Title
Visual Content Creation by Generative Adversarial Networks

Author
Azadi, Samaneh

Publication Date
2021

Peer reviewed|Thesis/dissertation

Visual Content Creation by Generative Adversarial Networks

by

Samaneh Azadi

A thesis 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 Trevor Darrell, Chair
Professor Alexei Efros
Professor Bruno Olshausen

Spring 2021

Visual Content Creation by Generative Adversarial Networks

Abstract

Visual Content Creation by Generative Adversarial Networks

by

Samaneh Azadi

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

We live in a world made up of different objects, people, and environments interacting with each other: people who work, write, eat, and drink; vehicles that move on land, water, or in the air; rooms that are furnished with chairs, tables, and carpets. This vast amount of information can be easily collected from the recorded videos and photographs shared online. However, it still remains a challenge to teach an intelligent machinery agent to reliably analyze and understand this extensive collection of data. Generative models are one of the most compelling methods towards modeling visual realism from the large corpus of available images, which operate by teaching a machine to create new contents. These models are not only beneficial in understanding the visual world, but more deeply in visual synthesis and content creation. They can assist human users in manipulating and editing an existing visual content. In the last few years, Generative Adversarial Networks (GANs) as an important type of generative models have made remarkable enhancements in learning complex data manifolds by generating data points from scratch. The GAN training procedure pits two neural networks against each other, a generator and a discriminator. The discriminator is trained to distinguish between the real samples and the generated ones. The generator is trained to fool the discriminator into thinking its outputs are real. The network learns the real-world distribution while generating high-quality images, translating a text phrase into an image, or transforming images from one domain to another. This dissertation investigates algorithms to improve the performance of such models in creating new visual content specifically in structural and compositional domains in a wide range from hand-designed fonts to natural complex scenes. In Chapter 2, we consider text as a visual element and propose tools to synthesize new glyphs in a font domain and transfer the style of the seen characters to the generated ones. From Chapter 3, we focus on the domain of natural images and propose GAN models capable of synthesizing complex scene images with lots of variations in the number of objects, their locations, shapes, etc. In Chapter 4, we explore the role of compositionality in the GAN frameworks and propose a new method to learn a function that maps images of different objects sampled from their marginal distributions into a combined sample that captures the joint distribution of object pairs. Despite all the

improvements in training GANs, it still remains a challenge to fully optimize the GAN generator in a two-player adversarial game, resulting in samples that do not always follow the target distribution. In Chapter 5, instead of trying to improve the training procedure, we propose an approach to improve the quality of the trained generator by post-processing its generated samples using information from the optimized discriminator.

To my best friends ever,
Maman, Baba, Sahar,
& Mohammad,
for supporting me in every possible way

and to my darling daughter, Jaanaan,
for motivating me to grow and keep trying.

Contents

Contents ii

List of Figures v

List of Tables xiii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Background . . . 1
- 1.3 Thesis Goal and Contributions . . . 2

2 Few-Shot Glyph Synthesis and Font Style Transfer 5
- 2.1 Introduction . . . 5
- 2.2 Multi-Content GAN Architecture . . . 7
  - 2.2.1 Conditional Generative Adversarial Networks . . . 8
  - 2.2.2 Glyph Network . . . 8
  - 2.2.3 Ornamentation Network . . . 9
  - 2.2.4 End-to-End Network . . . 10
- 2.3 Font Dataset . . . 12
- 2.4 Experiments and Results . . . 12
  - 2.4.1 Image Translation Baseline . . . 12
  - 2.4.2 Ablation Study . . . 14
  - 2.4.3 Automatic Learning of Correlations between Contents . . . 14
  - 2.4.4 Number of Observed Letters . . . 15
  - 2.4.5 Perceptual Evaluation . . . 15
  - 2.4.6 Ground Truth Glyph Ornamentation . . . 17
  - 2.4.7 Generalization on Synthetic Color Font Dataset . . . 17
- 2.5 Related Work . . . 18
- 2.6 Discussion . . . 19

3 Unconditional Synthesis of Complex Scenes 25
- 3.1 Introduction . . . 25

3.2 Semantic Bottleneck GAN (SB-GAN) . . . 27
3.2.1 Semantic bottleneck synthesis . . . 28
3.2.2 Semantic image synthesis . . . 29
3.2.3 End-to-end framework . . . 29
3.3 Experiments and Results . . . 30
3.3.1 Qualitative results . . . 33
3.3.2 Quantitative evaluation . . . 35
3.3.3 Perceptual evaluation . . . 36
3.4 Related Work . . . 36
3.5 Discussion . . . 38

4 Image-Conditional Binary Composition **47**
4.1 Introduction . . . 47
4.2 Background: Conditional GAN . . . 49
4.3 Compositional GAN . . . 49
4.3.1 Supervising composition by decomposition . . . 50
4.3.2 Example-Specific Meta-Refinement (ESMR) . . . 52
4.4 Implementation Details . . . 53
4.4.1 Relative spatial transformer network . . . 53
4.4.2 Relative Appearance Flow Network (RAFN) . . . 54
4.4.3 Inpainting network . . . 54
4.4.4 Full model . . . 55
4.5 Experiments . . . 56
4.5.1 Synthetic data sets . . . 58
4.5.1.1 Ablation study and baselines . . . 58
4.5.1.2 User evaluations . . . 60
4.5.2 Real data sets . . . 60
4.5.2.1 Qualitative analysis and baselines . . . 61
4.5.2.2 User evaluations . . . 62
4.5.3 Generalization to unseen categories . . . 62
4.5.4 Extension to adding more objects . . . 62
4.6 Related Work . . . 63
4.7 Discussion . . . 64

5 Discriminator Rejection Sampling **68**
5.1 Introduction . . . 68
5.2 Background . . . 69
5.2.1 Evaluation metrics: Inception Score (IS) and Fréchet Inception Distance (FID) . . . 69
5.2.2 Self-Attention GAN . . . 69
5.2.3 Rejection Sampling . . . 70
5.3 Rejection sampling for GANs . . . 71

5.3.1 Rejection sampling for GANs: the idealized version 71
5.3.2 Discriminator Rejection Sampling: the practical scheme 72
5.4 Experiments 74
5.4.1 Mixture of 25 Gaussians 75
5.5 Ablation Study 75
5.5.1 ImageNet Dataset 76
5.6 Nearest Neighbors from ImageNet 79
5.7 Discussion 79

6 Discussions **86**

A My Earlier Works in Computer Vision **88**

Bibliography **90**

List of Figures

2.1 Schematic of our Glyph Network to be trained on our 10K font data set. 6

2.2 Schematic of our end-to-end MC-GAN model including **(a)** GlyphNet and **(b)** OrnaNet. Inputs and Outputs are illustrated in white, network layers in green, and loss functions are shown in blue. We use a leave-one-out approach among all observed letters of a word like *TOWER* (in orange) to construct a batch of input image stacks to be fed into G_1: For each input stack in the batch, we extract the left out generated glyph. In addition, the remaining 21 glyphs will be generated by feeding in all observed letters together. After a reshape and gray-scale channel repetition, $\mathcal{T}$, these extracted generated glyphs, $\tilde{\text{A}}, \tilde{\text{B}}, \cdots, \tilde{\text{Z}}$ will be fed into OrnaNet. 7

2.3 Random subset of our 10K gray-scale font dataset 11

2.4 Example synthetic color gradient fonts 11

2.5 Ablation study on our MC-GAN model components: For each exemplar font, we show ground truth (**1st row**), observed letters (red squares in the 1st row), predictions of a baseline image translation network (**2nd row**), predictions of our end-to-end model with randomly initialized (RI) OrnaNet and $\lambda_2 = \lambda_3 = \lambda_4 = 0$ (**3rd row**), with pre-trained (PT) OrnaNet weights and $\lambda_2 = \lambda_3 = \lambda_4 = 0$ (**4th row**), selectively disabled loss terms (**rows 5-7**), and the full end-to-end MC-GAN model (**bottom row**). Style transfer improvements by λ_3 are highlighted in blue and degradation in the predictions by omitting each individual regularizer is highlighted in red. 13

2.6 Effect of number of observed glyphs on the quality of GlyphNet predictions. Red line is passing through median of each distribution. 14

2.7 Distributions $(\alpha|\beta)$ over SSIM scores for generating letter α given β in blue and given any other letter rather than β in red. Distributions for the most informative given letters β in generating each glyph α is shown in the left of each column while the least informative givens are presented in the right. 16

2.8 Text Effect Transfer (Yang *et al.,* 2016) failure example on clean input glyphs. . 17

2.9 Failure cases on clean input glyphs. 17

2.10 Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.,* 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results. 20
2.11 Continue - Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.,* 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results. 21
2.12 Continue - Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.,* 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results. 22
2.13 Continue - Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.,* 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results. 23
2.14 Comparison between image translation and our end-to-end multi-content GAN on our synthetic color font data set. For each example, ground truth and given letters are shown in the **1st row**, image translation outputs in the **2nd row** and MC-GAN in the **last row**. 24

3.1 (a) Examples of non-complex images from ImageNet synthesized by the state-of-the-art BigGAN model (Brock *et al.,* 2019). Although these samples look decent, the complex scenes synthesized by BigGAN (e.g., from the Cityscapes dataset) are blurry and defective in local structure (e.g., cars are blended together) (b). Zoom in for more detail. (c) A complex scene synthesized by our model respects both local and global structural integrity of the scene. (d) Schematic of our unconditional Semantic Bottleneck GAN. We progressively train the adversarial segmentation synthesis network to generate realistic segmentation maps from scratch, then synthesize a photo-realistic image using a conditional image synthesis network. End-to-end coupling of these two components results in state-of-the-art unconditional synthesis of complex scenes. 26

3.2 Schematic of Semantic Bottleneck GAN. Starting from random noise, we synthesize a segmentation layout and use a discriminator to bias the segmentation synthesis network towards realistic looking segmentation layouts. The generated layout is then provided as input to a conditional image synthesis network to synthesize the final image. A second discriminator is used to bias the conditional image synthesis network towards realistic images paired with real segmentation layouts. Finally, a third unconditional discriminator is used to bias the conditional image synthesis network towards generating images that match the real image distribution. 27
3.3 Images synthesized on Cityscapes-5K. Best viewed on screen; zoom in for more detail. Although both models capture the general scene layout, SB-GAN (1st row) generates more convincing objects, e.g. buildings and cars. 31
3.4 Images synthesized on Cityscapes-25K. Best viewed on screen; zoom in for more detail. Images synthesized by BigGAN (3rd row) are blurry and sometimes defective in local structures. 31
3.5 Images synthesized on ADE-Indoor. This dataset is very challenging, causing mode collapse for the BigGAN model (3rd row). In contrast, samples generated by SB-GAN (1st row) are generally of higher quality and much more structured than those of ProGAN (2nd row). 32
3.6 The effect of SB-GAN on improving the performance of the state-of-the-art semantic image synthesis model (SPADE) on ground truth segmentations of Cityscapes-25K validation set. For SB-GAN, we train the entire model end-to-end, extract the trained SPADE sub-network, and synthesize samples conditioned on the ground truth labels. 34
3.7 The effect of SB-GAN on improving the performance of the state-of-the-art semantic image synthesis model (SPADE) on ground truth segmentations of ADE-Indoor validation set. For SB-GAN, we train the entire model end-to-end, extract the trained SPADE sub-network, and synthesize samples conditioned on the ground truth labels. 39
3.8 The effect of fine-tuning on the baseline setup for the Cityscapes-25K dataset. We observe improvements in both the global structure of the segmentations and the performance of semantic image synthesis, resulting in images of higher quality. 40
3.9 The effect of fine-tuning (FT) on the baseline setup for ADE-Indoor dataset. Analogously to the results on Cityscapes-25K, we observe improvements in both the global structure of the segmentations and the performance of semantic image synthesis. 41

3.10 Architectural differences between our unconditional semantic bottleneck synthesis network and the conditional semantic layout synthesis network in Hong *et al.* (2018) and Li *et al.* (2019). (a) Schematic of our unconditional Semantic Bottleneck GAN. We progressively train an adversarial segmentation synthesis network to generate realistic segmentation maps from scratch, then synthesize a photo-realistic image using a conditional image synthesis network. End-to-end coupling of these two components results in state-of-the-art unconditional synthesis of complex scenes. For more detail about our conditional image synthesis network, one can refer to Section 3.2.2. (b) Schematic of the hierarchical text-to-image synthesis models inferring a semantic layout (Hong *et al.,* 2018; Li *et al.,* 2019). From an encoding of the input sentence, object bounding boxes are generated sequentially using an auto-regressive decoder, and are refined by a synthesized binary shape mask in the next step. The final image is synthesized given the constructed semantic layout and the text description. Note that whereas (b) conditionally generates masks only for objects, our model (a) unconditionally generates segmentation maps for the entire scene. 42
3.11 Segmentations and their corresponding images synthesized by SB-GAN trained on the Cityscapes-25K dataset. 43
3.12 Segmentations and their corresponding images synthesized by SB-GAN trained on the Cityscapes-25K dataset. 44
3.13 Segmentations and their corresponding images synthesized by SB-GAN trained on the ADE-Indoor dataset. 45
3.14 Segmentations and their corresponding images synthesized by SB-GAN trained on the ADE-Indoor dataset. 46

4.1 Binary Composition examples. *Top Row*: The first object or the background image, *Middle Row*: The second object or the foreground image, *Bottom Row*: The generated composite image. 48
4.2 (a) The CoDe training network includes the composition network getting a self-consistent supervisory signal from the decomposition network. This network is trained on all training images, (b) ESMR: At test time, the weights of the trained composition and decomposition networks are fine-tuned given only one test example of X and one test example of Y. The decomposition network provides the self-supervision required for updating the weights of the composition network at test time. The layers of the composition generator are presented in pink and the decomposition generator in yellow. 49

4.3 Schematic of our binary compositional GAN model at training and test times: (a) Our training model includes the inpainting networks (for unpaired data), RAFN (for paired data), the relative STN model, and the CoDe and mask prediction networks. X_s and Y_s stand for the respective object segments of real composite images in an unpaired training setup. X_r indicates the input image in an arbitrary viewpoint different from its corresponding composite image, and Y_{mask} is the binary segmentation mask of the object images in Y encoding the target viewpoint, (b) A toy example of a real composite image C, its object cutouts C_X and C_Y, and their segmentation masks, (c) We convert an unpaired training data to a paired setup by inpainting the object segment cutouts of the real composite image. The inpainted segments and their cropped variants at the center of the image are then used for training STN, (d) At test time, we fine-tune the weights of the CoDe network given only one test example from the X domain and one test example from the Y domain. The weights of the mask prediction network and STN are not updated on test examples. Each of the above modules is represented by a different color, and repeating the same module in different parts of this diagram is for the illustration purpose. 51

4.4 Relative Spatial Transformer Network: First input with three RGB channels (e.g., image of a chair) concatenated channel-wise with the second RGB image (e.g., image of a table). The network generates two transformed images each with three RGB channels. The orange feature maps are the outputs of the `conv2d` layer (represented along with their corresponding number of channels and dimensions), and the yellow maps are the outputs of the `max-pool2d` followed by a `ReLU`. The blue layers also represent fully connected layers. 53

4.5 Relative Appearance Flow Network: Input is an image of a chair with three RGB channels concatenated channel-wise with the table foreground mask. Output is the appearance flow for synthesizing a new viewpoint of the chair. All layers are convolutional. 55

4.6 Test results on (a) the chair-table and (b) the basket-bottle composition tasks trained with either paired or unpaired data. In the chair-table examples, we use X^{RAFN} as the input to all models."NN" stands for the nearest neighbor image in the paired training set, and "NoInpaint" shows the results of the unpaired model without the inpainting network. In both paired and unpaired cases, $\hat{c}^{\text{before}}$ and $\hat{c}^{\text{after}}$ show outputs of the generator before and after ESMR, respectively. Also, $\hat{c}_s^{\text{after}}$ represents summation of masked transposed inputs after the ESMR step. . 57

4.7 (a) Ablation Study: output of our compositional GAN model without the component specified on top of each column. Input is the channel-wise concatenation of the bottle and basket shown in the first two columns, (b) Baselines: As the input (9th column), each bottle is added to the basket after being scaled and translated with constant parameters. The Pix2Pix and CycleGAN outputs are shown in the last two columns. 59

4.8 (a) Test examples for the face-sunglasses composition task. *Top two rows*: input sunglasses and face images, *3rd and 4th rows*: the output of our compositional GAN for the paired and unpaired models, respectively, *Last row*: images generated by the ST-GAN (Lin *et al.,* 2018) model, (b) Test examples for the street scene-car composition task. *Top two rows*: input cars and street scenes, *3rd and 4th rows*: the output of our compositional GAN after the meta-refinement approach. Here, $\hat{c}^{\text{after}}$ shows the output of the composition generator and $\hat{c}_{\text{s}}^{\text{after}}$ represents the summation of the masked transposed inputs, *Last row*: images generated by ST-GAN. 61
4.9 Generalization of the compositional GAN model trained on bottle-basket examples to a similar unseen category of cans to be composed with baskets. The first two rows indicate the new test examples, and the last row show the generated composite images. 62
4.10 Test examples for an iterative extension of the street scenes-cars composition task to more than two objects. The first two rows show the input street and the input car images in the first iteration. The 3rd row illustrates the output of our compositional generator, used as the street scene inputs for the next iteration to be composed with the cars shown in the 4th row. The 5th row indicates the final generated composite image of the two cars added to the street scenes iteratively. 63
4.11 Failure test cases for both the paired and unpaired models on the chair-table composition task. 64
4.12 Test results on (a) the chair-table and (b) bottle-basket composition tasks trained with either paired or unpaired data. "NN" stands for the nearest neighbor image in the paired training set, and "NoInpaint" shows the results of the unpaired model without the inpainting network. In both paired and unpaired cases, $\hat{c}^{\text{before}}$ and $\hat{c}^{\text{after}}$ show outputs of the generator before and after the ESMR approach, respectively. Also, $\hat{c}_s^{\text{after}}$ represents summation of masked transposed inputs after ESMR. 65
4.13 Test examples for the face-sunglasses composition task. First two columns show the input sunglasses and face images, 3rd and 4th columns show the output of our compositional GAN for the paired and unpaired models, respectively. Last column shows images generated by the ST-GAN (Lin *et al.,* 2018) model. 66
4.14 Test examples for the street scenes-cars composition task. First two columns show the input car and street images, 3rd and 4th columns show the output of our compositional generator before and after the inference meta-refinement step, respectively. The 5th column shows our model's output by directly adding the masked inputs. The 6th and 7th columns correspond with images generated by the ST-GAN (Lin *et al.,* 2018) model and the nearest neighbor training images. 67

5.1 **Left:** For a uniform proposal distribution and Gaussian target distribution, the blue points are the result of rejection sampling and the red points are the result of naively throwing out samples for which the density ratio ($p_d(x)/p_g(x)$) is below a threshold. The naive method underrepresents the density of the tails. **Right:** the DRS algorithm. *KeepTraining* continues training using early stopping on the validation set. *BurnIn* computes a large number of density ratios to estimate their maximum. $\widetilde{D}^*$ is the logit of D^*. $\hat{F}$ is as in Equation 5.7. $\bar{M}$ is an empirical estimate of the true maximum M. 70
5.2 **(A)** Histogram of the sigmoid inputs, $\hat{F}(x)$ (left plot), and acceptance probabilities, $\sigma(\hat{F}(x))$ (center plot), on 20K fake samples before (purple) and after (green) adding the constant γ to all $F(x)$. Before adding gamma, 98.9% of the samples had an acceptance probability $<$ 1e-4. **(B)** Histogram of $\max_j p(y_j|x_i)$ from a pre-trained Inception network where $p(y_j|x_i)$ is the predicted probability of sample x_i belonging to the y_j category (from $1,000$ ImageNet categories). The green bars correspond to $25,000$ accepted samples and the red bars correspond to $25,000$ rejected samples. The rejected images are less recognizable as belonging to a distinct class. 74
5.3 Real samples from 25 2D-Gaussian Distributions (*left*) as well as fake samples generated from a trained GAN model without (*middle*) and with DRS (*right*). Results are computed as an average over five models randomly initialized and trained independently. 75
5.4 Different models generating 10,000 samples from a 2D grid of Gaussian components."No FT" stands for the discriminator not being trained to convergence. . . 77
5.5 Synthesized images with the highest (*left*) and lowest (*right*) acceptance probability scores. 79
5.6 **(A)** Inception Score and FID during ImageNet training, computed on 50,000 samples. **(B)** Each row shows images synthesized by interpolating in latent space. The color bar above each row represents the acceptance probabilities for each sample: red for high and white for low. Subjective visual quality of samples with high acceptance probability is considerably better: objects are more coherent and more recognizable as belonging to a specific class. There are fewer indistinct textures, and fewer scenes without recognizable objects. 80
5.7 Inception Score versus the rate of accepting samples on average (*left*), and the acceptance probability assigned to each sample x_i by DRS versus the maximum probability of belonging to one of the 1K categories based on a pre-trained Inception network, $\max_j p(y_j|x_i)$ (*right*). 81
5.8 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 81
5.9 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 82
5.10 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 82

5.11 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 83
5.12 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 83
5.13 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 84
5.14 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 84
5.15 Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features 85

List of Tables

3.1 FID of the synthesized samples (lower is better), averaged over 5 random sets of samples. Images were synthesized at resolution of 256x512 on Cityscapes and 256x256 on ADE-Indoor. 33
3.2 FID of the synthesized samples (lower is better), averaged over 5 random sets of samples. Images were synthesized at resolution of 128x256 on Cityscapes and 128x128 on ADE-Indoor. 33
3.3 FID of the synthesized samples when conditioned on the ground truth labels. For SB-GAN, we train the entire model end-to-end and extract the trained SPADE. 35
3.4 Ablation study of various components of SB-GAN. We report FID scores of SB-GAN before fine-tuning, fine-tuning only the semantic bottleneck synthesis component, fine-tuning only the image synthesis component, and full end-to-end fine-tuning. Experiments are performed on the Cityscapes-5K dataset at a resolution of 128×256. 36
3.5 Average perceptual evaluation scores when each evaluators has selected a quality score in the range of 1 (terrible quality) to 4 (high quality) for each image. . . . 36

4.1 AMT user evaluation comparing components of our model on the synthetic datasets. 2nd column: number of test images, 3rd column: % preferences to after vs. before refinement, 4th column: % preferences to paired training vs. unpaired. 60
4.2 AMT user evaluation comparing our model with ST-GAN on the real datasets. 2nd column: number of test images, 3rd and 4th columns: % preferences respectively to paired and upaired training vs. ST-GAN. 62

5.1 Results with and without DRS on 10,000 generated samples from a model of a 2D grid of Gaussian components. 76
5.2 Ablation study on 10,000 generated samples from a 2D grid of Gaussian components. The third to sixth columns represent % of high-quality samples within x standard deviations. "No FT" stands for the discriminator not being trained to convergence. 77
5.3 Results with and without DRS on 50K ImageNet samples. Low FID and high IS are better. 78

Acknowledgments

First and foremost, I would like to start by expressing my sincere gratitude to my advisor, Trevor Darrell, for taking me as his student and supporting me through my PhD. I am grateful for the opportunity he gave me to join his group although I was coming from a background different from computer vision once I applied to the PhD program at Berkeley. I would like to thank him for teaching me how to think broad and come up with cool research ideas and for giving me the freedom under his guidance to find the research direction that interests me the most. Additionally, I would like to thank Alyosha Efros and Bruno Olshausen for serving on my qualification and thesis committees and Jitendra Malik for chairing my qualification exam. I learned a lot from all of you and enjoyed being part of the discussions at the vision reading groups, courses, and seminars at Berkeley.

I would also like to thank those who mentored me and provided me a new insight into how to be a good researcher, in particular Jiashi Feng, Stefanie Jegelka, Matthew Fisher, and Ian Goodfellow. I started my PhD working with Jiashi and Stefanie once they were both post-docs at Berkeley. I learned the initial steps of doing research and writing a scientific paper in computer vision from them. Matt was my mentor during my internship at Adobe Research and motivated me to start working on GANs. Besides his great mentorship and precise attitude in research, I was amazed by his personality for always being motivated while relaxed and selfless. Working with Ian, The GANfather, during an internship at Google Brain was a priceless experience where he showed me how to constantly look at the big picture of a research project while questioning its fundamentals and details simultaneously.

Before starting my PhD, I was fortunate to have the opportunity to work with Suvrit Sra and Pieter Abbeel. Coming from a background of Electrical Engineering, Suvrit and Pieter showed me the beauties of computer science, machine learning, and AI through their courses at Berkeley and the projects I did under their supervision back then.

I would also like to thank all my other collaborators at Berkeley, Adobe, and Google Brain: Augustus Odena, Deepak Pathak, Mario Lucic, Michael Tschannen, Anna Rohrbach, Eli Shechtman, Eric Tzeng, Seth Dong Huk Park, Sayna Ebrahimi, Catherine Olsson, and Sylvain Gelly. I learned a lot from every single one of them, and this dissertation would not have been completed without them. In addition, I am grateful for all my peers and friends at SDH7, who were there whenever I had a concern and made beautiful memories remained from sitting in the bay and sharing scones and chocolates: Lisa Anne Hendricks, Evan Shelhamer, Jeff Donahue, Judy Hoffman, Jun-Yan Zhu, Shiry Ginosar, Marcus Rohrbach, Parsa Mahmoudieh, Yang Gao, and Kate Rakelly.

I am also thankful to my other friends at Berkeley for providing emotional support, bringing cheer and joy to our life, and making this long journey look shorter: Maedeh Golshirazi, Fereshteh Radaei, Negar Mehr, Maryam Farahmand, and many others. I am honored to have them in my life.

Finally, I would like to express my deepest gratitude to my family. My dear parents, Zahra Eghlidos and Ahmadreza Azadi, for their love and continuous support and for teaching me to be strong. My one and only sister, Sahar, for being on my side from thousands of miles away

and encouraging me to be persistent and keep up trying. My beloved husband, Mohammad Sahraeian, who has gone through all ups and downs of life with me in the last few years, for all his supports, understanding, and encouragements. I have shared so many moments of successes, failures, smiles, and tears along this journey with him and am so grateful for having him in my life. My beautiful daughter, Jaanaan, for inspiring me to try to make a better version of myself as a human and for allowing me to grow with her. Thank you all for your unconditional love. This thesis is dedicated to you.

Chapter 1

Introduction

1.1 Motivation

The technological advancements in online digital resources have made a large extent of visual content, such as photos and videos, accessible to millions of users in the blink of an eye. But how can we, as researchers, use this extensive collection of visual data to train an artificial agent to have a deep understanding of its surrounding environment? One approach towards this goal is by teaching the machine to create a new visual content that follows the regulations and structure governing the natural world projected into the existing visual data, similar to a child or a student who learns to draw a new scene after having enough observations of a similar scene. For example, painting the sky at the top and the ocean at the bottom confirms the child's understanding of these concepts ("sky" vs. "ocean") and their relationships. The family of these methods is called "generative modeling". A noteworthy advantage of training these creative artificial agents would be also in assisting and inspiring amateur human users and even artists to manipulate and edit an existing visual content, generate new designs, unleash their creativity, and effectively express their thoughts visually.

A good sketch is better than a long speech;
Napoleon Bonaparte

1.2 Background

Generative Adversarial Networks (GANs) (Goodfellow *et al.,* 2014) as an important type of generative models have made remarkable enhancements in learning complex data manifolds by generating data points from scratch and have been used in a wide variety of settings including image generation (Denton *et al.,* 2015; Yang *et al.,* 2017; Karras *et al.,* 2017a) and representation learning (Radford *et al.,* 2016; Salimans *et al.,* 2016; Liu & Tuzel, 2016; Chen *et al.,* 2016). The GAN training procedure pits two neural networks against each other, a generator G and a discriminator D. The discriminator is trained to distinguish between

the real samples and the generated ones. The generator is trained to fool the discriminator into thinking its outputs are real. It takes as input a sample from the prior $z \in Z \sim p_z$ and produces a sample $G(z) \in X$. The discriminator takes an observation $x \in X$ as input and produces a probability $D(x)$ that the observation is real. The observation is sampled either according to the density p_d (the data generating distribution) or p_g (the implicit density given by the generator and the prior). Using the standard non-saturating variant, the discriminator and generator are then trained using the following loss functions:

$$\begin{aligned} L_D &= -\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{z\sim p_z}[1 - \log D(G(z))] \\ L_G &= -\mathbb{E}_{z\sim p_z}[\log D(G(z))] \end{aligned}$$

The GAN training procedure is thus a two-player differentiable game, and the game dynamics are largely what distinguishes the study of GANs from the study of other generative models. Conditional Generative Adversarial Networks (cGANs) have emerged as a powerful method for generating images conditioned on a given input. The input cue could be in the form of an image (Isola *et al.,* 2017a; Zhu *et al.,* 2017a; Liu *et al.,* 2017; Azadi *et al.,* 2017b; Wang *et al.,* 2017a; Pathak *et al.,* 2016), a text phrase (Zhang *et al.,* 2017a; Reed *et al.,* 2016b; Reed *et al.,* 2016a; Johnson *et al.,* 2018) or a class label layout (Mirza & Osindero, 2014; Odena *et al.,* 2016; Antoniou *et al.,* 2017).

Training GANs is notoriously hard and recent efforts focused on improving neural architectures (Wang & Gupta, 2016; Karras *et al.,* 2017b; Zhang *et al.,* 2019; Chen *et al.,* 2019a), loss functions (Arjovsky *et al.,* 2017), regularization (Gulrajani *et al.,* 2017a; Miyato *et al.,* 2018), large-scale training (Brock *et al.,* 2019), self-supervision (Chen *et al.,* 2019b), and sampling (Brock *et al.,* 2019; Azadi *et al.,* 2019a). Improving the performance of GANs by disentangling structure and style has been studied by (Wang & Gupta, 2016) where structure is represented by a surface normal map and style is the texture mapped onto the structure. Another compelling approach which enables generation of high-resolution images is based on progressive training: a model is trained to first synthesize lower-resolution images (e.g. 8×8), then the resolution is gradually increased until the desired resolution is achieved (Karras *et al.,* 2017b). Recently, (Brock *et al.,* 2019) showed that GANs significantly benefit from large-scale training, both in terms of model size and batch size. We note that these models are able to synthesize high-quality images in settings where objects are very prominent and centrally placed or follow some well-defined structure, as the corresponding distribution is easier to capture.

1.3 Thesis Goal and Contributions

The structure shared between different elements of a domain plays an important role in learning its distribution and creating new visual contents from that domain. For instance, the well-defined structure of a human face with specific features and range of distances between different parts of the face (e.g., between the two eyes, or between the nose and the lips) has been an influential factor in the success of recent GAN models in synthesizing the human

faces (Karras *et al.,* 2019; Karras *et al.,* 2020). Our goal in this thesis has been to study the role of structure in content creation in a wide range from hand-designed fonts to natural complex scenes.

We start in the Chapter 2 focusing on glyph synthesis and font style transfer. Artists invest significant time into designing glyphs that are visually compatible with each other in their shape, structure, and texture. This process is labor intensive and artists often design only the subset of glyphs that are necessary for a specific task, like a movie ad title, making it difficult to be extended to other texts after the design is created. We developed an end-to-end GAN that learns the font manifold of the 26 English alphabets and automatically synthesizes the missing glyphs from a few observed letters. Leveraging the consistent structure of the A to Z letters across all fonts, we were able to predict unseen glyph shapes of a specific font, then learn to ornament the glyphs with a spectrum of styles and textures.

From Chapter 3, we switch gears to the domain of natural images where the scenes layout and objects shapes vary significantly making the image generation problem more error-prone and challenging. Recent techniques mostly perform well on the domains or categories with a specific structure such as center-aligned faces (Karras *et al.,* 2017a; Karras *et al.,* 2019) or single objects at the center of the scene (Brock *et al.,* 2019). As image resolution and complexity increase, the coherence of synthesized images decreases, i.e., samples contain convincing local textures, but lack a consistent global structure. Label-conditional GANs (Wang *et al.,* 2018; Park *et al.,* 2019) were recently proposed to alleviate some of these issues by synthesizing high-quality scenes using a strong conditioning mechanism based on semantic segmentation labels during the scene generation process. Global structure encoded in the segmentation layout of the scene is what allows these models to focus primarily on generating convincing local content consistent with that structure.

In Chapter 3, we couple the high-fidelity generation capabilities of label-conditional image synthesis methods with the flexibility of unconditional generative models and propose a Semantic Bottleneck GAN model for unconditional synthesis of complex scenes. This model generates a novel content image represented as a realistic segmentation layout and then synthesizes a photorealistic scene conditioned on that layout. This enables the model to synthesize an unlimited number of novel complex scenes, while still maintaining high-fidelity output characteristic of image-conditional models. When trained end-to-end, the model yields samples which have a coherent global structure as well as fine local details.

In Chapter 4, we investigate the importance of structure to model compositionality in natural images considering the fact that an image is a 2D projection of a composition of multiple objects interacting in a 3D visual world. We explore the role of compositionality in GAN frameworks and propose a new method which learns a function that maps images of different objects sampled from their marginal distributions (e.g., chair and table) into a combined sample (table–chair) that captures the joint distribution of object pairs. Here, we specifically focus on the composition of a pair of objects. For instance, given an image of a chair and an image of a table, our formulation is able to generate an image containing the same chair–table pair arranged in a realistic manner. The structure here defines the relative scaling of the objects, their spatial layout, occlusion ordering, and viewpoint transformation.

We propose a novel self-consistent composition-by-decomposition framework consisting of two conditional GANs to compose a pair of objects.

Despite the efforts made to improve the performance of the GAN models as discussed above, the synthesized samples do not always follow the target distribution and could be defective in the structure due to the challenging GAN optimization as a two-player adversarial game. In Chapter 5, instead of trying to improve the GAN training procedure, we propose an approach to improve the quality of the trained generator by post-processing its generated samples using information from the optimized discriminator. This practical method focuses on using the discriminator as part of a probabilistic rejection sampling scheme to discard faulty generated samples. We show that under quite strict assumptions, this scheme allows us to recover the data distribution exactly.

Chapter 2

Few-Shot Glyph Synthesis and Font Style Transfer

2.1 Introduction

Text is a prominent visual element of 2D design. Artists invest significant time into designing glyphs that are visually compatible with other elements in their shape and texture. This process is labor intensive and artists often design only the subset of glyphs that are necessary for a title or an annotation, which makes it difficult to alter the text after the design is created, or to transfer an observed instance of a font to your own project. In this work, we propose a neural network architecture that automatically synthesizes the missing glyphs from a few image examples [1].

Early research on glyph synthesis focused on geometric modeling of outlines (Suveeranont & Igarashi, 2010; Campbell & Kautz, 2014; Phan *et al.,* 2015), which is limited to particular glyph topology (e.g., cannot be applied to decorative or hand-written glyphs) and cannot be used with image input. With the rise of deep neural networks, researchers have looked at modeling glyphs from images (Baluja, 2016; Upchurch *et al.,* 2016; Lyu *et al.,* 2017; Chang & Gu, 2017). We improve this approach by leveraging recent advances in conditional generative adversarial networks (cGANS) (Isola *et al.,* 2016), which have been successful in many generative applications, but produce significant artifacts when directly used to generate fonts (Figure 5.4, 2nd row). Instead of training a single network for all possible typeface ornamentations, we show how to use our multi-content GAN architecture to retrain a customized network for each observed character set with only a handful of observed glyphs.

Our network operates in two stages, first modeling the overall glyph shape and then synthesizing the final appearance with color and texture, enabling transfer of fine decorative elements. Some recent texture transfer techniques directly leverage glyph structure as guiding channels to improve the placement of decorative elements (Yang *et al.,* 2016). While this

[1] This chapter is based on joint work done with Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell (Azadi *et al.,* 2018) presented at CVPR 2018

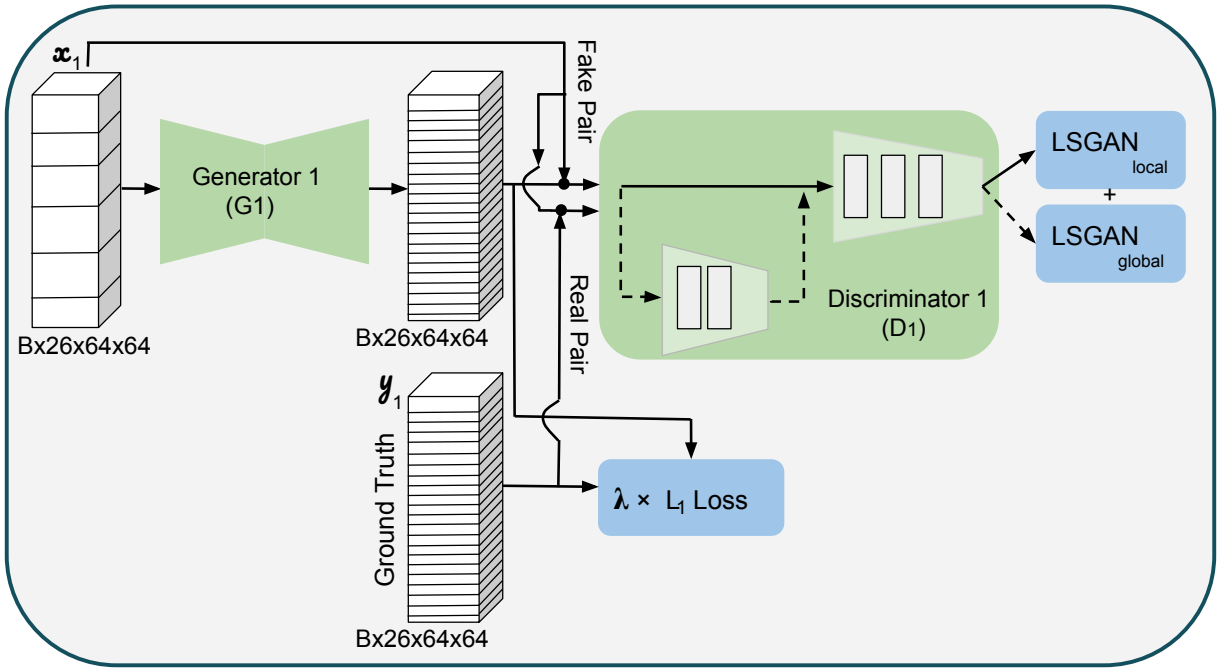


Figure 2.1: Schematic of our Glyph Network to be trained on our 10K font data set.

approach provides good results on clean glyphs it tends to fail on automatically-generated glyphs, as the artifacts of the synthesis procedure make it harder to obtain proper guidance from the glyph structure. Instead, we propose to train an ornamentation network jointly with the glyph generation network, enabling our ornament synthesis approach to learn how to decorate automatically generated glyphs with color and texture and also fix issues that arise during glyph generation. We demonstrate that users strongly preferred the output of our glyph ornamentation network in the end-to-end glyph synthesis pipeline.

Our Contributions. We propose the first end-to-end solution to synthesizing ornamented glyphs from images of a few example glyphs in the same style. To enable this, we develop a novel stacked cGAN architecture to predict the coarse glyph shapes, and a novel ornamentation network to predict color and texture of the final glyphs. These networks are trained jointly and specialized for each typeface using a very small number of observations, and we demonstrate the benefit of each component in our architecture (Figure 5.4). We use a perceptual evaluation to demonstrate the benefit of our jointly-trained network over effect transfer approaches augmented with a baseline glyph-outline inference network (Section 2.4.5).

Our Multi-Content GAN (*MC-GAN*) code and dataset are available at https://github.com/azadis/MC-GAN.

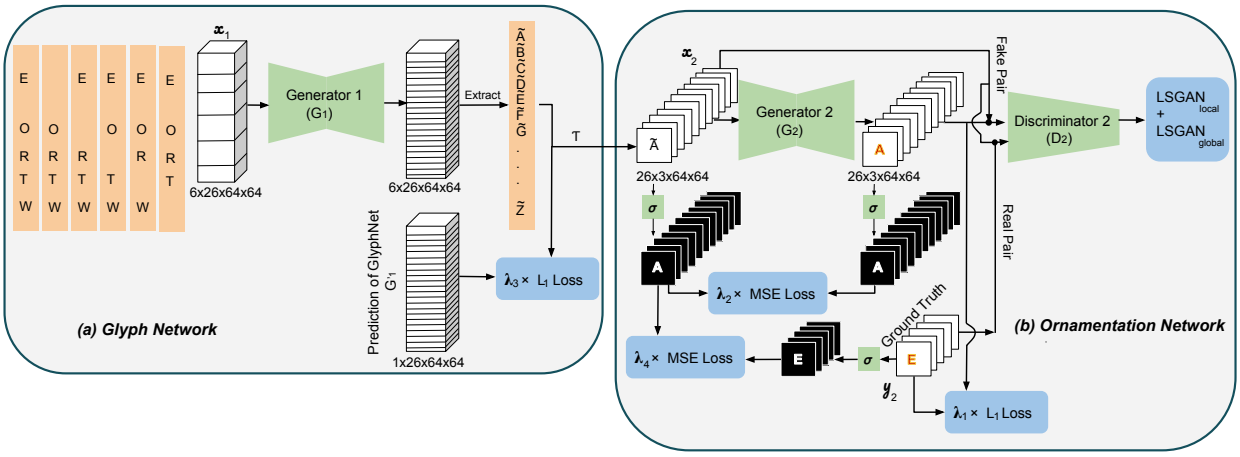


Figure 2.2: Schematic of our end-to-end MC-GAN model including **(a)** GlyphNet and **(b)** OrnaNet. Inputs and Outputs are illustrated in white, network layers in green, and loss functions are shown in blue. We use a leave-one-out approach among all observed letters of a word like *TOWER* (in orange) to construct a batch of input image stacks to be fed into G_1: For each input stack in the batch, we extract the left out generated glyph. In addition, the remaining 21 glyphs will be generated by feeding in all observed letters together. After a reshape and gray-scale channel repetition, $\mathcal{T}$, these extracted generated glyphs, $\tilde{\mathrm{A}}, \tilde{\mathrm{B}}, \cdots, \tilde{\mathrm{Z}}$ will be fed into OrnaNet.

2.2 Multi-Content GAN Architecture

We propose an end-to-end network to take a subset of stylized images of specific categories (such as font glyphs) and predict the whole set of stylistically similar images. We have specifically designed our model for the font generation problem to predict the set of letters from A to Z for in-the-wild fonts with a few observed letters. We divide this problem into two parts: glyph generation and texture transfer. Our first network, called GlyphNet, predicts glyph masks while our second network, called OrnaNet, fine-tunes color and ornamentation of the generated glyphs from the first network. Each sub-network follows the conditional generative adversarial network (cGAN) architecture (Isola *et al.*, 2016) modified for its specific purpose of stylizing glyphs or ornamentation prediction. We assume the label for each observed letter is known for the model and thus, skip the need for categorizing each letter into the 26 letters. In the following sections, we will first summarize the cGAN model, and then discuss our proposed GlyphNet and OrnaNet architectures and stack them together in an end-to-end final design which we refer to as *MC-GAN*.

2.2.1 Conditional Generative Adversarial Networks

Starting from a random noise vector z, generative adversarial networks (GANs) (Goodfellow *et al.,* 2014a) train a model to generate images y following a specific distribution by adversarially training a generator versus a discriminator ($z \rightarrow y$). While the discriminator tries to distinguish between real and fake images, the generator opposes the discriminator by trying to generate realistic-looking images. In the conditional GAN (cGAN) scenario (Isola *et al.,* 2016; Mirza & Osindero, 2014), this mapping is modified by feeding an observed image x alongside the random noise vector to the generator ($\{x, z\} \rightarrow y$), and thus, the adversary between generator and discriminator is formulated as the following loss function:

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{x,y\sim p_{\text{data}}(x,y)}[\log D(x, y)] + \mathbb{E}_{x\sim p_{\text{data}}(x),z\sim p_z(z)}[1 - \log D(x, G(x, z))], \quad (2.1)$$

where G and D minimize and maximize this loss function, respectively.

Given the ground truth output of the generator, it is also beneficial to force the model to generate images which are close to their targets through an L_1 loss function in addition to fooling the discriminator. The generator's objective can be summarized as:

$$G^* = \arg\min_G \max_D \mathcal{L}_{\text{cGAN}}(G, D) + \lambda \mathcal{L}_{L_1}(G), \quad (2.2)$$

where $\mathcal{L}_{L_1}(G) = \mathbb{E}_{x,y\sim p_{\text{data}}(x,y),z\sim p_z(z)}[\|y - G(x, z)\|_1]$.

We follow this conditional GAN setting in each of our sub-networks to generate the whole set of letters with a consistent style, y, by observing only a few examples fed in as a stack, x. Similar to (Isola *et al.,* 2016), we ignore random noise as the input to the generator, and dropout is the only source of randomness in the network.

2.2.2 Glyph Network

Generalizing all 26 capital letters of a font from a few example glyphs requires capturing correlations and similarities among source letters and the unseen ones. Our GlyphNet learns such correlations automatically in order to generate a whole set of stylistically similar glyphs. We study this behavior in Section 2.4.3.

Due to the style similarity among all content images, we add one input channel for each individual glyph in our proposed GlyphNet resulting in a "glyph stack" in both input and the generated output (as illustrated in Figure 2.1). A basic tiling of all 26 glyphs into a single image, however, fails to capture correlations among them specifically for those far from each other along the image length. This occurs due to the smaller size of convolution receptive fields than the image length within a reasonable number of convolutional layers.

With our novel input glyph stack design, correlation between different glyphs are learned across network channels in order to transfer their style automatically. We employ our generator, G_1, based on the image transformation network introduced in (Johnson *et al.,* 2016) including six ResNet blocks.

We consider 64×64 glyphs in gray-scale resulting in the input and output dimension of $B \times 26 \times 64 \times 64$ for the 26 capital English alphabets, with B indicating batch size. Following

the PatchGAN model proposed by (Isola *et al.,* 2016), we apply a 21×21 local discriminator with three convolutional layers on top of the generated output stack in order to discriminate between real and fake local patches resulting in a receptive field size equal to 21. In parallel, we add two extra convolutional layers as a global discriminator, resulting in a receptive field covering the whole image to distinguish between realistic font images and generated ones. In Figure 2.1, our local and global discriminators are shown within one discriminator block and will be referred as D_1.

For higher quality results and to stabilize GAN training (Zhu *et al.,* 2017a), we use two least squares GAN (LSGAN) loss functions (Mao *et al.,* 2017a) on our local and global discriminators added with an L_1 loss penalizing deviation of generated images $G_1(x_1)$ from their ground truth y_1:

$$\begin{aligned}\mathcal{L}(G_1) &= \lambda\mathcal{L}_{L_1}(G_1) + \mathcal{L}_{\text{LSGAN}}(G_1, D_1) = \lambda\mathbb{E}_{x_1,y_1\sim p_{\text{data}}(x_1,y_1)}[\|y_1 - G_1(x_1)\|_1] \\ &+ \mathbb{E}_{y_1\sim p_{\text{data}}(y_1)}[(D_1(y_1)-1)^2] + \mathbb{E}_{x_1\sim p_{\text{data}}(x_1)}[D_1(G_1(x_1))^2],\end{aligned}$$

where

$$\mathcal{L}_{\text{LSGAN}}(G_1, D_1) = \mathcal{L}^{\text{local}}_{\text{LSGAN}}(G_1, D_1) + \mathcal{L}^{\text{global}}_{\text{LSGAN}}(G_1, D_1).$$

We train this network on our collected 10K font data set (Section 2.3) where in each training iteration, x_1 includes a randomly chosen subset of y_1 glyphs with the remaining input channels being zeroed out. We will refer to this *trained model* as G_1' in the following sections. We explored adding in a separate input indicator channel denoting which of the glyphs are present, but did not find this to significantly affect the quality of the generator.

While we pre-train the GlyphNet using the conditional discriminator, we will remove this discriminator when training the joint network (Section 2.2.4).

2.2.3 Ornamentation Network

Our second sub-network, OrnaNet, is designed to transfer ornamentation of the few observed letters to the gray-scale glyphs through another conditional GAN network consisting of a generator, G_2, and a discriminator, D_2. Feeding in the glyphs as input images, x_2, this network generates outputs, $G_2(x_2)$, enriched with desirable color and ornamentation. The main difference between our proposed OrnaNet and GlyphNet lies in the dimension and type of inputs and outputs, as well as in how broad vs. specific the model is in generating images with a particular style; the generator and conditional discriminator architecture is otherwise identical to GlyphNet.

While GlyphNet is designed to generalize glyph correlations across all our training fonts, OrnaNet is specialized to apply only the specific ornamentation observed in a given observed font. It is trained only on the small number of observations available. Moreover, inputs and outputs of the OrnaNet include a batch of images with three RGB channels (similar to the format of the input and output images used in (Isola *et al.,* 2016)) where the the input channels are repeats of the gray-scale glyphs. In the next section, we will describe how to

combine our GlyphNet and OrnaNet in an end-to-end manner in order to generate stylized glyphs in an ornamented typeface.

2.2.4 End-to-End Network

The goal of our end-to-end model, illustrated in Figure 2.2, is to generalize both style and ornamentation of the observed letters to the unobserved ones. For this purpose, we generate all 26 glyphs including the observed ones through the pre-trained GlyphNet and feed them to the OrnaNet (initialized with random weights) to be fine-tuned. To accomplish this, we use a leave-one-out approach to cycle all possible unobserved letters:

For instance, given 5 observed letters of the word *TOWER* shown in Figure 2.2, we first use 4 letters *T, O, W, E* as the given channels in a $1 \times 26 \times 64 \times 64$ input stack and feed it to the pre-trained GlyphNet to generate all 26 letters and then extract the one fake glyph, $\tilde{R}$, not included in the input stack. Repeating this process would generate all of the 5 observed letters from the pre-trained GlyphNet. Similarly, we extract the 21 remaining letters from the pre-trained model by feeding in a $1 \times 26 \times 64 \times 64$ input stack filled with all 5 observed letters simultaneously while zeroing out all other channels. This whole process can be summarized by passing 6 input stacks each with dimension $1 \times 26 \times 64 \times 64$ through GlyphNet as a batch, extracting the relevant channel from each output stack, and finally concatenating them into one $1 \times 26 \times 64 \times 64$ output. After a reshape transformation and gray-scale channel repetition, represented by $\mathcal{T}$, we can transform this generated output to 26 images with dimension $3 \times 64 \times 64$ and feed them as a batch, x_2, to OrnaNet. This leave-one-out approach enables OrnaNet to generate high quality stylized letters from coarse generated glyphs.

To stabilize adversarial training of the OrnaNet generator (G_2) and discriminator (D_2), we likewise use an LSGAN loss added with an L_1 loss function on generated images of the observed letters, x_2, and their ground truth, y_2. Moreover, to generate a set of color images with clean outlines, we minimize the mean square error (MSE) between binary masks of the outputs and inputs of the generator in OrnaNet which are fake color letters, $G_2(x_2)$, and fake gray-scale glyphs, x_2, respectively. Binary masks are obtained by passing images through a sigmoid function, indicated as σ in equation 2.3. In summary, the loss function applied on top of the OrnaNet in the end-to-end scenario can be written as:

$$\begin{aligned} \mathcal{L}(G_2) &= \mathcal{L}_{\text{LSGAN}}(G_2, D_2) + \lambda_1 \mathcal{L}_{L_1}(G_2) + \lambda_2 \mathcal{L}_{\text{MSE}}(G_2) \\ &= \mathbb{E}_{y_2 \sim p_{\text{data}}(y_2)}[(D_2(y_2) - 1)^2] + \mathbb{E}_{x_2 \sim p_{\text{data}}(x_2)}[D_2(G_2(x_2))^2] \\ &+ \mathbb{E}_{x_2, y_2 \sim p_{\text{data}}(x_2, y_2)}\big[\lambda_1 \|y_2 - G_2(x_2)\|_1 + \lambda_2 (\sigma(y_2) - \sigma(G_2(x_2)))^2\big], \end{aligned} \tag{2.3}$$

where $x_2 = \mathcal{T}(G_1(x_1))$ and

$$\mathcal{L}_{\text{LSGAN}}(G_2, D_2) = \mathcal{L}^{\text{local}}_{\text{LSGAN}}(G_2, D_2) + \mathcal{L}^{\text{global}}_{\text{LSGAN}}(G_2, D_2).$$

.

Figure 2.3: Random subset of our 10K gray-scale font dataset

Figure 2.4: Example synthetic color gradient fonts

In the final end-to-end training, we do not use discriminator D_1 in the GlyphNet and instead, OrnaNet plays the role of a loss function by back propagating the gradients of the objective in equation 2.3 to improve style of the generated glyphs. Adding a weighted L_1 loss on top of the generator in GlyphNet, G_1, also penalizes deviating from the predictions of the pre-trained GlyphNet, G_1'. We also add an MSE loss function between binary masks of fake versions of the observed glyphs, $\mathcal{T}(G_1(x_1))$, and masks of their corresponding ground truth glyphs, y_2. Putting this all together, the gradients of the following loss functions would be passed through GlyphNet in addition to the gradient coming from OrnaNet:

$$\begin{aligned}\mathcal{L}(G_1) &= \lambda_3 \mathcal{L}_{w,L_1}(G_1) + \lambda_4 \mathcal{L}_{\text{MSE}}(G_1) \qquad (2.4)\\ &= \mathbb{E}_{x_1 \sim p_{\text{data}}(x_1), y_2 \sim p_{\text{data}}(y_2)} \big[\lambda_3 \sum_{i=1}^{26} w_i \times |G_1^i(x_1) - G_1'^i(x_1)| + \lambda_4 (\sigma(y_2) - \sigma(\mathcal{T}(G_1(x_1)))^2\big],\end{aligned}$$

where w_i allows us to apply different weights to observed vs. unobserved glyphs. Ratio between different terms in loss functions in equation 2.3, equation 2.4 is defined based on hyper-parameters λ_1 to λ_4. Moreover, as mentioned in Section 2.2.2, $G_1'(x)$ indicates the prediction of the pre-trained GlyphNet before being updated through end-to-end training.

2.3 Font Dataset

We have collected a dataset including 10K gray-scale Latin fonts each with 26 capital letters. We process the dataset by finding a bounding box around each glyph and resize it so that the larger dimension reaches 64 pixels, then pad to create 64×64 glyphs. A few exemplar fonts from our dataset are depicted in Figure 2.3. These fonts contain rich information about inter-letter correlations in font styles, but only encode glyph outlines and not font ornamentations. To create a baseline dataset of ornamented fonts, we apply random color gradients and outlining on the gray-scale glyphs, two random color gradients on each font, resulting in a 20K color font data set. A few examples are shown in Figure 2.4. Size of this data set can be arbitrarily increased through generating more random colors. These gradient fonts do not have the same distribution as in-the-wild ornamentations but can be used for applications such as network pre-training.

2.4 Experiments and Results

We demonstrate the quality of our end-to-end model predictions on multiple fonts with various styles and decorations. First, we study the advantage of various components of our model through different ablation studies. Next, we will show the significant improvement obtained by our model in transferring ornamentations on our synthesized glyphs compared with patch-based text effect transfer approach (Yang *et al.*, 2016). In the following experiments, we have set $\lambda_1 = 300, \lambda_2 = 300$ if epoch < 200 and $\lambda_2 = 3$ otherwise, $\lambda_3 = 10, \lambda_4 = 300$, while $w_i = 10$ if i is an observed glyph and $w_i = 1$ otherwise.

For evaluation, we download ornamented fonts from the web[1]. For all experiments in sections 2.4.1 and 2.4.2, we have made sure that all font examples used in these studies were not included in our 10K font training set by manually inspecting nearest neighbors computed over the black-and-white glyphs.

2.4.1 Image Translation Baseline

To illustrate the significant quality improvement of our end-to-end approach, we have implemented a baseline image-to-image translation network (Isola *et al.*, 2016) for this task. In this baseline approach, we consider channel-wise letters in input and output stacks with dimensions $B \times 78 \times 64 \times 64$, where B stands for training batch size and 78 corresponds to the 26 RGB channels. The input stack is given with "observed" color letters while all letters are generated in the output stack. We train this network on our color font data set where we have applied randomly chosen color gradients on each gray-scale font. Feeding in a random subset of RGB letters of an arbitrary font into this model during test time, it is expected to generate stylistically similar 26 letters. Results of this model are shown in the second rows of Figure 5.4 for each example font. Observe that while the network has learned rough

[1] http://www6.flamingtext.com/All-Logos

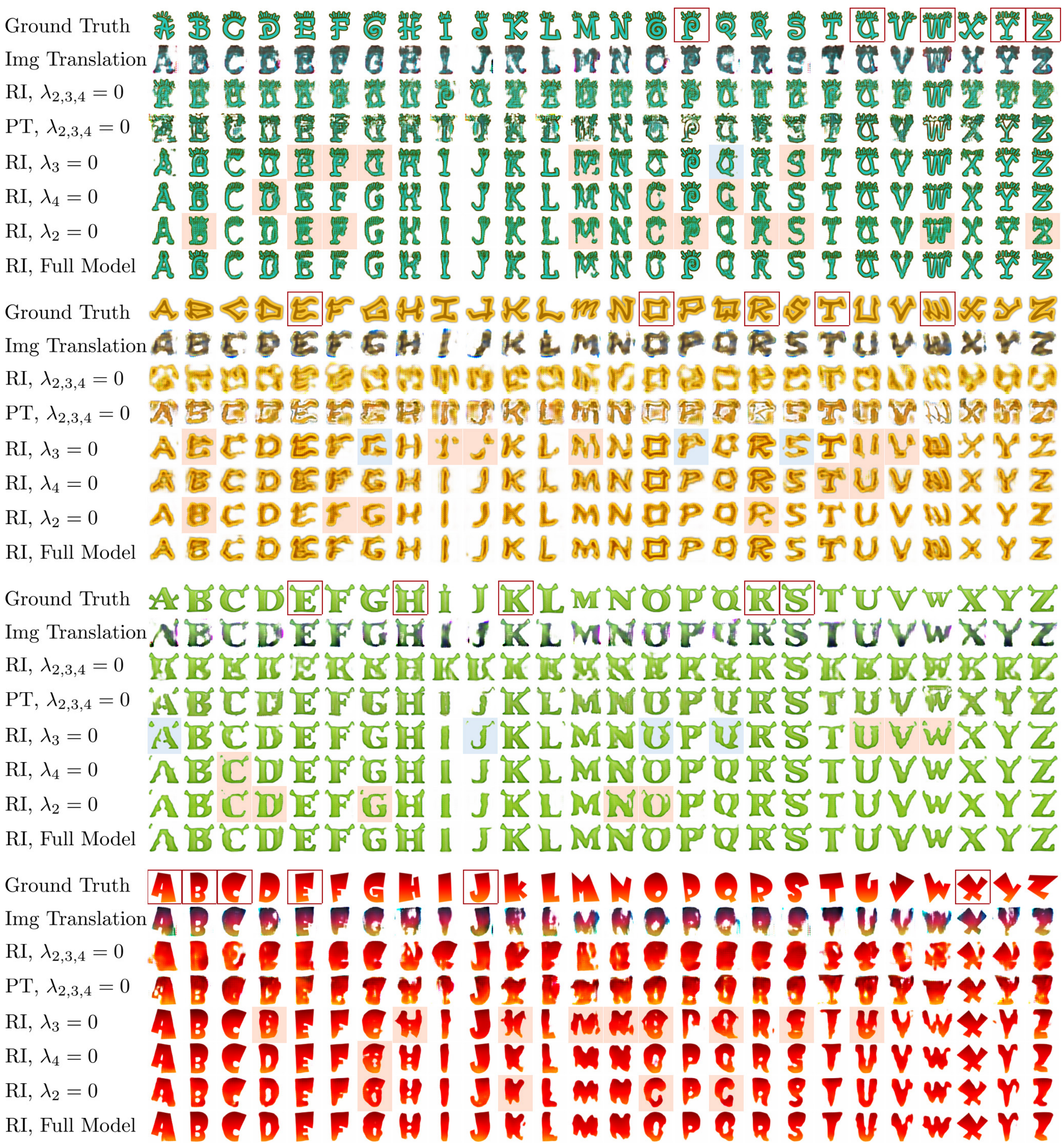


Figure 2.5: Ablation study on our MC-GAN model components: For each exemplar font, we show ground truth (**1st row**), observed letters (red squares in the 1st row), predictions of a baseline image translation network (**2nd row**), predictions of our end-to-end model with randomly initialized (RI) OrnaNet and $\lambda_2 = \lambda_3 = \lambda_4 = 0$ (**3rd row**), with pre-trained (PT) OrnaNet weights and $\lambda_2 = \lambda_3 = \lambda_4 = 0$ (**4th row**), selectively disabled loss terms (**rows 5-7**), and the full end-to-end MC-GAN model (**bottom row**). Style transfer improvements by λ_3 are highlighted in blue and degradation in the predictions by omitting each individual regularizer is highlighted in red.

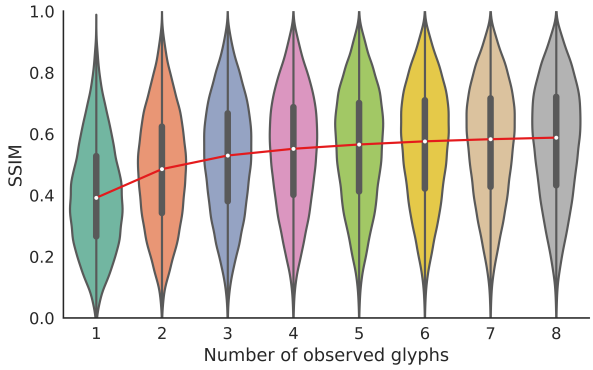


Figure 2.6: Effect of number of observed glyphs on the quality of GlyphNet predictions. Red line is passing through median of each distribution.

aspects of glyph structure, the predictions do not follow a consistent color or ornamentation scheme, as the network is not able to effectively specialize for the provided ornamentation style. Similar artifacts are observed even when evaluating on a test set derived from our simplified color-gradient dataset (see Section 2.4.7).

2.4.2 Ablation Study

In Figure 5.4 we demonstrate the incremental improvement of our proposed regularizers, $\mathcal{L}_{w,L_1}(G_1), \mathcal{L}_{MSE}(G_1)$, and $\mathcal{L}_{MSE}(G_2)$. We found that pre-training on our OrnaNet on gradient-based ornamentations was not helpful, and that the best result comes from a random initialization of OrnaNet and using all the proposed loss terms.

As mentioned in Section 2.2.4, $\mathcal{L}_{w,L_1}(G_1)$ prevents network predictions from going far from the original pre-trained predictions of the GlyphNet. However, it also reduces the freedom in modifying the style of the new glyphs during the end-to-end training. We show this trade-off in the fourth rows of each instance font in Figure 5.4 by highlighting letters with additional artifacts in red and improved letters in blue when this regularizer is excluded from the network. The other two MSE loss regularizers weighted by λ_2 and λ_4 prevent blurry predictions or noisy artifacts to appear on the generated gray-scale and color letters.

2.4.3 Automatic Learning of Correlations between Contents

Automatic learning of the correlations existing between different letters is a key factor in transferring style of the few observed letters in our multi-content GAN. In this section, we study such correlations through the structural similarity (SSIM) metric on a random subset of our 10K font data set consisting of 1500 examples. For each instance, we randomly keep one of the 26 glyphs and generate the rest through our pre-trained GlyphNet.

Computing the structural similarity between each generated glyph and its ground truth, we find 25 distributions over its SSIM scores when a single letter has been observed at a time. In Figure 2.7, we illustrate the distributions $\alpha|\beta$ of generating letter α when letter β is observed (in blue) vs when any other letter rather than β is given (in red). Distributions for the two most informative given letters and the two least informative ones in generating each of the 26 letters are shown in this figure. For instance, looking at the fifth row of the figure, letters F and B are the most constructive in generating letter E compared with other letters while I and W are the least informative ones. As other examples, O and C are the most guiding letters for constructing G as well as R and B for generating P.

2.4.4 Number of Observed Letters

Here, we investigate the dependency of quality of GlyphNet predictions on the number of observed letters. Similar to Section 2.4.3, we use a random subset of our font data set with 1500 example fonts and for each font, we generate 26 letters given n observed ones from our pre-trained GlyphNet. The impact of changing n from 1 to 8 on the distribution of SSIM scores between each unobserved letter and its ground truth is shown in Figure 2.6. The slope of the red line passing through the median of each distribution is decreasing as n increases and reaches to a stable point once the number of observations for each font is close to 6. This study confirms the advantage of our multi-content GAN method in transferring style when we have very few examples.

2.4.5 Perceptual Evaluation

To evaluate the performance of our model, we compare the generated letters of our end-to-end multi-content GAN against the output of the patch-based synthesis method in (Yang *et al.,* 2016). Since this model is designed only for transferring text decorations on clean glyphs, it is not fully comparable with our approach which synthesizes unobserved letters. To explore this method, we use the predictions of our pre-trained GlyphNet as the input to this algorithm. Moreover, this model transfers stylization from only one input decorated glyph, while our method uses all observed examples simultaneously. Therefore, to enable a fair comparison in transferring ornamentations, we allow their model to choose the most similar glyph among the observed instances to the generated glyph mask using a simple image-space distance metric.

We generated the output of both methods on 33 font examples downloaded from the web and asked 11 people to choose which character set they preferred when presented with the observed letters and the full glyph results of both methods. Overall users preferred the result of our method 80.0% of the time. We visualize these examples in Figures 2.10, 2.11, 2.12, 2.13 including ground truth and given letters (first rows), predictions of the text effect transfer method (Yang *et al.,* 2016)which are applied on top of the glyphs synthesized by our GlyphNet (second rows), and predictions of our full end-to-end model in the last rows. The

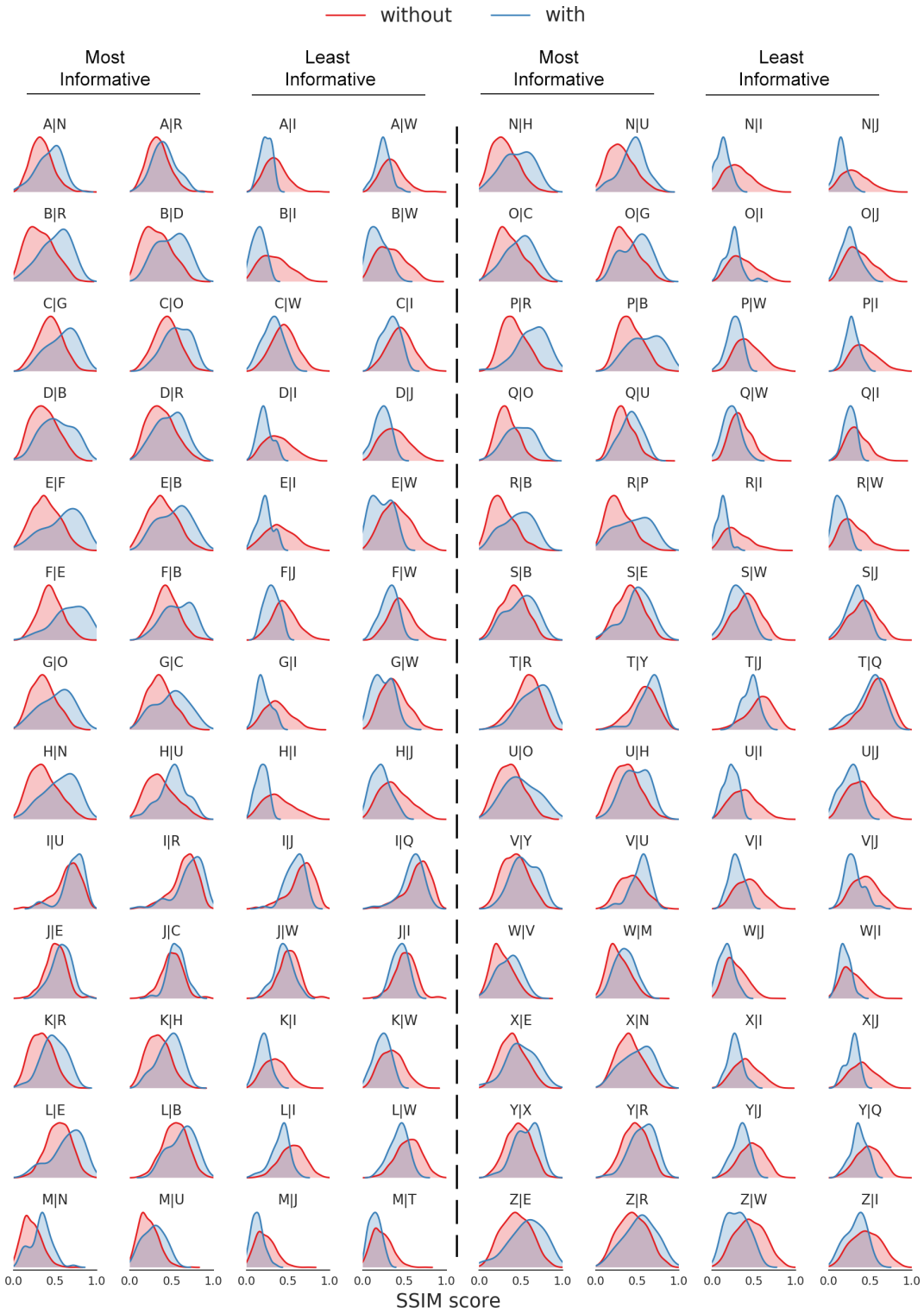


Figure 2.7: Distributions $(\alpha|\beta)$ over SSIM scores for generating letter α given β in blue and given any other letter rather than β in red. Distributions for the most informative given letters β in generating each glyph α is shown in the left of each column while the least informative givens are presented in the right.

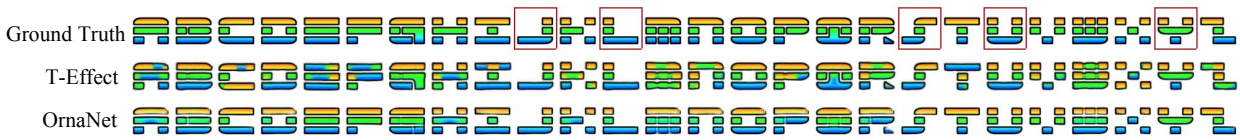


Figure 2.8: Text Effect Transfer (Yang *et al.*, 2016) failure example on clean input glyphs.

Figure 2.9: Failure cases on clean input glyphs.

two best and two worst scoring results for each method are shown on the top and bottom examples of Figure 2.10.

The text effect transfer approach is designed to generate text patterns on clean glyphs but mostly fails to transfer style given our synthesized gray-scale letters. In addition, due to their dependency on a patch matching based algorithm, they often cannot transfer style correctly when the shape of the given and new letters are not very similar (e.g., they cannot transfer straight line patterns when there is a curvature in their new input glyph as clear from the sixth and seventh examples in Figure 2.10).

2.4.6 Ground Truth Glyph Ornamentation

We further compare the performance of our ornamentation network against patch-based synthesis in the case where we are given correct grayscale glyphs (i.e. the ground-truth for GlyphNet). Figure 2.8 indicates a failure mode of patch-based effect transfer, where spatial patterns present in the input are often violated. Figure 2.9 represents a failure mode of both methods: our method averages over the distinct colors present and does not always generate the observed ornamentation such as eyes, while patch-based effect transfer better preserves the input color distrubtion but can still fail to capture the frequency of stylistic elements.

2.4.7 Generalization on Synthetic Color Font Dataset

In this section, we compare our end-to-end multi-content GAN approach with the image translation method discussed in Section 2.4.1. In Figure 2.14, we demonstrate the results on multiple examples from our color font data set where we have applied random color gradients on the gray-scale glyph outlines. By looking at the nearest neighbor examples, we have made sure that the fonts shown in this experiment were not used during training of our Glyph Network.

Given a subset of color letters in the input stack of GlyphNet with dimension $1 \times 78 \times 64 \times 64$ including RGB channels, we generate all 26 RGB letters from the pre-trained GlyphNet on our color font data set. Results are denoted as "Image Translation" in Figure 2.14. Our MC-GAN results are outputs of our end-to-end model fine-tuned on each exemplar font. The image translation method cannot generalize well in transferring these gradient colors at test time by observing only a few examples although other similar random patterns have been seen during training.

2.5 Related Work

Font glyph synthesis from few examples has been a long-studied problem. Earlier methods (Suveeranont & Igarashi, 2010; Campbell & Kautz, 2014; Phan *et al.,* 2015) mostly relied on explicit shape modeling to construct the transformation between existing and novel glyphs. Glyph part models for radicals (Zhou *et al.,* 2011) and strokes (Lian *et al.,* 2016) were designed specifically for Chinese characters. Based on a shape representation, machine learning techniques, including statistical models (Phan *et al.,* 2015) and bilinear factorization (Tenenbaum & Freeman, 1997), have been used to infer and transfer stroke styles and composition rules. More recently, with the rise of deep learning, convolutional neural networks have also been applied to novel glyph synthesis. Promising results were obtained with conventional model structures (Baluja, 2016; Upchurch *et al.,* 2016) as well as generative adversarial networks (GANs) (Lyu *et al.,* 2017; Chang & Gu, 2017). All these networks only predict glyph shape, a goal also targeted by our glyph network. We adopt a distinct multi-content representation in our glyph network which proves to effectively capture the common style among multiple glyphs.

Transferring artistic styles of color and texture to new glyphs is a challenging problem distinct from inferring the overall glyph shape. The problem was investigated in (Yang *et al.,* 2016) with the assumption that the unstylized glyph shape is given. A patch-based texture synthesis algorithm is employed to map sub-effect patterns to correlated positions on text skeleton for effect generation. Style transfer has been more actively studied on general images with the aid of convolutional neural networks (CNNs). CNN features are successfully used to represent image styles, and serve as the basis for optimization (Gatys *et al.,* 2016; Li & Wand, 2016a; Liao *et al.,* 2017). Recently, networks trained with feed-forward structure and adversarial loss have achieved much improved efficiency (Li & Wand, 2016b; Johnson *et al.,* 2016) and generalization ability (Huang & Belongie, 2017; Li *et al.,* 2017b). Our proposed ornamentation network is the first to employ deep networks for text effect transfer.

Several problems in graphics and vision require synthesizing data that is consistent with partial observations. These methods typically focus on learning domain-specific priors to accomplish this task. For example, given a single-view image, encoder-decoder architectures have been proposed to hallucinate novel views of faces (Kulkarni *et al.,* 2015; Tran *et al.,* 2017), bodies (Zhao *et al.,* 2017), and other rigid objects (Zhou *et al.,* 2016; Park *et al.,* 2017). CNNs were also used to complete missing regions in images (Pathak *et al.,* 2016)

and new stereo and lightfield views (Flynn *et al.,* 2016; Kalantari *et al.,* 2016) given a set of input images. Similarly, 3D models can be completed from a partial 3D shape (Dai *et al.,* 2017; Sung *et al.,* 2015). Our problem is different since different glyphs in the same font share the same style, but not structure (unlike one object under different viewpoints). Various geometry modeling techniques have been proposed for learning structural priors from example 3D shapes (Huang *et al.,* 2015; Kalogerakis *et al.,* 2012) and transferring style from a few examples to an input model (Lun *et al.,* 2016). Font data provides a cleaner factorization of style and content that we leverage in our approach.

2.6 Discussion

We propose the first end-to-end approach to synthesizing ornamented glyphs from a few examples. Our method takes a few example images as an input stack and predicts coarse shape and fine ornamentations for the remaining glyphs. We train two networks: one for the shape and one for the texture, and demonstrate that by training them jointly, we can produce results that are strongly preferred by users over existing texture transfer approaches that focus on glyphs. A surprising discovery of this work is that one can efficiently leverage GANs to address a multi-content style transfer problem. In many practical settings, however, fonts need to be generated at extremely high resolution, motivating extensions to this approach such as hierarchical generation or directly synthesizing smooth vector graphics. In the future, we would also like to explore other problems where content has to be stylized consistently from a few examples. For example, modifying a particular human face (style) to have a specific expression (content), consistent stylization of shapes such as emoticons, or transferring materials to consistent sets of objects such as clothing or furniture.

Figure 2.10: Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.*, 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results.

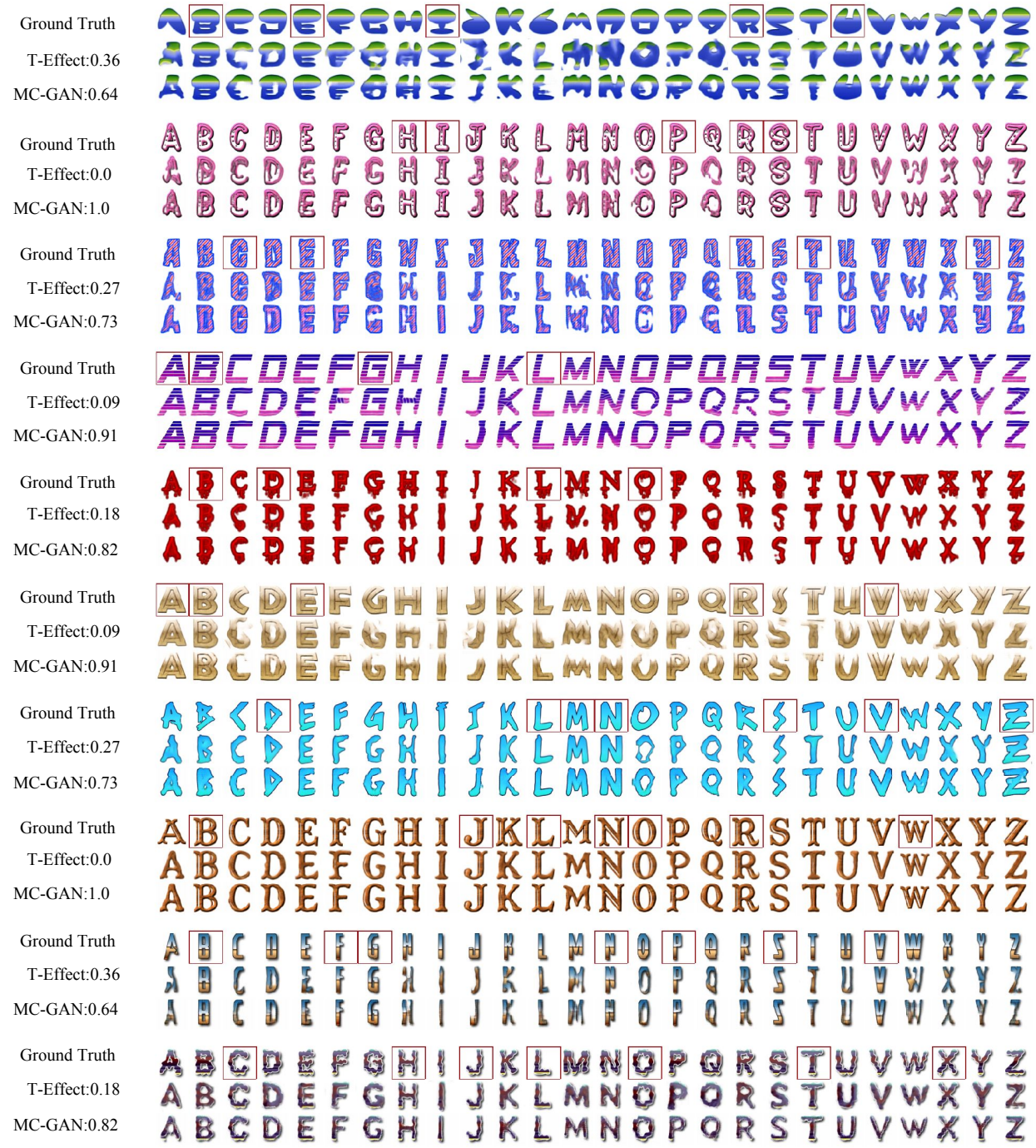


Figure 2.11: Continue - Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.*, 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results.

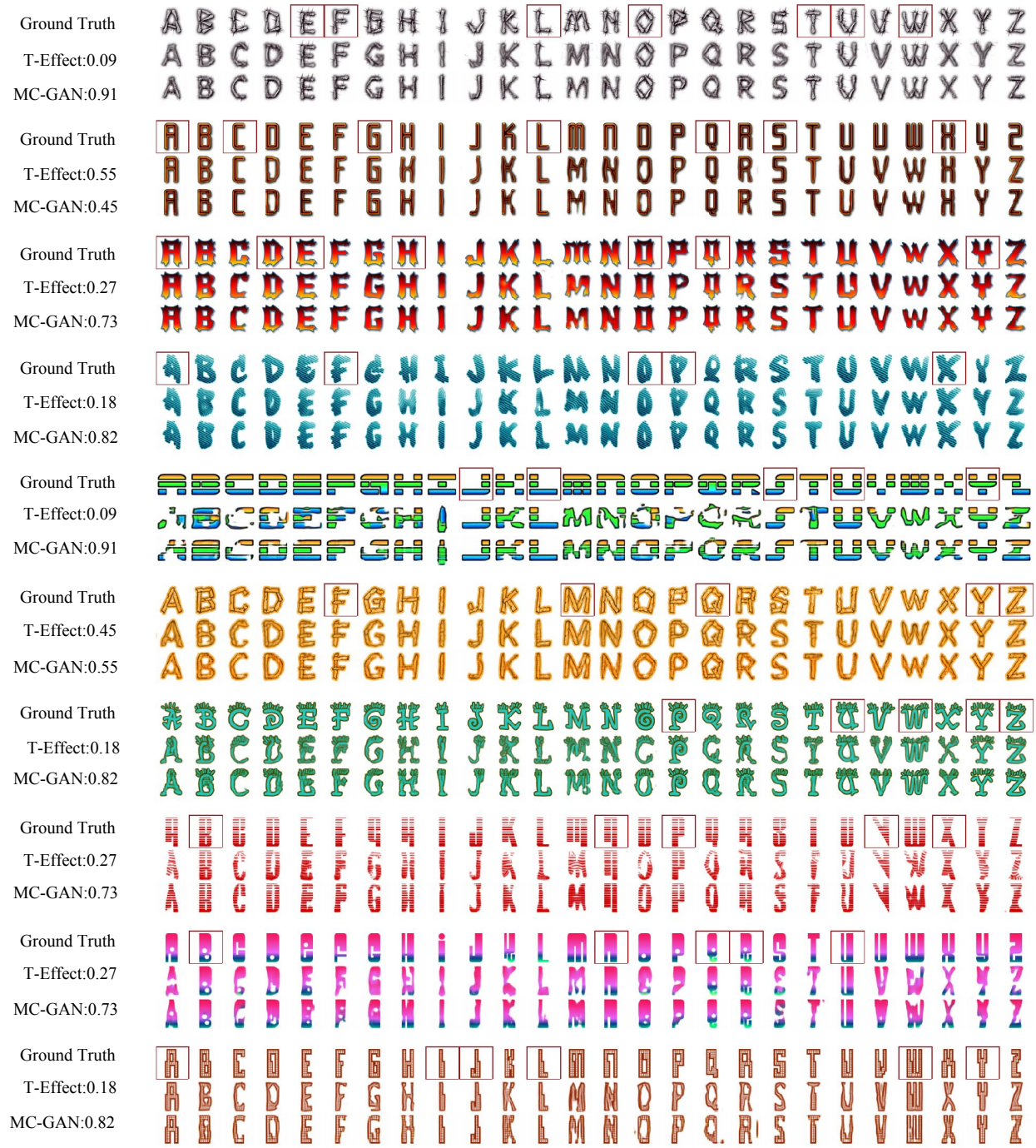


Figure 2.12: Continue - Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.*, 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results.

Figure 2.13: Continue - Comparison of our end-to-end MC-GAN model (**3rd rows**) with the text effect transfer approach (Yang *et al.*, 2016) using GlyphNet synthesized glyphs (**2nd rows**). Ground truth glyphs and the observed subset are illustrated in the **1st row** of each example font. Scores next to each example reveal the percentage of people who preferred the given results.

Figure 2.14: Comparison between image translation and our end-to-end multi-content GAN on our synthetic color font data set. For each example, ground truth and given letters are shown in the **1st row**, image translation outputs in the **2nd row** and MC-GAN in the **last row**.

Chapter 3

Unconditional Synthesis of Complex Scenes

3.1 Introduction

Significant strides have been made on generative models for image synthesis, with a variety of methods based on Generative Adversarial Networks (GANs) (Goodfellow *et al.,* 2014b) achieving state-of-the-art performance. At lower resolutions or in specialized domains, GAN-based methods are able to synthesize samples which are near-indistinguishable from real samples (Brock *et al.,* 2019). However, generating complex, high-resolution scenes from scratch remains a challenging problem, as shown in Figure 4.1-(a) and (b). As image resolution and complexity increase, the coherence of synthesized images decreases — samples lack consistent local or global structures.

Stochastic decoder-based models, such as conditional GANs, were recently proposed to alleviate some of these issues. In particular, both Pix2PixHD (Wang *et al.,* 2018) and SPADE (Park *et al.,* 2019) are able to synthesize high-quality scenes using a strong conditioning mechanism based on semantic segmentation labels during the scene generation process. Global structure encoded in the segmentation layout of the scene is what allows these models to focus primarily on generating convincing local content consistent with that structure.

A key practical drawback of such conditional models is that they require full segmentation layouts as input. Thus, unlike unconditional generative approaches which synthesize images from randomly sampled noise, these models are limited to generating images from a set of scenes that is prescribed in advance, typically either through segmentation labels from an existing dataset, or scenes that are hand-crafted by experts.

Contributions To overcome these limitations, we propose a new model, the Semantic Bottleneck GAN (SB-GAN), which couples high-fidelity generation capabilities of label-conditional models with the flexibility of unconditional image generation. This in turn

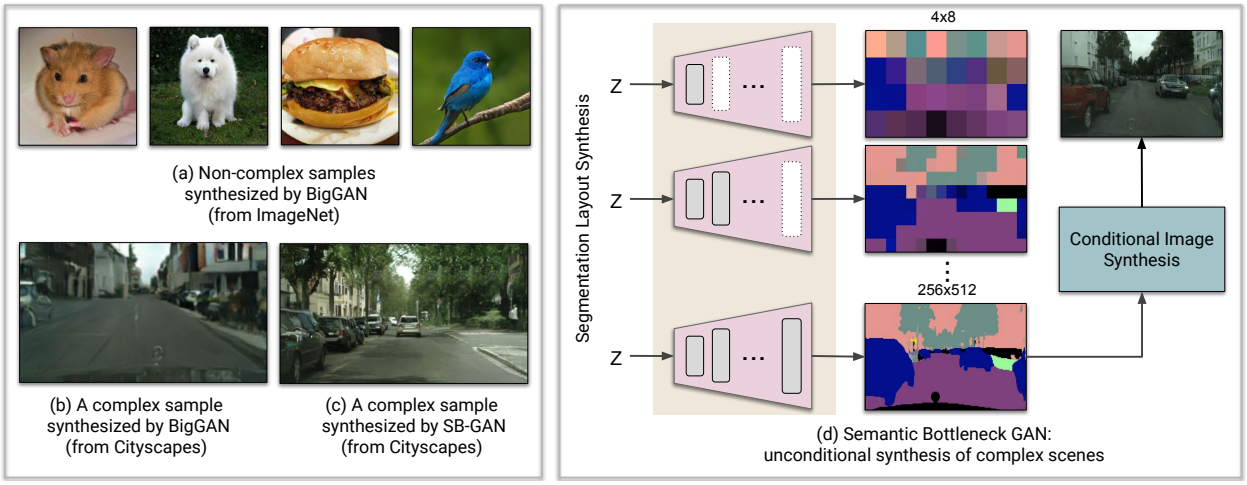


Figure 3.1: (a) Examples of non-complex images from ImageNet synthesized by the state-of-the-art BigGAN model (Brock *et al.*, 2019). Although these samples look decent, the complex scenes synthesized by BigGAN (e.g., from the Cityscapes dataset) are blurry and defective in local structure (e.g., cars are blended together) (b). Zoom in for more detail. (c) A complex scene synthesized by our model respects both local and global structural integrity of the scene. (d) Schematic of our unconditional Semantic Bottleneck GAN. We progressively train the adversarial segmentation synthesis network to generate realistic segmentation maps from scratch, then synthesize a photo-realistic image using a conditional image synthesis network. End-to-end coupling of these two components results in state-of-the-art unconditional synthesis of complex scenes.

enables our model to synthesize an unlimited number of novel complex scenes, while still maintaining high-fidelity output characteristic of image-conditional models [1].

Our SB-GAN first unconditionally generates a pixel-wise semantic label map of a scene (i.e. for each spatial location it outputs a class label), and then generates a realistic scene image by conditioning on that semantic map, Figure 4.1-(d). By factorizing the task into these two steps, we are able to separately tackle the problems of producing convincing segmentation layouts (i.e. a useful global structure) and filling these layouts with convincing appearances (i.e. local structure). When trained end-to-end, the model yields samples which have a coherent global structure as well as fine local details, e.g., Figure 4.1-(c). Empirical evaluation shows that our Semantic Bottleneck GAN achieves a new state-of-the-art on two complex datasets with relatively small number of training images, Cityscapes and ADE-Indoor, as measured both by the Fréchet Inception Distance (FID) and by perceptual evaluations. Additionally, we observe that the conditional segmentation-to-image synthesis component of our SB-GAN jointly trained with segmentation layout synthesis significantly

[1]This chapter is based on joint work done with Michael Tschannen, Eric Tzeng, Sylvain Gelly, Trevor Darrell, and Mario Lucic (Azadi *et al.*, 2019d) presented at Synthetic Data Generation Workshop at ICLR 2021.

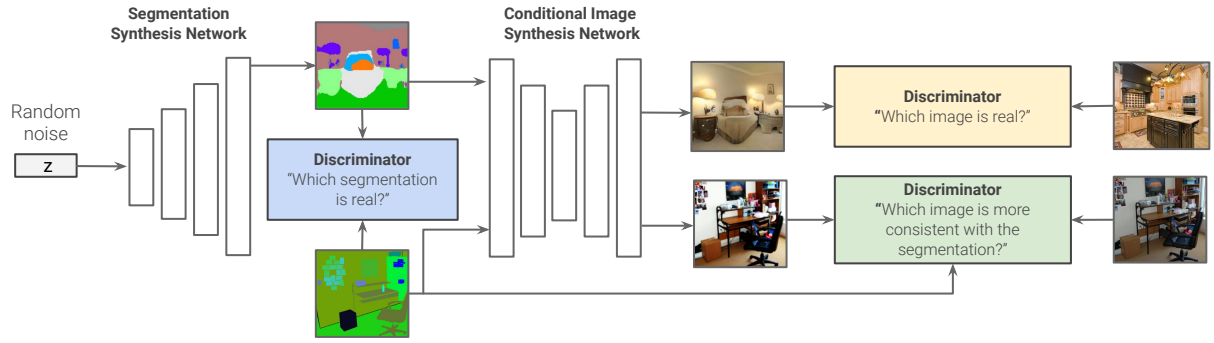


Figure 3.2: Schematic of Semantic Bottleneck GAN. Starting from random noise, we synthesize a segmentation layout and use a discriminator to bias the segmentation synthesis network towards realistic looking segmentation layouts. The generated layout is then provided as input to a conditional image synthesis network to synthesize the final image. A second discriminator is used to bias the conditional image synthesis network towards realistic images paired with real segmentation layouts. Finally, a third unconditional discriminator is used to bias the conditional image synthesis network towards generating images that match the real image distribution.

improves the state-of-the-art semantic image synthesis network (Park *et al.*, 2019), resulting in higher-quality outputs when conditioning on ground truth segmentation layouts.

Key Challenges While both unconditional generation and image-to-image translation are well-explored learning problems, fully unconditional generation of the segmentation maps is a notoriously hard task: (i) Semantic categories do not respect any ordering relationships and the network is therefore required to capture the intricate relationship between segmentation classes, their shapes, and their spatial dependencies. (ii) As opposed to RGB values, semantic categories are discrete, hence non-differentiable which poses a challenge for end-to-end training (Sec. 3.2.2) (iii) Naively combining state-of-the-art unconditional generation and image-to-image translation models leads to poor performance. However, by carefully designing an additional discriminator component and a corresponding training protocol, we not only manage to improve the performance of the end-to-end model, but also the performance of each component separately (Sec. 3.2.3).

We emphasize that despite these challenges our approach scales to 256×256 resolution and 95 semantic categories, whereas existing state-of-the-art GAN models directly generating RGB images at that resolution already suffer from considerable instability (Sec. 3.3).

3.2 Semantic Bottleneck GAN (SB-GAN)

We propose an unconditional Semantic Bottleneck GAN architecture to learn the distribution of complex scenes. To tackle the problems of learning both the global layout and the

local structure, we divide this synthesis problem into two parts: an unconditional segmentation map synthesis network and a conditional segmentation-to-image synthesis model. Our first network is designed to coarsely learn the scene distribution by synthesizing semantic layouts. It generates per-pixel semantic categories following the progressive GAN model architecture (ProGAN) (Karras *et al.,* 2017b). This fully unconditional generation of the segmentation maps is novel, very challenging, and a careful design is crucial, as described in Section 3.2.1. The second network populates the synthesized semantic layouts with texture by predicting RGB pixel values using Spatially-Adaptive Normalization (SPADE), following the architecture of the state-of-the-art semantic synthesis network in (Park *et al.,* 2019). We assume the ground truth segmentation masks are available for all or part of the target scene dataset. In the following sections, we will first discuss our semantic bottleneck synthesis pipeline and summarize the SPADE network for image synthesis. We will then couple these two networks in an end-to-end design which we refer to as Semantic Bottleneck GAN (SB-GAN).

3.2.1 Semantic bottleneck synthesis

Our goal is to learn a (coarse) estimate of the scene distribution from samples corresponding to real segmentation maps with K semantic categories. Starting from random noise, we generate a tensor $Y \in 1, K^{N\times 1\times H\times W}$ which represents a per-pixel segmentation class, with H and W indicating the height and width, respectively, of the generated map and N the batch size. In practice, we progressively train from a low to a high resolution using the ProGAN architecture (Karras *et al.,* 2017b) coupled with the Improved WGAN loss function (Gulrajani *et al.,* 2017a) on the ground truth discrete-valued segmentation maps, illustrated in Figure 4.1-(d). Similar to ProGAN, to increase the spatial resolution of the generated segmentation maps during training, we incrementally add layers to the generator and the discriminator. In contrast to ProGAN, in which the generator outputs continuous RGB values, we predict per-pixel discrete semantic class labels. This task is extremely challenging as it requires the network to capture the intricate relationship between segmentation classes and their spatial dependencies. To this end, we apply the Gumbel-softmax trick (Jang *et al.,* 2017; Maddison *et al.,* 2016) coupled with a straight-through estimator (Jang *et al.,* 2017), described in detail below.

We synthesize segmentation layouts by first generating per-pixel probability scores of belonging to each of the K semantic classes and then sampling a semantic class per pixel. The per-pixel probability scores are computed by applying a softmax function to the last layer of the generator (i.e. logits) which results in probability maps $P^{ij} \in [0, 1]^K$, with $\sum_{k=1}^{K} P_k^{ij} = 1$ for each spatial location $(i, j) \in 1, H \times 1, W$. To sample a semantic class from this multinomial distribution, we would ideally apply the following well-known procedure at each spatial location: (1) sample k i.i.d. samples, G_k, from the standard Gumbel distribution, (2) add these samples to each logit, and (3) take the index of the maximal value. This reparametrization indeed allows for an efficient forward-pass, but is not differentiable. Nevertheless, the `max` can be replaced with the `softmax` function and the quality of the approximation can be

controlled by varying the *temperature hyperparameter* τ — the smaller the τ, the closer the approximation is to the categorical distribution (Jang *et al.*, 2017):

$$S_k^{ij} = \frac{\exp\{(\log P_k^{ij} + G_k)/\tau\}}{\sum_{i=1}^{K} \exp\{(\log P_i^{ij} + G_i)/\tau\}}. \tag{3.1}$$

Similar to the real samples, the synthesized samples fed to the GAN discriminator should still contain *discrete* category labels. As a result, for the forward pass, we compute $\arg\max_k S_k$, while for the backward pass, we use the soft predicted scores S_k directly, a strategy known as straight-through estimation (Jang *et al.*, 2017).

3.2.2 Semantic image synthesis

Our second sub-network converts the synthesized semantic layouts into photo-realistic images using spatially-adaptive normalization (Park *et al.*, 2019). The segmentation masks are employed to spread the semantic information throughout the generator by modulating the activations with a spatially adaptive learned transformation. We follow the same generator and discriminator architectures and loss functions used in (Park *et al.*, 2019), where the generator contains a series of SPADE residual blocks with upsampling layers. The loss functions to train SPADE are summarized as:

$$\begin{aligned} L_{D_{\text{SPD}}} &= -\mathbb{E}_{y,x}[\min(0, -1 + D_{\text{SPD}}(y, x))] - \mathbb{E}_y[\min(0, -1 - D_{\text{SPD}}(y, G_{\text{SPD}}(y)))] \\ L_{G_{\text{SPD}}} &= -\mathbb{E}_y[D_{\text{SPD}}(y, G_{\text{SPD}}(y)))] + \lambda_1 L_1^{\text{VGG}} + \lambda_2 L_1^{\text{Feat}}, \end{aligned} \tag{3.2}$$

where G_{SPD}, D_{SPD} stand for the SPADE generator and discriminator, and L_1^{VGG} and L_1^{Feat} represent the VGG and discriminator feature matching L_1 loss functions, respectively (Park *et al.*, 2019; Wang *et al.*, 2018). We pre-train this network using pairs of real RGB images, x, and their corresponding real segmentation masks, y, from the target scene data set.

In the next section, we will describe how to employ the synthesized segmentation masks in an end-to-end manner to improve the performance of both the semantic bottleneck and the semantic image synthesis sub-networks.

3.2.3 End-to-end framework

After training semantic bottleneck synthesis model to synthesize segmentation masks and the semantic image synthesis model to stochastically map segmentations to photo-realistic images, we adversarially fine-tune the parameters of both networks in an end-to-end approach by introducing an unconditional discriminator network on top of the SPADE generator (see Figure 3.2).

This second discriminator, D_2, has the same architecture as the SPADE discriminator, but is designed to distinguish between real RGB images and the fake ones generated from the *synthesized* semantic layouts. Unlike the SPADE conditional GAN loss, which examines

pairs of input segmentations and output images, (y, x) in equation 3.2, the GAN loss on D_2, L_{D_2}, is unconditional and only compares real images to synthesized ones, as shown in equation 3.3:

$$\begin{aligned} L_{D_2} &= -\mathbb{E}_x[\min(0, -1 + D_2(x))] - \mathbb{E}_z[\min(0, -1 - D_2(G(z)))] \\ L_G &= -\mathbb{E}_z[D_2(G(z))] + L_{G_{\text{SPD}}} + \lambda L_{G_{\text{SB}}}, \quad G(z) = G_{\text{SPD}}(G_{\text{SB}}(z)) \end{aligned} \tag{3.3}$$

where G_{SB} represents the semantic bottleneck synthesis generator, and $L_{G_{\text{SB}}}$ is the improved WGAN loss to pretrain G_{SB} described in Section 3.2.1. In contrast to the conditional discriminator in SPADE, which enforces consistency between the input semantic map and the output image, D_2 is primarily concerned with the overall quality of the final output. The hyper parameter λ determines the ratio between the two generators during fine-tuning. The parameters of both generators, G_{SB} and G_{SPD}, as well as the corresponding discriminators, D_{SB} and D_{SPD}, are updated in this end-to-end fine-tuning.

We illustrate our final end-to-end network in Figure 3.2. Jointly fine-tuning the two networks in an end-to-end fashion allows the two networks to reinforce each other, leading to improved performance. The gradients with respect to RGB images synthesized by SPADE are back-propagated to the segmentation synthesis model, thereby encouraging it to synthesize segmentation layouts that lead to higher quality final images. Hence, SPADE plays the role of a loss function for synthesizing segmentations, but in the RGB space, hence providing a goal that was absent from the initial training. Similarly, fine-tuning SPADE with synthesized segmentations allows it to adapt to a more diverse set of scene layouts, which improves the quality of generated samples.

3.3 Experiments and Results

We evaluate the performance of the proposed approach on two datasets containing images with complex scenes, where the ground truth segmentation masks are available during training (possibly only for a subset of the images). We also study the role of the two network components, semantic bottleneck and semantic image synthesis, on the final result. We compare the performance of SB-GAN against the state-of-the-art BigGAN model (Brock *et al.*, 2019) as well as a ProGAN (Karras *et al.*, 2017b) baseline that has been trained on the RGB images directly. We evaluate our method using Fréchet Inception Distance (FID) as well as a perceptual evaluation.

Datasets We study the performance of our model on the Cityscapes and ADE-indoor datasets as the two domains with complex scene images.

- Cityscapes-5K (Cordts *et al.*, 2016a) contains street scene images in German cities with training and validation set sizes of 3,000 and 500 images, respectively. Ground truth segmentation masks with 33 semantic classes are available for all images in this dataset.

Figure 3.3: Images synthesized on Cityscapes-5K. Best viewed on screen; zoom in for more detail. Although both models capture the general scene layout, SB-GAN (1st row) generates more convincing objects, e.g. buildings and cars.

Figure 3.4: Images synthesized on Cityscapes-25K. Best viewed on screen; zoom in for more detail. Images synthesized by BigGAN (3rd row) are blurry and sometimes defective in local structures.

- Cityscapes-25K (Cordts *et al.*, 2016a) contains street scene images in German cities with training and validation set sizes of 23,000 and 500 images, respectively with 19 semantic classes. Cityscapes-5K is a subset of this dataset, providing 3,000 images in the training set here as well as the entire validation set. Fine ground truth annotations are only provided for this subset, with the remaining 20,000 training images containing only coarse annotations. We extract the corresponding fine annotations for the rest of training images using the state-of-the-art segmentation model (Yu *et al.*, 2017a) trained on the training annotated samples from Cityscapes-5K.

- ADE-Indoor is a subset of the ADE20K dataset (Zhou *et al.*, 2017) containing 4,377 challenging training images from indoor scenes and 433 validation images with 95 semantic categories.

Figure 3.5: Images synthesized on ADE-Indoor. This dataset is very challenging, causing mode collapse for the BigGAN model (3rd row). In contrast, samples generated by SB-GAN (1st row) are generally of higher quality and much more structured than those of ProGAN (2nd row).

Evaluation We use the Fréchet Inception Distance (FID) (Heusel *et al.*, 2017) as well as a perceptual evaluation of the quality of the generated samples. To compute FID, the real data and generated samples are embedded in a specific layer of a pre-trained Inception network. Then, a multivariate Gaussian is fit to the data, and the distance is computed as $(x, g) = ||\mu_x - \mu_g||_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{\frac{1}{2}})$, where μ and Σ denote the empirical mean and covariance, and subscripts x and g denote the real and generated data respectively. FID is sensitive to both the addition of spurious modes and to mode dropping (Sajjadi *et al.*, 2018; Lucic *et al.*, 2018). On the Cityscapes dataset, we ran five trials where we computed FID on 500 random synthetic images and 500 real validation images, and report the average score. On ADE-Indoor, this is repeated on batches of 433 images.

Implementation details In all our experiments, we set $\lambda_1 = \lambda_2 = 10$, and $\lambda = 10$. The initial generator and discriminator learning rates for training SPADE both in the pretraining and end-to-end steps are 10^{-4} and $4 \cdot 10^{-4}$, respectively. The learning rate for the semantic bottleneck synthesis sub-network is set to 10^{-3} in the pretraining step and to 10^{-5} in the end-to-end fine-tuning on Cityscapes, and to 10^{-4} for ADE-Indoor. The temperature hyperparameter, τ, is always set to 1. For BigGAN, we followed the setup by Lucic *et al.*

Table 3.1: FID of the synthesized samples (lower is better), averaged over 5 random sets of samples. Images were synthesized at resolution of 256x512 on Cityscapes and 256x256 on ADE-Indoor.

	ProGAN	SB-GAN w/o FT	SB-GAN
Cityscapes-5k	92.57	83.20	**65.49**
Cityscapes-25k	63.87	71.13	**62.97**
ADE-Indoor	104.83	91.80	**85.27**

Table 3.2: FID of the synthesized samples (lower is better), averaged over 5 random sets of samples. Images were synthesized at resolution of 128x256 on Cityscapes and 128x128 on ADE-Indoor.

	ProGAN	BigGAN	SB-GAN
Cityscapes-5k	178.19	-	**57.48**
Cityscapes-25k	56.7	64.82	**54.92**
ADE-Indoor	85.94	156.65	**81.39**

(2019)[2], where we modified the code to allow for non-square images of Cityscapes. We used one class label for all images to have an unconditional BigGAN model. For both datasets, we varied the batch size (using values in $\{128, 256, 512, 2048\}$), learning rate, and location of the self-attention block. We trained the final model for 50K iterations.

3.3.1 Qualitative results

In Figures 3.3, 3.4, and 3.5, we provide qualitative comparisons of the competing methods on the three aforementioned datasets. We observe that both Cityscapes-5K and ADE-Indoor are very challenging for the state-of-the-art ProGAN and BigGAN models, likely due to the complexity of the data and small number of training instances. Even at a resolution of 128×128 on the ADE-Indoor dataset, BigGAN suffers from mode collapse, as illustrated in Figure 3.5. In contrast, SB-GAN significantly improves the structure of the scene distribution and provides samples of higher quality. On Cityscapes-25K, the performance improvement of SB-GAN is more modest due to the large number of training images available. It is worth emphasizing that in this case only 3K ground truth segmentations are available to train SB-GAN. Compared to BigGAN, images synthesized by SB-GAN are sharper and contain more

[2]Configuration as in https://github.com/google/compare_gan/blob/master/example_configs/biggan_imagenet128.gin

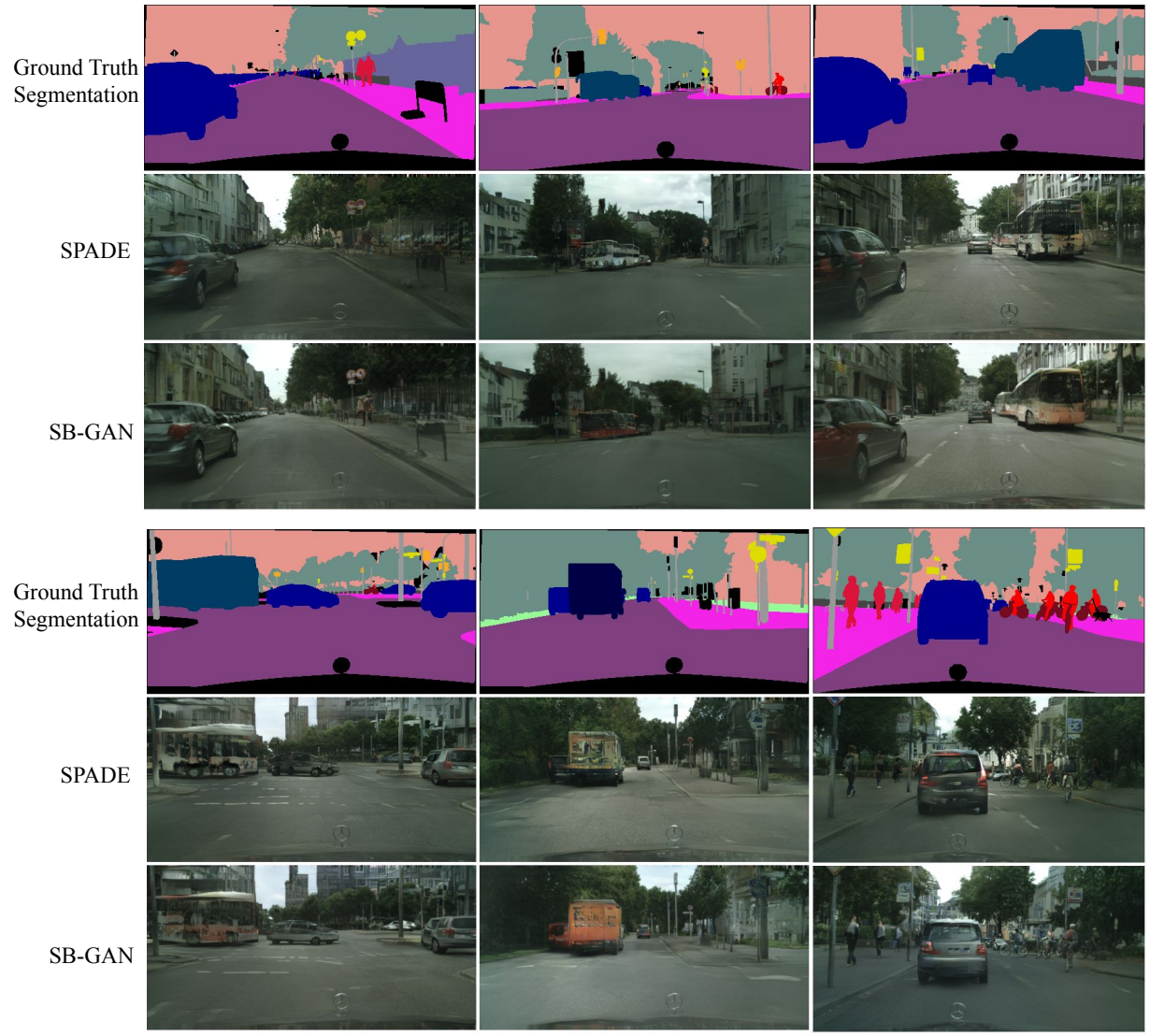


Figure 3.6: The effect of SB-GAN on improving the performance of the state-of-the-art semantic image synthesis model (SPADE) on ground truth segmentations of Cityscapes-25K validation set. For SB-GAN, we train the entire model end-to-end, extract the trained SPADE sub-network, and synthesize samples conditioned on the ground truth labels.

structural details (e.g., one can zoom-in on the synthesized cars). In Figures 3.11, 3.12, 3.13, and 3.14, we show additional synthetic results from our proposed SB-GAN model including both the synthesized segmentations and their corresponding synthesized images from the Cityscapes-25K and ADE-Indoor datasets.

3.3.2 Quantitative evaluation

To provide a thorough empirical evaluation of the proposed approach, we generate samples for each dataset and report the FID scores of the resulting images (averaged across 5 sets of generated samples). We evaluate SB-GAN both before and after end-to-end fine-tuning, and compare our method to two strong baselines, ProGAN (Karras *et al.,* 2017b) and BigGAN (Brock *et al.,* 2019).

The results are detailed in Tables 3.1 and 3.2. First, in the low-data regime, even without fine-tuning, our Semantic Bottleneck GAN produces higher quality samples and significantly outperforms the baselines on Cityscapes-5K and ADE-Indoor. The advantage of our proposed method is even more striking on smaller datasets. While competing methods are unable to learn a high-quality model of the underlying distribution without having access to a large number of samples, SB-GAN is less sensitive to the number of training data points. Secondly, we observe that by jointly training the two components, SB-GAN produces state-of-the-art results across all three datasets.

We were not able to successfully train BigGAN at a resolution of 256×512 due to instability observed during training and mode collapse. Table 3.2 shows the results for a lower-resolution setting, for which we were able to successfully train BigGAN. We report the results before the training collapses. BigGAN is, to a certain extent, able to capture the distribution of Cityscapes-25K, but fails completely on ADE-Indoor. Interestingly, BigGAN fails to capture the distribution of Cityscapes-5K even at 128×128 resolution.

Generating by conditioning on real segmentations To independently assess the impact of end-to-end training on the conditional image synthesis sub-network, we evaluate the quality of generated samples when conditioning on ground truth validation segmentations from each dataset. Comparisons to the baseline network SPADE (Park *et al.,* 2019) are provided in Table 3.3 and Figures 3.6 and 3.7. We observe that the image synthesis component of SB-GAN consistently outperforms SPADE across all three datasets, indicating that fine-tuning on data sampled from the segmentation generator improves the conditional image generator.

Table 3.3: FID of the synthesized samples when conditioned on the ground truth labels. For SB-GAN, we train the entire model end-to-end and extract the trained SPADE.

	SPADE	SB-GAN
Cityscapes-5K	72.12	**60.39**
Cityscapes-25K	60.83	**54.13**
ADE-Indoor	50.30	**48.15**

Fine-tuning ablation study To dissect the effect of end-to-end training, we perform a study on different components of SB-GAN. In particular, we consider three settings: (1) SB-GAN before end-to-end fine-tuning, (2) fine-tuning only the semantic bottleneck synthesis

component, (3) fine-tuning only the conditional image synthesis component, and (4) fine-tuning all jointly. The results on the Cityscapes-5K dataset (resolution 128×256) are reported in Table 5.2. Finally, the impact of fine-tuning on the quality of samples can be observed in Figures 3.8 and 3.9.

Table 3.4: Ablation study of various components of SB-GAN. We report FID scores of SB-GAN before fine-tuning, fine-tuning only the semantic bottleneck synthesis component, fine-tuning only the image synthesis component, and full end-to-end fine-tuning. Experiments are performed on the Cityscapes-5K dataset at a resolution of 128×256.

No FT	FT SB	FT SPADE	FT Both
70.15	66.22	63.04	**58.67**

3.3.3 Perceptual evaluation

We used Amazon Mechanical Turk (AMT) to assess the performance of each method on each dataset using ∼600 pairs of (synthesized images, human evaluators) with a total of 200 unique synthesized images. For each image, evaluators were asked to assign a score between 1 to 4 to each image, indicating low-to-high quality images, respectively. The results are summarized in Table 3.5 and are consistent with our FID-based evaluations.

Table 3.5: Average perceptual evaluation scores when each evaluators has selected a quality score in the range of 1 (terrible quality) to 4 (high quality) for each image.

	ProGAN	BigGAN	SB-GAN
CITYSCAPES-5K	2.08	-	**2.48**
CITYSCAPES-25K	2.53	2.27	**2.61**
ADE-INDOOR	2.35	1.96	**2.49**

3.4 Related Work

Generative Adversarial Networks (GANs) Training GANs is notoriously hard and recent efforts focused on improving neural architectures (Wang & Gupta, 2016; Karras *et al.*, 2017b; Zhang *et al.*, 2019; Chen *et al.*, 2019a), loss functions (Arjovsky *et al.*, 2017), regularization (Gulrajani *et al.*, 2017a; Miyato *et al.*, 2018), large-scale training (Brock *et al.*, 2019), self-supervision (Chen *et al.*, 2019b), and sampling (Brock *et al.*, 2019; Azadi *et al.*, 2019a). Improving the performance of GANs by disentangling structure and style has been studied by Wang & Gupta (2016) where structure is represented by a surface normal map and style is the texture mapped onto the structure. Another compelling approach which enables

generation of high-resolution images is based on progressive training: a model is trained to first synthesize lower-resolution images (e.g. 8×8), then the resolution is gradually increased until the desired resolution is achieved (Karras *et al.,* 2017b). Recently, Brock *et al.* (2019) showed that GANs significantly benefit from large-scale training, both in terms of model size and batch size. We note that these models are able to synthesize high-quality images in settings where objects are very prominent and centrally placed or follow some well-defined structure, as the corresponding distribution is easier to capture. In contrast, when the scenes are more complex and the amount of data is limited, the task becomes extremely challenging for these state-of-the-art models. We aim to improve the performance in the context of complex scenes and a small number of training examples by disentangling the image generation problem into learning the structure represented by semantic layouts and filling in the RGB details using a semantic image synthesis model. A similar idea was proposed by a concurrent work (Volokitin *et al.,* 2020) with substantial differences in the model and results.

GANs on discrete domains GANs for discrete domains have been investigated in several works (Yu *et al.,* 2017b; Lin *et al.,* 2017a; Bojchevski *et al.,* 2018; Lu *et al.,* 2018). Training in this domain is even more challenging as the samples from discrete distributions are not differentiable with respect to the network parameters. This problem can be somewhat alleviated by using the Gumbel-softmax distribution, which is a continuous approximation to a multinomial distribution parameterized in terms of the softmax function (Kusner & Hernández-Lobato, 2016). We will show how to apply a similar principle to learn the distribution of discrete segmentation masks.

Conditional image synthesis In conditional image synthesis one aims to generate images by conditioning on an input which can be provided in the form of an image (Isola *et al.,* 2017b; Zhu *et al.,* 2017b; Azadi *et al.,* 2018; Azadi *et al.,* 2019c; Liu *et al.,* 2017), a text phrase (Reed *et al.,* 2016b; Zhang *et al.,* 2017b; Qiao *et al.,* 2019; Ashual & Wolf, 2019; Hong *et al.,* 2018), a scene graph (Johnson *et al.,* 2018; Ashual & Wolf, 2019), a class label, or a semantic layout (Odena *et al.,* 2017; Chen & Koltun, 2017; Wang *et al.,* 2018; Park *et al.,* 2019). These conditional GAN methods learn a mapping that translates samples from the source distribution into samples from the target domain.

The text-to-image synthesis models proposed in (Hong *et al.,* 2018; Li *et al.,* 2019) decompose the synthesis task into multiple steps. Given the text description, a semantic layout is constructed by generating object bounding boxes and refining each box by estimating object shapes. Then, an image is synthesized conditionally on the generated semantic layout from the first step. Our work shares the same high-level idea of decomposing the image generation problem into the semantic layout synthesis and the conditional semantic-layout-to-image synthesis. However, we note that the above approaches, as opposed to ours, are conditional and require supervision in the form of textual descriptions. Secondly, they are sequential in nature and synthesize masks of a few different objects (e.g. person, elephant), but not a fully fine-grained semantic map (e.g. missing sky, grass, etc.). In stark contrast,

our approach unconditionally synthesizes the *full semantic layout* of the entire scene from a noise input in an end-to-end network design. Due to the above distinctions, their segmentation synthesis models differ significantly from ours in terms of architecture and design as shown in Figure 3.10.

3.5 Discussion

We proposed an end-to-end Semantic Bottleneck GAN model that synthesizes semantic layouts from scratch, and then generates photo-realistic scenes conditioned on the synthesized layouts. Through extensive quantitative and qualitative evaluations, we showed that this novel end-to-end training pipeline significantly outperforms the state-of-the-art models in unconditional synthesis of complex scenes. In addition, Semantic Bottleneck GAN strongly improves the performance of the state-of-the-art semantic image synthesis model in synthesizing photo-realistic images from ground truth segmentations. As a future work, one could explore novel ways to train GANs with discrete outputs, especially to deal with the non-differentiable nature of the generated outputs.

Figure 3.7: The effect of SB-GAN on improving the performance of the state-of-the-art semantic image synthesis model (SPADE) on ground truth segmentations of ADE-Indoor validation set. For SB-GAN, we train the entire model end-to-end, extract the trained SPADE sub-network, and synthesize samples conditioned on the ground truth labels.

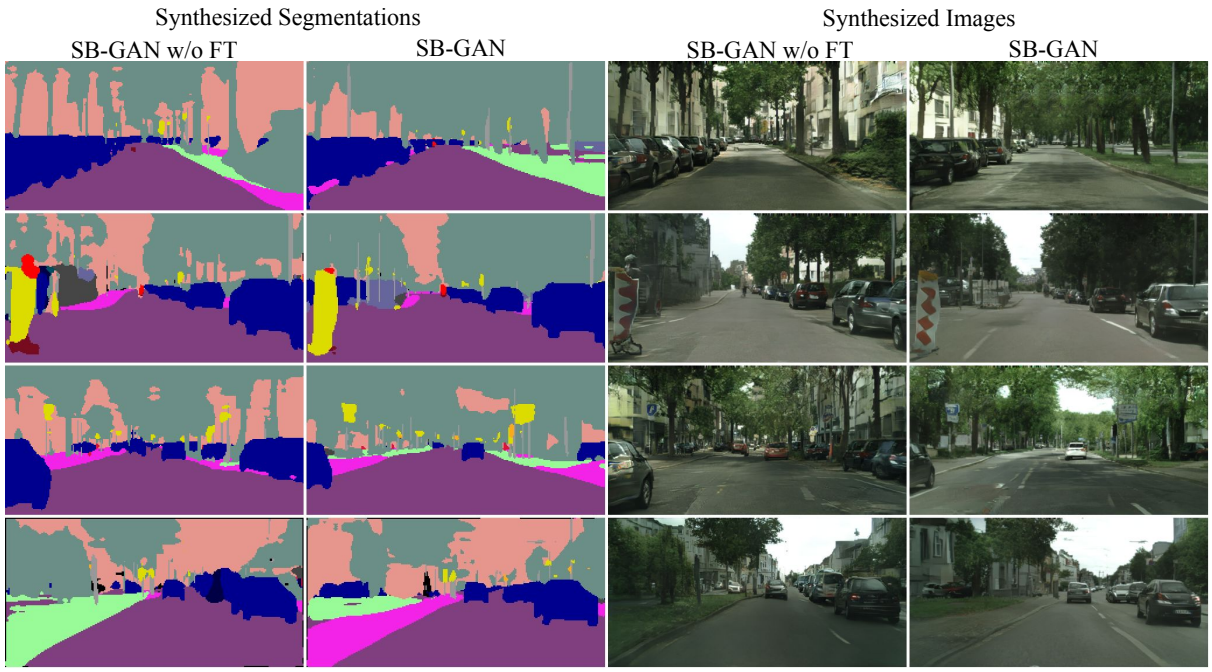


Figure 3.8: The effect of fine-tuning on the baseline setup for the Cityscapes-25K dataset. We observe improvements in both the global structure of the segmentations and the performance of semantic image synthesis, resulting in images of higher quality.

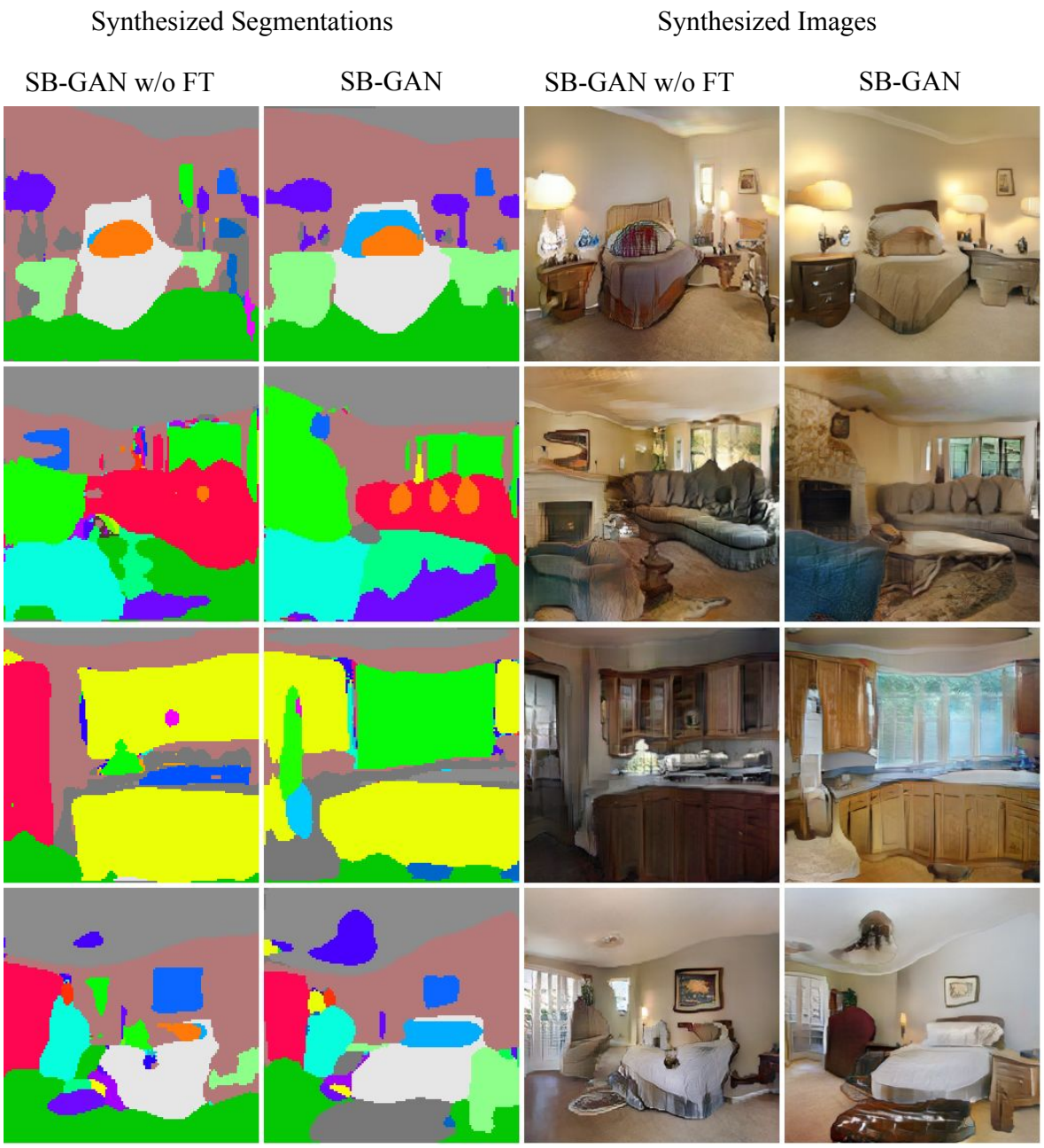


Figure 3.9: The effect of fine-tuning (FT) on the baseline setup for ADE-Indoor dataset. Analogously to the results on Cityscapes-25K, we observe improvements in both the global structure of the segmentations and the performance of semantic image synthesis.

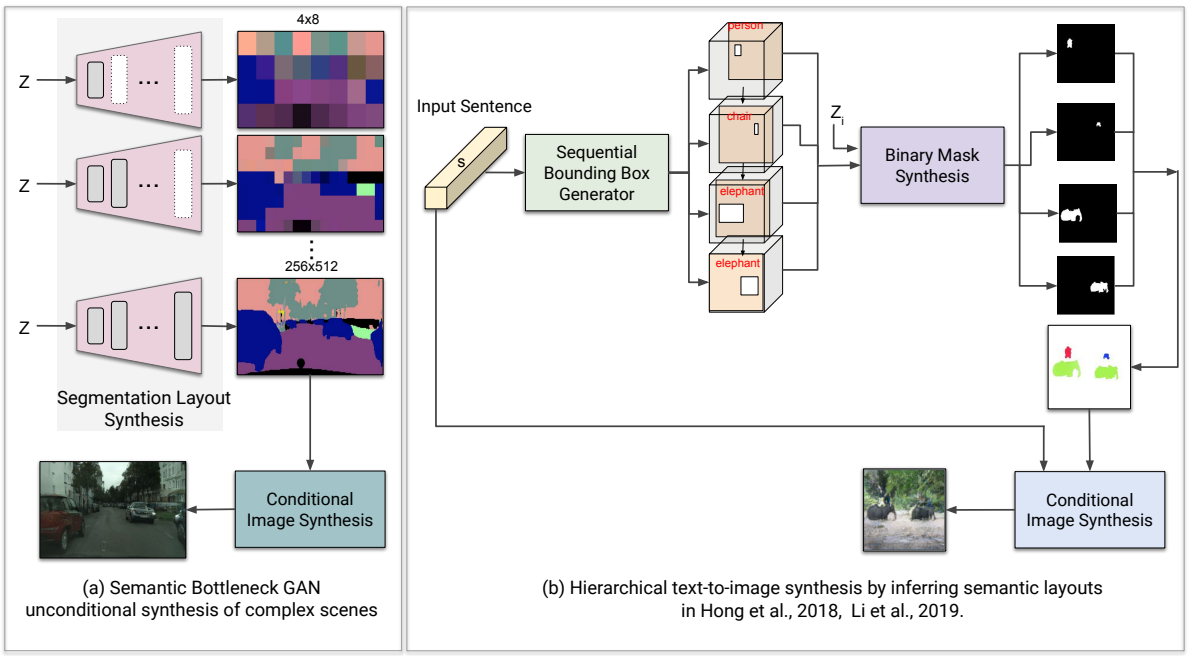


Figure 3.10: Architectural differences between our unconditional semantic bottleneck synthesis network and the conditional semantic layout synthesis network in Hong *et al.* (2018) and Li *et al.* (2019). (a) Schematic of our unconditional Semantic Bottleneck GAN. We progressively train an adversarial segmentation synthesis network to generate realistic segmentation maps from scratch, then synthesize a photo-realistic image using a conditional image synthesis network. End-to-end coupling of these two components results in state-of-the-art unconditional synthesis of complex scenes. For more detail about our conditional image synthesis network, one can refer to Section 3.2.2. (b) Schematic of the hierarchical text-to-image synthesis models inferring a semantic layout (Hong *et al.*, 2018; Li *et al.*, 2019). From an encoding of the input sentence, object bounding boxes are generated sequentially using an auto-regressive decoder, and are refined by a synthesized binary shape mask in the next step. The final image is synthesized given the constructed semantic layout and the text description. Note that whereas (b) conditionally generates masks only for objects, our model (a) unconditionally generates segmentation maps for the entire scene.

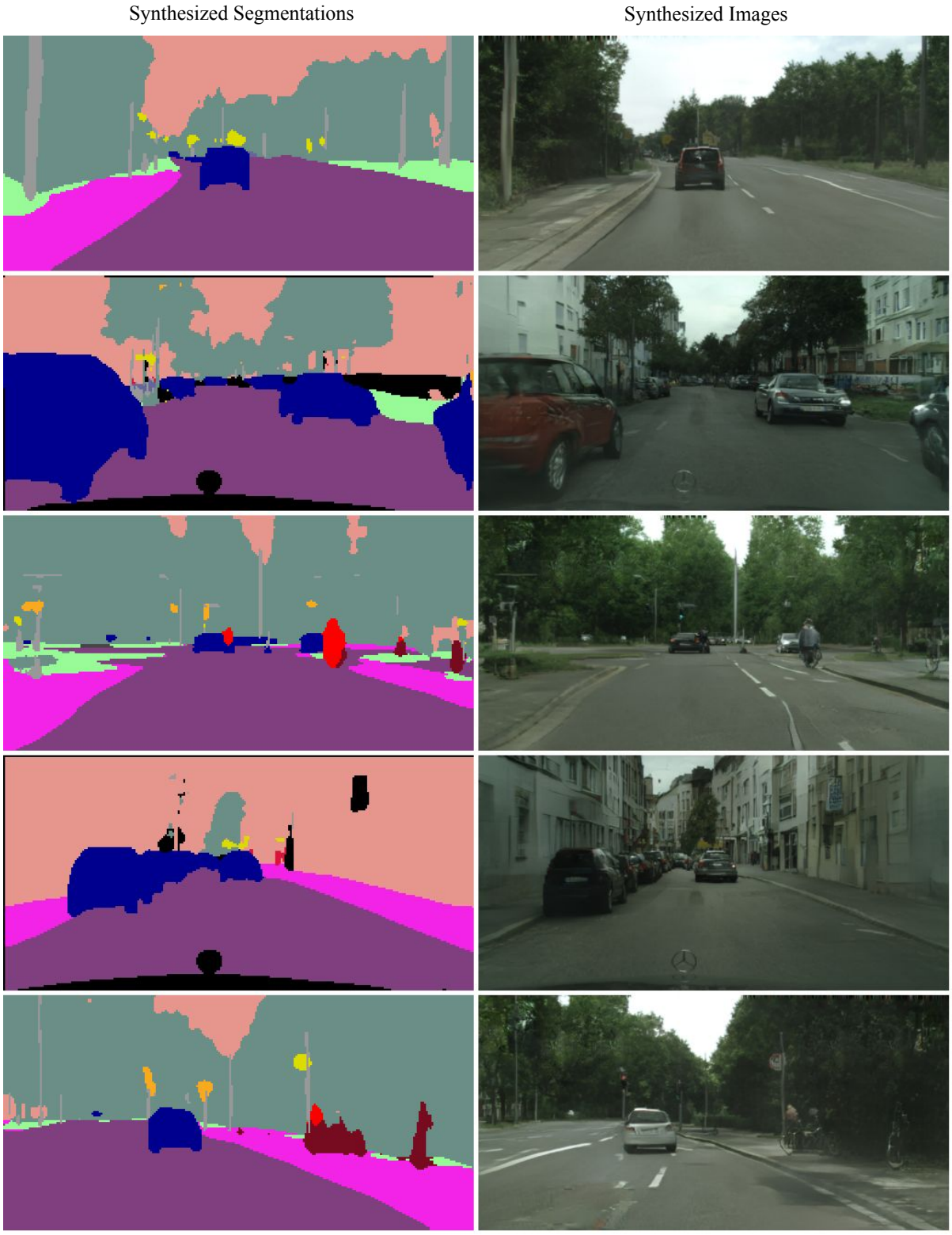


Figure 3.11: Segmentations and their corresponding images synthesized by SB-GAN trained on the Cityscapes-25K dataset.

Figure 3.12: Segmentations and their corresponding images synthesized by SB-GAN trained on the Cityscapes-25K dataset.

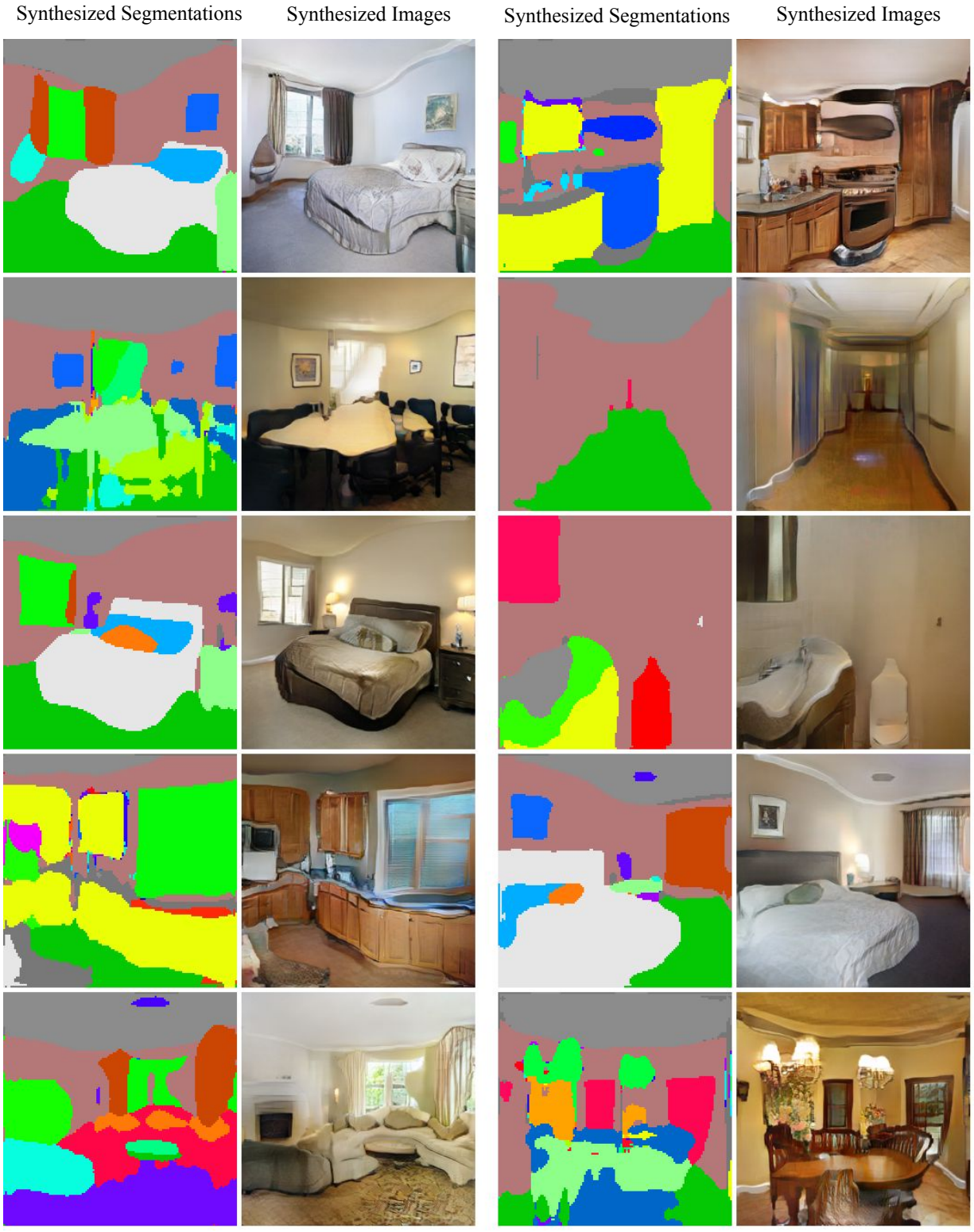


Figure 3.13: Segmentations and their corresponding images synthesized by SB-GAN trained on the ADE-Indoor dataset.

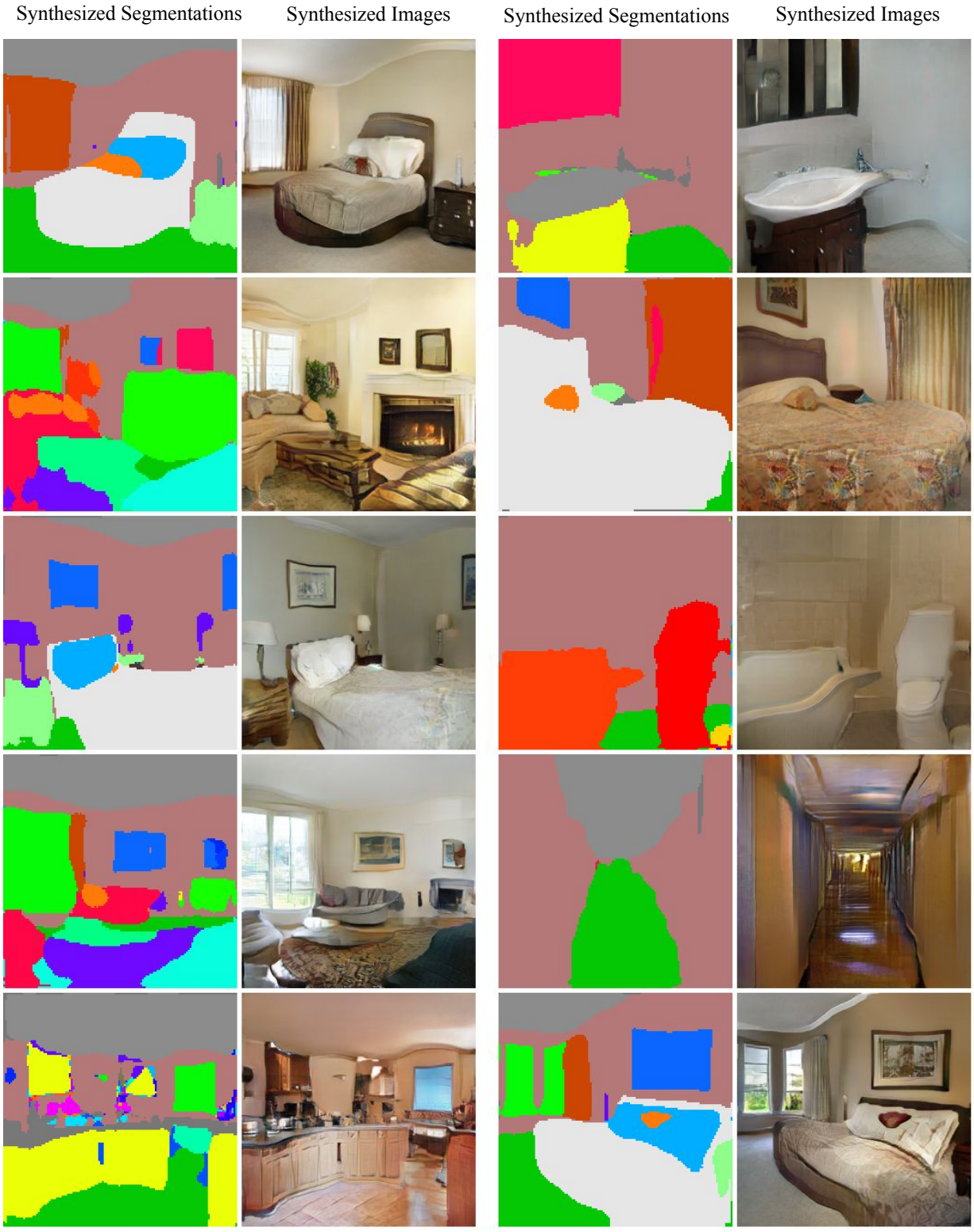


Figure 3.14: Segmentations and their corresponding images synthesized by SB-GAN trained on the ADE-Indoor dataset.

Chapter 4

Image-Conditional Binary Composition

4.1 Introduction

Conditional Generative Adversarial Networks (cGANs) have emerged as a powerful method for generating images conditioned on a given input. The input cue could be in the form of an image (Isola *et al.,* 2017a; Zhu *et al.,* 2017a; Liu *et al.,* 2017; Azadi *et al.,* 2017b; Wang *et al.,* 2017a; Pathak *et al.,* 2016), a text phrase (Zhang *et al.,* 2017a; Reed *et al.,* 2016b; Reed *et al.,* 2016a; Johnson *et al.,* 2018) or a class label layout (Mirza & Osindero, 2014; Odena *et al.,* 2016; Antoniou *et al.,* 2017). The goal in most of these GAN instances is to learn a mapping that *translates* a given sample from the source distribution to generate a sample from the output distribution. This primarily involves transforming either a single object of interest (apples to oranges, horses to zebras, label to image, etc.) or changing the style and texture of the input image (day to night, etc.). However, these direct transformations do not capture the fact that a natural image is a 2D projection of a *composition* of multiple objects interacting in a 3D visual world. In this chapter, we explore the role of compositionality in GAN frameworks and propose a new method which learns a function that maps images of different objects sampled from their marginal distributions (e.g., chair and table) into a combined sample (table-chair) that captures the joint distribution of object pairs. Here, we specifically focus on the composition of a pair of objects. [1]

Modeling compositionality in natural images is a challenging problem due to the complex interactions among different objects with respect to relative scaling, spatial layout, occlusion or viewpoint transformation. Recent work using spatial transformer networks (Jaderberg *et al.,* 2015) within a GAN framework (Lin *et al.,* 2018) decomposes this problem by operating in a geometric warp parameter space to find a geometric modification for a foreground object. However, this approach is only limited to a fixed background and does not consider more complex interactions in the real world.

We consider the task of composing two input object images into a joint image that

[1] This chapter is based on joint work done with Deepak Pathak, Sayna Ebrahimi, and Trevor Darrell (Azadi *et al.,* 2019c; Azadi *et al.,* 2019b) published at IJCV 2020.

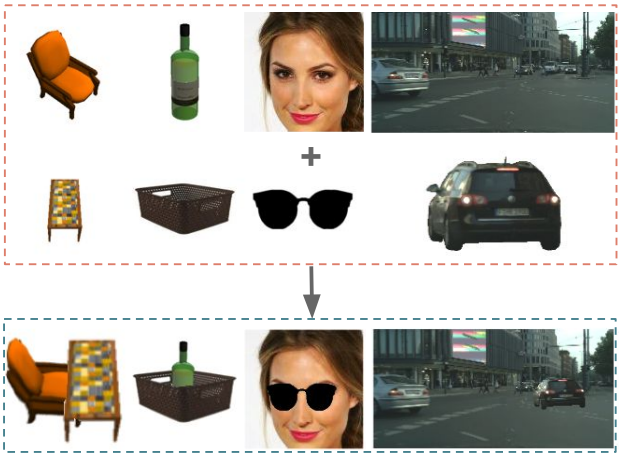


Figure 4.1: Binary Composition examples. *Top Row*: The first object or the background image, *Middle Row*: The second object or the foreground image, *Bottom Row*: The generated composite image.

captures their realistic interactions. For instance, given an image of a chair and an image of a table, our formulation is able to generate an image containing the same chair-table pair arranged in a realistic manner. To the best of our knowledge, this is the first work that addresses the problem of generating a composed image from two given inputs using a GAN, trainable under paired and unpaired scenarios. In an unpaired training setup, one does not have access to the paired examples of the same object instances with their combined compositional image. For instance, to generate the joint image from the image of a given table and a chair, we might not have any example of that particular chair beside that particular table while we might have images of other chairs and other tables together.

Our key insight is to leverage the idea that a successful composite image of two objects should not only be realistic in appearance but can also be decomposed back into individual objects. Hence, we use decomposition as a supervisory signal to train composition, thereby enforcing a self-consistency constraint (Zhu *et al.*, 2017a) through a composition-by-decomposition (CoDe) network. Moreover, we use this self-consistent CoDe network for an example-specific meta refinement (ESMR) approach at test time to generate sharper and more accurate composite images: We fine-tune the weights of the composition network on each given test example by the self-supervision provided from the decomposition network.

Through qualitative and quantitative experiments, we evaluate our compositional GAN approach in two training scenarios: (a) paired: when we have access to paired examples of individual object images with their corresponding composed image, (b) unpaired: when we have a dataset from the joint distribution without being paired with any of the images from the marginal distributions.

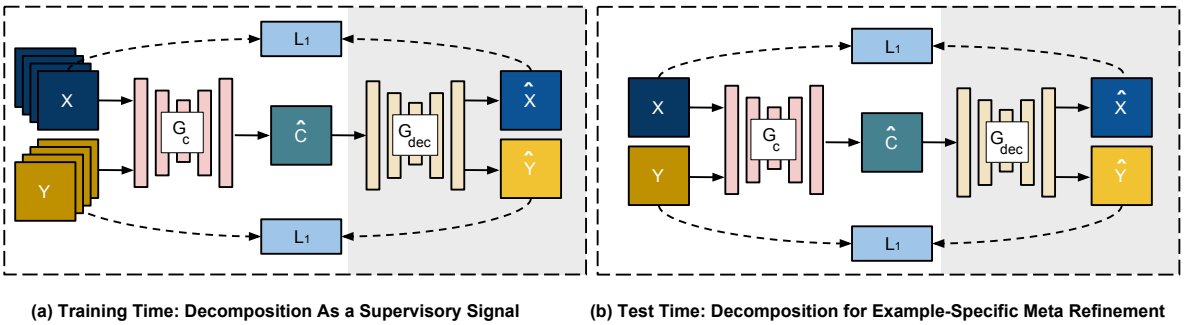


Figure 4.2: (a) The CoDe training network includes the composition network getting a self-consistent supervisory signal from the decomposition network. This network is trained on all training images, (b) ESMR: At test time, the weights of the trained composition and decomposition networks are fine-tuned given only one test example of X and one test example of Y. The decomposition network provides the self-supervision required for updating the weights of the composition network at test time. The layers of the composition generator are presented in pink and the decomposition generator in yellow.

4.2 Background: Conditional GAN

We briefly review the conditional Generative Adversarial Networks before discussing our compositional setup. Given a random noise vector z, GANs generate images c of a specific distribution using a generator G which is trained adversarially with respect to a discriminator D. While the generator tries to produce realistic images, the discriminator opposes the generator by learning to distinguish between real and fake images. In conditional GAN models (cGANs), an auxiliary information x is fed into the model, in the form of an image or a label, alongside the noise vector, i.e., $\{x, z\} \rightarrow c$ (Goodfellow, 2016; Mirza & Osindero, 2014). The objective of cGANs would be therefore an adversarial loss function formulated as

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

where G, D minimize and maximize this objective, respectively.

The convergence of the above GAN objective and consequently the quality of the generated images would be improved if an L_1 loss penalizing deviation of the generated images from their ground-truth is added. Thus, the generator's objective function would be summarized as

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathbb{E}_{x,c\sim p_{\text{data}}(x,c),z\sim p_z(z)}[\|c - G(x, z)\|_1]$$

4.3 Compositional GAN

Conditional GANs, discussed in Section 4.2, have been applied to several image translation problems such as day to night, horse to zebra, and sketch to portraint (Isola *et al.*,

2017a; Zhu *et al.,* 2017a). However, the composition problem is more challenging than just translating images from one domain to another because the model additionally needs to handle the relative scaling, spatial layout, occlusion, and viewpoint transformation of the individual objects to generate a realistic composite image. Here, we propose Compositional GAN for generating a composite image given two individual object images.

Let x be an image containing the first object, y be an image of the second object and c be the image from their joint distribution. During training, we are given datasets $X = \{x_1, \cdots, x_n\}$ and $Y = \{y_1, \cdots, y_n\}$ from the marginal distribution of the two objects, and $C = \{c_1, \cdots, c_n\}$ from their joint distribution containing both objects. We further assume that the segmentation masks of objects are available for both individual images in X, Y as well as the composite images in C. Our binary compositional GAN model is conditioned on *two* input images (x, y) to generate an image from the target distribution $p_{\text{data}}(c)$. The goal is to ensure that the generated composite image $\hat{c}$ contains the objects in x, y with the same color, texture, and structure while also looking realistic with respect to set C. Note that instead of learning a generative model of all possible compositions, our aim is to learn a mode of the distribution.

Since the conditional GANs are not adequate for transforming objects spatially, we explicitly model the scale and shift transformations by translating the object images (x, y) to (x^T, y^T) based on a relative Spatial Transformer Network (STN) (Jaderberg *et al.,* 2015). Moreover, in specific domains where a viewpoint transformation of the first object relative to the second one is required, we propose a Relative Appearance Flow Network (RAFN). Details of our relative STN and RAFN are provided in Sections 4.4.1, 4.4.2.

4.3.1 Supervising composition by decomposition

The central idea of our approach is to supervise the composition of two images (x^T, y^T) via a self-consistency loss function ensuring that the generated composite image, $\hat{c}$, can further be decomposed back into the respective individual object images. The composition is performed using a conditional GAN, $(G_{\text{c}}, D_{\text{c}})$, that takes the two RGB images (x^T, y^T) concatenated channel-wise as the input to generate the corresponding composite output, $\hat{c}$, with the two input images appropriately composed. This generated image will then be fed into a another conditional GAN, $(G_{\text{dec}}, D_{\text{dec}})$, to be decomposed back into its constituent objects, $(\hat{x}^T, \hat{y}^T)$ using a self-consistency L_1 loss function. Both the composition and decomposition networks are conditional GAN models with the generators following an encoder-decoder design similar to (Isola *et al.,* 2017a). The schematic of our self-consistent Composition-by-Decomposition (CoDe) network is illustrated in Figure 4.2-(a).

In addition to the decomposition network, the generated composite image will be given to a mask prediction network, G^M_{dec}, that predicts the probability of each pixel in the composite image to belong to each of the input objects or background. The argmax of these probabilities over object ids results in the estimated masks of the two objects as $\hat{M}_x$ and $\hat{M}_y$. The two decomposition generators G_{dec} and G^M_{dec} share their weights in their encoder network but are different in the decoder. A GAN loss with a gradient penalty (Gulrajani *et al.,* 2017b) is applied on top of the generated images $\hat{c}, \hat{x}^T, \hat{y}^T$ to make them look realistic in addition to

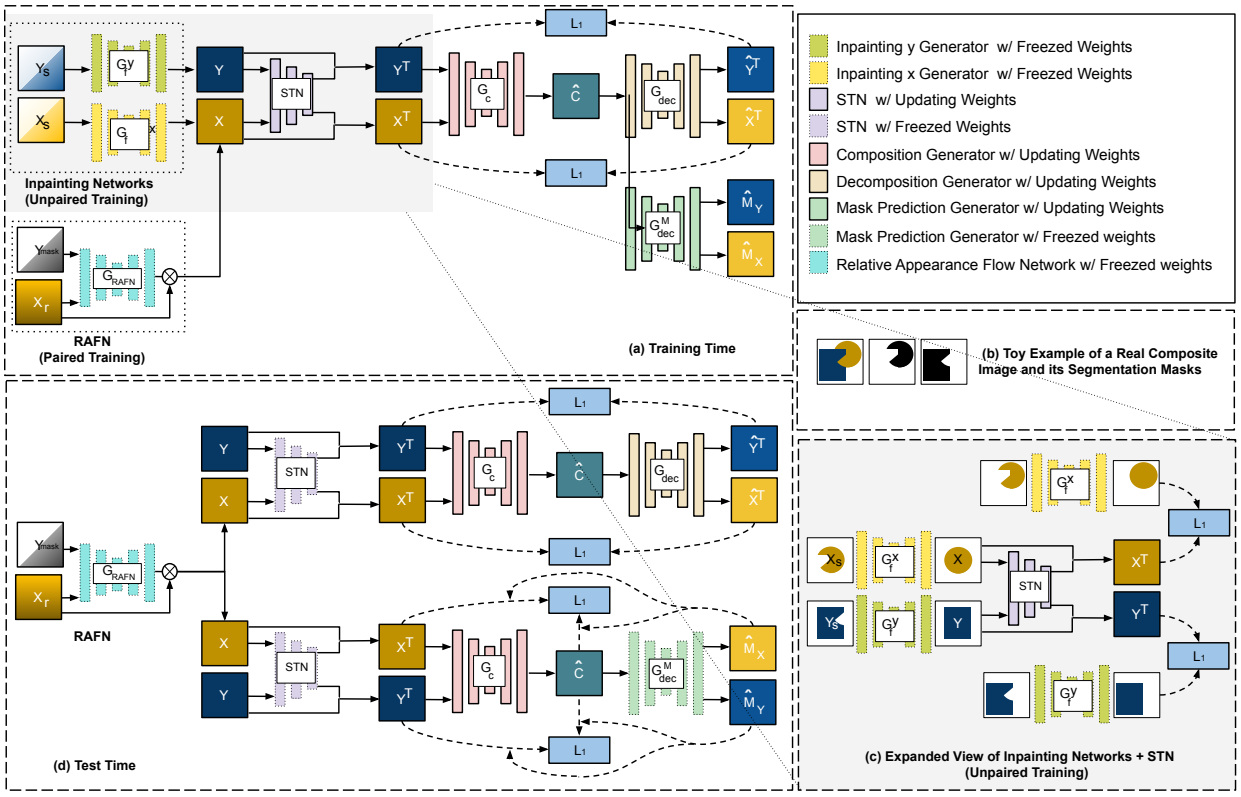


Figure 4.3: Schematic of our binary compositional GAN model at training and test times: (a) Our training model includes the inpainting networks (for unpaired data), RAFN (for paired data), the relative STN model, and the CoDe and mask prediction networks. X_s and Y_s stand for the respective object segments of real composite images in an unpaired training setup. X_r indicates the input image in an arbitrary viewpoint different from its corresponding composite image, and Y_{mask} is the binary segmentation mask of the object images in Y encoding the target viewpoint, (b) A toy example of a real composite image C, its object cutouts C_X and C_Y, and their segmentation masks, (c) We convert an unpaired training data to a paired setup by inpainting the object segment cutouts of the real composite image. The inpainted segments and their cropped variants at the center of the image are then used for training STN, (d) At test time, we fine-tune the weights of the CoDe network given only one test example from the X domain and one test example from the Y domain. The weights of the mask prediction network and STN are not updated on test examples. Each of the above modules is represented by a different color, and repeating the same module in different parts of this diagram is for the illustration purpose.

multiple L_1 loss functions penalizing deviation of the generated images from their ground-truth. Further details of our training model are provided in Section 4.4.4.

Extension to Unpaired Data: We train our Compositional GAN framework in two scenarios: (1) when inputs-output are *paired* in the training set, i.e., each composite image in C has corresponding individual object images in X, Y, and (2) when training data is *unpaired*, i.e., images in C do not correspond to images in X and Y. We convert the unpaired data to a paired one by cutting out the respective object segments from each composite image in C to get the corresponding paired individual object images. Although these new object cutouts would be paired with the composite image, they are incomplete and not amodal because of occlusion in the composite image. Hence, we synthesize the missing part of these individual object cutouts using self-supervised inpainting networks (Pathak *et al.,* 2016) which are trained on object images from X and Y, described in Section 4.4.3.

4.3.2 Example-Specific Meta-Refinement (ESMR)

The compositional GAN model not only should learn to compose two object with each other, but it also needs to preserve the color, texture and other properties of the individual objects in the composite image. While our framework is able to handle the former, it suffers at times to preserve color and texture of held-out objects at test time. We propose to handle this issue by performing per-example refinement at test time. Since our training algorithm gets supervision by decomposing the composite image back into individual objects, we can use the same supervisory signal to refine the generated composite image $\hat{c}$ for unseen test examples as well. Hence, we continue to optimize the network parameters using the decomposition of the generated image back into the two test objects to remove artifacts and generate sharper results. This example-specific meta-refinement (ESMR), depicted in Figure 4.2-(b), improves the quality of the composite image at inference.

Given the segmentation masks of the input object images, we again ignore background for simplicity. We freeze the weights of the relative STN, RAFN, and G^M_{dec}, while only refining the weights of the CoDe layers. A GAN loss is applied on the outputs of the generators given the real samples from our training set. The objective function for our ESMR approach would be thus summarized as

$$\begin{aligned}\mathcal{L}(G) &= \lambda(\|\hat{x}^T - x^T\|_1 + \|\hat{M}_x \odot \hat{c} - \hat{M}_x \odot x^T\|_1 \\ &+ \|\hat{y}^T - y^T\|_1 + \|\hat{M}_y \odot \hat{c} - \hat{M}_y \odot y^T\|_1) \\ &+ [\mathcal{L}_{\text{cGAN}}(G_c, D_c) + \mathcal{L}_{\text{cGAN}}(G_{\text{dec}}, D_{\text{dec}})],\end{aligned}$$

where $\hat{x}^T, \hat{y}^T$ are the generated decomposed images, and x^T, y^T are the transposed inputs. Moreover, $\hat{M}_x$ and $\hat{M}_y$ indicate the object masks predicted by the mask decomposition network given the generated composite image $\hat{c}$ as discussed in Section 4.3.1. Here, the decomposition and mask prediction networks reinforce each other in generating sharper outputs and predicting more accurate segmentation masks. The mask prediction loss (i.e., second

and fourth L_1 loss terms) provides an extra supervision for occlusion ordering of the two objects in the composite image during meta-refinement optimization at inference. We quantify it through an ablation study in Section 5.5 where eliminating the mask prediction network results in an incorrect occlusion ordering. On the other hand, ignoring the self-consistency L_1 loss from the decomposition network (i.e., first and third loss terms) results in a composite image with the object shapes deviated from their corresponding inputs, as shown in the ablation study.

4.4 Implementation Details

In this section, we provide more details on the components of our training network including the relative STN, RAFN, inpainting, and our full end-to-end model.

4.4.1 Relative spatial transformer network

Given the segmentation masks of the objects for images in sets X, Y, and C, we crop and scale all input objects to be at the center of the image in all training images. To *relatively* translate the center-oriented input objects, (x, y), to an appropriate spatial layout, we train our variant of the spatial transformer network (STN) (Jaderberg *et al.*, 2015). This Relative STN simultaneously takes the *two* RGB images concatenated channel-wise and translates them relatively to each other to (x^T, y^T) based on their spatial relation encoded in the training composite images, as represented in Figure 4.4.

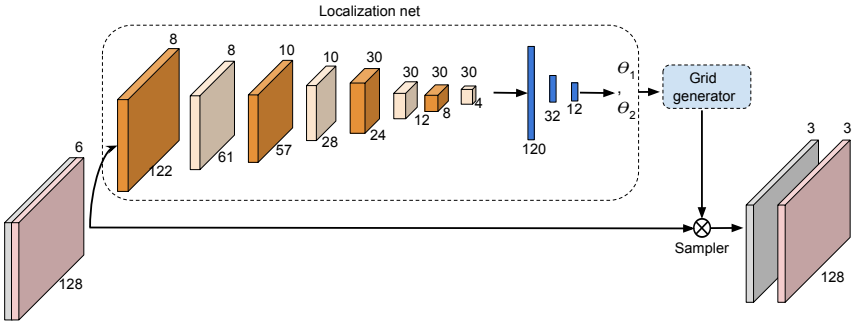


Figure 4.4: Relative Spatial Transformer Network: First input with three RGB channels (e.g., image of a chair) concatenated channel-wise with the second RGB image (e.g., image of a table). The network generates two transformed images each with three RGB channels. The orange feature maps are the outputs of the `conv2d` layer (represented along with their corresponding number of channels and dimensions), and the yellow maps are the outputs of the `max-pool2d` followed by a `ReLU`. The blue layers also represent fully connected layers.

4.4.2 Relative Appearance Flow Network (RAFN)

In specific domains where the relative viewpoint of the objects should be changed accordingly to generate a natural composite image, we introduce our relative appearance flow network orthogonal to our main CoDe pipeline. Irrespective of the paired or unpaired training data, we propose a relative encoder-decoder appearance flow network, G_{RAFN}, based on the AFN model introduced in (Zhou *et al.,* 2016). The AFN model of (Zhou *et al.,* 2016) uses an explicit rotating angle parameter for synthesizing images in their target view. However, our RAFN model synthesizes a new viewpoint of the first object, x, given the viewpoint of the second one, y, *encoded in its binary mask*. Our RAFN is trained on a set of images in X with arbitrary azimuth angles $\alpha \in \{-180°, -170°, \cdots, 180°\}$ along with their target images with arbitrary new azimuth angles $\theta \in \{-180°, -170°, \cdots, 180°\}$ and a set of foreground masks of images in Y in the same target viewpoints. The architecture of our RAFN is illustrated in Figure 4.5, and its loss function is formulated as

$$\begin{aligned} \mathcal{L}(G_{\text{RAFN}}) &= \mathcal{L}_{L_1}(G_{\text{RAFN}}) + \lambda \mathcal{L}_{\text{BCE}}(G^M_{\text{RAFN}}) \\ &= \mathbb{E}_{(x,y)}[\|x - G_{\text{RAFN}}(M^{\text{fg}}_y, x^r)\|_1] + \lambda \mathbb{E}_x[\hat{M}^{\text{fg}}_x \log M^{\text{fg}}_x + (1 - \hat{M}^{\text{fg}}_x) \log(1 - M^{\text{fg}}_x)] \end{aligned} \tag{4.1}$$

Here, G_{RAFN} is the encoder-decoder network predicting the appearance flow vectors, which after a bilinear sampling step generates the synthesized view. Also, G^M_{RAFN} is an encoder-decoder mask prediction network sharing the weights of its encoder with G_{RAFN}, while its decoder is designed for predicting the foreground masks of the synthesized images. Moreover, x is the ground-truth image for the first object in the target viewpoint while x^r indicates the input image in an arbitrary view. The predicted foreground mask of RAFN is represneted by $\hat{M}^{\text{fg}}_x$, while M^{fg}_x, M^{fg}_y are the ground-truth segmentation masks for objects in x, y, respectively.

4.4.3 Inpainting network

As discussed in Section 4.3.1, when paired training data is not available, we generate a synthetic paired data from the composite images via inpainting. This happens in two steps. First, we train GAN-based inpainting networks, G^x_{f} and G^y_{f}, for each of the individual object domain X and Y respectively. These inpainting networks can be trained in a self-supervised manner (Pathak *et al.,* 2016). For instance, to train G^x_{f}, we apply a random binary mask on an image x from X to zero out the pixel values inside the applied mask region. Given the masked image and the original image x, we train a conditional GAN, $(G^x_{\text{f}}, D^x_{\text{f}})$, to fill in the masked regions of the image. Another cGAN network, $(G^y_{\text{f}}, D^y_{\text{f}})$, would be trained similarly to fill in the masked regions of images in Y. Instead of random binary masks, one can also use randomly sampled object masks from images in Y to zero out pixel values of images in X and vice-versa. The loss function for each inpainting network would be:

$$\mathcal{L}(G_{\text{f}}) = \mathcal{L}_{L_1}(G_{\text{f}}) + \lambda \mathcal{L}_{\text{cGAN}}(G_{\text{f}}, D_{\text{f}}) \tag{4.2}$$

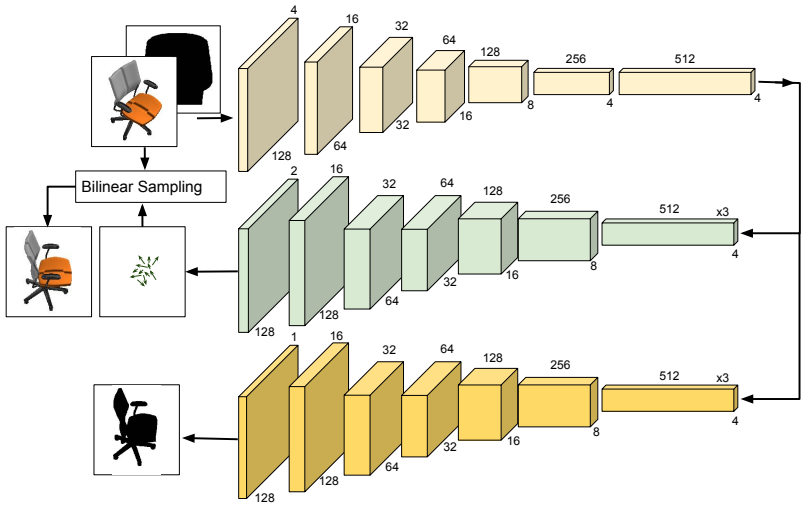


Figure 4.5: Relative Appearance Flow Network: Input is an image of a chair with three RGB channels concatenated channel-wise with the table foreground mask. Output is the appearance flow for synthesizing a new viewpoint of the chair. All layers are convolutional.

In the second step, we use the trained inpainting network G_{f}^{x} and G_{f}^{y} to generate synthetic paired data from composite images. We sample an image from the composite set C, and generate two object cutouts using the segmentation mask. These cutouts are then cropped and centered, as shown in Figure 4.3b-4.3c, to generate X_s and Y_s. We then use our pre-trained inpainting networks from step one to fill in X_s and Y_s. The inpainted images are treated as synthetic paired object images corresponding to C.

Finally when training the CoDe model with this synthetically generated paired data, we add an additional L_1 loss on STN (see Figure 4.3c) to penalize the deviation of the inpainted cropped segments from the inpainted non-cropped segments both paired with the same composite image. Using inpainted segments instead of the occluded ones to be paired with the real composite image in C reinforces the CoDe model in learning object occlusions and spatial layouts more accurately, as shown in Figure 4.6.

4.4.4 Full model

An schematic of our full network, G, is represented in Figure 4.3 and the objective function is composed of:

- Pixel-reconstruction L_1-loss functions on the outputs of the composition generator,

decomposition generator, and the relative STN model:

$$\begin{aligned}\mathcal{L}_{L_1}(G_{\mathrm{c}}) &= \mathbb{E}_{(x,y,c)}\big[\|c-\hat{c}\|_1\big],\\ \mathcal{L}_{L_1}(G_{\mathrm{dec}}) &= \mathbb{E}_{(x,y)}\big[\|(x^T,y^T)-G_{\mathrm{dec}}(\hat{c})\|_1\big],\\ \mathcal{L}_{L_1}(\mathrm{STN}) &= \mathbb{E}_{(x,y)}\big[\|(x^c,y^c)-(x^T,y^T)\|\big],\end{aligned}$$

where $(x^T, y^T) = \mathrm{STN}(x, y)$ and $\hat{c} = G_{\mathrm{c}}(x^T, y^T)$. Moreover, $(x^c,\ y^c)$ are the ground-truth transposed input objects corresponding to the composite image, c, in the paired scenario (or equivalently, the inpainted object segments of c in the unpaired case). Also, $\mathcal{L}_{L_1}(G_{\mathrm{dec}})$ indicates the self-consistency constraint penalizing deviation of decomposed images from their corresponding inputs,

- A cross-entropy mask prediction loss as $\mathcal{L}_{\mathrm{CE}}(G^M_{\mathrm{dec}})$ to assign a label to each pixel of the generated composite image, $\hat{c}$, corresponding with the $\{x, y, \mathrm{background}\}$ classes,
- Conditional GAN loss functions for both the composition and decomposition networks:

$$\mathcal{L}_{\mathrm{cGAN}}(G_{\mathrm{c}}, D_{\mathrm{c}}) = \mathbb{E}_{(x,y,c)}\big[\log D_{\mathrm{c}}(x^T, y^T, c)\big] + \mathbb{E}_{(x,y)}\big[1-\log D_{\mathrm{c}}(x^T, y^T, \hat{c})\big],$$

$$\begin{aligned}\mathcal{L}_{\mathrm{cGAN}}(G_{\mathrm{dec}}, D_{\mathrm{dec}}) &= \mathbb{E}_{(x,y)}\big[\log D_{\mathrm{dec}}(\hat{c}, x^c) + \log D_{\mathrm{dec}}(\hat{c}, y^c)\big]\\ &+ \mathbb{E}_{(x,y)}\big[(1-\log D_{\mathrm{dec}}(\hat{c}, \hat{x}^T)) + (1-\log D_{\mathrm{dec}}(\hat{c}, \hat{y}^T))\big].\end{aligned} \tag{4.3}$$

We also added the gradient penalty introduced by (Gulrajani *et al.*, 2017b) to improve the convergence of the GAN loss functions. In summary, the objective for the full end-to-end model is

$$\begin{aligned}\mathcal{L}(G) &= \lambda_1[\mathcal{L}_{L_1}(G_{\mathrm{c}}) + \mathcal{L}_{L_1}(G_{\mathrm{dec}}) + \mathcal{L}_{L_1}(\mathrm{STN})]\\ &+ \lambda_2\mathcal{L}_{\mathrm{CE}}(G^M_{\mathrm{dec}}) + \lambda_3[\mathcal{L}_{\mathrm{cGAN}}(G_{\mathrm{c}}, D_{\mathrm{c}}) + \mathcal{L}_{\mathrm{cGAN}}(G_{\mathrm{dec}}, D_{\mathrm{dec}})]\end{aligned}$$

4.5 Experiments

In this section, we study the performance of our compositional GAN model in both the paired and unpaired training regimes through multiple qualitative and quantitative experiments on synthetic and real data sets. We will present: (1) images generated directly from the composition network, $\hat{c}$, before and after the ESMR step, (2) images generated directly based on the predicted segmentation masks as $\hat{c}_s = \hat{M}_x \odot x^T + \hat{M}_y \odot y^T$. In all of our experiments, the training hyper-parameters are $\lambda_1 = 100$, $\lambda_2 = 50$, $\lambda_3 = 1$, and the inference $\lambda = 100$. Furthermore, similar to (Isola *et al.*, 2017a), we ignore random noise as the input to the generator, and dropout is the only source of randomness in the network.

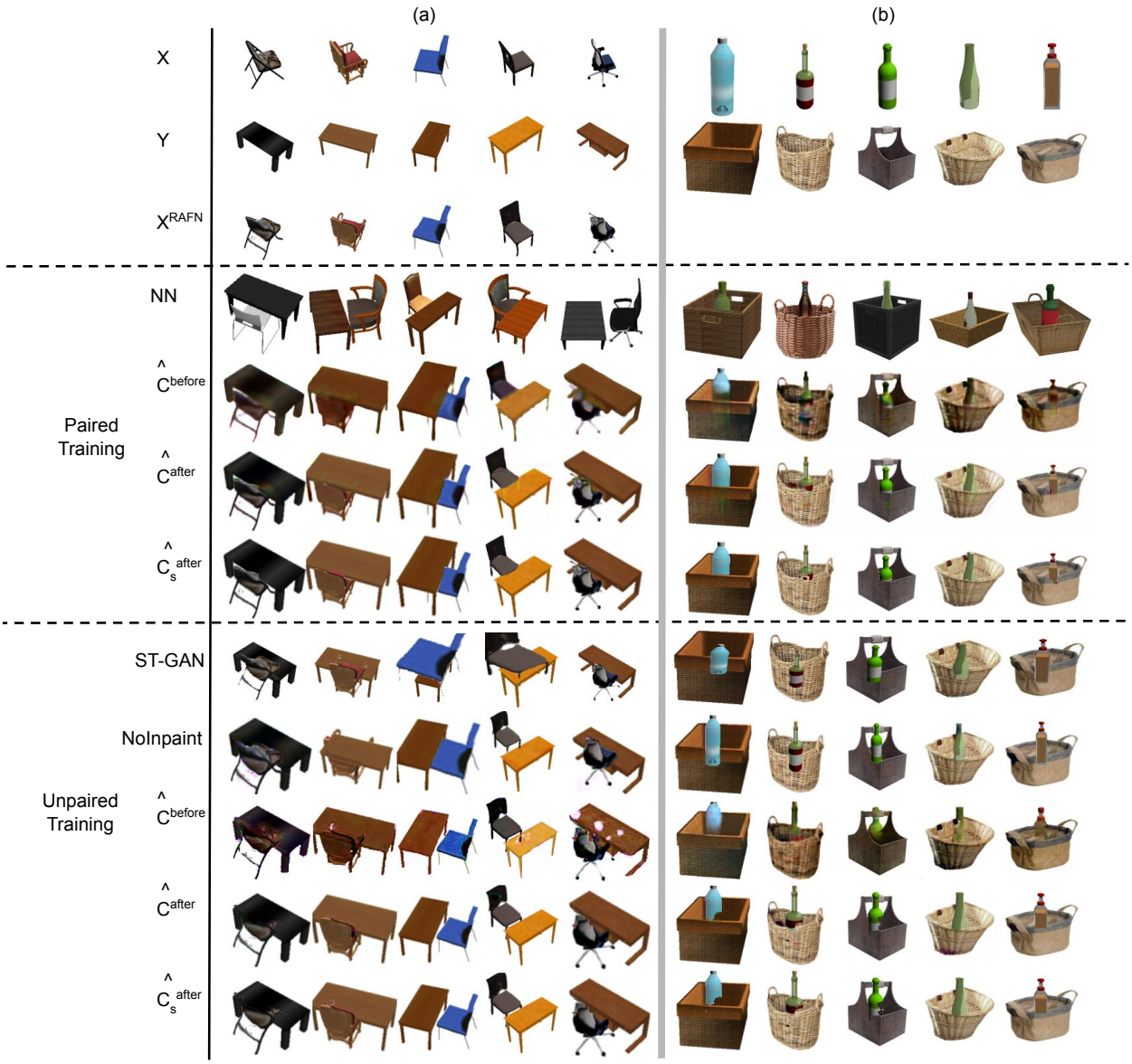


Figure 4.6: Test results on (a) the chair-table and (b) the basket-bottle composition tasks trained with either paired or unpaired data. In the chair-table examples, we use X^{RAFN} as the input to all models."NN" stands for the nearest neighbor image in the paired training set, and "NoInpaint" shows the results of the unpaired model without the inpainting network. In both paired and unpaired cases, $\hat{c}^{\text{before}}$ and $\hat{c}^{\text{after}}$ show outputs of the generator before and after ESMR, respectively. Also, $\hat{c}_s^{\text{after}}$ represents summation of masked transposed inputs after the ESMR step.

4.5.1 Synthetic data sets

In this section, we use the Shapenet dataset (Chang *et al.*, 2015) and study two composition tasks: (1) a chair next to a table, (2) a bottle in a basket. On the chair-table data set, we deal with all four composition challenges, i.e., spatial layout, relative scaling, occlusion, and viewpoint transformation. In the basket-bottle experiment, the main challenge is to predict the correct occluding pixels as well as the relative scaling of the two objects.

Composing a chair with a table: We manually made a collection of 1K composite images from Shapenet chairs and tables to use as the real joint set, C, in the paired and unpaired training schemes. Chairs and tables in the input-output sets can pose in random azimuth angles in the range $[-180°, 180°]$ at steps of $10°$. As discussed in section 4.4.2, given the segmentation mask of an arbitrary table in a random viewpoint and an input chair, our relative appearance flow network synthesizes the chair in the viewpoint consistent with the table. The synthesized test chairs as X^{RAFN} are presented in the third row of Figure 4.6-a.

Composing a bottle with a basket: We manually composed Shapenet bottles with baskets to prepare a training set of 100 joint examples and trained the model both with and without the paired data.

4.5.1.1 Ablation study and baselines

To study the role of different components of our network, we visualize the predicted outputs at different steps in Figure 4.6. Trained with either paired or unpaired data, we illustrate output of the generator before and after the ESMR step discussed in section 4.3.2, as $\hat{c}^{\text{before}}$ and $\hat{c}^{\text{after}}$, respectively. The ESMR step sharpens the synthesized images at test time and removes the artifacts generated by the model. Given generated images after being refined and their segmentation masks predicted by our pre-trained mask decomposition network, we also represent outputs as the direct summation of the segments, $\hat{c}_s^{\text{after}}$. Our results from the model trained with unpaired data are comparable with those from paired data. Moreover, we depict the performance of the model without our inpainting network in the ninth row, where occlusions are not correct in multiple examples. More test examples are illustrated in Figure 4.12 and a few failure test examples in Figure 4.11 for both paired and unpaired training models.

In addition, to make sure that our network does not memorize its training samples and can be generalized for each new input test example, we find the nearest neighbor composite example in the training set based on the features of its constituent objects extracted from a pre-trained VGG19 network (Simonyan & Zisserman, 2014). The nearest neighbor examples for each test case are shown in the fourth row of Figure 4.6. We further repeat the experiments with each component of the model removed at a time to study their effect on the final composite image. The qualitative results are illustrated in Figure 5.4. The first and second columns show bottle and basket images concatenated channel-wise as the input of the network. In the third column, there is no pixel reconstruction loss on the composite image resulting in a wrong color and defective occlusion. The fourth column shows a faulty

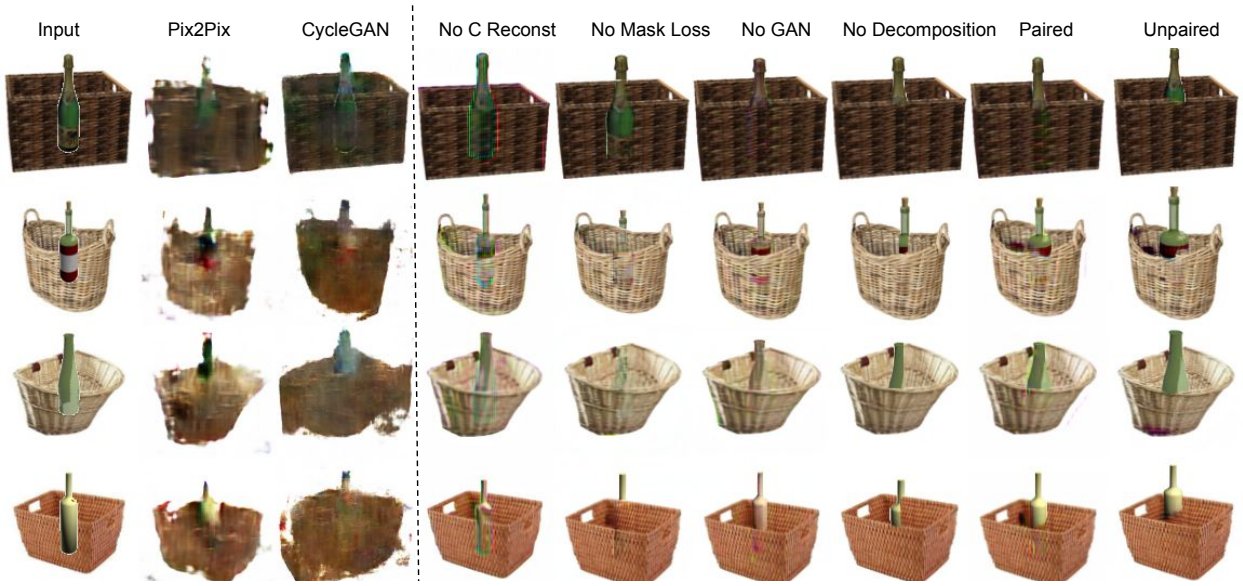


Figure 4.7: (a) Ablation Study: output of our compositional GAN model without the component specified on top of each column. Input is the channel-wise concatenation of the bottle and basket shown in the first two columns, (b) Baselines: As the input (9th column), each bottle is added to the basket after being scaled and translated with constant parameters. The Pix2Pix and CycleGAN outputs are shown in the last two columns.

occlusion ordering of the two objects when the cross-entropy mask loss is eliminated during training. Moreover, as shown in the fifth column, removing the GAN loss in training and inference generates lower quality outputs with a different color compared with the input image. As illustrated in the sixth column, training the network without the decomposition generator and the self-consistency L_1 loss function results in the deviation of the bottle shape from its corresponding input. The results before ESMR in both paired and unpaired cases are shown in the seventh and ninth columns followed by the full model results represented in the eighth and tenth columns, respectively.

Our compositional GAN pipeline can address the challenging problem of generating a realistic composite image by learning the spatial layout, relative scaling, occlusion, and viewpoint of the two object images. However, the conditional GAN models such as CycleGAN (Zhu *et al.,* 2017a) and Pix2Pix (Isola *et al.,* 2017a) address the image translation problem from one domain to another by only changing the appearance of the input image. Here, we compare our model with these two networks in the basket-bottle composition task. We use the mean scaling and translation parameters of our training set to place the bottle and basket examples in one RGB input image, illustrated in the ninth column of Figure 5.4. The Pix2Pix network includes a ResNet generator trained on our paired training data with an adversarial loss added with an L_1 loss regularizer. Since the scaling and layout of the objects in the input image are different from the corresponding target composite image,

Table 4.1: AMT user evaluation comparing components of our model on the synthetic datasets. 2nd column: number of test images, 3rd column: % preferences to after vs. before refinement, 4th column: % preferences to paired training vs. unpaired.

Inputs	# test images	after-vs-before refinement	paired-vs-unpaired
Chair-Table	90	71.3%	57%
Basket-Bottle	45	64.2%	57%

ResNet model works better than a U-Net with skip connections. However, it still generates unrealistic images as presented in the tenth column of Figure 5.4. We followed the same approach to train the CycleGAN model on the unpaired data and represent the results in the last column of Figure 5.4. Our qualitative results confirm the difficulty of learning the transformation between samples from the input distribution and the real composite domain for the Pix2Pix or CycleGAN networks.

4.5.1.2 User evaluations

We conducted an Amazon Mechanical Turk (AMT) evaluation (Zhang *et al.,* 2016) to compare the performance of our algorithm in different scenarios including training with and without paired data and before and after ESMR. Results from 60 evaluators are summarized in Table 4.1, revealing that even without paired examples during training, our compositional GAN model performs comparably well. In addition, the benefit of the ESMR module in generating higher-quality images is clear from the table.

4.5.2 Real data sets

In this section, we use two real datasets to show our model performs equally well when one object is fixed and the other one is relatively scaled and linearly transformed to generate a composed image: (1) a pair of sunglasses to be aligned with a face, (2) a car to be added to a street scene. The problem here is thus similar to the case studies of ST-GAN (Lin *et al.,* 2018) with a background and foreground object.

Composing a face with sunglasses: Here, we used the CelebA dataset (Liu *et al.,* 2015) and cropped the images to 128×128 pixels. We hand-crafted 180 composite images of celebrity faces from the training split aligned with sunglasses downloaded from the web. In the unpaired scenario, the training set of individual faces in X contains 6K images from the CelebA training split, distinct from the faces in our composite set. As the baseline, we trained ST-GAN (Lin *et al.,* 2018) with 10K celebrity faces of the celebA dataset with eyeglasses.

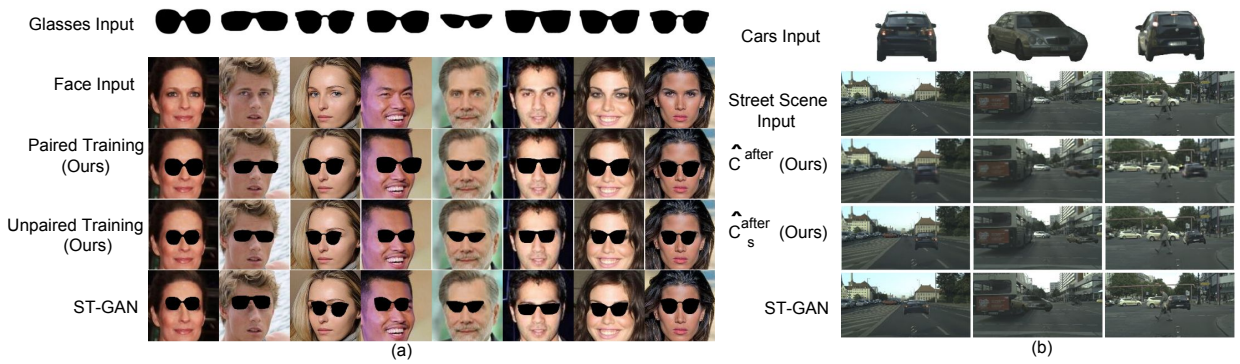


Figure 4.8: (a) Test examples for the face-sunglasses composition task. *Top two rows*: input sunglasses and face images, *3rd and 4th rows*: the output of our compositional GAN for the paired and unpaired models, respectively, *Last row*: images generated by the ST-GAN (Lin *et al.,* 2018) model, (b) Test examples for the street scene-car composition task. *Top two rows*: input cars and street scenes, *3rd and 4th rows*: the output of our compositional GAN after the meta-refinement approach. Here, $\hat{c}^{\text{after}}$ shows the output of the composition generator and $\hat{c}_{\text{s}}^{\text{after}}$ represents the summation of the masked transposed inputs, *Last row*: images generated by ST-GAN.

Composing a street scene with a car: We used the Cityscapes dataset (Cordts *et al.,* 2016b) and extracted two sets of non-overla- pping street scenes from the training set to be used as domains X and C in our compositional setup. In addition, we extracted a set of car images for domain Y using the instance segmentation masks available in the Cityscapes dataset and scaled them to be in a fixed size range. For the scenes collected as our real composite images in set C, we used the available segmentation mask of one of the cars in the scene as the mask of the foreground object. We manually filtered the real samples that do not include any cars larger than a specific dimension. We also flipped images in all three sets for data augmentation. Overall, we have 500 composite images in set C, 1100 cars images in set Y, and 1500 street scenes in set X, all down-scaled to 128×256 pixels. To train the ST-GAN model as a baseline, we used 1600 Cityscapes street scenes as the real composite images.

4.5.2.1 Qualitative analysis and baselines

In Figure 4.8, we compare the performance of our model with ST-GAN (Lin *et al.,* 2018), which assumes images of faces (or street scenes) as a fixed background and warps the glasses (or the cars) in a geometric warp parameter space. Our results in the paired and unpaired cases, shown in Figure 4.8, look more realistic in terms of the scale and location of the foreground objects. More examples are presented in Figures 4.13 and 4.14. In the street scene-car composition, we do not have any paired data and can only evaluate the unpaired model.

Table 4.2: AMT user evaluation comparing our model with ST-GAN on the real datasets. 2nd column: number of test images, 3rd and 4th columns: % preferences respectively to paired and upaired training vs. ST-GAN.

Inputs	# test images	paired-vs-ST-GAN	unpaired-vs-ST-GAN
Face-Sunglasses	75	84%	73%
Street Scene-Car	80	-	61%

4.5.2.2 User evaluations

To confirm our qualitative observations, we asked 60 evaluators to score our model predictions versus ST-GAN, with the results summarized in Table 4.2. This experiment confirms the superiority of our network to the state-of-the-art model in composing a background image with a foreground.

4.5.3 Generalization to unseen categories

Here, we study the generalization of our model to the composition of unseen (but similar) categories. To this aim, we use a model trained on the *bottle-basket* composition task and feed it new pairs of *can-basket* images at test time. The results are illustrated in Figure 4.9. These qualitative examples confirm that the model effectively understands the 3D geometry of the composition problem without explicitly training for it and is able to generalize from *bottles* to *cans*.

Figure 4.9: Generalization of the compositional GAN model trained on bottle-basket examples to a similar unseen category of cans to be composed with baskets. The first two rows indicate the new test examples, and the last row show the generated composite images.

4.5.4 Extension to adding more objects

In this section, we extend our binary compositional GAN model to iteratively compose more objects through a qualitative study on the street scene-car composition task. Fig-

Figure 4.10: Test examples for an iterative extension of the street scenes-cars composition task to more than two objects. The first two rows show the input street and the input car images in the first iteration. The 3rd row illustrates the output of our compositional generator, used as the street scene inputs for the next iteration to be composed with the cars shown in the 4th row. The 5th row indicates the final generated composite image of the two cars added to the street scenes iteratively.

ure 4.10 illustrates the output of our compositional generator after an iterative composition of two cars with street scenes. Final results seem to be qualitatively plausible suggesting that a trained binary model generalizes iteratively to add more objects.

4.6 Related Work

Image composition is a challenging problem in computer graphics where objects from different images are to be overlaid in one single image. The appearance and geometric differences between these objects are the obstacles that can result in non-realistic composed images. (Zhu *et al.*, 2015) addressed the composition problem by training a discriminator that could distinguish realistic composite images from synthetic ones. (Tsai *et al.*, 2017) developed an end-to-end deep CNN for image harmonization to automatically capture the context and semantic information of the composite image. This model outperformed its precedents (Sunkavalli *et al.*, 2010; Xue *et al.*, 2012) which transferred statistics of hand-crafted features to harmonize the foreground and the background in the composite image. Recently, (Lin *et al.*, 2018) used spatial transformer networks as a generator by performing geometric corrections to warp a masked object to adapt to a fixed background image. Moreover, (Johnson *et al.*, 2018) computed a scene layout from given scene graphs which revealed explicit reasoning about relationships between objects and converted the layout to an output

image. In the image-conditional composition problem which we address, each object should be rotated, scaled, and translated while partially occluding the other object to generate a realistic composite image.

4.7 Discussion

In this chapter, we proposed a novel Compositional GAN model addressing the problem of object composition in conditional image generation. Our model captures the relative affine and viewpoint transformations needed to be applied to each input object (in addition to the pixels occlusion ordering) to generate a realistic joint image. We used a decomposition network as a supervisory signal to improve the task of composition both at training and test times. We evaluated our compositional GAN through multiple qualitative experiments and user evaluations for two cases of paired versus unpaired training data on synthetic and real data sets. This work can be extended toward modeling photometric effects (e.g., lighting) in addition to generating images composed of multiple (more than two) and/or non-rigid objects.

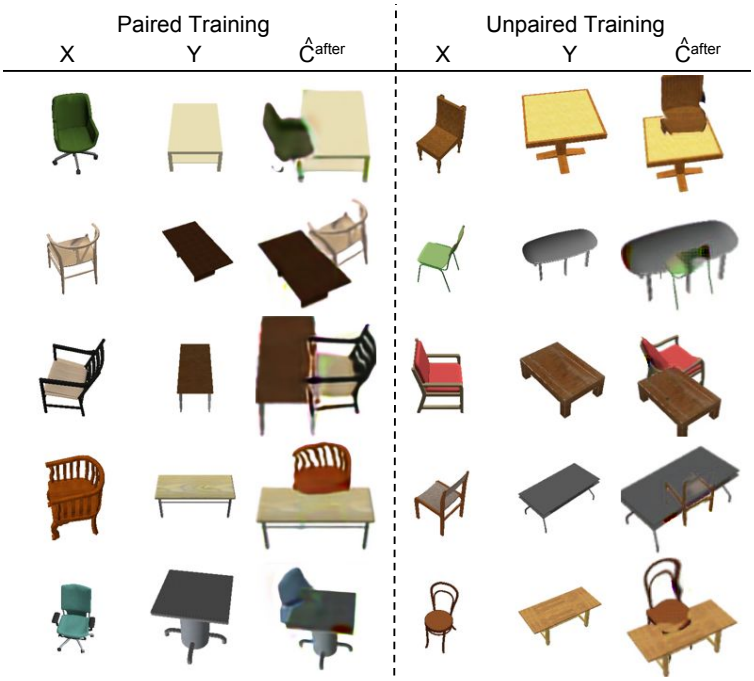


Figure 4.11: Failure test cases for both the paired and unpaired models on the chair-table composition task.

Figure 4.12: Test results on (a) the chair-table and (b) bottle-basket composition tasks trained with either paired or unpaired data. "NN" stands for the nearest neighbor image in the paired training set, and "NoInpaint" shows the results of the unpaired model without the inpainting network. In both paired and unpaired cases, $\hat{c}^{\text{before}}$ and $\hat{c}^{\text{after}}$ show outputs of the generator before and after the ESMR approach, respectively. Also, $\hat{c}_s^{\text{after}}$ represents summation of masked transposed inputs after ESMR.

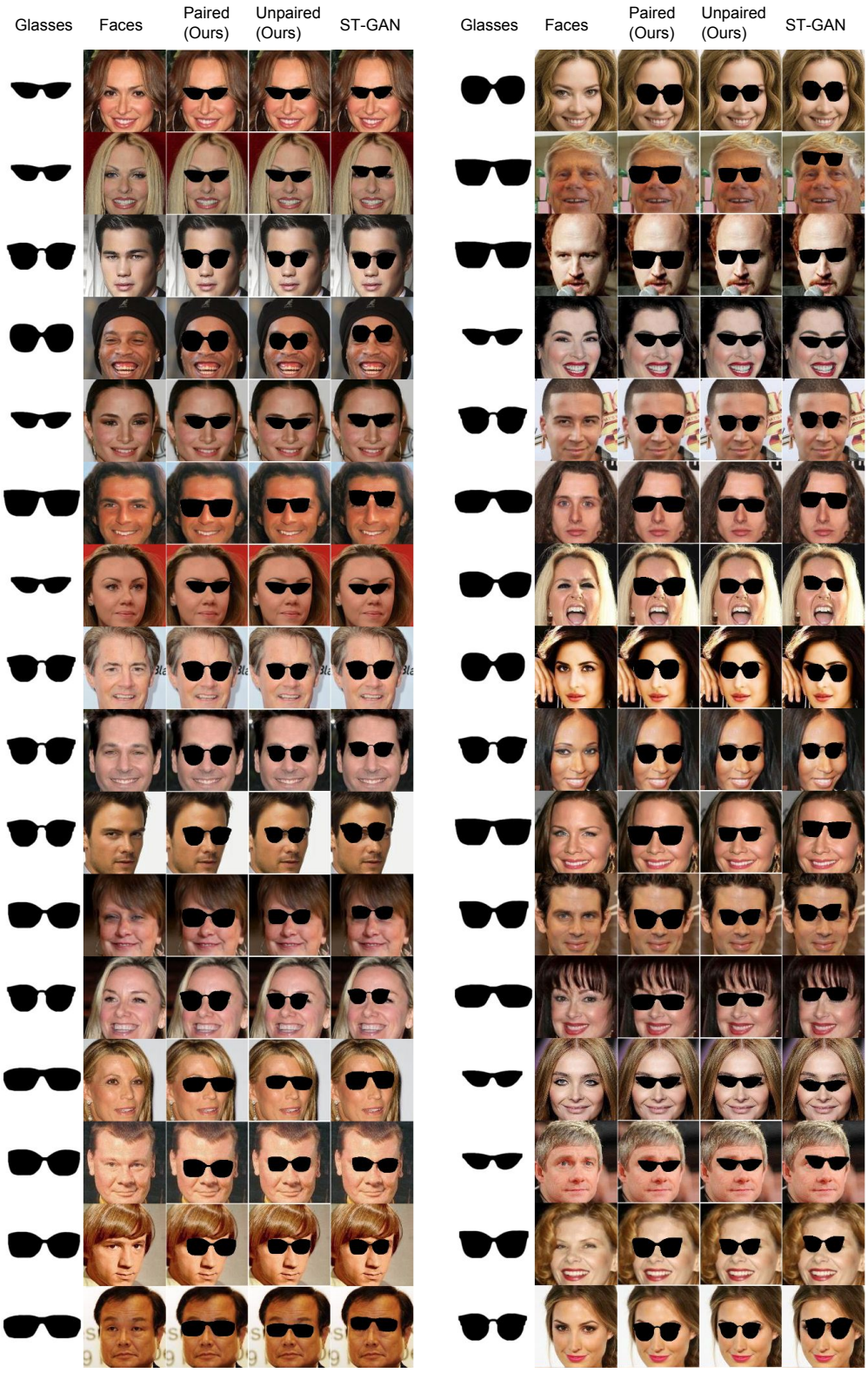


Figure 4.13: Test examples for the face-sunglasses composition task. First two columns show the input sunglasses and face images, 3rd and 4th columns show the output of our compositional GAN for the paired and unpaired models, respectively. Last column shows images generated by the ST-GAN (Lin *et al.*, 2018) model.

Figure 4.14: Test examples for the street scenes-cars composition task. First two columns show the input car and street images, 3rd and 4th columns show the output of our compositional generator before and after the inference meta-refinement step, respectively. The 5th column shows our model's output by directly adding the masked inputs. The 6th and 7th columns correspond with images generated by the ST-GAN (Lin *et al.*, 2018) model and the nearest neighbor training images.

Chapter 5

Discriminator Rejection Sampling

5.1 Introduction

The GAN training procedure is a two-player differentiable game, and the game dynamics are largely what distinguishes the study of GANs from the study of other generative models. These game dynamics have well-known and heavily studied stability issues. Addressing these issues is an active area of research (Mao *et al.,* 2017b; Arjovsky *et al.,* 2017; Gulrajani *et al.,* 2017c; Odena *et al.,* 2018; Li *et al.,* 2017a).

However, we are interested in studying something different: Instead of trying to improve the training procedure, we (temporarily) accept its flaws and attempt to improve the quality of trained generators by post-processing their samples using information from the trained discriminator. It's well known that (under certain very strict assumptions) the equilibrium of this training procedure is reached when sampling from the generator is identical to sampling from the target distribution and the discriminator always outputs 1/2. However, these assumptions don't hold in practice. In particular, GANs as presently trained don't learn to reproduce the target distribution (Arora & Zhang, 2017). Moreover, trained GAN discriminators aren't just identically 1/2 — they can even be used to perform chess-type skill ratings of other trained generators (Olsson *et al.,* 2018).

We ask if the information retained in the weights of the discriminator at the end of the training procedure can be used to "improve" the generator. At face value, this might seem unlikely. After all, if there is useful information left in the discriminator, why doesn't it find its way into the generator via the training procedure? Further reflection reveals that there are many possible reasons. First, the assumptions made in various analyses of the training procedure surely don't hold in practice (e.g. the discriminator and generator have finite capacity and are optimized in parameter space rather than density-space). Second, due to the concrete realization of the discriminator and the generator as neural networks, it may be that it is harder for the generator to model a given distribution than it is for the discriminator to tell that this distribution is not being modeled precisely. Finally, we may simply not train GANs long enough in practice for computational reasons.

In this chapter, we focus on using the discriminator as part of a probabilistic rejection sampling scheme [1]. In particular, we make the following contributions:

- We propose a rejection sampling scheme using the GAN discriminator to approximately correct errors in the GAN generator distribution.
- We show that under quite strict assumptions, this scheme allows us to recover the data distribution exactly.
- We then examine where those strict assumptions break down and design a practical algorithm – called DRS – that takes this into account.
- We conduct experiments demonstrating the effectiveness of DRS. First, as a baseline, we train an improved version of the Self-Attention GAN, improving its performance from the best published Inception Score of 52.52 up to 62.36, and from a Fréchet Inception Distance of 18.65 down to 14.79. We then show that DRS yields further improvement over this baseline, increasing the Inception Score to 76.08 and decreasing the Fréchet Inception Distance to 13.75.

5.2 Background

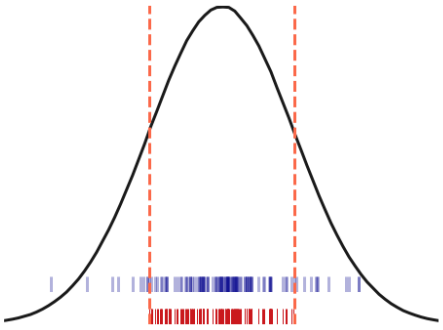
5.2.1 Evaluation metrics: Inception Score (IS) and Fréchet Inception Distance (FID)

The two most popular techniques for evaluating GANs on image synthesis tasks are the Inception Score and the Fréchet Inception Distance. The Inception Score (Salimans *et al.,* 2016) is given by $\exp(\mathbb{E}_x \mathrm{KL}(p(y|x)||p(y)))$, where $p(y|x)$ is the output of a pre-trained Inception classifier (Szegedy *et al.,* 2014). This measures the ability of the GAN to generate samples that the pre-trained classifier confidently assigns to a particular class, and also the ability of the GAN to generate samples from all classes. The Fréchet Inception Distance (FID) (Heusel *et al.,* 2017), is computed by passing samples through an Inception network to yield "semantic embeddings", after which the Fréchet distance is computed between Gaussians with moments given by these embeddings.

5.2.2 Self-Attention GAN

We use a Self-Attention GAN (SAGAN) (Zhang *et al.,* 2019) in our experiments. We do so because SAGAN is considered state of the art on the ImageNet conditional-image-synthesis task (in which images are synthesized conditioned on class identity). SAGAN differs from a vanilla GAN in the following ways: First, it uses large residual networks (He

[1] This chapter is based on joint work done with Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena (Azadi *et al.,* 2019a) presented at ICLR 2019.



Data: generator **G** and discriminator **D**
Result: Filtered samples from **G**
$D^* \leftarrow$ `KeepTraining(`D`)`
$\bar{M} \leftarrow$ `BurnIn(`$\boldsymbol{G}$, D^*`)`
$samples \leftarrow \emptyset$
while $|samples| < N$ **do**
  $x \leftarrow$ `GetSample(`$\boldsymbol{G}$`)`
  $ratio \leftarrow e^{\widetilde{D}^*(x)}$
  $\bar{M} \leftarrow$ `Maximum(`$\bar{M}$, $ratio$`)`
  $p \leftarrow \sigma(\hat{F}(x, \bar{M}, \epsilon, \gamma))$
  $\psi \leftarrow$ `RandomUniform(`$0,1$`)`
  if $\psi \leq p$ **then**
    `Append(`X, $samples$`)`
  end
end

Figure 5.1: **Left:** For a uniform proposal distribution and Gaussian target distribution, the blue points are the result of rejection sampling and the red points are the result of naively throwing out samples for which the density ratio $(p_d(x)/p_g(x))$ is below a threshold. The naive method underrepresents the density of the tails. **Right:** the DRS algorithm. *KeepTraining* continues training using early stopping on the validation set. *BurnIn* computes a large number of density ratios to estimate their maximum. $\widetilde{D}^*$ is the logit of D^*. $\hat{F}$ is as in Equation 5.7. $\bar{M}$ is an empirical estimate of the true maximum M.

et al., 2016) instead of normal convolutional layers. Second, it uses spectral normalization (Miyato *et al.,* 2018) in the generator and the discriminator and a much lower learning rate for the generator than is conventional (Heusel *et al.,* 2017). Third, SAGAN makes use of self-attention layers (Wang *et al.,* 2017b), in order to better model long range dependencies in natural images. Finally, this whole model is trained using a special hinge version of the adversarial loss (Lim & Ye, 2017; Miyato & Koyama, 2018; Tran *et al.,* 2017):

$$\begin{aligned} L_D &= -\mathbb{E}_{(x,y)\sim p_{\text{data}}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{\text{data}}}[\min(0, -1 - D(G(z),y))] \\ L_G &= -\mathbb{E}_{z\sim p_z, y\sim p_{\text{data}}}[D(G(z),y))] \end{aligned} \tag{5.1}$$

5.2.3 Rejection Sampling

Rejection sampling is a method for sampling from a target distribution $p_d(x)$ which may be hard to sample from directly. Samples are instead drawn from a proposal distribution $p_g(x)$, which is easier to sample from, and which is chosen such that there exists a finite value M such that $Mp_g(x) > p_d(x)$ for $\forall x \in \text{domain}(p_d(x))$. A given sample y drawn from p_g is

kept with acceptance probability $p_d(y)/Mp_g(y)$, and rejected otherwise. See the blue points in Figure 5.1 (Left) for a visualization. Ideally, $p_g(x)$ should be close to $p_d(x)$, otherwise many samples will be rejected, reducing the efficiency of the algorithm (MacKay, 2003).

In Section 5.3, we explain how to apply this rejection sampling algorithm to the GAN framework: in brief, we draw samples from the trained generator, $p_g(x)$, and then reject some of those samples using the discriminator to attain a closer approximation to the true data distribution, $p_d(x)$. An independent rejection sampling approach was proposed by (Grover *et al.*, 2018) in the latent space of variational autoencoders for improving samples from the variational posterior.

5.3 Rejection sampling for GANs

In this section we introduce our proposed rejection sampling scheme for GANs (which we call Discriminator Rejection Sampling, or DRS). We'll first derive an idealized version of the algorithm that will rely on assumptions that don't necessarily hold in realistic settings. We'll then discuss the various ways in which these assumptions might break down. Finally, we'll describe the modifications we made to the idealized version in order to overcome these challenges.

5.3.1 Rejection sampling for GANs: the idealized version

Suppose that we have a GAN and our generator has been trained to the point that p_g and p_d have the same support. That is, for all $x \in X$, $p_g(x) \neq 0$ if and only if $p_d(x) \neq 0$. If desired, we can make p_d and p_g have support everywhere in X if we add low-variance Gaussian noise to the observations. Now further suppose that we have some way to compute $p_d(x)/p_g(x)$. Then, if $M = \max_x p_d(x)/p_g(x)$, then $Mp_g(x) > p_d(x)$ for all x, so we can perform rejection sampling with p_g as the proposal distribution and p_d as the target distribution as long as we can evaluate the quantity $p_d(x)/Mp_g(x)$[2]. In this case, we can exactly sample from p_d (Casella *et al.*, 2004), though we may have to reject many samples to do so.

But how can we evaluate $p_d(x)/Mp_g(x)$? p_g is defined only implicitly. One thing we can do is to borrow an analysis from the original GAN paper (Goodfellow *et al.*, 2014), which assumes that we can optimize the discriminator in the space of density functions rather than via changing its parameters. If we make this assumption, as well as the assumption that the discriminator is defined by a sigmoid applied to some function of x and trained with a cross-entropy loss, then by Proposition 1 of that paper, we have that, for any fixed generator and in particular for the generator G that we have when we stop training, training

[2]Why go through all this trouble when we could instead just pick some threshold T and throw out x when $D^*(x) < T$? This doesn't allow us to recover p_d in general. If, for example, there is x' s.t. $p_g(x') > p_d(x') > 0$, we still want some probability of observing x'. See the red points in Figure 5.1 (Left) for a visual explanation.

the discriminator to completely minimize its own loss yields

$$D^*(x) = \frac{p_d(x)}{p_d(x) + p_g(x)} \tag{5.2}$$

We will discuss the validity of these assumptions later, but for now consider that this allows us to solve for $p_d(x)/p_g(x)$ as follows: As noted above, we can assume the discriminator is defined as:

$$D(x) = \sigma(x) = \frac{1}{1 + e^{-\widetilde{D}(x)}}, \tag{5.3}$$

where $D(x)$ is the final discriminator output after the sigmoid, and $\widetilde{D}(x)$ is the logit. Thus,

$$\begin{aligned} D^*(x) = \frac{1}{1 + e^{-\widetilde{D}^*(x)}} &= \frac{p_d(x)}{p_d(x) + p_g(x)} \\ 1 + e^{-\widetilde{D}^*(x)} &= \frac{p_d(x) + p_g(x)}{p_d(x)} \\ p_d(x) + p_d(x)e^{-\widetilde{D}^*(x)} &= p_d(x) + p_g(x) \\ p_d(x)e^{-\widetilde{D}^*(x)} &= p_g(x) \\ \frac{p_d(x)}{p_g(x)} &= e^{\widetilde{D}^*(x)} \end{aligned} \tag{5.4}$$

Now suppose one last thing, which is that we can tractably compute $M = \max_x p_d(x)/p_g(x)$. We would find that $M = p_d(x^*)/p_g(x^*) = e^{\widetilde{D}^*(x^*)}$ for some (not necessarily unique) x^*. Given all these assumptions, we can now perform rejection sampling as promised. If we define $\widetilde{D}^*_M := \widetilde{D}^*(x^*)$, then for any input x, the acceptance probability $p_d(x)/Mp_g(x)$ can be written as $e^{\widetilde{D}^*(x) - \widetilde{D}^*_M} \in [0, 1]$. To decide whether to keep any particular example, we can just draw a random number ψ uniformly from $[0, 1]$ and accept the sample if $\psi < e^{\widetilde{D}^*(x) - \widetilde{D}^*_M}$.

5.3.2 Discriminator Rejection Sampling: the practical scheme

As we hinted at, the above analysis has a number of practical issues. In particular:

1. Since we can't actually perform optimization over density functions, we can't actually compute D^*. Thus, our acceptance probability won't necessarily be proportional to $p_d(x)/p_g(x)$.

2. At least on large datasets, it's quite obvious that the supports of p_g and p_d are not the same. If the support of p_g and p_d has a low volume intersection, we may not even want to compute D^*, because then $p_d(x)/p_g(x)$ would just evaluate to 0 most places.

3. The analysis yielding the formula for D^* also assumes that we can draw infinite samples from p_d, which is not true in practice. If we actually optimized D all the way given a finite data-set, it would give nonzero results on a set of measure 0.

4. In general it won't be tractable to compute M.

5. Rejection sampling is known to have too low an acceptance probability when the target distribution is high dimensional (MacKay, 2003).

This section describes the Discriminator Rejection Sampling (DRS) procedure, which is an adjustment of the idealized procedure, meant to address the above issues.

On the difficulty of actually computing D^*: Given that items 2 and 3 suggest we may not want to compute D^* exactly, we should perhaps not be too concerned with item 1, which suggests that we can't. The best argument we can make that it is OK to approximate D^* is that doing so seems to be successful empirically. We speculate that training a regularized D with SGD gives a final result that is further from D^* but perhaps is less over-fit to the finite sample from p_d used for training. We also hypothesize that the D we end up with will distinguish between "good" and "bad" samples, even if those samples would both have zero density under the true p_d. We qualitatively evaluate this hypothesis in Figures 5.5 and 5.6. We suspect that more could be done theoretically to quantify the effect of this approximation, but we leave this to future work.

On the difficulty of actually computing M: It's nontrivial to compute M, at the very least because we can't compute D^*. In practice, we get around this issue by estimating M from samples. We first run an estimation phase, in which 10,000 samples are used to estimate $\widetilde{D}^*_M$. We then use this estimate in the sampling phase. Throughout the sampling phase we update our estimate of $\widetilde{D}^*_M$ if a larger value is found. It's true that this will result in slight overestimates of the acceptance probability for samples that were processed before a new maximum was found, but we choose not to worry about this too much, since we don't find that we have to increase the maximum very often in the sampling phase, and the increase is very small when it does happen.

Dealing with acceptance probabilities that are too low: Item 5 suggests that we may end up with acceptance probabilities that are too low to be useful when performing this technique on realistic data-sets. If $\widetilde{D}^*_M$ is very large, the acceptance probability $e^{\widetilde{D}^*(x)-\widetilde{D}^*_M}$ will be close to zero, and almost all samples will be rejected, which is undesirable. One simple way to avoid this problem is to compute some $F(x)$ such that the acceptance probability can be written as follows:

$$\frac{1}{1+e^{-F(x)}} = e^{\widetilde{D}^*(x)-\widetilde{D}^*_M} \tag{5.5}$$

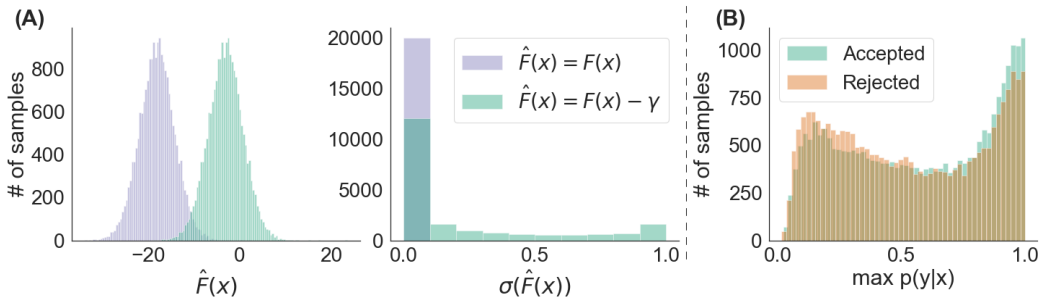


Figure 5.2: **(A)** Histogram of the sigmoid inputs, $\hat{F}(x)$ (left plot), and acceptance probabilities, $\sigma(\hat{F}(x))$ (center plot), on 20K fake samples before (purple) and after (green) adding the constant γ to all $F(x)$. Before adding gamma, 98.9% of the samples had an acceptance probability $<$ 1e-4. **(B)** Histogram of $\max_j p(y_j|x_i)$ from a pre-trained Inception network where $p(y_j|x_i)$ is the predicted probability of sample x_i belonging to the y_j category (from 1,000 ImageNet categories). The green bars correspond to 25,000 accepted samples and the red bars correspond to 25,000 rejected samples. The rejected images are less recognizable as belonging to a distinct class.

If we solve for $F(x)$ in the above equation we can then perform the following rearrangement:

$$\begin{aligned} F(x) &= \widetilde{D}^*(x) - \log(e^{\widetilde{D}^*_M} - e^{\widetilde{D}^*(x)}) \\ &= \widetilde{D}^*(x) - \log(\frac{e^{\widetilde{D}^*_M}}{e^{\widetilde{D}^*_M}} e^{\widetilde{D}^*_M} - \frac{e^{\widetilde{D}^*_M}}{e^{\widetilde{D}^*_M}} e^{\widetilde{D}^*(x)}) = \widetilde{D}^*(x) - \widetilde{D}^*_M - \log(1 - e^{\widetilde{D}^*(x) - \widetilde{D}^*_M}) \end{aligned} \tag{5.6}$$

In practice, we instead compute

$$\hat{F}(x) = \widetilde{D}^*(x) - \widetilde{D}^*_M - \log(1 - e^{\widetilde{D}^*(x) - \widetilde{D}^*_M - \epsilon}) - \gamma \tag{5.7}$$

where ϵ is a small constant added for numerical stability and γ is a hyperparameter modulating overall acceptance probability. For very positive γ, all samples will be rejected. For very negative γ, all samples will be accepted. See Figure 5.2 for an analysis of the effect of adding γ. A summary of our proposed algorithm is presented in Figure 5.1 (Right).

5.4 Experiments

In this section we justify the modifications made to the idealized algorithm. We do this by conducting two experiments in which we show that (according to popular measures of how well a GAN has learned the target distribution) Discriminator Rejection Sampling yields improvements for actual GANs. We start with a toy example that yields insight into how DRS can help, after which we demonstrate DRS on the ImageNet dataset (Russakovsky *et al.*, 2015).

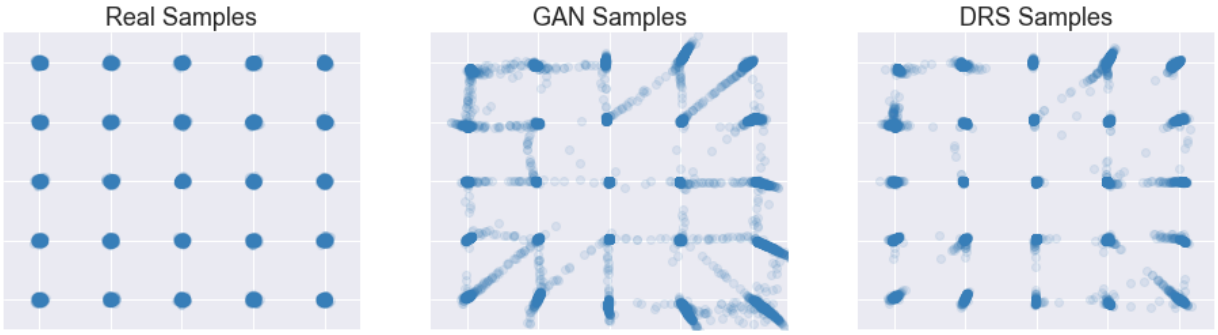


Figure 5.3: Real samples from 25 2D-Gaussian Distributions (*left*) as well as fake samples generated from a trained GAN model without (*middle*) and with DRS (*right*). Results are computed as an average over five models randomly initialized and trained independently.

5.4.1 Mixture of 25 Gaussians

We investigate the impact of DRS on a low-dimensional synthetic data set consisting of a mixture of twenty-five 2D isotropic Gaussian distributions (each with standard deviation of 0.05) arranged in a grid (Dumoulin *et al.*, 2016; Srivastava *et al.*, 2017; Lin *et al.*, 2017b). We train a GAN model where the generator and discriminator are neural networks with four fully connected layers with ReLu activations. The prior is a 2D Gaussian with mean of 0 and standard deviation of 1 and the GAN is trained using the standard loss function. We generate 10,000 samples from the generator with and without DRS. The target distribution and both sets of generated samples are depicted in Figure 5.3. Here, we have set γ dynamically for each batch, to the 95^{th} percentile of $\hat{F}(x)$ for all x in the batch.

To measure performance, we assign each generated sample to its closest mixture component. As in Srivastava *et al.* (2017), we define a sample as "high quality" if it is within four standard deviations of its assigned mixture component. As shown in Table 5.1, DRS increases the fraction of high-quality samples from 70% to 90%. As in Dumoulin *et al.* (2016) and Srivastava *et al.* (2017) we call a mode "recovered" if at least one high-quality sample was assigned to it. Table 5.1 shows that DRS does not reduce the number of recovered modes – that is, it does not trade off quality for mode coverage. It does reduce the standard deviation of the high-quality samples slightly, but this is a good thing in this case (since the standard deviation of the target Gaussian distribution is 0.05). It also confirms that DRS does not accept samples only near the center of each Gaussian but near the tails as well. An ablation study of our proposed algorithm is presented in Appendix 5.5.

5.5 Ablation Study

We have evaluated four different rejection sampling schemes on the mixture-of-Gaussians dataset, represented in Figure 5.4:

Table 5.1: Results with and without DRS on 10,000 generated samples from a model of a 2D grid of Gaussian components.

	# of recovered modes	% "high quality"	std of "high quality" samples
Without DRS	24.8 ± 0.4	70 ± 9	0.11 ± 0.01
With DRS	24.8 ± 0.4	$\mathbf{90 \pm 2}$	$\mathbf{0.10 \pm 0.01}$

1. Always reject samples falling below a hard threshold and DO NOT train the Discriminator to "convergence".
2. Always reject samples falling below a hard threshold and train the Discriminator to convergence.
3. Use probabilistic sampling as in eq 8 and DO NOT train the Discriminator to convergence.
4. Our original DRS algorithm, in which we use probabilistic sampling and train the Discriminator to convergence.

In (1) and (2), we were careful to set the hard threshold so that the actual acceptance rate was the same as in (3) and (4). Broadly speaking, (4) performs best, (3) performs OK but yields less good samples than (4), (2) yields the same number of good samples as (3), but completely fails to sample from 5 of the 25 modes. (1) actually yields the most good samples for the modes it hits, but it only hits 4 modes!

These results show that both continuing to train D so that it can approximate D^* and performing sampling as in equation 5.7, which we have already motivated theoretically, is helpful in practice. For each method, we provide the number of samples within 1, 2, 3 and 4 standard deviations and the number of modes hit in Table 5.2. For reference, we also compute these statistics for the ground truth distribution and the unfiltered samples from GAN.

5.5.1 ImageNet Dataset

Since it is presently the state-of-the-art model on the conditional ImageNet synthesis task, we have reimplemented the Self-Attention GAN (Zhang *et al.*, 2019) as a baseline. After reproducing the results reported by Zhang *et al.* (2019) (with the learning rate of $1e^{-4}$), we fine-tuned a trained SAGAN with a much lower learning rate ($1e^{-7}$) for both generator and discriminator. This improved both the Inception Score and FID significantly as can be seen in the Improved-SAGAN column in Table 5.3. Plots of Inception score and FID during training are given in Figure 5.6(A).

Since SAGAN uses a hinge loss and DRS requires a sigmoid output, we added a fully-connected layer "on top of" the trained discriminator and trained it to distinguish real images

Table 5.2: Ablation study on 10,000 generated samples from a 2D grid of Gaussian components. The third to sixth columns represent % of high-quality samples within x standard deviations. "No FT" stands for the discriminator not being trained to convergence.

	# of recovered modes	% in "1 std"	% in "2 std"	% in "3 std"	% in "4 std"
Ground Truth	25	39.3	86.6	98.9	99.9
Vanilla GAN	25	27.3	53.1	66.2	75.6
Threshold (No FT)	4	38.5	92.6	99.4	99.8
Threshold	20	34.8	70.2	83.6	89.3
DRS (No FT)	25	31.5	60.2	73.6	81.2
DRS	25	35.3	65.8	81.8	89.8

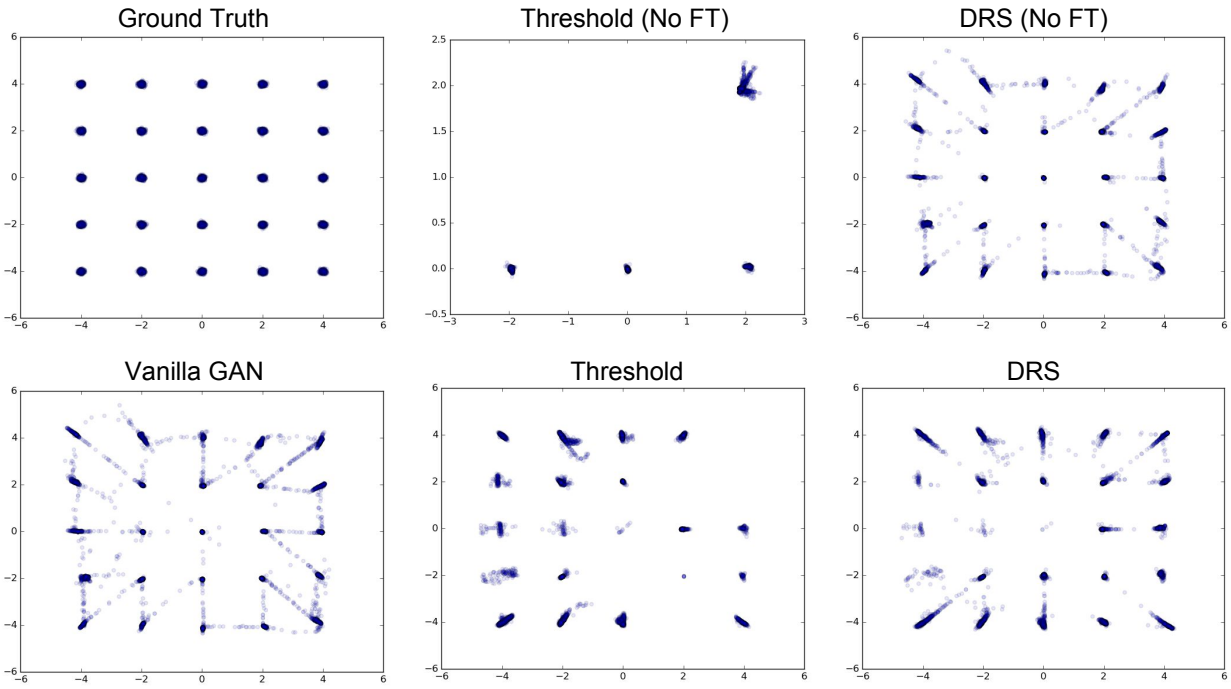


Figure 5.4: Different models generating 10,000 samples from a 2D grid of Gaussian components."No FT" stands for the discriminator not being trained to convergence.

from fake ones using the binary cross-entropy loss. We trained this extra layer with 10,000 generated samples from the model and 10,000 examples from ImageNet.

We then generated 50,000 samples from normal SAGAN and Improved SAGAN with and without DRS, repeating the sampling process 4 times. We set γ dynamically to the 80^{th} percentile of the $F(x)$ values in each batch. The averages of Inception Score and FID

Table 5.3: Results with and without DRS on 50K ImageNet samples. Low FID and high IS are better.

	SAGAN		Improved-SAGAN	
	IS	FID	IS	FID
Without DRS	52.34 ± 0.45	18.21 ± 0.14	62.36 ± 0.35	14.79 ± 0.06
With DRS	61.44 ± 0.09	17.14 ± 0.09	$\mathbf{76.08 \pm 0.30}$	$\mathbf{13.57 \pm 0.13}$

over these four trials are presented in Table 5.3. Both scores were substantially improved for both models, indicating that DRS can indeed be useful in realistic settings involving large data-sets and sophisticated GAN variants.

Qualitative Analysis of ImageNet results: From a pool of 50,000 samples, we visualize the "best" and the "worst" 100 samples based on their acceptance probabilities. Figure 5.5 shows that the subjective visual quality of samples with high acceptance probability is considerably better. Figure 5.2(B) also shows that the accepted images are on average more recognizable as belonging to a distinct class.

We also study the behavior of the discriminator in another way. We choose an ImageNet category randomly, then generate samples from that category until we have found two images $G(z_1), G(z_2)$ such that $G(z_1)$ appears visually realistic and $G(z_2)$ appears visually unrealistic. Here, z_1 and z_2 are the input latent vectors. We then generate many images by interpolating in latent space between the two images according to $z = \alpha z_1 + (1 - \alpha) z_2$ with $\alpha \in \{0, 0.1, 0.2, \ldots, 1\}$. In Figure 5.6, the first and last columns correspond with $\alpha = 1$ and $\alpha = 0$, respectively. The color bar in the figure represents the acceptance probability assigned to each sample. In general, acceptance probabilities decrease from left to right. There is no reason to expect a priori that the acceptance probability should decrease monotonically as a function of the interpolated z, so it says something interesting about the discriminator that most rows basically follow this pattern.

In addition, we represent Inception score as a function of acceptance rate in Figure 5.7-left. Different acceptance rates are achieved by changing γ from the 0^{th} percentile of $F(x)$ (acceptance rate = 100%) to its 90^{th} percentile (acceptance rate = 14%). Decreasing the acceptance rate filters more non-realistic samples and increases the final Inception score. After an specific rate, rejecting more samples does not gain any benefit in collecting a better pool of samples.

Moreover, Figure 5.7-right shows the correlation between the acceptance probabilities that DRS assigns to the synthesized samples and the recognizability of those samples from the view-point of a pre-trained Inception network. The latter is measured by computing $\max_j p(y_j|x_i)$ which is the probability of sample x_i belonging to the category y_j from the 1,000 ImageNet classes. As expected, there is a large mass of the recognizable images accepted with high acceptance probabilities on the top right corner. The small mass of images which cannot

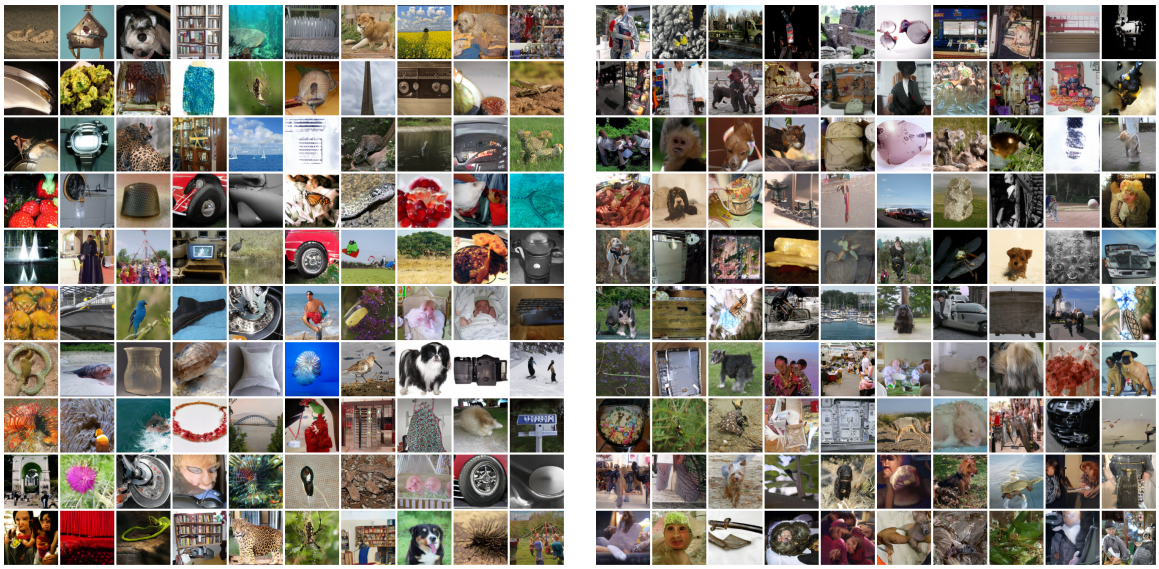


Figure 5.5: Synthesized images with the highest (*left*) and lowest (*right*) acceptance probability scores.

be easily classified into one of the 1,000 categories while having high acceptance probability scores (the top left corner of the graph) can be due to the non-optimal GAN discriminator in practice. Therefore, we expect that improving the discriminator performance boosts the final inception score even more substantially.

5.6 Nearest Neighbors from ImageNet

To confirm that our Discriminator Rejection Sampling is not duplicating the training samples, we show the nearest neighbor of a few visually-realistic generated samples in the ImageNet training data in Figures 5.8-5.15. The nearest neighbors are found based on their fc7 features from the pre-trained VGG16 model.

5.7 Discussion

We have proposed a rejection sampling scheme using the GAN discriminator to approximately correct errors in the GAN generator distribution. We've shown that under strict assumptions, we can recover the data distribution exactly. We've also examined where those assumptions break down and designed a practical algorithm (Discriminator Rejection Sampling) to address that. Finally, we have demonstrated the efficacy of this algorithm on a mixture of Gaussians and on the state-of-the-art SAGAN model.

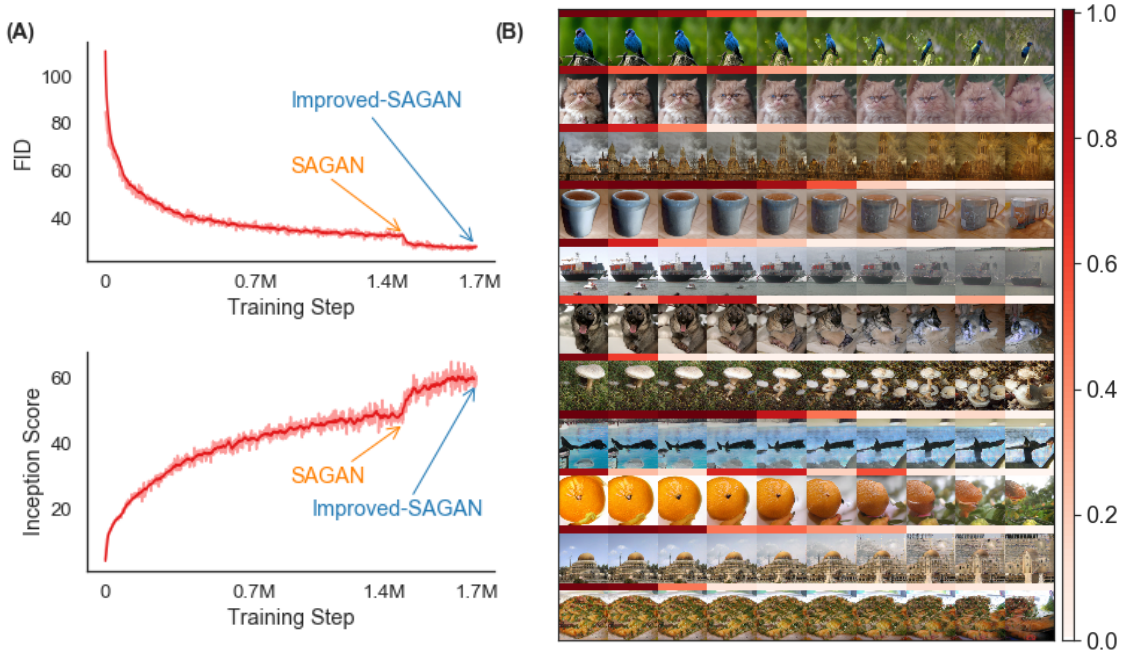


Figure 5.6: **(A)** Inception Score and FID during ImageNet training, computed on 50,000 samples. **(B)** Each row shows images synthesized by interpolating in latent space. The color bar above each row represents the acceptance probabilities for each sample: red for high and white for low. Subjective visual quality of samples with high acceptance probability is considerably better: objects are more coherent and more recognizable as belonging to a specific class. There are fewer indistinct textures, and fewer scenes without recognizable objects.

Opportunities for future work include the following:

- There's no reason that our scheme can only be applied to GAN generators. It seems worth investigating whether rejection sampling can improve e.g. VAE decoders. This seems like it might help, because VAEs may have trouble with "spreading mass around" too much.
- In one ideal case, the critic used for rejection sampling would be a human. Can we use better proxies for the human visual system to improve rejection sampling's effect on image synthesis models?
- It would be interesting to theoretically characterize the efficacy of rejection sampling under the breakdown-of-assumptions that we have described earlier. For instance, if one can't recover D^* but can train some other critic that has bounded divergence from D^*, how does the efficacy depend on this bound?

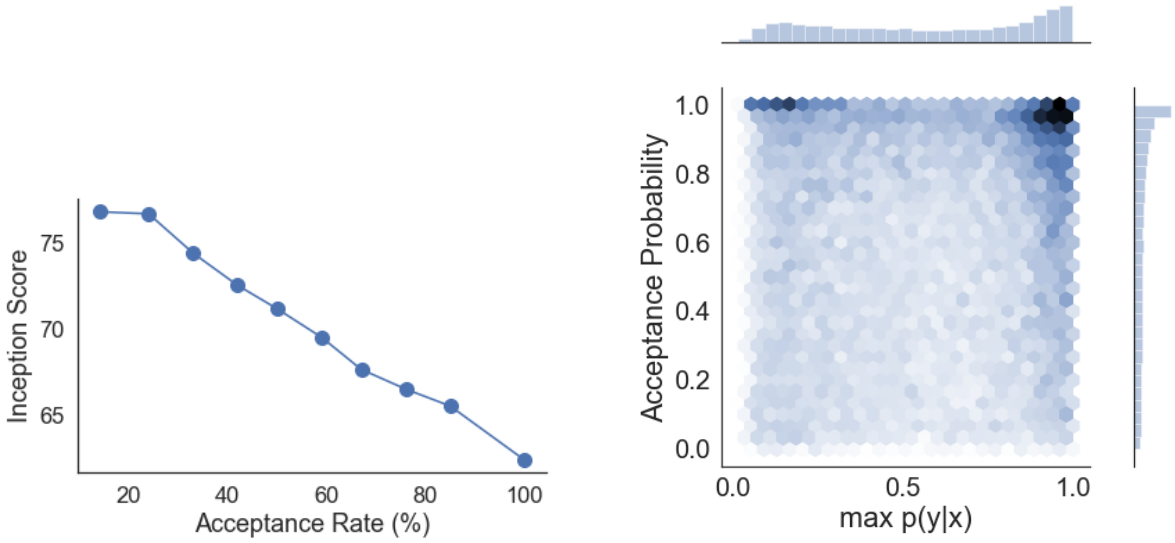


Figure 5.7: Inception Score versus the rate of accepting samples on average (*left*), and the acceptance probability assigned to each sample x_i by DRS versus the maximum probability of belonging to one of the 1K categories based on a pre-trained Inception network, $\max_j p(y_j|x_i)$ (*right*).

Figure 5.8: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Figure 5.9: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Figure 5.10: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Figure 5.11: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Figure 5.12: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Figure 5.13: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Figure 5.14: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Figure 5.15: Nearest neighbors of the top left generated image in ImageNet training set in terms of VGG16 fc7 features

Chapter 6

Discussions

In this thesis, we explored learning the manifold of visual data in multiple structural and compositional domains using Generative Adversarial Networks. Leveraging the structure shared between different elements of each domain, we were able to improve the performance of the state of the art methods in image synthesis. In Chapter 2, we considered text as a visual data and proposed a model for font synthesis and style transfer when only a few characters of a font are provided. The consistent structure of each character across different fonts (such as “A” in the English alphabet) and correlation between specific characters (such as “C" and “G") are the essential factors we considered to design our proposed network. In Chapters 3 and 4, we focused on complex natural scenes and composing a pair of objects into a convincing layout, respectively. In Chapter 5, we investigated the defective contents that can be generated by a trained GAN and proposed a practical method to reject these samples to improve the generator distribution.

Below, we discuss the existing challenges and possible future directions for visual content creation.

Image manipulation There are still several challenges remained unresolved in image editing and compositing. For instance, an automatic approach to modify a human gesture while interacting with an object or performing a specific task is missing from the existing content creation tools. Assume you have the picture of a person walking in a park in a summer day and you intend to transfer the same image to a winter rainy day. Although the existing image-to-image translation models (Zhu *et al.*, 2017b; Isola *et al.*, 2017b) are capable of transferring the style and color scheme of the image, they cannot change the pose of the walking person to hold an umbrella to keep off the rain-drops.

Structural domain shift One advantage of a content creative system is transferring images from one domain associated with a big corpus of visual data to a target domain with a limited set of descriptive images (Hoffman *et al.*, 2018). Although these models have made enhancements in closing the gap between the two domains in terms of their appearance, they cannot change the content of images in the source domain in favor of the

target domain. As an example, consider the domain differences between two street scene datasets, one containing streets with multiple vehicles and the other containing streets with sidewalks and pedestrians. It is still very challenging to add pedestrians to or change the number of vehicles in a scene from the first domain such that it gets closer to the target domain content-wise.

Novel composition of objects or attributes Assume we have a dataset containing different types of birds where each bird category is biased to a specific living environment. For example, birds with open wings and a large body size always appear in the sky while all birds with long thin beaks swim in the water. Can we have a model that composes these different attributes and generates a new type of bird with open wings while swimming in the water? Training a model that generalizes beyond what it has observed in the training domain to apply an unseen composition of objects and attributes could help us remove the existing biases in the datasets and further improve the performance of the computer vision systems relying on them.

Content creation as a threat What we have seen so far are all good uses of content creation, but we should note that content creation can be also seen as a threat. With the enhancements made in content creative systems in specific domains like human faces (Karras *et al.,* 2019) and the emergence of DeepFakes [1], it has been a challenge to rely on what we see on the web. Recent studies (Wang *et al.,* 2020) have shown that content creation is not a big problem for now and CNN-generated images have common artifacts that can be easily detected by a simple classifier. However, we should note that content creation will be a big problem soon where GANs and detectors can be adversaries for each other playing a cat-and-mouse game. Therefore, robust and explainable detectors would be required to be able to trust the extensive collection of visual data accessible online.

[1] https://github.com/deepfakes/faceswap

Appendix A

My Earlier Works in Computer Vision

I started my Ph.D. being amazed by the insight that computer vision provides into key real-world challenges when demand for visual recognition system is higher than ever before. Autonomous driving cars require reliable pedestrian, road, and object detectors. Companies and consumers need person and scene recognition for autonomous photo organization to cope with the explosion of personal photos. Next generation personal robotics demands accurate recognition of basic objects for interaction in a home or office environment. Combination of computer vision and deep learning provides us powerful tools to mimic human ability in recognizing and detecting people and scenes with which we have daily interactions. However, it still remains a challenge to reliably recognize and localize all objects in any arbitrary environment or real-time applications. Training deep convolutional neural networks heavily relies on precisely labeled data sets while many of the available real world data sets contain mislabeled samples. These errors substantially hinder the learning of a very accurate model to recognize objects. In the first two years of my Ph.D., I developed algorithms based on contextual image-level or object-level correlations existing in the data sets to recognize and localize objects more accurately and efficiently.

Noisy supervision in visual systems Learning a deep convolutional model capable of producing robust image representations in presence of noisy supervision is promising due to the availability of millions of online images with user-supplied tags. We proposed a novel auxiliary image regularizer (AIR) to address this issue of deception of training annotations (Azadi *et al.,* 2015). Intuitively, our proposed regularizer exploits the mutual context information among training images and automatically retrieves useful auxiliary examples to collaboratively facilitate the training of the classification model. The AIR regularizer can be deemed as seeking some "nearest neighbors" within the training examples to regularize the fitting of a deep CNN model to noisy samples and improve its classification performance in the presence of noise.

Using regularization is a common practice to enhance robustness of the models. However, an effective regularizer for training a deep CNN model was absent especially for handling a learning problem with faulty labels. Our proposed auxiliary image regularizer was among

the first to introduce an effective regularizer for deep CNN models to handle label noise. We used a group sparse norm to automatically select auxiliary images by constructing groups of input image features. Imposing such group sparsity regularization on the classifier response enables it to actively select the relevant and useful features, which gives higher learning weights to the informative groups in the classification task and forces the weights of irrelevant or noisy groups toward zero. The activated auxiliary images implicitly provide the guiding information for training deep models.

Besides taking benefit from auxiliary image-level information to vaccinate a classification model from noisy labels disruption, object-level correlations in the scenes seems to be the next clue for the enhancement of visual recognition systems. But how should we consider and formulate these interactions for a more challenging problem as object detection?

Label- and instance-level relations Object detection is still more complicated than image classification as it aims at both localizing and classifying objects. Accurate localization of objects in each image requires both well-processed candidate object locations and selected refined boxes with precise locations. Looking at the object detection problem as a summarization and representation task, the set of all predicted bounding boxes per image should be as informative and non-repetitive as possible. To improve location and category specifications of final detected bounding boxes per image, we proposed a novel end-to-end network taking care of both spatial layout and category-level analogy between object proposals without increasing the number of network parameters (Azadi *et al.,* 2017a).

We formulated the discriminative and contextual information as well as mutual relation between boxes into a Determinantal Point Process (DPP) loss function that can be employed on top of any deep network architecture for object detection. Applying our diversity-ignited loss layer reinforces the model to find more accurate object instances with minimum overlap with each other and thus boosts performance of the network in both localizing and classifying objects compared with the state-of-the-art models (Ren *et al.,* 2015). The unified training and inference detection scheme of our proposed model makes it integrable into any system that requires a detection component.

Switch gears to visual content creation While object recognition systems rely on existing data sets and their erroneously labeled samples to recognize object instances, generative networks are capable to pass through a reverse path by generating arbitrary images given their instance ids or an input text (Ramesh *et al.,* 2021). These two opposite directions can be complementary to each other if designed appropriately. The feedback coming from generated samples through a generative network can enhance the performance of an object recognition system and at the same time, the observation from instance-level learning can enrich image generation. I was fascinated by the fundamental changes that GANs introduced in data generation and anticipated a bright horizon for its impact on the field as well as its creative applications, thus, it became the focus of my thesis as described in Chapters 1 to 6.

Bibliography

1. Antoniou, A., Storkey, A. & Edwards, H. Data Augmentation Generative Adversarial Networks. *arXiv preprint arXiv:1711.04340* (2017).
2. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein Generative Adversarial Networks (2017).
3. Arora, S. & Zhang, Y. Do GANs actually learn the distribution? An empirical study. *CoRR* **abs/1706.08224.** arXiv: 1706.08224. http://arxiv.org/abs/1706.08224 (2017).
4. Ashual, O. & Wolf, L. *Specifying object attributes and relations in interactive scene generation* in *ICCV* (2019).
5. Azadi, S., Feng, J. & Darrell, T. *Learning detection with diverse proposals* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), 7149–7157.
6. Azadi, S., Feng, J., Jegelka, S. & Darrell, T. Auxiliary image regularization for deep cnns with noisy labels. *arXiv preprint arXiv:1511.07069* (2015).
7. Azadi, S., Olsson, C., Darrell, T., Goodfellow, I. & Odena, A. Discriminator Rejection Sampling. *arXiv preprint arXiv:1810.06758* (2019).
8. Azadi, S., Pathak, D., Ebrahimi, S. & Darrell, T. Compositional GAN (Extended Abstract): Learning Image-Conditional Binary Composition (2019).
9. Azadi, S., Pathak, D., Ebrahimi, S. & Darrell, T. Compositional GAN: Learning Image-Conditional Binary Composition. *arXiv preprint arXiv:1807.07560* (2019).
10. Azadi, S. *et al.* Multi-Content GAN for Few-Shot Font Style Transfer. *arXiv preprint arXiv:1712.00516* (2017).
11. Azadi, S. *et al.* Multi-Content GAN for Few-Shot Font Style Transfer. *CVPR* (2018).
12. Azadi, S. *et al.* Semantic bottleneck scene generation. *arXiv preprint arXiv:1911.11357* (2019).
13. Baluja, S. Learning Typographic Style. *arXiv preprint arXiv:1603.04000* (2016).
14. Bojchevski, A., Shchur, O., Zügner, D. & Günnemann, S. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816* (2018).

15. Brock, A., Donahue, J. & Simonyan, K. Large scale gan training for high fidelity natural image synthesis (2019).

16. Campbell, N. D. & Kautz, J. Learning a manifold of fonts. *ACM Transactions on Graphics (TOG)* **33,** 91 (2014).

17. Casella, G., Robert, C. P., Wells, M. T., *et al.* in *A Festschrift for Herman Rubin* 342–347 (Institute of Mathematical Statistics, 2004).

18. Chang, A. X. *et al. ShapeNet: An Information-Rich 3D Model Repository* tech. rep. arXiv:1512.03012 [cs.GR] (Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015).

19. Chang, J. & Gu, Y. Chinese Typography Transfer. *arXiv preprint arXiv:1707.04904* (2017).

20. Chen, Q. & Koltun, V. *Photographic image synthesis with cascaded refinement networks* in *ICCV* (2017).

21. Chen, T., Lucic, M., Houlsby, N. & Gelly, S. *On Self Modulation for Generative Adversarial Networks* in *ICLR* (2019).

22. Chen, T., Zhai, X., Ritter, M., Lucic, M. & Houlsby, N. *Self-Supervised GANs via Auxiliary Rotation Loss* in *CVPR* (2019).

23. Chen, X. *et al. InfoGAN: interpretable representation learning by information maximizing Generative Adversarial Nets* in *NIPS* (2016).

24. Cordts, M. *et al. The Cityscapes dataset for semantic urban scene understanding* in *CVPR* (2016).

25. Cordts, M. *et al. The Cityscapes Dataset for Semantic Urban Scene Understanding* in *CVPR* (2016).

26. Dai, A., Qi, C. R. & Nießner, M. Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017).

27. Denton, E. L., Chintala, S., Szlam, A. & Fergus, R. *Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks* in *NIPS* (2015).

28. Dumoulin, V. *et al.* Adversarially Learned Inference. *ArXiv e-prints.* arXiv: 1606.00704 [stat.ML] (June 2016).

29. Flynn, J., Neulander, I., Philbin, J. & Snavely, N. *DeepStereo: Learning to Predict New Views From the World's Imagery* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). http://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Flynn_DeepStereo_Learning_to_CVPR_2016_paper.html.

30. Gatys, L. A., Ecker, A. S. & Bethge, M. *Image style transfer using convolutional neural networks* in *CVPR* (2016), 2414–2423.

31. Goodfellow, I. J. *et al.* Generative Adversarial Networks. *ArXiv e-prints.* arXiv: 1406.2661 [stat.ML] (June 2014).
32. Goodfellow, I. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160* (2016).
33. Goodfellow, I. *et al. Generative adversarial nets* in *NIPS* (2014).
34. Goodfellow, I. *et al. Generative adversarial nets* in *NeurIPS* (2014).
35. Grover, A., Gummadi, R., Lazaro-Gredilla, M., Schuurmans, D. & Ermon, S. *Variational Rejection Sampling* in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics* **84** (2018).
36. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. *Improved training of Wasserstein GANs* in *NeurIPS* (2017).
37. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. *Improved training of wasserstein gans* in *NIPS* (2017).
38. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. Improved Training of Wasserstein GANs. *CoRR* **abs/1704.00028.** arXiv: 1704.00028. http://arxiv.org/abs/1704.00028 (2017).
39. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.
40. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *ArXiv e-prints.* arXiv: 1706.08500 [cs.LG] (June 2017).
41. Heusel, M. *et al. GANs trained by a two time-scale update rule converge to a Nash equilibrium* in *NeurIPS* (2017).
42. Hoffman, J. *et al. Cycada: Cycle-consistent adversarial domain adaptation* in *International conference on machine learning* (2018), 1989–1998.
43. Hong, S., Yang, D., Choi, J. & Lee, H. *Inferring semantic layout for hierarchical text-to-image synthesis* in *CVPR* (2018).
44. Huang, H., Kalogerakis, E. & Marlin, B. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. *Computer Graphics Forum* **34** (2015).
45. Huang, X. & Belongie, S. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *arXiv preprint arXiv:1703.06868* (2017).
46. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004* (2016).
47. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. *Image-to-image translation with conditional adversarial networks* in *CVPR* (2017).

48. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. *Image-to-image translation with conditional adversarial networks* in *CVPR* (2017).
49. Jaderberg, M., Simonyan, K., Zisserman, A., *et al. Spatial transformer networks* in *NIPS* (2015).
50. Jang, E., Gu, S. & Poole, B. *Categorical reparameterization with Gumbel-softmax* in *ICLR* (2017).
51. Johnson, J., Alahi, A. & Fei-Fei, L. *Perceptual losses for real-time style transfer and super-resolution* in *ECCV* (2016).
52. Johnson, J., Gupta, A. & Fei-Fei, L. Image Generation from Scene Graphs. *CVPR* (2018).
53. Kalantari, N. K., Wang, T.-C. & Ramamoorthi, R. Learning-Based View Synthesis for Light Field Cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)* **35** (2016).
54. Kalogerakis, E., Chaudhuri, S., Koller, D. & Koltun, V. A Probabilistic Model of Component-Based Shape Synthesis. *ACM Transactions on Graphics* **31** (2012).
55. Karras, T., Aila, T., Laine, S. & Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).
56. Karras, T., Aila, T., Laine, S. & Lehtinen, J. *Progressive growing of GANs for improved quality, stability, and variation* in *ICLR* (2017).
57. Karras, T., Laine, S. & Aila, T. *A style-based generator architecture for generative adversarial networks* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 4401–4410.
58. Karras, T. *et al. Analyzing and improving the image quality of stylegan* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 8110–8119.
59. Kulkarni, T. D., Whitney, W. F., Kohli, P. & Tenenbaum, J. *Deep convolutional inverse graphics network* in *Advances in Neural Information Processing Systems* (2015), 2539–2547.
60. Kusner, M. J. & Hernández-Lobato, J. M. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051* (2016).
61. Li, C. & Wand, M. *Combining Markov random fields and convolutional neural networks for image synthesis* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 2479–2486.
62. Li, C. & Wand, M. *Precomputed real-time texture synthesis with Markovian generative adversarial networks* in *European Conference on Computer Vision* (2016), 702–716.
63. Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y. & Póczos, B. *MMD GAN: Towards deeper understanding of moment matching network* in *Advances in Neural Information Processing Systems* (2017), 2203–2213.

64. Li, W. *et al. Object-driven text-to-image synthesis via adversarial training* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 12174–12182.

65. Li, Y. *et al.* Universal Style Transfer via Feature Transforms. *arXiv preprint arXiv:1705.08086* (2017).

66. Lian, Z., Zhao, B. & Xiao, J. *Automatic generation of large-scale handwriting fonts via style learning* in *SIGGRAPH ASIA 2016 Technical Briefs* (2016), 12.

67. Liao, J., Yao, Y., Yuan, L., Hua, G. & Kang, S. B. Visual Attribute Transfer through Deep Image Analogy. *arXiv preprint arXiv:1705.01088* (2017).

68. Lim, J. H. & Ye, J. C. Geometric gan. *arXiv preprint arXiv:1705.02894* (2017).

69. Lin, C.-H., Yumer, E., Wang, O., Shechtman, E. & Lucey, S. ST-GAN: Spatial Transformer Generative Adversarial Networks for Image Compositing. *arXiv preprint arXiv:1803.01837* (2018).

70. Lin, K., Li, D., He, X., Zhang, Z. & Sun, M.-T. *Adversarial ranking for language generation* in *NeurIPS* (2017).

71. Lin, Z., Khetan, A., Fanti, G. & Oh, S. PacGAN: The power of two samples in generative adversarial networks. *arXiv preprint arXiv:1712.04086* (2017).

72. Liu, M.-Y., Breuel, T. & Kautz, J. *Unsupervised image-to-image translation networks* in *NIPS* (2017).

73. Liu, M.-Y. & Tuzel, O. *Coupled generative adversarial networks* in *Advances in neural information processing systems* (2016), 469–477.

74. Liu, Z., Luo, P., Wang, X. & Tang, X. *Deep Learning Face Attributes in the Wild* in *ICCV* (2015).

75. Lu, S., Zhu, Y., Zhang, W., Wang, J. & Yu, Y. Neural text generation: past, present and beyond (2018).

76. Lucic, M., Kurach, K., Michalski, M., Gelly, S. & Bousquet, O. *Are GANs Created Equal? A Large-scale Study* in *NeurIPS* (2018).

77. Lucic, M. *et al. High-Fidelity Image Generation With Fewer Labels* in *ICML* (2019).

78. Lun, Z., Kalogerakis, E., Wang, R. & Sheffer, A. Functionality Preserving Shape Style Transfer. *ACM Transactions on Graphics* **35** (2016).

79. Lyu, P. *et al.* Auto-Encoder Guided GAN for Chinese Calligraphy Synthesis. *arXiv preprint arXiv:1706.08789* (2017).

80. MacKay, D. J. *Information theory, inference and learning algorithms* (Cambridge university press, 2003).

81. Maddison, C. J., Mnih, A. & Teh, Y. W. *The concrete distribution: A continuous relaxation of discrete random variables* in *ICLR* (2016).

82. Mao, X. *et al. Least squares generative adversarial networks* in *ICCV* (2017).
83. Mao, X. *et al. Least squares generative adversarial networks* in *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017), 2813–2821.
84. Mirza, M. & Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
85. Miyato, T. & Koyama, M. cGANs with Projection Discriminator. *ArXiv e-prints.* arXiv: 1802.05637 (Feb. 2018).
86. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. Spectral normalization for generative adversarial networks. *ICLR* (2018).
87. Odena, A. *et al.* Is Generator Conditioning Causally Related to GAN Performance? *ArXiv e-prints.* arXiv: 1802.08768 [stat.ML] (Feb. 2018).
88. Odena, A., Olah, C. & Shlens, J. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585* (2016).
89. Odena, A., Olah, C. & Shlens, J. *Conditional image synthesis with auxiliary classifier gans* in *ICML* (2017).
90. Olsson, C., Bhupatiraju, S., Brown, T., Odena, A. & Goodfellow, I. Skill Rating for Generative Models. *ArXiv e-prints.* arXiv: 1808.04888 [stat.ML] (Aug. 2018).
91. Park, E., Yang, J., Yumer, E., Ceylan, D. & Berg, A. C. Transformation-grounded image generation network for novel 3D view synthesis. *arXiv preprint arXiv:1703.02921* (2017).
92. Park, T., Liu, M.-Y., Wang, T.-C. & Zhu, J.-Y. *Semantic image synthesis with spatially-adaptive normalization* in *CVPR* (2019).
93. Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T. & Efros, A. *Context Encoders: Feature Learning by Inpainting* in *CVPR* (2016).
94. Phan, H. Q., Fu, H. & Chan, A. B. *Flexyfont: Learning transferring rules for flexible typeface synthesis* in *Computer Graphics Forum* **34** (2015), 245–256.
95. Qiao, T., Zhang, J., Xu, D. & Tao, D. *MirrorGAN: Learning Text-to-image Generation by Redescription* in *CVPR* (2019).
96. Radford, A., Metz, L. & Chintala, S. *Unsupervised representation learning with deep convolutional generative adversarial networks* in *ICLR* (2016).
97. Ramesh, A. *et al.* Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092* (2021).
98. Reed, S. *et al. Generative Adversarial Text-to-Image Synthesis* in *ICML* (2016).
99. Reed, S. E. *et al. Learning what and where to draw* in *NIPS* (2016).
100. Ren, S., He, K., Girshick, R. & Sun, J. *Faster R-CNN: Towards real-time object detection with region proposal networks* in *NIPS* (2015).

101. Russakovsky, O. *et al.* ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **115,** 211–252 (2015).

102. Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O. & Gelly, S. *Assessing Generative Models via Precision and Recall* in *NeurIPS* (2018).

103. Salimans, T. *et al.* Improved Techniques for Training GANs. *ArXiv e-prints.* arXiv: 1606.03498 [cs.LG] (June 2016).

104. Salimans, T. *et al. Improved techniques for training gans* in *NIPS* (2016).

105. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

106. Srivastava, A., Valkoz, L., Russell, C., Gutmann, M. U. & Sutton, C. *VEEGAN: Reducing mode collapse in GANs using implicit variational learning* in *Advances in Neural Information Processing Systems* (2017), 3308–3318.

107. Sung, M., Kim, V. G., Angst, R. & Guibas, L. Data-driven Structural Priors for Shape Completion. *Transactions on Graphics (Proc. of SIGGRAPH Asia)* (2015).

108. Sunkavalli, K., Johnson, M. K., Matusik, W. & Pfister, H. *Multi-scale image harmonization* in *ACM Transactions on Graphics (TOG)* (2010).

109. Suveeranont, R. & Igarashi, T. *Example-Based Automatic Font Generation.* in *Smart Graphics* (2010), 127–138.

110. Szegedy, C. *et al.* Going Deeper with Convolutions. *CoRR* **abs/1409.4842.** http://arxiv.org/abs/1409.4842 (2014).

111. Tenenbaum, J. B. & Freeman, W. T. *Separating style and content* in *Advances in neural information processing systems* (1997), 662–668.

112. Tran, D., Ranganath, R. & Blei, D. M. Hierarchical Implicit Models and Likelihood-Free Variational Inference. *ArXiv e-prints.* arXiv: 1702.08896 [stat.ML] (Feb. 2017).

113. Tran, L., Yin, X. & Liu, X. *Disentangled representation learning gan for pose-invariant face recognition* in *CVPR* **4** (2017), 7.

114. Tsai, Y.-H. *et al. Deep image harmonization* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

115. Upchurch, P., Snavely, N. & Bala, K. From A to Z: supervised transfer of style and content using deep neural network generators. *arXiv preprint arXiv:1603.02003* (2016).

116. Volokitin, A., Konukoglu, E. & Van Gool, L. *Decomposing Image Generation into Layout Prediction and Conditional Synthesis* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020), 372–373.

117. Wang, S.-Y., Wang, O., Zhang, R., Owens, A. & Efros, A. A. *CNN-generated images are surprisingly easy to spot... for now* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 8695–8704.

118. Wang, T.-C. *et al.* High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *arXiv preprint arXiv:1711.11585* (2017).

119. Wang, T.-C. *et al. High-resolution image synthesis and semantic manipulation with conditional gans* in *CVPR* (2018).

120. Wang, X., Girshick, R., Gupta, A. & He, K. Non-local neural networks (2017).

121. Wang, X. & Gupta, A. Generative Image Modeling using Style and Structure Adversarial Networks. *ECCV* (2016).

122. Xue, S., Agarwala, A., Dorsey, J. & Rushmeier, H. Understanding and improving the realism of image composites. *ACM Transactions on Graphics (TOG)* (2012).

123. Yang, J., Kannan, A., Batra, D. & Parikh, D. LR-GAN: Layered recursive generative adversarial networks for image generation. *arXiv preprint arXiv:1703.01560* (2017).

124. Yang, S., Liu, J., Lian, Z. & Guo, Z. Awesome Typography: Statistics-Based Text Effects Transfer. *arXiv preprint arXiv:1611.09026* (2016).

125. Yu, F., Koltun, V. & Funkhouser, T. *Dilated Residual Networks* in *CVPR* (2017).

126. Yu, L., Zhang, W., Wang, J. & Yu, Y. *Seqgan: Sequence generative adversarial nets with policy gradient* in *AAAI* (2017).

127. Zhang, H., Goodfellow, I., Metaxas, D. & Odena, A. *Self-attention generative adversarial networks* in *ICML* (2019).

128. Zhang, H. *et al. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks* in *ICCV* (2017).

129. Zhang, H. *et al. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks* in *ICCV* (2017).

130. Zhang, R., Isola, P. & Efros, A. A. *Colorful Image Colorization* in *ECCV* (2016).

131. Zhao, B., Wu, X., Cheng, Z.-Q., Liu, H. & Feng, J. Multi-View Image Generation from a Single-View. *arXiv preprint arXiv:1704.04886* (2017).

132. Zhou, B., Wang, W. & Chen, Z. *Easy generation of personal Chinese handwritten fonts* in *Multimedia and Expo (ICME), 2011 IEEE International Conference on* (2011), 1–6.

133. Zhou, B. *et al. Scene parsing through ade20k dataset* in *CVPR* (2017).

134. Zhou, T., Tulsiani, S., Sun, W., Malik, J. & Efros, A. A. *View synthesis by appearance flow* in *ECCV* (2016).

135. Zhu, J.-Y., Krahenbuhl, P., Shechtman, E. & Efros, A. A. *Learning a discriminative model for the perception of realism in composite images* in *ICCV* (2015).

136. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* in *ICCV* (2017).

137. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* in *ICCV* (2017).