown in the below equation [21].

$$L(x_i, \theta, \phi) = \mathbb{E}_{z\sim q_\phi(z|x_i)}[log p_\theta(x_i|z)] - D_{KL}[q_\phi(z|x_i)||p_\theta(x_i|z)] \tag{2.5}$$

VAEs [33] provide better training stability than GANs [34] and better sampling than autoregressive models [35]. Nevertheless, some practical and theoretical issues with VAEs are still unaddressed. The fundamental disadvantage of variational approaches is the tendency to provide an unacceptable trade-off in sample quality and reconstruction quality due to an insufficient approximate posterior distribution or an excessively simplified posterior distribution. The research investigations in [36, 37] improved the variational posterior to solve the ambiguity of produced samples. Tomczak et al. [38] introduced it as a novel before learning the greater strength of hidden representations. Furthermore, [39] asserted that the inherent over-regularization caused by the KL divergence element in the VAE goal frequently results in an error between $L(x_i, \theta, \phi)$ and the genuine probability.

2.3 Generative Adversarial Generating Network

The basic idea behind Generative Adversarial Networks (GANs) [34] is to construct a game in which there is no winner between two participants: the Generator and a Discriminator. Each participant player is a differentiable function governed by some parameters [21]. It learns data distributions using an adversarial training procedure grounded on game theory [30]. Generator G attempts to produce false but believable pictures. In addition, discriminator D learns to pick out actual and fake picture data [34]. The answer regarding

this game is to achieve a Nash equilibrium between both participants [21]. The remaining sections will go over unconditional GANs (2.3.1), conditional GANs (2.3.2), and how to train GANs (2.3.3).

2.3.1 Unconditional GANs

Unconditional GANs are the original GANs proposed [34] and can be considered unconditional GANs. The method applies Multilayer Perceptron (MLP) [40] to build the properly organized probabilistic model using latent noise variables z and actual data x as inputs [21]. Since Convolutional Neural Network (CNN) [41] has shown to be more successful than the MLP in expressing visual features, Deep Convolutional Generative Adversarial Networks (DGANs) have been recommended by researchers [42] to learn more accurate models of pictures while enhancing earlier GAN capability.

Generator G constructs a false image $G(z)$ is similar to the distribution of the fact data as it can get by sampling random noise z from the model prior distribution $p(z)$ as shown in Figure2.3(a). Discriminator D then receives the genuine sample x from the collected data, and the false sample arbitrary as inputs and output return a probability of zero and one. It indicates that the data receives is a genuine or counterfeit image. In other words, D tries to identify the created fake sample from $G(z)$, and G desires to make samples that would mislead D. As a result, the purpose of the optimization problem is as follows:

The Generator G uses a random noise z sampled from the model's prior distribution $p(z)$ to build a false picture $G(z)$ that is as close to the distribution of real data as shown in Figure2.3(a). Discriminator D then randomly accepts the actual sample x from the dataset and the false sample as input and outputs a probability between 0 and 1, indicating whether the input is a real or fake picture. In other words, D wants to distinguish the generated false sample $G(z)$, whereas G seeks to make samples to confuse D. Consequently, the objective optimization problem is as shown below:

$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (2.6)$$

Previous Equation 2.6 depicts x referred to as the actual data, z to as the random noise vector, G(z) as the false samples created through Generator a G, D(x) as the chance of D representing input as valid, and D(G(z)) as the possibility that D discriminates between G representing inputs [33].

The above Equation 2.6 represent x as the Real data whereas the z is the random noise vector, $G(z)$ represents the fake samples created by Generator a G, $D(x)$ indicates the probability that D's input is Real, and $D(G(z))$ denotes the probability that D Discriminates between G's inputs [34].

2.3.2 Conditional GAN

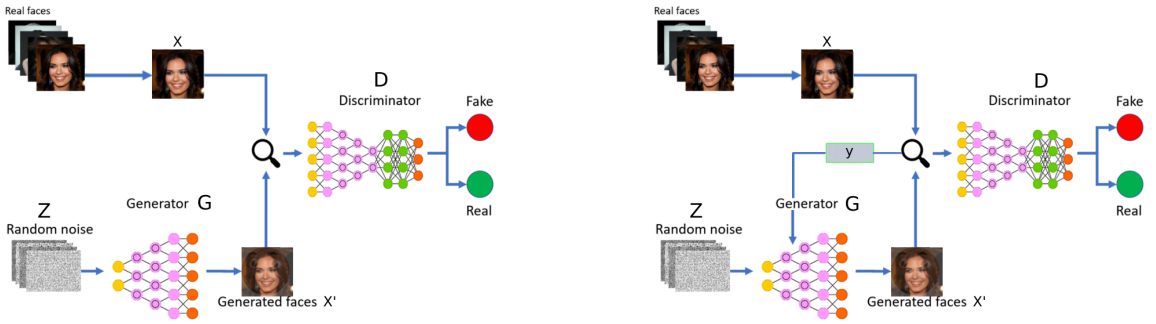
During the unconditional GAN, there is no regulator to obtain we want or desire output from the latent space z vector. As a result, Conditional GAN (cGAN) [42] advocated concatenating extra data y with z to create the picture G(z|y) displayed in Fig 2.3(b). The conditional input y can be anything, comprising data labels, words, and picture properties. Hence, we may use the extra data to navigate the created results toward the right way perspective. The objective function is defined as follows:

In the unconditional GAN, there is no control of what we want to generate because the only input is the random noise vector $\boldsymbol{Z}$ [21]. Therefore, Conditional GAN (cGAN) [43] proposed adding additional information y concatenated with z to generate image $G(z|y)$ shown in Fig 2.3(b). The conditional input y can be any information, such as data labels, text and attributes of images. In this way, we can use the additional information to adjust the generated results in a desirable direction. The objective function is described as

$$\min_{G}\max_{D} L(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))] \tag{2.7}$$

2.3.3 Training GANs

GAN modifies the values of both G and D through the training process utilizing gradient-based optimization techniques, including SGD, Adam, and Root Mean Square Propagation (RMSProp) [20]. The D cannot differentiate between



(a) *The structure of unconditional GANs* (b) *The structure of conditional GANs*

Figure 2.3: *Structural differences between conditional and unconditional GAN.*

the produced sample x' = G(z) and the genuine sample x, which means that given the Nash equilibrium point in this scenario. GAN training is frequently restricted in mode collapse, making convergence a challenge.

During the training process, GAN updates the parameters of G and D using gradient-based optimization methods such as SGD, Adam, and Root Mean Square Propagation (RMSProp) [21]. The entire optimization goal is achieved when D cannot distinguish between the generated sample $\dot{x} = G(z)$ and real sample x, i.e., when the Nash equilibrium is found in this status. In practice, the training of GANs is often trapped in mode collapse, and it is difficult to achieve convergence.

A GAN key challenge is training instability, and many recent works have focused on developing novel cost functions having better non-vanishing or non-exploding gradients overall. The Wasserstein GAN (WGAN) [44] gives an entirely new cost function dependent on the Wasserstein distance to remedy the mode collapse problem [34]. Another approach, Wasserstein GAN with a Gradient Penalty (WGAN-GP) [45] , uses the Lipschitz norm in WGAN using a gradient penalty rather than weight clipping. Least Squares Generative Adversarial Networks (LSGAN) [46] discover that maximizing the least-squares cost function is equivalent to calculating a Pearson X^2 divergence. The Energy-Based GAN (EBGAN) [47] recommended modeling a substitute discriminator with an energy function. This approach can likewise apply the discriminator alongside an auto-encoder and the reconstruction cost Mean squared error (MSE) to evaluate the genuine and produced pictures. The Boundary Equilibrium GAN (BEGAN) [48] technique extends from the EBGAN method.

It uses an Auto-Encoder as the discriminator. Another way is A A Relativistic GAN (RSGAN) [49] analyzes the likelihood that the actual data is more genuine than the produced data, thus rendering the cost function.

2.4 Multi-modal Unsupervised Image-to-image Translation

2.4.1 Assumptions

The model has two assumptions the following.

The first presumption is that the content and style of pictures are constructed up of several picture spaces $x_i \in X_i$. The x_i The i_{th} picture is referred to by X_i, while the matching picture space is indicated by Xi [15]. The purpose is to find the conditional distributions $p(x_{1\to2}|x_2)$ and $p(x_{2\to1}|x_1)$ that will result in the trained translating models $p(x_{1\to2}|x_2)$ and $p(x_{2\to1}|x_1)$. As a result, $p(x_1)$ and $p(x_2)$ correspond to the marginal distributions of both x_1 and x_2, respectively.

The second assumption said that $x_i \in X_i$ has constructed out of a content latent space $c_i \in C_i$ and a style latent space $s_i \in S_i$ for each picture in the datasets [15]. Images from various domains have a similar content space but not an identical style space, as seen in 2.4. As a result, $x_1 = G_1^*(c, s_1)$ and $x_2 = G_2^*(c, s_2)$ create two identical corresponding pictures (x_1, x_2) from the separate generators. The generator functions G_1^* and G_2^* have inverse encoders, $E_1^* = (G_1^*)^{-1}$ and $E_2^* = (G_2^*)^{-1}$, respectively. As this implies, we want to train neural networks to execute encoder and generator tasks [15].

2.4.2 Model

Assuming two domain collections of unpaired pictures in X_1 and X_2, our objective is to build two mappings $M_{x_1\to x_2}$: $X_1 \mapsto X_2$ and $M_{x_2\to x_1}$: $X_2 \mapsto X_1$, which will translate the image across the two domains [50]. It has an Encoder E_i and a Decoder G_i for each of the domains X_i where $(i = 1, 2)$. The translation should be natural and clear and with a specific focus on preserving essential key attribute contents. It makes no strong assumptions about the two domains other than the existence of some shared key content in both set domains. It

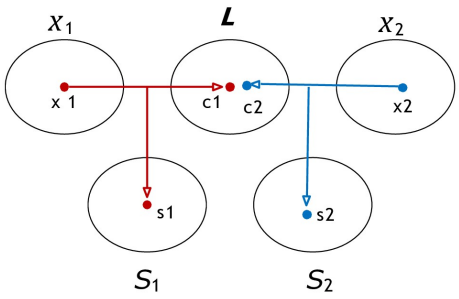


Figure 2.4: *Overview of Assumption of MUNIT , images in different domains consists of images similar Attribute in one domain . Each domain latent space picture in Xi gets divided into a content ci and a style code si. Images from multiple domains have an identical content space but not a similar style space.Each domain latent space images in X_i are factorized into a content code c_i and a style code s_i [15]. Images from different domains have a common shared content space but not a style space [15].*

really should be noted that in nature, such content can be both local and global, making it difficult to describe or image explicitly. Therefore, we use deep neural networks to learn certain key content directly.

2.4.3 Overview of Networks and loss

As part of the training steps, the MUNIT model comprises two parts of the training phases.

In the first step, an auto-encoder is trained. The auto-encoders take images from both domains(edges and shoes) as input and the Encoder E converts to the form of the a content c_i and a style code s_i, since $(c_i, s_i) = (E_i^c(x_i), E_i^s(x_i)) = E_i(x_i)$ [15]. As shown in 2.5., decoder D recognizes the latent codes and converts them right back into a picture. Following training, the auto-encoder generates overcomplete latent codes for every single domain, designated by Z_{x_1} and Z_{x_2} [50].

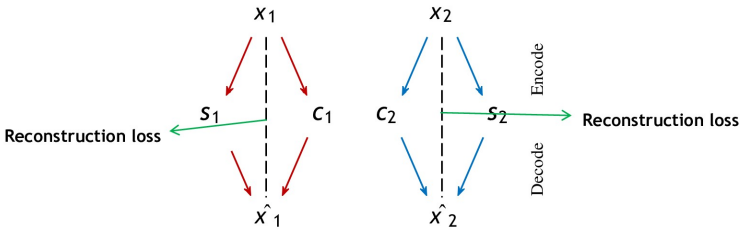


Figure 2.5: *The overview of Auto-Encoder Architecture [15].*

The second component of our network is image translating, which is done

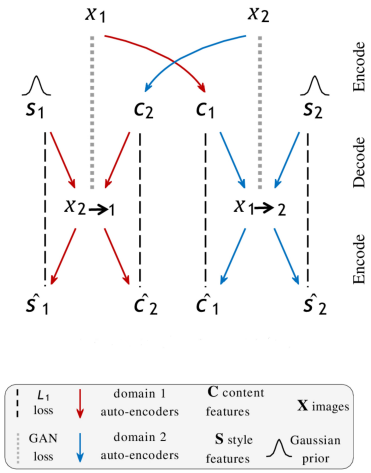


Figure 2.6: *Overview of image Translator Network architecture. The image translation is executed by exchanging encoder-decoder pairs. For example, to translate an image $x_1 \in X_1$ to X_2, its content latent code $c_1 = E_1^c(x_1)$ is extracted and a style latent code s_2 is chosen at random from the prior distribution $q(s_2) \sim N(0,1)$ [15].*

by changing encoder-decoder pairs. It is a network of latent code translators that communicates across Z_{x_1} and Z_{x_2} and is formed of a pair of translators: Z_{x_1} and Z_{x_2}. We see in detail $T_{x_1 \to x_2}$ and $T_{x_2 \to x_1}$ in the Figure 2.6 for more understanding.

We employ two loss functions: one is a bidirectional reconstruction loss which assures the encoders and decoders are inverses, and the other is an adversarial loss that correlates the distribution of translated pictures to the picture distribution in the domain of interest [15].

- **Bidirectional reconstruction loss** goal is to learn inverse encoder and decoder pairs and incorporate them to ensure that the pictures are rebuilt in both directions, picture → latent → picture and latent → picture → latent [15], as shown in Figure 2.5 and 2.6.
  - The image reconstruction loss has calculated the difference between a give image sampled from both domains(edge and shoe) of data distribution and reconstructed the image after encoding and decoding [15].

$$L_{recon}^{x_1} = E_{x_1 \sim p(x_1)}[||G_1(E_1^c(x_1), E_1^s(x_1)) - x_1||_1] \tag{2.8}$$

$$L_{recon}^{x_2} = E_{x_2 \sim p(x_2)}[||G_2(E_2^c(x_2), E_2^s(x_1)) - x_2||_1] \tag{2.9}$$

- The latent reconstruction loss has calculated the difference between a given latent code (i.e., style and content) sampled from latent distribution and reconstructed latent code after decoding and encoding [15].

$$L_{recon}^{c_1} = E_{c_1 \sim p(c_1), s_2 \sim p(s_2)} \left[||E_2^c(G_2(c_1, s_2)) - c_1||_1 \right] \tag{2.10}$$

$$L_{recon}^{s_1} = E_{c_2 \sim p(c_2), s_1 \sim p(s_1)} \left[||E_2^c(G_2(c_1, s_2)) - s_1||_1 \right] \tag{2.11}$$

$$L_{recon}^{c_2} = E_{c_2 \sim p(c_2), s_1 \sim p(s_1)} \left[||E_2^c(G_2(c_2, s_1)) - c_2||_1 \right] \tag{2.12}$$

$$L_{recon}^{s_2} = E_{c_1 \sim p(c_1), s_2 \sim p(s_2)} \left[||E_2^c(G_2(c_1, s_2)) - s_2||_1 \right] \tag{2.13}$$

- **Adversarial loss**: let's take $T_{x_1 \to x_2}$ in Figure 2.6 as an example. The network translates and has also been applied to converted latent code. [15].

$$\begin{aligned} L_{GAN}^{x_2}(G_2, D_2) = {} & E_{c_1 \sim p(c_1), s_2 \sim p(s_2)} \left[\log(1 - D_2(G_2(c_1, s_2))) \right] + \\ & E_{x_2 \sim p(x_2)} \left[\log D_2(x_2) \right] \end{aligned} \tag{2.14}$$

$$\begin{aligned} L_{GAN}^{x_1}(G_1, D_1) = {} & E_{c_2 \sim p(c_2), s_1 \sim p(s_1)} \left[\log(1 - D_1(G_1(c_2, s_1))) \right] + \\ & E_{x_1 \sim p(x_1)} \left[\log D_1(x_1) \right] \end{aligned} \tag{2.15}$$

D_2 is the discriminator function in this case, and it differentiates between the genuine pictures x_2 and the translated images. D1 operates similarly [15].

We collaboratively train the encoders, decoders, and discriminators to optimize the final goal. The two-loss terms described above are combined to generate the total loss function [15]:

$$\begin{aligned} \min_{E_1, E_2, G_1, G_2} \max_{D_1, D_2} L(E_1, E_2, G_1, G_2, D_1, D_2) = {} & L_{GAN}^{x_1} + L_{GAN}^{x_2} + \lambda_x (L_{recon}^{x_1} + L_{recon}^{x_2}) + \\ & \lambda_c (L_{recon}^{c_1} + L_{recon}^{c_2}) + \lambda_s (L_{recon}^{s_1} + L_{recon}^{s_2}) \end{aligned} \tag{2.16}$$

Where λ_x, λ_c and λ_s are the weights that control the reconstruction image [15].

2.5 Image to Image Translation

In this part, we describe terms utilized throughout our thesis to understand the problem. The following definitions are for (GAN) and Image translation .

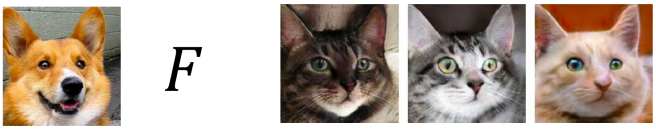
- **Attributes** are the parts of an image that contain meaningful information and features described in greater detail. In other words, an image contains descriptive information and features. For example, skin color.

- **Domain** is a collection of pictures that share identical characteristics and Attributes.

- **Uni-modal image-to-image translation** is to learning one-to-one mapping. The model learns to provide predictable output from a given domain input picture. [30]. As shown in Fig 2.7(a), which will serve as an example.

- **Multi-modal image-to-image translation** aim is to find a one-to-many mapping between a given and the target domain, which allows the model to produce a diverse set of outputs [30]. as shown in Fig 2.7(b), which we shall use as an example.

(a) *Uni-modal image-to-image translation*



(b) *Multi-modal image-to-image translation*

Figure 2.7: *Uni-modal and Multi-modal image translation are two examples. [15]*

- **Domain-independent features** include elements relating to the underlying spatial structure and the content code. In other words, the content must remain untouched during the image translated into the target style [30].

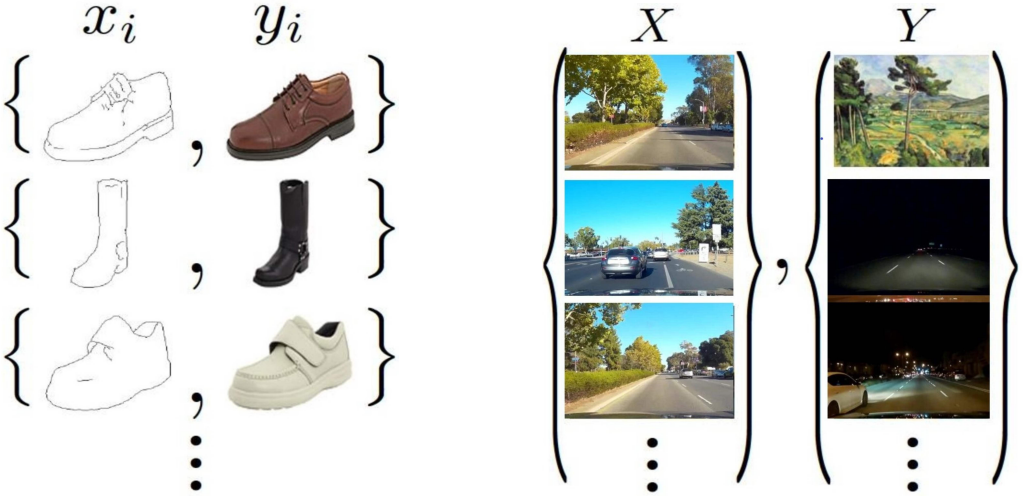
- **Domain-specific features**, commonly known as style code, are concerned with the display of the structure as well as images from the same scene but from distinct domains.

- **Image translation** is an approach to making a new picture from an old one and changing its unique features.

- **Image generation** a method of creating an image using a random noise vector.

- **Paired image-to-image translation** occurs when given picture **X** and counterpart picture **Y** are provided inputs depending on the availability of the training data collection a matched picture pair [30]. As shown in fig 2.8(a), which we will give as an intuition.

- **Unpaired image-to-image translation** unpaired image-to-image translation training data collection comprises a given picture **X** and a matching picture **Y** from two different domains [30]. As illustrated in Fig 2.8(b), which we shall use as an example.



(a) *Paired image-to-image translation*

(b) *Unpaired image-to-image translation*

Figure 2.8: *Examples of image translation Paired and Unpaired* [1 and 15].

Chapter 3

Literature Reviews

This chapter covers the most closely related to our thesis. We started reviewing image translator techniques constructed using generative adversarial networks using paired domain-trained data called Supervised image translation, and those trained with the unpaired domain are known as Unsupervised Image Translation. The supervised translation techniques divide into two ways Directional and Bidirectional Translations. Finally, we will cover Unsupervised image translation approaches.

3.1 Image-to-image translation

The I2I has advanced tremendously in the past few years. There are several reasons for this. Image-to-image translation has made significant progress in the past few years. There are several reasons for this. First, there has been an increase in the availability of image-translating tools and services. Second, there has been an increase in the number of people interested in image translating. Third, additional studies focus on I2I more than ever before. Before researchers began working on the overall general objective of image translating, they concentrated on ways of learning or color transfer. Currently, some works combine images to create composite images that draw from different sources and style transfer from one domain to another. Next, we will discuss image translation methods for both supervised and Unsupervised image translation.

The supervised image-to-image translation methods require a collection of corresponding coupled (a, b) pairs of training pictures from two input domains (A, B) is required for supervised image-to-image translation algorithms [30]. The approach learns through a probability distribution that uses the compiler $F{:}A \longrightarrow B$ for each picture an $a \in A$ and a similar image from another field

$b \in B$. Instance, supervised translation generates a high-quality picture by using domain images conditioned on class labels or input data picture. The supervised translation is further subdivided into Directional and Bidirectional translations, as explained in the following sections.

Pix2Pix [2] is a Directional translation, a conditional generative adversarial network-based supervised image translation technique. Pix2Pix requires paired pictures to learn individual mapping(1-to-1 mapping) and employs two different data sets: one for consideration as input and a second for condition input [2]. Another instance is the picture conversion of a semantic label such as x into a realistic-looking image. The generator implements a "U-Net" design with skip connections to each layer. The discriminator uses a convolution-based "Patch GAN" that acts like a classifier. The objective function applies cGAN with the L_1 norm instead of L_2, which results in lesser blurring. However, the resulting photos are still low in quality and not clear.

Bicycle-GAN zhu2017toward is a bidirectional translation approach that uses paired training images and a model assumption based on multi-model cross-domain translation. It uses a Conditional Variational Auto-Encoder (CVAE) alongside a Conditional Latent Regressor (CLR) to provide accurate and varied output results. Bicycle-GAN forces a bijective link between the two latent encoding spaces zhu2017toward. In addition, resulting in tackling the mode collapse issues [52]. It is built based on Least Squares Generative Adversarial Networks (LSGAN) and helps to stabilize the training process.

he supervised image-to-image translation methods [2 and 51] mentioned above need paired images as training data (one picture before the conversion and one image post-conversion). In many situations, gathering such coupled training pictures takes time or perhaps impossible. It is a unique approach to addressing this problem.

As a result, for multi-purpose image-to-image translation, an unsupervised translation technique needs to be used. To that purpose, Cycle-GAN [1] is an assumption that comes from dual learning in human language interpreting, in which translating networks for two separate directions have together trained and created cycle consistency loss. It combines two generators and two dis-

criminators to learn the mapping between the two separate sets of domains of pictures. The model used two objective function terms: adversarial and cyclic consistency losses. Cycle-GAN approaches have been successful at image translation with simple style appearance changes, such as a horse to a zebra or a zebra to a horse, but commonly fail when the translation wants for geometric deformation.

An auto-encoder has an Encoder and a Decoder. The decoder turns the input pictures into a latent representation.The generator receives this compressed vector to produce high-quality images. The Unsupervised Image-to-Image Translation Network (UNIT) [19] have created using an auto-encoder. It's an assumption shared latent space concept that images in different domains can map to the same latent code in a latent space. Its framework combines Coupled generative adversarial networks Coupled generative adversarial networks (COGAN) and VAEs. Two GANs have trained to convert pictures by encoding that into fully shared latent space and decoding their latent codes as pictures in the domain of interest. However, there are two main drawbacks to this framework. The translated outputs image result is a lack of diversity or unimodal and additional problems caused by the existing training unstable between the generator and discriminator neural network.

The Multimodal unsupervised image-to-image translation (MUNIT) [15] approach is one of the disentangled representation methods to address the UNIT diversity problem. According to the MUNIT framework, the Latent space of images is disentanglement into two sections: content and style code. MUNIT assumption states that pictures from various domains have a similar content space but not an identical style space. It has built two auto-encoders, and the latent code from each auto-encoder has split into the form of content and style code.

MUNIT [15] suffers from content and photo-realism loss during bidirectional reconstruction. The Matting Laplacian [53] technique generates an alpha matte that extracts foreground pictures with white, and the background is black, or vice versa, depending on the demands. We compute the local affine transform factor using this matting Laplacian matrix. Moniz et al. [16] added

an affine transformer to MUNIT Architecture that aids in preserving the geometrical shape of content images during style translation, resulting in more realistic and smooth images. This method uses just one form of image translation uni-modal (one-to-one mapping), does not support multimodal image translation (one-to-many), and is not unsupervised. It is semi-supervised.

Most of the success in the above-mentioned image translation methods [1, 15, 16 and 19] have come from transferring style image elements with resolution quality 256x256 or greater than pixel size 256x256. Additionally, those techniques required a large amount of computational memory. They employed eight **NVIDIA TITAN V** with 12GB GPUs for their job. Due to limited computational computer memory and GPU resource, running the MUNIT model on a general-purpose computer with a consumer-grade GPU takes a lengthy time. For addressing the computational memory issue, the most common way is reducing the image resolution quality. The MUNIT framework trained with low image resolution quality, which causes the generator loss slowly decrease, and the discriminator prevents them from learning new information that would help them perform better in loss reduction. It was hard to get an equilibrium for the training model. Because discriminators have trained with original images, the generator cannot learn new information. As a result, at lower image resolutions, the MUNIT model shows training instability between two neural networks, Generator and Discriminator.

Chapter 4

Methodology

4.1 Introduction

One of the most essential and frequently faced challenges with computer graphics and vision is image translation. With so much interest in images for neural networks in the graphics world today, it's reasonable to wonder if artificial intelligence can learn image translation, especially in unsupervised image translation. A computer can learn to translate an edge to a shoe if the translator has only seen a collection of image edges and shoes without pairings between the two domains of an image set.

The interesting unpaired domain translation problem has attracted the interest of a variety of computer vision and graphics researchers in recent years, including theDomain Transfer Network (DTN) [54], MUNIT [15], Cycle GAN [1], UNIT [19], and others [16]. However, most success in unpaired image translation has come from changing or transferring style image elements with resolution quality 256x256 and above the size of a pixel rather than images with low-resolution quality. The Cycle GAN-based horse-to-zebra translating photographs is a symbolic example, with the network only able to make slight style adjustments to the horse/zebra visuals.

For image translation, we suggest a deep generative model network. Our goal the training will become more stable, and for the image-generating network will learn meaningful information about what the translation image looks like from the unpaired domain, which contains images of low-resolution quality. The network has been trained on two sets of images, for example, the edge and the shoe, each represented by a collection image of datasets. There is no image pairing between the two domains to facilitate image translation, nor is there any pair corresponding relation between any images. Once trained, the

network takes an image from the set of a source domain and turns it into the target.

There is no image-corresponding relationship between the source and destination domains for the translation (i.e., unsupervised image translation). One of the challenges to alleviate is how to correct the network training stability of the images, relating them to facilitate their translation. For that purpose, we perform picture translating in a multi-modal Unsupervised image-to-image Translation (MUNIT), a common latent space shared between two domains. Figure 2.5 shows the latent space code obtained by an auto-encoder trained before image translation.

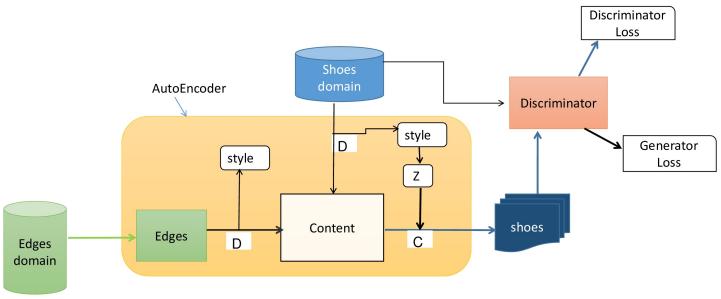


Figure 4.1: Workflows of our method show, D and C represent Decompose and Combine, respectively. To translate a picture from the edges of the source domain into the shoes of the target domain, the image's latent space is first separated into its content and style. To produce alternative target pictures, a generator integrates the content and multiple style vectors (z, the random samples from $N(0,1)$). We train the model with adversarial objectives that ensure the discriminator attempts to differentiate between converted and genuine pictures in the target domain.

More crucially, a suitable image translation from edge to shoe should translate a given image edge to a shoe that is obviously from that specific input edge. As a result, some of the similar edge attributes should be kept during an edge-to-shoe translation, while others can be modified. This feature is the main challenge: it is uncertain which characteristics will be preserved/altered; it should depend upon the provided image domains, and our network must learn it without supervision. To that purpose, on MUNIT network is applied to use spectral normalization to address this challenge.

- The MUNIT framework is training with low image resolution quality, such as 128x128. The output images are blurred, and the holistic structure

is entirely pointless. As a result, at lower image resolutions, the MUNIT model shows training instability between the generator and discriminator.

4.2 Our Proposed Approach

In the discriminator section, we proposed using Spectral Normalization (SN) [55] on Multi-modal Unsupervised image-to-image translation (MUNIT). Spectral normalization is a weight normalization technique that helps to stabilize discriminator training. It controls the Lipschitz constant of the discriminator to avoid the exploding gradient problem. However, to understand more detail about spectral normalization, we will first need to review a couple of concepts of the Lipshitz constant and spectral norm.

Lipschitz continuity is a type of uniform continuity that is particularly strong for functions. It is a property of a function where the function's rate of change is bounded by a constant value. In other words, if you take two points on the graph of a Lipschitz continuous function, the difference between the values of the function at those points cannot be greater than a constant time the distance between the points.

Definition 4.2.1. *Let say $A \subseteq \mathbb{R}$ and $F: A \to \mathbb{R}$ say F is Lipschitz on A if there exist $K \in \mathbb{R}$ such that for each $x_1, x_2 \in A$ we have*

$$||f(x_1) - f(x_2)||_2 \leq k||x_1 - x_2||_2 \tag{4.1}$$

we can rearrange Equation 4.1 to say

$$\frac{||f(x_1) - f(x_2)||}{||x_1 - x_2||} \leq k \tag{4.2}$$

So, the slope of the secant line is $f(x_1)$, $f(x_2)$. The slope of a function has an absolute value that is always less than K, which is the Lipschitz constant. Its Lipschitz function has limited first derivatives.

Let's look at Figure 4.2 to illustrate Lipschitz continuity geometrically. Suppose $F(x) = sin(x)$, F is Lipschitz continuous, then we can draw a cone centered at every point on its graph such that the graph lies outside of this cone.

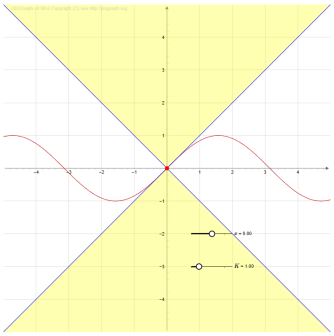
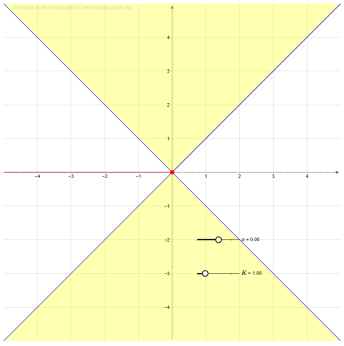


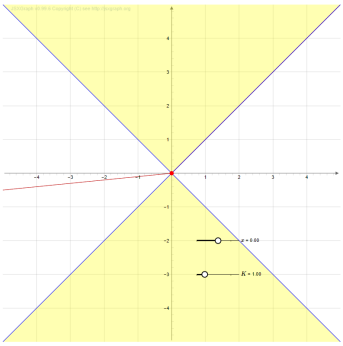
Figure 4.2: There exists a dual cone (yellow) whose origin may be shifted across the graph so that the entire graph continuously remains within the double cone boundary according to the Lipschitz continuous of ***sin*** function.

Let's look other Figure 4.3 to illustrate Lipschitz continuity geometrically. Suppose function $f(x)$ represents the Relu activation function see Figure 4.3(a) whereas $g(x)$ denotes the LeakyRelu Activation function see Figure 4.3(b) used for the activation function.

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} ; \quad g(x) = \begin{cases} ax & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} = \textit{a small positive slope}$$



(a)



(b)

Figure 4.3: The Lipschitz continuity ensures that ReLu and LeakyRelu function always remains entirely outside the yellow cone.

Spectral norm

Mathematically define as :

$$||A|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$$

$||A||$ is represented spectrum norm. It calculates how far away the matrix A may stretch given a vector x dimension.

Let us consider the Singular Value Decomposition (SVD) of a matrix as the

factorization of that matrix into three pieces of matrices. So that every mxn rectangular matrix factors into U, V, and S. The factor U is an orthogonal matrix, the factor S in the middle is a diagonal matrix, and the factor V^T on the right is also an orthogonal matrix.

$$A_{mxn} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

$$A_{mxn} = U_{mxm} S_{mxn} V^T_{nxn}$$

$$U = \begin{bmatrix} u_{11} & \cdots & u_{m1} \\ & \ddots & \\ u_{m1} & \cdots & u_{mm} \end{bmatrix}, V = \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ & \ddots & \\ v_{n1} & \cdots & v_{nn} \end{bmatrix}, S = \begin{bmatrix} \delta_1 & & & 0 \\ & \delta_r & & \\ & & \ddots & \\ 0 & & \cdots & 0 \end{bmatrix}$$

S contains δ instead of square root eigenvalues are called singular values or spectral norms. The matrices U and V, each of which includes a Singular value vector known as an eigenvector. The maximum Singular value of a matrix is known as the spectral norm δ(A).

Spectral normalization

Let us build a basic discriminator using the following formalization of a convolutional neural network using the simplified version of deep neural network input [55].

$$f(x,\theta) = W^{L+1} a_L(W^L(a_{L-1}(W^{L-1}(...a_1(W^1 x).....)))) \\ where\ a = activation\ function,\ \ W = weights\ for\ each\ layer \tag{4.3}$$

The above equation depicts a discriminator neural network with many functions. Take one layer from the neural network and, for each layer L, it outputs $y = Ax$ followed by an activation function $"a"$ that determines whether the layer is ReLU or Leaky-ReLU.

Let us write down the Lipschitz norm of a certain g function that can be determined using its derivative, which is formally denoted by:

$$\|g\|_{Lip} = sup_h \sigma(\bigtriangledown_g(h))$$

Remember gradient of $\bigtriangledown_g$ depends on where you are taking the derivatives. As a result, if you change h, you will get different gradients with their own spectrum norm. We can take a supremum (sup) overall h since the gradients depend on where you take the derivative that gives the Lipschitz constant.

The Lipschitz constant for $g = Ax$ is the spectral norm of W [55].

$$\|g\|_{Lip} = sup_h \sigma(\bigtriangledown(h)) = sup_h \sigma(W) = \sigma(W)$$
$$\bigtriangledown(h) = \bigtriangledown Wh = W$$

The activation function ReLU and the Leaky ReLU Lipschitz constant are always one. As a result, such activation functions exist in the discriminator in Equation 4.3's deep neural network f.

$$\begin{aligned}\|f\|_{Lip} \leq & \|(h_L \mapsto W^{L+1}h_L)\|_{Lip} * \|a\|_{Lip} * \|(h_{L-1}W^L \mapsto h_{L-1})\|_{Lip} \\ & ....\|a_1\|_{Lip} * \|(h_0 W^L \mapsto h_0)\|_{Lip} = \\ & \prod_{l=1}^{L+1} = \|(h_{l-1}W^L \mapsto h_{l-1})\|_{Lip} ||f||_{Lip} = \prod_{l=1}^{L+1} \sigma(W^l)\end{aligned} \tag{4.4}$$

In Equation 4.4, h_L that represented as a hidden layer. There is some property using the "$\leq$" sign that indicates the composition of two functions Lipschitz norm is the product of the Lipschitz norm of those two functions. mathematically represents as $\|g_1 * g_2\|_{Lip} \leq \|g_1\|_{Lip} * \|g_2\|_{Lip}$. So, that is the reason we are using this inequality. In networks, the Lipschitz norm of activation function $||a||_{lip}$ value is equal to 1. Then, we have a linear function that gives the product of spectrum norm W^l. So, the spectral norm of these W^l of your weights has a problem that is not going to be 1, or it is not going to be bound because you can have different W^l.

The weight for every single layer has normalized with the spectral norm (W) during spectral normalization. As a result, the Lipschitz constant for all layers and the network as a whole is one [55] .

$$\begin{aligned}W'_{SN}(W) &= W/\sigma(W) \\ \sigma(W'_{sn}(W)) &= 1 \\ \|f\|_{Lip} &= 1\end{aligned} \tag{4.5}$$

If we normalize each W^l using Equation 4.5 and the fact that $\sigma(W'_{SN}(W)) = 1$

to see that $||f||_{Lip}$ is bounded from above by 1.

Spectral Normalization allows us to re-normalize the weight anytime it is modified. It builds a network that reduces gradient explosion issues. however, the greatest singular value of W is σ(W). Since finding eigenvectors is expensive to solve. There is a cheap and effective technique called **Power Iteration.**

Power Iteration

Power Iteration is a straightforward approach for determining the largest eigenvalue of a matrix. It is simple to find the k^{th} power of A given an eigenvector u.

$$u_{k+1} = Au_k$$
$$u_k = A^k u_0$$
$$u_1 = Au_0 = \lambda u_0$$
$$u_2 = Au_1 = \lambda^2 u_0$$
$$u_k = \lambda^k u_0$$
$$so, u_k \textit{ is much easier}$$

We tend to simply describe a vector v as a linear combination of A's eigenvectors and determine its k^{th} power using A.

$$v_0 = cu_1 + cu_2 + cu_3 + ... + cu_n; \textit{ when } u_1, ... u_n \textit{ is eigenvectors}$$
$$v_k = Av_k$$
$$v_k = A^k v_0$$
$$= c_1 A^k u_1 + c_2 A^k u_2 + ... + c_n A^k u_n$$
$$= c_1 \lambda_1^k u_1 + c_2 \lambda_2^k u_2 + ... + c_n \lambda_n^k u_n$$

Consider u to be A's eigenvectors. These eigenvectors can be reordered, with u becoming the eigenvector containing the largest eigenvalue (dominant eigenvector). A vector b is able to be split down into A's eigenvectors. The k^{th} power of A with v will be close to:

$$b_0 = c_1 v_1 + c_2 v_2 + c_3 v_3 + ... + c_m v_m$$

Let b_0 be the initial guess (with uniform probability), then $c_1 \neq 0$ with proba-

bility 1. Finally, we have got λ, u eigenvectors

$$\begin{aligned} A^K b_0 &= c_1 A^k u_1 + c_2 A^k u_2 + ... + c_n A^k u_n \\ &= c_1 \lambda_1^k u_1 + c_2 \lambda_2^k u_2 + ... + c_n \lambda_n^k u_n \\ &= c_1 \lambda_1^k (v_1 + \frac{C_2}{C_1}(\frac{\lambda_2}{\lambda_1})^k v_2 + ... + \frac{C_m}{C_1}(\frac{\lambda_m}{\lambda_1})^k v_m) \;\; when \; \frac{\lambda_j}{\lambda_1} < 1 \; for \; j > 1 \\ &\longrightarrow \;\; c_1 \lambda_1^k v_1 \end{aligned}$$

Applying the following loop, starting with b, b will eventually converge to several of the dominating eigenvectors. To summarize, we can begin with a random vector b. We may estimate the dominating eigenvector by multiplying it by A.

$$b_{k+1} = \frac{Ab_k}{||Ab_k||}$$

The exact same thing may be implemented with ũ and ṽ repeatedly calculated in the following order (in which A can be represented now as W).

* ũ is first randomly initialized
* ṽ $\leftarrow W^T$ũ / $||W^T$ũ$||_2$, where ũ $\leftarrow W^T$ṽ / $||W^T$ṽ$||_2$ (iteratively)
* Calculate Spectrum norm is $\sigma(W) \approx$ ũ$^T W$ṽ

Finally, Spectra Normalization is done by dividing the weight matrix w_i by its spectral norm $\sigma(W)$ [55], Mathematically written as

$$\frac{w_i}{u_i^T w_i v_i} \tag{4.6}$$

where u_i and v_i are the left/right Singular vectors of w_i corresponding to its largest singular value.

The spectral normalization algorithm should now look easy. We randomly initialize vectors ũ and ṽ for each weight in the discriminator network layer. Because the discriminator weights change slowly, we only need to execute a single power iteration on the current version of these vectors for each layering step of learning. For this reason, spectral normalization is more computationally efficient.

4.3 Implementation Details

In the previous chapter, section 2.4.3, we saw the Auto-Encoder network architecture and the goal of Auto-encoder in the MUNIT. This section shows the Architecture with detailed information and Hyper-parameters.

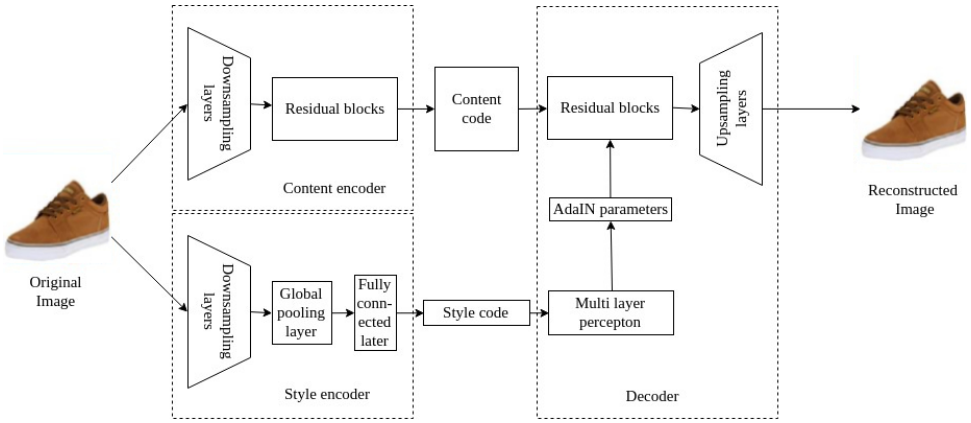


Figure 4.4: Auto-Encoder Architecture

4.3.1 Auto-Encoder

The Auto-Encoder architecture is designed by the encoder and decoder, as shown in more detail in Figure 4.4. Now we can discuss each detailed part of the encoder and decoder. We can talk about each specific component in depth.

(A) **Encoder**

An Encoder's purpose in each domain is to encode images to a latent code. The latent code consists of both content code and style code. Separate sections are required to generate such codes (i.e., content and style code), such as naming the content encoder, which produces the content code, and the style encoder, which generates the style code.

- **Content Encoder**

  The content encoder is composed of two main sections: Down-sampling, which consists of several strides of convolutional layers to reduce the dimensionality of the input data so that it can be processed more quickly of the data (image). In the other section, Multiple residual blocks further processed it. Instance Normalization (IN) comes after all convolutional layers.

- 7 × 7 convolutional blocks with stride 1 and 64 filters.
- 4 × 4 convolutional blocks with stride 2 and 128 filters.
- 4 × 4 convolutional blocks with stride 2 and 256 filters.
- Three residual blocks each consisting of two 3 × 3 convolutional blocks with 256 filters.

• **Style Encoder**

The style encoder consists of downsampling that aids in multiple strides convolutional layers that reduce the input, followed by a Global Average Pooling (GAP) and Fully Connected Layers (FC Layers). MUNIT and our the approach did not use IN layers in the style encoder because IN eliminates the original feature mean and variance, which contains crucial style information.

- 7 × 7 convolutional blocks with stride 1 and 64 filters.
- 4 × 4 convolutional blocks with stride 2 and 128 filters.
- Three 4 × 4 convolutional blocks with stride 2 and 256 filters.
- Global average pooling layer.
- Fully connected layer with 8 filters

(B) **Decoder**

In MUNIT architecture, Auto-Encoder contains another component called Decoder, which we will discuss in detail now. The decoder's primary objective is to reconstruct the input picture from its content and style code. During the image construction, a collection of residual blocks analyses the content code and eventually creates the reconstructed image using several up-sampling and convolutional layers. Inspired by past studies that use affine transformation parameters in normalization layers to indicate styles, we utilize residual blocks with Adaptive Instance Normalization (AdaIN) layers with parameters that would be produced automatically by an MLP from the style code.

- Three residual blocks each consisting of two 3 × 3 convolutional blocks with 256 filters.

- 2 × 2 nearest-neighbor up-sampling layer followed by a 5 × 5 convolutional layers with stride 1 and 128 filters
- 2 × 2 nearest-neighbor up-sampling layers followed by a 5 × 5 convolutional layers with stride 1 and 64 filters.
- 7 × 7 convolutional block with stride 1 and 3 filters.

4.3.2 Discriminator

Discriminators can be used as binary classifiers to distinguish between real and generate fake images. The goal is to determine if the image is true or restored.

The Least Squares GAN (LSGAN) model was created to overcome the vanishing gradient issue triggered by the initial GAN model's minimax and nonsaturated loss. It ap the least squares [21]. The objective of the LSGAN function is to minimize the difference between the Genuine sample distribution and the produced sample distribution [21]. There are two advantages of using LSGAN instead of the original GAN: primarily LSGAN can create excellent quality samples; next, LSGAN enables the learning process to produce a sufficient amount of variety.

We use multi-scale discriminators with three scales. The discriminator training with a multi-scale discriminator helps the network of generators to create a more globally consistent repairing image and finer details, and the overall image-repairing effect seems more natural. In subsection 2.4.3, the two discriminators, D_1 and D_2, are mentioned, which helps the network improves image reconstruction. As a result, according to the sequence stated, the MUNIT discriminator architecture consists of the following layers:

- 4x4 convolutional block with stride 2 and 64 filters
- 4x4 convolutional block with stride 2 and 128 filters
- 4x4 convolutional block with stride 2 and 256 filters
- 4x4 convolutional block with stride 2 and 512 filters

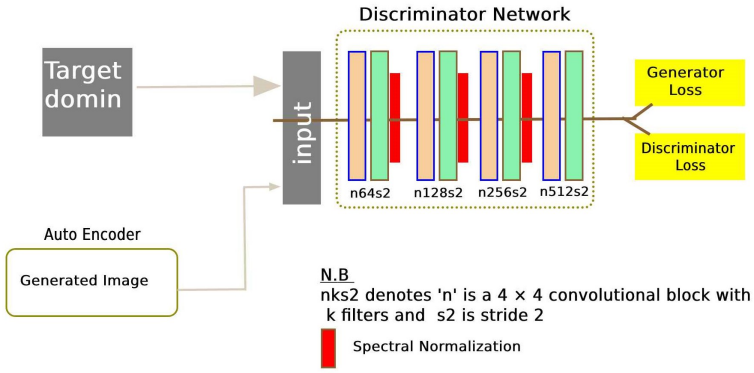


Figure 4.5: The overall Network Architectures of MUNIT

.

The main distinction between our study and earlier research [15 and 16] is in the discriminator section, As seen in Figure Figure 4.6. The SN method is used in our study to normalize the discriminator weights throughout the training phase, with the spectral norm of each layer constrained. Spectral normalization operates in our solution by dividing the weight matrix Wi by its spectral norm [55], as seen in Equation 4.6 in detail. SN outperforms conventional regularization strategies and enhances training stability for GANs [55].

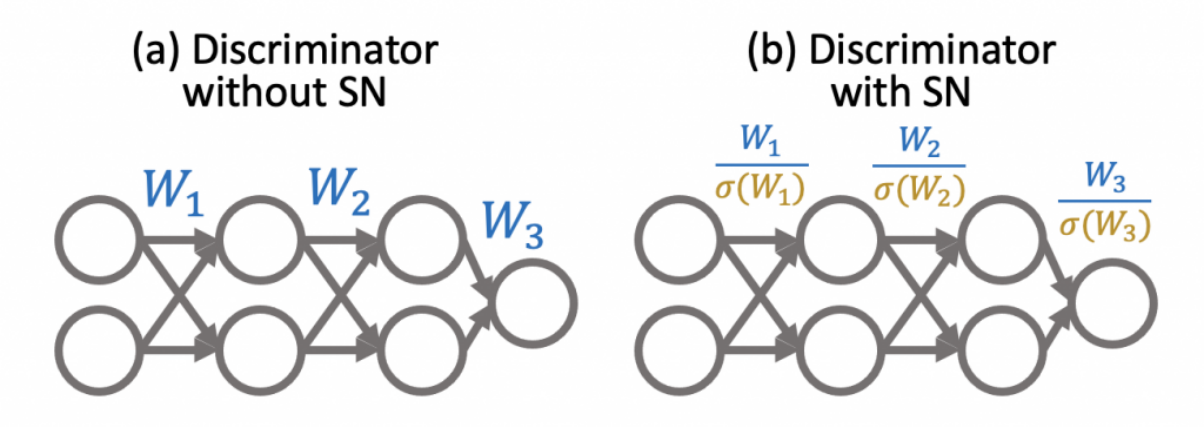


Figure 4.6: Spectral normalization divides the weights W_i by their spectral norms $\sigma(W_i)$

.

N.B: - The MUNIT model working principle is GAN, and GAN is a form of two primary neural networks: a generator that seeks to produce a picture from latent space code, and a discriminator is a classifier that distinguishes between actual and generated images, as we saw in greater depth in section 2.3 of chapter 2. In the case of the Auto-Encoder, the encoder generates latent

space of content and style code, whereas the decoder creates an image from the encoder to build latent space code. In other words, the decoder functions as a GAN generator.

4.4 Evaluation Matrices

In this study, the evaluation metric was the training loss of the generator and discriminator. We collect samples at intervals of 1000 during training and generate an image with those two losses value (the generator and discriminator loss).

(A) **Generator loss**

The generator loss consists of three losses: image reconstruction, image translation, and latent reconstruction loss.

- Images Reconstruction loss

Image reconstruction loss, which we observe in bidirectional reconstruction loss in Equations 2.8 and 2.9.

- Latent reconstruction loss

The latent reconstruction loss consists of two losses: style loss (explained in Equations 2.11 and 2.13), and content loss (described in Equations 2.10 and 2.12).

The latent reconstruction loss contains two losses namely; Style loss as described in Equation 2.11 and 2.13 and Content loss whereas we stated in Equation 2.10 and 2.12 .

- Translate images loss

We train the model with adversarial objectives, which result in a loss known as Translate image loss. To compute translation image loss between model-generated output and target data distribution.

$$L_{GAN}^{x_2}(G_1) = E_{c_1 \sim p(c_1), s_2 \sim p(s_2)} \left[\log(1 - D_2(G_2(c_1, s_2)))\right] \tag{4.7}$$

$$L_{GAN}^{x_1}(G_2) = E_{c_2 \sim p(c_2), s_1 \sim p(s_1)} \left[\log(1 - D_1(G_1(c_2, s_1)))\right] \tag{4.8}$$

Therefore, the total of the three losses, namely image reconstruction loss, latent reconstruction loss, and translate images loss, is known as gener-

ator loss.

(B) **Discriminator**

The discriminator loss is the sum of two discriminators, As described in subsection 2.4.3.

$$L_{GAN}^{x_2}(D_2) = E_{c_1 \sim p(c_1), s_2 \sim p(s_2)} [\log(1 - D_2(G_2(c_1, s_2)))] + E_{x_2 \sim p(x_2)}[\log D_2(x_2)] \tag{4.9}$$

$$L_{GAN}^{x_1}(D_1) = E_{c_2 \sim p(c_2), s_1 \sim p(s_1)} [\log(1 - D_1(G_1(c_2, s_1)))] + E_{x_1 \sim p(x_1)}[\log D_1(x_1)] \tag{4.10}$$

Chapter 5

Experiment and Result

In this chapter, we conducted four experiments to answer chapter 1 contains our research questions, as described in section 1.1.we start with details of the experimental setup of each experiment that used hardware and software tools, datasets, and Hyper-parameters used for this research work. Finally, The experiment results will explain with brief justifications.

5.1 Experimental Setup

The suggested spectrum normalization modules in the Multi-modal image-to-image translation systems are run on the Ubuntu operating system. It uses Anaconda3 IDE to execute and uses Python Programming Language with the 3.8.5 version. Python is the most popular programming language for machine learning operations and the most supported library for different purposes. Its easy syntax, built-in functions, and broad package support have helped it become the most popular programming language. Therefore, the most widely used machine learning and data science tools like Torch version 1.8.1, NumPy version 1.19.2, matplotlib, and Scipy. In our experiment, Nvidia's CUDA cores are preferable over Central Processing Units (CPUs) for high computational cost activities such as parallel processing, real-time image upscaling, executing petaflops of computations per second, and high-definition video rendering, encoding, and decoding. We installed the cuDNN library, which enables high-performance GPU acceleration. It eliminates the requirement to adjust GPU performance at a low level, freeing up time to focus on designing programs for training neural networks. widespread deep learning frameworks like as Keras, TensorFlow, and PyTorch are supported by cuDNN acceleration.

Num.	Name	Detail Specification .
1	Manufacture	Acer - AN515-54-70KK
2	Operating Sysytem	Ubuntu 20.04.4 LTS
2	RAM	16GB DDR4
3	Hard Disk	512 GB NVMe SSD
4	CPU Model name CPU op-mode CPU(s) Thread(s) per core Core(s) per socket Vendor ID CPU family	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 64-bit 12 2 6 GenuineIntel 6
5	L1d cache L1i cache L2 cache L3 cache	192 KiB 192 KiB 1.5 MiB 12 MiB
6	GPU Model GPU Engine Specs Standard Memory Config	GEFORCE RTX 2060 1920NVIDIA CUDA® Cores 6 GB GDDR6

Table 5.1: Hardware specification

5.2 Data sets

The datasets utilized in our experiment are collected from publicly available datasets from the University of California, Berkeley, free accessible from the following link: http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/edges2shoes.tar.gz and previous researches [1, 2 and 15] had also used Edges-to-shoes data-set from this site. It contains the paired form images with sets of pictures of edges with corresponding shoes. However, we divided the paired form of the data-set that contains images with edges to shoes into an unpaired form comprising edges images in one domain and the other shoe images collected in a separate part of the domain. We train the model for edges to shoes with low image resolution qualities, such as 128x128. We used two datasets, one with 5,000 images (i.e., 5,000 edges in one domain and 5,000 shoes in another) and the other with 20,000 images (i.e., 20,000 edges in one domain set and 20,000 shoes in another).

5.3 Hyper-parameters

In all four experiments, the input pictures are 128x128 image resolution quality. We utilize the Adam optimizer with parameters β_1 = 0.5, β_2 = 0.999, and an initial learning rate of 0.0001 15. For experiment 10 epoch, the learning rate is reduced by half. We use a batch size of 1 in all experiments content, and style reconstruction weights are set at λ_x = 10, λ_c = 1, λs = 1, respectively 15. The generator network uses activation function is ReLU and the discriminator uses Leaky ReLU with slope 0.2. 15. We chose the style code that has a size of 8 across 128x128 pixel images.

5.4 Experimental Results

In this section, we conducted four experiments using two datasets; one data set contains 5000 images, and using these images, we conducted two experiments with 5 and 10 epochs. Furthermore, another data set contains 20000 images, and we have performed two experiments using 20000 images with different epoch numbers, 5 and 10. To evaluate the results of the four experiments, we picked up samples from batches in an epoch at 1000-number intervals. In every 1000 intervals, both models save generator output results with generator and discriminator training loss. For illustration, assume we have 10000 images of dogs on which to train our network to identify different breeds of dogs. Let's assume we set the batch size to 1. This implies that an image of dogs will be sent to the network as a group, or batch, all at once. Given that a single epoch is one single pass of all the data through the network, it will take 10000 batches to make up a full epoch. We have 10000 images divided by a batch size of 1, which equals 10000 total batches. As a result, we have taken the first sample batch at $\mathbf{1}^{st}$ batches from the 10000 batches. The second, take a sample at $\mathbf{1000}^{th}$ from the 10000 batches. Next, the third take a third sample at $\mathbf{2000}^{th}$ from the 10000 batches. We can keep doing this procedure 1000 times each interval until we have propagated all samples through the network. Therefore, for every interval sample, both models generate output translating multi-modal images from edges-to-shoes datasets with a training loss generator and discriminator. To compare the performance change improvement of

our model generator loss with MUNIT generator loss and our model discriminator loss with MUNIT discriminator loss utilizing Relative change. Relative change defines the change as a percentage of the value of the losses in the earlier method, i.e.

$$Relative\ change = \frac{\text{MUNIT}\ training\ loss - Our\ Model\ training\ loss}{\text{MUNIT}\ training\ loss} x100\%$$

After calculating the relative change of each interval sample of training loss of two neural networks (i.e., generator and discriminator), we calculated the mean. The mean refers to the number you obtain when you sum up a given set of interval samples of training loss value and then divide this sum by the total number of interval samples of training loss in the experiment.

5.4.1 Experiment One

In the first experiment, we use a data set containing 5000 images, and the image resolution quality is 128x128. We evaluated our proposed technique and MUNIT model on a data set of 5000 images with 5 epochs of hyper-parameters and a batch size of 1. The initial learning rate is reduced by half after two epochs in this experiment. For every 1000 interval sample, we collected 25 samples from 25,000 batches (i.e., Model training 5000 images with batch size 1, for in a single epoch is one single pass of all the data through the network, it will take 50000 batches to make up a full epoch. Thus, there are 25,000 batches across all five epochs). When we compared training loss in our proposed method to MUNIT models, we achieved the overall training loss generator loss was decreased by 5.049 % on average, and the discriminator was reduced by 2.882 % on average. However, our model requires more time complexity than the MUNIT model. When putting it, as a result, it shows an increase of 8 %.

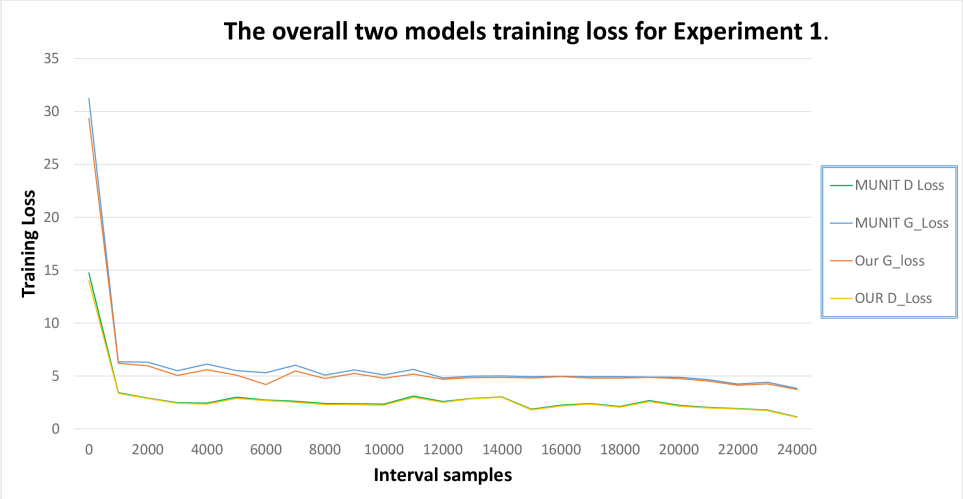


Figure 5.1: The graph shows the overall training loss for two models (i.e., MUNIT and our proposed method) of the generator and discriminator. Its plotted using the results of experiment 1.

Figure 5.1 graph depicts the x-axis representing the interval sampling of each model train and the y-axis each model's training loss value at interval sampling. For every 1000 interval sampling, both models (i.e., MUNIT and our proposed method) generate an image with a generator and discriminator training loss. The following three graphs (i.e., Figure 5.4 from Experiment 5.4.2, Figure 5.5 from Experiment 5.4.3, and Figure 5.6 from Experiment 5.4.4) are represented by the same x-axis and y-axis as Figure 5.1. Therefore, it demonstrates the two models (i.e., MUNIT and our proposed) relationship between generator and discriminator training loss for each interval sample of 1000.

The training loss impact of the two models can be seen in two segments on the graph in Figure 5.1 above. For both models, the initial learning rate hyper-parameter was mentioned in section 5.3, and we used it for up to two epochs (from 0 to 10000 interval samples). Our suggested technique has a lesser training loss than the MUNIT model and produces more photo realistic photos than the MUNIT model .

(a)

Figure 5.2: *The images in Figures 5.2(a), above show that images are the MUNIT model produced during the first half of the training. We have used Experiment 1 to demonstrate. We did not include the additional experiments 2, 3, and 4 because they are similar.*

(a)

Figure 5.3: *The images in Figures 5.3(a), above show that images are our model produced during the first half of the training.We have used Experiment 1 to demonstrate. We did not include the additional experiments 2, 3, and 4 because they are similar.*

In the second section of the graph, which ranges from 2 to 5 Epochs (10000 to 25000 interval samples), the initial learning for both models are reduced by half. In this case, our models indicate a little decreased training loss. Additionally, as seen in Figure 5.1, the graph results of our model and the MUNIT overlap.

5.4.2 Experiment Two

In the Second experiment, we use a data set containing 5000 images, and the image resolution quality is 128x128. We evaluated our proposed technique and MUNIT model on a data set of 5000 images with 10 epochs of hyper-parameters and a batch size of 1. The initial learning rate is reduced by half after five epochs in this experiment. For every 1000 interval sample, we collected 50 samples from 50,000 batches (i.e., Model training 5000 images with batch size 1, for in a single epoch is one single pass of all the data through the network, it will take 5,000 batches to make up a full epoch. Thus, there are 50,000 batches across all ten epochs). When we compared training loss in our proposed method to MUNIT models, we achieved the overall training loss

generator loss was decreased by 5.032 % on average, and the discriminator was reduced by 2.864 % on average. However, our model requires more time complexity than the MUNIT model. When putting it, as a result, it shows an increase of 10 %.

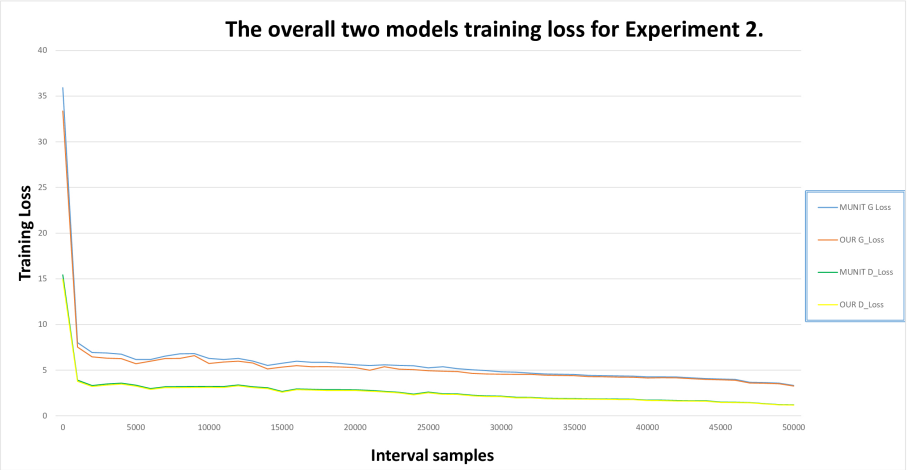


Figure 5.4: The graph shows the overall training loss for two models (i.e., MUNIT and our proposed method) of the generator and discriminator. Its plotted using the results of experiment 2.

The training loss impact of the two models can be seen in two segments on the graph in Figure 5.4 above. For both models, the initial learning rate hyper-parameter was mentioned in section 5.3, and we used it for up to five epochs (from 0 to 50,000 interval samples). Our suggested technique has a lesser training loss than the MUNIT model and produces more photo realistic photos than the MUNIT model. In the second section of the graph, which ranges from 5 to 10 Epochs (25,000 to 50,000 interval samples), the initial learning for both models are reduced by half. In this case, our models indicate a little decreased training loss. Additionally, as seen in Figure 5.4, the graph results of our model and the MUNIT overlap.

5.4.3 Experiment Three

In the third experiment, we use a data set containing 5000 images, and the image resolution quality is 128x128. We evaluated our proposed technique and MUNIT model on a dataset of 20000 images with 5 epochs of hyper-parameters and a batch size of 1. The initial learning rate is reduced by half after two

epochs in this experiment. For every 1000 interval sample, we collected 100 samples from 100,000 batches (i.e., Model training 20,000 images with batch size 1, for in a single epoch is one single pass of all the data through the network, it will take 20,000 batches to make up a full epoch. Thus, there are 100,000 batches across all five epochs). When we compared training loss in our proposed method to MUNIT models, we achieved the overall training loss generator loss was decreased by 4.745 % on average, and the discriminator was reduced by 2.787 % on average. However, our model requires more time complexity than the MUNIT model. When putting it, as a result, it shows an increase of 12 %.

The training loss impact of the two models can be seen in two segments on the graph in Figure 5.5 above. For both models, the initial learning rate hyper-parameter was mentioned in section 5.3, and we used it for up to two epochs (from 0 to 50,000 interval samples). Our suggested technique has a lesser training loss than the MUNIT model and produces more photo realistic photos than the MUNIT model. In the second section of the graph, which ranges from 2 to 5 Epochs (50,000 to 100,000 interval samples), the initial learning for both models are reduced by half. In this case, our models indicate a little decreased training loss. Additionally, as seen in Figure 5.5, the graph results of our model and the MUNIT overlap.

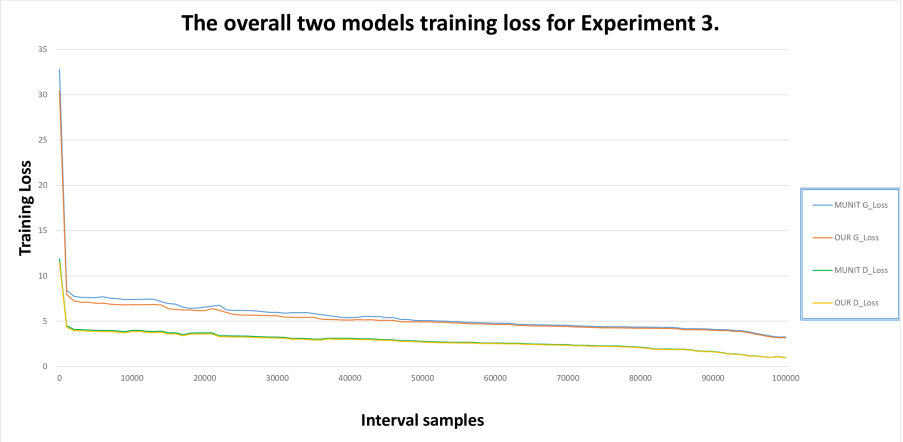


Figure 5.5: The graph shows the overall training loss for two models (i.e., MUNIT and our proposed method) of the generator and discriminator. Its plotted using the results of experiment 3.

5.4.4 Experiment Four

In the third experiment, we use a data set containing 5000 images, and the image resolution quality is 128x128. We evaluated our proposed technique and MUNIT model on a data-set of 20,000 images with 10 epochs of hyper-parameters and a batch size of 1. The initial learning rate is reduced by half after five epochs in this experiment. For every 1000 interval sample, we collected 200 samples from 200,000 batches (i.e., Model training 20,000 images with batch size 1, for in a single epoch is one single pass of all the data through the network, it will take 20,000 batches to make up a full epoch. Thus, there are 200,000 batches across all ten epochs). When we compared training loss in our proposed method to MUNIT models, we achieved the overall training loss generator loss was decreased by 3.092 % on average, and the discriminator was reduced by 2.497 % on average. However, our model requires more time complexity than the MUNIT model. When putting it, as a result, it shows an increase of 13.2 %.

The training loss impact of the two models can be seen in two segments on the graph in Figure 5.6 above. For both models, the initial learning rate hyper-parameter was mentioned in section 5.3, and we used it for up to five epochs (from 0 to 100,000 interval samples). Our suggested technique has a lesser training loss than the MUNIT model and produces more photo realistic photos than the MUNIT model. In the second section of the graph, which ranges from 5 to 10 Epochs (100,000 to 200,000 interval samples), the initial learning for both models are reduced by half. In this case, our models indicate a little decreased training loss. Additionally, as seen in Figure 5.6, the graph results of our model and the MUNIT overlap.

Table 5.2 shows the initial and final loss function values of the generator and discriminator neural networks during training with the MUNIT approach and our recommended method.

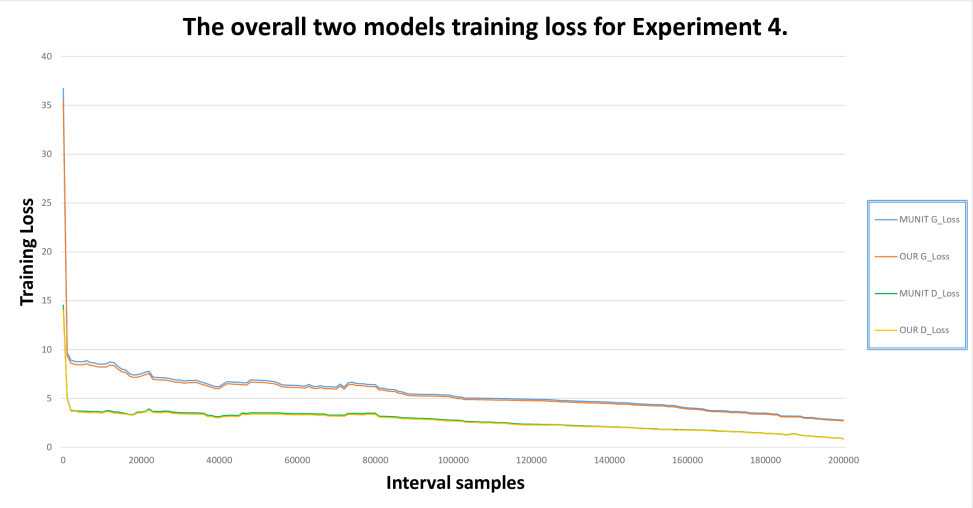


Figure 5.6: The graph shows the overall training loss for two models (i.e., MUNIT and our proposed method) of the generator and discriminator. Its plotted using the results of experiment 4.

Image Resolution 12x128 with 5,0000 images					
No Epoch	Model	Generator Loss Initially	Discriminator Loss Initially	Generator Loss Finally	Discriminator Loss Finally
5	MUNIT	31.225929	14.727692	3.8286	1.123418
	OUR	29.35237326	13.9913074	3.72216492	1.084323054
10	MUNIT	35.915462	15.418649	3.327647	1.19526
	OUR	33.35828111	14.925252323	3.257766413	1.1653785
Image Resolution 128x128 with 20,000 images					
No Epoch	Model	Generator Loss Initially	Discriminator Loss Initially	Generator Loss Finally	Discriminator Loss Finally
5	MUNIT	32.817451	11.872171	3.260404	0.9761414
	OUR	30.387066	11.46851719	3.174329334	0.953690148
10	MUNIT	36.72538	14.544095	2.775576	0.8519808
	OUR	35.45835439	14.04232372	2.701468121	0.8429233

Table 5.2: The results of the Experiment show a comparison of Our proposed method and MUNIT in Training of loss value of generator and discriminator on Edges$\longrightarrow$shoes Datasets.

5.5 Discussion

In this section, we discuss all experiments to address two research questions raised in chapter 1, at subsection 1.2.1. We obtained the following answers to those questions based on the experimental results:

Q1: What does the effect of Spectral normalization apply on Multi-modal Un-

supervised image to image translation using low image resolution quality?

In the first half of the four whole experiments, we observed the image produced by MUNIT is blurred image outputs, unwanted geometric shapes, and information loss holistic structure, as we can see in Figure 5.2. The reason is the Generator network loss is slowly reducing, which means the generator starts to find a way to fool the Discriminator even though the generation is still immature. On the other hand, there is a weight imbalance between Generator and Discriminator. It's hard to get equilibrium for the training model. Because Discriminator is trained with original images, the generator cannot learn new information. Due to this reason, for every 1000 interval samples MUNIT model generates an images are far from the ground truth of the images structure and images that has a high error of training loss of the Generator and Discriminator. However, In our proposed model, as we can see in Figure 5.3, unlike the MUNIT model, we are unable to fully observe the generated images in four experiments because spectrum normalization on the discriminator helps change and bounds the weight that fits the network model and gives the generator some gradient signal to guide it toward creating a better and more realistic image.

In the second half of the graph, which includes all four experiments, we can see how the two training loss models for generator and discriminator networks overlap the MUNIT model and our model. Our model is slightly less accurate than the MUNIT model. Therefore, in four experiments, we compared the total training loss generator and discriminator of our suggested technique to that of the MUNIT model, and we were able to lower the overall training loss values.

Furthermore, our suggested model requires more time complexity than the MUNIT model in all four Experiments. We used Power Iteration in our approach to get the eigenvectors during spectral normalization. However, finding the eigenvectors requires more computation time. Due to this reason, our proposed method has more time complexity than the MUNIT model.

Finally, We show that our proposed methodology reduces the loss function while at the same time avoiding the generated blurred image outputs and unwanted geometric shapes of image attributes observed for low-resolution

images.

5.6 Threats to Validity

During the experiment, we considered two categories of validity.

(A) **Internal Validity**

Internal validity is one of the most important kinds of validity since it involves the mathematical reasoning of connections between the independent and dependent variables. The use of measuring or implementing equipment, techniques, and training datasets might present a threat to internal validity. Threats to measurement and particular threats to instruments may affect the validity of experimental results. Such threats might come from the type of computer used to execute the experiments, the type and number of training datasets and the development tools used. However, in this study, all methodologies are tested on the same system. As a result, the threats will not affect the study outcomes or comparisons.

(B) **External Validity**

External validity is concerned with whether the study findings can be extended to a new circumstance, other participants, locations, and more. External validity may be affected as a result of:

- Parameter selections (strides, padding, kernel size, up sampling, down sampling, residual blocks, and so on): This risk may have an impact on trials using the suggested strategy and previous MUNIT method. In this study, both approaches presented identical parameter settings. As a result, Threats related to parameter selection do not affect comparisons.
- Edge-to-shoes Picking Datasets: - All training datasets would need to be consistent across all experiments. In this investigation, the exact same datasets are utilized for both the proposed and MUNIT approaches. As a consequence, the threats will have no effect on the outcomes of the experiments or evaluations conducted in this study.

Chapter 6

Conclusion and Recommendations

The translation of images from one to another is an exciting topic in computer graphics and vision. We believe it is as important to computer graphics as image-to-image translation is to computer vision. We are considering our work for unsupervised image translation with low image resolution quality to stabilize the training. We end the thesis in this chapter with a summary of the primary contributions of our approaches for learning image translation with low-quality resolution. Next, we'll talk about the direction of future work.

6.1 Conclusions

This paper's work suggested a proposed method for doing unsupervised image-to-image translation for low picture resolution while avoiding the training instability reported in the Multi-modal unsupervised Image-Image Translation (MUNIT) method. During the MUNIT architecture, the generator's loss is slowly reducing, which means the generator starts to find a way to fool the discriminator even though the generation is still immature. In addition, the discriminator prevents the generator network from learning new information. Our proposed method applies to lower GPU memory systems compared to the original MUNIT model proposed. This was achieved by using spectral normalization. The suggested approach has reduced training loss compared to the MUNIT approach on lower-resolution images. Therefore, our proposed method has lower training instability observed for the MUNIT method at lower-resolution images.

Finally, we are interested in investigating image-to-image translation with low-resolution quality for gaining meaningful information and reducing the generator loss of information during the translation task. Giving an artist such

alternatives to enforce greater control losses information and receive meaningful information will make a more practical help for image translation.

6.2 Future work

Our ultimate goal is to support machine learning with all the innovative design tasks performed by artists/modelers, the target domain of inspiration, without the ground truth. The thesis is the initial step in this road map, providing an outline for future research studies.

The most challenging problem in the MUNIT framework is the disentanglement of content and style from low image resolution quality. We are eager to investigate additional solutions and approaches to tackle the problem. An attention mechanism would be a suitable choice since it could enable the network to focus on the appropriate image content regions to extract features from the image. Another option would be to require explicit latent code separating using a regularization term. Furthermore, the MUNIT architecture supports other domains like motion pictures, 3D shapes and texts.

In the future, we would want to consider our approach to be more time-consuming than the MUNIT model and additional research is to reduce this time complexity.

REFERENCES

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[2] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[3] A. Solanki, A. Nayyar, and M. Naved, *Generative Adversarial Networks for Image-to-Image Translation*. Academic Press, 2021.

[4] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*, pp. 649–666, Springer, 2016.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423, 2016.

[7] Q. Yang, N. Li, Z. Zhao, X. Fan, E. I. Chang, Y. Xu, *et al.*, "Mri cross-modality neuroimage-to-neuroimage translation," *arXiv preprint arXiv:1801.06940*, 2018.

[8] X. Guo, Z. Wang, Q. Yang, W. Lv, X. Liu, Q. Wu, and J. Huang, "Gan-based virtual-to-real image translation for urban scene semantic segmentation," *Neurocomputing*, vol. 394, pp. 127–135, 2020.

[9] R. Li, W. Cao, Q. Jiao, S. Wu, and H.-S. Wong, "Simplified unsupervised image translation for semantic segmentation adaptation," *Pattern Recognition*, vol. 105, p. 107343, 2020.

[10] M. S. Uddin, "Real-world single image super-resolution under rainy condition," *arXiv preprint arXiv:2206.08345*, 2022.

[11] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 701–710, 2018.

[12] Z. Zheng, C. Wang, Z. Yu, N. Wang, H. Zheng, and B. Zheng, "Unpaired photo-to-caricature translation on faces in the wild," *Neurocomputing*, vol. 355, pp. 71–81, 2019.

[13] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "Cartoongan: Generative adversarial networks for photo cartoonization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9465–9474, 2018.

[14] X. Wang and J. Yu, "Learning to cartoonize using white-box cartoon representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8090–8099, 2020.

[15] X. Huang, M. Liu, S. J. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," *CoRR*, vol. abs/1804.04732, 2018.

[16] M. Oza, H. Vaghela, and S. Bagul, "Semi-supervised image-to-image translation," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, pp. 16–20, 2019.

[17] A. Royer, K. Bousmalis, S. Gouws, F. Bertsch, I. Mosseri, F. Cole, and K. Murphy, "Xgan: Unsupervised image-to-image translation for many-to-many mappings," in *Domain Adaptation for Visual Understanding*, pp. 33–49, Springer, 2020.

[18] T. Lindvall, *Lectures on the coupling method*. Courier Corporation, 2002.

[19] M. Liu, T. M. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *CoRR*, vol. abs/1703.00848, 2017.

[20] D. Foster, *Generative deep learning: teaching machines to paint, write, compose, and play*. O'Reilly Media, 2019.

[21] Y. Pang, J. Lin, T. Qin, and Z. Chen, "Image-to-image translation: Methods and applications," *IEEE Transactions on Multimedia*, vol. 24, pp. 3859–3881, 2021.

[22] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 1395–1403, 2015.

[23] A. Oussidi and A. Elhassouny, "Deep generative models: Survey," in *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pp. 1–8, IEEE, 2018.

[24] H.-M. Chu, C.-K. Yeh, and Y.-C. F. Wang, "Deep generative models for weakly-supervised multi-label classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[25] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5485–5493, 2017.

[26] M. Tschannen, E. Agustsson, and M. Lucic, "Deep generative models for distribution-preserving lossy compression," *Advances in neural information processing systems*, vol. 31, 2018.

[27] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[28] L. Jiang, H. Zhang, and Z. Cai, "A novel bayes model: Hidden naive bayes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1361–1371, 2009.

[29] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[30] A. Alotaibi, "Deep generative adversarial networks for image-to-image translation: A review," *Symmetry*, vol. 12, no. 10, p. 1705, 2020.

[31] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep boltzmann machines," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 693–700, JMLR Workshop and Conference Proceedings, 2010.

[32] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[33] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[35] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," in *International conference on machine learning*, pp. 881–889, PMLR, 2015.

[36] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*, pp. 1530–1538, PMLR, 2015.

[37] E. Nalisnick, L. Hertel, and P. Smyth, "Approximate inference for deep latent gaussian mixtures," in *NIPS Workshop on Bayesian Deep Learning*, vol. 2, p. 131, 2016.

[38] J. Tomczak and M. Welling, "Vae with a vampprior," in *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223, PMLR, 2018.

[39] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint arXiv:1711.01558*, 2017.

[40] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, classifiaction," 1992.

[41] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

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

[43] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[44] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[45] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[46] 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*, pp. 2794–2802, 2017.

[47] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[48] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[49] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018.

[50] K. Yin, Z. Chen, H. Huang, D. Cohen-Or, and H. Zhang, "Logan: Unpaired shape transform in latent overcomplete space," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–13, 2019.

[51] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," *Advances in neural information processing systems*, vol. 30, 2017.

[52] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 35–51, 2018.

[53] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 228–242, 2008.

[54] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," *arXiv preprint arXiv:1611.02200*, 2016.

[55] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.