STYLE SYNTHESIZING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Yarkın Deniz Çetin
January 2020

Style Synthesizing Conditional Generative Adversarial Networks
By Yarkın Deniz Çetin
January 2020

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Selim Aksoy(Advisor)

Ramazan Gökberk Cinbiş(Co-Advisor)

Ahmet Oğuz Akyüz

Hamdi Dibeklioğlu

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

STYLE SYNTHESIZING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

Yarkın Deniz Çetin
M.S. in Computer Engineering
Advisor: Selim Aksoy
Co-Advisor: Ramazan Gökberk Cinbiş
January 2020

Neural style transfer (NST) models aim to transfer a particular visual style to a image while preserving its content using neural networks. Style transfer models that can apply arbitrary styles without requiring style-specific models or architectures are called universal style transfer (UST) models. Typically a UST model takes a content image and a style image as inputs and outputs the corresponding stylized image. It is, therefore, required to have a style image with the required characteristics to facilitate the transfer. However, in practical applications, where the user wants to apply variations of a style class or a mixture of multiple style classes, such style images may be difficult to find or simply non-existent.

In this work we propose a conditional style transfer network which can model multiple style classes. While our model requires training examples (style images) for each class at training time, it does not require any style images at test time. The model implicitly learns the manifold of each style and is able to generate diverse stylization outputs corresponding to a single style class or a mixture of the available style classes.

This requires the model to be able to learn one-to-many mappings, from an input single class label to multiple styles. For this reason, we build our model based on generative adversarial networks (GAN), which have been shown to generate realistic data in highly complex and multi-modal distributions in numerous domains. More specifically, we design a conditional GAN model that takes a semantic conditioning vector specifying the desired style class(es) and a noise vector as input and outputs the statistics required for applying style transfer.

In order to achieve style transfer, we adapt a preexisting encoder-decoder based

universal style transfer model. The encoder component extracts convolutional feature maps from the content image. These features are first whitened and then colorized using the statistics of the input style image. The decoder component then reconstructs the stylized image from the colorized features. In our adaptation, instead of using full covariance matrices, we approximate the whitening and coloring transforms using diagonal elements of the covariance matrices. We then remove the dependence to the input style image by learning to generate the statistics via our GAN model.

In our experiments, we use a subset of the WikiArt dataset to train and validate our approach. We demonstrate that our approximation method achieves stylization results similar to the preexisting model but with higher speeds and using a fraction of target style statistics. We also show that our conditional GAN model leads to successful style transfer results by learning the manifold of styles corresponding to each style class. We additionally show that the GAN model can be used to generate novel style class combinations, which are highly correlated with the corresponding actual stylization results that are not seen during training.

Keywords: style transfer, neural style transfer, universal style transfer, generative models, generative adversarial networks, conditional generative adversarial networks.

ÖZET

STİL SENTEZLEYİCİ KOŞULLU ÇEKİŞMELİ ÜRETİCİ AĞLAR

Yarkın Deniz Çetin
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Selim Aksoy
İkinci Tez Danışmanı: Ramazan Gökberk Cinbiş
Ocak 2020

Nöral sinir transfer modelleri, nöral ağlar kullanarak bir resmin içeriğini koruyarak, bu resme belirli bir sanatsal stili aktarmayı amaçlar. Stile özgün model ya da mimari gerektirmeden keyfi stiller aktarabilen modellere evrensel stil aktarımı (ESA) modelleri olarak bilnmektedir. ESA modelleri tipik olarak bir içerik ve stil resmini girdi olarak alıp, stillendirilmiş resmi çıktı olarak veririler. Bu nedenle, stil aktarımı için istenilen özelliklere sahip bir stil resminin bulunması gerekmektedir. Ancak bir stilin çeşitlemelerini ya da stillerin kombinasyonlarını aktarılması gereken uygulamalarda, buna uygun bir stil resmi bulunmayabilir ya da bulunması zor olabilir. Bu çalışmada, bir stil resmine gereksinim duymadan stil aktarımı yapabilen bir ağ sunuyoruz. Ağımız stil resmi yerine bir koşullandırma etiketi kabul etmekte ve stil transferini bu koşullandırmaya göre yapmaktadır. Koşullandırma etiketi birden çok stili barındırabilmektedir ve belirli bir etikete koşullandırılmış çeşitli stiller üretebilmektedir. Modelin tek bir koşul etiketiden birçok stile haritalamayı öğrenebilmesi gerekmektedir. Bu nedenle modelimiz, karmaşık ve çok modlu dağılımları gerçekçi bir şekilde üretebilen üretici çekişmeli ağlar (ÇÜA) üzerine kuruludur. Modelimiz girdi olarak istenilen stil sınıflarını belirten anlamsal bir koşullandırma vektörü alan ve stillendirmeyi yapmak için gerekli olan istatistkleri üreten bir koşullu çekişmeli üretici ağdır. Stil aktarımı yapabilmek amacıyla daha önceden geliştirilmiş, otokodlayıcı tabanlı bir stil aktarma modelini uyarlıyoruz. Bu model önce kodlayıcı ile içerik resminin evrişimsel öznitelik vektörlerini çıkartarak bunların üzerine ağartma dönüşümü uygulayarak çalışmaktadır. Daha sonra ağartılmış öznitelikler stil resminin öznitelikleriyle renklendirme dönüşünümüne sokularak stilr resminin kodları elde edilir ve son

olarak kod çözücü, bu koddan stil aktarılmış resmi oluşturmak amacıyla kullanılır. Önerdiğimiz uyarlamada, kovaryans matrislerinin tam hallerini, aynı matrisin yalnızca köşegen elemanlarını kullanarak yakınsıyoruz. Aynı zamanda ÇÜA temelli modelimiz ile özniteliklerin istatistklerini direkt olarak üreterek modelin stil resmi girdisine olan ihtiyacını ortadan kaldırıyoruz. Eğitim ve doğrulama deneylerinde WikiArt verikümesinin bir alt kümesini kullanıyoruz. Hedeflenen stil istatistiklerinin yalnızca küçük bir bölümünü kullanan yakınsama metodumuzun, orijinal metotdan daha hızlı çalıştığını ve orijinal modelle benzer sonuçlar elde ettiğini gösteriyoruz. Aynı zamanda ÇÜA'nın geliştirdiğimiz gerçek stil resimlerine oldukça benzer, eğitim kümesinde bulunmayan stil kombinasyonları üretebildiğini göstermekteyiz.

Anahtar sözcükler: stil transferi, sinirsel stil transferi, evrensel stil transferi, üretici modeller, çekişmeli üretici ağlar, koşullu çekişmeli üretici ağlar.

Acknowledgement

Working in this thesis was a profound experience I will never forget. However this journey would not be possible without the efforts, experience and guidance of Dr. Ramazan Gökberk Cinbiş. I will be always grateful to his seemingly endless patience and understanding throughout this adventure.

I give my special thanks to Dr. Selim Aksoy for his always helpful attitude and agreeing to be my supervisor, Dr. Ahmet Oğuz Akyüz and Dr. Hamdi Dibeklioğlu for agreeing to be in my thesis jury.

I am grateful to have co-workers to study beside and would like to thank, Bulut, Bülent, Gencer and Yiğit for keeping company while giving me great insights on my research. I would like to thank our department secretary Ebru Ateş who, with her kind personality, helped me through the bureaucratic mazes of my master studies.

I am also grateful to Armağan Yavuz and Taleworlds for their support for completing my education and my team members there for their support and understanding.

Finally I would like to thank Computer Engineering Department of Bilkent University, Computer Engineering Department of Middle East Technical University and TÜBİTAK for providing me funding throughout this study. I thank Onur Tırtır for his help with gathering datasets which made this work possible. This work was supported in part by the TUBITAK Grant 116E445. Part of the numerical computations which made this study possible are computed on ImageLab at METU.

This journey was only made possible with the loving presence and undying support of my family; Nazlıcan, my mother and my father.

to Erkin—

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Creation of a painting or an image with a certain artistic style is a challenging task, which can typically be achieved only by people with specific skills and training. Neural Style Transfer [3] introduces the problem of developing computational approaches that can imitate stylistic image creation by transforming existing images.

In this chapter, we make an introduction to the problem of style transfer and main challenges in it. Then we define a novel type of style transfer problem, provide a brief summary of our approach and explain our contributions. We conclude the chapter by giving an outline of the thesis.

1.1 Style Transfer Overview

The main goal of style transfer is commonly defined as transfering the *artistic style* of one particular image, into another. Here, the term artistic style is broad, as it encompasses many aspects of art creation and can be described in various ways. For example, in the unsupervised super-resolution study [1], high resolution is considered as a style, that is being applied to low-resolution images. Style transfer in computer vision typically considered under texture synthesis. In these works

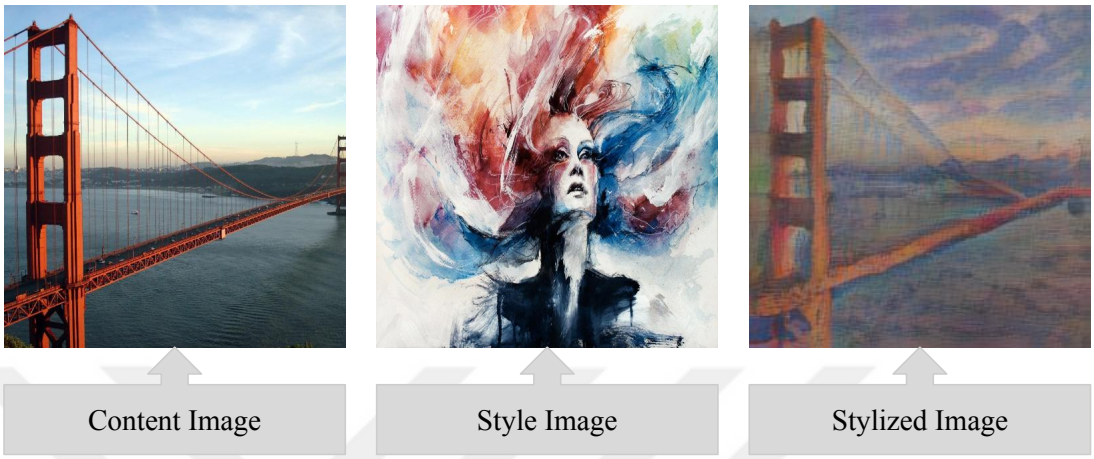


Figure 1.1: Figure showing an example of style transfer. Here the resulting stylized image is obtained using Li et al. [2].

artistic style is considered highly correlated with structure of a texture [4, 5, 6, 7]. Gatys uses a statistically guided definition for style and equates the image of style with its summary of extracted features [3]. However, despite this difficulty in defining artistic style formally and canonically, widely style transfer is considered as the transfer of brush sizes, stroke patterns and color palettes across images.

Therefore, in style transfer, there are two types of input images. The *style image* provides the style information implicitly through an image and the *content image* gives the content information. A style transfer model is expected to extract and use the style information from the style image and apply it to the content image without altering the semantic content and global spatial structure of the original image. In Figure 1.1, style transfer is illustrated through an example showing an input pair of content and style images, and, the resulting stylized image.

A number of different style transfer approaches have been proposed in the literature. Early methods based on traditional computer vision methods uses techniques such as stroke-based [8], region-based [9, 10, 11] and example-based rendering [12]. More recent methods use neural style transfer starting with Gatys et al. [3]. Methods which use neural networks typically split into one of the two categories. These are image optimization and model optimization methods [13].

Image optimization methods try to optimize an image using pre-trained networks while model based methods use feed-forward modes of the networks to perform the transfer.

Style transfer has many potential applications in media generation. It has started to find use in computer generated media, animations and entertainment software. Chen et al. [14] proposes a convolutional style transfer network which can preserve temporal information in video using short-term and long-term coherence losses. Another model, proposed by Gao et al. [15] can perform style transfer in video in real time. Their method also preserves temporal coherency of the style transfer. Applications of the real-time style transfer methods can extend to interactive media as well. For example in 2019, Google announced a plugin which utilizes style transfer with optical flow to create style transferred game renders in real-time [16].

Main challenges. A fundamental challenge lies in the definition of style itself as it is hard to describe style in a quantitative manner. From a technical point of view, the style transfer problem is ill-posed as there is no unique solution given a style image.

As a consequence, evaluating style transfer models is inherently difficult, as well. Ideally, the output of a style transfer model should be rated based on its artistic quality. As this is simply difficult to quantize, most works in this domain resort to qualitative analyses and user studies for evaluation purposes. While the rigor of these singular experiments is always an open question, they can provide useful insight about the method. In this thesis, however, we solely use quantitative metrics for its advantage of making systematic model tuning and evaluation possible. Quantitative evaluation also makes our experimental results much more reproducible.

1.2 Our Semantic Style Transfer Problem

Problem definition.. To our knowledge, all existing style generation methods requires source style images for facilitating style transfer. While this provides a finer control on the outcome images, sometimes images of the given style or style combinations might not be available. In this respect, style transfer models are limited as they can only transfer styles which already exist in the real world.

Towards removing the dependency to explicit style image input and creating a semantically controlled stylization approach, we consider the problem of building style class label conditional style transfer models. More specifically, we aim to train a model that learns the manifold of predefined style classes through provided per-class style training examples. At test time, we want to be able to generate (novel) stylizations of a content image, by applying variations of either one of the existing styles or a novel mixture of them. In Figure 1.2 we provide an overview of our framework, and, illustrate the train and inference time operations.

Our approach.

Due to the complexity and high dimensionality of the image domain, learning to generate images pertaining to predetermined styles is inherently a different problem. Therefore, in our approach, instead of directly learning a generative model that produces the final stylized image, we aim to operate in an intermediate feature representation level. Additionally, our approximation method enables us to represent this intermediate features with fewer dimensions. This in turn increases both the evaluation and training performance as the number of parameters of the network is smaller than networks which generate images directly.

In the original work a pre-trained VGG based networks as an autoencoder which are based on [2] generates the final stylized image given a content and style images as inputs. The encoder network encodes any image to its feature space. The intermediate features i.e. the code generated by the encoder then fed into a architecturally symmetric decoder which creates the final image. For stylization, the features of the style and content image are combined using whitening and

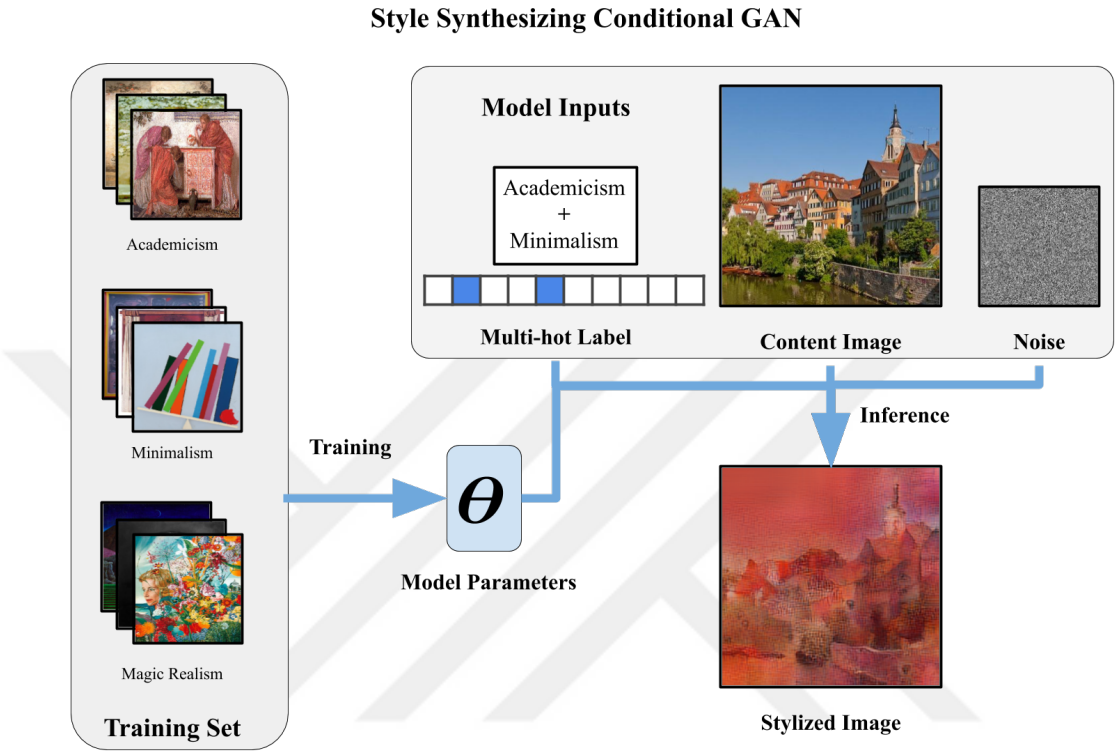


Figure 1.2: Here we see the basic framework of our method. The training set consists of style images from pre-selected style categories and their respective one-hot style labels. Using this dataset we train our model. In inference mode, we input our model an any-hot label representing styles, a content image and a noise.

coloring transforms. The resulting feature vector is then passed into the decoder to create final stylized image.

In our method, instead of the VGG based encoder network we use a GAN based generator to create tensors pertaining to the intermediate feature space. Instead of a style image, our network takes a Gaussian noise vector and style conditioning labels and outputs an approximation of the style features. This approximation greatly requires a reduced dimension input to approximate the style features and further explained in Chapter 3. For the content features we use the same technique as the original model. Finally we combine our generated style features with the content features using whitening and coloring transforms. We then, pass the combined feature vector to decoder same as the original model

to create the final image.

Essentially our model is trained using a pre-determined set of styles and their respective category labels. Then the model is used to generate stylized images. The inputs of our model are; a content image, category label in the form of multi-hot vector and random noise. Our model outputs the full stylized image with style conditioned on the conditioning label.

Contributions. To our current knowledge this work is the first one generating novel styles using label conditional generative models. While there exists other GAN-based style transfer models [17], or other models which combine multiple styles such as [18], these models do not generate new class-level style combinations and instead produce deterministic style transfer.

Our second contribution is the approximation method which we propose for our stylization framework. Instead of using the full feature matrices for performing the whitening and coloring transform as in the original Universal Style Transfer model [2], we approximate the features by encoding the prominent features of a given style. This approximation method provides performance gains and ease of style generation.

1.3 Outline

In Chapter 2 brief historical overview of style transfer and generative methods are provided. Chapter 3 presents both the preliminaries and core of our work to the reader. Chapter 4 contains the relevant experiments and their evaluations of our method. Chapter 5 concludes the thesis with a brief discussion.

Chapter 2

Related Work

In this chapter, we provide an overview of neural style transfer and deep generative models related to our work. First, we give an general outline of traditional and neural style transfer approaches, loosely based on Jing et al. [13]. In our discussion, we also give an overview of the state-of-the art approaches in universal style transfer. Finally, we explain and discuss generative adversarial nets (GAN) [19] since we later use GANs in construction of our style synthesizing approach.

2.1 Traditional Style Transfer Approaches

Before the introduction of neural network based style transfer, a number of works was published on *non-photorealistic rendering* (NPR). NPR is one of the fields of research related to style transfer. There existed several pre-neural network NPR schemes which fundamentally provided style transfer-like functionality. Below, we provide a brief overview of them.

Stroke based rendering. Stroke based rendering (SBR) is based on placing digital brush strokes on a canvas to imitate image stylization [8]. In SBR, the algorithm tries to match to the style of a given image, through iterative stroking on a canvas according to an objective function. These SBR methods, however,

lack the flexibility of *neural style transfer* (NST) based models and typically need to be deliberately designed for each style separately.

Region based techniques. Region based rendering for image stylization uses semantic information on images such as locations of certain objects to position strokes [9, 10]. Similarly, [11] uses transform image to canonical geometric shapes and manipulate these to achieve artistic style. Region based algorithms have the same limitations as SBR in terms of flexibility.

Example based rendering. Example based rendering aims to learn a mapping between content to style images using a training set of comparing image pairs [20]. However, in real world such pairs of stylized and unstylized versions of images are difficult to find. However, this method, given large amounts of training data, can generalize to many artistic styles.

Image processing and filtering. Since the styles can be structural patterns on images, image processing filters can be used as a mean for style transfer. For example, in Winnemöller et al. [21] uses difference of Gaussians for contrast enhancement to facilitate style transfer. This type of methods are relatively easy to implement but typically lack in style diversity.

2.2 Neural Style Transfer

Style can be generalized as *texture* therefore changing the style of an image can be also seen as changing its texture properties. *Convolutional* neural networks provide detailed image statistics by learning *filters* which can differentiate between content and style.

2.2.1 Single Style Models

Single style transfer is concerned with a single style image and the each style image requires complete re-evaluation or re-training of the model.

Gatys et. al. [3] proposes a method that works by matching the Gram-matrix statistics of transferred and style images. More specifically, the approach uses backpropagation to match the second order statistics of the style and transferred images. The statistics are acquired using a pre-trained VGG network.

This model uses the gram matrix which is also used in our model. [3] uses gram matrix to calculate correlation matrix of feature maps. The aim of the method is to match the gram matrix of the style image with the generated image. The gram matrix is defined as:

$$\mathcal{G}(F(I_s)) = F(I_s)F(I_s)^\intercal \tag{2.1}$$

where F is the output of the convolutional map and I_s is the style image. Here $F(I_s)$ has a dimension of $m \times n$ where m and n are the number of channels and number of pixels in each feature map, respectively.

The method utilizes iterative back propagation, for each style image. Similar to optimizing model weights; this optimization routine is typically slow. For each image, the output pixels are initialized randomly and whole process starts from scratch, making style transfer very slow in practice.

Another approach to single style method is to train a neural network to perform real-time style transfer [1, 22] (Figure 2.1). These single-style-models (SSM) are trained for a single style image and aim to perform the same transform as in [3] through a stylization network. Improvements in stylization quality are made by introduction of instance normalization (IN) [23]. Instead of batch-normalization, IN does not normalize across samples in a batch and only performs *spatial* normalization within each sample independently.

A Markov random field based approach by [24] increase texture preservation

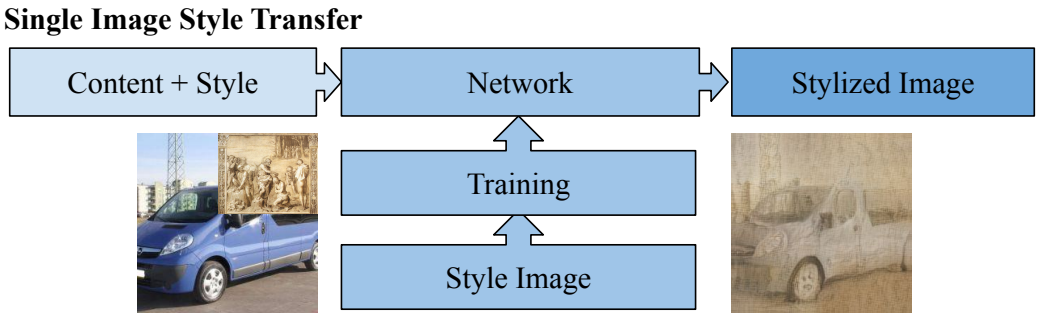


Figure 2.1: Single style transfer method based on [1] requires training a separate model for each particular style image.

over structurally complex images using adversarial training. However, this approach does not provide better results in non-texture style images, compared to baselines.

2.2.2 Multi Style Models

While some of the single style approaches provide fast stylization, they require training a separate model for each style. Generally, different style images belonging to same style group, usually share many qualities such as color palette, brush type, etc. Exploiting this phenomenon, Dumoulin et al. [25] uses shifting and scaling on the IN layer of [23] to represent up tp 32 specific styles (not style categories) using a single network. They propose a conditional instance normalization scheme to train a style transfer network. This model is also capable of linearly combining different styles. Figure 2.2 shows an example which uses parametric inputs for performing style transfer.

Chen et al. [26] uses an approach that decouples style and content by using different network modules to learn the content and style information. They use convolutional modules called *stylebanks* to learn individual styles. This approach also allows incremental training for adding more styles as the content modules could be frozen after the initial training and stylebanks are trained as usual for new styles. There is a similar model by Zhang et al. [27] that can do style transfer

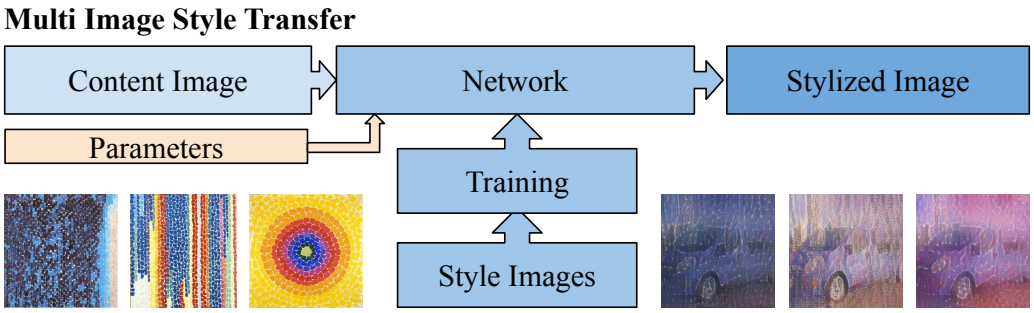


Figure 2.2: Multi image style can reuse styling learned from multiple styles and apply similar styles to content images with use of conditioning parameters.

in real time.

The main disadvantage of the aforementioned models is that the model size increases as more styles are *embedded* into the model. Aiming to tackle this disadvantage, approaches combining the image generation from content and style features are proposed. Li et al. [28] uses a model which can transfer any of the N pre-selected styles by combining the feature maps of the content and style images and passing the combined features to a decoder which creates the final image. This is similar to the work [2] and only differs in the operation which combines the style and content images. An illustration is given in Figure 2.3.

Our model is essentially a multi style model, governed by a conditioning layer. Our network can be trained with arbitrary number of style examples and style groups, without changing the model architecture except for the conditioning label input layer. The model can also be used to generate styles which are not available in the dataset as well, effectively synthesizing style, through mixing known style categories. No prior work directly aims to learn a multi-style transfer model conditioned on style category, to the best of our knowledge.

2.2.3 Universal Style Models

Universal style transfer requires a single model to perform style transfer for all possible pairs of content and style images. The premise of universal style transfer

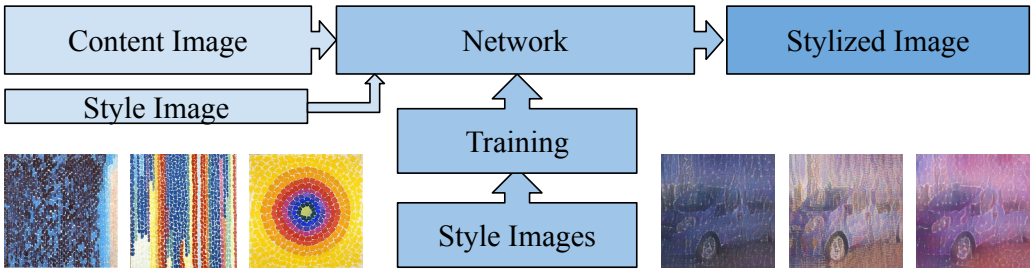


Figure 2.3: Multi image style transfer using style images as stylization input instead of parameters.

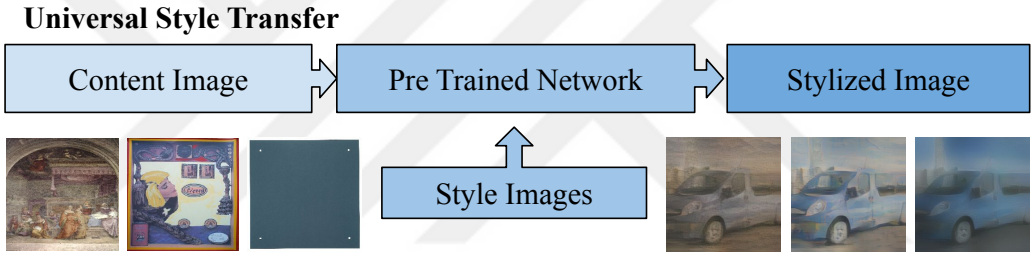


Figure 2.4: Universal style transfer uses a single trained model for all possible styles. Here images from multiple classes are successfully stylized by [2].

is shown in Figure 2.4. The first universal style model is proposed by Chen and Schmidt [29]. This method extracts activation patches for style and content images using a pre-trained VGG network. The method then swaps the content patch, which extracted from the content image with the most similar style patch extracted from the style image. This process is called *style swap*. The activation map obtained this way is then reconstructed using a model optimization or an image optimization method. This approach can transfer from arbitrary styles as the patches only extracted from the given images and there is no training involved other than the pre-trained VGG network. This model tries to optimize the similarity between style patch an content patch activations. This optimization scheme heavily biases model to preserve content over style since the style patches are selected based on their similarity to the content patches.

A method for training a universal style model for [25] is proposed by Huang and Belongie [30]. They propose to modify conditional instance normalization

into adaptive instance normalization (AdaIN). AdaIN transfers first and second order statistics between content and style images. This transferred features is then passed into a decoder to generate the final image. This method is the first method to achieve real-time universal style transfer. However, modifying feature maps using only first and second order statistics has its limits in terms of style complexity being transferred.

Li et al. [2], uses an approach similar to [30]. More specifically, instead of using AdaIN to modify the feature activations, they use *whitening* and *coloring* transforms. Basically the work shows that whitening transforms removes the style information of a given feature map obtained from pre-trained VGG activations. The whitened content image, without any style information, then *recolorized* with coloring transform using the coloring matrix extracted from the style image. This model does not suffer from the generalization limitations of [30] and can apply an arbitrary style, given the style image as an input, efficiently. The model also incorporates an α parameter to control the amount of stylization. As our work is based on [2], we provide a detailed explanation of whitening and coloring transforms in the next chapter.

2.2.4 Semantic Style Transfer

Semantic style transfer aims to form a semantic relationship between the content image and the style image and perform style transfer according to this mapping. For example, style of the eyes can be mapped to eye patches in the content image. For this reason these methods depend on region based methods. Chamapanard [31] improves upon the patch-based algorithm of [32] and creates a better semantic match between the content and style image. Here, the semantic segmentation of the images can be fed into the network manually or from a dedicated semantic segmentation network [33, 34]. It is shown that semantic information can increase stylization quality [35] by mapping semantically similar structures from style image to content image.

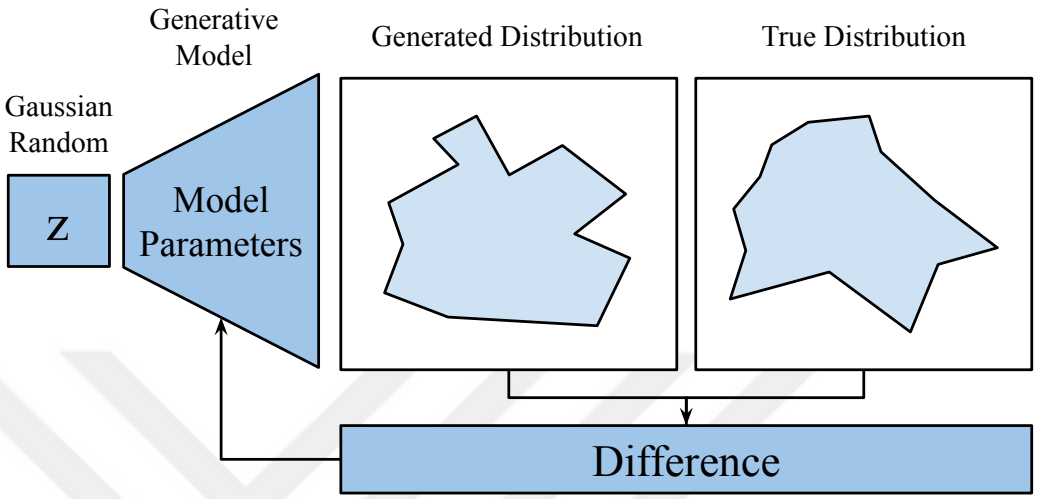


Figure 2.5: Diagram explaining implicit generative models. Purpose is to model the true data distribution using a model. Generative models try to map a (usually) Gaussian distribution to the true data distribution using parametric functions. The parameters are updated according to the loss function representing the dissimilarity between true and synthetic data distributions.

2.3 Neural Network based Generative Models

In this section, we provide an overview of prominent approaches for neural network based generative modeling. Generative modeling is one of the most important areas in artificial intelligence and machine learning research. The ability to generate novel data has been sought after by many academics over the years. A source of motivation for studying them is their practical application. Examples include image generation, super-resolution, music and voice generation. Another motivation for studying of the generative models is that they can be used to understand and model the underlying data distributions better as a generative model inherently aims to learn the manifold underlying a data distribution. In Figure 2.5 a diagram describing parametric generative models can be seen.

Generative modeling problem is an ill posed problem as the classes, latent vectors or any other source of image generation inherently have less information than the generated data. Therefore generative models are required to solve one-to-many mapping problems. Towards addressing this challenge, a number of

important models have been proposed in the past few years, such as variational auto encoders, autoregressive models and generative adversarial networks. We summarize these prominent approaches in the following sub-sections.

2.3.1 Variational Autoencoders

Variatonal Autoencoders (VAE), proposed by Kingma and Wellington [36] and Rezende et al. [37], VAEs use a similar architecture to conventional autoencoders [38, 39, 40] and try to model the data generation process. VAEs can provide better control on the statistics of latent representation of the images [41] by encouraging statistical independence. One of the setbacks of the VAE is the tendency to generate blurry or noisy images because it uses a form of least squares as the reconstruction loss, which is part of the training objective [42].

2.3.2 Autoregressive Models

Autoregressive models aim to model data based on random processes dependent on previous outputs. For example, Pixel RNN proposed by Oord et al. [43] uses the previously generated pixels in an image to guess the next pixel. This models can complete occluded images [44] or can be adapted to create high quality images [45]. A study on modeling and generating raw audio is proposed by Oord et al. which is based on PixelRNN [46]. WaveNet is fully autoregressive and outputs are conditioned on samples from previous timesteps. Another useful application of autoregressive models is in natural language processing (NLP). An attention based auto-regressive model called "transformer" [47] uses encoder and decoder structures with autoregression instead of convolution and recurrence, which was common use of language modeling.

2.3.3 Generative Adversarial Networks

Generative adversarial networks (GAN) first proposed by [19] can model high-dimensional distributions of data and suited well for complex data generation. GANs are further advanced to generate state-of-the-art results in the generative modeling domain with contributions from [48, 49, 50] and many others.

A GAN model consists of two networks which are trained against each other to perform better than the competing network. The first network is the generator $\mathcal{G}$ which aims to generate realistic data. The other one is the discriminator $\mathcal{D}$ which tries to identify between real data versus data generated by the generator.

In this process generator never sees the real data directly and it depends on the gradient flow coming from the discriminator for its model updates. Basically the discriminator can be thought as a learnable loss function which guides the generator to learn the true data distribution. GANs are architecture agnostic and they can be formed with fully connected, convolutional or other kinds of components.

Training formulation of a GAN can be denoted as solving

$$\max_{\mathcal{D}} \min_{\mathcal{G}} V(\mathcal{G}, \mathcal{D}) \tag{2.2}$$

where

$$V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim p_{data}(x)}[\log \mathcal{D}(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)))].$$

Here V is the function the models are trying to minimize and maximize which have inputs $\mathcal{G}$ and $\mathcal{D}$ denoting generator and discriminator networks. The p_{data} and p_z are the distribution of the true data and noise respectively.

A more generalized version of the training objective for GAN is given by Nagarajan and Kolter [51] as the following:

$$L(\theta, \psi) = \mathbb{E}_{p_{\mathcal{D}(x)}}[f(-\mathcal{D}_\psi(x))]) + \mathbb{E}_{p(z)}[f(\mathcal{D}_\psi(\mathcal{G}_\theta(z)))] \tag{2.3}$$

for some real valued function f. Here θ and ψ are the weights of the given generator and discriminator respectively.

As stated in [52], the goal of GAN training is to find a Nash-equilibrium (θ^*, ψ^*) for the given value function given in Eq. (2.2).

Non-conditional GANs. Early GANs used fully-connected layers to generate data. The very first GAN in [19] uses FC layers to generate images from MNIST, CIFAR-10 and TFD. Other non conditional GANs which can generate higher dimension images and 3D volumetric data [53] has also been proposed. Notably, LAPGAN [54], DCGAN [55] improve image quality with the use of convolutional layers.

Conditional GANs. Non-conditional GANs are implicitly conditioned to model a single data distribution. Therefore a new GAN model is trained for each distribution (e.g. dogs as opposed to animals). Mirza and Osindero [56] introduce conditional GANs (CGAN) to make generator and classifier conditioned on classes. This allows GANs to represent multi-modal data better. Odena et al. [57] use it for class conditional image generation.

CGANs modifies the Eq. (2.2) with additional variable y as the class of the generated data. Hence Eq. (2.2) becomes:

$$\max_{\mathcal{D}} \min_{\mathcal{G}} \mathbb{E}_{x \sim p_{data}(x)}[\log \mathcal{D}(x|y)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z|y)))] \tag{2.4}$$

so that both generator and discriminator can model classes differently.

In our study we use CGAN with Projection Discriminator introduced by Miyato and Koyama [58] as our CGAN framework for generating stylized images with different styles. Methods prior to this study, the discriminators generally concatenated the class label y to the generated data. [58] provides a model constructed from a probabilistic assumptions.

There are also works on learning attribute conditional generative models. For example, a work by Karras et al. [17] which uses a conditioned GAN to generate high quality face images. Their model allows control of the certain features such

as pose, identity, complexion.

GAN training. GAN training is a challenging task and an open research problem. Local convergence is not always possible as shown by [51, 59]. For this reason typically a collection of regularizations and architectural decisions is used to make GAN both trainable and representative of the true data distribution.

Unlike discriminative models, whose convergence can be detected by tracking loss function, convergence is difficult to detect in GANs. For simple GANs introduced in [19] convergence is generally determined by discriminator having 50% accuracy. However, in Wasserstein GANs (WGAN) [60], for example, discriminators do not provide class labels as their output is not bound between 0 and 1. In this case detecting convergence is not trivial.

There are several methods proposed to make convergence possible, faster and models more stable. Batch normalization [61] can improve GAN results [55] as it behaves more stable against changing batch statistics. A normalization technique called spectral normalization (SN) [62] is now common in training Wasserstein GANs. Spectral normalization works by normalizing the weights in a layer by its spectral norm, i.e. maximum singular value gradient of the weight in its domain. This term help the discriminator to be Lipschitz continuous [63]. As it is central for training our model, SN is explained thoroughly in Chapter 3.1.

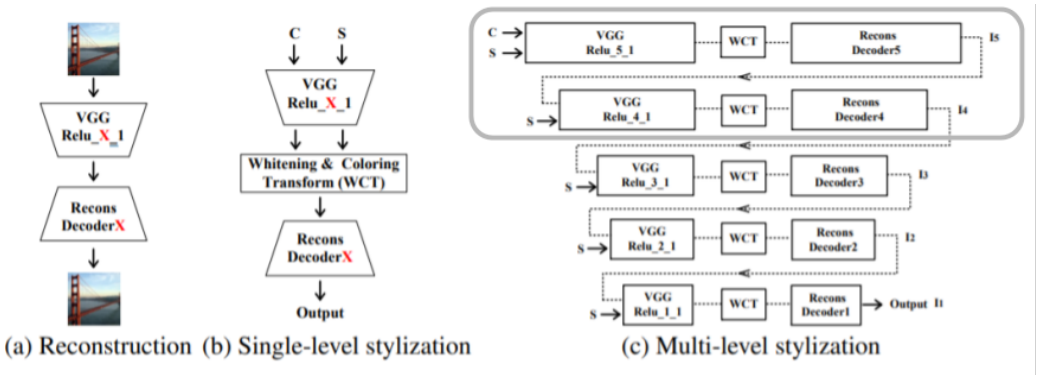
Chapter 3

Method

Here we present our method for generating images. Section 3.1 presents the preliminary methods which our method is built upon. The original UST model [2] and two statistical transforms called whitening and coloring transforms are explained extensively. We conclude the preliminaries by explaining projection CGAN and spectral normalization. In Section 3.2 we present our covariance matrix approximation method which is one of our main contributions. Finally, in Section 3.3 we describe our model architecture and discuss the effectiveness of our design.

3.1 Preliminaries

In this section the methods and techniques which are used in our model will be explained. While some background information is already given in the previous section, here we provide the mathematical details of the methods.



(a) Reconstruction (b) Single-level stylization (c) Multi-level stylization

Figure 3.1: Figure taken from [2]. (a) Demonstrates the VGG auto-encoders, trained for reconstructing a given image. (b) Shows a single-level stylization network. A content and a style image is provided to the model. The model performs style transfer using whitening and coloring transforms. (c) The VGG auto-encoders are concatenated to achieve stylization at every feature space level. The gray box highlights the VGG_4 and VGG_5 networks we use in our study.

3.1.1 Original Style Transfer Network

Our model is mainly based on the universal style transfer method developed by [2].

Reconstruction Decoders. In our model we use reconstruction decoders trained in [2]. These decoders are trained using pixel reconstruction loss, i.e. pixel-wise mean square error (MSE) and feature loss, which are based on MSE between feature maps of various levels. The decoders for this model is trained on the following objective proposed in [1]:

$$L = \|I_o - I_i\|_2^2 + \lambda \|\Phi(I_o) - \Phi(I_i)\|_2^2 \tag{3.1}$$

where I_i, I_o are input and output images for the auto-encoder. Φ is the VGG encoder and creates the feature space using Relu_X_1 layer. Additionally λ is a weight parameter to control the balance between these losses. The decoders (and encoders) are frozen after this stage. We had hardware limitations which severely hindered with experimentation. We believe that the quality of the network does not increase substantially to warrant the extra memory and computational usage. For this reason throughout this paper we select the largest two networks which are used in original paper. These are denoted as *VGG ReLU_4_1* and *VGG*

ReLU_5_1 and their decoder counterparts. These models are highlighted gray in Fig. 3.1. We call these networks VGG_4 and VGG_5. For stylization constant α we use 0.7. Using only the last two networks also makes the prediction easier as we only predict the mean values of the last 2 networks. The feature map depths of the VGG_4 and VGG_5 are $512 \times 32 \times 32$ and $512 \times 16 \times 16$ respectively for input image with size $3 \times 256 \times 256$, all written in $channel \times height \times width$ ordering.

3.1.2 Whitening and Coloring Transforms

Let I_c and I_s be the content and style images respectively. Then using the pre-trained VGG encoder, we extract f_s and f_c from the images. These are flattened feature maps of the given images at a certain level after the activation functions. In our case we choose ReLU_4_1 and Relu_5_1 of the VGG19 network from [64] for creating the feature maps. ReLU_4_1 and Relu_5_1 are the ReLU activated feature output maps after the convolutional layers conv_4_4 and conv_5_4 respectively. Both have channel depth of 512.

We use whitening and coloring transforms for extracting and applying style information on the images. The purpose of whitening and coloring transforms is to match the covariance matrix of f_c to covariance matrix of f_s [2].

For whitening transform we obtain $\hat{f}_c$ such that $\hat{f}_c \hat{f}_c^\top = I$

$$\hat{f}_c = E_c D_c^{(-\frac{1}{2})} E_c^\top f_c \tag{3.2}$$

In Eq. (3.2) D_c is the diagonal matrix comprised of eigenvalues of the covariance matrix $f_c f_c^\top \in \mathbb{R}$. E_c is the orthogonal matrix which satisfies the $f_c f_c^\top = E_c D_c E_c^\top$.

For the coloring transform inverse of the whitening is applied. The feature map of the resulting image denoted as f_{cs}. Coloring transform aims to match the correlations of f_{cs} and f_s such that $\hat{f}_{cs} \hat{f}_{cs}^\top = f_s f_s^\top$.

$$\hat{f_{cs}} = E_s D_s^{(\frac{1}{2})} E_s^\top \hat{f}_c \tag{3.3}$$

Similarly D_s is the diagonal matrix comprised of eigenvalues of the covariance matrix $f_s f_s^\top \in \mathbb{R}$. E_s is the orthogonal matrix which satisfies the $f_s f_s^\top = E_s D_s E_s^\top$ The resulting f_{cs} is re-centered using the mean vector of the style features m_s with $\hat{f}_{cs} = \hat{f}_{cs} + m_s$

3.1.3 CGAN with Projection Transform

The standard CGAN works according to the Eq. (2.4). The loss function given in Eq. (2.2) is modified into

$$\begin{aligned} \mathcal{L}_{\mathcal{D}} = &-\mathbb{E}_{x\sim p_{data}(y)}[\mathbb{E}_{x\sim p_{data}(x|y)}[\log \mathcal{D}(x,y)]] \\ &- \mathbb{E}_{z\sim p_z(y)}[\mathbb{E}_{z\sim p_z(\mathcal{G}(z)|y)}[\log(1-\mathcal{D}(\mathcal{G}(z,y)))]] \end{aligned} \tag{3.4}$$

following this, one can decompose Eq. (3.4) i.e. the output of the discriminator as log likelihoods:

$$f(x,y) = \log \frac{p_{data}(x|y)p_{data}(y)}{p_{\mathcal{G}(z)}(\mathcal{G}(z,y)|y)p_{\mathcal{G}(z)}(\mathcal{G}(z,y)} \tag{3.5}$$

$$f(x,y;\theta) = y^\top V\phi(x;\theta_\Phi) + \psi(\phi(x;\theta_\Phi),\theta_\Psi) \tag{3.6}$$

where V is the embedding matrix of y, $\phi(\cdot,\theta_\Phi)$ is a vector output function of x and $\phi(\cdot,\theta_\Phi)$ is a scalar function.

In Eq. 3.5 and Eq. 3.6, all variables denoted with θ $(\theta_\Phi,\theta_\Psi)$ and V are learnable parameters optimized in the learning process. The study uses the term projection discriminator as it uses a linear projection of the conditioning labels y instead of concatenation. The authors claim that the method allows an implicit regularization on the generator when the generator distribution and target distribution are relatively simple while acknowledging the lack of theoretical grounding. Nonetheless, their method provides better results in conditioning in ImageNet dataset [65].

Algorithm 1: Power Iteration Algorithm

Result: $\sigma(W) \simeq \tilde{u}^\top W \tilde{v}$
$\tilde{v} \leftarrow$ random matrix
$\tilde{u} \leftarrow$ random matrix
while *iteration* $<$ *max iteration count* **do**
  $\tilde{v} \leftarrow W^\top \tilde{u} / ||W^\top \tilde{u}||_2$
  $\tilde{u} \leftarrow W^\top \tilde{v} / ||W^\top \tilde{v}||_2$
  increment iteration
end

3.1.4 Spectral Normalization

Spectral normalization achieves Lipschitz continuity over the discriminator $\mathcal{D}$ by normalizing the weights. In [62] spectral normalization is used to generate the ideal discriminator theorized by [59]. Lipschitz norm $||g||_{Lip}$ is equal to $sup_h \sigma(\nabla g(h))$ where $\sigma(A)$ is the spectral norm of the matrix A. The spectral normalization is given in Eq. 3.7.

$$\bar{W}_{SN} = W/\sigma(W) \tag{3.7}$$

Since $\sigma(W)$ is largest singular value, it can be computed by singular value decomposition. However this is deemed expensive and instead the regularization method relies on power iterations [66, 67]. Algorithm 1 can compute sufficiently approximate values for $\sigma(W)$ with iteration count 1 as demonstrated in [67].

3.2 Diagonal Covariance Approximation

In our style transfer model explained later in this chapter, we develop a conditional generative model that synthesizes the statistics needed for the coloring transform. Full covariance matrix, however, is difficult to model generatively. In this sub section, we propose an approximation to the aforementioned style transfer approach that drastically reduces the dimensionality of the statistics that need to be synthesized.

We modify Eq. (3.2) such that instead of computing E_s, D_s from covariance

matrix of F_s, we only use the mean vectors of the given F_s and diagonal of its covariance matrix. For symmetry we use the same method on content features F_c when whitening.

Given F_s, an $m \times n$ matrix where m is the number of channels and n is the number of pixels in feature domain. In the coloring transform when computing the coloring matrix normally we compute:

$$\Sigma_s = (F_s - \mu_s)(F_s - \mu_s)^\top / (n-1)$$

$$U_s D_s E_s^\top = \Sigma_s$$

In our method we do not use the F_s matrix and instead directly use the diagonal of $\Sigma_{\tilde{s}}$ for SVD decomposition.

$$U_{\tilde{s}} D E_{\tilde{s}}^\top = diag(\Sigma_s) \tag{3.8}$$

Since $diag(\Sigma_s)$ has form

$$diag(E_s) = \begin{bmatrix} \Sigma_{11} & 0 & \cdots & 0 \\ 0 & \Sigma_{22} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \Sigma_{mm} \end{bmatrix}_{m \times m}$$

U_s and E_s are combination of one-hot vectors as can be seen in Fig. 3.2. In this figure we observe the D_s is relatively preserved as well as the final stylized result. The high-valued features are preserved in F_s which allows our approximation to work.

Our motivation behind this approach is to exploit the low cross-correlation between the variable of VGG. Since the non-diagonal entries in the covariance matrix denotes the covariance of two different channels, they carry some information about the distribution. However assuming independence, we can set these entries to zero. Effectively performing an element-wise multiplication between covariance and an identity matrix of the same size.

$$\Sigma_{\tilde{s}} = \Sigma_s \odot I \tag{3.9}$$

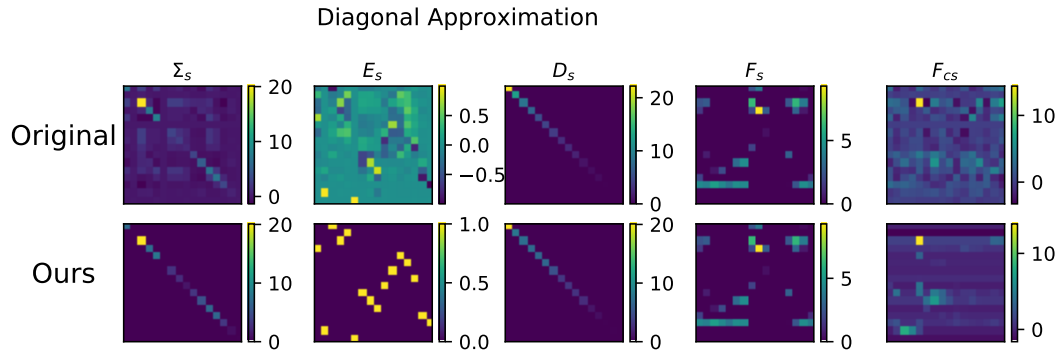


Figure 3.2: Here we observe the differences between our proposed method and the original [2]. The covariance matrix Σ_c is from a content image. The F_{cs} is obtained through coloring with a sample style image. (The matrices are cropped to 16×16 for better visualization)

We construct the square matrix $D_{\tilde{s}}$ using the diagonal entries of the Σ. Then we use $D_{\tilde{s}}$ and $E_{\tilde{s}}$ to calculate the coloring matrix S.

$$S = E_{\tilde{s}} D_{\tilde{s}}^{\frac{1}{2}} E_{\tilde{s}}^{\top}$$

Since in Eq. 3.2 we require $\mu_s = \Sigma_i^n(F_s(i))/n$ and we do not have F_s we also predict the means of F_s to complete a valid coloring transform.

Therefore instead of requiring a $m \times n$ matrix our approximation only requires two $m \times 1$ matrices for coloring. For $m = 512$ this is an n-fold decrease as in our required parameters since, $n_4 = 1024$ and $n_5 = 256$ for VGG_4 and VGG_5 decoders respectively.

3.3 Conditional Styling Generative Adversarial Network (CS-GAN)

In this section we explain our main model Conditional Styling Generative Adversarial Network (CS-GAN) and compare our method to pre-existing methods and highlight the differences of our model. We describe our complete architecture and

training method of the model. Finally, we provide an ablation study, justifying our model's design and training process.

We aim to generate stylized image given a any-hot encoding of the desired *style category(ies)*. For this purpose we propose a GAN based model which can generate $diag(\Sigma)$ and μ_s.

One of the main differences of our model compared to existing multi-style and universal style transfer approaches is that our model does not use style images directly to generate stylized images. Instead of extracting style statistics from style images we use the Σ generated by our model with the help of our approximation method. Since our network does not need to predict full covariance matrices only two vectors of size 512 is required to enable the image stylization and generation. This reduces the complexity of our model architecture drastically. The reduction in output vector enables us to generate these vectors using conventional fully connected layers instead of convolutional layers and this allows us to better model the output vector space. This is because while convolutional networks are efficient for modeling data with spatial continuity such as images, this is most probably not the case for our necessary vectors for approximating the covariance matrices.

Model Architecture. Our model uses a fully-connected architecture as opposed to convolutional architectures in [55]. The model uses multiple modern techniques for training GANs. The input of the model is a label vector denoted y which is an one-hot vector encoding one of the predetermined styles. For data diversification we use standard Gaussian noise vector z with dimensions 20×1. The network produces two vector pairs $(\sqrt{diag(\Sigma)}, \mu_s)$. Here we produce the square-root of the diagonal elements for the reasons we explain in the next paragraph. We convert $\sqrt{diag(\Sigma)}$ back to $diag(\Sigma)$ before feeding these vectors to the decoders VGG_4 and VGG_5. These decoders require the inputs $diag(\Sigma_4), \mu_4$ and $diag(\Sigma_5), \mu_5$ respectively. We have two GAN networks, one for generating the mean vectors of the style features to be used in colorizing transform and one for generating the diagonal entries of the style feature covariance matrix. Our first generator can

written as:

$$\mu_{4,5} = \mathcal{G}_{\mu}(z, y) \tag{3.10}$$

The GAN for generating the diagonal entries of the covariance matrix $(\mathcal{G}_{\Sigma}, \mathcal{D}_{\Sigma})$ has near identical architecture with the GAN generating the mean vectors $(\mathcal{G}_{\mu}, \mathcal{D}_{\mu})$. However, there is one key difference. $\mathcal{G}_{\Sigma}$ does not predict the diagonal entries of the covariance matrix instead it predicts the square root of the covariance matrices. The square root is also concave ($-\sqrt{x}$ is convex for minimization purposes) and it maps the entries in a narrower range. We also know that all entries of the diagonal of covariance matrices must be non negative since it is positive semi-definite. Similarly our second generator can be written as:

$$\sqrt{diag(\Sigma)}_{4,5} = \mathcal{G}_{\Sigma}(z, y) \tag{3.11}$$

The order of layers i.e. linear, normalization and ReLU configuration is partially inspired pix2pix network by Isola et al. [68]. Instead of using dropout for creating variance in the generation as in pix2pix, we use a similar technique that applies feature-wise linear modulation layers proposed by Perez et al. [69].

As the discriminator architecture we use the general architecture proposed in projection GAN [58]. Our discriminator is similarly composed of fully connected layers. We use three fully connected layers with Leaky ReLU activation. Similar to the generator we use skip connections for better gradient flow. As projection for the conditioning labels, we use a linear layer instead of embedding matrices proposed in original paper [58] as they're equivalent with linear layers having the advantage of representing any-hot vectors with continuous values. We use spectral normalization and batch normalization in the network.

In both networks we use Leaky ReLU proposed by Maas [70] which is built upon ReLU [71] with leak coefficient of 0.2. Leaky ReLU provides advantages against dying ReLU problem. Also for better gradient flow, we use skip connections between Linear-Normalization-ReLU modules which perform well as demonstrated in [72].

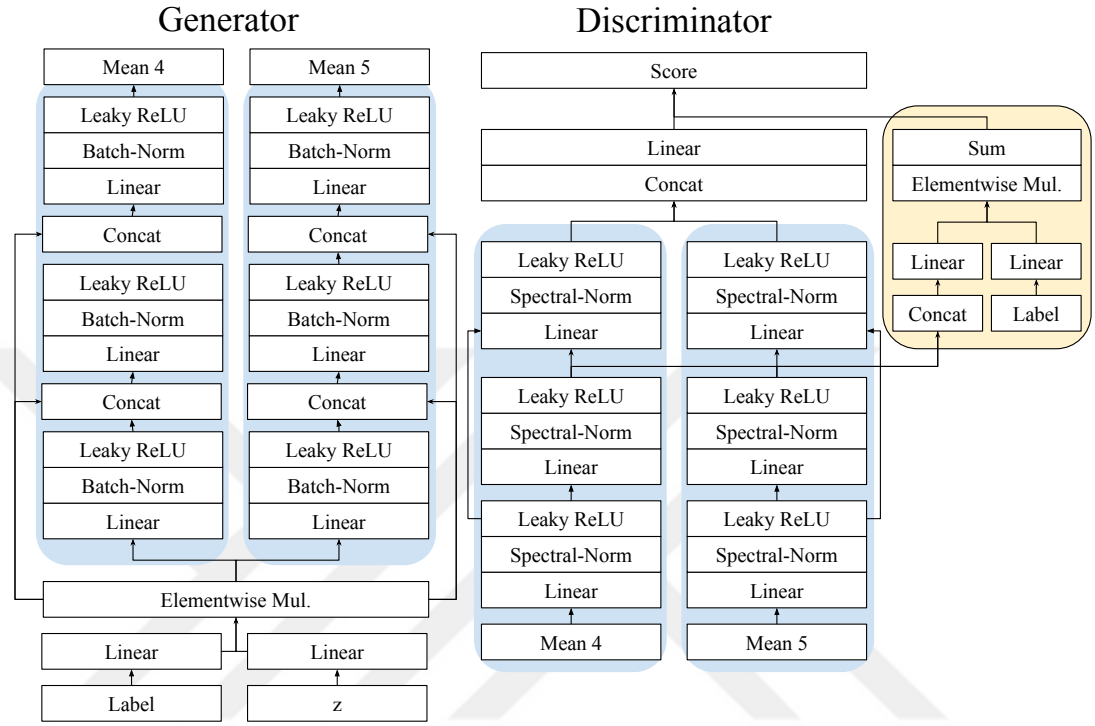


Figure 3.3: Architecture of the GAN generating the means. **Left.** Generator network. Notice the symmetrical generation for both VGG_4 and VGG_5 means. **Right.** Discriminator network. The projection conditioning is denoted by the yellow area.

For training both G_Σ and G_μ generators we use the same discriminator architecture. For both of our GANs the discriminators can be written as

$$score_\mu = \mathcal{D}(\mu_{4,5}, y), \;\; score_\Sigma = \mathcal{D}(\Sigma_{4,5}, y) \tag{3.12}$$

The architectures of the both G_Σ and G_μ networks are same, in Fig. 3.3. We can see the details of the architecture for the generator and discriminator for the means. ($\mathcal{G}_\mu$ and $\mathcal{D}_\mu$)

Model Training. We train our models using Adam optimizer proposed by Kingma and Ba [73]. For our optimizer we select the parameters β_1 and β_2 as 0.5 and 0.999 respectively. We utilize the two time-scale update rule (TTUR) and select different learning rates for generator and discriminator for better optimization [74]. We also use learning rate decay with decay constant of 0.999 per epoch.

Also we use different update-per-epoch for generator and discriminator. We use the historical buffer used by Shrivastava et al. [75]. This method forces discriminator to remember failing modes of the generator and helps avoiding the mode-collapse. History buffer was one of the most effective ways to combat mode-collapse in our networks and is very straightforward to implement.

As the loss function for GAN training we use hinge loss described in Eq. (3.13) and (3.14) for the discriminator and generator respectively.

$$L_{\mathcal{D}} = ReLU(1 - \mathcal{D}(p_{data}, p_y)) + ReLU(1 + \mathcal{D}(\mathcal{G}(z, y), y)) \tag{3.13}$$

$$L_{\mathcal{G}} = -\mathcal{D}(\mathcal{G}(z, y), y) \tag{3.14}$$

Hinge loss terms are robust to outliers in the dataset as they are clamping the minimum losses to zero with ReLU layers. They are shown to stabilize the training process with HingeGAN [62]. Our method for training is shown in Algorithm 2. For our training we set $k = 2$.

Algorithm 2: Random Batch Algorithm

```
Result: G(z, y)
G ← initialize weights
D ← initialize weights
HistoryBuffer ← G(z, y)
while epoch < max epochs do
    FakeData ← G(z, y) + HistoryBuffer
    optimize(G(z, y))
    for i=0 → k do
        optimize(D(RealData, FakeData))
    end
    HistoryBuffer[RandomIndex] ← FakeData[RandomIndex]
    epoch ← epoch + 1
end
```

Chapter 4

Dataset and Experiments

In this chapter we provide the details of our experiments results and their respective discussions. Firstly, we explain our training process. Secondly, we discuss our the evaluation methods. Finally, we present the experiments we conducted.

4.1 Style Dataset

In the experiments we use the WikiArt dataset, which consists over 126K entries. The dataset contains images and their meta-data, which includes basic information such as its name, author and year as well as manually annotated qualitative features such as style, genre and technique.

The images collected from 10 styles which consisted at least 1000 entries per style. From these we generated a train and test split with ratio of 0.8 and 0.2 respectively. The styles are chosen from images which consists of unique styles, i.e. none of the images in the dataset has multiple styles. We denote this dataset as ***S10-1000***. In Fig. 4.1 the distribution of the dataset can be seen. Also in Table 4.1 we provide number of samples in each selected style. Fig 4.2 shows sample images from the selected styles.

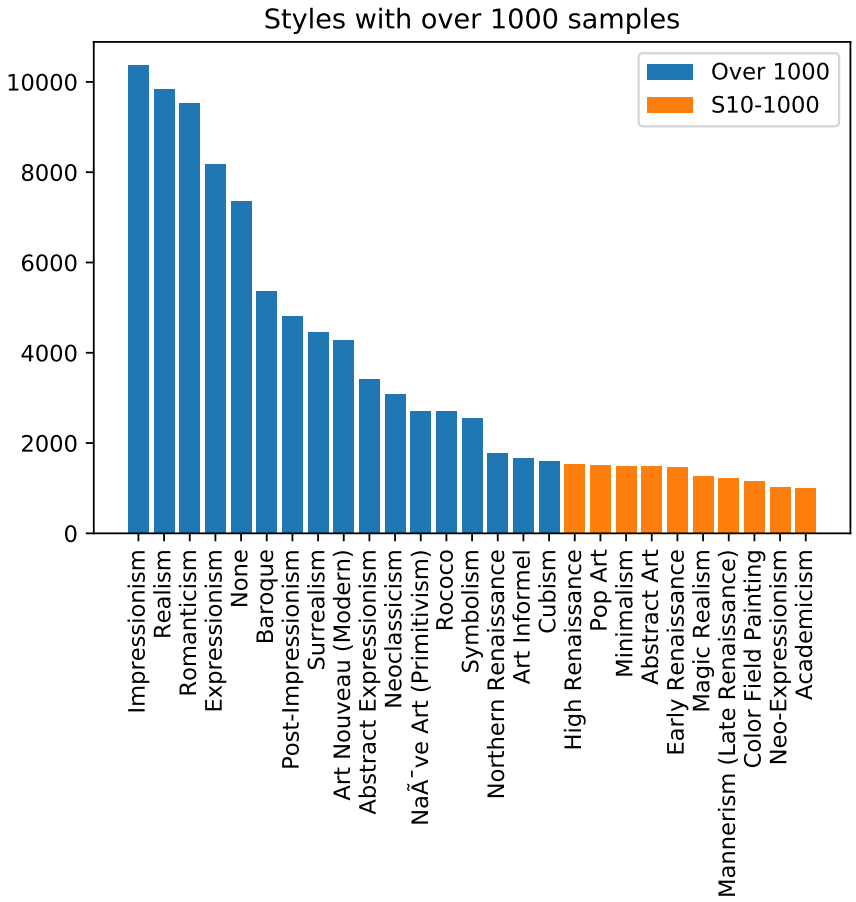


Figure 4.1: The distribution of styles with sample larger than 1000.

4.2 Training Process

We train our model on GTX 1070 implemented in PyTorch 1.2.0 [78]. We use a batch size of 1024. In Fig. 4.3 are the loss function for generator and discriminator. We also keep track of the MSE between generated means, however this is actually a heuristic metric since generator does not have any information on style image it should transfer. We relied on MSE for detecting gradient explosions where model completely fails to generate a distribution similar to the original data. Even though there are methods which use MSE for enhancing GAN training [76], we do not use MSE in our training process.

In training we used TTUR which smooths the training process and leads to a

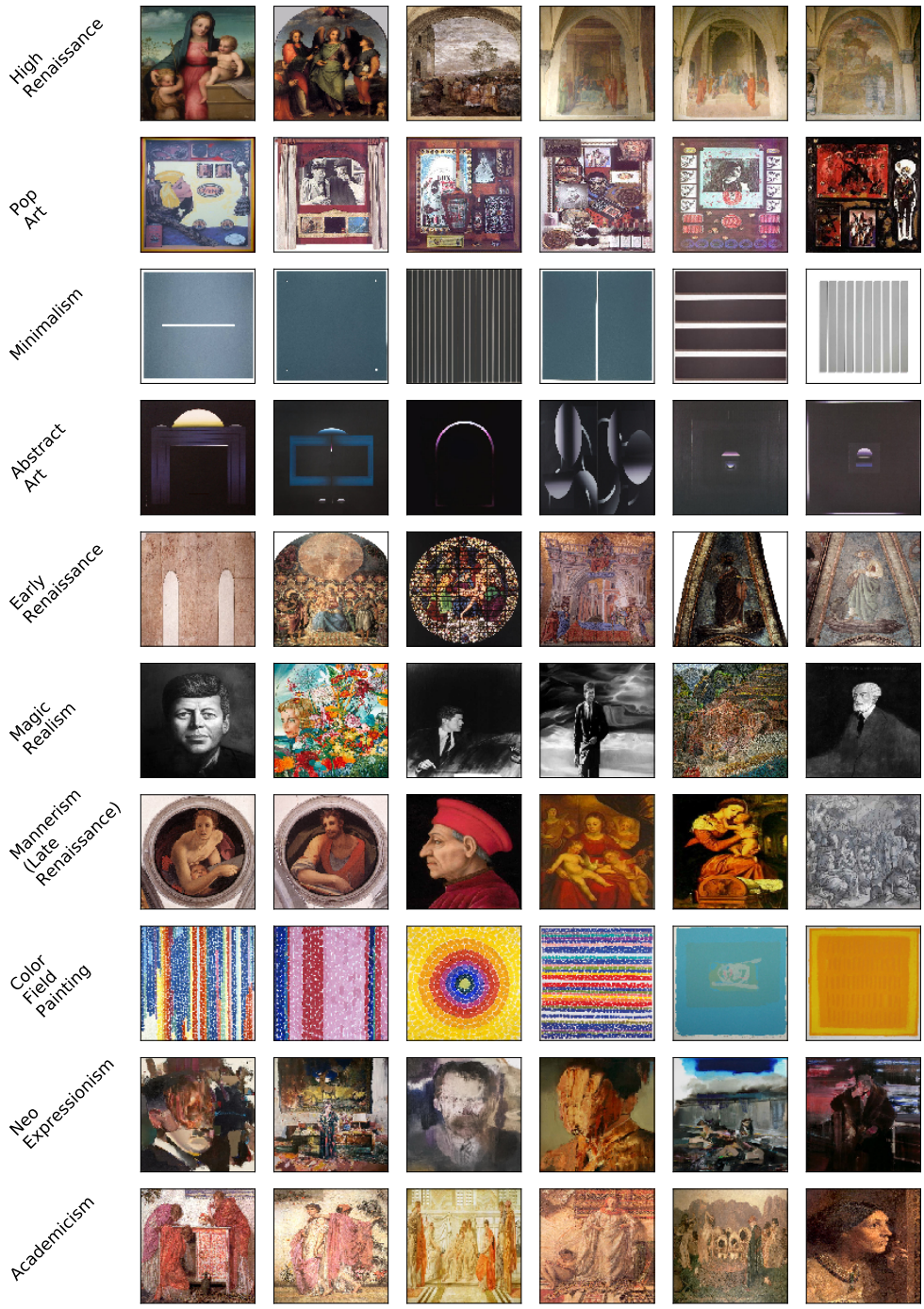


Figure 4.2: Characteristic images from selected styles. Half of the styles belong to single era, namely Renaissance. The other half, i.e. neo-expressionism, color field painting, pop art, magic realism and minimalism, are more modern. This selection is on purpose to demonstrate the ability of modeling both similar and distinct styles.

Table 4.1: The styles in the S10-1000 dataset and their respective counts.

Style	Count
High Renaissance	1554
Pop Art	1517
Minimalism	1499
Abstract Art	1496
Early Renaissance	1468
Magic Realism	1266
Mannerism	1228
Color Field Painting	1151
Neo-Expressionism	1013
Academicism	1003

more stable convergence. In both figures the discriminator losses do not increase showing that the discriminator is able to assign real-fake values correctly. TTUR is required so that the generator can only make smaller updates cannot escape the discriminators effective domain. This leads to better training of the discriminator as it allows more time for the discriminator to learn previous generator failure modes.

In Fig. 4.3 we also test the effectiveness of using spectral normalization as well. The training without spectral normalization causes instabilities in the training and generator non-convergence as predicted by [67]. The computation cost introduced by the spectral normalization was undetectably small since our network only uses few linear layers.

We experimented with different batch sizes and found that the batch sizes between 128 and 1024 leads to better convergence. We also used relativistic losses for our network which proved effective in many GANs yet these led to non-convergence in our experiments [77].

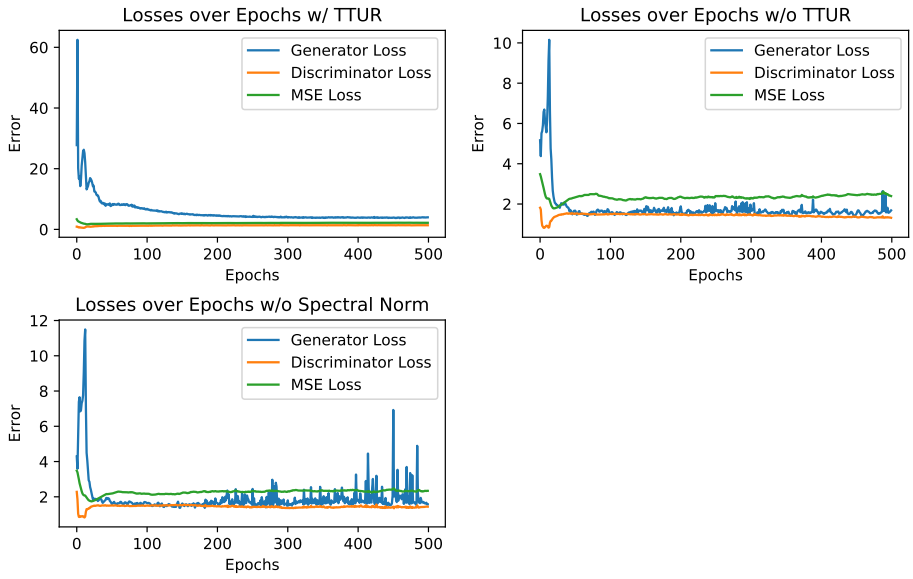


Figure 4.3: The training losses over 500 epochs. **Top Left.** Training loss with TTUR used, learining rate set to 10^{-4} for generator and 10^{-3} for discriminator. **Top Right.** No TTUR used, with both learning rates set to 10^{-4}. **Bottom Left.** Training without spectral normalization, notice the instabilities in the generator loss.

4.3 Experiments

We demonstrate the capabilities of our model through experiments. We first detail the evaluation methods namely the Frechet Inception Distance (FID) and then compare the original method to our approximation to demonstrate its effectiveness. We then compare our complete model to approximation. Finally we demonstrate the multi-style capabilities of our network.

4.3.1 Evaluation Methods

The true evaluation metric for style transfer is still one of the open questions of neural style transfer. There are two possible ways to evaluate transfer quality, qualitative and quantitative. Qualitative methods rely on human judgment and

can be subject to many external factors, such as age, gender and state of the mind of the observer at the time of observation. Accounting these external factors frequently require large population samples which may not be feasible. Quantitative evaluation depends on proposed metrics for images such as performance based on time complexity, absolute duration per image and distance between activation maps.

Evaluating GANs is yet another challenge as many GANs are constructed from unlabeled data and lack the necessary classifiers to evaluate them. One of the proposed methods is the Inception Score [79]. Inception score uses the *Inception-v3* model proposed by Szgedy et al. [80]. In inception score, images with meaningful content which also have a conditional label distribution $p(y|x)$ with low entropy, gets a higher score [79]. This means that an image generated in one class should have higher probability assigned to it by a classifier for being in that class versus other possible classes. An image with large difference between highest probability class and lower probability classes is considered a better, more realistic image.

Another requirement for high quality GANs is the intra-class variance. This requires the integral of the marginal probability distribution to have a high entropy [79]. The generated images belonging to a class should have similar probability of being in the same class.

Combining these requirements forms the following metric:

$$IS = \exp\left(-\mathbb{E}_x\left[D_{\mathrm{KL}}(p(y|x)||p(y))\right]\right) \tag{4.1}$$

The exponentiation makes IS of different models easier to compare. Although it can provide a good metric to measure realistic images, it has setbacks which disallows our use of IS. Image must belong one of the classes inception model is trained on. In WikiArt dataset, such classes does not exist as many images contains several classes.

We therefore use the Frechet Inception Distance proposed by Heusel et al. [74]

based on the original inception score. FID is given by

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||_2^2 + Tr(C + C_w - 2(CC_w)^{\frac{1}{2}}) \tag{4.2}$$

Here (m, C) is the Gaussian obtained from the model data distribution $p(\cdot)$ and (m_w, C_w) is the Gaussian obtained from the real data distribution. Specifically m and m_w denote the feature-wise means of the generated and real data respectively. C and C_w are the covariance matrices obtained from the respective data. However, FID needs at least channel-size dataset to work. This means it requires 2048 images for each class. However our test dataset contains fewer than 300 images per class for this reason we use a shallower pooling layer from the *Inception-v3* instead of the final pooling layer. We also believe that earlier layers can capture image stylization better and thus use the first pooling layer with 64 channels. This makes our FID scores incomparable with other studies, however it provides a reliable benchmark for both [2], which we consider as a baseline, and our work.

4.3.2 Baseline FID Measures

Since we do not have an FID based baseline for [2], we create our own baseline and compute a distance matrix for the images generated in the original model.

Experimental Setup. For our experimental setup we use S10-1000 dataset to stylize images based on all available style images, then compare their cross-class FID scores. We use a single content image to eliminate randomness in the measures. The FID is obtained through the first max pooling features as opposed to final average pooling features in [74].

Results. Mean FID difference between classes as shown as a matrix in Fig. 4.5. Sample outputs from the reduced model can be seen in Fig. 4.4.

In Fig. 4.5 we can see that more dissimilar styles have higher FID values such as early renaissance with low color contrast versus the minimalist case where color have higher values. Also the palettes of the renaissance era styles are restricted with their materials in contrast to more modern styles such as minimalism and

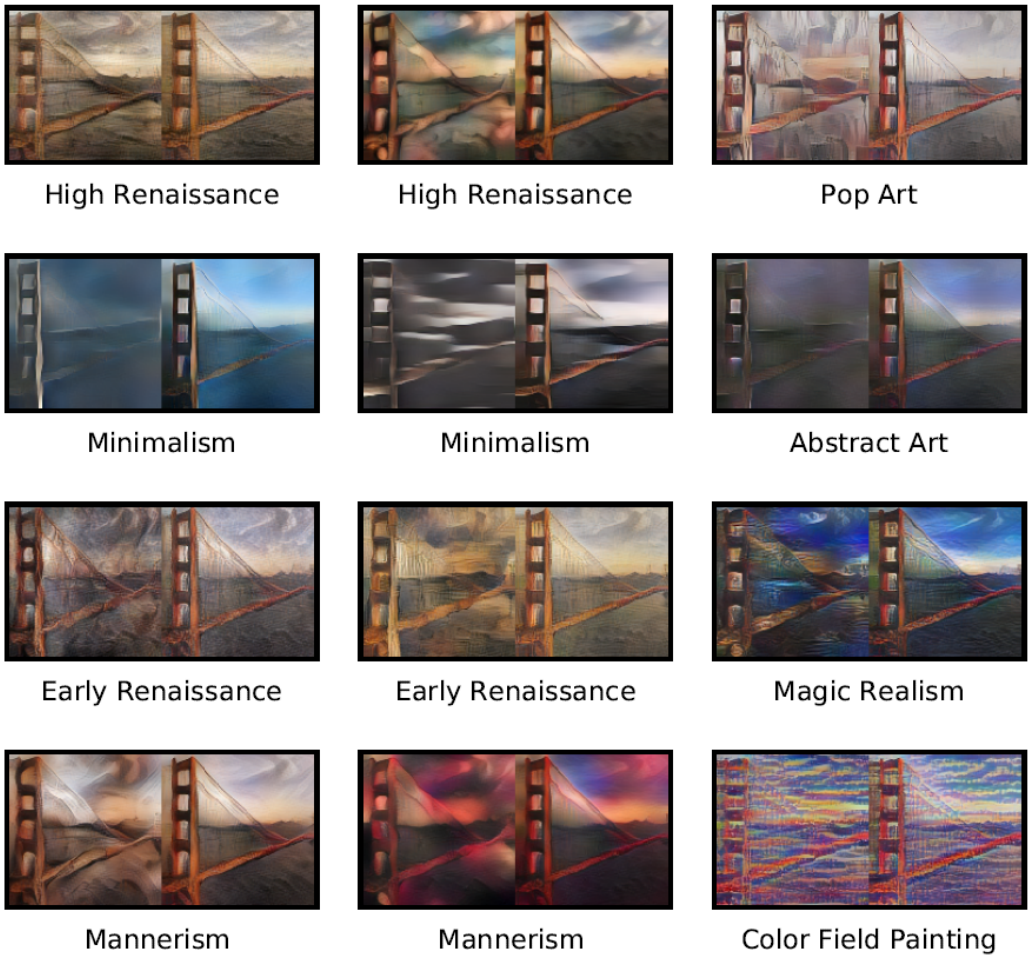


Figure 4.4: **Left.** The images generated by the [2]. **Right.** Images generated by the diagonal approximation for all styles in S10-1000. Only VGG_4 and VGG_5 are used.

color field painting. Also as we use the approximation of the covariance matrix i.e. its diagonal, there are some differences in the FID distances. However, as can be seen in Fig. 4.6, the difference in quality is perceptually low.

4.3.3 Diagonal Stylization

Experimental Setup. In the first experiments we compare our style mean shifting algorithm with the original algorithm provided in [2]. We directly feed

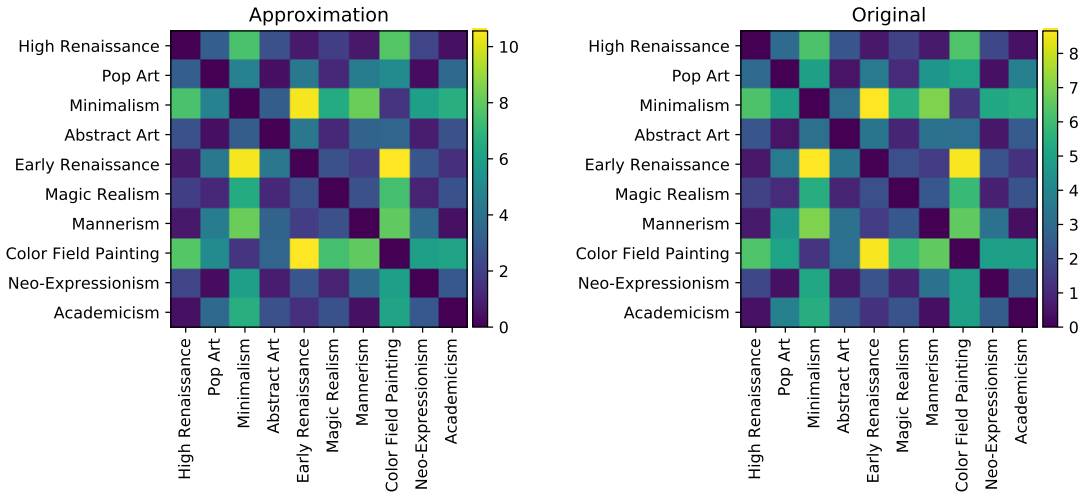


Figure 4.5: **Left.** The output of the FID matrix for approximation model. Highest FID belongs to the classes Early Renaissance and Minimalism. Highest distance is 10.56. **Right.** Same FID matrix of the original model. Notice the similarity.

Table 4.2: FID Distances between models.

Model	**Against**	**Average FID**
Original	Diagonal Only	0.36
Original	CS-GAN	1.76
Original	Random	4.4

the style image features into the original model and our network with the real-mean array we obtained form pre-trained VGG encoder. In this experiment we use the truncated model (i.e. $dec_{4,5}$) which contains only the largest 2 of the 5 networks in [2].

Results. In Fig. 4.7 we see that our method approximates the stylization quite well and only has distance differences lower than 1. Our approximation models the stylization process well. However especially in Minimalism and Color Field Painting, our model performs worse or different than modified [2]. There are several reasons for this. First of all minimalism as a style encompasses many different styles of many distinct authors. Older art tend to more scholastic in the sense that practitioners of a certain style had small variance. The other reason is the usage of with many different color palettes which effects the output of VGG

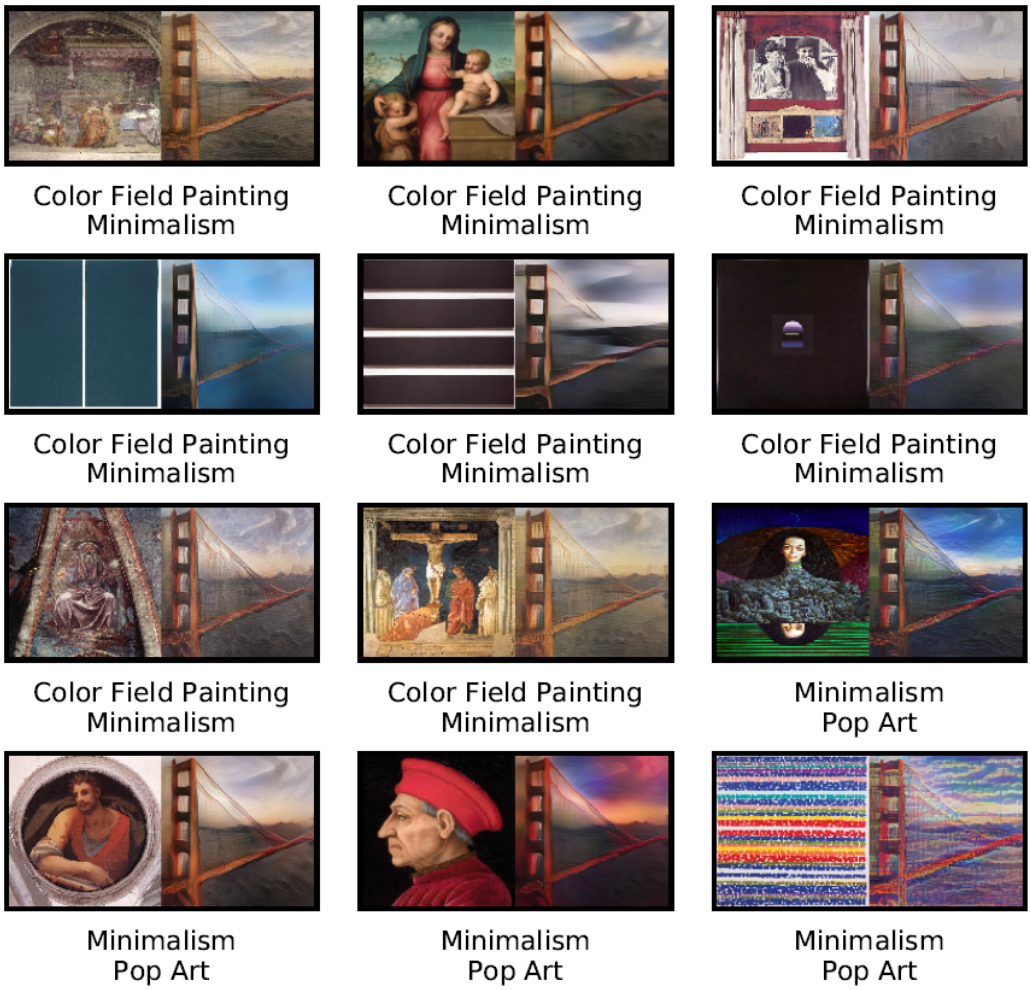


Figure 4.6: The images generated by the modified version of the [2] for all styles in S10-1000. Only VGG_4 and VGG_5 are used.

model significantly.

4.3.4 CS-GAN

Experimental Setup. For the CS-GAN experiment we generate samples equal to the number of samples generated by our test images. Among different snapshots of the trained model over epochs, we picked the model with the least generator loss for generating images.

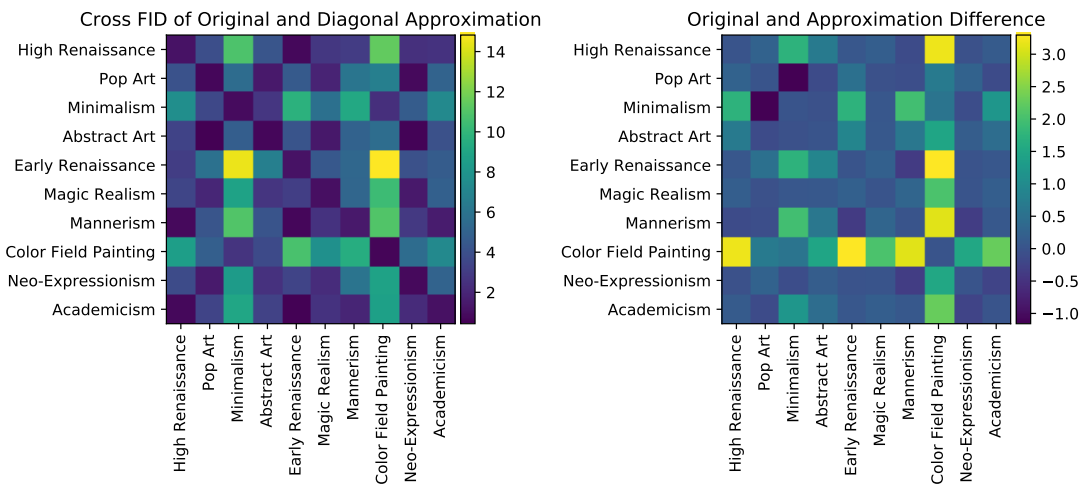


Figure 4.7: The output of the FID difference matrix for stylized and generated images. **Left.**The output of diagonal approximation model. **Right.** Subtraction of Original FID matrix from diagonal approximation matrix.

Results. In Fig. 4.9 we see that the model creates a FID matrix similar to both original and diagonal approximation models. Further analysis on the matrix shows that the CS-GAN FID matrix is approximately linearly scaled version of original FID matrix with scale factor 1.65. We can also see that our model had more problem imitating the Minimalism. For example in our model Minimalism and Pop-Art are less distinct from the original model.

Overall our model can recreate the stylization to achieve similar FID matrix. Also as seen in Table 4.2, the method approximates the distribution achieved by the original model as FID implies. In Fig. 4.8 examples show that our model can indeed recreate some of the stylizations present in the test dataset effectively. We also observe a very diverse image generation which implies that our model can generate a wide variety of images and does not suffer from mode collapse.

The diverse images generated can lie outside of the standard [0..1] range for RGB images, this causes some amount of distortion as RGB's max out and create visually unappealing artifacts in the output image. We have tried to map output images to [0..1] yet this caused a drastic reduction in image contrast and reduced visual quality further. For this reason we present our results without mapping them to the standard RGB range while using CS-GAN generated stylizations.

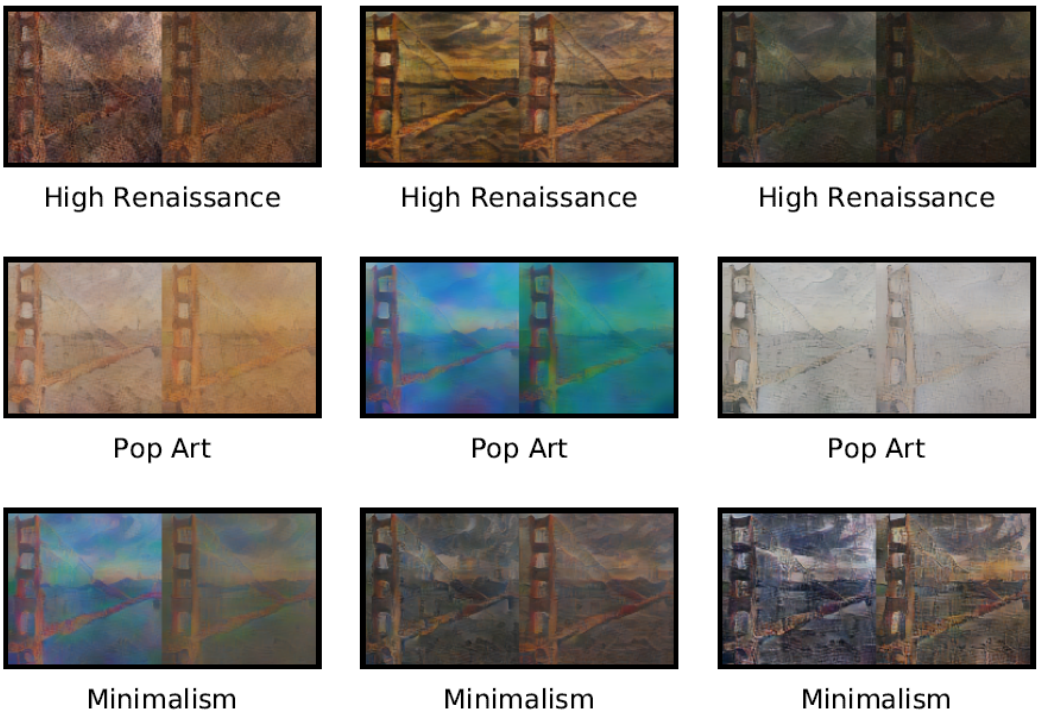


Figure 4.8: **Left.** Single style images generated by CS-GAN. **Right.** Images with closest means and covariance diagonals from the test dataset .

It should be noted that quality of our network, CS-GAN can be improved by using the non-truncated version of the autoencoder in [2].

On the performance side, we compare the original style transfer model's average image generation speed versus CS-GAN. As seen in Table 4.3 there is a 32% improvement as seen in Table 4.3. The absolute performance boost actually increases as we add more VGG layers to the auto-encoder architecture. This gain originates from the absence of computation for style features. The encoder does not have to generate style features for each image.

4.3.5 CS-GAN Multi-Hot

Our conditional GAN can take multi-hot labels as inputs as well. We perform this experiment to measure the models ability to combine different styles to form

Table 4.3: Performance measures of original and our model.

Model	Time	Improvement
Original	1.717 ms	-
CS-GAN + Decoder	1.295 ms	32%
CS-GAN	0.005 ms	-

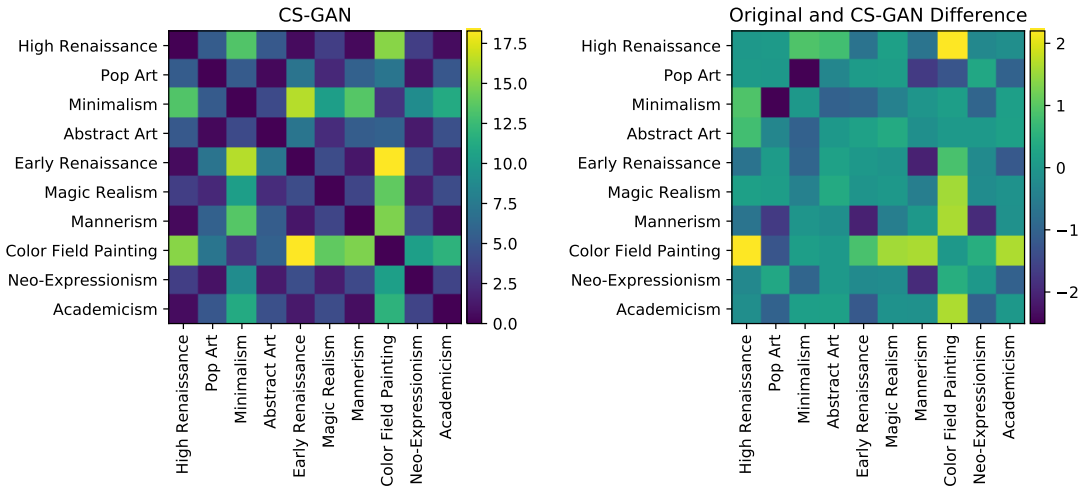


Figure 4.9: **Left.** Output of the FID matrix for CS-GAN. Observe the similar stucture with Fig. 4.5. **Right.** The FID differences between CS-GAN and original model. The original model is scaled by 1.65 to center the distances.

genuine styles.

Experimental Setup. This experiment uses the same CS-GAN trained for one-hot experiment. For evaluation we search the WikiArt for images with two styles. These two styles must be already included in our S10-1000. Given these conditions we find two multi-style groups, Color Field Painting-Pop Art and Minimalism-Pop Art which contains 28 and 11 images respectively. We then combine the one-hot labels with addition to create multi-hot vectors. We do not average the multi-hot vectors to $||y|| = 1$ as this creates very low contrast images.

Results. The images generated by the resulting means of multihot vectors is shown in Fig. 4.11. While the general variety of the created images are still preserved, the style images have less correspondence to the image with the closest mean. This can be attributed to the lack of real samples. The model can combine

Figure 4.10: CS-GAN outputs for multi-hot encodings. Each image pair has an image generated by our model(Left) and its closest mean in the test dataset(Right). **Top Left Half.** Color Field Painting and Pop Art. **Bottom Right Half.** Minimalism and Pop Art. Note that the two styled samples have very low presence in the dataset.

multiple styles together to form a novel style which is previously unknown to the model.

In some stylings the stylized image has max outs in some pixels, this is partially caused by feeding a conditioning label with higher element sum, i.e. higher norm. Normalization of the input label reduces the contrast of the styling and creates lower quality images. We believe that if the network was trained with multi-hot embeddings with internal L1 or L2 normalization, it could better preserve the image contrast at reconstruction. However such training were not possible because of the dataset limitations.

In Fig. 4.12, we present the progressive style transfers to highlight the effects

of combining styles. For this particular images we fix the noise z for all examples to suppress the variance caused by different noise samples. One important thing to note is that the images do not combine linearly and instead generate novel stylizations. This effect can be observed in first and last rows of the figure. This figure is a qualitative assessment of our model in terms of generating novel styles.

For more examples see Appendix A.

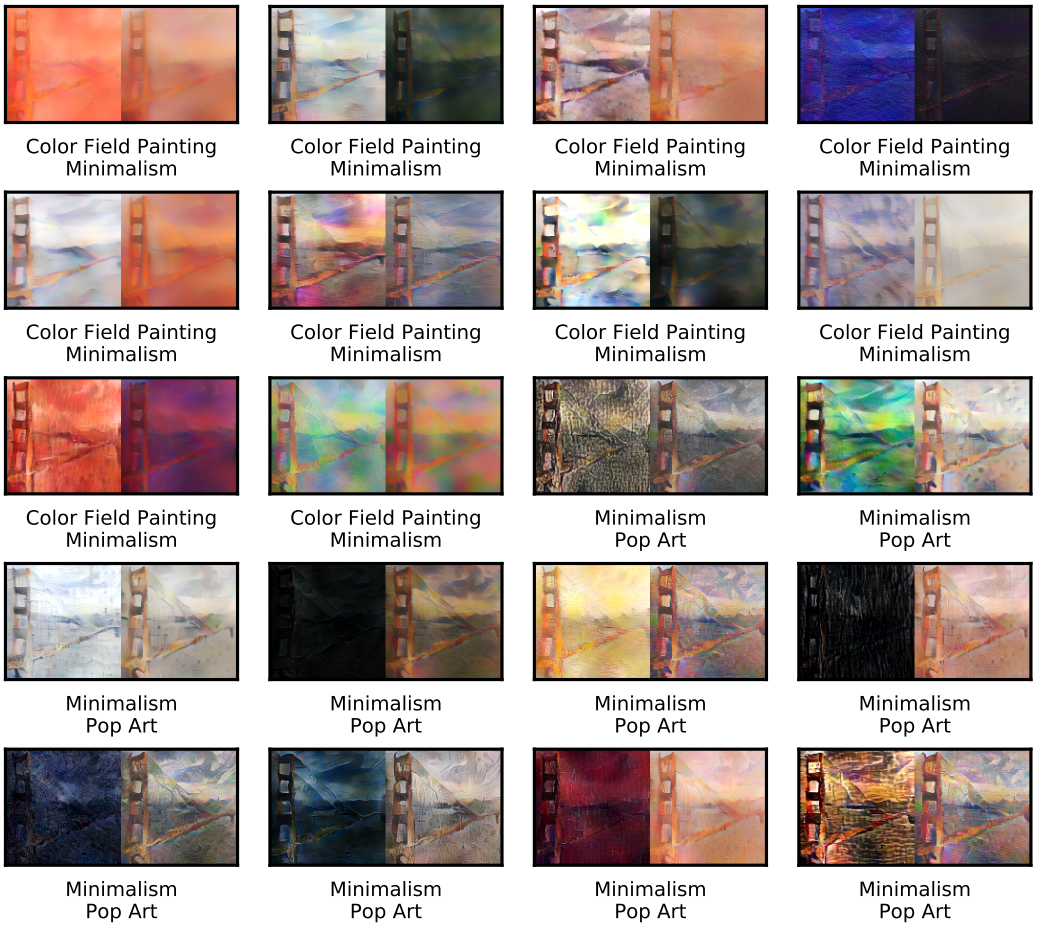


Figure 4.11: More CS-GAN outputs for multi-hot encodings with a different content image. Each image pair has an image generated by our model(Left) and its closest mean in the test dataset(Right). **Top Left Half.** Color Field Painting and Pop Art. **Bottom Right Half.** Minimalism and Pop Art. Note that the two styled samples have very low presence in the dataset.

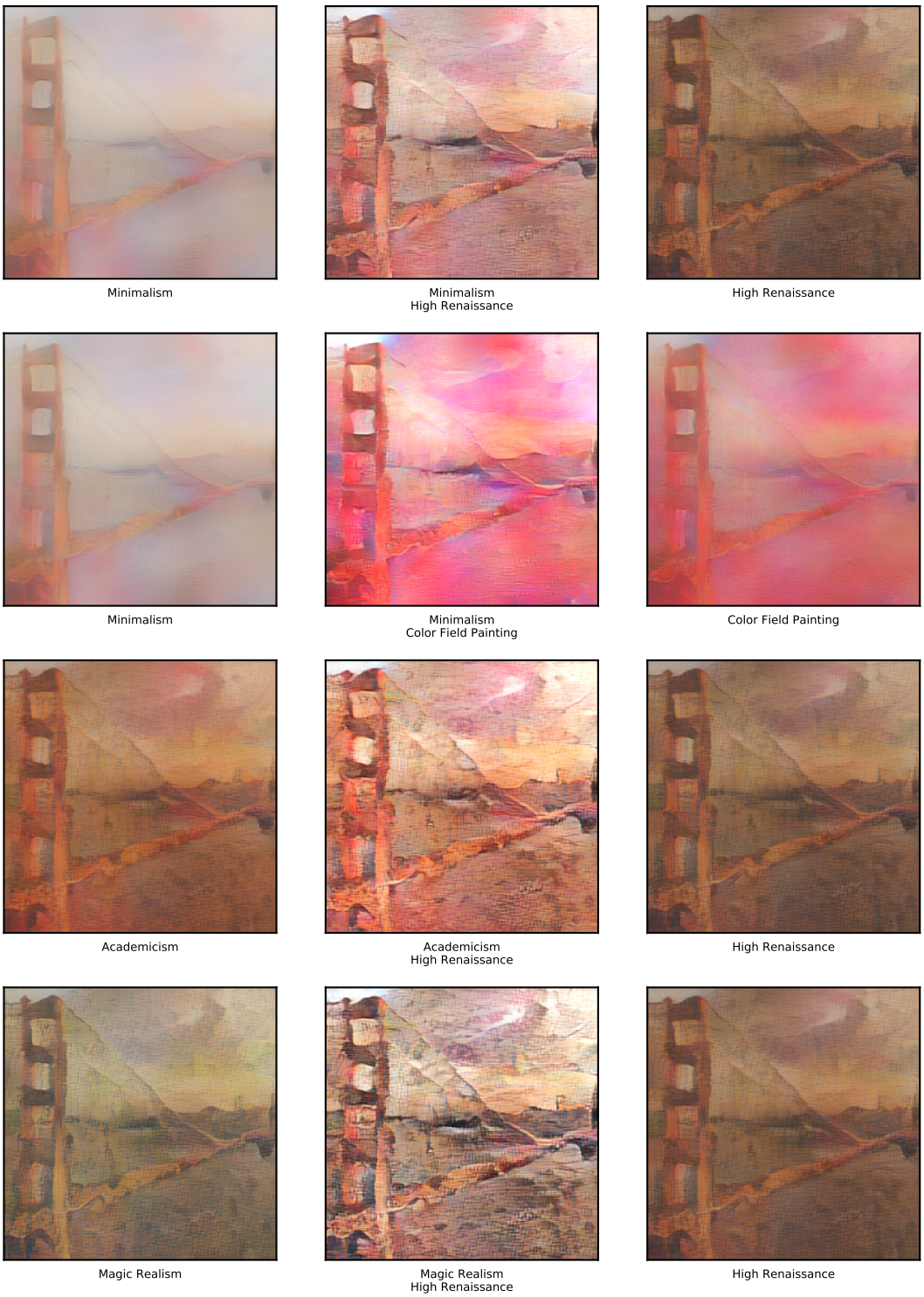


Figure 4.12: Image stylizations with fixed noise z for different style combinations. Notice that the styles combinations are not mere linear combinations of two images.

Chapter 5

Conclusion

In this chapter, we briefly summarize the main results and implications of this study. We also provide a brief discussion of possible future study directions.

5.1 Discussion

In this thesis, we have proposed a framework for universal multihot-to-style transfer. The framework uses a GAN based style-transfer network for universal style transfer. Our model directly infers correlation matrix which will be used for style transfer from the one-hot vectors itself. The model draws many inspirations from previous works. Building upon these features we have been able to provide a cross domain model, which can synthesize stylized images from one-hot or multihot style category inputs.

We have shown that our multi-style model can also create novel stylizations by combining conditioning labels. This flexibility allows our model to generate outputs involving niche styles. Its generative nature also enables the model to generate styling which does not have a real life correspondence and can be used in creative applications.

While our model can create visually comparable images, we have observed a discrepancy in terms of FID scores, compared to the original model. We believe that this discrepancy between original and ours stems from our approximation method. While our method enables GAN training by reducing complexity n-fold, we clearly loose style information partially. This trade-off, however, enables us to train our generative model successfully without observing (significant) GAN training problems, such as instability.

Overall, our network provides a framework using which style transfer can be controlled through semantic input from the user. Another contribution of our model is providing an approximation method, which is efficient and quality preserving.

5.2 Future Work

We believe there are at least two main ways in which we can improve this work in future work. First is designing and training a GAN which can better approximate the covariance matrices of the style images. For this purpose convolutional GANs can be used to generate data in higher dimensions. Other means of approximating the full covariance matrix can be done through low rank matrix multiplication methods. To this end, one method that we have planned but eventually did not include in the final version of the model is the following:

$$F_{\tilde{s}} = \mathcal{G}(z, y)$$

$$\Sigma_{\tilde{s}} = F_{\tilde{s}} F_{\tilde{s}}^{\top}$$

where $F_{\tilde{s}}$ is a low rank approximation of F_s e.g. for $F_{s\ m\times n}$ we can generate $F_{\tilde{s}\ m\times j}$ where $j << n$.

A second improvement to our style transfer model can be made by using better semantic conditioning vectors where styles can be represented densely. This would allow conditioning vectors created from word2vec embeddings and could give finer control over the styles. Such an approach can also improve the ability to generate

novel styles creatively, using semantic inputs. For a finer control, however, domain specific corpus collection might be required and our network should be adapated to accommodate a higher number of styles.

Appendix A

Style Transfer Examples

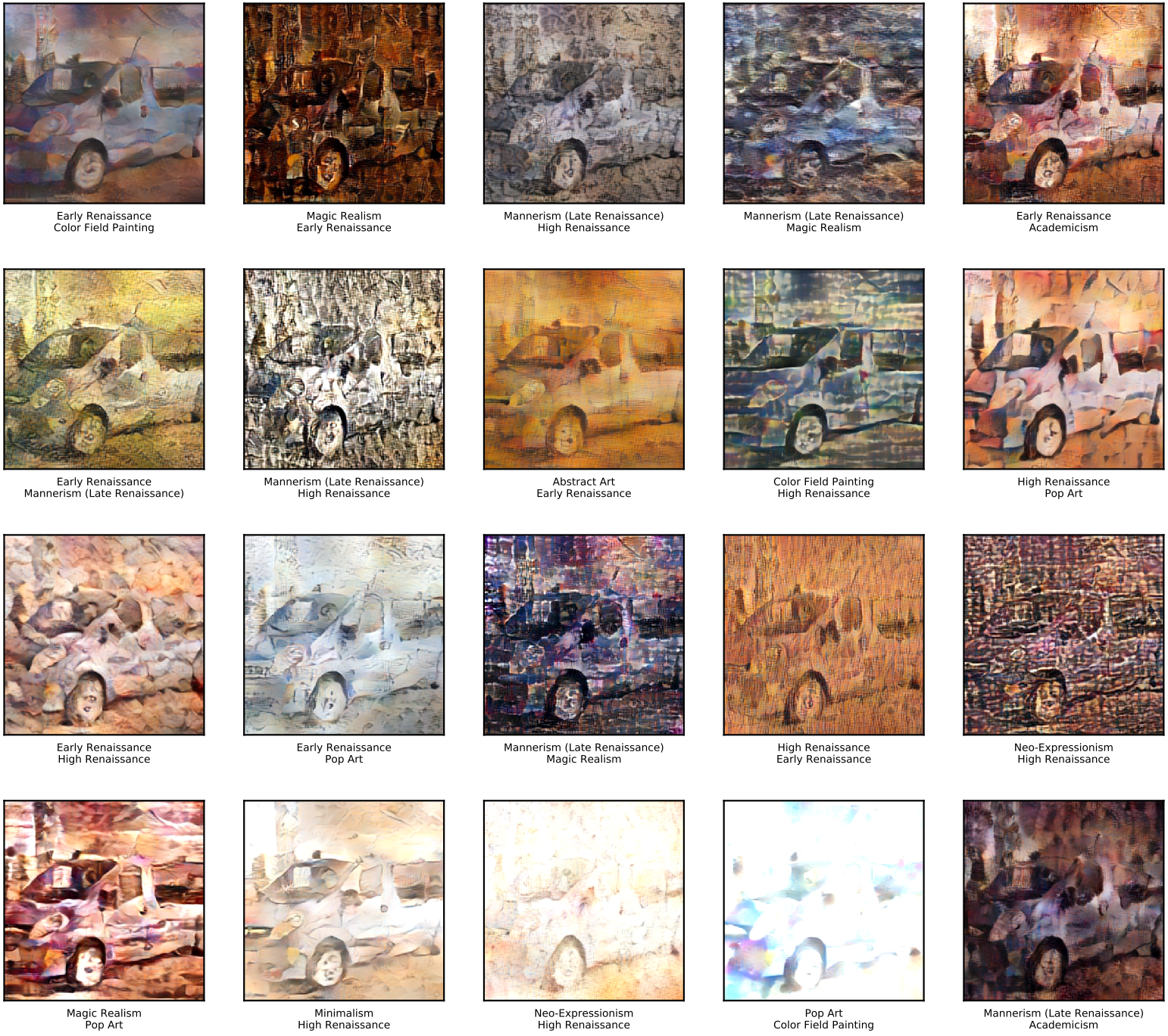


Figure A.1: Randomized multi style outputs for CS-GAN.

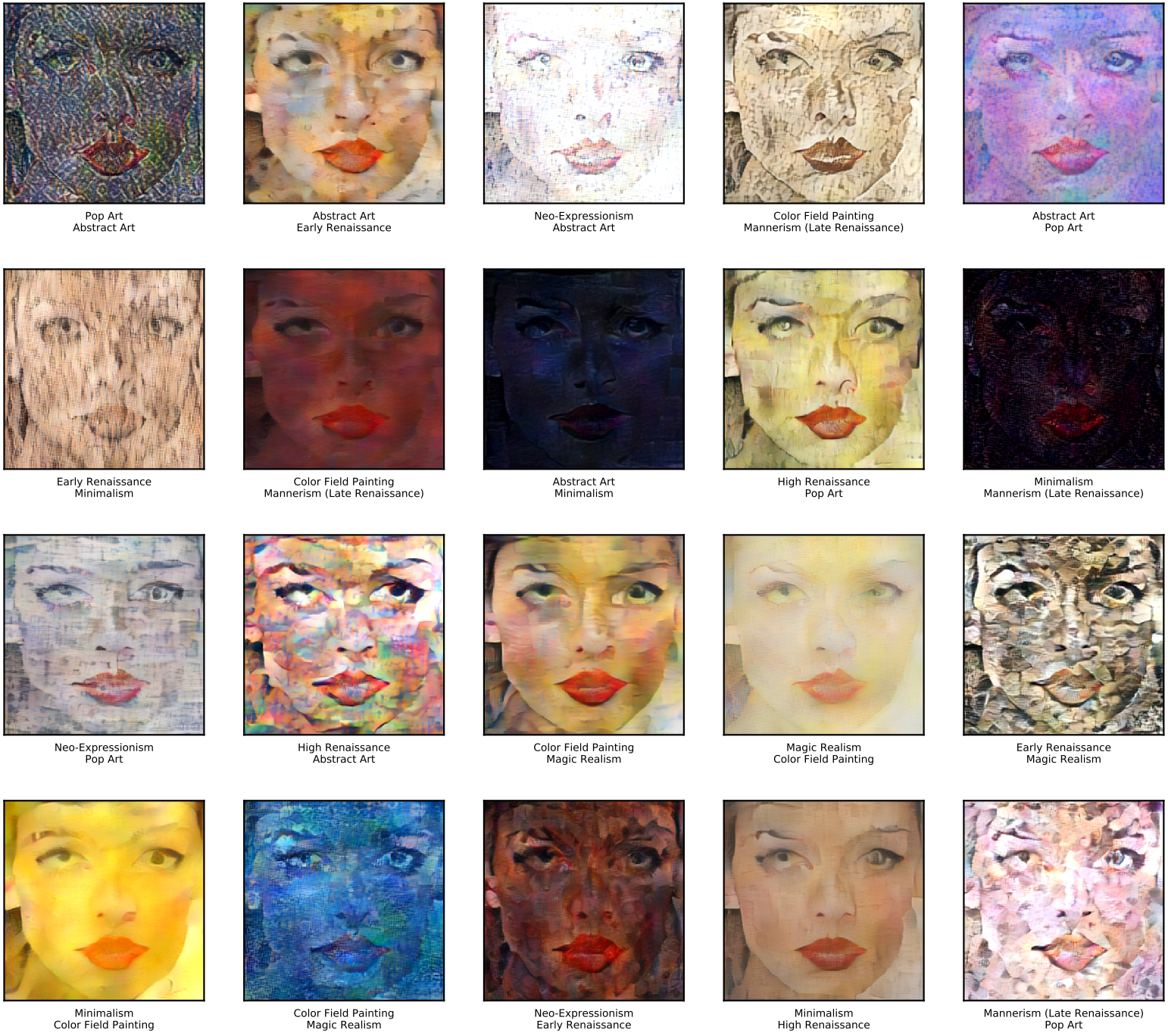


Figure A.2: Randomized multi style outputs for CS-GAN.

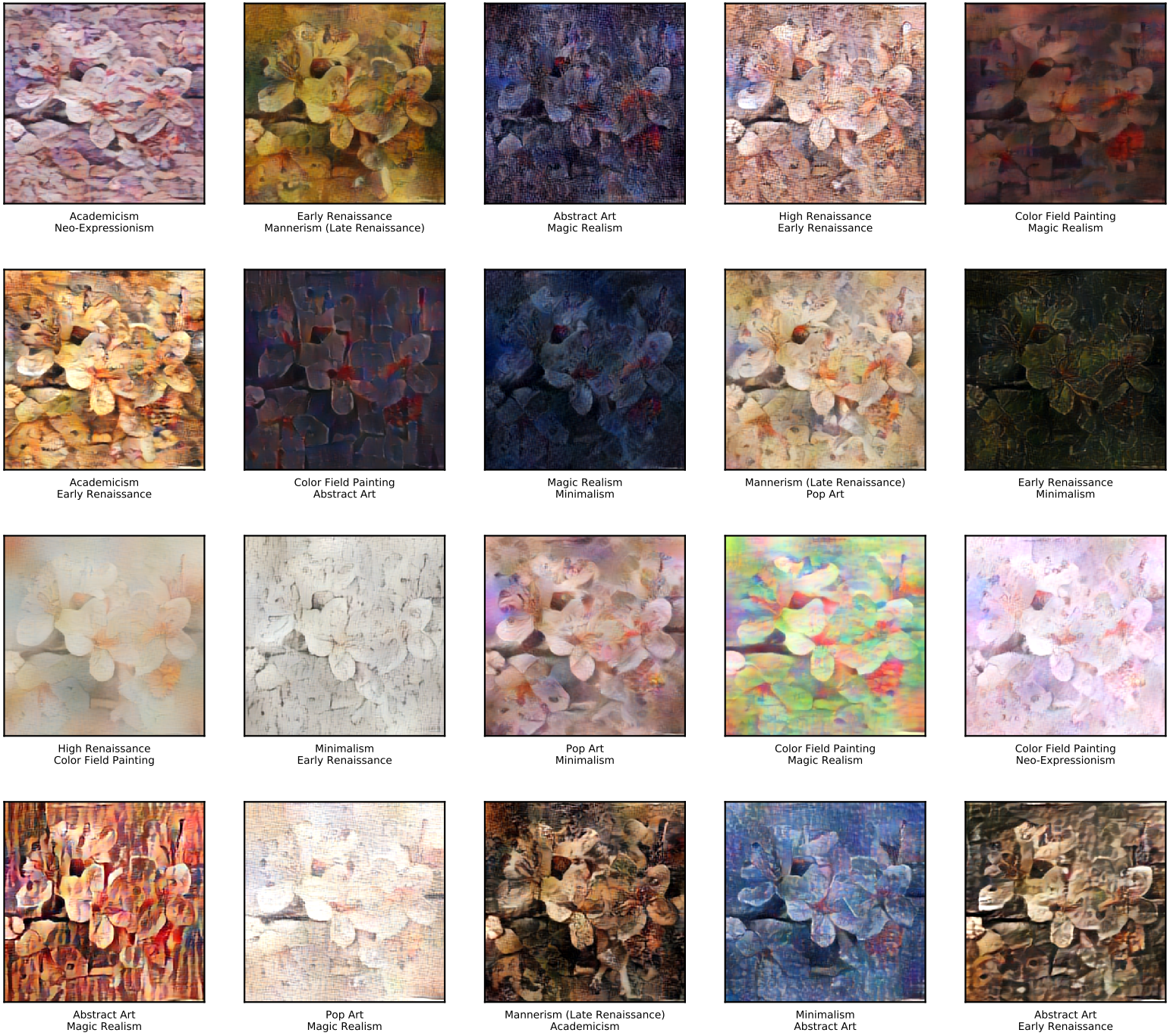


Figure A.3: Randomized multi style outputs for CS-GAN.

Bibliography

[1] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016.

[2] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. Yang, "Universal style transfer via feature transforms," *CoRR*, vol. abs/1705.08086, 2017.

[3] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *CoRR*, vol. abs/1508.06576, 2015.

[4] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, (New York, NY, USA), p. 341–346, Association for Computing Machinery, 2001.

[5] I. Drori, D. Cohen-Or, and H. Yeshurun, "Example-based style synthesis," pp. II– 143, 07 2003.

[6] O. Frigo, N. Sabater, J. Delon, and P. Hellier, "Split and match: Example-based adaptive patch sampling for unsupervised style transfer," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 553–561, June 2016.

[7] M. Elad and P. Milanfar, "Style transfer via texture synthesis," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2338–2351, 2017.

[8] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH 98*, 1998.

[9] A. Kolliopoulos, *Image segmentation for stylized nonphotorealistic rendering and animation*. PhD thesis, University of Toronto, 2005.

[10] B. Gooch, G. Coombe, and P. Shirley, "Artistic vision," *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR 02*, 2002.

[11] Y.-Z. Song, P. L. Rosin, P. M. Hall, and J. Collomosse, "Arty shapes," in *Proceedings of the Fourth Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics'08, (Aire-la-Ville, Switzerland, Switzerland), pp. 65–72, Eurographics Association, 2008.

[12] M. Zhao and S.-C. Zhu, "Portrait painting using active templates," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, NPAR '11, (New York, NY, USA), p. 117–124, Association for Computing Machinery, 2011.

[13] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song, "Neural style transfer: A review," *CoRR*, vol. abs/1705.04058, 2017.

[14] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1105–1114, 2017.

[15] C. Gao, D. Gu, F. Zhang, and Y. Yu, "Reconet: Real-time coherent video style transfer network," *CoRR*, vol. abs/1807.01197, 2018.

[16] R. Poplin and A. Prins, "Behind the scenes with stadia's style transfer ml," Mar 2019.

[17] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *CoRR*, vol. abs/1812.04948, 2018.

[18] Z. Wang, L. Zhao, W. Xing, and D. Lu, "Glstylenet: Higher quality style transfer combining global and local pyramid features," *CoRR*, vol. abs/1811.07260, 2018.

[19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *ArXiv*, vol. abs/1406.2661, 2014.

[20] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, (New York, NY, USA), pp. 327–340, ACM, 2001.

[21] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," *ACM Trans. Graph.*, vol. 25, pp. 1221–1226, July 2006.

[22] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," *CoRR*, vol. abs/1603.03417, 2016.

[23] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," *CoRR*, vol. abs/1701.02096, 2017.

[24] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," *CoRR*, vol. abs/1604.04382, 2016.

[25] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," *CoRR*, vol. abs/1610.07629, 2016.

[26] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," *CoRR*, vol. abs/1703.09210, 2017.

[27] H. Zhang and K. Dana, "Multi-style generative network for real-time transfer," *arXiv preprint arXiv:1703.06953*, 2017.

[28] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. Yang, "Diversified texture synthesis with feed-forward networks," *CoRR*, vol. abs/1703.01664, 2017.

[29] T. Q. Chen and M. Schmidt, "Fast patch-based style transfer of arbitrary style," *CoRR*, vol. abs/1612.04337, 2016.

[30] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," *CoRR*, vol. abs/1703.06868, 2017.

[31] A. J. Champandard, "Semantic style transfer and turning two-bit doodles into fine artworks," *CoRR*, vol. abs/1603.01768, 2016.

[32] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," *CoRR*, vol. abs/1601.04589, 2016.

[33] J. Ye, Z. Feng, Y. Jing, and M. Song, "Finer-net: Cascaded human parsing with hierarchical granularity," in *2018 IEEE International Conference on Multimedia and Expo, ICME 2018, San Diego, CA, USA, July 23-27, 2018*, pp. 1–6, 2018.

[34] H. Zhang, K. J. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," *CoRR*, vol. abs/1803.08904, 2018.

[35] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *CoRR*, vol. abs/1705.01088, 2017.

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

[37] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[38] Y. Lecun and F. Soulie Fogelman, "Modeles connexionnistes de l'apprentissage," *Intellectica, special issue apprentissage et machine*, vol. 2, 01 1987.

[39] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics*, vol. 59, pp. 291–294, 1988.

[40] R. S. Zemel and G. E. Hinton, "Learning population codes by minimizing description length," *Neural Computation*, vol. 7, pp. 549–564, May 1995.

[41] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early visual concept learning with unsupervised deep learning," 2016.

[42] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," 2016.

[43] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *CoRR*, vol. abs/1601.06759, 2016.

[44] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *CoRR*, vol. abs/1606.05328, 2016.

[45] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *CoRR*, vol. abs/1701.05517, 2017.

[46] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.037628, 2017.

[48] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *CoRR*, vol. abs/1609.04802, 2016.

[49] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *CoRR*, vol. abs/1612.03242, 2016.

[50] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *CoRR*, vol. abs/1605.05396, 2016.

[51] V. Nagarajan and J. Z. Kolter, "Gradient descent GAN optimization is locally stable," *CoRR*, vol. abs/1706.04156, 2017.

[52] L. M. Mescheder, "Which training methods for gans do actually converge?," *CoRR*, vol. abs/1801.04406, 2018.

[53] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," *CoRR*, vol. abs/1610.07584, 2016.

[54] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *CoRR*, vol. abs/1506.05751, 2015.

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

[56] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[57] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," 2016.

[58] T. Miyato and M. Koyama, "cgans with projection discriminator," *CoRR*, vol. abs/1802.05637, 2018.

[59] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *ArXiv*, vol. abs/1701.04862, 2017.

[60] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[61] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," pp. 448–456, 2015.

[62] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *CoRR*, vol. abs/1802.05957, 2018.

[63] Z. Zhou, Y. Song, L. Yu, and Y. Yu, "Understanding the effectiveness of lipschitz constraint in training of gans via gradient analysis," *CoRR*, vol. abs/1807.00751, 2018.

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[65] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[66] G. H. Golub and H. A. van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1, pp. 35 – 65, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

[67] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," 2017.

[68] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[69] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," *CoRR*, vol. abs/1709.07871, 2017.

[70] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[71] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 11–13 Apr 2011.

[72] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[74] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR*, vol. abs/1706.08500, 2017.

[75] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2242–2251, 2016.

[76] A. Lucas, S. L. Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *CoRR*, vol. abs/1806.05764, 2018.

[77] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard GAN," *CoRR*, vol. abs/1807.00734, 2018.

[78] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[79] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016.

[80] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.