From Rule-based to Learning-based Image-Conditional Image Generation

by

© *Zili Yi*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Doctor of Philosophy

Department of *Computer Science*

Memorial University of Newfoundland

Supervisor: Dr. Minglun Gong

May, 2018

St. John's Newfoundland

Contents

Abstract

Visual contents, such as movies, animations, computer games, videos and photos, are massively produced and consumed nowadays. Most of these contents are the combination of materials captured from real-world and contents synthesized by computers. Particularly, computer-generated visual contents are increasingly indispensable in modern entertainment and production. The generation of visual contents by computers is typically conditioned on real-world materials, driven by the imagination of designers and artists, or a combination of both. However, creating visual contents manually are both challenging and labor intensive. Therefore, enabling computers to automatically or semi-automatically synthesize needed visual contents becomes essential. Among all these efforts, a stream of research is to generate novel images based on given image priors, e.g., photos and sketches. This research direction is known as image-conditional image generation, which covers a wide range of topics such as image stylization, image completion, image fusion, sketch-to-image generation, and extracting image label maps. In this thesis, a set of novel approaches for image-conditional image generation are presented.

The thesis starts with an exemplar-based method for facial image stylization in Chapter 2. This method involves a unified framework for facial image stylization based on a single style exemplar. A two-phase procedure is employed, where the first phase searches a dense and semantic-aware correspondence between the input and the exemplar images, and the second phase conducts edge-preserving texture transfer. While this algorithm has the merit of requiring only a single exemplar, it is constrained to face photos. To perform generalized image-to-image translation, Chapter 3 presents a data-driven and learning-based method.

Inspired by the dual learning paradigm designed for natural language translation [115], a novel dual Generative Adversarial Network (DualGAN) mechanism is developed, which enables image translators to be trained from two sets of unlabeled images from two domains. This is followed by another data-driven method in Chapter 4, which learns multi-scale manifolds from a set of images and then enables synthesizing novel images that mimic the appearance of the target image dataset. The method is named as Branched Generative Adversarial Network (BranchGAN) and employs a novel training method that enables unconditioned generative adversarial networks (GANs) to learn image manifolds at multiple scales. As a result, we can directly manipulate and even combine latent manifold codes that are associated with specific feature scales. Finally, to provide users more control over image generation results, Chapter 5 discusses an upgraded version of iGAN [126] (iGAN-HD) that significantly improves the art of manipulating high-resolution images through utilizing the multi-scale manifold learned with BranchGAN.

Acknowledgements

After an intensive period of several months, today is the day: writing this note of thanks is the finishing touch on my dissertation. It has been a period of intense learning for me, not only in the scientific arena, but also on a personal level. Writing this dissertation has had a big impact on me. I would like to reflect on the people who have supported and helped me so much throughout the last four years.

I would first like to single out my supervisor, Dr. Minglun Gong, I want to thank you for your excellent cooperation and for all of the opportunities I was given to conduct my research and further my dissertation. You supported me greatly and were always willing to help me. You definitely provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

Then I would particularly like to thank my colleagues in Simon Fraser University, Dr. Hao Zhang, Dr. Ping Tan, Mr. Zhiqin Chen and Mr. Ligeng Zhu for their wonderful collaboration. Particularly thank Dr. Hao Zhang for his valuable guidance.

In addition, I would like to thank my fellow students, Yang Li, Songyuan Ji, Hao Cai, Wendong Mao, Shiyao Wang, Xue Cui, Xin Huang, Jun Zhou, Mingjie Wang and Yiming Qian for their valuable support and help, and also the happy time we spent together. I would like to thank the supervisory committee members, Dr. Wolfgang Banzhaf and Dr. Yuanzhu Chen, and the committee members of my comprehensive exam, Dr. Mohamed Shehata, Dr. Paul Gillard, Dr. Antonina Kolokolova, Dr. Manrique Mata-Montero, Dr. Andrew Vardy and Dr. Krishnamurthy Vidyasankar for the precious advice that helps enhance my basic training and perfect my thesis proposal. I would like to thank my colleagues in Altumview

System Inc., Dr. Jie Liang, Dr. Jane Z. Wang, Dr. Xing Wang, Dr. Jiannan Zheng and Dr. Pegah Kharazmi for exchange of ideas. I would like to thank the administrative assistants Andrew Kim, Dawn Quinton, Erin Manning, Darlene Oliver, Sharene Bungay and Brenda Hillier for their warm-hearted helps and assistance all the time.

I would also like to thank my wife, Baihui Zhang, and my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, there are my friends. We were not only able to support each other by deliberating over our problems and findings, but also happily by talking about things other than just our papers.

Thank you very much, everyone!

Zili Yi

Coquitlam, June 02, 2018.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Human brains are wired for visual content. They typically process visual information 60,000 times faster than text. Every day, people watch videos, play digital games and share photos on social media for entertainments. In the meantime, professionals employ visual contents to exchange ideas and boost productivity. However, the creation of visual contents typically require professional skills and laborious operation. In this scenario, computer-generated visual contents play an increasingly important role in satisfying human needs for visual contents.

Image synthesis is a highly broad area. Specifically, image synthesis could be conditioned on prior materials such as texts, images, 3D geometric models, categories or manifolds. This thesis only discusses about image-conditional image synthesis. Image-conditional image generation involves creating images through restructuring, retouching or manipulating input digital images. The study includes a wide range of topics such as artistic stylization, image fusion, image completion, sketch-to-image generation, and image segmentation (converting image to label maps). It is also referred to as image-to-image translation in scenarios where the prior is a single image.

In this thesis, a series of approaches are presented for image-conditional image synthesis. In Chapter 2, a rule-based approach is presented for facial image stylization. The method enables generating different styles, whereas it is heuristically defined for facial images only. The following chapter discusses a general-purpose learning paradigm for image-to-image translation, assuming that sufficient data are available for training. It achieves great success in a wide range of applications including converting aerial images to maps, day scenes to night scenes, sketches to photos and label maps to real images. Whereas the limitation is that the results image are mostly low resolution: see Chapter 3. The third method, BranchGAN, aims at high-resolution image synthesis with a multi-scale process. It enables generation of high-resolution images, scale-aware image fusion and coarse-to-fine image synthesis: see Chapter 4. On the basis of BranchGAN, an approach that allows users to interactively manipulating image priors for image generation is presented: see Chapter 5. Figure 1.1 illustrates the frameworks of the presented methods.

1.2 Related Work in Image-Conditional Image Synthesis

The last two decades have seen the proliferation of a number of image-conditional image synthesis algorithms, which could be categorized through different properties.

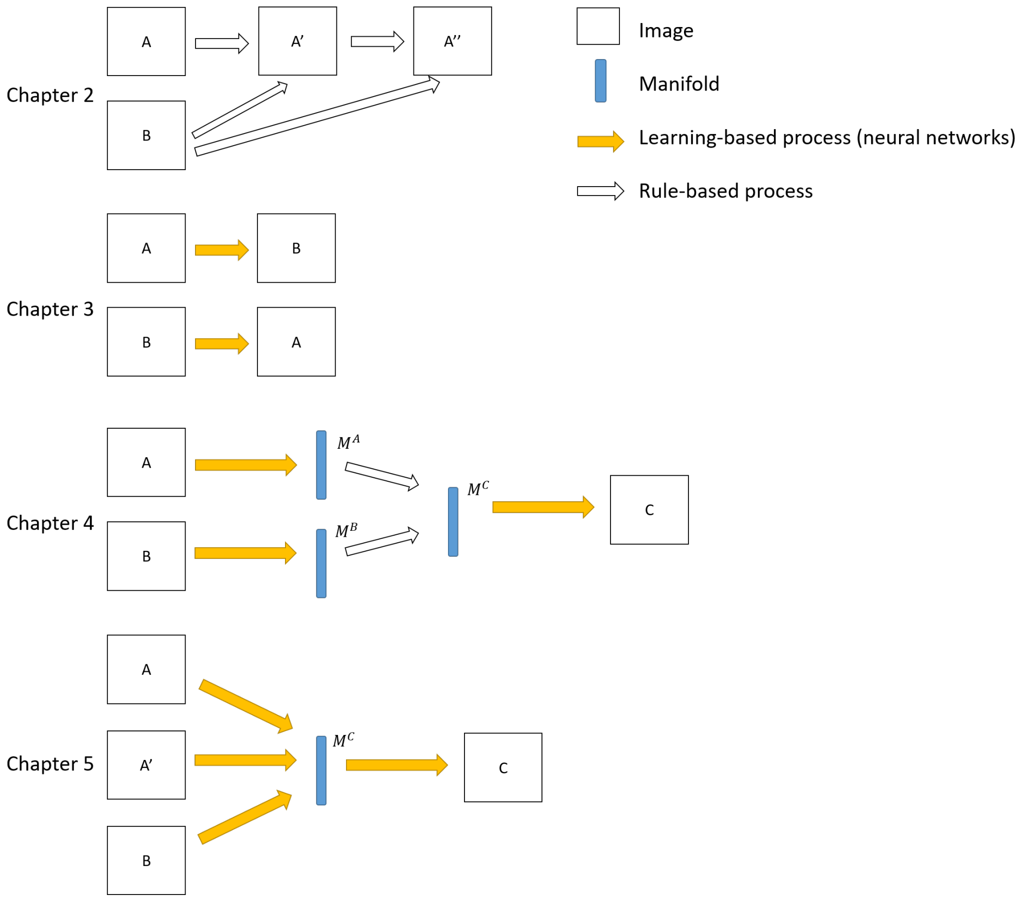


Figure 1.1: A simplified illustration of the frameworks of image-conditional image synthesis approaches presented in each of the chapters. The method presented in Chapter 2 conditions image generation on a content image (A) and a style image (B). Chapter 3 describes a method for cross-domain image-to-image translation (A to B, or B to A). Chapter 4 discusses a method for scale-aware image fusion, which synthesizes a novel image (C) by selectively combining the scale-disentangled manifolds (M^A and M^B) of two input images (A and B). Chapter 5 presents a method for interactive image generation which conditions image generation on user-manipulated image priors such as a color map (A), a color map mask (A') and/or an edge map (B).

1.2.1 Photorealistic and non-photorealistic

Based on whether the synthesized images are photorealistic or non-photorealistic, image-conditional image synthesis methods could be divided into photorealistic or non-photorealistic methods. Photorealistic methods typically involve pure color transformation or low-level filtering [55, 14, 91]. A special type of photorealistic methods, photorealistic rendering aims to simulate images (usually rendered from 3D graphic models) that looks exactly like the real-world objects. Synthesizing naturalistic images by directly simulating the lighting and imaging process of physical world with computational techniques such as photon-mapping or radiation, is widely used for this purpose. Nonetheless, in recent years, increasing number of researchers endeavor to achieve this goal through manipulations in 2D domain only. An example is to transform non-photorealistic images to natural images [93].

Non-photorealistic (NPR) focuses on converting photographs into painterly or artistic stylized images, which is an area of computer graphics that aims at enabling various expressive styles for digital art. In contrast to photorealism, NPR is motivated by artistic styles (e.g., drawing, painting), technical visualization and animated cartoons [112, 100, 73, 64, 87, 28, 110, 59, 124]. NPR is widely used in video games and movies, scientific visualization, architectural illustration and experimental animation.

1.2.2 Low-level, middle-level and high-level methods

Low-level algorithms employ low-level image processing operators such as Gamma correction, color adjustment, placement of marks (e.g., strokes, hatches and stipples) or simple filtering operators [100, 64, 14]. For stylized video generation, low-level methods based on optical flow are used [95]. Early in the last decade, mid-level operators such as superpixel segmentor, perceptual saliency estimators and other sophisticated filtering operators yielded improved style diversity and robustness [112, 73, 87, 28, 110, 82]. As the field matures, it sees a increased fusion of higher-level operators such as image parser for content-based and semantic-aware image synthesis [42, 91, 55, 59, 124].

1.2.3 Exemplar-based, data-driven and expert-tuned methods

Exemplar-based image synthesis eases the careful design of application-specific operators and enables multiple styles in a single paradigm. In this way, exemplar-based image synthesis facilitates the functionality extension of an image synthesis system [42, 100, 28, 110]. As a special case of example-based image synthesis, data-driven image synthesis usually requires significant amounts of templates, examples, or elements [55, 59, 124]. Expert-tuned methods do not explicitly employ an exemplar. Instead, they involve a complex pipeline that is designed, established and fine-tuned with expertise knowledge [112, 73, 88, 64, 87, 91, 14], most of which are application-specific.

1.2.4 Semi-automatic and fully-automatic methods

Semi-automatic methods require users' assistance during image synthesis and conventionally provide carefully-designed user interface, soliciting user's marking, clicking, circling, dragging, or touching [42, 88, 87, 28]. Most methods intend to function without human assistance and provide fully automated procedure [112, 100, 73, 64, 110, 91, 55, 14, 59, 124].

1.2.5 Rule-based and learning-based methods

Rule-based methods for image synthesis normally involve a sequence of manipulations and can be unified as a workflow of processing units, where each processing unit is deliberately designed or considerately fine-tuned for specific applications [88, 87, 28, 112, 100, 73, 64, 110, 55, 14, 59, 124]. However, Deep Neural Networks, especially the recently-prevalent FCN and GAN, are end-to-end architectures that directly condition image synthesis on priors, and learn to synthesize images. In addition, image synthesis can also benefit from the advanced cognitive power and modeling potent of Deep Neural Networks.

Since 2014, academic predecessors have started to experiment deep learning for image synthesis. Leon et al. proposed a novel method to synthesize images of target style, through optimizing a sophisticated loss objective adding up the content loss and the style loss. The content loss measures the difference of contents between the synthesized and the content prior, whereas the style loss measures the disparity of styles between the synthesized and the style exemplar. The computation of loss function relies upon a pre-trained Convolutional Neural Network (CNN) for image classification [23]. The method turns out to be effective in generating stylized images that mimics world-famous painting artworks. As a result, the paper triggered an upsurge of research on learning-based image synthesis. In 2015, Justin et al. proposed a novel framework based on Fully Convolutional Network (FCN) [68] known as perceptive loss, enabling synthesizing stylized images in real-time [41].

On the other hand, the invention of Generative Adversarial Network (GAN) in 2014 [79] motivated a hot wave for image representation learning [79, 107] and high-quality natural image synthesis [96, 80]. In 2015, the emergence of FCN [68] further empowered GAN for end-to-end image translation, or conditional image synthesis, achieving record-breaking results for texture synthesis [58, 102], portrait stylization [98] and general-purpose image-to-image translation[39]. FCN is also used separately for end-to-end image translation tasks such as portrait segmentation, super-resolution, stylization [91] and sketch synthesis [121].

Another stream of research for image representation learning and image synthesis, known as PixelRNN, was started by Oord et al. [75]. PixelRNN takes the advantage of the modeling power of Long-Short-Term-Memory (LSTM) networks and attempts to model image distribution in a pixelwise manner. In specific, a two-dimensional multi-layer LSTM network is employed to predict the color of each pixel based on all the top-left pixels it has scanned.

1.3 Deep Neural Networks for Image Synthesis

Deep Neural Network is a branch of machine learning that endeavors to learn highly abstractive representation of data. Typically, Deep Neural Network involves multiple neuron layers other than the input layer and output layer. In a deep network, the signals are taken by the input layer and then pass through multiple processing layers before reaching the output, thus allowing sophisticated function modeling. Various deep neural architectures have been proposed and applied to challenging tasks such as image classification, object detection, image parsing and video analysis, including Deep Belief Networks (DBN) [34], Convolutional Neural Networks (CNN) [53] and Recurrent Neural Networks (RNN). Considerable improvements have been made over previous non-neural-network-based methods. In last few years, Deep Neural Networks prove to be highly effective in learning representations of image structures and play an increasingly essential role in image synthesis.

In the following paragraphs, I will present a few generative neural models related to end-to-end image synthesis. They include Deep Belief Network, Fully Convolutional Network, PixelRNN, Generative Adversarial Network and Variational Auto-Encoder.

1.3.1 Deep Belief Network (DBN)

DBN is a probabilistic, generative model composed of multiple layers of hidden units. It can be considered to be stacked with multiple learning modules that are typically made of Restricted Boltzmann Machines (RBM). An RBM is an undirected, generative energy-based model with a "visible" input layer and a hidden layer, and connections between the layers but not within layers.

A DBN can be efficiently trained in a layer-by-layer and unsupervised manner. The training method for RBMs, known as contrastive divergence learning, is first proposed by Geoffrey Hinton et al. [34] and provides an approximation to the maximum likelihood method. As a generative model, it can be used to learn embeddings of any vectorized meta data such as facial images [97], hand-writing digits [34] and speech [44].

1.3.2 Fully Convolutional Network (FCN)

CNN, first proposed by Yann Lecun et al. [53], consists of one or more convolutional layers and pooling layers with prediction layer on top. This architecture allows CNNs to take advantage of the 2D structure of input data. CNN is normally trained with back-propagation. In comparison with other deep architectures, CNNs are easier to train and have many fewer parameters to estimate, making them a highly effective architecture to use. CNN has been the primary method chosen to process visual and other two-dimensional data. CNN has shown superior results in a wide variety of topics like image classification/localization [49], object detection, image parsing and image generation.

FCN, first developed by Long et al. [68], is a type of “fully convolutional” networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. FCN is suitable for spatially dense prediction tasks like image segmentation, image colorization and image stylization. FCN normally involves a skip architecture that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed prediction.

1.3.3 PixelRNN

Long Short-Term Memory (LSTM) network, first published in 1997 [36], is a branch of Recurrent Neural Network (RNN). LSTM prevents back-propagated errors from vanishing or exploding, by introducing a recurrent gate known as forget gate, thus enabling errors to propagate backwards through unlimited numbers of discrete time steps. In this case, LSTM can learn tasks that require both long-term and short-term memories. LSTM is typically trained using Connectionist Temporal Classification (CTC). LSTM has demonstrated its considerable power for sequential-data-related tasks like handwriting recognition, speech recognition, text-to-speech synthesis, currently holding the best records of those tasks.

As a branch of LSTM, PixelRNN is a generative model for pixelwise color prediction. PixelRNN attempts to model images from a specific domain using a bi-dimensional multi-layer LSTM network to predict the color of a pixel based on all the scanned pixels in top-left [75].

1.3.4 Generative Adversarial Network (GAN)

GAN is first introduced by Goodfellow et al. in 2014 [79]. GAN is a system made up of two neural networks, a generator and a discriminator, competing against each other in a zero-sum game framework [79]. Normally, the generator maps from a latent space to a particular data distribution of interest. In particular, the discriminator learns to differentiate between real instances and the synthesized "fake" instances, while the generative network learns to "fool" the discriminator by synthesizing "fake" instances that mimic real data. As a result, the generator can effectively learn the distribution of real data and reverse engineer the target dataset.

1.3.5 Variational Auto-Encoder (VAE)

Variational Auto-Encoder models inherit Auto-Encoder architecture, but make strong assumptions concerning the distribution of latent variables. They use variational approach for latent representation learning, which results in an additional loss component and specific training algorithm called Stochastic Gradient Variational Bayes (SGVB) [48]. It assumes that the data is generated by a directed graphical model $p(\mathbf{x}|\mathbf{z})$ and that the encoder is learning an approximation $q_\phi(\mathbf{z}|\mathbf{x})$ to the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ where ϕ and θ denote the parameters of the encoder (recognition model) and decoder (generative model) respectively.

1.4 Contributions of the Thesis

This thesis aims at general-purpose solutions for image-conditional image generation, with the attempt to assure the high-quality of the outputs and relieve the requirements for massive data resources. The method presented in Chapter 2 can transfer multiple styles and only requires a single style exemplar for each style. DualGAN, described in Chapter 3, is a general-purpose solution for image-to-image translation and yet it does not require any paired data. BranchGAN (Chapter 4) learns to disentangle image representations by scales with unsupervised data only, while assuring the "high-resolution" property. Similar to BranchGAN, iGAN-HD (Chapter 5)) can generate high-quality results and only requires a collection of unsupervised data of the same category. The latter three methods are all learning-based and GAN-family methods. More information about the novelty of each method and their relations to prior works will be explained separately in each of the chapters.

Two of the presented methods have been published and well recognized by peer researchers. The method presented in Chapter 2 was published on *The Visual Computer* in 2016 with the title of "Artistic stylization of face photos based on a single exemplar", and that presented in Chapter 3 was published on *the International Conference on Computer Vision (ICCV)* in 2017 with the title of "DualGAN: Unsupervised Dual Learning for Image-to-Image Translation". Google scholar indicates that DualGAN has attracted over 130 citations in no more than 18 months. The methods presented in Chapter 4 and Chapter 5 will also be submitted to related conference or journals for publication.

1.5 Organization of the Dissertation

The rest of this thesis is organized as the following. It starts with an exemplar-based method for facial image stylization in Chapter 2. While this algorithm has the merit of requiring only a single exemplar, it is constrained to facial images. To perform generalized image-to-image translation, Chapter 3 presents a data-driven and learning-based method, referred as DualGAN. This is followed by another data-driven method, BranchGAN, in Chapter 4, which learns multi-scale manifolds from a set of images. To provide users more control over image generation results, Chapter 5 discusses an upgraded version of iGAN, referred as iGAN-HD, that significantly improves the art of manipulating high-resolution images. Finally Chapter 6 summarizes the achievements of the thesis work and discusses several future directions in image synthesis.

Chapter 2

Exemplar-based Facial Image Stylization

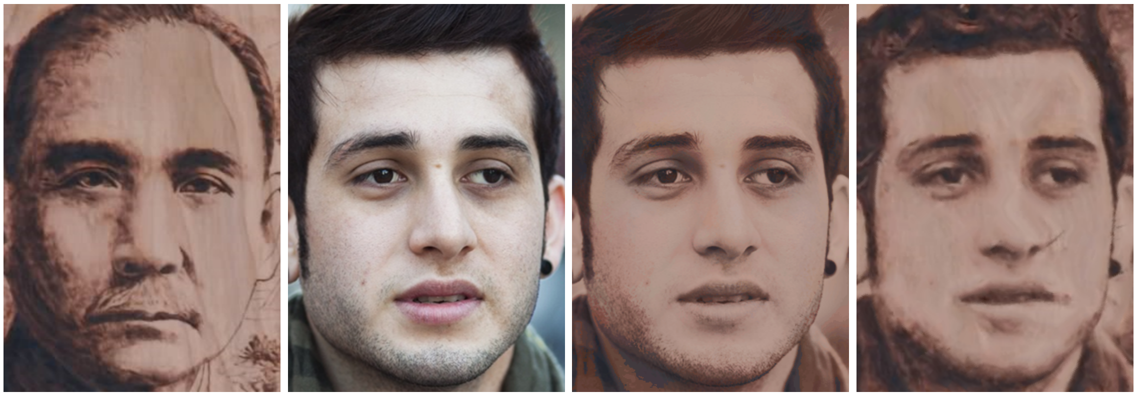
In this Chapter, we propose a rule-based algorithm for fully automatic face photo stylization based on a single style exemplar. Constrained by the “single-exemplar” condition, where the numbers and varieties of patch samples are limited, we introduce flexibility in sample selection while preserving identity and content of the input photo. Based on the observation that many styles are characterized by unique color selections and texture patterns, we employ a two-phase procedure. The first phase searches a dense and semantic-aware correspondence between the input and the exemplar images so that colors in the exemplar can be transferred to the input. The second phase conducts edge-preserving texture transfer, which preserves edges and contours of the input and mimics the textures of the exemplar at multiple scales. Experimental results demonstrate compelling visual effects and notable improvements over other state-of-the-art methods which are adapted for the same task.

2.1 Overview

Faces are common objects in artworks and paintings. Compared with manual painting or drawing of faces, which requires laborious operation and advanced technique, automated artistic face synthesis by computers is fast and inexpensive. Previous algorithms, which stylize a given face photo automatically or semi-automatically [60, 62, 69, 57, 25, 122, 61], are mostly style-specific. Different styles convey different visual features and entail different aesthetic standards. However, styles differ among time, nations, regions, media and artists. There exist great numbers of artistic styles in historical and modern art, thus making it impractical to implement a style-specific rendering algorithm for each style. In this case, exemplar-based face stylization becomes important. By using exemplar-based face stylization, the algorithm can be easily expanded by importing one or multiple available “style” exemplars. This advantage on convenience makes a general-purpose face stylization method very useful in a number of scenarios, even though the stylization results may not be as fine-tuned as certain style-specific approaches.

Three factors make face stylization a challenging task. First, error tolerance is small since human eyes are exceptionally sensitive to facial information. Secondly, the geometry and appearance of faces vary by race, individual, expression, age, and pose. Other factors, such as glasses, also result in variation. A robust stylization algorithm should work well for varieties of inputs even when the input and the exemplar are severely disparate. Thirdly, artists often apply different treatments to different facial parts (e.g., the mouth, nose, eyes, eyebrows, chin and hair) to achieve compelling visual effects. As a result, the stylization algorithm needs to be semantic-aware.

To address these challenges, we propose a novel two-phase style transfer procedure.



(a) (b) (c) (d)

Figure 2.1: The overall process of our approach: (a) the exemplar (pyrography style), (b) the input, (c) result of semantic-aware color transfer, (d) result of edge-preserving texture transfer.

The first phase processes the image with a semantic-aware color transfer (SACT) technique, which considers both the global color space alignment and local semantic correspondences. The second phase employs an edge-preserving texture transfer (EPTT) scheme, which attempts to preserve edges and contours of the input while mimicking the target texture at multiple scales. As a result, the algorithm can automatically stylize the input into an image of the exemplar style, while keeping the identity, content and large-scale structure of the input; see Figure 2.1.

Focusuing on painterly and artistic stylization of face images with a single exemplar, our method makes the following three contributions:

- Our algorithm can stylize an artistic face using a single exemplar image, which differs from many existing data-driven face synthesis approaches that require large-scale datasets [124, 92, 59].

- We find a method to combine the local and global constraints for color style transfer. Previous algorithms for color style transfer are either local [92] or global [69], which are not suitable for face stylization.

- We propose an edge-preserving texture transfer algorithm for effective texture transfer, which plays a key role in achieving compelling visual effects for our task.

We discuss related research in Section 2.2 and present our algorithm in Section 2.3. Section 2.4 shows our experiment results and compares them with those of previous methods. The chapter concludes in Section 2.5, with potential future directions identified.

2.2 Related Work

2.2.1 Non-Photorealistic Rendering (NPR)

Non-photorealistic rendering is an active field in computer graphics. Motivated by the Art and history of Art, the main objective of this field is to either simulate traditional media or simplify an image in an illustrative form. Previous algorithms mostly focus on a specific art-style, e.g., pencil drawing [122, 69], tessellation [61], halftoning [60], stippling [63], oil painting [25, 43] and water color painting [57]. Each style involves unique sets of colors, tones, dots, sketches and textures. Unlike these style-specific rendering algorithms, our method intends to cover a large range of non-photorealistic styles by using exemplar-based style transfer techniques.

Our work is similar to existing face synthesis approaches, such as [124, 59, 108]. Zhang. et al. [124] propose a data-driven cartoon face synthesis approach by using a large set of pre-designed face elements (e.g., mouth, nose, eye, chin line, eyebrow and hair). Li.

et al. [59] synthesize animated faces by searching across a set of exemplars and extracting best-matched patches. Wang et al. propose a novel data-driven face sketch synthesis method using a multiscale Markov Random Fields (MRF) model [108]. Different from ours, these approaches require large sets of exemplar photos.

2.2.2 Style transfer

The assessment of an artwork depends on two factors: content and form. In a style transfer system, the content is provided in the input image and the form (or style) is defined by an exemplar. The task is to combine the content of the input photo and the style of the exemplar to generate a novel piece of artwork. The problem to solve in this chapter is basically a style transfer problem.

Our objective is similar to the style-transfer approach by Shin et al. [92], which targets on photograph style transfer for headshot portraits based on multiple stylized exemplars. However, the difference is that we aim at non-photorealistic style transfer based on a single exemplar. Greater variation among non-photorealistic styles makes the task more challenging.

2.2.3 Color style transfer

Color is a key element in artistic and painterly stylization. Endowing an image with a specified color style defined by an exemplar is an essential technique in stylization. Previous color modification algorithms include histogram equalization, histogram specification and color histogram matching [72, 14]. Neumann et al. [72] propose a lightness, hue and saturation histogram match scheme for color style transfer. The method proposed by Ha-

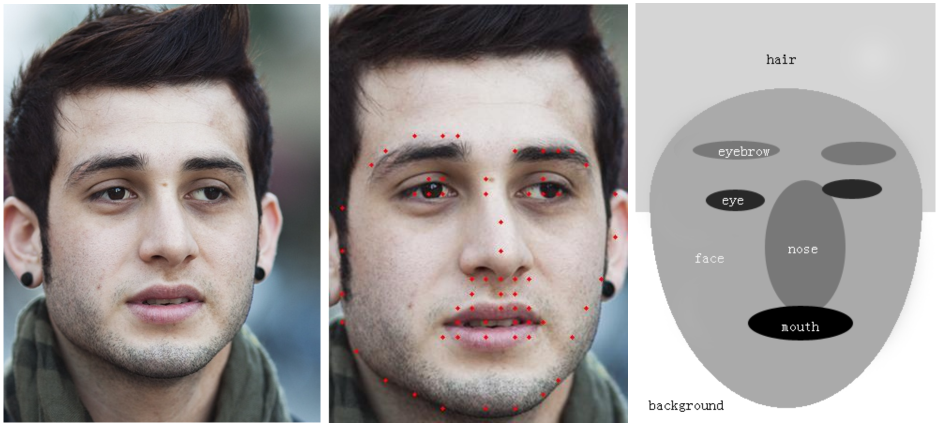
Cohen et al. can automatically transfer color styles across images that share contents, in which semantic information is used as guidance [29]. The method proposed by Cohen-Or et al. [14] enhances the color harmony among pixels in an image by optimizing a cost function that seeks neighboring coherency while remaining faithful to original colors as much as possible. Motivated by their work, in the first color transfer phase, we optimize a cost function that considers both global color distribution (intensity histogram) and local geometry correspondences. An additional constraint in our method is the color consistency among semantically identical regions and we do not explicitly pursue a targeted color distribution.

2.2.4 Texture synthesis/transfer

Texture synthesis aims to synthesize a customized size of texture given a finite sample of texture. Sophisticated synthesis algorithms such as image quilting [21], non-parametric sampling [22] and optimization-based texture synthesis [46] are widely used. One task of texture synthesis is to avoid unnatural artifacts introduced by the synthesis. As for texture transfer, an additional constraint is that the synthesized texture should be consistent with the color variation of the input image. Image quilting [17, 21] can be applied to texture transfer but does not maintain the structures in the input image very well. To preserve facial structures during the second texture transfer phase, we extend upon image melding [17].

2.3 Single Exemplar Stylization

Our approach is based on the following key observation: many artistic styles can be generated through color and texture transfer, except for the ones that highlight edges or exaggerate shapes. In our approach, color and texture are transferred in two separate phases, where



(a) original input image (b) cropped input image (c) semantic label map

Figure 2.2: Automatic cropping and semantic label map generation: (a) original input image, (b) cropped input image obtained by automatic cropping and scaling based on 68 detected face landmarks (shown as red dots), (c) semantic label map obtained by fitting landmarks.

color transfer is performed first and then followed by texture transfer.

We assume that both the input and the exemplar are images of fixed size ($500x700$ in our implementation) and are mostly covered by frontal faces. When the images supplied by users do not satisfy this condition, an automatic pre-processing step is applied first, which crops and resizes the images with the guidance of 68 face landmarks detected using an open-source face landmark detection library [128, 13]; see Figure 2.2.

2.3.1 Semantics-Aware Color Transfer

Semantics-Aware Color Transfer (SACT) assigns a color for each pixel in the input image by finding a dense correspondence between the input and the exemplar. For pixel $\mathbf{p} = (p_x, p_y)$ in the input image I, its correspondence $\mathbf{q_p}$ in the exemplar image E is found by minimizing a cost function that consists of three terms: semantic term e_{sem}, geometry term e_{geo} and color term e_{clr}.

$$\mathbf{q_p} = \arg\min_{\mathbf{q} \in E} \Big(\alpha_1 e_{sem}(l_{\mathbf{p}}, l_{\mathbf{q}}) + \alpha_2 e_{geo}(\mathbf{p}, \mathbf{q}) + \alpha_3 e_{clr}(\mathbf{p}, \mathbf{q}) \Big), \tag{2.1}$$

where $\alpha_i, (i = 1, 2, 3)$ are weights of these energy terms. $l_{\mathbf{p}}$ (or $l_{\mathbf{q}}$) is the label assigned for pixel $\mathbf{p}$ (or $\mathbf{q}$), which specifies the region that the pixel belongs to (e.g., mouth, nose, eye, eyebrow, face, hair and background). To assign these labels, each region is approximated using an ellipse, which is computed by fitting the corresponding face landmarks; see Figure 2.2.

$e_{sem}(\cdot, \cdot)$ evaluates the incompatibility between two labels. For example, nose and eye are considered highly incompatible since they have distinct colors and features, whereas nose and face are compatible. To specify these semantic relations, a look up table is heuristically defined; see Table 2.1.

$e_{geo}(\mathbf{p}, \mathbf{q})$ measures the geometric cost between the pixel coordinates of $\mathbf{p}$ and $\mathbf{q}$. Directly using the Euclidean distantance between $\mathbf{p}$ and $\mathbf{q}$ as the cost does not tolerate the pose and shape differences between the input and the exemplar faces, nor does it take into account the bilateral symmetry of faces. Hence, before measuring the distance, we first warp the exemplar to align with the input face based on extracted facial landmarks. The warping is performed by first constructing a set of displacement vectors from facial landmarks and then warping the exemplar with the feature-based warping algorithm [10]; see Figure 2.3 (c), (d). This warping operation ensures that the facial landmarks of the warped exemplar align with those of the input image. Furthermore, to utilize bilateral symmetry, we also generate a mirrored version of the exemplar, which is also warp-aligned to the input for computing the second distance value. The final cost is set to the smaller of the two

Table 2.1: The look up table for semantic cost function $e_{sem}(l_{\mathbf{p}}, l_{\mathbf{q}})$. There is no penalty for transferring colors within the same semantic regions and hence the corresponding costs are zero. The infinity cost values can forbid color transfer between the incompatible regions.

$l_{\mathbf{q}}$ $l_{\mathbf{p}}$	mouth	eye	eyebrow	nose	face	hair	background
mouth	0	$+\infty$	$+\infty$	$+\infty$	1	$+\infty$	$+\infty$
eye	$+\infty$	0	$+\infty$	$+\infty$	1	$+\infty$	$+\infty$
eyebrow	$+\infty$	$+\infty$	0	$+\infty$	1	$+\infty$	$+\infty$
nose	$+\infty$	$+\infty$	$+\infty$	0	0	$+\infty$	$+\infty$
face	1	$+\infty$	$+\infty$	1	0	1	$+\infty$
hair	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0	0	$+\infty$
background	$+\infty$	$+\infty$	$+\infty$	$+\infty$	1	1	0

distances, i.e.:

$$e_{geo}(\mathbf{p}, \mathbf{q}) = \min\Big(\|\mathbf{p} - Warp(\mathbf{q})\|, \|\mathbf{p} - Warp(Mirror(\mathbf{q}))\|\Big), \tag{2.2}$$

where function $Warp(\mathbf{q})$ outputs the coordinates of $\mathbf{q}$ after warping and $\|\cdot\|$ computes the L^2 norm.

The last term $e_{clr}(\mathbf{p}, \mathbf{q})$ measures the color cost between pixels $\mathbf{p}$ and $\mathbf{q}$. To accommodate the overall intensity differences between the two images, we first apply histogram equalization based on the pixel intensities and then compute the color cost term using Eq. 2.3:

$$e_{clr}(\mathbf{p}, \mathbf{q}) = \tan\Big(\frac{\pi}{2} \cdot \frac{|I^{equ}(\mathbf{p}) - E^{equ}(\mathbf{q})|}{256}\Big), \tag{2.3}$$

where I^{equ} and E^{equ} denote the equalized grayscale version of the input and exemplar images, respectively. The tangent function is applied to boost penalty for large intensity disparity.

With the cost function defined, the best correspondence of pixel $\mathbf{p}$ that minimizes Eq. 2.1 is searched through enumerating and testing all pixels in the exemplar E. Once found, its color is assigned to pixel $\mathbf{p}$, i.e., $T(\mathbf{p}) = E(\mathbf{q}_\mathbf{p})$, where T refers the color transfer result. Since the cost function takes into account both local semantic information and global intensity histogram alignment, the above procedure can effectively transfer the color style of the exemplar image to the input image in a pixelwise manner.

Figure 2.3 shows the results of color transfer and the impacts of individual terms. It shows that, when using the geometry term without bilateral symmetry, the result is the same as warping the exemplar to the input image. After adding the color term, the result

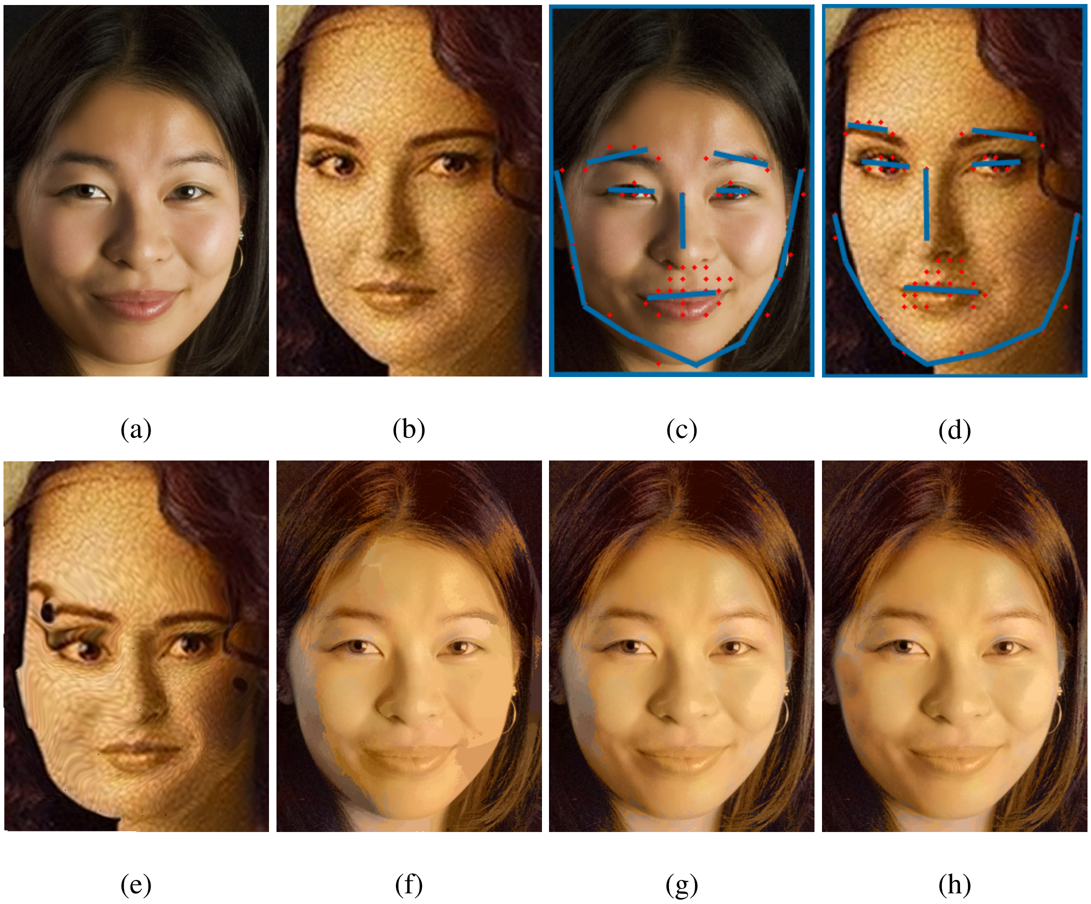


Figure 2.3: Results of Semantic Aware Color Transfer: (a) Input image. (b) Exemplar. (c) Control lines (as shown in blue) constructed from landmarks for the input. (d) Control lines (as shown in blue) constructed for the exemplar. (e) Result obtained using geometry term only and without bilateral symmetry, (f) using geometry (without bilateral symmetry) and color terms, (g) using geometry (with symmetry) and color terms, (h) using all three terms. Note that the light directions are different in input and exemplar photos, where the input has shadow on the right cheek and the exemplar has shadow on the left.

resembles the input image, but still cannot accommodate the lighting direction difference between the input and the exemplar. Utilizing bilateral symmetry addresses this problem. The result obtained using all three terms achieves the best visual effects and ensures that colors are transferred from semantically compatible regions. Nevertheless, it does not have the same texture style as in the exemplar image. In addition, color bleeding artifacts along the boundaries of the face can be spotted: see Figure 2.3 (h). They are mainly caused by two reasons: 1) there are no suitable correspondence in the warped exemplar or the mirror-warped exemplar for certain pixels; 2) the automatically constructed control line segments are not precise enough to perfectly align the high contrast contours in the input and exemplar images, and hence some pixels may be mapped to the background or to a wrong region. It is noteworthy that these artifacts become less noticeable after the patch-based texture transfer step.

2.3.2 Edge-Preserving Texture Transfer

From a signal processing viewpoint, the synthesis of paintings and artworks can be considered as a texture synthesis problem. One thing to be noted is that texture details may be evident at many scales, and the textures at each scale may have distinct characteristics. Hence, we need a texture synthesis approach that can deal with texture structures at multiple scales.

The problem gets more complicated for texture synthesis on face portraits since the geometry and structure of human faces introduce additional constrains. Edges such as chin line, hairline, eye/eyebrow boundaries, and mouth/nose curves are essential for keeping face identities. Previous texture synthesis/transfer approaches mainly focus on avoiding

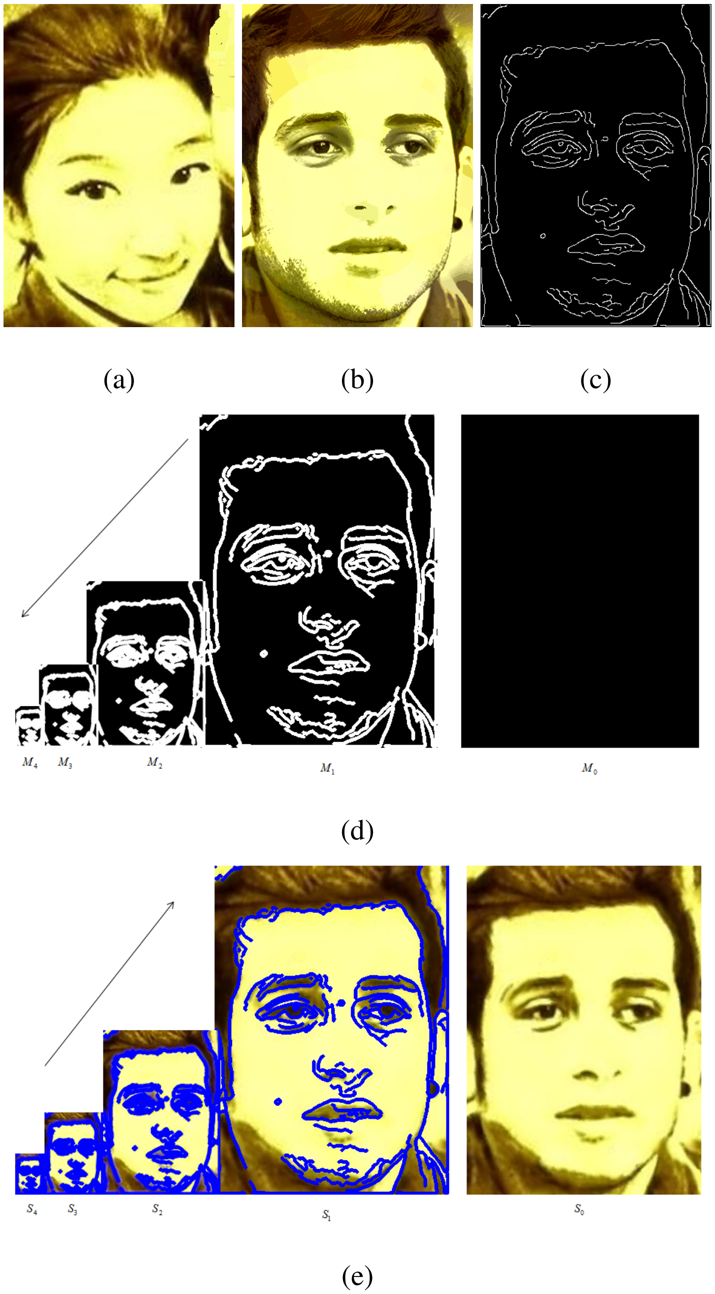


Figure 2.4: Illustration of Edge-Preserving Texture Synthesis: (a) Exemplar image; (b) color transfer result, which is used to initialize S; (c) edge map; (d) edge mask pyramid at different levels; (e) the corresponding synthesis results (masked pixels are marked with blue). Note how texture details in the hair region are generated at coarse levels.

artifacts and minimizing intensity differences between the input and the exemplar, but not so on preserving edges and contours [22, 21, 65]. Hence, we here introduce an edge-preserving texture synthdge-preserving texture synthesis approach, which is extended from Image Melding [17].

We begin with a brief explanation of the optimization-based texture synthesis [50], which both Image Melding [17] and our Edge-Preserving Texture Transfer (EPTT) are based on. This approach optimizes an objective function that measures the similarity between the synthesized texture S and the exemplar E over a set of overlapping local patches. This objective function takes the following form:

$$S^* = \arg\min_{S} \sum_{s_i \in S} \Big(\min_{e_j \in E} \big(D(s_i, e_j) + \lambda D(\Delta s_i, \Delta e_j) \big) \Big), \tag{2.4}$$

where s_i refers to the square patch (10×10 pixels in our implementation) in S whose top-left corner is at pixel i. e_j is a patch of the same size in E. Distance between two patches, $D(\cdot, \cdot)$ is measured as the sum of squared color difference and $D(\Delta\cdot, \Delta\cdot)$ measures the sum of squared gradient differences. λ is the coefficient for the gradient term, typically set as 1.0. The color difference is measured in the CIELAB color space since it approximates human perception. Two additional gradient channels for horizontal and vertical gradients based on the L value are computed using Sobel operator.

As shown in Eq. 2.4, to evaluate the quality of a given synthesized image S, we need to enumerate all overlapping patches s_i in S, find their nearest neighbors in the exemplar E, and sum together the total distances between the corresponding patches. The image S^* that yields the smallest total distance is the desired synthesis result. To optimize this objective function, two steps are alternatively performed. The first step finds the approximate nearest neighbors for all patches in S using the Generalized PatchMatch algorithm [9]. The second step, referred to as voting, updates S by averaging together the overlapping nearest neighbor patches found from exemplar E [111]. The above two steps are repeated until the termination condition is met.

In our practice, we use the result of color transfer T to initialize S, which guides the synthesis result to follow the shape of the input face. Since the original texture synthesis approach does not preserve important facial contours very well, we modified the procedure to make it better preserving edges during the synthesis. In addition, a coarse-to-fine processing scheme is employed so that structures at different scales can be properly handled.

As shown in Figure 2.4, given the input image, we first compute an edge map using Canny Edge Detector with automatically selected thresholds [32]. An edge mask pyramid is then generated by downsizing the edge map by a factor of two each time. During the downsizing, a pixel in the coarse level is labeled as an edge pixel, as long as one of the four corresponding pixels in the finer level is an edge pixel. Texture synthesis is then performed at the coarsest level first for non-edge pixels only. Once the synthesis process converges, we move to the next finer level to repeat the synthesis process; see Algorithm 1. As shown in Figure 2.4, such a coarse-to-fine processing scheme brings two benefits: 1) texture pattens at different scales can be properly synthesized; and 2) areas close to facial contours are processed at finer levels only and hence the edge structured in the input image can be properly preserved.

The overall algorithm is given in Algorithm 1. It shows that within each resolution level, the nearest neighbor searching and voting are repeated alternatively to update S^k. After each iteration, T^k is used to reset S^k for areas masked by M^k and Poisson blending [77] is employed. Specifically, we view masked regions of T^k as destination and unmasked

Algorithm 1 Edge Preserving Texture Synthesis

```
Input: color transfer result $T$, edge map $M$, exemplar $E$;
Output: texture synthesis result $S$;
for each scale $k = 1$ to $n$ do
    downsample $T^{k-1}$, $E^{k-1}$, and $M^{k-1}$ by a factor of 2 to obtain $T^k$, $E^k$, and $M^k$;
end for
for each scale $k = n$ to 0 do
    if $k$ equals $n$ then
        initialize $S^n$ using $T^n$;
    else
        initialize $S^k$ by scaling $S^{k+1}$;
    end if
    repeat
        compute the nearest neighbor field between $S^k$ and $E^k$;
        update $S^k$ by voting for areas not masked by $M^k$;
        set $S^k$ to $T^k$ for areas masked by $M^k$;
        apply Poisson blending to remove seams between the two areas;
    until the changes to $S^k$ is below a threshold or a specific number of repetitions is reached;
end for
Synthesize the edge pixels using the inverse of the edge map $M$ as mask;
```

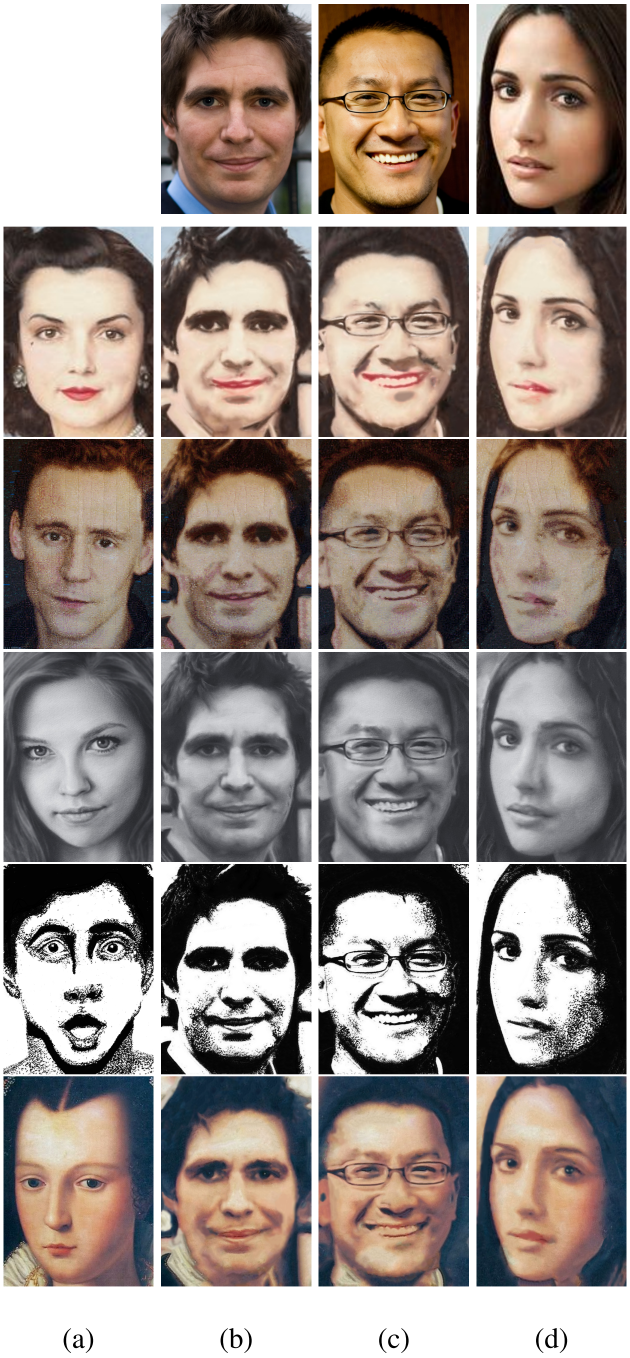


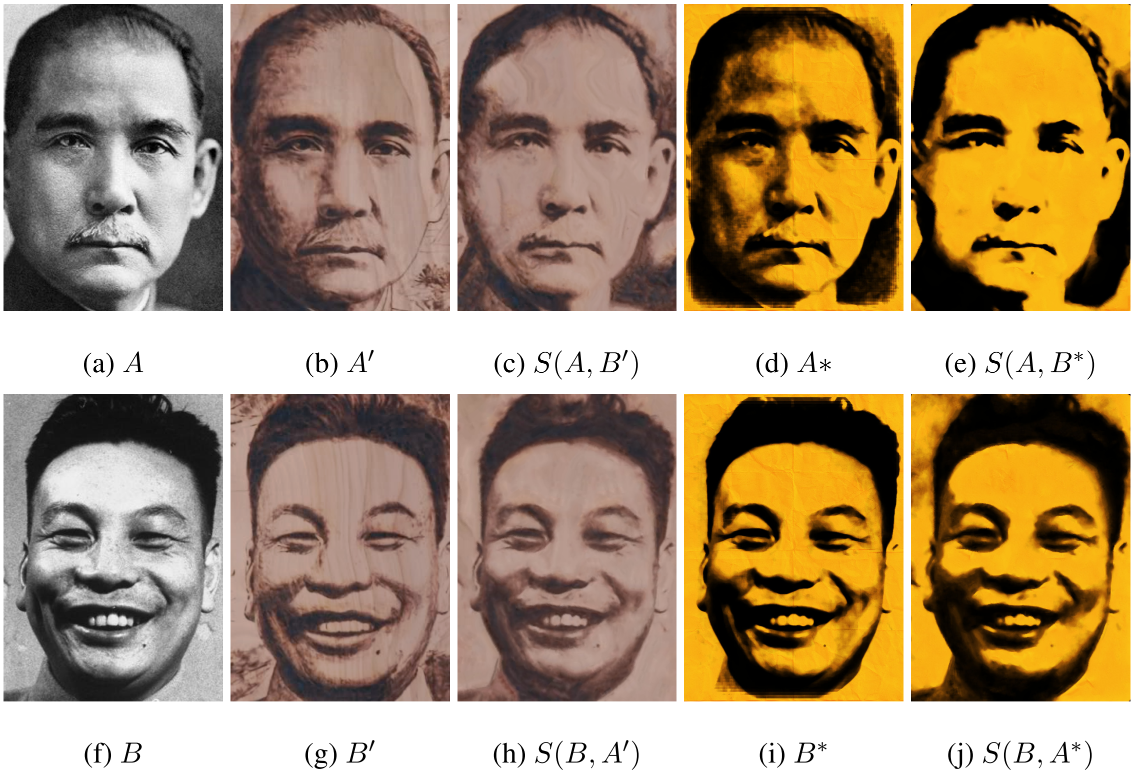
Figure 2.5: Experiment results by our approach. Top row are input images. For other rows, each represents an exemplar (a) and the corresponding stylized results (b-d). (b-d) illustrates the original input image at the top and corresponding stylized results below it. In specific, (b) demonstrates a face with cluttered background. (c) shows a face wearing glasses. (d) illustrates a non-frontal face.

regions of S^k as source, then seamlessly clone the source to the destination by interpolating destination pixels with a guidance gradient field directly taken from the source. As a result, details along edge structures captured in T^k and the synthesized texture in S^k can be seamlessly blended together.

2.4 Experiment Results

Figure 2.5 shows our style transfer for common art styles. The input images are selected from public dataset supplied by Shih et al. [92]. They are of numerous sources (captured under uncontrolled lightening condition and with different devices) and of various subjects (gender, race, beards, age, glasses, pose, facial expression and hair style). Further, the photos could be noisy, and the background can be cluttered which makes the dataset challenging. The exemplar images are collected from the Internet. They are various in colors, tone and texture; and the styles range across oil painting, water color painting, mosaic style, pencil drawing and stippling. The stylization results demonstrate that our approach successfully transfers the color, texture and tone from the exemplars to the inputs. The visual effects are convincing even when the input and the exemplar are significantly disparate, e.g., a face wearing glasses (Figure 2.5(c)) and a non-frontal face (Figure 2.5(d)) are properly stylized by frontal exemplar faces without glasses. Nevertheless, limitations can be found in some synthesis results, e.g. texture structures are not well preserved in the third row of Figure 2.5, especially for Figure 2.5(d), where artifacts are shown.

In addition, we located two wood pyrography and the photos that the artist based on [1]. Through using the pyrography of A to stylize the photo of B, we can compare our exemplar-based stylization results with the real artworks, which can be treated at the "ground truth".



(a) A (b) A' (c) $S(A, B')$ (d) $A*$ (e) $S(A, B^*)$

(f) B (g) B' (h) $S(B, A')$ (i) B^* (j) $S(B, A^*)$

Figure 2.6: Comparison between our automatic stylization results with real artworks and the results of style-specific approaches. A and B are input images. A' and B' are wood pyrography created by the same artist based on A and B [1]. A^* and B^* are automatically stylized using Rollip [2]. $S(x, y)$ denotes the output of our algorithm using image x as input and image y as exemplar.

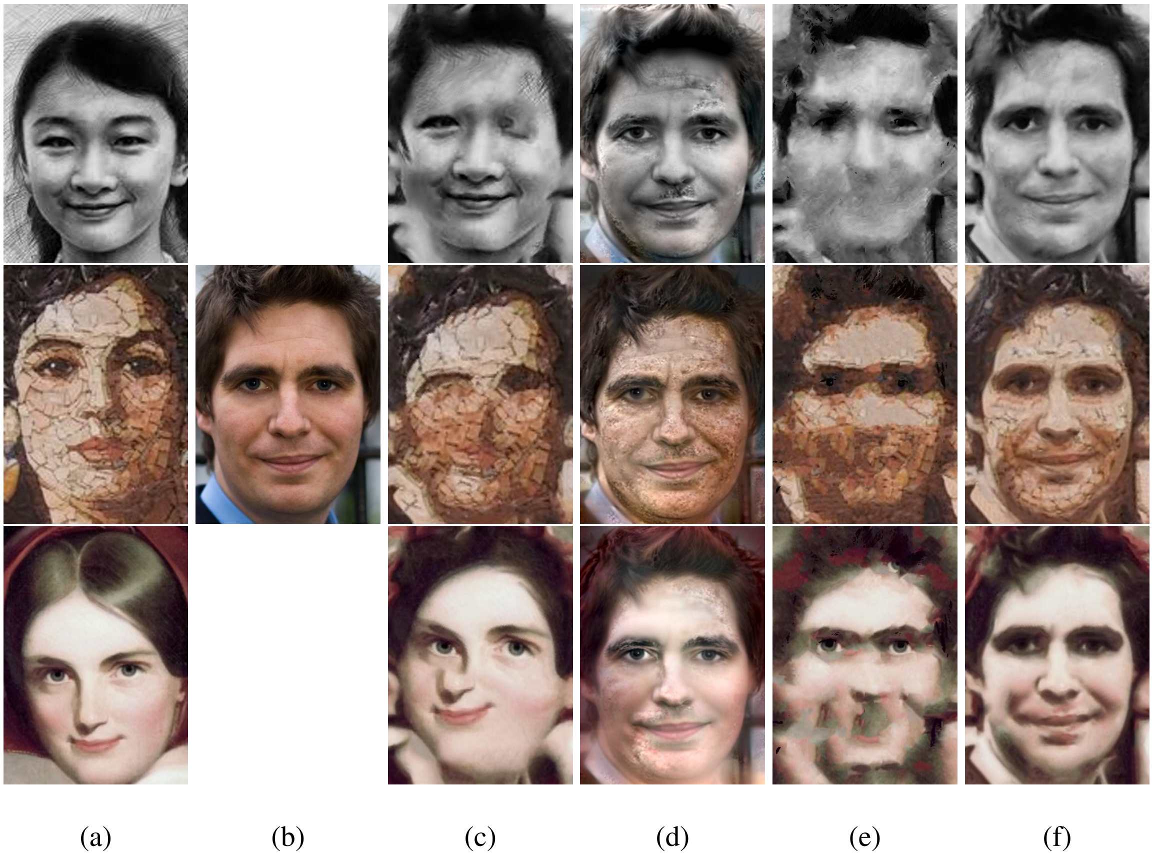


Figure 2.7: Comparison to other methods. (a) Exemplars. (b) Input image, which is the same for all three rows. (c) Image Melding [17] initiated with our SACT. (d) Shih et al. [92] (e) Our SACT + Image Quilting [21]. (f) Ours.

Similarly, we can compare our outputs with those of a style-specific synthesis approach [2]. Figure 2.6 shows that in both cases, our results resemble those created by the artist and the style-specific algorithm, even though our approach uses a different image with the same style as the exemplar. However, it could be spotted that fine edges and detailed textures are not very well preserved, which is due to the voting procedure during texture synthesis.

Figure 2.7 further compares our method with related work in style transfer [21, 92, 17]. Image quilting is originally applied to texture transfer [21]. We ran their code directly on our task (Figure 2.7 (e)). The method proposed by Shih et al. [92] is initially designed for photorealistic style transfer. Here we implement their approach and apply it for our non-photorealistic style transfer. Image Melding can be applied to numerous tasks, such as image completion, texture interpolation, image blending and texture preserving warping [17]. We adapted it for our style transfer task by viewing our task as a single-source image synthesis problem, in which the exemplar also serves as the source (please refer to [92] for details). Here the color transfer result T is set as the initial solution.

As shown in Figure 2.7, our method achieved visually pleasing results, whereas others either lost the identity of the input face or entail apparent artifacts. For example, Image Melding [17] severely distorted the face. The method by Shih et al. [92] blurred the edges and failed to transfer the texture properly. In Image Quilting result, the majority of face contours were broken.

2.5 Summary

A novel algorithm is presented for single-exemplar face stylization. The algorithm can maintain the identity, content and structure of the input face, while imposing the style of

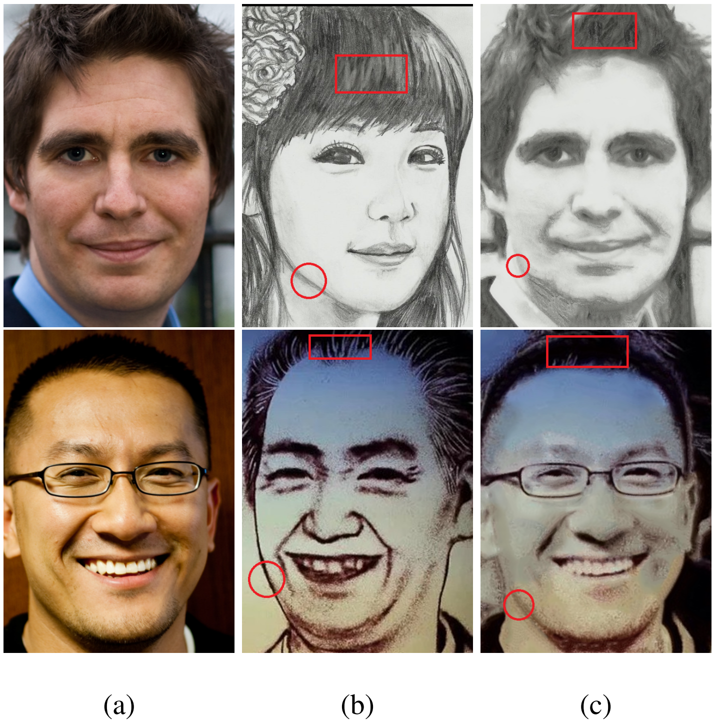


Figure 2.8: Limitations of our approach: (a) Input. (b) Exemplar. (c) Results by our method. Red rectangles highlight zones where textures are poorly transferred. The detailed strokes featured in rectangular zones are not transferred due to lack of color variation in the corresponding hair regions of the input image. Circular zones are featured with sharp edges strokes, which are lost in the stylized results since the corresponding contours in the input images have much lower color contrast.

the exemplar. The robustness of our method is tested using inputs of varieties of subjects. We achieve visually convincing effects for a variety of art styles, which include certain styles of pencil drawing, oil painting, mosaic, sand painting, stippling, water color painting, and pyrography. Even though at fine level, textures such as brush stokes or stipple dots generated by our approach may not be as clean or precisely structured as those obtained by style-specific approaches, our approach has its merits in terms of flexibility and extendability. Qualitative evaluation is performed using both real artworks and outputs of a style-specific synthesis algorithm. The comparison with related methods also demonstrates the advantage of our approach on face stylization tasks.

Whereas, our method fails to transfer styles with sharp lines or curves. As shown in Figure 2.8, the sharp lines and edges alone the chin or from the hair region are not successfully transferred. These are probably due to nature of optimized-based texture synthesis, which attempts to mix all the nearest neighbors.

Chapter 3

Unsupervised Learning for Image-to-Image Translation

While the algorithm presented in the previous Chapter has the merit of requiring only a single exemplar, it is constrained to face photos. To perform generalized image-to-image translation, here a learning-based method is presented based on Conditional Generative Adversarial Networks (conditional GANs). Conditional GANs for cross-domain image-to-image translation have made much progress recently [54, 58, 107, 70, 39, 98]. Depending on the task complexity, thousands to millions of labeled image pairs are needed to train a conditional GAN. However, human labeling is expensive, even impractical, and large quantities of data may not always be available. Inspired by dual learning from natural language translation [115], we develop a novel *dual-GAN* mechanism, which enables image translators to be trained from two sets of *unlabeled* images from two domains. In our architecture, the primal GAN learns to translate images from domain U to those in domain V, while the dual GAN learns to invert the task. The closed loop made by the primal and dual tasks allows images from either domain to be translated and then reconstructed. Hence a loss function that accounts for the reconstruction error of images can be used to train the translators. Experiments on multiple image translation tasks with unlabeled data show considerable performance gain of DualGAN over a single GAN. For some tasks, DualGAN can even achieve comparable or slightly better results than conditional GAN trained on fully labeled data.

3.1 Overview

Many image processing and computer vision tasks, e.g., image segmentation, stylization, and abstraction, can be posed as image-to-image translation problems [39], which convert one visual representation of an object or scene into another. Conventionally, these tasks have been tackled separately due to their intrinsic disparities [54, 58, 107, 70, 39, 98]. It is not until the past two years that general-purpose and end-to-end deep learning frameworks, most notably those utilizing fully convolutional networks (FCNs) [68] and conditional generative adversarial nets (cGANs) [39], have been developed to enable a *unified* treatment of these tasks.

Up to date, these general-purpose methods have all been supervised and trained with a large number of *labeled* and *matching* image *pairs*. In practice however, acquiring such training data can be time-consuming (e.g., with pixelwise or patchwise labeling) and even unrealistic. For example, while there are plenty of photos or sketches available, photo-sketch image pairs depicting the same people under the same pose are scarce. In other image translation settings, e.g., converting daylight scenes to night scenes, even though labeled and matching image pairs can be obtained with stationary cameras, moving objects in the scene often cause varying degrees of content discrepancies.

In this chapter, we aim to develop an *unsupervised* learning framework for general-purpose image-to-image translation, which only relies on *unlabeled* image data, such as two sets of photos and sketches for the photo-to-sketch conversion task. The obvious technical challenge is how to train a translator without any data characterizing correct translations. Our approach is inspired by *dual learning* from natural language processing [115]. Dual learning trains *two* "opposite" language translators (e.g., English-to-French and French-to-English) simultaneously by minimizing the *reconstruction loss* resulting from a *nested* application of the two translators. The two translators represent a primal-dual pair and the nested application forms a closed loop, allowing the application of reinforcement learning. Specifically, the reconstruction loss measured over monolingual data (either English or French) would generate informative feedback to train a bilingual translation model.

Our work develops a dual learning framework for image-to-image translation for the first time and differs from the original NLP dual learning method of Xia et al. [115] in two main aspects. First, the NLP method relied on pre-trained (English and French) language models to indicate how confident the the translator outputs are natural sentences in their respective target languages. With general-purpose processing in mind and the realization that such pre-trained models are difficult to obtain for many image translation tasks, our work develops GAN discriminators [26] that are trained adversarially with the translators to capture domain distributions. Hence, we call our learning architecture *DualGAN*. Furthermore, we employ FCNs as translators which naturally accommodate the 2D structure of images, rather than sequence-to-sequence translation models such as LSTM or Gated

Recurrent Unit (GUT).

Taking two sets of unlabeled images as input, each characterizing an image domain, DualGAN simultaneously learns two reliable image translators from one domain to the other and hence can operate on a wide variety of image-to-image translation tasks. The effectiveness of DuanGAN is validated through comparison with both GAN (with an image-conditional generator and the original discriminator) and conditional GAN [39]. The comparison results demonstrate that, for some applications, DualGAN can outperform supervised methods trained on labeled data.

3.2 Related Work

Since the seminal work by Goodfellow et al. [26] in 2014, a series of GAN-family methods have been proposed for a wide variety of problems. The original GAN can learn a generator to capture the distribution of real data by introducing an adversarial discriminator that evolves to discriminate between the real data and the fake [26]. Soon after, various conditional GANs (cGAN) have been proposed to condition the image generation on class labels [71], attributes [76, 117], texts [80], and images [54, 58, 107, 70, 39, 98].

Most image-conditional models were developed for specific applications such as super-resolution [54], texture synthesis [58], style transfer from normal maps to images [107], and video prediction [70], whereas few others were aiming for general-purpose processing [39, 98]. The general-purpose solution for image-to-image translation proposed by Isola et al. [39] requires significant number of labeled image pairs. The unsupervised mechanism for cross-domain image conversion presented by Taigman et al. [98] can train an image-conditional generator without paired images, but relies on a sophisticated pre-trained function that maps images from either domain to an intermediate representation, which requires labeled data in other formats.

Dual learning was first proposed by Xia et al. [115] to reduce the requirement on labeled data in training English-to-French and French-to-English translators. The French-to-English translation is the dual task to English-to-French translation, and they can be trained side-by-side. The key idea of dual learning is to set up a dual-learning game which involves two agents, each of whom only understands one language, and can evaluate how likely the translated are natural sentences in targeted language and to what extent the reconstructed are consistent with the original. Such a mechanism is played alternatively on both sides, allowing translators to be trained from monolingual data only.

Despite of a lack of parallel bilingual data, two types of feedback signals can be generated: the membership score which evaluates the likelihood of the translated texts belonging to the targeted language, and the reconstruction error that measures the disparity between the reconstructed sentences and the original. Both signals are assessed with the assistance of application-specific domain knowledge, i.e., the pre-trained English and French language models.

In our work, we aim for a general-purpose solution for image-to-image conversion and hence do not utilize any domain-specific knowledge or pre-trained domain representations. Instead, we use a domain-adaptive GAN discriminator to evaluate the membership score of translated samples, whereas the reconstruction error is measured as the mean of absolute difference between the reconstructed and original images within each image domain.

In CycleGAN, a concurrent work by Zhu et al. [125], the same idea for unpaired image-to-image translation is proposed, where the primal-dual relation in DualGAN is referred to as a cyclic mapping and their *cycle consistency loss* is essentially the same as our reconstruction loss. Superiority of CycleGAN has been demonstrated on several tasks where paired training data hardly exist, e.g., in object transfiguration and painting style and season transfer.

Recent work by Liu and Tuzel [67], which we refer to as coupled GAN or CoGAN, also trains two GANs together to solve image translation problems without paired training data. Unlike DualGAN or CycleGAN, the two GANs in CoGAN are not linked to enforce cycle consistency. Instead, CoGAN learns a joint distribution over images from two domains. By sharing weight parameters corresponding to high level semantics in both generative and discriminative networks, CoGAN can enforce the two GANs to interpret these image semantics in the same way. However, the weight-sharing assumption in CoGAN and similar approaches, e.g., [7, 66], does not lead to effective general-purpose solutions as its applicability is task-dependent, leading to unnatural image translation results, as shown in comparative studies by CycleGAN [125].

DualGAN and CycleGAN both aim for general-purpose image-to-image translations without requiring a joint representation to bridge the two image domains. In addition, DualGAN trains both primal and dual GANs at the same time, allowing a reconstruction error term to be used to generate informative feedback signals.

3.3 Method

Given two sets of unlabeled and unpaired images sampled from domains U and V, respectively, the primal task of DualGAN is to learn a generator $G_A : U \to V$ that maps an image $u \in U$ to an image $v \in V$, while the dual task is to train an inverse generator $G_B : V \to U$. To realize this, we employ two GANs, the primal GAN and the dual GAN. The primal

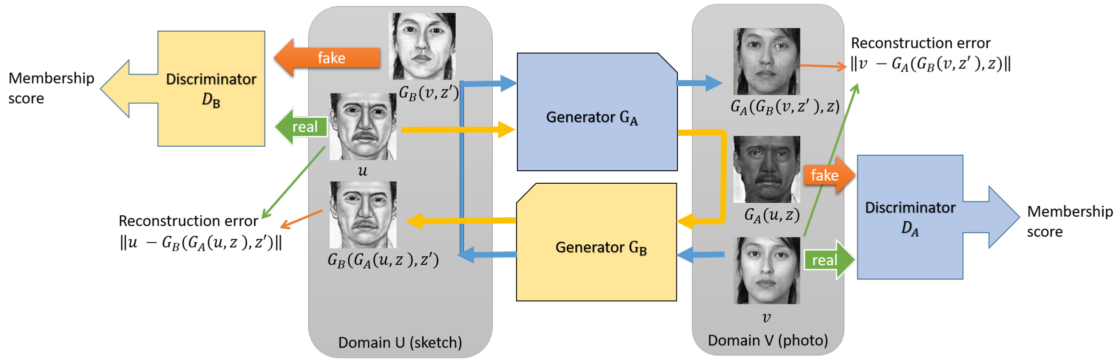


Figure 3.1: Network architecture and data flow chart of DualGAN for image-to-image translation.

GAN learns the generator G_A and a discriminator D_A that discriminates between G_A's fake outputs and real members of domain V. Analogously, the dual GAN learns the generator G_B and a discriminator D_B. The overall architecture and data flow are illustrated in Figure 3.1.

As shown in Figure 3.1, image $u \in U$ is translated to domain V using G_A. How well the translation $G_A(u, z)$ fits in V is evaluated by D_A, where z is random noise, and so is z' that appears below. $G_A(u, z)$ is then translated back to domain U using G_B, which outputs $G_B(G_A(u, z), z')$ as the reconstructed version of u. Similarly, $v \in V$ is translated to U as $G_B(v, z')$ and then reconstructed as $G_A(G_B(v, z'), z)$. The discriminator D_A is trained with v as positive samples and $G_A(u, z)$ as negative examples, whereas D_B takes u as positive and $G_B(v, z')$ as negative. Generators G_A and G_B are optimized to emulate "fake" outputs to blind the corresponding discriminators D_A and D_B, as well as to minimize the two *reconstruction losses* $\|G_A(G_B(v, z'), z) - v\|$ and $\|G_B(G_A(u, z), z') - u\|$.

3.3.1 Objective

As in the traditional GAN, the objective of discriminators is to discriminate the generated fake samples from the real ones. Nevertheless, here we use the loss format advocated by Wasserstein GAN (WGAN) [5] rather than the sigmoid cross-entropy loss used in the original GAN [26]. It is proven that the former performs better in terms of generator convergence and sample quality, as well as in improving the stability of the optimization [5]. The corresponding loss functions used in D_A and D_B are defined as:

$$l_A^d(u,v) = D_A(G_A(u,z)) - D_A(v), \tag{3.1}$$

$$l_B^d(u,v) = D_B(G_B(v,z')) - D_B(u), \tag{3.2}$$

where $u \in U$ and $v \in V$.

The same loss function is used for both generators G_A and G_B as they share the same objective. Previous works on conditional image synthesis found it beneficial to replace L_2 distance with L_1, since the former often leads to blurriness [52, 115]. Hence, we adopt L_1 distance to measure the recovery error, which is added to the GAN objective to force the translated samples to obey the domain distribution:

$$\begin{aligned} l^g(u,v) = \lambda_U \|u - G_B(G_A(u,z),z')\| + \\ \lambda_V \|v - G_A(G_B(v,z'),z)\| \\ -D_B(G_B(v,z')) - D_A(G_A(u,z)), \end{aligned} \tag{3.3}$$

where $u \in U$, $v \in V$, and λ_U, λ_V are two constant parameters. Depending on the application, λ_U and λ_V are typically set to a value within $[100.0, 1,000.0]$. If U contains natural images and V does not (e.g., aerial photo-maps), we find it more effective to use smaller λ_U than λ_V.

3.3.2 Network configuration

DualGAN is constructed with identical network architecture for G_A and G_B. The generator is configured with equal number of downsampling (pooling) and upsampling layers. In addition, we configure the generator with skip connections between mirrored downsampling and upsampling layers as in [83, 39], making it a U-shaped net. Such a design enables low-level information to be shared between input and output, which is beneficial since many image translation problems implicitly assume alignment between image structures in the input and output (e.g., object shapes, textures, clutter, etc.). Without the skip layers, information from all levels has to pass through the bottleneck, typically causing significant loss of high-frequency information. Furthermore, similar to [39], we did not explicitly provide the noise vectors z, z'. Instead, they are provided only in the form of dropout and applied to several layers of our generators at both training and test phases.

For discriminators, we employ the Markovian PatchGAN architecture as explored in [58], which assumes independence between pixels distanced beyond a specific patch size and models images only at the patch level rather than over the full image. Such a configuration is effective in capturing local high-frequency features such as texture and style, but less so in modeling global distributions. It fulfills our needs well, since the recovery loss encourages preservation of global and low-frequency information and the discriminators are designated to capture local high-frequency information. The effectiveness of this configuration has been verified on various translation tasks [115]. Similar to [115], we run this discriminator convolutionally across the image, averaging all responses to provide the ultimate output. An extra advantage of such a scheme is that it requires fewer parameters, runs faster, and has no constraints over the size of the input image. The patch size at which the discriminator operates is fixed at 70×70, and the image resolutions were mostly 256×256, same as pix2pix [39].

3.3.3 Training procedure

To optimize the DualGAN networks, we follow the training procedure proposed in WGAN [5]; see Algorithm 2. We train the discriminators n_{critic} steps, then one step on generators. We employ mini-batch Stochastic Gradient Descent and apply the RMSProp solver, as momentum based methods such as Adam would occasionally cause instability [5], and RMSProp is known to perform well even on highly non-stationary problems [99, 5]. We typically set the number of critic iterations per generator iteration n_{critic} to $\{2, 3, 4\}$ and assign batch size to 1-4, without noticeable differences on effectiveness in the experiments. The clipping parameter c is normally set in $[0.01, 0.1]$, varying by application.

Training for traditional GANs needs to carefully balance between the generator and the discriminator, since, as the discriminator improves, the sigmoid cross-entropy loss is locally saturated and may lead to vanishing gradients. Unlike in traditional GANs, the Wasserstein loss is differentiable almost everywhere, resulting in a better discriminator. At each iteration, the generators are not trained until the discriminators have been trained for n_{critic} steps. Such a procedure enables the discriminators to provide more reliable gradient information [5].

3.4 Experimental Results and Evaluation

To assess the capability of DualGAN in general-purpose image-to-image translation, we conduct experiments on a variety of tasks, including photo-sketch conversion, label-image

Algorithm 2 DualGAN training procedure

```
Require: Image set $U$, image set $V$, GAN $A$ with generator parameters $\theta_A$ and discriminator parameters $\omega_A$, GAN $B$ with generator parameters $\theta_B$ and discriminator parameters $\omega_B$, clipping parameter $c$, batch size $m$, and $n_{critic}$
Randomly initialize $\omega_i$, $\theta_i$, $i \in \{A, B\}$
repeat
    for $t = 1, \ldots, n_{critic}$ do
        sample images $\{u^{(k)}\}_{k=1}^{m} \subseteq U$, $\{v^{(k)}\}_{k=1}^{m} \subseteq V$
        update $\omega_A$ to minimize $\frac{1}{m}\sum_{k=1}^{m} l_A^d(u^{(k)}, v^{(k)})$
        update $\omega_B$ to minimize $\frac{1}{m}\sum_{k=1}^{m} l_B^d(u^{(k)}, v^{(k)})$
        $clip(\omega_A, -c, c)$, $clip(\omega_B, -c, c)$
    end for
    sample images $\{u^{(k)}\}_{k=1}^{m} \subseteq U$, $\{v^{(k)}\}_{k=1}^{m} \subseteq V$
    update $\theta_A$, $\theta_B$ to minimize $\frac{1}{m}\sum_{k=1}^{m} l^g(u^{(k)}, v^{(k)})$
until convergence
```

translation, and artistic stylization.

To compare DualGAN with GAN and cGAN [39], four labeled datasets are used: PHOTO-SKETCH [108, 123], DAY-NIGHT [51], LABEL-FACADES [101], and AERIAL-MAPS, which was directly captured from Google Map [39]. These datasets consist of corresponding images between two domains; they serve as ground truth (GT) and can also be used for supervised learning. However, none of these datasets could guarantee accurate feature alignment at the pixel level. For example, the sketches in SKETCH-PHOTO dataset were drawn by artists and do not accurately align with the corresponding photos, moving objects and cloud pattern changes often show up in the DAY-NIGHT dataset, and the labels in LABEL-FACADES dataset are not always precise. This highlights, in part, the difficulty in obtaining high quality matching image pairs.

DualGAN enables us to utilize abundant unlabeled image sources from the Web. Two unlabeled and unpaired datasets are also tested in our experiments. The MATERIAL dataset includes images of objects made of different materials, e.g., stone, metal, plastic, fabric, and wood. These images were manually selected from Flickr and cover a variety of illumination conditions, compositions, color, texture, and material sub-types [90]. This dataset was initially used for material recognition, but is applied here for material transfer. The OIL-CHINESE painting dataset includes artistic paintings of two disparate styles: oil and Chinese. All images were crawled from search engines and they contain images with varying quality, format, and size. We reformat, crop, and resize the images for training and evaluation. In both of these datasets, no correspondence is available between images from different domains.

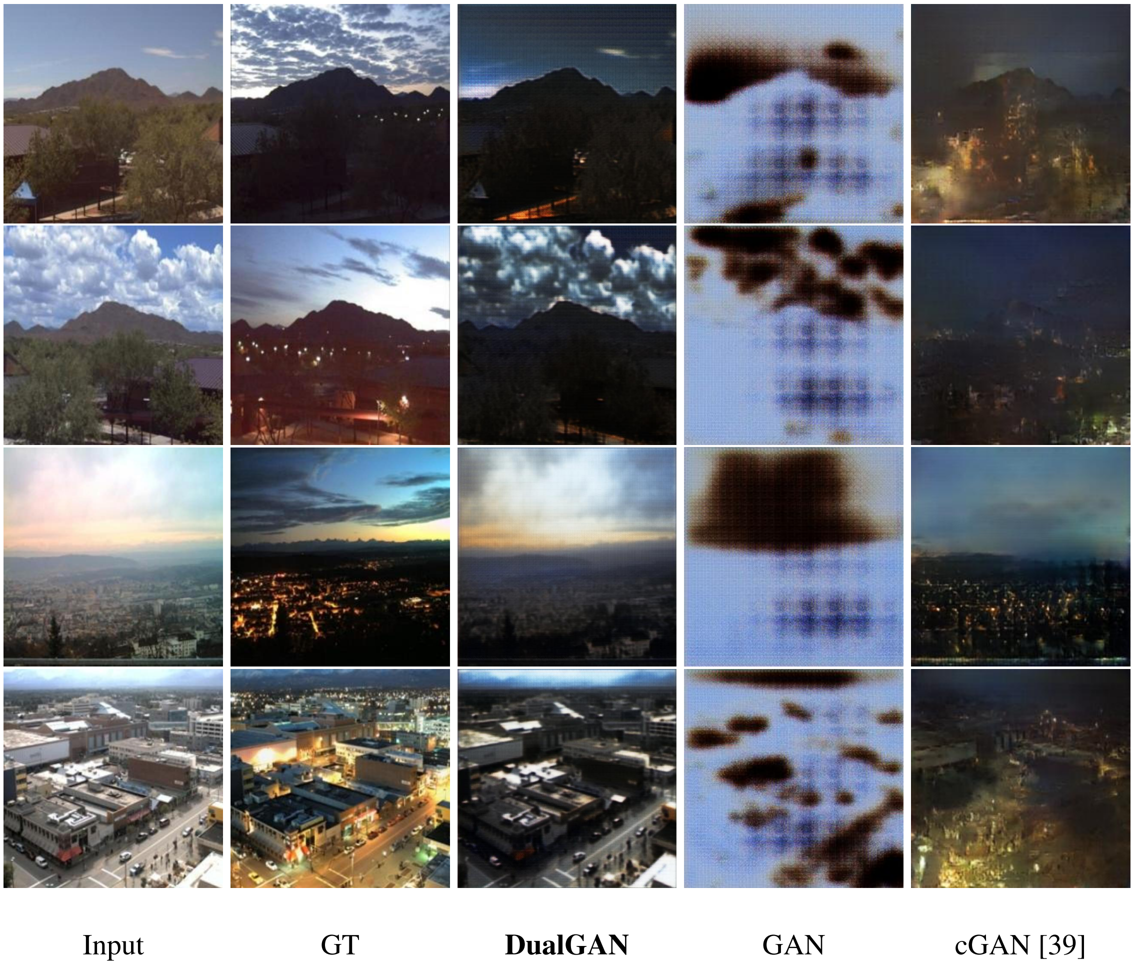


Figure 3.2: Results of day→night translation. cGAN [39] is trained with labeled data, whereas DualGAN and GAN are trained in an unsupervised manner. DualGAN successfully emulates the night scenes while preserving textures in the inputs, e.g., see differences over the cloud regions between our results and the ground truth (GT). In comparison, results of cGAN and GAN contain much less details.

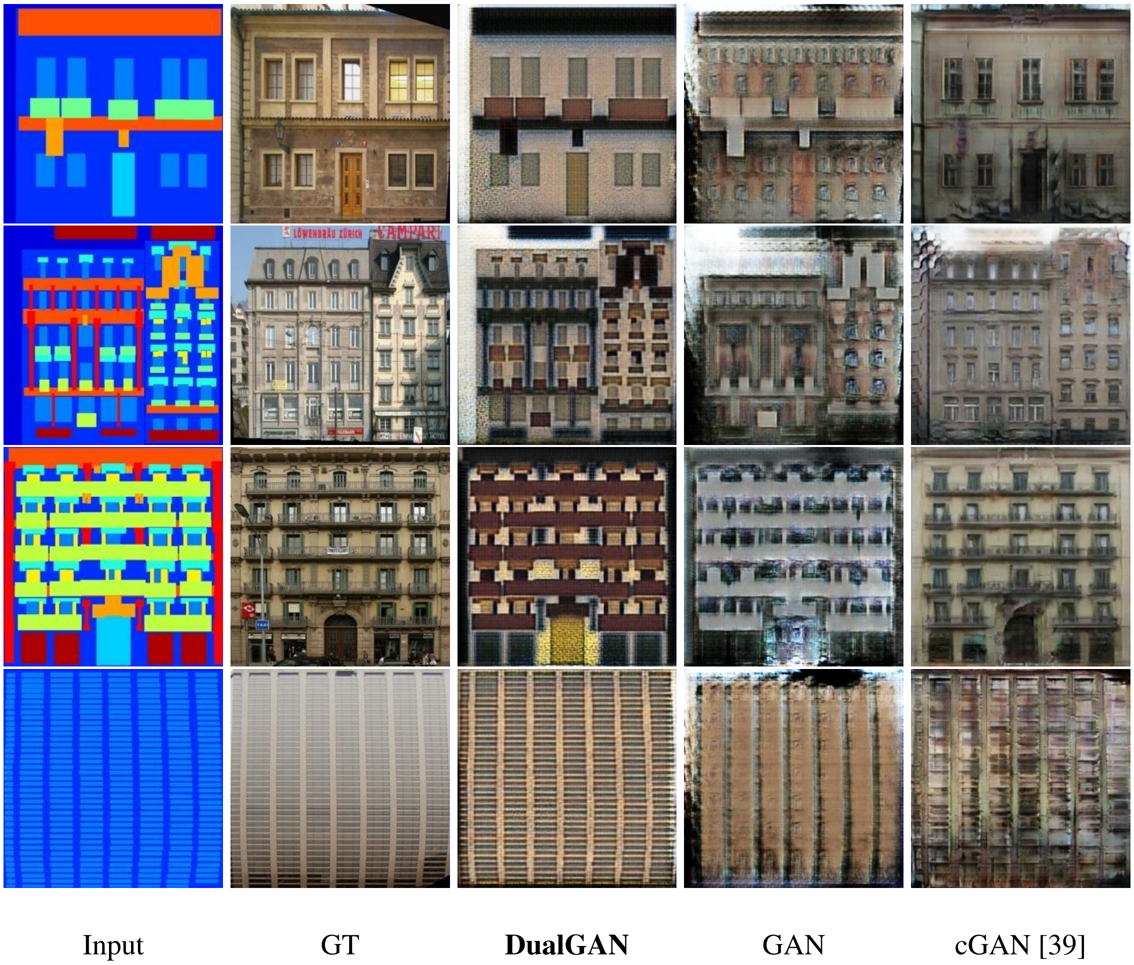


Figure 3.3: Results of label→facade translation. DualGAN faithfully preserves the structures in the label images, even though some labels do not match well with the corresponding photos in finer details. In contrast, results from GAN and cGAN contain many artifacts. Over regions with label-photo misalignment, cGAN often yields blurry output (e.g., the roof in second row and the entrance in third row).

3.4.1 Qualitative evaluation

Using the four labeled datasets, we first compare DualGAN with GAN and cGAN [39] on the following translation tasks: day→night (Figure 3.2), labels↔facade (Figures 3.3 and 3.10), face photo↔sketch (Figures 3.4 and 3.5), and map↔aerial photo (Figures 3.8 and 3.9). In all these tasks, cGAN was trained with labeled (i.e., paired) data, where we ran the model and code provided in [39] and chose the optimal loss function for each task: L_1 loss for facade→label and $L_1 + cGAN$ loss for the other tasks (see [39] for more details). In contrast, DualGAN and GAN were trained in an unsupervised way, i.e., we decouple the image pairs and then reshuffle the data. The results of GAN were generated using our approach by setting $\lambda_U = \lambda_V = 0.0$ in Eq. (3.3), noting that this GAN is different from the original GAN model [26] as it employs a conditional generator.

All three models were trained on the same training datasets and tested on novel data that does not overlap those for training. All the training were carried out on a single GeForce GTX Titan X GPU. At test time, all models ran in well under a second on this GPU.

Compared to GAN, in almost all cases, DualGAN produces results that are less blurry, contain fewer artifacts, and better preserve content structures in the inputs and capture features (e.g., texture, color, and/or style) of the target domain. We attribute the improvements to the reconstruction loss, which forces the inputs to be reconstructable from outputs through the dual generator and strengthens feedback signals that encodes the targeted distribution.

In many cases, DualGAN also compares favorably over the supervised cGAN in terms of sharpness of the outputs and faithfulness to the input images; see Figures 3.2, 3.3, 3.4, 3.5, and 3.8. This is encouraging since the supervision in cGAN does utilize additional image and pixel correspondences. On the other hand, when translating between photos and semantic-based labels, such as map↔aerial and label↔facades, it is often impossible to infer the correspondences between pixel colors and labels based on targeted distribution alone. As a result, DualGAN may map pixels to wrong labels (see Figures 3.9 and 3.10) or labels to wrong colors/textures (see Figures 3.3 and 3.8).

Figures 3.6 and 3.7 show image translation results obtained using the two unlabeled datasets, including oil↔Chinese, plastic→metal, metal→stone, leather→fabric, as well as wood↔plastic. The results demonstrate that visually convincing images can be generated by DualGAN when no corresponding images can be found in the target domains. As well, the DualGAN results generally contain less artifacts than those from GAN.

More results could be found in Figures 3.11, 3.13, 3.15, 3.14, 3.12, 3.16, 3.17.

3.4.2 Quantitative evaluation

To quantitatively evaluate DualGAN, we set up two user studies through Amazon Mechanical Turk (AMT). The "material perceptual" test evaluates the material transfer results, in which we mix the outputs from all material transfer tasks and let the Turkers choose the best match based on which material they believe the objects in the image are made of. For a total of 176 output images, each was evaluated by ten Turkers. An output image is rated as a success if at least three Turkers selected the target material type. Success rates of various material transfer results using different approaches are summarized in Table 3.1, showing that DualGAN outperforms GAN by a large margin.

In addition, we run the AMT "realness score" evaluation for sketch→photo, label map→facades, maps→aerial photo, and day→night translations. To eliminate potential

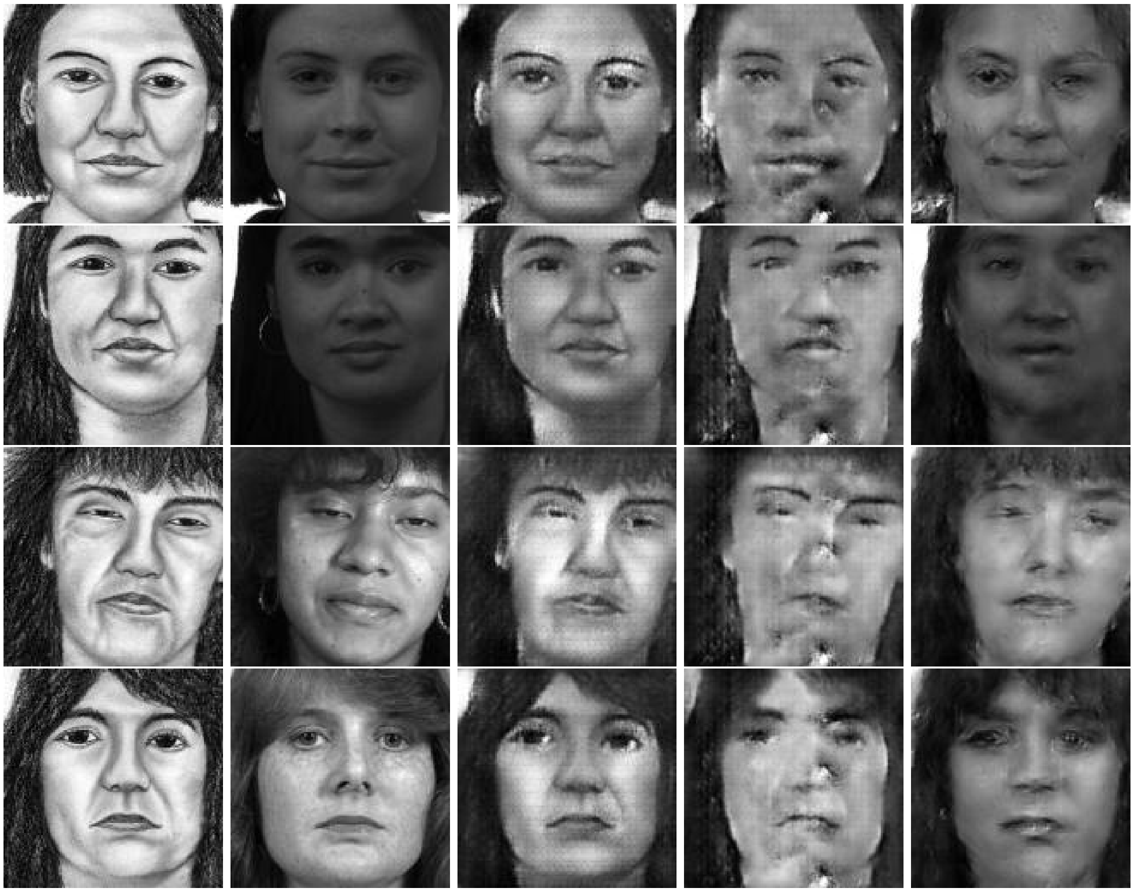


Figure 3.4: Photo→sketch translation for faces. Results of DualGAN are generally sharper than those from cGAN, even though the former was trained using unpaired data, whereas the latter makes use of image correspondence.

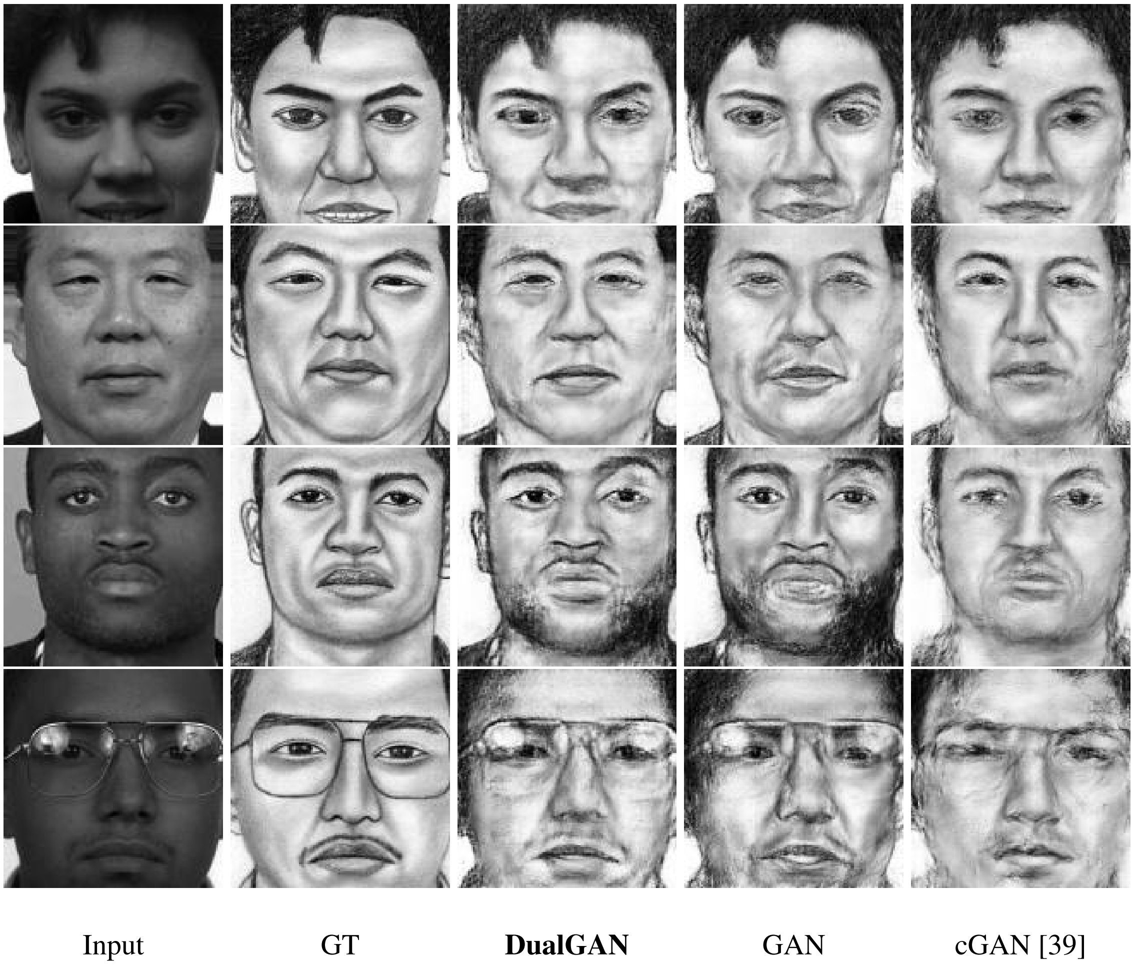


Figure 3.5: Results for sketch→photo translation of faces. More artifacts and blurriness are showing up in results generated by GAN and cGAN than DualGAN.

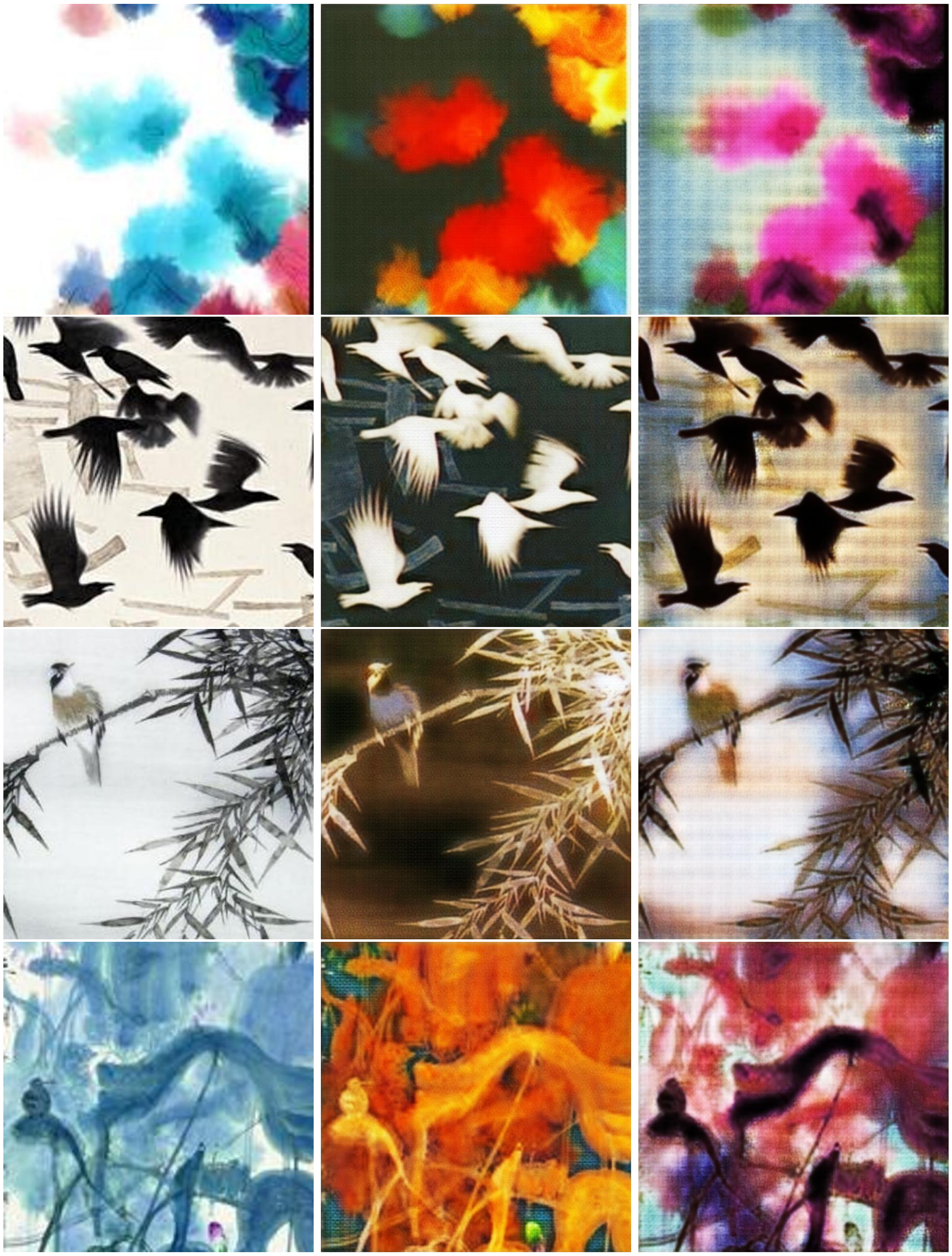
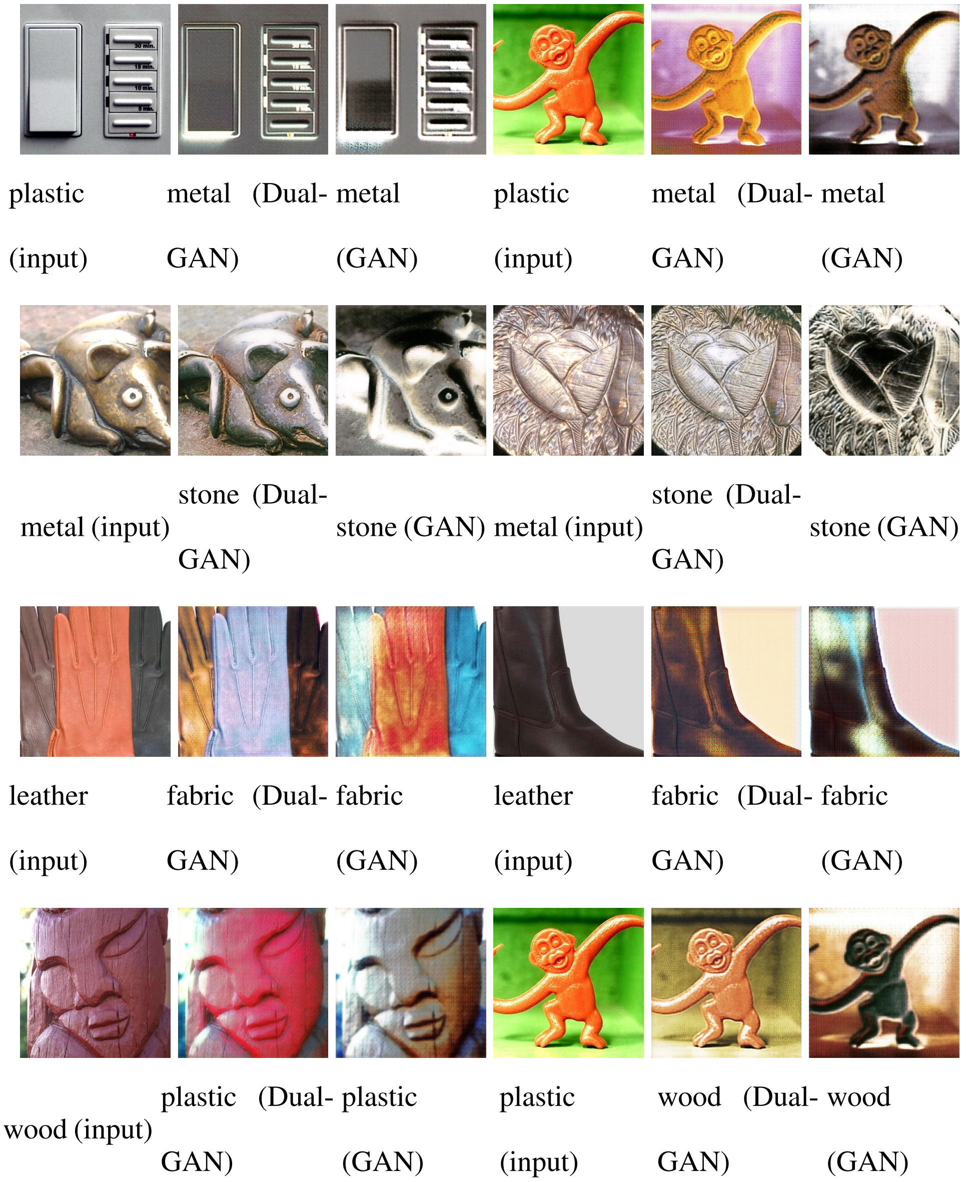


Figure 3.6: Experimental results for translating Chinese paintings to oil paintings (without GT available). The background grids shown in the GAN results imply that the outputs of GAN are not as stable as those of DualGAN.



plastic (input) | metal (Dual-GAN) | metal (GAN) | plastic (input) | metal (Dual-GAN) | metal (GAN)

metal (input) | stone (Dual-GAN) | stone (GAN) | metal (input) | stone (Dual-GAN) | stone (GAN)

leather (input) | fabric (Dual-GAN) | fabric (GAN) | leather (input) | fabric (Dual-GAN) | fabric (GAN)

wood (input) | plastic (Dual-GAN) | plastic (GAN) | plastic (input) | wood (Dual-GAN) | wood (GAN)

Figure 3.7: Experimental results for various material transfer tasks. From top to bottom, plastic→metal, metal→stone, leather→fabric, and plastic↔wood.

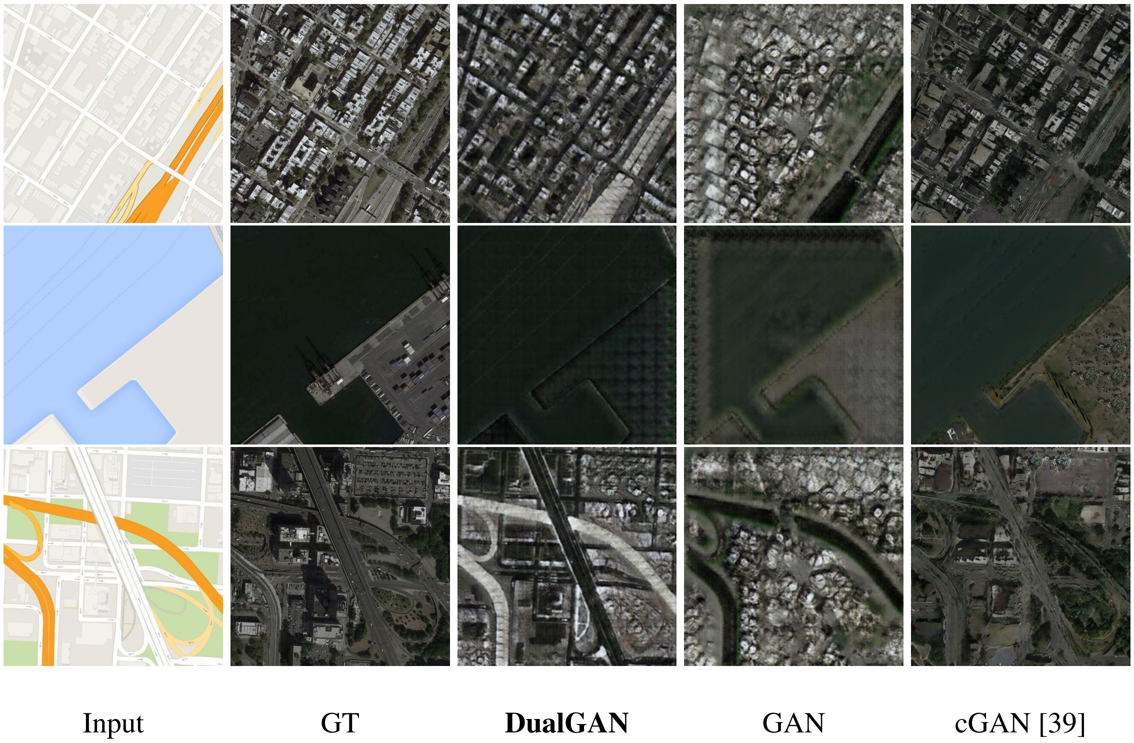


Figure 3.8: Map→aerial photo translation. Without image correspondences for training, DualGAN may map the orange-colored interstate highways to building roofs with bright colors. Nevertheless, the DualGAN results are sharper than those from GAN and cGAN.

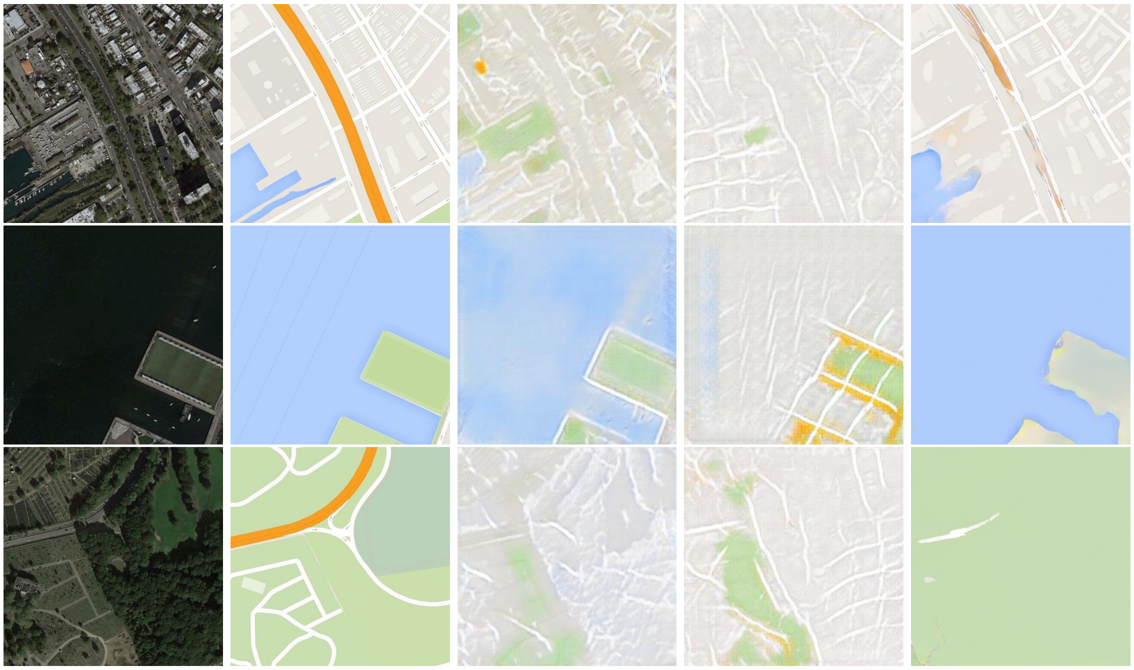
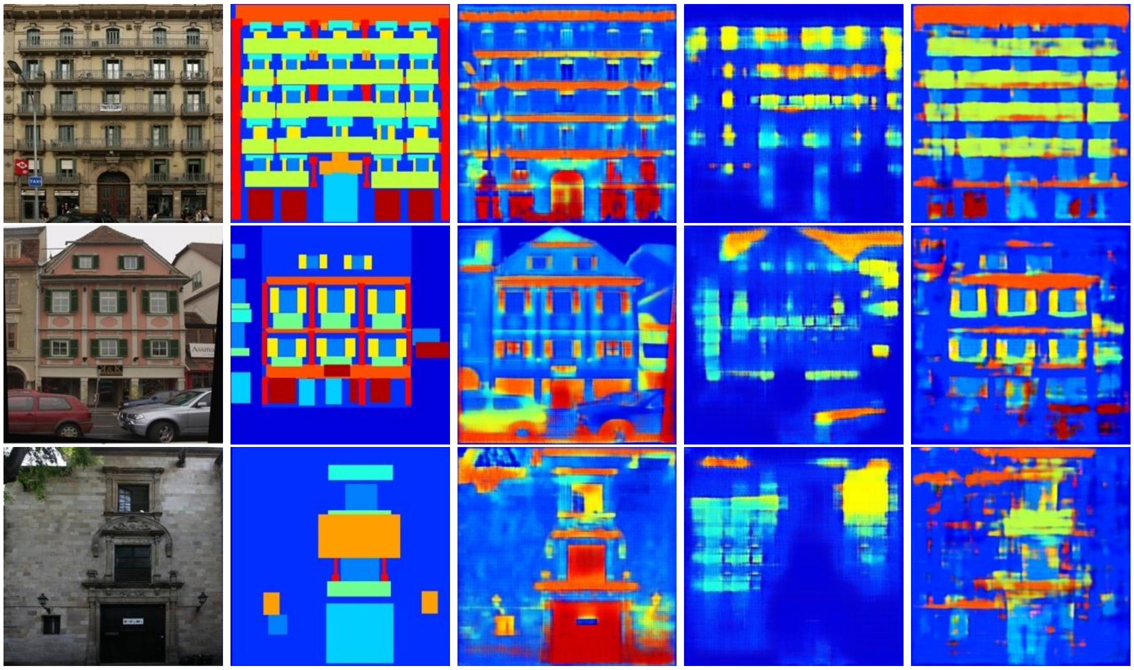


Figure 3.9: Results for aerial photo→map translation. DualGAN performs better than GAN, but not as good as cGAN. With additional pixel correspondence information, cGAN performs well in terms of labeling local roads, but still cannot detect interstate highways.



Input GT **DualGAN** GAN) cGAN [39]

Figure 3.10: Facades→label translation. While cGAN correctly labels various bulding components such as windows, doors, and balconies, the overall label images are not as detailed and structured as DualGAN's outputs.

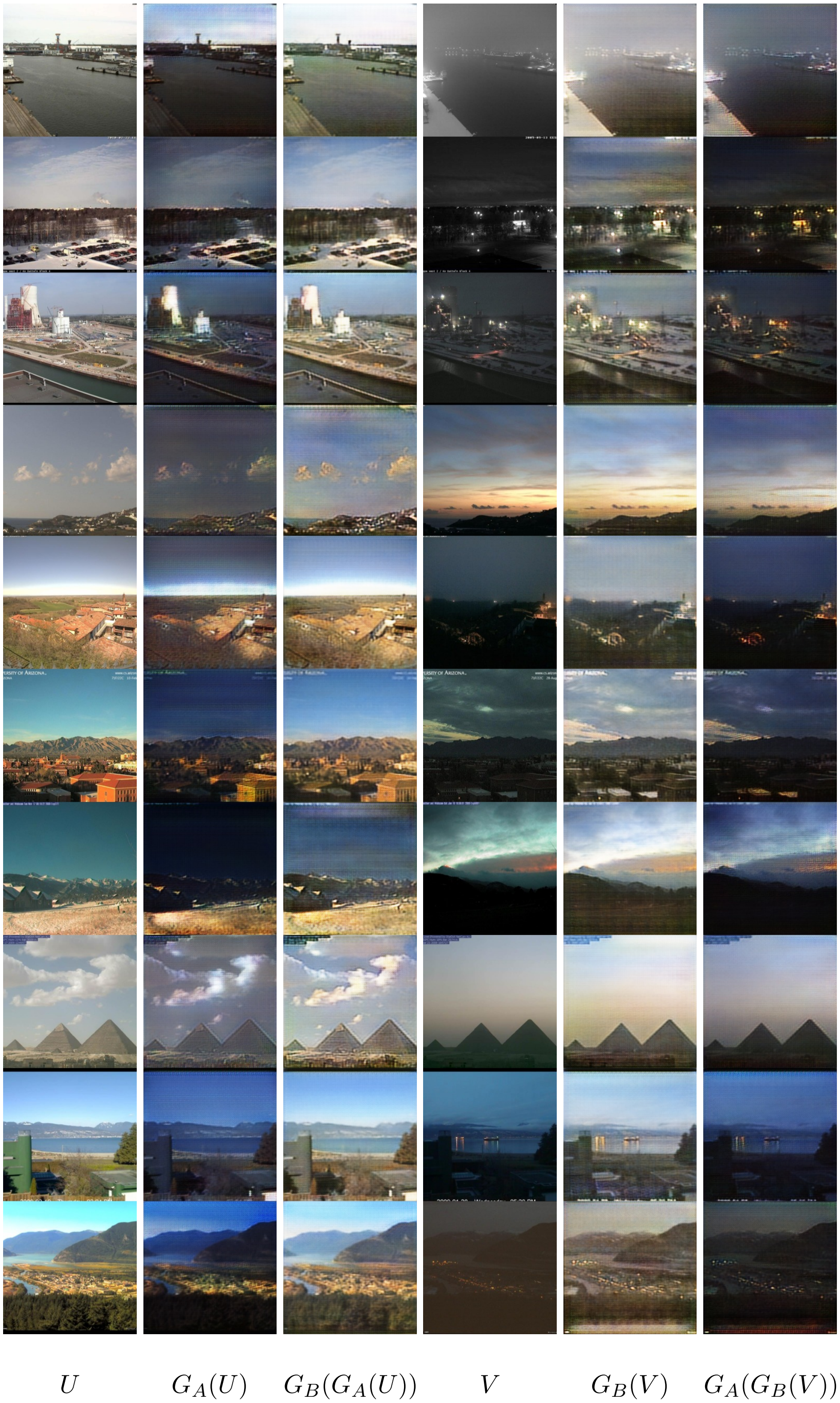


Figure 3.11: day scenes→night scenes translation results by DualGAN

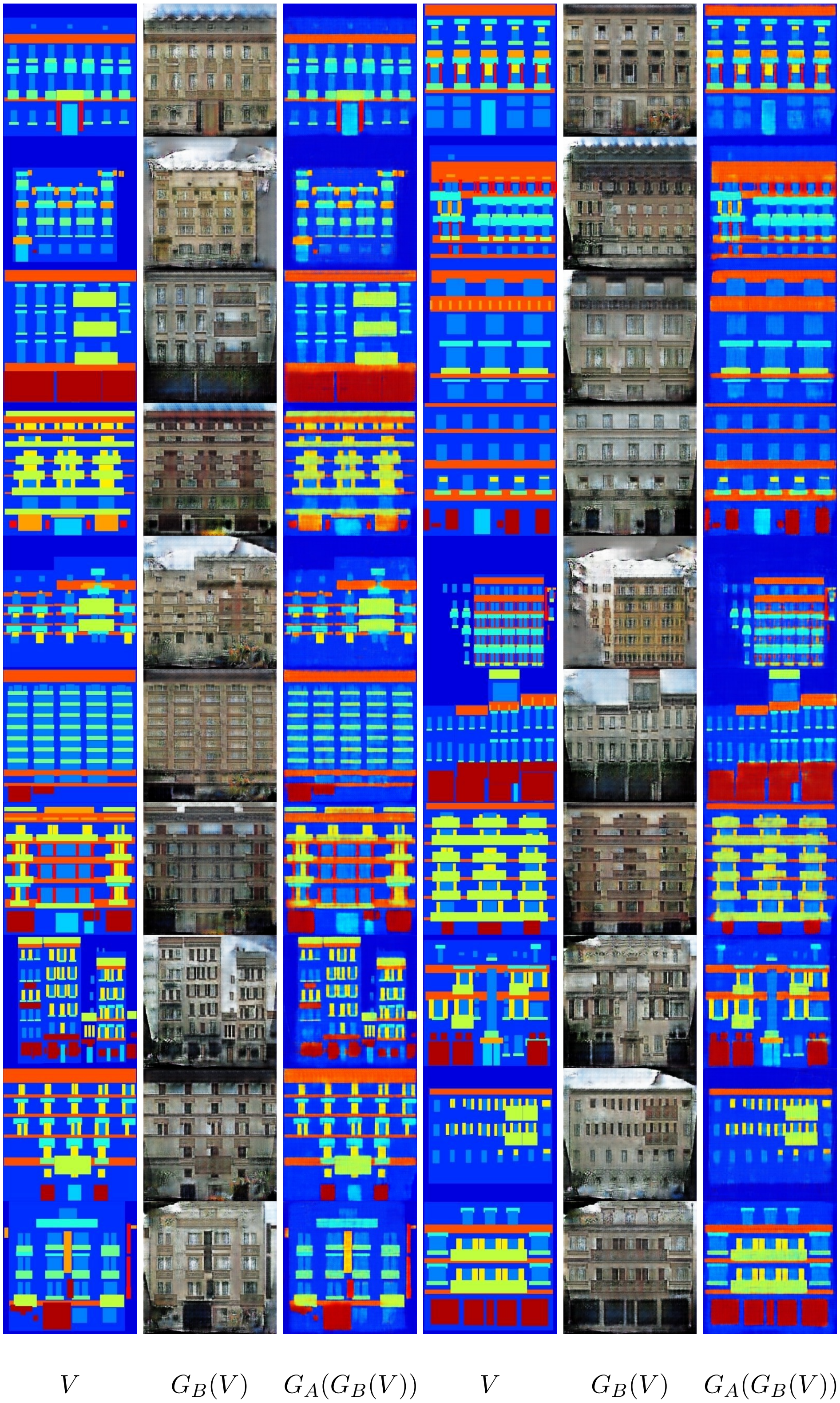


Figure 3.12: label map→photo translation results by DualGAN

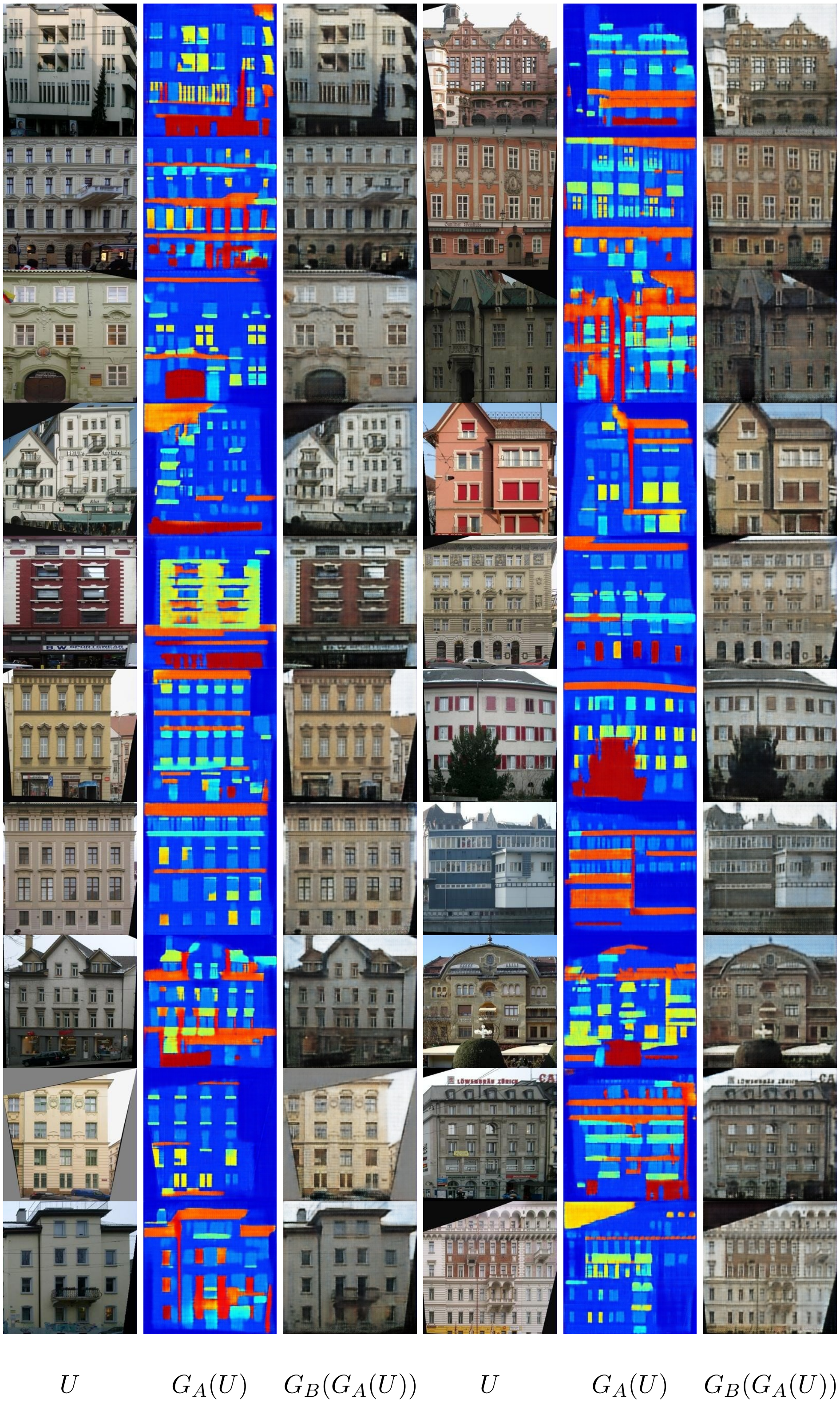


Figure 3.13: photo→label map translation results by DualGAN

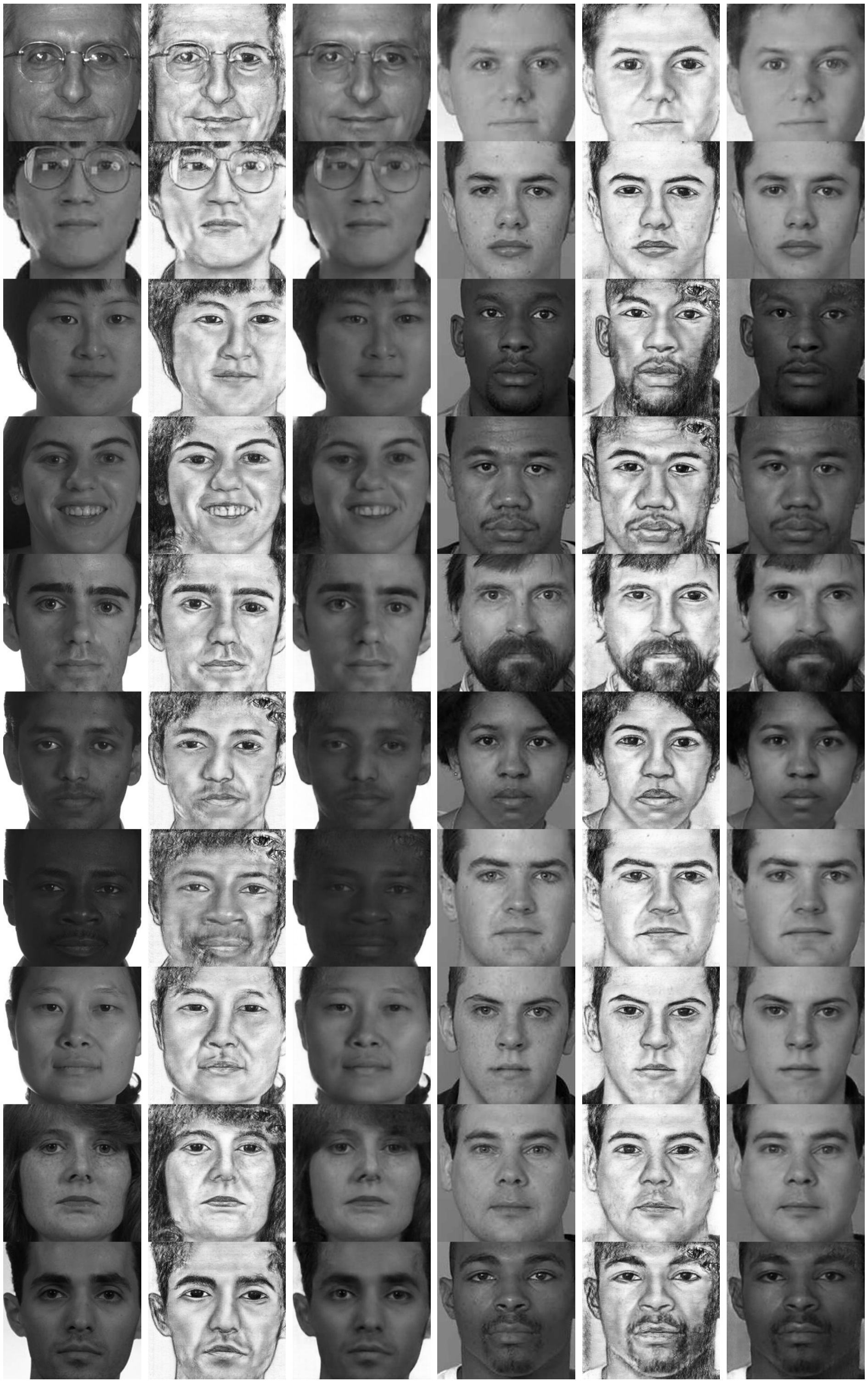


Figure 3.14: Photo→sketch translation results by DualGAN

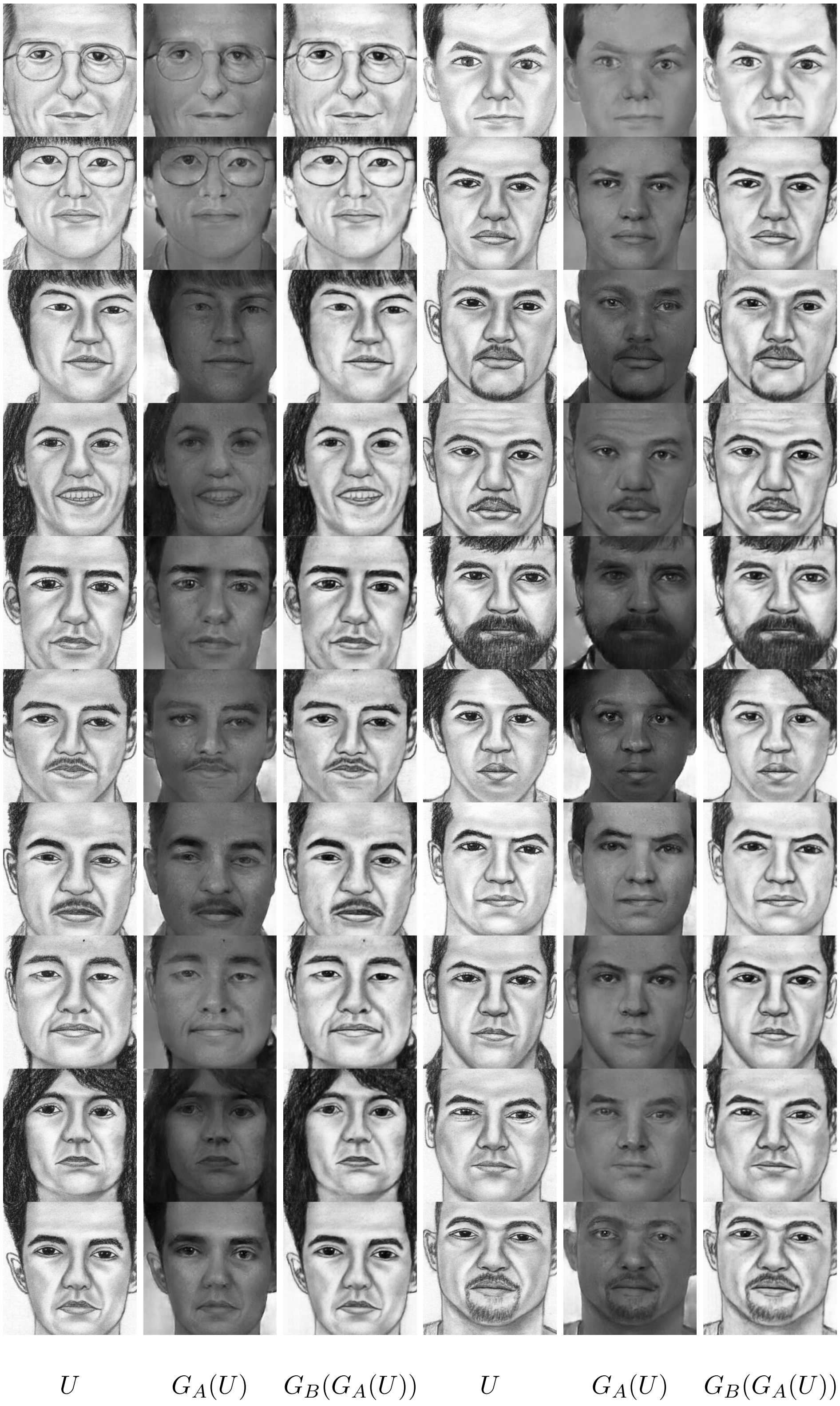


Figure 3.15: sketch→photo translation results by DualGAN

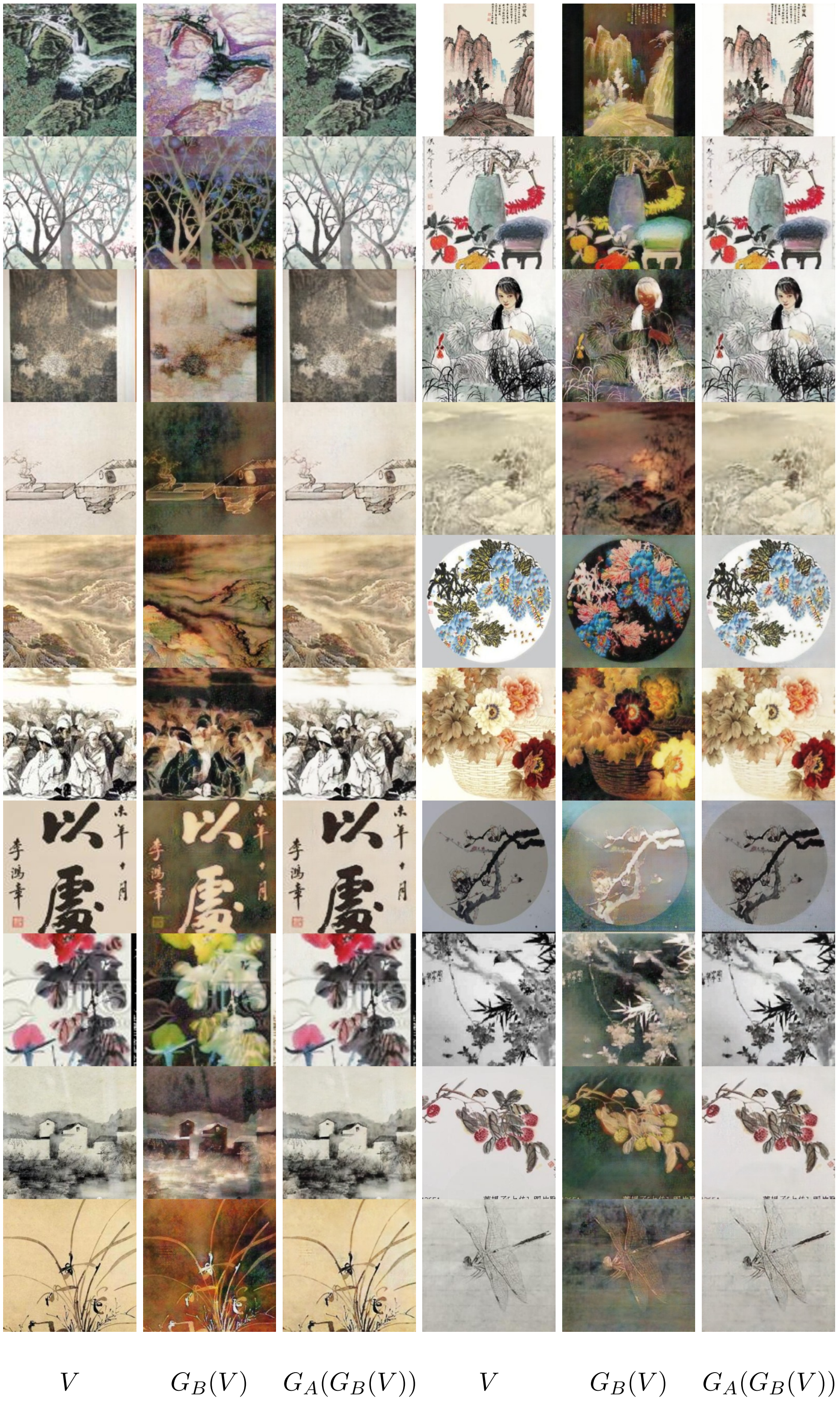


Figure 3.16: Chinese paintings→oil paintings translation results by DualGAN

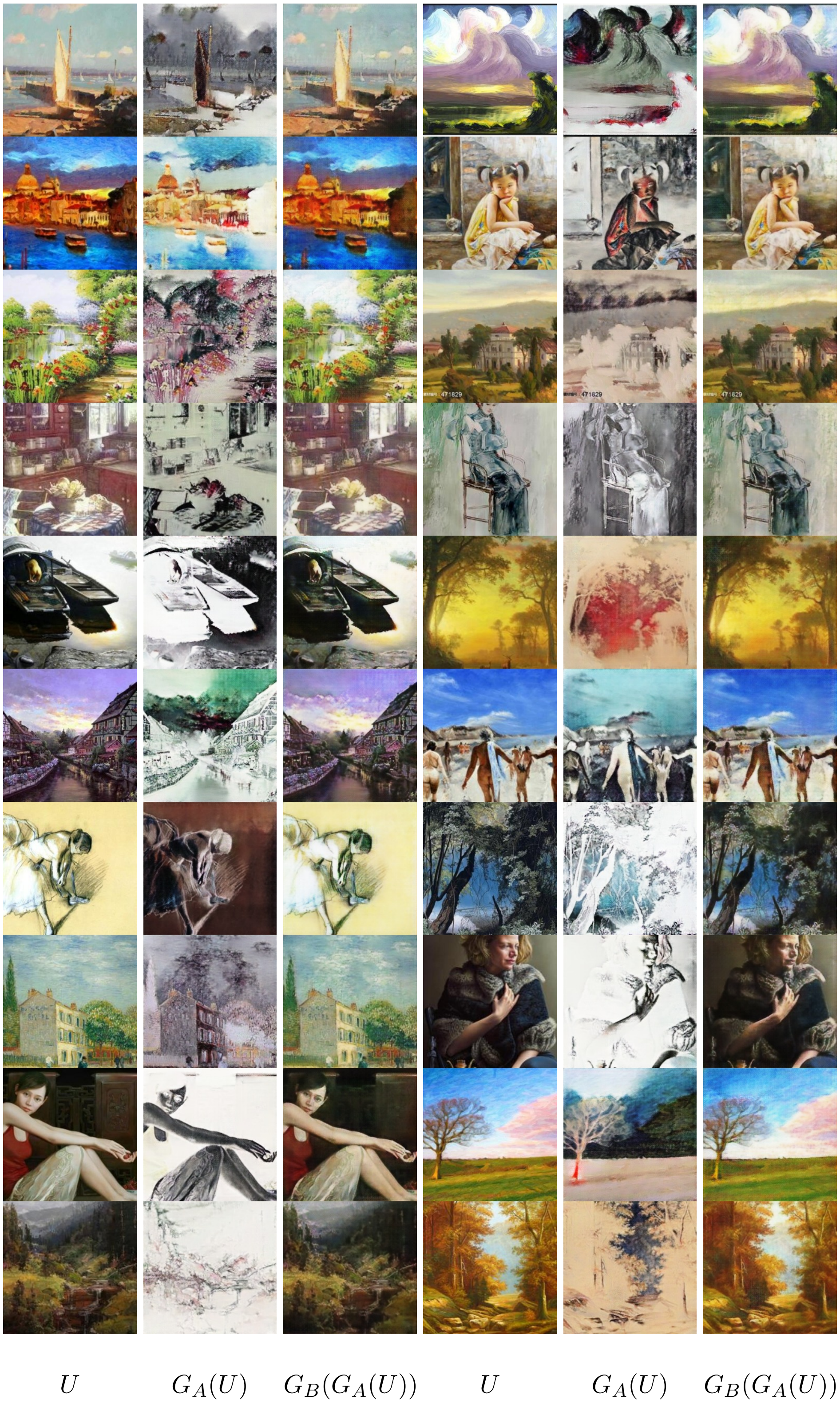


Figure 3.17: Oil painting→Chinese painting translation results by DualGAN

bias, for each of the four evaluations, we randomly shuffle real photos and outputs from all three approaches before showing them to Turkers. Each image is shown to 20 Turkers, who were asked to score the image based on to what extent the synthesized photo looks real. The “realness” score ranges from 0 (totally missing), 1 (bad), 2 (acceptable), 3 (good), to 4 (compelling). The average score of different approaches on various tasks are then computed and shown in Table 3.2. The AMT study results show that DualGAN outperforms GAN on all tasks and outperforms cGAN on two tasks as well. This indicates that cGAN has little tolerance to misalignment and inconsistency between image pairs, but the additional pixel-level correspondence does help cGAN correctly map labels to colors and textures.

Finally, we compute the segmentation accuracies for facades→label and aerial→map tasks, as reported in Tables 3.3 and 3.4. The comparison shows that DualGAN is outperformed by cGAN, which is expected as it is difficult to infer proper labeling without image correspondence information from the training data.

3.5 Summary

We propose DualGAN, a novel unsupervised dual learning framework for general-purpose image-to-image translation. The unsupervised characteristic of DualGAN enables many real world applications, as demonstrated in this work, as well as in the concurrent work CycleGAN [125]. Experimental results suggest that the DualGAN mechanism can significantly improve the outputs of GAN for various image-to-image translation tasks. With unlabeled data only, DualGAN can generate comparable or even better outputs than conditional GAN [39] which is trained with labeled data providing image and pixel-level corre-

Task	DualGAN	GAN
plastic→wood	**2**/11	0/11
wood→plastic	**1**/11	0/11
metal→stone	**2**/11	0/11
stone→metal	**2**/11	0/11
leather→fabric	**3**/11	2/11
fabric→leather	**2**/11	1/11
plastic→metal	**7**/11	3/11
metal→plastic	**1**/11	0/11

Table 3.1: Success rates of various material transfer tasks based on the AMT “material perceptual” test. There are 11 images in each set of transfer result, with noticeable improvements of DualGAN over GAN.

	Avg. “realness” score			
Task	DualGAN	cGAN[39]	GAN	GT
sketch→photo	**1.87**	1.69	1.04	3.56
day→night	**2.42**	1.89	0.13	3.05
label→facades	1.89	**2.59**	1.43	3.33
map→aerial	2.52	**2.92**	1.88	3.21

Table 3.2: Average AMT “realness” scores of outputs from various tasks. The results show that DualGAN outperforms GAN in all tasks. It also outperforms cGAN for sketch→photo and day→night tasks, but still lag behind for label→facade and map→aerial tasks. In the latter two tasks, the additional image correspondence in training data would help cGAN map labels to the proper colors/textures.

	Per-pixel acc.	Per-class acc.	Class IOU
DualGAN	0.27	0.13	0.06
cGAN [39]	**0.54**	**0.33**	**0.19**
GAN	0.22	0.10	0.05

Table 3.3: Segmentation accuracy for the facades→label task. DualGAN outperforms GAN, but is not as accurate as cGAN. Without image correspondence (for cGAN), even if DualGAN segments a region properly, it may not assign the region with a correct label.

	Per-pixel acc.	Per-class acc.	Class IOU
DualGAN	0.42	0.22	0.09
cGAN [39]	**0.70**	**0.46**	**0.26**
GAN	0.41	0.23	0.09

Table 3.4: Segmentation accuracy for the aerial→map task, for which DualGAN performs less than satisfactorily.

spondences. Source codes of DualGAN have been released on duxingren14/DualGAN on github.

On the other hand, our method is outperformed by conditional GAN or cGAN [39] for certain tasks which involve semantics-based labels. This is due to the lack of pixel and label correspondence information, which cannot be inferred from the target distribution alone. In the future, we intend to investigate whether this limitation can be lifted with the use of a small number of labeled data as a warm start.

Chapter 4

Learning Multi-Scale Image Manifold for Scale-Aware Image Fusion

This Chapter presents another learning-based approach: BranchGAN. It is a novel training method that enables unconditioned generative adversarial networks (GANs) to learn image manifolds at multiple scales. What is unique about BranchGAN is that it is trained in *multiple branches*, progressively covering both the breadth and depth of the network, as resolutions of the training images increase to reveal finer-scale features. Specifically, each noise vector, as input to the generator network, is *explicitly* split into several sub-vectors, each corresponding to and trained to learn image representations at a particular scale. During training, we progressively "de-freeze" the sub-vectors, one at a time, as a new set of higher-resolution images is employed for training and more network layers are added. A consequence of such an explicit sub-vector designation is that we can *directly* manipulate and even combine latent (sub-vector) codes that are associated with specific feature scales.

Experiments demonstrate the effectiveness of our training method in multi-scale, *disentangled* learning of image manifolds and synthesis, without any extra labels and without compromising quality of the synthesized high-resolution images. We further demonstrate two new applications enabled by BranchGAN.

4.1 Overview

In recent years, unconditioned generative adversarial networks (GANs) [26] have been intensively studied as a means for unsupervised representation learning and data synthesis. Compared to their conditional counterparts [71, 117, 74, 80, 120, 40, 127], unconditioned GANs place less burden on the training data but are less steerable at the same time. In unconditional GAN, a well-trained generator could synthesize novel data by sampling a random noise vector from the learned manifold as input and altering values "parameterizing" the dimensions of the manifold. However, this synthesis process is typically uncontrollable and counterintuitive, since we have little understanding how each manifold dimension impacts the synthesized output.

For manifold learning of images or other visual forms, the notion of *feature scales* is of paramount importance. The ability to learn multi-scale or scale-invariant features often leads to a deeper and richer understanding of representations and distributions of images. In the last few years, *scale-aware unconditioned GANs* have been developed, e.g., StackGAN [120], Laplacian Pyramid GAN [18] and progressively growing GAN [45], where correlated GANs are trained in a coarse-to-fine manner, using lower- and then higher-resolution images, with the goal of improving the quality of the final full-resolution images. However, factors which impact image features at various scales remain entangled in the networks. In a few most recent works, under the setting of conditional GANs, attempts have been made to disentangle the latent codes which correspond to different image attributes [19, 116, 117].

In this chapter, we introduce a novel training method that enables unconditioned GANs to learn image manifolds at multiple scales. What is unique about our learning paradigm is that each noise vector, as input to the generator, is *explicitly* split into a prescribed number of sub-vectors, e.g., 5 for learning 256×256 images and 6 for 512×512 images, where each sub-vector corresponds to and is trained to learn image representations at a particular scale. A direct consequence of such a sub-vector designation is that we can *directly* manipulate and even combine latent (sub-vector) codes that are associated with specific feature scales, leading to novel applications of unconditional adversarial learning that were not possible before. Figure 4.1 shows an example of *cross-scale image fusion*, where we intentionally synthesize an image by integrating the coarse-scale features of one image with finer-scale features of another. At the high level, our learning method employs the standard GAN framework which comes with an unconditioned generator and a discriminator, following the standard GAN training paradigm as described in [26, 79]. To achieve multi-scale learning, our network is trained *progressively*, bearing some similarity to Karras et al. [45]. However, instead of progressing only on the number of network layers and the resolutions of the training images, as in [45], our network training also progresses over the sub-vectors corresponding to increasingly finer scales of the images. Specifically, as shown in Figure 4.2, we start by only training for the sub-vector corresponding to the coarsest level features, i.e., using the lowest-resolution images, while keeping the other sub-vectors “frozen”. Then we progressively “de-freeze” the sub-vectors, one at a time, as a set of higher-resolution images is employed for training and more network layers are added.

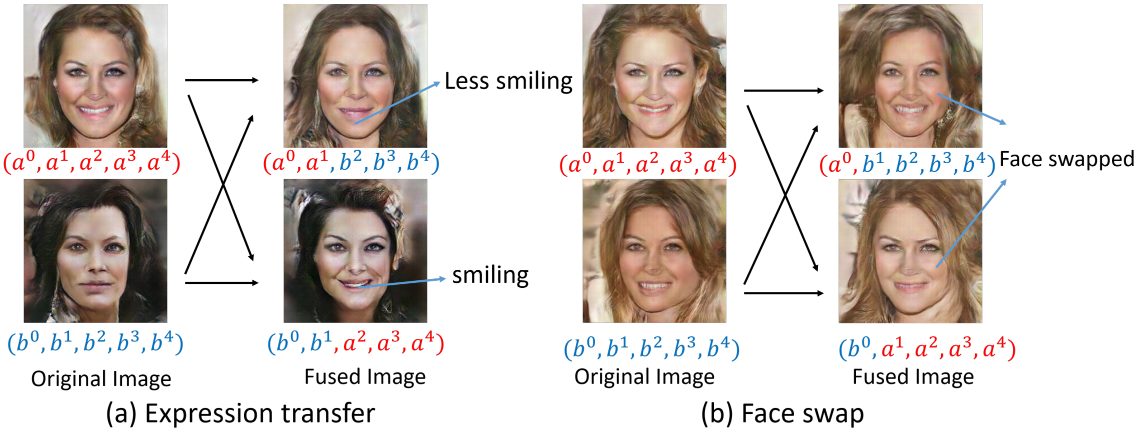

$(a^0, a^1, a^2, a^3, a^4)$
$(b^0, b^1, b^2, b^3, b^4)$
Original Image
$(a^0, a^1, b^2, b^3, b^4)$
Less smiling
$(b^0, b^1, a^2, a^3, a^4)$
smiling
Fused Image
(a) Expression transfer
$(a^0, a^1, a^2, a^3, a^4)$
$(b^0, b^1, b^2, b^3, b^4)$
Original Image
$(a^0, b^1, b^2, b^3, b^4)$
Face swapped
$(b^0, a^1, a^2, a^3, a^4)$
Fused Image
(b) Face swap

Figure 4.1: *Cross-scale image fusion* by directly combining coarse-scale features in one image with finer-scale features from another. Please note that $\mathbf{x}^0(\mathbf{x} \in \{\mathbf{a}, \mathbf{b}\})$ encodes image-wide structures and $\mathbf{x}^t(t \in \{1, 2, 3, 4\})$ encodes increasingly fine-scale features. Given a pair of images, we compose new images by cross-combining coarse-scale structures and fine-scale features of the two, accomplishing expression transfer (a) and face swap (b).

When training for a particular scale, the network weights learned from previous trainings, for coarser scales, are used to initialize the training. Note that after training, these weights are often changed to adapt to the new training data. In contrast to previous works on scale-aware unconditioned GANs that train multiple GANs [120, 18], our method trains only one GAN. Unlike [45], our progressive training is not only over the network depth (adding layers as image resolutions increase), but also over the dimensions of the image manifold, explicitly designating dimensions to image scales. Another way to view the progression is that it is over the "breadth" (and depth) of the network, leading to different scale-specific "*training branches*". Hence, we refer to our network as a *BranchGAN*.

What had motivated our key idea of branched GAN training and what made it work effectively is a phenomenon we observed when experimenting with multi-branch data generators, which we coined "*branch suppression*". Roughly speaking, we found that when multiple noise vectors, with their respective training branches, are at play, GAN training typically results in one dominant branch while the other branches are either fully or partially *suppressed*; see more details in Section 4.3.2. In other words, the already-trained weights (branches) will have priority in maintaining their role in encoding the image structures that are already encoded and suppress the other branches. When de-freezing one sub-vector during progressive training, "*branch suppression*" helps inhibit the ability of the newly de-frozen branch in the network to encode coarser-scale structures, thus "encouraging" it to encode the finer-scale structures in the new set of higher-resolution training images. Note that the inhibition or suppression is not absolute; the network weights in previously trained branches are still altered.

For the first time, BranchGAN allows direct manipulation of scale-specific manifold dimensions by disentangling multi-scale image representations, without introducing extra

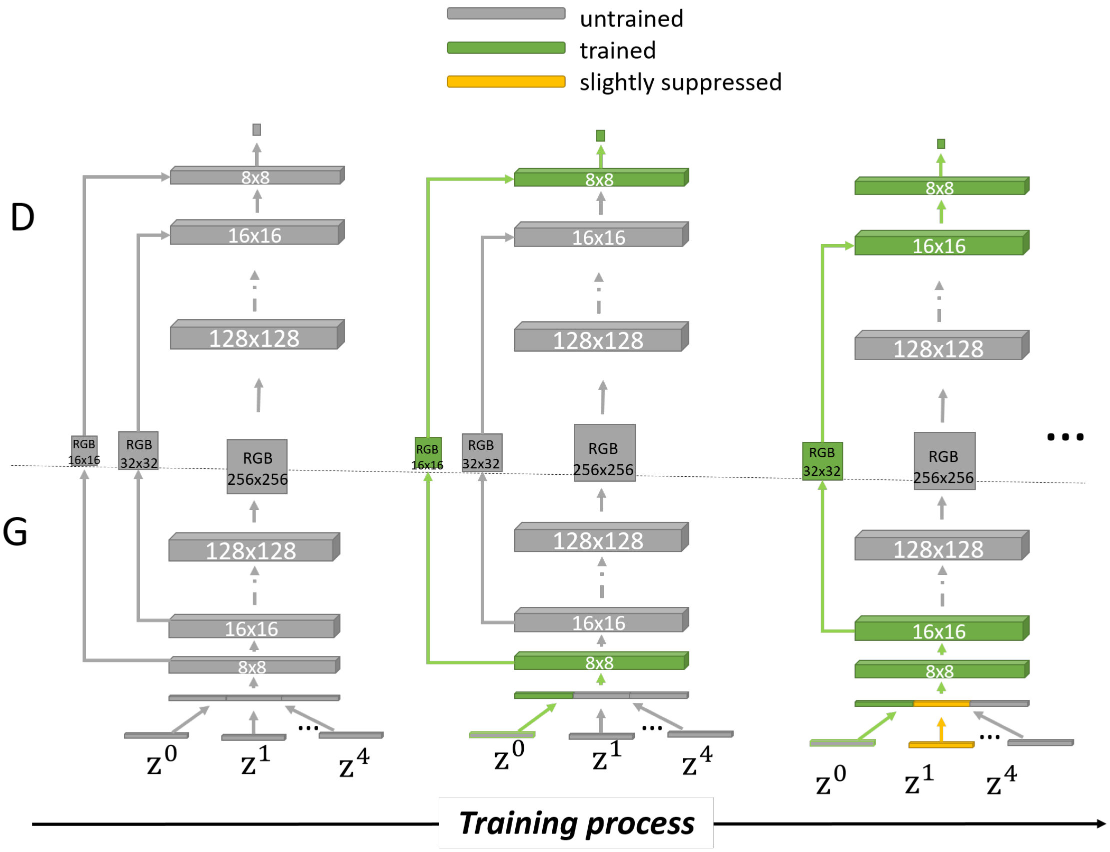


Figure 4.2: Training pipeline of BranchGAN. We start with both the generator (G) and discriminator (D) having a low spatial resolution. During first training period, we feed $\mathbf{z}^0$ with random vectors of uniform distribution and $\mathbf{z}^t$ ($t > 0$) with zero vector $\mathbf{0}$, thus making the linear-layer weights corresponding to $\mathbf{z}^t$ ($t > 0$) untrainable. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. Meanwhile, we "de-freeze" more $\mathbf{z}$ vectors for training by feeding them with non-zero uniform-random vectors. This process is repeated until the target resolution is reached.

labels. We test our novel training method on several high-quality image datasets and verify its effectiveness in learning multi-scale image representations. We also show that Branch-GAN adds capabilities to GANs in coarse-to-fine image synthesis and scale-aware image fusion. As a side contribution, we release one high-resolution (800×600) image dataset to boost research on high-quality image synthesis and manifold learning. Finally, we believe that our observation on "*branch suppression*" may offer insight in other contexts of training multi-branch convolutional and/or generative networks.

4.2 Related work

4.2.1 Multi-scale image representations

An inherent property of visual objects is that they only exist as meaningful entities over certain ranges of scale in an image. How to describe image structures at multiple scales remains an essential and challenging problem in image analysis, image compression, image processing and image synthesis. Early methods for multi-scale image representing such as Discrete Fourier Transformation (DFT) [3] and Discrete Wavelet Transformation (DWT) [89] are widely used in disentangling small-scale details and large-scale structures. Additionally, image quality metrics such as MS-SSIM [109] is widely used to evaluate image structures by scales. In this chapter, DFT is employed to disentangle image structures by frequencies (or scales).

Another scale-independent representation of images is the layer activations of a well-trained Convolutional Neural Network (CNN) [53, 49, 94]. In CNN, top activation layers roughly represent large-scale image structures such as objects and scenes, and bottom activations represent small-scale details such as edges, colors or textures. Other than CNN, stacked models such as Deep Belief Network (DBN) [34], Stacked AutoEncoders (SAE) [105, 104] or multi-scale sparse Coding [86] can also be utilized to retrieve multi-scale representations of images, though the effectiveness could be limited. In this chapter, we employ a pre-trained $VGG19$ network [94] to extract image features at multiple scales.

4.2.2 Coarse-to-fine image synthesis

Scale-aware image synthesis has been explored in StackGAN [120], LAPGAN [18], and Progressive GAN [45]. These methods attempt to synthesize higher-quality images, rather than to learn multi-scale image manifolds. LAPGAN [18] attempts to add noise separately/progressively to increase the variation of outputs, but the noise are added through dropout layers (except for the first z vector); this is neither controllable nor explicit. Our task is to explicitly learn scale-disentangled image representations, which differs from the goal of these methods. Whereas, we extend the idea of progressive growing [45] from progressively adding layers to progressively growing both layers and branches.

4.2.3 User-controllability in image synthesis

One stream to make image synthesis more controllable is conditional GANs or semi-conditional GANs, which condition image synthesis on attributes [117, 19, 116], classes [74], texts [80, 120], or images [40, 127]. These methods either require extra labels or paired images, or need strict inherent relations between priors and outputs. Our method, as a type of unconditional GAN, conditions image generation on random noise of uniform distribution and does not require any extra labels or priors.

For unconditional GANs, the method known as iGAN [126] provides a way for users to synthesize or manipulate realistic images in a more controllable way. In iGAN, users could add certain constraints on the appearance of desired images (e.g., draw edges, add color strokes or set up an exemplar image) and a manifold point is then optimized to satisfy these constraints. Nonetheless, a sophisticated optimization method is required, as gradient descent is particularly vulnerable to local minima. To resolve this issue, an extra network that predicts manifold points from images is needed. Our method attempts to raise user-controllability as iGAN [126] does, though we intend to improve the manifold itself rather build a system on top of it. It is therefore possible to combine our method with iGAN. Chen et al. proposed infoGAN that attempts to learn interpretable and disentangled latent space with unlabeled data [12], though the latent space are not disentangled by scales.

4.2.4 Surround suppression

"*Branch suppression*" refers to a phenomenon observed in the training of unconditional GANs when the generator propagates confidence from input to output through multiple branches. We found that, under certain circumstance, one branch evolves to dominate the output of the generator while other branches become suppressed. This observation reminds us of another phenomenon known as "surround suppression" [15], which refers to observations that the relative firing rate of a neuron may under certain conditions decrease when a particular stimulus is enlarged. The distinctness between the two is that "*branch suppression*" is observed throughout the training of GANs, whereas surround suppression is typically observed in biological neuron systems such as human brain and sensory neurons. Limited by our knowledge on the subject matter, it remains unclear whether there are any meaningful links between the two forms of suppresions. Nevertheless, it is a curious question whether "*branch suppression*" exists in other "multi-branch" neural networks, such as ResNet [31], DenseNet [37], and capsule network [33].

4.3 Our method

4.3.1 Entangling of scales in traditional GAN

To examine how each dimension of the latent manifold space of unconditional GANs impacts the output image, we did an investigation as shown in Figure 4.3. The metrics used to evaluate the impact significance (IS) of each dimension on the output image are IS_DFT and IS_VGG, which measure the variance of the output image when a given manifold dimension is changed but other dimensions are fixed. In specific, IS_DFT measures image variance within image frequency domain, whereas IS_VGG measures image variance within VGG feature domain (the layers activations of $VGG19$ network). Since both DFT and VGG network can extract multi-scale features, the two metrics can therefore evaluate scale-wise impact significance. Note that layer $poolx, x \in \{1, 2, ..., 5\}$ activations correspond to increasingly large scale features, and higher frequency corresponds to smaller spatial scale.

$$\begin{aligned} IS\_DFT_{z_x}(f1, f2) &= \sum_{h,w,d} \mathop{\mathbb{E}}_{\mathbf{c} \sim U(-1,1)} \sigma_{z_x \sim U(-1,1), \overline{\boldsymbol{z}_x} \leftarrow \boldsymbol{c}} \big(DFT_{f1}^{f2}(G(\boldsymbol{z})) \big) \\ IS\_DFT_{z_x}(f1, f2) &= IS\_DFT_{z_x}(f1, f2) / \mathop{\mathbb{E}}_{z_x \in \mathbf{z}} IS\_DFT_{z_x}(f1, f2) \end{aligned} \tag{4.1}$$

and

$$\begin{aligned} IS\_VGG_{z_x}(L^{VGG}) &= \sum_{h,w,d} \mathop{\mathbb{E}}_{\mathbf{c} \sim U(-1,1)} \sigma_{z_x \sim U(-1,1), \overline{\mathbf{z}_x} \leftarrow \boldsymbol{c}} \big(L^{VGG}(G(\boldsymbol{z})) \big) \\ IS\_VGG_{z_x}(L^{VGG}) &= IS\_VGG_{z_x}(L^{VGG}) / \mathop{\mathbb{E}}_{z_x \in \mathbf{z}} IS\_VGG_{z_x}(L^{VGG}) \end{aligned} \tag{4.2}$$

where $\overline{\mathbf{z}_x}$ is the dimension set of $\mathbf{z}$ excluding z_x. $\sigma_{z_x \sim U(-1.0,1.0), \overline{\mathbf{z}_x} \leftarrow \mathbf{c}} f(\mathbf{z})$ refers to the deviation of the value of $f(\mathbf{z})$ when z_x follows the uniform distribution $U(-1.0, 1.0)$ and $\overline{\mathbf{z}_x}$ is fixed as a constant vector $\mathbf{c}$. $G(\mathbf{z})$ is the output image of Generator G given $\mathbf{z}$. h, w, d are the height, width and depth of images (or layer activations). $\mathbb{E}(\cdot)$ is the expectation operator. In Eq. 4.1, $DFT_{f1}^{f2}(\cdot)$ refers to the DFT of an image, and $(f1, f2)$ is a frequency range. In Eq. 4.2, $L^{VGG}(\cdot)$ is the activation of L^{VGG} layer of a pre-trained $VGG19$ network [94] in terms of an input image (resized to 224×224), where the candidate $L^{VGG} \in \{pool1, pool2, pool3, pool4, pool5\}$. In order to avoid the impact of image size, IS_DFT (or IS_VGG) is further normalized (divided by their expectation values). Greater value of IS_DFT (or IS_VGG) means greater impact of a manifold dimension on the output.

To assess consistency of the two metrics with human perception, we conducted a user study. We selected 48 pairs of images (18 for each dataset) with different level of variation. We hired 20 Turks to rate each pair in terms of level of variation. In the test, three options with elaborate explanations were shown to the Turkers: (a) large-scale variation; (b) median-scale variation; or (c) small-scale variation. The label with the most votes is treated as ground-truth. Then we compare the human-labeled results with those estimated by IS_DFT and IS_VGG (the scale level with highest score is used) and compute the percentages of agreements as shown in Table 4.1. The agreement rates of these metrics are quite high from this preliminary study (significantly better than random). Though not perfect enough, the proposed metrics are the only metrics that we could think out to measure image variation by scale.

Further, a traditional GAN is trained for evaluation. Without loss of generality, we use the training loss and network architecture of dcgan [79] and train it with progressive

IS_DFT & Human	IS_VGG & Human	IS_DFT & IS_VGG
41/48	36/48	39/48

Table 4.1: Agreement rate of different metrics.

growing methodology [45] on the $celeba\_hq\_256 \times 256$ dataset [45] (down-sampled from the original 1024×1024 resolution). The generator takes a 100-dimensional $\mathbf{z}$ vector as input.

To get overview of how each of the 100 manifold dimensions impacts the output image, we compute the IS_DFT and IS_VGG of each dimension, summarize the number of dimensions by value of impact significance and generate a few histograms: see Figure 4.4. From Figure 4.3 and 4.4, we have three observations: First, the all-scale contributions of different manifold dimensions to the output differ but do not differ much; Second, the scale-wise contribution of a chosen manifold dimension does not vary much, which means the manifold is far from being scale-disentangled; Third, the way in which each dimension affects the output is neither attribute-specific nor scale-specific. BranchGAN is therefore proposed to address these issues.

4.3.2 Branch suppression

We observed "branch suppression" in all kinds of multi-branch generators as shown in Figure 4.5, among which some are fully suppressed, some are partially suppressed. In "branch suppression", the already-trained weights (branches) will have priority in maintaining their role in encoding the image structures that are already well encoded and suppress the other branches. To explain it in more details, we present a few examples of branch suppression

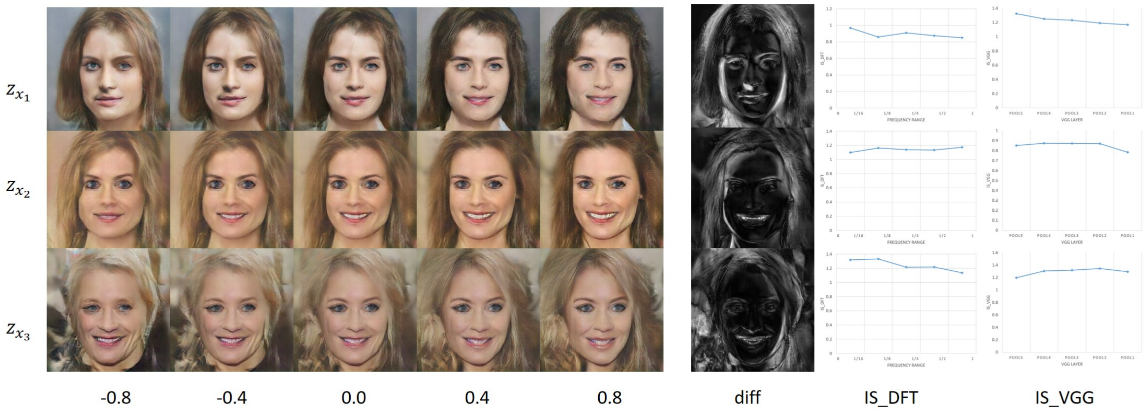


Figure 4.3: Column 1-5: $G(\mathbf{z})$, given $z_{x_t} \in \{-0.8, -0.4, 0, 0.4, 0.8\}$ and $\overline{\mathbf{z}_{x_t}} \leftarrow \mathbf{c}$ ($\mathbf{c}$ is constant and $t \in \{1, 2, 3\}$ is the Row number). Column 6: difference image $\sigma(G(\mathbf{z}))$. Column 7: $IS\_DFT_{\mathbf{z}_{x_t}}$ by frequencies. Column 8: $IS\_VGG_{\mathbf{z}_{x_t}}$ by layers. Observing from Column 7 and 8, we find that the value of impact significance of a specific dimension vary no more than 0.15 by scales. (Please magnify the **electronic** page to get the figures clearer.)

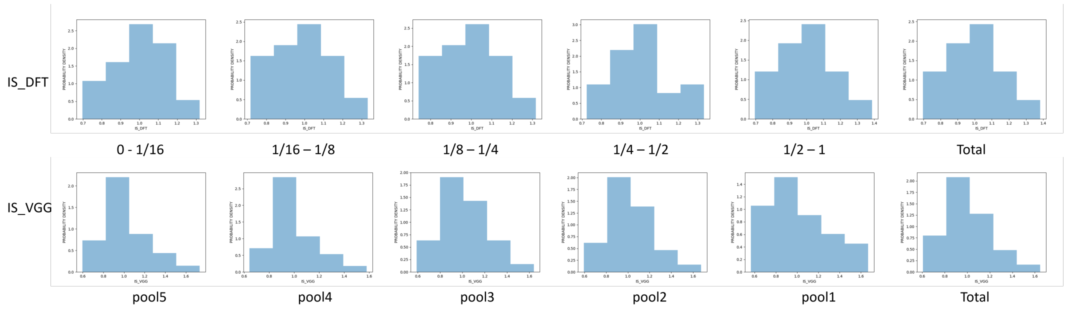


Figure 4.4: The statistic of dimension numbers by value of IS_DFT (upper) and IS_VGG (lower). As shown, most dimensions have IS_DFT in $(0.7, 1.3)$ and IS_VGG in $(0.6, 1.6)$. (Please magnify the **electronic** page to get the figures clearer.)

in Figure 4.5.

4.3.3 BranchGAN

The architecture of the generator is the same as shown in the bottom row of Figure 4.5 except that we increase the number of $\mathbf{z}$ sub-vectors (each sub-vector is 30-dimensional). For 256×256 image generation, we use 5 $\mathbf{z}$ sub-vectors. The number of $\mathbf{z}$ sub-vectors is subject to change according to the resolution of output images. We use generator and discriminator networks that are mirror images of each other and always grow in synchrony, and use the sigmoid-cross-entropy loss as dcgan [79] for training (please find more details in 4.3.4).

The overall training procedure of BranchGAN has been clearly and elaborately presented in Section 4.1. In brief, BranchGAN progressively adds the depth (or layers) and broadth (or $\mathbf{z}$ sub-vector) to the generator, and then start training with images of higher-resolution. During the process, "branch suppression" helps encourage the newly-added sub-vector to encode the finer-scale structures: see Figure 4.2. At each scale, a two-staged sub-procedure is used to avoid sudden shock to already well-trained, smaller-resolution layers: see Figure 4.6. Note that all already-added layers of the discriminator are trainable throughout the sub-procedure. Datasets will be made public.

4.3.4 Networks architecture and hyperparameters

Source codes of BranchGAN have been released on duxingren14/BranchGAN on github. The detailed information about the networks architecture of generators and discriminators are presented in Table 4.3, 4.2, 4.4, 4.5. The non-architecture hyper-parameters are listed

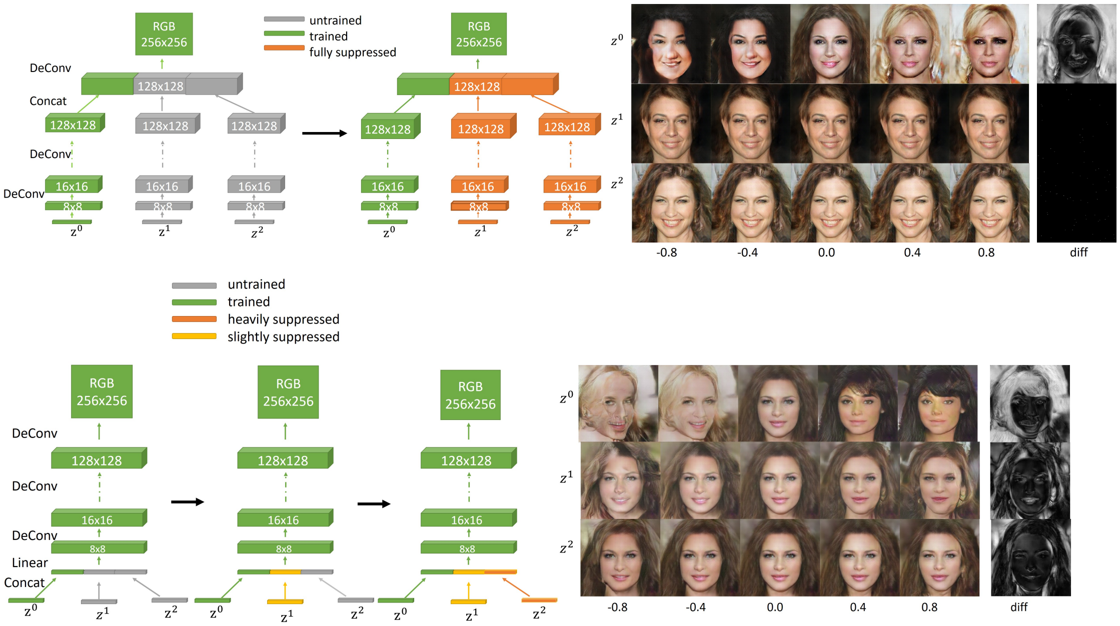


Figure 4.5: Two examples of branch suppression in GANs. In these examples, we employ the training loss and discriminator of dcgan [79]. Here we change the architecture of the generator a bit by conditioning image generation on split **z** vectors ($\mathbf{z}^t, t \in \{1, 2, 3\}$). In the upper row, the left branch is already well trained for image generation, and the middle and right branches are initialized randomly (see more details about the initialization in the 4.3.4). Then we train the GAN by following the standard GAN training procedure as in [79]. After the training converges, the left branch dominates the output while the other are fully suppressed, as seen from the difference image on the right. In the lower row, the generator architecture is the same as traditional GAN except that the z vector is split. We train the left branch till converging, then de-freeze the middle branch for training till converging, and finally the right branch. Note that the number of training steps for each stage are equal and the pre-trained weights (or branches) are not frozen even after new branches are de-frozen. As a result, the middle branch is slightly suppressed and the right branch is severely suppressed as seen from the rightmost difference images. (Please magnify the **electronic** page to get the figures clearer.)

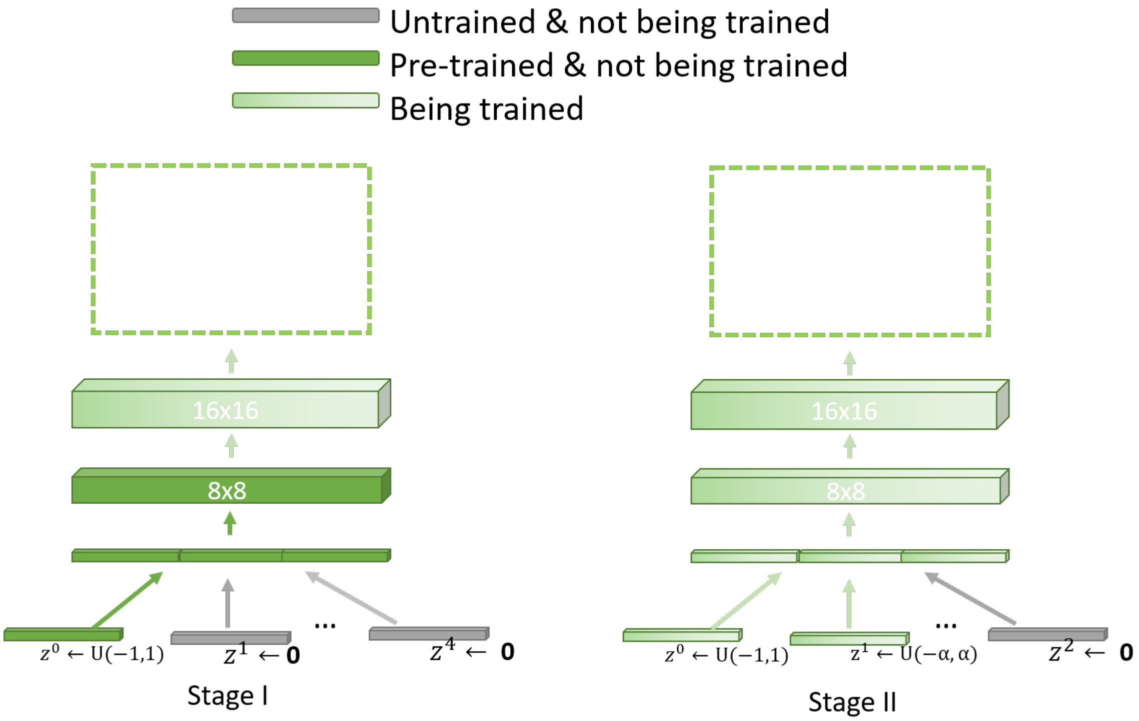

Untrained & not being trained
Pre-trained & not being trained
Being trained
16x16
8x8
...
$z^0 \leftarrow$ U(−1,1)
$z^1 \leftarrow \mathbf{0}$
$z^4 \leftarrow \mathbf{0}$
Stage I
16x16
8x8
...
$z^0 \leftarrow$ U(−1,1)
$z^1 \leftarrow$ U(−α, α)
$z^2 \leftarrow \mathbf{0}$
Stage II

Figure 4.6: The training sub-procedure at a specific scale level. After a new layer is added to the generator, we first only train the last layer of the generator while holding other layers untrainable (Stage I). After the last layer is well-trained, we then de-freeze all pre-trained layers (the branches in green) plus the newly-added branch for training (Stage II). To avoid sudden shock to already well-trained layers, we feed the newly de-frozen **z** sub-vector with a random vector following uniform distribution $U(-\alpha, \alpha)$, where α increases smoothly from 0.0 to 1.0 throughout Stage II of the sub-procedure.

	activation size	filter size
input	[30], [30], [30], [30], [30]	NA
concat	[150]	NA
linear	[32768]	[32768,150]
reshape	[8,8,512]	NA
$deconv + instance\_BN + lrelu$	[16,16,256]	[5,5,512,256]
$deconv + instance\_BN + lrelu$	[32,32,128]	[5,5,256,128]
$deconv + instance\_BN + lrelu$	[64,64,64]	[5,5,128,64]
$deconv + instance\_BN + lrelu$	[128,128,64]	[5,5,64,64]
deconv+sigmoid (output)	[256,256,3]	[5,5,64,3]

Table 4.2: Network architecture of the generator for 256×256 image synthesis.

in Table 4.6. Please note that lrelu is leaky relu layer.

4.3.5 Initialization, “freeze” and “defreeze”

For the untrained linear or deconv/conv/linear layers, the filter weights are initialized with normally random numbers $N(\mu, \sigma)$ and biases are initialized with 0. For instance normalization layer, we initialize the $scale$ with 1.0 and assign the $center$ with 0.0.

We “freeze” certain branches (or weights) by feeding the corresponding $\mathbf{z}$ vector with

	activation size	filter size
input	[256,256,3]	NA
$conv + instance\_BN + lrelu$	[128,128,64]	[5,5,3,64]
$conv + instance\_BN + lrelu$	[64,64,64]	[5,5,64,64]
$conv + instance\_BN + lrelu$	[32,32,128]	[5,5,64,128]
$conv + instance\_BN + lrelu$	[16,16,256]	[5,5,128,256]
$conv + instance\_BN + lrelu$	[8,8,512]	[5,5,256,512]
reshape	[32768]	NA
linear	[1]	[32768,1]

Table 4.3: Network architecture of the discriminator for 256×256 image synthesis.

	activation size	filter size
input	[300,400,3]	NA
$conv + instance\_BN + lrelu$	[150,200,64]	[5,5,3,64]
$conv + instance\_BN + lrelu$	[75,100,64]	[5,5,64,64]
$conv + instance\_BN + lrelu$	[38,50,64]	[5,5,64,64]
$conv + instance\_BN + lrelu$	[19,25,128]	[5,5,64,128]
$conv + instance\_BN + lrelu$	[10,13,256]	[5,5,128,256]
$conv + instance\_BN + lrelu$	[5,7,512]	[5,5,256,512]
$conv + instance\_BN + lrelu$	[17920]	NA
linear	[1]	[17920,1]

Table 4.4: Network architecture of the discriminator for 400×300 image synthesis.

	activation size	filter size
input	[30], [30], [30], [30], [30]	NA
concat	[150]	NA
linear	[17920]	[17920,150]
reshape	[5,7,512]	NA
$deconv + instance\_BN + lrelu$	[10,13,256]	[5,5,512,256]
$deconv + instance\_BN + lrelu$	[19,25,128]	[5,5,256,128]
$deconv + instance\_BN + lrelu$	[37,50,64]	[5,5,128,64]
$deconv + instance\_BN + lrelu$	[75,100,64]	[5,5,64,64]
$deconv + instance\_BN + lrelu$	[150,200,64]	[5,5,64,64]
deconv+sigmoid (output)	[300,400,3]	[5,5,64,3]

Table 4.5: Network architecture of the generator for 400×300 image synthesis.

name	value	notation
Optimizer learning rate beta1 beta2	AdamOptimizer 0.0002 0.5 0.999	constant lr
#scale	5 for $Celeb\_hq\_256 \times 256$ and $LSUNchurch\_outdoor$ 6 for $Celeb\_hq\_512 \times 512$ and $car\_400 \times 300$	
#epoch/scale	20 for $Celeb\_hq\_256 \times 256$ and $LSUNchurch\_outdoor$ 12 for $Celeb\_hq\_512 \times 512$ and $car\_400 \times 300$	
#batch/epoch	subject to dataset size and batch size we use full dataset for each epoch	
batch size	20 for $Celeb\_hq\_256 \times 256$ and $LSUNchurch\_outdoor$ 12 for $Celeb\_hq\_512 \times 512$ and $car\_400 \times 300$	limited by GPU memory

Table 4.6: Non-architecture training hyperparameters.

$\mathbf{0}$. Here we intend to explain the reason why feeding zero vector makes the corresponding weights untrainable.

Therefore, the activations of linear layer when fed with $\mathbf{0}$ are given by $f(\mathbf{0}) \equiv \theta\mathbf{0} + \mathbf{0} \equiv \mathbf{0}$, where θ are the linear weights. The activations of conv/deconv layer are given by $f(\mathbf{0}) \equiv \theta \circledast \mathbf{0} + \mathbf{0} \equiv \mathbf{0}$ (or $\theta \circledast \mathbf{0} \equiv \mathbf{0}$ after concatenation), where θ are filter weights. So we have the gradients $\nabla f_\theta(\mathbf{0}) \equiv \mathbf{0}$. For instance normalization layer and leaky relu layer, $g(\mathbf{0}) \equiv lrelu((\mathbf{0} - \mathbf{0}) \cdot \mathbf{1.0} + \mathbf{0.0}) \equiv \mathbf{0}$. We have the gradients $g_\beta(\mathbf{0}) \equiv \mathbf{0}$, where β is the scale.

In this way, the branches (or weights) could be "frozen" when fed with $\mathbf{0}$. To "defreeze" these branches (or weights), simply feed them with non-zero vectors.

4.4 Experimental Results and Evaluation

4.4.1 Evaluation

We evaluate our method on three datasets: LSUN *church_outdoor* [119], *celeba_hq* [45] and *car* (prepared by us). The original car dataset has resolution of 800×600 pixels. To speed up the training, we use the downsampled version of $celeba\_hq$ and car ($celeba\_hq\_256 \times 256$ and $car\_400 \times 300$). All qualitative and quantitative evaluation results are shown in Figures 4.7, 4.9, and 4.10. From these figures, we clearly observe that our goal of scale-disentangling is well achieved, where each sub-vector learns image representations at a particular scale. Figure 4.8 further shows the statistic results of impact significance. We find that the value of IS_VGG and IS_DFT see much greater variance than those of traditional GAN, which further proves the effectiveness of BranchGAN in r12cm

	celeba_hq		car		church_outdoor	
	FID	MRE	FID	MRE	FID	MRE
Layer+Branch growing (ours)	12.86	**0.169**	**18.97**	**0.178**	23.47	**0.193**
Layer-growing	**12.56**	0.181	19.15	0.183	**23.29**	0.204

Table 4.7: A comparison between BranchGAN and progressive GAN. scale-disentangled manifold learning.

4.4.2 Experimental results on higher resolution datasets

We experiment our method on data of higher resolution: see results in Figure 4.11.

4.4.3 Comparison

We can add a comparison to progressive GAN. As shown in Table 4.7, we compute the Frechet Inception Score (FID) and Minimum Reconstruction Error (MRE) [78] of models trained with our branch+layer growing mechanism vs. those trained with layer growing only. We apply the same experimental setting for both methods, e.g., # manifold dimensions, # training steps, architectures of the discriminators and generators, and datasets. The results show that their FIDs are comparable, whereas BranchGAN beats progressive GAN in MRE, implying that the two compared GANs lead to similar image quality while BranchGAN increases variation of the outputs. We would expect that by integrating with techniques such as pixel normalization, WGAN-GP, minibatch standard deviation, and adequate training steps for full convergence, BranchGAN can achieve better results.

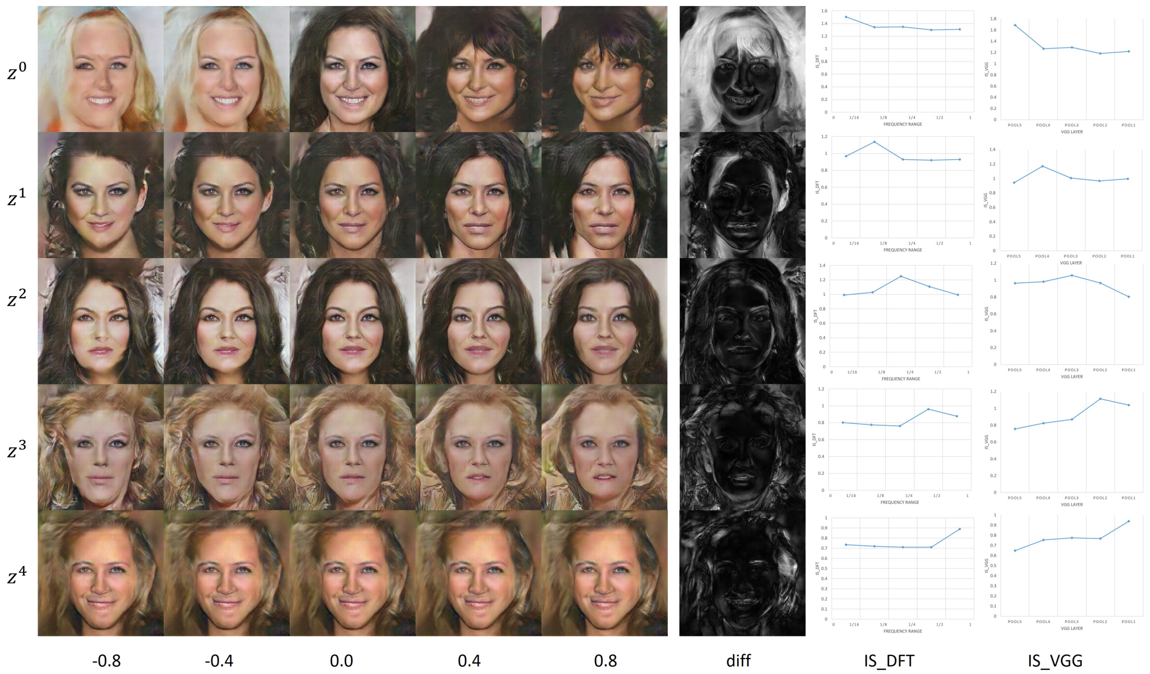


Figure 4.7: Experimental results on $celeba\_hq\_256\times256$ dataset. Column 1-5: $G(\mathbf{z})$, given $z_{\mathbf{z}^t} \in \{-0.8\mathbf{I}, -0.4\mathbf{I}, 0.0\mathbf{I}, 0.4\mathbf{I}, 0.8\mathbf{I}\}$ and $\overline{z_{\mathbf{z}^t}} \leftarrow \mathbf{c}$ ($\mathbf{c}$ is constant and $t \in \{0, 1, 2, 3, 4\}$). Column 6: difference image $\sigma(G(\mathbf{z}))$. Column 7: $IS\_DFT_{\mathbf{z}^t} = \mathbb{E}_{\mathbf{z}_x^t \in \mathbf{z}^t} IS\_DFT_{\mathbf{z}_x^t}$. Column 8: $IS\_VGG_{\mathbf{z}^t} = \mathbb{E}_{\mathbf{z}_x^t \in \mathbf{z}^t} IS\_VGG_{\mathbf{z}^x}$. Note that $\mathbf{I} = (1.0, 1.0, ..., 1.0)$. We find that $IS\_DFT_{\mathbf{z}^0}$ (or $IS\_VGG_{\mathbf{z}^0}$) sees its maximum value in range $(0, 1/16)$ (or $pool5$), which corresponds to coarsest scale. Accordingly, $IS\_DFT_{\mathbf{z}^t}$ (or $IS\_VGG_{\mathbf{z}^t}$) ($t = 1, 2, 3, 4$) see their greatest value corresponding to increasingly small-scales. (Please magnify the **electronic** page to get the figures clearer.)

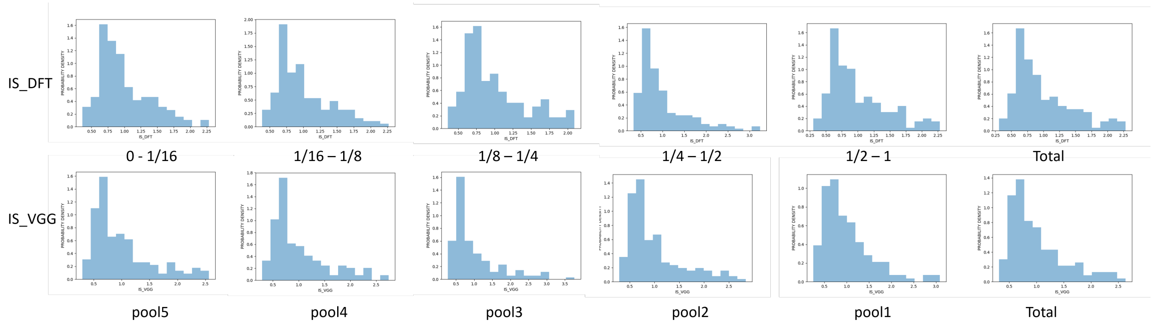


Figure 4.8: Statistic results of manifold (trained on $celeba\_hq\_256 \times 256$ dataset) dimensions by value of impact significance. In contrast to those of traditional GAN as shown in 4.4, our scale-disentangled manifolds see much greater variance of impact significance. The IS_DFT value varies in between $(0.2, 3.0)$, and IS_VGG varies in between $(0.2, 3.5)$. (Please magnify the **electronic** page to get the figures clearer.)

4.4.4 Applications

The multi-scale image manifolds learnt with BranchGAN facilitate coarse-to-fine image synthesis and scale-aware image fusion. For coarse-to-fine image synthesis, we develop a UI-based application that enables users to synthesize desired images by interactively selecting the image of interest: see Figure 4.12. The progressively-improved appearance of the synthesized faces verifies the advantage of scale-disentangled manifolds in enhancing user controllability. The results of *scale-aware image fusion* demonstrated in Figure 4.1 further highlights the amazing effectiveness of scale-disentangled manifolds learned with BranchGAN.

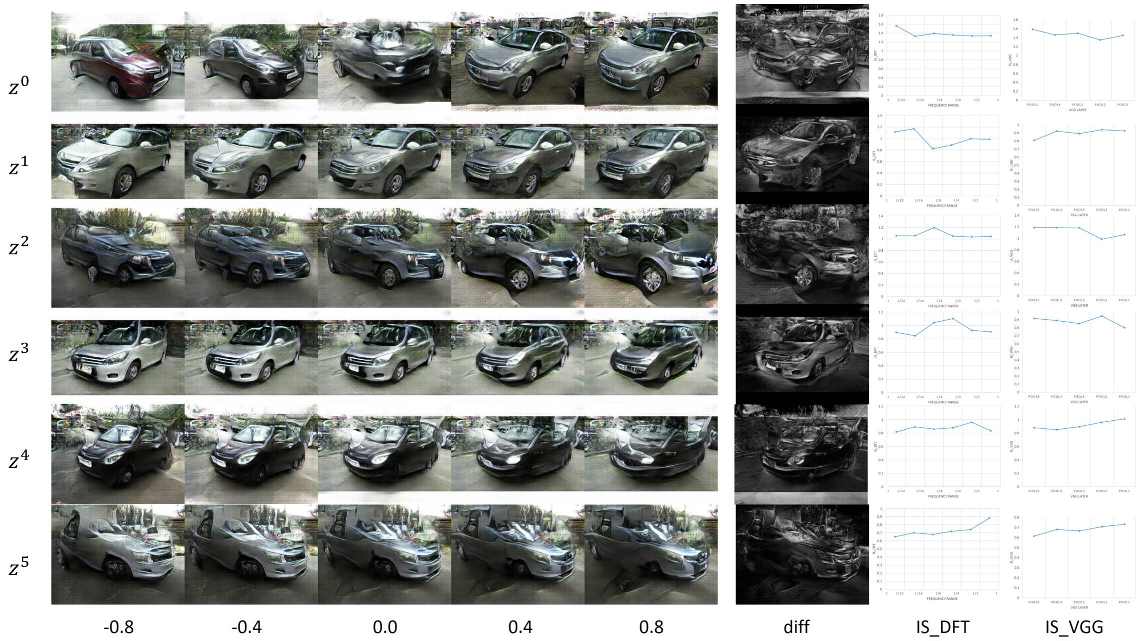


Figure 4.9: Experimental results on $car\_400 \times 300$ dataset. The notation set is the same as the one used in Figure 4.7. $IS\_DFT_{\mathbf{z}^t}$ (or $IS\_VGG_{\mathbf{z}^t}$) ($t = 0, 1, 2, 3, 4, 5$) see their greatest value corresponding to increasingly small-scales. (Please magnify the **electronic** page to get the figures clearer.)

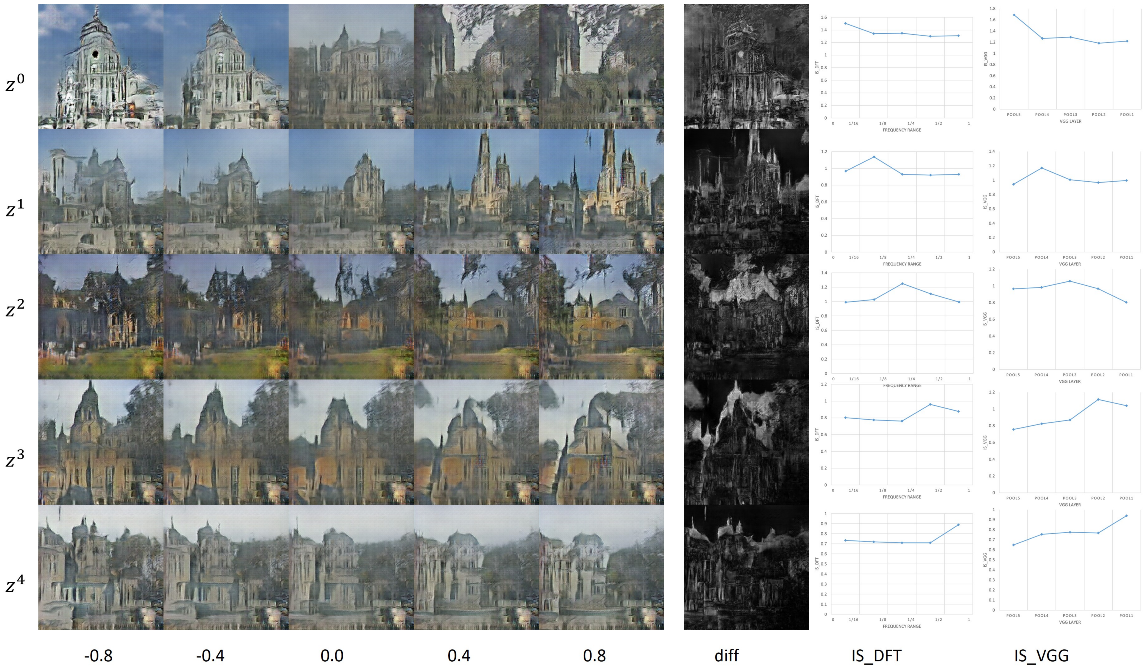


Figure 4.10: Experimental results on LSUN $church\_outdoor\_256 \times 256$ dataset. The notation set is the same as the one used in Figure 4.7. $IS\_DFT_{\mathbf{z}^t}$ (or $IS\_VGG_{\mathbf{z}^t}$) ($t = 0, 1, 2, 3, 4$) also see their greatest value corresponding to increasingly small scales. (Please magnify the **electronic** page to get the figures clearer.)

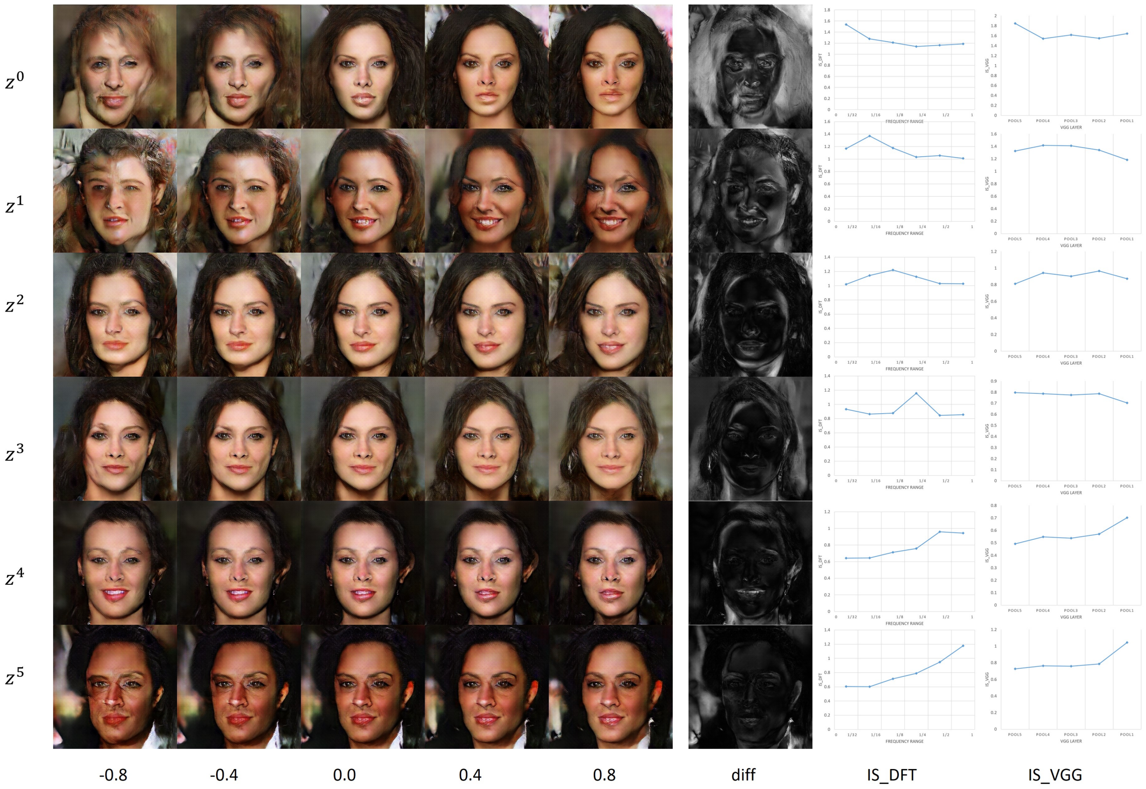


Figure 4.11: Experimental results on $celeba\_hq\_512 \times 512$ dataset. Column 1-5: $G(\mathbf{z})$, given $z_{\mathbf{z}^t} \in \{-0.8\mathbf{I}, -0.4\mathbf{I}, 0.0\mathbf{I}, 0.4\mathbf{I}, 0.8\mathbf{I}\}$ and $\overline{z_{\mathbf{z}^t}} \leftarrow$ **c** (**c** is constant and $t \in \{0, 1, 2, 3, 4, 5\}$). Column 6: difference image $\sigma(G(\mathbf{z}))$. Column 7: $IS\_DFT_{\mathbf{z}^t} = \mathbb{E}_{\mathbf{z}^t_x \in \mathbf{z}^t} IS\_DFT_{\mathbf{z}^t_x}$. Column 8: $IS\_VGG_{\mathbf{z}^t} = \mathbb{E}_{\mathbf{z}^t_x \in \mathbf{z}^t} IS\_VGG_{\mathbf{z}^x}$. Note that $\mathbf{I} = (1.0, 1.0, ..., 1.0)$. We find that $IS\_DFT_{\mathbf{z}^0}$ (or $IS\_VGG_{\mathbf{z}^0}$) sees its maximum value in range $(0, 1/32)$ (or $pool5$), which corresponds to coarsest scale. Accordingly, $IS\_DFT_{\mathbf{z}^t}$ (or $IS\_VGG_{\mathbf{z}^t}$) ($t = 1, 2, 3, 4, 5$) see their greatest value corresponding to increasingly small-scales. (Please magnify the **electronic** page to get the figures clearer.)

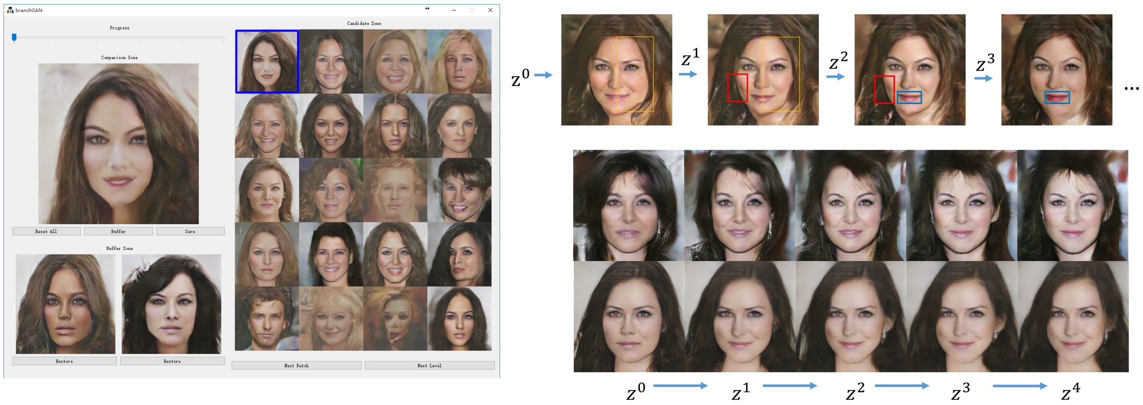


Figure 4.12: Left: UI of the application. Right: images synthesized in coarse-to-fine manner. With a target in mind, the user selects a most-matching face from a batch of randomly-generated faces on the right. At coarse level, the batch of images are mapped from different $\mathbf{z}^0$ values. If the user is satisfied with a coarse-level image, he or she selects it and move on to the next scale. Then the value of $\mathbf{z}_0$ is fixed and a batch of images mapped from different $\mathbf{z}^1$ values are displayed for selection. As a result, the user could progressively improve the appearance of the synthesized face, as seen from the image sequences selected by the user with the goal of synthesizing a female face on the right. The bounding box pair of the same colors highlights image variance (yellow bbox → thinner face, red bbox → less dimple, blue bbox → red lip).

4.5 Summary

Based on our observation of branch suppression when training multi-branch GANs, we design a novel training procedure for unconditional GANs that enables multi-scale image manifold learning. The experimental results on a couple of high-quality datasets verify the effectiveness of our method in "partially" disentangling image manifolds by scales, in which each manifold sub-vector encodes image structures of a specific scale. In the future, we will boost our goal for the scale-independent image manifold learning, where the manifold is fully scale-disentangled. Meanwhile, we will explore the potential value of multi-scale image manifolds in tasks such as image compression, image filtering and image denoising. A third application of multi-scale image manifold is to combine it with iGAN [126] for scale-aware image editing. This would allow image structures of specific scales to be manipulated, while maintaining features in other scales.

Chapter 5

Interactively Manipulating Image Priors for Image Generation

Approaches presented in the previous three chapters automatically synthesize result images based on conditional images. In some applications, however, it is desirable to provide users the control on the output images. Nevertheless, unprofessional manipulations easily produce artifacts and undermine the realism of natural images. One possible way to make image manipulation easier for unprofessional users is to set user inputs as constraints for optimization-based image generation, rather than “fully” accept the free-style inputs as the final result. The previous method known as iGAN [126] serves this purpose quite well. In specific, iGAN searches across the image manifold for the optimal manifold point that best fits user inputs, and then infer the result from the optimal manifold point. In this way, the user inputs are well satisfied on condition that the naturalism of the output is ensured. Despite the success achieved by iGAN, the output suffers poor quality and limited application.

In this chapter, we develop an upgraded version of iGAN (iGAN-HD) that significantly improves the state-of-the-art for manipulating high-resolution images. For high-resolution image manipulation, the manifold learned with classic GANs does not satisfy the needs, as the latent manifold space entangled global structures and local details (see Chapter 4). As a result, any manipulation on local image patch will lead to global changes on image appearance. In response to this issue, we choose and develop the multi-scale manifold learned with BranchGAN (see Chapter 4) which well disentangles global structure and local details. We enrich the edit operations on natural images, as well as integrating a few techniques for GAN training that are recently developed for high-quality image generation. We evaluate iGAN-HD on various tasks, and verify its effectiveness on manipulating images up to 512x512. Quantitative comparisons further validate the advantage of iGAN-HD over other configurations.

5.1 Overview

Image editing/manipulation is a well established field in computer graphics. Whereas, tools like Photoshop typically requires users to have expertise skills. For unprofessional users, even a simple manipulation on natural images could exert artifacts and make the edited image look fake. The major reason is that classic visual manipulations provided in Photoshop do not place any constraints on user inputs, thus not ensuring the output to stay close to natural image manifolds. Nonetheless, on the background of advanced development of natural image manifold learning with Generative Adversarial Networks (GANs) [26], it becomes doable to use the natural image manifold to limit the “free-style” edits. The method known as iGAN [126] is the first model to do this. Specifically, rather than directly perform manipulations upon images, iGAN performs manipulations through gradient-based optimization upon the image manifold, thus avoiding the output to “fall off” the natural image manifold. iGAN provides a paradigm for manipulation on natural image manifolds, which successfully enables free-style user manipulation while maintaining the naturalism of outputs.

However, as iGAN uses the original DCGAN training procedure which get all layers of the generator trained in a single decay, which proves to be incapable of generating images larger than 256x256. To avoid loss of image quality (e.g., texture and details) after optimization, a technique known as “edit transfer” is used, which performs interactive edits to a newly generated image and then transfers the resulting changes of shape and color back to the original image. The technique proves to be highly unstable in practice and tends to exert poor results for images with cluttered background. In addition, the GAN model used by iGAN only works well on structured datasets such as product images. Finally, the brush tools provided in iGAN only allow rough changes in color and shape but do not enable more complex manipulations.

To the end, we propose an upgraded version of iGAN which is named as iGAN-HD. In iGAN-HD, we significantly boost the quality of output images using the “progressive growing” training procedure (see [45] for more details) rather than the naive “all-in-one” training procedure. For high-quality image manipulation, we find out that the classic scale-entangled manifold does not work well in practice, as any patch-wise edits will result in changes of the whole image. The primary reason is that the image manifold learned with classic GANs does not disentangle representations of large-scale structures and finer-scale details. In response, we utilize the multi-scale image manifolds learned with branched adversarial learning (see Chapter 4) to relieve the effects of local edits upon global structures.

Additionally, we significantly enrich the manipulations provided in the user interfaces. Applications of iGAN-HD include generation of images from scratch based on users scribbles and editing of existing images.

5.2 Related Work

5.2.1 Image editing

Image editing is a well-established area in computer graphics, where an image is altered to satisfy users' needs. Previous editing operations such as color retouching [81, 56], sketch drawing [11, 84] and liquifying [113] are mostly based on low-level principles, without considering validness of global structures. More sophisticated editing methods such as image warping [4, 6, 114], patch merging or blending [30, 77] and structural image editing [8] attempt to seek global harmonization. Nonetheless, these methods do not enforce high-level validness about natural images and require professional skills to achieve impressive results. Unprofessional edits easily produce artifacts like unrealistic colors, edge distortion, oversmoothing or prominent repetitions [126].

5.2.2 Image manifold learning

For image manifold learning, the last decades have seen advanced development of approaches such as Restricted Boltzman Machine [35], Deep Belief Network [85], Auto-Encoder [104], Variational Auto-Encoder [48] and GANs [26], among which the GAN-family models are superior to others in terms of high-quality of images generated from manifolds. While these approaches are setup to generate an image starting from a random vector, they do not provide interactive tools for users to control the generation process. iGAN and the proposed iGAN-HD are methods that attempt to make image generation from the image manifold more controllable.

Among the representation-learning GANs, info-GAN [12] and BranchGAN (see Chapter 4) are specially designed to learn meaningful or interpretable manifolds. In specific, info-GAN attempts to learn attribute-disentangled image manifold and BranchGAN aims at learning multi-scale image manifolds. In iGAN-HD, we employ the multi-scale manifold learned with BranchGAN as presented in Chapter 4. When the user only edits a local patch, the multi-scale manifold will only optimize locally to fit the changes brought by the edits, instead of modifying the output globally. We modify the BranchGAN a bit by replacing the originally-used classic training method with WGAN-GP [27] to stablize the training and avoid modal collapse.

Another stream of research attempts to increase the resolution of generated images. Hierarchical GANs [120, 18, 38] define a generator and discriminator for each level of an image pyramid. Durugkar et al. [20] and Wang et al. [106] use one generator and multiple discriminators concurrently. Ghosh et al. [24] do the opposite with multiple generators and one discriminator. The most recent work by Karras et al. [45] introduces progressively-growing training procedure for GANs and they only use one generator and discriminator to generate images up to 1024x1024. In this work, we adopt the progressive-growing training mechanism for image manifold learning.

5.2.3 Manipulation on image manifold

iGAN is proposed by Zhu et al. for interactive manipulation on image manifold [126]. However, iGAN suffers from issues such as poor image quality, inflexible manipulations and limited generality as discussed. In iGAN-HD, we adopt the idea of performing all manipulations through gradient-based optimization and train a network that maps images to latent vectors. Nevertheless, the major differences are, (1) we abandon the technique of "edit transfer", (2) employ multi-scale image manifolds rather than classic manifolds, (3) use more sophisticated training techniques like progressive growing and WGAN-GP, and (4) provide additional interaction tools to facilitate users' editing operations.

5.3 Method

5.3.1 Overview

Figure 5.1 illustrates the overall process of our approach. The operations provided in the user interface include color brushing, edge drawing, image warping, liquifying, and patch blending. Each of the manipulations will produce a temporary editing result. Given a well-trained GAN generator G which takes a manifold vector $\mathbf{z}$ as input and outputs an image, the final editing results are inferred from the optimal manifold point $\mathbf{z}^*$ that best mimics the temporary editing result. We discover that all manipulations could be grouped into three paradigms, which respectively produces an image based on a masked color map (a), an edge map (b) or a complete color map (c). The workflow of each paradigm is shown in Fig. 5.1 (a), (b), and (c). Note that Paradigm (c) differs from Paradigm (a) and (b) in terms of the initialization of manifold point $\mathbf{z}_0$. In specific, Paradigm (c) initializes $\mathbf{z}_0$ with a pretrained

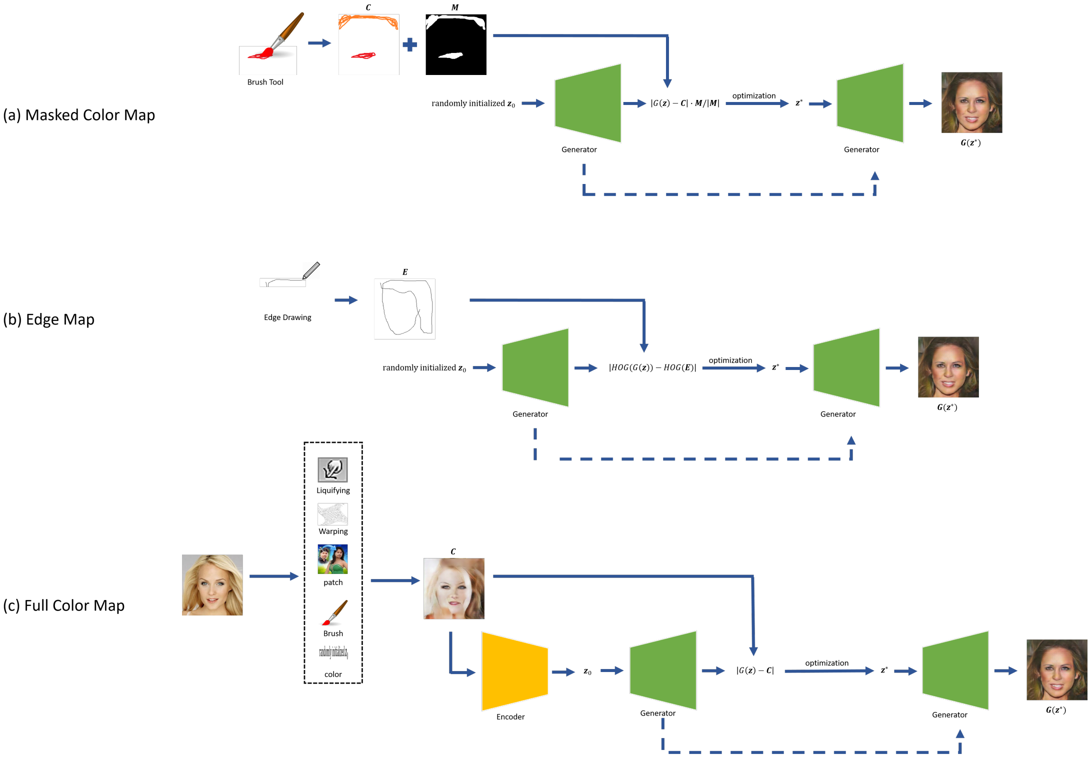


Figure 5.1: Overview of our approach.

encoder that maps an image to $\mathbf{z}$, while Paradigm (a) and (b) initialize $\mathbf{z}_0$ randomly. The objective losses of three paradigms are respectively defined as follows.

$$\mathbf{z}^*_{mask}(\mathbf{C}, \mathbf{M}) = \arg\min |\mathbf{C} - G(\mathbf{z})| \cdot \mathbf{M}/|\mathbf{M}| \tag{5.1}$$

where $\mathbf{C}$ refers to color map and $\mathbf{M}$ is the mask. $|\mathbf{M}|$ denotes the number of non-masked pixels $\sum_{h,w} \mathbf{M}(h, w)$ (h, w are the row and column of the mask).

$$\mathbf{Z}^*_{edge}(\mathbf{E}) = \arg\min |HOG(G(\mathbf{z})) - HOG(\mathbf{E})| \tag{5.2}$$

where E is the edge map and $HOG(\cdot)$ is the differentiable HOG operator [16] which extracts a HOG descriptor from an image.

$$\mathbf{z}^*_{color}(\mathbf{C}) = \arg\min |\mathbf{C} - G(\mathbf{z})| \tag{5.3}$$

where $\mathbf{C}$ is the color map.

5.3.2 Learning multi-scale image manifold

We employ BranchGAN (see Chapter 4) that allows learning scale-disentangled image representations without introducing extra labels. The generator used by BranchGAN is a multi-branch generator which takes multiple scale-separate manifold vectors as input. The principle of "branch suppression" during the process of layer-and-branch growing helps enforce each latent vector to encode structure of separate scale (please see more detials in [118]). To simplify the notation, we concatenate all the latent vectors and refer to the concatenated vector as $\mathbf{z}$. In addition, we replace the original training loss and per-step training mechanism with WGAN-GP [27] to avoid modal collapse.

5.3.3 Predicting manifold vectors from priors

For Paradigm (c), we use a hybrid method to predict latent vector from priors. We trained a network P that predicts the manifold vector based on an input image, by minimizing the training loss $l(\theta_P)$ as shown in Eq. 5.4. A well-trained Network P can only predict the near-optimal manifold vector $\mathbf{z}_0$ of a given image. We then set $\mathbf{z}_0$ as the initial value of $\mathbf{z}$ and start optimizing the objective described in Eq.5.3 to gain a finer approximation of $\mathbf{z}^*$. The two-phase hybrid method proves to be more effective than separate settings [126]. During optimization, the gradient is generated by the objective loss and propagated backward to the latent vector.

$$l(\theta_P) = |G(P(\mathbf{C}, \theta_P)) - \mathbf{C})| \tag{5.4}$$

where θ_P are weights of network P, and $\mathbf{C}$ is the image used for training.

For Paradigm (a) and (b), we initialize the $\mathbf{z}$ with a random noise vector. It is probable that different initial values of $\mathbf{z}$ generate disparate results. To avoid being trapped in local minima, the user could re-initialize $\mathbf{z}$ and redo the optimization. The optimization is played the same way as Paradigm (c).

The paradigms could be joined when multiple constraints are placed. For example, the prior could be an edge map plus a masked color map. In these cases, the latent vector is optimized to satisfy multiple constraints. The objective is set as Eq. 5.5, where α is used to balance between terms. We set $\alpha = 10$ in our experiment.

$$\begin{gathered}\mathbf{z}^*(\mathbf{C}, \mathbf{M}, \mathbf{E}) = \arg\min |\mathbf{C} - G(\mathbf{z})| \cdot \mathbf{M}/|\mathbf{M}| + \alpha|HOG(G(\mathbf{z})) - HOG(\mathbf{E})| \\ \mathbf{z}^*(\mathbf{C}, \mathbf{E}) = \arg\min |\mathbf{C} - G(\mathbf{z})| + \alpha|HOG(G(\mathbf{z})) - HOG(\mathbf{E})|\end{gathered} \tag{5.5}$$

5.3.4 User interface

Fig. 5.2 illustrates the user interface of iGAN-HD. It consists of a main window showing the edit zone (left) and the display zone (right). The edit zone provides various editing operations for drawing edges, producing color strokes or manipulating an existing image. It uses a canvas to facilitate and visualize the edits, whereas the display zone presents the result generated based on the priors. Two types of manipulation tools are provided: Color tools for manipulating the color map & mask and edge tools to produce the edge map. Color tools consist of warping (point-based warping and grid warping), patch editing (loading patch from external or selecting a patch in the current color map), color adjustment

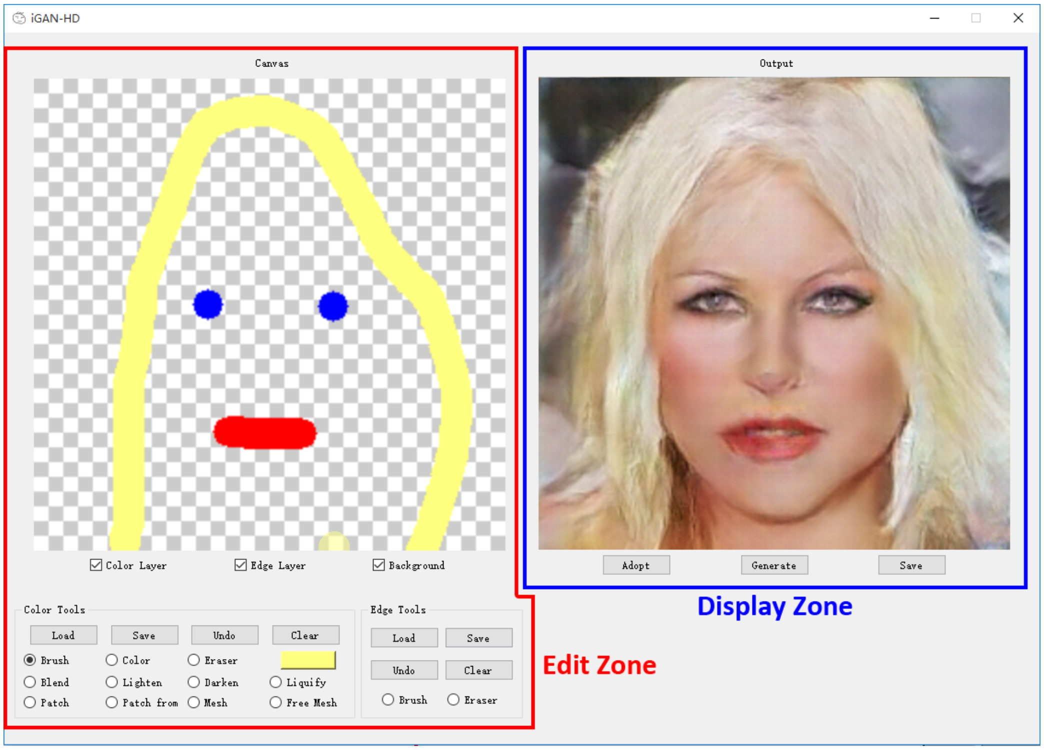


Figure 5.2: The user interface of iGAN-HD.

(lightening, darkening, colorizing or blending), liquify and brush. The user can undo an edit at any time. Erasers are provided for both the color and edge manipulation.

In our experiment settings, we generate images with 256x256 resolution or larger, using a TITAN XP GPU. As higher-resolution images require much more computing resources and GPU memory, real-time response is not enabled. Therefore, we setup a “generate” button which computes and display the result based on the current edits. In addition, we generate only one result at a time. If the user is not satisfied with the result, he/she could re-press the “generate” button to obtain a different result. Due to randomly sampled vector z, the results generated using the same set of constraints can be different.

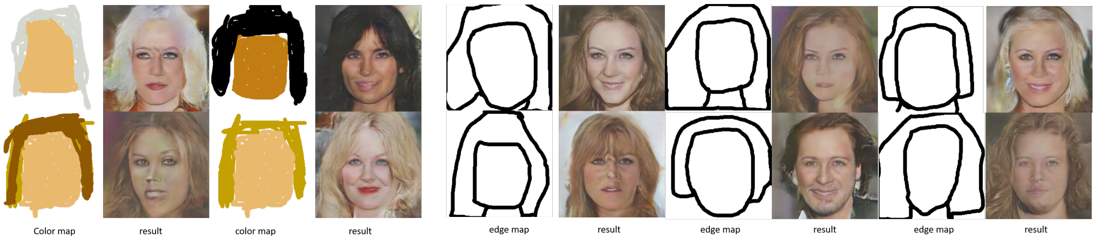


Figure 5.3: The edge maps and color maps drawn by users and the corresponding image generation results.

5.4 Experimental Results and Evaluation

We experiment iGAN-HD on three high-resolution datasets: *celeba_hq* (512x512 and 256x256) [45], *car* (400x300) (see Chapter 4), *lsun church_outdoor* (256x256) [119]. All the training and tests are upon a TITAN XP GPU. Limited by GPU memory, we can only set the image size up to 512x512. Fig. 5.3, 5.4, 5.7, 5.5 and 5.6 illustrate the image generation and editing results based various user inputs.

5.4.1 Comparison

To verify the advantage of multi-scale manifolds learned with BranchGAN (see Chapter 4) over classic manifolds, we did a controlled experiment on various manifold learning methods. We trained four unsupervised GAN models using different training mechanism. We unify all experimental conditions except for the training method, such as the number

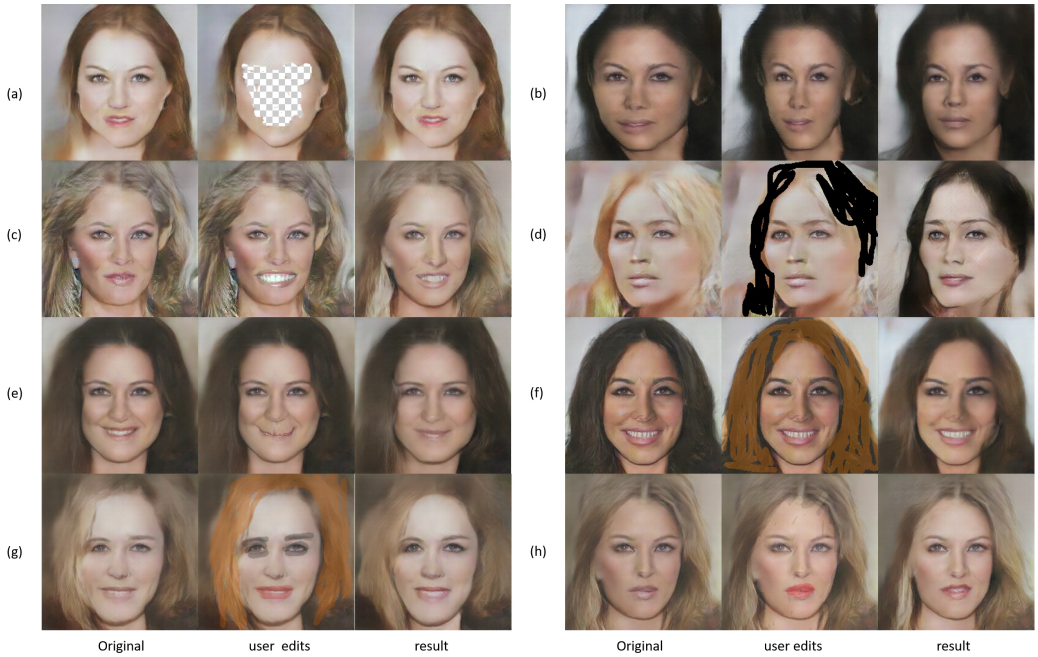


Figure 5.4: Edits by users and the corresponding results. (a) face erased. (b) face slimmed. (c) mouth replaced with a patch from another image. (d) hair darkened. (e) mouth shut. (f) hair brownified. (g) hair brownified & eyeshadowed & lips reddened. (h) face lightened & lips reddened.

of manifold dimensions, the architecture of discriminator and generator, and the dataset for training. The architecture of discriminator and generator used in the control experiment are shown in Table 5.1 and Table 5.2. Note that instance-norm refers to Instance Normalization [103]. We use $celeba_{hq}$ (256x256) for the experiment and the number of manifold dimensions is set as 150. For all the training, we use Adam Optimizer [47] with constant learning rate of 0.0001 and set beta1=0.5, beta2=0.99.

The training methods for each model are DCGAN [79] (used by iGAN), WGAN-GP

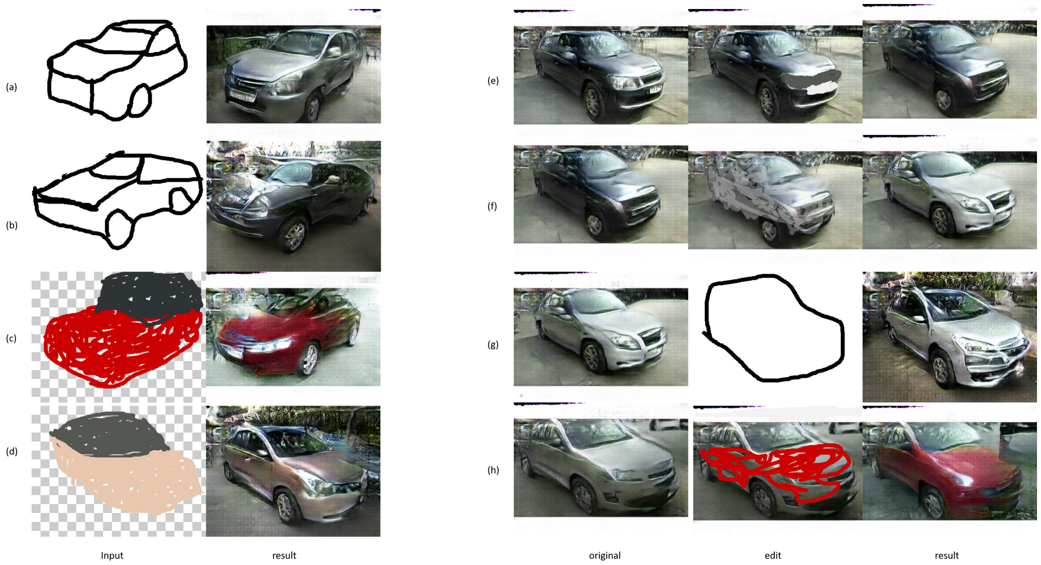


Figure 5.5: Car image generation and editing results. (a)-(b), results based on edge maps. (c)-(d), results based on masked color maps. (e)-(h), image editing results. (e) license plate erased. (f) car lightened. (g) extra edge map added. (h) .

[27], WGAN-GP+layer growing and WGAN-GP+layer&branch growing (please see [45] and [118] for more details about the technique of layer growing and branch growing). To evaluate these methods, we use the minimum loss that measures to what extent the latent manifold vector could be optimized to minimize the objective loss as defined in Eq. 5.3, 5.2 and 5.1. We test each model by feeding a fixed set of user inputs and compute the average minimum loss: see Table 5.3. We find that without using layer growing, DCGAN and WGAN-GP performs equally bad. The technique of layer growing helps improve the results, as it enables GAN to generate high-resolution images. The technique of branch growing further improves the results, as it enables multi-scale manifold learning.

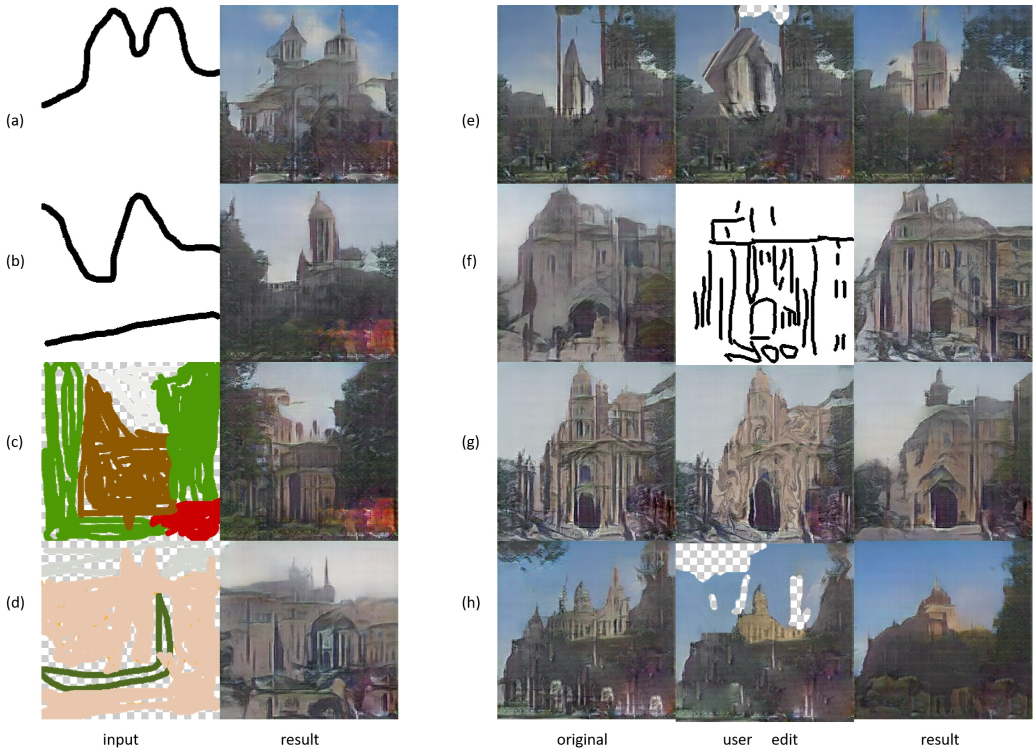


Figure 5.6: Church image generation and editing results. (a)-(b), results based on edge maps. (c)-(d), results based on masked color maps. (e)-(h), image editing results.

5.5 Summary

We presented iGAN-HD, the upgraded version of iGAN, which enables unprofessional users to manipulate natural images with free-style inputs while maintaining the naturalism of the results. We improve iGAN on three major aspects. First, we take advantages of a few sophisticated techniques to increase the resolution and quality of generated results. In addition, we use multi-scale manifolds learned with BranchGAN instead of the classic manifold by DCGAN and prove its advantages in our experiments. Finally, we significantly enrich the interactive operations to enable more complex edits in color, shape and structure. The codes will be made public on Github.

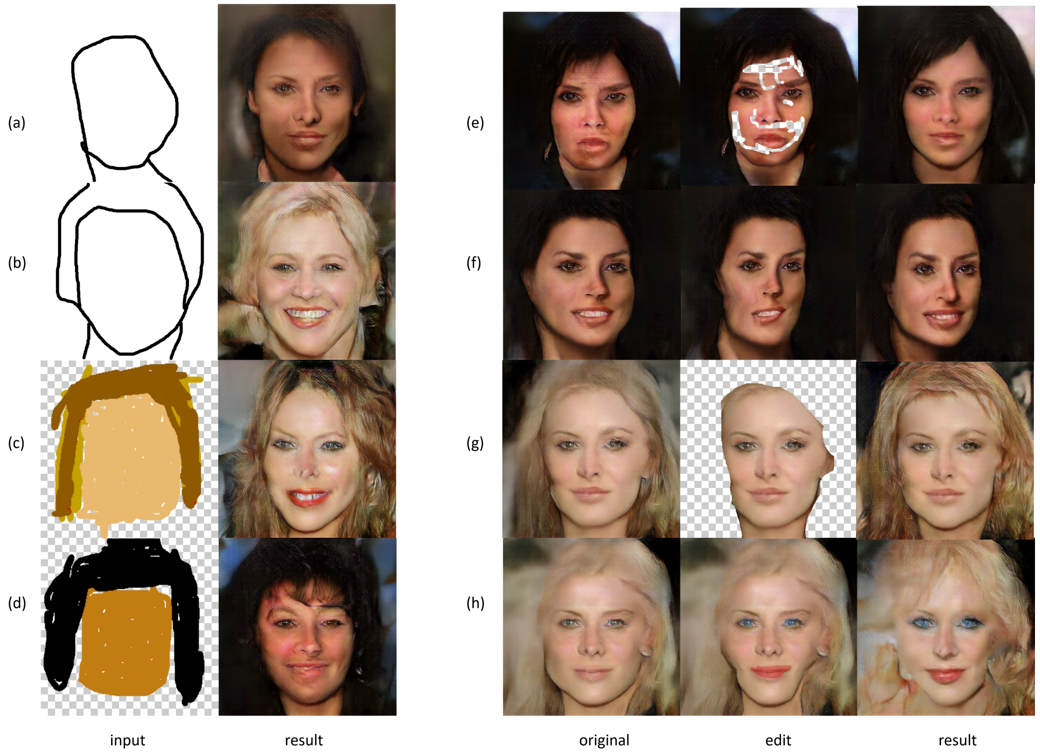


Figure 5.7: Face image (512x512) generation and editing results. (a)-(b), results based on edge maps. (c)-(d), results based on masked color maps. (e)-(h), image editing results.

	Activation Size	Filter Size
input	(256,256,3)	–
conv+instance-norm+lrelu	(128,128,64)	(5,5,3,64)
conv+instance-norm+lrelu	(64,64,64)	(5,5,64,64)
conv+instance-norm+lrelu	(32,32,128)	(5,5,64,128)
conv+instance-norm+lrelu	(16,16,256)	(5,5,128,256)
conv+instance-norm+lrelu	(8,8,512)	(5,5,256,512)
reshape	(32768)	–
linear	(1)	(32768,1)

Table 5.1: The architecture of the discriminator.

	Activation Size	Filter Size
input	(30), (30), (30), (30), (30) or (150)	–
linear	(32768)	(32768,150)
reshape	(8,8,512)	–
deconv+instance-norm+lrelu	(16,16,256)	(5,5,512,256)
deconv+instance-norm+lrelu	(32,32,128)	(5,5,256,128)
deconv+instance-norm+lrelu	(64,64,64)	(5,5,128,64)
deconv+instance-norm+lrelu	(128,128,64)	(5,5,64,64)
deconv+sigmoid (output)	(256,256,3)	(5,5,64,3)

Table 5.2: The architecture of the generator.

dataset	$celeba\_hq$		car	$church\_outdoor$
image size	256x256	512x512	400x300	256x256
DCGAN [79] (used by iGAN)	0.24	0.26	0.29	0.27
WGAN-GP [27]	0.25	0.28	0.27	0.26
WGAN-GP + Layer Growing [45]	0.18	0.17	0.19	0.22
WGAN-GP + Layer Growing + Branch Growing [118]	**0.15**	**0.14**	**0.17**	**0.18**

Table 5.3: Average minimum loss when using different GAN manifolds.

Chapter 6

Conclusions and Future Work

This thesis presents one rule-based method and three learning-based methods for image-conditional image synthesis. In specific, the first method is rule-based and exemplar-based mechanism for content-specific image generation, whereas the others are learning-based and data-driven for general-purpose image generation. As a results, the first approach can only generate non-photorealistic (artistic-stylized) images, whereas the others do not have such constraints. In terms of demand for user interactions, the first these methods can synthesize images in a fully-automated manner, whereas the last approach is semi-automated. It is notable that all four approaches are high-level methods that take content-level image structures into consideration. Compared to previous methods, the presented approaches can either produce more visually appealing results, or require less resources (e.g., paired images, labeled data) and involve less assumptions (e.g., style-specific). Our algorithms have also enabled some novel visual effects, unachievable by any prior works.

In the future, research on image-conditional image synthesis will focus on properties including “higher-resolution”, “self-quality-assured”, “greater flexibility about inquiries” and “increasing novelty and diversity”. Below, we discuss several potential future directions, building on our current image-conditional image synthesis algorithms.

“Self-quality-assured image synthesis”. Current image synthesis models such as perceptual loss, VAE, GAN and pixelRNN all suffer from unexpectable failure products. Failure products is a primary reason preventing fully-automated image synthesis system to be accepted by users. In this scenario, an extra quality-assuring system or quality assessment system is needed to guarantee the outputs to be semantically valid, aesthetically compelling or naturalism-preserved. In the future, we would like to explore how to effectively train image-quality assessors with unlabeled data only.

“Conditioning image synthesis on multi-modal inputs”. Enabling users to interact with an image synthesis system in various manners including speech, posture, texts, drawing and even mental image is a long-term goal. The first step would be developing algorithms that condition image synthesis on multi-modal inputs (e.g., texts+drawing, speech+posture). In the future, we would like to investigate novel neural architectures that could effectively generate images based on multi-modal priors.

Bibliography

[1] burnt picture. http://bbs.wenxuecity.com/art/3047.html. Accessed: 2016-02-29.

[2] Rollip. http://www.rollip.com/start/?src=ws. Accessed: 2016-02-17.

[3] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[4] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164. ACM Press/Addison-Wesley Publishing Co., 2000.

[5] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[6] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, volume 26, page 10. ACM, 2007.

[7] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba. Cross-modal scene networks. *CoRR*, abs/1610.09003, 2016.

[8] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[9] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *Computer Vision–ECCV 2010*, pages 29–43. Springer, 2010.

[10] T. Beier and S. Neely. Feature-based image metamorphosis. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 35–42. ACM, 1992.

[11] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: Internet image montage. In *ACM Transactions on Graphics (TOG)*, volume 28, page 124. ACM, 2009.

[12] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[13] H.-C. Choi, D. Sibbing, and L. Kobbelt. Nonparametric facial feature localization using segment-based eigenfeatures. *Computational Intelligence and Neuroscience*, 501:164290, 2015.

[14] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu. Color harmonization. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 624–630. ACM, 2006.

[15] W. contributors. Surround suppression — wikipedia, the free encyclopedia, 2018. [Online; accessed 13-March-2018].

[16] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[17] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.*, 31(4):82, 2012.

[18] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using laplacian pyramid of adversarial networks. In *NIPS*, pages 1486–1494, 2015.

[19] C. Donahue, Z. C. Lipton, A. Balsubramani, and J. McAuley. Semantically decomposing the latent spaces of generative adversarial networks. In *ICLR*, 2018.

[20] I. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

[21] 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*, pages 341–346. ACM, 2001.

[22] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[23] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[24] A. Ghosh, V. Kulharia, V. Namboodiri, P. H. Torr, and P. K. Dokania. Multi-agent diverse generative adversarial networks. *arXiv preprint arXiv:1704.02906*, 2017.

[25] B. Gooch, G. Coombe, and P. Shirley. Artistic vision: painterly rendering using computer vision techniques. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 83–ff. ACM, 2002.

[26] I. e. a. Goodfellow. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[28] Y.-w. Guo, J.-h. Yu, X.-d. Xu, J. Wang, and Q.-s. Peng. Example based painting generation. *Journal of Zhejiang University-Science A*, 7(7):1152–1159, 2006.

[29] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Optimizing color consistency in photo collections. *ACM Transactions on Graphics (TOG)*, 32(4):38, 2013.

[30] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 4. ACM, 2007.

[31] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[32] P. V. Henstock and D. M. Chelberg. Automatic gradient threshold determination for edge detection using a statistical model, a description of the model and comparison of algorithms. *DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING TECHNICAL REPORTS*, 1996.

[33] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *ICANN*, pages 44–51. Springer, 2011.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[35] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[37] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[38] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 4, 2017.

[39] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[40] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv*, 2017.

[41] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[42] B. Joshi, K. Stewart, and D. Shapiro. Bringing Impressionism to Life with Neural Style Transfer in Come Swim. *ArXiv e-prints*, Jan. 2017.

[43] M. Kagaya, W. Brendel, Q. Deng, T. Kesterson, S. Todorovic, P. J. Neill, and E. Zhang. Video painting with space-time-varying style parameters. *Visualization and Computer Graphics, IEEE Transactions on*, 17(1):74–87, 2011.

[44] S. Kang, X. Qian, and H. Meng. Multi-distribution deep belief network for speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8012–8016. IEEE, 2013.

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

[46] A. Kaspar, B. Neubert, D. Lischinski, M. Pauly, and J. Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015.

[47] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[50] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *ACM SIGGRAPH*, pages 277–286, New York, NY, USA, 2003. ACM.

[51] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)*, 33(4):149, 2014.

[52] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[54] C. e. a. Ledig. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[55] J.-Y. Lee, K. Sunkavalli, Z. Lin, X. Shen, and I. So Kweon. Automatic content-aware color and tone stylization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2470–2478, 2016.

[56] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 689–694. ACM, 2004.

[57] J.-P. Lewis. Texture synthesis for digital painting. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 245–252. ACM, 1984.

[58] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[59] H. Li, G. Liu, and K. N. Ngan. Guided face cartoon synthesis. *Multimedia, IEEE Transactions on*, 13(6):1230–1239, 2011.

[60] H. Li and D. Mould. Contrast-aware halftoning. In *Computer Graphics Forum*, volume 29, pages 273–280. Wiley Online Library, 2010.

[61] H. Li and D. Mould. Artistic tessellations by growing curves. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, pages 125–134. ACM, 2011.

[62] H. Li and D. Mould. Content-sensitive screening in black and white. In *GRAPP*, pages 166–172, 2011.

[63] H. Li and D. Mould. Structure-preserving stippling by priority-based error diffusion. In *Proceedings of Graphics Interface 2011*, pages 127–134. Canadian Human-Computer Communications Society, 2011.

[64] P. Li, H. Sun, B. Sheng, and J. Shen. Image stylization with enhanced structure on gpu. *Science China Information Sciences*, 55(5):1093–1105, 2012.

[65] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001.

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

[67] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[68] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[69] C. Lu, L. Xu, and J. Jia. Combining sketch and tone for pencil drawing production. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, pages 65–73. Eurographics Association, 2012.

[70] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

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

[72] L. Neumann and A. Neumann. Color style transfer techniques using hue, lightness and saturation histogram matching. In *Computational Aesthetics*, pages 111–122. Citeseer, 2005.

[73] L. Northam, P. Asente, and C. S. Kaplan. Stereoscopic 3d image stylization. *Computers & Graphics*, 37(5):389–402, 2013.

[74] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv*, 2016.

[75] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

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

[77] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[78] G.-J. Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv preprint arXiv:1701.06264*, 2017.

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

[80] S. e. a. Reed. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 3, 2016.

[81] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

[82] C. Richardt, C. Stoll, N. A. Dodgson, H.-P. Seidel, and C. Theobalt. Coherent spatiotemporal filtering, upsampling and rendering of rgbz videos. In *Computer Graphics Forum*, volume 31, pages 247–256. Wiley Online Library, 2012.

[83] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[84] K. Ryokai, S. Marti, and H. Ishii. I/o brush: drawing with everyday objects as ink. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 303–310. ACM, 2004.

[85] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. *AISTATS*, 2009.

[86] P. Sallee and B. A. Olshausen. Learning sparse multiscale image representations. In *NIPS*, pages 1351–1358, 2003.

[87] A. Semmo, D. Limberger, J. E. Kyprianidis, and J. Döllner. Image stylization by oil paint filtering using color palettes. In *Proceedings of the workshop on Computational Aesthetics*, pages 149–158. Eurographics Association, 2015.

[88] A. Semmo, D. Limberger, J. E. Kyprianidis, and J. Döllner. Image stylization by interactive oil paint filtering. *Computers & Graphics*, 55:157–171, 2016.

[89] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on signal processing*, 41(12):3445–3462, 1993.

[90] L. Sharan, R. Rosenholtz, and E. Adelson. Material perception: What can you see in a brief glance? *Journal of Vision*, 9(8):784–784, 2009.

[91] X. Shen, A. Hertzmann, J. Jia, S. Paris, B. Price, E. Shechtman, and I. Sachs. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, volume 35, pages 93–102. Wiley Online Library, 2016.

[92] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. Style transfer for headshot portraits. *ACM Transactions on Graphics (TOG)*, 33(4):148, 2014.

[93] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016.

[94] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[95] N. Snavely, C. L. Zitnick, S. B. Kang, and M. Cohen. Stylizing 2.5-d video. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 63–69. ACM, 2006.

[96] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[97] J. M. Susskind, G. E. Hinton, J. R. Movellan, and A. K. Anderson. Generating facial expressions with deep belief nets. In *Affective Computing*. InTech, 2008.

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

[99] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

[100] D. Tschumperlé. Tensor-directed simulation of strokes for image stylization with hatching and contours. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 793–796. IEEE, 2011.

[101] R. Tyleček and R. Šára. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, pages 364–374. Springer, 2013.

[102] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*, 2016.

[103] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[104] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103. ACM, 2008.

[105] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11(Dec):3371–3408, 2010.

[106] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *arXiv preprint arXiv:1711.11585*, 2017.

[107] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pages 318–335. Springer, 2016.

[108] X. Wang and X. Tang. Face photo-sketch synthesis and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(11):1955–1967, 2009.

[109] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402. Ieee, 2003.

[110] C. Wei and D. Mould. Coordinated particle systems for image stylization. In *Proceedings of Graphics Interface 2014*, pages 225–233. Canadian Information Processing Society, 2014.

[111] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):463–476, 2007.

[112] H. WinnemöLler, J. E. Kyprianidis, and S. C. Olsen. Xdog: an extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 36(6):740–753, 2012.

[113] H. Winnemöller, A. Orzan, L. Boissieux, and J. Thollot. Texture design and draping in 2d images. In *Computer Graphics Forum*, volume 28, pages 1091–1099. Wiley Online Library, 2009.

[114] G. Wolberg. *Digital image warping*, volume 10662. IEEE computer society press Los Alamitos, CA, 1990.

[115] Y. Xia, D. He, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma. Dual learning for machine translation. *arXiv preprint arXiv:1611.00179*, 2016.

[116] T. Xiao, J. Hong, and J. Ma. DNA-GAN: Learning disentangled representations from multi-attribute images. In *ICLR*, 2018.

[117] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016.

[118] Z. Yi, Z. Chen, H. Zhang, X. Huang, and M. Gong. Branched generative adversarial networks for multi-scale image manifold learning. *arXiv preprint arXiv:1803.08467*, 2018.

[119] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv*, 2015.

[120] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, pages 5907–5915, 2017.

[121] L. Zhang, L. Lin, X. Wu, S. Ding, and L. Zhang. End-to-end photo-sketch generation via fully convolutional representation learning. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 627–634. ACM, 2015.

[122] W. Zhang, X. Wang, and X. Tang. Lighting and pose robust face sketch synthesis. In *ECCV 2010*, pages 420–433. Springer, 2010.

[123] W. Zhang, X. Wang, and X. Tang. Coupled information-theoretic encoding for face photo-sketch recognition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 513–520. IEEE, 2011.

[124] Y. Zhang, W. Dong, O. Deussen, F. Huang, K. Li, and B.-G. Hu. *Data-driven face cartoon stylization.* ACM, 2014.

[125] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV), to appear*, 2017.

[126] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, pages 597–613. Springer, 2016.

[127] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv*, 2017.

[128] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.