Learning to Synthesize and Manipulate Natural Images

By

Junyan Zhu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexei A. Efros, Chair
Professor Jitendra Malik
Professor Ren Ng
Professor Michael DeWeese

Fall 2017

Learning to Synthesize and Manipulate Natural Images

Abstract

Learning to Synthesize and Manipulate Natural Images

by

Junyan Zhu

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Science

University of California, Berkeley

Professor Alexei A. Efros, Chair

Humans are avid consumers of visual content. Every day, people watch videos, play digital games and share photos on social media. However, there is an asymmetry – while everybody is able to consume visual data, only a chosen few are talented enough to effectively express themselves visually. For the rest of us, most attempts at creating or manipulating realistic visual content end up quickly "falling off" the manifold of natural images. In this thesis, we investigate a number of data-driven approaches for preserving visual realism while creating and manipulating photographs. We use these methods as training wheels for visual content creation. We first propose to model visual realism directly from large-scale natural images. We then define a class of image synthesis and manipulation operations, constraining their outputs to look realistic according to the learned models. The presented methods not only help users easily synthesize more visually appealing photos but also enable new visual effects not possible before this work.

Part I describes discriminative methods for modeling visual realism and photograph aesthetics. Directly training these models requires expensive human judgments. To address this, we adopt active and unsupervised learning methods to reduce annotation costs. We then apply the learned model to various graphics tasks, such as automatically generating image composites and choosing the best-looking portraits from a photo album.

Part II presents approaches that directly model the natural image manifold via generative models and constrain the output of a photo editing tool to lie on this manifold. We build real-time data-driven exploration and editing interfaces based on both simpler image averaging models and more recent deep models.

Part III combines the discriminative learning and generative modeling into an end-to-end image-to-image translation framework, where a network is trained to map inputs (such as user sketches) directly to natural looking results. We present a new algorithm that can learn the translation in the absence of paired training data, as well as a method for producing diverse outputs given the same input image. These methods enable many new applications, such as turning user sketches into photos, season transfer, object transfiguration, photo style transfer, and generating real photographs from painting and computer graphics renderings.

To my parents and fiancée for their love and support

Contents

List of Figures

List of Tables

Acknowledgments

Alyosha sometimes says, "if you wait until the last minute, it will only take a minute." I think he might have gotten it wrong this time. It actually took me more than one minute to write this acknowledgment.

Nevertheless, I would like to thank Alyosha Efros for his guidance, inspiration, enthusiasm, and support throughout my years at CMU and Berkeley. If a student is a generator and an advisor a discriminator, he is probably the best discriminator a student could have. First, he uses all the researchers, mentors, and students whom he has interacted with as valuable training data. Second, during each optimization step, he not only tells me the difference between those scholars and myself, but also provides helpful suggestions on how to close the gap. Third, his guidance is always encouraging and constructive, preventing me from crashing due to gradient explosion. Perhaps this adversarial mentorship sounds a bit complicated. In the end, it is probably just some nearest neighbor mumbo jumbo. I hope I have become closer to one of his training points throughout the years.

As we all know, initialization matters in learning a non-convex function. For that, I would like to thank Zhuowen Tu and Shi-Min Hu for bringing a sophomore to the wonderful world of computer vision and computer graphics. Without their fantastic work, I would not have been attracted to these fields in the first place.

I have had the privilege of conducting and discussing my research with many Berkeley and CMU faculty members, including my thesis members – Jitendra Malik, Ren Ng, Mike DeWeese – as well as Ravi Ramamoorthi, Trevor Darrell, Dawn Song, Bruno Olshausen, John Canny, Abhinav Gupta, Yaser Sheikh, and Kayvon Fatahalian.

Over the summers, I have had the fortune to intern with many researchers, including Aseem Agarwala, Jue Wang, Oliver Wang, Eli Shechtman, Michael Rubinstein, Ce Liu, and Bill Freeman. I would like to especially thank Eli Shechtman, who has shared not only his view of vision and graphics, but also invaluable practical knowledge. His daily support helped push me through the ups and downs during my Ph.D., especially when experiments were not working out. I was extremely fortunate to be able to treat him as a second advisor.

Special thanks to my close collaborators who actually made this thesis possible. I would like to thank Phillip Isola for his philosophical discussions and grand vision. I am amazed that he can always find the positive side of ideas. I also appreciate Philipp Krähenbühl's many useful suggestions over the years, ranging from mathematics and optimization to marriage and bodybuilding. Perhaps the latter two will turn out more helpful. I would like to thank Yong Jae Lee for helping me publish my first paper in graduate school. His valuable guidance and encouragement made the transition from CMU to Berkeley a much easier process.

I would like to thank Tinghui Zhou for his inspiration on cycle-consistency. More importantly, for a very long time, Tinghui was the only reason that my cat and I were not homeless. Tinghui rented his lovely condo to me at a labmate-only discount. If I am lucky enough, I may have a chance to be the last roommate in Tinghui's life. I am looking forward to that.

I would like to thank Richard Zhang for saving me many times from "disaster mode", from projects to teaching to dissertation writing. I enjoyed the pleasant summer we spent next to Lake Sammamish, as well as "playing the nice game." Richard and I once wrote a paper in the first week of our internship, starting during the 800 mile drive from Berkeley to Seattle. We even had to sleep at the company in the last day. Our paper was accepted, but we had to spend another few months polishing the project. I would not recommend this submission procedure in the future.

Special thanks to Taesung Park for fixing many of my bugs in our code base. Our projects would not have been possible without his extraordinary talent and hard work. I enjoyed our Venice trip: the spectacular sunrise, the colorful Burano, the horse2zebra live demo, and the last-minute poster printing. I believe CycleGAN is just the beginning, and I am expecting many more amazing works from his Ph.D.

I would like to acknowledge Ting-Chun Wang for his knowledge on light fields and cameras. I enjoyed numerous lunches and dinners in the food corner, along with the nice discussion about manga, games, and life. We also managed to break a camera, right before a SIGGRAPH deadline.

I would like to thank Shiry Ginosar for valuable discussions, her artistic taste, and paper writing help. She is the best labmate one can have, as she always brought gifts from her trips, including the adorable statue of Ganesha and her recent little dog gift. I was also extremely happy that she loved our Monet paintings to photos results.

I always enjoyed late night discussions with Deepak Pathak. I thought I could do some work at midnight, as nobody would be in the lab. However, Deepak had the same thought. Instead, we would often just chat until 6 am, without doing any work. Nevertheless, I appreciated Deepak's wisdom and creativity, and I predict that

he will become a star scholar soon. We have many unfinished ideas to write, and I hope we can finish some of them before we get too old.

I would also thank all the students and postdocs at Berkeley who provided tremendous support for my study and research. For me, they are the biggest reason why Berkeley is such a great place for doing a Ph.D. I would like to thank many BAIR folks including Andrew Owens, Mathieu Aubry, Olivier Duchenne, Dinesh Jayaraman, Judy Hoffman, Allan Jabri, Bharath Hariharan, Georgia Gkioxari, Shubham Tulsiani, Pulkit Agrawal, Saurabh Gupta, Abhishek Kar, Eric Kuo, Ke Li, Christian Häne, Amir Zamir, Kate Rakelly, Angjoo Kanazawa, Zhe Cao, Shizhan Zhu, Yangqing Jia, Ross Girshick, Jeff Donahue, Evan Shelhamer, Jonathan Long, Eric Tzeng, Lisa Anne Hendricks, Marcus Rohrbach, Anna Rohrbach, Yang Gao, Samaneh Azadi, Ronghang Hu, Huazhe Xu, Dequan Wang, Fisher Yu, Chi Jin, Yi Wu, Qifan Pu, Hezheng Yin, Chang Lan, Sergey Karayev, Daniel Seita, Hyun Oh Song, Jiashi Feng, Biye Jiang, as well as graphics folks including Ling-Qi Yan, Weilun Sun, Rachel Albert, Alex Hall, Pratul Srinivasan, Luxin Yang, Xuaner (Cecilia) Zhang, Fu-Chung Huang, Sean Arietta, Eric Yao, Woojong Koh, Michael Tao, Jiamin Bai, among others. Thanks Zhuang for helping out our GANs project. Thanks David Fouhey for creating many cats memes and the "submission" meme. Thanks Ning Zhang for all the advice, from graduate school application to thesis writing. I am so glad we were born in the same town in Shanghai, and later attended the same high school, college, and graduate school. Berkeley always has impressive undergrads. For that, I would like to thank Hemang Jangle for our stylization project, Tongzhou Wang for our image-to-image project, and Xinyang Geng, Angela Lin and Tianhe Yu for our colorization project.

I would like to thank my old friends at the CMU vision and graphics labs, including Xinlei Chen, Jacob Walker, Hanbyul Joo, Gunhee Kim, Kris Kitani, Yuxiong Wang, Yong He, Yan Gu, Ersin Yumer, Natasha Banerjee, among others. Special thanks to Carl Doersch and Abhinav Shrivastava for many helpful comments and feedback.

I would like to thank my coauthors from other groups and institutions: Jiajun Wu, Yan Xu and Eric Chang for collaboration in multiple instance learning projects, Chaowei Xiao, Warren He, and Bo Li for many helps in adversarial example projects, Ming-Yu Liu for the support in the pix2pixHD project, and Nima Kalantari and Manmohan Chandraker for the guidance in light field projects.

I would like to thank Ian Goodfellow for his encouragement and support for my image generation-related research. I personally learned a lot from him, both through his incredible works and his insightful book. I would like to thank the researchers who made Caffe (Yangqing Jia, Evan Shelhamer, etc.), Torch (NYU/Facebook), and PyTorch (Soumith Chintala, etc.) available. Nothing in this thesis could be built without these wonderful tools. I also thank the many GitHub users who contributed

to our research repositories (e.g., iGAN, CycleGAN, pix2pix, and ideepcolor), which improved the quality of our research codebase.

I would like to thank all my friends (e.g., Biye Jiang, Ling-Qi Yan, Luxin Yang, Angjoo Kanazawa, among others) who helped take care of my cat Aquarius as I was traveling. As Richard noticed, I travelled a lot. But my cat could not have been happier.

Finally, I am grateful to my parents and my fiancée Yijia (and our lovely dog Arya) for their love and support during this wonderful journey. The unbroken bonds between us made me the person who I am today.

Chapter 1

Introduction

Every day, people consume astounding amounts of visual content, as they watch videos, play digital games, and share photos on social media. For example, Facebook alone reports 3 million photo uploads per day, and YouTube sees 300 hours of video uploaded every single minute. As of this writing, an estimated 4.7 trillion photos have been taken since the invention of photography, of which around 20% are from the past 12 months.

The availability of this big visual data has enabled researchers to develop powerful *visual understanding* methods, which aim at compressing visual data, such as images or videos, into abstract representations. For example in Figure 1.1 (top), an image is converted into a single word "street" via scene classification [242], into multiple words "pedestrians", "motorcycles", and "buildings" via object detection [63], or even into a human-like sentence such as "A group of people riding motorcycles on a busy city street" via image captioning [43, 226]. However, there is another side of visual intelligence: *visual synthesis*. It operates in the opposite direction, from compact concepts back into visual data. In this thesis, I would like to teach machines to imagine the realistic visual world from both low-level visual concepts (e.g., texture, lighting, and shape) as well as high-level semantic concepts (e.g. objects and scenes).

But why would *visual synthesis* be useful? Indeed, it can help address the one-sided nature of communication between humans and vast amounts of visual data. While we all perceive information in the visual form (through photographs, paintings, sculpture, videos, etc.), only a chosen few are talented enough to effectively express themselves visually. This imbalance manifests itself even in the most mundane tasks. Consider an online shopping scenario, as shown in the bottom of Figure 1.1. A user looking for shoes has found a pair that mostly suits her, but perhaps she would like them to be a little taller, or wider, or in a different color. How can she communicate her preference to the shopping website? How can we recreate the

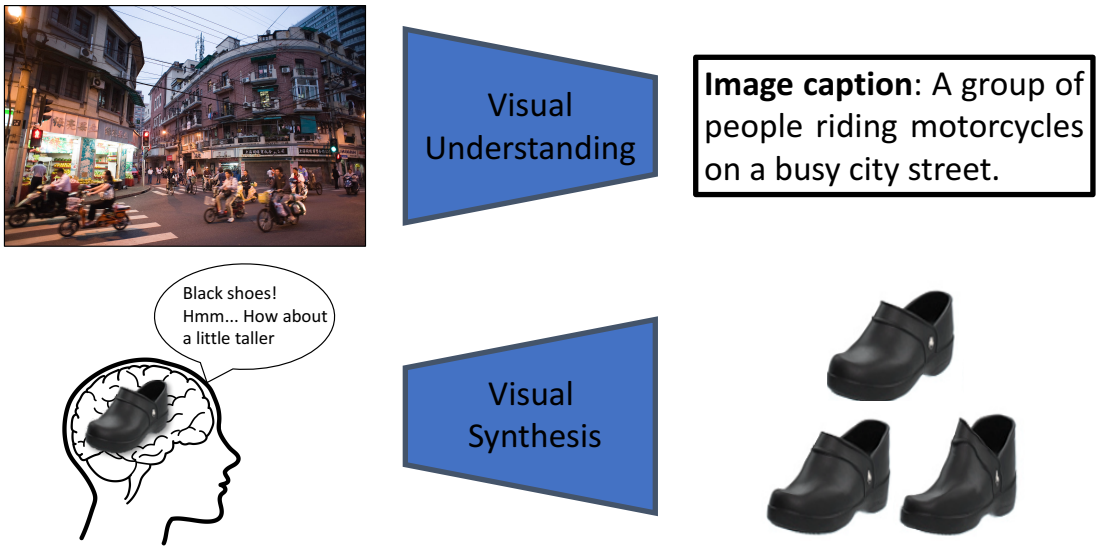


Figure 1.1: Due to vast amounts of visual data and the recent advance of deep neural networks [66], computer vision algorithms have remarkably excelled at *visual understanding* tasks such as generating a human-like sentence from a natural photo. In this thesis, we focus on the opposite direction, *visual synthesis*, with the goal of creating and manipulating natural photographs.

vivid visual world representing the user's mental picture? If the user is an artist, then a few minutes with an image editing program will allow her to transform the shoe into the desired one, and then use image-based search to find it. However, for most of us, even a simple image manipulation in Photoshop presents insurmountable difficulties. One reason is the lack of "safety wheels" in visual synthesis and editing: any less-than-perfect edit immediately makes the image look completely unrealistic, as shown in Figure 1.2(c). To put another way, the classic visual synthesis and manipulation paradigm does not prevent the user from "falling off" the manifold of natural images.

In this dissertation, we investigate a number of data-driven visual synthesis approaches for preserving visual realism while creating and manipulating photographs. Most prior works rely on low-level visual cues such as color and texture for modeling visual realism [88] or hand-crafted engineering to reduce artifacts for individual applications [8, 156]. Unlike these methods, we propose to model visual realism

directly from large-scale collections of natural images. We then use the learned models as training wheels for visual content creation. We define a class of image synthesis and manipulation operations, constraining their outputs to look realistic according to the learned visual realism models.

We first build our methods on two classic machine learning paradigms: discriminative learning and generative modeling. For discriminative learning, we train a classifier to predict photo realism and aesthetics. The classifier can then be used for choosing the best-looking photos as well as computing the optimal parameters of a graphics program. For generative modeling, we directly enforce the synthesis results to lie on the manifold characterized by the learned image generation models, while still satisfying user constraints. We explore several variants of this idea, from a 19th-century-old image averaging model [59] to a state-of-the-art deep generative model [67, 162]. We present real-time applications such as visual data exploration and image editing.

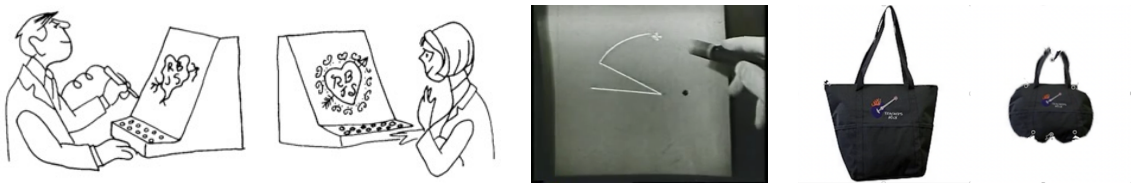
In the above methods, visual realism modeling and image synthesis algorithms serve as two independent system components which are designed and optimized separately. To take advantage of end-to-end learning [66], we combine the realism classifier and image synthesis program into a single image-to-image translation pipeline. Inspired by a recently proposed method known as generative adversarial networks [67], we train a graphics program to translate inputs (such as user sketches) directly to output results, while simultaneously learning a realism classifier to distinguish the synthesized results from natural photographs.

Through many qualitative results and human perceptual studies, we demonstrate that our proposed methods help users easily synthesize more visually appealing photos, compared to the previous state-of-the-art. We also show that our methods enable many new visual effects not possible before, such as turning a running horse video into a zebra video, generating real photographs from Impressionist paintings, and converting an image captured at night into day images with different types of lighting, sky and clouds.

Below we first review previous works on visual synthesis and manipulation (Section 1.1) and visual realism modeling (Section 1.2). We then give a overview of this dissertation in Section 1.3.

1.1 Visual Synthesis and Manipulation

Our goal is to build machines capable of helping humans create and manipulate natural photographs, a field with a rich history. Figure 1.2(a) shows a cartoon from influential work, "The Computer as a Communication Device" [131, 132], written



(a) Communication Device [132] (b) Sketchpad [198] (c) Photoshop tools

Figure 1.2: Early works and modern tools for visual synthesis and manipulation. Decades of research since seminal works such as (a) "The Computer as a Communication Device" [132] and (b) Sketchpad project [198] were proposed, it is still hard for novice users to synthesize and manipulate natural images even with the most advanced tools like Photoshop. Without artistic skills and extensive training, a user often produces noticeable artifacts such as the unnatural deformation of the handbag shown in (c).

by the visionary psychologist and computer science pioneer J.C.R. Licklider at MIT. Already in 1960th, he imagined that we could build a machine to help us communicate our ideas and emotions in pictures, even though we are not good at creating visual objects. Along the same line, the Sketchpad project [198], proposed by Ivan Sutherland, presented a human-machine interface, where a computer can turn a user's intent into simple drawings of objects.

Since then, image synthesis and manipulation have become well-established areas in computer graphics, where an image is created or manipulated to achieve a certain goal specified by a user. Examples of basic editing and synthesis include changing the color properties of an image either globally [166] or locally [127], or synthesizing texture regions [48, 49] and even larger coherent image content [75]. More advanced editing methods also exist such as image warping [90], content-aware image resizing [8], or structured image editing [12] that intelligently reshuffle the pixels in an image following a user's edits. While achieving impressive results in the hands of an expert, when these types of methods fail, they produce results that look nothing like a real image. This is because they rely on low-level principles (e.g., the similarity of color, gradients or patches) and do not capture higher-level information about natural images.

For example, in Figure 1.2(c), a user intends to modify the shape of the handbag to realize a particular design idea, using an advanced commercially available image warping tool. However, a few misplaced control points and sub-optimal user controls produce a less than satisfactory result, both appearing unnatural and far from the user's original objective. Two things are possibly going wrong here. First, the algorithm does not realize that it is currently editing a handbag, rather than a shoe

or a cat. Indeed, many other users have used the same algorithm for editing different object instances from different object categories. Second, more importantly, the algorithm has no clue about what makes a real handbag look like a real handbag. These failures motivate us to propose new methods that can understand both the semantics and the visual realism of the image.

1.2 Learning Visual Realism

The human ability to very quickly decide whether a given image is "realistic", *i.e.* a likely sample from our visual world, is impressive. Indeed, this is what makes good computer graphics and photographic editing so difficult. So many things must be "just right" for a human to perceive an image as realistic, while a single thing going wrong will likely hurtle the image down into the Uncanny Valley [144]. Computers, on the other hand, find distinguishing between "realistic" and "artificial" images incredibly hard.

So what makes an image appear realistic? This is one of the most longstanding problems in computer vision. Back in 1999, Huang and Mumford [88] studied the local statistics of images, ranging from a simple histogram of intensity values to the joint distribution of texture features such as wavelets. Since then, much of the early works were based on generative models [52, 160, 251]. Learning a generative model for full images was challenging at the time due to their high dimensionality. Therefore, these works focused on modeling local properties via filter responses and small patch-based representations. These local patch models work well for low-level imaging tasks such as denoising and deblurring, but are inadequate for capturing the higher-level visual information required for assessing photorealism.

Computer graphics researchers look at the same problem from a different angle. They ask the question "what makes an image look fake?". Researchers investigate common artifacts such as unrealistic colors, exaggerated stretching, obvious repetitions, and over-smoothing. To fix individual artifacts, researchers study low-level visual cues like boundary, color, and texture, and manually design specific algorithms to suppress the artifacts appearing in previous algorithms. However, each new fix may introduce additional artifacts, which require further fixes. As a result, new systems have become more and more complicated and brittle to use in practice. This is because the spectrum of unrealistic images is much larger than the spectrum of natural ones. Any heuristic that works for one aspect may fail in another aspect. Rather than relying on the hand-crafted engineering, in this thesis we aim to learn visual realism directly from large-scale databases of natural images based on the recent advances of highly accurate discriminative classifiers [116] and generative

models [67] that can generate full images.

1.3 Dissertation Overview

In this dissertation, we explore three kinds of approaches: discriminative learning [245, 246], generative modeling [247, 248], and an end-to-end image-to-image translation framework [249, 250] combining the previous two.

Part I. Discriminative Learning of Visual Realism and Aesthetics Chapters 2 and 3 describe discriminative methods for modeling visual realism and photograph aesthetics. Directly training these models requires expensive human judgments. To address this, we adopt active and unsupervised learning methods to reduce annotation costs. We then apply the learned model to various graphics tasks, such as automatically generating image composites and choosing the best-looking portraits from a photo album.

Part II. Generative Modeling for Visual Exploration and Synthesis Chapters 4 and 5 present approaches that directly model the natural image manifold via generative models and constrain the output of a photo editing tool to lie on this manifold. We build real-time data-driven exploration and editing interfaces based on both simpler image averaging models and more recent deep models.

Part III. Image-to-Image Translation Chapters 6 and 7 combine the discriminative learning and generative modeling into an end-to-end image-to-image translation framework, where a network is trained to map inputs (such as user sketches) directly to natural looking results. We present a new algorithm that can learn the translation in the absence of paired training data, as well as a method for producing diverse outputs given the same input image. These methods enable many new applications, such as turning user sketches into photos, season transfer, object transfiguration, photo style transfer, and generating real photographs from painting and computer graphics renderings.

Discussion In Chapter 8, we summarize the contributions of this dissertation and discuss several future directions in deep image synthesis and manipulation.

Part I

Discriminative Learning of Visual Realism and Aesthetics

Chapter 2

Modeling Photo Aesthetics with Active Learning

In this chapter, we aim to learn a discriminative model of photo aesthetics and use the model to choose and produce the best-looking photos. As a case study, we investigate the human perception of portraits regarding different facial expressions, as portraits are among the most popular subjects in photography. Directly training these discriminative models incurs expensive annotations, as we have to collect human subjective ratings across facial expressions and individual subjects. To address this challenge, we introduce an active learning method to reduce the annotation efforts dramatically. We then use the learned model for selecting the most attractive expressions from large video/photo collections. We capture video of a subject's face while engaged in a task designed to elicit a range of positive emotions. We then use crowdsourcing to score the captured expressions for their attractiveness. We use these scores to train a model to automatically predict attractiveness of different expressions of a given person. We also train a cross-subject model that evaluates portrait attractiveness of held-out test subjects and show how it can be used to automatically mine attractive photos from personal photo collections. Furthermore, we show how, with a little bit of extra crowdsourcing, we can substantially improve the cross-subject model by "fine-tuning" it to a new individual, using active learning. Finally, we demonstrate a training app that helps people learn how to mimic their best expressions.

2.1 Introduction

Human faces are one of the most common subjects of photographs. Unfortunately, many of us feel anxiety when a camera is pointed in our direction. What should I do to look good? Will my smile look attractive or awkward? We have all experienced the disappointment of not looking our best in other people's photos. While models and actors are taught how to look good when a camera is pointed at them, the rest of us suffer from a lack of feedback; we simply don't know which of our expressions look good to other people. Self-perception in a mirror can be misleading; the image is horizontally flipped, but more importantly, our perception of ourselves is often very different than that of others [195] since our perception is influenced by our self-image and internal emotions.

There are a number of approaches to editing and improving faces in photographs as a post-process [101,128,230]; however, we often do not have control of photographs taken by others and posted publicly, and many people are not comfortable with the idea of manipulating expressions in photographs. Instead, our goal is to help people look better in photographs *at the time they are taken*. Specifically, our method offers users feedback on how their range of facial expressions are perceived by others, so that they can be better prepared when a camera is pointed at them. Our method can also be used to select the most flattering pictures of people from a photo collection or video.

Our approach begins by capturing a user's range of facial expressions that are appropriate for portraits. We capture a video of the user while they are shown a twelve minute compendium of videos selected to elicit a range of neutral and positive emotions [70]. We then use a novel data-driven computer vision model that automatically predicts the scores of the expressions along two axes: attractiveness and seriousness. (We include the serious attribute so that users can see their best expressions across a range of scenarios, from big smiles in social settings to more neutral expressions for professional portraits.) While this method provides a reasonable approximation of the scores of a user's expressions, it cannot capture all the subtle differences between expressions and variation among users. We therefore also describe a novel crowdsourced, active learning scheme to both customize our model to the user's data and select the user's top expressions across a range of seriousness levels. This active learning scheme reduces the cost of data collection by an order of magnitude over random sampling, to about $5.

We provide a number of interfaces and visualizations to inform the user of the results of our models. The first visualization simply shows the user their most attractive expressions across twenty five levels of seriousness (Figures 2.1,2.4). Next, we offer a number of tools to explore and visualize the data more deeply. For example,

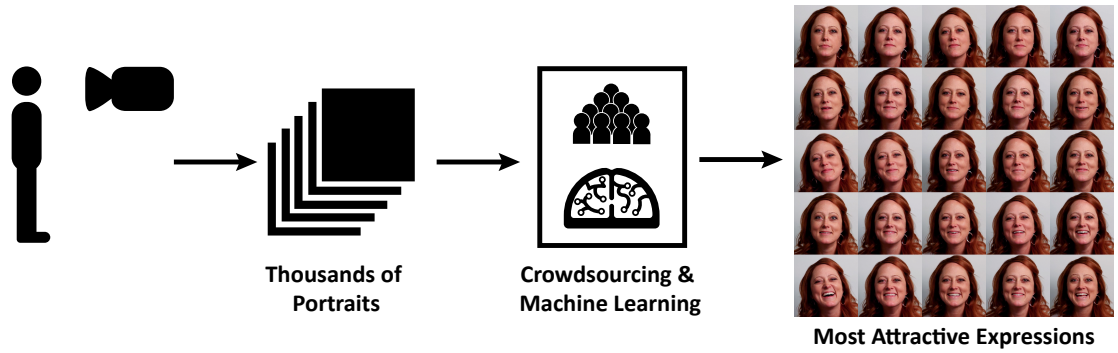


Figure 2.1: We collect thousands of portraits by capturing video of a subject while they watch movie clips designed to elicit a range of positive emotions. We use crowdsourcing and machine learning to train models that can predict attractiveness scores of different expressions. These models can be used to select a subject's best expressions across a range of emotions, from more serious professional portraits to big smiles.

the user can select an expression and suggest a change, e.g., opening the eyes more widely, and see a similar expression with more open eyes and the corresponding change in attractiveness score. The user can also visualize the differences between slices of the data, e.g., the difference between the most and least attractive expressions that contain open eyes. Finally, we also provide an expression training application, called "Mirror Mirror", for practicing expressions in front of a webcam. The user can see their attractiveness and seriousness scores in real-time, and can practice mimicking their best expressions by selecting one and using a visualization that cross-fades between aligned versions of the current and selected expressions.

We test our method on input videos of eleven subjects, and numerically evaluate our methods on hold-out data. We also include a demonstration of the training app to show that subjects can use it to mimic selected expressions. Finally, we apply our method to select the most attractive expressions of a subject from videos downloaded from the internet, as well as personal photo collections.

2.2 Background

The perception of facial expressions is a well-studied topic [24]. The diversity of facial expressions are organized by the Facial Action Coding System (FACS) proposed by Ekman and Friesen [51]; each action unit describes a specific facial motion (e.g., "cheek raiser") and its underlying muscular basis. More recent work [46] suggests that

there are an even larger range of facial expressions than those encoded by FACS. Of particular interest to our application is the difference between an insincere, voluntary smile and a spontaneous smile, which adds a slight narrowing of the eyes. Studies show that a small percentage of people are able to fake spontaneous smiles (also known as "Duchenne smiles") [72, 117], which should yield better portraits. The muscular differences in other subtle smile variations (e.g., amused, polite, nervous) have also been observed [5].

Another area of related research is the differences in social judgments elicited by different faces. Oosterhof and Todorov [152] algorithmically generate different face shapes and measure their perceived traits (attractive, trustworthy, etc.) as scored by humans. They find that most traits approximately lie in a two-dimensional space that can be modeled as a linear combination of two principal components: valence and dominance. We instead model differences of traits between expressions of a single person, and we choose axes that are more relevant to our application (attractive and serious). However, our experiments also show that other traits that may be desirable in a portrait (e.g., trustworthy, confident) are strongly correlated to our chosen axes. Predicting, ranking, and improving the attractiveness or memorability of the faces of different people is a common research topic [4, 68, 102, 110, 128, 230]. We instead focus on the attractiveness of different expressions of *the same person*.

There is significant work in the computer vision literature on the automatic recognition of facial expressions [153]; most of this work focuses on FACS recognition. In contrast, Dibeklioglu et al. [38] predict whether a portrait contains a genuine Duchenne smile. Both Shah and Kwatra [183] and Albuquerque et al. [3] identify smiles from multiple portraits for the purposes of selecting or generating better photographs. None of these techniques can provide a continuous rating of attractiveness of the various facial expressions of an individual. Fiss et al. [54] select facial expressions from a video stream that best serve as candid portraits. However, they optimize for portraits that convey the moment, and many of the selected expressions are not attractive. Also, their method requires temporal features such as optical flow, and cannot be used on photo collections, which we demonstrate in Section 2.8. Finally, our approach to using crowdsourcing to collect ranking and scoring data for subjective attributes of images is inspired by Parikh and Grauman [154], and similar to recent work on font attributes [148] and fashion style [111].

2.3 Overview

Our system has a number of components that can be organized into two main

steps: training and testing.

Training: We begin by collecting a large set of aligned and white-balanced images of unique facial expressions for 11 subjects (Section 2.4). The first step is to score each image along two attributes: attractiveness and seriousness. We use crowdsourcing to collect randomly-sampled pairwise comparisons for each subject and attribute (Section 2.4.3), and then perform MAP estimation to compute attribute scores for each image of each subject (Section 2.5.1). Since we are particularly interested in accurate ranking of the most attractive expressions across different levels of seriousness, we collect additional crowdsourced pairwise comparisons for the highest scoring expressions and re-estimate scores to obtain an even more accurate ranking (Section 2.5.1). These scores for a single subject are used to train a single-subject regression model (Section 2.5.2) that can estimate attribute scores for an image of the same subject. The model takes as input features of a single image (computed in Section 2.4.2), and can operate on previously unseen images of the subject. Finally, we take the scores for all 11 subjects and train a cross-subject regressive model that can operate on images of any subject (Section 2.5.3). This model is more general since it can score a new person's expressions without any additional crowdsourcing; however, it is less accurate than the single-subject model.

Testing: Our system offers a number of applications, such as expression training (Section 2.6) and visualization (Section 2.7), for subjects that are not in our training data. For some applications (e.g., Figure 2.17), we can simply use the cross-subject model to compute attributes. In situations requiring higher accuracy, we first collect images of the new subject's expressions, and use the cross-subject model to compute baseline attribute scores. We use the seriousness scores as-is, since the cross-subject model is accurate enough for this attribute. For attractiveness we use an active learning scheme (Section 2.5.4) to collect a small number of crowdsourced pairwise comparisons. During this step we re-estimate attractiveness scores for each of the subject's images using both the pairwise comparisons and the cross-subject model as a rough prior. Finally, we train an improved single-subject model from the new scores.

2.4 Collecting Portrait Data

Our first goal is to collect a set of portrait expressions of a subject and rate them along attributes that provide useful feedback for portrait posing. However, we first need to determine the range of expressions we wish to capture, and select criteria for good portraits. Clearly, attractiveness is a common goal in most casual portraiture. Also, while most work on facial expression analysis [51, 152] include negative attributes like anger and sadness, these attributes are generally not desired

Figure 2.2: Left: our video capture set-up. Subjects watch videos (played by an iPad on top of a camera) while we record them. Right: example subject expressions.

in contemporary portraits. We therefore restrict our focus to positive attributes. Along with attractiveness there are a number of positive attributes for portraits; for example, we may wish a professional portrait to appear confident, or a sales person may wish to appear trustworthy.

In initial experiments, we collected measurements on portraits for attractive, confident, and trustworthy attributes. However, like previous work [152], we found these attributes to be highly correlated, and therefore redundant. Oosterhof and Todorov show that most attributes can be represented as linear combination of two attributes: valence and dominance. Valence is roughly parallel to attractiveness, while dominance is roughly parallel to aggressiveness. We therefore kept the attractive attribute, and chose to add a second attribute that is parallel to aggressiveness but also useful for our portrait application. We found that the highest rated portraits for attractiveness consistently had large smiles; however, it is also useful to be able to pose well for more neutral expressions without large smiles. We therefore added the "serious" attribute, since it is both a useful control for smile strength, and is nearly parallel to aggressiveness.

In the rest of this section, we first describe how we capture portraits that span a range of positive expressions. Next, we pre-process the portraits to normalize their position and color, extract image features used for predicting attribute scores, and eliminate data redundancy. Finally, we use crowdsourcing to collect pairwise comparisons of portraits along the attractive and serious attributes.

2.4.1 Collecting a Personal Portrait Dataset

We start by collecting a large range of positive facial expressions that may be appropriate for portraits for each subject. We hand-edited together a 12-minute

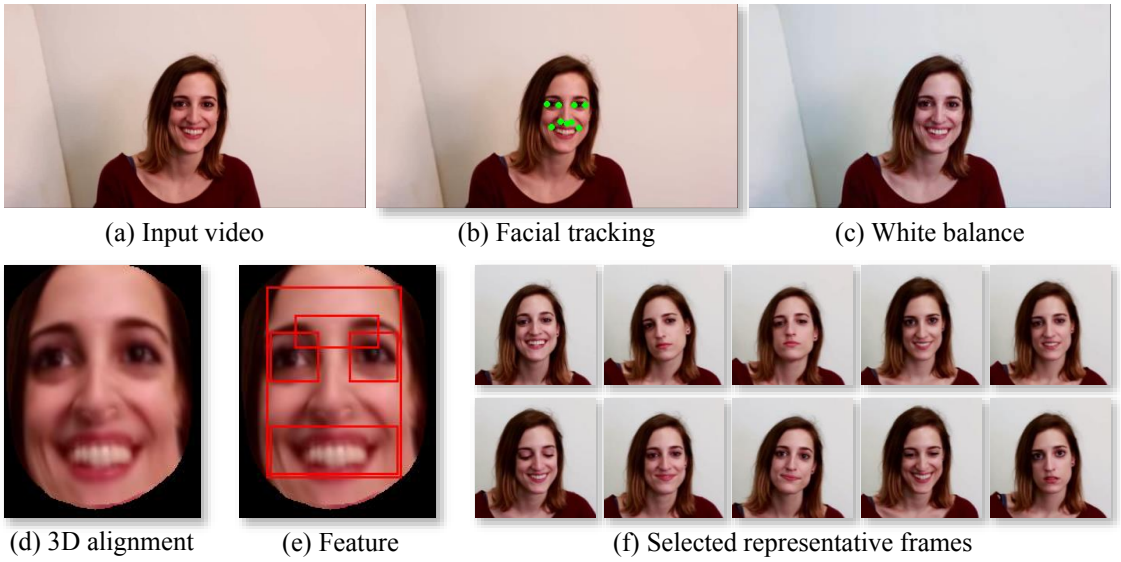


Figure 2.3: We pre-process the input video to align the faces, compute features, and reduce data redundancy.

compendium of short videos that ranged across several categories, including funny, scientific, and inspirational topics. The video is shown on an iPad mounted directly above a SLR camera capturing video, so that it appears the subject is looking at the camera (Figure 2.2). We also asked the subject to make their best portrait expression in several posed categories, such as confident, big open-mouthed smile, etc. Video is often used to elicit emotions for facial analysis [70, 142]. An alternative is to engage in a conversation with the subject [54]; however, mouth motions can make stills unsuitable for portraits. In total, we collected the data of 11 subjects including both male and female subjects ranging in age from 23 to 50.

2.4.2 Pre-Processing

We perform several pre-processing steps (Figure 2.3) for each captured video to align the facial data, compute facial features and reduce data redundancy.

Facial tracking and pose normalization: We first perform tracking and pose alignment to place the face in a common reference frame. We use a recently developed face tracker [225] that accurately estimates nine facial feature points and localizes different facial parts such as eyes, mouth and nose (Figure 2.3b). We apply a median filter with a window size of 5 frames to smooth the estimated points and suppress tracking temporal jitter. Then we align the tracked face to a 3D template model released by [236]. In particular, we estimate a 3D-to-2D transformation matrix

Figure 2.4: Visualizations of the most attractive expressions for three subjects across a range of seriousness (the upper-left is the most serious, the lower-right the least, and seriousness decreases in reading order; attractiveness scores are shown in red). The frames are automatically selected from 12 minutes of video using a combination of crowdsourcing and machine learning.

between the pre-annotated 3D points in the 3D model and the detected 2D facial points using least squares. Finally, we warp the 2D face into a frontal view (174×224) using the computed transformation matrix. We exclude frames for which the tracker reports tracking failures.

Feature extraction: We extract HOG (Histogram of Oriented Gradients) [32] features to capture visual properties of facial expressions in different parts of the face at different scales. Figure 2.3e shows five bounding boxes we use for HOG extraction, which capture two eyes (4×6 cells), eyebrows and wrinkles (2×6 cells), the mouth (2×6 cells) and the whole face (8×6 cells). The cell size for HOG is 8 pixels. Combining features of different parts results in a 3720-dimensional feature vector.

Select representative expressions: Each video typically contains around $16,000$ frames with highly redundant sampling of common expressions; collecting ratings for each frame is impractical. Therefore, we implement a simple greedy algorithm to select unique expressions from the input video. The algorithm starts by randomly selecting a frame I_i from the video, and then removes any other frame I_j which is very similar to the current frame (i.e., $D(I_i, I_j) > T$ where $D(\cdot, \cdot)$ is an appearance similarity function between two expressions and T is a threshold). After the first iteration, we repeatedly select another random frame and remove duplicates until all frames have been processed. The similarity function $D(I_i, I_j)$ is a weighted dot product between the HOG vectors of frames I_i and I_j (after first centering and whitening the HOG vectors [73]). As in previous work [107], we weigh the mouth regions four times as strongly as the other features. We set the threshold

T by binary search with the goal of extracting 200 to 250 unique expressions, which we observe empirically to be a good range for avoiding duplicates while avoiding the elimination of subtle but significant facial expression differences. Figure 2.3f shows several examples of the remaining frames.

White Balance: Some of our videos are not properly white balanced. To reduce the distortion in color space, we white-balance the selected representative frames using Adobe Lightroom before we collect the annotation data.

2.4.3 Crowdsourcing Pairwise comparisons

We next collect human response data that allows us to score the unique expressions along the attractive and serious axes for each portrait subject. We use Amazon Mechanical Turk to collect pairwise comparisons (e.g., "Is expression A more attractive than B?"). Pairwise comparisons are a common approach [205] to collecting subjective scoring data since it is much harder for people to provide absolute scores.

We use separate MTurk HITs (Human Intelligence Task) for attractive and serious attributes, and each HIT only includes portraits from one subject. We provide instructions with two examples of labeled pairwise comparisons from a subject not used in our experiments. Each HIT includes two control questions with obvious answers, along with fourteen unknown comparisons. We discard HITs with incorrect obvious answers, and ban users who fail more than 25% of these tests. No single worker is allowed to complete more than 20 HITs. We pay \$0.06 per HIT. Our system always uses this structure for generating HITs; however, we sample expressions to form pairwise comparisons in different ways (random and active) and at different scales in different parts of our system. We discuss this sampling in the next section.

2.5 Portrait Evaluation

One of the main goals of our system is to output a visualization of the subject's best portrait expressions from a very large input collection of portraits, such as the frames of a video. Our visualization (Figure 2.4) shows the most attractive expressions across 25 discretized seriousness levels; seriousness scores decrease from the upper left to the lower right in reading order (left-to-right, top-to-bottom), and the most attractive image within each seriousness level is shown. These images can be used directly, or the user can select one and use our training app to learn how to mimic its expression.

Supporting these goals requires two types of portrait evaluation. First, we need a function that can score a portrait for both its attractiveness and seriousness. This

score is shown to the user in our expression training app, and could be used to identify the best moment to trigger the shutter on a camera. Second, we need a method to select the most attractive portraits from a large set, i.e., rank them by attractiveness. This ranking is used to visualize a subject's best expressions, and could be used to select the best stills from a video. A ranking can trivially be derived from a scoring function; however, for our application there is a difference in accuracy requirements. For our ranking, the relative ordering of two non-attractive expressions is not important; instead, we want high confidence in our ranking of the top few expressions. At the same time, the scoring function should be reasonably accurate for any portrait.

To accomplish these goals, our method begins by first computing scores for the representative expressions chosen in Section 2.4.2 using crowdsourced pairwise comparisons. We then use these scored images to compute both single-subject (Section 2.5.2) and cross-subject (Section 2.5.3) predictive models. Finally, using the cross-subject model as a rough prior, we learn a more accurate single-subject model with an *active learning* scheme that selects a small number of pairwise comparisons that most increase ranking accuracy (Section 2.5.4).

2.5.1 Scoring Representative Expressions

We estimate attractiveness scores $A = \{a_1, ..., a_n\}$ and seriousness scores $S = \{s_1, ..., s_n\}$ for each of n representative expressions. We denote the pairwise comparison annotations as a count matrix $C = \{c_{i,j}\}$, where $c_{i,j}$ indicates expression I_i is preferred over expression I_j by $c_{i,j}$ times. We use the Bradley-Terry model [18], which models the probability of choosing I_i over I_j as a sigmoid function of the score difference between two expressions, i.e., $P(I_i > I_j) = f(a_i - a_j)$ where $f(u) = \frac{1}{1+exp(-u/\sigma)}$.

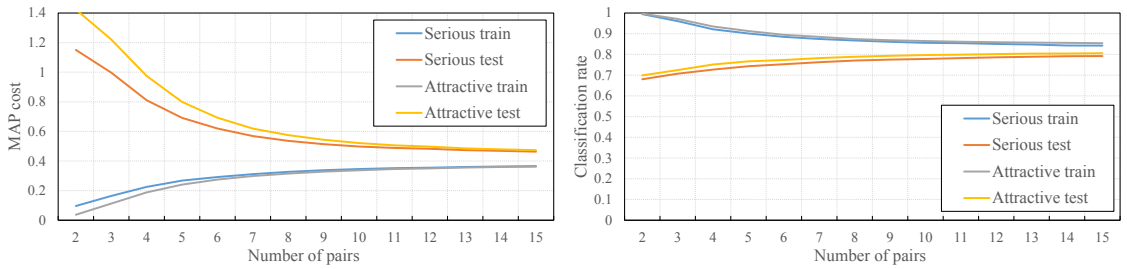
The scores can be estimated by solving a maximum a-posteriori (MAP) problem [205]

$$A^* = \arg\min_A \left(-\log \Pr(C|A) - \log(\Pr(A))\right), \tag{2.1}$$

where $-\log \Pr(C|A)$ is the negative log likelihood of the pairwise comparison data given the model,

and $-\log(\Pr(A))$ is a model prior term. For now we assume A is a uniform distribution; we improve this prior in Section 2.5.4. We can therefore rewrite Equation (2.1) as

$$A^* = \arg\min_A -\sum_{i,j} c_{i,j} \log(f(a_i - a_j)). \tag{2.2}$$



(a) MAP cost (b) Classification rate

Figure 2.5: Convergence of (a) (MAP minimization in Equation 2.2 and (b) classification rate with varying numbers of training pairs per image, for both training and testing data, and serious and attractive attributes.

We solve this equation using gradient descent (with σ in $f(u)$ set to 1), and then normalize scores to $[0, 1]$ for each subject. The same method is used to estimate seriousness scores S.

Convergence

We need to collect enough pairwise comparisons per subject so that the minimization of the MAP energy in Equation 2.2 converges to its minimum. As in previous work [148], we find that convergence occurs in a linear rather than quadratic number of pairwise comparisons. To determine the actual number required, we reserve 5 pairwise comparisons per expression as hold-out test data, and vary the number of randomly sampled training pairs per expression from 2 to 15. (Note that one pair compares two expressions, so 15 pairs means that we sample $15 \times 2 \times n$ expressions in total, i.e., each expression is seen 30 times.) We evaluate this convergence test on three subjects and report the average MAP cost and the classification rate (percentage of pairwise comparisons correctly predicted) as a function of the number of training pairs (2 to 15). The MAP cost is reported for both testing data (the 5 pairs held-out) and the portion of training data used. As shown in Figure 2.5, both metrics converge after about 10 pairs per expression.

Ranking

We can use the scores to rank and select the most attractive expressions across a range of serious levels, as in Figure 2.4. However, MAP convergence does not necessarily mean that the scores are accurate enough to select the best expressions. To explore this question, we first define a *rank error metric* that measures the success

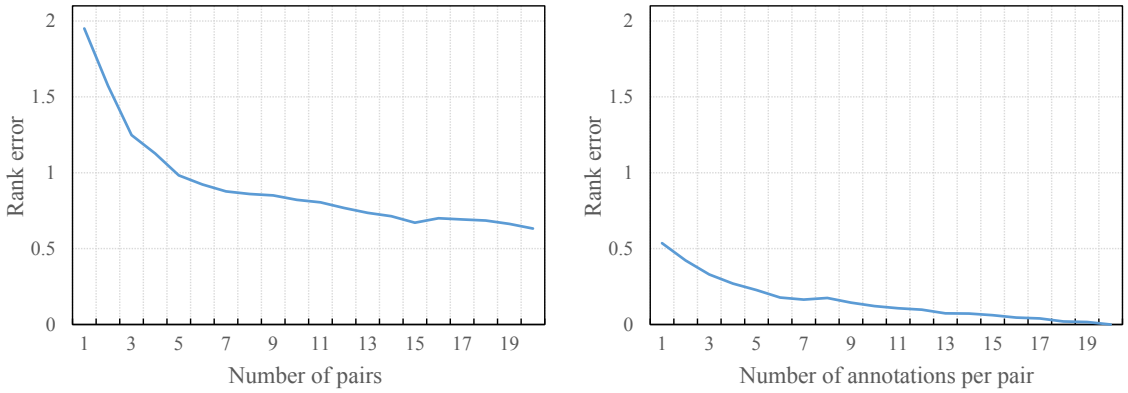
	attractive corr	attractive error	serious corr	serious error
SVR	0.88	0.064	0.90	0.060
GBR	0.88	0.064	0.89	0.063

Table 2.1: Accuracy of the single-subject regression model, reported as correlation and mean absolute error, for two regression methods.

of a selection algorithm. We assume the seriousness score of each expression is known, and there are K serious levels (each level is a range of serious values, as computed in Section 2.5.5). Given a "correct" attractiveness ranking within each serious level we can compute the deviation from this ranking as $\frac{1}{K}\sum_{k=0}^{K}(\pi_k - 1)$, where π_k is the rank of the chosen expression in the k'th serious level in the "correct" ranking. This equation takes the mean of the difference of the rank of the chosen expression (which is 1) and its correct rank π_k. This metric is only concerned with the highest-rated expression in each serious level, since this is the only image shown in our target visualization.

Unfortunately, it is impossible to know whether we have collected enough pairwise comparisons from the crowd to know the "correct" ranking. We therefore generate a baseline ranking as follows. First, we randomly sample 20 pairs per expression for both attractiveness scores and seriousness scores. With this sampling, the MAP error has converged, but the rank error may not have. We therefore generate additional samples that can fine-tune the ranking. We fix the seriousness scores, since these are only used to place expressions into 25 levels, and discard all but the top 10 expressions in each bin. For each pair of these 10 expressions in each bin, we collect an additional 20 pairwise comparisons. That is, we collect 20 redundant opinions for each possible pair. We then re-rank the expressions using this data and our MAP minimization (Equation 2.2). We show rank error relative to this correct ranking in Figure 2.6, both for the initial random-sampled comparisons, and the ranking refinement. We can see while 20 random samples is enough to minimize MAP, it does not minimize rank error. Rank error is reduced to around 0.1 after 10 refinement samples, which means that 9 out of 10 visualization expressions are correct.

This method of generating a "correct" ranking is expensive: $87.8 per subject. We therefore collect this data for only three subjects, as a reference for comparing more efficient methods. In Section 2.5.4, we show how an active learning scheme can reduce this cost to about $5.



(a) Rank error (random sampling) (b) Rank error (refinement)

Figure 2.6: Rank error convergence from method in Section 2.5.1. (a) Mean rank error after varying the number of randomly-sampled pairs per expression. (b) Mean rank error with different numbers of additional pairwise comparisons per expression within each serious level.

2.5.2 Single-subject predictive model

Now that we have scores for the representative expressions of a single subject, the next step is to build a model that can predict attractiveness and seriousness scores for new photos of the same subject.

We train a subject-specific regression model that predicts scores from facial appearance. We use the HOG features described in section 2.4.3, and treat scores estimated in Section 2.5.1 as ground-truth. We experimented with two popular regression models — Support Vector Regression (SVR) [191] and Gradient Boosted Regression Trees (GBR) [56] — and evaluate both methods on all the 11 subjects using 10-fold cross-validation where each fold has 20 to 25 test images. We report correlation and mean absolute errors in Table 2.1. The two methods produce similar results, and we use SVR since it is more efficient. We also tried adding tracking landmark point coordinates (normalized by face size) to our feature vector, as suggested by Khosla et al. [110], but found that it barely boosted prediction performance.

A natural criticism of our approach is that smile and open-eye detectors could be adequate for predicting attractive expressions. To explore this question we use an off-the-shelf smile detector [98], and build our own open-eye detector using the facial tracker landmarks by taking the mean distance of the two points on top of each eye from their corresponding points on the bottom. Larger distances correspond to open eyes; we experimentally confirmed that this metric works well. We train an SVR on our score data using only the 2-dimensional output of the smile and

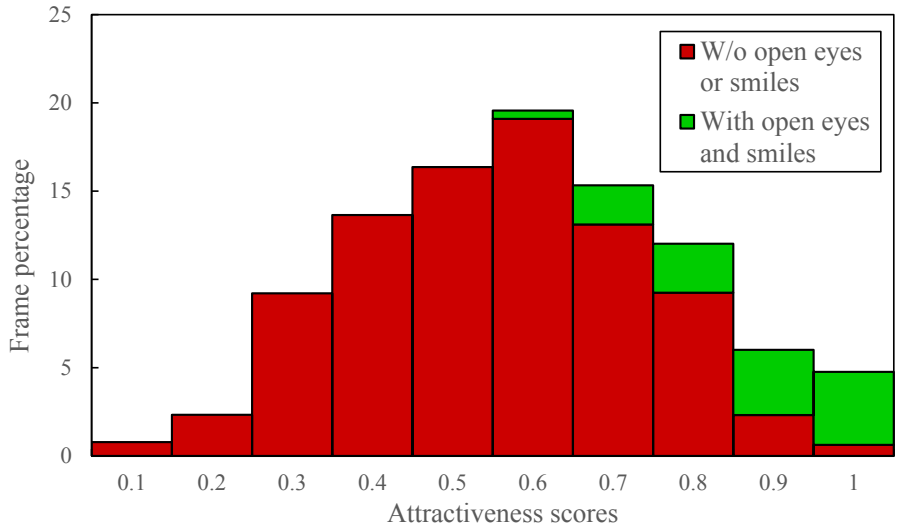


Figure 2.7: Attractiveness scores for three subjects, discretized into 10 bins. Green portions of the histogram indicate open eyes and smiles; the red are the rest.

open-eye detector, combined. Its correlation with the correct attractiveness scores is only 0.47, indicating that our model (with correlation 0.88) is understanding much more than smile size and blinks. The smile detector output has a -0.51 correlation with seriousness, so it is somewhat effective at modeling that attribute.

Our smile and open-eye detectors may not be state-of-the-art; we simulate "ideal" detectors by manually selecting expressions with open eyes and smiles. We show a histogram of the expressions by attractiveness score in Figure 2.7. We can see that while open-eye and smile detectors can filter out the worst images, they miss many of the more attractive expressions.

2.5.3 Cross-subject predictive model

Our single-subject model can predict attractiveness and seriousness scores for one subject given 10 pairs per expression for both attractive and serious attributes. This crowdsourcing costs on average $21.6 for a single subject to achieve a good scoring function, and $87.8 to accurately rank the top expressions, which is too expensive for real-world applications. Given differences between humans and their facial expressions, it is challenging to build a sufficiently accurate completely automatic model for new subjects without any crowdsourcing. However, we should be able to share information between the single-subject models to build a reasonably effective cross-subject model that can at least serve as an initial condition. We therefore combine features and labels

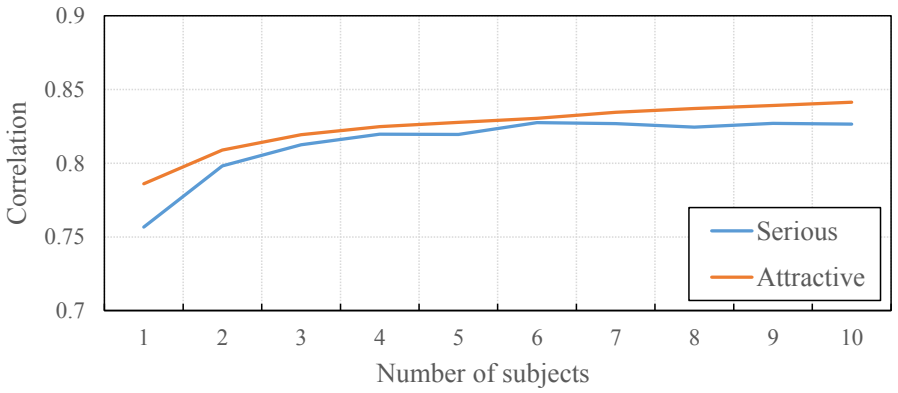


Figure 2.8: Correlation between the expression scores computed in Section 2.5.1 and scores from cross-subject models trained with fewer numbers of subjects. Since there are multiple ways to select x subjects (e.g., for $x = 3$, there are $\binom{11}{3}$ combinations), we randomly select at most 50 combinations and average them to produce plot values.

from different subjects, and train a cross-subject SVR model to predict attractive and seriousness scores using the same method as in Section 2.5.2.

To evaluate the model we hold-out one subject and train on the others, and then average the results of all 11 subjects. The correlation score between the single-subject scores and cross-subject prediction is 0.84 for "attractive", and 0.83 for "serious". The cross-subject model can also be evaluated by its rank error of 1.00; this is significantly higher than the rank errors in Figure 2.6, and suggests that this model alone is not sufficient to accurately select the most attractive expressions.

Adding data for more subjects may improve the cross-subject model. We plot the correlation between the scores computed in Section 2.5.1 and versions of the cross-subject model trained with fewer subjects (from 1 to 10) in Figure 2.8. We can see that seriousness has converged. Attractiveness has mostly converged, but adding a few more subjects will probably slightly improve the model. Also, while our subjects do include a variety of races, genders, and ages, it is likely that there are people for whom our current model will not perform well.

In the end, we use the cross-subject serious model to predict subject-specific seriousness scores since high accuracy is usually not required for this attribute (in our main visualization seriousness scores are only used to assign portraits to serious levels). In the next section we improve the attractiveness score with a small amount of crowdsourcing guided by active learning.

2.5.4 Active Learning

We wish to collect a small amount of crowdsourced data to improve the ranks and scores for photos of new subjects that are first computed with the cross-subject model. The problem of selecting the optimal data to collect during a learning procedure is called *active learning*, and is well-studied. Though most of the literature addresses collecting class labels for objects, several papers address pairwise comparisons while learning to rank data [2,95,130]. Most of these techniques address learning a ranking function that operates on data features, and thus can generalize to new data. In our case, we only wish to rank existing representative expressions. Chen et al. [29] update the Bradley-Terry model we use in Section 2.5.1 to better handle the crowdsourced setting by taking worker quality into account. We could use their method to produce rankings, but our situation is still unique for several reasons. For one, we are most interested in accurate ranking of the most attractive expressions. Two, our expressions are organized into serious levels, and relative ranking within a serious level is most important; on the other hand, the scores of expressions in different serious levels should still be comparable. Three, while there are subtle differences in the attractiveness of expressions across different subjects, there are also significant commonalities (e.g., open eyes and smiles are usually more attractive). We can therefore use scores from the cross-subject model to predict scores that can serve as a prior.

Nonetheless, our active learning scheme follows the same principles of most previous work. We more frequently sample pairs with high uncertainty [2], which corresponds to pairs with similar attractiveness scores. We add to this scheme a preference for sampling more attractive expressions, and a preference for sampling images of similar seriousness scores. (While only sampling pairs within the same serious level would quickly optimizing ranking error, the scores of different levels would drift from each other; we therefore use a soft preference.) Finally, we use scores from the cross-subject model as a prior.

Our method is initialized by computing baseline seriousness and attractiveness scores $S^0 = \{s_1^0, ..., s_n^0\}$ and $A^0 = \{a_1^0, ..., a_n^0\}$ from the cross-subject model. We fix the seriousness scores and do not attempt to improve them, since they are already reasonably accurate and only used to assign expressions to serious levels. We then iterate through active learning rounds $t = 1, ..., T$. In each round we first select n pairs to sample via crowdsourcing. These samples are selected by sampling a probability distribution

$$\Pr(I_i, I_j) \sim e^{-||a_i - a_j||^2/2\sigma_a^2} \cdot e^{-||s_i - s_j||^2/2\sigma_s^2} \cdot e^{-[(1-\tilde{a}_i)^2 + (1-\tilde{a}_j)^2]/2\sigma_h^2} \tag{2.3}$$

where

$$\tilde{a}_i \propto \frac{a_i \sum_j e^{-||s_j - s_i||^2/2\sigma_{\tilde{a}}^2}}{\sum_j a_j e^{-||s_j - s_i||^2/2\sigma_{\tilde{a}}^2}} \tag{2.4}$$

and $\sigma_{\tilde{a}}$ in Equation 2.4 is set to the std. deviation of the seriousness scores. The first factor prefers to sample expressions with similar attractiveness scores, i.e., similar ranks. The second factor prefers to sample similar seriousness scores. The third factor prefers to sample more attractive expressions, according to the current estimate of their scores. We use $\tilde{a}_i$ because directly using a_i leads to under-sampling the more serious levels, since serious and attractiveness scores are negatively correlated. Equation 2.4 normalizes each score a_i by a local weighted average of attractiveness scores, where scores with similar seriousness scores (i.e., close to s_i) are weighted higher. As a result, attractiveness scores that are unusually high for the local range of seriousness are more likely to be sampled. Note that we rescale $\tilde{a}_i$ to [0, 1] after we calculate Equation 2.4. We use σ_a, σ_s and σ_h to weight the relative importance of each factor. (We describe how each parameter is set later.)

Once we have selected samples within a round t, we update the scoring model before iterating. First, new crowdsourced labels are added to the existing crowdsourced annotation data: $c_{i,j} = c_{i,j} + 1$. Next, we minimize Equation 2.1 to compute scores. However, in this case we can use the cross-subject model as a more suitable prior than a uniform distribution. We assume a Gaussian distribution $\Pr(A) \sim N(A^0, \sigma_c^2 I)$ as the prior model of A, where I is the identity matrix. That is, we encourage each expression's score to be similar to the cross-subject score. We can thus re-write the MAP Equation 2.1 as

$$\begin{aligned} A^t &= \arg\min_A -\log \Pr(C|A) - \log(\Pr(A)) \\ &= \arg\min_A -\sum_{i,j} c_{i,j} \log(f(a_i - a_j)) + \frac{1}{2\sigma_c^2} \sum_i ||a_i - a_i^0||^2 \end{aligned} \tag{2.5}$$

where parameter σ_c controls the emphasis of the cross-subject prior relative to the data-fitting term, and σ in the sigmoid function f is set to the std. deviation of the prior scores A^0. We solve Equation 2.5 using gradient descent. Notice that $-\log \Pr(C|A)$ increases its influence as we sample more pairs; we start from the cross-subject model and increasingly rely on personalized crowdsourced data as it arrives. Many expressions with low attractiveness scores may never be sampled at all, and simply be scored by the cross-subject model. On the other hand, highly attractive pairs of expressions may be sampled multiple times with different workers.

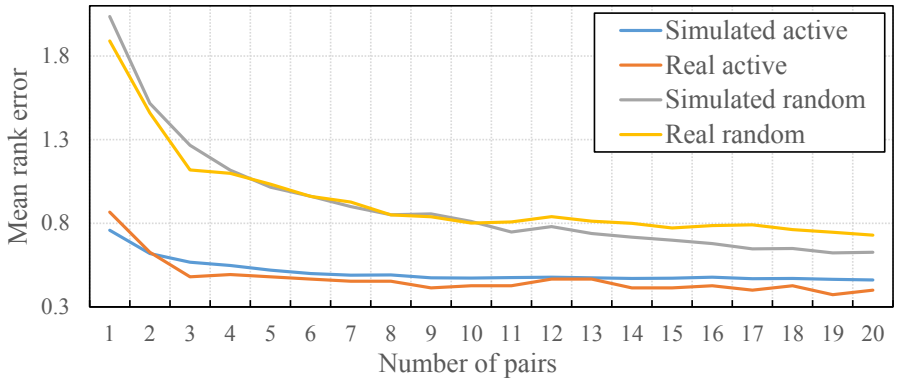


Figure 2.9: Mean rank error averaged across three subjects versus the number of pairwise comparisons per expression for four conditions: active learning versus random sampling, across both real (crowdsourced) and simulated data.

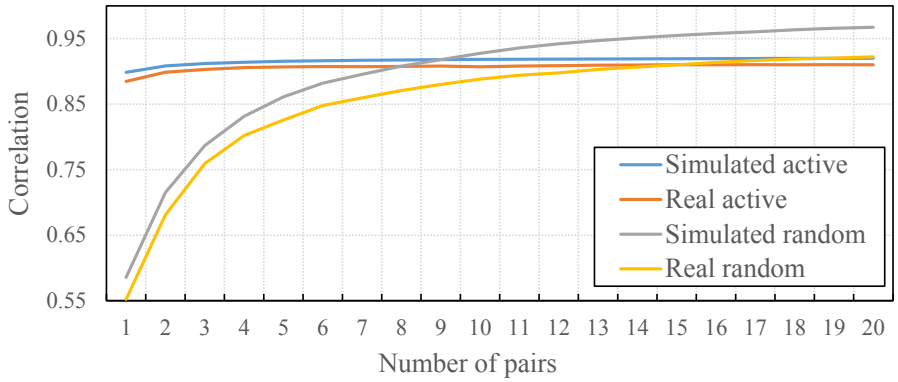


Figure 2.10: Correlation between the scores computed in Section 2.5.1 and scores computed using either active learning or random sampling, across both real and simulated data. Correlations are averaged across three subjects.

Simulated Pairwise Comparisons

Our method has four parameters; we set these to minimize the ranking error on pairwise comparisons generated with a simulation, since online optimization with crowdsourcing would be prohibitively expensive. We take the scores generated by random-sampling pairs in Section 2.5.1, and assume they are ground-truth. We then simulate a Mechanical Turk active learning experiment by generating pairwise labels according to these scores, plus some noise. We label $c_{i,j} = 1$ if a Gaussian random number generator (with bias $a_i - a_j$ and variance σ^i_{worker}) produces a positive

number, as suggested by Thurstone's Law [205]. We model each worker's labeling noise with a Gaussian kernel σ^i_{worker}, where the noise std. deviation of the i'th worker (σ^i_{worker}) is sampled from another Gaussian distribution $N(\sigma_{worker}, \sigma^2_{worker})$. We fit the overall variation in worker noise (σ_{worker}) to actual data from our random sampling experiments by performing a grid search on σ_{worker} between $[0.0, 0.8]$. We can see in Figure 2.9 that our simulation is fairly accurate compared to crowdsourced data. We then set the parameters σ^2_a, σ^2_s, σ^2_h and σ^2_c to values that minimize the ranking error by the end of round 20. The optimized parameters are $\sigma^2_a = 0.02, \sigma^2_s = 0.5, \sigma^2_h = 0.1$, and $\sigma^2_c = 0.5$. Note that the simulation is only run once to set these parameters; it does not need to be run again for new subjects.

Evaluation

We can now evaluate performance over a series of sampling rounds, where each round samples n pairs. We consider four conditions: active learning versus random sampling, across both simulation data and real Mechanical Turk data. Performance can be measured with both mean rank error and the correlation with the attractiveness scores computed in Section 2.5.1, averaged across three subjects. We show these performance metrics in Figures 2.9 and 2.10.

We can see that active learning strongly outperforms random sampling, especially in early rounds, for both simulated and real data. Our active learning scheme can achieve a reasonable accuracy (0.52) with just 5 pairs per expression, while random sampling still has rank error 0.73 after 20 pairs. Using 5 pairs within active learning reduces the crowdsourcing cost to \$5.6, on average, for a subject.

Also, after 5 pairs the active learning scheme gives accurate scores, with a correlation over 0.9.

Our method for ranking portraits has a number of components. The active learning probability for selecting pairwise comparisons in Equation 2.5 has three different factors, and we also use our cross-subject model as a prior. How much do each of these components contribute to the success of our method? We answer this question by turning off individual components and comparing performance using the simulated pairwise comparison data described in Section 2.5.4 (Figure 2.11). We can see that each part of our method does contribute to reducing mean rank error more quickly. The cross-subject prior has the most significant effect, while comparing expressions with similar seriousness scores has the least significant effect.

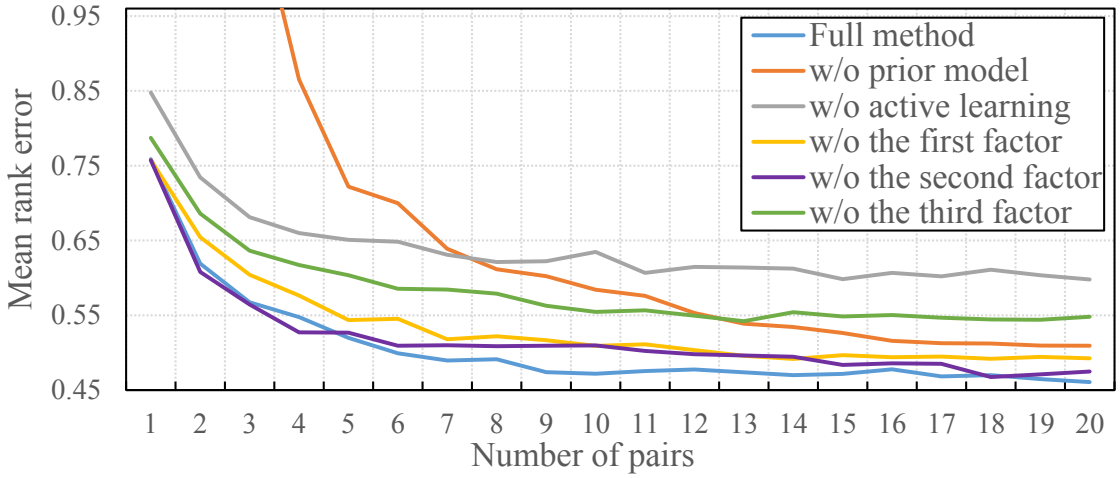


Figure 2.11: Performance achieved after removing individual components of our active learning scheme, computed on simulated pairwise comparisons. We compare: (1) the full active learning scheme, (2) active learning without a cross-subject prior, (3) random sampling plus a cross-subject prior, and (4-6) active learning with a cross-subject prior while removing one of the three factors in Equation 2.3.

.

2.5.5 Visualization details

Finally, we give some technical details on how the visualization in Figure 2.4 is generated.

We first divide seriousness scores into K serious levels, and display the most attractive expression for each serious level. We could simply evenly sample the range of seriousness scores to create serious levels. However, the most attractive expressions tend to be less serious, while there are larger numbers of serious expressions in our input data. The most serious levels may not contain any expressions that are attractive. We therefore divide the seriousness scores into levels based on the idea that the sum of attractiveness scores in each serious level should be about the same.

To compute the number of expressions in each serious level, we first sort attractiveness scores so that their associated seriousness scores are in descending order. We compute the sum of all attractiveness scores, and divide by K to get the target sum of attractiveness for each serious level. Then, we iterate through the sorted attractiveness scores and sum them until we reach expression a_i such that the sum exceeds the target sum for a serious level; the number of expressions in this serious level is set to either i or $i+1$, depending on which minimizes the difference between the current and target sum. The process is repeated until all expressions are assigned to

serious levels. We also found it useful to increase the influence of the most attractive expressions during this binning process by first exponentiating each attractiveness score to a power p. We set $k = 25$ and $p = 4$ in all our experiments.

2.6 Expression Training App

We demonstrate a simple app, called "Mirror Mirror", for training subjects to mimic their best expressions. The app takes input from a webcam and displays the current expression along with its attractiveness and seriousness scores, computed in real-time (about 15fps). Seriousness scores are computed with the cross-subject model after computing features for each input frame; attractiveness is computed from the improved single-subject model computed after active learning. We place a SeeEye2Eye[1] device, which contains a pair of mirrors, on the monitor so that the subject can simultaneously look into the camera and see the camera output.

In training mode the app shows the visualization in Figure 2.4, along with scores of each portrait. The subject can select a target expression to mimic. The app then shows three windows; the current expression, the target expression, and an aligned and a blended cross-fade between the two. The cross-fade oscillates between the target and current expression once per two seconds, so that the subject can examine differences between the two expressions. The target expression is aligned to the current expression and blended to remove visible seams and color differences that might distract from perceiving expression differences. We also show a similarity score between the current and target expression that the user can try to increase. The system automatically saves frames when similarity scores reach new highs; the subject can also pause the system to see fine-grained differences at a frozen moment of time. We show examples in Figure 2.12 that demonstrate that subjects can accurately mimic target expressions using our interface.

After alignment we blend the target expression into the current one by performing color histogram transfer between the two images; we then blend with Laplacian pyramids [22]. We compute the similarity score between the target and current expression with a weighted sum of the difference in attractiveness scores, the difference in seriousness scores, and the projection errors of face alignment landmarks.

[1] http://www.bodelin.com/se2e

Figure 2.12: Two examples from two subjects of using the cross-fade ability of the expression training app to mimic target expressions; the subjects triggered the capture themselves once they were happy with their expression. We show, from left to right, the target expression aligned to the captured expression, the captured expression, the target expression composited into the current expression, and a 50% blend between the previous two images.

Figure 2.13: We show a comparison of average images of unattractive (left) and attractive (right) portraits organized into 10 bins by eye size (top to bottom, we show 6 of 10 bins). Eyes of equivalent size look different between the two sides.

2.7 Data Analysis and Visualization

In this section we use our collected and rated portraits to provide users with useful visualizations, glean insights on the properties of attractive portraits, and explore differences between crowd and subject perception of attractiveness.

2.7.1 Eyes open

In previous work [3,216] it is common to assume that open eyes yield good images, and closed eyes do not. Our analysis shows that the situation is more nuanced. In Section 2.5.2 we created a simple open-eye detector, and found its correlation with attractiveness scores is only 0.45. It is also useful to visualize the difference between attractive and un-attractive photos with the same eye size (Figure 2.13). We show average images of a single subject grouped into attractive (right) and unattractive (left) clusters by score. The y-axis of the visualization is organized by how open the

eyes are; very open eyes are at the top, and closed eyes at the bottom. If we look at the middle bins, we can see a substantial difference in the appearance of good and bad eyes, even though they are open to the same degree. On the left, the eyes appear drugged; the upper eyelid is lowered more substantially than on the right, while the lower eyelid is lower. These bad images usually correspond to expressions in transition (e.g., half-way through a blink). On the right, we can see the same eye size made naturally. Note that smiles often involve narrowing of the eyes.

This observation is consistent with a recent viral video on principles of portrait posing by Peter Hurley[2] that recommends "squinching" (raising the lower rather than the upper eyelid to narrow the eyes). We can see that good eyes of the same size as bad eyes exhibit more squinching.

2.7.2 Subject Preferences and Poses

When subjects are asked to rank their own best portraits, are their opinions consistent with the crowd? We asked four subjects to rank their top three portraits from the visualization in Figure 2.4. Their average rank compared to the first, second, and third choices of the crowd are 10, 11, and 10.7. These ranks suggest that subject preferences are not generally consistent with other viewers. An open question is whether friends of the subject, rather than strangers, would also have different opinions.

Second, we examine the success of subjects at posing upon demand. The beginning of our video designed to elicit emotions asks subjects to first pose for three styles of portraits; an open-mouth smile, a closed-mouth smile, and a neutral professional photo. Then, for seven subjects we look at the top ten attractive portraits, and use the video timeline to determine if they came from portrait posing, or from natural responses to videos. We find that, on average, 7.9 of these ten expressions come from natural response, and 2.1 expressions are posed. The mean rank of the single top posed expression in these top ten is 6.6, versus 1.4 for natural expressions. This difference suggests that subjects do not generally show their best expressions when asked to pose. An alternate explanation is that subjects choose to convey something different with their expressions than what the crowd wishes to see.

2.7.3 Improving Expressions

A subject may like a specific expression, but wish to see if there are similar expressions that the crowd finds more attractive. We therefore generate the visualization

[2] http://www.youtube.com/watch?v=ff7nltdBCHs

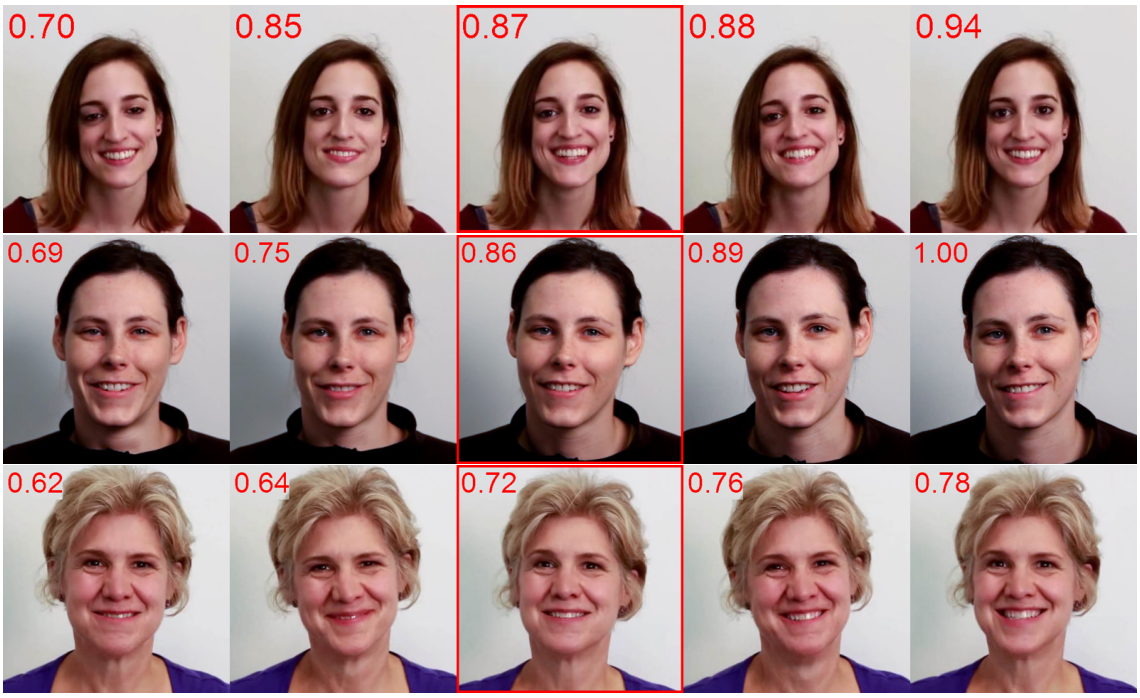


Figure 2.14: Given a query image (middle) we show expressions that are similar but less or more attractive; expressions are sorted by attractiveness score in increasing order. We show examples for three subjects. (Zoom to see subtle differences; attractiveness scores are shown in red.)

shown in Figure 2.14, where a query expression is shown in the middle, and less and more attractive expressions that are similar to the query are shown on the left and right, respectively. This visualization lets the subject see subtle differences between similar expressions and how they may be improved (or worsened).

To create the visualization from a query expression we retrieve the top two most similar expressions who scores are higher than the query, and two that are lower. Similarity is computed as in Section 2.4.2.

2.7.4 Changing One Feature

Another scenario arises when a subject is interested in a specific expression, but wishes to know how changing one feature of the face affects attractiveness. For example, the subject can ask to see different eye or smile sizes, with all other aspects of the face the same. In Figure 2.15 we show examples of different eye sizes, increasing from left to right, for a specific query image (middle). We can see in the first row

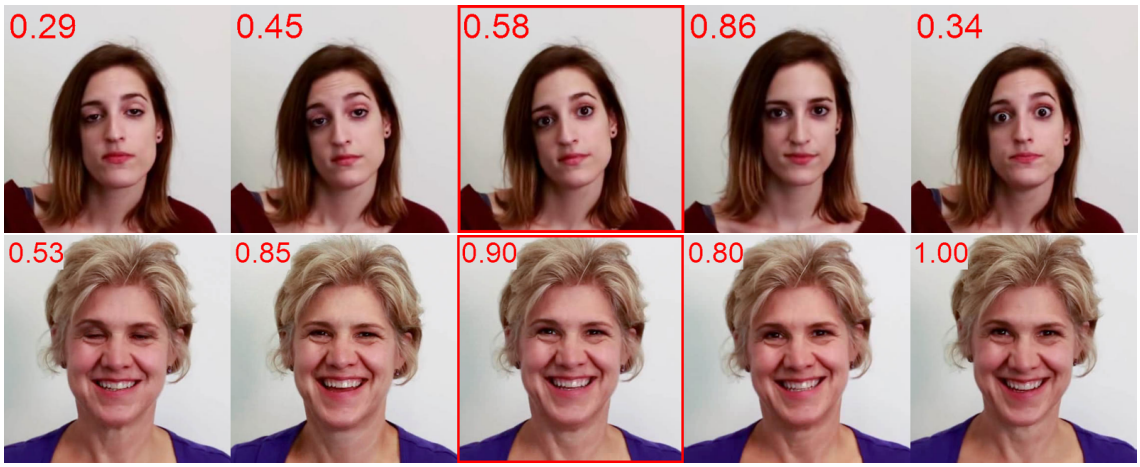


Figure 2.15: Given a query image (middle) we show expressions that are similar but with different eye sizes, increasing from left to right, for two subjects. (Zoom to see subtle differences; attractiveness scores are shown in red.)

that increasing the eye size slightly increases attractiveness, but opening the eyes too widely introduces awkwardness.

To create the visualization from a query expression we select the top two most similar expressions whose eye sizes are larger, and two that are smaller. However, in this case we turn off the HOG eye window when computing similarity, since we do not want the eye appearance to be too similar.

2.8 Results

We tested our method on nine subjects who are not the authors, and two authors; three of these were also tested using the more expensive, randomly-sampled method in Section 2.5.1. We have already numerically evaluated our active learning scheme, and shown results in Figure 2.4. Note that all our results shown in Figures are generated by active learning, rather than random-sampling. We tested our training app on four subjects, and show results of mimicking expressions in Figure 2.12.

We also show that our method works on imagery that we did not capture specifically for this work; in each case, we use only the cross-subject model without any additional crowdsourcing. First, we downloaded a YouTube video[3] on portrait posing; in this video the photographer freezes the frame nine times to indicate good portraits. We select the ten most attractive frames after running peak detection

[3] https://www.youtube.com/watch?v=yrC9eUwPIoo

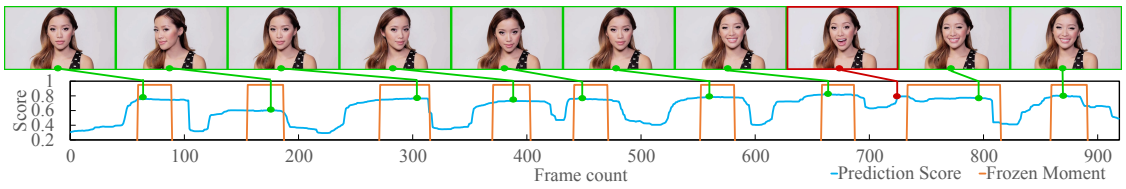


Figure 2.16: The ten most attractive expressions selected by our algorithm run on an Internet video about portrait posing (top). We also show a plot of attractiveness over the frames of the video (bottom); the orange rectangles indicate freeze frames used by the photographer to indicate expressions they select. Remarkably, nine out of ten of our selects come from these freeze-frame regions of the video.

on the attractiveness score signal (to avoid repeating multiple frames of the most attractive expression). Remarkably, nine out of ten selected expressions are the same as those selected by the photographer (Figure 2.16). We show a plot of the attractiveness scores rated by our cross-subject model over time in Figure 2.16.

Next, we try two personal photo collections (Figure 2.17). The first comes from a public person photo dataset [57], which already has faces labeled. The second comes from a personal photo collection; we use Picasa to isolate and identify the subjects, and then automatically remove non-frontal faces (angles larger than 15°) using the pose estimates from the face tracker. We compute the attractiveness score on all faces of specific subjects, and show the ten most and least attractive photos. Note that these photo collections are already partially filtered, so there are fewer very bad photos.

Finally, we add an experiment combining our method with the Photobios feature in Picasa [107]. We filter the representative expressions to images with attractiveness scores greater than 0.6, and set their dates in order of decreasing seriousness. The resulting Photobio shows a smooth animation of attractive expressions from the most serious to the least.

2.9 Discussion

We describe a method that uses a combination of crowdsourcing and machine learning to provide users feedback on their best portrait expressions, and to select their most flattering ones from photo collections and videos. While the graphics and vision communities have focused extensively on improving photos through post-processing, we believe there are numerous opportunities to improve photos *before* they are taken. For example, we could identify which photos or very short videos are most effective at eliciting attractive expressions, and play them before snapping a

Figure 2.17: We show two rows for each of three subjects from personal photo collections: the ten most attractive, and the ten least. We select these expressions from 111, 101, and 85 images of each subject, respectively.

picture. Our large, and often unexplored, collections of photos and videos also offer a large opportunity for identifying flattering content.

Limitations and future work Our method has a number of limitations. While our videos were selected to elicit a wide range of expressions, there is no guarantee that our input video is not missing good expressions of subjects, or that all good expressions can be triggered by watching videos. Also, we only investigate the influence of expression on attractiveness; there are many other factors, such as lighting, camera viewpoint and angle, makeup, and hair. These other factors may not be independent of expression. Though we demonstrate some results on faces captured from an angle, our current methods are not trained on profile or near-profile views.

The most fundamental question about our expression training app is whether it actually helps people pose better for portraits. Conducting this user study accurately would require evaluating the attractiveness of photos from portrait photography

sessions before and after using the app; the second session should not be immediately after the training session, to avoid improvements that are only short-term. We leave this more ambitious user study to future work. Our expression training app is only a proof-of-concept for now; it remains an open question whether people can be trained to make certain expressions, or how training compares to other alternatives (such as remembering certain happy or funny moments).

Finally, while we describe methods to select the best expressions, a subject may wish to slightly modify an expression to increase its attractiveness. Using our scoring model to optimize image edits or warps is a promising avenue for future work.

Chapter 3

Learning Visual Realism without Human Supervision

The above active learning method can reduce the labeling cost by a large margin. However, it is not scalable to typical graphics scenarios, which include many parameters rather than just one (e.g., facial expression). Active learning methods still require a prohibitive amount of human-labeled data, as these parameters are combinatorial and the space of plausible generated images is so vast. To further address the annotation issue, this chapter introduces a method for learning the perception of visual realism directly from large amounts of *unlabeled* data, without using any labels regarding human perception. We focus on image compositing, as it contains many factors such as the inserted object instance, as well as the lighting and color of the inserted objects. In particular, we train a Convolutional Neural Network (CNN) model [124] that distinguishes natural photographs from automatically generated composite images. The model learns to predict visual realism of a scene regarding color, lighting and texture compatibility, without any human annotations pertaining to it. Our model outperforms previous works that rely on hand-crafted heuristics for the task of classifying realistic vs. unrealistic photographs. Furthermore, we apply our learned model to compute optimal parameters of a compositing method, to maximize the visual realism score predicted by our CNN model. We demonstrate its advantage against existing methods via a human perception study.

3.1 Introduction

As mentioned in Chapter 1, the human ability to very quickly decide whether a given image is "realistic", *i.e.* a likely sample from our visual world, is very impressive.

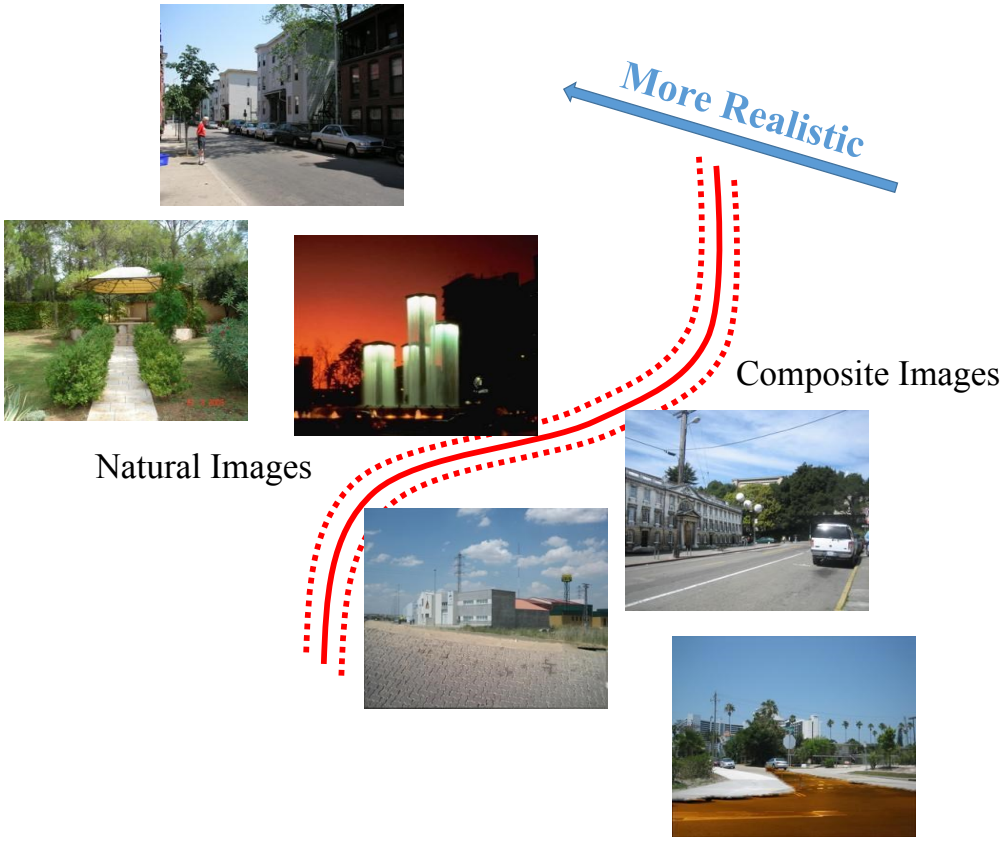


Figure 3.1: We train a discriminative model to distinguish natural images (top left) and automatically generated image composites (bottom right). The red boundary illustrates the decision boundary between two. Our model is able to predict the *degree* of perceived visual realism of a photo, whether it's an actual natural photo, or a synthesized composite. For example, the composites close to the boundary appear more realistic.

Indeed, this is what makes good computer graphics and photographic editing so difficult. So many things must be "just right" for a human to perceive an image as realistic, while a single thing going wrong will likely hurtle the image down into the Uncanny Valley [144].

Computers, on the other hand, find distinguishing between "realistic" and "artificial" images incredibly hard. Much heated online discussion was generated by recent results suggesting that image classifiers based on Convolutional Neural Network (CNN) are easily fooled by random noise images [147, 199]. But in truth, no existing

method (deep or not) has been shown to reliably tell whether a given image resides on the manifold of natural images. This is because the spectrum of unrealistic images is much larger than the spectrum of natural ones. Indeed, if this was not the case, photo-realistic computer graphics would have been solved long ago.

In this chapter, we are taking a small step in the direction of characterizing the space of natural images. We restrict the problem setting by choosing to ignore the issues of image layout, scene geometry, and semantics and focus purely on appearance. For this, we use a large dataset of automatically generated image composites, which are created by swapping similarly-shaped object segments of the same object category between two natural images [119]. This way, the semantics and scene layout of the resulting composites are kept constant, only the object appearance changes. Our goal is to predict whether a given image composite will be perceived as realistic by a human observer. While this is admittedly a limited domain, we believe the problem still reveals the complexity and richness of our vast visual space, and therefore can give us insights about the structure of the manifold of natural images.

Our insight is to train a high-capacity discriminative model (a Convolutional Neural Network) to distinguish natural images (assumed to be realistic) from automatically-generated image composites (assumed to be unrealistic). Clearly, the latter assumption is not quite valid, as a small number of "lucky" composites will, in fact, appear as realistic as natural images. But this setup allows us to train on a very large visual dataset without the need of costly human labels. One would reasonably worry that a classifier trained in this fashion might simply learn to distinguish natural images from composites, regardless of their perceived realism. But, interestingly, we have found that our model appears to be picking up on cues about visual realism, as demonstrated by its ability to rank image composites by their perceived realism, as measured by human subjects. For example, Figure 3.1 shows two composites which our model placed close to the decision boundary – these turn out to be composites which most of our human subjects thought were natural images. On the other hand, the composite far from the boundary is clearly seen by most as unrealistic. Given a large corpus of natural and composite training images, we show that our trained model is able to predict the degree of realism of a new image. We observe that our model mainly characterizes the visual realism in terms of color, lighting and texture compatibility.

We also demonstrate that our learned model can be used as a tool for creating better image composites automatically via simple color adjustment. Given a low-dimensional color mapping function, we directly optimize the visual realism score predicted by our CNN model. We show that this outperforms previous color adjustment methods on a large-scale human subjects study. We also demonstrate how our model can be used to choose an object from a category that best fits a given

background at a specific location.

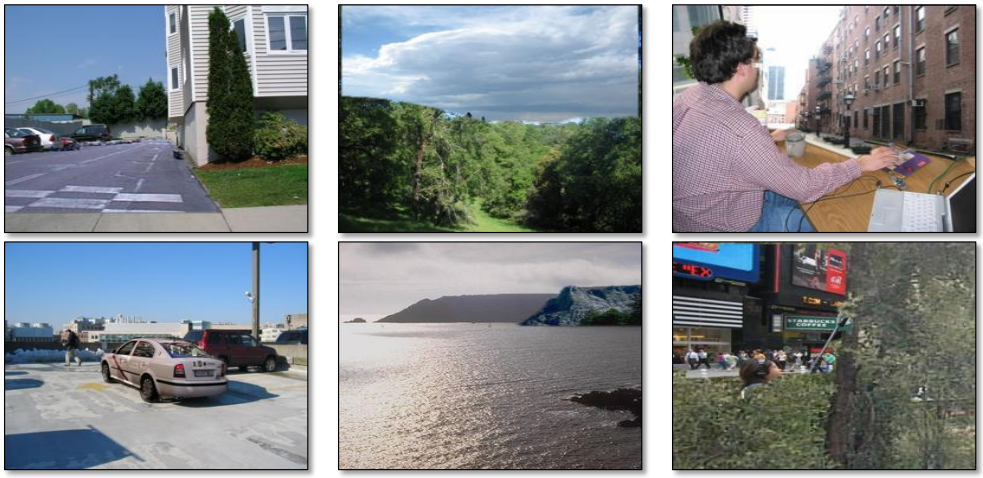
3.2 Background

Our work attempts to characterize properties of images that look realistic. This is closely related to the extensive literature on natural image statistics. Much of that work is based on generative models [52, 160, 251]. Learning a generative model for full images is challenging due to their high dimensionality, so these works focus on modeling local properties via filter responses and small patch-based representations. These models work well for low-level imaging tasks such as denoising and deblurring, but they are inadequate for capturing higher level visual information required for assessing photo realism.

Other methods take a discriminative approach [80, 135, 168, 181, 227]. These methods can generally attain better results than generative ones by carefully simulating examples labeled with the parameters of the data generation process (e.g. joint velocity, blur kernel, noise level, color transformation). Our approach is also discriminative, however, we generate the negative examples in a non-task-specific way and without recording the parameters of the process. Our intuition is that using large amounts of data leads to an emergent ability of the method to evaluate photo realism *from the data itself*.

In this work we demonstrate our method on the task of assessing realism of image composites. Traditional image compositing methods try to improve realism by suppressing artifacts that are specific to the compositing process. These include transition of colors from the foreground to the background [22, 156], color inconsistencies [119, 165, 167, 227], texture inconsistencies [34, 100], and suppressing "bleeding" artifacts [202]. Some work best when the foreground mask aligns tightly with the contours of the foreground object [119, 165, 167, 227], while others need the foreground mask to be rather loose and the two backgrounds not too cluttered or too dissimilar [34, 76, 120, 156, 202]. These methods show impressive visual results and some are used in popular image editing software like Adobe Photoshop, however they are based on hand-crafted heuristics and, more importantly, do not directly try to improve (or measure) the realism of their results. A recent work [201] explored the perceptual realism of outdoor composites but focused only on lighting direction inconsistencies.

The work most related to ours, and a departure point for our approach, is Lalonde and Efros [119] who study color compatibility in image composites. They too generate a dataset of image composites and attempt to rank them on the basis of visual realism. However, they use simple, hand-crafted color-histogram based features and



(a) Fully Supervised (b) Partially Supervised (c) Unsupervised

Figure 3.2: Example composite images for CNN training: (a) image composites generated by fully supervised foreground and background masks, (b) image composites generated by a hybrid ground truth mask and object proposal, (c) image composites generated by a fully unsupervised proposal system. See text for details. Best viewed in color.

do not do any learning.

Our method is also superficially related to work on digital image forensics [106,158] that try to detect digital image manipulation operations such as image warping, cloning, and compositing, which are not perceptible to the human observer. But, in fact, the goals of our work are entirely different: rather than detecting which of the realistic-looking images are fake, we want to predict which of the fake images will look realistic.

3.3 Learning the Perception of Realism

Our goal is developing a model that could predict whether or not a given image will be judged to be realistic by a human observer. However, training such a model directly would require a prohibitive amount of human-labeled data, since the negative (unrealistic) class is so vast. Instead, our idea is to train a model for a different "pretext" task, which is: 1) similar to the original task, but 2) can be trained with large amounts of unsupervised (free) data. The "pretext" task we propose is to discriminate between natural images and computer-generated image composites. A

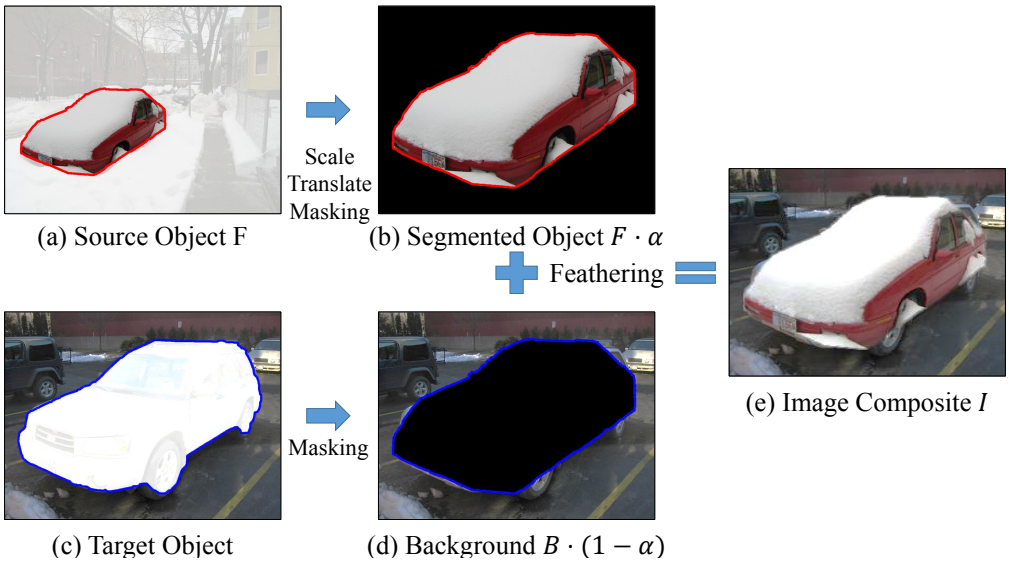


Figure 3.3: We generate a composite image by replacing the target object (c) by the source object F (a). We rescale and translate the source object to match the location and scale of the target object (c). We generate the final composite (e) by combining the segmented object (b) and the masked background (d).

high-capacity convolutional neural network (CNN) classifier is trained using only automatically-generated "free" labels (i.e. natural vs. generated). While this "pretext" task is different from the original task we wanted to solve (realistic vs. unrealistic), our experiments demonstrate that it performs surprisingly well on our manually-annotated test set (c.f. Section 3.6).

We use the network architecture of the recent VGG model [189], a 16-layer model with small 3×3 convolution filters. We initialize the weights on the ImageNet classification challenge [36] and then fine-tune on our binary classification task. We optimize the model using back-propagation with Stochastic Gradient Descent (SGD) using Caffe [97].

3.3.1 Automatically Generating Composites

To generate training data for the CNN model, we use the LabelMe image dataset [174] because it contains many categories along with detailed annotation for object segmentation. For each natural image in the LabelMe dataset, we generate a few composite images as follows.

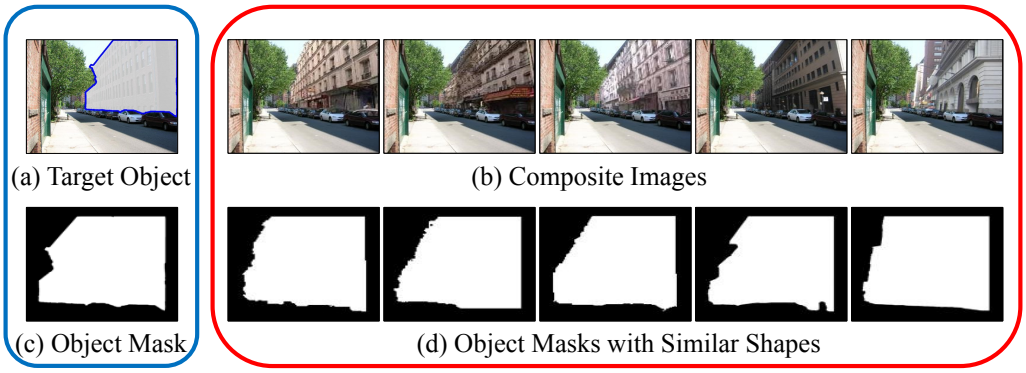


Figure 3.4: Given an original photo with target object (a) and its object mask (c), we search for source objects whose object mask matches well the shape of target object, and replace the target object with them. We show the nearest neighbor object masks in (d) and their corresponding generated composites (b).

Generate a Single Composite Figure 3.3 illustrates the process of generating a single composite image, which follows [119]. Starting with a background image B (Figure 3.3c) that contains an object of interest (target object), we locate a source object F (Figure 3.3a) with a similar shape elsewhere in the dataset, and then rescale and translate the source object F so that the source object matches the target location. (Figure 3.3b). We assume the object is well segmented and the alpha map α of the source object is known (Figure 3.3d). We apply a simple feathering based on a distance transform map to the object mask α of the source object. We generate the final composite by combining the source object and background $I = \alpha \cdot F + (1-\alpha) \cdot B$.

Generate Composite Dataset For each target object in each image, we search for source objects with similar shapes by computing the SSD of blurred and subsampled (64×64) object masks. Take Figure 3.4, for example. We replace the original building with other buildings with similar outlines. The purpose of the rough matching of object shape is to make sure that the generated composites are already close to the manifold of natural images. However, this procedure requires detailed segmentation annotations for both source and target objects. We call this procedure *FullySupervised* as it requires full annotation of object masks.

An alternative way is to use automatic image segmentation produced by an "object proposal" method (in our implementation we used Geodesic Object Proposals [114]). In this case, training images are still generated using human labeled segmentation for the target objects, but source objects are obtained by searching for object proposal

segments with similar shapes to the target objects in all images. This requires much fewer segmented training images. We name this procedure *PartiallySupervised*. The third way is fully automatic: we use object proposals for both source and target objects. In particular, we randomly sample an object proposal for a given image, and replace it by other object proposals with the most similar shapes from the dataset. This procedure is fully unsupervised and we call it *Unsupervised*. Later, we show that this fully automatic procedure only performs slightly worse than *FullySupervised* w.r.t human annotations, in terms of predicting visual realism (Section 3.6). We also experimented with randomly cutting and pasting objects from one image to the other without matching object masks. In this case, the CNN model we trained mainly picked up artifacts of high-frequency edges that appear in image composites and performed significantly worse. In our experiments, we used $\sim 11,000$ natural images containing $\sim 25,000$ object instances from the largest 15 categories of objects in the LabelMe dataset. For *FullySupervised* and *PartiallySupervised*, we generated a composite image for each annotated object in the image. For *Unsupervised*, we randomly sample a few object proposals as target objects, and generate a composite image for each of them.

Figure 3.2 shows some examples of image composites generated by all three methods. Notice that some composite images are artifact-free and appear quite realistic, which forces the CNN model to pick up not only the artifacts of the segmentation and blending algorithms, but also the compatibility between the visual content of the inserted object and its surrounding scene. Different from previous work [119], we do not manually remove any structurally inconsistent images. We find that composites generated by *FullySupervised* are usually correct with regards to semantics and geometry, but sometimes suffer from inconsistent lighting and color. *PartiallySupervised* also often generates meaningful scenes, but sometimes tends to paste an object into parts of another object. While *Unsupervised* tends to generate scenes with incorrect semantics, the number of scenes that can be generated is not restricted by the limited amount of human annotation.

Ranking of Training Images Interestingly, our trained CNN model is able to rank visually appealing image composites higher than unrealistic photos with visual artifacts. In Figure 3.5, we use our model to rank the training composites by their realism score prediction. The top row shows high-quality composites that are difficult for humans to spot while the bottom row shows poor composites due to incorrect segmentation and color inconsistency. We demonstrate that our model matches to human perception with quantitative experiments in Section 3.6.

(a) Most realistic composites ranked by our model

(b) Least realistic composites ranked by our model

Figure 3.5: Ranking of generated training composites in terms of realism scores. Best viewed in color.

3.4 Improving Image Composites

Let $f(I;\theta)$ be our trained CNN classifier model predicting the visual realism of an image I. We can use this classifier to guide an image compositing method to produce more realistic outputs. This optimization not only improves object composition, but also reveals many of the properties of our learned realism model.

We formulate the object composition process as $I_g = \alpha \cdot g(F) + (1-\alpha) \cdot B$ where F is the source object, B is the background scene, and $\alpha \in [0,1]$ is the alpha mask for the foreground object. For this task, we assume that the foreground object is well segmented and placed at a reasonable location. The color adjustment model $g(\cdot)$ adjusts the visual properties of the foreground to be compatible with the background image. Color plays an important role in the object composition process [119]. Even if an object fits well to the scene, the inconsistent lighting will destroy the illusion of realism.

The goal of a color adjustment is to optimize the adjustment model $g(\cdot)$, such that the resulting composite appears realistic. We express this in the following objective function:

$$E(g,F) = -f(I_g;\theta) + w \cdot E_{reg}(g), \tag{3.1}$$

where f measures the visual realism of the composite and E_{reg} imposes a regularizer

on the space of possible adjustments. A desired image composite should be realistic while staying true to identity of the original object (e.g. do not turn a white horse to be yellow). The weight w controls the relative importance between the two terms (we set it to $w = 50$ in all our experiments). We apply a very simple brightness and contrast model to the source object F for each channel independently. For each pixel we map the foreground color values $F^p = (c_1^p, c_2^p, c_3^p)$ to $g(F^p) = (\lambda_1 c_1^p + \beta_1, \lambda_2 c_2^p + \beta_2, \lambda_3 c_3^p + \beta_3)$. The regularization term for this model can be formulated as:

$$E_{reg}(g) = \frac{1}{N} \sum_p \Big(\|I_g^p - I_0^p\|_2 + \sum_{i,j} \|(\lambda_i - 1) \cdot c_i^p + \beta_i - (\lambda_j - 1) \cdot c_j^p - \beta_j\|_2 \Big) \tag{3.2}$$

where N is the number of foreground pixels in the image, and $I_0 = \alpha \cdot F + (1-\alpha) \cdot B$ is the composite image without recoloring, I_g^p and I_0^p denotes the color values for pixel p in the composite image. The first term penalizes large change between the original object and recolored object, and the second term discourages independent color channel variations (roughly hue change).

Note that the discriminative model θ has been trained and fixed during this optimization.

Optimizing Color Compatibility We would like to optimize color adjustment function $g^* = \arg\min_g E(g, F)$. Our objective (Equation 3.1) is differentiable, if the color adjustment function g is also differentiable. This allows us to optimize for color adjustment using gradient-descent.

To optimize the function, we decompose the gradient into $\frac{\partial E}{\partial g} = -\frac{\partial f(I_g, \theta)}{\partial I_g} \cdot \frac{\partial I_g}{\partial g} + \frac{\partial E_{reg}}{\partial g}$. Notice that $-\frac{\partial f(I_g, \theta)}{\partial I_g}$ can be computed through backpropagation of CNN model from the loss layer to the image layer while the other parts have a simple close form of gradient. See our online arXiv paper at https://arxiv.org/abs/1510.00477 for the gradient derivation. We optimize the cost function using L-BFGS-B [23]. Since the objective is non-convex, we start from multiple random initializations and output the solution with the minimal cost.

In Section 3.6.1, we compare our model to existing methods, and show that our method generates perceptually better composites. Although our color adjustment model is relatively simple, our learned CNN model provides guidance towards better color compatible composite.

Selecting Best-fitting Objects Imagine that a user would like to place a car on a street scene (e.g. as in [120]). Which car should she choose? We could choose an object $F^* = \arg\min_F E(g, F)$. For this, we essentially generate a composite image

for each candidate car instance and select the object with minimum cost function (Equation 3.1). We show our model can select more suitable objects for composition task in Section 3.6.2.

3.5 Implementation

CNN Training We used the VGG model [189] from the authors' website, which is trained on ImageNet [36]. We then fine-tune the VGG Net on our binary classification task (natural photos vs. composites). We optimize the CNN model using SGD. The learning rate α is initialized to 0.0001 and reduced by factor 0.1 after $10,000$ iterations. We set the learning rate for $fc8$ layer to be 10 times higher than the lower layers. The momentum is 0.9, the batch size 50, and the maximum number of iterations $25,000$.

Dataset Generation For annotated objects and object proposals in the LabelMe dataset [174], we only consider objects whose pixels occupy between $5\% \sim 50\%$ of image pixels. For human annotation, we exclude occluded objects whose object label strings contain the words "part", "occlude", "regions" and "crop".

3.6 Experiments

We first evaluate our trained CNN model in terms of classifying realistic photos vs. unrealistic ones.

Evaluation Dataset We use a public dataset of 719 images introduced by Lalonde and Efros [119], which comprises of 180 natural photographs, 359 unrealistic composites, and 180 realistic composites. The images were manually labeled by three human observers with normal color vision. All methods are evaluated on a binary realistic vs. unrealistic classification task with 359 unrealistic photos versus 360 realistic photos (which include natural images plus realistic composites). Our method assigns a visual realism score to each photo. Area under ROC curve is used to evaluate the classification performance. We call our method *RealismCNN*. Although trained on a different loss function (i.e. classifying natural photos vs. automatically generated image composites), with no human annotations for visual realism, our model outperforms previous methods that build on matching low-level visual statistics including color std/mean [165], color palette, texture and color histogram [119]. Notice that Lalonde and Efros [119] also requires a mask for the inserted object, making the task much easier, but less useful.

Methods without object mask	
Color Palette [119] (no mask)	0.61
VGG Net [189] + SVM	0.76
PlaceCNN [242] + SVM	0.75
AlexNet [116] + SVM	0.73
RealismCNN	0.84
RealismCNN + SVM	**0.88**
Human	0.91
Methods using object mask	
Reinhard *et al.* [165]	0.66
Lalonde and Efros [119] (with mask)	0.81

Table 3.1: Area under ROC curve comparing our method against previous methods [119,165]. Note that several methods take advantage of human annotation (object mask) as additional input while our method assumes no knowledge of the object mask.

Supervised Training Without any human annotation for visual realism, our model already outperforms previous methods. But it would be more interesting to see how our *RealismCNN* model improves with a small additional amount of human realism labeling. For this, we use the human annotation (realistic photos vs. unrealistic photos) provided by [119], and train a linear SVM classifier [27] on top of the $fc7$ layer's 4096 dimensional features extracted by our *RealismCNN* model, which is a common way to adapt a pre-trained deep model to a relatively small dataset. We call this *RealismCNN + SVM*. Figure 3.6 shows a few composites ranked with this model. In practice, $fc6$ and $fc7$ layers give similar performance, and higher compared to lower layers. We evaluate our SVM model using 10-fold cross-validation. This adaptation further improves the accuracy of visual realism prediction. As shown in Table 3.1, *RealismCNN + SVM* (0.88) outperforms existing methods by a large margin. We also compare our SVM model with other SVM models trained on convolutional activation features ($fc7$ layer) extracted from different CNN models including AlexNet [116] (0.75), PlaceCNN [242] (0.73) and original VGG Net [189] (0.76). As shown in Table 3.1, our *Realism + SVM* model reports much better results, which suggests that training a discriminative model using natural photos, and automatically generated image composites can help learn better feature representation for predicting visual realism.

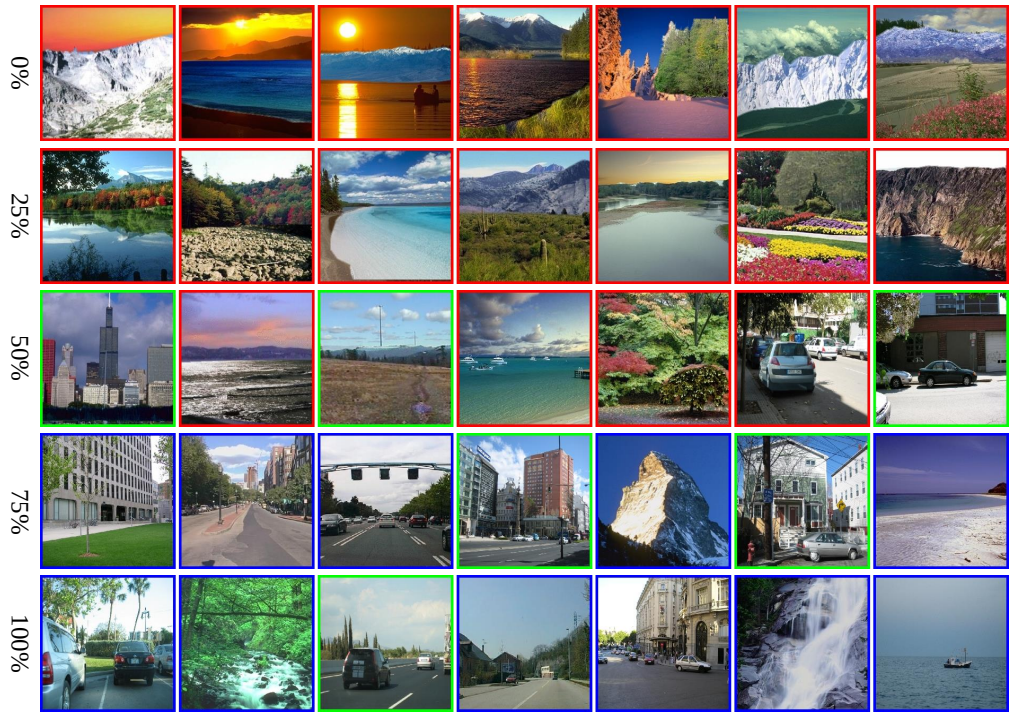


Figure 3.6: Ranking of photos according to our model's visual realism prediction. The color of image border encodes the human annotation: green: realistic composites; red: unrealistic composites; blue: natural photos. The different rows contain composites corresponding to different rank percentiles of scores predicted with *RealismCNN + SVM*.

Human Performance Judging an image as photo-realistic or not can be ambiguous even for humans. To measure the human performance on this task, we collected additional annotations for the 719 images in [119] using Amazon Mechanical Turk. We collected on average 13 annotations for each image by asking a simple question "Does this image look realistic?" and allowing the worker to choose one of four options: 1 (definitely unrealistic), 2 (probably unrealistic), 3 (probably realistic) and 4 (definitely realistic). We then average the scores of human response and compare the MT workers' ratings to the "ground truth" labels provided in the original dataset [119]. Humans achieve a score of 0.91 in terms of area under ROC curve, suggesting our model achieves performance that is close to level of human agreement on this dataset.

	RealismCNN	RealismCNN + SVM
FullySupervised	0.84	0.88
PartiallySupervised	0.79	0.84
Unsupervised	0.78	0.84

Table 3.2: Area under ROC curve comparing different dataset generation procedures. *FullySupervised* uses annotated objects for both source object and target object. *PartiallySupervised* uses annotated objects only for target object, but using object proposals for source object. *Unsupervised* uses object proposals for both cases.

Dataset Generation Procedure The CNN we reported so far was trained on the image composites generated by the *FullySupervised* procedure. In Table 3.2, we further compare the realism prediction performance when training with other procedures described in Section 3.3.1. We find that *FullySupervised RealismCNN* gives better results when no human realism labeling is available. With SVM supervised training (using human annotations), the margin between different dataset generation methods becomes smaller. This suggests that we can learn the feature representation using fully unsupervised data (without any masks), and improve it using small amounts of human rating annotations.

Indoor Scenes The Lalonde and Efros dataset [119] contains mainly photographs of natural outdoor environments. To complement this dataset, we construct a new dataset that contains 720 indoor photos with man-made objects from the LabelMe dataset. Similar to [119], our new dataset contains 180 natural photos, 180 realistic composites, and 360 unrealistic composites. To better model indoor scenes, we train our CNN model on $\sim$21,000 natural images (both indoor and outdoor) that contain $\sim$42,000 object instances from more than 200 categories of objects in the LabelMe dataset. We use MTurk to collect human labels for realistic and unrealistic composites (13 annotations per image). Without SVM training, our *RealismCNN* alone achieves 0.83 on the indoor dataset, which is consistent with our results on the Lalonde and Efros dataset.

3.6.1 Optimizing Color Compatibility

Generating a realistic composite is a challenging problem. Here we show how our model can recolor the object so that it better fits the background.

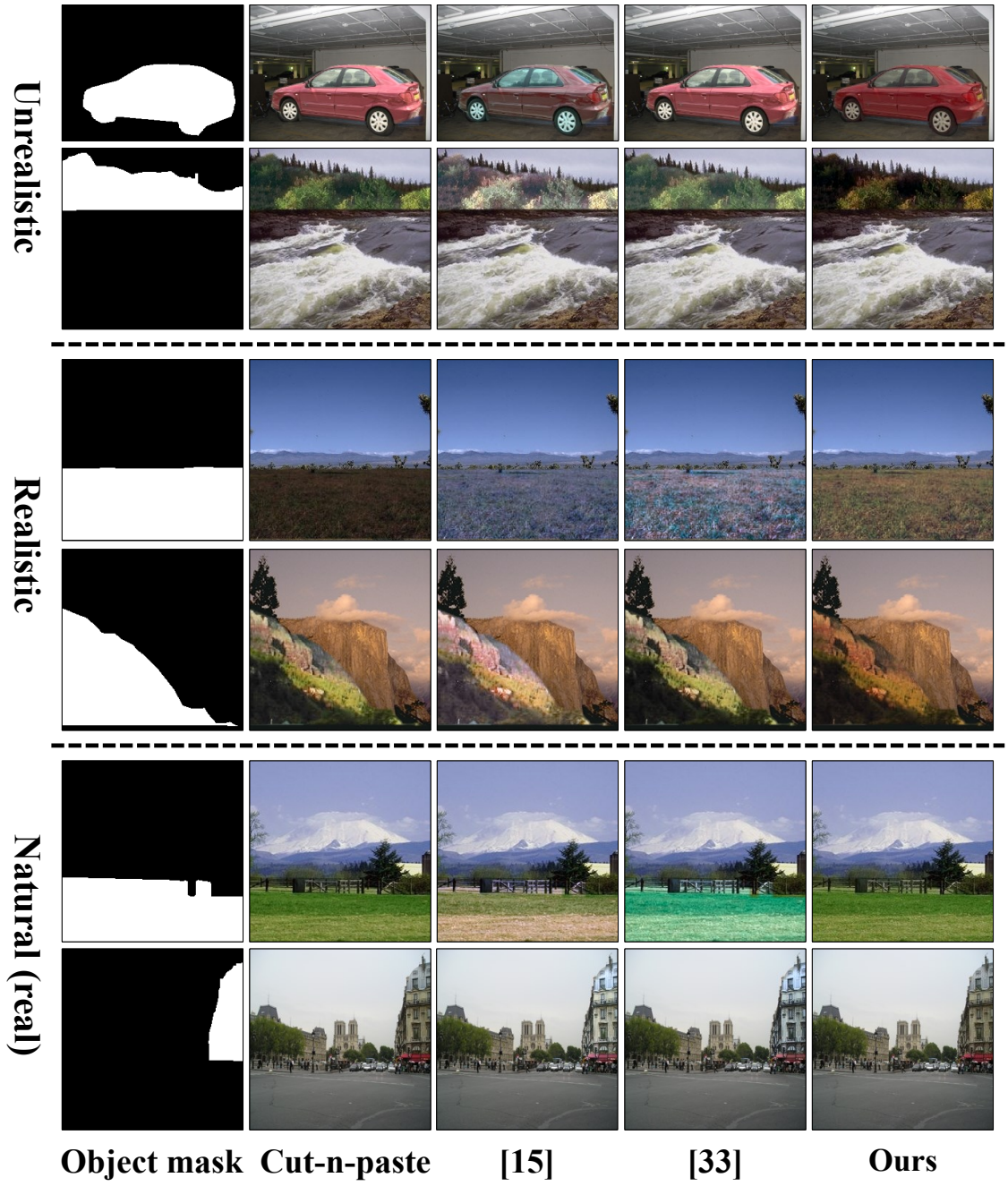


Figure 3.7: Example composite results: from left to right: objects mask, cut-and-paste, Lalonde and Efros [119], Xue et al. [227] and our method.

Dataset, Baselines and Evaluation We use the dataset from [119] that provides a foreground object, its mask, and a background image for each photo. Given an input, we recolor the foreground object using four methods: simple cut-and-paste, Lalonde and Efros [119], Xue et al. [227] and our color adjustment model described in Section 3.4. We use the *FullySupervised* version of *RealismCNN* model without SVM training. We follow the same evaluation setting as in [227] and use Amazon Mechanical Turk to collect pairwise comparisons between pairs of results (the question we ask is "Given two photos generated by two different methods, which photo looks more realistic?"). We collected in total 43140 pairwise annotations (10 annotations for each pair of methods for all 719 images). We use the Thurstone's Case V Model [205] to obtain a realism score for each method per image from the pairwise annotations, and normalize the scores so that their standard deviation for each image is 1. Finally, we compute the average scores over all the photos. We report these average human rating scores for three categories of images: unrealistic composites, realistic composites and natural photos. We use natural photos for sanity check since an ideal color adjustment algorithm should not modify the color distribution of an object in a natural photo. For natural photos, if no color adjustment is applied, the "cut-and-paste" result does not alter the original photo.

Results Table 3.3 compares different methods in terms of average human ratings. On average, our method outperforms other existing color adjustment methods. Our method significantly improves the visual realism of unrealistic photos. Interestingly, none of the methods can notably improve realistic composites although our model still performs best among the three color adjustment methods. Having a sense of visual realism informs our color adjustment model as to when, and how much, it should recolor the object. For both realistic composites and natural photos, our method typically does not change much the color distribution since these images are correctly predicted as already being quite realistic. On the other hand, the other two methods try to always match the low-level statistics between the foreground object and background, regardless of how realistic the photo is before recoloring. Figure 3.7 shows some example results.

Hard Negative Mining We observe that our color optimization method performs poorly for some images once we turn off the regularization term E_{reg}. (See Figure 3.8 for examples). We think this is because some of the resulting colors (without E_{reg}) never appear in any training data (positive or negative). To avoid this unsatisfactory property, we add newly generated color adjustment results as the negative data, and retrain the CNN with newly added data, similar to hard negative mining in object detection literature [53]. Then we use this new CNN model to recolor the

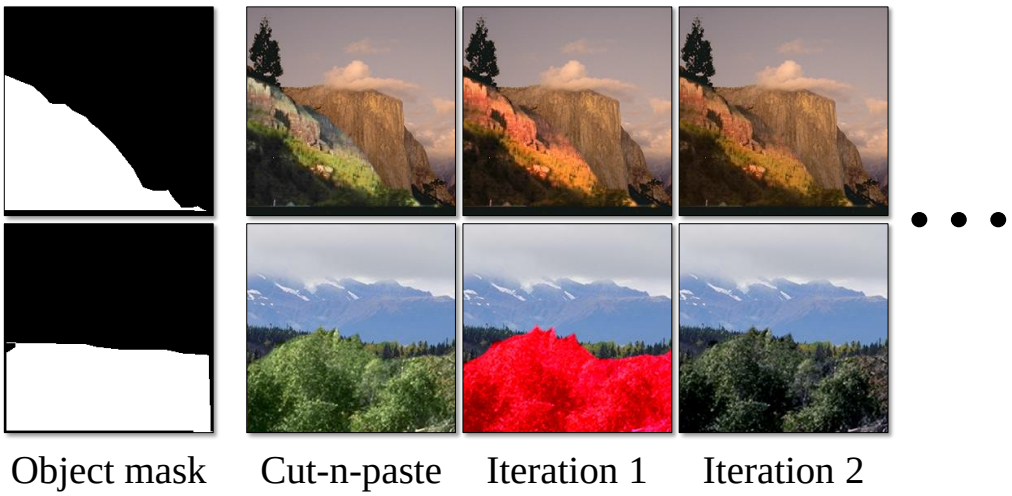


Figure 3.8: From left to right: object mask, cut-and-paste, results generated by $CNNIter1$ and $CNNIter2$ without the regularization term E_{reg}.

object again. We repeat this process three times, and obtain three CNN models named as $CNNIter1$, $CNNIter2$ and $CNNIter3$. We compare these three models (with E_{reg} added back) using the same MTurk experiment setup, and obtain the following results: $CNNIter1$: -0.162, $CNNIter2$: 0.045, and $CNNIter3$: 0.117. As shown in Figure 3.8, the hard negative mining avoids extreme coloring, and produces better results in general. We use $CNNIter3$ with E_{reg} to produce the final results in Table 3.3 and Figure 3.7.

3.6.2 Selecting Suitable Object

We can also use our *RealismCNN* model to select the best-fitting object from a database given a location and a background image. In particular, we generate multiple possible candidate composites for one category (e.g. a car) and use our model to select the most realistic one among them.

We randomly select 50 images from each of the 15 largest object categories in the LabelMe dataset and build a dataset of 750 background images. For each background photo, we generate 25 candidate composite images by finding 25 source objects (from all other objects in the same category) with the most similar shapes to the target object, as described in Section 3.3.1. Then the task is to pick the object that fits the background best. We select the foreground object using three methods: using *RealismCNN*, as described in Section 3.4; select the object with the most similar

(a) Best-fitting object selected by *RealismCNN*

(b) Object with most similar shape

(c) Random selected objects

Figure 3.9: For the same photo and the same location, we produce different composites using objects selected by three methods: (a) *RealismCNN*, (b) the object with the most similar shape, and (c) a randomly selected object.

shape (denoted *Shape*); and randomly select the object from 25 candidates (denoted *Random*).

We follow the same evaluation setting described in Section 3.6.1. We collect 22500 human annotations, and obtain the following average Human ratings: *RealismCNN*: 0.285, *Shape*: -0.033, and *Random*: -0.252. Figure 3.9 shows some example results for the different methods. Our method can suggest more suitable objects for the composition task.

	Unrealistic Composites	Realistic Composites	Natural Photos
cut-and-paste	-0.024	0.263	0.287
[119]	0.123	-0.299	-0.247
[227]	-0.410	-0.242	-0.237
ours	0.311	0.279	0.196

Table 3.3: Comparison of methods for improving composites by average human ratings. We use the authors' code to produce results for Lalonde and Efros [119] and Xue et al [227]. We follow the same evaluation setting as in [227] and obtain human ratings from pairwise comparisons using Thurstone's Case V Model [205].

3.7 Discussion

In this chapter, we present a learning approach for characterizing the space of natural images, using a large dataset of automatically created image composites. We show that our learned model can predict whether a given image composite will be perceived as realistic or not by a human observer. Our model can also guide automatic color adjustment and object selection for image compositing.

Many factors play a role in the perception of realism. While our learned model mainly picks up on purely visual cues such as color compatibility, lighting consistency, and segment compatibility, high-level scene cues (semantics, scene layout, perspective) are also important factors. Our current model is not capable of capturing these cues as we generate composites by replacing the object with an object from the same category and with a similar shape. Further investigation in these high-level cues will be required.

Part II

Generative Modeling for Visual Exploration and Synthesis

Chapter 4

Visual Exploration via Image Averaging

This chapter studies a quite simple and old image generation model: image averaging, pioneered by Sir Francis Galton [59] in 1878. The image averaging model constrains the output result to be a convex combination of the database images. Building on this simple model, we propose an interactive framework that allows a user to rapidly explore and visualize a large image collection using the medium of average images. Our interactive, real-time system provides a way to summarize large amounts of visual data by weighted averages of an image collection, with the weights reflecting user-indicated importance. We capture not only the mean of the distribution, but also a set of modes, discovered via interactive exploration. We pose this exploration in terms of a user interactively "editing" the average image using various types of strokes, brushes, and warps, with each user interaction providing a new constraint for updating the average. New weighted averages can be spawned and edited either individually or jointly. Together, these tools allow the user to simultaneously perform two fundamental operations on visual data, user-guided clustering and user-guided alignment, within the same framework. We show that our system is useful for various computer vision and graphics applications.

4.1 Introduction

The world is drowning in a data deluge [*The Economist*, Feb 25th 2010], and much of that data is visual. As mentioned in Chapter 1, an estimated 4.7 trillion photographs have been taken since the invention of photography, of which around 20% within the past 12 months.For example, Facebook alone reports 3 million photo

(a) 'Kids with Santa' © Jason Salavon

(b) result of Google query: "Kids with Santa"

(c) average image of (b)

(d) spectral clustering averages of (b)

(e) Results of our system: discovered modes (top) with example images and automatically propagated correspondences (bottom)

Figure 4.1: Average images, such as 'Kids with Santa' © Jason Salavon (a) are a creative way to visualize image data. However, attempts to replicate this technique with data automatically scraped from the Internet (b) does not lead to good results (c,d). In this work, we propose an interactive framework for discovering visually informative modes in the data and providing visual correspondences within each mode (e).

uploads per day, and YouTube sees 300 hours of video uploaded every single minute. Additionally, there is the data that hasn't made it onto the Internet (yet), such as the 24/7 video feeds from millions of surveillance cameras in convenience stores and ATMs. In fact, there is so much visual data out there already that much of it *might never be seen by a human being!* But unlike other types of "Big Data", such as text or consumer records, much of the visual content cannot be easily indexed, searched or hyperlinked, making it Internet's "digital dark matter" [157].

How can we explore this vast visual space, to see what's out there? One way is to anchor off the image meta-data (keywords, surrounding text, GPS, etc) as a proxy for indexing visual content. For example, typing "wedding kiss" into GOOGLE IMAGE SEARCH will return pages upon pages of photos that have somehow been associated with the words "wedding" and "kiss", most of them actually depicting formal kissing of various sorts (Figure 4.6a). While this semantic focusing vastly narrows down the

data (from `{all_photos}` down to `{all_wedding_kiss_photos}`), the resulting set is still far too vast to take in by mere visual inspection. How can we capture the visual *gestalt* of this data, its Platonic ideal?

The main inspiration and a departure point for this chapter is the recent surge in the use of data analytics and visualization techniques in contemporary art [210]. In particular, the simple technique of **image averaging** has been used extensively, and to great effect, by several well-known contemporary visual artists, such as Krzysztof Pruszkowski [161], Jason Salavon [176], James Campbell [25], and Idris Khan [109]. An average image of a set of photographs is obtained by simply computing the average color at each (x, y) pixel position independently across the set. This has the effect of capturing the overall similarities within the set while blurring out the individual differences[1]. For example, Figure 4.1a shows a piece by Salavon titled *'Kids with Santa'*, from his *100 Special Moments* series [176], which is an average computed over a hundred photos the artist manually picked from the Internet. Notice how, although the average is quite blurred, one can definitely decipher a figure dressed as Santa Claus, with another figure sitting on his knee – the individuality and uniqueness of each "special moment" usurped to tell a universal story.

Alas, we discovered that trying to replicate Salavon's averages automatically turns out to be quite difficult, e.g. simply downloading the top hundred images using a Google query "kids with Santa" (Figure 4.1b) and averaging them does not work well (Figure 4.1c). This is because the data is too varied (e.g. there are close-ups vs. long-range views, Santa could be in vastly different poses, or be missing altogether), and even images that depict the same type of scene (e.g. sitting Santa with kid on left knee) are not spatially aligned resulting in an extremely blurry average. Trying to reduce the data variability by first clustering the images using popular techniques, gives a slight improvement (Figure 4.1d), but nowhere close to Salavon's hand-picked average.

Of course, it is an artist's role to "actively guide analytical reasoning and encourage a contextualized reading of their subject matter" [210], but the only active guidance available in image averaging is the choice of which images to include – a blunt and inefficient instrument. What if we wanted to bring into focus different parts of the average image, such as Santa's face, or that of the kid? Antonio Torralba (a computer scientist who is also an accomplished artist) has been working on *centered average images* [203] where a dataset is first centered (aligned) on a particular object (e.g. a face, a spoon, etc) before the average is computed. Torralba's beautiful averages [203]

[1] In fact, this technique is almost as old photography itself, going back to Sir Francis Galton who, "having obtained photographs of several persons alike in most respects, but differing in minor detail", created "composite portraits" by "throwing faint images of the several portraits, in succession, upon the same sensitised photographic plate" as way of "extracting typical characteristics from them" [59].

contrast a sharper focal point with an eerie, dream-like background. Unfortunately, to achieve this effect requires that the object in focus be carefully hand-labeled in all images in the dataset before computing the average, making this approach completely impractical for large-scale data.

In this chapter, we propose an interactive framework that allows a user to rapidly explore and visualize a large image collection using average images. The idea is to summarize the visual data by *weighted* averages of an image collection, with the weights reflecting user-indicated image and feature importance. The aim is to capture not just the mean of the distribution, but a set of modes and projections (Figure 4.1e top), discovered via interactive exploration. The user interactively "edits" the average image using various types of brushes and warps, similar to a normal image editor, with each user interaction providing a new constraint to update the average. The user can also spawn and edit new weighted averages either individually or jointly, in which case an image that is weighted highly in one average will automatically be down-weighted in the others.

Alternatively, one can view the proposed framework as a new way to perform two fundamental operations on visual data: *user-guided clustering* and *user-guided alignment*. Automatic (i.e. unsupervised) image clustering and image alignment are, of course, two key problems in computer vision, both largely unsolved. By bringing the user into the loop, we demonstrate how to address these two problems *jointly*, within the same framework. And while our main driving application is in Big Visual Data exploration and improved artistic expression, we also demonstrate our framework to be of value in various other computer vision and graphics tasks.

4.2 Background

Our work builds on ideas from a number of different areas:

Image stacks: Given a stack of (typically) registered images of the same subject matter, methods such as Photomontage [1] offer various multi-image pixel operations on the stack, resulting in a combined "best" image. Our approach shares the idea of operating on an image stack, except our stack is 1) made up of semantically, but not necessarily visually similar data, 2) not typically well aligned, and 3) far too large for any manual per-image user guidance. Also related, a computer vision technique called *congealing* [86, 123] jointly aligns a stack of images of the same object category by iteratively bringing each image into closer alignment with the average. A recent extension [141] can even jointly align and cluster simple digit images. The *congealing* pipeline is fully automatic and while it works quite well for simple, unimodal categories (e.g. digits) and small mis-alignments, due to its

iterative nature, it often falls into local minima on more complex image data.

Image clustering and data mining: A standard way to model and visualize multi-modal data is by clustering. However, clustering is a highly under-constrained problem [11], so most clustering algorithms, such as k-means, spectral clustering [185], make strong distributional assumptions about the data and/or the distance metric, which often produce results that do not correspond to what is anticipated by the user. The present work can be thought of as a type of interactive clustefring [11] where the user can refine clustering results via interactive feedback. Other efforts aim to mine visual collections by finding a small number of important or "iconic" images [15, 188] or visual elements [41]. We differ in that our aim is to capture the *gestalt* of the data, not sample from it (although our system provides the latter as well). Unsupervised sub-category discovery approaches [40, 84] use discriminative clustering to find multiple modes for a given visual category. However, these methods do not provide local image alignment, resulting in poor clusters (see Sec. 6.6.2).

Data-assisted content editing and content creation: There has been recent interest in using large amounts of online visual data as content for computer graphics. For example, Hays and Efros [75] use millions of Flickr images as data for filling holes in a given scene, whereas Sketch2Photo [28] allows a user to generate new visual content by employing sketch and text queries to find and compose together content from existing photographs. There are also methods that use large amounts of data to provide artistic guidance to the user as part of the content creation process. Most related to our work is ShadowDraw [126], which helps users draw better by providing real-time average "shadow" suggestions of what to draw next by matching what has already been drawn against a large database of existing imagery. Our sketching brush tool is very much inspired by ShadowDraw, but whereas ShadowDraw is fundamentally a reactive, bottom-up process – first asking the user to draw something and only then providing suggestions, our system aims to be top-down – first giving the user a sense of what is in the data, and only then asking him to refine it. More importantly, the target goals are fundamentally different: while ShadowDraw aims to help the user in drawing, our AverageExplorer aims to help the user in exploring and aligning large image collections, while also facilitating a number of other applications (Sec. 6.6.2). Inspired by successful content-based image retrieval systems like Fast Multiresolution Image Querying [94] and BlobWorld [26], our system also takes paint strokes and user-specified regions as input (Sec. 4.3.3).

Exploratory data visualization: There is a large body of work on exploring and mining Big Data in a visual way (see [35] for a survey), but the vast majority is on visualizing *non-visual* data, which is rather different from the problem we are trying to address. Several works show beautiful ways of visualizing a *specific type* of visual data, such as photos of the same location [193], or of the same person [108]. Of

the few attempts to visualize generic, large-scale visual data, the most related is the "Visual Dictionary of Tiny Images" [204], that aims to provide a visual summary of 80 million images using an atlas of 53464 tiny average image tiles, each corresponding to one English noun. While the online demo is fun to explore, semantic concepts (words) often do not correspond to coherent visual concepts, making many of the average images noisy and uninformative. In this work, we only use semantics (e.g. keyword tags) as an initialization, and then let the user interactively discover visual concepts hidden in the data.

4.3 Approach

AverageExplorer is a real-time interactive system that allows the user to easily explore and navigate a large image collection through the manipulation of *average image(s)*. The input is a (potentially very large) collection of images, typically representing the same semantic concept ("cats", "shoes", "Paris", etc.) but with wide variation in appearance, e.g. Internet images retrieved using a search engine. The output is a set of average images that depict different modes in the data, as well as feature correspondences between images within each mode. The user is provided with a set of brush tools to iteratively "edit" each average image. The objective is to summarize the image collection with weighted average(s) of images, in which the weights reflect the user's suggested importance.

AverageExplorer has five main components: (1) the user interface, which displays one or more average images that reveal different modes in the data; (2) a method to generate/update an average image, which continuously takes the user's edits and re-ranks the database images accordingly; (3) a set of brush tools (explorer, coloring, and sketching), which the user applies to the average image to denote what she deems important; (4) cluster spawning, which dynamically creates a new cluster at anytime for simultaneous exploration of multiple average images; and (5) image alignment, which automatically warps each image in the dataset to better align it with the user-specified constraints.

4.3.1 User interface

The AverageExplorer interface is composed of the current average image, a button for each brush tool, a button to generate a new cluster, and a retrieval display of the top most similar (i.e. highest-weighted) images to the current average. The average image and retrieved images are updated in real-time as the user continuously provides edits. When the user makes an edit, it is highlighted on both the average image and in each of the retrieved images. If more than one average image is being edited,

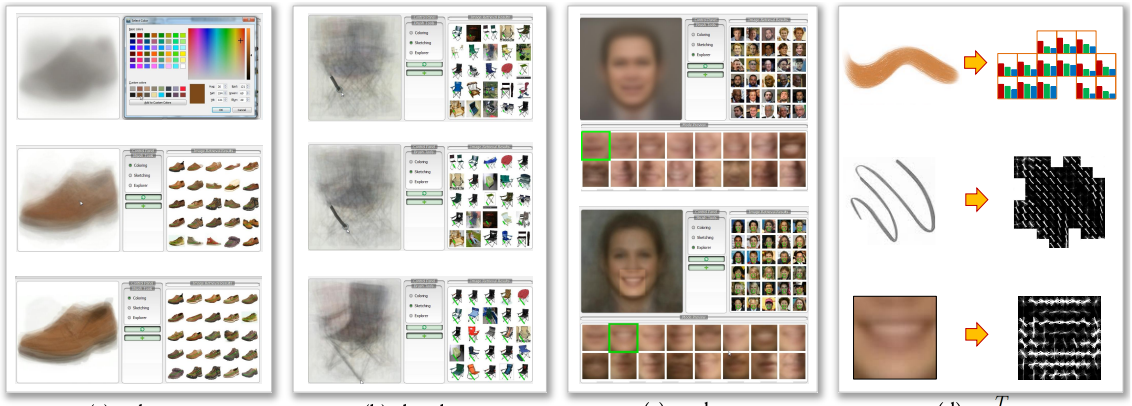


Figure 4.2: We propose a set of brush tools that can be used to edit the average image to interactively explore the data. Each user interaction provides a new constraint to update the average.

the user can switch focus between them by pressing the 'tab' key or clicking on the corresponding cluster with the mouse cursor (see video).

4.3.2 Generating the average image

Given a database of N images $\{I_1, \ldots, I_N\}$, we continuously update its average image in real-time as the user interacts with the system. We create the average image I_{avg} by computing a weighted average of the database images that reflects the score (i.e. weight) s_i of each image (for now, we assume that the images are spatially aligned; in Sec. 4.3.5, we will relax this assumption):

$$I_{avg} = \frac{\sum_{i=1}^{N} s_i \cdot I_i}{\sum_{i=1}^{N} s_i}. \tag{4.1}$$

We initialize $s_i = 1/N, \forall i$ so, we start with I_{avg} being a simple pixel-wise mean of the entire image collection. Once editing begins, the score s_i for each image I_i is updated to reflect how well that image matches the user's edits, and is computed cumulatively:

$$s_i = \sum_{t=1}^{T} match(\mathbf{w}^t_{user}, I_i), \tag{4.2}$$

where s_i is the cumulative score of image i after T edits, $\mathbf{w}^t_{user}$ represents the user edit at time t, and $match(\cdot)$ returns how similar a given image I_i is to the user edit $\mathbf{w}^t_{user}$. For now, we will define $match(\mathbf{w}^t_{user}, I_i) = \mathbf{w}^t_{user} \cdot \phi(I_i)$, i.e. the dot-product between

the user edit $\mathbf{w}^t_{user}$ and image I_i in some feature space defined by $\phi(\cdot)$. (the exact representation of $\mathbf{w}^t_{user}$ and $\phi(\cdot)$ depends on the type of user edit and will be defined in Sec. 4.3.3). Intuitively, each user edit tells the system which visual patterns should be present (or emphasized) within the spatial region where the edit has occurred. An image that has similar visual patterns as the user's edit will produce a high $match(\cdot)$ value, while an image that has dissimilar visual patterns will produce a low $match(\cdot)$ value. Because the image scores are computed cumulatively, we only need to update the score to reflect the latest edit, which allows our system to update the average image very quickly and smoothly from one edit to the next.

To reduce the effect of noisy matches produced when editing has just started (e.g. with a single edit, there can be a few matches that agree extremely well at the local region level, but not at the global image level), we apply the following nonlinear function (also used by [126] for a similar purpose) to each image score: $s^*_i = \max(0, s_i - \alpha \cdot \bar{s})^\gamma$, where s^*_i is the updated image score, $\bar{s}$ is the average of the top K image scores, $\alpha = 0.2 + 0.05 \cdot T$, $\gamma = 0.1 \cdot T$, and $K = 20$. As α increases, fewer images will have score greater than 0, and as γ increases, the distribution of those positive scores will become more peaked. The combined effect makes the average image blurry initially and sharper over time (i.e. as T increases).

4.3.3 Brush tools

AverageExplorer provides three brush tools to navigate the data: coloring, sketching, and explorer. Each tool allows the user to dynamically update the average image. After each edit, the weight of each database image is changed according to how similar it is to all of the user's edits provided thus far.

Coloring Brush

The coloring brush allows the user to paint on the average image by adding color strokes. The user chooses a color (mouse right button) from either a standard color palette or a "data-driven palette" which contains the most common colors for the region currently under the brush across the dataset. The user can adjust the size of the brush by scrolling the mouse wheel, and color by holding down the left mouse button. This tool is most useful when the user wants to constrain the color of a specific spatial region (e.g. to specify the color of a person's hair or eyes). We encode the user's color stroke at the current iteration T as $\mathbf{w}^T_{user} = H_c$, which is a normalized, 5-dimensional (x,y,R,G,B) histogram containing a uniformly-sampled 4x4x4 RGB histogram within each 8x8 pixel block of the stroke (see Figure 4.2d). Each database image I_i is encoded in the same way, $\phi(I_i) = H_{c,I_i}$; a normalized 5-dimensional (x,y,R,G,B) histogram computed in the same spatial region as the user's color stroke.

$match(\mathbf{w}^t_{user}, I_i) = H_c \cdot H_{c,I_i}$, a dot-product between the two histograms, encoding the degree of their similarity.

As the user paints, the average image is updated dynamically at 30 fps. Figure 4.2(a) shows an example usage: a user selects the brown color, clicks on the center of the average image and starts painting, giving high weight to the brown images in the dataset, which changes the average image accordingly.

Sketching Brush

The sketching brush allows the user to add line strokes to the average image. The user can choose the size of the brush by scrolling the mouse wheel, and sketch by holding down the left mouse button. It is most useful for adding fine details (e.g. outlining the shape of the chin or drawing glasses when exploring faces). We encode the user's sketch at the current iteration T as $\mathbf{w}^T_{user} = H_g$, which is a histogram with spatial and orientation bins encoding the gradients under the stroke region. We use the standard Histogram of Gradients (HOG) representation [32] with 8x8 pixel spatial bins (see Figure 4.2d). Each database image I_i is encoded in the same way, $\phi(I_i) = H_{g,I_i}$; a HOG feature computed in the same spatial region as the user's line stroke. Thus, $match(\mathbf{w}^t_{user}, I_i) = H_g \cdot H_{g,I_i}$, a dot-product between the two histograms, encoding the degree of their similarity.

As with color brush, as the user sketches, we dynamically update the average image at 30 fps. Figure 4.2(b) shows an example usage: the user sketches a diagonal stroke to denote a particular chair leg shape, which gives high weight to the database chair images that have similar shaped legs and updates the average image and the top retrieved images accordingly.

Explorer Brush

The coloring and sketching brushes are useful to emphasize and sharpen the features that are already visible in the average image. However, if the user wants to explore information that may be hidden in the data and not immediately visible through the average, she is essentially limited to guessing the correct stroke. The explorer brush attempts to overcome this limitation; it is thus our most important tool. The main idea is to collect local patches situated in the same spatial position across all database images, and cluster them into a set of visually-informative modes. As part of the explorer tool, the user can pick a single mode, and see the global average computed using only the images that are assigned to (conditioned on) that mode, as illustrated on Figure 4.3. This give the user a local tool to interactively explore the different components that make up an average image.

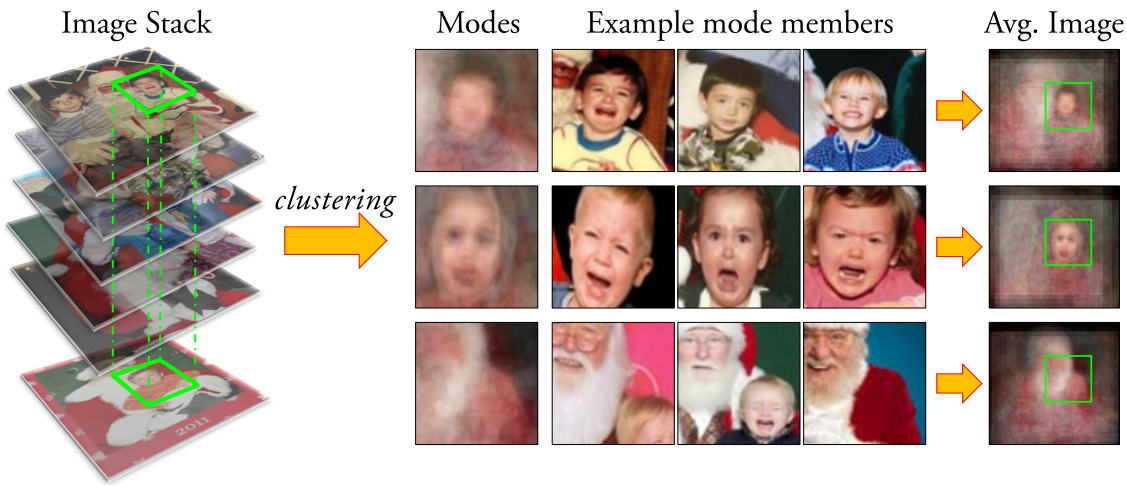


Figure 4.3: Our explorer brush groups local image patches (shown in green bounding boxes) from the same rough spatial position across all database images. Then, for each group, our system averages the full images assigned to the group to create an average image.

Specifically, given a mouse cursor position, we find the dominant local modes (groups) in the image stack data for that rough spatial location. To find the modes, we adapt the mid-level discriminative patch discovery approach of [190]. It mines mid-level visual patterns that are frequently-occurring but also discriminative (sufficiently different from the rest of the "natural visual world"). We first sample a thousand "seed" patches centered at the mouse cursor position from a random subset of the database images. For each seed, we compute distances to patches from (roughly) the same spatial position in all remaining database images, and also compute distances to random patches in random images (downloaded from Flickr), which represent the "natural visual world". For each potential group (seed patch and its k nearest neighbors) we compute the inlier score u, which is the ratio of the database images (inliers) to the Flickr images (outliers) in the k nearest neighbor patches ($k = 50$ in our experiments). The inlier score measures the uniqueness of the nearest neighbors. A group that includes many Flickr patches will not be unique as it will capture common visual patterns found in the natural world, whereas a group comprised mostly of patches from the database will capture unique visual patterns, and thus be good for matching and alignment (see Figure 4.3; the top-ranked modes are discriminative and lead to accurate matches).

We then rank each group of seed patch p_j and nearest neighbors $\{d_1, \ldots, d_N\}$ by: $(\sum_{m=1}^{N} s_m \cdot z(d_m)) \cdot u_j$, where N is the number of detections (one per image), s_m is

the score of the image containing patch d_m, $z(d_m)$ is the similarity score between p_j and d_m, and u_j is the inlier score. This scoring function will assign high rank to unique groups whose nearest neighbors match well to the seed patch *and* come from highly weighted images (due to previous edits). This reinforces the gradual change of the average image, since the top suggestions (i.e. highest ranked groups) are likely to come from images that contributed highly to the previous average image (see next paragraph for details on how to choose between different groups). We retain groups that have inlier score u greater than 0.75, and remove near-duplicate groups that have spatial overlap of more than 25% between any 30 patches of their members. This typically results in 10-50 groups that represent the main local modes of the data at that spatial position.

The user interaction proceeds as follows: As the mouse hovers over a particular part of the average image, the top-ranked local modes are displayed below the average image, as average local patches (Figure 4.2c). The user can interactively change the size of the local patches by scrolling the mouse wheel. By default, the main panel displays the average image according to the top-ranked local mode. But the user can explore the average images of the lower-ranked modes by pressing the 'tab' key, which shifts down to the next mode (Figure 4.2c). Once the user finds an interesting position on the average image and picks the preferred local mode, she can use this mode as an extra constraint on the average image by pressing the left mouse button. Specifically, we encode the user's mode constraint at the current iteration T as $\mathbf{w}_{user}^T = H_e$, which is a histogram of the gradients in the seed patch of the given mode, again using HOG. Each database image I_i is encoded in the same way, $\phi(I_i) = H_{e,I_i}$; a HOG feature computed in the same spatial region as the selected mode. Thus, $match(\mathbf{w}_{user}^t, I_i) = H_e \cdot H_{e,I_i}$, a dot-product between the user edit's and database image's HOGs. This has the effect of constraining the average image to give more weight to the images from that local mode.

Real-time speed-up: Unlike the previous tools, the computation required for the explorer tool is too expensive to run in real time due to large amount of patch nearest neighbor searching across the entire dataset, the discriminative patch mining, and the fact that the user wants to explore the space rapidly. To achieve real-time performance, we are forced to pre-compute the matches offline. Specifically, we first sample seed patches on a dense regular grid (4-pixel stride) at multiple scales (32x32, 48x48, 80x80, 128x128 pixel patches) for the entire image collection. We then compute distances (dot-product in HOG space) to all patches within a 2x length (64 to 256 pixels) region surrounding the seed patch in all remaining database images. For each seed, we store the top matching patch per image with match score greater than 0.5, and record both the match score as well as the position and scale of the matched patch. During online processing, for each seed in the current mouse cursor position,

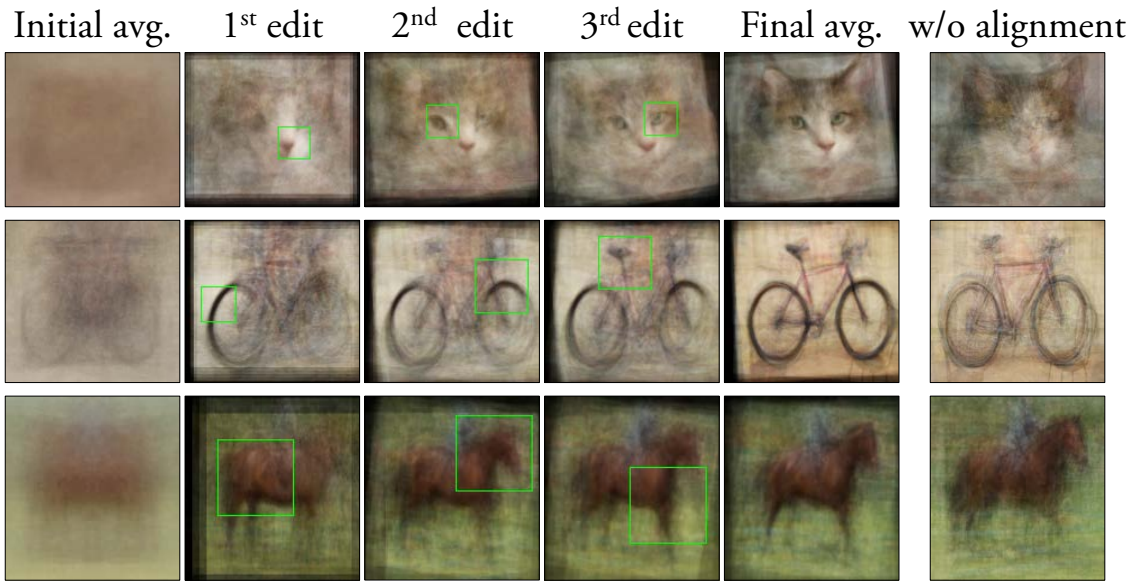


Figure 4.4: We align each database image to produce a sharper average image. Notice how just clustering (without alignment) is insufficient to produce a sharp average (right-most column).

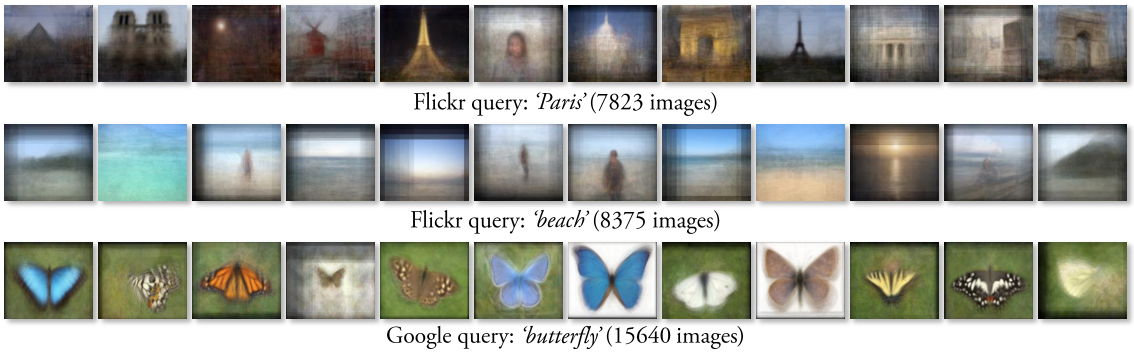


Figure 4.5: Examples of interactively discovered modes in the data using AverageExplorer.

our system selects the top-matching patch in each database image to create the average image.

Figure 4.2d shows example usage: given a database of face images initially with uniform weight (top), the user explores different types of mouths (bottom). The user has chosen the second mode, which changes the average image accordingly.

4.3.4 Interactive Clustering

At any time during data exploration, the user can spawn a new cluster. This is particularly useful when the user wants to explore multiple modes in the data at the same time. For example, when exploring human faces, the user might want to separate, say, people with oval faces from people with rounder faces. To this end, we present a tool for interactive clustering, in which a user's edit on one average image influences the average images of the other clusters (i.e. modes). The goal is to simultaneously produce sharper average images for all clusters, which means that each cluster should consist of images that are similar to each other and dissimilar to images in other clusters. E.g. if the user edits an average image of a face to be rounder, then the other average images should become oval-shaped.

To simultaneously update all clusters with each edit, we compute a cluster-specific weight for each image. The average image for each cluster is created as in Sec. 4.3.2, but the weight of each image can now be different for each cluster. We normalize the cluster-specific weights in such a way that an image cannot contribute highly to all clusters. Specifically, we initialize a new cluster c by assigning each image with uniform weight $s_{i,c} = 1/N, \forall i$. We then normalize the weights such that the total cluster-specific weights of an image sum to one: i.e. $\sum_j s_{i,j} = 1$, where j indexes the clusters. In effect, the normalization limits high contribution of an image to one or a few clusters, which in turn makes each cluster tighter. To give a simple example, if there are two clusters, and the user colors one average image white, then the second average image will become black, since all the highest weighted images for the first cluster will be white, while the highest weighted images for the second cluster will be the opposite (i.e. black).

4.3.5 Image Alignment

Thus far, we have assumed that the semantic concepts ("cats", "shoes", "Paris", etc) depicted in each database image will be spatially aligned. In practice, this will rarely be the case (even within the same mode), especially when working with Internet images retrieved using a search engine. If the database images are not spatially well aligned, the resulting average will be blurry, which, in turn, will lead to meaningless user edits. AverageExplorer provides a two-step solution to mitigate this problem: robustness to mis-alignment and image warping.

First, we update the matching function so that it is robust to (some) spatial misalignment between the database images using max-pooling [16]:

$$match(\mathbf{w}^t_{user}, I_i) = \max_{p \in \mathcal{P}} \mathbf{w}^t_{user} \cdot \phi_p(I_i), \tag{4.3}$$

where p indexes over the possible x, y locations in $\mathcal{P}$, and $\phi_p(I_i)$ is the feature space representation of image I_i computed over the spatial location defined by p. We set $\mathcal{P}$ to be the locations that cover up to 64 pixels in both x, y directions surrounding the user's edit.

We then apply non-linear warping to align the database images. For this, we use Moving Least Squares (MLS) [180], which provides a simple closed-form solution that yields fast deformations for real-time performance. MLS estimates a deformation function that maps a set of source control points to a set of target control points. We use the center of mass of each user edit in the average image as a target control point and the center of mass of the corresponding matching region in the database image as a source control point. We then warp the database image such that its control points (from all T edits thus far) align to the corresponding control points in the average image. The deformation function is applied to every pixel in the database image (see [180] for more details). Note that the warping does not affect the matching function defined in Eqn. 4.3; it is used only to align the images more accurately to form a sharper average.

We compute the average image with the warped images I_i^{MLS}:

$$I_{avg} = \frac{\sum_{i=1}^{N} s_i \cdot I_i^{MLS}}{\sum_{i=1}^{N} s_i}. \tag{4.4}$$

Our iterative, user-guided process leads to image stacks that have increasingly better image alignment, producing not only sharper average images (Figure 4.4), but also surprisingly high-quality local feature correspondences (Figure 4.1e bottom), which can be useful for rapid data annotation (Sec. 6.6.2, "Image annotation").

Finally, after each user edit, we update each database image with its warped version. For non-linear warping, we need to recompute the database image features (HOG, color histograms, and discriminative patches), which is prohibitively expensive. Therefore, we only translate each image according to the mean offset between the source and target control points.

4.4 Results and Applications

We demonstrate how our system can be used to explore and visualize large image collections, and show several potential applications.

Datasets: We experimented with a variety of visual image collections from multiple sources. They are: Labeled Faces in the Wild (LFW) [87], which has 13233 images with 5749 different people; *Cat* database [239] (10000 images); Query-based collections downloaded from Google Images and Flickr using the following keywords:

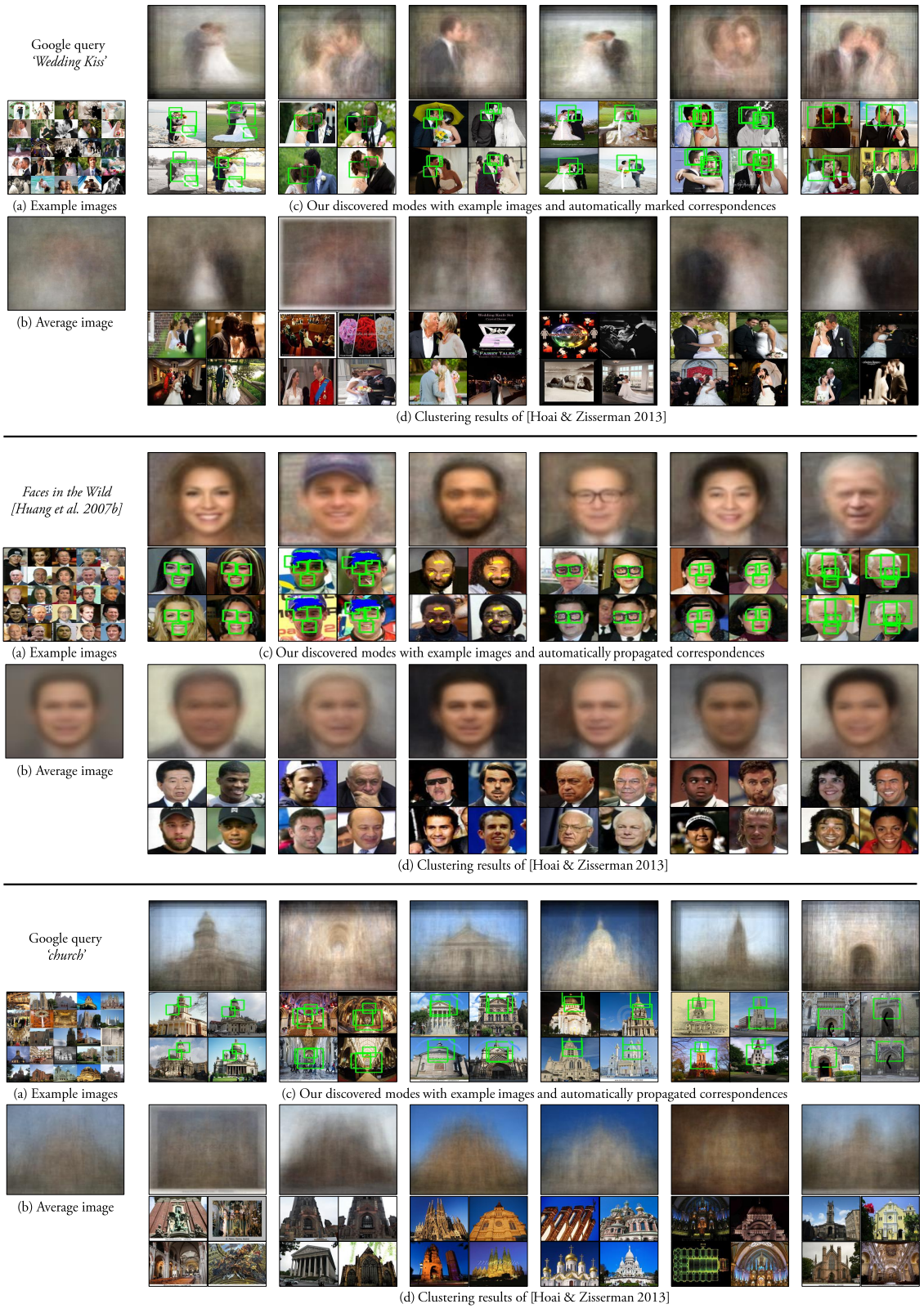


Figure 4.6: *Interactive exploration and alignment.* For each dataset we show (a) example images, (b) global average image, (c) our discovered modes with 4 top retrieved images and automatically marked correspondences (an experienced user spent on average 124 seconds discovering each mode using on average 3.4 user edits)

(a) [Divvala et al. 2012] modes (b) Our discovered modes

Figure 4.7: We compare the averages generated by the visual subcategory learning approach of [40] (a) to our averages (b) using the PASCAL 2007 horse dataset. See text for more details.

'Church' (11007 images), *'Paris'* (7823 images), *'Butterfly'* (15640 images), *'Beach'* (8375 images), *'Wedding kiss'* (16868 images), and *'Kids with Santa'* (1640 images); YouTube videos: 50 clips of 1 minute summary of the *Cobert Report*, sampled at 2 fps (5232 frames total); and 362 PASCAL 2007 *Horse* images.

Implementation details: We resize each image to the average size of its dataset. To compute $\mathbf{w}_{user}$ for coloring and sketching we first create a tight bounding box surrounding the user's stroke or selected region. We then compute the color or HOG histograms over 8x8 spatial bins, and zero-out any cells that do not spatially overlap with the users' stroke/region. We whiten the HOG descriptors, as described in [74], which makes dot product similarity computations more visually meaningful. When computing the average image, we ignore any pixels in the warped images that fall outside of the dimensions of the average image. We run our system in real-time on a PC with Intel i7-4770K processor (3.50GHz, 4 cores) and 16GB RAM. For a 10000 image dataset, the pre-computation processing of features and discriminative patch discovery can be done on a 150-core cluster in about 10 hours. Our system is publicly available on our project page.

Interactive exploration and alignment: We first show how AverageExplorer can help to uncover meaningful patterns in visual data. We compare against the standard global average of the dataset, as well as several baselines: 1) k-means clustering, 2) spectral clustering [185], and 3) discriminative sub-category discovery algorithm of [84], which uses negative images (irrelevant to the category at hand) to learn a better distance metric for clustering. For all baselines, we represent each image using the standard HOG descriptor (8x8 pixel cells) concatenated with a tiny color image (original RGB image resized by 1/8 in width and height) to spatially encode gradient and color information. For [84], we generate a weighted average

	M=3	M=6	M=12	M=24	mean
Random Accuracy (%)	55.3	64.9	67.1	64.4	62.9
Manual Accuracy (%)	66.0	65.2	**72.6**	**68.6**	68.1
Ours Accuracy (%)	**67.4**	**68.3**	71.0	67.8	**68.6**
Manual Time (minutes)	24	28	39	54	36.25
Ours Time (minutes)	6	9	15	28	14.5

Table 4.1: *User study.* Human selection accuracy and timing results with M representative images. See text for details.

image for each cluster, where each cluster instance is weighted according to the score that indicates how representative it is to that cluster (see [84] for more details). For k-means and spectral clustering, we create weighted averages by weighting each image by its total intra-cluster affinity.

Figure 4.6 shows results of our system on 3 sample datasets: *'Wedding Kiss'*, *Faces*, and *'Church'*. In each case, we show a global average image, six of our discovered modes with four top retrieved images each (showing correspondences) and results of [84] (see project web page for more detailed comparisons, including those to k-means and spectral clustering). As discussed earlier, the global average image can only summarize the data coarsely, displaying only the rough global shape, since it weighs all pixels equally. Standard clustering techniques, e.g. k-means and spectral clustering, try to represent different modes in the data, but often focus on the non-important parts of the image (e.g. see k-means cluster faces based on their background color in Figure 3 of the supp. material). [84] does better, as it learns a better distance metric to discover sub-clusters within each category. For example, for faces, the representative cluster instances are mostly consistent visually. However, it is not able to cope with misalignment, which limits matching accuracy and resulting in blurry averages. Our averages, on the other hand, are sharp and clearly depict key modes in the data. For example, we can discover and align faces of people wearing hats, beards, and glasses. This is possible due to our system finding accurate correspondences between visually-similar features and providing real-time feedback to the user for interactive refinement of the average image. Notice how our approach allows us to discover visual modes in the data that might have otherwise gone unnoticed, such as a gay wedding kiss (the rightmost mode in the top figure). We also conducted a simple study to determine which of the average representations were preferred by the users. The results show overwhelming preference for AverageExplorer results against all other baselines. Figure 4.5 shows more examples of the modes discovered by our system.

We also compare our method to the visual subcategory learning approach of [40], which is similar to [84] but also performs global alignment (translation and scaling) for more accurate clustering. In this setting, we simultaneously edit 15 clusters.

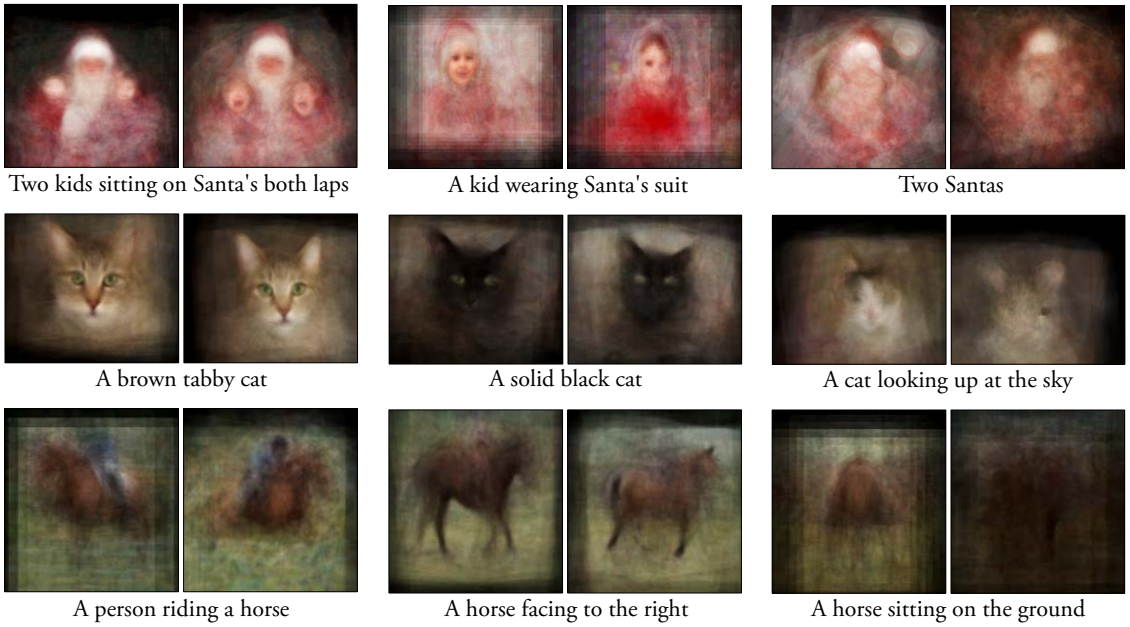


Figure 4.8: *Average images created by users given a text query.* The last two columns show failure cases. The users produced blurry and distorted averages for 'difficult' queries that have insufficient data.

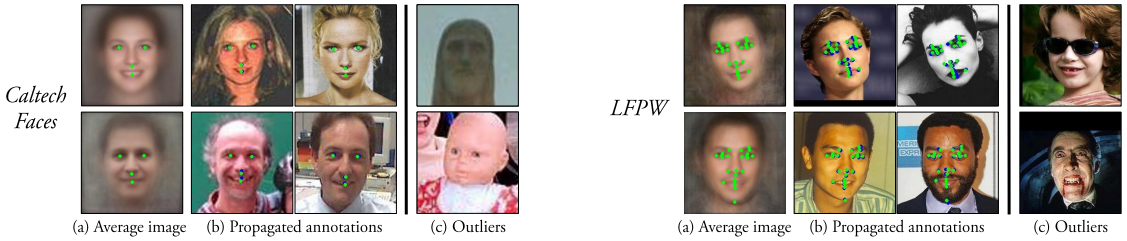


Figure 4.9: *Qualitative keypoint annotation results:* We show examples of the annotated average image (a), and corresponding propagated annotations to cluster instances (b) for Caltech Faces dataset [6] (left) and LFPW dataset [13] (right). The blue points are human-marked annotations and green points are our annotations. The outlier images in (c) were not assigned to any cluster. They correspond to atypical face images, such as a baby doll or Dracula (sharp teeth and heavy make-up).

Figure 4.7 shows the average images for the 15 clusters discovered from the PASCAL 2007 *Horse* category by the baseline (a) and our approach (b). Note that both ours and the baseline operate on the human-annotated horse bounding box region within each image. Even so, the object regions are only coarsely aligned and contain many different modes. Both methods are able to discover a diverse set of modes. However,

	detected faces	annotated clusters	annotated images	annotated points
BioID	1506	27 (∼55 imgs/cluster)	98.6%	93.2%
LFPW	798	22 (∼32 imgs/cluster)	88.2%	71.1%
Caltech Faces	6476	58 (∼101 imgs/cluster)	90.8%	71.0%

Table 4.2: *Keypoint annotation statistics.* Since AverageExplorer accurately aligns images of the same mode, it can be used to efficiently propagate keypoint annotations. For example, given 6476 face images from the Caltech Faces dataset, we are able to annotate 90.8% of the images (and 71.0% of the keypoints) by annotating only 58 average images.

	Global average	k-means	Spectral clustering	[84]	[86]	[84] + [86]	Ours	error between annotators
BioID	4.80	3.90	3.69	3.93	4.42	3.73	**1.93**	N/A (1 annotator)
LFPW	5.97	5.75	5.70	5.69	5.83	5.28	**3.38**	2.40 (2-3 annotators)
Caltech Faces	5.05	4.74	4.88	4.86	5.05	4.65	**2.65**	2.43 (1-7 annotators)

Table 4.3: *Keypoint annotation mean pixel error rates.* The error rates always compares the annotations produced by each method to the ground-truth human annotations. Lower numbers are better. For global average and [86], we annotate a single average image. For all other methods, we annotate 27, 22, and 58 average images on BioID, LFPW, and Caltech Faces, respectively. The last column shows the error between the different human annotators.

our averages are significantly sharper due to more accurate interactive clustering, local warping and alignment between intra-cluster images.

Visual data representation user study: We conducted a user study comparing different ways to compactly represent a visual dataset: 1) Ours: M average images generated with our system by an experienced user; 2) Manual: M manually picked iconic images that represent the database; and 3) Random: M randomly sampled images from the database. The bottom two rows in Table 4.1 show the time it took to create/select the images with $M = \{3, 6, 12, 24\}$ for Ours and Manual (Random is automatic).

For each experiment, we presented the subjects with M images (on the same screen) which "collectively represent some concept". We also displayed 30 test images and asked the subjects to identify 15 of them which are likely to be examples of the same (unspecified) concept as the "training" images above. We picked two concepts: *'Paris'* and *'Kids with Santa'*, with $M = \{3, 6, 12, 24\}$. We used 15 randomly sampled non-Paris Flickr images and 15 "Santa Claus" internet images as distracters for *'Paris'* and *'Kids with Santa'*, respectively. Table 4.1 shows the result. Across both concepts, the users were able to select the correct images around 68% of the time using both Ours and Manual, and 61% of the time using Random. This suggests

that our system is able to produce a concise representation of the database that is as informative as manually selected iconic images, but using significantly less time.

User experience study: We next design an experiment to evaluate the user experience of our system. We invited 5 novice users to create average images that correspond to text descriptions such as "a kid wearing Santa suit". Each user spent 5-10 minutes getting familiar with the user interface, using an image collection unrelated to the study. We then asked each user to create average images for the following text descriptions in three datasets: 1) Kids with Santa: "a kid wearing Santa's suit", "two kids sitting on Santa's both laps", and "two Santas" (difficult); 2) Cat: "a brown tabby cat", "a solid black cat" and "a cat looking up at the sky" (difficult); 3) Horse: "a person riding a horse", "a horse facing to the right", and "a horse sitting on the ground" (difficult). We allowed each user to spend 4 minutes per average image. We label a text query as "difficult" if there is insufficient data to create an average corresponding to the text description. This is to measure how long (in the allotted 4-minutes) it takes a user to realize that the corresponding mode does not exist in the database. On average, the users spent 201 seconds on the difficult query (as reference, they spent 108 seconds on the other queries). 93.3% of time, the users believed that it was impossible to create the difficult mode. We show examples of the user-generated averages in Figure 4.8. For the difficult queries, the users were unable to create an average image that corresponded to the text description. The resulting average images were distorted, blurry, or irrelevant to the text description despite the users spending more time to create them.

Rapid image dataset annotation: Our system's ability to accurately align images suggests potential applications in computer vision, such as rapid annotation of keypoints, for example, to train an object detector. Object/keypoint detection requires a lot of training data and the standard approach to keypoint annotation is to mark every image independently, by hand. This can be extremely tedious and time-consuming, especially for very large datasets. With AverageExplorer, we show that one can significantly accelerate this process. We evaluate our system by annotating human face keypoints on three widely-used datasets: BioID [96], Labeled Face Parts in the Wild (LFPW) [13] and Caltech Web Faces [6]. For each dataset, we first run the Viola-Jones face detector to get a coarse alignment of the faces, and resize each face to 200x200 pixels. We then cluster and align the faces using our system, and annotate the keypoints on the resulting average images, which are then automatically propagated to the corresponding cluster members. This way, only a few clusters need to be annotated manually, instead of each image.

Table 4.2 shows keypoint annotation statistics. Using AverageExplorer, we are able to annotate 88-98% of the images and 71-94% of the keypoints by annotating only 22-58 average images. This is a huge saving in effort compared to the standard approach

for annotation, which would require annotating 798-6476 images. Figure 4.9 (a) and (b) show examples of annotated averages and propagated annotations, respectively.

Table 4.3 shows pixel error rates compared to the ground-truth human annotations (GT) for ours and several baselines. For each method, we compute the error by averaging the average pair-wise difference for all keypoints across all images. Computing a single global average given the entire image stack produces high error rates. Clustering methods (k-means, spectral clustering [185], and discriminative sub-category discovery [84]) do slightly better, but without any form of alignment these methods cannot be used to make fine keypoint correspondences. *Congealing* [86], which is an automatic algorithm that jointly aligns a set of images, could also be used for keypoint annotation, but only if all data represents a single visual mode. Since our interactive method can do both, clustering *and* alignment at the same time, it is best suited for the annotation task. As can be seen in Table 4.3, our method aligns the images well, producing pixel error rates that are only slightly worse than the average error between the different human annotators. The images that we missed are those that could not be aligned well, such as outlier images that do not belong to any mode (see Figure 4.9c).

4.5 Discussion

Our work is but a small step in an exciting new direction of interactive Visual Data Exploration. We hope that the ideas and the prototype system presented in this chapter will inspire others to explore this ripe research topic. Here, we will first sketch some potential applications of our system and then discuss its current limitations.

Interactive portraits: We believe our system could be a fun alternative to still image portraits, e.g., displayed on social networking sites like FACEBOOK (see Figure 4.10). Social networking sites have lots of portraits for each user. Instead of the user selecting a single portrait, we envision each web visitor exploring and browsing a collection [58] of the user's face images (e.g., detected via a face detector). Our system could add an element of human interaction to the portraits, allowing each visitor to freely explore, e.g., different hairstyles, expressions, clothing, etc. of the user's portrait as s/he likes. See our video for this idea in action.

Visual data analytics: Analytics is the discovery and communication of meaningful (potentially hidden) patterns in data. Due to AverageExplorer's progressive updates, in which the user's edits cumulatively change the average image, we can "fix" (i.e. condition on) one region and observe the resulting different modes in other regions. This could allows us to do a simple form of conditional visual analytics. For example,

Figure 4.10: *Interactive portraits.* AverageExplorer could be used in social networking sites to explore the owner's face portraits.

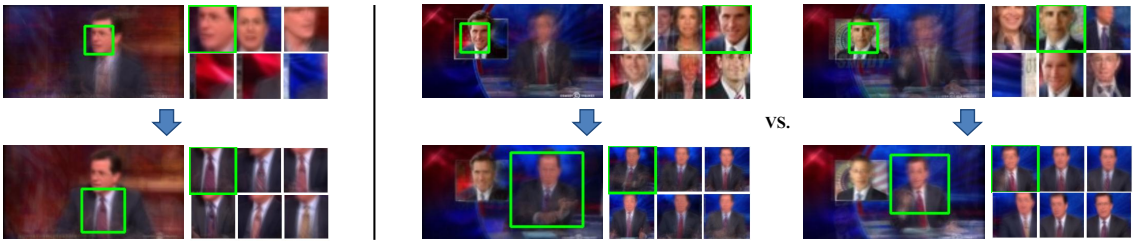


Figure 4.11: *Visual data-driven analytics.* We demonstrate a simple form of conditional data-driven analytics. We can discover Stephen Colbert's different ties (left) and explore his posture when discussing Romney versus Obama (right). The green boxes show the user selected regions via our explorer tool and the average patches to the right display the corresponding main modes.

given a large collection of The Colbert Report footage, we could discover what Stephen Colbert's ties look like by first fixing the region of interest on his face and then exploring the region beneath his face (Figure 4.11 left). In a similar manner, we could also observe what Stephen Colbert's posture or expression looks like when he is discussing Romney versus Obama (Figure 4.11 right).

Limitations Following are the main limitations of our current system. First, our system assumes that the image collection to be explored is already in some kind of "rough" alignment. This is typically not a problem for object-centric datasets like those one might get by using an Internet Search Engine, e.g. searching for "car" will mostly return images with a large car at the center of the image. However, for more scene-centric datasets, like Google StreetView, cars will be a small part of the image and not in the center, which will make them almost impossible to find using our

system.

Second, our system is critically relying on good visual matching. When a user's edit is incorrectly matched, the warping algorithm will produce a distorted or blurry average image. Unfortunately, there is no way to rectify an incorrect match with ensuing edits. This greedy matching approach can be problematic when the system produces a mismatch of two repeated objects (e.g. two faces) that are close to each other, and the user still desires to discover them both. One possible solution to this mismatching issue would be to develop an efficient global matching method that recomputes matching at each time step using all existing user edits and their geometric relationships.

Third, while AverageExplorer has good tools for refining clusters and making sharper averages, it is not as well-equipped at starting visual exploration "from scratch". For a dataset with a fairly complex visual concept, the initial average image is likely to be a gray nothing, making it hard to know where to start the exploration. The Explorer tool and the Cluster Spawning tool aim to address this very problem, but often they are either too fine (former) or too coarse (latter) to provide a robust solution. Some sort of a multi-scale exploration strategy might be a fruitful direction.

Lastly, speed and memory are the biggest obstacles to scaling our system up to millions of images, since our system must run in real-time. We are looking at the use of large-memory GPUs for speed up, as well as optimizing the off-line/on-line processing pipeline.

Chapter 5

Visual Manipulation with Deep Generative Models

Simple image averaging models can serve as an intuitive and artistic medium for visual data exploration. However, the generated results often look blurry and far from realistic, even with our alignment algorithm. Furthermore, novel results cannot always be represented as simple combinations of dataset images. To produce more natural and diverse results, we turn to recently developed deep generative models, due to the quality and complexity of their samples. In this chapter, we propose to learn the natural image manifold directly from data, using a Generative Adversarial Network [67]. We then define a class of image editing operations, constraining their outputs to lie on that learned manifold at all times. The model automatically adjusts the outputs based on user desires, while keeping all edits as realistic as possible. All our manipulations are expressed in terms of constrained optimization and are applied in near-real time. We evaluate our algorithm on the task of realistic photo manipulation in shape and color. The presented method can further be used for changing one image to look like another, as well as generating novel imagery from scratch based on a user's scribbles.

5.1 Introduction

As mentioned in Chapter 1, today's visual communication is sadly one-sided. We all perceive information in the visual form (through photographs, paintings, sculpture, etc), but only a chosen few are talented enough to effectively express themselves visually. This imbalance manifests itself even in the most mundane tasks. Consider an online shopping scenario: a user looking for shoes has found a pair that mostly

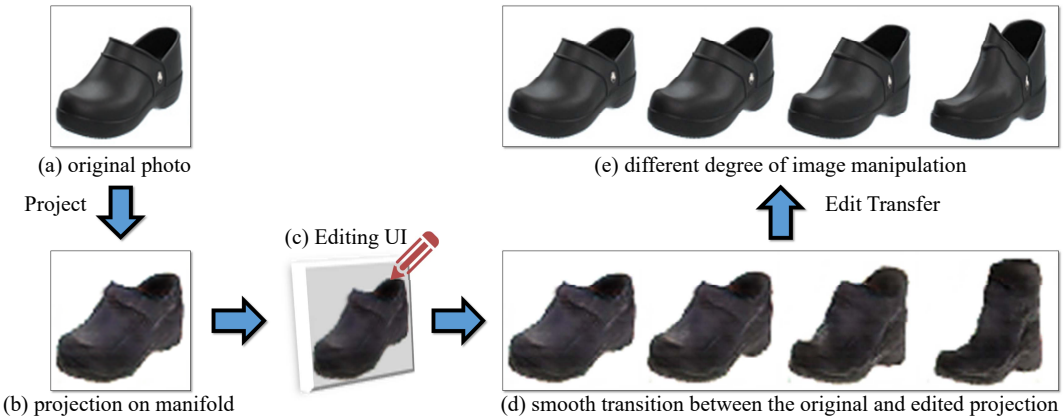


Figure 5.1: We use generative adversarial networks (GAN) [67, 162] to perform image editing on the natural image manifold. We first project an original photo (a) onto a low-dimensional latent vector representation (b) by regenerating it using GAN. We then modify the color and shape of the generated image (d) using various brush tools (c) (for example, dragging the top of the shoe). Finally, we apply the same amount of geometric and color changes to the original photo to achieve the final result (e). **See interactive image editing video on our website.**

suits her but she would like them to be a little taller, or wider, or in a different color. How can she communicate her preference to the shopping website? If the user is also an artist, then a few minutes with an image editing program will allow her to transform the shoe into what she wants, and then use image-based search to find it. However, for most of us, even a simple image manipulation in Photoshop presents insurmountable difficulties. One reason is the lack of "safety wheels" in image editing: any less-than-perfect edit immediately makes the image look completely unrealistic. To put another way, classic visual manipulation paradigm does not prevent the user from "falling off" the manifold of natural images.

Understanding and modeling the natural image manifold has been a long-standing open research problem. But in the last two years, there has been rapid advancement, fueled largely by the development of the generative adversarial networks [67]. In particular, several recent papers [37, 44, 67, 113, 162] have shown visually impressive results sampling random images drawn from the natural image manifold. However, there are two reasons preventing these advances from being useful in practical applications at this time. First, the generated images, while good, are still not quite photo-realistic (plus there are practical issues in making them high resolution). Second, these generative models are setup to produce images by sampling a latent

vector-space, typically at random. So, these methods are not able to create and/or manipulate visual content in a user-controlled fashion.

In this chapter, we use the generative adversarial neural network to learn the manifold of natural images, but we do not actually employ it for image generation. Instead, we use it as a constraint on the output of various image manipulation operations, to make sure the results lie on the learned manifold at all times. This enables us to reformulate several editing operations, specifically color and shape manipulations, in a natural and data-driven way. The model automatically adjusts the output keeping all edits as realistic as possible (Figure 5.1).

We show three applications based on our system: (1) Manipulating an existing photo based on an underlying generative model to achieve a different look (shape and color); (2) "Generative transformation" of one image to look more like another; (3) Generate a new image from scratch based on user's scribbles and warping UI.

All manipulations are performed in a straightforward manner through gradient-based optimization, resulting in a simple and fast image editing tool. We hope that this work inspires further research in data-driven generative image editing, and thus release the code and data at our website.

5.2 Background

Image editing and user interaction: As described in Chapter 1, image editing is a well established area in computer graphics where an input image is manipulated to achieve a certain goal specified by the user. Examples of basic editing include changing the color properties of an image either globally [166] or locally [127]. More advanced editing methods such as image warping [9, 90, 115] or structured image editing [12] intelligently reshuffle the pixels in an image following user's edits. While achieving impressive results in the hands of an expert, when these types of methods fail, they produce results that look nothing like a real image. Common artifacts include unrealistic colors, exaggerated stretching, obvious repetitions and over-smoothing. This is because they rely on low-level principles (e.g., similarity of color, gradients or patches) and do not capture higher-level information about natural images.

Image morphing: There are a number of techniques for producing a smooth visual transition between two input images. Traditional morphing methods [221] combine an intensity blend with a geometric warp that requires a dense correspondence. In Regenerative Morphing [184] the output sequence is regenerated from small patches sampled from the source images. Thus, each frame is constrained to look similar to the two sources. Exploring Photobios [107] presented an alternative way to transition between images, by finding a shortest path in a large image collection based on

pairwise image distances. Here we extend this idea and produce a morph that is both close to the two sources and stays on, or close to, the natural image manifold.

Natural image statistics: Generative models of local image statistics have long been used as a prior for image restoration problems such as image denoising and deblurring. A common strategy is to learn local filter or patch models, such as Principal Components, Independent Components, Mixture of Gaussians or wavelet bases [149, 159, 251]. Some methods attempt to capture full-image likelihoods [173] through dense patch overlap, though the basic building block is still small patches that do not capture global image structures and long range relations. Zhu et al. [246] recently showed that discriminative deep neural networks learn a much stronger prior that captures both low-level statistics, as well as higher order semantic or color-balance clues. This deep prior can be directly used for a limited set of editing operations (e.g. compositing). However it does not extend to the diversity of editing operations considered in this work.

Neural generative models: There is a large body of work on neural network based models for image generation. Early classes of probabilistic models of images include restricted Boltzmann machines (e.g., [82]) and their deep variants [175], auto-encoders [82, 211] and more recently, stochastic neural networks [14, 69, 113] and deterministic networks [45]. Generative adversarial networks (GAN), proposed by Goodfellow et al. [67], learn a generative network jointly with a second discriminative adversarial network in a mini-max objective. The discriminator tries to distinguish between the generated samples and natural image samples, while the generator tries to *fool* the discriminator producing highly realistic looking images. Unfortunately in practice, GAN does not yield a stable training objective, so several modifications have been proposed recently, such as a multi-scale generation [37] and a convolution-deconvolution architecture with batch normalization [162]. While the above methods attempt to generate an image starting from a random vector, they do not provide tools to change the generation process with intuitive user controls. In this chapter we try to remedy this by learning a generative model that can be easily controlled via a few intuitive user edits.

5.3 Learning the Natural Image Manifold

Let us assume that all natural images lie on an ideal low-dimensional manifold $\mathbb{M}$ with a distance function $S(x_1, x_2)$ that measures the perceptual similarity between two images $x_1, x_2 \in \mathbb{M}$. Directly modeling this ideal manifold $\mathbb{M}$ is extremely challenging, as it involves training a generative model in a highly structured and complex million dimensional space. Following the recent success of deep generative networks in

(a) random samples (b) random jittering (c) linear interpolation

Figure 5.2: GAN as a manifold approximation. (a) Randomly generated examples from a GAN, trained on the shirts dataset; (b) random jittering: each row shows a random sample from a GAN (the first one at the left), and its variants produced by adding Gaussian noise to z in the latent space; (c) interpolation: each row shows two randomly generated images (first and last), and their smooth interpolations in the latent space.

generating natural looking images, we approximate the image manifold by learning a model using generative adversarial networks (GAN) [67,162] from a large-scale image collection. Beside the high quality results, GAN has a few other useful properties for our task we will discuss next.

Generative Adversarial Networks: A GAN model consists of two neural networks: (1) a generative network $G(z;\theta_g)$ that generates an image $x \in \mathbb{R}^{H\times W\times C}$ given a random vector $z \in \mathbb{Z}$, where $\mathbb{Z}$ denotes a d-dimensional latent space, and (2) a discriminative network $D(x;\theta_d)$ that predicts a probability of a photo being real ($D=1$) or generated ($D=0$). For simplicity, we denote $G(z;\theta_G)$ and $D(x;\theta_D)$ as $G(z)$ and $D(x)$ in later sections. One common choice of $\mathbb{Z}$ is a multivariate uniform distribution $Unif[-1,1]^d$. D and G are learned using a min-max objective [67]. GAN works well when trained on images of a certain class. We formally define $\tilde{\mathbb{M}} = \{G(z)|z \in \mathbb{Z}\}$ and use it as an approximation to the ideal manifold $\mathbb{M}$ (i.e $\tilde{\mathbb{M}} \approx \mathbb{M}$). We also approximate the distance function of two generated images as an Euclidean distance between their corresponding latent vectors, i.e., $S(G(z_1), G(z_2)) \approx \|z_1 - z_2\|^2$.

GAN as a manifold approximation: We use GAN to approximate an ideal manifold for two reasons: first, it produces high-quality samples (see Figure 5.2 (a) for example). Though lacking visual details sometimes, the model can synthesize appealing samples with a plausible overall structure. Second, the Euclidean distance in the latent space often corresponds to a perceptually meaningful visual similarity (see Figure 5.2 (b) for examples). We therefore argue that GAN is a powerful generative model for modeling the image manifold.

Traversing the manifold: Given two images on the manifold $G(z_0), G(z_N)) \in$

$\tilde{\mathbb{M}}$, one would like to seek a sequence of $N+1$ images $\left[G(z_0), G(z_1), \ldots G(z_N)\right]$ with a smooth transition. This is often done by constructing an image graph with images as nodes, and pairwise distance function as the edge, and computing a shortest path between the starting image and end image [107]. In our case, we minimize $\sum_{t=0}^{N-1} S(G(z_t), G(z_{t+1}))$ where S is the distance function. In our case $S(G(z_1), G(z_2)) \approx \|z_1 - z_2\|^2$, so a simple linear interpolation $\left[(1-\frac{t}{N})\cdot z_0 + \frac{t}{N}\cdot z_N\right]_{t=0}^{N}$ is the shortest path. Figure 5.2 (c) shows a smooth and meaningful image sequence generated by interpolating between two points in the latent space. We will now use this approximation of the manifold of natural images for realistic photo editing.

5.4 Approach

Figure 5.1 illustrates the overview of our approach. Given a real photo, we first project it onto our approximation of the image manifold by finding the closest latent feature vector z of the GAN to the original image. Then, we present a real-time method for gradually and smoothly updating the latent vector z so that it generates a desired image that both satisfies the user's edits (e.g. a scribble or a warp; more details in Section 5.5) and stays close to the natural image manifold. Unfortunately, in this transformation the generative model usually looses some of the important low-level details of the input image. We therefore propose a dense correspondence method that estimates both per-pixel color and shape changes from the edits applied to the generative model. We then transfer these changes to the original photo using an edge-aware interpolation technique and produce the final manipulated result.

5.4.1 Projecting an Image onto the Manifold

A real photo x^R lies, by definition, on the ideal image manifold $\mathbb{M}$. However for an approximate manifold $\tilde{\mathbb{M}}$, our goal here is to find a generated image $x^* \in \tilde{\mathbb{M}}$ close to x^R in some distance metric $\mathcal{L}(x_1, x_2)$ as

$$x^* = \arg\min_{x \in \tilde{\mathbb{M}}} \mathcal{L}(x,\, x^R). \tag{5.1}$$

For the GAN manifold $\tilde{\mathbb{M}}$ we can rewrite the above equation as follows:

$$z^* = \arg\min_{z \in \tilde{\mathbb{Z}}} \mathcal{L}(G(z),\, x^R). \tag{5.2}$$

Our goal is to reconstruct the original photo x^R using the generative model G by minimizing the reconstruction error, where $\mathcal{L}(x_1, x_2) = \|\mathcal{C}(x_1) - \mathcal{C}(x_2)\|^2$ in some

differentiable feature space $\mathcal{C}$. If $\mathcal{C}(x) = x$, then the reconstruction error is simply pixel-wise Euclidean error. Previous work [44, 99] suggests that using deep neural network activations leads to a reconstruction of perceptually meaningful details. We found that a weighted combination of raw pixels and *conv4* features ($\times 0.002$) extracted from AlexNet [116] trained on ImageNet [36] to perform best.

Projection via optimization: As both the feature extractor $\mathcal{C}$ and the generative model G are differentiable, we can directly optimize the above objective using L-BFGS-B [23]. However, the cascade of $\mathcal{C}(G(z))$ makes the problem highly non-convex, and as a result, the reconstruction quality strongly relies on a good initialization of z. We can start from multiple random initializations and output the solution with the minimal cost. However the number of random initializations required to obtain a stable reconstruction is prohibitively large (more than 100), which makes real-time processing impossible. We instead train a deep neural network to minimize equation 5.2 directly.

Projection via a feedforward network: We train a feedforward neural network $P(x; \theta_P)$ that directly predicts the latent vector z from a x. The training objective for the predictive model P is written as follows:

$$\theta_P^* = \arg\min_{\theta_P} \sum_n \mathcal{L}(G(P(x_n^R; \theta_P)),\, x_n^R), \tag{5.3}$$

where x_n^R denotes the n-th image in the dataset. The architecture of the model P is equivalent to the discriminator D of the adversarial networks, and only varies in the final number of network outputs. Objective 5.3 is reminiscent of an auto-encoder pipeline, with a encoder P and decoder G. However, the decoder G is fixed throughout the training. While the optimization problem 5.2 is exactly the same as the learning objective 5.3, the learning based approach often performs better and does not fall into local optima. We attribute this behavior to the regularity in the projection problem and the limited capacity of the network P. Projections of similar images will share similar network parameters and produce a similar result. In some sense the loss for one image provides information for many more images that share a similar appearance [62]. However, the learned inversion is not always perfect, and can often be improved further by a few additional steps of optimization.

A hybrid method: The hybrid method takes advantage of both approaches above. Given a real photo x^R, we first predict $P(x^R; \theta_P)$ and then use it as the initialization for the optimization objective (Equation 5.2). So the predictive model we have trained serves as a fast bottom-up initialization method for a non-convex optimization problem. Figure 5.3 shows a comparison of these three methods. See Section 5.7.4 for a more quantitative evaluation.

Figure 5.3: Projecting real photos onto the image manifold using GAN. Top row: original photos (from handbag dataset); 2nd row: reconstruction using optimization-based method; 3rd row: reconstruction via learned deep encoder P; bottom row: reconstruction using the hybrid method (ours). We show the reconstruction loss below each image.

5.4.2 Manipulating the Latent Vector

With the image x_0^R projected onto the manifold $\tilde{\mathbb{M}}$ as $x_0 = G(z_0)$ via the projection methods just described, we can start modifying the image on that manifold. We update the initial projection x_0 by simultaneously matching the user intentions while staying on the manifold, close to the original image x_0.

Each editing operation is formulated as a constraint $f_g(x) = v_g$ on a local part of the output image x. The editing operations g include color, shape and warping constraints, and are further described in Section 5.5.1. Given an initial projection x_0, we find a new image $x \in \mathbb{M}$ close to x_0 trying to satisfy as many constraints as possible

$$x^* = \arg\min_{x \in \mathbb{M}} \Big\{ \underbrace{\sum_g \|f_g(x) - v_g\|^2}_{\text{data term}} + \underbrace{\lambda_s \cdot S(x, x_0)}_{\substack{\text{manifold} \\ \text{smoothness}}} \Big\}, \tag{5.4}$$

where the data term measures deviation from the constraint and the smoothness term enforces moving in small steps on the manifold, so that the image content is not altered too much. We set $\lambda_s = 5$ in our experiments.

The above equation simplifies to the following on the approximate GAN manifold

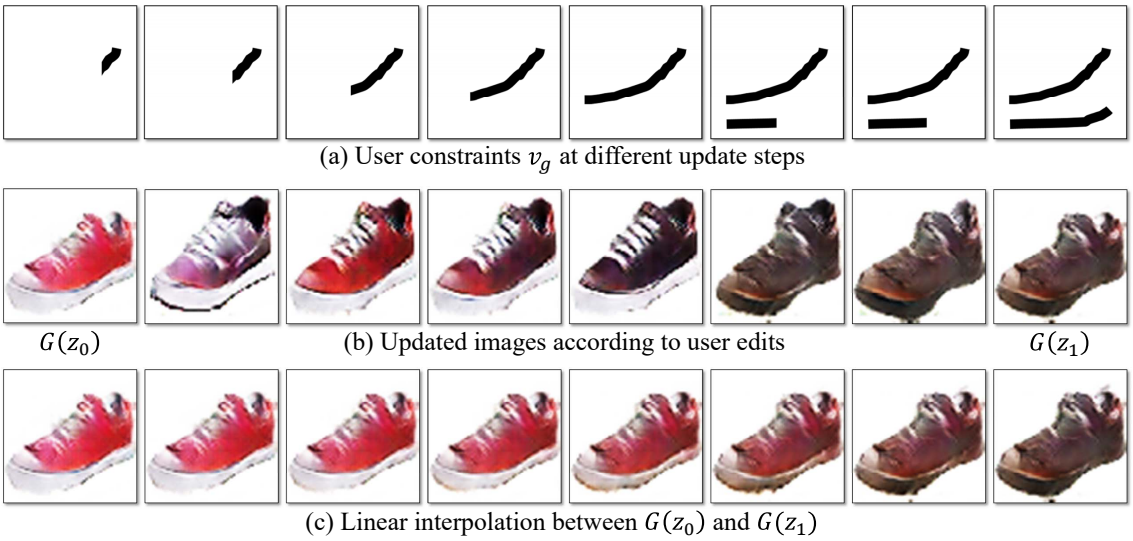


Figure 5.4: Updating latent vector given user edits. (a) Evolving user constraint v_g (black color strokes) at each update step; (b) intermediate results at each update step ($G(z_0)$ at leftmost, and $G(z_1)$ at rightmost); (c) a smooth linear interpolation in latent space between $G(z_0)$ and $G(z_1)$.

$\tilde{\mathbb{M}}$:

$$z^* = \arg\min_{z \in \mathbb{Z}} \Big\{ \underbrace{\sum_g \|f_g(G(z)) - v_g\|^2}_{\text{data term}} + \underbrace{\lambda_s \cdot \|z - z_0\|^2}_{\substack{\text{manifold} \\ \text{smoothness}}} + E_D \Big\}. \tag{5.5}$$

Here the last term $E_D = \lambda_D \cdot \log(1 - D(G(z)))$ optionally captures the visual realism of the generated output as judged by the GAN discriminator D. This further pushes the image towards the manifold of natural images, and slightly improves the visual quality of the result. By default, we turn off this term to increase frame rates.

Gradient descent update: For most constraints Equation 5.5 is non-convex. We solve it using gradient descent, which allows us to provide the user with a real-time feedback as she manipulates the image. As a result, the objective 5.5 evolves in real-time as well. For computational reasons, we only perform a few gradient descent updates after changing the constraints v_g. Each update step takes $50 - 100$ ms, which ensures an interactive feedback. Figure 5.4 shows one example of the update of z. Given an initial red shoe as shown in Figure 5.4, the user gradually scribbles a black color stroke (i.e. specifies a region is black) on the shoe image (Figure 5.4 a). Then our update method smoothly changes the image appearance (Figure 5.4 b) by adding more and more of the user constraints. Once the final result $G(z_1)$ is computed, a

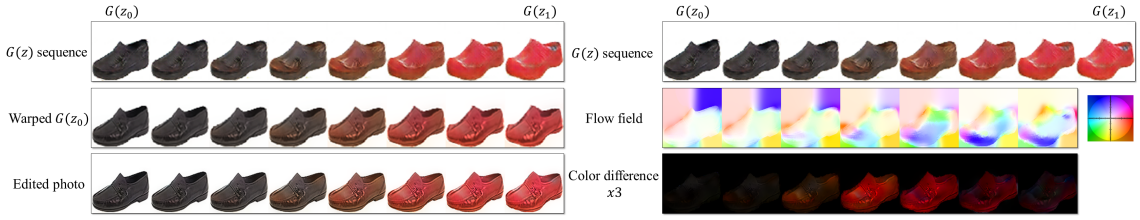


Figure 5.5: Edit transfer via Motion+Color Flow. Following user edits on the left shoe $G(z_0)$ we obtain an interpolation sequence in the generated latent space $G(z)$ (top right). We then compute the motion and color flows (right middle and bottom) between neighboring images in $G(z)$. These flows are concatenated and, as a validation, can be applied on $G(z_0)$ to obtain a close reconstruction of $G(z)$ (left middle). The bottom left row shows how the edit is transferred to the original shoe using the same concatenated flow, to obtain a sequence of edited shoes.

user can see the interpolation sequence between the initial point z_0 and z_1 (Figure 5.4 c), and select any intermediate result as the new starting point.

While this editing framework allows us to modify any generated image on the approximate natural image manifold $\tilde{\mathbb{M}}$, it does not directly provide us a way to modify the original high resolution image x_0^R. In the next section we show how edits on the approximate manifold can be transferred to the original image.

5.4.3 Edit Transfer

Give the original photo x_0^R (e.g. a black shoe) and its projection on the manifold $G(z_0)$, and a user modification $G(z_1)$ by our method (e.g. the generated red shoe). The generated image $G(z_1)$ captures the roughly change we want, albeit the quality is degraded w.r.t the original image.

Can we instead adjust the original photo and produce a more photo-realistic result x_1^R that exhibits the changes in the generated image? A straightforward way is to transfer directly the pixel changes (i.e. $x_1^R = x_0^R + (G(z_1) - G(z_0))$. We have tried this approach and it introduces new artifacts due to the misalignment of the two images. To address this issue, we develop a dense correspondence algorithm to estimate both the geometric and color changes induced by the editing process.

Specifically, given two generated images $G(z_0)$ and $G(z_1)$, we can generate any number of intermediate frames $\left[G((1-\frac{t}{N})\cdot z_0 + \frac{t}{N}\cdot z_1)\right]_{t=0}^{N}$, where consecutive frames only exhibit minor visual variations.

Motion+Color flow algorithm: We then estimate the color and geometric changes

by generalizing the brightness constancy assumption in traditional optical flow methods [20, 21]. This results in the following motion+color flow objective[1]:

$$\iint \underbrace{\|I(x,y,t) - A \cdot I(x+u, y+v, t+1)\|^2}_{\text{data term}} + \underbrace{\sigma_s(\|\nabla u\|^2 + \|\nabla v\|^2)}_{\text{spatial reg}} + \underbrace{\sigma_c \|\nabla A\|^2}_{\text{color reg}} dxdy, \quad (5.6)$$

where $I(x, y, t)$ denotes the RGB values $(r, g, b, 1)^T$ of pixel (x, y) in the generated image $G((1 - \frac{t}{N}) \cdot z_0 + \frac{t}{N} \cdot z_1)$. (u, v) is the flow vector with respect to the change of t, and A denotes a 3×4 color affine transformation matrix. The data term relaxes the color constancy assumption by introducing a locally affine color transfer model A [186] while the spatial and color regularization terms encourage smoothness in both the motion and color change. We solve the objective by iteratively estimating the flow (u, v) using a traditional optical flow algorithm, and computing the color change A by solving a system of linear equations [186]. We iterate 3 times. We produce 8 intermediate frames (i.e. $N = 7$).

We estimate the changes between nearby frames, and concatenate these changes frame by frame to obtain long-range changes between any two frames along the interpolation sequence $z_0 \to z_1$. Figure 5.5 shows a warping sequence after we apply the flow to the initial projection $G(z_0)$.

Transfer edits to the original photo: After estimating the color and shape changes in the generated image sequence, we apply them to the original photo and produce an interesting transition sequence of photo-realistic images as shown in Figure 5.5. As the resolution of the flow and color fields are limited to the resolution of the generated image (i.e. 64×64), we upsample those edits using a guided image filter [78].

5.5 User Interface

The user interface consists of a main window showing the current edited photo, a display showing thumbnails of all the candidate results, and a slider bar to explore the interpolation sequence between the original photo and the final result.

Candidate results: Given the objective (Equation 5.5) derived with the user guidance, we generate multiple different results by initializing z as random perturbations of z_0. We generate 64 examples and show the best 9 results sorted by the objective cost (Equation 5.5).

[1]For simplicity, we omit the pixel subscript (x, y) for all the variables.

Relative edits: Once a user finishes one edit, she can drag a slider to see all the intermediate results interpolated between the original and the final manipulated photo. We call this "relative edits" as it allows a user to explore more alternatives with a single edit. Similar to relative attributes [154], a user can express ideas like changing the handle of the handbag to be more red, or making the heel of the shoes slightly higher, without committing to a specific final state.

5.5.1 Editing constraints

Our system provides three constraints to edit the photo in different aspects: coloring, sketching and warping. All constraints are expressed as brush tools. In the following, we explain the usage of each brush, and the corresponding constraints.
Coloring brush: The coloring brush allows the user to change the color of a specific region. The user selects a color from a palette and can adjust the brush size. For each pixel marked with this brush we constrain the color $f_g(I) = I_p = v_g$ of a pixel p to the selected values v_g.
Sketching brush: The sketching brush allows the user to outline the shape or add fine details. We constrain $f_g(I) = HOG(I)_p$ a differentiable HOG descriptor [33] at a certain location p in the image to be close to the user stroke (i.e. $v_g = HOG(stroke)_p$). We chose the HOG feature extractor because it is binned, which makes it robust to sketching inaccuracies.
Warping brush: The warping brush allows the user to modify the shape more explicitly. The user first selects a local region (a window with adjustable size), and then drag it to another location. We then place both a color and sketching constraint on the displaced pixels encouraging the target patch to mimic the appearance of the dragged region.

Figure 5.8 shows a few examples where the coloring and sketching brushed were used in the context of interactive image generation. Figure 5.1 shows the result of the warping brush that was used to pull the topline of the shoe up. Figure 5.6 shows a few more examples.

5.6 Implementation Details

Network architecture: We follow the same architecture of deep convolutional generative adversarial networks (DCGAN) [162]. DCGAN mainly builds on multiple convolution, deconvolution and ReLU layers, and eases the min-max training via batch normalization [92]. We train the generator G to produce a $64 \times 64 \times 3$ image given a 100-dimensional random vector. Notice that our method can also use other

generative models (e.g. variational auto-encoder [113] or future improvements in this area) to approximate the natural image manifold.
Computational time: We run our system on a Titan X GPU. Each update of the vector z takes $50 \sim 100$ milliseconds, which allows the real-time image editing and generation. Once an edit is finished, it takes $5 \sim 10$ seconds for our edit transfer method to produce high-resolution final result.

5.7 Results

We first introduce the statistics of our dataset. We then show three main applications: realistic image manipulation, generative image transformation, and generating a photo from scratch using our brush tools. Finally, we evaluate our image reconstruction methods, and perform a human perception study to understand the realism of generated results.
Datasets: We experiment with multiple photo collections from various sources as follows: "shoes" dataset [232], which has 50K shoes collected from Zappos.com (the shoes are roughly centered but not well aligned, and roughly facing left, with frontal to side view); "church outdoor" dataset (126K images) from the LSUN challenge [233]; "outdoor natural" images (150K) from the MIT Places dataset [242]; and two query-based product collections downloaded from Amazon, including "handbags" (138K) and "shirts" (137K). The downloaded handbags and shirts are roughly centered but no further alignment has been performed.

5.7.1 Image Manipulation

Our main application is photo-realistic image manipulation using the brush interactions described in Section 5.5.1. See Figure 5.6 for a few examples where the brush edits are depicted on the left (dashed line for the sketch tool, color scribble for the color brush and a red square with arrow for the warp tool).

5.7.2 Generative Image Transformation

An interesting outcome of the editing process is the sequence of intermediate generated images that can be seen as a new kind of image morphing [182, 184, 221]. We call it "generative transformation". We use this sequence to transform the shape and color of one image to look like another image automatically, i.e., *without* any user edits. This is done by applying the motion+color flow on either of the sources. Figure 5.7 shows a few "generative transform" examples.

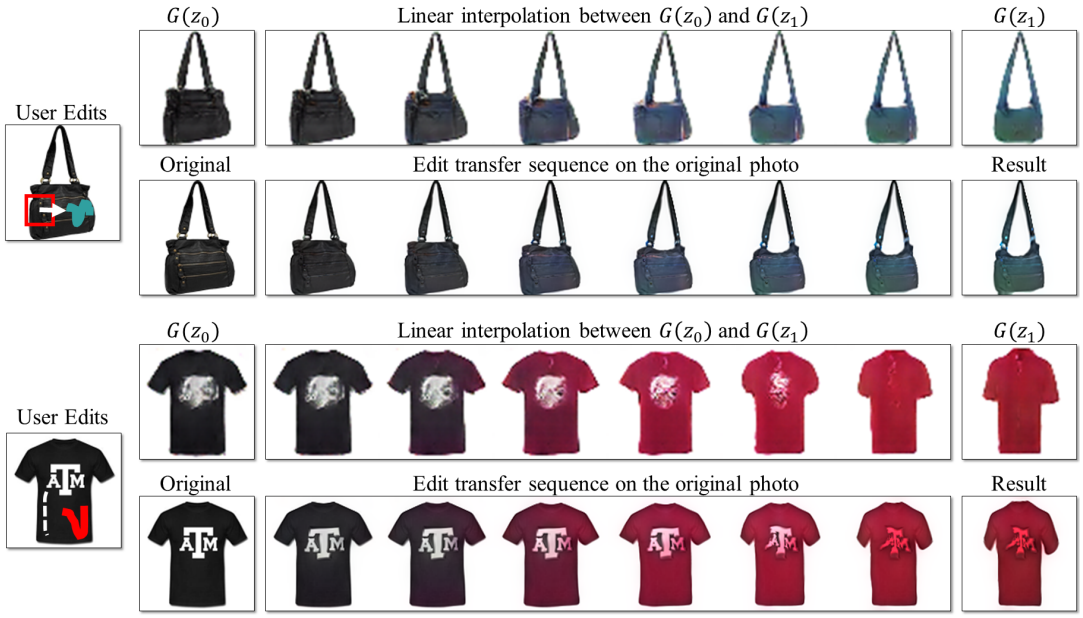


Figure 5.6: Image manipulation examples: for each example, we show the original photo and user edits on the left. The top row on the right shows the generated sequence and the bottom row shows the edit transfer sequence on the original image.

5.7.3 Interactive Image Generation

Another byproduct of our method is that if there is no image to begin with and all we have are the user brush strokes, the method would generate a natural image that best satisfies the user constraints. This could be useful for dataset exploration and browsing. The difference with previous sketch-to-image retrieval methods [196] or AverageExplorer [248], is that due to potentially contradicting user constraints, the result may look very different than any single image from the dataset or an average of such images, and more of a realistic hybrid image [170]. See some examples in Figure 5.8.

5.7.4 Evaluation

Image reconstruction evaluation: We evaluate three image reconstruction methods described in Section 5.4.1: optimization-based, network-based and our hybrid approach that combines the last two. We run these on 500 test images per category, and evaluate them by the reconstruction error $\mathcal{L}(x, x^R)$ defined in Equation 5.1.

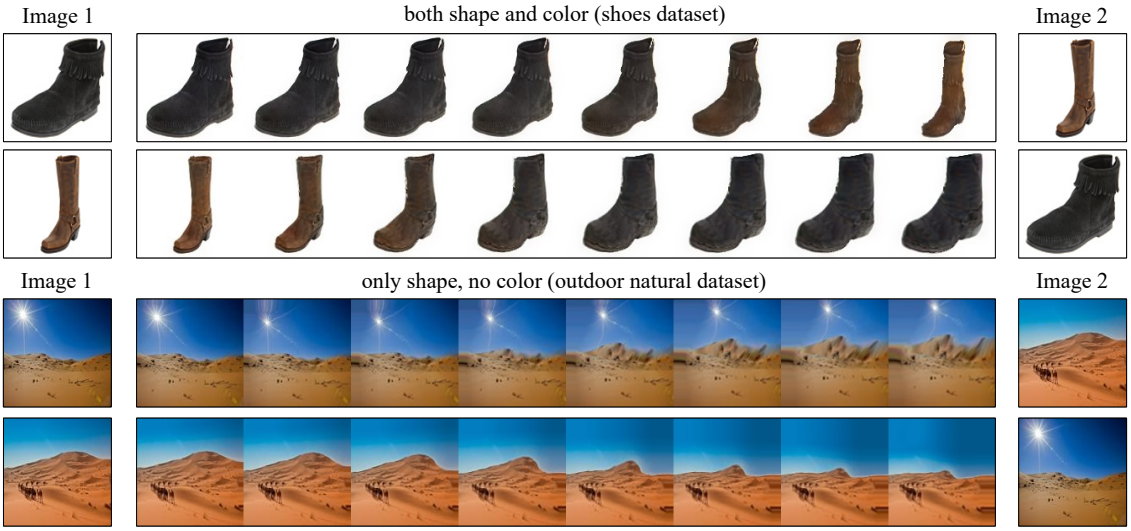


Figure 5.7: Generative image transformation. In both rows, the source on the left is transformed to have the shape and color (or just shape in the 2nd example) of the one on the right.

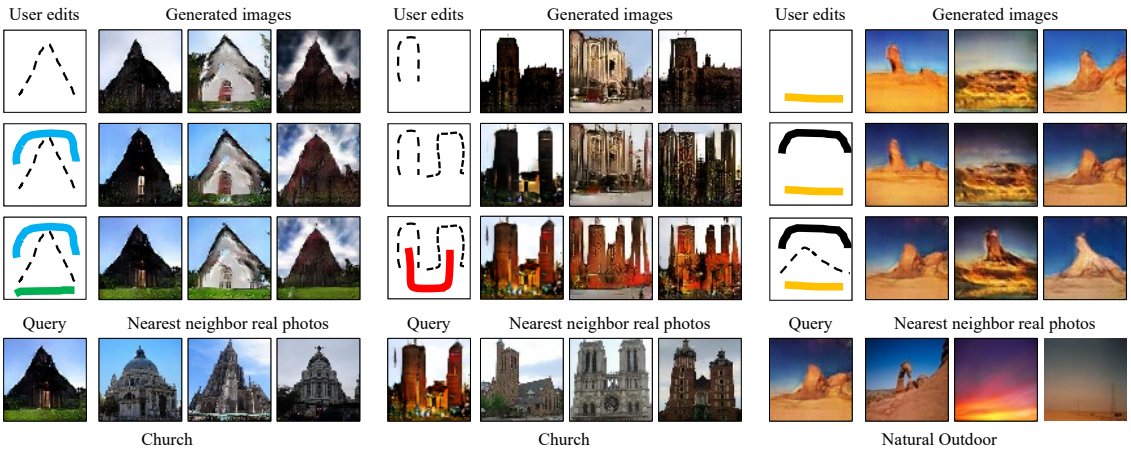


Figure 5.8: Interactive image generation. The user uses the brush tools to generate an image from scratch (top row) and then keeps adding more scribbles to refine the result (2nd and 3rd rows). In the last row, we show the most similar real images to the generated images. (dashed line for the sketch tool, and color scribble for the color brush)

	Shoes	Church	Outdoor Natural	Handbags	Shirts
Optimization-based	0.155	0.319	0.176	0.299	0.284
Network-based	0.210	0.338	0.198	0.302	0.265
Hybrid (ours)	**0.140**	**0.250**	**0.145**	**0.242**	**0.184**

Table 5.1: Average per-dataset image reconstruction error measured by $\mathcal{L}(x, x^R)$.

Table 5.1 shows the mean reconstruction error of these three methods on 5 different datasets. We can see the optimization-based and neural network-based methods perform comparably, where their combination yields better results. See Figure 5.3 for a qualitative comparison.

Class-specific model: So far, we have trained the generative model on a particular class of images. As a comparison, we train a cross-class model on three datasets altogether (i.e. shoes, handbags, and shirts), and observe that the model achieves worse reconstruction error compared to class-specific models (by $\sim 10\%$). We also have tried to use a class-specific model to reconstruct images from a different class. The mean cross-category reconstruction errors are much worse: shoes model used for shoes: 0.140 vs. shoes model for handbags: 0.398, and for shirts: 0.451. However, we expect a model trained on many categories (e.g. $1,000$) to generalize better to novel objects.

Perception study: We perform a small perception study to compare the photo realism of four types of images: real photos, generated samples produced by GAN, our method (shape only), and our method (shape+color). We collect 20 annotations for 400 images by asking Amazon Mechanical Turk workers if the image look realistic or not. Real photos: 91.5%, DCGAN: 14.3%, ours (shape+color): 25.9%; ours (shape only): 48.7%. DCGAN model alone produces less photo-realistic images, but when combined with our edit transfer, the realism significantly improves.

5.8 Discussion

We presented a step towards image editing with a direct constraint to stay close to the manifold of real images. We approximate this manifold using the state-of-the-art in deep generative models (DCGAN). We show how to make interactive edits to the generated images and transfer the resulting changes in shape and color back to the original image. Thus, the quality of the generated results (low resolution, missing texture and details) and the types of data DCGAN is applicable to (works well on structured datasets such as product images and worse on more general imagery),

limits how far we can get with this editing approach. However our method is not tied to a particular generative method and will improve with the advancement of this field. Our current editing brush tools allow rough changes in color and shape but not texture and more complex structure changes. We leave these for future work.

Part III

Image-to-Image Translation

Chapter 6

Unpaired Image-to-Image Translation

This chapter introduces image-to-image translation, a class of vision and graphics problems, where the goal is to learn the mapping between an input domain to an output domain. We briefly discuss a baseline algorithm [93] that can learn to produce a single output given paired training data. However, for many tasks, paired training data will not be available. To address this challenge, we present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. Our goal is to learn a mapping $G : X \rightarrow Y$, such that the distribution of images from $G(X)$ is indistinguishable from the distribution Y. Because this mapping is highly under-constrained, we couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a *cycle consistency loss* to enforce $F(G(X)) \approx X$ (and vice versa). Qualitative results are presented on several tasks where paired training data does not exist, including collection style transfer, object transfiguration, season transfer and photo enhancement, etc. Quantitative comparisons against several prior methods demonstrate the superiority of our approach.

6.1 Introduction

What did Claude Monet see as he placed his easel by the bank of the Seine near Argenteuil on a lovely spring day in 1873 (Figure 6.1, top-left)? A color photograph, had it been invented, may have documented a crisp blue sky and a glassy river reflecting it. Monet conveyed his *impression* of this same scene through wispy brush strokes and a bright palette.

What if Monet had happened upon the little harbor in Cassis on a cool summer

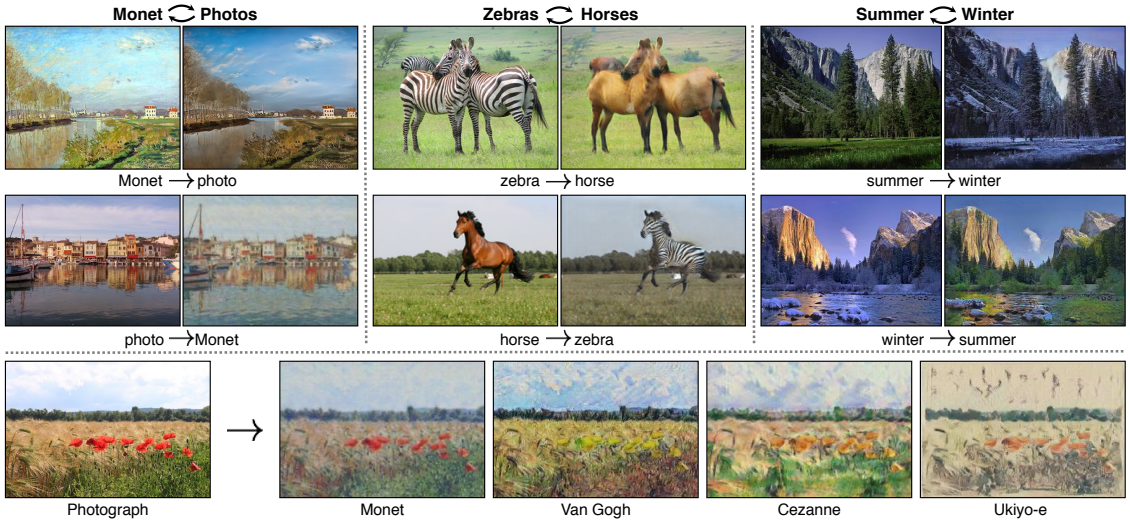


Figure 6.1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

evening (Figure 6.1, bottom-left)? A brief stroll through a gallery of Monet paintings makes it possible to imagine how he would have rendered the scene: perhaps in pastel shades, with abrupt dabs of paint, and a somewhat flattened dynamic range.

We can imagine all this despite never having seen a side by side example of a Monet painting next to a photo of the scene he painted. Instead we have knowledge of the set of Monet paintings and of the set of landscape photographs. We can reason about the stylistic differences between these two sets, and thereby imagine what a scene might look like if we were to "translate" it from one set into the other.

In this chapter, we present a method that can learn to do the same: capturing special characteristics of one image collection and figuring out how these characteristics could be translated into the other image collection, all in the absence of any paired training examples.

This problem can be more broadly described as image-to-image translation [93], converting an image from one representation of a given scene, x, to another, y, e.g., grayscale to color, image to semantic labels, edge-map to photograph. Years of research in computer vision, image processing, computational photography, and graphics have produced powerful translation systems in the supervised setting, where example image

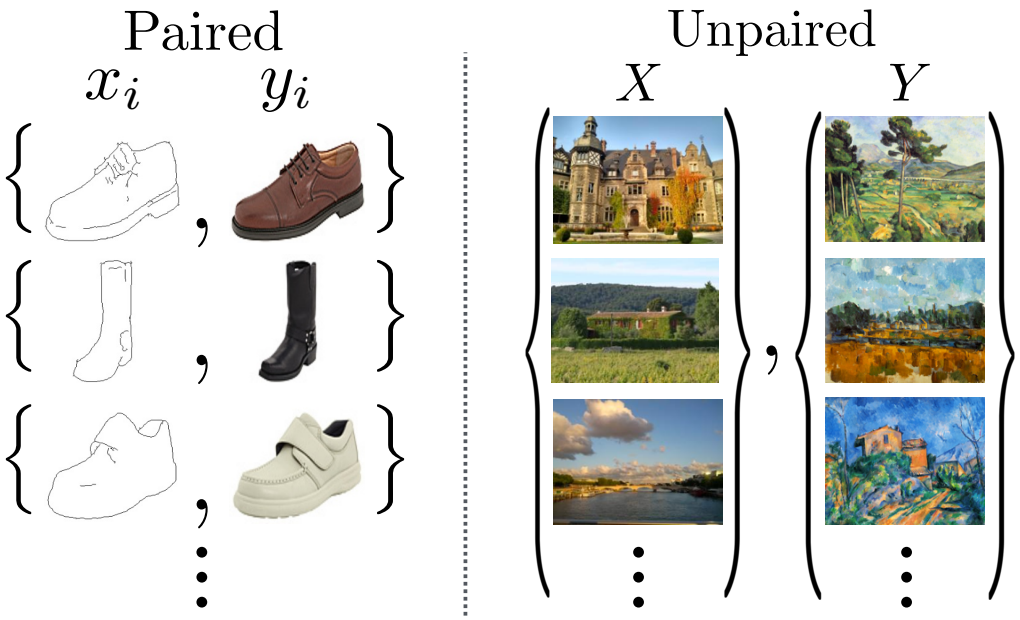


Figure 6.2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [93]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}$ ($y_j \in Y$), with no information provided as to which x_i matches which y_j.

pairs $\{x, y\}$ are available (Figure 6.2, left), e.g., [50,81,93,99,118,136,186,220,224,237]. However, obtaining paired training data can be difficult and expensive. For example, only a couple of datasets exist for tasks like semantic segmentation (e.g., [31]), and they are relatively small. Obtaining input-output pairs for graphics tasks like artistic stylization can be even more difficult since the desired output is highly complex, typically requiring artistic authoring. For many tasks, like object transfiguration (e.g., zebra↔horse, Figure 6.1 top-middle), the desired output is not even well-defined.

We therefore seek an algorithm that can learn to translate between domains without paired input-output examples (Figure 6.2, right). We assume there is some underlying relationship between the domains – for example, that they are two different renderings of the same underlying scene – and seek to learn that relationship. Although we lack supervision in the form of paired examples, we can exploit supervision at the level of sets: we are given one set of images in domain X and a different set in domain Y. We may train a mapping $G : X \rightarrow Y$ such that the output $\hat{y} = G(x)$, $x \in X$, is indistinguishable from images $y \in Y$ by an adversary trained to classify $\hat{y}$ apart from y. In theory, this objective can induce an output distribution over $\hat{y}$ that matches the empirical distribution $p_{data}(y)$ (in general, this requires G to be stochastic) [67]. The

optimal G thereby translates the domain X to a domain $\hat{Y}$ distributed identically to Y. However, such a translation does not guarantee that an individual input x and output y are paired up in a meaningful way – there are infinitely many mappings G that will induce the same distribution over $\hat{y}$. Moreover, in practice, we have found it difficult to optimize the adversarial objective in isolation: standard procedures often lead to the well-known problem of mode collapse, where all input images map to the same output image and the optimization fails to make progress [65].

These issues call for adding more structure to our objective. Therefore, we exploit the property that translation should be "cycle consistent", in the sense that if we translate, e.g., a sentence from English to French, and then translate it back from French to English, we should arrive back at the original sentence [19]. Mathematically, if we have a translator $G : X \rightarrow Y$ and another translator $F : Y \rightarrow X$, then G and F should be inverses of each other, and both mappings should be bijections. We apply this structural assumption by training both the mapping G and F simultaneously, and adding a *cycle consistency loss* [244] that encourages $F(G(x)) \approx x$ and $G(F(y)) \approx y$. Combining this loss with adversarial losses on domains X and Y yields our full objective for unpaired image-to-image translation.

We apply our method to a wide range of applications, including collection style transfer, object transfiguration, season transfer and photo enhancement. We also compare against previous approaches that rely either on hand-defined factorizations of style and content, or on shared embedding functions, and show that our method outperforms these baselines. Our code is available at `https://github.com/junyanz/CycleGAN`. Check out more results at `https://junyanz.github.io/CycleGAN/`.

6.2 Background

Generative Adversarial Networks (GANs) [67, 240] have achieved impressive results in image generation [37, 162], image editing [247], and representation learning [140, 162, 177]. Recent methods adopt the same idea for conditional image generation applications, such as text2image [164], image inpainting [155], and future prediction [139], as well as to other domains like videos [213] and 3D data [222]. The key to GANs' success is the idea of an *adversarial loss* that forces the generated images to be, in principle, indistinguishable from real images. This is particularly powerful for image generation tasks, as this is exactly the objective that much of computer graphics aims to optimize. We adopt an adversarial loss to learn the mapping such that the translated image cannot be distinguished from images in the target domain.

Image-to-Image Translation The idea of image-to-image translation goes back

at least to Hertzmann et al.'s Image Analogies [81], who employ a non-parametric texture model [49] on a single input-output training image pair. More recent approaches use a *dataset* of input-output examples to learn a parametric translation function using CNNs, e.g. [136]. Our approach builds on the "pix2pix" framework of Isola et al. [93], which uses a conditional generative adversarial network [67] to learn a mapping from input to output images. Similar ideas have been applied to various tasks such as generating photographs from sketches [178] or from attribute and semantic layouts [104]. However, unlike these prior works, we learn the mapping without paired training examples.

Unpaired Image-to-Image Translation Several other methods also tackle the unpaired setting, where the goal is to relate two data domains, X and Y. Rosales et al. [172] propose a Bayesian framework that includes a prior based on a patch-based Markov random field computed from a source image, and a likelihood term obtained from multiple style images. More recently, CoGAN [134] and cross-modal scene networks [10] use a weight-sharing strategy to learn a common representation across domains. Concurrent to our method, Liu et al. [133] extends this framework with a combination of variational autoencoders [113] and generative adversarial networks. Another line of concurrent work [17, 187, 200] encourages the input and output to share certain "content" features even though they may differ in "style". They also use adversarial networks, with additional terms to enforce the output to be close to the input in a predefined metric space, such as class label space [17], image pixel space [187], and image feature space [200].

Unlike the above approaches, our formulation does not rely on any task-specific, predefined similarity function between the input and output, nor do we assume that the input and output have to lie in the same low-dimensional embedding space. This makes our method a general-purpose solution for many vision and graphics tasks. We directly compare against several prior and contemporary approaches in Section 6.6.1.

Cycle Consistency The idea of using transitivity as a way to regularize structured data has a long history. In visual tracking, enforcing simple forward-backward consistency has been a standard trick for decades [197]. In the language domain, verifying and improving translations via "back translation and reconsiliation" is a technique used by human translators [19] (including, humorously, by Mark Twain [207]), as well as by machines [77]. More recently, higher-order cycle consistency has been used in structure from motion [234], 3D shape matching [89], co-segmentation [215], dense semantic alignment [243, 244], and depth estimation [64]. Of these, Zhou et al. [244] and Godard et al. [64] are most similar to our work, as they use a *cycle consistency loss* as a way of using transitivity to supervise CNN training. In this work, we are introducing a similar loss to push G and F to be consistent with each other. Concurrent with our work, in these same proceedings, Yi et al. [231] independently

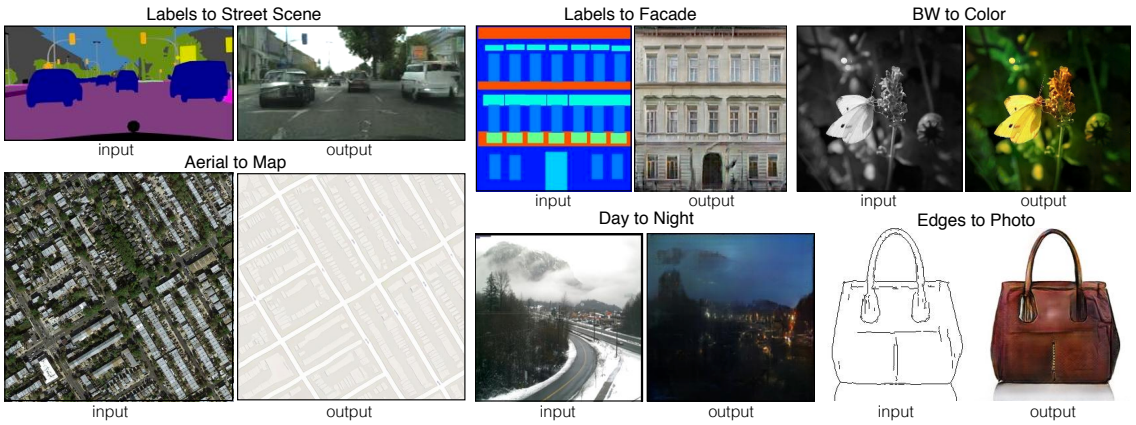


Figure 6.3: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the `pix2pix` method on several. In each case the models use the same architecture and objective, and simply train on different data.

use a similar objective for unpaired image-to-image translation, inspired by dual learning in machine translation [77].

Neural Style Transfer [60,61,99,208] is another way to perform image-to-image translation, which synthesizes a novel image by combining the content of one image with the style of another image (typically a painting) based on matching the Gram matrix statistics of pre-trained deep features. Our main focus, on the other hand, is learning the mapping between two image collections, rather than between two specific images, by trying to capture correspondences between higher-level appearance structures. Therefore, our method can be applied to other tasks, such as painting→ photo, object transfiguration, etc. where single sample transfer methods do not perform well. We compare these two methods in Section 6.6.2.

6.3 Paired Image-to-Image Translation

Here, we first briefly review the `pix2pix` framework (Isola et al. [93]), an image-conditional GANs that learn a mapping from observed image x to y, $G : x \rightarrow y$ where x and y follows the data distribution $x \sim p_{\text{data}}(x)$ and $y \sim p_{\text{data}}(y)$. The training data contains input-output pairs $\{(x_i, y_i)\}_{i=1}^{N}$ with the joint distribution

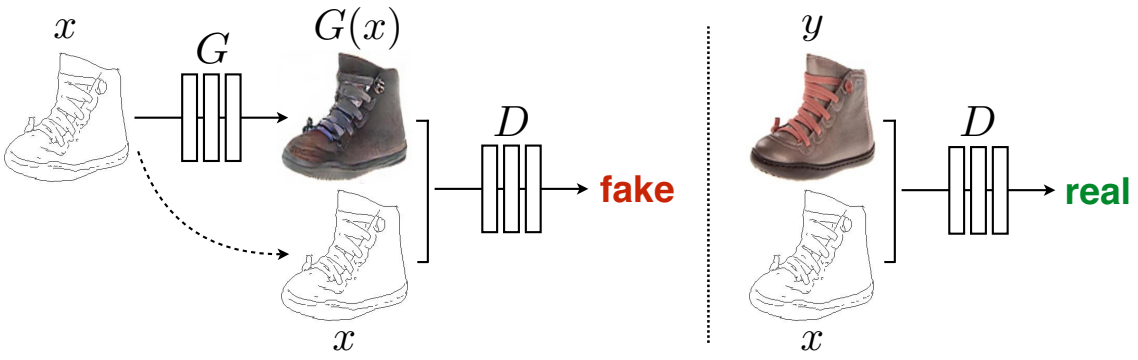


Figure 6.4: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

$(x, y) \sim p_{\text{data}}(x, y)$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes". This training procedure is diagrammed in Figure 6.4.

The adversarial loss of `pix2pix` can be expressed as

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{(x,y)\sim p_{\text{data}}(x,y)}[\log D(x, y)] + \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(1 - D(x, G(x))], \quad (6.1)$$

where G tries to minimize this objective against an adversarial D that tries to maximize it, i.e. $G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D)$.

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance [155]. The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L2 sense. The `pix2pix` framework uses a L1 distance rather than L2 as L1 encourages less blurring:

$$\mathcal{L}_{\text{L1}}(G) = \mathbb{E}_{(x,y)\sim p_{\text{data}}(x,y)}[\|y - G(x)\|_1]. \quad (6.2)$$

The final objective of `pix2pix` can be formulated as:

$$G^* = \arg\min_G \max_D \mathcal{L}_{\text{cGAN}}(G, D) + \lambda \mathcal{L}_{\text{L1}}(G). \quad (6.3)$$

The `pix2pix` method implements the generator as an U-Net [171] and the discriminator as a fully convolutional network [136] for classifying 70×70 overlapping patches. Please find more details in the original paper https://arxiv.org/abs/1611.07004. Figure 6.3 shows a few example results generated by `pix2pix` framework on various image-to-image translation tasks.

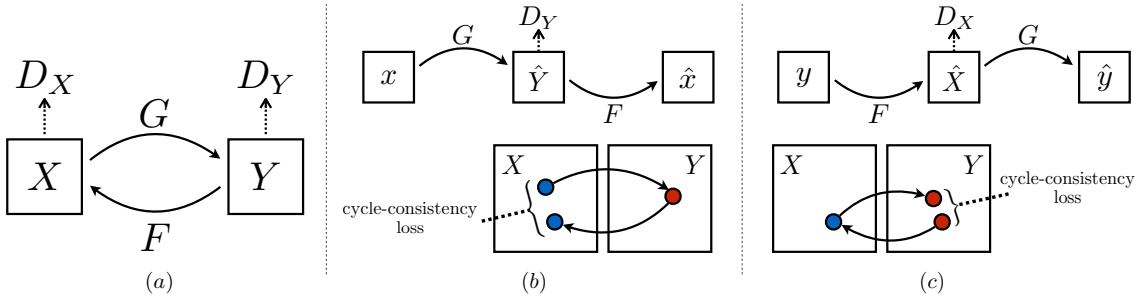


Figure 6.5: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

6.4 Unpaired Image-to-Image Translation

Different from the paired image-to-image translation described in Section 6.3, our goal here is to learn mapping functions between two domains X and Y given training samples $\{x_i\}_{i=1}^N$ where $x_i \in X$ and $\{y_j\}_{j=1}^M$ where $y_j \in Y$[1]. Note that no paired data is provided for the model training. We denote the data distribution as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. As illustrated in Figure 6.5 (a), our model includes two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$. In addition, we introduce two adversarial discriminators D_X and D_Y, where D_X aims to distinguish between images $\{x\}$ and translated images $\{F(y)\}$; in the same way, D_Y aims to discriminate between $\{y\}$ and $\{G(x)\}$. Our objective contains two types of terms: *adversarial losses* [67] for matching the distribution of generated images to the data distribution in the target domain; and *cycle consistency losses* to prevent the learned mappings G and F from contradicting each other.

[1]We often omit the subscript i and j for simplicity.

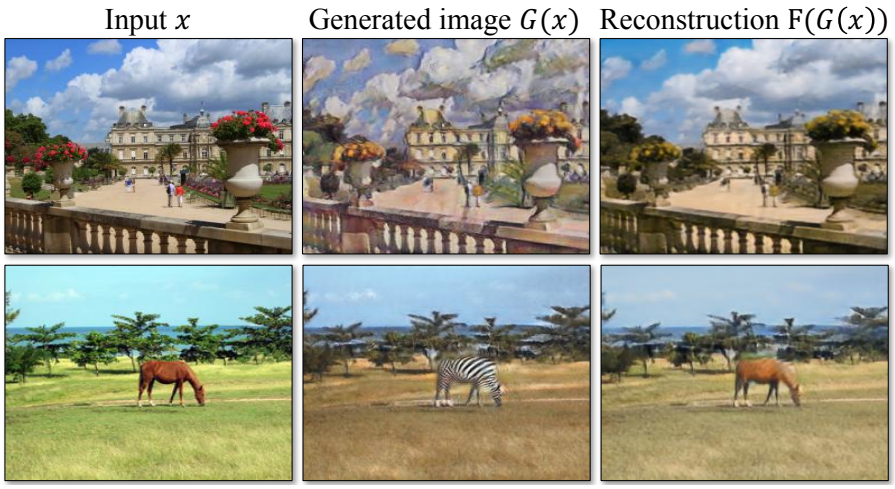


Figure 6.6: The generated images $G(x)$ and the reconstructed images $F(G(x))$ from various experiments. From top to bottom: photo↔Cezanne and horses↔zebras.

6.4.1 Adversarial Loss

We apply adversarial losses [67] to both mapping functions. For the mapping function $G : X \to Y$ and its discriminator D_Y, we express the objective as:

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))], \quad (6.4)$$

where G tries to generate images $G(x)$ that look similar to images from domain Y, while D_Y aims to distinguish between translated samples $G(x)$ and real samples y. G aims to minimize this objective against an adversary D that tries to maximize it, i.e. $\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$. We introduce a similar adversarial loss for the mapping function $F : Y \to X$ and its discriminator D_X as well: i.e. $\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$.

6.4.2 Cycle Consistency Loss

Adversarial training can, in theory, learn mappings G and F that produce outputs identically distributed as target domains Y and X respectively (strictly speaking, this requires G and F to be stochastic functions) [65]. However, with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution. Thus, adversarial losses

alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i. To further reduce the space of possible mapping functions, we argue that the learned mapping functions should be cycle-consistent: as shown in Figure 6.5 (b), for each image x from domain X, the image translation cycle should be able to bring x back to the original image, i.e. $x \to G(x) \to F(G(x)) \approx x$. We call this *forward cycle consistency*. Similarly, as illustrated in Figure 6.5 (c), for each image y from domain Y, G and F should also satisfy *backward cycle consistency*: $y \to F(y) \to G(F(y)) \approx y$. We can incentivize this behavior using a *cycle consistency loss*:

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \quad (6.5)$$

In preliminary experiments, we also tried replacing the L1 norm in this loss with an adversarial loss between $F(G(x))$ and x, and between $G(F(y))$ and y, but did not observe improved performance.

The behavior induced by the cycle consistency loss can be observed in Figure 6.6: the reconstructed images $F(G(x))$ end up matching closely to the input images x.

6.4.3 Full Objective

Our full objective is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F), \quad (6.6)$$

where λ controls the relative importance of the two objectives. We aim to solve:

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y). \quad (6.7)$$

Notice that our model can be viewed as training two "autoencoders" [83]: we learn one autoencoder $F \circ G : X \to X$ jointly with another $G \circ F : Y \to Y$. However, these autoencoders each have special internal structure: they map an image to itself via an intermediate representation that is a translation of the image into another domain. Such a setup can also be seen as a special case of "adversarial autoencoders" [137], which use an adversarial loss to train the bottleneck layer of an autoencoder to match an arbitrary target distribution. In our case, the target distribution for the $X \to X$ autoencoder is that of domain Y.

In Section 6.6.1, we compare our method against ablations of the full objective, including the adversarial loss $\mathcal{L}_{\text{GAN}}$ alone and the cycle consistency loss $\mathcal{L}_{\text{cyc}}$ alone, and empirically show that both objectives play critical roles in arriving at high-quality results. We also evaluate our method with only cycle loss in one direction, and show that a single cycle is not sufficient to regularize the training for this under-constrained problem.

6.5 Implementation

Network Architecture We adapt the architecture for our generative networks from Johnson et al. [99] who have shown impressive results for neural style transfer and super-resolution. This network contains two stride-2 convolutions, several residual blocks [79], and two fractionally-strided convolutions with stride $\frac{1}{2}$. We use 6 blocks for 128×128 images, and 9 blocks for 256×256 and higher-resolution training images. Similar to Johnson et al. [99], we use instance normalization [209]. For the discriminator networks we use 70×70 PatchGANs [93,125,129], which aim to classify whether 70×70 overlapping image patches are real or fake. Such a patch-level discriminator architecture has fewer parameters than a full-image discriminator, and can be applied to arbitrarily-sized images in a fully convolutional fashion [93].

Training details We apply two techniques from recent works to stabilize our model training procedure. First, for $\mathcal{L}_{GAN}$ (Equation 6.4), we replace the negative log likelihood objective by a least-squares loss [138]. This loss is more stable during training and generates higher quality results. In particular, for a GAN loss $\mathcal{L}_{\text{GAN}}(G, D, X, Y)$, we train the G to minimize $\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$ and train the D to minimize $\mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2]$.

Second, to reduce model oscillation [65], we follow Shrivastava et al's strategy [187] and update the discriminators using a history of generated images rather than the ones produced by the latest generative networks. We keep an image buffer that stores the 50 previously generated images.

For all the experiments, we set $\lambda = 10$ in Equation 6.6. We use the Adam solver [112] with a batch size of 1. All networks were trained from scratch with a learning rate of 0.0002. We keep the same learning rate for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs. Please see our online arXiv paper `https://arxiv.org/abs/1703.10593` for more details about the datasets, architectures, and training procedures.

6.6 Results

We first compare our approach against recent methods for unpaired image-to-image translation on paired datasets where ground truth input-output pairs are available for evaluation. We then study the importance of both the adversarial loss and the cycle consistency loss, and compare our full method against several variants. Finally, we demonstrate the generality of our algorithm on a wide range of applications where paired data does not exist. For brevity, we refer to our method as `CycleGAN`. The

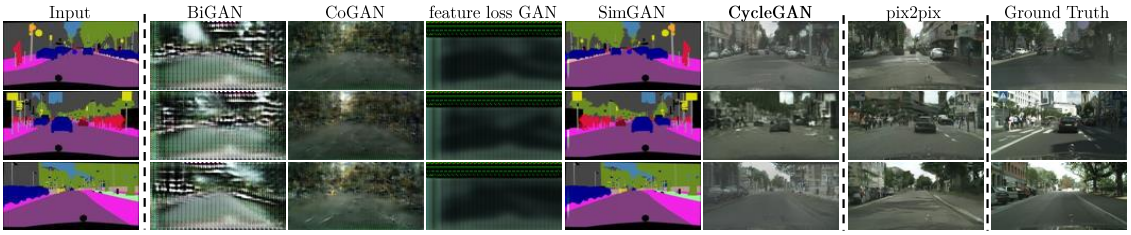


Figure 6.7: Different methods for mapping labels↔photos trained on Cityscapes images. From left to right: input, BiGAN/ALI [42, 47], CoGAN [134], feature loss + GAN, SimGAN [187], CycleGAN (ours), pix2pix [93] trained on paired data, and ground truth.

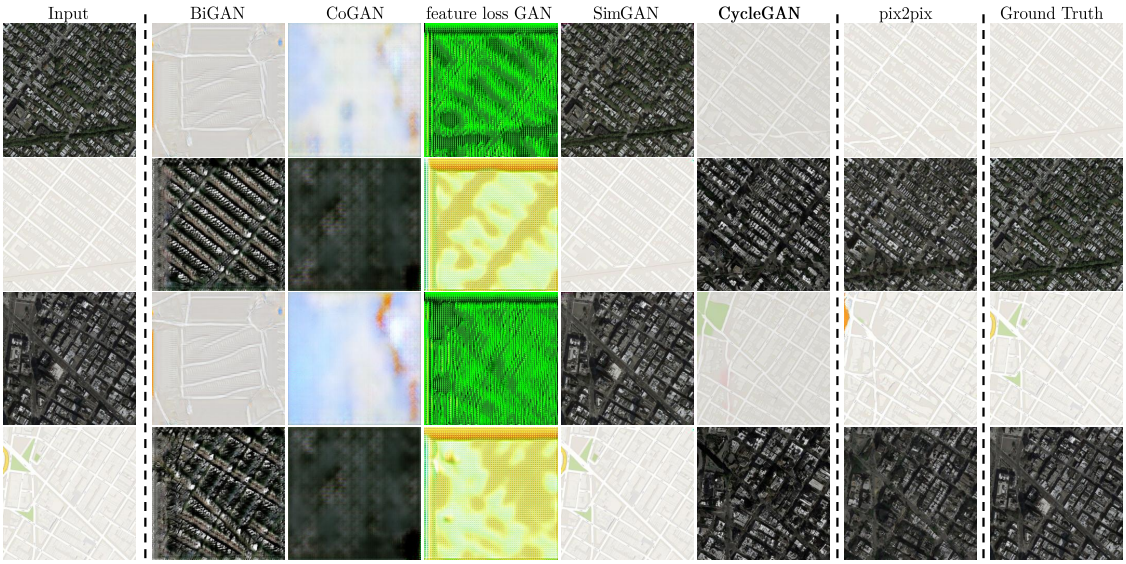


Figure 6.8: Different methods for mapping aerial photos↔maps on Google Maps. From left to right: input, BiGAN/ALI [42, 47], CoGAN [134], feature loss + GAN, SimGAN [187], CycleGAN (ours), pix2pix [93] trained on paired data, and ground truth.

code and full results can be viewed at https://github.com/junyanz/CycleGAN.

6.6.1 Evaluation

Using the same evaluation datasets and metrics as "pix2pix" [93], we compare our method against several baselines both qualitatively and quantitatively. The tasks include semantic labels↔photo on the Cityscapes dataset [31], and map↔aerial photo on data scraped from Google Maps. We also perform ablation study on the full loss function.

Metrics

AMT perceptual studies On the map↔aerial photo task, we run "real vs fake" perceptual studies on Amazon Mechanical Turk (AMT) to assess the realism of our outputs. We follow the same perceptual study protocol from Isola et al. [93], except we only gather data from 25 participants per algorithm we tested. Participants were shown a sequence of pairs of images, one a real photo or map and one fake (generated by our algorithm or a baseline), and asked to click on the image they thought was real. The first 10 trials of each session were practice and feedback was given as to whether the participant's response was correct or incorrect. The remaining 40 trials were used to assess the rate at which each algorithm fooled participants. Each session only tested a single algorithm, and participants were only allowed to complete a single session. Note that the numbers we report here are not directly comparable to those in [93] as our ground truth images were processed slightly differently [2] and the participant pool we tested may be differently distributed from those tested in [93] (due to running the experiment at a different date and time). Therefore, our numbers should only be used to compare our current method against the baselines (which were run under identical conditions), rather than against [93].

FCN score Although perceptual studies may be the gold standard for assessing graphical realism, we also seek an automatic quantitative measure that does not require human experiments. For this we adopt the "FCN score" from [93], and use it to evaluate the Cityscapes labels→photo task. The FCN metric evaluates how interpretable the generated photos are according to an off-the-shelf semantic segmentation algorithm (the fully-convolutional network, FCN, from [136]). The FCN predicts a label map for a generated photo. This label map can then be compared against the input ground truth labels using standard semantic segmentation metrics described below. The intuition is that if we generate a photo from a label map of

[2] We train all the models on 256×256 images while in pix2pix [93], the model was trained on 256×256 patches of 512×512 images, and run convolutionally on the 512×512 images at test time. We choose 256×256 in our experiments as many baselines cannot scale up to high resolution images, and CoGAN cannot be tested fully convolutionally.

"car on road", then we have succeeded if the FCN applied to the generated photo detects "car on road".

Semantic segmentation metrics To evaluate the performance of photo→labels, we use the standard metrics from the Cityscapes benchmark, including per-pixel accuracy, per-class accuracy, and mean class Intersection-Over-Union (Class IOU) [31].

Baselines

CoGAN [134] This method learns one GAN generator for domain X and one for domain Y, with tied weights on the first few layers for shared latent representation. Translation from X to Y can be achieved by finding a latent representation that generates image X and then rendering this latent representation into style Y.

SimGAN [187] Like our method, Shrivastava et al. [187] uses an adversarial loss to train a translation from X to Y. The regularization term $\|X - G(X)\|_1$ was used to penalize making large changes at pixel level.

Feature loss + GAN We also test a variant of SimGAN [187] where the L1 loss is computed over deep image features using a pretrained network (VGG-16 `relu4_2` [189]), rather than over RGB pixel values. Computing distances in deep feature space, like this, is also sometimes referred to as using a "perceptual loss" [44, 99].

BiGAN/ALI [42, 47] Unconditional GANs [67] learn a generator $G : Z \to X$, that maps random noise Z to images X. The BiGAN [47] and ALI [42] propose to also learn the inverse mapping function $F : X \to Z$. Though they were originally designed for mapping a latent vector z to an image x, we implemented the same objective for mapping a source image x to a target image y.

pix2pix [93] We also compare against pix2pix [93], which is trained on paired data, to see how close we can get to this "upper bound" without using any paired training data.

For a fair comparison, we implement all the baselines using the same architecture and details as our method, except for CoGAN [134]. CoGAN builds on generators that produce images from a shared latent representation, which is incompatible with our image-to-image network. We use the public implementation of CoGAN instead [3].

Comparison against baselines As can be seen in Figure 6.7 and Figure 6.8, we were unable to achieve compelling results with any of the baselines. Our method, on the other hand, is able to produce translations that are often of similar quality to the fully supervised pix2pix.

Table 6.1 reports performance regarding the AMT perceptual realism task. Here, we see that our method can fool participants on around a quarter of trials, in both

[3] https://github.com/mingyuliutw/CoGAN

Loss	Map → Photo % Turkers labeled *real*	Photo → Map % Turkers labeled *real*
CoGAN [134]	0.6% ± 0.5%	0.9% ± 0.5%
BiGAN/ALI [42, 47]	2.1% ± 1.0%	1.9% ± 0.9%
SimGAN [187]	0.7% ± 0.5%	2.6% ± 1.1%
Feature loss + GAN	1.2% ± 0.6%	0.3% ± 0.2%
CycleGAN (ours)	**26.8% ± 2.8%**	**23.2% ± 3.4%**

Table 6.1: AMT "real vs fake" test on maps↔aerial photos at 256 × 256 resolution.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [134]	0.40	0.10	0.06
BiGAN/ALI [42, 47]	0.19	0.06	0.02
SimGAN [187]	0.20	0.10	0.04
Feature loss + GAN	0.06	0.04	0.01
CycleGAN (ours)	**0.52**	**0.17**	**0.11**
pix2pix [93]	0.71	0.25	0.18

Table 6.2: FCN-scores for different methods, evaluated on Cityscapes labels→photo.

the maps→aerial photos direction and the aerial photos→maps direction at 256 × 256 resolution[4]. All baselines almost never fooled participants.

Table 6.2 assesses the performance of the labels→photo task on the Cityscapes and Table 6.3 assesses the opposite mapping (photos→labels). In both cases, our method again outperforms the baselines.

Analysis of the loss function In Table 6.4 and Table 6.5, we compare against ablations of our full loss. Removing the GAN loss substantially degrades results, as

[4]We also train CycleGAN and pix2pix at 512 × 512 resolution, and observe the comparable performance: maps→aerial photos: CycleGAN: 37.5% ± 3.6% and pix2pix: 33.9% ± 3.1%; aerial photos→maps: CycleGAN: 16.5% ± 4.1% and pix2pix: 8.5% ± 2.6%

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [134]	0.45	0.11	0.08
BiGAN/ALI [42, 47]	0.41	0.13	0.07
SimGAN [187]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**
pix2pix [93]	0.85	0.40	0.32

Table 6.3: Classification performance of photo→labels for different methods on cityscapes.

Figure 6.9: Different variants of our method for mapping labels↔photos trained on cityscapes. From left to right: input, cycle-consistency loss alone, adversarial loss alone, GAN + forward cycle-consistency loss ($F(G(x)) \approx x$), GAN + backward cycle-consistency loss ($G(F(y)) \approx y$), CycleGAN (our full method), and ground truth. Both *Cycle alone* and *GAN + backward* fail to produce images similar to the target domain. *GAN alone* and *GAN + forward* suffer from mode collapse, producing identical label maps regardless of the input photo.

Loss	**Per-pixel acc.**	**Per-class acc.**	**Class IOU**
Cycle alone	0.22	0.07	0.02
GAN alone	0.51	0.11	0.08
GAN + forward cycle	**0.55**	**0.18**	**0.12**
GAN + backward cycle	0.39	0.14	0.06
CycleGAN (ours)	0.52	0.17	0.11

Table 6.4: Ablation study: FCN-scores for different variants of our method, evaluated on Cityscapes labels→photo.

does removing the cycle-consistency loss. We therefore conclude that both terms are critical to our results. We also evaluate our method with the cycle loss in only one direction: GAN+forward cycle loss $\mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1]$, or GAN+backward cycle loss $\mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]$ (Equation 6.5) and find that it often incurs training instability and causes mode collapse, especially for the direction of the mapping that was removed. Figure 6.9 shows several qualitative examples.

Image reconstruction quality In Figure 6.6, we show a few random samples of the reconstructed images $F(G(x))$. We observed that the reconstructed images were very close to the original inputs x, at both training and testing time, even in cases where one domain represents significantly more diverse information, such as map↔aerial photos.

Additional results on paired datasets Figure 6.10 shows some example results on other paired datasets used in "pix2pix" [93], such as architectural labels↔photos

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Cycle alone	0.10	0.05	0.02
GAN alone	0.53	0.11	0.07
GAN + forward cycle	0.49	0.11	0.07
GAN + backward cycle	0.01	0.06	0.01
CycleGAN (ours)	**0.58**	**0.22**	**0.16**

Table 6.5: Ablation study: classification performance of photo→labels for different losses, evaluated on Cityscapes.

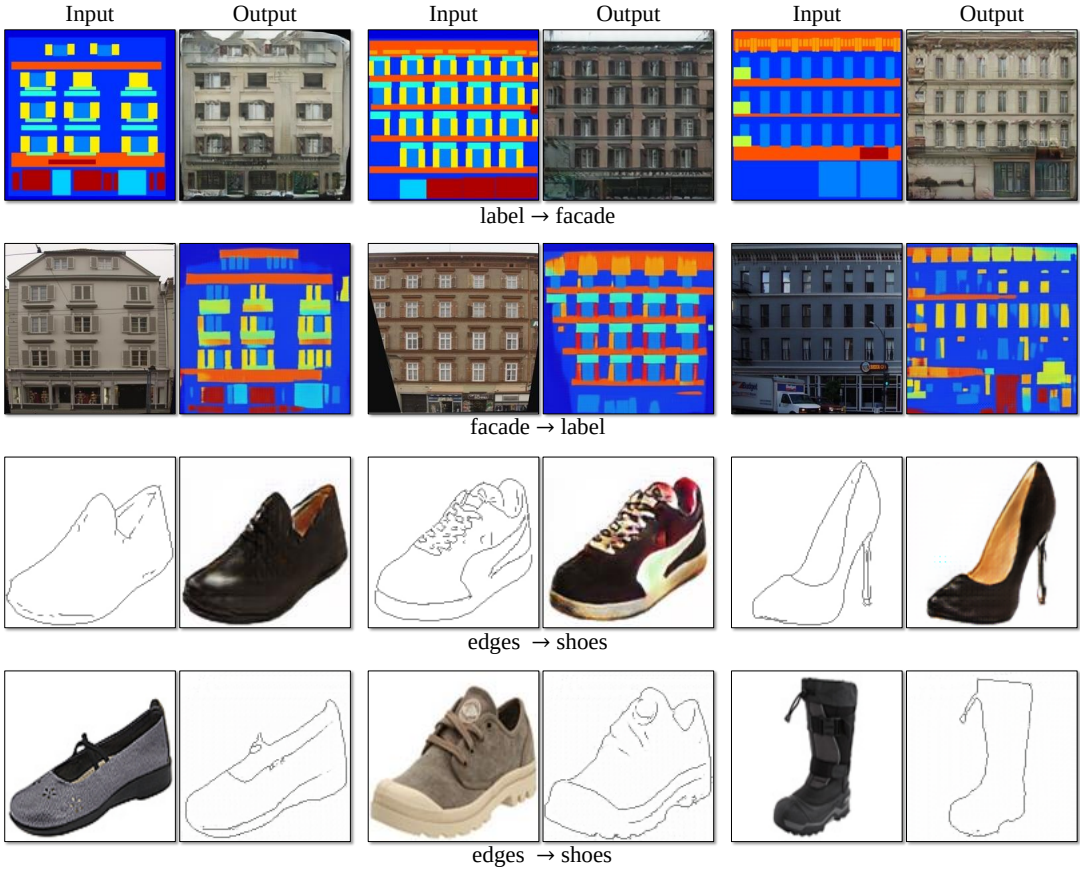


Figure 6.10: Example results of CycleGAN on paired datasets used in "pix2pix" [93] such as architectural labels↔photos and edges↔shoes.

from the CMP Facade Database [163], and edges↔shoes from the UT Zappos50K dataset [232]. The image quality of our results is close to those produced by the fully supervised pix2pix while our method learns the mapping without paired supervision.

6.6.2 Applications

We demonstrate our method on several applications where paired training data does not exist. Please refer to our online arXiv version https://arxiv.org/abs/1703.10593 for more details about the datasets. We observe that translations on training data are often more appealing than those on test data, and full results of all applications on both training and test data can be viewed on our project website.

Collection style transfer (Figure 6.12 and Figure 6.13) We train the model on landscape photographs downloaded from Flickr and WikiArt. Note that unlike recent work on "neural style transfer" [61], our method learns to mimic the style of an entire *collection* of artworks, rather than transferring the style of a single selected piece of art. Therefore, we can learn to generate photos in the style of, e.g., Van Gogh, rather than just in the style of Starry Night. The size of the dataset for each artist/style was 526, 1073, 400, and 563 for Cezanne, Monet, Van Gogh, and Ukiyo-e.

Object transfiguration (Figure 6.15) The model is trained to translate one object class from ImageNet [36] to another (each class contains around 1000 training images). Turmukhambetov et al. [206] proposes a subspace model to translate one object into another object of the same category, while our method focuses on object transfiguration between two visually similar categories.

Season transfer (Figure 6.15) The model is trained on 854 winter photos and 1273 summer photos of Yosemite downloaded from Flickr.

Photo generation from paintings (Figure 6.14) For painting→photo, we find that it is helpful to introduce an additional loss to encourage the mapping to preserve color composition between the input and output. In particular, we adopt the technique of Taigman et al. [200] and regularize the generator to be near an identity mapping when real samples of the target domain are provided as the input to the generator: i.e. $\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(x) - x\|_1]$.

Without $\mathcal{L}_{\text{identity}}$, the generator G and F are free to change the tint of input images when there is no need to. For example, when learning the mapping between Monet's paintings and Flickr photographs, the generator often maps paintings of daytime to photographs taken during sunset, because such a mapping may be equally valid under the adversarial loss and cycle consistency loss. The effect of this *identity mapping loss* are shown in Figure 6.11.

In Figure 6.14, we show additional results translating Monet's paintings to photographs. This figure and Figure 6.11 show results on paintings that were included in the *training set*, whereas for all other experiments in the chapter, we only evaluate and show test set results. Because the training set does not include paired data, coming up with a plausible translation for a training set painting is a nontrivial

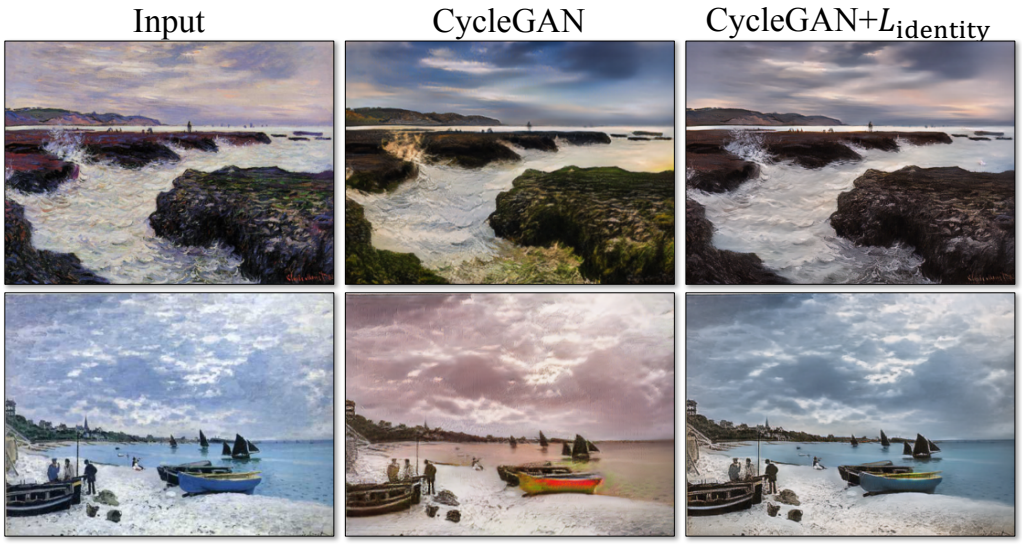


Figure 6.11: The effect of the *identity mapping loss* on Monet's painting→ photos. From left to right: input paintings, CycleGAN without identity mapping loss, CycleGAN with identity mapping loss. The identity mapping loss helps preserve the color of the input paintings.

task. Indeed, since Monet is no longer able to create new paintings, generalization to unseen, "test set", paintings is not a pressing problem.

Photo enhancement (Figure 6.16) We show that our method can be used to generate photos with shallower depth of field. We train the model on flower photos downloaded from Flickr. The source domain consists of photos of flower taken by smartphones, which usually have deep DoF due to small aperture. The target contains photos captured by DSLRs with larger aperture. Our model successfully generates photos with shallower depth of field from the photos taken by smartphones.

Comparison with Gatys et al. [61] In Figure 6.17, we compare our results with neural style transfer [61] on photo stylization. For each row, we first use two representative artworks as the style images for [61]. Our method, on the other hand, is able to produce photos in the style of entire *collection*. To compare against neural style transfer of an entire collection, we compute the average Gram Matrix across the target domain, and use this matrix to transfer the "average style" using [61].

Figure 6.18 demonstrates similar comparisons for other translation tasks. We observe that Gatys et al. [61] requires finding target style images that closely match the desired output, but still often fails to produce photo-realistic results, while our method succeeds to generate natural looking results, similar to the target domain.

6.7 Discussion

Although our method can achieve compelling results in many cases, the results are far from uniformly positive. Several typical failure cases are shown in Figure 6.19. On translation tasks that involve color and texture changes, like many of those reported above, the method often succeeds. We have also explored tasks that require geometric changes, with little success. For example, on the task of dog→cat transfiguration, the learned translation degenerates to making minimal changes to the input (Figure 6.19). This might be caused by our generator architecture choices which are tailored for good performance on the appearance changes. Handling more varied and extreme transformations, especially geometric changes, is an important problem for future work.

Some failure cases are caused by the distribution characteristics of the training datasets. For example, the horse → zebra example (Figure 6.19, right) has got confused, because our model was trained on the *wild horse* and *zebra* synsets of ImageNet, which does not contain images of a person riding a horse or zebra.

We also observe a lingering gap between the results achievable with paired training data and those achieved by our unpaired method. In some cases, this gap may be very hard – or even impossible – to close: for example, our method sometimes permutes the labels for tree and building in the output of the photos→labels task. To resolve this ambiguity may require some form of weak semantic supervision. Integrating weak or semi-supervised data may lead to substantially more powerful translators, still at a fraction of the annotation cost of the fully-supervised systems.

Nonetheless, in many cases completely unpaired data is plentifully available and should be made use of. This chapter pushes the boundaries of what is possible in this "unsupervised" setting.

While most of the current image-to-image translation methods focus on 2D images, in the future, we plan to explore the similar idea on other types of visual data such as light fields [103, 218] and videos [55, 219], which require additional constraints like temporal consistency and spatial-angular photo-consistency.

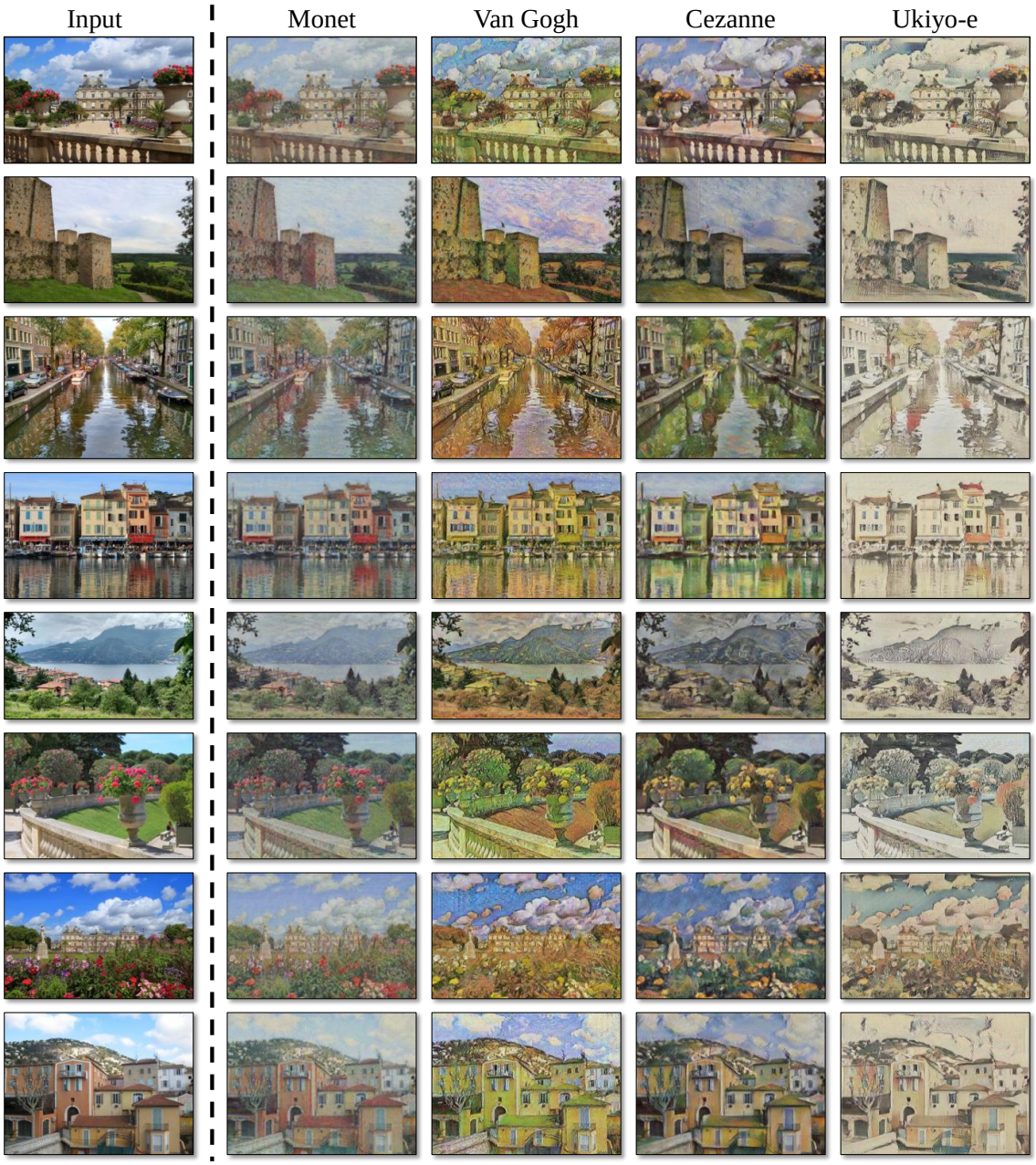


Figure 6.12: Collection style transfer I: we transfer input images into the artistic styles of Monet, Van Gogh, Cezanne, and Ukiyo-e. Please see our website for additional examples.

Figure 6.13: Collection style transfer II: we transfer input images into the artistic styles of Monet, Van Gogh, Cezanne, Ukiyo-e. Please see our website for additional examples.

Figure 6.14: Relatively successful results on mapping Monet's paintings to photographs. Please see our website for additional examples.

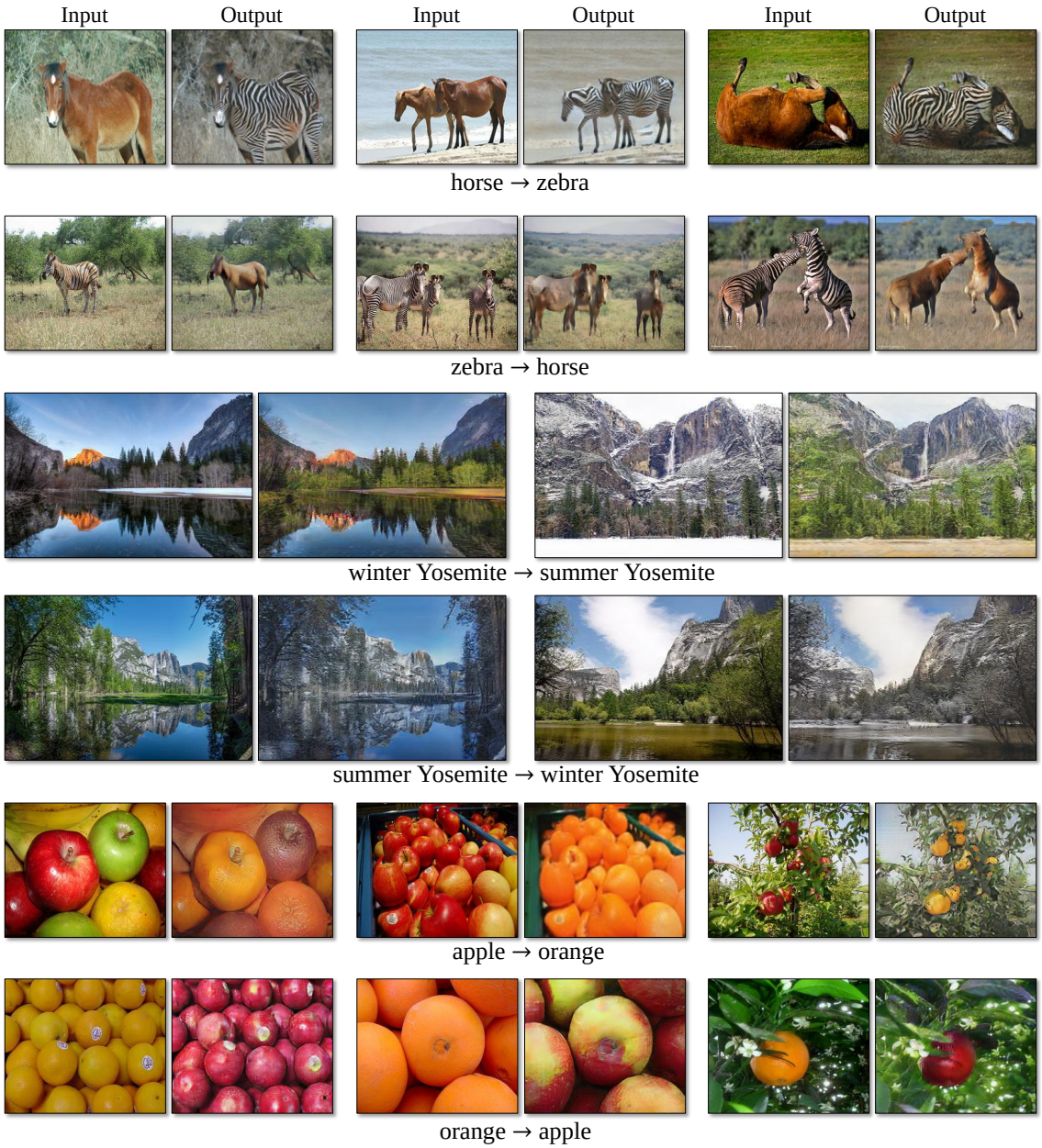


Figure 6.15: Our method applied to several translation problems. These images are selected as relatively successful results – please see our website for more comprehensive and random results. In the top two rows, we show results on object transfiguration between horses and zebras, trained on 939 images from the *wild horse* class and 1177 images from the *zebra* class in Imagenet [36]. Also check out the horse→zebra demo video at https://youtu.be/9reHvktowLY. The middle two rows show results on season transfer, trained on winter and summer photos of Yosemite from Flickr. In the bottom two rows, we train our method on 996 *apple* images and 1020 *navel orange* images from ImageNet.

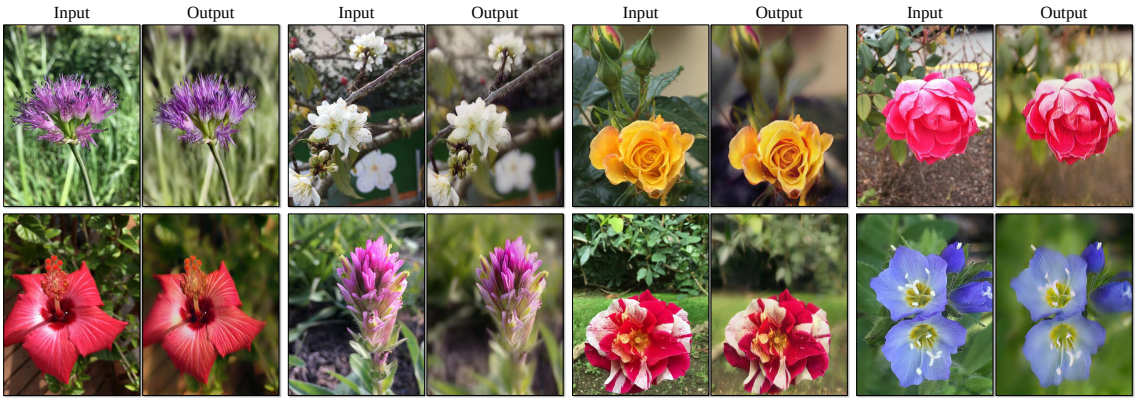


Figure 6.16: Photo enhancement: mapping from a set of iPhone snaps to professional DSLR photographs, the system often learns to produce shallow focus. Here we show some of the most successful results in our test set – average performance is considerably worse. Please see our website for more comprehensive and random examples.

Figure 6.17: We compare our method with neural style transfer [61] on photo stylization. Left to right: input image, results from [61] using two different representative artworks as style images, results from [61] using the entire collection of the artist, and CycleGAN (ours).

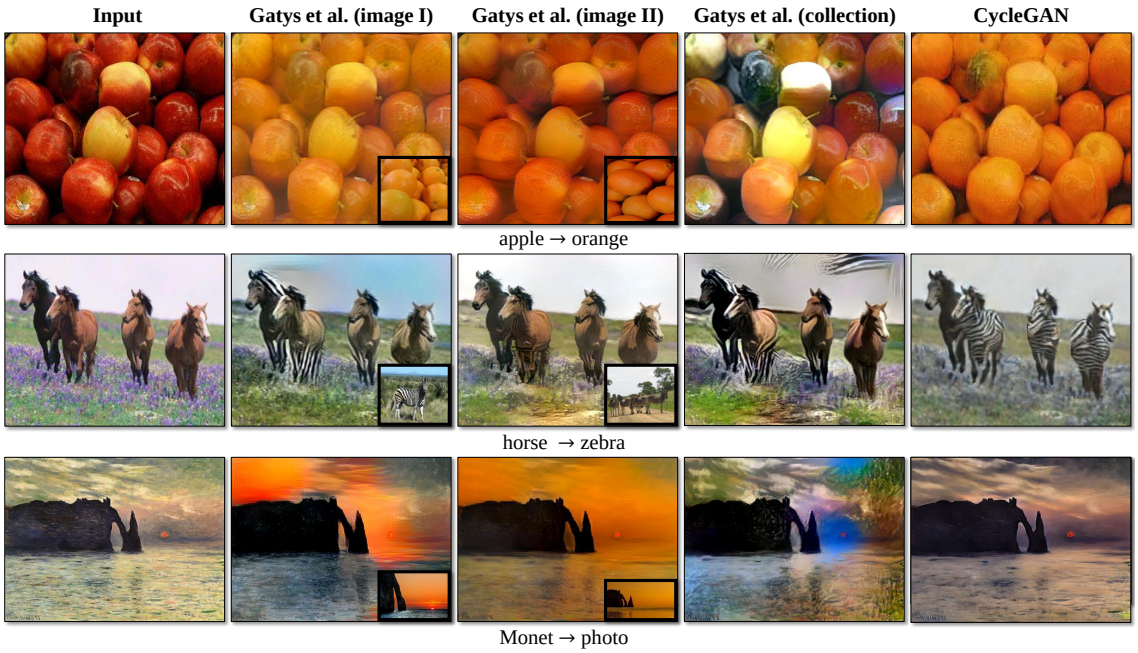


Figure 6.18: We compare our method with neural style transfer [61] on various applications. From top to bottom: apple→orange, horse→zebra, and Monet→photo. Left to right: input image, results from [61] using two different images as style images, results from [61] using all the images from the target domain, and CycleGAN (ours).

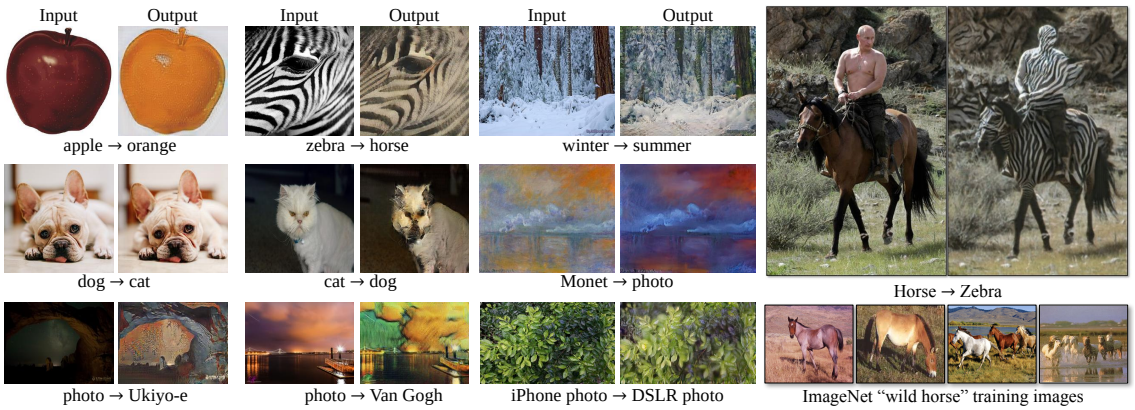


Figure 6.19: Typical failure cases of our method. Please see our website for more comprehensive results.

Chapter 7

Multimodal Image-to-Image Translation

The above method can only produce one plausible result given an input image. However, many image-to-image translation problems are inherently ambiguous, as a single input image may correspond to multiple possible outputs. In this chapter, we aim to model a *distribution* of possible outputs, in a conditional generative modeling setting. The ambiguity of the mapping is distilled into a low-dimensional latent vector, which can be randomly sampled at test time. A generator learns to map the given input, combined with this latent code, to the output. We explicitly encourage the connection between output and the latent code to be invertible. This helps prevent a many-to-one mapping from the latent code to the output during training, also known as the problem of mode collapse, and produces diverse results. We explore several variants of this approach by employing different training objectives, network architectures, and methods of injecting the latent code. Our proposed method encourages bijective consistency between the latent encoding and output modes. We present a systematic comparison of our method and other variants on both perceptual realism and diversity.

7.1 Introduction

Deep learning techniques have made rapid progress in conditional image generation. For example, networks have been used to inpaint missing image regions [93, 155, 229], create sentence conditioned generations [235], add color to grayscale images [91, 93, 122, 237], and sketch-to-photo [93, 179]. However, most techniques in this space have focused on generating a *single* result conditioned on the input. In

Figure 7.1: Multimodal image-to-image translation using our proposed method: given an input image from one domain (night image of a scene), we aim to model a *distribution* of potential outputs (corresponding day images) in the target domain, producing both realistic and diverse results.

this work, our focus is to model a *distribution* of potential results, as many of these problems may be multimodal or ambiguous in nature. For example, in a night-to-day translation task (see Figure 7.1), an image captured at night may correspond to many possible day images with different types of lighting, sky and clouds. There are two main goals of the conditional generation problem: producing results which are (1) perceptually realistic and (2) diverse, while remaining faithful to the input. This multimodal mapping from a high-dimensional input to a distribution of high-dimensional outputs makes the conditional generative modeling task challenging. In existing approaches, this leads to the common problem of mode collapse [65], where the generator learns to generate only a small number of unique outputs. We systematically study a family of solutions to this problem, which learn a low-dimensional latent code for aspects of possible outputs which are not contained in the input image. The generator then produces an output conditioned on both the given input and this learned latent code.

We start with the `pix2pix` framework [93] which has previously been shown to produce good-quality results for a variety of image-to-image translation tasks. The trains a generator network, conditioned on the input image, with two losses: (1) a regression loss to produce a similar output to the known paired ground truth image and (2) a learned discriminator loss to encourage realism. The authors note that trivially appending a randomly drawn latent code did not help produce diverse results, and using random dropout at test time only helped marginally. Instead, we propose encouraging a bijection between the output and latent space. Not only do we perform the direct task of mapping from the input and latent code to the output,

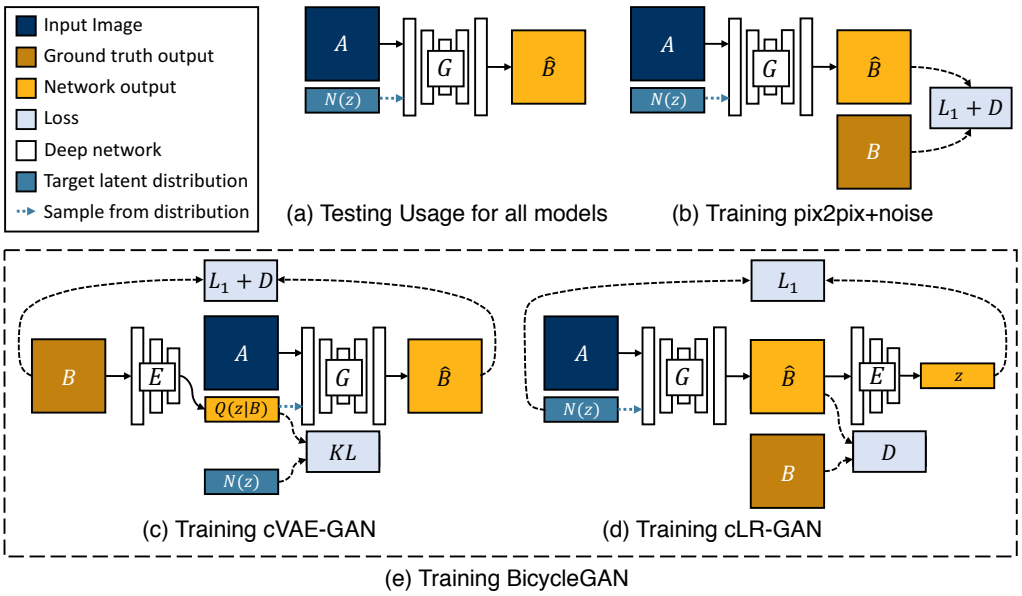


Figure 7.2: Overview: (a) Test time usage of all the methods. To produce a sample output, a latent code $\mathbf{z}$ is first randomly sampled from a known distribution (e.g., a standard normal distribution). A generator G maps input image $\mathbf{A}$ (blue) and latent sample $\mathbf{z}$ to produce output sample $\hat{\mathbf{B}}$ (yellow). (b) `pix2pix+noise` [93] baseline, with additional input $\mathbf{B}$ (brown) that corresponds to $\mathbf{A}$. (c) `cVAE-GAN` (and `cAE-GAN`) start from ground truth target image $\mathbf{B}$ and encode it into the latent space. The generator then attempts to map the input image $\mathbf{A}$ along with a sampled $\mathbf{z}$ back into original image $\mathbf{B}$. (d) `cLR-GAN` randomly samples a latent code from a known distribution, uses it to map $\mathbf{A}$ into the output $\hat{\mathbf{B}}$, and then tries to reconstruct the latent code from it. (e) Our hybrid `BicycleGAN` method combines constraints in both directions.

we also jointly learn an encoder from the output back to the latent space. This discourages two different latent codes from generating the same output (non-injective mapping). During training, the learned encoder is trained to find a latent code vector that corresponds to the ground truth output image, while passing enough information to the generator to resolve any ambiguities about the mode of output. For example, when generating a day image from a night one (Figure 7.1), the latent vector may encode information about the sky color, lighting effects on the ground and the density and shape of clouds. In both cases, applying the encoder and generator, in either order, should be consistent, resulting in either the same latent code or the same image.

In this work, we instantiate this idea by exploring several objective functions inspired by the literature in unconditional generative modeling:

- `cVAE-GAN` (Conditional Variational Autoencoder GAN): One popular approach to model multimodal output distribution is by learning the distribution of latent encoding given the output as popularized by VAEs [113]. In the conditional setup (similar to unconditional analogue [121]), we enforce that the latent distribution encoded by the desired output image maps back to the output via conditional generator. The latent distribution is regularized using KL-divergence to be close to a standard normal distribution so as to sample random codes during inference. This variational objective is then optimized jointly with the discriminator loss.

- `cLR-GAN` (Conditional Latent Regressor GAN): Another approach to enforce mode-capture in latent encoding is to explicitly model the inverse mapping. Starting from a randomly sampled latent encoding, the conditional generator should result into an output which when given itself as input to the encoder should result back into the same latent code, enforcing self-consistency. This method could be seen as a conditional formulation of the "latent regressor" model [42, 47] and also related to InfoGAN [30].

- `BicycleGAN`: Finally, we combine both these approaches to enforce the connection between latent encoding and output in both directions *jointly* and achieve improved performance. We show that our method can produce both diverse and visually appealing results across a wide range of image-to-image translation problems, significantly more diverse than other baselines, including naively adding noise in the `pix2pix` framework. In addition to the loss function, we study the performance with respect to several encoder networks, as well as different ways of injecting the latent code into the generator network.

We perform a systematic evaluation of these variants by using real humans to judge photo-realism and an automated distance metric to assess output diversity. Code and data are available at `https://github.com/junyanz/BicycleGAN`.

7.2 Background

Generative modeling Parametric modeling of the natural image distribution is a challenging problem. Classically, this problem has been tackled using autoencoders [83, 212] or Restricted Boltzmann machines [192]. Variational autoencoders [113] provide an effective approach to model stochasticity within the network

by reparametrization of a latent distribution. Other approaches to modeling stochasticity include autoregressive models [150, 151] which can model multimodality via sequential conditional prediction. These approaches are trained with a pixel-wise independent loss on samples of natural images using maximum likelihood and stochastic back-propagation. This is a disadvantage because two images, which are close regarding a pixel-wise independent metric, may be far apart on the manifold of natural images. Generative adversarial networks [67] overcome this issue by learning the loss function using a discriminator network, and have recently been very successful [7, 30, 37, 42, 47, 162, 164, 235, 240, 247]. Our method builds on the conditional version of VAE [113] and InfoGAN [30] or latent regressor [42] model via alternating joint optimization to learn diverse and realistic samples. We revisit this connection in Section 7.3.4.

Conditional image generation Potentially, all of the methods defined above could be easily conditioned. While conditional VAEs [194], PixelCNN [151], conditional autoregressive models [150, 151] have shown promise [71, 214, 228], image-to-image conditional GANs have lead to a substantial boost in the quality of the results. However, the quality has been attained at the expense of multimodality, as the generator learns to largely ignore the random noise vector when conditioned on a relevant context [93, 155, 179, 223, 229, 249]. In fact, it has even been shown that ignoring the noise leads to more stable training [93, 139, 155].

Explicitly-encoded multimodality One way to express multiple modes in the output is to encode them, conditioned on some mode-related context in addition to the input image. For example, color and shape scribbles and other interfaces were used as conditioning in iGAN [247], `pix2pix` [93], Scribbler [179] and interactive colorization [238]. While there has been some degree of success for generating multimodal outputs in unconditional and text-conditional setups [39, 67, 121, 146, 164], conditional image-to-image generation is still far from achieving the same results, unless explicitly encoded as discussed above. In this work, we learn conditional generation models for modeling multiple modes of output by enforcing tight connections in both image and latent space.

7.3 Multimodal Image-to-Image Translation

Our goal is to learn a multi-modal mapping between two image domains, for example, edges and photographs, or day and night images, etc. Consider the input domain $\mathcal{A} \subset \mathbb{R}^{H \times W \times 3}$ which is to be mapped to an output domain $\mathcal{B} \subset \mathbb{R}^{H \times W \times 3}$.

During training, we are given a dataset of paired instances from these domains, $\{(\mathbf{A} \in \mathcal{A}, \mathbf{B} \in \mathcal{B})\}$, which is representative of a joint distribution $p(\mathbf{A}, \mathbf{B})$. It is important to note that there could be multiple plausible paired instances $\mathbf{B}$ that would correspond to an input instance $\mathbf{A}$, but the training dataset usually contains only one such pair. However, given a new instance $\mathbf{A}$ during test time, our model should be able to generate a diverse set of output $\widehat{\mathbf{B}}$'s, corresponding to different modes in the distribution $p(\mathbf{B}|\mathbf{A})$.

While conditional GANs have achieved success in image-to-image translation tasks [93, 249], they are primarily limited to generating deterministic output $\widehat{\mathbf{B}}$ given the input image $\mathbf{A}$. On the other hand, we would like to learn the mapping that could sample the output $\widehat{\mathbf{B}}$ from true conditional distribution given $\mathbf{A}$, and produce results which are both diverse and realistic. To do so, we learn a low-dimensional latent space $\mathbf{z} \in \mathbb{R}^Z$, which encapsulates the ambiguous aspects of the output mode which are not present in the input image. For example, a sketch of a shoe could map to a variety of colors and textures, which could get compressed in this latent code. We then learn a deterministic mapping $G : (\mathbf{A}, \mathbf{z}) \rightarrow \mathbf{B}$ to the output. To enable stochastic sampling, we desire the latent code vector $\mathbf{z}$ to be drawn from some prior distribution $p(\mathbf{z})$; we use a standard Gaussian distribution $\mathcal{N}(0, I)$ in this work.

We first discuss a simple extension of existing methods and discuss its strengths and weakness, motivating the development of our proposed approach in the subsequent subsections.

7.3.1 Baseline: `pix2pix+noise`

The recently proposed `pix2pix` model [93] has shown high quality results in image-to-image translation setting. It uses conditional adversarial networks [67, 143] to help produce perceptually realistic results. GANs train a generator G and discriminator D by formulating their objective as an adversarial game. The discriminator attempts to differentiate between real images from the dataset and fake samples produced by the generator. Randomly drawn noise $\mathbf{z}$ is added to attempt to induce stochasticity. We illustrate the formulation in Figure 7.2(b) and describe it below.

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{\mathbf{A},\mathbf{B}\sim p(\mathbf{A},\mathbf{B})}[\log(D(\mathbf{A}, \mathbf{B}))] + \mathbb{E}_{\mathbf{A}\sim p(\mathbf{A}),\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(\mathbf{A}, G(\mathbf{A}, \mathbf{z})))] \tag{7.1}$$

To encourage the output of the generator to match the input as well as stabilize the GANs training, we use an ℓ_1 loss between the output and the ground truth image.

$$\mathcal{L}_1^{\text{image}}(G) = \mathbb{E}_{\mathbf{A},\mathbf{B}\sim p(\mathbf{A},\mathbf{B}),\mathbf{z}\sim p(\mathbf{z})}||\mathbf{B} - G(\mathbf{A}, \mathbf{z})||_1 \tag{7.2}$$

The final loss function uses the GAN and ℓ_1 terms, balanced by λ.

$$G^* = \arg\min_G \max_D \quad \mathcal{L}_{\text{GAN}}(G, D) + \lambda \mathcal{L}_1^{\text{image}}(G) \tag{7.3}$$

In this scenario, there is no incentive for the generator to make use of the noise vector which encodes random information. It was also noted in the preliminary experiments in [93] that the generator simply ignored the added noise and hence the noise was removed in their final experiments. This observation is consistent with [139, 155] and the mode collapse phenomenon observed in unconditional cases [65, 177]. In this chapter, we explore different ways to explicitly enforce the generator to use the latent encoding by making it capture relevant information than being random.

7.3.2 Conditional Variational Autoencoder GAN: `cVAE-GAN`

One way to force the latent code $\mathbf{z}$ to be "useful" is to directly map the ground truth $\mathbf{B}$ to it using an encoding function E. The generator G then uses both the latent code and the input image $\mathbf{A}$ to synthesize the desired output $\widehat{\mathbf{B}}$. The overall model can be easily understood as the reconstruction of $\mathbf{B}$, with latent encoding $\widehat{\mathbf{z}}$ concatenated with the paired $\mathbf{A}$ in the middle – similar to an autoencoder [83]. This interpretation is better shown in Figure 7.2(c).

This approach has been successfully investigated in Variational Auto-Encoders [113] in the unconditional scenario without the adversarial objective. Extending it to conditional scenario, the distribution $Q(\mathbf{z}|\mathbf{B})$ of latent code $\mathbf{z}$ using the encoder E with a Gaussian assumption, $Q(\mathbf{z}|\mathbf{B}) \triangleq E(\mathbf{B})$. To reflect this, Equation 7.1 is modified to sampling $\mathbf{z} \sim E(\mathbf{B})$ using the re-parameterization trick, allowing direct back-propagation [113].

$$\mathcal{L}_{\text{GAN}}^{\text{VAE}} = \mathbb{E}_{\mathbf{A},\mathbf{B}\sim p(\mathbf{A},\mathbf{B})}[\log(D(\mathbf{A},\mathbf{B}))] + \mathbb{E}_{\mathbf{A},\mathbf{B}\sim p(\mathbf{A},\mathbf{B}),\mathbf{z}\sim E(\mathbf{B})}[\log(1 - D(\mathbf{A}, G(\mathbf{A},\mathbf{z})))] \tag{7.4}$$

We make the corresponding change in the ℓ_1 loss term in Equation 7.2 as well to obtain $\mathcal{L}_1^{\text{VAE}}(G) = \mathbb{E}_{\mathbf{A},\mathbf{B}\sim p(\mathbf{A},\mathbf{B}),\mathbf{z}\sim E(\mathbf{B})}||\mathbf{B} - G(\mathbf{A},\mathbf{z})||_1$. Further, the latent distribution encoded by $E(B)$ is encouraged to be close to random gaussian so as to sample $\mathbf{z}$ at inference (i.e., $\mathbf{B}$ is not known).

$$\mathcal{L}_{\text{KL}}(E) = \mathbb{E}_{\mathbf{B}\sim p(\mathbf{B})}[\mathcal{D}_{\text{KL}}(E(\mathbf{B})||\,\mathcal{N}(0, I))], \tag{7.5}$$

where $\mathcal{D}_{\text{KL}}(p||q) = -\int p(z) \log \frac{p(z)}{q(z)} dz$. This forms our `cVAE-GAN` objective, a conditional version of the VAE-GAN [121] as

$$G^* = \arg\min_{G,E} \max_D \quad \mathcal{L}_{\text{GAN}}^{\text{VAE}}(G, D, E) + \lambda \mathcal{L}_1^{\text{VAE}}(G, E) + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}(E). \tag{7.6}$$

As a baseline, we also consider the deterministic version of this approach, i.e., dropping KL-divergence and encoding $\mathbf{z} = E(\mathbf{B})$. We call it `cAE-GAN` and show comparison in the experiments. However, there is no guarantee in `cAE-GAN` on the distribution of the latent space $\mathbf{z}$, which makes test-time sampling of $\mathbf{z}$ difficult.

7.3.3 Conditional Latent Regressor GAN: `cLR-GAN`

We explore another method of enforcing the generator network to utilize the latent code embedding $\mathbf{z}$, while staying close to the actual test time distribution $p(\mathbf{z})$, but from the latent code's perspective. We start from the latent code $\mathbf{z}$, as shown in Figure 7.2(d), and then enforce that $E(G(\mathbf{A}, \mathbf{z}))$ map back to the randomly drawn latent code with an ℓ_1 loss. Note that the encoder E here is producing a point estimate for $\widehat{\mathbf{z}}$, whereas the encoder in the previous section was predicting a Gaussian distribution.

$$\mathcal{L}_1^{\text{latent}}(G, E) = \mathbb{E}_{\mathbf{A}\sim p(\mathbf{A}), \mathbf{z}\sim p(\mathbf{z})} ||\mathbf{z} - E(G(\mathbf{A}, \mathbf{z}))||_1 \tag{7.7}$$

We also include the discriminator loss $L_{\text{GAN}}(G, D)$ (Equation 7.1) on $\widehat{\mathbf{B}}$ to encourage the network to generate realistic results, and the full loss can be written as:

$$G^* = \arg\min_{G,E} \max_{D} \quad \mathcal{L}_{\text{GAN}}(G, D) + \lambda_{\text{latent}} \mathcal{L}_1^{\text{latent}}(G, E) \tag{7.8}$$

The ℓ_1 loss for the ground truth image $\mathbf{B}$ is not used. In this case, since the noise vector is randomly drawn, we do not want the predicted $\widehat{\mathbf{B}}$ to be the ground truth, but rather a realistic result. The above objective bears similarity to the "latent regressor" model [30, 42, 47], where the generated sample $\widehat{\mathbf{B}}$ is encoded to generate a latent vector.

7.3.4 Our Hybrid Model: `BicycleGAN`

We combine the `cVAE-GAN` and `cLR-GAN` objectives in a hybrid model. For `cVAE-GAN`, the encoding is learned from real data, but a random latent code may not yield realistic images at test time – the KL loss may not be well optimized, and perhaps more importantly, the adversarial classifier D does not have a chance to see randomly sampled results during training. In `cLR-GAN`, the latent space is easily sampled from a simple distribution, but the generator is trained without the benefit of seeing ground truth input-output pairs. We propose to train with constraints in both directions, aiming to take advantage of both cycles ($\mathbf{B} \rightarrow \mathbf{z} \rightarrow \widehat{\mathbf{B}}$ and $\mathbf{z} \rightarrow \widehat{\mathbf{B}} \rightarrow \widehat{\mathbf{z}}$),

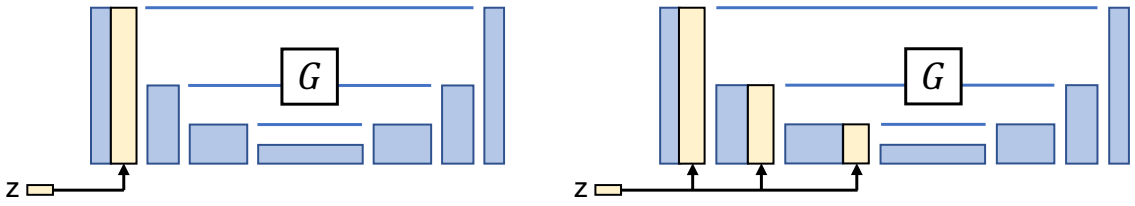


Figure 7.3: Alternatives for injecting z into generator: Latent code z is injected by spatial replication and concatenation into the generator network. We tried two alternatives, (left) injecting into the input layer and (right) every intermediate layer in the encoder.

hence the name `BicycleGAN`.

$$G^* = \arg\min_{G,E}\max_{D} \quad \mathcal{L}_{\text{GAN}}^{\text{VAE}}(G, D, E) + \lambda\mathcal{L}_1^{\text{VAE}}(G, E) \\ +\mathcal{L}_{\text{GAN}}(G, D) + \lambda_{\text{latent}}\mathcal{L}_1^{\text{latent}}(G, E) + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}(E), \tag{7.9}$$

where the hyper-parameters λ, λ_{latent}, and λ_{KL} control the importance of each term.

In the unconditional GAN setting, it has been observed that using samples both from random $\mathbf{z}$ and encoded vector $E(\mathbf{B})$ help further improves the results [121]. Hence, we also report one variant which is the full objective shown above (Equation 7.9), but without the reconstruction loss on the latent space $\mathcal{L}_1^{\text{latent}}$. We call it `cVAE-GAN++`, as it is based on `cVAE-GAN` with additional loss $\mathcal{L}_{\text{GAN}}(G, D)$, that encourages the discriminator to also see the randomly generated samples.

7.4 Implementation Details

The code and additional results are publicly available on our website. Please refer to our online arXiv paper at https://arxiv.org/abs/1711.11586 [250] for more details about the datasets, architectures, and training procedures.

Network architecture For generator G, we use the U-Net [171], which contains an encoder-decoder architecture, with symmetric skip connections. The architecture has been shown to produce strong results in the unimodal image prediction setting, when there is spatial correspondence between input and output pairs. For discriminator D, we use two PatchGAN discriminators [93] at different scales, which aims to predict real vs. fake for 70×70 and 140×140 overlapping image patches, respectively. For the encoder E, we experiment with two networks: (1) `E_CNN`: CNN with a few convolutional and downsampling layers and (2) `E_ResNet`: a classifier with several residual blocks [79].

Training details We build our model on the Least Squares GANs (LSGANs) variant [138], which uses a least-squares objective instead of a cross entropy loss. LSGANs produce high quality results with stable training. We also find that not conditioning the discriminator D leads to better results (also discussed in [155]), and hence choose to do the same for all methods. We set the parameters $\lambda_{\text{image}} = 10$, $\lambda_{\text{latent}} = 0.5$ and $\lambda_{\text{KL}} = 0.01$ in all our experiments. In practice, we tie the weights for the generators in the `cVAE-GAN` and `cLR-GAN`. We observe that using two separate discriminators yields slightly better visual results compared to discriminators with shared weights. We train our networks from scratch using Adam [112] with a batch size of 1 and with learning rate of 0.0002. We choose latent dimension to be 8 across all the datasets.

Injecting the latent code z to generator G. How to propagate the information encoded by latent code $\mathbf{z}$ to the image generation process is critical to our applications. We explore two choices as shown in Figure 7.3: (1) `add_to_input`: We simply extend a Z-dimensional latent code $\mathbf{z}$ to an $H \times W \times Z$ spatial tensor and concatenate it with the $H \times W \times 3$ input image. (2) `add_to_all`: Alternatively, we add $\mathbf{z}$ to each intermediate layer of the network G.

7.5 Experiments

Datasets We test our method on several image-to-image translation problems from prior work, including edges $\rightarrow$ photos [232, 247], Google maps $\rightarrow$ satellite [93], labels $\rightarrow$ images [31], and outdoor night $\rightarrow$ day images [118]. These problems are all one-to-many mappings. We train all the models on 256×256 images. Code and data are available at https://github.com/junyanz/BicycleGAN.

Methods We train the following models described in Section 7.3: `pix2pix+noise`, `cAE-GAN`, `cVAE-GAN`, `cVAE-GAN++`, `cLR-GAN` and the hybrid model `BicycleGAN`.

7.5.1 Qualitative Evaluation

We show qualitative comparison results on Figure 7.5. We observe that `pix2pix+noise` typically produces a single realistic output, but does not produce any meaningful variation. `cAE-GAN` adds variation to the output, but typically reduces quality of results, as shown for an example on facades on Figure 7.4. We observe more variation in the `cVAE-GAN`, as the latent space is encouraged to encode information about ground truth outputs. However, the space is not densely populated, so drawing random samples may cause artifacts in the output. The `cLR-GAN` shows less variation in the output, and sometimes suffers from mode collapse. When combining these methods, however, in the hybrid method `BicycleGAN`, we observe results which are

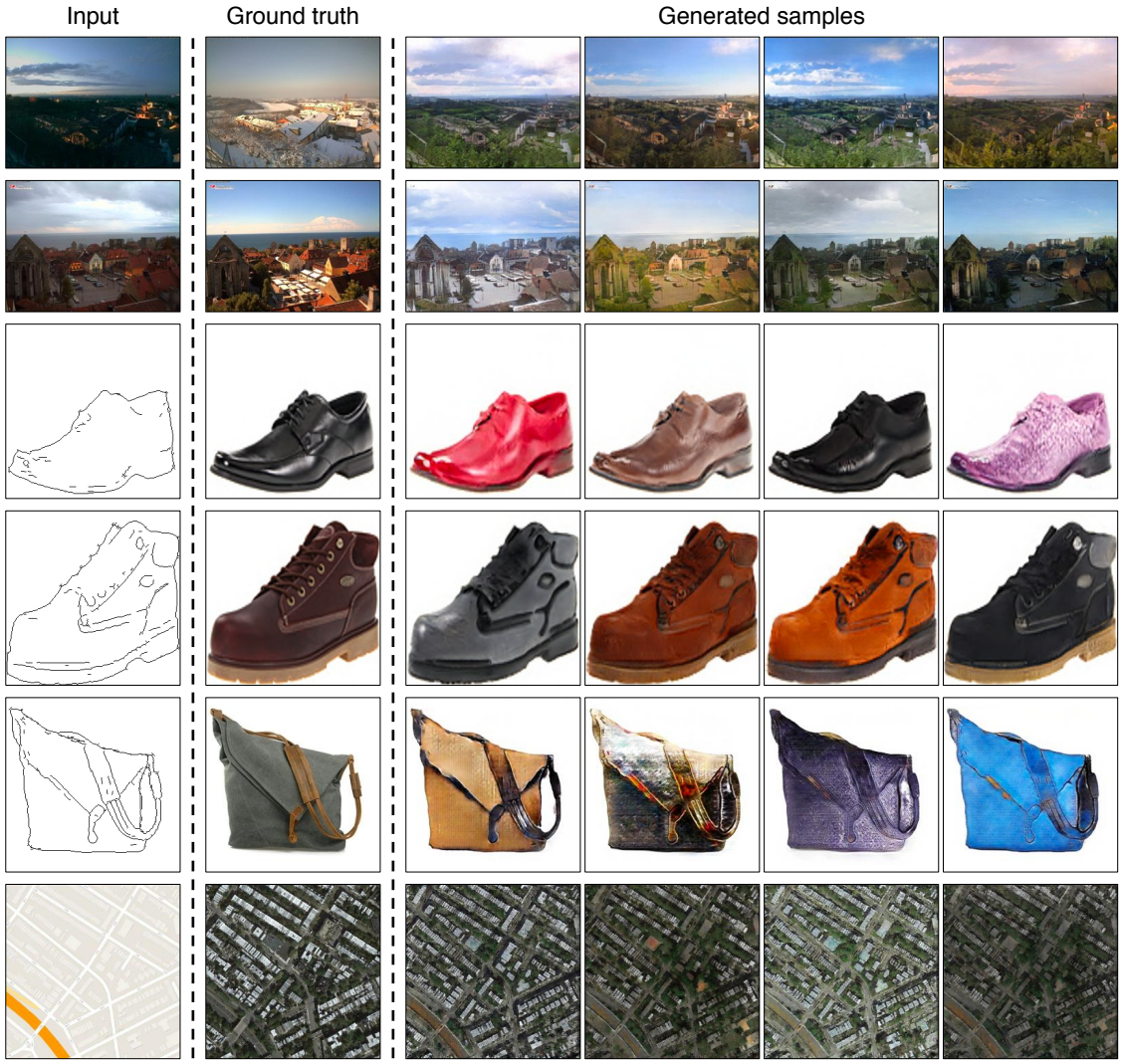


Figure 7.4: Example results: We show example results of our hybrid model `BicycleGAN`. The left column show shows the input. The second shows the ground truth output. The final four columns show randomly generated samples. We show results of our method on night→day, edges→shoes, edges→handbags, and maps→satellites. Models and additional examples are available at https://junyanz.github.io/BicycleGAN.

both diverse and realistic. We show example results in Figure 7.4. Please see our website for a full set of results.

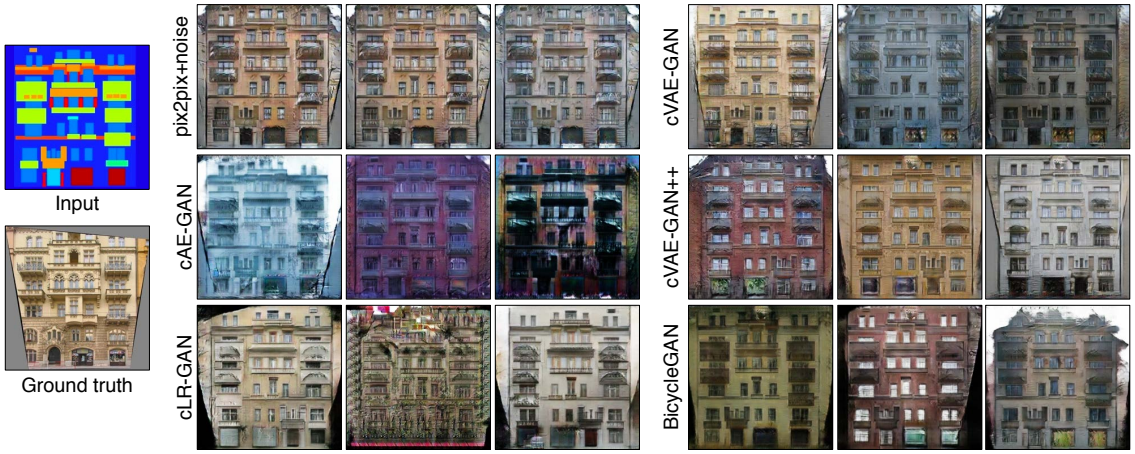


Figure 7.5: Qualitative method comparison: We compare results on the labels → facades dataset across different methods. The `BicycleGAN` method produces results which are both realistic and diverse.

	Realism	**Diversity**
Method	AMT Fooling Rate [%]	VGG-16 Distance
Random real images	50.0%	3.520±.021
`pix2pix+noise` [93]	27.93±2.40 %	0.338±.002
`cAE-GAN`	13.64±1.80 %	2.304±.012
`cVAE-GAN`	24.93±2.27 %	1.350±.013
`cVAE-GAN++`	29.19±2.43 %	1.425±.014
`cLR-GAN`	29.23±2.48 %	[1]1.374±.022
`BicycleGAN`	34.33±2.69 %	1.469±.014

Figure 7.6: Realism vs Diversity: We measure diversity using average feature distance in the VGG-16 space using cosine distance summed across five layers, and realism using a real vs. fake AMT test on the Google maps → satellites task.

7.5.2 Quantitative Evaluation

We perform a quantitative analysis on the diversity, realism, and latent space distribution on our six variants and baselines. We quantitatively test the Google maps → satellites dataset.

Diversity We randomly draw samples from our model and compute average distance in a deep feature space. In the context of style transfer, image super-

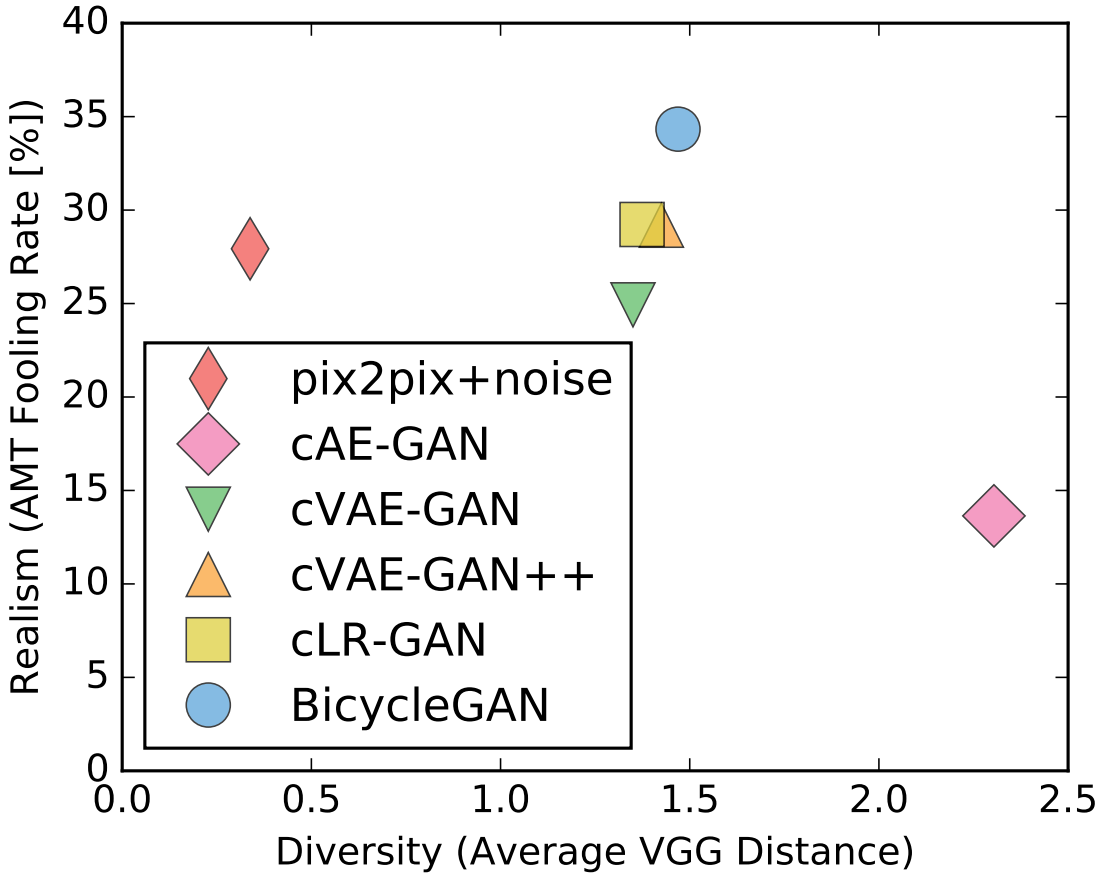


Figure 7.7: We show all the methods' realism and diversity on a 2D plot. The `pix2pix+noise` baseline produces little diversity. Using only `cAE-GAN` method produces large artifacts during sampling. The hybrid `BicycleGAN` method, which combines `cVAE-GAN++` and `cLR-GAN`, produces results which have higher realism while maintaining diversity.

resolution [99], and feature inversion [44], pretrained networks have been used as a "perceptual loss" and explicitly optimized over. In the context of generative modeling, they have been used as a held-out "validation" score, for example to assess how semantic samples from a generative model [177] or the semantic accuracy of a grayscale colorization [237].

In Figure 7.7, we show the diversity-score using the cosine distance, averaged across spatial dimensions, and summed across the five `conv` layers preceding the `pool` layers on the VGG-16 network [189], pre-trained for Imagenet classification [36].

The maximum score is 5.0, as all the feature responses are nonnegative. For each method, we compute the average distance between 1900 pairs of randomly generated output $\widehat{\mathbf{B}}$ images (sampled from 100 input $\mathbf{A}$ images). Random pairs of ground truth real images in the $\mathbf{B} \in \mathcal{B}$ domain produce an average variation of 3.520 using cosine distance. As we are measuring samples $\widehat{\mathbf{B}}$ which correspond to a specific input, a system which stays faithful to the input should definitely not exceed this score.

The `pix2pix` system [93] produces a single point estimate. Adding noise to the system `pix2pix+noise` produces a diversity score of 0.338, confirming the finding in [93] that adding noise does not produce large variation. Using an `cAE-GAN` model to encode ground truth image $\mathbf{B}$ into latent code $\mathbf{z}$ does increase the variation. The `cVAE-GAN`, `cVAE-GAN++`, and `BicycleGAN` models all place explicit constraints on the latent space, and the `cLR-GAN` model places an implicit constraint through sampling. These four methods all produce similar diversity scores. We note that high diversity scores may also indicate that nonrealistic images are being generated, causing meaningless variation. Next, we investigate the visual realism of our samples.

Perceptual Realism To judge the visual realism of our results, we use human judgments, as proposed in [237] and later used in [93, 249]. The test sequentially presents a real and generated image to a human for 1 second each, in a random order, asks them to identify the fake, and measures the "fooling" rate. Figure 7.7(left) shows the realism across methods. The `pix2pix+noise` model achieves high realism score, but without large diversity, as discussed in the previous section. The `cAE-GAN` helps produce diversity, but this comes at a large cost to the visual realism. Because the distribution of the learned latent space is unclear, random samples may be from unpopulated regions of the space. Adding the KL-divergence loss in the latent space, used in the `cVAE-GAN` model recovers the visual realism. Furthermore, as expected, checking randomly drawn $\mathbf{z}$ vectors in the `cVAE-GAN++` model slightly increases realism. The `cLR-GAN`, which draws $\mathbf{z}$ vectors from the predefined distribution randomly, produces similar realism and diversity scores. However, the `cLR-GAN` model resulted in large mode collapse - approximately 15% of the outputs produced the same result, independent of the input image. The full hybrid `BicycleGAN` gets the best of both worlds, as it does not suffer from mode collapse and also has the highest realism score by a significant margin.

Encoder architecture The `pix2pix` framework [93] have conducted extensive ablation studies on discriminators and generators. Here we focus on the performance of two encoders `E_CNN` and `E_ResNet` for our applications on maps and facades datasets, and we find that `E_ResNet` can better encode the output image, regarding the image reconstruction loss $||\mathbf{B} - G(\mathbf{A}, E(B))||_1$ on validation datasets as shown in Table 7.1.

Methods of injecting latent code We evaluate two ways of injecting latent

Encoder	E_ResNet	E_ResNet	E_CNN	E_CNN
Injecting **z**	add_to_all	add_to_input	add_to_all	add_to_input
label→photo	0.292 ± 0.058	0.292 ± 0.054	0.326 ± 0.066	0.339 ± 0.069
map → satellite	0.268 ± 0.070	0.266 ± 0.068	0.287 ± 0.067	0.272 ± 0.069

Table 7.1: The encoding performance with respect to the different encoder architectures and methods of injecting **z**. Here we report the reconstruction loss $||\mathbf{B} - G(\mathbf{A}, E(B))||_1$

.

Figure 7.8: Different label → facades results trained with varying length of the latent code $|\mathbf{z}| \in \{2, 8, 256\}$.

code **z**: add_to_input and add_to_all (Section 7.4), regarding the same reconstruction loss $||\mathbf{B} - G(\mathbf{A}, E(B))||_1$. Table 7.1 shows that two methods give similar performance. This indicates that the U_Net [171] can already propagate the information well to the output without the additional skip connections from **z**.

Latent code length We study the BicycleGAN model results with respect to the varying number of dimensions of latent codes $\{2, 8, 256\}$ in Figure 7.8. A low-dimensional latent code may limit the amount of diversity that can be expressed by the model. On the contrary, a high-dimensional latent code can potentially encode more information about an output image at the cost of making sampling quite difficult. The optimal length of **z** largely depends on individual datasets and applications, and how much ambiguity there is in the output.

7.6 Discussion

In conclusion, we have evaluated a few methods for combating the problem of mode collapse in the conditional generative setting. We find that by combining multiple objectives for encouraging a bijective mapping between the latent and output spaces, we obtain results which are more realistic and diverse. We see many interesting avenues of future work, including directly enforcing a distribution in the

latent space that encodes semantically meaningful attributes to allow for image-to-image transformations with user controllable parameters.

Chapter 8

Discussion

In conclusion, we have described a few data-driven approaches for learning visual realism directly from large-scale image collections. We use the learned visual realism models for realistic image synthesis and editing. Our extensive experiments have shown that the presented approaches can produce more visually appealing results compared to previous methods that rely on hand-crafted engineering or heuristics. Our algorithms have also enabled many new visual synthesis and manipulation effects, unachievable by any prior works.

Below, we discuss several potential future directions, building on our current image synthesis and manipulation algorithms.

Image Search by Mental Picture While working on interactive visual synthesis projects, we realized that in addition to content creation, the user-guided image generation is a potentially powerful tool for image *retrieval*. We performed a preliminary experiment with the AverageExplorer system [248], seeing if it could assist online shoppers in finding a desired item, such as a specific shoe. As shown in Figure 8.1, the user may start with a set of black high heels (left), and then decide on a different color. Simply drawing a stroke with the desired red color on the generated image (center) retrieves the available red shoes of similar styles. If the user then desires a heel with more support, adding an edge stroke at the bottom of the shoe switches the heel style (right).

Visual Domain Adaptation To goal of domain adaptation is to adapt existing models to new environments. Most current work focuses on adaptation at the *feature* level. By learning domain invariant feature representations, models are less likely to degrade in performance when presented with a domain shift. However, these models are often difficult to interpret, as the changes are in an abstract feature

Figure 8.1: Image search with mental picture: our image synthesis system could be potentially used as an image search interface for efficient browsing

(a) GTA5 (b) GTA5 → Cityscapes (c) CityScapes (d) CityScapes → GTA5

Figure 8.2: Unpaired Image-to-Image translation results on GTA5 ↔ Cityscapes

space. In addition, the additional constraints are only applied at a high level, so the models may fail to bridge the low-level domain gap. To address these issues, a few recent methods investigate *pixel*-level domain adaptation, building, in part, upon our image-to-image translation methods [93, 249]. Our models excel at translating images from one domain to another, even in the absence of paired training data [249]. See Figure 8.2 for CycleGAN results on translating between two semantic segmentation datasets: GTA5 [169] ↔ Cityscapes [31]. Intuitively, matching statistics at the pixel-level perfectly would also match statistics at the feature-level by definition. These proposed methods have shown state-of-the-art results across various adaptation tasks, including semantic segmentation of street-view imagery [85, 145], person re-identification [241], digit classification [85], and synthetic to real adaptation [145]. However, our image-to-image models can struggle to handle large spatial misalignment across domains, in which feature-level domain adaptation may still be needed. In the future, we would like to explore how to effectively combine pixel-level and feature-level domain adaptation, to align both low-level statistics and high-level semantics across domains.

High-resolution Visual Synthesis and Manipulation Existing GAN-based image synthesis systems have enabled a wide range of vision and graphics applications, but most of the results are limited in resolution and far from perfect. The main reason is that it is difficult to train GANs to produce high-resolution images, due to training instability and optimization issues. These challenges call for better network

architectures, as well as more robust loss functions and stable training procedures. Two recent works have succeeded in the megapixel regime, synthesizing 1024×1024 images in the unconditional setting [105], and 2048×1024 images in the conditional setting [217]. These breakthroughs may enable high-resolution photorealistic editing applications. Both of these works include coarse-to-fine architectures, curriculum learning procedures, and a few carefully designed implementation details for robust training. In the future, end-to-end training with more compact, simple, and memory-efficient network architectures are a potential next step in this direction.

Bibliography

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 294–302. ACM, 2004.

[2] Nir Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13(1):137–164, 2012.

[3] Georgia Albuquerque, Timo Stich, Anita Sellent, and Marcus Magnor. The good, the bad and the ugly: Attractive portraits from video sequences. In *European Conference on Visual Media Production*, London, UK, November 2008.

[4] Hani Altwaijry and Serge Belongie. Relative ranking of facial attractiveness. In *IEEE Winter Confererence on Applications of Computer Vision*, pages 117–124, 2013.

[5] Zara Ambadar, Jeffrey F. Cohn, and Lawrence Ian Reed. All smiles are not created equal: Morphology and timing of smiles perceived as amused, polite, and embarrassed/nervous. *Journal of Nonverbal Behavior*, 33(1):17–34, 2009.

[6] Anelia Angelova, Y Abu-Mostafam, and Pietro Perona. Pruning training sets for learning of object categories. In *CVPR*, 2005.

[7] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.

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

[9] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *to appear in SIGGRAH*, 2007.

[10] Yusuf Aytar, Lluis Castrejon, Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Cross-modal scene networks. *arXiv preprint arXiv:1610.09003*, 2016.

[11] Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. In *Algorithmic Learning Theory*, pages 316–328. Springer, 2008.

[12] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 28(3):24, 2009.

[13] Peter N Belhumeur, David W Jacobs, David J Kriegman, and Neeraj Kumar. Localizing parts of faces using a consensus of exemplars. In *CVPR*, 2011.

[14] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *ICML*, pages 226–234, 2014.

[15] T.L. Berg and A.C. Berg. Finding iconic images. In *2nd Workshop on Internet Vision*, 2009.

[16] Y-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in vision algorithms. In *International Conference on Machine Learning*, 2010.

[17] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. 2017.

[18] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[19] Richard W Brislin. Back-translation for cross-cultural research. *Journal of cross-cultural psychology*, 1(3):185–216, 1970.

[20] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36. Springer, 2004.

[21] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *IJCV*, 61(3):211–231, 2005.

[22] Peter J Burt and Edward H Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics (TOG)*, 2(4):217–236, 1983.

[23] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[24] Andy Calder, Gillian Rhodes, Mark Johnson, and James Haxby. *Oxford Handbook of Face Perception*. Oxford University Press, 2012.

[25] Jim Campbell. http://jimcampbell.tv/portfolio/still_image_works/, 2002.

[26] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[27] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[28] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: Internet image montage. *ACM Transactions on Graphics (TOG)*, 28(5):124, 2009.

[29] Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *ACM International Conference on Web Search and Data Mining*, pages 193–202, 2013.

[30] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.

[31] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition*, 2016.

[32] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, 2005.

[33] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE, 2005.

[34] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image melding: combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG)*, 31(4), 2012.

[35] Maria Cristina Ferreira de Oliveira and Haim Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394, 2003.

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[37] Emily L Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *Neural Information Processing Symposium*, 2015.

[38] Hamdi Dibeklioglu, Theo Gevers, and Albert Ali Salah. Are you really smiling at me? spontaneous versus posed enjoyment smiles. In *European Conference on Computer Vision*, number 3, pages 525–538, 2012.

[39] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *ICLR*, 2017.

[40] Santosh K. Divvala, Alexei A. Efros, and Martial Hebert. How important are 'deformable parts' in the deformable parts model? In *Parts and Attributes Workshop, ECCV*, 2012.

[41] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A Efros. What makes paris look like paris? *SIGGRAPH*, 31(4):101, 2012.

[42] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[43] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[44] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*, 2016.

[45] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, pages 1538–1546, 2015.

[46] Shichuan Du, Yong Tao, and Aleix M. Martinez. Compound facial expressions of emotion. *Proceedings of the National Academy of Sciences*, 2014.

[47] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. 2017.

[48] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, pages 341–346. ACM, 2001.

[49] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, volume 2, pages 1033–1038. IEEE, 1999.

[50] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *International Conference on Computer Vision*, pages 2650–2658, 2015.

[51] P. Ekman and W. V. Friesen. *The Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, 1978.

[52] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

[53] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.

[54] Juliet Fiss, Aseem Agarwala, and Brian Curless. Candid portrait selection from video. *ACM Transactions on Graphics (TOG)*, 30(6):128:1–128:8, 2011.

[55] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Computer Vision and Pattern Recognition*, pages 5515–5524, 2016.

[56] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[57] A. Gallagher and T. Chen. Clothing cosegmentation for recognizing people. In *Computer Vision and Pattern Recognition*, 2008.

[58] Andrew C Gallagher and Tsuhan Chen. Clothing cosegmentation for recognizing people. In *Computer Vision and Pattern Recognition*, 2008.

[59] Francis Galton. Composite portraits made by combining those of many different persons into a single figure. *Nature*, (18):97–100, 1878.

[60] Leon A Gatys, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*, 2016.

[61] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *CVPR*, 2016.

[62] Samuel J Gershman and Noah D Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, 2014.

[63] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[64] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[65] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[66] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[67] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Symposium*, 2014.

[68] Douglas Gray, Kai Yu, Wei Xu, and Yihong Gong. Predicting facial beauty without landmarks. In *European Conference on Computer Vision*, pages 434–447. 2010.

[69] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, 2015.

[70] JJ Gross and RW Levenson. Emotion elicitation using films. *Cognition & Emotion*, 1995.

[71] Sergio Guadarrama, Ryan Dahl, David Bieber, Mohammad Norouzi, Jonathon Shlens, and Kevin Murphy. Pixcolor: Pixel recursive colorization. In *BMVC*, 2017.

[72] Sarah D. Gunnery, Judith A. Hall, and Mollie A. Ruben. The deliberate duchenne smile: Individual differences in expressive control. *Journal of Nonverbal Behavior*, 37(1):1–13, 2012.

[73] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. Discriminative decorrelation for clustering and classification. In *European Conference on Computer Vision*, pages 459–472. 2012.

[74] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.

[75] James Hays and Alexei A Efros. Scene completion using millions of photographs. In *SIGGRAPH*, volume 26, page 4, 2007.

[76] James Hays and Alexei A Efros. Scene completion using millions of photographs. *ACM Trans. on Graphics*, 26(3), 2007.

[77] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Neural Information Processing Symposium*, pages 820–828, 2016.

[78] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *ECCV*, pages 1–14. Springer, 2010.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[80] Y. Hel-Or and D. Shaked. A discriminative approach for wavelet denoising. *IEEE Transactions on Image Processing*, 17(4):443–457, 2008.

[81] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *SIGGRAPH*, pages 327–340. ACM, 2001.

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

[83] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[84] M. Hoai and A. Zisserman. Discriminative Sub-categorization. In *CVPR*, 2013.

[85] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[86] G. Huang, V. Jain, and E. Learned-Miller. Unsupervised Joint Alignment of Complex Images. In *ICCV*, 2007.

[87] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Technical report, University of Massachusetts, Amherst, 2007.

[88] Jinggang Huang and David Mumford. Statistics of natural images and models. In *Computer Vision and Pattern Recognition*, volume 1, pages 541–547. IEEE, 1999.

[89] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, volume 32, pages 177–186. Wiley Online Library, 2013.

[90] Takeo Igarashi, Tomer Moscovich, and John F Hughes. As-rigid-as-possible shape manipulation. *SIGGRAPH*, 24(3):1134–1141, 2005.

[91] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (TOG)*, 35(4), 2016.

[92] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37, pages 448–456, 2015.

[93] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition*, 2017.

[94] Charles E Jacobs, Adam Finkelstein, and David H Salesin. Fast multiresolution image querying. In *SIGGRAPH*, 1995.

[95] Kevin G. Jamieson and Robert D. Nowak. Active ranking using pairwise comparisons. In *Neural Information Processing Symposium*, pages 2240–2248, 2011.

[96] Oliver Jesorsky, Klaus J Kirchberg, and Robert W Frischholz. Robust face detection using the hausdorff distance. In *Audio-and video-based biometric person authentication*, pages 90–95. Springer, 2001.

[97] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678, 2014.

[98] Bihan Jiang, Michel François Valstar, and Maja Pantic. Action unit detection using sparse appearance descriptors in space-time video volumes. In *IEEE Conference on Automatic Face and Gesture Recognition*, pages 314–321, 2011.

[99] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. 2016.

[100] Micah K Johnson, Kevin Dale, Shai Avidan, Hanspeter Pfister, William T Freeman, and Wojciech Matusik. Cg2real: Improving the realism of computer generated images using a large collection of photographs. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1273–1285, 2011.

[101] Neel Joshi, Wojciech Matusik, Edward H Adelson, and David J Kriegman. Personal photo enhancement using example images. *ACM Transactions on Graphics (TOG)*, 29(2):1–15, 2010.

[102] Amit Kagian, Gideon Dror, Tommer Leyvand, Isaac Meilijson, Daniel Cohen-Or, and Eytan Ruppin. A machine learning predictor of facial attractiveness revealing human-like psychophysical biases. *Vision research*, 48(2):235–43, 2008.

[103] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *SIGGRAPH Asia*, 2016.

[104] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016.

[105] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[106] Eric Kee, James F O'brien, and Hany Farid. Exposing photo manipulation from shading and shadows. *ACM Transactions on Graphics (TOG)*, 33(5):165, 2014.

[107] Ira Kemelmacher-Shlizerman, Eli Shechtman, Rahul Garg, and Steven M Seitz. Exploring photobios. *ACM Transactions on Graphics (TOG)*, 30(4):61, 2011.

[108] Ira Kemelmacher-Shlizerman, Eli Shechtman, Rahul Garg, and Steven M. Seitz. Exploring photobios. In *SIGGRAPH*, 2011.

[109] Idris Khan. www.skny.com/artists/idris-khan/images/, 2005.

[110] Aditya Khosla, Wilma A. Bainbridge, Antonio Torralba, and Aude Oliva. Modifying the memorability of face photographs. In *International Conference on Computer Vision*, 2013.

[111] M Hadi Kiapour, Kota Yamaguchi, Alexander C Berg, and Tamara L Berg. Hipster wars: Discovering elements of fashion styles. In *European Conference on Computer Vision*, pages 472–488. 2014.

[112] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

[113] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

[114] Philipp Krähenbühl and Vladlen Koltun. Geodesic object proposals. In *ECCV*, pages 725–739. 2014.

[115] Philipp Krähenbühl, Manuel Lang, Alexander Hornung, and Markus Gross. A system for retargeting of streaming video. In *ACM Transactions on Graphics (TOG)*, volume 28, page 126. ACM, 2009.

[116] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Symposium*, pages 1097–1105, 2012.

[117] Eva Gabriele Krumhuber and Antony Stephen Reid Manstead. Can duchenne smiles be feigned? new evidence on felt and false smiles. *Emotion*, 9(6):807–820, 2009.

[118] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)*, 33(4):149, 2014.

[119] Jean-François Lalonde and Alexei A Efros. Using color compatibility for assessing image realism. In *ICCV*, pages 1–8, 2007.

[120] Jean-François Lalonde, Derek Hoiem, Alexei A Efros, Carsten Rother, John Winn, and Antonio Criminisi. Photo clip art. In *ACM Transactions on Graphics (TOG)*, volume 26, page 3, 2007.

[121] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

[122] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. *ECCV*, 2016.

[123] E. Learned-Miller. Data Driven Image Models through Continuous Joint Alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2006.

[124] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[125] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *Computer Vision and Pattern Recognition*, 2017.

[126] Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. Shadowdraw: real-time user guidance for freehand drawing. *SIGGRAPH*, 30(4):27, 2011.

[127] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 689–694. ACM, 2004.

[128] Tommer Leyvand, Daniel Cohen-Or, Gideon Dror, and Dani Lischinski. Data-driven enhancement of facial attractiveness. *ACM Transactions on Graphics (TOG)*, 27(3):38:1–38:9, 2008.

[129] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *ECCV*, 2016.

[130] Lucy Liang and Kristen Grauman. Beyond comparing image pairs: Setwise active learning for relative attributes. In *Computer Vision and Pattern Recognition*, 2014.

[131] Joseph CR Licklider. Man-computer symbiosis. *IRE transactions on human factors in electronics*, (1):4–11, 1960.

[132] Joseph CR Licklider and Robert W Taylor. The computer as a communication device. *Science and technology*, 76(2):1–3, 1968.

[133] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *Neural Information Processing Symposium*, 2017.

[134] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Neural Information Processing Symposium*, pages 469–477, 2016.

[135] Yiming Liu, Jue Wang, Sunghyun Cho, Adam Finkelstein, and Szymon Rusinkiewicz. A no-reference metric for evaluating the quality of motion deblurring. *ACM Transactions on Graphics (TOG)*, 32(6):175, 2013.

[136] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[137] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *International Conference on Learning Representations*, 2016.

[138] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *International Conference on Computer Vision*, 2017.

[139] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *ICLR*, 2016.

[140] Michael F Mathieu, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Neural Information Processing Symposium*, pages 5040–5048, 2016.

[141] Marwan Mattar, Allen Hanson, and Erik G Learned-Miller. Unsupervised joint alignment and clustering using bayesian nonparametrics. In *Computer Vision and Pattern Recognition*, 2012.

[142] Daniel McDuff, Rana El Kaliouby, and Rosalind W. Picard. Crowdsourcing facial responses to online videos. *IEEE Transactions on Affective Computing*, 3(4):456–468, 2012.

[143] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[144] Masahiro Mori, Karl F MacDorman, and Norri Kageki. The uncanny valley. *Robotics & Automation Magazine, IEEE*, 19(2):98–100, 2012.

[145] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. *arXiv preprint arXiv:1712.00479*, 2017.

[146] Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, 2017.

[147] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition*, 2015.

[148] Peter O'Donovan, Janis Libeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Transactions on Graphics (TOG)*, 33(4), 2014.

[149] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, June 1996.

[150] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning*, 2016.

[151] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *Neural Information Processing Symposium*, 2016.

[152] Nikolaas N. Oosterhof and Alexander Todorov. The functional basis of face evaluation. *Proceedings of the National Academy of Sciences*, 105(32):11087–11092, 2008.

[153] Maja Pantic and Leon J. M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1424–1445, 2000.

[154] Devi Parikh and Kristen Grauman. Relative attributes. In *International Conference on Computer Vision*, pages 503–510. IEEE, 2011.

[155] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition*, 2016.

[156] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM Transactions on graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[157] Pietro Perona. Vision of a visipedia. *Proceedings of the IEEE*, 98(8):1526–1534, 2010.

[158] Alin C Popescu and Hany Farid. Exposing digital forgeries by detecting traces of resampling. *Signal Processing, IEEE Transactions on*, 53(2):758–767, 2005.

[159] Javier Portilla and Eero P. Simoncelli. A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. *IJCV*, 40(1):49–70, October 2000.

[160] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12:1338–1351, 2003.

[161] Krzysztof Pruszkowski. http://www.gallerywm.com/prusz_index.html, 1986.

[162] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations*, 2016.

[163] Radim Šára Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, Saarbrucken, Germany, 2013.

[164] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[165] Erik Reinhard, Ahmet Oguz Akuyz, Mark Colbert, Charles E Hughes, and Matthew O'Connor. Real-time color blending of rendered and captured video. In *Interservice/Industry Training, Simulation and Education Conference*, volume 18, 2004.

[166] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Comput. Graph. Appl.*, September 2001 2001.

[167] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Comput. Graph. Appl.*, 21(5):34–41, September 2001.

[168] Liu Ren, Alton Patrick, Alexei A Efros, Jessica K Hodgins, and James M Rehg. A data-driven approach to quantifying natural human motion. *ACM Trans. on Graphics*, 24(3):1090–1097, 2005.

[169] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016.

[170] Eric Risser, Charles Han, Rozenn Dahyot, and Eitan Grinspun. Synthesizing structured image hybrids. *SIGGRAPH*, 29(4):85:1–85:6, 2010.

[171] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[172] Rómer Rosales, Kannan Achan, and Brendan J Frey. Unsupervised image translation. In *International Conference on Computer Vision*, pages 472–478, 2003.

[173] Stefan Roth and Michael J. Black. Fields of Experts: A Framework for Learning Image Priors. In *CVPR*, 2005.

[174] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008.

[175] Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep boltzmann machines. In *AISTATS*, 2009.

[176] Jason Salavon. www.salavon.com/work/specialmoments/, 2004.

[177] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *arXiv preprint arXiv:1606.03498*, 2016.

[178] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, 2017.

[179] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, 2017.

[180] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *SIGGRAPH*, 25:533–540, 2006.

[181] Uwe Schmidt, Carsten Rother, Sebastian Nowozin, Jeremy Jancsary, and Stefan Roth. Discriminative non-blind deblurring. In *Computer Vision and Pattern Recognition*, June 2013.

[182] Steven M. Seitz and Charles R. Dyer. View morphing. In *SIGGRAPH*, pages 21–30, New York, 1996.

[183] Rajvi Shah and Vivek Kwatra. All smiles : Automatic photo enhancement by facial expression analysis. In *European Conference on Visual Media Production*, 2012.

[184] Eli Shechtman, Alex Rav-Acha, Michal Irani, and Steve Seitz. Regenerative morphing. In *CVPR*, San-Francisco, CA, June 2010.

[185] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[186] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):200, 2013.

[187] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. *Computer Vision and Pattern Recognition*, 2017.

[188] I. Simon, N. Snavely, and S. Seitz. Scene Summarization for Online Image Collections. In *ICCV*, 2007.

[189] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[190] S. Singh, A. Gupta, and A. A. Efros. Unsupervised Discovery of Mid-Level Discriminative Patches. In *ECCV*, 2012.

[191] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[192] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.

[193] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, 2006.

[194] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

[195] I N Springer, J Wiltfang, J T Kowalski, P a J Russo, M Schulze, S Becker, and S Wolfart. Mirror, mirror on the wall: self-perception of facial beauty versus judgement by others. *Journal of cranio-maxillo-facial surgery*, 40(8):773–6, 2012.

[196] Xinghai Sun, Changhu Wang, Chao Xu, and Lei Zhang. Indexing billions of images for sketch-based retrieval. In *ACM Multimedia*, 2013.

[197] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.

[198] Ivan E Sutherland. Sketchpad a man-machine graphical communication system. *Transactions of the Society for Computer Simulation*, 2(5):R–3, 1964.

[199] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[200] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *International Conference on Learning Representations*, 2017.

[201] Minghui Tan, Jean-François Lalonde, Lavanya Sharan, Holly Rushmeier, and Carol O'sullivan. The perception of lighting inconsistencies in composite outdoor scenes. *ACM Transactions on Applied Perception*, 12(4):18:1–18:18, September 2015.

[202] Michael W Tao, Micah K Johnson, and Sylvain Paris. Error-tolerant image compositing. *IJCV*, 103(2):178–189, 2013.

[203] Antonio Torralba. http://people.csail.mit.edu/torralba/gallery/, 2001.

[204] Antonio Torralba, Hector J. Bernal, Rob Fergus, Yair Weiss, and William Freeman. http://groups.csail.mit.edu/vision/tinyimages/, 2008.

[205] K. Tsukida and M. R. Gupta. How to analyze paired comparison data. Technical Report UWEETR-2011-0004, Dept. of Electrical Engineering, University of Washington, 2011.

[206] Daniyar Turmukhambetov, Neill DF Campbell, Simon JD Prince, and Jan Kautz. Modeling object appearance using context-conditioned component analysis. In *Computer Vision and Pattern Recognition*, pages 4156–4164, 2015.

[207] Mark Twain. *The Jumping Frog: in English, then in French, and then Clawed Back into a Civilized Language Once More by Patient, Unremunerated Toil.* 1903.

[208] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning*, 2016.

[209] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[210] Fernanda B Viégas and Martin Wattenberg. Artistic data visualization: Beyond visual analytics. In *Online Communities and Social Computing*, pages 182–191. Springer, 2007.

[211] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[212] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, 2008.

[213] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Neural Information Processing Symposium*, pages 613–621, 2016.

[214] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, 2016.

[215] Fan Wang, Qixing Huang, and Leonidas J Guibas. Image co-segmentation via consistent functional maps. In *International Conference on Computer Vision*, pages 849–856, 2013.

[216] Jue Wang and Michael F. Cohen. Very low frame-rate video streaming for face-to-face teleconference. In *Proceedings of the Data Compression Conference*, pages 309–318, 2005.

[217] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. *arXiv preprint arXiv:1711.11585*, 2017.

[218] Ting-Chun Wang, Jun-Yan Zhu, Ebi Hiroaki, Manmohan Chandraker, Alexei Efros, and Ravi Ramamoorthi. A 4D light-field dataset and CNN architectures for material recognition. In *European Conference on Computer Vision*, 2016.

[219] Ting-Chun Wang, Jun-Yan Zhu, Nima Khademi Kalantari, Alexei A. Efros, and Ravi Ramamoorthi. Light field video capture using a learning-based hybrid imaging system. *ACM Transactions on Graphics (TOG)*, 36(4), 2017.

[220] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. *ECCV*, 2016.

[221] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[222] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Neural Information Processing Symposium*, pages 82–90, 2016.

[223] Wenqi Xian, Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *arXiv preprint arXiv:1706.02823*, 2017.

[224] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[225] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition*, pages 532–539, 2013.

[226] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[227] Su Xue, Aseem Agarwala, Julie Dorsey, and Holly Rushmeier. Understanding and improving the realism of image composites. *ACM Transactions on Graphics (TOG)*, 31(4):84, 2012.

[228] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Neural Information Processing Symposium*, 2016.

[229] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Computer Vision and Pattern Recognition*, 2017.

[230] Fei Yang, Jue Wang, Eli Shechtman, Lubomir Bourdev, and Dimitri Metaxas. Expression flow for 3d-aware face component transfer. *ACM Transactions on Graphics (TOG)*, 30(4):60, 2011.

[231] Zili Yi, Hao Zhang, Tang Gong, Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *International Conference on Computer Vision*, 2017.

[232] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *CVPR*, pages 192–199, 2014.

[233] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[234] Christopher Zach, Manfred Klopschitz, and Marc Pollefeys. Disambiguating visual relations using loop constraints. In *Computer Vision and Pattern Recognition*, pages 1426–1433. IEEE, 2010.

[235] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *International Conference on Computer Vision*, 2017.

[236] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: High-resolution capture for modeling and animation. *ACM Transactions on Graphics (TOG)*, 23(3), 2004.

[237] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[238] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *SIGGRAPH*, 2017.

[239] Weiwei Zhang, Jian Sun, and Xiaoou Tang. Cat head detection-how to effectively exploit shape and texture features. In *European Conference on Computer Vision*. 2008.

[240] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *ICLR*, 2017.

[241] Zhun Zhong, Liang Zheng, Zhedong Zheng, Shaozi Li, and Yi Yang. Camera style adaptation for person re-identification. *arXiv preprint arXiv:1711.10295*, 2017.

[242] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Neural Information Processing Symposium*, 2014.

[243] Tinghui Zhou, Yong Jae Lee, Stella X Yu, and Alyosha A Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *Computer Vision and Pattern Recognition*, pages 1191–1200, 2015.

[244] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Computer Vision and Pattern Recognition*, pages 117–126, 2016.

[245] Jun-Yan Zhu, Aseem Agarwala, Alexei A Efros, Eli Shechtman, and Jue Wang. Mirror mirror: Crowdsourcing better portraits. *ACM Transactions on Graphics (TOG)*, 33(6), 2014.

[246] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Learning a discriminative model for the perception of realism in composite images. In *ICCV*, 2015.

[247] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016.

[248] Jun-Yan Zhu, Yong Jae Lee, and Alexei A Efros. Averageexplorer: Interactive exploration and alignment of visual data collections. *SIGGRAPH*, 33(4), 2014.

[249] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision*, 2017.

[250] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Neural Information Processing Symposium*. 2017.

[251] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision*, pages 479–486, 2011.