Machine Learning for Deep Image Synthesis

Taesung Park

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-143
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-143.html

May 20, 2021

Acknowledgement

First and foremost, I would like to thank my advisor Alyosha. His advice on research and life will stay in my heart forever.I also can't express how much I am grateful for my Ph.D. brothers Jun-Yan Zhu and Richard Zhang, who basically shaped the way I do research. I learned so much, from practical things like defining typesetting macros to brainstorming next research directions. I am also grateful to my internship hosts Ming-Yu Liu and Eli Shechtman for the opportunities. I would also like to thank all my colleagues and project partners.

Machine Learning for Deep Image Manipulation

By

Taesung Park

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexei A. Efros, Chair
Professor Jitendra Malik
Professor Angjoo Kanazawa
Professor Jaakko Lehtinen

Spring 2021

Abstract

Machine Learning for Deep Image Manipulation

by

Taesung Park

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Alexei A. Efros, Chair

Common types of image editing methods focus on low-level characteristics. In this thesis, I leverage machine learning to enable image editing that operates at a higher conceptual level. Fundamentally, the proposed methods aim to factor out the visual information that must be maintained in the editing process from the information that may be edited by incorporating the generic visual knowledge. As a result, the new methods can transform images in human-interpretable ways, such as turning one object into another, stylizing photographs into a specific artist's paintings, or adding sunset to a photo taken in daylight.

We explore designing such methods in different settings with varying amounts of supervision: per-pixel labels, per-image labels, and no labels. First, using per-pixel supervision, I propose a new deep neural network architecture that can synthesize realistic images from scene layouts and optional target styles. Second, using per-image supervision, I explore the task of domain translation, where an input image of one class is transformed into another. Lastly, I design a framework that can still discover disentangled manipulation of structure and texture from a collection of unlabeled images. We present convincing visuals in a wide range of applications including interactive photo drawing tools, object transfiguration, domain gap reduction between virtual and real environment, and realistic manipulation of image textures.

Contents

List of Figures iii

List of Tables vi

Acknowledgments vii

1 Introduction 1

2 Learning Representation for Image Editing from Paired Examples 4

2.1 Introduction 5
2.2 Related Work 6
2.3 Semantic Image Synthesis 7
2.4 Experiments 12
2.5 Conclusion 18
2.6 Additional Implementation Details 18
2.7 Additional Ablation Study 23
2.8 Additional Results 24

3 Discovering Structure and Style from Unpaired Examples 31

3.1 Related work 33
3.2 Pixel Cycle-Consistency Loss for Unpaired Translation 36
3.2.1 Adversarial Loss 37
3.2.2 Cycle Consistency Loss 37
3.2.3 Full Objective 39
3.2.4 Implementation 39
3.2.5 Results 40
3.2.6 Applications 46
3.2.7 Limitations and Discussion 48
3.3 Contrastive Loss for
Unpaired Translation 56

3.3.1 Methods 57
3.3.2 Experiments 61
3.3.3 Discussion 74
3.4 Applying Image Translation for Domain Adaptation 78
3.4.1 Related Work 78
3.4.2 Cycle-Consistent Adversarial Domain Adaption 80
3.4.3 Experiments 82
3.5 Conclusion 87

4 Learning Structure and Texture without Per-Image Labels 88
4.1 Introduction 88
4.2 Related Work 90
4.3 Method 92
4.3.1 Accurate and realistic reconstruction 93
4.3.2 Decomposable latent codes 93
4.3.3 Co-occurrent patch statistics 94
4.3.4 Overall training and architecture 94
4.4 Experiments 95
4.4.1 Experimental setup 95
4.4.2 Image embedding 97
4.4.3 Swapping to produce image hybrids 98
4.4.4 Analysis of our method 101
4.4.5 Image editing via latent space operations 101
4.4.6 Interactive user interface for image editing 103
4.4.7 Architecture 104
4.4.8 Training details 107
4.4.9 Additional visual results 107
4.4.10 Additional comparison to existing methods 108
4.4.11 Corruption study of Self-Similarity Distance and SIFID 115
4.5 Discussion 116
4.5.1 Discussion on broader social impact 117

5 Conclusions and Discussion 119

Bibliography 120

List of Figures

1.1 Why do we need machine learning for image editing? 1
1.2 Dilemma in utilizing machine learning in image editing 2

2.1 SPADE overview 4
2.2 SPADE normalization module 8
2.3 SPADE results on uniform labels 9
2.4 SPADE architecture 10
2.5 SPADE results and comparison on COCO-Stuff 11
2.6 SPADE results and comparison on ADE20K outdoor and Cityscapes dataset 11
2.7 SPADE results on the Flickr Landscapes dataset 12
2.8 SPADE results on COCO-Stuff dataset 15
2.9 SPADE multimodal synthesis 16
2.10 SPADE design 19
2.11 SPADE Residual Block 19
2.12 SPADE generator architecture 20
2.13 SPADE discriminator architecture 20
2.14 SPADE encoder architecture 21
2.15 SPADE architecture overview 21
2.16 SPADE additional results on COCO-Stuff (I) 25
2.17 SPADE additional results on COCO-Stuff (II) 26
2.18 SPADE additional comparison on ADE20K 27
2.19 SPADE additional comparison on ADE20K-outdoor 28
2.20 SPADE additional comparison on Cityscapes 29
2.21 Multi-modal synthesis results of SPADE on Flickr Landscapes 30

3.1 Difference between paired and unpaired setting 32
3.2 CycleGAN design diagram 37
3.3 Reconstructed images of CycleGAN 38
3.4 CycleGAN overview 41

3.5 CycleGAN comparison on Cityscapes 41
3.6 CycleGAN comparison on Maps 42
3.7 CycleGAN ablation study on Cityscapes 45
3.8 CycleGAN on paired datasets 46
3.9 Effect of the identity mapping loss of CycleGAN 47
3.10 CycleGAN: Collection style transfer I 50
3.11 CycleGAN: Collection style transfer II 51
3.12 CycleGAN: Turning Monet's paintings into photographs 52
3.13 CycleGAN on several translation problems 53
3.14 CycleGAN on photo enhancement 54
3.15 Comparison of CycleGAN to neural style transfer 54
3.16 CycleGAN and neural style transfer methods on various applications . . 55
3.17 CycleGAN failure modes 55
3.18 Patchwise Contrastive Learning for one-sided translation 56
3.19 Patchwise Contrastive Loss 58
3.20 Result of CUT compared to the existing methods 62
3.21 Additional results of CUT 63
3.22 Results of CUT on additional datasets 64
3.23 CUT on GTA→Cityscapes 66
3.24 CUT ablations 68
3.25 Qualitative ablation results of CUT 69
3.26 Identity loss $\mathcal{L}_{\text{PatchNCE}}(G, H, Y)$ on domain Y adds stability 69
3.27 Visualizing the learned similarity of CUT by G_{enc} 71
3.28 Distribution matching of CUT 72
3.29 Additional applications of CUT 72
3.30 High-res single image translation of CUT 74
3.31 CUT additional result on high-res painting to photo translation (I) . . . 75
3.32 CUT additional result on high-res painting to photo translation (II) . . . 76
3.33 CyCADA - design 80
3.34 CyCADA: Cross Season Image Translation 84
3.35 CyCADA: GTA5 to Cityscapes Semantic Segmentation 85
3.36 CyCADA: GTA5 to CityScapes Image Translation 86

4.1 Swapping Autoencoder: disentangling texture from structure 89
4.2 Swapping Autoencoder architecture 92
4.3 Swapping Autoencoder embedding and reconstruction 97
4.4 Swapping Autoencoder results on LSUN Church and Bedroom 98
4.5 Swapping Autoencoder comparison to existing methods 99
4.6 Swapping Autoencoder perceptual study 100

4.7 Swapping Autoencoder continuous interpolation 102
4.8 Swapping Autoencoder UI 103
4.9 Swapping Autoencoder encoder and generator detailed architecture . . . 105
4.10 Swapping Autoencoder Co-occurrence patch discriminator design 106
4.11 Swapping Autoencoder of the FFHQ model 110
4.12 Swapping Autoencoder region editing on faces 110
4.13 Swapping Autoencoder results on Flickr Mountains 111
4.14 Swapping Autoencoder global editing 112
4.15 Swapping Autoencoder user-edited results on Flickr Mountains 112
4.16 Swapping Autoencoder additional results on LSUN 113
4.17 Swapping Autoencoder comparison to existing methods 114
4.18 Swapping Autoencoder user-guided image translation 115
4.19 Swapping Autoencoder validation of the self-similarity matrix distance . 116

List of Tables

2.1 SPADE comparison to existing methods 12
2.2 SPADE user study 14
2.3 Effectiveness of SPADE 17
2.4 Variations on the SPADE architecture 17
2.5 SPADE ablation study 23

3.1 AMT "real vs fake" test on maps$\leftrightarrow$aerial photos at 256×256 resolution. 44
3.2 FCN-scores for different methods, evaluated on Cityscapes labels$\rightarrow$photo. 44
3.3 Classification performance of photo$\rightarrow$labels for different methods on cityscapes. 44
3.4 Ablation study: FCN-scores for different variants of our method, evaluated on Cityscapes labels$\rightarrow$photo. 45
3.5 Ablation study: classification performance of photo$\rightarrow$labels for different losses, evaluated on Cityscapes. 46
3.6 Comparison of CUT to baselines 65
3.7 Adaptation between seasons using CyCADA 84
3.8 Domain Adaptation between GTA5 and Cityscapes using CyCADA . . . 85

4.1 Swapping Autoencoder realism of swap-generated images 99
4.2 Swapping Autoencoder FID of swapping 109
4.3 Swapping Autoencoder FID of swapping on the training set 109
4.4 Swapping Autoencoder's fake detectability 118

Acknowledgments

First and foremost, I would like to thank my advisor Alyosha. I did not realize at the beginning of my grad school how lucky I was to have him as my advisor. Without him, I do not think I could have such a happy and fulfilling journey in the past five years. Coming back from military service, I had little background knowledge in both computer vision and computational photography. Alyosha patiently waited for me to grow up and supported me academically, financially, and mentally. I remember the night of my first paper submission. He stayed the whole night with me, sleeping on the couch behind me. I almost do not want to graduate, because I know I will miss him. Still, his advice on research and life will stay in my heart forever.

I also can't express how much I am grateful for my Ph.D. brothers Jun-Yan Zhu and Richard Zhang, who basically shaped the way I do research. I learned so much, from practical things like defining typesetting macros to brainstorming next research directions. In particular, Jun-Yan has co-authored all papers with me. Richard has shown me how a research project can have a real impact on products and helped me decide on my next career. My research would not have been the same if it were not for them.

I am also grateful to my internship hosts Ming-Yu Liu and Eli Shechtman for the opportunities. At the beginning of my internship at NVIDIA with Ming-Yu, I had an audacious goal of continuing a previous project to submit to an unrelated conference. He graciously let me finish the project. Then we sprinted together. Even after my internship, he continued on refining the project so that we could demo it at SIGGRAPH Real-Time Live and also on the NVIDIA webpage that millions of people view. He also chatted with me about many aspects of life. Eli had hosted me for two years, including the part-time involvement during school years. Even though the first project ended up taking a very long time, he always accommodated my requests and provided deep insights into the project. After all, I am going to spend more time with him in my next career.

I would also like to thank all my colleagues and project partners, Phillip, Ting-Chun, Judy, Eric, Kate, Trevor, Cynthia, and Oliver. I thank my committee members Jitendra Malik and Angjoo Kanazawa for providing deep insights and leading at the group meetings and discussions. I am also grateful for the moments I had with Jaakko Lehtinen and Tongzhou Wang for the many late night experiments and discussions. I will also remember the chats, hikes, and runs I had with Deepak Pathak, Allan Jabri, Shiry Ginosar, Tinghui Zhou, Yu Sun, David Fouhey, Andrew Owens, Tim Brooks, Bill Peebles, Arnab Ghosh, Jacob Huh, Karttikeya Mangalam, and Shubham Goel.

I thank all my friends and family members to help me go through. A special mention to the members of KSEA (Korean American Scientists and Engineers Association) for socially supporting me. I express my gratitude to the Presidential Science Scholarship and Samsung Scholarship for providing me with the means to pursue 11 years of university programs.

Lastly, I thank my mom, dad, and Dayoung for embracing me with love.

Chapter 1

Introduction

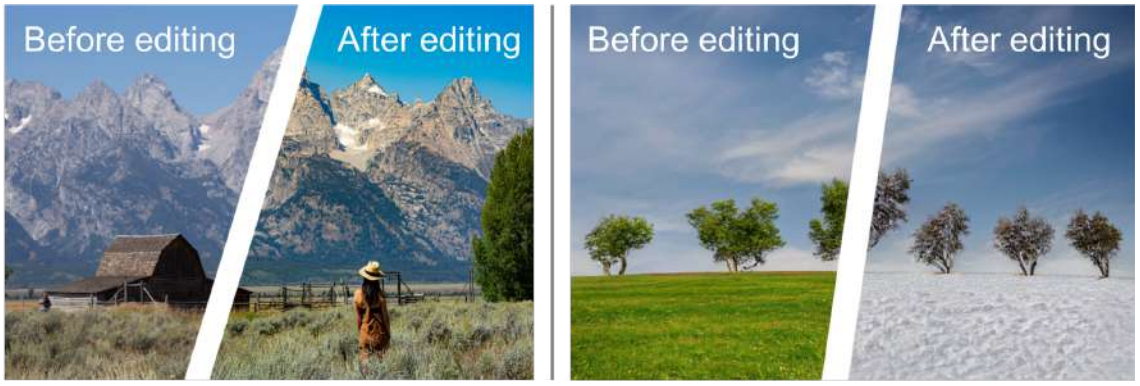


Figure 1.1: *(left)* Typical image editing operations focus on low-level features, such as changing brightness and contrast. *(right)* I aim to enable entirely new types of editing operations, such as covering the input scene with snow. The editing was performed using Park *et al.*, 2020 [1] (see Chapter 4).

With the rise of digital cameras, smartphones and social media, photography has become widely accessible to the general public. As a hobbiest photographer, I also carry a digital camera, whether it's a smartphone or a full-fledged DSLR, and attempt to record my favorite life moments. However, more often than not, the photos straight out of the camera don't turn out to be as I wanted. Sometimes they don't replicate the real world as it is. Even if they do, they may still not capture my subjective sensation of that moment.

Likely due to this, the usage of photo editing software is widespread. For example, 10 out of the top 40 most downloaded application on App Store in 2019 were photo editing tools [2]. However, the range of possible operations by these apps are still limited. For example, they offer ways to change low-level characteristics such as brightness and contrast, but there is no good way to edit images at a higher conceptual level, such as adding snow to the image (Figure 1.1).

Why are high level editing operations challenging? Most existing photo editing methods do not possess *generic visual knowledge* acquired prior to the editing tasks at use time. Instead, they rely on hand-designed techniques to reshuffle and transform the input pixels to simulate the desired editing effect. Certainly, there exist many classic works in designing these techniques, such as finding the closest match in reference images by looking at nearby patches [3–5], combining different images across frequency bands [6–8], or morphing images by warping and blending [9–11]. However, to add snow, its visual appearance needs to be acquired *a priori*, independent of the test input images. This is where machine learning can come into play. We can gain visual knowledge from a separate dataset, and use that for creating new realistic visual that is nowhere present in the input images.

However, there is subtlety in using machine learning for image editing. There seems to be a fundamental conflict: what information should be gleaned from the dataset versus information that must be retained from the input image? If the output image relies too much on the dataset, it will retain no resemblance to the input, so can hardly be called "editing", whereas relying too much on the input lessens the value of the dataset (Figure 1.2). Unconditional image generation models like GANs [12, 13] are examples of such cases: they are exceptionally good at learning from the dataset, but do not provide a well-grounded method for applying the acquired knowledge for editing *existing* images. This conflict can be viewed as a disentanglement problem. Starting from image pixels, one needs to factor out the visual information which is specific to a given image from information that is applicable across the dataset.

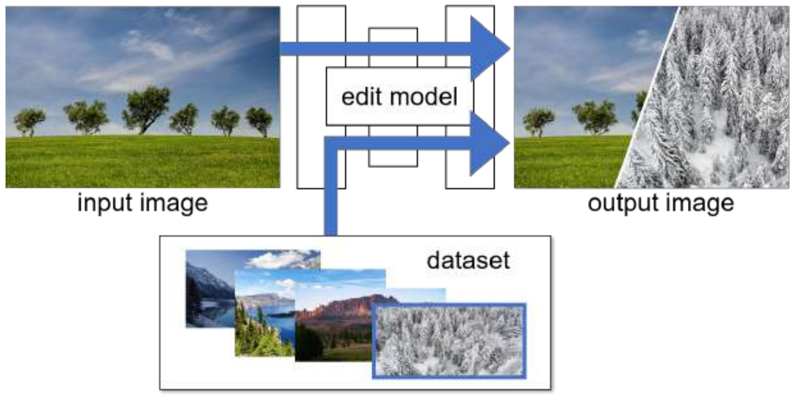


Figure 1.2: The desired image editing model must harmonize the input and visuals of the training dataset. The output image needs to contain new visual obtained from the dataset, but still be in resemblance to the original input image.

The presentation of the thesis is organized in terms of gradually reducing the amount of required training time supervision. In Chapter 2, I start out by studying the simplest setting: for every pixel and image in the dataset, we directly supervise what an ideal output should be. While such supervised approach can produce powerful results, collecting ground truth targets for every single pixel is very costly, if not impossible. In this regard, Chapter 3 proceeds by reducing the amount of supervision, from per-pixel to per-image. By doing so, we attain a more versatile image manipulation model that can be used for various tasks, ranging from learning a style of an artist to improving visual recognition of perception modules trained in virtual environment. Lastly, in Chapter 4, I propose an approach that can still discover disentangled representation of structure and texture in a fully unsupervised setting with no per-pixel or per-image labels.

Chapter 2

Learning Representation for Image Editing from Paired Examples

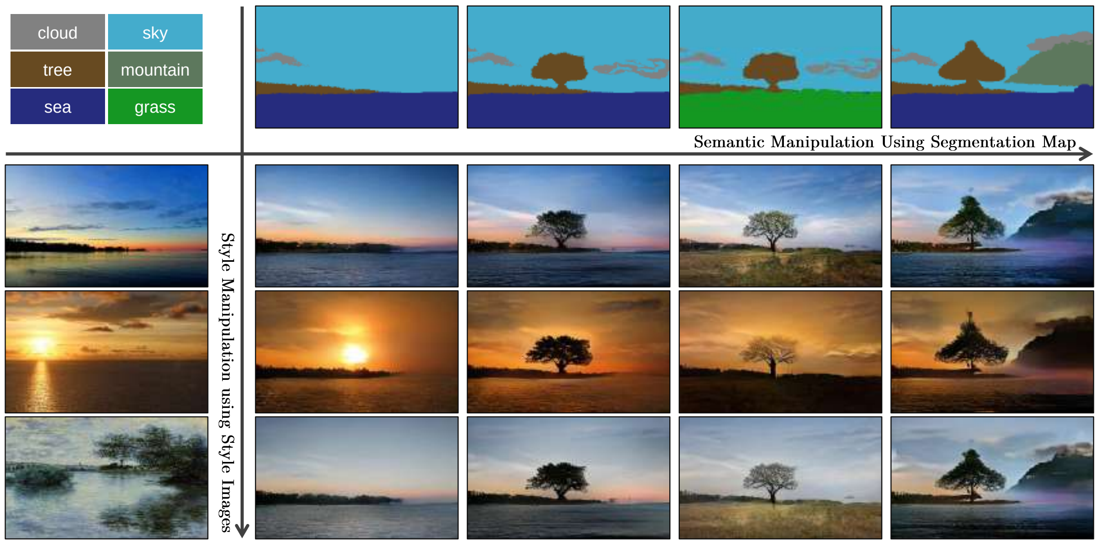

cloud
sky
tree
mountain
sea
grass
Semantic Manipulation Using Segmentation Map
Style Manipulation using Style Images

Figure 2.1: Our model allows user control over both semantic and style as synthesizing an image. The semantic (e.g., the existence of a tree) is controlled via a label map (the top row), while the style is controlled via the reference style image (the leftmost column). Please visit our website for interactive image synthesis demos.

How can we train a model that takes an image as input and produces an edited image as output? A conceptually straightforward way is to employ supervised learning. That is, our model can be trained on a dataset consisting of desired input – output pairs. In this scenario, we are essentially providing an exampler output for each pixel. The focus of this chapter is designing an effective framework, an architectural design in particular, that can produce convincing output images.

We propose spatially-adaptive normalization, a simple but effective layer for

synthesizing photorealistic images given an input semantic layout. Previous methods directly feed the semantic layout as input to the deep network, which is then processed through stacks of convolution, normalization, and nonlinearity layers. We show that this is suboptimal as the normalization layers tend to "wash away" semantic information. To address the issue, we propose using the input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation. Experiments on several challenging datasets demonstrate the advantage of the proposed method over existing approaches, regarding both visual fidelity and alignment with input layouts. Finally, our model allows user control over both semantic and style. Code is available at `https://github.com/NVlabs/SPADE`[1].

2.1 Introduction

Conditional image synthesis refers to the task of generating photorealistic images conditioning on certain input data. Seminal work computes the output image by stitching pieces from a single image (e.g., Image Analogies [4]) or using an image collection [15–19]. Recent methods directly learn the mapping using neural networks [20–27]. The latter methods are faster and require no external database of images.

We are interested in a specific form of conditional image synthesis, which is converting a semantic segmentation mask to a photorealistic image. This form has a wide range of applications such as content generation and image editing [20–22]. We refer to this form as semantic image synthesis. In this paper, we show that the conventional network architecture [20, 22], which is built by stacking convolutional, normalization, and nonlinearity layers, is at best suboptimal because their normalization layers tend to "wash away" information contained in the input semantic masks. To address the issue, we propose *spatially-adaptive normalization*, a conditional normalization layer that modulates the activations using input semantic layouts through a spatially-adaptive, learned transformation and can effectively propagate the semantic information throughout the network.

We conduct experiments on several challenging datasets including the COCO-Stuff [28, 29], the ADE20K [30], and the Cityscapes [31]. We show that with the help of our spatially-adaptive normalization layer, a compact network can synthesize significantly better results compared to several state-of-the-art methods. Additionally, an extensive ablation study demonstrates the effectiveness of the proposed normalization layer against several variants for the semantic image synthesis

[1]This work was first published as *Semantic Image Synthesis with Spatially Adaptive Normalization* in CVPR, 2019 [14].

task. Finally, our method supports multi-modal and style-guided image synthesis, enabling controllable, diverse outputs, as shown in Figure 2.1. Also, please see our SIGGRAPH 2019 Real-Time Live demo and try our online demo by yourself.

2.2 Related Work

Deep generative models can learn to synthesize images. Recent methods include generative adversarial networks (GANs) [12] and variational autoencoder (VAE) [32]. Our work is built on GANs but aims for the conditional image synthesis task. The GANs consist of a generator and a discriminator where the goal of the generator is to produce realistic images so that the discriminator cannot tell the synthesized images apart from the real ones.

Conditional image synthesis exists in many forms that differ in the type of input data. For example, class-conditional models [27, 33–36] learn to synthesize images given category labels. Researchers have explored various models for generating images based on text [24, 37–39]. Another widely-used form is image-to-image translation based on a type of conditional GANs [20, 40–46], where both input and output are images. Compared to earlier non-parametric methods [4, 15, 18], learning-based methods typically run faster during test time and produce more realistic results. In this work, we focus on converting segmentation masks to photorealistic images. We assume the training dataset contains registered segmentation masks and images. With the proposed spatially-adaptive normalization, our compact network achieves better results compared to leading methods.

Unconditional normalization layers have been an important component in modern deep networks and can be found in various classifiers, including the Local Response Normalization in the AlexNet [47] and the Batch Normalization (BatchNorm) in the Inception-v2 network [48]. Other popular normalization layers include the Instance Normalization (InstanceNorm) [49], the Layer Normalization [50], the Group Normalization [51], and the Weight Normalization [52]. We label these normalization layers as unconditional as they do not depend on external data in contrast to the conditional normalization layers discussed below.

Conditional normalization layers include the Conditional Batch Normalization (Conditional BatchNorm) [53] and Adaptive Instance Normalization (AdaIN) [54]. Both were first used in the style transfer task and later adopted in various vision tasks [26, 27, 34, 35, 44, 55–59]. Different from the earlier normalization techniques, conditional normalization layers require external data and generally operate as follows. First, layer activations are normalized to zero mean and unit deviation.

Then the normalized activations are *denormalized* by modulating the activation using a learned affine transformation whose parameters are inferred from external data. For style transfer tasks [53, 54], the affine parameters are used to control the global style of the output, and hence are uniform across spatial coordinates. In contrast, our proposed normalization layer applies a spatially-varying affine transformation, making it suitable for image synthesis from semantic masks. Wang *et al.* proposed a closely related method for image super-resolution [57]. Both methods are built on spatially-adaptive modulation layers that condition on semantic inputs. While they aim to incorporate semantic information into super-resolution, our goal is to design a generator for style and semantics disentanglement. We focus on providing the semantic information in the context of modulating normalized activations. We use semantic maps in different scales, which enables coarse-to-fine generation. The reader is encouraged to review their work for more details.

2.3 Semantic Image Synthesis

Let $\mathbf{m} \in \mathbb{L}^{H \times W}$ be a semantic segmentation mask where $\mathbb{L}$ is a set of integers denoting the semantic labels, and H and W are the image height and width. Each entry in $\mathbf{m}$ denotes the semantic label of a pixel. We aim to learn a mapping function that can convert an input segmentation mask $\mathbf{m}$ to a photorealistic image.

Spatially-adaptive denormalization. Let $\mathbf{h}^i$ denote the activations of the i-th layer of a deep convolutional network for a batch of N samples. Let C^i be the number of channels in the layer. Let H^i and W^i be the height and width of the activation map in the layer. We propose a new conditional normalization method called the SPatially-Adaptive (DE)normalization[2] (SPADE). Similar to the Batch Normalization [48], the activation is normalized in the channel-wise manner and then modulated with learned scale and bias. Figure 2.2 illustrates the SPADE design. The activation value at site $(n \in N, c \in C^i, y \in H^i, x \in W^i)$ is

$$\gamma^i_{c,y,x}(\mathbf{m}) \frac{h^i_{n,c,y,x} - \mu^i_c}{\sigma^i_c} + \beta^i_{c,y,x}(\mathbf{m}) \tag{2.1}$$

where $h^i_{n,c,y,x}$ is the activation at the site before normalization and μ^i_c and σ^i_c are the

[2]Conditional normalization [53, 54] uses external data to denormalize the normalized activations; i.e., the denormalization part is conditional.

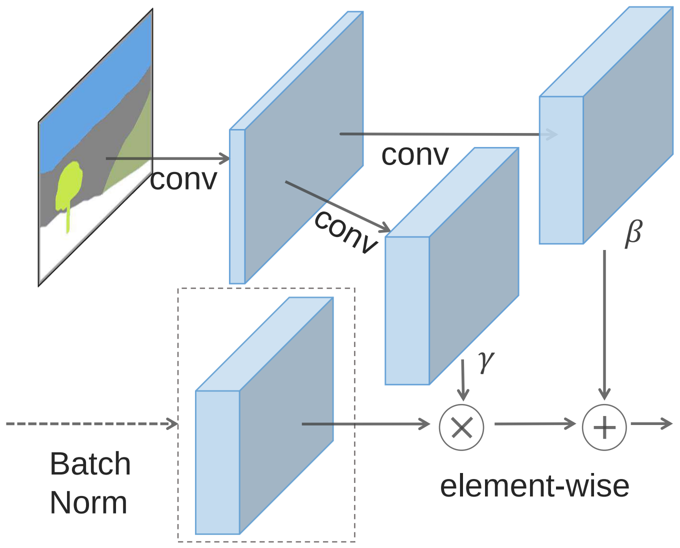


Figure 2.2: In the SPADE, the mask is first projected onto an embedding space and then convolved to produce the modulation parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. Unlike prior conditional normalization methods, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are not vectors, but tensors with spatial dimensions. The produced $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are multiplied and added to the normalized activation element-wise.

mean and standard deviation of the activations in channel c:

$$\mu_c^i = \frac{1}{NH^iW^i} \sum_{n,y,x} h^i_{n,c,y,x} \tag{2.2}$$

$$\sigma_c^i = \sqrt{\frac{1}{NH^iW^i} \sum_{n,y,x} \Big((h^i_{n,c,y,x})^2 - (\mu_c^i)^2 \Big)}. \tag{2.3}$$

The variables $\gamma^i_{c,y,x}(\mathbf{m})$ and $\beta^i_{c,y,x}(\mathbf{m})$ in (2.1) are the learned modulation parameters of the normalization layer. In contrast to the BatchNorm [48], they depend on the input segmentation mask and vary with respect to the location (y, x). We use the symbol $\gamma^i_{c,y,x}$ and $\beta^i_{c,y,x}$ to denote the functions that convert $\mathbf{m}$ to the scaling and bias values at the site (c, y, x) in the i-th activation map. We implement the functions $\gamma^i_{c,y,x}$ and $\beta^i_{c,y,x}$ using a simple two-layer convolutional network, whose design is in the appendix.

In fact, SPADE is related to, and is a generalization of several existing normalization layers. First, replacing the segmentation mask $\mathbf{m}$ with the image class label and making the modulation parameters spatially-invariant (i.e., $\gamma^i_{c,y_1,x_1} \equiv \gamma^i_{c,y_2,x_2}$ and $\beta^i_{c,y_1,x_1} \equiv \beta^i_{c,y_2,x_2}$ for any $y_1, y_2 \in \{1, 2, ..., H^i\}$ and $x_1, x_2 \in \{1, 2, ..., W^i\}$), we arrive at the form of the Conditional BatchNorm [53]. Indeed, for any spatially-invariant conditional data, our method reduces to the Conditional BatchNorm. Similarly,

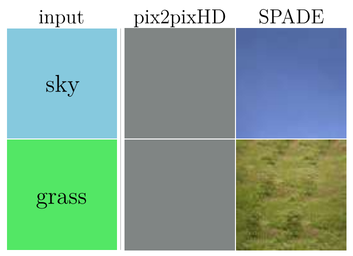


Figure 2.3: Comparing results given uniform segmentation maps: while the SPADE generator produces plausible textures, the pix2pixHD generator [22] produces two identical outputs due to the loss of the semantic information after the normalization layer.

we can arrive at the AdaIN [54] by replacing **m** with a real image, making the modulation parameters spatially-invariant, and setting $N = 1$. As the modulation parameters are adaptive to the input segmentation mask, the proposed SPADE is better suited for semantic image synthesis.

SPADE generator. With the SPADE, there is no need to feed the segmentation map to the first layer of the generator, since the learned modulation parameters have encoded enough information about the label layout. Therefore, we discard encoder part of the generator, which is commonly used in recent architectures [20, 22]. This simplification results in a more lightweight network. Furthermore, similarly to existing class-conditional generators [26, 34, 35], the new generator can take a random vector as input, enabling a simple and natural way for multi-modal synthesis [43, 44].

Figure 2.4 illustrates our generator architecture, which employs several ResNet blocks [60] with upsampling layers. The modulation parameters of all the normalization layers are learned using the SPADE. Since each residual block operates at a different scale, we downsample the semantic mask to match the spatial resolution.

We train the generator with the same multi-scale discriminator and loss function used in pix2pixHD [22] except that we replace the least squared loss term [61] with the hinge loss term [26, 62, 63]. We test several ResNet-based discriminators used in recent unconditional GANs [34, 35, 64] but observe similar results at the cost of a higher GPU memory requirement. Adding the SPADE to the discriminator also yields a similar performance. For the loss function, we observe that removing any loss term in the pix2pixHD loss function lead to degraded generation results.

Why does the SPADE work better? A short answer is that it can better preserve semantic information against common normalization layers. Specifically, while normalization layers such as the InstanceNorm [49] are essential pieces in almost all the state-of-the-art conditional image synthesis models [22], they tend to wash away semantic information when applied to uniform or flat segmentation masks.

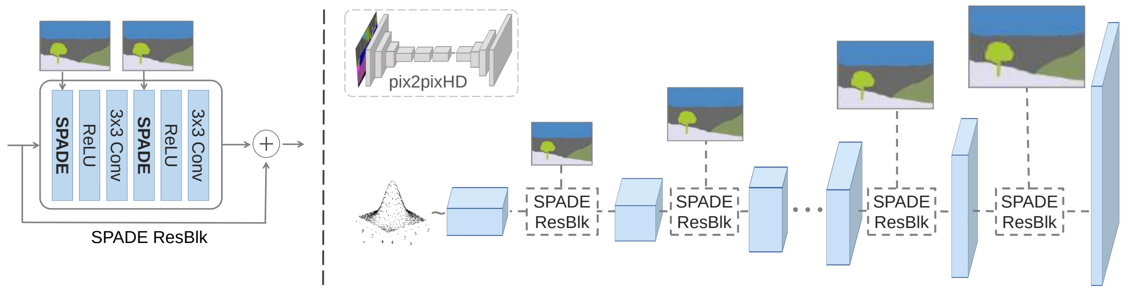


Figure 2.4: In the SPADE generator, each normalization layer uses the segmentation mask to modulate the layer activations. *(left)* Structure of one residual block with the SPADE. *(right)* The generator contains a series of the SPADE residual blocks with upsampling layers. Our architecture achieves better performance with a smaller number of parameters by removing the downsampling layers of leading image-to-image translation networks such as the pix2pixHD model [22].

Let us consider a simple module that first applies convolution to a segmentation mask and then normalization. Furthermore, let us assume that a segmentation mask with a single label is given as input to the module (e.g., all the pixels have the same label such as sky or grass). Under this setting, the convolution outputs are again uniform, with different labels having different uniform values. Now, after we apply InstanceNorm to the output, the normalized activation will become all zeros no matter what the input semantic label is given. Therefore, semantic information is totally lost. This limitation applies to a wide range of generator architectures, including pix2pixHD and its variant that concatenates the semantic mask at all intermediate layers, as long as a network applies convolution and then normalization to the semantic mask. In Figure 2.3, we empirically show this is precisely the case for pix2pixHD. Because a segmentation mask consists of a few uniform regions in general, the issue of information loss emerges when applying normalization.

In contrast, the segmentation mask in the SPADE Generator is fed through spatially adaptive modulation *without* normalization. Only activations from the previous layer are normalized. Hence, the SPADE generator can better preserve semantic information. It enjoys the benefit of normalization without losing the semantic input information.

Multi-modal synthesis. By using a random vector as the input of the generator, our architecture provides a simple way for multi-modal synthesis [43, 44]. Namely, one can attach an encoder that processes a real image into a random vector, which will be then fed to the generator. The encoder and generator form a VAE [32], in which the encoder tries to capture the style of the image, while the generator combines the encoded style and the segmentation mask information via the SPADEs

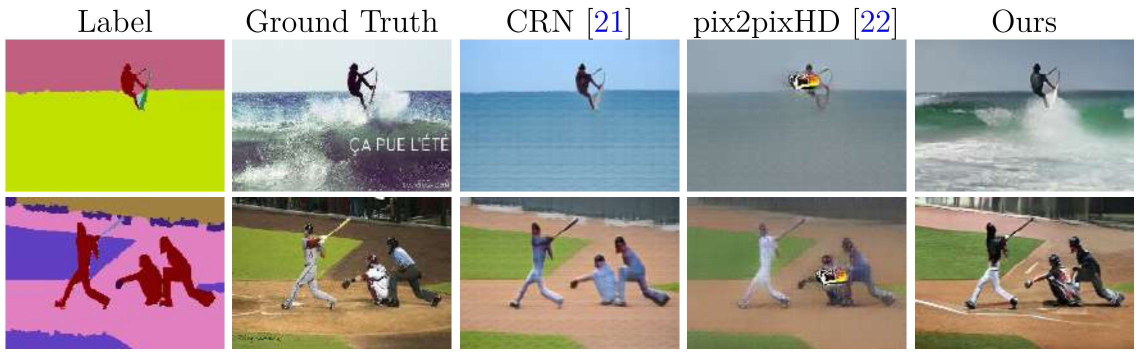


Figure 2.5: Visual comparison of semantic image synthesis results on the COCO-Stuff dataset. Our method successfully synthesizes realistic details from semantic labels.

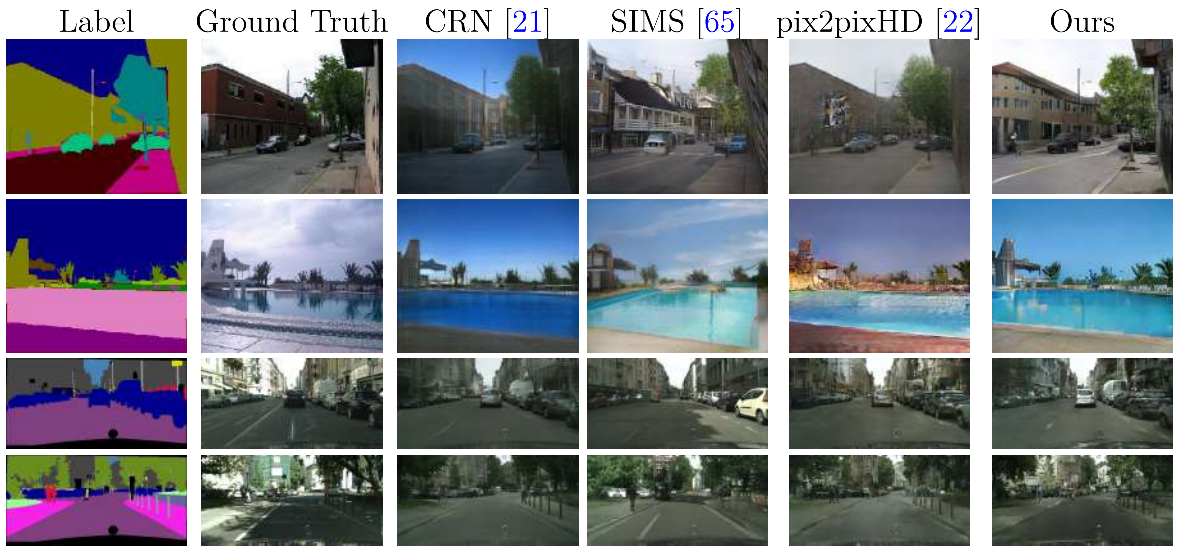


Figure 2.6: Visual comparison of semantic image synthesis results on the ADE20K outdoor and Cityscapes datasets. Our method produces realistic images while respecting the spatial semantic layout at the same time.

	COCO-Stuff			ADE20K			ADE20K-outdoor			Cityscapes		
Method	mIoU	accu	FID	mIoU	accu	FID	mIoU	accu	FID	mIoU	accu	FID
CRN [21]	23.7	40.4	70.4	22.4	68.8	73.3	16.5	68.6	99.0	52.4	77.1	104.7
SIMS [65]	N/A	N/A	N/A	N/A	N/A	N/A	13.1	74.7	67.7	47.2	75.5	**49.7**
pix2pixHD [22]	14.6	45.8	111.5	20.3	69.2	81.8	17.4	71.6	97.8	58.3	81.4	95.0
Ours	**37.4**	**67.9**	**22.6**	**38.5**	**79.9**	**33.9**	**30.8**	**82.9**	**63.3**	**62.3**	**81.9**	71.8

Table 2.1: Our method outperforms the current leading methods in semantic segmentation (mIoU and accu) and FID [66] scores on all the benchmark datasets. For the mIoU and accu, higher is better. For the FID, lower is better.

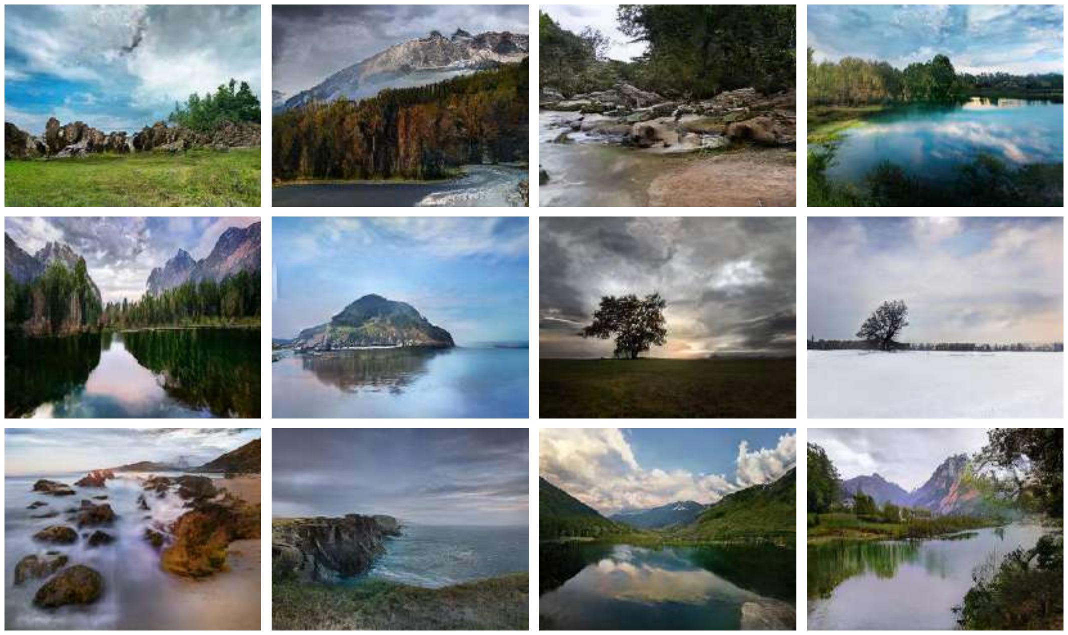


Figure 2.7: Semantic image synthesis results on the Flickr Landscapes dataset. The images were generated from semantic layout of photographs on the Flickr website.

to reconstruct the original image. The encoder also serves as a style guidance network at test time to capture the style of target images, as used in Figure 2.1. For training, we add a KL-Divergence loss term [32].

2.4 Experiments

Implementation details. We apply the Spectral Norm [63] to all the layers in both generator and discriminator. The learning rates for the generator and discriminator are 0.0001 and 0.0004, respectively [66]. We use the ADAM solver [67] with $\beta_1 = 0$ and $\beta_2 = 0.999$. All the experiments are conducted on an NVIDIA DGX1 with 8 32GB V100 GPUs. We use synchronized BatchNorm, i.e., these statistics are collected from all the GPUs.

Datasets. We conduct experiments on several datasets.

- *COCO-Stuff* [28] is derived from the COCO dataset [29]. It has $118,000$ training images and $5,000$ validation images captured from diverse scenes. It has 182 semantic classes. Due to its vast diversity, existing image synthesis models perform poorly on this dataset.
- *ADE20K* [30] consists of $20,210$ training and $2,000$ validation images. Similarly to the COCO, the dataset contains challenging scenes with 150 semantic classes.
- *ADE20K-outdoor* is a subset of the ADE20K dataset that only contains outdoor scenes, used in Qi *et al.* [65].
- *Cityscapes* dataset [31] contains street scene images in German cities. The training and validation set sizes are $3,000$ and 500, respectively. Recent work has achieved photorealistic semantic image synthesis results [23,65] on the Cityscapes dataset.
- *Flickr Landscapes.* We collect $41,000$ photos from Flickr and use $1,000$ samples for the validation set. To avoid expensive manual annotation, we use a well-trained DeepLabV2 [68] to compute input segmentation masks.

We train the competing semantic image synthesis methods on the same training set and report their results on the same validation set for each dataset.

Performance metrics. We adopt the evaluation protocol from previous work [21,22]. Specifically, we run a semantic segmentation model on the synthesized images and compare how well the predicted segmentation mask matches the ground truth input. Intuitively, if the output images are realistic, a well-trained semantic segmentation model should be able to predict the ground truth label. For measuring the segmentation accuracy, we use both the mean Intersection-over-Union (mIoU) and the pixel accuracy (accu). We use the state-of-the-art segmentation networks for each dataset: DeepLabV2 [68,69] for COCO-Stuff, UperNet101 [70] for ADE20K, and DRN-D-105 [71] for Cityscapes. In addition to the mIoU and the accu segmentation performance metrics, we use the Fréchet Inception Distance (FID) [66] to measure the distance between the distribution of synthesized results and the distribution of real images.

Baselines. We compare our method with 3 leading semantic image synthesis models: the pix2pixHD model [22], the cascaded refinement network (CRN) [21], and the semi-parametric image synthesis method (SIMS) [65]. The pix2pixHD is the current state-of-the-art GAN-based conditional image synthesis framework. The CRN uses a deep network that repeatedly refines the output from low to high resolution, while the SIMS takes a semi-parametric approach that composites real segments from a training set and refines the boundaries. Both the CRN and SIMS are mainly trained using image reconstruction loss. For a fair comparison, we train the CRN

Dataset	Ours vs. CRN	Ours vs. pix2pixHD	Ours vs. SIMS
COCO-Stuff	79.76	86.64	N/A
ADE20K	76.66	83.74	N/A
ADE20K-outdoor	66.04	79.34	85.70
Cityscapes	63.60	53.64	51.52

Table 2.2: User preference study. The numbers indicate the percentage of users who favor the results of the proposed method over those of the competing method.

and pix2pixHD models using the implementations provided by the authors. As image synthesis using the SIMS requires many queries to the training dataset, it is computationally prohibitive for a large dataset such as the COCO-stuff and the full ADE20K. Therefore, we use the results provided by the authors when available.

Quantitative comparisons. As shown in Table 2.1, our method outperforms the current state-of-the-art methods by a large margin in all the datasets. For the COCO-Stuff, our method achieves an mIoU score of 35.2, which is about 1.5 times better than the previous leading method. Our FID is also 2.2 times better than the previous leading method. We note that the SIMS model produces a lower FID score but has poor segmentation performances on the Cityscapes dataset. This is because the SIMS synthesizes an image by first stitching image patches from the training dataset. As using the real image patches, the resulting image distribution can better match the distribution of real images. However, because there is no guarantee that a perfect query (e.g., a person in a particular pose) exists in the dataset, it tends to copy objects that do not match the input segments.

Qualitative results. In Figures 2.5 and 2.6, we provide qualitative comparisons of the competing methods. We find that our method produces results with much better visual quality and fewer visible artifacts, especially for diverse scenes in the COCO-Stuff and ADE20K dataset. When the training dataset size is small, the SIMS model also renders images with good visual quality. However, the depicted content often deviates from the input segmentation mask (e.g., the shape of the swimming pool in the second row of Figure 2.6).

In Figures 2.7 and 2.8, we show more example results from the Flickr Landscape and COCO-Stuff datasets. The proposed method can generate diverse scenes with high image fidelity. More results are included in the appendix.

Human evaluation. We use the Amazon Mechanical Turk (AMT) to compare the perceived visual fidelity of our method against existing approaches. Specifically, we give the AMT workers an input segmentation mask and two synthesis outputs from different methods and ask them to choose the output image that looks more like a corresponding image of the segmentation mask. The workers are given unlimited time

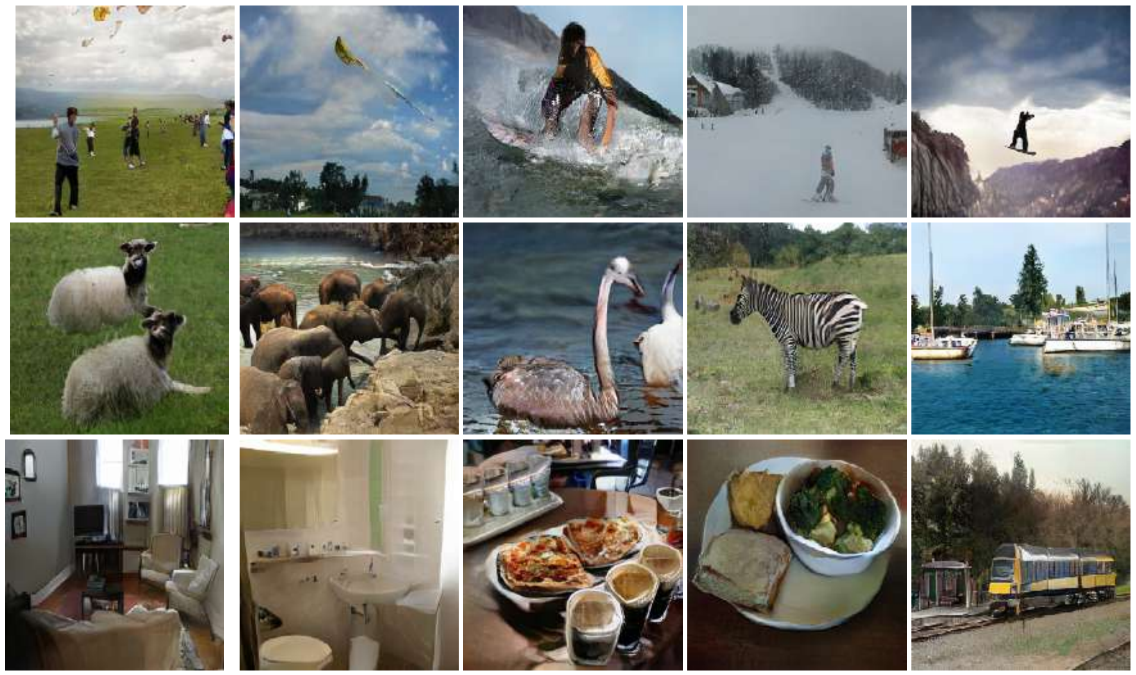


Figure 2.8: Semantic image synthesis results on COCO-Stuff. Our method successfully generates realistic images in diverse scenes ranging from animals to sports activities.

to make the selection. For each comparison, we randomly generate 500 questions for each dataset, and each question is answered by 5 different workers. For quality control, only workers with a lifetime task approval rate greater than 98% can participate in our study.

Table 2.2 shows the evaluation results. We find that users strongly favor our results on all the datasets, especially on the challenging COCO-Stuff and ADE20K datasets. For the Cityscapes, even when all the competing methods achieve high image fidelity, users still prefer our results.

Effectiveness of the SPADE. For quantifying importance of the SPADE, we introduce a strong baseline called pix2pixHD++, which combines all the techniques we find useful for enhancing the performance of pix2pixHD except the SPADE. We also train models that receive the segmentation mask input at all the intermediate layers via feature concatenation in the channel direction, which is termed as pix2pixHD++ w/ Concat. Finally, the model that combines the strong baseline with the SPADE is denoted as pix2pixHD++ w/ SPADE.

As shown in Table 2.3, the architectures with the proposed SPADE consistently outperforms its counterparts, in both the decoder-style architecture described in Figure 2.4 and more traditional encoder-decoder architecture used in the pix2pixHD. We also find that concatenating segmentation masks at all intermediate layers, a

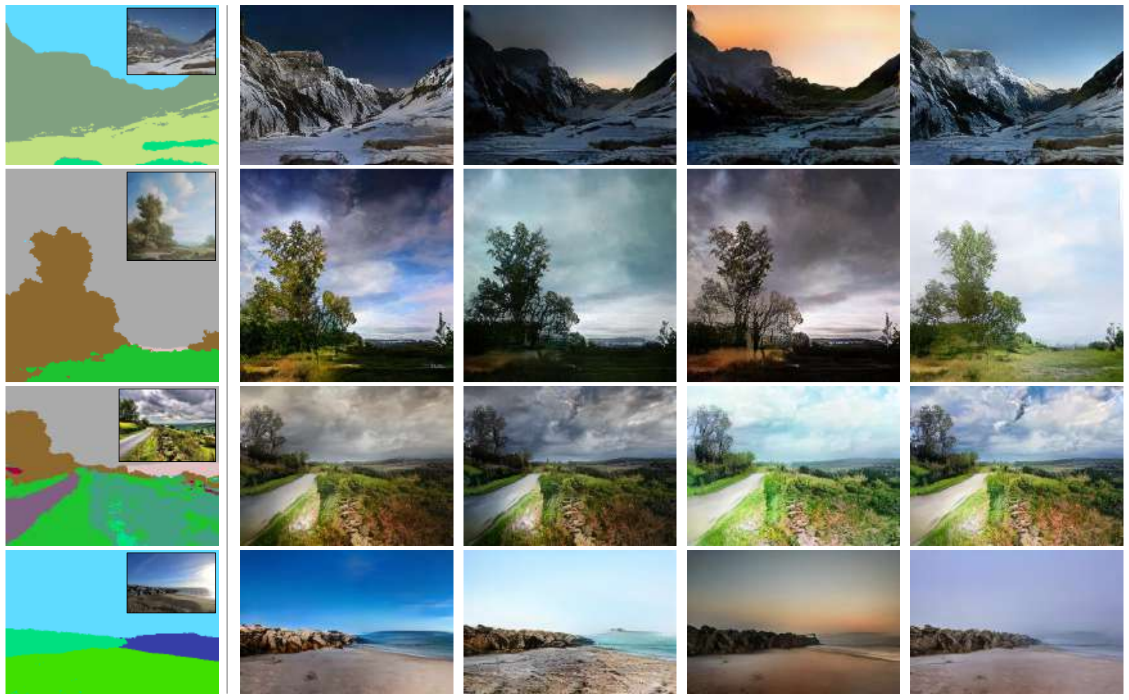


Figure 2.9: Our model attains multimodal synthesis capability when trained with the image encoder. During deployment, by using different random noise, our model synthesizes outputs with diverse appearances but all having the same semantic layouts depicted in the input mask. For reference, the ground truth image is shown inside the input segmentation mask.

Method	#param	COCO.	ADE.	City.
decoder w/ SPADE (Ours)	96M	**35.2**	38.5	62.3
compact decoder w/ SPADE	61M	**35.2**	38.0	**62.5**
decoder w/ Concat	79M	31.9	33.6	61.1
pix2pixHD++ w/ SPADE	237M	34.4	**39.0**	62.2
pix2pixHD++ w/ Concat	195M	32.9	38.9	57.1
pix2pixHD++	183M	32.7	38.3	58.8
compact pix2pixHD++	103M	31.6	37.3	57.6
pix2pixHD [22]	183M	14.6	20.3	58.3

Table 2.3: The mIoU scores are boosted when the SPADE is used, for both the decoder architecture (Figure 2.4) and encoder-decoder architecture of pix2pixHD++ (our improved baseline over pix2pixHD [22]). On the other hand, simply concatenating semantic input at every layer fails to do so. Moreover, our compact model with smaller depth at all layers outperforms all the baselines.

Method	COCO	ADE20K	Cityscapes
segmap input	35.2	38.5	62.3
random input	35.3	38.3	61.6
kernelsize 5x5	35.0	39.3	61.8
kernelsize 3x3	35.2	38.5	62.3
kernelsize 1x1	32.7	35.9	59.9
#params 141M	35.3	38.3	62.5
#params 96M	35.2	38.5	62.3
#params 61M	35.2	38.0	62.5
Sync BatchNorm	35.0	39.3	61.8
BatchNorm	33.7	37.9	61.8
InstanceNorm	33.9	37.4	58.7

Table 2.4: The SPADE generator works with different configurations. We change the input of the generator, the convolutional kernel size acting on the segmentation map, the capacity of the network, and the parameter-free normalization method. The settings used in the paper are boldfaced.

reasonable alternative to the SPADE, does not achieve the same performance as SPADE. Furthermore, the decoder-style SPADE generator works better than the strong baselines even with a smaller number of parameters.

Variations of SPADE generator. Table 2.4 reports the performance of several variations of our generator. First, we compare two types of input to the generator where one is the random noise while the other is the downsampled segmentation map. We find that both of the variants render similar performance and conclude that the modulation by SPADE alone provides sufficient signal about the input mask.

Second, we vary the type of parameter-free normalization layers before applying the modulation parameters. We observe that the SPADE works reliably across different normalization methods. Next, we vary the convolutional kernel size acting on the label map, and find that kernel size of 1x1 hurts performance, likely because it prohibits utilizing the context of the label. Lastly, we modify the capacity of the generator by changing the number of convolutional filters. We present more variations and ablations in the appendix.

Multi-modal synthesis. In Figure 2.9, we show the multimodal image synthesis results on the Flickr Landscape dataset. For the same input segmentation mask, we sample different noise inputs to achieve different outputs. More results are included in the appendix.

Semantic manipulation and guided image synthesis. In Figure 2.1, we show an application where a user draws different segmentation masks, and our model renders the corresponding landscape images. Moreover, our model allows users to choose an external style image to control the global appearances of the output image. We achieve it by replacing the input noise with the embedding vector of the style image computed by the image encoder.

2.5 Conclusion

We have proposed the spatially-adaptive normalization, which utilizes the input semantic layout while performing the affine transformation in the normalization layers. The proposed normalization leads to the first semantic image synthesis model that can produce photorealistic outputs for diverse scenes including indoor, outdoor, landscape, and street scenes. We further demonstrate its application for multi-modal synthesis and guided image synthesis.

2.6 Additional Implementation Details

Generator. The architecture of the generator consists of a series of the proposed SPADE ResBlks with nearest neighbor upsampling. We train our network using 8 GPUs simultaneously and use the synchronized version of the BatchNorm. We apply the Spectral Norm [63] to all the convolutional layers in the generator. The architectures of the proposed SPADE and SPADE ResBlk are given in Figure 2.10 and Figure 2.11, respectively. The architecture of the generator is shown in Figure 2.12.

Discriminator. The architecture of the discriminator follows the one used in the pix2pixHD method [22], which uses a multi-scale design with the InstanceNorm

(IN). The only difference is that we apply the Spectral Norm to all the convolutional layers of the discriminator. The details of the discriminator architecture is shown in Figure 2.13.

Image Encoder. The image encoder consists of 6 stride-2 convolutional layers followed by two linear layers to produce the mean and variance of the output distribution as shown in Figure 2.14.

Learning objective. We use the learning objective function in the pix2pixHD work [22] except that we replace its LSGAN loss [61] term with the Hinge loss term [26, 62, 63]. We use the same weighting among the loss terms in the objective function as that in the pix2pixHD work.

When training the proposed framework with the image encoder for multi-modal

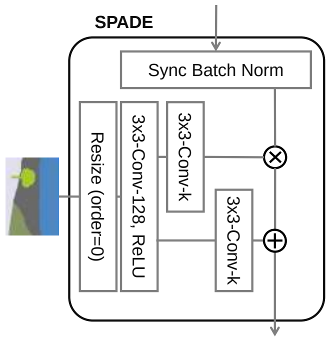


Figure 2.10: SPADE Design. The term 3x3-Conv-k denotes a 3-by-3 convolutional layer with k convolutional filters. The segmentation map is resized to match the resolution of the corresponding feature map using nearest-neighbor downsampling.

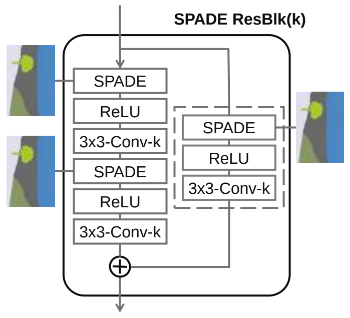


Figure 2.11: SPADE ResBlk. The residual block design largely follows that in Mescheder *et al.* [35] and Miyato *et al.* [34]. We note that for the case that the number of channels before and after the residual block is different, the skip connection is also learned (dashed box in the figure).

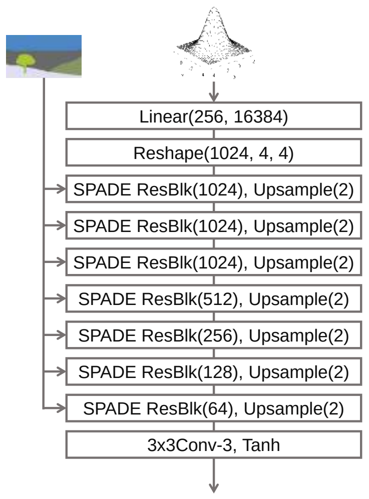

Linear(256, 16384)
Reshape(1024, 4, 4)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(512), Upsample(2)
SPADE ResBlk(256), Upsample(2)
SPADE ResBlk(128), Upsample(2)
SPADE ResBlk(64), Upsample(2)
3x3Conv-3, Tanh

Figure 2.12: SPADE Generator. Different from prior image generators [20,22], the semantic segmentation mask is passed to the generator through the proposed SPADE ResBlks in Figure 2.11.

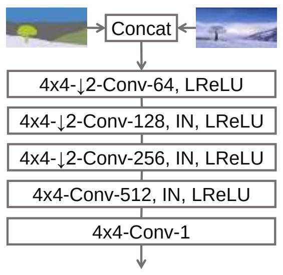


Figure 2.13: Our discriminator design largely follows that in the pix2pixHD [22]. It takes the concatenation the segmentation map and the image as input. It is based on the PatchGAN [20]. Hence, the last layer of the discriminator is a convolutional layer.

synthesis and style-guided image synthesis, we include a KL Divergence loss:

$$\mathcal{L}_{\text{KLD}} = \mathcal{D}_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

where the prior distribution $p(\mathbf{z})$ is a standard Gaussian distribution and the variational distribution q is fully determined by a mean vector and a variance vector [32]. We use the reparamterization trick [32] for back-propagating the gradient from the generator to the image encoder. The weight for the KL Divergence loss is 0.05.

In Figure 2.15, we overview the training data flow. The image encoder encodes a real image to a mean vector and a variance vector. They are used to compute the noise input to the generator via the reparameterization trick [32]. The generator also

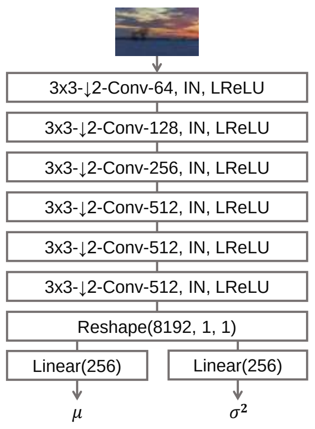

3x3-↓2-Conv-64, IN, LReLU
3x3-↓2-Conv-128, IN, LReLU
3x3-↓2-Conv-256, IN, LReLU
3x3-↓2-Conv-512, IN, LReLU
3x3-↓2-Conv-512, IN, LReLU
3x3-↓2-Conv-512, IN, LReLU
Reshape(8192, 1, 1)
Linear(256)
Linear(256)
μ
σ²

Figure 2.14: The image encoder consists a series of convolutional layers with stride 2 followed by two linear layers that output a mean vector μ and a variance vector σ.

takes the segmentation mask of the input image as input with the proposed SPADE ResBlks. The discriminator takes concatenation of the segmentation mask and the output image from the generator as input and aims to classify that as fake.

Training details. We perform 200 epochs of training on the Cityscapes and ADE20K datasets, 100 epochs of training on the COCO-Stuff dataset, and 50 epochs of training on the Flickr Landscapes dataset. The image sizes are 256×256, except the Cityscapes at 512×256. We linearly decay the learning rate to 0 from epoch 100

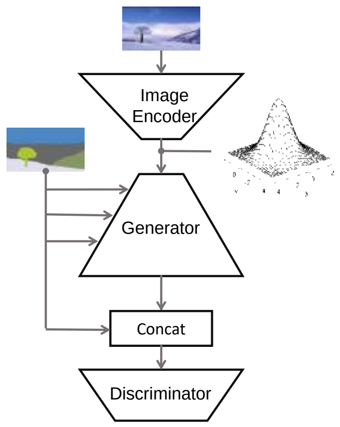


Figure 2.15: The image encoder encodes a real image to a latent representation for generating a mean vector and a variance vector. They are used to compute the noise input to the generator via the reparameterization trick [32]. The generator also takes the segmentation mask of the input image as input via the proposed SPADE ResBlks. The discriminator takes concatenation of the segmentation mask and the output image from the generator as input and aims to classify that as fake.

to 200 for the Cityscapes and ADE20K datasets. The batch size is 32. We initialize the network weights using thes Glorot initialization [72].

2.7 Additional Ablation Study

Method	COCO.	ADE.	City.
Ours	35.2	38.5	62.3
Ours w/o Perceptual loss	24.7	30.1	57.4
Ours w/o GAN feature matching loss	33.2	38.0	62.2
Ours w/ a deeper discriminator	34.9	38.3	60.9
pix2pixHD++ w/ SPADE	34.4	39.0	62.2
pix2pixHD++	32.7	38.3	58.8
pix2pixHD++ w/o Sync BatchNorm	27.4	31.8	51.1
pix2pixHD++ w/o Sync BatchNorm, and w/o Spectral Norm	26.0	31.9	52.3
pix2pixHD [22]	14.6	20.3	58.3

Table 2.5: Additional ablation study results using the mIoU metric: the table shows that both the perceptual loss and GAN feature matching loss terms are important. Making the discriminator deeper does not lead to a performance boost. The table also shows that all the components (Synchronized BatchNorm, Spectral Norm, TTUR, the Hinge loss objective, and the SPADE) used in the proposed method helps our strong baseline, pix2pixHD++.

Table 2.5 provides additional ablation study results analyzing the contribution of individual components in the proposed method. We first find that both of the perceptual loss and GAN feature matching loss inherited from the learning objective function of the pix2pixHD [22] are important. Removing any of them leads to a performance drop. We also find that increasing the depth of the discriminator by inserting one more convolutional layer to the top of the pix2pixHD discriminator does not improve the results.

In Table 2.5, we also analyze the effectiveness of each component used in our strong baseline, the pix2pixHD++ method, derived from the pix2pixHD method. We found that the Spectral Norm, synchronized BatchNorm, TTUR [66], and the hinge loss objective all contribute to the performance boost. Adding the SPADE to the strong baseline further improves the performance. Note that the pix2pixHD++ w/o Sync BatchNorm and w/o Spectral Norm still differs from the pix2pixHD in that it uses the hinge loss objective, TTUR, a large batch size, and the Glorot initialization [72].

2.8 Additional Results

In Figure 2.16, 2.17, and 2.18, we show additional synthesis results from the proposed method on the COCO-Stuff and ADE20K datasets with comparisons to those from the CRN [21] and pix2pixHD [22] methods.

In Figure 2.19 and 2.20, we show additional synthesis results from the proposed method on the ADE20K-outdoor and Cityscapes datasets with comparison to those from the CRN [21], SIMS [65], and pix2pixHD [22] methods.

In Figure 2.21, we show additional multi-modal synthesis results from the proposed method. As sampling different $\mathbf{z}$ from a standard multivariate Gaussian distribution, we synthesize images of diverse appearances.

In the accompanying video, we demonstrate our semantic image synthesis interface. We show how a user can create photorealistic landscape images by painting semantic labels on a canvas. We also show how a user can synthesize images of diverse appearances for the same semantic segmentation mask as well as transfer the appearance of a provided style image to the synthesized one.

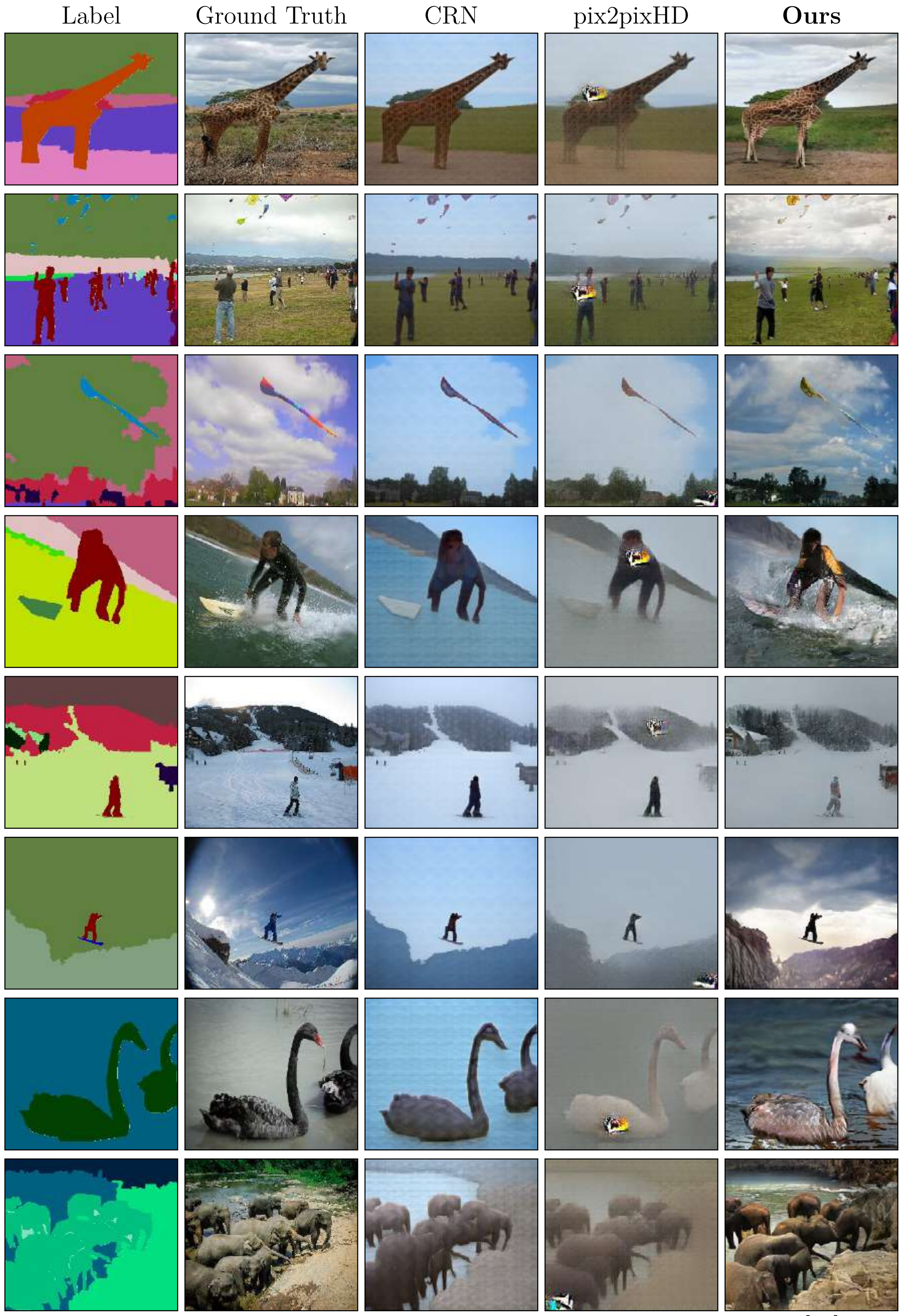


Figure 2.16: Additional results with comparison to those from the CRN [21] and pix2pixHD [22] methods on the COCO-Stuff dataset.

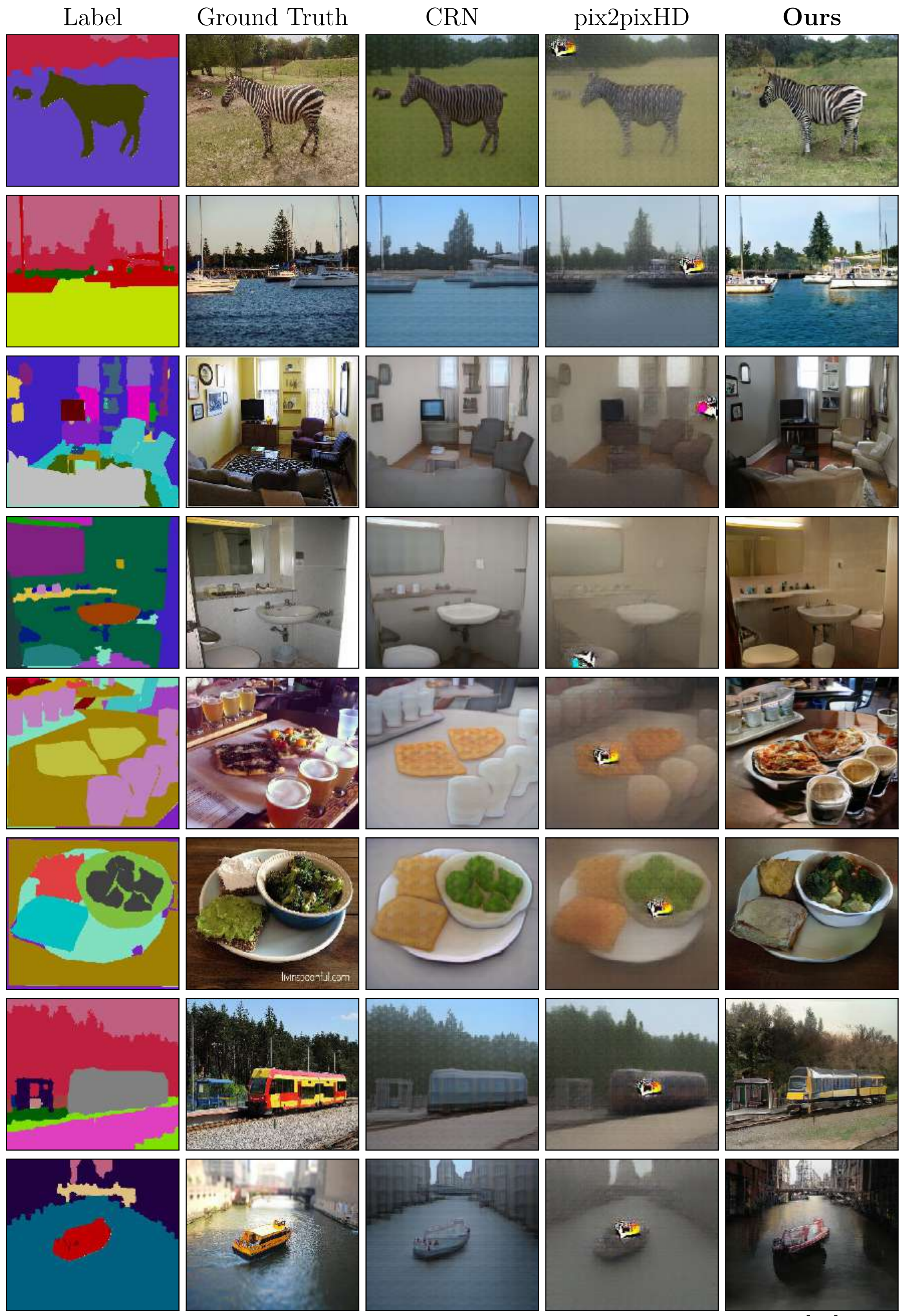


Figure 2.17: Additional results with comparison to those from the CRN [21] and pix2pixHD [22] methods on the COCO-Stuff dataset.

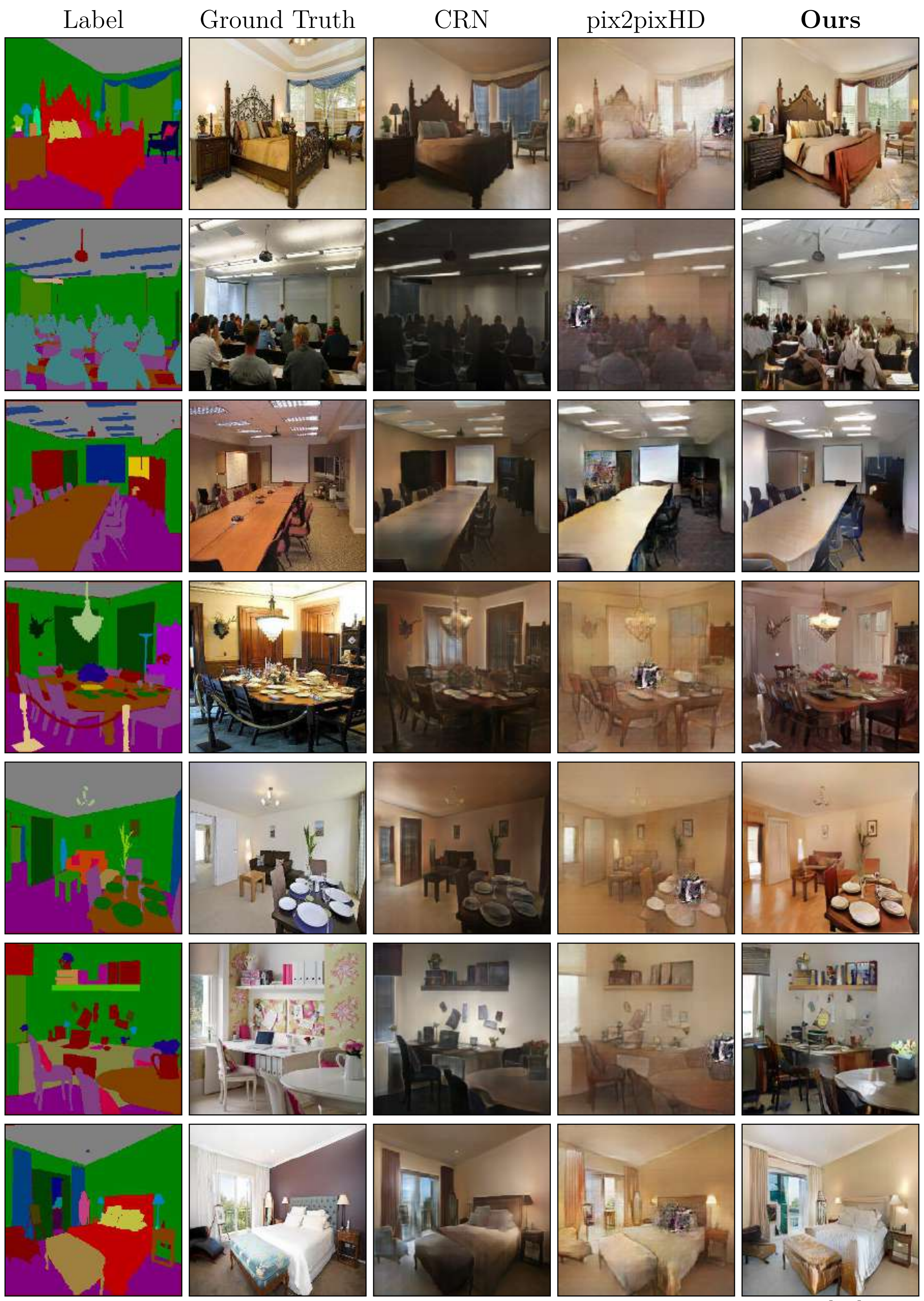


Figure 2.18: Additional results with comparison to those from the CRN [21] and pix2pixHD [22] methods on the ADE20K dataset.

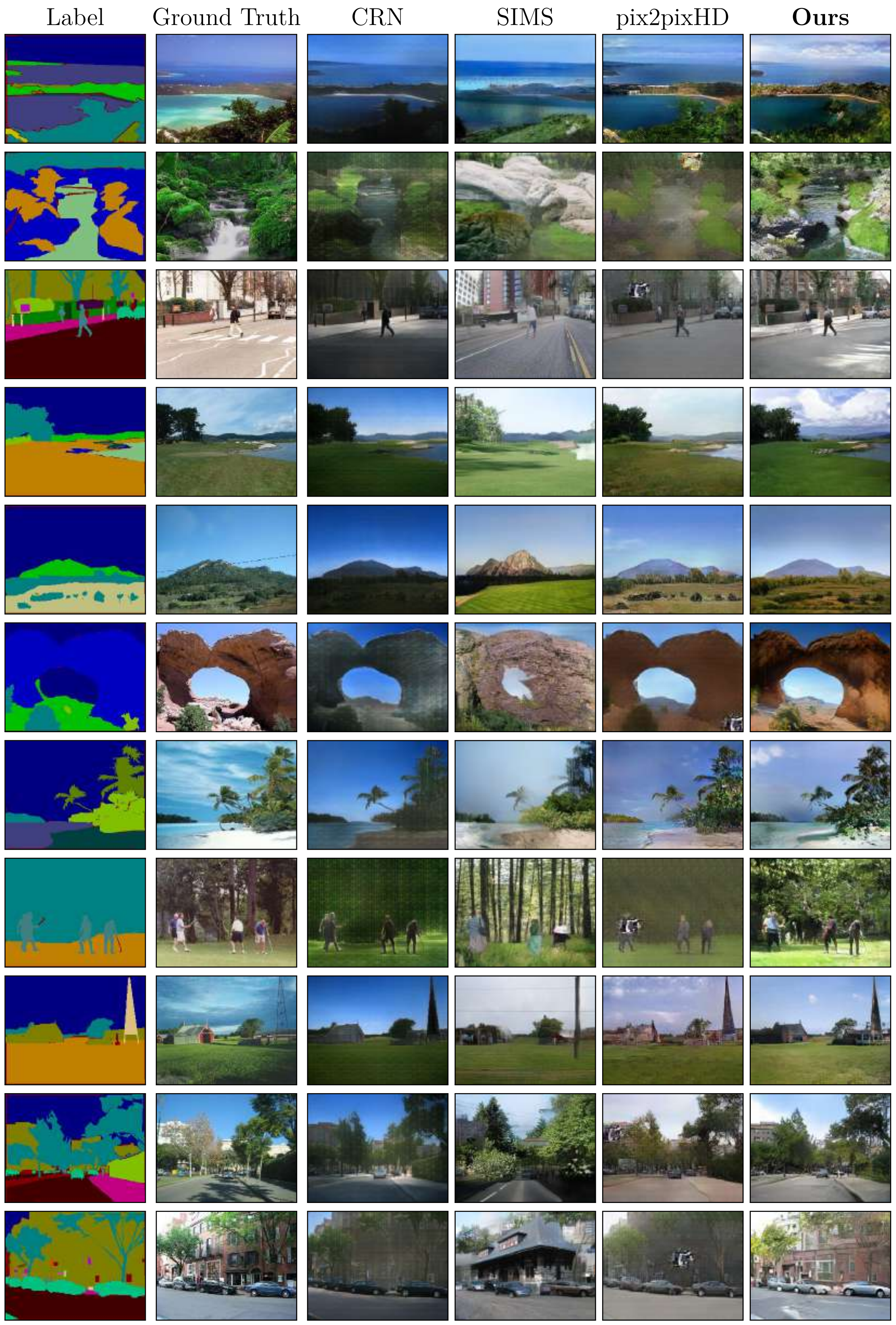


Figure 2.19: Additional results with comparison to those from the CRN [21], SIMS [65], and pix2pixHD [22] methods on the ADE20K-outdoor dataset.

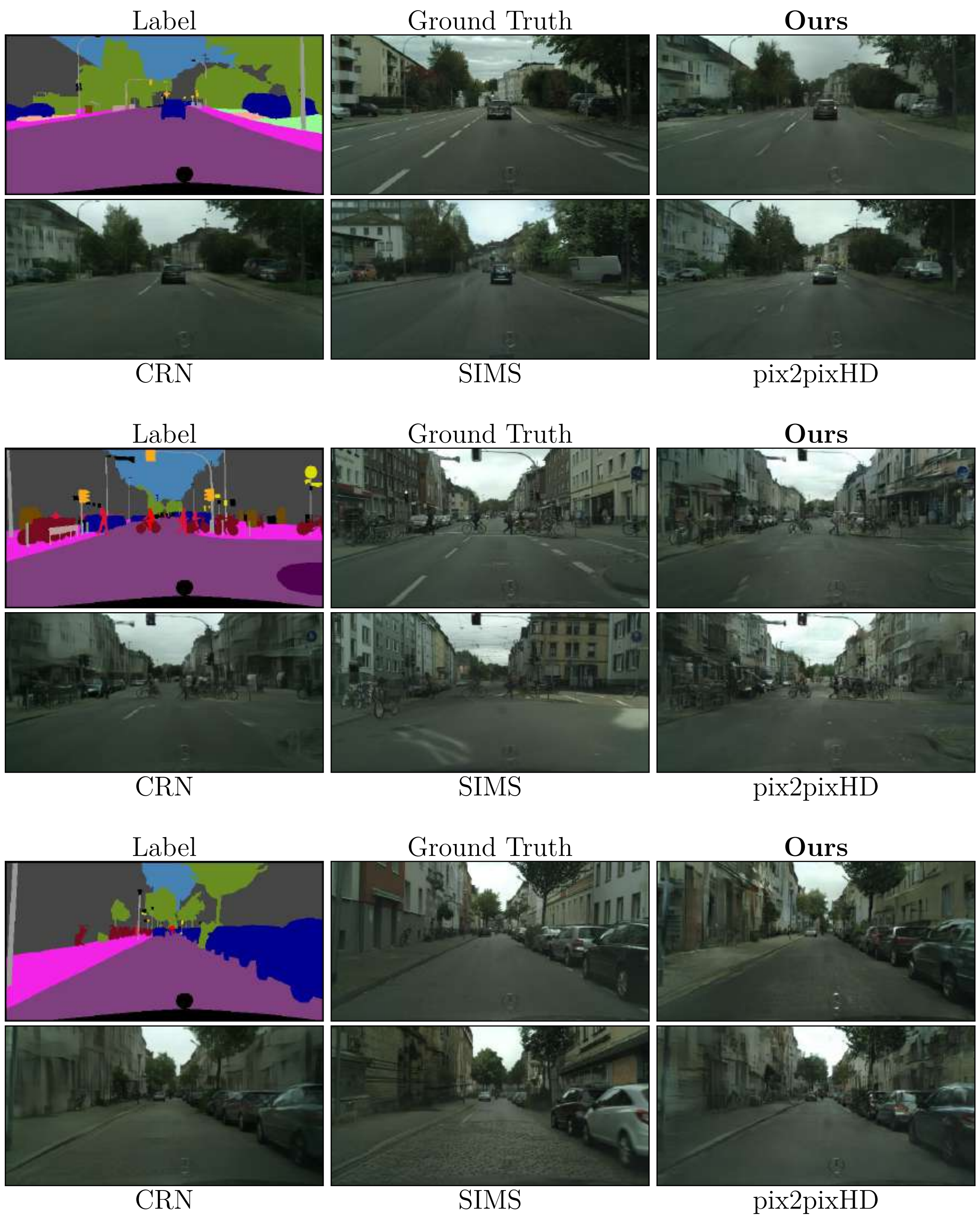


Figure 2.20: Additional results with comparison to those from the CRN [21], SIMS [65], and pix2pixHD [22] methods on the Cityscapes dataset.

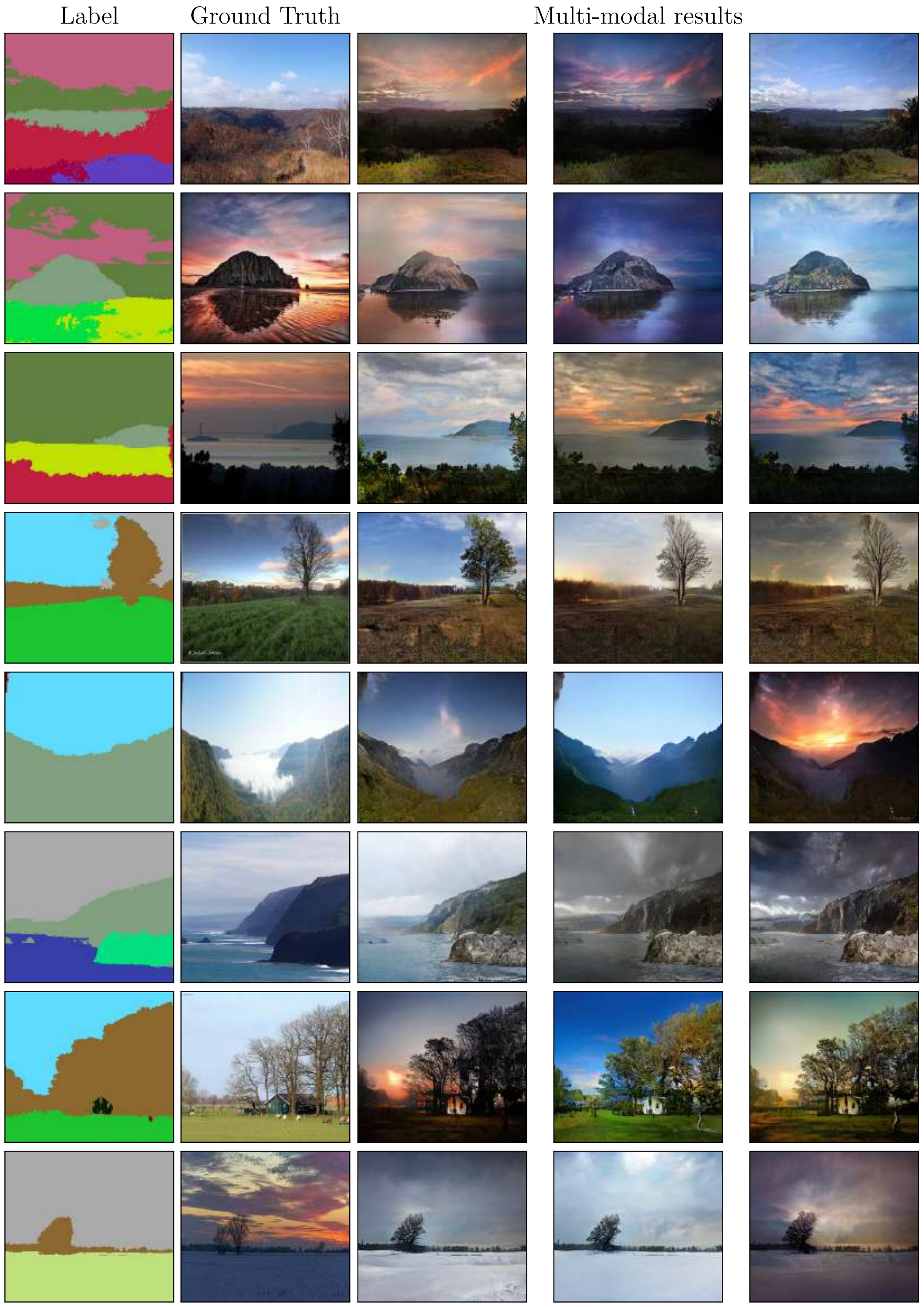


Figure 2.21: Additional multi-modal synthesis results on the Flickr Landscapes Dataset. By sampling latent vectors from a standard Gaussian distribution, we synthesize images of diverse appearances.

Chapter 3

Discovering Structure and Style from Unpaired Examples

While the per-pixel supervision provided a conceptually simple yet powerful method that directly learns controllable generation of style and content, its method cannot be used in many scenarios. For example, let's say the user would like to transform a horse in a picture into a zebra in the same pose. To utilize per-pixel supervision of the last chapter, one needs to prepare pairs of horse and zebra photos with the same pose, camera angle, background and illumination, but this would be impossible. As such, there are many cases in which a source image of one category should be translated into another category, but providing explicit examples of the input and output pairs is very difficult. Still, we can still provide a class label per each image, and prepare a dataset of *unpaired examples*.

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 using an adversarial loss. Note that this mapping is highly under-constrained, as there are combinatorial number of mappings $|Y|^{|X|}$ that satisfy this constraint. In particular, we would like to construct a mapping that avoids unnecessary modification to the input image: a horse must be changed into a zebra, but the pose and background should be preserved.

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

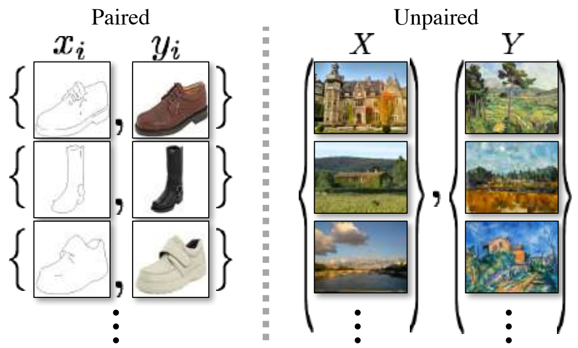


Figure 3.1: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [73]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}^M$ ($y_j \in Y$), with no information provided as to which x_i matches which y_j.

$y \in Y$ 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) [12]. 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 [74].

These issues call for adding more structure to our objective. We present two frameworks in this direction. In Section 3.2, 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 [75]. 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* [76] 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. In Section 3.3, we constrain the image translation function by simultaneously finding an embedding space that is invariant to domain differences, but is sensitive to the differences among patches of the same image. We enforce that a visual patch after translation should be the most similar to the corresponding input patch, compared to all the other patches of the same input image.

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.

3.1 Related work

Generative Adversarial Networks (GANs) [12,77] have achieved impressive results in image generation [78,79], image editing [80], and representation learning [79, 81, 82]. Recent methods adopt the same idea for conditional image generation applications, such as text2image [37], image inpainting [83], and future prediction [84], as well as to other domains like videos [85] and 3D data [86]. The key to GANs' success is the idea of an *adversarial loss* that forces the generated images to be, in principle, indistinguishable from real photos. This loss 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 images cannot be distinguished from images in the target domain.

Paired Image-to-Image Translation The idea of image-to-image translation goes back at least to Hertzmann et al.'s Image Analogies [4], who employ a non-parametric texture model [3] 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., [87]). Our approach builds on the "pix2pix" framework of Isola et al. [73], which uses a conditional generative adversarial network [12] to learn a mapping from input to output images. Similar ideas have been applied to various tasks such as generating photographs from sketches [88] or from attribute and semantic layouts [40]. However, unlike the above prior work, 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. [89] 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 [90] and cross-modal scene networks [91] use a weight-sharing strategy to learn a common representation across domains. Concurrent to Chapter 3.2, Liu et al. [42] extends the above framework with a combination of variational autoencoders [32] and generative adversarial networks [12]. Another line of concurrent work [92–94] encourages the input and output to share specific "content" features even though they may differ in "style".

These methods 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 [94], image pixel space [92], and image feature space [93]. More recently, many unpaired image-to-image translation methods were proposed by leveraging the cycle-consistency constraint or preservation of relationship among images. Please see below for more detailed discussion.

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 Chapter 3.2.5.

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 [95, 96]. In the language domain, verifying and improving translations via "back translation and reconciliation" is a technique used by human translators [75] (including, humorously, by Mark Twain [97]), as well as by machines [98]. More recently, higher-order cycle consistency has been used in structure from motion [99], 3D shape matching [100], co-segmentation [101], dense semantic alignment [76,102], and depth estimation [103]. Of these, Zhou et al. [76] and Godard et al. [103] 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. [104] independently use a similar objective for unpaired image-to-image translation, inspired by dual learning in machine translation [98].

Including our work of Chapter 3.2, the *cycle-consistency* has become the de facto method for enforcing correspondence [41, 104, 105], which learns an inverse mapping from the output domain back to the input and checks if the input can be reconstructed. Alternatively, UNIT [42] and MUNIT [44] propose to learn a shared intermediate "content" latent space. Recent works further enable multiple domains and multi-modal synthesis [43, 106–109] and improve the quality of results [110–114]. In all of the above examples, cycle-consistency is used, often in multiple aspects, between (a) two image domains [41,104,105] (b) image to latent [42–44,106,108], or (c) latent to image [43, 44]. While effective, the underlying bijective assumption behind cycle-consistency is sometimes too restrictive. Perfect reconstruction is difficult to achieve, especially when images from one domain have additional information compared to the other domain.

Neural Style Transfer [115–118] 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 primary 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 Chapter 3.2.6.

Relationship preservation. An interesting alternative approach is to encourage relationships present in the input be analogously reflected in the output. For example, perceptually similar patches *within* an input image should be similar in the output [111], output and input images share similar content regarding a pre-defined distance [119–121], vector arithmetic between input images is preserved using a margin-based triplet loss [122], distances *between* input images should be consistent in output images [123], the network should be equivariant to geometric transformations [124]. Among them, TraVeLGAN [122], DistanceGAN [123] and GcGAN [124] enable one-way translation and bypass cycle-consistency. However, they rely on relationship between entire images, or often with predefined distance functions. Here we seek to replace cycle-consistency by instead learning a cross-domain similarity function *between input and output patches* through information maximization, without relying on a pre-specified distance.

Emergent perceptual similarity in deep network embeddings. Defining a "perceptual" distance function between high-dimensional signals, e.g., images, has been a longstanding problem in computer vision and image processing. The majority of image translation work mentioned uses a per-pixel reconstruction metric, such as ℓ_1. Such metrics do not reflect human perceptual preferences and can lead to blurry results. Recently, the deep learning community has found that the VGG classification network [125] trained on ImageNet dataset [126] can be re-purposed as a "perceptual loss" [116, 127–131], which can be used in paired image translation tasks [14, 21, 22], and was known to outperform traditional metrics such as SSIM [132] and FSIM [133] on human perceptual tests [130]. In particular, the Contextual Loss [131] boosts the perceptual quality of pretrained VGG features, validated by human perceptual judgments [134]. In these cases, the frozen network weights cannot adapt to the data on hand. Furthermore, the frozen similarity function may not be appropriate when comparing data *across* two domains, depending on the pairing. By posing our constraint via mutual information, our method makes use of negative samples from the data, allowing the cross-domain similarity function to adapt to the particular input and output domains, and bypass using a pre-defined similarity function.

Contrastive representation learning. Traditional unsupervised learning has sought to learn a compressed code which can effectively reconstruct the input [135]. Data imputation – holding one subset of raw data to predict from another – has emerged as a more effective family of pretext tasks, including denoising [136], context prediction [83, 137], colorization [138, 139], cross-channel encoding [140], frame prediction [141, 142], and multi-sensory prediction [143, 144]. However, such methods suffer from the same issue as before — the need for a pre-specified, hand-designed loss function to measure predictive performance.

Recently, a family of methods based on *maximizing mutual information* has emerged to bypass the above issue [145–153]. These methods make use of noise contrastive estimation [154], learning an embedding where associated signals are brought together, in *contrast* to other samples in the dataset (note that similar ideas go back to classic work on metric learning with Siamese nets [155]). Associated signals can be an image with itself [146, 153, 156–158], an image with its downstream representation [148, 149], neighboring patches within an image [147, 151, 159], or multiple views of the input image [152], and most successfully, an image with a set of transformed versions of itself [145, 150]. The design choices of the InfoNCE loss, such as the number of negatives and how to sample them, hyperparameter settings, and data augmentations all play a critical role and need to be carefully studied. We are the first to use InfoNCE loss for the conditional image synthesis tasks. As such, we draw on these important insights, and find additional pertinent factors, unique to image synthesis.

3.2 Pixel Cycle-Consistency Loss for Unpaired Translation

[1]

Our goal 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 $y_j \in Y$[2]. We denote the data distribution as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. As illustrated in Figure 3.2 (a), our model includes two mappings $G : X \to Y$ and $F : Y \to X$. 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

[1]This work was first published as *Unpaired image-to-image translation using cycle-consistent adversarial networks* in ICCV, 2017 [41].

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

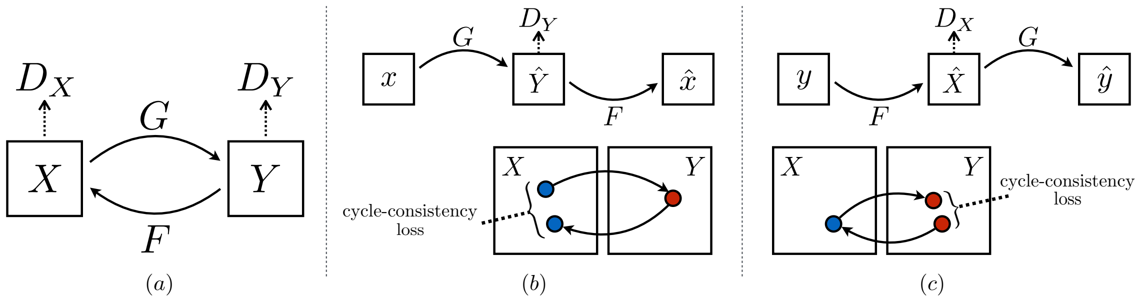


Figure 3.2: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, 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: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

discriminate between $\{y\}$ and $\{G(x)\}$. Our objective contains two types of terms: *adversarial losses* [12] 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.

3.2.1 Adversarial Loss

We apply adversarial losses [12] to both mapping functions. For the mapping function $G : X \rightarrow Y$ and its discriminator D_Y, we express the objective as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) &= \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &\quad + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))], \end{aligned} \tag{3.1}$$

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 $F : Y \rightarrow X$ and its discriminator D_X as well: i.e., $\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$.

3.2.2 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,

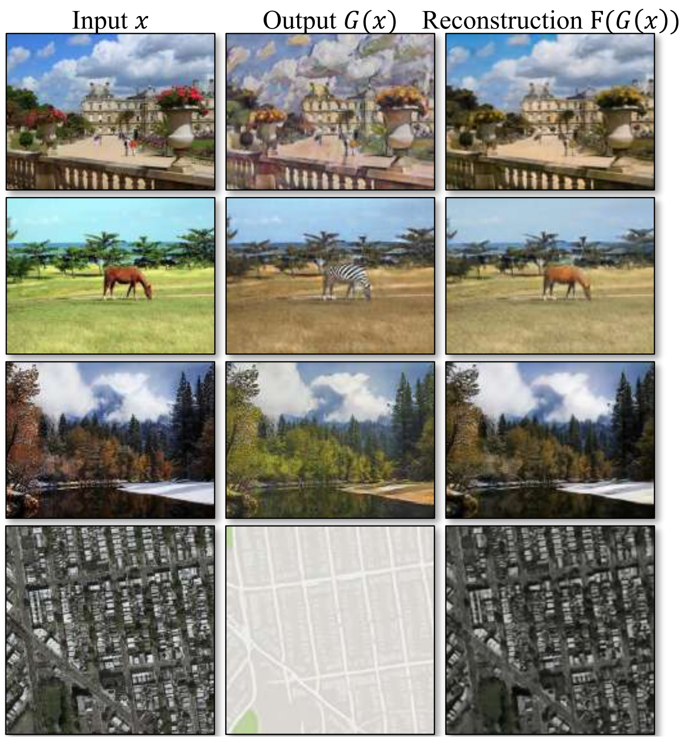


Figure 3.3: The input images x, output images $G(x)$ and the reconstructed images $F(G(x))$ from various experiments. From top to bottom: photo↔Cezanne, horses↔zebras, winter→summer Yosemite, aerial photos↔Google maps.

this requires G and F to be stochastic functions) [74]. 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 3.2 (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., $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. We call this *forward cycle consistency*. Similarly, as illustrated in Figure 3.2 (c), for each image y from domain Y, G and F should also satisfy *backward cycle consistency*: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. We incentivize this behavior using a *cycle consistency loss*:

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G,F) = {} & \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x))-x\|_1] \\ & + \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y))-y\|_1]. \end{aligned} \tag{3.2}$$

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 3.3: the reconstructed images $F(G(x))$ end up matching closely to the input images x.

3.2.3 Full Objective

Our full objective is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda\mathcal{L}_{\text{cyc}}(G, F),\end{aligned} \tag{3.3}$$

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

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y). \tag{3.4}$$

Notice that our model can be viewed as training two "autoencoders" [135]: 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 structures: 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" [160], 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 the domain Y.

In Chapter 3.2.5, 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.

3.2.4 Implementation

Network Architecture We adopt the architecture for our generative networks from Johnson et al. [116] who have shown impressive results for neural style transfer and super-resolution. This network contains three convolutions, several residual blocks [60], two fractionally-strided convolutions with stride $\frac{1}{2}$, and one convolution that maps features to RGB. We use 6 blocks for 128×128 images and 9 blocks for 256×256 and higher-resolution training images. Similar to Johnson et al. [116], we use instance normalization [161]. For the discriminator networks we use 70×70 PatchGANs [73, 162, 163], 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 work on arbitrarily-sized images in a fully convolutional fashion [73].

Training details We apply two techniques from recent works to stabilize our model training procedure. First, for $\mathcal{L}_{\text{GAN}}$ (Equation 3.1), we replace the negative log likelihood objective by a least-squares loss [61]. 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 [74], we follow Shrivastava et al.'s strategy [92] and update the discriminators using a history of generated images rather than the ones produced by the latest generators. We keep an image buffer that stores the 50 previously created images.

For all the experiments, we set $\lambda = 10$ in Equation 3.3. We use the Adam solver [164] 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.

3.2.5 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 PyTorch and Torch code, models, and full results can be found at our website.

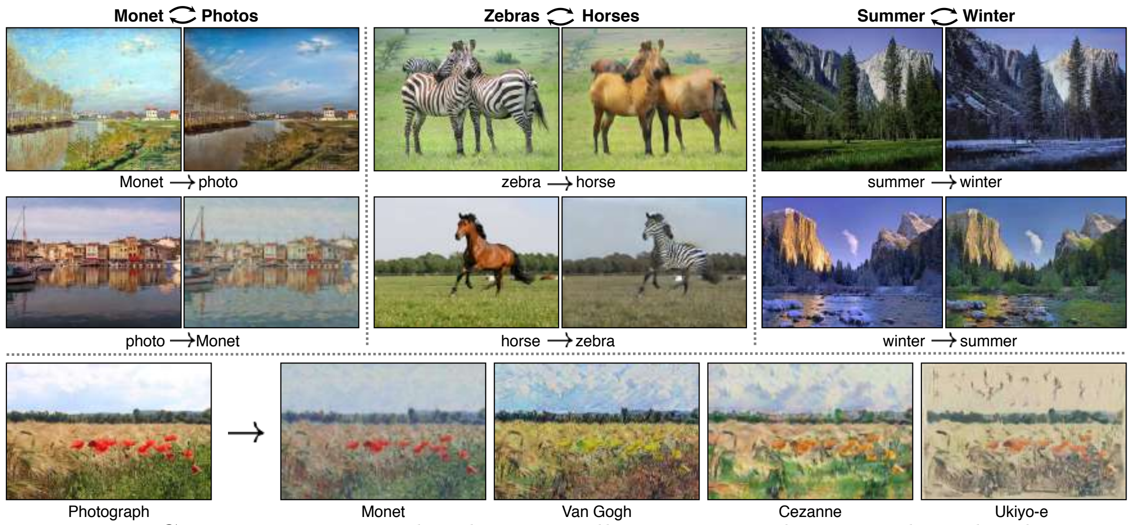


Figure 3.4: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite 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.

Evaluation

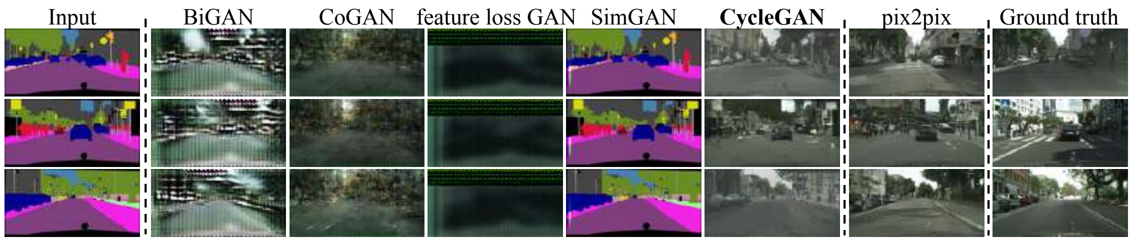


Figure 3.5: Different methods for mapping labels↔photos trained on Cityscapes images. From left to right: input, BiGAN/ALI [165, 166], CoGAN [90], feature loss + GAN, SimGAN [92], CycleGAN (ours), pix2pix [73] trained on paired data, and ground truth.

Using the same evaluation datasets and metrics as "pix2pix" [73], 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.

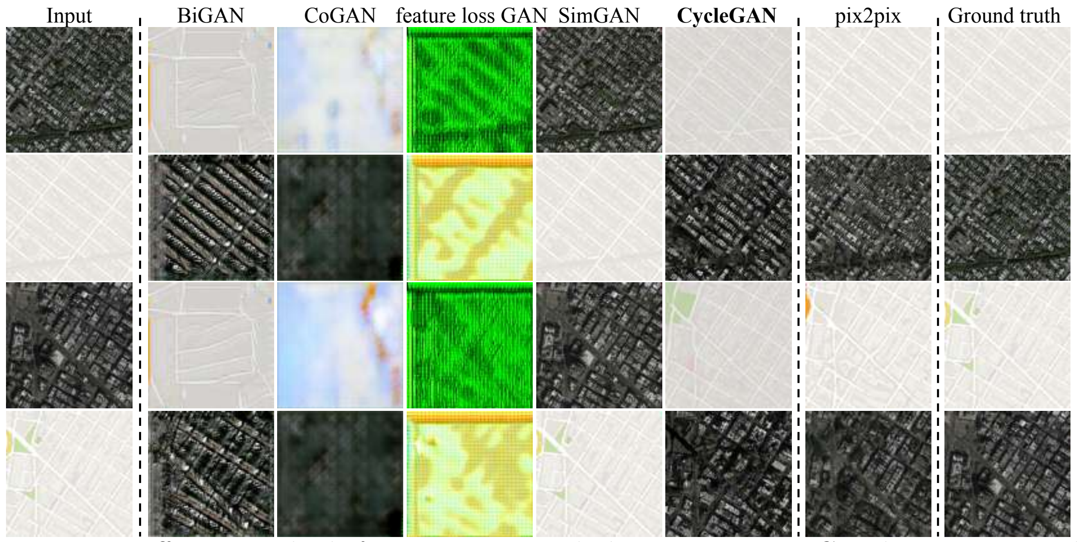


Figure 3.6: Different methods for mapping aerial photos↔maps on Google Maps. From left to right: input, BiGAN/ALI [165,166], CoGAN [90], feature loss + GAN, SimGAN [92], CycleGAN (ours), pix2pix [73] trained on paired data, and ground truth.

Evaluation 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. [73], 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. The numbers we report here are not directly comparable to those in [73] as our ground truth images were processed slightly differently [3] and the participant pool we tested may be differently distributed from those tested in [73] (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 [73].

FCN score Although perceptual studies may be the gold standard for assessing

[3]We train all the models on 256×256 images while in pix2pix [73], 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.

graphical realism, we also seek an automatic quantitative measure that does not require human experiments. For this, we adopt the "FCN score" from [73], 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 [87]). 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 "car on the road", then we have succeeded if the FCN applied to the generated photo detects "car on the road".

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

Baselines **CoGAN** [**90**] 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 representations. 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 [**92**] Like our method, Shrivastava et al. [92] uses an adversarial loss to train a translation from X to Y. The regularization term $\|x - G(x)\|_1$ i s used to penalize making large changes at pixel level.

Feature loss + GAN We also test a variant of SimGAN [92] where the L1 loss is computed over deep image features using a pretrained network (VGG-16 `relu4_2` [167]), rather than over RGB pixel values. Computing distances in deep feature space, like this, is also sometimes referred to as using a "perceptual loss" [116, 127].

BiGAN/ALI [**165,166**] Unconditional GANs [12] learn a generator $G : Z \to X$, that maps a random noise z to an image x. The BiGAN [166] and ALI [165] 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 [**73**] We also compare against pix2pix [73], which is trained on paired data, to see how close we can get to this "upper bound" without using any paired data.

For a fair comparison, we implement all the baselines using the same architecture and details as our method, except for CoGAN [90]. CoGAN builds on generators that produce images from a shared latent representation, which is incompatible with

Loss	**Map → Photo** % Turkers labeled *real*	**Photo → Map** % Turkers labeled *real*
CoGAN [90]	0.6% ± 0.5%	0.9% ± 0.5%
BiGAN/ALI [165, 166]	2.1% ± 1.0%	1.9% ± 0.9%
SimGAN [92]	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 3.1: AMT "real vs fake" test on maps↔aerial photos at 256 × 256 resolution.

Loss	**Per-pixel acc.**	**Per-class acc.**	**Class IOU**
CoGAN [90]	0.40	0.10	0.06
BiGAN/ALI [165, 166]	0.19	0.06	0.02
SimGAN [92]	0.20	0.10	0.04
Feature loss + GAN	0.06	0.04	0.01
CycleGAN (ours)	**0.52**	**0.17**	**0.11**
pix2pix [73]	0.71	0.25	0.18

Table 3.2: FCN-scores for different methods, evaluated on Cityscapes labels→photo.

our image-to-image network. We use the public implementation of CoGAN instead.

Comparison against baselines

As can be seen in Figure 3.5 and Figure 3.6, we were unable to achieve compelling results with any of the baselines. Our method, on the other hand, can produce translations that are often of similar quality to the fully supervised pix2pix.

Table 3.1 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 the maps→aerial photos direction and the aerial photos→maps direction at 256×256 resolution[4]. All the baselines almost never fooled participants.

Table 3.2 assesses the performance of the labels→photo task on the Cityscapes and Table 3.3 evaluates the opposite mapping (photos→labels). In both cases, our method again outperforms the baselines.

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

Loss	**Per-pixel acc.**	**Per-class acc.**	**Class IOU**
CoGAN [90]	0.45	0.11	0.08
BiGAN/ALI [165, 166]	0.41	0.13	0.07
SimGAN [92]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**
pix2pix [73]	0.85	0.40	0.32

Table 3.3: Classification performance of photo→labels for different methods on cityscapes.

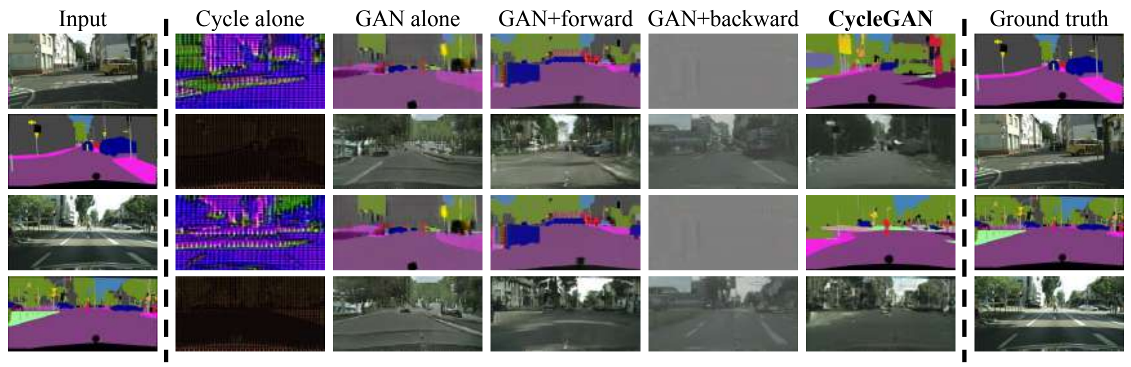


Figure 3.7: Different variants of our method for mapping labels↔photos trained on cityscapes. From left to right: input, cycle-consistency loss alone, adversarial loss alone, GAN + forward cycle-consistency loss ($F(G(x)) \approx x$), GAN + backward cycle-consistency loss ($G(F(y)) \approx y$), CycleGAN (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 3.4: Ablation study: FCN-scores for different variants of our method, evaluated on Cityscapes labels→photo.

Analysis of the loss function

In Table 3.4 and Table 3.5, we compare against ablations of our full loss. Removing the GAN loss substantially degrades results, as 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 3.2) and find that it often incurs training instability and causes mode collapse, especially for the direction of the mapping that was removed. Figure 3.7 shows several qualitative examples.

Image reconstruction quality

In Figure 3.3, we show a few random samples of the reconstructed images $F(G(x))$. We observed that the reconstructed images were often close to the original inputs x, at both training and testing time, even in cases where one domain represents

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

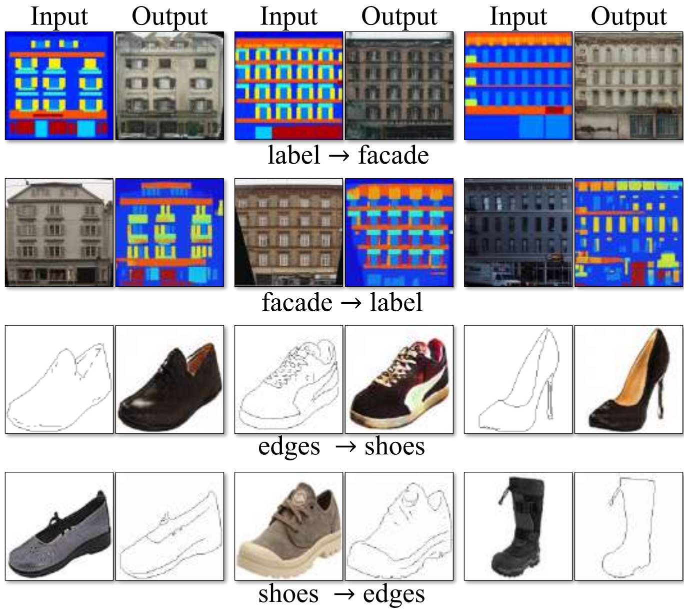


Figure 3.8: Example results of CycleGAN on paired datasets used in "pix2pix" [73] such as architectural labels↔photos and edges↔shoes.

significantly more diverse information, such as map↔aerial photos.

Additional results on paired datasets

Figure 3.8 shows some example results on other paired datasets used in "pix2pix" [73], such as architectural labels↔photos from the CMP Facade Database [168], and edges↔shoes from the UT Zappos50K dataset [169]. 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.

3.2.6 Applications

We demonstrate our method on several applications where paired training data does not exist. We observe that translations on training data are often more appealing

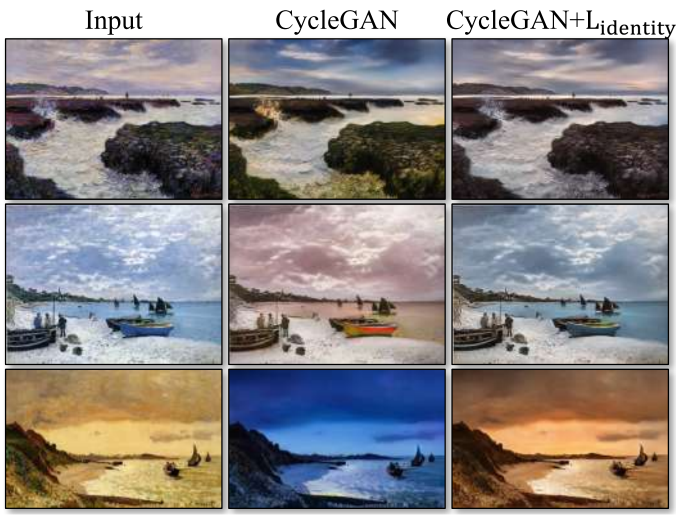


Figure 3.9: The effect of the *identity mapping loss* on Monet's painting→ photos. From left to right: input paintings, CycleGAN without identity mapping loss, CycleGAN with identity mapping loss. The identity mapping loss helps preserve the color of the input paintings.

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 3.10 and Figure 3.11) We train the model on landscape photographs downloaded from Flickr and WikiArt. Unlike recent work on "neural style transfer" [115], 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 3.13) The model is trained to translate one object class from ImageNet [126] to another (each class contains around 1000 training images). Turmukhambetov et al. [170] propose 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 3.13) The model is trained on 854 winter photos and 1273 summer photos of Yosemite downloaded from Flickr.

Photo generation from paintings (Figure 3.12) 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. [93] 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 3.9.

In Figure 3.12, we show additional results translating Monet's paintings to photographs. This figure and Figure 3.9 show results on paintings that were included in the *training set*, whereas for all other experiments in the paper, 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 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 3.14) 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 flower photos taken by smartphones, which usually have deep DoF due to a small aperture. The target contains photos captured by DSLRs with a larger aperture. Our model successfully generates photos with shallower depth of field from the photos taken by smartphones.

Comparison with Gatys et al. [115] In Figure 3.15, we compare our results with neural style transfer [115] on photo stylization. For each row, we first use two representative artworks as the style images for [115]. Our method, on the other hand, can 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" with Gatys et al [115].

Figure 3.16 demonstrates similar comparisons for other translation tasks. We observe that Gatys et al. [115] requires finding target style images that closely match the desired output, but still often fails to produce photorealistic results, while our method succeeds to generate natural-looking results, similar to the target domain.

3.2.7 Limitations and Discussion

Although our method can achieve compelling results in many cases, the results are far from uniformly positive. Figure 3.17 shows several typical failure cases. On translation tasks that involve color and texture changes, as 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$\rightarrow$cat transfiguration, the learned translation degenerates into making minimal changes to the input (Figure 3.17). This failure might be caused by our generator architectures 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, our method has got confused in the horse $\rightarrow$ zebra example (Figure 3.17, right), 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$\rightarrow$labels task. Resolving 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 paper pushes the boundaries of what is possible in this “unsupervised" setting.

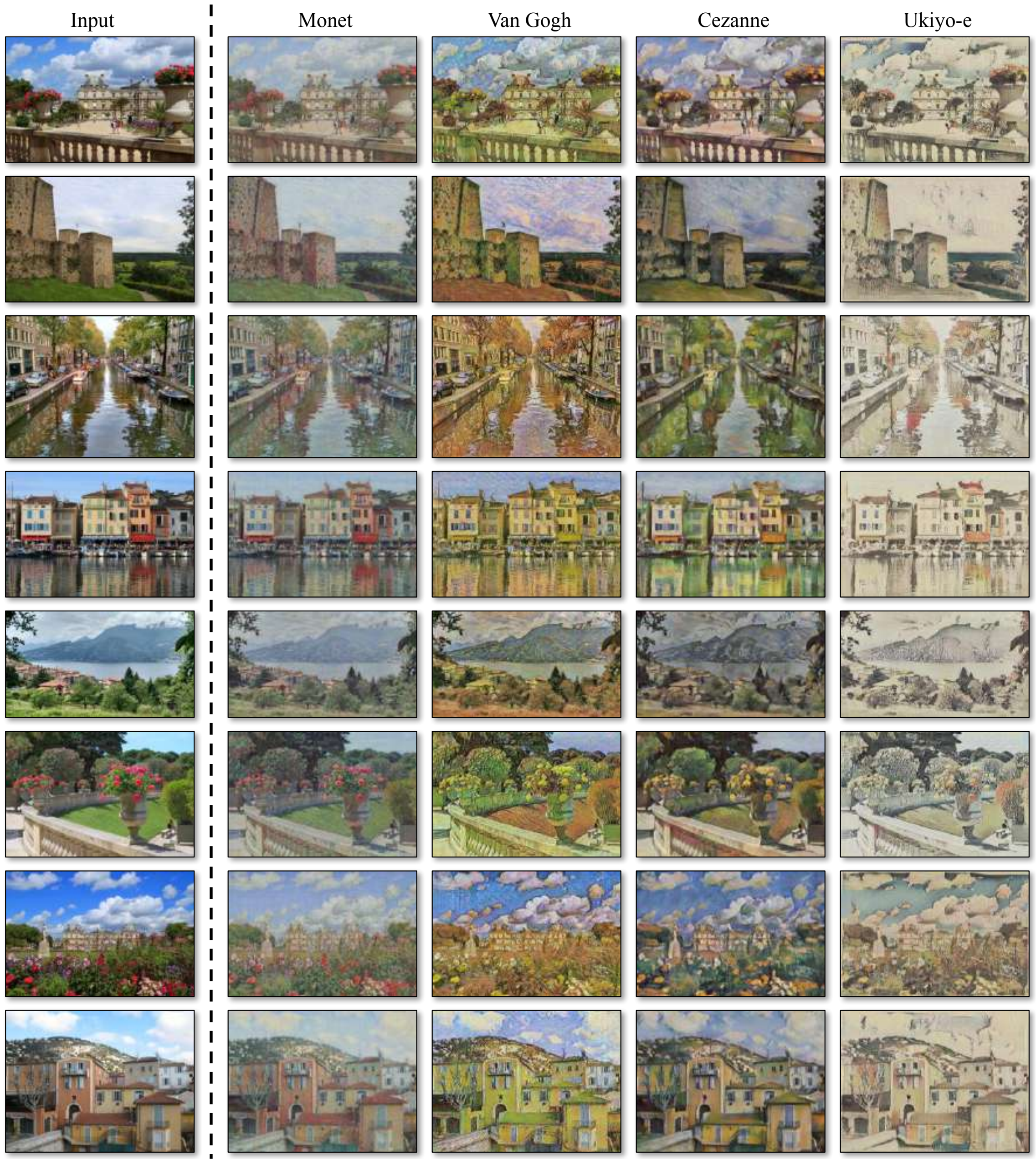


Figure 3.10: Collection style transfer I: we transfer input images into the artistic styles of Monet, Van Gogh, Cezanne, and Ukiyo-e. Please see our website for additional examples.

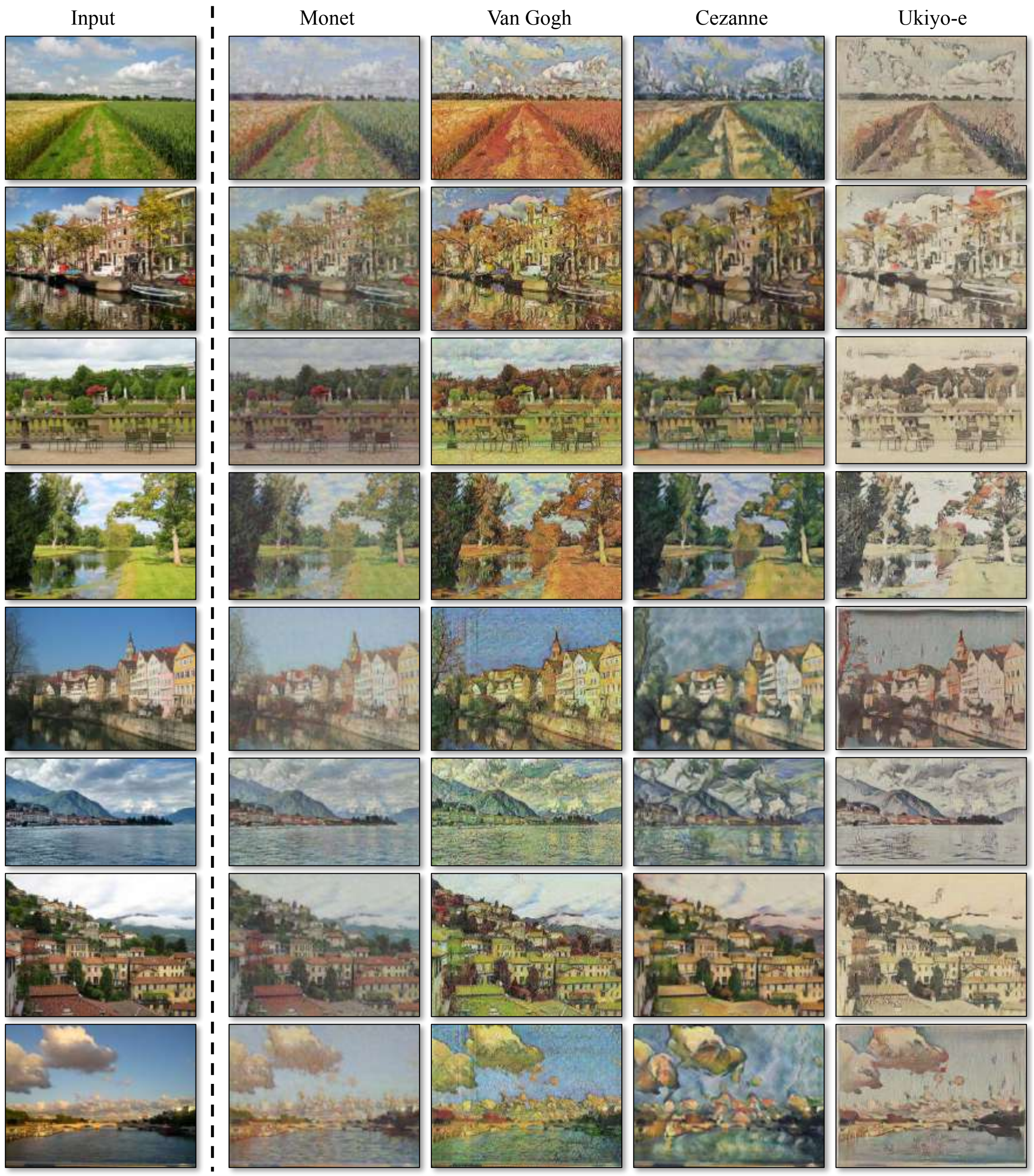


Figure 3.11: Collection style transfer II: we transfer input images into the artistic styles of Monet, Van Gogh, Cezanne, Ukiyo-e. Please see our website for additional examples.

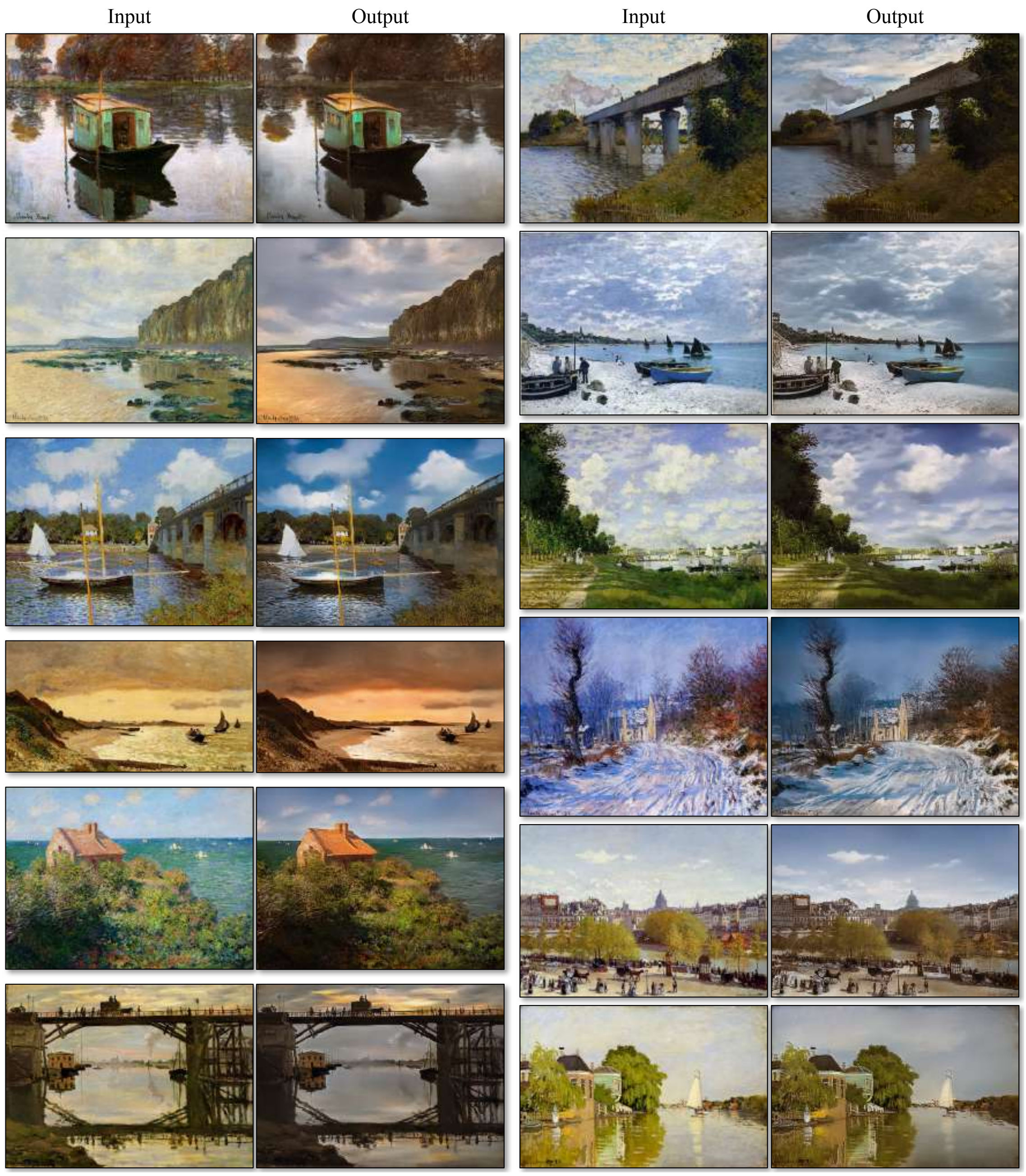


Figure 3.12: Relatively successful results on mapping Monet's paintings to a photographic style. Please see our website for additional examples.

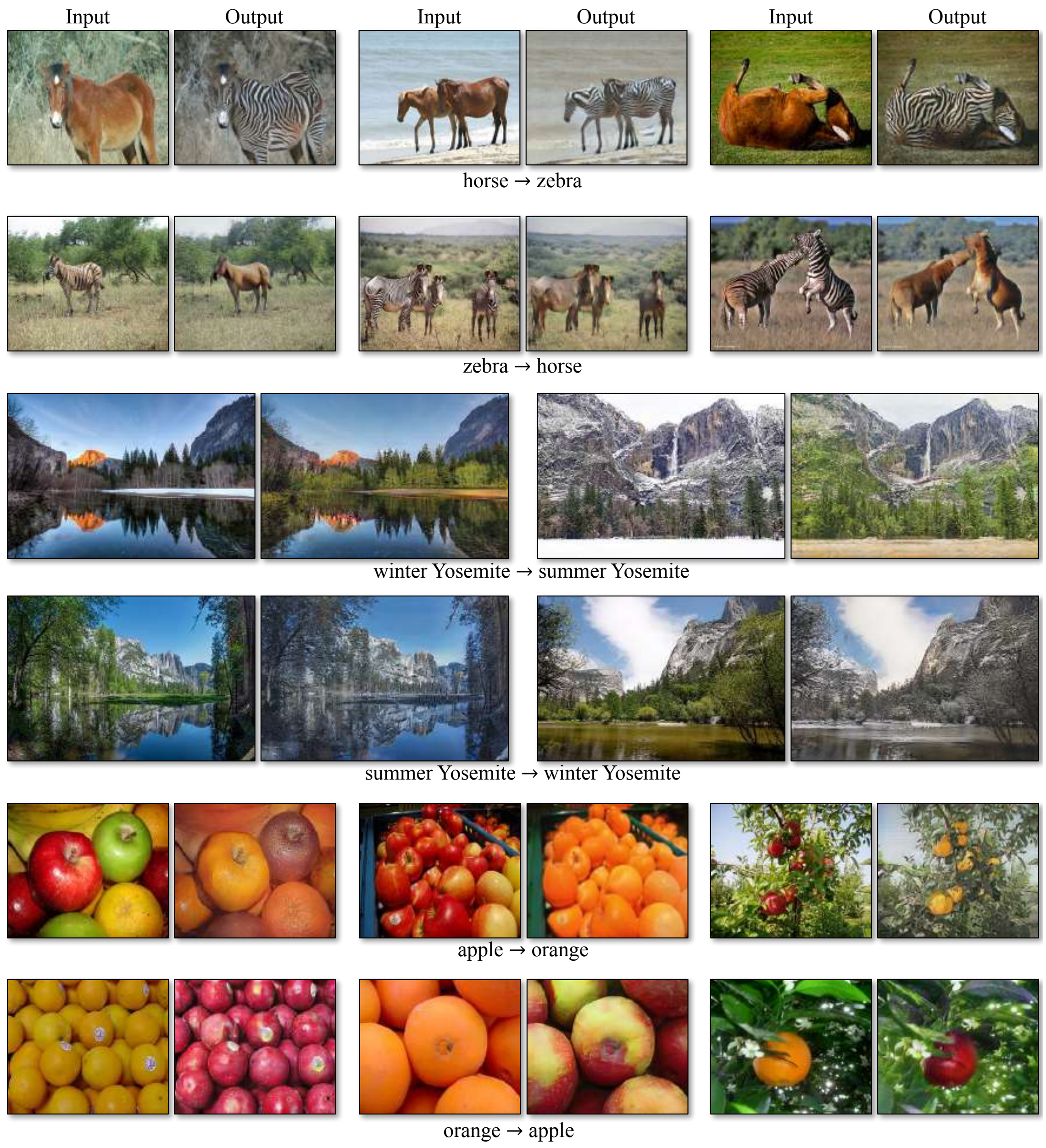


Figure 3.13: Our method applied to several translation problems. These images are selected as relatively successful results – please see our website for more comprehensive and random results. In the top two rows, we show results on object transfiguration between horses and zebras, trained on 939 images from the *wild horse* class and 1177 images from the *zebra* class in Imagenet [126]. Also check out the horse→zebra demo video. 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.

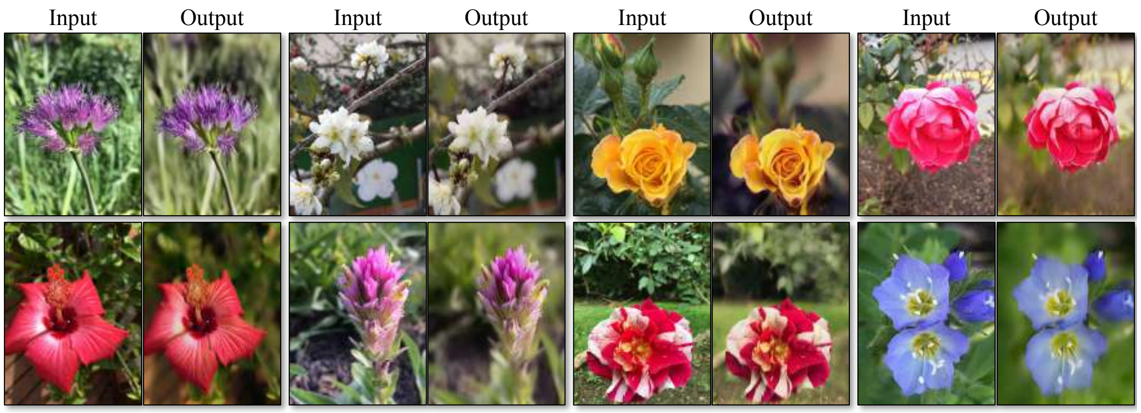


Figure 3.14: Photo enhancement: mapping from a set of smartphone snaps to professional DSLR photographs, the system often learns to produce shallow focus. Here we show some of the most successful results in our test set – average performance is considerably worse. Please see our website for more comprehensive and random examples.

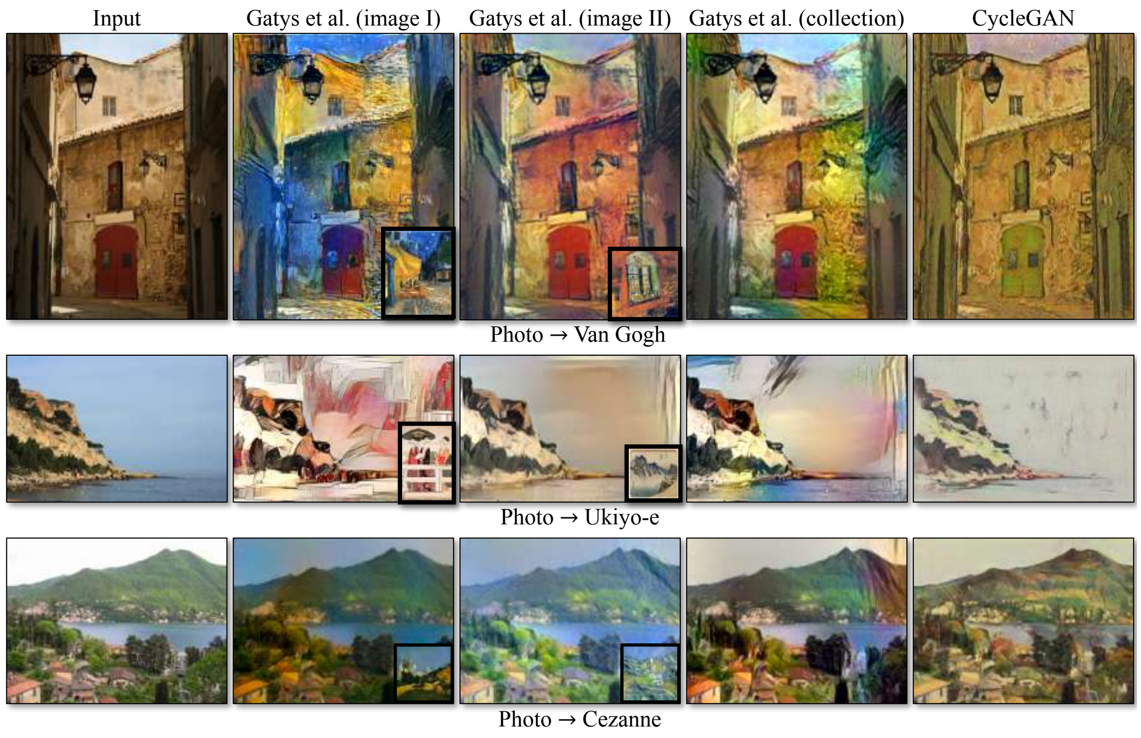


Figure 3.15: We compare our method with neural style transfer [115] on photo stylization. Left to right: input image, results from Gatys et al. [115] using two different representative artworks as style images, results from Gatys et al. [115] using the entire collection of the artist, and CycleGAN (ours).

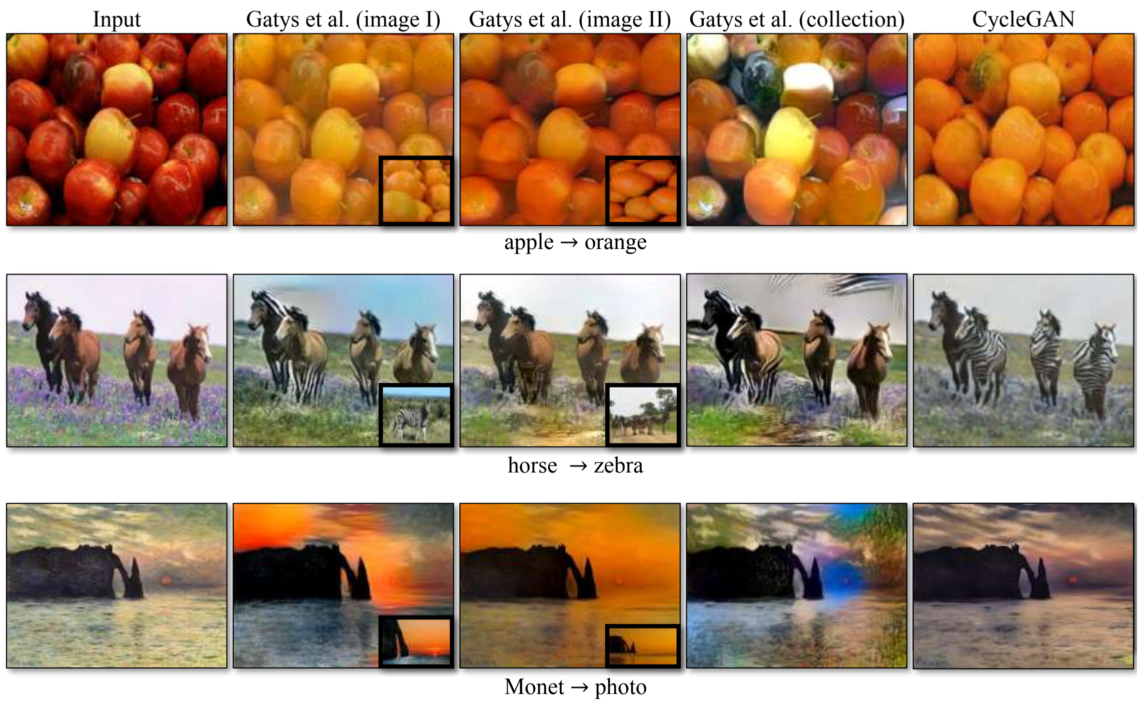


Figure 3.16: We compare our method with neural style transfer [115] on various applications. From top to bottom: apple→orange, horse→zebra, and Monet→photo. Left to right: input image, results from Gatys et al. [115] using two different images as style images, results from Gatys et al. [115] using all the images from the target domain, and CycleGAN (ours).

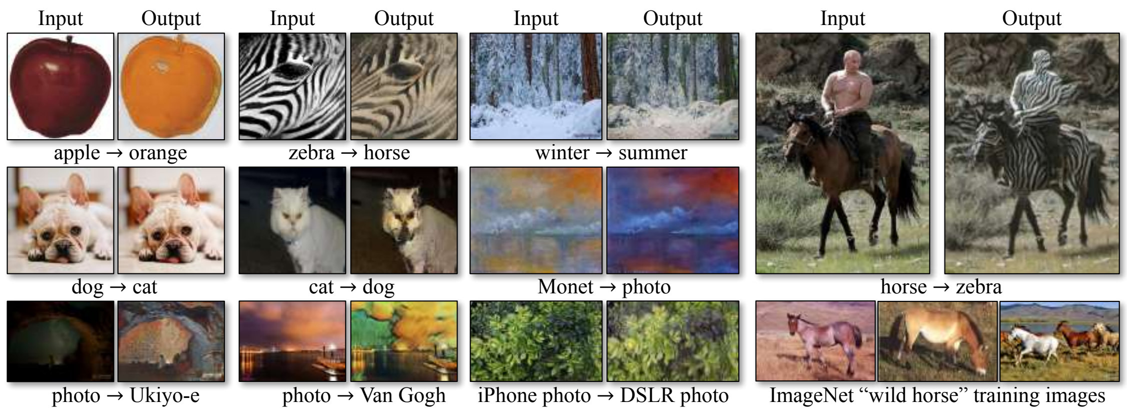


Figure 3.17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our website for more comprehensive results.

3.3 Contrastive Loss for Unpaired Translation

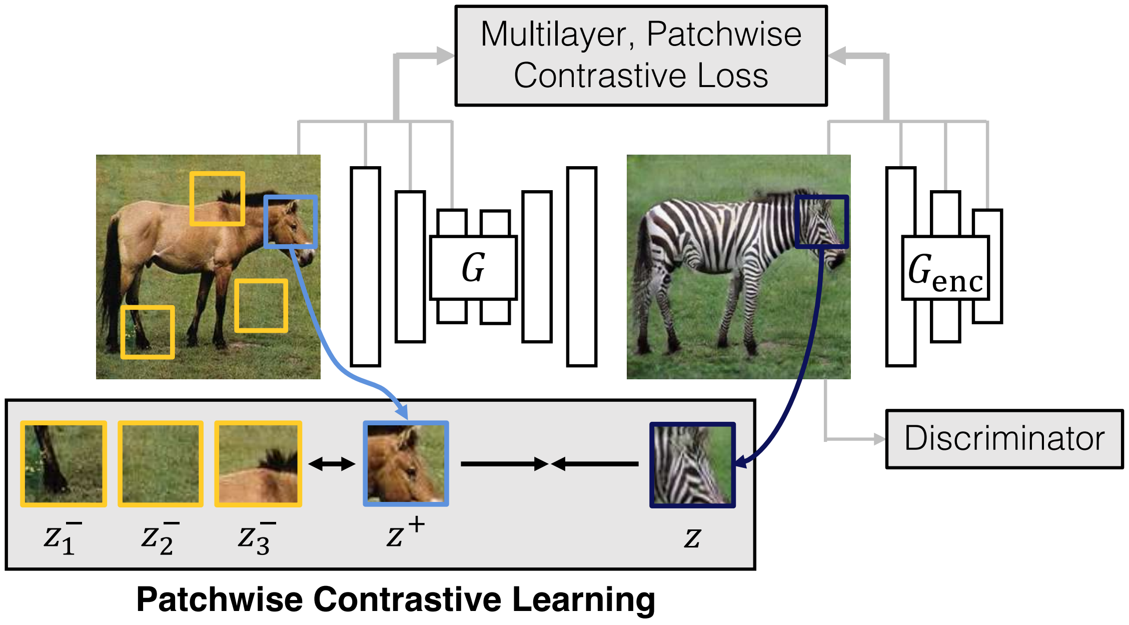

Multilayer, Patchwise Contrastive Loss
G
G_{enc}
Discriminator
z_1^-
z_2^-
z_3^-
z^+
z
Patchwise Contrastive Learning

Figure 3.18: **Patchwise Contrastive Learning for one-sided translation.** A generated **output patch** should appear closer to its **corresponding input patch**, in comparison to other **random patches**. We use a multilayer, patchwise contrastive loss, which maximizes *mutual information* between corresponding input and output patches. This enables one-sided translation in the unpaired setting.

Consider the image-to-image translation problem in Figure 3.18. We wish for the output to take on the *appearance* of the target domain (a zebra), while retaining the structure, or *content*, of the specific input horse. This is, fundamentally, a disentanglement problem: separating the content, which needs to be preserved across domains, from appearance, which must change. Typically, target appearance is enforced using an adversarial loss [12, 20], while content is preserved using cycle-consistency [41, 104, 105]. However, cycle-consistency assumes that the relationship between the two domains is a bijection, which is often too restrictive. In this paper, we propose an alternative, rather straightforward way of maintaining correspondence in content but not appearance – by maximizing the mutual information between corresponding input and output patches.

In a successful result, given a specific patch on the output, for example, the generated zebra forehead highlighted in blue, one should have a good idea that it came from the horse forehead, and not the other parts of the horse or the background vegetation. We achieve this by using a type of contrastive loss function, InfoNCE loss [151], which aims to learn an embedding or an encoder that *associates* corresponding patches to each other, while *disassociating* them from others. To do so, the encoder learns to pay attention to the commonalities between the two domains, such as object parts and shapes, while being invariant to the differences, such as the textures of the animals. The two networks, the generator and encoder, conspire together to generate an image such that patches can be easily traceable to the input.

Contrastive learning has been an effective tool in unsupervised visual representation learning [145, 146, 151, 153]. In this work, we demonstrate its effectiveness in a conditional image synthesis setting and systematically study several key factors to make it successful. We find it pertinent to use it on a multilayer, patchwise fashion. In addition, we find that drawing negatives *internally* from within the input image, rather than externally from other images in the dataset, forces the patches to better preserve the content of the input. Our method requires neither memory bank [146, 153] nor specialized architectures [147, 171].

Extensive experiments show that our faster, lighter model outperforms both prior one-sided translation methods [123, 124] and state-of-the-art models that rely on several auxiliary networks and multiple loss functions. Furthermore, since our contrastive representation is formulated within the same image, our method can even be trained on single images. Our code and models are available at GitHub.[5]

3.3.1 Methods

We wish to translate images from input domain $\mathcal{X} \subset \mathbb{R}^{H\times W\times C}$ to appear like an image from the output domain $\mathcal{Y} \subset \mathbb{R}^{H\times W\times 3}$. We are given a dataset of unpaired instances $X = \{\boldsymbol{x} \in \mathcal{X}\}, Y = \{\boldsymbol{y} \in \mathcal{Y}\}$. Our method can operate even when X and Y only contain a single image each.

Our method only requires learning the mapping in one direction and avoids using inverse auxiliary generators and discriminators. This can largely simplify the training procedure and reduce training time. We break up our generator function G into two components, an encoder G_{enc} followed by a decoder G_{dec}, which are applied sequentially to produce output image $\hat{\boldsymbol{y}} = G(\boldsymbol{z}) = G_{\text{dec}}(G_{\text{enc}}(\boldsymbol{x}))$.

Adversarial loss. We use an adversarial loss [12], to encourage the output to be visually similar to images from the target domain, as follows:

$$\mathcal{L}_{\text{GAN}}(G, D, X, Y) = \mathbb{E}_{\boldsymbol{y}\sim Y} \log D(\boldsymbol{y}) + \mathbb{E}_{\boldsymbol{x}\sim X} \log(1 - D(G(\boldsymbol{x}))). \tag{3.5}$$

[5]This work was first published as *Contrastive Learning for Unpaired Image-to-Image Translation* in ECCV, 2020 [172].

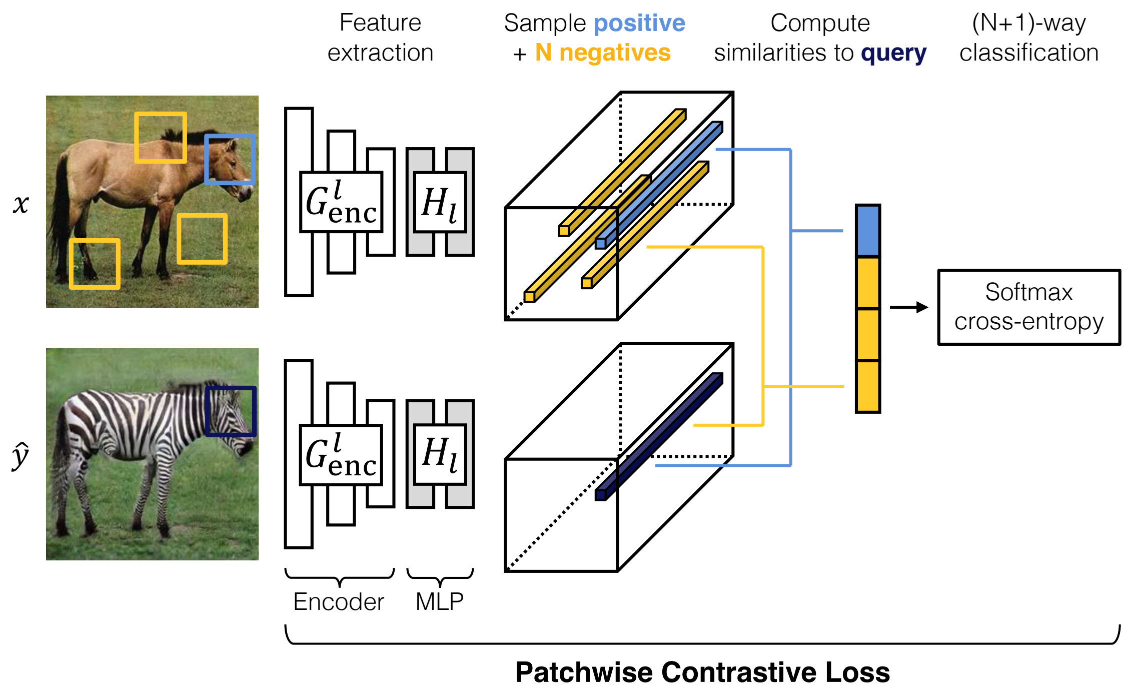


Figure 3.19: **Patchwise Contrastive Loss.** Both images, $\boldsymbol{x}$ and $\hat{\boldsymbol{y}}$, are encoded into feature tensor. We sample a **query** patch from the output $\hat{\boldsymbol{y}}$ and compare it to the input patch at the same location. We set up an (N+1)-way classification problem, where N negative patches are sampled from the same input image at different locations. We reuse the encoder part G_{enc} of our generator and add a two-layer MLP network. This network learns to project both the input and output patch to a shared embedding space.

Mutual information maximization. We use a noise contrastive estimation framework [151] to maximize mutual information between input and output. The idea of contrastive learning is to associate two signals, a "query" and its "positive" example, in contrast to other points within the dataset, referred to as "negatives". The query, positive, and N negatives are mapped to K-dimensional vectors $\boldsymbol{v}, \boldsymbol{v}^+ \in \mathbb{R}^K$ and $\boldsymbol{v}^- \in \mathbb{R}^{N \times K}$, respectively. $\boldsymbol{v}_n^- \in \mathbb{R}^K$ denotes the n-th negative. We normalize vectors onto a unit sphere to prevent the space from collapsing or expanding. An $(N+1)$–way classification problem is set up, where the distances between the query and other examples are scaled by a temperature $\tau = 0.07$ and passed as logits [146, 153]. The cross-entropy loss is calculated, representing the probability of the positive example being selected over negatives.

$$\ell(\boldsymbol{v}, \boldsymbol{v}^+, \boldsymbol{v}^-) = -\log\left[\frac{\exp(\boldsymbol{v}\cdot\boldsymbol{v}^+/\tau)}{\exp(\boldsymbol{v}\cdot\boldsymbol{v}^+/\tau) + \sum_{n=1}^{N}\exp(\boldsymbol{v}\cdot\boldsymbol{v}_n^-/\tau)}\right]. \tag{3.6}$$

Our goal is to associate the input and output data. In our context, query refers to an output. positive and negatives are corresponding and noncorresponding input. Below, we explore several important design choices, including how to map the images into vectors and how to sample the negatives.

Multilayer, patchwise contrastive learning. In the unsupervised learning setting, contrastive learning has been used both on an image and patch level [147,171]. For our application, we note that not only should the whole images share content, but also corresponding patches between the input and output images. For example, given a patch showing the legs of an output zebra, one should be able to more strongly associate it to the corresponding legs of the input horse, more so than the other patches of the horse image. Even at the pixel level, the colors of a zebra body (black and white) can be more strongly associated to the color of a horse body than to the background shades of grass. Thus, we employ a *multilayer, patch-based* learning objective.

Since the encoder G_{enc} is computed to produce the image translation, its feature stack is readily available, and we take advantage. Each layer and spatial location within this feature stack represents a patch of the input image, with deeper layers corresponding to bigger patches. We select L layers of interest and pass the feature maps through a small two-layer MLP network H_l, as used in SimCLR [145], producing a stack of features $\{\boldsymbol{z}_l\}_L = \{H_l(G^l_{\text{enc}}(\boldsymbol{x}))\}_L$, where G^l_{enc} represents the output of the l-th chosen layer. We index into layers $l \in \{1, 2, ..., L\}$ and denote $s \in \{1, ..., S_l\}$, where S_l is the number of spatial locations in each layer. We refer to the corresponding feature as $\boldsymbol{z}_l^s \in \mathbb{R}^{C_l}$ and the other features as $\boldsymbol{z}_l^{S\setminus s} \in \mathbb{R}^{(S_l-1)\times C_l}$, where C_l is the number of channels at each layer. Similarly, we encode the output image $\hat{\boldsymbol{y}}$ into $\{\hat{\boldsymbol{z}}_l\}_L = \{H_l(G^l_{\text{enc}}(G(\boldsymbol{x})))\}_L$.

We aim to match corresponding input-output patches at a specific location. We can leverage the other patches *within* the input as negatives. For example, a zebra leg should be more closely associated with an input horse leg than the other patches of the same input, such as other horse parts or the background sky and vegetation. We name it as the *PatchNCE* loss, as illustrated in Figure 3.19.

$$\mathcal{L}_{\text{PatchNCE}}(G, H, X) = \mathbb{E}_{\boldsymbol{x}\sim X}\sum_{l=1}^{L}\sum_{s=1}^{S_l}\ell(\hat{\boldsymbol{z}}_l^s, \boldsymbol{z}_l^s, \boldsymbol{z}_l^{S\setminus s}). \tag{3.7}$$

Alternatively, we can also leverage image patches from the rest of the dataset. We encode a random negative image from the dataset $\tilde{\boldsymbol{x}}$ into $\{\tilde{\boldsymbol{z}}_l\}_L$, and use the following *external* NCE loss. In this variant, we maintain a large, consistent dictionary of negatives using an auxiliary moving-averaged encoder, following MoCo [146]. MoCo enables negatives to be sampled from a longer history, and performs more effective than end-to-end updates [147, 151] and memory bank [153].

$$\mathcal{L}_{\text{external}}(G, H, X) = \mathbb{E}_{\boldsymbol{x} \sim X, \tilde{\boldsymbol{z}} \sim Z^-} \sum_{l=1}^{L} \sum_{s=1}^{S_l} \ell(\hat{\boldsymbol{z}}_l^s, \boldsymbol{z}_l^s, \tilde{\boldsymbol{z}}_l), \tag{3.8}$$

where dataset negatives $\tilde{\boldsymbol{z}}_l$ are sampled from an external dictionary Z^- from the source domain, whose data are computed using a moving-averaged encoder $\hat{H}_l$ and moving-averaged MLP $\hat{H}$. We refer our readers to the original work for more details [146].

In Chapter 3.3.2, we show that our encoder G_{enc} learns to capture domain-invariant concepts, such as animal body, grass, and sky for horse $\rightarrow$ zebra, while our decoder G_{dec} learns to synthesize domain-specific features such as zebra stripes. Interestingly, through systematic evaluations, we find that using internal patches only outperforms using external patches. We hypothesize that by using internal statistics, our encoder does not need to model large intra-class variation such as white horse vs. brown horse, which is not necessary for generating output zebras. Single image internal statistics has been proven effective in many vision tasks such as segmentation [173], super-resolution, and denoising [174, 175].

Final objective. Our final objective is as follows. The generated image should be realistic, while patches in the input and output images should share correspondence. Figure 3.18 illustrates our minimax learning objective. Additionally, we may utilize PatchNCE loss $\mathcal{L}_{\text{PatchNCE}}(G, H, Y)$ on images from domain $\mathcal{Y}$ to prevent the generator from making unnecessary changes. This loss is essentially a learnable, domain-specific version of the identity loss, commonly used by previous unpaired translation methods [41, 121].

$$\mathcal{L}_{\text{GAN}}(G, D, X, Y) + \lambda_X \mathcal{L}_{\text{PatchNCE}}(G, H, X) + \lambda_Y \mathcal{L}_{\text{PatchNCE}}(G, H, Y). \tag{3.9}$$

We choose $\lambda_X = 1$ when we jointly train with the identity loss $\lambda_Y = 1$, and choose a larger value $\lambda_X = 10$ without the identity loss ($\lambda_Y = 0$) to compensate for the absence of the regularizer. We find that the former configuration, named *Contrastive Unpaired Translation (CUT)* hereafter, achieves superior performance to existing methods, whereas the latter, named *FastCUT*, can be thought as a faster and lighter version of CycleGAN. Our model is relatively simple compared to recent methods that often use 5-10 losses and hyper-parameters.

Discussion. Li et al. [176] has shown that cycle-consistency loss is the upper bound of conditional entropy $\mathrm{H}(X|Y)$ (and $\mathrm{H}(Y|X)$). Therefore, minimizing cycle-consistency loss encourages the output $\hat{\boldsymbol{y}}$ to be more dependent on input $\boldsymbol{x}$. This is related to our objective of maximizing the mutual information $\mathrm{I}(X, Y)$, as $\mathrm{I}(X, Y) = \mathrm{H}(X) - \mathrm{H}(X|Y)$. As entropy $\mathrm{H}(X)$ is a constant and independent of the generator G, maximizing mutual information is equivalent to minimizing the conditional entropy. Notably, using contrastive learning, we can achieve a similar goal without introducing inverse mapping networks and additional discriminators. In the unconditional modeling scenario, InfoGAN [68] shows that simple losses (e.g., L2 or cross-entropy) can serve as a lower bound for maximizing mutual information between an image and a low-dimensional code. In our setting, we maximize the mutual information between two high-dimensional image spaces, where simple losses are no longer effective. Liang et al. [114] proposes an adversarial loss based on Siamese networks that encourages the output to be closer to the target domain than to its source domain. The above method still builds on cycle-consistency and two-way translations. Different from the above work, we use contrastive learning to enforce content consistency, rather than to improve the adversarial loss itself. To measure the similarity between two distributions, the Contextual Loss [131] used softmax over cosine disntances of features extracted from pre-trained networks. In contrast, we learn the encoder with the NCE loss to associate the input and output patches at the same location.

3.3.2 Experiments

We test across several datasets. We first show that our method improves upon baselines in unpaired image translation. We then show that our method can extend to *single-image* training. Full results are available at our website.

Training details. We follow the setting of CycleGAN [41], except that the ℓ_1 cycle-consistency loss is replaced with our contrastive loss. In detail, we used LSGAN [61] and Resnet-based generator [116] with PatchGAN [20]. We define our encoder as the first half of the generator, and accordingly extract our multilayer features from five evenly distributed points of the encoder. For single image translation, we use a StyelGAN2-based generator [177]. To embed the encoder's features, we apply a two-layer MLP with 256 units at each layer. We normalize the vector by its L2 norm.

To show the effect of the proposed patch-based contrastive loss, we intentionally match the architecture and hyperparameter settings of CycleGAN, except the loss function. This includes the ResNet-based generator [116] with 9 residual blocks, PatchGAN discriminator [20], Least Square GAN loss [61], batch size of 1, and Adam optimizer [67] with learning rate 0.002.

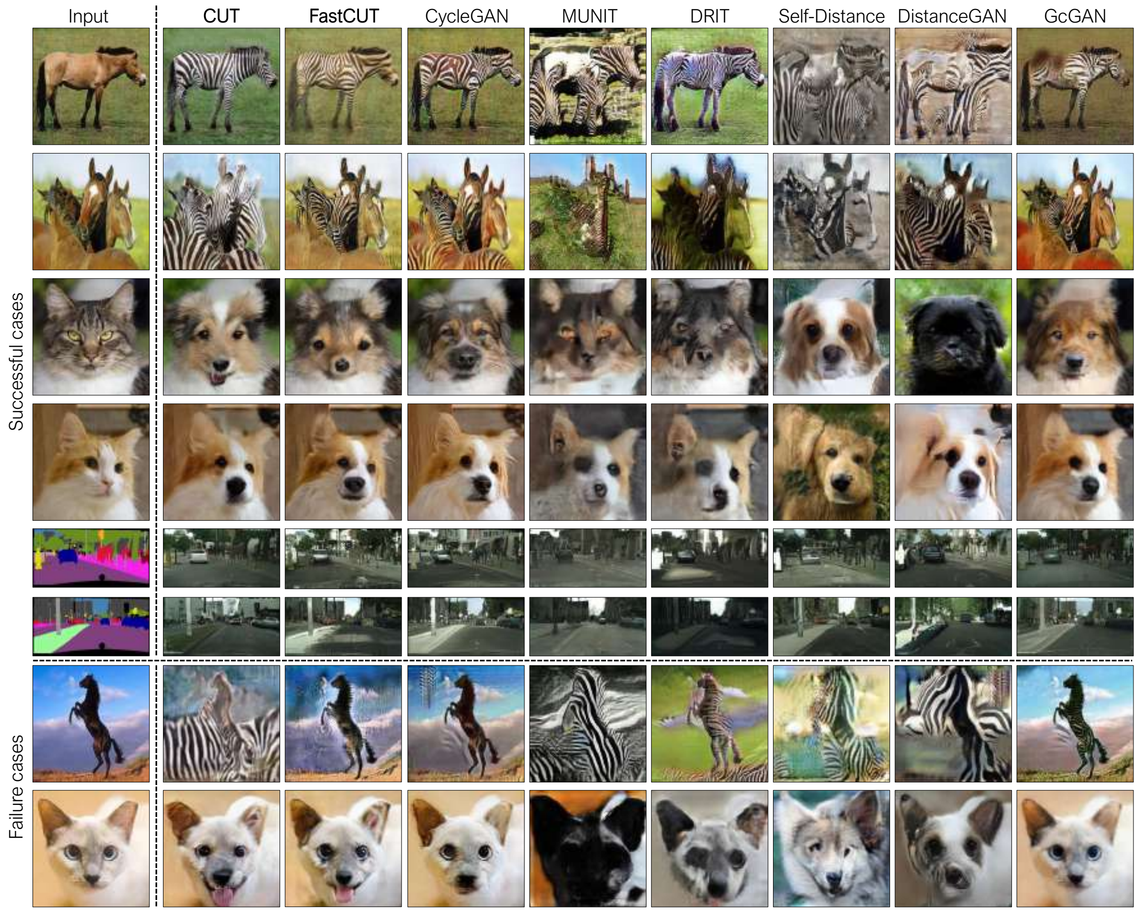


Figure 3.20: **Results**. We compare our methods (CUT and FastCUT) with existing methods on the horse→zebra, cat→dog, and Cityscapes datasets. CycleGAN [41], MUNIT [42], and DRIT [108], are two-sided methods, while SelfDistance, DistanceGAN [123], and GcGAN [124] are one-sided. We show successful cases above the dotted lines. Our full version CUT is able to add the zebra texture to the horse bodies. Our fast variant FastCUT can also generate competitive results at the least computational cost of training. The final rows show failure cases. In the first, we are unable to identify the unfamiliar pose of the horse and instead add texture to the background. In the second, the method hallucinates a tongue.

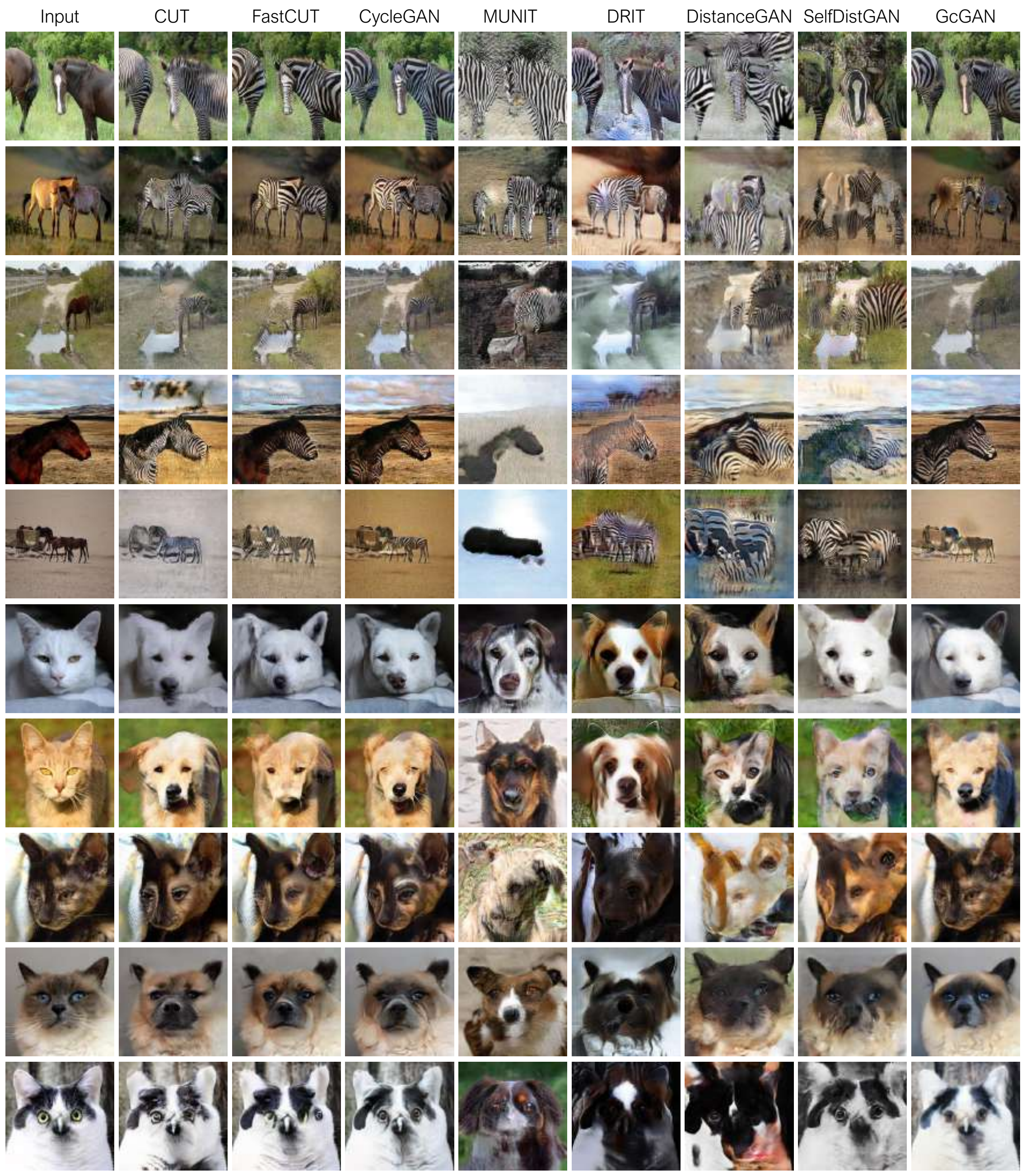


Figure 3.21: **Randomly selected Horse→Zebra and Cat→Dog results.**

Our full model CUT is trained up to 400 epochs, while the fast variant FastCUT is trained up to 200 epochs, following CycleGAN. Moreover, inspired by GcGAN [124], FastCUT is trained with flip-equivariance augmentation, where the input image to

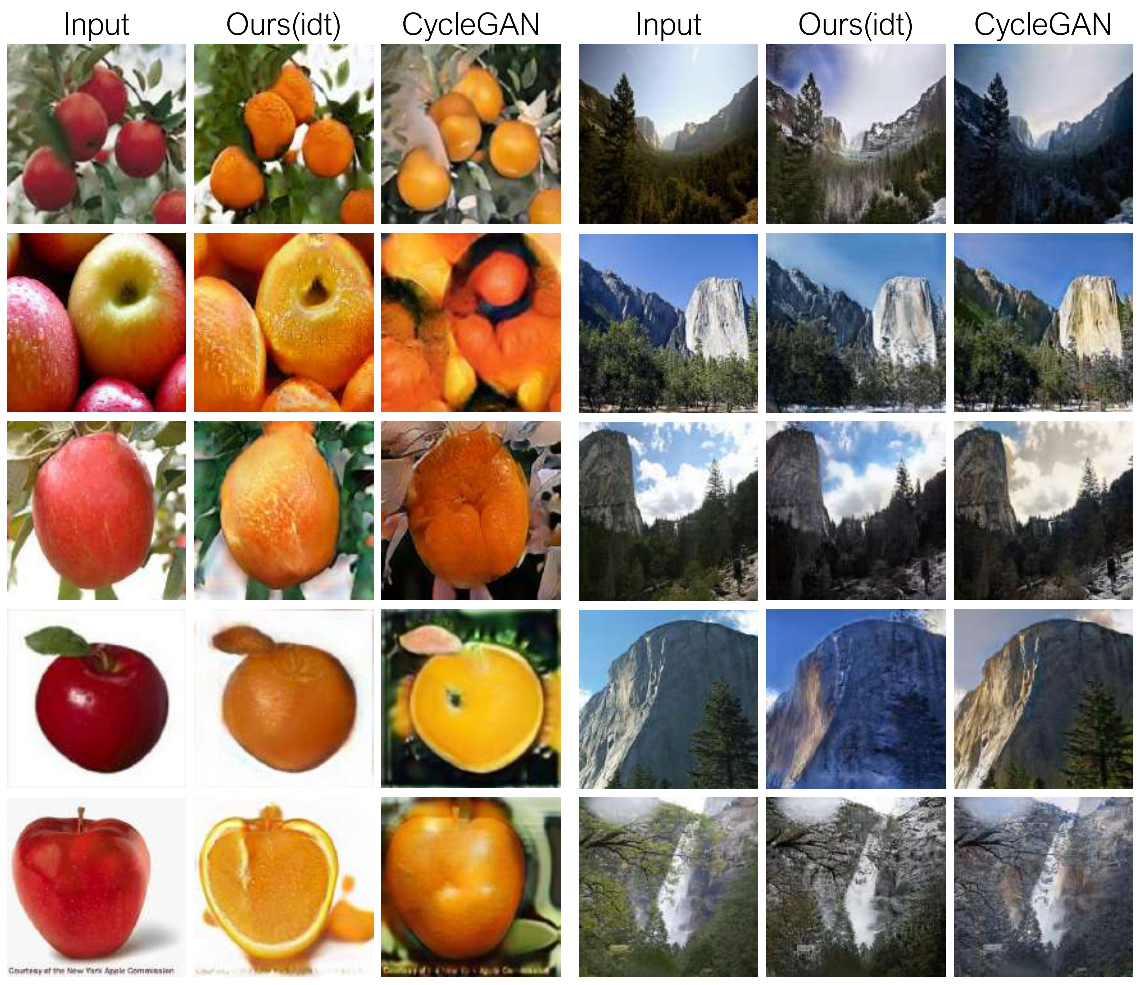


Figure 3.22: **Apple→Orange** and **Summer→Winter Yosemite**. CycleGAN models were downloaded from the authors' public code repository. Apple→Orange shows that CycleGAN may suffer from color flipping issue.

Method	**Cityscapes**				**Cat→Dog**	**Horse→Zebra**		
	mAP↑	**pixAcc↑**	**classAcc↑**	**FID↓**	**FID↓**	**FID↓**	**sec/iter↓**	**Mem(GB)↓**
CycleGAN [41]	20.4	55.9	25.4	76.3	85.9	77.2	0.40	4.81
MUNIT [42]	16.9	56.5	22.5	91.4	104.4	133.8	0.39	3.84
DRIT [108]	17.0	58.7	22.2	155.3	123.4	140.0	0.70	4.85
Distance [123]	8.4	42.2	12.6	81.8	155.3	72.0	**0.15**	2.72
SelfDistance [123]	15.3	56.9	20.6	78.8	144.4	80.8	0.16	2.72
GCGAN [124]	21.2	63.2	26.6	105.2	96.6	86.7	0.26	2.67
CUT	**24.7**	**68.8**	**30.7**	**56.4**	**76.2**	**45.5**	0.24	3.33
FastCUT	19.1	59.9	24.3	68.8	94.0	73.4	**0.15**	**2.25**

Table 3.6: **Comparison with baselines** We compare our methods across datasets on common evaluation metrics. CUT denotes our model trained with the identity loss ($\lambda_X = \lambda_Y = 1$), and FastCUT without it ($\lambda_X = 10, \lambda_Y = 0$). We show FID, a measure of image quality [66] (lower is better). For Cityscapes, we show the semantic segmentation scores (mAP, pixAcc, classAcc) to assess the discovered correspondence (higher is better for all metrics). Based on quantitative measures, CUT produces higher quality and more accurate generations with light footprint in terms of training speed (seconds per sample) and GPU memory usage. Our variant FastCUT also produces competitive results with even lighter computation cost of training.

the generator is horizontally flipped, and the output features are flipped back before computing the PatchNCE loss. Our encoder G_{enc} is the first half of the CycleGAN generator [41]. In order to calculate our multi-layer, patch-based contrastive loss, we extract features from 5 layers, which are RGB pixels, the first and second downsampling convolution, and the first and the fifth residual block. The layers we use correspond to receptive fields of sizes 1×1, 9×9, 15×15, 35×35, and 99×99. For each layer's features, we sample 256 random locations, and apply 2-layer MLP to acquire 256-dim final features. For our baseline model that uses MoCo-style memory bank [146], we follow the setting of MoCo, and used momentum value 0.999 with temperature 0.07. The size of the memory bank is 16384 per layer, and we enqueue 256 patches per image per iteration.

Unpaired image translation

Datasets We conduct experiments on the following datasets.

- *Cat→Dog* contains 5,000 training and 500 val images from AFHQ Dataset [178].
- *Horse→Zebra* contains 2,403 training and 260 zebra images from ImageNet [126] and was introduced in CycleGAN [41].
- *Cityscapes* [31] contains street scenes from German cities, with 2,975 training and 500 validation images. We train models at 256×256 resolution. Unlike previous datasets listed, this does have corresponding labels. We can leverage this to

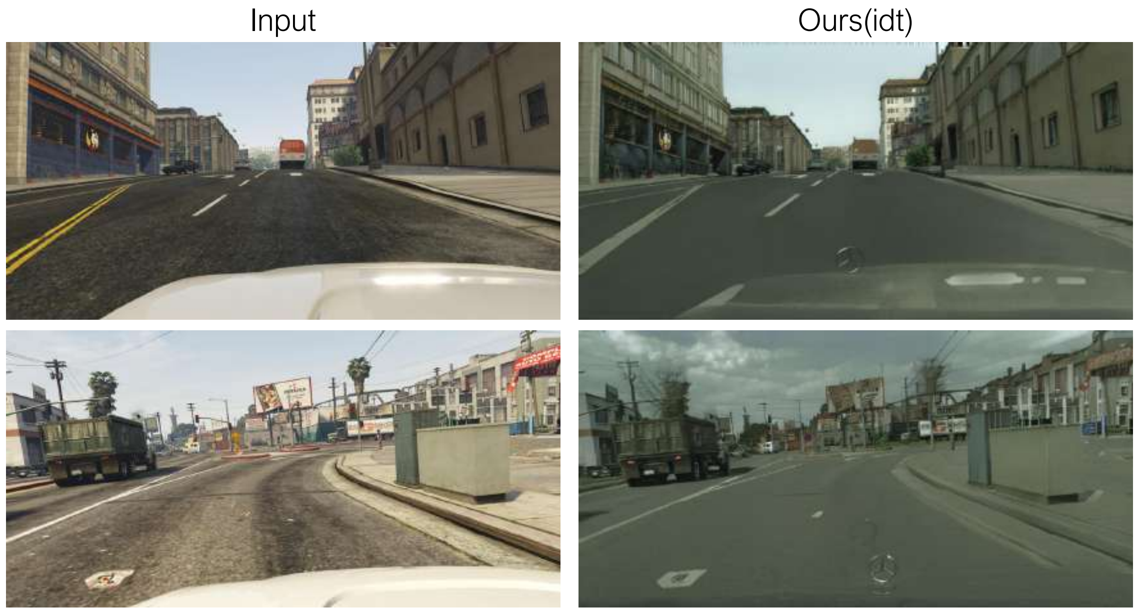


Figure 3.23: **GTA→Cityscapes** results at 1024 × 512 resolution. The model was trained on 512 × 512 crops.

.

measure how well our unpaired algorithm discovers correspondences.

In Figure 3.22 and Figure 3.23, we show qualitative results on additional datasets, compared against baseline method CycleGAN [41]. Our method provides better or comparable results, demonstrating its flexibility across a variety of datasets.

- *Apple→Orange* contains 996 apple and 1,020 orange images from ImageNet and was introduced in CycleGAN [41].
- *Yosemite Summer→Winter* contains 1,273 summer and 854 winter images of Yosemite scraped using the FlickAPI was introduced in CycleGAN [41].
- *GTA→Cityscapes* GTA contains 24,966 images [179] and Cityscapes [31] contains 19,998 images of street scenes from German cities. The task was originally used in CyCADA [180].

Evaluation protocol. We adopt the evaluation protocols from [41, 66], aimed at assessing *visual quality* and *discovered correspondence*. For the first, we utilize the widely-used Fréchet Inception Distance (FID) metric, which empirically estimates the distribution of real and generated images in a deep network space and computes the divergence between them. Intuitively, if the generated images are realistic, they should have similar summary statistics as real images, in any feature space. For *Cityscapes* specifically, we have ground truth of paired label maps. If accurate correspondences are discovered, the algorithm should generate images that are recognizable as the correct class. Using an off-the-shelf network to test "semantic interpretability" of image translation results has been commonly used [20, 139]. We use the pretrained semantic segmentation network DRN [71]. We train the DRN at 256x128 resolution, and compute mean average precision (mAP), pixel-wise accuracy (pixAcc), and average class accuracy (classAcc).

Fréchet Inception Distance (FID [66]) throughout this paper is computed by resizing the images to 299-by-299 using bilinear sampling of PyTorch framework, and then taking the activations of the last average pooling layer of a pretrained Inception V3 [181] using the weights provided by the TensorFlow framework. We use the default setting of `https://github.com/mseitzer/pytorch-fid`. All test set images are used for evaluation, unless noted otherwise.

Semantic segmentation metrics on the Cityscapes dataset are computed as follows. First, we trained a semantic segmentation network using the DRN-D-22 [71] architecture. We used the recommended setting from `https://github.com/fyu/drn`, with batch size 32 and learning rate 0.01, for 250 epochs at 256x128 resolution. The output images of the 500 validation labels are resized to 256x128 using bicubic downsampling, passed to the trained DRN network, and compared against the ground truth labels downsampled to the same size using nearest-neighbor sampling.

Comparison to baselines. In Table 3.6, we show quantitative measures of our and Figure 3.20, we compare our method to baselines. We present two settings of our method in Equation 3.9: CUT with the identity loss ($\lambda_X = \lambda_Y = 1$), and FastCUT without it ($\lambda_X = 10, \lambda_Y = 0$). On image quality metrics across datasets, our methods outperform baselines. In addition, our Cityscapes semantic segmentation scores are higher, suggesting that our method is able to find correspondences between output and input.

Speed and memory. Since our model is one-sided, our method is memory-efficient and fast. For example, our method with the identity loss was 40% faster and 31% more memory-efficient than CycleGAN at training time, using the same architectures as CycleGAN (Table 3.6). Furthermore, our faster variant FastCUT is 63% faster and 53% lighter, while achieving superior metrics to CycleGAN. Table 3.6 contains the speed and memory usage of each method measured on NVIDIA GTX 1080Ti, and shows that FastCUT achieves competitive FIDs and segmentation scores with a lower time and memory requirement. Therefore, our method can serves as a practical, lighter alternative in scenarios, when an image translation model is jointly trained with other components [180, 182].

Method	Training settings					Testing datasets		
	Id	Negs	Layers	Int	Ext	Horse→Zebra	Cityscapes	
						FID↓	FID↓	mAP↑
CUT (default)	✓	255	All	✓	✗	45.5	**56.4**	**24.7**
no id	✗	255	All	✓	✗	39.3	68.5	22.0
no id, 15 neg	✗	15	All	✓	✗	44.1	59.7	23.1
no id, 15 neg, last	✗	15	Last	✓	✗	**38.1**	114.1	16.0
last	✓	255	Last	✓	✗	441.7	141.1	14.9
int and ext	✓	255	All	✓	✓	56.4	64.4	20.0
ext only	✓	255	All	✗	✓	53.0	110.3	16.5
ext only, last	✓	255	Last	✗	✓	60.1	389.1	5.6

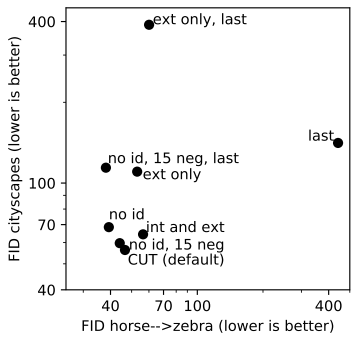


Figure 3.24: **Ablations.** The PatchNCE loss is trained with negatives from each layer output of the same (internal) image, with the identity preservation regularization. *(Left)* We try removing the identity loss [**Id**], using less negatives [**Negs**], using only the last layer of the encoder [**Layers**], and varying where patches are sampled, internal [**Int**] vs external [**Ext**]. *(Right)* We plot the FIDs on horse→zebra and Cityscapes dataset. Removing the identity loss (`no id`) and reducing negatives (`no id, 15 neg`) still perform strongly. In fact, our variant FastCUT does not use the identity loss. However, reducing number of layers (`last`) or using external patches (`ext`) hurts performance.

Ablation study and analysis

We find that in the image synthesis setting, similarly to the unsupervised learning setting [145–147], implementation choices for contrastive loss are important. Here, try various settings and ablations of our method, summarized in Figure 3.24. By default, we use the ResNet-based generator used in CycleGAN [41], with patchNCE using (a) negatives sampled from the input image, (b) multiple layers of the encoder, and (c) a PatchNCE loss $\mathcal{L}_{\text{PatchNCE}}(G, H, Y)$ on domain Y. In Figure 3.24, we show results using several variants and ablations, taken after training for 400 epochs. We show qualitative examples in Figure 3.25.

Internal negatives are more effective than external. By default, we sample negatives from *within* the same image (internal negatives). We also try adding negatives from other images, using a momentum encoder [146]. However, the external negatives, either as addition (`int and ext`) or replacement of internal negatives (`ext only`), hurts performance. In Figure 3.25, we see a loss of quality, such as repeated texture in the Cityscapes dataset, indicating that sampling negatives from the same image serves as a stronger signal for preserving content.

Importance of using multiple layers of encoder. Our method uses multiple layers of the encoder, every four layers from pixels to the 16th layer. This is consistent with the standard use of ℓ_1+VGG loss, which uses layers from the pixel level up

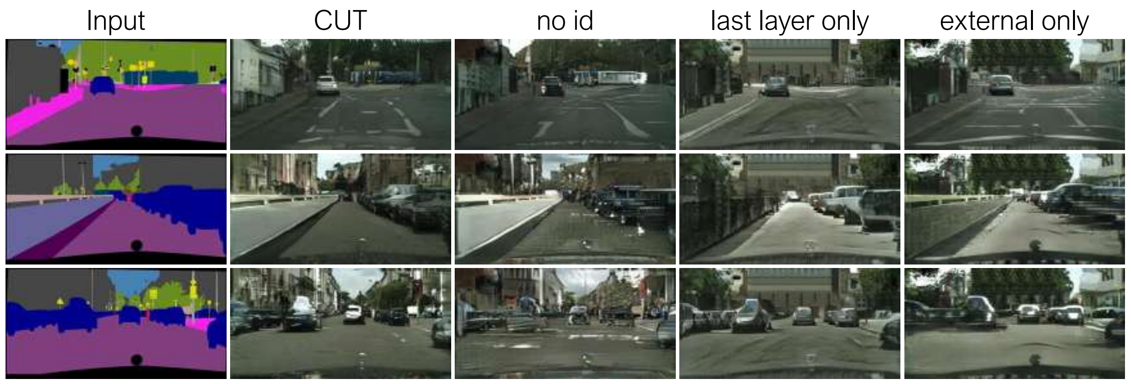


Figure 3.25: **Qualitative ablation results** of our full method (CUT) are shown: *without* the identity loss $\mathcal{L}_{\text{PatchNCE}}(G, H, Y)$ on domain Y (`no id`), using only one layer of the encoder (`last layer only`), and using external instead of internal negatives (`external only`). The ablations cause noticeable drop in quality, including repeated building or vegetation textures when using only external negatives or the last layer output.

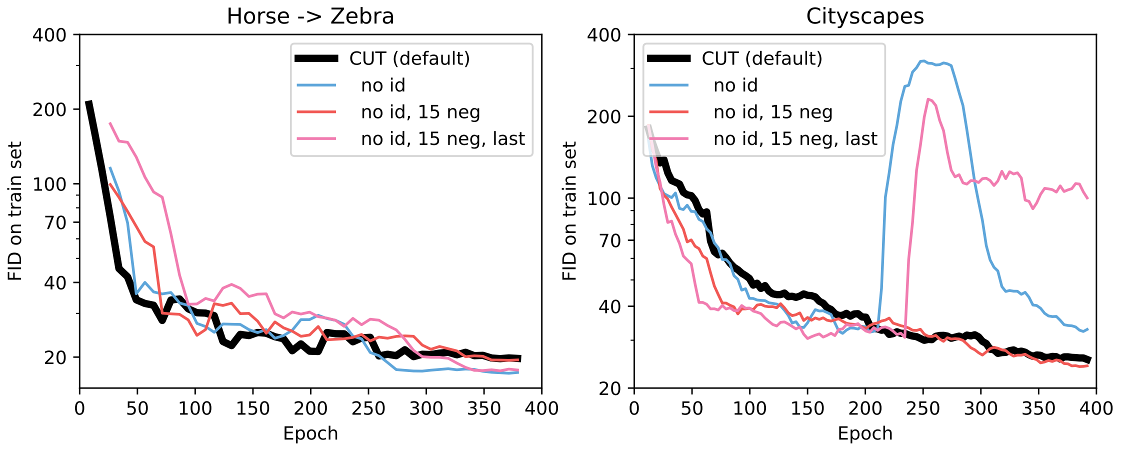


Figure 3.26: **Identity loss $\mathcal{L}_{\text{PatchNCE}}(G, H, Y)$ on domain Y adds stability.** This regularizer encourages an image from the output domain $\boldsymbol{y}$ to be unchanged by the generator. Using it (shown in **bold, black** curves), we observe better stability in comparison to other variants. On the left, our variant without the regularizer, `no id`, achieves better FID. However, we see higher variance in the training curve. On the right, training without the regularizer can lead to collapse.

to a deep convolutional layer. On the other hand, many contrastive learning-based unsupervised learning papers map the whole image into a single representation. To emulate this, we try only using the last layer of the encoder (`last`), and try a variant using external negatives only (`ext only, last`). Performance is drastically reduced in both cases. In unsupervised representation learning, the input images are fixed.

For our application, the loss is being used as a signal for synthesizing an image. As such, this indicates that the dense supervision provided by using multiple layers of the encoder is important when performing image synthesis.

$\mathcal{L}_{\textbf{PatchNCE}}(G, H, Y)$ regularizer stabilizes training. Given an image from the output domain $\boldsymbol{y} \in \mathcal{Y}$, this regularizer encourages the generator to leave the image unchanged with our patch-based contrastive loss. We also experiment with a variant without this regularizer, `no id`. As shown in Figure 3.24, removing the regularizer improves results for the horse→zebra task, but decreases performance on Cityscapes. We further investigate by showing the training curves in Figure 3.26, across 400 epochs. In the Cityscapes results, the training can collapse without the regularizer (although it can recover). We observe that although the final FID is sometimes better without, the training is more stable with the regularizer.

Visualizing learned similarity by encoder $G_{\textbf{enc}}$ To further understand why our encoder network G_{enc} has learned to perform horse→ zebra task, we study the output space of the 1st residual block for both horse and zebra features. As shown in Figure 3.27. Given an input and output image, we compute the distance between a query patch's feature vector $\boldsymbol{v}$ (highlighted as red or blue dot) to feature vectors $\boldsymbol{v}^-$ of all the patches in the input using $\exp(\boldsymbol{v} \cdot \boldsymbol{v}^- / \tau)$ (Equation 3.6). Additionally, we perform a PCA dimension reduction on feature vectors from both horse and zebra patches. In (d) and (e), we show the top three principal components, which looks similar before and after translation. This indicates that our encoder is able to bring the corresponding patches from two domains into a similar location in the feature embedding space.

Distribution matching In Figure 3.28, we show an interesting phenomenon of our method, caused by the training set imbalance of the horse→zebra set. We use an off-the-shelf DeepLab model [68] trained on COCO-Stuff [28], to measure the percentage of pixels that belong to horses and zebras[6]. The training set exhibits dataset bias [183]. On average, zebras appear in more close-up pictures than horses and take up about twice the number of pixels (37% vs 18%). To perfectly satisfy the discriminator, a translation model should attempt to match the statistics of the training set. Our method allows the flexibility for the horses to change the size, and the percentage of output zebra pixels (31%) better matches the training distribution (37%) than the CycleGAN baseline (19%). On the other hand, our fast variant *FastCUT* uses a larger weight ($\lambda_X = 10$) on the Patch NCE loss and flip-equivariance augmentation, and hence behaves more conservatively and more similar

[6]Pretrained model from https://github.com/kazuto1011/deeplab-pytorch

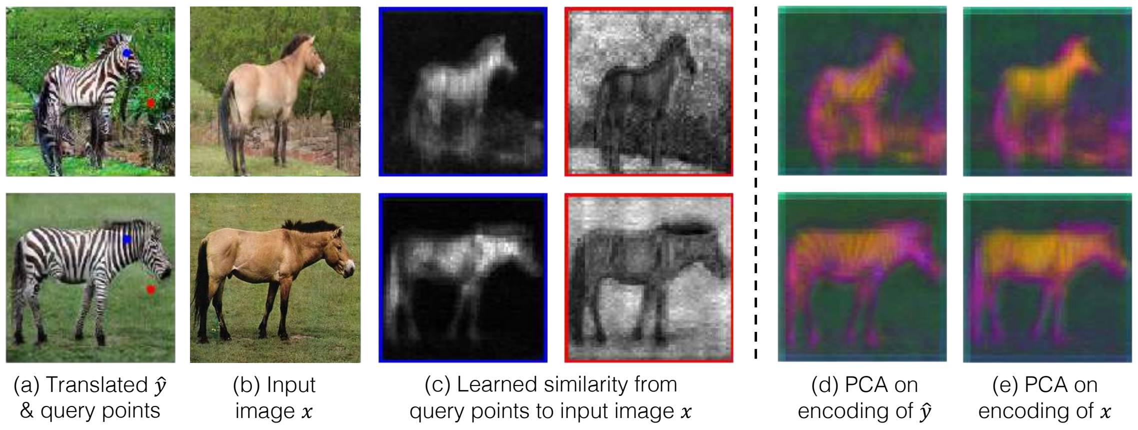

(a) Translated $\hat{y}$ & query points
(b) Input image x
(c) Learned similarity from query points to input image x
(d) PCA on encoding of $\hat{y}$
(e) PCA on encoding of x

Figure 3.27: **Visualizing the learned similarity by G_{enc}.** Given query points (blue or red) on an output image (a) and input (b), we visualize the learned similarity to patches on the input image by computing $\exp(\boldsymbol{v} \cdot \boldsymbol{v}^{-}/\tau)$ in (c). Here $\boldsymbol{v}$ is the query patch in the output and $\boldsymbol{v}^{-}$ denotes patches from the input. This suggests that our encoder may learn cross-domain correspondences implicitly. In (d) and (e), we visualize the top 3 PCA components of the shared embedding.

to CycleGAN. The strong distribution matching capacity has pros and cons. For certain applications, it can create introduce undesired changes (e.g., zebra patterns on the background for horse→zebra). On the other hand, it can enable dramatic geometric changes for applications such as Cat→Dog.

Additional Ablation studies Here we present additional ablation studies on more subtle design choices. We run all the variants on horse2zebra datasets [41]. The FID of our original model is **46.6**. We compare it to the following two variants of our model:

- Ours without weight sharing for the encoder G_{enc} and MLP projection network H: for this variant, when computing features $\{\boldsymbol{z}_l\}_L = \{H_l(G^l_{\text{enc}}(\boldsymbol{x}))\}_L$, we use two separate encoders and MLP networks for embedding input images (e.g., horse) and the generated images (e.g., zebras) to feature space. They do not share any weights. The FID of this variant is **50.5**, worse than our method. This shows that weight sharing helps stabilize training while reducing the number of parameters in our model.
- Ours without updating the decoder G_{dec} using *PatchNCE* loss: in this variant, we exclude the gradient propagation of the decoder G_{dec} regarding *PatchNCE* loss $\mathcal{L}_{\text{PatchNCE}}$. In other words, the decoder G_{dec} only gets updated through the adversarial loss $\mathcal{L}_{\text{GAN}}$. The FID of this variant is **444.2**, and the results contain severe artifacts. This shows that our $\mathcal{L}_{\text{PatchNCE}}$ not only helps learn the encoder

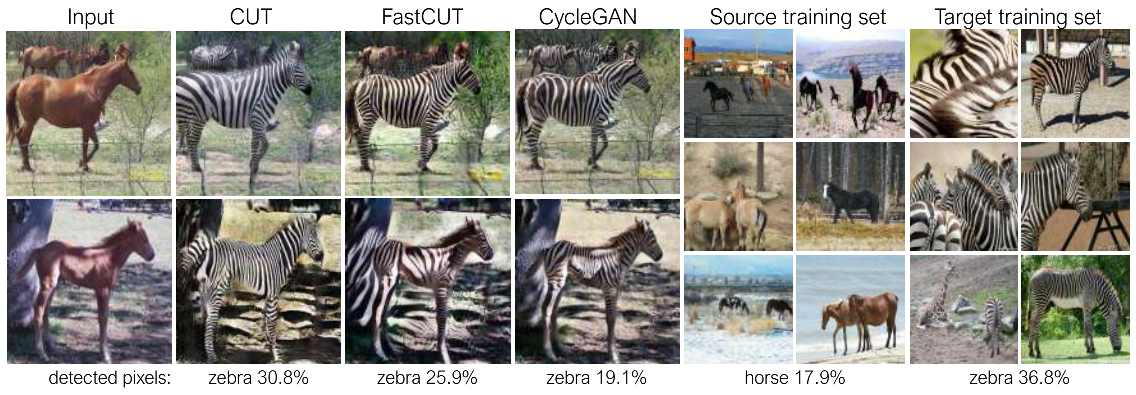


Figure 3.28: **Distribution matching.** We measure the percentage of pixels belonging to the horse/zebra bodies, using a pre-trained semantic segmentation model. We find a distribution mismatch between sizes of horses and zebras images – zebras usually appear larger (36.8% vs. 17.9%). Our full method CUT has the flexibility to enlarge the horses, as a means of better matching of the training statistics than CycleGAN [41]. Our faster variant FastCUT, trained with a higher PatchNCE loss ($\lambda_X = 10$) and flip-equivariance augmentation, behaves more conservatively like CycleGAN.

G_{enc}, as done in previous unsupervised feature learning methods [146], but also learns a better decoder G_{dec} together with the GAN loss. Intuitively, if the generated result has many artifacts and is far from realistic, it would be difficult for the encoder to find correspondences between the input and output, producing a large *PatchNCE* loss.

Additional applications . Figure 3.29 shows additional results: Parisian street → Burano's brightly painted houses and Russian Blue cat → Grumpy cat.

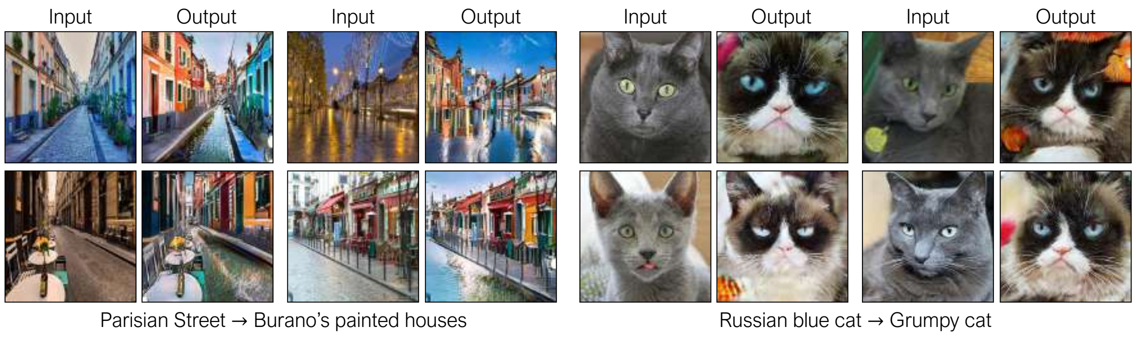

Input
Output
Input
Output
Input
Output
Input
Output
Parisian Street → Burano's painted houses
Russian blue cat → Grumpy cat

Figure 3.29: **Additional applications** on Parisian street → Burano's colored houses and Russian Blue cat → Grumpy cat.

High-resolution single image translation

Finally, we conduct experiments in the single image setting, where both the source and target domain only have one image each. Here, we transfer a Claude Monet's painting to a natural photograph. Recent methods [184, 185] have explored training unconditional models on a single image. Bearing the additional challenge of respecting the structure of the input image, conditional image synthesis using only one image has not been explored by previous image-to-image translation methods. Our painting $\rightarrow$ photo task is also different from neural style transfer [116, 128] (photo $\rightarrow$ painting) and photo style transfer [186, 187] (photo $\rightarrow$ photo).

Since the whole image (at HD resolution) cannot fit on a commercial GPU, at each iteration we train on 16 random crops of size 128×128. We also randomly scale the image to prevent overfitting. Furthermore, we observe that limiting the receptive field of the discriminator is important for preserving the structure of the input image, as otherwise the GAN loss will force the output image to be identical to the target image. Therefore, the crops are further split into 64×64 patches before passed to the discriminator. Lastly, we find that using gradient penalty [13, 35] stabilizes optimization. We call this variant SinCUT.

Figure 3.30 shows a qualitative comparison between our results and baseline methods including two neural style transfer methods (Gatys et al. [128] and STROTSS [188]), one leading photo style transfer method WCT^2 [187], and a CycleGAN baseline [41] that uses the ℓ_1 cycle-consistency loss instead of our contrastive loss at the patch level. The input paintings are high-res, ranging from 1k to 1.5k. We observe that Gatys et al. [128] fails to synthesize realistic textures. Existing photo style transfer methods such as WCT^2 can only modify the color of the input image. Our method SinCUT outperforms CycleGAN and is comparable to a leading style transfer method [188], which is based on optimal transport and self-similarity. Interestingly, our method is not originally designed for this application. This result suggests the intriguing connection between image-to-image translation and neural style transfer.

In Figure 3.31 and Figure 3.32, we show additional comparison results for our method, Gatys et al. [128], STROTSS [188], WCT^2 [187], and CycleGAN baseline [41]. Note that the CycleGAN baseline adopts the same augmentation techniques as well as the same generator/discriminator architectures as our method. The image resolution is at 1-2 Megapixels. Please zoom in to see more visual details.

Both figures demonstrate that our results look more photorealistic compared to CycleGAN baseline, Gatys et al [128], and WCT^2. The quality of our results is on par with results from STROTSS [188]. Note that STROTSS [188] compares to and outperforms recent style transfer methods (e.g., [131, 189]).

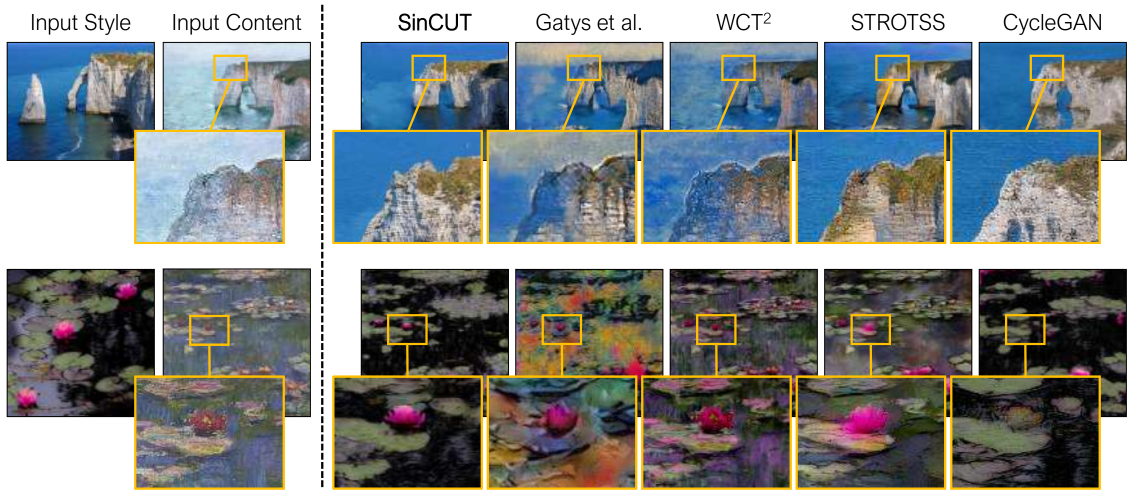


Figure 3.30: **High-res painting to photo translation.** We transfer Claude Monet's paintings to reference natural photographs. The training only requires a single image from each domain. We compare our results (SinCUT) to recent style and photo transfer methods including Gatys et al. [128], WCT^2 [187], STROTSS [188], and patch-based CycleGAN [41]. Our method generates can reproduce the texture of the reference photo while retaining structure of input painting. Our generation is at 1k $\sim$ 1.5k resolution.

Training details. At each iteration, the input image is randomly scaled to a width between 384 to 1024, and we randomly sample 16 crops of size 128×128. To avoid overfitting, we divide crops into 64×64 tiles before passing them to the discriminator. At test time, since the generator network is fully convolutional, it takes the input image at full size.

We found that adopting the architecture of StyleGAN2 [177] instead of CycleGAN slightly improves the output quality, although the difference is marginal. Our StyleGAN2-based generator consists of one downsampling block of StyleGAN2 discriminator, 6 StyleGAN2 residual blocks, and one StyleGAN2 upsampling block. Our discriminator has the same architecture as StyleGAN2. Following StyleGAN2, we use non-saturating GAN loss [190] with R1 gradient penalty [35]. Since we do not use style code, the style modulation layer of StyleGAN2 was removed.

3.3.3 Discussion

We propose a straightforward method for encouraging content preservation in unpaired image translation problems – by maximizing the mutual information between input and output with contrastive learning. The objective learns an embedding to bringing together corresponding patches in input and output, while pushing away

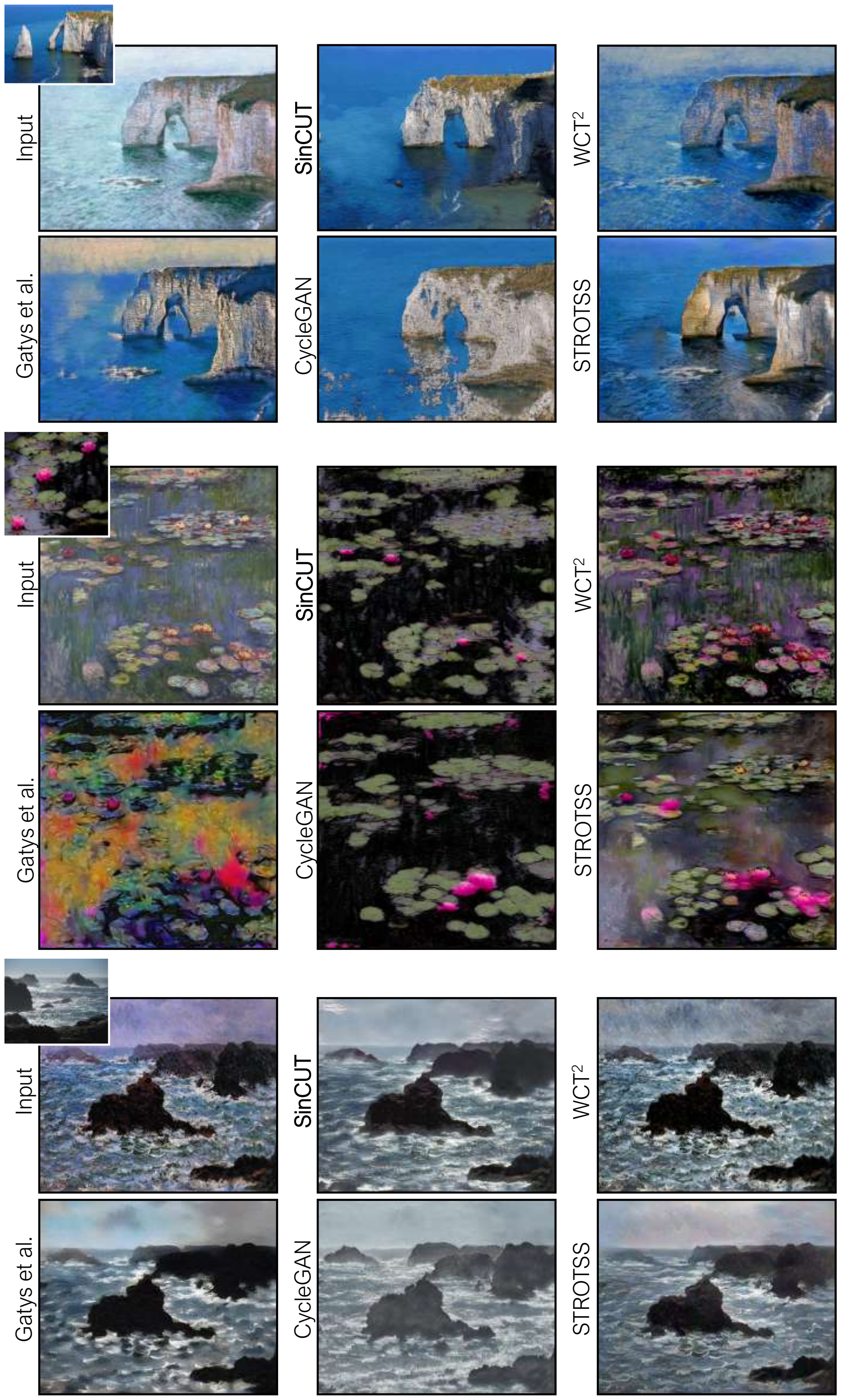


Figure 3.31: **High-res painting to photo translation (I).** We transfer Monet's paintings to reference natural photos shown as insets at top-left corners. The training only requires a single image from each domain. We compare our results to recent style and photo transfer methods including Gatys et al. [128], WCT^2 [187], STROTSS [188], and our modified patch-based CycleGAN [41]. Our method can reproduce the texture of the reference photos while retaining structure of the input paintings. Our results are at 1k $\sim$ 1.5k resolution.

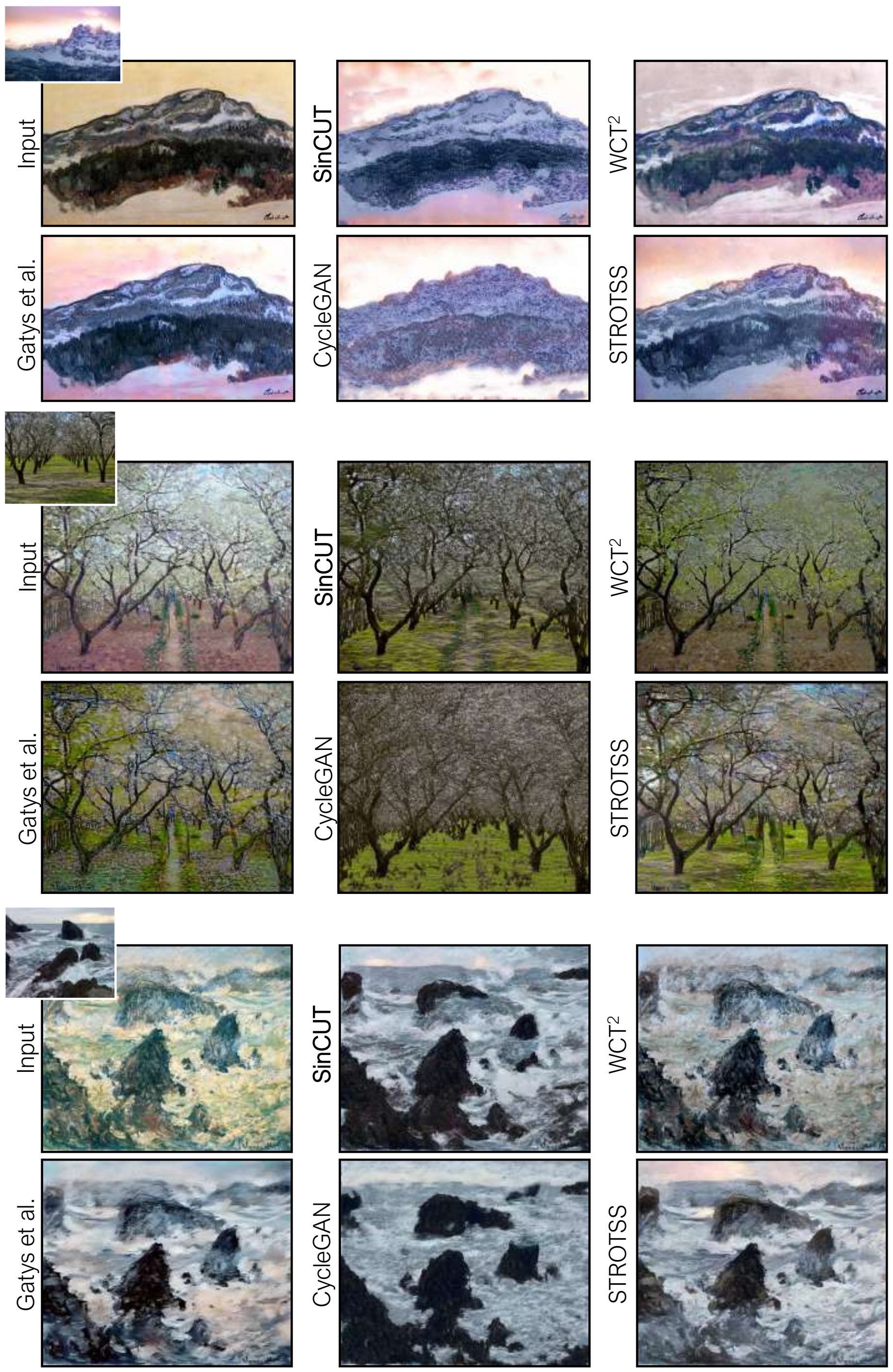


Figure 3.32: **High-res painting to photo translation (II).** We transfer Monet's paintings to reference natural photos shown as insets at top-left corners. The training only requires a single image from each domain. We compare our results to recent style and photo transfer methods including Gatys et al. [128], WCT^2 [187], STROTSS [188], and our modified patch-based CycleGAN [41]. Our method can reproduce the texture of the reference photos while retaining structure of the input paintings. Our results are at 1k $\sim$ 1.5k resolution.

noncorresponding “negative” patches. We study several important design choices. Interestingly, drawing negatives from *within* the image itself, rather than other images, provides a stronger signal. Our method *learns a cross-domain similarity function* and is the first image translation algorithm, to our knowledge, to not use any pre-defined similarity function (such as ℓ_1 or perceptual loss). As our method does not rely on cycle-consistency, it can enable one-sided image translation, with better quality than established baselines. In addition, our method can be used for *single-image* unpaired translation.

3.4 Applying Image Translation for Domain Adaptation

The versatility of unpaired image translation opens up applications other than image editing. In this section, we show that CycleGAN [41] can be used in domain adaptation, by reducing the visual gap between two domains.[7]

3.4.1 Related Work

The problem of visual domain adaptation was introduced along with a pairwise metric transform solution by [191] and was further popularized by the broad study of visual dataset bias [192]. Early deep adaptive works focused on feature space alignment through minimizing the distance between first or second order feature space statistics of the source and target [193,194]. These latent distribution alignment approaches were further improved through the use of domain adversarial objectives whereby a domain classifier is trained to distinguish between the source and target representations while the domain representation is learned so as to maximize the error of the domain classifier. The representation is optimized using the standard minimax objective [195], the symmetric confusion objective [196], or the inverted label objective [197]. Each of these objectives is related to the literature on generative adversarial networks [198] and follow-up work for improved training procedures for these networks [199, 200].

The feature-space adaptation methods described above focus on modifications to the discriminative representation space. In contrast, other recent methods have sought adaptation in the pixel-space using various generative approaches. One advantage of pixel-space adaptation, as we have shown, is that the result may be more human interpretable, since an image from one domain can now be visualized in a new domain. CoGANs [201] jointly learn a source and target representation through explicit weight sharing of certain layers while each source and target has a unique generative adversarial objective. [202] uses an additional reconstruction objective in the target domain to encourage alignment in the unsupervised adaptation setting.

In contrast, another approach is to directly convert the target image into a source style image (or visa versa), largely based on Generative Adversarial Networks (GANs) [198]. Researchers have successfully applied GANs to various applications such as image generation [77–79], image editing [80] and feature learning [81, 165].

[7]This work was first published as *CyCADA: Cycle-consistent adversarial domain adaptation* at ICML, 2018 [180].

Recent work [40, 73, 88] adopt conditional GANs [203] for these image-to-image translation problems [73], but they require input-output image pairs for training, which is in general not available in domain adaptation problems.

There also exist lines of work where such training pairs are not given. [204] learns a source to target encoder-decoder along with a generative adversarial objective on the reconstruction which is is applied for predicting the clothing people are wearing. The Domain Transfer Network [205] trains a generator to transform a source image into a target image by enforcing consistency in the embedding space. [206] instead uses an L1 reconstruction loss to force the generated target images to be similar to their original source images.This works well for limited domain shifts where the domains are similar in pixel-space, but can be too limiting for settings with larger domain shifts. [207] use a content similarity loss to ensure the generated target image is similar to the original source image; however, this requires prior knowledge about which parts of the image stay the same across domains (e.g. foreground). Our method does not require pre-defining what content is shared between domains and instead simply translates images back to their original domains while ensuring that they remain identical to their original versions. BiGAN [165] and ALI [166] take an approach of simultaneously learning the transformations between the pixel and the latent space. More recently, Cycle-consistent Adversarial Networks (CycleGAN) [208] produced compelling image translation results such as generating photorealistic images from impressionism paintings or transforming horses into zebras at high resolution using the cycle-consistency loss. This loss was simultaneously proposed by [104] and [209] to great effect as well. Our motivation comes from such findings about the effectiveness of the cycle-consistency loss.

Few works have explicitly studied visual domain adaptation for the semantic segmentation task. Adaptation across weather conditions in simple road scenes was first studied by [210]. More recently, a convolutional domain adversarial based approached was proposed for more general drive cam scenes and for adaptation from simulated to real environments [211]. [212] learns a multi-source model through concatenating all available labeled data and learning a single large model and then transfers to a sparsely labeled target domain through distillation [213]. [214] use an adversarial objective to align both global and class-specific statistics, while mining additional temporal data from street view datasets to learn a static object prior. [215] instead perform segmentation adaptation by aligning label distributions both globally and across superpixels in an image.

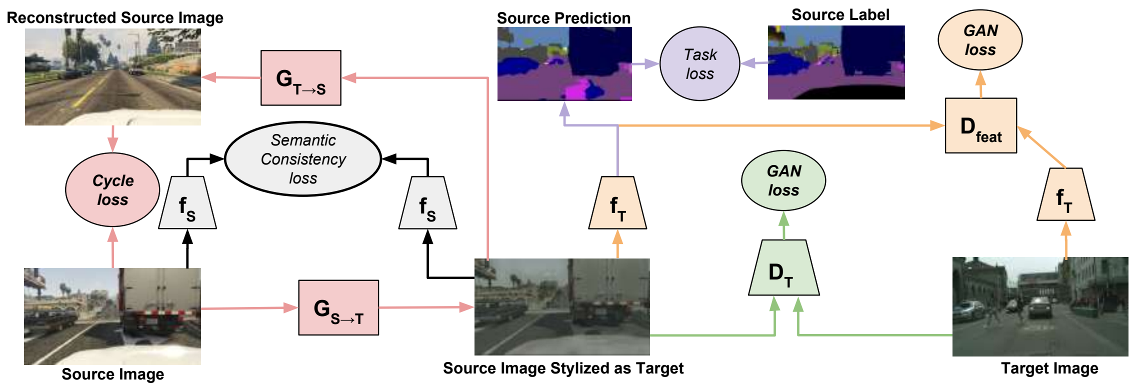


Figure 3.33: Cycle-consistent adversarial adaptation of pixel-space inputs. By directly remapping source training data into the target domain, we remove the low-level differences between the domains, ensuring that our task model is well-conditioned on target data. We depict here the image-level GAN loss (green), the feature level GAN loss (orange), the source and target semantic consistency losses (black), the source cycle loss (red), and the source task loss (purple). For clarity the target cycle is omitted.

3.4.2 Cycle-Consistent Adversarial Domain Adaption

We consider the problem of unsupervised adaptation, where we are provided source data X_S, source labels Y_S, and target data X_T, but no target labels. The goal is to learn a model f that can correctly predict the label for the target data X_T.

We can begin by simply learning a source model f_S that can perform the task on the source data. For K-way classification with a cross-entropy loss, this corresponds to

$$\mathcal{L}_{\text{task}}(f_S, X_S, Y_S) = -\mathbb{E}_{(x_s,y_s)\sim(X_S,Y_S)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log\left(\sigma(f_S^{(k)}(x_s))\right) \tag{3.10}$$

where σ denotes the softmax function. However, while the learned model f_S will perform well on the source data, typically domain shift between the source and target domain leads to reduced performance when evaluating on target data. To mitigate the effects of domain shift, we follow previous adversarial adaptation approaches and learn to map samples across domains such that an adversarial discriminator is unable to distinguish the domains. By mapping samples into a common space, we enable our model to learn on source data while still generalizing to target data.

To this end, we introduce a mapping from source to target $G_{S\to T}$ and train it to produce target samples that fool an adversarial discriminator D_T. Conversely, the adversarial discriminator attempts to classify the real target data from the source target data. This corresponds to the loss function

$$\mathcal{L}_{\text{GAN}}(G_{S\to T}, D_T, X_T, X_S) = \mathbb{E}_{x_t\sim X_T}\left[\log D_T(x_t)\right] + \mathbb{E}_{x_s\sim X_S}\left[\log(1 - D_T(G_{S\to T}(x_s)))\right] \tag{3.11}$$

This objective ensures that $G_{S\to T}$, given source samples, produces convincing target samples. In turn, this ability to directly map samples between domains allows us to learn a target model f_T by minimizing $\mathcal{L}_{\text{task}}(f_T, G_{S\to T}(X_S), Y_S)$ (see Figure 3.33 green portion).

However, while previous approaches that optimized similar objectives have shown effective results, in practice they can often be unstable and prone to failure. Although the GAN loss in Equation 3.11 ensures that $G_{S\to T}(x_s)$ for some x_s will resemble data drawn from X_T, there is no way to guarantee that $G_{S\to T}(x_s)$ preserves the structure or content of the original sample x_s.

In order to encourage the source content to be preserved during the conversion process, we impose a cycle-consistency constraint on our adaptation method [104, 208, 209] (see Figure 3.33 red portion). To this end, we introduce another mapping from target to source $G_{T\to S}$ and train it according to the same GAN loss $\mathcal{L}_{\text{GAN}}(G_{T\to S}, D_S, X_S, X_T)$. We then require that mapping a source sample from source to target and back to the source reproduces the original sample, thereby enforcing cycle-consistency. In other words, we want $G_{T\to S}(G_{S\to T}(x_s)) \approx x_s$ and $G_{S\to T}(G_{T\to S}(x_t)) \approx x_t$. This is done by imposing an L1 penalty on the reconstruction error, which is referred to as the *cycle-consistency loss*:

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G_{S\to T}, G_{T\to S}, X_S, X_T) = {} & \mathbb{E}_{x_s\sim X_S}\left[||G_{T\to S}(G_{S\to T}(x_s)) - x_s||_1\right] \\ & + \mathbb{E}_{x_t\sim X_T}\left[||G_{S\to T}(G_{T\to S}(x_t)) - x_t||_1\right].\end{aligned} \tag{3.12}$$

Optionally, as we have access to source labeled data, we explicitly encourage high semantic consistency before and after image translation. We pretrain a source task model f_S, fixing the weights, we use this model as a noisy labeler by which we encourage an image to be classified in the same way after translation as it was before translation according to this classifier. Let us define the predicted label from a fixed classifier, f, for a given input X as $p(f, X) = \arg\max(f(X))$. Then we can define the semantic consistency before and after image translation as follows:

$$\begin{aligned}\mathcal{L}_{\text{sem}}(G_{S\to T}, G_{T\to S}, X_S, X_T, f_S) = {} & \mathcal{L}_{\text{task}}(f_S, G_{T\to S}(X_T), p(f_S, X_T)) \\ & + \mathcal{L}_{\text{task}}(f_S, G_{S\to T}(X_S), p(f_S, X_S))\end{aligned} \tag{3.13}$$

See Figure 3.33 black portion. This can be viewed as analogously to content losses in style transfer [128] or in pixel adaptation [216], where the shared content to preserve is dictated by the source task model f_S. However, utilizing the semantic consistency loss in the training loop is hard in practice, because of the memory overhead of loading the semantic information as well as the classifier f. For this reason, the semantic consistency loss was left as an idea, and not used in our experiments.

Taken together, these loss functions form our complete objective:

$$\begin{aligned}
&\mathcal{L}_{\text{CyCADA}}(f_T, X_S, X_T, Y_S, G_{S\to T}, G_{T\to S}, D_S, D_T) \\
&\quad = \mathcal{L}_{\text{task}}(f_T, G_{S\to T}(X_S), Y_S) \\
&\quad + \mathcal{L}_{\text{GAN}}(G_{S\to T}, D_T, X_T, X_S) + \mathcal{L}_{\text{GAN}}(G_{T\to S}, D_S, X_S, X_T) \\
&\quad + \mathcal{L}_{\text{cyc}}(G_{S\to T}, G_{T\to S}, X_S, X_T).
\end{aligned} \tag{3.14}$$

This ultimately corresponds to solving for a target model f_T according to the optimization problem

$$f_T^* = \arg\min_{f_T} \min_{\substack{G_{S\to T}\\ G_{T\to S}}} \max_{D_S, D_T} \mathcal{L}_{\text{CyCADA}}(f_T, X_S, X_T, Y_S, G_{S\to T}, G_{T\to S}, D_S, D_T). \tag{3.15}$$

We have introduced a method for unsupervised adaptation which generalizes adversarial objectives to be viewed as operating at the pixel. In addition, we introduce the use of cycle-consistency together with semantic transformation constraints to guide the mapping from one domain to another. In this work, we apply CyCADA to semantic segmentation. We implement G as a pixel-to-pixel convnet, f as a convnet classifier or a Fully-Convolutional Net (FCN) and D as a convnet with binary outputs.

3.4.3 Experiments

We evaluate CyCADA on the task of semantic image segmentation, using the SYNTHIA [217], GTA [218] and CityScapes [219] datasets.

The task is to assign a semantic label to each pixel in the input image, e.g. *road*, *building*, etc. We limit our evaluation to the unsupervised adaptation setting, where labels are only available in the source domain, but we are evaluated solely on our performance in the target domain.

For each experiment, we use three metrics to evaluate performance. Let n_{ij} be the number of pixels of class i predicted as class j, let $t_i = \sum_j n_{ij}$ be the total number of pixels of class i, and let N be the number of classes. Our three evaluation metrics are, mean intersection-over-union (mIoU), frequency weighted intersection-over-union (fwIoU), and pixel accuracy, which are defined as follows:
$\text{mIoU} = \frac{1}{N} \cdot \frac{\sum_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$, $\text{fwIoU} = \frac{1}{\sum_k t_k} \cdot \frac{\sum_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$, pixel acc. $= \frac{\sum_i n_{ii}}{\sum_i t_i}$.

Cycle-consistent adversarial adaptation is general and can be applied at any layer of a network. Since optimizing the full CyCADA objective in Equation 3.14 end-to-end is memory-intensive in practice, we train our model in stages. First, we perform image-space adaptation and map our source data into the target domain. Next, using the adapted source data with the original source labels, we learn a task model that is suited to operating on target data. We do not use the semantic loss for the segmentation experiments as it would require loading generators, discriminators, and an additional semantic segmenter into memory all at once for two images. We did not have the required memory for this at the time of submission, but leave it to future work to deploy model parallelism or experiment with larger GPU memory.

For our first evaluation, we consider the SYNTHIA dataset [217], which contains synthetic renderings of urban scenes. We use the SYNTHIA video sequences, which are rendered across a variety of environments, weather conditions, and lighting conditions. This provides a synthetic testbed for evaluating adaptation techniques. For comparison with previous work, in this work we focus on adaptation between seasons. We use only the front-facing views in the sequences so as to mimic dashcam imagery, and adapt from fall to winter. The subset of the dataset we use contains 13 classes and consists of 10,852 fall images and 7,654 winter images.

To further demonstrate our method's applicability to real-world adaptation scenarios, we also evaluate our model in a challenging synthetic-to-real adaptation setting. For our synthetic source domain, we use the GTA5 dataset [218] extracted from the game Grand Theft Auto V, which contains 24966 images. We consider adaptation from GTA5 to the real-world Cityscapes dataset [219], from which we used 19998 images without annotation for training and 500 images for validation. Both of these datasets are evaluated on the same set of 19 classes, allowing for straightforward adaptation between the two domains.

Image-space adaptation also affords us the ability to visually inspect the results of the adaptation method. This is a distinct advantage over opaque feature-space adaptation methods, especially in truly unsupervised settings—without labels, there is no way to empirically evaluate the adapted model, and thus no way to verify that adaptation is improving task performance. Visually confirming that the conversions between source and target images are reasonable, while not a *guarantee* of improved task performance, can serve as a sanity check to ensure that adaptation is not completely diverging. This process is diagrammed in Figure 3.33.

Cross-season adaptation

We start by exploring the abilities of pixel space adaptation alone (using FCN8s architecture) for the setting of adapting across seasons in synthetic data. For this

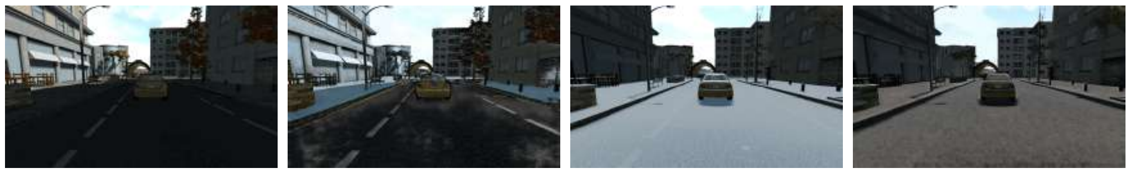


Figure 3.34: **Cross Season Image Translation.** Example image-space conversions for the SYNTHIA seasons adaptation setting. We show real samples from each domain (Fall and Winter) alongside conversions to the opposite domain.

SYNTHIA Fall → Winter																
	sky	building	road	sidewalk	fence	vegetation	pole	car	traffic sign	pedestrian	bicycle	lanemarking	traffic light	**mIoU**	**fwIoU**	**Pixel acc.**
Source only	91.7	80.6	79.7	12.1	71.8	44.2	26.1	42.8	49.0	38.7	45.1	41.3	24.5	49.8	71.7	82.3
FCNs in the wild	92.1	86.7	91.3	20.8	72.7	**52.9**	**46.5**	64.3	**50.0**	**59.5**	**54.6**	**57.5**	**26.1**	59.6	—	—
CyCADA pixel-only	**92.5**	**90.1**	**91.9**	**79.9**	**85.7**	47.1	36.9	**82.6**	45.0	49.1	46.2	54.6	21.5	**63.3**	**85.7**	**92.1**
Oracle (Train on target)	93.8	92.2	94.7	90.7	90.2	64.4	38.1	88.5	55.4	51.0	52.0	68.9	37.3	70.5	89.9	94.5

Table 3.7: Adaptation between seasons in the SYNTHIA dataset. We report IoU for each class and mean IoU, freq-weighted IoU and pixel accuracy. Our CyCADA method achieves state-of-the-art performance on average across all categories. *FCNs in the wild is by [211].

we use the SYNTHIA dataset and adapt from fall to winter weather conditions. Typically in unsupervised adaptation settings it is difficult to interpret what causes the performance improvement after adaptation. Therefore, we use this setting as an example where we may directly visualize the shift from fall to winter and inspect the intermediate pixel level adaptation result from our algorithm. In Figure 3.34 we show the result of pixel only adaptation as we generate a winter domain image (b) from a fall domain image (a), and visa versa (c-d). We may clearly see the changes of adding or removing snow. This visually interpretable result matches our expectation of the true shift between these domains and indeed results in favorable final semantic segmentation performance from fall to winter as shown in Table 3.7. We find that CyCADA achieves state-of-the-art performance on this task with image space adaptation alone, however does not recover full supervised learning performance (train on target). Some example errors includes adding snow to the sidewalks, but not to the road, while in the true winter domain snow appears in both locations. However, even this mistake is interesting as it implies that the model is learning to distinguish road from sidewalk during pixel adaptation, despite the lack of pixel annotations.

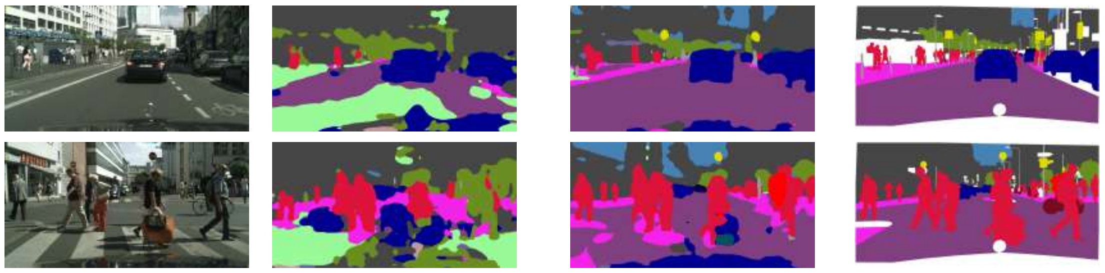

(a) Test Image (b) Source Prediction (c) CyCADA Prediction (d) Ground Truth

Figure 3.35: **GTA5 to CityScapes Semantic Segmentation.** Each test CityScapes image (a) along with the corresponding predictions from the source only model (b) and our CyCADA model (c) are shown and may be compared against the ground truth annotation (d).

Cycle-consistent adversarial adaptation achieves state-of-the-art adaptation performance. We see that under the fwIoU and pixel accuracy metrics, CyCADA approaches oracle performance, falling short by only a few points, despite being entirely unsupervised. This indicates that CyCADA is extremely effective at correcting the most common classes in the dataset. This conclusion is supported by inspection of the individual classes in Table 3.7, where we see the largest improvement on common classes such as *road* and *sidewalk*.

GTA5 → Cityscapes																						
	Architecture	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorbike	bicycle	**mIoU**	**fwIoU**	**Pixel acc.**
Source only	A	26.0	14.9	65.1	5.5	12.9	8.9	6.0	2.5	70.0	2.9	47.0	24.5	0.0	40.0	12.1	1.5	0.0	0.0	0.0	17.9	41.9	54.0
FCNs [211]	A	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1	—	—
CyCADA	A	**83.5**	**38.3**	**76.4**	**20.6**	**16.5**	**22.2**	**26.2**	**21.9**	**80.4**	**28.7**	**65.7**	**49.4**	4.2	**74.6**	**16.0**	**26.6**	**2.0**	**8.0**	0.0	**34.8**	**73.1**	**82.8**
Oracle	A	96.4	74.5	87.1	35.3	37.8	36.4	46.9	60.1	89.0	54.3	89.8	65.6	35.9	89.4	38.6	64.1	38.6	40.5	65.1	60.3	87.6	93.1
Source only	B	42.7	**26.3**	51.7	5.5	6.8	13.8	23.6	6.9	75.5	11.5	36.8	49.3	0.9	46.7	3.4	5.0	0.0	5.0	1.4	21.7	47.4	62.5
CyCADA	B	**63.7**	24.7	**69.3**	**21.2**	**17.0**	**30.3**	**33.0**	**32.0**	**80.5**	**25.3**	**62.3**	**62.0**	**15.1**	**73.1**	**19.8**	**23.6**	**5.5**	**16.2**	**28.7**	**37.0**	**63.8**	**75.4**
Oracle	B	97.3	79.8	88.6	32.5	48.2	56.3	63.6	73.3	89.0	58.9	93.0	78.2	55.2	92.2	45.0	67.3	39.6	49.9	73.6	67.4	89.6	94.3

Table 3.8: Adaptation between GTA5 and Cityscapes, showing IoU for each class and mean IoU, freq-weighted IoU and pixel accuracy. CyCADA significantly outperforms baselines, nearly closing the gap to the target-trained oracle on pixel accuracy. *FCNs in the wild is by [211]. We compare our model using two base semantic segmentation architectures (A) VGG16-FCN8s [220] base network and (B) DRN-26 [221].

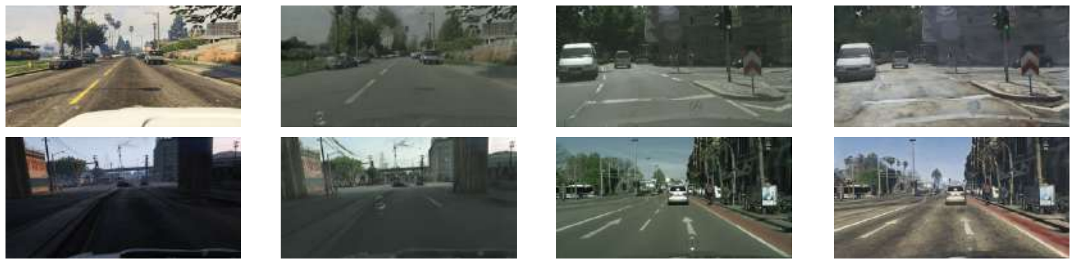

(a) GTA5 (b) GTA5 → Cityscapes (c) CityScapes (d) CityScapes → GTA5

Figure 3.36: **GTA5 to CityScapes Image Translation.** Example images from the GTA5 (a) and Cityscapes (c) datasets, alongside their image-space conversions to the opposite domain, (b) and (d), respectively. Our model achieves highly realistic domain conversions.

Synthetic to real adaptation

To evaluate our method's applicability to real-world adaptation settings, we investigate adaptation from synthetic to real-world imagery. The results of this evaluation are presented in Table 3.8 with qualitative results shown in Figure 3.35. Once again, CyCADA achieves state-of-the-art results, recovering approximately 40% of the performance lost to domain shift. CyCADA also improves or maintains performance on all 19 classes. Examination of fwIoU and pixel accuracy as well as individual class IoUs reveals that our method performs well on most of the common classes. Although some classes such as *train* and *bicycle* see little or no improvement, we note that those classes are poorly represented in the GTA5 data, making recognition very difficult. We compare our model against [206] for this setting, but found this approach did not converge and resulted in worse performance than the source only model.

We visualize the results of image-space adaptation between GTA5 and Cityscapes in Figure 3.36. The most obvious difference between the original images and the adapted images is the saturation levels—the GTA5 imagery is much more vivid than the Cityscapes imagery, so adaptation adjusts the colors to compensate. We also observe texture changes, which are perhaps most apparent in the road: in-game, the roads appear rough with many blemishes, but Cityscapes roads tend to be fairly uniform in appearance, so in converting from GTA5 to Cityscapes, our model removes most of the texture. Somewhat amusingly, our model has a tendency to add a hood ornament to the bottom of the image, which, while likely irrelevant to the segmentation task, serves as a further indication that image-space adaptation is producing reasonable results.

3.5 Conclusion

We explored how an image of one domain can be transformed into a another domain, by utilizing the GAN framework [12] to ensure the output looks indistinguishable to members of the target domain, while constraining the mapping function via the pixel-based cycle consistency loss (Section 3.2) or the patch-based contrastive loss (Section 3.3). The image translation model can be trained on many interesting domains and applications, including turning paintings into photographs, photographs into style of an artist, one object class to another, or computer graphics to realistic images, all maintaining the overall structure of the input. Furthermore, we presented a cycle-consistent adversarial domain adaptation method (Section 3.4) that utilizes cycle-consistency loss at pixel level to enhance perception modules trained in synthetic environment.

Chapter 4

Learning Structure and Texture without Per-Image Labels

The methods proposed in the previous chapters still rely on either per-pixel (Chapter 2) or per-image (Chapter 3) supervision. In this chapter, we explore if similarly disentangled representation can be discovered without any labels. In this regard, we propose the Swapping Autoencoder, a deep model designed for image manipulation that can be trained on a collection of unlabeled images. The key idea is to encode an image into two independent components and enforce that any swapped combination maps to a realistic image. In particular, we encourage the components to represent structure and texture, by enforcing one component to encode co-occurrent patch statistics across different parts of the image. As our method is trained with an encoder, finding the latent codes for a new input image becomes trivial, rather than cumbersome. As a result, our method enables us to manipulate real input images in various ways, including texture swapping, local and global editing, and latent code vector arithmetic. Experiments on multiple datasets show that our model produces better results and is substantially more efficient compared to recent generative models. [1]

4.1 Introduction

Traditional photo-editing tools, such as Photoshop, operate solely within the confines of the input image, i.e. they can only "recycle" the pixels that are already there. The promise of using machine learning for image manipulation has been

[1]This work was first published as *Swapping Autoencoder for Deep Image Manipulation* at NeurIPS, 2020 [1].

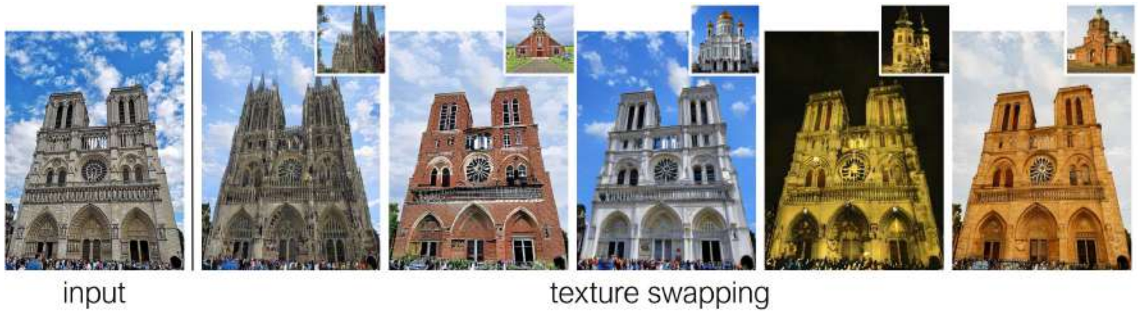


Figure 4.1: Our Swapping Autoencoder learns to disentangle texture from structure for image editing tasks. One such task is texture swapping, shown here. Please see our project webpage for a demo video of our editing method.

to incorporate the *generic visual knowledge* drawn from external visual datasets into the editing process. The aim is to enable new class of editing operations, such as inpainting large image regions [83, 222, 223], synthesizing photorealistic images from layouts [14, 20, 22], replacing objects [41, 224], or changing the time photo is taken [46, 225].

However, learning-driven image manipulation brings in its own challenges. For image editing, there is a fundamental conflict: what information should be gleaned from the dataset versus information that must be retained from the input image? If the output image relies too much on the dataset, it will retain no resemblance to the input, so can hardly be called "editing", whereas relying too much on the input lessens the value of the dataset. This conflict can be viewed as a disentanglement problem. Starting from image pixels, one needs to factor out the visual information which is specific to a given image from information that is applicable across different images of the dataset. Indeed, many existing works on learning-based image manipulation, though not always explicitly framed as learning disentanglement, end up doing so, using paired supervision [14, 20, 22, 226], domain supervision [41, 44, 109, 225], or inductive bias of the model architecture [227, 228].

In our work, we aim to discover a disentanglement suitable for image editing in an *unsupervised setting*. We argue that it is natural to explicitly factor out the visual patterns within the image that must change consistently with respect to each other. We operationalize this by learning an autoencoder with two modular latent codes, one to capture the *within-image visual patterns*, and another to capture the rest of the information. We enforce that any arbitrary combination of these codes map to a realistic image. To disentangle these two factors, we ensure that all image patches with the same within-image code appear coherent with each other. Interestingly, this coincides with the classic definition of visual texture in a line of works started by Julesz [229–235]. The second code captures the remaining information, coinciding with structure. As such, we refer to the two codes as *texture* and *structure* codes.

A natural question to ask is: why not simply use unconditional GANs [12] that have been shown to disentangle style and content in unsupervised settings [13, 177, 228]? The short answer is that these methods do not work well for editing *existing* images. Unconditional GANs learn a mapping from an easy-to-sample (typically Gaussian) distribution. Some methods [177, 227, 236] have been suggested to *retrofit* pre-trained unconditional GAN models to find the latent vector that reproduces the input image, but we show that these methods are inaccurate and magnitudes slower than our method. The conditional GAN models [14, 20, 41, 44] address this problem by starting with input images, but they require the task to be defined *a priori*. In contrast, our model learns an embedding space that is useful for image manipulation in several downstream tasks, including synthesizing new image hybrids (see Figure 4.1), smooth manipulation of attributes or domain transfer by traversing latent directions (Figure 4.7), and local manipulation of the scene structure (Figure 4.8).

To show the effectiveness of our method, we evaluate it on multiple datasets, such as LSUN churches and bedrooms [237], FlickrFaces-HQ [13], and newly collected datasets of mountains and waterfalls, using both automatic metrics and human perceptual judgments. We also present an interactive UI (please see our video in the project webpage) that showcases the advantages of our method.

4.2 Related Work

Conditional generative models , such as image-to-image translation [20, 41], learn to directly synthesize an output image given a user input. Many applications have been successfully built with this framework, including image inpainting [83, 223, 238, 239], photo colorization [139, 240–242], texture and geometry synthesis [243–245], sketch2photo [88], semantic image synthesis and editing [14, 21, 22, 246]. Recent methods extent it to multi-domain and multi-modal setting [43, 44, 109, 247, 248]. However, it is challenging to apply such methods to on-the-fly image manipulation, because for each new application and new user input, a new model needs to be trained. We present a framework for both image synthesis and manipulation, in which the task can be defined by one or a small number of examples at run-time. While recent works [184, 185] propose to learn a single-image GANs for image editing, our model can be quickly applied to a test image without extensive computation of single-image training.

Deep image editing via latent space exploration modifies the latent vector of a pre-trained, unconditional generative model (e.g., a GAN [12]) according to the desired user edits. For example, iGAN [80] obtains the latent code using an encoder-based initialization followed by Quasi-Newton optimization, and updates the code according to new user constraints. Similar ideas have been explored in other tasks like image inpainting, face editing, and deblurring [249–252]. More recently, instead of using the input latent space, GANPaint [236] adapts layers of a pre-trained GAN for each input image and updates layers according to a user's semantic control [253]. Image2StyleGAN [227] and StyleGAN2 [177] reconstruct the image using an extended embedding space and noise vectors. Our work differs in that we allow the code space to be learned rather than sampled from a fixed distribution, thus making it much more flexible. In addition, we train an encoder together with the generator, which allows for significantly faster reconstruction.

Disentanglement of content and style generative models. Deep generative models learn to model the data distribution of natural images [12, 32, 254–257], many of which aim to represent content and style as independently controllable factors [13, 177, 258–260]. Of special relevance to our work are models that use code swapping during training [13, 82, 259, 261–263]. Our work differs from them in three aspects. First, while most require human supervision, such as class labels [82], pairwise image similarity [262], images pairs with same appearances [259], or object locations [263], our method is fully unsupervised. Second, our decomposable structure and texture codes allow each factor be extracted from the input images to control different aspects of the image, and produce higher-quality results when mixed. Note that for our application, image quality and flexible control are critically important, as we focus on image manipulation rather than unsupervised feature learning. Recent image-to-image translation methods also use code swapping but require ground truth domain labels [108, 264, 265]. In concurrent work, Anokhin et al. [225] and ALAE [266] propose models very close to our code swapping scheme for image editing purposes.

Style transfer. Modeling style and content is a classic computer vision and graphics problem [4,226]. Several recent works revisited the topic using modern neural networks [115–117, 267], by measuring content using perceptual distance [115, 127], and style as global texture statistics, e.g., a Gram matrix. These methods can transfer low-level styles such as brush strokes, but often fail to capture larger scale semantic structures. Photorealistic style transfer methods further constrain the result to be represented by local affine color transforms from the input image [186, 187, 268], but such methods only allow local color changes. In contrast, our learned decomposition can transfer semantically meaningful structure, such as the architectural details of a church, as well as perform other image editing operations.

4.3 Method

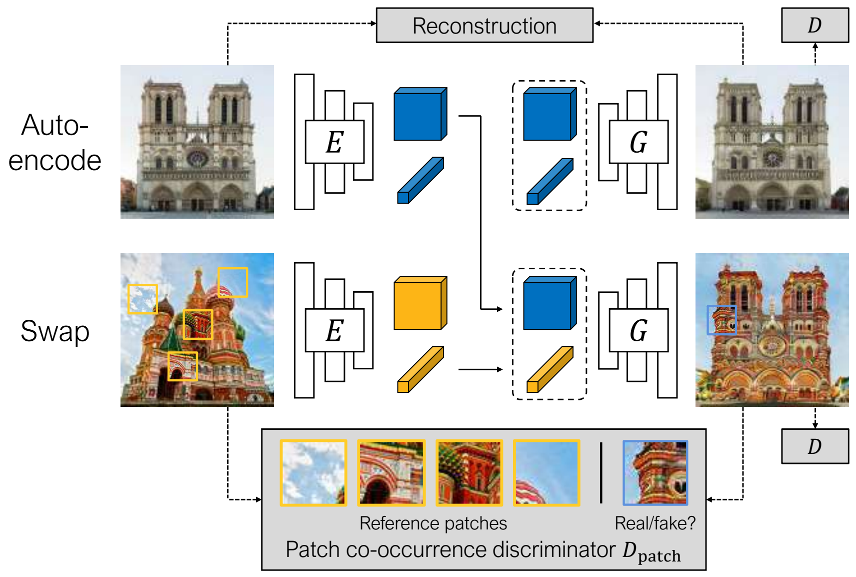


Figure 4.2: **Swapping Autoencoder** consists of autoencoding (*top*) and swapping (*bottom*) operation. **(Top)** An encoder E embeds an input (Notre-Dame) into two codes. The structure code (□) is a tensor with spatial dimensions; the texture code (▱) is a 2048-dimensional vector. Decoding with generator G should produce a realistic image (enforced by discriminator D) matching the input (reconstruction loss). **(Bottom)** Decoding with the texture code from a second image (Saint Basil's Cathedral) should look realistic (via D) and match the texture of the image, by training with a patch co-occurrence discriminator D_{patch} that enforces the output and reference patches look indistinguishable.

What is the desired representation for image editing? We argue that such representation should be able to reconstruct the input image easily and precisely. Each code in the representation can be independently modified such that the resulting image both looks realistic and reflects the unmodified codes. The representation should also support both global and local image editing.

To achieve the above goals, we train a swapping autoencoder (shown in Figure 4.2) consisting of an encoder E and a generator G, with the core objectives of 1) accurately reconstructing an image, 2) learning independent components that can be mixed to create a new hybrid image, and 3) disentangling texture from structure by using a patch discriminator that learns co-occurrence statistics of image patches.

4.3.1 Accurate and realistic reconstruction

In a classic autoencoder [135], the encoder E and generator G form a mapping between image $\boldsymbol{x} \sim \mathcal{X} \subset \mathbb{R}^{H\times W\times 3}$ and latent code $\boldsymbol{z} \sim \mathcal{Z}$. As seen in the top branch of Figure 4.2, our autoencoder also follows this framework, using an image reconstruction loss:

$$\mathcal{L}_{\text{rec}}(E, G) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{X}} \|\boldsymbol{x} - G(E(\boldsymbol{x}))\|_1. \tag{4.1}$$

In addition, we wish for the image to be realistic, enforced by a discriminator D. The non-saturating adversarial loss [12] for the generator G and encoder E is calculated as:

$$\mathcal{L}_{\text{GAN,rec}}(E, G, D) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{X}} - \log\left(D(G(E(\boldsymbol{x})))\right). \tag{4.2}$$

4.3.2 Decomposable latent codes

We divide the latent space $\mathcal{Z}$ into two components, $\boldsymbol{z} = (\boldsymbol{z}_s, \boldsymbol{z}_t)$, and enforce that swapping components with those from other images still produces realistic images, using the GAN loss [12].

$$\mathcal{L}_{\text{GAN,swap}}(E, G, D) = \mathbb{E}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X},\boldsymbol{x}^1\neq\boldsymbol{x}^2} - \log\left(D(G(\boldsymbol{z}_s^1, \boldsymbol{z}_t^2))\right), \tag{4.3}$$

where $\boldsymbol{z}_s^1$, $\boldsymbol{z}_t^2$ are the first and second components of $E(\boldsymbol{x}^1)$, $E(\boldsymbol{x}^2)$, respectively. Furthermore, as shown in Figure 4.2, we design the shapes of $\boldsymbol{z}_s$ and $\boldsymbol{z}_t$ asymmetrically such that $\boldsymbol{z}_s$ is a tensor with spatial dimensions, while $\boldsymbol{z}_t$ is a vector. In our model, $\boldsymbol{z}_s$ and $\boldsymbol{z}_t$ are intended to encode structure and texture information, and hence named *structure* and *texture* code, respectively, for convenience. At each training iteration, we randomly sample two images $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, and enforce $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{GAN,rec}}$ on $\boldsymbol{x}^1$, and $\mathcal{L}_{\text{GAN,swap}}$ on the hybrid image of $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$.

A majority of recent deep generative models [13, 177, 255, 256, 269–271], such as in GANs [12] and VAEs [32], attempt to make the latent space Gaussian to enable random sampling. In contrast, we do not enforce such constraint on the latent space of our model. Our swapping constraint focuses on making the "distribution" around a *specific* input image and its plausible variations well-modeled.

Under ideal convergence, the training of the Swapping Autoencoder encourages several desirable properties of the learned embedding space $\mathcal{Z}$. First, the encoding function E is optimized toward injection, due to the reconstruction loss, in that different images are mapped to different latent codes. Also, our design choices encourage that different codes produce different outputs via G: the texture code must capture the texture distribution, while the structure code must capture location-specific information of the input images (see Section 4.4.7 for more details). Lastly, the joint distribution of the two codes of the swap-generated images is factored by construction, since the structure codes are combined with random texture codes.

4.3.3 Co-occurrent patch statistics

While the constraints above are sufficient for our swapping autoencoder to learn a factored representation, the resulting representation will not necessarily be intuitive for image editing, with no guarantee that $\boldsymbol{z}_s$ and $\boldsymbol{z}_t$ actually represent structure and texture. To address this, we encourage the texture code $\boldsymbol{z}_t$ to maintain the same texture in any swap-generated images. We introduce a patch co-occurrence discriminator D_{patch}, as shown in the bottom of Figure 4.2. The generator aims to generate a hybrid image $G(\boldsymbol{z}_s^1, \boldsymbol{z}_t^2)$, such that any patch from the hybrid cannot be distinguished from a group of patches from input $\boldsymbol{x}^2$.

$$\mathcal{L}_{\text{CooccurGAN}}(E, G, D_{\text{patch}}) = \mathbb{E}_{\boldsymbol{x}^1, \boldsymbol{x}^2 \sim \mathcal{X}} - \log\left(D_{\text{patch}}\left(\texttt{crop}(G(\boldsymbol{z}_s^1, \boldsymbol{z}_t^2)), \texttt{crops}(\boldsymbol{x}^2)\right)\right), \tag{4.4}$$

where `crop` selects a random patch of size 1/8 to 1/4 of the full image dimension on each side (and `crops` is a collection of multiple patches). Our formulation is inspired by Julesz's theory of texture perception [229, 230] (long used in texture synthesis [232, 234]), which hypothesizes that images with similar marginal and joint feature statistics appear perceptually similar. Our co-occurence discriminator serves to enforce that the joint statistics of a learned representation be consistently transferred. Similar ideas for modeling co-occurences have been used for propagating a single texture in a supervised setting [245], self-supervised representation learning [272], and identifying image composites [273].

4.3.4 Overall training and architecture

Our final objective function for the encoder and generator is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rec}} + 0.5\mathcal{L}_{\text{GAN,rec}} + 0.5\mathcal{L}_{\text{GAN,swap}} + \mathcal{L}_{\text{CooccurGAN}}$. The discriminator objective and design follows StyleGAN2 [177]. The co-occurrence patch discriminator first extracts features for each patch, and then concatenates them to pass to the final classification layer. The encoder consists of 4 downsampling ResNet [60] blocks to produce the tensor $\boldsymbol{z}_s$, and a dense layer after average pooling to produce the vector $\boldsymbol{z}_t$. As a consequence, the structure code $\boldsymbol{z}_s$, is limited by its receptive field at each location, providing an inductive bias for capturing local information. On the other hand, the texture code $\boldsymbol{z}_t$, deprived of spatial information by the average pooling, can only process aggregated feature distributions, forming a bias for controlling global style. The generator is based on StyleGAN2, with weights modulated by the texture code. Please see Section 4.4.7 for a detailed specification of the architecture, as well as details of the discriminator loss function.

4.4 Experiments

The proposed method can be used to efficiently embed a given image into a factored latent space, and to generate hybrid images by swapping latent codes. We show that the disentanglement of latent codes into the classic concepts of "style" and "content" is competitive even with style transfer methods that address this specific task [187, 188], while producing more photorealistic results. Furthermore, we observe that even without an explicit objective to encourage it, vector arithmetic in the learned embedding space $\mathcal{Z}$ leads to consistent and plausible image manipulations [13,274,275]. This opens up a powerful set of operations, such as attribute editing, image translation, and interactive image editing, which we explore.

We first describe our experimental setup. We then evaluate our method on: (1) quickly and accurately embedding a test image, (2) producing realistic hybrid images with a factored latent code that corresponds to the concepts of texture and structure, and (3) editability and usefulness of the latent space. We evaluate each aspect separately, with appropriate comparisons to existing methods.

4.4.1 Experimental setup

Datasets.

For existing datasets, our model is trained on LSUN Churches, Bedrooms [237], Animal Faces HQ (AFHQ) [248], Flickr Faces HQ (FFHQ) [13], all at resolution of 256`px` except FFHQ at 1024`px`. In addition, we introduce new datasets, which are Portrait2FFHQ, a combined dataset of 17k portrait paintings from `wikiart.org` and FFHQ at 256`px`, Flickr Mountain, 0.5M mountain images from `flickr.com`, and Waterfall, of 90k 256`px` waterfall images. Flickr Mountain is trained at 512`px` resolution, but the model can handle larger image sizes (e.g., 1920×1080) due to the fully convolutional architecture. We describe the datasets in details.

LSUN Church [237] consists of 126,227 images of outdoor churches. The images are in the dataset are 256`px` on the short side. During training, 256×256 cropped images are used. A separate validation set of 300 images is used for comparisons against baselines.

LSUN Bedroom [237] consists of 3,033,042 images of indoor bedrooms. Like LSUN Church, the images are trained at 256×256 resolution. The results are shown with the validation set.

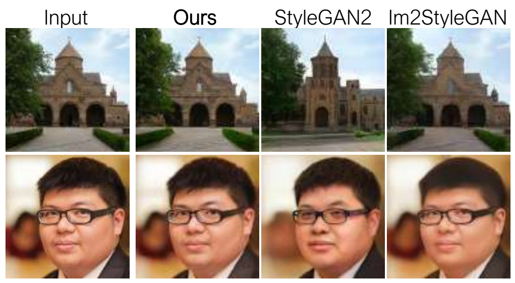
Flickr Faces HQ [13] consists of 70,000 high resolution aligned face images from `flickr.com`. Our model is initially trained at 512×512 resolution, and finetuned at 1024 resolution. The dataset designated 10,000 images for validation, but we train our model on the entire 70,000 images, following the practice of StyleGAN [13] and StyleGAN2 [177]. For evaluation, we used randomly selected 200 images from the validation set, although the models are trained with these images.

Animal Faces HQ [248] contains a total of 15,000 images equally split between cats, dogs, and a wildlife category. Our method is trained at 256×256 resolution on the combined dataset without domain labels. The results are shown with a separate validation set.

Portrait2FFHQ consists of FFHQ [13] and a newly collected 19,863 portrait painting images from `wikiart.org`. The model is trained at 512×512 resolution on the combined dataset. The results of the paper are generated from separately collected sample paintings. We did not check if the same painting belongs in the training set. The test photographs are from CelebA [276]. All images are aligned to match the facial landmarks of FFHQ dataset.

Flickr Waterfall is a newly collected dataset of 90,345 waterfall images. The images are downloaded from the user group "Waterfalls around the world" on `flickr.com`. The validation set is 399 images collected from the user group "*Waterfalls*". Our model is trained at 256×256 resolution.

Flickr Mountains is a newly collected dataset of 517,980 mountain images from Flickr. The images are downloaded from the user group "Mountains Anywhere" on `flickr.com`. For testing, separately downloaded sample images were used. Our model is trained at 512×512 resolution.



Method	Runtime (sec) (↓)	LPIPS Reconstruction (↓)			
		Church	FFHQ	Waterfall	Average
Ours	**0.101**	0.227	**0.074**	**0.238**	**0.180**
Im2StyleGAN	495	**0.186**	0.174	0.281	0.214
StyleGAN2	96	0.377	0.215	0.384	0.325

Figure 4.3: **Embedding examples and reconstruction quality**. We project images into embedding spaces for our method and baseline GAN models, Im2StyleGAN [13, 227] and StyleGAN2 [177]. Our reconstructions better preserve the detailed outline (e.g., doorway, eye gaze) than StyleGAN2, and appear crisper than Im2StyleGAN. This is verified on average with the LPIPS metric [130]. Our method also reconstructs images much faster than recent generative models that use iterative optimization.

Baselines.

To use a GAN model for downstream image editing, one must embed the image into its latent space [80]. We compare our approach to two recent solutions. Im2StyleGAN [227] present a method for embedding into StyleGAN [13], using iterative optimization into the "W^+-space" of the model. The StyleGAN2 model [177] also includes an optimization-based method to embed into its latent space and noise vectors. One application of this embedding is producing hybrids. StyleGAN and StyleGAN2 present an emergent hierarchical parameter space that allows hybrids to be produced by mixing parameters of two images. We additionally compare to image stylization methods, which aim to mix the "style" of one image with the "content" from another. STROTSS [188] is an optimization-based framework, in the spirit of the classic method of Gatys et al. [115]. We also compare to WCT^2 [187], a recent state-of-the-art *photorealistic* style transfer method based on a feedforward network.

4.4.2 Image embedding

The first step of manipulating an image with a generative model is projecting it into its latent spade. If the input image cannot be projected with high fidelity, the embedded vector cannot be used for editing, as the user would be editing a different image. Figure 4.3 illustrates both example reconstructions and quantitative measurement of reconstruction quality, using LPIPS [130] between the original and embedded images. Note that our method accurately preserves the doorway pattern (top) and facial features (bottom) without blurriness. Averaged across datasets and on 5 of the 6 comparisons to the baselines, our method achieves *better*

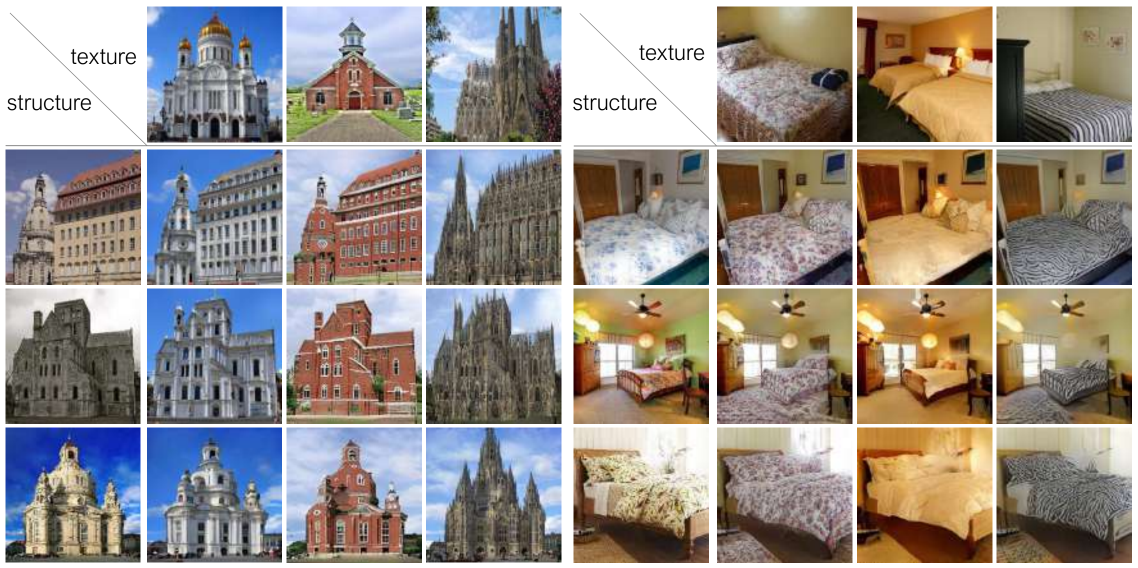


Figure 4.4: **Image swapping**. Each row shows the result of combining the structure code of the leftmost image with the texture code of the top image (trained on LSUN Church and Bedroom). Our model generates realistic images that preserve texture (e.g., material of the building, or the bedsheet pattern) and structure (outline of objects).

reconstruction quality than the baselines. An exception is on the Church dataset, where Im2StyleGAN obtains a better reconstruction score. Importantly, as our method is designed with test-time embedding in mind, it only requires a single feedforward pass, at least 1000× faster than the baselines that require hundreds to thousands of optimization steps. Next, we investigate how *useful* the embedding is by exploring manipulations with the resulting code.

4.4.3 Swapping to produce image hybrids

In Figure 4.4, we show example hybrid images with our method, produced by combining structure and texture codes from different images. Note that the textures of the top row of images are consistently transferred; the sky, facade, and window patterns are mapped to the appropriate regions on the structure images on the churches, and similarly for the bedsheets.

Realism of image hybrids. In Table 4.1, we show results of comparison to existing methods. As well as generative modeling methods [13, 177, 227]. For image hybrids, we additionally compare with SOTA style transfer methods [187, 188], although they are not directly applicable for controllable editing by embedding

Method	Runtime (sec) (↓)	Human Perceptual Study (AMT Fooling Rate) (↑)			
		Church	FFHQ	Waterfall	**Average**
Swap Autoencoder (Ours)	**0.113**	**31.3±2.4**	**19.4±2.0**	**41.8±2.2**	**31.0±1.4**
Im2StyleGAN [13, 227]	990	8.5±2.1	3.9±1.1	12.8±2.4	8.4±1.2
StyleGAN2 [177]	192	24.3±2.2	13.8±1.8	35.3±2.4	24.4±1.4
STROTSS [188]	166	13.7±2.2	3.5±1.1	23.0±2.1	13.5±1.2
WCT^2 [187]	1.35	***27.9±2.3***	**22.3±2.0**	35.8±2.4	***28.6±1.3***

Table 4.1: **Realism of swap-generated images** We study how realistic our swap-generated swapped appear, compared to state-of-the-art generative modeling approaches (Im2StyleGAN and StyleGAN2) and stylization methods (STROTSS and WCT^2). We run a perceptual study, where each method/dataset is evaluated with 1000 human judgments. We **bold** the best result per column and ***bold+italicize*** methods that are within the statistical significance of the top method. Our method achieves the highest score across all datasets. Note that WCT^2 is a method tailored especially for photorealistic style transfer and is within the statistical significance of our method in the perceptual study. Runtime is reported for 1024×1024 resolution.

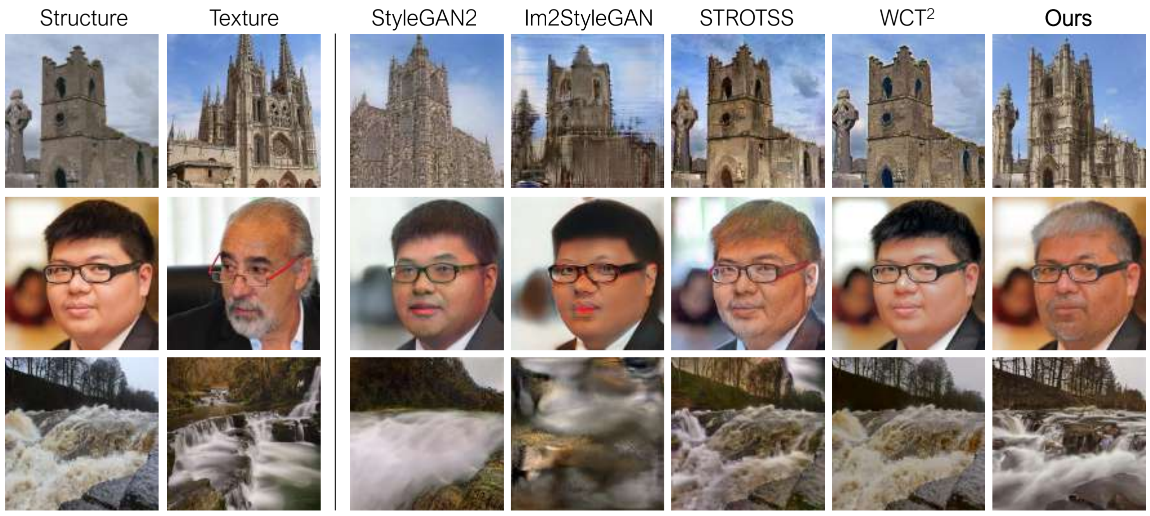


Figure 4.5: **Comparison of image hybrids.** Our approach generates realistic results that combine scene structure with elements of global texture, such as the shape of the towers (church), the hair color (portrait), and the long exposure (waterfall).

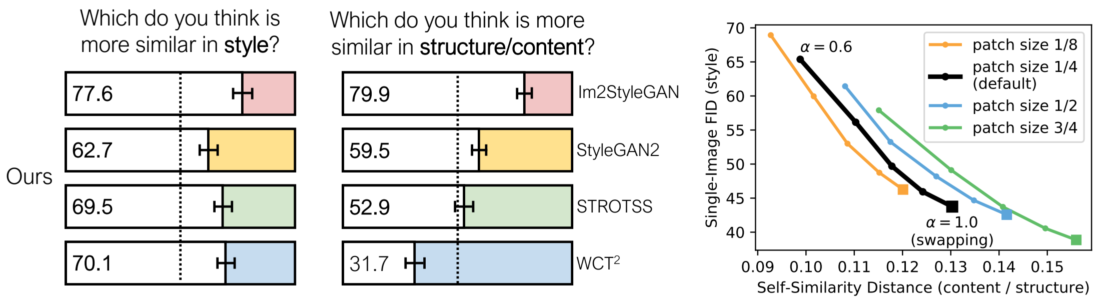


Figure 4.6: **Style and content**. **(Left)** Results of our perceptual study where we asked users on AMT to choose which image better reflects the "style" or "content" of a provided reference image, given two results (ours and a baseline). Our model is rated best for capturing style, and second-best for preserving content, behind WCT^2 [187], a photorealistic style transfer method. Most importantly, our method was rated strictly better in both style and content matching than both image synthesis models Im2StyleGAN [13, 227] and StyleGAN2 [177]. **(Right)** Using the self-similarity distance [188] and SIFID [184], we study variations of the co-occurrence discriminator's patch size in training with respect to the image size. As patch size increases, our model tends to make more changes in swapping (closer to the target style and further from input structure). In addition, we gradually interpolate the texture code, with interpolation ratio α, away from a full swapping $\alpha = 1.0$, and observe that the transition is smooth.

images (Chapter 4.4.5). We run a human perceptual study, following the test setup used in [20, 139, 184]. A real and generated image are shown sequentially for one second each to Amazon Mechanical Turkers (AMT), who choose which they believe to be fake. We measure how often they fail to identify the fake. An algorithm generating perfectly plausible images would achieve a fooling rate of 50%. We gather 15,000 judgments, 1000 for each algorithm and dataset. Our method achieves more realistic results across all datasets. The nearest competitor is the WCT^2 [187] method, which is designed for photorealistic style transfer. Averaged across the three datasets, our method achieves the highest fooling rate ($31.0 \pm 1.4\%$), with WCT^2 closely following within the statistical significance ($28.6 \pm 1.3\%$). We show qualitative examples in Figure 4.5.

Style and content. Next, we study how well the concepts of *content* and *style* are reflected in the structure and texture codes, respectively. We employ a Two-alternative Forced Choice (2AFC) user study to quantify the quality of image hybrids in content and style space. We show participants our result and a baseline result, with the style or content reference in between. We then ask a user which image is more similar in style, or content respectively. Such 2AFC tests were used to train the

LPIPS perceptual metric [130], as well as to evaluate style transfer methods in [188]. As no true automatic perceptual function exists, human perceptual judgments remain the "gold standard" for evaluating image synthesis results [20,21,139,184]. Figure 4.6 visualizes the result of 3,750 user judgments over four baselines and three datasets, which reveal that our method outperforms all baseline methods with statistical significance in style preservation. For content preservation, our method is only behind WCT^2, which is a photorealistic stylization method that makes only minor color modifications to the input. Most importantly, our method achieves the best performance with statistical significance in both style and content among models that can embed images, which is required for other forms of image editing.

4.4.4 Analysis of our method

Next we analyze the behavior of our model using automated metrics. Self-similarity Distance [188] measures structural similarity in deep feature space based on the self-similarity map of ImageNet-pretrained network features. Single-Image FID [184] measures style similarity by computing the Fréchet Inception Distance (FID) between two feature distributions, each generated from a single image. SIFID is similar to Gram distance, a popular metric in stylization methods [115,234], but differs by comparing the mean of the feature distribution as well as the covariance.

Specifically, we vary the size of cropped patches for the co-occurrence patch discriminator in training. In Figure 4.6 *(right)*, the max size of random cropping is varied from 1/8 to 3/4 of the image side length, including the default setting of 1/4. We observe that as the co-occurrence discriminator sees larger patches, it enforces stronger constraint, thereby introducing more visual change in both style and content. Moreover, instead of full swapping, we gradually interpolate one texture code to the other. We observe that the SIFID and self-similarity distance both change gradually, in all patch settings. Such gradual visual change can be clearly observed in Figure 4.7, and the metrics confirm this.

4.4.5 Image editing via latent space operations

Even though no explicit constraint was enforced on the latent space, we find that modifications to the latent vectors cause smooth and predictable transformations to the resulting images. This makes such a space amenable to downstream editing in multiple ways. First, we find that our representation allows for controllable image manipulations by **vector arithmetic** in the latent space. Figure 4.7 shows that adding the same vector smoothly transforms different images into a similar style, such as gradually adding more snow *(top)*. Such vectors can be conveniently derived

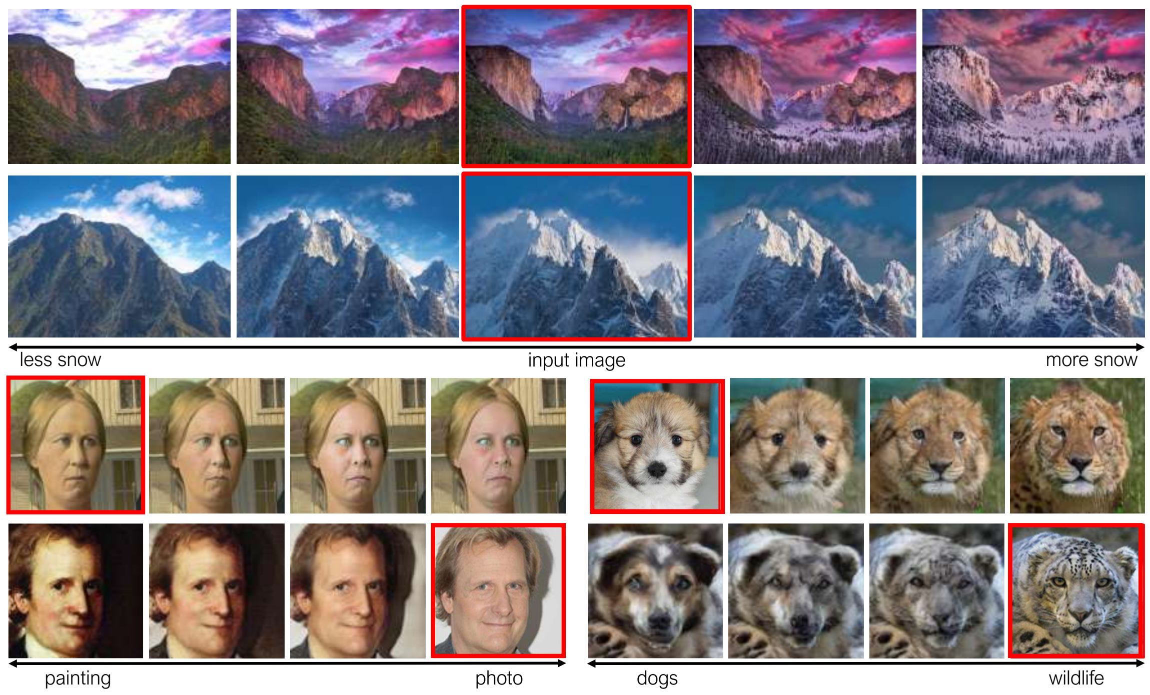


Figure 4.7: **Continuous interpolation**. **(top)** A manipulation vector for *snow* is discovered by taking mean difference between 10 user-collected photos of snowy and summer mountain. The vector is simply added to the texture code of the input image (red) with some gain. **(bottom)** Multi-domain, continuous transformation is achieved by applying the average vector difference between the texture codes of two domains, based on annotations from the training sets. We train on Portrait2FFHQ and AFHQ [248] datasets.

by taking the mean difference between the embeddings of two groups of images.

In a similar mechanism, the learned embedding space can also be used for **image-to-image translation** tasks (Figure 4.7), such as transforming paintings to photos. Image translation is achieved by applying the domain translation vector, computed as the mean difference between the two domains. Compared to most existing image translation methods, our method does not require that all images are labeled, and also allows for multi-domain, fine-grained control simply by modifying the vector magnitude and members of the domain at test time. Finally, the design of the structure code $\boldsymbol{z}_s$ is directly amenable **local editing** operations, due to its spatial nature.

4.4.6 Interactive user interface for image editing

Based on the proposed latent space exploration methods, we built a sample user interface for creative user control over photographs. Figure 4.8 shows three editing modes that our model supports. Please see a demo video on our webpage. We demonstrate three operations: (1) **global style editing**: the texture code can be transformed by adding predefined manipulation vectors that are computed from PCA on the train set. Like GANSpace [228], the user is provided with knobs to adjust the gain for each manipulation vector. (2) **region editing**: the structure code can also be manipulated the same way of using PCA components, by treating each location as individual, controllable vectors. In addition, masks can be automatically provided to the user based on the self-similarity map at the location of interest to control the extent of structural manipulation. (3) **cloning**: the structure code can be directly edited using a brush that replaces the code from another part of the image, like the Clone Stamp tool of Photoshop.

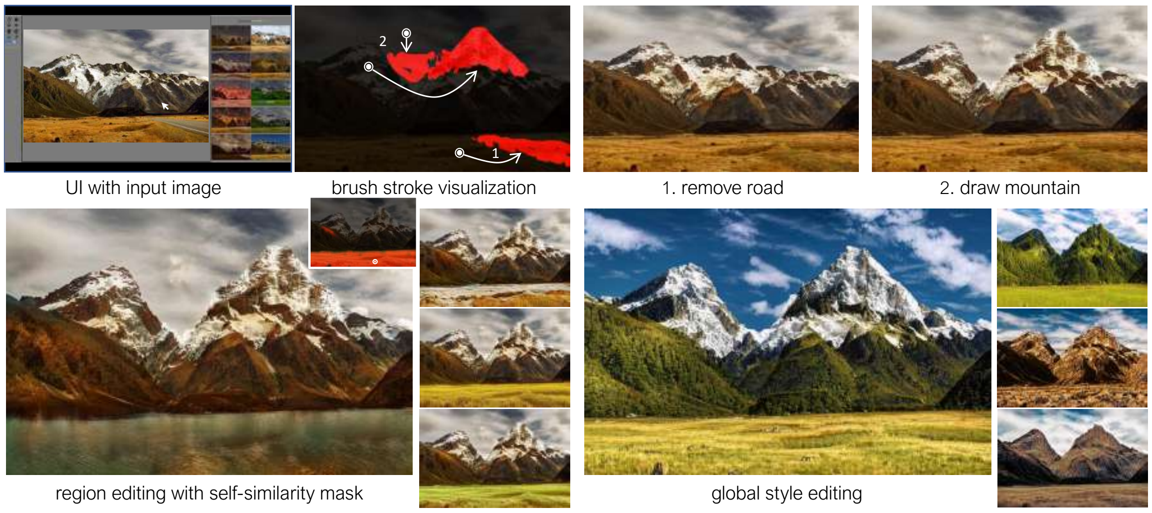


Figure 4.8: **Example Interactive UI. (top, cloning)** using an interactive UI, part of the image is "redrawn" by the user with a brush tool that extracts structure code from user-specified location. **(left, region editing)** the bottom region is transformed to lake, snow, or different vegetation by adding a manipulation vector to the structure codes of the masked region, which is auto-generated from the self-similarity map at the specified location. **(right, global style editing)** the overall texture and style can be changed using vector arithmetic with principal directions of PCA, controlled by the sliders on the right pane of the UI. (best viewed zoomed in)

4.4.7 Architecture

The **encoder** maps the input image to structure and texture codes, as shown in Figure 4.9 (left). For the structure code, the network consists of 4 downsampling residual blocks [60], followed by two convolution layers. For the texture code, the network branches off and adds 2 convolutional layers, followed by an average pooling (to completely remove spatial dimensions) and a dense layer. The asymmetry of the code shapes is designed to impose an inductive bias and encourage decomposition into orthogonal tensor dimensions. Given an $256\times$ 256 image, the structure code is of dimension $16 \times 16 \times 8$ (large spatial dimension), and texture code is of dimension $1 \times 1 \times 2048$ (large channel dimension).

The texture code is designed to be agnostic to positional information by using reflection padding or no padding ("valid") in the convolutional layers (rather than zero padding) followed by average pooling. On the other hand, each location of the structure code has a strong inductive bias to encode information in its neighborhood, due to its fully convolutional architecture and limited receptive field.

The **generator** maps the codes back to an image, as shown in Figure 4.9 (right). The network uses the structure code in the main branch, which consists of 4 residual blocks and 4 upsampling residual blocks. The texture code is injected using the weight modulation/demodulation layer from StyleGAN2 [177]. We generate the output image by applying a convolutional layer at the end of the residual blocks. This is different from the default setting of StyleGAN2, which uses an output skip, but more similar to the residual net setting of StyleGAN2 discriminator. Lastly, to enable isolated local editing, we avoid normalizations such as instance or batch normalization [48, 161].

The **discriminator** architecture is identical to StyleGAN2, except with no minibatch discrimination, to enable easier fine-tuning at higher resolutions with smaller batch sizes.

The **co-occurrence patch discriminator** architecture is shown in Figure 4.10 and is designed to determine if a patch in question ("real/fake patch") is from the same image as a set of reference patches. Each patch is first independently encoded with 5 downsampling residual blocks, 1 residual block, and 1 convolutional layer. The representations for the reference patches are averaged together and concatenated with the representation of the real/fake patch. The classification applies 3 dense layers to output the final prediction.

The detailed design choices of the layers in all the networks follow StyleGAN2 [177], including weight demodulation, antialiased bilinear down/upsampling [277], equalized learning rate, noise injection at every layer, adjusting variance of residual blocks by the division of $\sqrt{2}$, and leaky ReLU with slope 0.2.

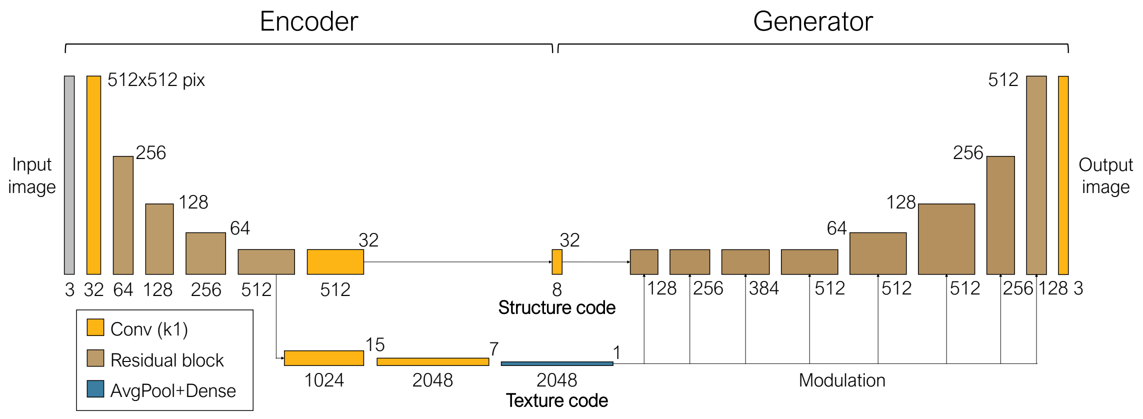


Figure 4.9: **Encoder and generator architecture**. The encoder network first applies 4 downsampling residual blocks [60] to produce an intermediate tensor, which is then passed to two separate branches, producing the structure code and texture code. The structure code is produced by applying 1-by-1 convolutions to the intermediate tensor. The texture code is produced by applying strided convolutions, average pooling, and then a dense layer. Given an $H \times H$ image, the shapes of the two codes are $H/16 \times H/16 \times 8$, and $1 \times 1 \times 2048$, respectively. The case for a 512×512 image is shown. To prevent the texture code from encoding positional information, we apply reflection padding for the residual blocks, and then no padding for the conv blocks. The generator consists of 4 residual blocks and then 4 upsampling residual blocks, followed by 1-by-1 convolution to produce an RGB image. The structure code is given in the beginning of the network, and the texture code is provided at every layer as modulation parameters. We use zero padding for the generator. The detailed architecture follows StyleGAN2 [177], including weight demodulation, bilinear upsampling, equalized learning rate, noise injection at every layer, adjusting variance of residual blocks by the division of $\sqrt{2}$, and leaky ReLU with slope 0.2.

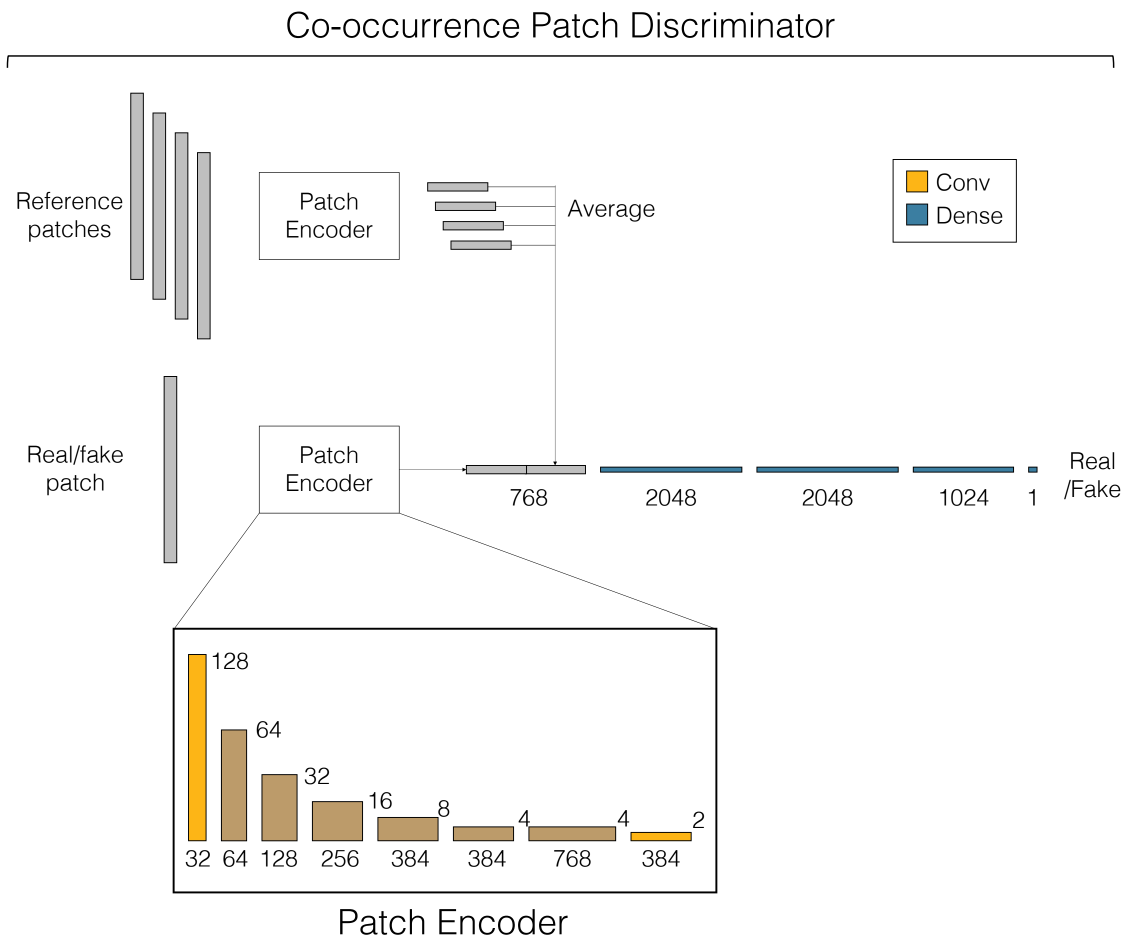


Figure 4.10: **Co-occurrence patch discriminator architecture**. The co-occurrence patch discriminator consists of the feature extractor, which applies 5 downsampling residual blocks, 1 residual block, and 1 convolutional layer with valid padding to each input patch, and the classifier, which concatenates the flattened features in channel dimension and then applies 3 dense layers to output the final prediction. Since the patches have random sizes, they are upscaled to the same size before passed to the co-occurrence discriminator. All convolutions use kernel size 3×3. Residual blocks use the same design as those of the image discriminator. For the reference patches, more than one patch is used, so the extracted features are averaged over the batch dimension to capture the aggregated distribution of the reference texture.

4.4.8 Training details

At each iteration, we sample a minibatch of size N and produce $N/2$ reconstructed images and $N/2$ hybrid images. The reconstruction loss is computed using $N/2$ reconstructed images. The loss for the image discriminator is computed on the real, reconstructed, and hybrid images, using the adversarial loss $\mathbb{E}-\log(D(\boldsymbol{x})) + \mathbb{E}-\log(1 - D(\boldsymbol{x}_{\text{fake}}))$, where $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{fake}}$ are real and generated (both reconstructed and hybrid) images, respectively. For the details of the GAN loss, we follow the setting of StyleGAN2 [177], including the non-saturating GAN loss [12] and lazy R1 regularization [35, 177]. In particular, R1 regularization is also applied to the co-occurrence patch discriminator. The weight for R1 regularization was 10.0 for the image discriminator (following the setting of [35, 177]) and 1.0 for the co-occurrence discriminator. Lastly, the co-occurrence patch discriminator loss is computed on random crops of the real and swapped images. The size of the crops are randomly chosen between 1/8 and 1/4 of the image dimensions for each side, and are then resized to 1/4 of the original image. For each image (real or fake), 8 crops are made. For the query image (the first argument to D_{patch}), each crop is used to predict co-occurrence, producing $8N$ predictions at each iteration. For the reference image (the second argument to D_{patch}), the feature outputs are averaged before concatenated with the query feature. Both discriminators use the binary cross-entropy GAN loss.

We use ADAM [67] with 0.002 learning rate, $\beta_1 = 0.0$ and $\beta_2 = 0.99$. We use the maximum batch size that fits in memory on 8 16GB Titan V100 GPUs: 64 for images of 256×256 resolution, 16 for 512×512 resolution, and 16 for 1024×1024 resolution (with smaller network capacity). Note that only the FFHQ dataset was trained at 1024×1024 resolution; for the landscape datasets, we take advantage of the fully convolutional architecture and train with cropped images of size 512×512, and test on the full image. The weights on each loss term are simply set to be all 1.0 among the reconstruction, image GAN, and co-occurrence GAN loss.

4.4.9 Additional visual results

In Figure 4.1, 4.4, and 4.7, we have shown our results of swapping the texture and structure codes as well as manipulation results of the latent space. Here we show additional swapping and editing results.

Swapping. Here we show additional results of swapping on FFHQ (Figure 4.11), Mountains (Figure 4.13), and LSUN Church and Bedroom (Figure 4.16) dataset. For test images, the input images for the models trained on FFHQ (Figure 4.11, 4.12, and 4.14) and Mountains (Figure 4.13 and 4.15) are separately downloaded from pixabay.com using relevant keywords. The results on LSUN (Figure 4.16) are from the validation sets [237].

Editing. The latent space of our method can be used for image editing. For example, in Figure 4.14 and 4.15, we show the result of editing the texture code using an interactive UI that performs vector arithmetic using the PCA components. Editing the texture code results in changing global attributes like age, wearing glasses, lighting, and background in the FFHQ dataset (Figure 4.14), and time of day and grayscale in the Mountains dataset (Figure 4.15). On the other hand, editing the structure code can manipulate locally isolated attributes such as eye shape, gaze direction (Figure 4.12), or texture of the grass field (Figure 4.15). These results are generated by performing vector arithmetic in the latent space of the flattened structure code, masked by the region specified by the user in the UI, similar to region editing of Figure 4.8. In addition, the pond of Figure 4.15 is created by overwriting the structure code with the code of a lake from another image (cloning of Figure 4.8). More editing results of using the interactive UI can be found on our project webpage: https://taesungp.github.io/SwappingAutoencoder.

User-guided image translation. In Figure 4.18, we show the results of user-guided image translation, trained on Portrait2FFHQ and Animal Faces HQ [248]. For each dataset, the results are produced using the model trained on the mix of all domains and hence without any domain labels. By adjusting the gains on the principal components of the texture code with the interactive UI, the user controls the magnitude and style of translation. Interestingly, we found that the first principal axis of the texture code largely corresponds to the domain translation vector in the case of Portrait2FFHQ and AFHQ dataset, with the subsequent vectors controlling more fine-grained styles. Therefore, our model is suitable for the inherent multi-modal nature of image translation. For example, in Figure 4.18, the input cat and dog images are translated into six different plausible outputs.

4.4.10 Additional comparison to existing methods

In Table 4.2, we report the FIDs of the swapping results of our model and baselines on LSUN Church, FFHQ, and Waterfall datasets using the validation set. More visual comparison results that extend Figure 4.3 and 4.5 of the main paper are in Figure 4.17. Note that using FID to evaluate the results of this task is not sufficient, as it does not capture the relationship to input content and style images. For example, a low FID can be achieved simply by not making large changes to the input content image. Our model achieves the second-best FID, behind the photorealistic style transfer method WCT^2 [187]. However, the visual results of Figure 4.17 and human perceptual study of Figure 4.6 reveal that our method better captures the details of the reference style. In Table 4.3, we compare the FIDs of swapping on the training set with *unconditionally* generated StyleGAN and StyleGAN2 outputs. Note that randomly sampled images of StyleGAN and StyleGAN2 are not suitable for image editing, as it ignores the input image. The FID of swap-generated images of our method is placed between the FID of unconditionally generated StyleGAN and StyleGAN2 images.

Method	Church	FFHQ	Waterfall	Mean
Swap Autoencoder (Ours)	52.34	59.83	50.90	54.36
Im2StyleGAN [13, 227]	219.50	123.13	267.25	203.29
StyleGAN2 [177]	57.54	81.44	57.46	65.48
STROTSS [188]	70.22	92.19	108.41	83.36
WCT^2 [187]	35.65	39.02	35.88	36.85

Table 4.2: **FID of swapping on the validation set**. We compare the FIDs of content-style mixing on the validation sets. Note the utility of FID is limited in our setting, since it does not capture the quality of embedding or disentanglement. Our method achieves second-lowest FID, behind WCT^2 [187], a photorealistic style transfer method. Note that the values are not directly comparable to different datasets or to the training splits (Table 4.3), since the number of samples are different. Please see Figure 4.17 for visual results.

Method	Church	FFHQ	Waterfall	Mean
Swap Autoencoder (Ours)	3.91	3.48	3.04	
StyleGAN [13]	4.21	4.40*	6.09	
StyleGAN2 [177]	3.86*	2.84*	2.67	

Table 4.3: **FID of swapping on the training set, in the context of unconditional GAN**. We compute the FID of swapped images on the training set, and compare it with FIDs of unconditionally generated images of StyleGAN [13] and StyleGAN2 [177]. The result conveys how much realism the swap-generated images convey. Note that randomly sampled images of StyleGAN [13] and StyleGAN2 [177] models are not suitable for image editing. Asterisk(*) denotes FIDs reported in the original papers.

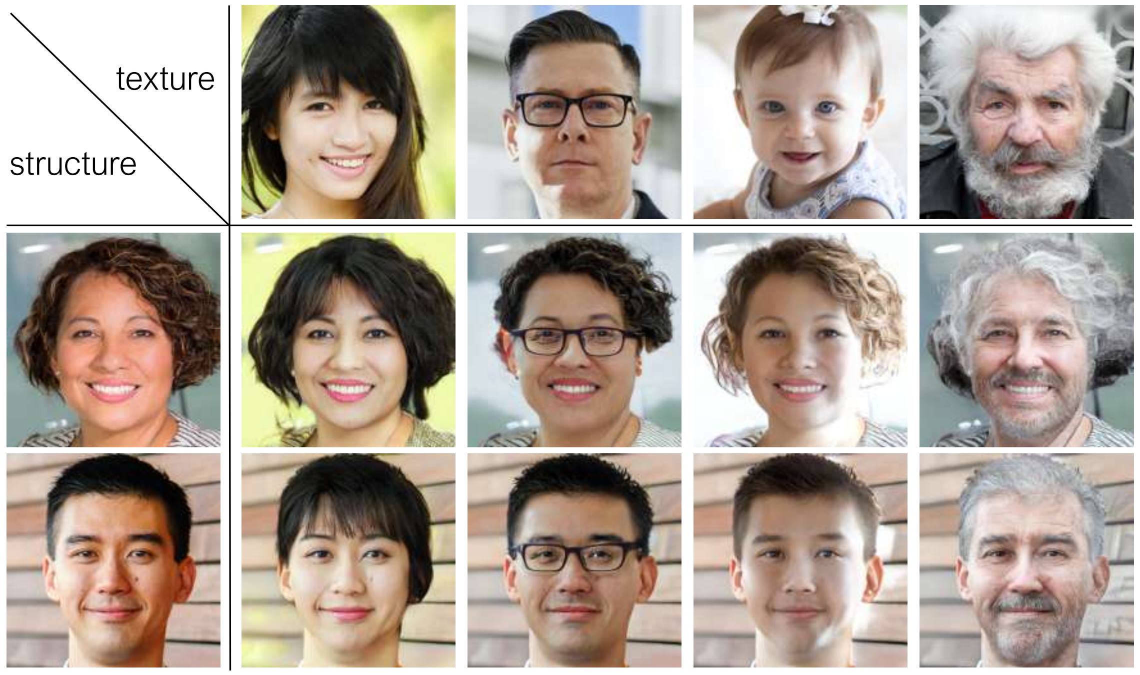


Figure 4.11: **Swapping results of our FFHQ model** (photos from pixabay.com).

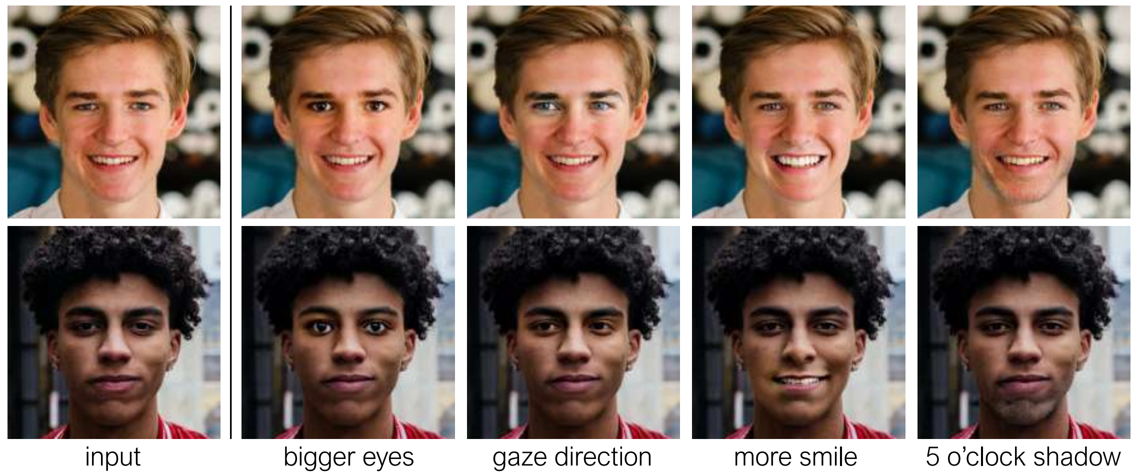


Figure 4.12: **Region editing**. The results are generated with vector arithmetic on the structure code. The vectors are discovered by a user with our UI, with each goal in mind.

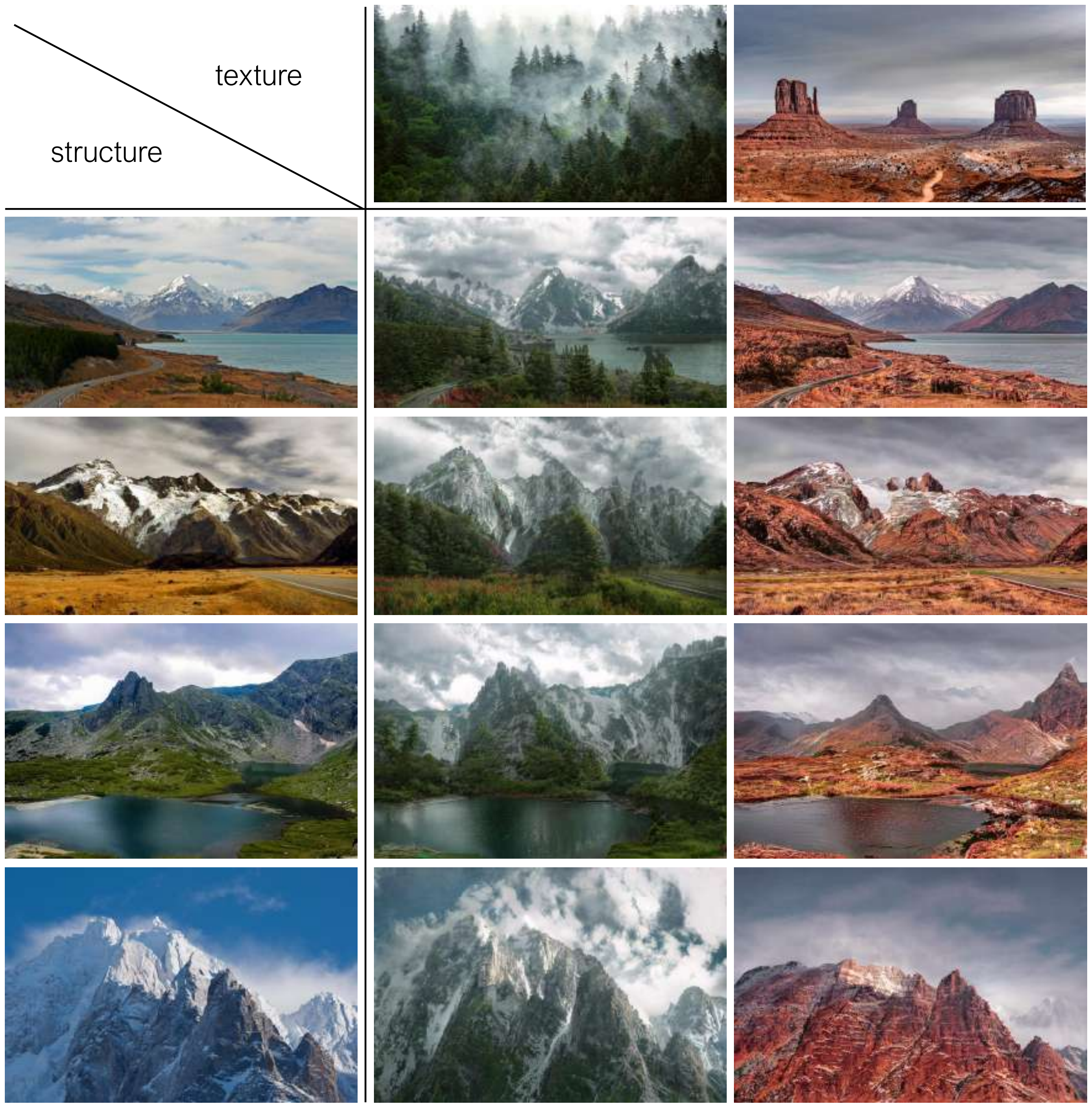


Figure 4.13: **Swapping results of our method trained on Flickr Mountains**. The model is trained and tested at 512px height.

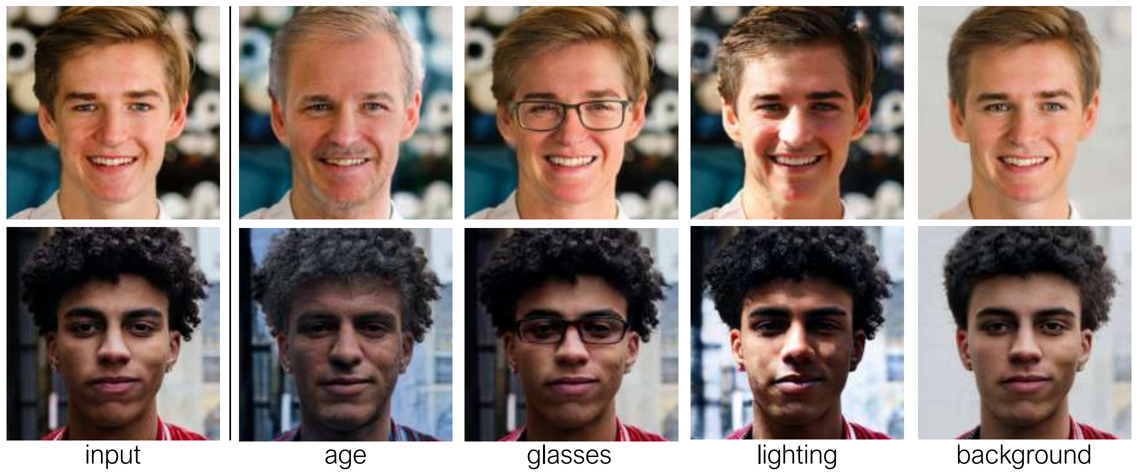


Figure 4.14: **Global editing**. The results are generated using vector arithmetic on the texture code. The vectors are discovered by a user with our UI, with each goal in mind.

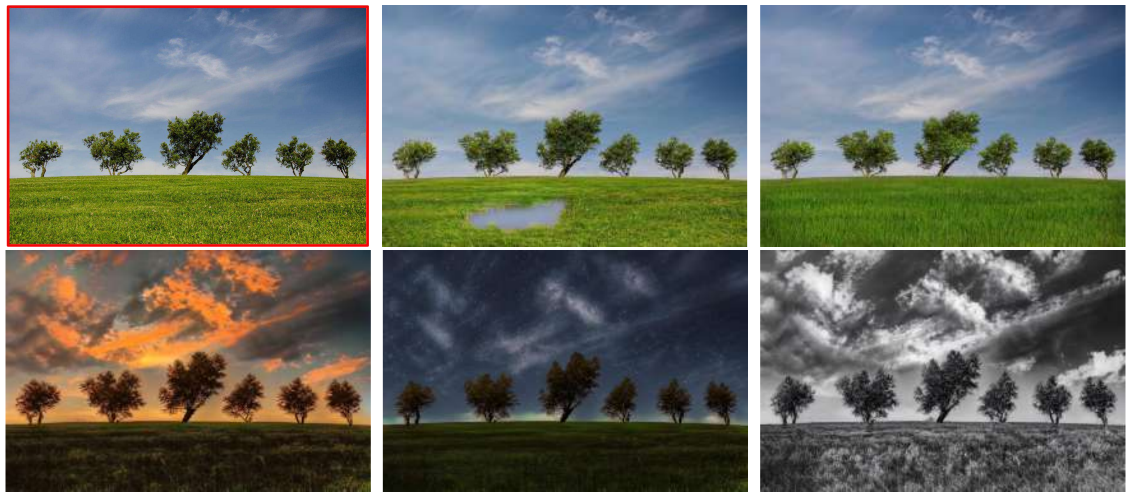
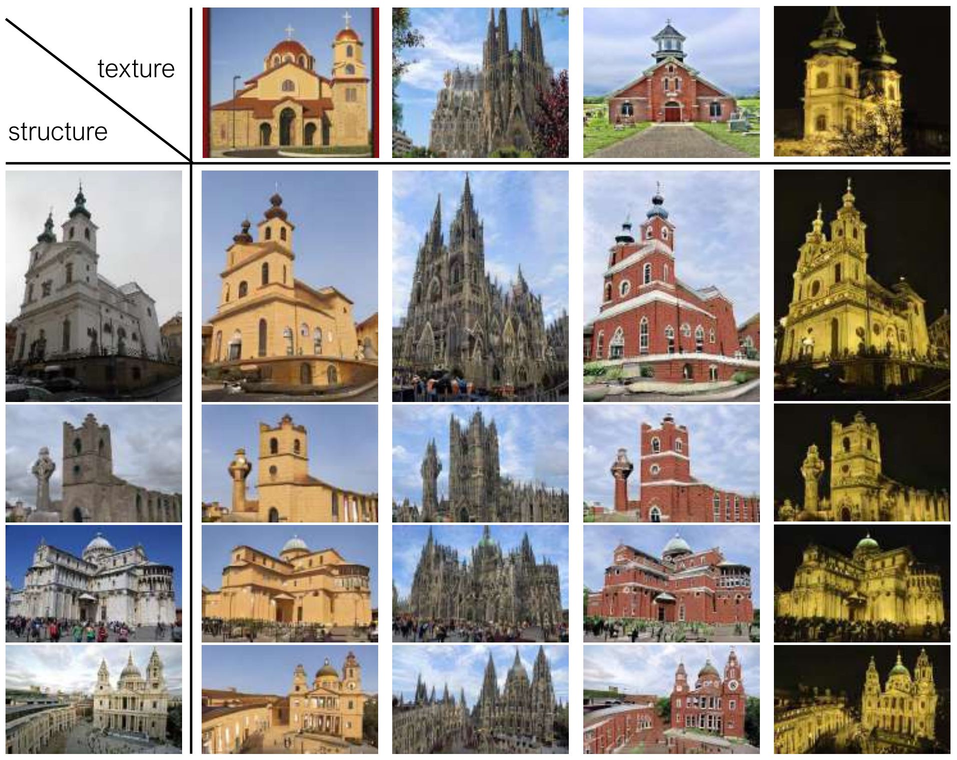


Figure 4.15: **User editing results of our method trained on Flickr Mountains**. For the input image in red, the top and bottom rows show examples of editing the structure and texture code, respectively. Please refer to Figure 4.8 on how editing is performed. The image is of 1536×1020 resolution, using a model trained at 512px resolution.



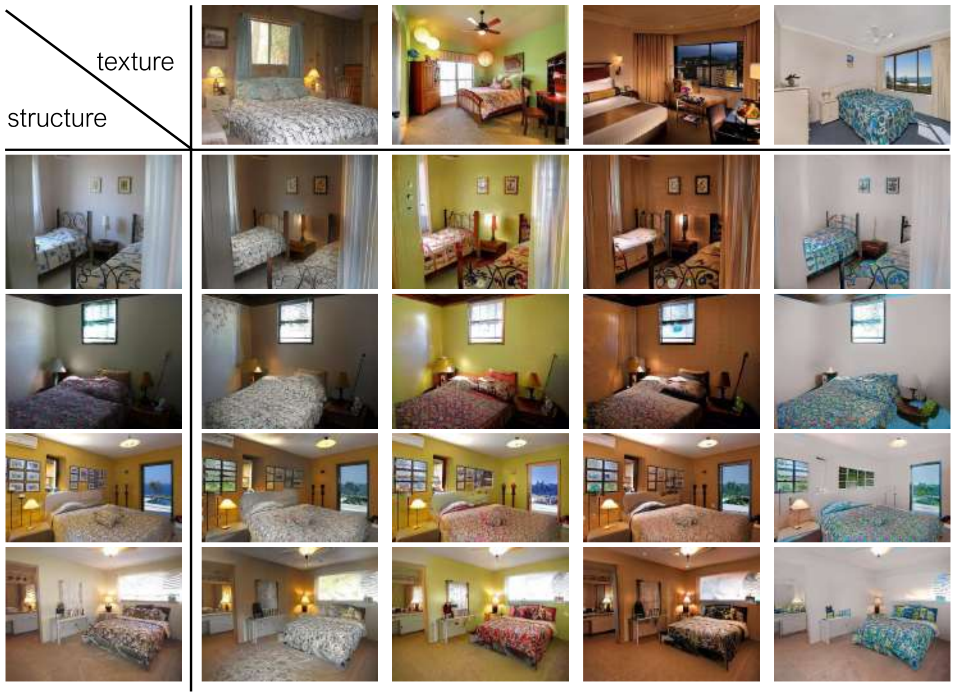


Figure 4.16: **Swapping results of LSUN** Churches (top) and Bedrooms (bottom) validation set. The model is trained with 256px-by-256px crops and tested at 256px resolution on the shorter side, keeping the aspect ratio.

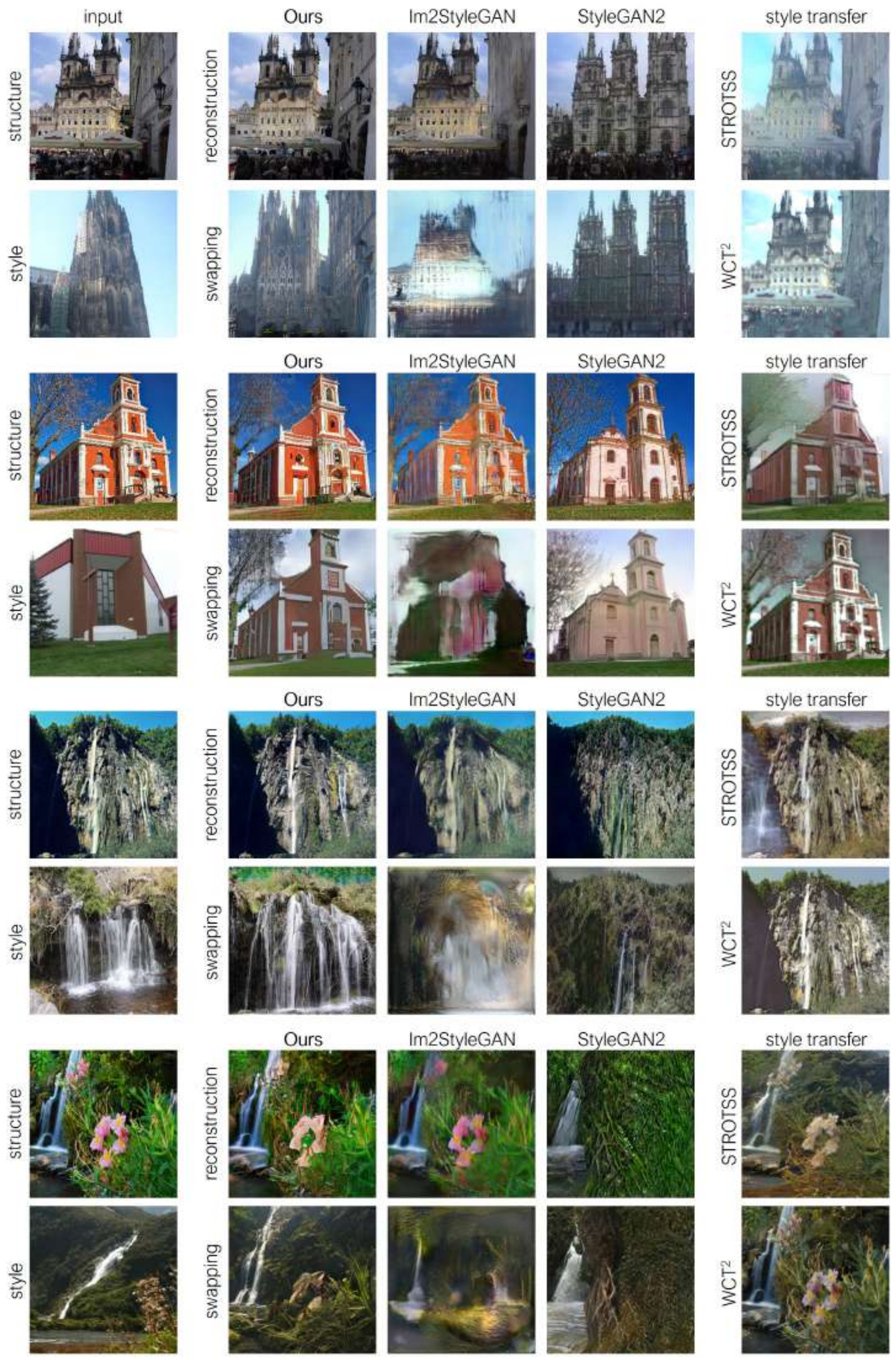


Figure 4.17: **Comparison to existing methods**. Random results on LSUN Churches and Flickr Waterfall are shown. In each block, we show both the reconstruction and swapping for ours, Im2StyleGAN [13, 227], and StyleGAN2 [177], as well as the style transfer results of STROTSS [188] and WCT^2 [187]. Im2StyleGAN has a low reconstruction error but performs poorly on the swapping task. StyleGAN2 generates realistic swappings, but fails to capture the input images faithfully. Both style transfer methods makes small changes to the input structure images.

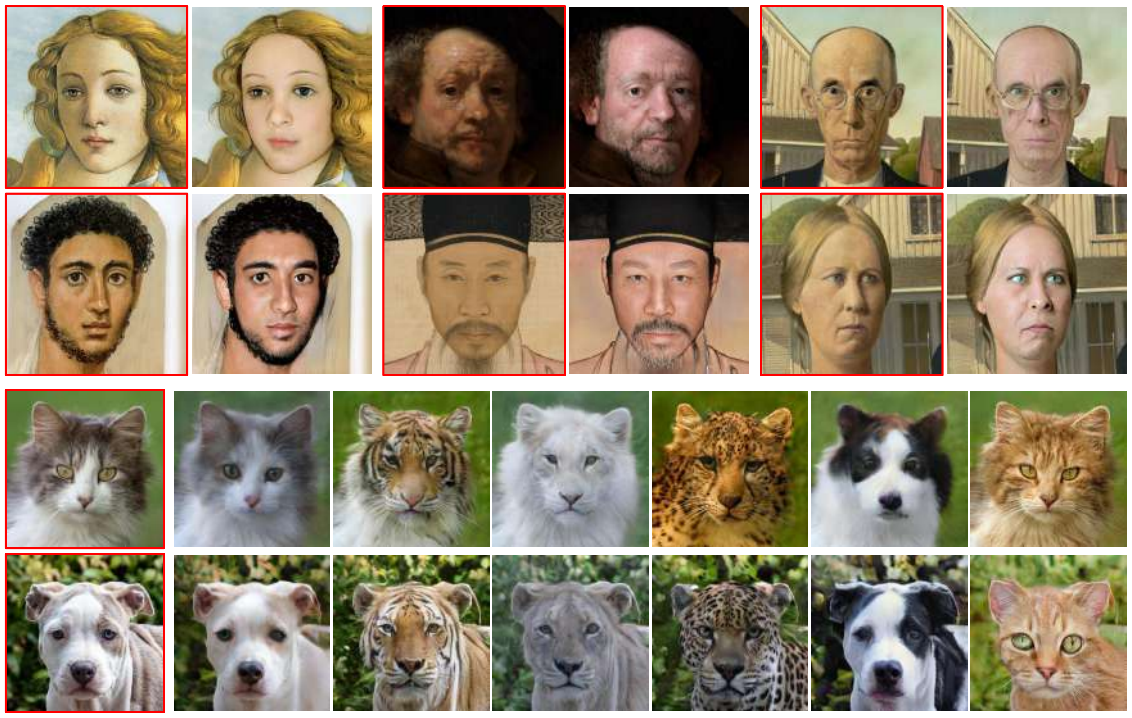


Figure 4.18: **User-guided image translation**. Using the interactive UI, the user controls the magnitude and style of the translated image. We show the edit results of turning paintings into photo (top) on the model trained on the Portrait2FFHQ dataset, and translating within the Animal Faces HQ dataset (bottom). The input images are marked in red. For the animal image translation, 6 different outputs are shown for the same input image.

4.4.11 Corruption study of Self-Similarity Distance and SIFID

In Figure 4.19, we validate our usage of Self-Similarity Matrix Distance [188] and Single-Image FID (SIFID) [184] as automated metrics for measuring distance in structure and style. Following FID [66], we study the change in both metrics under predefined corruptions. We find that the self-similarity distance shows a larger variation for image translation and rotation than blurring or adding white noise. In contrast, SIFID is more sensitive to blurring or white noise than translation or rotation. This confirms that the self-similarity captures structure, and SIFID captures style.

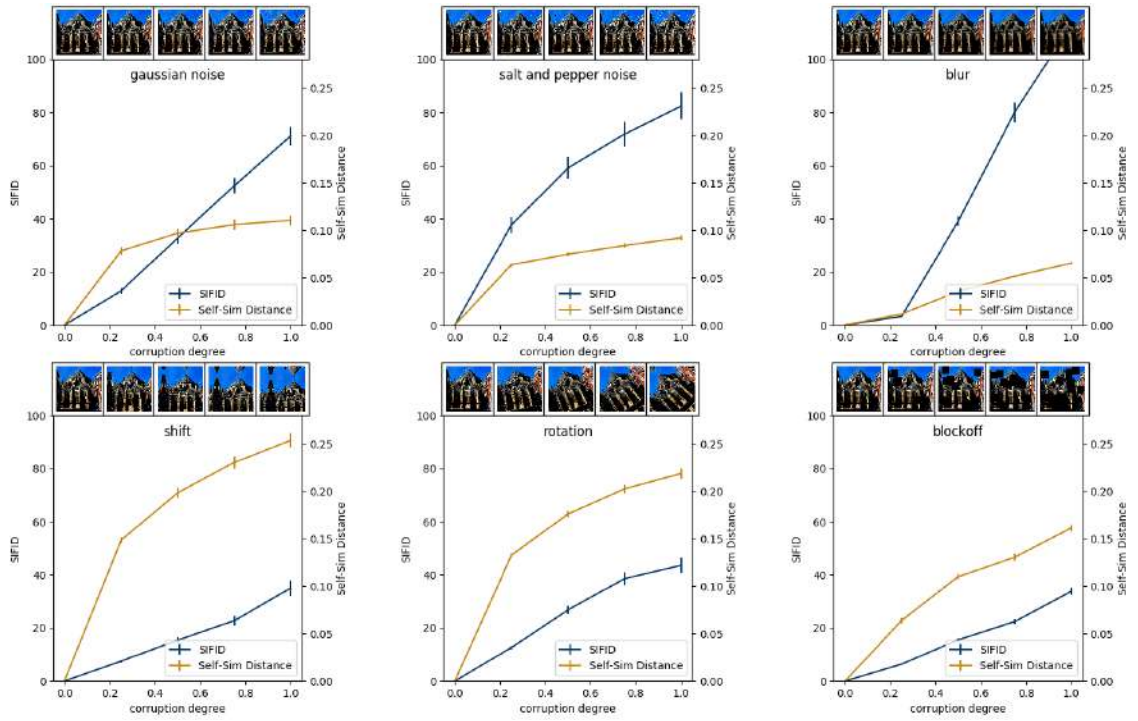


Figure 4.19: **Validating the Self-Similarity Matrix Distance and Single-Image FID**. We apply different types of corruptions and study the variation in the Self-Similarity Distance [188] and Single-Image FID [184]. SIFID shows higher sensitivity to overall style changes, such as Gaussian noise or blurring, than structural changes, such as shift and rotation. On the other hand, Self-Similarity Distance shows higher variation for structural changes. This empirically confirms our usage of the two metrics as measuring distance in structure and style.

4.5 Discussion

The main question we would like to address, is whether unconditional random image generation is required for high-quality image editing tasks. For such approaches, projection becomes a challenging operation, and intuitive disentanglement still remains a challenging question. We show that our method based on an auto-encoder model has a number of advantages over prior work, in that it can accurately embed high-resolution images in real-time, into an embedding space that disentangles texture from structure, and generates realistic output images with both swapping and vector arithmetic. We performed extensive qualitative and quantitative evaluations of our method on multiple datasets. Still, structured texture transfer remains challenging, such as the striped bedsheet of Figure 4.4. Furthermore, extensive analysis on the nature of disentanglement, ideally using reliable, automatic metrics will be beneficial as future work.

4.5.1 Discussion on broader social impact

From the sculptor's chisel to the painter's brush, tools for creative expression are an important part of human culture. The advent of digital photography and professional editing tools, such as Adobe Photoshop, has allowed artists to push creative boundaries. However, the existing tools are typically too complicated to be useful by the general public. Our work is one of the new generation of visual content creation methods that aim to democratize the creative process. The goal is to provide intuitive controls (see Chapter 4.4.6) for making a wider range of realistic visual effects available to non-experts.

While the goal of this work is to support artistic and creative applications, the potential misuse of such technology for purposes of deception – posing generated images as real photographs – is quite concerning. To partially mitigate this concern, we can use the advances in the field of image forensics [278], as a way of verifying the authenticity of a given image. In particular, Wang et al. [279] recently showed that a classifier trained to classify between real photographs and synthetic images generated by ProGAN [280], was able to detect fakes produced by other generators, among them, StyleGAN [13] and StyleGAN2 [177]. We take a pretrained model of [279] and report the detection rates on several datasets in Table 4.4. Our swap-generated images can be detected with an average rate greater than 90%, and this indicates that our method shares enough architectural components with previous methods to be detectable. However, these detection methods do not work at 100%, and performance can degrade as the images are degraded in the wild (e.g., compressed, rescanned) or via adversarial attacks. Therefore, the problem of verifying image provenance remains a significant challenge to society that requires multiple layers of solutions, from technical (such as learning-based detection systems or authenticity certification chains), to social, such as efforts to increase public awareness of the problem, to regulatory and legislative.

Method	Task	Dataset			
		Church	FFHQ	Waterfall	Average
Im2StyleGAN [13, 227]	reconstruct	99.3	100.0	92.4	97.2
	swap	100.0	100.0	97.7	99.2
StyleGAN2 [177]	reconstruct	99.7	100.0	94.4	98.0
	swap	99.8	100.0	96.6	98.8
Swap Autoencoder (Ours)	reconstruct	93.6	95.6	73.9	87.7
	swap	96.6	94.7	80.4	90.5

Table 4.4: **Detectability.** We run the CNN-generated image detector from Wang et al. [279] and report average precision (AP); chance is 50%. The CNN classifier is trained from ProGAN [280], the predecessor to StyleGAN [13]. Because our method shares architectural components, a classifier trained to detect a different method can also generalize to ours, with some dropoff, especially for the waterfall class. Notably, the performance on FFHQ faces remains high. However, performance is not reliably at 100% across all methods, indicating that future detection methods could potentially benefit from training on our method.

Chapter 5

Conclusions and Discussion

In this thesis, we discussed how machine learning could be utilized for image editing. While the existing photo editing tools focus on low-level operations, the proposed approaches based on machine learning can perform new high level operations such as synthesizing new realistic objects or transforming the input image into a entirely new style. We explored three different settings, with varying amounts of supervision provided in training our models, and observed that each has its own strengths and applications. I hope the confluence of image editing and machine learning opens up new possibilities in our imagination.

Nevertheless, the proposed approaches come with limitations. I see two opportunities that would further improve the utility of the proposed frameworks. First is embedding a notion of object constancy and physics into the model. Currently, neither the supervisory signal or the formulation convey concept about the underlying world represented by the input image. Because of this, there is no explicit way to maintain the underlying scene in the editing process. For example, the swapping results of Figure 4.13 often changes the mountain ridgeline. By incorporating 3D vision, the user may benefit from having an option to maintain the 3D scene shown by the image. Secondly, the expressiveness of structural editing can be further improved. In many settings discussed in this thesis, the structure of an image was deemed as the features that need to be maintained in the editing process. In Swapping Autoencoder, we briefly investigated how local editing could be performed with the structure code (Chapter 4.4.5, Figure 4.8, and Figure 4.12), and expanding further in this direction could enable useful modes of image editing.

Bibliography

[1] T. Park, J.-Y. Zhu, O. Wang, J. Lu, E. Shechtman, A. A. Efros, and R. Zhang, "Swapping autoencoder for deep image manipulation," in *Advances in Neural Information Processing Systems*, 2020.

[2] Apple Inc., "Best of 2019: The year's top apps," 2019. https://apps.apple.com/us/story/id1484100916.

[3] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV*, 1999.

[4] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *SIGGRAPH*, 2001.

[5] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," in *SIG*, 2009.

[6] P. J. Burt, "The pyramid as a structure for efficient computation," in *Multiresolution image processing and analysis*, pp. 6–35, Springer, 1984.

[7] P. J. Burt and R. J. Kolczynski, "Enhanced image capture through fusion," in *ICCV*, 1993.

[8] A. Oliva, A. Torralba, and P. G. Schyns, "Hybrid images," *TOG*, vol. 25, no. 3, pp. 527–532, 2006.

[9] G. Wolberg, *Digital Image Warping.* Los Alamitos, CA: IEEE Computer Society Press, 1990.

[10] T. Beier and S. Neely, "Feature-based image metamorphosis," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 35–42, 1992.

[11] S. Lee, G. Woberg, K.-Y. Chwa, and S. Y. Shin, “Image metamorphosis with scattered feature constraints,” *TVCG*, vol. 2, no. 4, pp. 337–354, 1996.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, pp. 2672–2680, 2014.

[13] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *CVPR*, 2019.

[14] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, “Semantic image synthesis with spatially-adaptive normalization,” in *CVPR*, 2019.

[15] M. Johnson, G. J. Brostow, J. Shotton, O. Arandjelovic, V. Kwatra, and R. Cipolla, “Semantic photo synthesis,” in *Computer Graphics Forum*, vol. 25, pp. 407–413, 2006.

[16] J. Hays and A. A. Efros, “Scene completion using millions of photographs,” in *SIG*, 2007.

[17] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi, “Photo clip art,” in *ACM transactions on graphics (TOG)*, vol. 26, p. 3, ACM, ACM, 2007.

[18] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, “Sketch2photo: internet image montage,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 124, 2009.

[19] T. B. Mathias Eitz, Kristian Hildebrand and M. Alexa, “Photosketch: A sketch based image query and compositing system,” in *ACM SIGGRAPH 2009 Talk Program*, 2009.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *CVPR*, 2017.

[21] Q. Chen and V. Koltun, “Photographic image synthesis with cascaded refinement networks,” in *ICCV*, vol. 1, p. 3, 2017.

[22] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-resolution image synthesis and semantic manipulation with conditional gans,” in *CVPR*, 2018.

[23] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, "Video-to-video synthesis," in *NIPS*, 2018.

[24] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *ICCV*, 2017.

[25] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan++: Realistic image synthesis with stacked generative adversarial networks," *PAMI*, 2018.

[26] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *ICML*, 2019.

[27] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *ICLR*, 2019.

[28] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[30] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *CVPR*, 2017.

[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[33] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *ICML*, 2017.

[34] T. Miyato and M. Koyama, "cGANs with projection discriminator," in *ICLR*, 2018.

[35] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?," in *ICML*, 2018.

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

[37] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” in *ICML*, 2016.

[38] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, “Attngan: Fine-grained text to image generation with attentional generative adversarial networks,” in *CVPR*, 2018.

[39] S. Hong, D. Yang, J. Choi, and H. Lee, “Inferring semantic layout for hierarchical text-to-image synthesis,” in *CVPR*, 2018.

[40] L. Karacan, Z. Akata, A. Erdem, and E. Erdem, “Learning to generate images of outdoor scenes from attributes and semantic layouts,” *arXiv preprint arXiv:1612.00215*, 2016.

[41] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *ICCV*, 2017.

[42] M.-Y. Liu, T. Breuel, and J. Kautz, “Unsupervised image-to-image translation networks,” in *NIPS*, pp. 700–708, 2017.

[43] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, “Toward multimodal image-to-image translation,” in *NIPS*, 2017.

[44] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, “Multimodal unsupervised image-to-image translation,” *ECCV*, 2018.

[45] B. Zhao, L. Meng, W. Yin, and L. Sigal, “Image generation from layout,” in *CVPR*, 2019.

[46] L. Karacan, Z. Akata, A. Erdem, and E. Erdem, “Manipulating attributes of natural scenes via hallucination,” *arXiv preprint arXiv:1808.07413*, 2018.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.

[48] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015.

[49] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization. arxiv 2016," *arXiv preprint arXiv:1607.08022*, 2016.

[50] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[51] Y. Wu and K. He, "Group normalization," in *ECCV*, 2018.

[52] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *NIPS*, 2016.

[53] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *ICLR*, 2016.

[54] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017.

[55] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," in *NIPS*, 2017.

[56] E. Perez, H. De Vries, F. Strub, V. Dumoulin, and A. Courville, "Learning visual reasoning without strong priors," in *ICML*, 2017.

[57] X. Wang, K. Yu, C. Dong, and C. Change Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 606–615, 2018.

[58] T. Chen, M. Lucic, N. Houlsby, and S. Gelly, "On self modulation for generative adversarial networks," in *International Conference on Learning Representations*, 2019.

[59] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[61] X. Mao, Q. Li, H. Xie, Y. R. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *ICCV*, pp. 2813–2821, IEEE, 2017.

[62] J. H. Lim and J. C. Ye, "Geometric gan," *arXiv preprint arXiv:1705.02894*, 2017.

[63] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[64] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[65] X. Qi, Q. Chen, J. Jia, and V. Koltun, "Semi-parametric image synthesis," in *CVPR*, 2018.

[66] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *NIPS*, 2017.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[68] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *PAMI*, vol. 40, no. 4, pp. 834–848, 2018.

[69] K. Nakashima, "Deeplab-pytorch." `https://github.com/kazuto1011/deeplab-pytorch`, 2018.

[70] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *ECCV*, 2018.

[71] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *CVPR*, 2017.

[72] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

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

[74] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[75] R. W. Brislin, "Back-translation for cross-cultural research," *Journal of cross-cultural psychology*, vol. 1, no. 3, pp. 185–216, 1970.

[76] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros, "Learning dense correspondence via 3d-guided cycle consistency," in *CVPR*, 2016.

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

[78] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *NIPS*, pp. 1486–1494, 2015.

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

[80] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *eccv*, 2016.

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

[82] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun, "Disentangling factors of variation in deep representation using adversarial training," in *NIPS*, 2016.

[83] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, pp. 2536–2544, 2016.

[84] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *ICLR*, 2016.

[85] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *NIPS*, 2016.

[86] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *NIPS*, 2016.

[87] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[88] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," *arXiv preprint arXiv:1612.00835*, 2016.

[89] R. Rosales, K. Achan, and B. J. Frey, "Unsupervised image translation," in *ICCV*, 2003.

[90] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *NIPS*, pp. 469–477, 2016.

[91] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba, "Cross-modal scene networks," *PAMI*, 2016.

[92] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *CVPR*, 2017.

[93] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," in *ICLR*, 2017.

[94] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *CVPR*, 2017.

[95] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *ICPR*, 2010.

[96] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," in *ECCV*, 2010.

[97] M. Twain, *The Jumping Frog: in English, then in French, and then Clawed Back into a Civilized Language Once More by Patient, Unremunerated Toil*, vol. 3. 1903.

[98] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W.-Y. Ma, "Dual learning for machine translation," in *NIPS*, 2016.

[99] C. Zach, M. Klopschitz, and M. Pollefeys, "Disambiguating visual relations using loop constraints," in *CVPR*, 2010.

[100] Q.-X. Huang and L. Guibas, "Consistent shape maps via semidefinite programming," in *Symposium on Geometry Processing*, 2013.

[101] F. Wang, Q. Huang, and L. J. Guibas, "Image co-segmentation via consistent functional maps," in *ICCV*, 2013.

[102] T. Zhou, Y. J. Lee, S. Yu, and A. A. Efros, "Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences," in *CVPR*, 2015.

[103] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

[104] Z. Yi, H. Zhang, P. T. Gong, *et al.*, "Dualgan: Unsupervised dual learning for image-to-image translation," *arXiv preprint arXiv:1704.02510*, 2017.

[105] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *ICML*, pp. 1857–1865, JMLR. org, 2017.

[106] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *CVPR*, 2018.

[107] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," in *ICML*, 2018.

[108] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. K. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representation," in *ECCV*, 2018.

[109] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, "Few-shot unsupervised image-to-image translation," in *ICCV*, 2019.

[110] H. Tang, H. Liu, D. Xu, P. H. Torr, and N. Sebe, "Attentiongan: Unpaired image-to-image translation using attention-guided generative adversarial networks," *arXiv preprint arXiv:1911.11897*, 2019.

[111] R. Zhang, T. Pfister, and J. Li, "Harmonic unpaired image-to-image translation," *arXiv preprint arXiv:1902.09727*, 2019.

[112] A. Gokaslan, V. Ramanujan, D. Ritchie, K. In Kim, and J. Tompkin, "Improving shape deformation in unsupervised image-to-image translation," in *ECCV*, 2018.

[113] W. Wu, K. Cao, C. Li, C. Qian, and C. C. Loy, "Transgaga: Geometry-aware unsupervised image-to-image translation," in *CVPR*, 2019.

[114] X. Liang, H. Zhang, L. Lin, and E. Xing, "Generative semantic manipulation with mask-contrasting gan," in *ECCV*, 2018.

[115] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, 2016.

[116] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.

[117] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” in *ICML*, 2016.

[118] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman, “Preserving color in neural artistic style transfer,” *arXiv preprint arXiv:1606.05897*, 2016.

[119] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, “Unsupervised pixel-level domain adaptation with generative adversarial networks,” in *CVPR*, 2017.

[120] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, “Learning from simulated and unsupervised images through adversarial training,” in *CVPR*, 2017.

[121] Y. Taigman, A. Polyak, and L. Wolf, “Unsupervised cross-domain image generation,” in *ICLR*, 2017.

[122] M. Amodio and S. Krishnaswamy, “Travelgan: Image-to-image translation by transformation vector learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8983–8992, 2019.

[123] S. Benaim and L. Wolf, “One-sided unsupervised domain mapping,” in *NeurIPS*, 2017.

[124] H. Fu, M. Gong, C. Wang, K. Batmanghelich, K. Zhang, and D. Tao, “Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping,” in *CVPR*, 2019.

[125] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.

[126] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR*, 2009.

[127] A. Dosovitskiy and T. Brox, “Generating images with perceptual similarity metrics based on deep networks,” in *NIPS*, 2016.

[128] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *cvpr*, pp. 2414–2423, 2016.

[129] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis,” in *CVPR*, 2017.

[130] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, pp. 586–595, 2018.

[131] R. Mechrez, I. Talmi, and L. Zelnik-Manor, "The contextual loss for image transformation with non-aligned data," in *ECCV*, 2018.

[132] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[133] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment," *IEEE transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, 2011.

[134] R. Mechrez, I. Talmi, F. Shama, and L. Zelnik-Manor, "Maintaining natural image statistics with the contextual loss," in *ACCV*, 2018.

[135] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[136] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[137] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

[138] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *CVPR*, pp. 6874–6883, 2017.

[139] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*, 2016.

[140] R. Zhang, P. Isola, and A. A. Efros, "Split-brain autoencoders: Unsupervised learning by cross-channel prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1058–1067, 2017.

[141] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *arXiv preprint arXiv:1605.08104*, 2016.

[142] I. Misra, C. L. Zitnick, and M. Hebert, “Shuffle and learn: unsupervised learning using temporal order verification,” in *European Conference on Computer Vision*, pp. 527–544, Springer, 2016.

[143] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *ICML*, 2011.

[144] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba, “Ambient sound provides supervision for visual learning,” in *European conference on computer vision*, pp. 801–816, Springer, 2016.

[145] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” *arXiv preprint arXiv:2002.05709*, 2020.

[146] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” *arXiv preprint arXiv:1911.05722*, 2019.

[147] O. J. Hénaff, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord, “Data-efficient image recognition with contrastive predictive coding,” *arXiv preprint arXiv:1905.09272*, 2019.

[148] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, “Learning deep representations by mutual information estimation and maximization,” *arXiv preprint arXiv:1808.06670*, 2018.

[149] S. Löwe, P. O’Connor, and B. Veeling, “Putting an end to end-to-end: Gradient-isolated learning of representations,” in *Advances in Neural Information Processing Systems*, pp. 3033–3045, 2019.

[150] I. Misra and L. van der Maaten, “Self-supervised learning of pretext-invariant representations,” *arXiv preprint arXiv:1912.01991*, 2019.

[151] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[152] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” *arXiv preprint arXiv:1906.05849*, 2019.

[153] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

[154] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304, 2010.

[155] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546, IEEE, 2005.

[156] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of Exemplar-SVMs for object detection and beyond," in *ICCV*, 2011.

[157] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros, "Data-driven visual similarity for cross-domain image matching," *SIGA*, vol. 30, no. 6, 2011.

[158] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1734–1747, 2015.

[159] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Learning visual groups from co-occurrences in space and time," *arXiv preprint arXiv:1511.06811*, 2015.

[160] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[161] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[162] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," *ECCV*, 2016.

[163] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *CVPR*, 2017.

[164] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[165] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *ICLR*, 2017.

[166] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[167] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[168] R. S. Radim Tylecek, "Spatial pattern templates for recognition of objects with regular structure," in *Proc. GCPR*, (Saarbrucken, Germany), 2013.

[169] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *CVPR*, 2014.

[170] D. Turmukhambetov, N. D. Campbell, S. J. Prince, and J. Kautz, "Modeling object appearance using context-conditioned component analysis," in *CVPR*, 2015.

[171] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," in *NeurIPS*, 2019.

[172] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for conditional image synthesis," in *ECCV*, 2020.

[173] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Crisp boundary detection using pointwise mutual information," in *ECCV*, 2014.

[174] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *CVPR*, 2011.

[175] A. Shocher, N. Cohen, and M. Irani, ""zero-shot" super-resolution using deep internal learning," in *CVPR*, 2018.

[176] C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin, "Alice: Towards understanding adversarial learning for joint distribution matching," in *NIPS*, 2017.

[177] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," *CVPR*, 2020.

[178] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," *arXiv preprint arXiv:1912.01865*, 2019.

[179] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” in *European conference on computer vision*, pp. 102–118, Springer, 2016.

[180] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation,” in *ICML*, 2018.

[181] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *CVPR*, 2016.

[182] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, “Rl-cyclegan: Reinforcement learning aware simulation-to-real,” in *CVPR*, 2020.

[183] A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in *CVPR*, pp. 1521–1528, IEEE, 2011.

[184] T. R. Shaham, T. Dekel, and T. Michaeli, “Singan: Learning a generative model from a single natural image,” in *ICCV*, 2019.

[185] A. Shocher, S. Bagon, P. Isola, and M. Irani, “Ingan: Capturing and remapping the" dna" of a natural image,” in *ICCV*, 2019.

[186] F. Luan, S. Paris, E. Shechtman, and K. Bala, “Deep photo style transfer,” in *CVPR*, 2017.

[187] J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, “Photorealistic style transfer via wavelet transforms,” in *ICCV*, 2019.

[188] N. Kolkin, J. Salavon, and G. Shakhnarovich, “Style transfer by relaxed optimal transport and self-similarity,” in *CVPR*, 2019.

[189] S. Gu, C. Chen, J. Liao, and L. Yuan, “Arbitrary style transfer with deep feature reshuffle,” in *CVPR*, 2018.

[190] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *ICLR*, 2016.

[191] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in *European conference on computer vision*, pp. 213–226, Springer, 2010.

[192] A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in *CVPR’11*, June 2011.

[193] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *CoRR*, vol. abs/1412.3474, 2014.

[194] M. Long and J. Wang, “Learning transferable features with deep adaptation networks,” in *icml*, 2015.

[195] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)* (D. Blei and F. Bach, eds.), pp. 1180–1189, JMLR Workshop and Conference Proceedings, 2015.

[196] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks,” in *International Conference in Computer Vision (ICCV)*, 2015.

[197] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[198] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, 2014.

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

[200] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” *CoRR*, vol. abs/1701.07875, 2017.

[201] M. Liu and O. Tuzel, “Coupled generative adversarial networks,” in *nips*, 2016.

[202] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, “Deep reconstruction-classification networks for unsupervised domain adaptation,” in *eccv*, pp. 597–613, Springer, 2016.

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

[204] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. Kweon, “Pixel-level domain transfer,” in *eccv*, 2016.

[205] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," in *iclr*, 2017.

[206] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[207] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

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

[209] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *arXiv preprint arXiv:1703.05192*, 2017.

[210] E. Levinkov and M. Fritz, "Sequential bayesian model update under structured scene prior for semantic road scenes labeling," in *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[211] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "FCNs in the wild: Pixel-level adversarial and constraint-based adaptation," *CoRR*, vol. abs/1612.02649, 2016.

[212] G. Ros, S. Stent, P. F. Alcantarilla, and T. Watanabe, "Training constrained deconvolutional networks for road scene semantic segmentation," *CoRR*, vol. abs/1604.01545, 2016.

[213] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[214] Y.-H. Chen, W.-Y. Chen, Y.-T. Chen, B.-C. Tsai, Y.-C. Frank Wang, and M. Sun, "No more discrimination: Cross city adaptation of road scene segmenters," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[215] Y. Zhang, P. David, and B. Gong, "Curriculum domain adaptation for semantic segmentation of urban scenes," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[216] Y. Taigman, A. Polyak, and L. Wolf, “Unsupervised cross-domain image generation,” in *International Conference on Learning Representations*, 2017.

[217] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, “The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[218] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” in *European Conference on Computer Vision (ECCV)* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9906 of *LNCS*, pp. 102–118, Springer International Publishing, 2016.

[219] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes dataset for semantic urban scene understanding,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[220] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *CVPR*, Nov. 2015.

[221] F. Yu, V. Koltun, and T. Funkhouser, “Dilated residual networks,” in *cvpr*, 2017.

[222] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Generative image inpainting with contextual attention,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[223] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in *ECCV*, 2018.

[224] S. Hong, X. Yan, T. S. Huang, and H. Lee, “Learning hierarchical semantic image manipulation through structured representations,” in *NeurIPS*, 2018.

[225] I. Anokhin, P. Solovev, D. Korzhenkov, A. Kharlamov, T. Khakhulin, A. Silvestrov, S. Nikolenko, V. Lempitsky, and G. Sterkin, “High-resolution daytime translation without domain labels,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[226] J. B. Tenenbaum and W. T. Freeman, “Separating style and content with bilinear models,” *Neural computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[227] R. Abdal, Y. Qin, and P. Wonka, “Image2stylegan: How to embed images into the stylegan latent space?,” in *ICCV*, 2019.

[228] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, “Ganspace: Discovering interpretable gan controls,” in *NIPS*, 2020.

[229] B. Julesz, “Visual pattern discrimination,” *IRE transactions on Information Theory*, vol. 8, no. 2, pp. 84–92, 1962.

[230] B. Julesz, E. N. Gilbert, L. A. Shepp, and H. L. Frisch, “Inability of humans to discriminate between visual textures that agree in second-order statistics-revisited,” *Perception*, vol. 2, no. 4, pp. 391–405, 1973.

[231] B. Julesz, “Textons, the elements of texture perception, and their interactions,” *Nature*, vol. 290, no. 5802, pp. 91–97, 1981.

[232] J. Portilla and E. P. Simoncelli, “A parametric texture model based on joint statistics of complex wavelet coefficients,” *International journal of computer vision*, vol. 40, no. 1, pp. 49–70, 2000.

[233] D. J. Heeger and J. R. Bergen, “Pyramid-based texture analysis/synthesis,” in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 229–238, 1995.

[234] L. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” in *NIPS*, vol. 12, 2015.

[235] T.-Y. Lin and S. Maji, “Visualizing and understanding deep texture representations,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2791–2799, 2016.

[236] D. Bau, H. Strobelt, W. Peebles, J. Wulff, B. Zhou, J.-Y. Zhu, and A. Torralba, “Semantic photo manipulation with a generative image prior,” *SIGGRAPH*, vol. 38, no. 4, pp. 1–11, 2019.

[237] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop,” *arXiv preprint arXiv:1506.03365*, 2015.

[238] S. Iizuka, E. Simo-Serra, and H. Ishikawa, “Globally and locally consistent image completion,” *TOG*, vol. 36, no. 4, p. 107, 2017.

[239] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, “High-resolution image inpainting using multi-scale neural patch synthesis,” in *CVPR*, 2017.

[240] G. Larsson, M. Maire, and G. Shakhnarovich, “Learning representations for automatic colorization,” in *ECCV*, 2016.

[241] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, “Real-time user-guided image colorization with learned deep priors,” *TOG*, vol. 9, no. 4, 2017.

[242] M. He, D. Chen, J. Liao, P. V. Sander, and L. Yuan, “Deep exemplar-based colorization,” *TOG*, vol. 37, no. 4, pp. 1–16, 2018.

[243] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang, “Non-stationary texture synthesis by adversarial expansion,” *TOG*, vol. 37, no. 4, 2018.

[244] E. Guérin, J. Digne, E. Galin, A. Peytavie, C. Wolf, B. Benes, and B. Martinez, “Interactive example-based terrain authoring with conditional generative adversarial networks,” *TOG*, vol. 36, no. 6, 2017.

[245] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays, “Texturegan: Controlling deep image synthesis with texture patches,” in *CVPR*, 2018.

[246] T. Portenier, Q. Hu, A. Szabó, S. A. Bigdeli, P. Favaro, and M. Zwicker, “Faceshop: Deep sketch-based face image editing,” *TOG*, vol. 37, no. 4, 2018.

[247] X. Yu, Y. Chen, S. Liu, T. Li, and G. Li, “Multi-mapping image-to-image translation via learning disentanglement,” in *NeurIPS*, 2019.

[248] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “Stargan v2: Diverse image synthesis for multiple domains,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[249] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Neural photo editing with introspective adversarial networks,” in *ICLR*, 2017.

[250] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, “Invertible conditional gans for image editing,” in *NIPS Workshop on Adversarial Training*, 2016.

[251] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *CVPR*, 2017.

[252] M. Asim, F. Shamshad, and A. Ahmed, "Blind image deconvolution using deep generative priors," *arXiv preprint arXiv:1802.04073*, 2018.

[253] D. Bau, J.-Y. Zhu, H. Strobelt, Z. Bolei, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Gan dissection: Visualizing and understanding generative adversarial networks," in *ICLR*, 2019.

[254] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, pp. 448–455, 2009.

[255] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," in *ICLR*, 2017.

[256] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *NIPS*, 2016.

[257] X. Xing, T. Han, R. Gao, S.-C. Zhu, and Y. N. Wu, "Unsupervised disentangling of appearance and geometry by deformable generator network," in *CVPR*, 2019.

[258] H. Kazemi, S. M. Iranmanesh, and N. Nasrabadi, "Style and content disentanglement in generative adversarial networks," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 848–856, IEEE, 2019.

[259] P. Esser, J. Haux, and B. Ommer, "Unsupervised robust disentangling of latent characteristics for image synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2699–2709, 2019.

[260] W. Wu, K. Cao, C. Li, C. Qian, and C. C. Loy, "Disentangling content and style via unsupervised geometry distillation," *arXiv preprint arXiv:1905.04538*, 2019.

[261] Q. Hu, A. Szabó, T. Portenier, P. Favaro, and M. Zwicker, "Disentangling factors of variation by mixing them," in *CVPR*, 2018.

[262] A. H. Jha, S. Anand, M. Singh, and V. Veeravasarapu, "Disentangling factors of variation with cycle-consistent variational auto-encoders," in *ECCV*, 2018.

[263] K. K. Singh, U. Ojha, and Y. J. Lee, "Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery," in *CVPR*, 2019.

[264] D. Kotovenko, A. Sanakoyeu, S. Lang, and B. Ommer, "Content and style disentanglement for artistic style transfer," in *ICCV*, 2019.

[265] J. Lin, Z. Chen, Y. Xia, S. Liu, T. Qin, and J. Luo, "Exploring explicit domain supervision for latent space disentanglement in unpaired image-to-image translation," *PAMI*, 2019.

[266] S. Pidhorskyi, D. A. Adjeroh, and G. Doretto, "Adversarial latent autoencoders," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[267] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1897–1906, 2017.

[268] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *ECCV*, 2018.

[269] Y. Bengio, "Deep learning of representations: Looking forward," in *International Conference on Statistical Language and Speech Processing*, 2013.

[270] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[271] H. Kim and A. Mnih, "Disentangling by factorising," in *ICML*, 2018.

[272] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Learning visual groups from co-occurrences in space and time," *arXiv preprint arXiv:1511.06811*, 2015.

[273] M. Huh, A. Liu, A. Owens, and A. A. Efros, "Fighting fake news: Image splice detection via learned self-consistency," in *ECCV*, 2018.

[274] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *ICLR*, 2019.

[275] A. Jahanian, L. Chai, and P. Isola, "On the"steerability" of generative adversarial networks," in *ICLR*, 2020.

[276] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015.

[277] R. Zhang, "Making convolutional networks shift-invariant again," in *ICML*, 2019.

[278] H. Farid, *Photo forensics.* MIT press, 2016.

[279] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot... for now," in *CVPR*, 2020.

[280] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *ICLR*, 2018.