HYBRID GENERATIVE MODELS FOR 2D AND 3D COMPUTER VISION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Omid Poursaeed

August 2020

HYBRID GENERATIVE MODELS FOR 2D AND 3D COMPUTER VISION

Omid Poursaeed, Ph.D.

Cornell University 2020

Deep Learning has made tremendous progress in the last decade, making breakthroughs in visual perception and imagination. Discriminative models have achieved human-level performance on several tasks. Generative models have also shown great promise for 2D and 3D synthesis; however, their performance still lags behind discriminative models. Several approaches have been proposed to enhance the performance of generative models. One promising direction is leveraging multiple representations to guide the synthesis process.

In this dissertation, we explore how generative models can benefit from hybrid representations, in which a stronger representation guides a weaker one. We propose models for 3D synthesis conditioned on images or point clouds, label-conditioned 2D generation, and 2.5D motion generation. A major drawback of current generative models is that they require gigantic datasets for training. We present a few-shot synthesis method for decreasing the model's dependence on labeled data. Finally, we discuss adversarial applications of generative models in which adversaries abuse visual realism to deceive humans and machines.

BIOGRAPHICAL SKETCH

Omid Poursaeed is a PhD candidate at Cornell University working with Prof. Serge Belongie. His research interests are in Computer Vision with focus on Generative Models, Adversarial Learning and 3D Deep Learning. He received his B.Sc. (with honor) from Sharif University of Technology in 2015. He is also the recipient of Jacobs Fellowship from Cornell University and DLI Doctoral Fellowship from Cornell Tech.

To my parents, Parvin and Hamidreza, and my brother Vahid.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help of a large number of people. First, I would like to thank members of the SE3 Computer Vision group for their support and helpful discussions and collaborations: Serge Belongie, Mohammad Moghimi, Xun Huang, Zekun Hao, Ryan Benmalek, Yin Cui, Kevin Musgrave, Kimberly Wilber, Boaguang Shi, Hani Altwaijri, Rui Qian and Boyi Li. I also had the chance to collaborate with other researchers at Cornell including Bharath Hariharan, Davis Wertheimer and Yixuan Li. I want to extend my gratitude to my thesis committee members, Thorsten Joachims and Ramin Zabih, for their feedback during the process of my thesis proposal and defense. I also appreciate the support from Prof. Qing Zhao and Prof. Lang Tong during the admission process. I had the opportunity to work with several undergraduate and masters students: Tianxing Jiang, Isay Katsman, Bicheng Gao, Quintessa Qiao, Yordanos Goshu, Nayun Xu and Xiaoyan Wu. Many of the papers in this dissertation were made possible with their help and efforts.

I had a great experience doing internships at several research labs in industry. I was fortunate to work with Vladimir G. Kim during two internships at Adobe Research. He helped me explore new research directions and his dedication and support were exemplary. I also appreciate the help from my other mentors including Eli Shechtman, Matthew Fisher, Jun Saito and Noam Aigerman. I would like to thank Ser-Nam Lim who gave me the opportunity to intern at Facebook AI, and Robinson Piramuthu, Hadi Kiapour and Shuai Kyle Zheng who supported me in my first internship at eBay research. Finally, I want to dedicate my dissertation to my parents, Parvin and Hamidreza, and my brother Vahid, without whom this journey would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Remarkable progress has been made in the field of computer vision since the resurgence of deep neural networks. State-of-the-art models can rival human performance in several discriminative tasks given enough supervision. However, these models require large amount of training data and are incapable of capturing the underlying data distribution. This has spurred interest in generative models that learn to capture the statistical distribution of data, allowing us to synthesize novel samples from the learned distribution. Generative models enable visual imagination which is a hallmark of computer vision and has applications such as unsupervised learning, domain adaptation and image editing. While generative models have shown great promise for 2D and 3D synthesis, they still fail when there are large variations in the data distribution. Various approaches have been proposed to enhance the performance of these models including better training schemes, loss functions and architectures.

In this dissertation, we explore how generative models can benefit from hybrid representations, in which a stronger representation guides a weaker one. First, we propose efficient 3D generative models by encouraging consistency across two 3D representations: surface atlas and implicit functions. We extend this idea to 2D by enforcing consistency between generative and discriminative representations. Then, we present a hybrid image-mesh model for 2.5D motion generation. State-of-the-art generative models require gigantic datasets for training. In order to reduce this dependency, we discuss a few-shot generative model leveraging autoencoders and affine transformations of input samples. Finally, we explore how visual realism provided by generative models can be

abused by adversaries to create realistic images that mislead machine learning models. We propose two approaches for generating adversarial examples based on additive perturbations and searching in the latent space of GANs.

In Chapter 2, we propose a hybrid surface representation for generative 3D modeling [202]. In the 3D domain, point clouds, meshes, voxel grids, surface atlas and implicit functions are commonly used to define shapes. We present an approach for encouraging consistency between surface atlas and implicit functions. We use both representations in a hybrid manner and add novel consistency losses that couple the two representations during joint training to ensure that the atlas embedding aligns with the implicit level-set. We show that these two representations reinforce one another. This results in smoother normals that are more consistent with the ground truth for the atlas representation, while also maintaining lower chamfer distance in the implicit representation. We demonstrate the advantage of our joint representation by using it to train 3D-shape autoencoders and reconstructing a surface from a single image. The resulting implicit and explicit surfaces are consistent with each other and quantitatively and qualitatively superior to either of the branches trained in isolation.

In Chapter 3, we present a hybrid generative-discriminative model for label-conditioned image generation [117]. For 2D images, a pixel grid is the most common representation, while we can also consider features learned by a ConvNet. In the latter case, these learned features can guide a generative model such that the generated features are consistent with the discriminative ones. Using this intuition, we propose a hybrid model consisting of a stack of GANs trained to invert the hierarchical representations of a discriminative network. Based on visual inspection, Inception scores and visual Turing test, we demon-

strate that our model is able to generate images of much higher quality than GANs without stacking.

In Chapter 4, we discuss a motion generation framework using an auxiliary mesh representation [206]. While most existing motion generation methods operate solely in the image space, we use a hybrid image-mesh representation which leads to more realistic deformations and interpolations. We express pose changes as deformations of a layered 2.5D template mesh, and devise a novel architecture that learns to predict mesh deformations matching the template to a target image. This enables us to extract a common low-dimensional structure from a diverse set of character poses. We combine recent advances in differentiable rendering as well as mesh-aware models to successfully align common template even if only a few character images are available during training. We demonstrate that our generative model can be used to synthesize in-between frames and to create data-driven deformation. Our template fitting procedure also outperforms state-of-the-art generic techniques for detecting image correspondences.

In Chapter 5, we propose a few-shot generation method that generalizes to novel classes from few examples. While state-of-the-art generative models have achieved impressive performance, they require gigantic datasets for training, which can be costly and time-consuming to collect. We introduce a strong, efficient, unsupervised baseline for few-shot image generation. Our key insight derives from our finding that while the latent space of powerful generative models, such as VAEs and GANs, does not generalize to new classes, the representation learned by vanilla autoencoders generalizes extremely well. Unfortunately, autoencoders cannot synthesize novel class images. To remedy this, we intro-

duce a new training method to encourage a more interpretable and informative latent space, which allows for meaningful interpolation. We demonstrate on three different settings (handwritten characters, faces and general objects) that the resulting *Interpolative Autoencoder* achieves simple, robust, highly general, and completely unsupervised few-shot image generation.

While generative models can be helpful in several applications such as unsupervised and semi-supervised learning, domain adaptation, data augmentation and others, they can also be exploited by adversaries who abuse visual realism to deceive humans and machines. In Chapter 6, we propose novel generative models for creating adversarial examples, slightly perturbed images resembling natural images but maliciously crafted to fool machine learning models [205]. Our approach can produce image-agnostic and image-dependent perturbations for both targeted and non-targeted attacks. We also demonstrate that similar architectures can achieve impressive results in fooling both classification and semantic segmentation models, obviating the need for hand-crafting attack methods for each task. We improve the state-of-the-art performance in universal perturbations by leveraging generative models in lieu of current iterative methods. Our attacks are considerably faster than iterative and optimization-based methods at inference time. Moreover, we are the first to present effective targeted universal perturbations.

In Chapter 7, we present an approach to create on-the-manifold adversarial modifications instead of additive perturbations [204]. We generate unrestricted adversarial examples by manipulating fine-grained aspects of image generation. Unlike existing unrestricted attacks that typically hand-craft geometric transformations, we learn stylistic and stochastic modifications leverag-

ing state-of-the-art generative models. This allows us to manipulate an image in a controlled, fine-grained manner without being bounded by a norm threshold. We demonstrate that our attacks can bypass certified defenses, yet our adversarial images look indistinguishable from natural images as verified by human evaluation. Adversarial training can be used as an effective defense without degrading performance of the model on clean images.

Chapter 8 concludes the dissertation and discusses future directions.

CHAPTER 2

COUPLING EXPLICIT AND IMPLICIT SURFACE REPRESENTATIONS FOR GENERATIVE 3D MODELING

2.1 Introduction

Many applications rely on a neural network to generate a 3D geometry [104, 60], where early approaches used point clouds [1], uniform voxel grids [162], or template mesh deformations [27] to parameterize the outputs. The main disadvantage of these representations is that they rely on a pre-selected discretization of the output, limiting network's ability to focus its capacity on high-entropy regions. Several recent geometry learning techniques address this limitation by representing 3D shapes as *continuous* mappings over vector spaces. Neural networks learn over a manifold of these mappings, creating a mathematically elegant and visually compelling generative models. Two prominent alternatives have been proposed recently.

The explicit surface representation defines the surface as an atlas – a collection of *charts*, which are maps from 2D to 3D, $\{f_i : \Omega_i \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3\}$, with each chart mapping a 2D patch Ω_i into a part of the 3D surface. the surface S is then defined as the union of all 3D patches, $S = \cup_i f_i(\Omega_i)$. In the context of neural networks, this representation has been explored in a line of works considering atlas-based architectures [97, 287] which exactly represent surfaces by having the network predict the charts $\{f_i^{\mathbf{x}}\}$, where the network also takes latent code, $\mathbf{x} \in \mathcal{X}$, as input, to describe the target shape. These predicted charts can then be queried at arbitrary 2D points, enabling approximating the resulting surface with, e.g., a polygonal mesh, by densely sampling the 2D domain with the vertices of a

mesh, and then mapping the resulting mesh to 3D via $f_i^{\mathbf{x}}$. This reconstruction step is suitable for an end-to-end learning pipeline where the loss is computed over the resulting surface. It can also be used as an input to a differentiable rasterization layer in case image-based losses are desired. On the other hand, the disadvantage of atlas-based methods is that the resulting surfaces tend to have visual artifacts due to inconsistencies at patch boundaries, and because even small perturbations in the mapping can lead to high-frequency normal variations, which are a major source of shading artifacts.

The implicit surface representation defines a volumetric function $g : \mathbb{R} \rightarrow \mathbb{R}^3$. This function is called an implicit function, with the surface S defined as its zero level set, $S = \{p \in \mathbb{R}^3 | g(p) = 0\}$. Many works train networks to predict implicit functions, either as signed distance fields [200, 49], or simply occupancy values [182]. They also typically use shape descriptor, $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$, as additional input to express different shapes: $g^{\hat{\mathbf{x}}}$. These methods tend to produce visually appealing results since they are smooth with respect to the 3D volume. They suffer from two main disadvantages; first, they do not immediately produce a surface, making them less suitable for end-to-end pipeline with surface-based or image-based losses; second, as observed in [200, 49, 182], their final output tends to produce a higher surface-to-surface distance to ground truth than atlas-based methods.

In this paper we propose to use both representations in a hybrid manner, with our network predicting both an explicit atlas $\{f_i\}$ and an implicit function g. For the two branches of the two representations we use the AtlasNet [97] and OccupancyNet [182] architectures. We use the same losses used to train these two networks (chamfer distance and occupancy, respectively) while adding

novel consistency losses that couple the two representations during joint training to ensure that the atlas embedding aligns with the implicit level-set. We show the two representations reinforce one another: OccupancyNet learns to shift its level-set to align it better with the ground truth surface, and AtlasNet learns to align the embedded points and their normals to the level-set. This results in smoother normals that are more consistent with the ground truth for the atlas representation, while also maintaining lower chamfer distance in the implicit representation. Our framework enables a straightforward extraction of the surface from the explicit representation, as opposed to the more intricate marching-cube-like techniques required to extract a surface from the implicit function. This enables us to add image-based losses on the output of a differentiable rasterizer. Even though these losses are only measured over AtlasNet output, we observe that they further improve the results for *both* representations, since the improvements propagate to OccupancyNet via consistency losses. Another advantage of reconstructing surfaces from the explicit representation is that it is an order of magnitude faster than running marching cubes on the implicit representation. We demonstrate the advantage of our joint representation by using it to train 3D-shape autoencoders and reconstruct a surface from a single image. The resulting implicit and explicit surfaces are consistent with each other and quantitatively and qualitatively superior to either of the branches trained in isolation.

2.2 Related Work

We review existing representations for shape generation that are used within neural network architectures. While target application and architecture details

might vary, in many cases an alternative representation can be seamlessly integrated into an existing architecture by modifying the layers of the network that are responsible for generating the output.

Generative networks designed for images operate over regular 2D grids and can directly extend to 3D voxel occupancy grids [162, 36, 90, 36, 60]. These models tend to be coarse and blobby, since the size of the output scales cubically with respect to the desired resolution. Hierarchical models [105, 223] alleviate this problem, but they still tend to be relatively heavy in the number of parameters due to multiple levels of resolution. A natural remedy is to only focus on surface points, hence point-based techniques were proposed to output a tensor with a fixed number of 3D coordinates [1, 246]. Very dense point clouds are required to approximate high curvature regions and fine-grained geometric details, and thus, point-based architectures typically generate coarse shapes. While polygonal meshes allow non-even tessellation, learning over this domain even with modest number of vertices remains a challenge [59]. One can predict vertex positions of a template [252], but this can only apply to analysis of very homogenous datasets. Similarly to volumetric cases, one can adaptively refine the mesh [269], using graph unpooling layers to add more mesh elements. The main limitation of these techniques is that they discretize the domain in advance and allocate same network capacity to each discrete element. Even hierarchical methods only provide opportunity to save time by not exploring finer elements in feature-less regions. In contrast, continuous, functional representations enable the network to learn the discretization of the output domain.

The explicit continuous representations view 3D shapes as 2D charts embedded in 3D [97, 287]. These atlas-based techniques tend to have visual artifacts

related to non-smooth normals and patch misalignments. For homogeneous shape collections, such as human bodies, this can be remedied by replacing 2D charts with a custom template (e.g., a human in a T-pose) and enforce strong regularization priors (e.g., isometry) [96], however, the choice of such a template and priors limits expressiveness and applicability of the method to non-homogeneous collections with diverse geometry and topology of shapes.

Another alternative is to use a neural network to model a space probing function that predicts occupancy [182] or clamped signed distance field [200, 49] for each point in a 3D volume. Unfortunately, these techniques cannot be trained with surface-based losses and thus tend to perform worse with respect to surface-to-surface error metrics.

Implicit representations also require marching cubes algorithm [173] to reconstruct the surface. Note that unlike explicit representation, where every sample lies on the surface, marching cubes requires sampling off-surface points in the volume to extract the level set. We found that this leads to a surface reconstruction algorithm that is about an order of magnitude slower than an explicit technique. This additional reconstruction step, also makes it impossible to plugin the output of the implicit representation into a differentiable rasterizer (e.g., [172, 130, 164]. We observe that using differentiable rasterizer to enforce additional image-based losses can improve the quality of results. Moreover, adding these losses just for the explicit output, still propagates the improvements to the implicit representation via the consistency losses.

In theory, one could use differentiable version of marching cubes [159] for reconstructing a surface from an implicit representation, however, this has not been used by prior techniques due to cubic memory requirements of this step

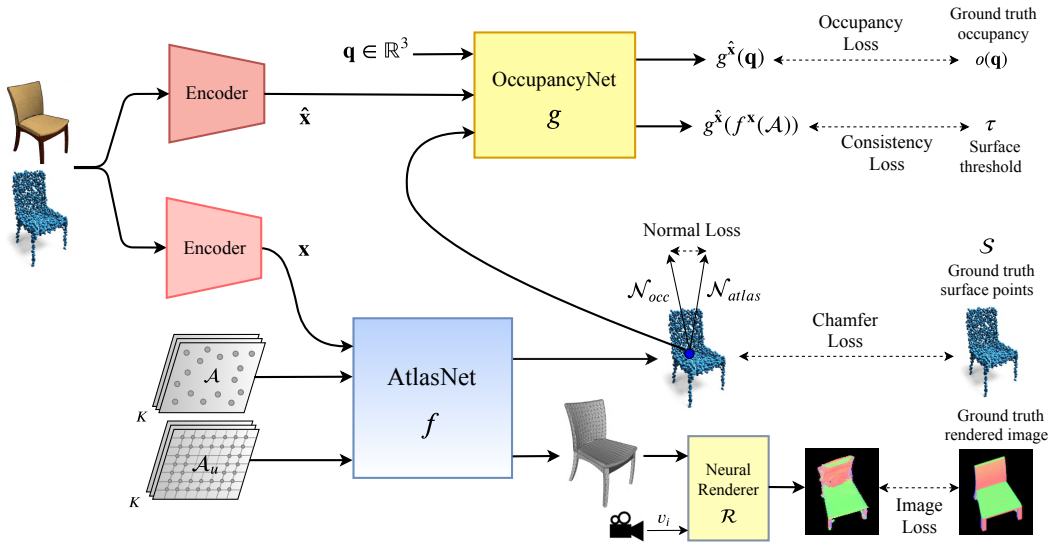


Figure 2.1: Model architecture. AtlasNet and OccupancyNet branches of the hybrid model are trained with Chamfer and occupancy losses as well as consistency losses for aligning surfaces and normals. A novel image loss is introduced to further improve generation quality.

(essentially, it would limit the implicit formulation to 32^3 grids as argued in prior work [182]). Several recent techniques use ray-casting to sample implicit functions for image-based losses. Since it is computationally intractable to densely sample the volume, these methods either interpolate a sparse set of samples [166] or use LSTM to learn the ray marching algorithm [240]. Both solutions are more computationally involved than simply projecting a surface point using differentiable rasterizer, as enabled by our technique.

2.3 Approach

We now detail the architecture of the proposed network, as well as the losses used within the training to enforce consistency across the two representations.

2.3.1 Architecture

Our network simultaneously outputs two surface representations. These two representations are generated from two branches of the network, where each branch uses a state-of-the-art architecture for the target representation.

For the explicit branch, we use AtlasNet [97]. AtlasNet represents K charts with neural functions $\{f_i^{\mathbf{x}}\}_{i=1}^{K}$, where each function takes a shape descriptor vector, $\mathbf{x} \in \mathcal{X}$, and a query point in the unit square, $\mathbf{p} \in [0,1]^2$, and outputs a point in 3D, i.e., $f_i^{\mathbf{x}} : [0,1]^2 \rightarrow \mathbb{R}^3$. We also denote the set of 3D points achieved by mapping all 2D points in $\mathcal{A} \subset [0,1]^2$ as $f^{\mathbf{x}}(\mathcal{A})$.

For the implicit branch, we use OccupancyNet [182], learning a neural function $g^{\hat{\mathbf{x}}} : \mathbb{R}^3 \rightarrow [0,1]$, which takes a query point $q \in \mathbb{R}^3$ and a shape descriptor vector $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$ and outputs the occupancy value. The point q is considered occupied (i.e., inside the shape) if $g^{\hat{\mathbf{x}}} \geq \tau$, where we set $\tau = 0.2$ following the choice of OccupancyNet.

2.3.2 Loss Functions

Our approach centers around losses that ensure geometric consistency between the output of the OccupancyNet and AtlasNet modules. We employ these consistency loses along with each branch's original fitting loss (Chamfer and occupancy loss) that was used to train the network in its original paper. Furthermore, we take advantage of AtlasNet's output lending itself to differentiable rendering in order to incorporate a rendering-based loss. These losses are summarized in Figure 2.1 and detailed below.

Consistency losses. First, to favor consistency between the explicit and implicit representations, we observe that the surface generated by AtlasNet should align with the τ-level set of OccupancyNet:

$$g^{\hat{\mathbf{x}}}\left(f_i^{\mathbf{x}}(\mathbf{p})\right) = \tau, \tag{2.1}$$

for all charts $f_i^{\mathbf{x}}$ and at every point $\mathbf{p} \in [0,1]^2$. Throughout this subsection we assume that $\mathbf{x}$ and $\hat{\mathbf{x}}$ are describing the same shape.

This observation motivates the following surface consistency loss:

$$\mathcal{L}_{\text{consistency}} = \sum_{\mathbf{p}\in\mathcal{A}} \mathcal{H}(g^{\hat{\mathbf{x}}}(f_i^{\mathbf{x}}(\mathbf{p})), \tau). \tag{2.2}$$

where $\mathcal{H}(\cdot,\cdot)$ is the cross entropy function, and $\mathcal{A}$ is the set of sample points in $[0,1]^2$.

Second, we observe that the gradient of the implicit representation should align with the surface normal of the explicit representation. The normals for both representations are differentiable and their analytic expressions can be defined in terms of gradients of the network. For AtlasNet, we compute surface normal at a point $\mathbf{p}$ as follows:

$$\mathcal{N}_{\text{atlas}} = \frac{\partial f_i^{\mathbf{x}}}{\partial u} \times \frac{\partial f_i^{\mathbf{x}}}{\partial v}\bigg|_{\mathbf{p}} \tag{2.3}$$

The gradient of OccupancyNet's at a point $\mathbf{q}$, is computed as:

$$\mathcal{N}_{\text{occ}} = \nabla_{\mathbf{q}} g^{\hat{\mathbf{x}}}(\mathbf{q}) \tag{2.4}$$

We now define the normal consistency loss by measuring the misalignment in their directions (note that the values are normalized to have unit magnitude):

$$\mathcal{L}_{\text{norm}} = \left| 1 - \frac{\mathcal{N}_{\text{atlas}}}{||\mathcal{N}_{\text{atlas}}||} \cdot \frac{\mathcal{N}_{\text{occ}}}{||\mathcal{N}_{\text{occ}}||} \right| \tag{2.5}$$

We evaluate this loss only at surface points predicted by the explicit representation (i.e., $\mathcal{N}_{\text{occ}}$ is evaluated at $\mathbf{q} = f^{\mathbf{x}}(\mathbf{p}), \mathbf{p} \in \mathcal{A}$).

Fitting losses. Each branch also has its own fitting loss, ensuring it adheres to the input geometry. We use the standard losses used to train each of the two surface representations in previous works.

For the explicit branch, we measure the distance between the predicted surface and the ground truth in standard manner, using Chamfer distance:

$$\mathcal{L}_{\text{chamfer}} = \sum_{\mathbf{p}\in\mathcal{A}} \min_{\hat{\mathbf{p}}\in S} |f^{\mathbf{x}}(\mathbf{p}) - \hat{\mathbf{p}}|^2 + \sum_{\hat{\mathbf{p}}\in S} \min_{\mathbf{p}\in\mathcal{A}} |f^{\mathbf{x}}(\mathbf{p}) - \hat{\mathbf{p}}|^2, \tag{2.6}$$

where $\mathcal{A}$ be a set of points randomly sampled from the K unit squares of the charts (here $f^{\mathbf{x}}$ uses one of the neural functions f_i depending on which of the K charts the point $\mathbf{p}$ came from). S is a set of points that represent the ground truth surface.

For the implicit branch, given a set of points $\{\mathbf{q}_i\}_{i=1}^N$ sampled in 3D space, with $o(\mathbf{q}_i)$ denoting their ground-truth occupancy values, the occupancy loss is defined as:

$$\mathcal{L}_{\text{occ}} = \sum_{i=1}^{N} \mathcal{H}(g^{\hat{\mathbf{x}}}(\mathbf{q}_i), o(\mathbf{q}_i)) \tag{2.7}$$

Finally, for many applications visual quality of a rendered 3D reconstruction plays a very important role (e.g., every paper on this subject actually presents a rendering of the reconstructed model for qualitative evaluations). Rendering implicit functions requires complex probing of volumes, while output of the explicit representation can be directly rasterized into an image. Thus, we chose to only include an image-space loss for the output of the explicit branch, comparing its differentiable rendering to the image produced by rendering the ground

truth shape. Note that this loss to still improves the representation learned by the implicit branch due to consistency losses.

To compute the image-space loss we first reconstruct a mesh from the explicit branch. In particular, we sample a set of 2D points $\mathcal{A}_u$ on a regular grid for each of the K unit squares. Each grid defines topology of the mesh, and mapping the corners of all grids with $f^{\mathbf{x}}$ gives a triangular 3D mesh that can be used with most existing differentiable rasterizers $\mathcal{R}$ (we use our own implementation inspired by SoftRas [165]). We render 25 images from different viewpoints produced by the cross product of 5 elevation and 5 azimuth uniformly sampled angles.

The image loss is defined as:

$$\mathcal{L}_{\text{img}} = \frac{1}{25} \sum_{i=1}^{25} \left\| \mathcal{R}(f^{\mathbf{x}}(\mathcal{A}_u), v_i) - \mathcal{R}(\mathcal{M}_{gt}, v_i) \right\|^2 \tag{2.8}$$

in which v_i is the i^{th} viewpoint and $\mathcal{M}_{gt}$ represents the ground truth mesh. Our renderer $\mathcal{R}$ outputs a normal map image (based on per-face normals), since they capture the shape better than silhouettes or gray-shaded images.

Our final loss is a weighted combination of the fitting and consistency losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{occ}} + \alpha \cdot \mathcal{L}_{\text{chamfer}} + \beta \cdot \mathcal{L}_{\text{img}} + \gamma \cdot \mathcal{L}_{\text{consistency}} + \delta \cdot \mathcal{L}_{\text{norm}} \tag{2.9}$$

We use $\alpha = 2.5 \times 10^4, \beta = 10^3, \gamma = 0.04$, and $\delta = 0.05$ in all experiments.

2.3.3 Pipeline and Training

Figure 2.1 illustrates the complete pipeline for training and inference: given an input image or a point cloud, the two encoders encode the input to two shape

features, $\mathbf{x}$ and $\hat{\mathbf{x}}$. For the AtlasNet branch, a set of points $\mathcal{A} \subset [0,1]^2$ is randomly sampled from K unit squares. These points are concatenated with the shape feature $\mathbf{x}$ and passed to AtlasNet. The Chamfer loss is computed between $f^{\mathbf{x}}(\mathcal{A})$ and the ground truth surface points, per Equation 2.6. For the OccupancyNet branch, similarly to [182], we uniformly sample a set of points $\{q_i\}_{i=1}^{N} \subset \mathbb{R}^3$ inside the bounding box of the object and use them to train OccupancyNet with respect to the fitting losses. To compute the image loss, the generated mesh $f^{\mathbf{x}}(\mathcal{A}_u)$ and the ground truth mesh $\mathcal{M}_{gt}$ are normalized to a unit cube prior to rendering.

For the consistency loss, the occupancy function $g^{\hat{\mathbf{x}}}$ is evaluated at the points generated by AtlasNet, $f^{\mathbf{x}}(\mathcal{A})$ and then penalized as described in Equation (2.2). AtlasNet's normals are evaluated at the sample points $\mathcal{A}$. OccupancyNet's normals are evaluated at the corresponding points, $f^{\mathbf{x}}(\mathcal{A})$. These are then plugged into the loss described in Equation (2.5). We train AtlasNet and OccupancyNet jointly with the loss function in equation (2.9), thereby coupling the two branches to one-another via the consistency losses.

Since we wish to show the merit of the hybrid approach, we keep the two branches' networks' architecture and training setup identical to the one used in the previous works that introduced those two networks. For AtlasNet, we sample random 100 points from each of $K = 25$ patches during training. At inference time, the points are sampled on 10×10 regular grid for each patch. For OccupancyNet, we use the 2500 uniform samples provided by the authors [182]. We use the Adam optimizer [134] with learning rates of 6×10^{-4} and 1.5×10^{-4} for AtlasNet (f) and OccupancyNet (g) respectively.

Metric	**Chamfer-L_1($\times 10^{-1}$)**									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	1.05	1.34	**0.91**	1.03	0.96	1.10	0.95	1.08	1.01	1.17
bench	1.38	1.50	**1.23**	1.26	1.27	1.31	1.26	1.29	1.32	1.38
cabinet	1.75	1.53	1.53	**1.47**	1.57	1.49	1.55	1.49	1.61	1.50
car	1.41	1.49	**1.28**	1.31	1.33	1.37	1.33	1.36	1.37	1.42
chair	2.09	2.06	1.96	**1.95**	2.02	2.01	1.99	1.99	2.04	2.03
display	1.98	2.58	**1.89**	2.14	1.92	2.24	1.90	2.19	1.94	2.29
lamp	3.05	3.68	**2.91**	3.02	2.93	3.09	2.91	3.06	2.99	3.21
sofa	1.77	1.81	**1.56**	1.58	1.61	1.63	1.59	1.61	1.68	1.71
table	1.90	1.82	1.73	**1.72**	1.80	1.78	1.78	1.76	1.83	1.79
telephone	1.28	1.27	**1.17**	1.18	1.22	1.21	1.19	1.19	1.24	1.24
vessel	1.51	2.01	**1.42**	1.53	1.46	1.60	1.46	1.58	1.48	1.69
mean	1.74	1.92	**1.60**	1.65	1.64	1.71	1.63	1.69	1.68	1.77
Metric	**Normal Consistency ($\times 10^{-2}$)**									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	83.6	84.5	85.5	**85.7**	85.3	85.6	84.8	85.3	84.3	85.0
bench	77.9	81.4	81.4	**82.5**	80.9	82.2	80.4	81.9	79.9	81.7
cabinet	85.0	88.4	88.3	**89.1**	88.1	89.0	87.2	88.7	86.8	88.6
car	83.6	85.2	86.2	**86.8**	85.8	86.5	85.3	86.0	84.9	85.8
chair	79.1	82.9	83.5	**84.0**	83.1	83.7	82.4	83.4	82.0	83.2
display	85.8	85.7	**87.0**	86.9	86.7	86.6	86.3	86.1	86.0	85.9
lamp	69.4	75.1	74.9	**76.0**	74.7	75.9	73.3	75.6	72.8	75.4
sofa	84.0	86.7	87.2	**87.5**	86.9	87.4	86.4	87.1	85.9	86.9
table	83.2	85.8	86.3	**87.4**	86.0	87.1	85.3	86.4	84.9	86.1
telephone	92.3	93.9	94.0	**94.5**	93.8	94.4	93.6	94.2	93.3	94.1
vessel	75.6	79.7	79.2	**80.6**	78.9	80.4	77.7	80.0	77.4	79.9
mean	81.8	84.5	84.9	**85.5**	84.6	85.4	83.9	85.0	83.5	84.8

Table 2.1: Quantitative results on single-view reconstruction. Variants of our hybrid model, with AtlasNet (AN) and OccupancyNet (ON) branches, are compared with vanilla AtlasNet and OccupancyNet using Chamfer-L_1 distance and Normal Consistenty score.

2.4 Results

We evaluate our network's performance on single view reconstruction as well as on point cloud reconstruction, using the same subset of shapes from ShapeNet

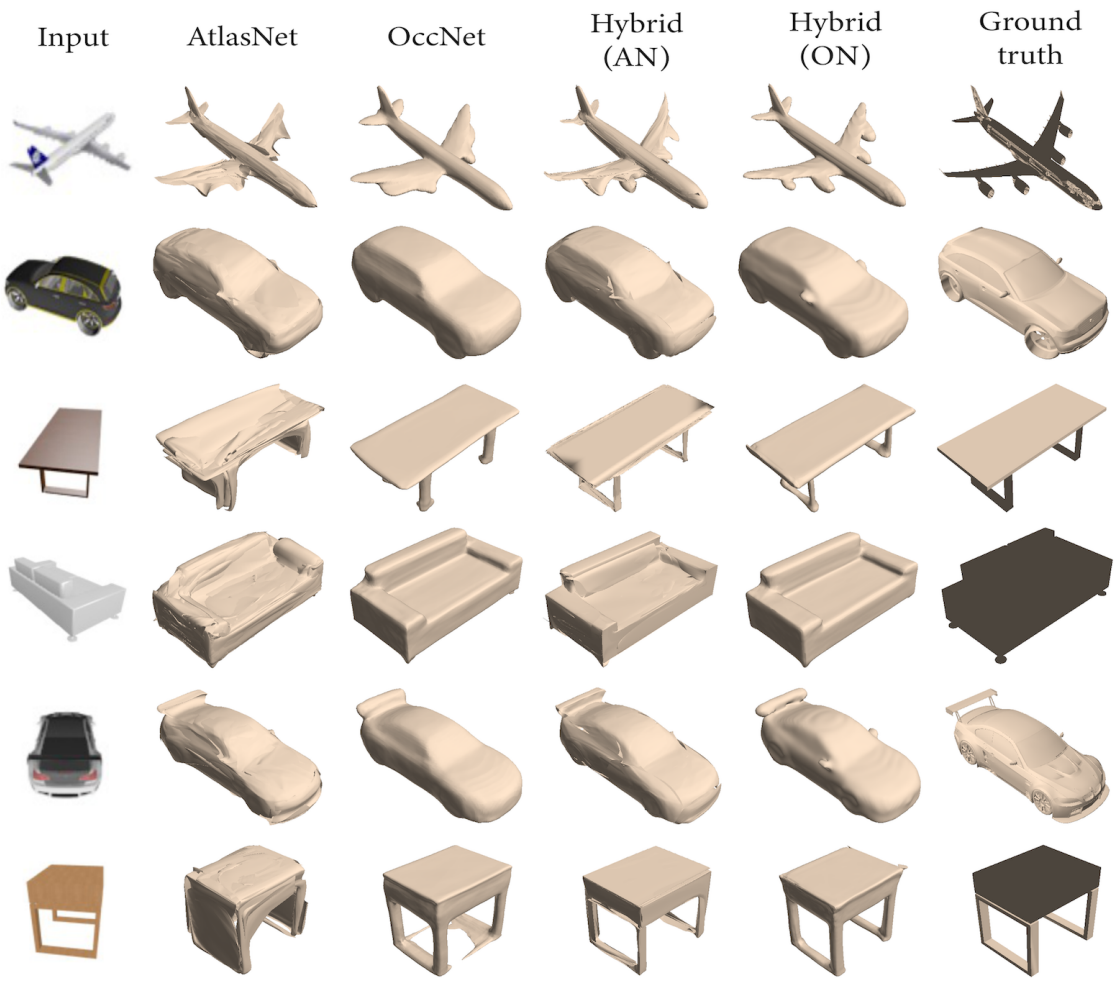


Figure 2.2: Comparison of meshes generated by vanilla AtlasNet and OccupancyNet with the AtlasNet (AN) and OccupancyNet (ON) branches of our hybrid model. Compared to their vanilla counterparts, our AtlasNet branch produces results with significantly less oscillatory artifacts, and our OccupancyNet branch produces results that better preserve thin features such as the chair legs.

[43] as used in Choy et al. [51]. For both tasks, following prior work (e.g., [97, 182]), we use simple encoder-decoder architectures. Similarly to [182], we quantitatively evaluate the results using the chamfer-L_1 distance and normal consistency score. The chamfer-L_1 distance is the mean of the accuracy and completeness metrics, with accuracy being the average distance of points on

the output mesh to their nearest neighbors on the ground truth mesh, and completeness similarly with switching the roles of source and target point sets. The normal consistency score is the mean absolute dot product of normals in the predicted surface and normals at the corresponding nearest neighbors on the true surface.

Single view reconstruction. To reconstruct geometry from a single-view image, we use a ResNet-18 [106] encoder each of the two branches to encode an input image into a shape descriptor which is then fed to the branch. We then train end-to-end with the loss (2.9), on the dataset of images provided by Choy et al. [51], using batch size of 7. Note that with our method the surface can be reconstructed from either the explicit AtlasNet (AN) branch or the implicit OccupancyNet (ON) branch. We show qualitative results (Figure 2.2) and error metrics (Table 2.1) for both branches. The surface generated by our AtlasNet branch, "Hybrid (AN)," provides a visually smoother surface than vanilla AtlasNet (AN), which is also closer to the ground truth – both in terms of chamfer distance, as well as its normal-consistency score. The surface generated by our OccupancyNet branch, "Hybrid (ON)", similarly yields a more accurate surface in comparison to vanilla OccupancyNet (ON). We observe that the hybrid implicit representation tends to be better at capturing thinner surfaces (e.g., see table legs in Figure 2.2) than its vanilla counterpart; this improvement is exactly due to having the implicit branch indirectly trained with the chamfer loss propagated from the AtlasNet branch.

Point cloud reconstruction. As a second application, we train our network to reconstruct a surface for a sparse set of 2500 input points. We encode the set of points to a shape descriptor using a PointNet [210] encoder for each of the

Metric	**Chamfer-L_1($\times 10^{-3}$)**									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	0.17	0.19	**0.15**	0.16	0.16	0.17	0.16	0.17	0.17	0.18
bench	0.49	**0.23**	0.31	0.25	0.34	0.25	0.33	0.24	0.37	0.24
cabinet	0.73	0.56	0.55	**0.51**	0.58	0.52	0.61	0.54	0.63	0.54
car	0.49	0.54	**0.42**	0.44	0.46	0.47	0.44	0.47	0.47	0.50
chair	0.52	**0.30**	0.36	0.33	0.39	0.33	0.38	0.33	0.41	0.32
display	0.61	0.45	0.47	**0.39**	0.50	0.41	0.48	0.40	0.52	0.42
lamp	1.53	1.35	1.42	**1.31**	1.46	1.33	1.44	1.31	1.49	1.34
sofa	0.32	0.34	**0.25**	0.26	0.28	0.29	0.26	0.27	0.30	0.31
table	0.58	0.45	0.46	**0.41**	0.48	0.42	0.47	0.42	0.50	0.43
telephone	0.22	0.12	0.14	**0.10**	0.15	0.10	0.16	0.11	0.18	0.11
watercraft	0.74	**0.38**	0.53	0.42	0.57	0.41	0.54	0.42	0.61	0.40
mean	0.58	0.45	0.46	**0.41**	0.49	0.43	0.48	0.43	0.51	0.44
Metric	**Normal Consistency ($\times 10^{-2}$)**									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	85.4	89.6	88.3	**90.1**	88.1	90.0	87.5	89.7	87.1	89.6
bench	81.5	**87.1**	85.6	86.7	85.3	86.8	85.0	86.8	84.7	86.9
cabinet	87.0	90.6	89.3	**91.1**	89.1	91.0	88.4	90.8	88.1	90.8
car	84.7	87.9	87.5	**88.6**	87.1	88.5	86.7	88.1	86.1	88.0
chair	84.7	**94.9**	88.7	94.3	88.2	94.5	87.6	94.5	87.1	94.6
display	89.7	91.9	91.8	**92.4**	91.5	92.3	91.0	92.2	90.8	92.1
lamp	73.1	79.5	77.1	**79.8**	76.9	79.7	76.4	79.6	76.0	79.5
sofa	89.1	92.2	91.8	**92.8**	91.6	92.7	91.3	92.5	91.0	92.4
table	86.3	91.0	88.8	**91.4**	88.6	91.3	88.3	91.3	88.0	91.2
telephone	95.9	97.3	97.4	**98.0**	97.2	97.8	96.8	97.6	96.5	97.5
watercraft	82.1	**86.7**	84.9	86.5	84.7	86.5	84.3	86.7	84.0	86.7
mean	85.4	89.8	88.3	**90.2**	88.0	90.1	87.6	90.0	87.2	89.9

Table 2.2: Quantitative results on auto-encoding. Variants of our hybrid model are compared with vanilla AtlasNet and OccupancyNet.

two branches, and train the encoder-decoder architecture end-to-end with the loss (2.9). We train with the same points as [182] with a batch size of 10. See results in Figure 2.3 and Table 2.2. As in the single view reconstruction task, the hybrid method surpasses the vanilla, single-branch architectures on average. While there are three categories in which vanilla OccupancyNet performs better,

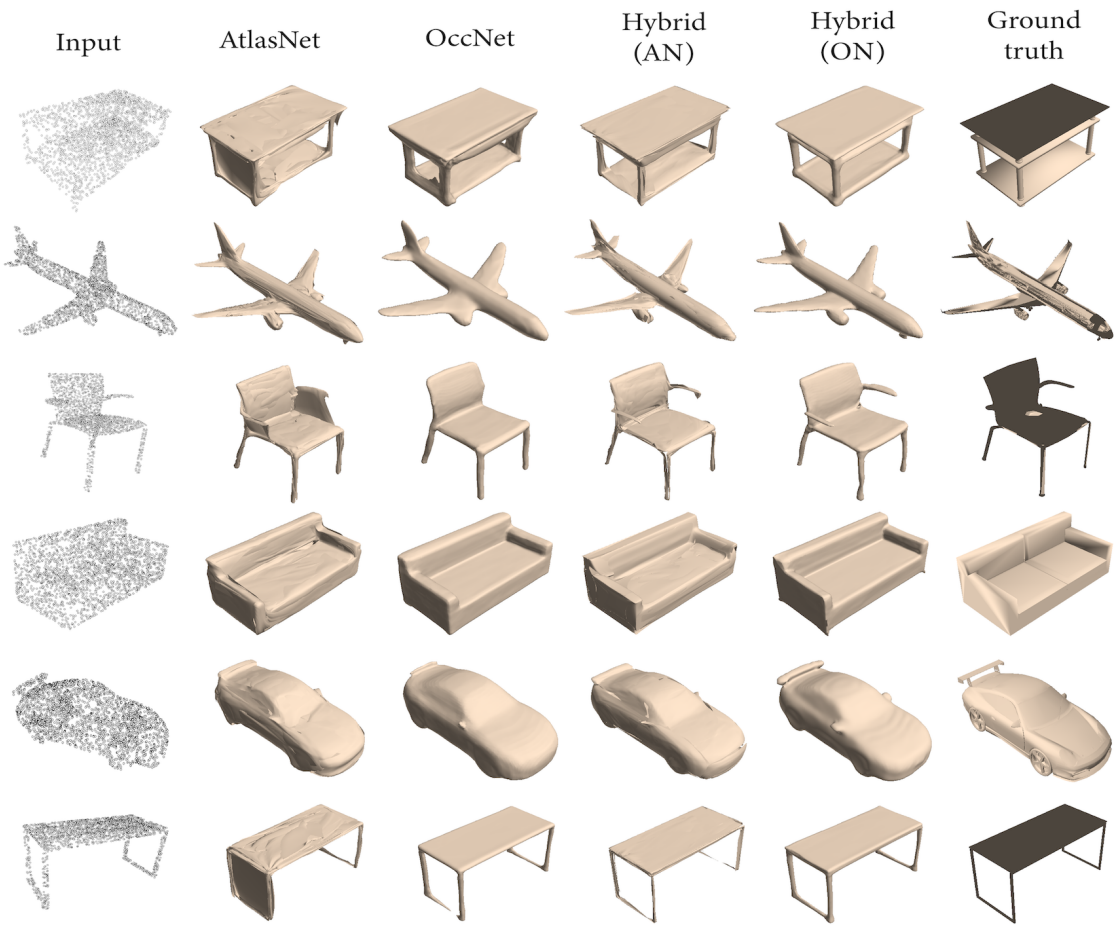


Figure 2.3: Reconstructing surfaces from point clouds. Our hybrid approach better reproduces fine features and avoids oscillatory surface normals.

we note that Hybrid AtlasNet consistently outperforms vanilla AtlasNet on all categories. This indicates that the hybrid training is mostly beneficial for the implicit representation, and *always* beneficial for the explicit representation; this in turn offers a more streamlined surface reconstruction process.

Ablation study on the loss functions. We evaluate the importance of the different loss terms via an ablation study for both tasks (see Tables 2.1, 2.2). First, we exclude the image-based loss function $\mathcal{L}_{\text{img}}$. Note that even without this loss,

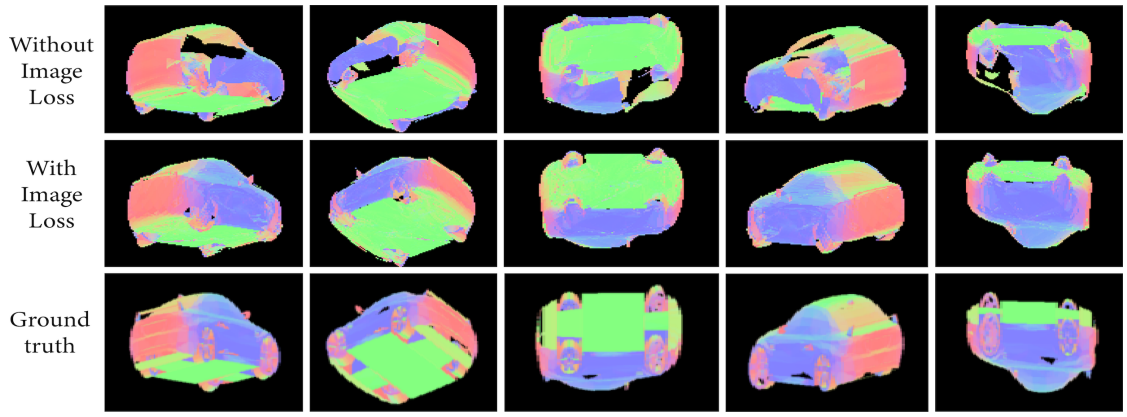


Figure 2.4: Impact of the image loss. Rendered images from different viewpoints are shown for models trained with and without the image loss. Evidently, the image loss significantly improves the similarity of the output to the ground truth.

hybrid AtlasNet still outperforms vanilla AtlasNet, attributing these improvements mainly to the consistency losses. Removing the normal-consistency loss $\mathcal{L}_{\text{norm}}$ results in decreased quality of reconstructions, especially the accuracy of normals in the predicted surface. Finally, once both terms $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$ are removed, we observe that still, the single level-set consistency term is sufficient to boost the performance within the hybrid training.

We also provide qualitative examples on how each loss term affects the quality of the generated point clouds and meshes. Figure 2.4 illustrates impact of the image loss. Generated meshes from the AN branch are rendered from different viewpoints as shown in Figure 2.1. Rendered images are colored based on per-face normals. As we observe, the image loss reduces artifacts such as holes, resulting in more accurate generation.

We next demonstrate the importance of the normal consistency loss in Figure 2.5. We colorize the generated point clouds (from the AN branch) with ground

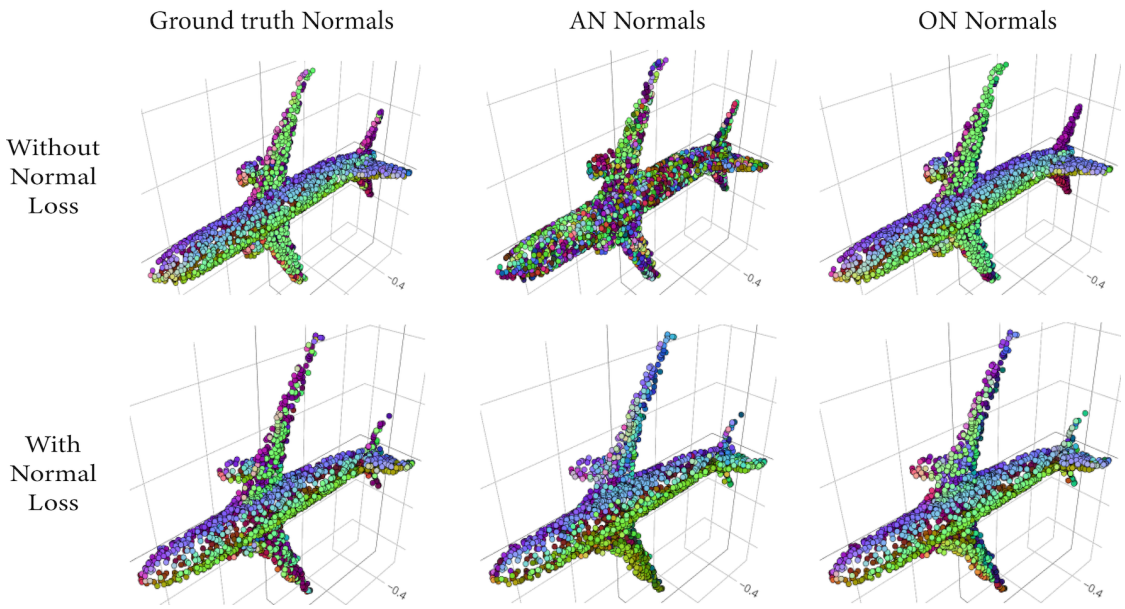


Figure 2.5: Impact of the normal consistency loss. The generated point clouds are colored based on the ground truth normals, AtlasNet's (AN) normals, and OccupancyNet's (ON) normals. The results are then compared between a model trained with the normal consistency loss and a model trained without that loss; AN's normals significantly improve when the loss is incorporated, as it encourages alignment with ON's normals which tend to be close to the ground truth normals.

truth normals as well as normals computed from the AN and ON branches (Equation 2.3 and 2.4). We show the change in these results between two models trained with and without normal consistency loss (Equation 2.5). As we observe, AN's normals are inaccurate for models without the normal loss. This is since the normals consistency loss drives AN's normals to align with ON's normals, which are generally close to the ground truth's.

Finally, Figure 2.6 exhibits the effect of the consistency loss. We evaluate the resulting consistency by measuring the deviation from the constraint in Equation 2.1, i.e., the deviation of the predicted occupancy probabilities from the

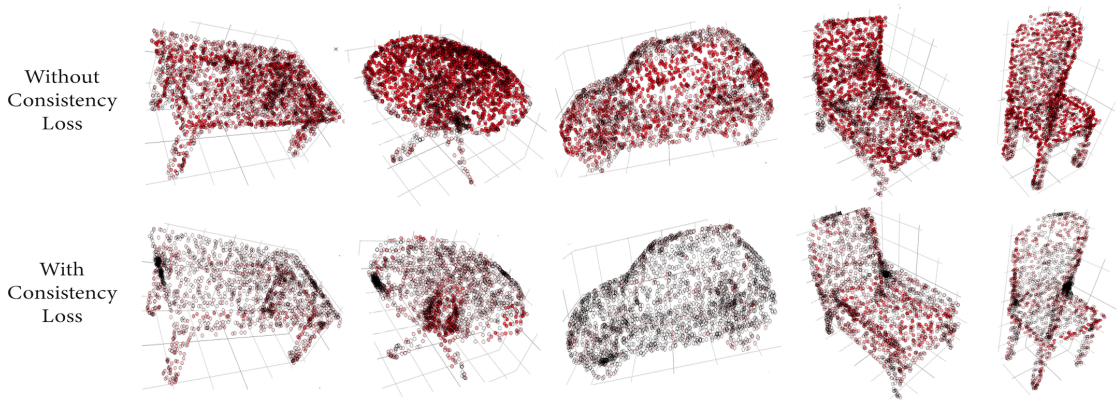


Figure 2.6: Impact of the consistency loss. Each point in the generated point cloud is colored based on deviation of its predicted occupancy probability from the threshold τ, with red indicating deviation.

threshold τ, sampled on the predicted AtlasNet surface. We then color the point cloud such that the larger the deviation the redder the point is. Evidently, for models trained without the consistency loss this deviation is significantly larger, than when the consistency loss is incorporated.

Surface reconstruction time. One drawback of the implicit representations is the necessary additional step of extracting the level set. Current approaches require sampling a large number of points near the surface which can be computationally expensive. Our approach allows to circumvent this issue by using the reconstruction from the explicit branch (which is trained to be consistent with the level set of the implicit representation). We see that the surface reconstructing time is about order of magnitude faster for explicit representation (Table 2.3). Our qualitative and quantitative results suggest that the quality of the explicit representation improves significantly when trained with the consistency losses.

	AN (explicit)	ON (implicit)
Single-view Reconstruction	0.037	0.400
Auto-encoding	0.025	0.428

Table 2.3: Average surface reconstruction time (in seconds) for the explicit (AN) and implicit (ON) representations. Our approach enables to pick the appropriate reconstruction routine at inference time depending on the application needs, where the quality of the reconstructed surfaces increases due to dual training.

2.5 Conclusion and Future Work

We presented a dual approach for generating consistent implicit/explicit surface representations using AtlasNet and OccupancyNet in a hybrid architecture, via novel consistency losses that encourage this consistency. Various tests demonstrate that surfaces generated by our network are of higher quality, namely smoother and closer to the ground truth compared with vanilla AtlasNet and OccupancyNet.

A main shortcoming of our method is that it only penalizes inconsistency of the branches, but does not guarantee perfect consistency; nonetheless, the experiments conducted show that both representations significantly improve by using this hybrid approach during training.

We believe this is an important step in improving neural surface generation, and are motivated to continue improving this hybrid approach, by devising tailor-made encoders and decoders for both representations, to optimize their synergy. In terms of applications, we see many interesting future directions that leverage the strengths of each approach, such as using AtlasNet to texture OccupancyNet's implicit level set, or using OccupancyNet to train AtlasNet's

surface to encapsulate specific input points.

CHAPTER 3

STACKED GENERATIVE ADVERSARIAL NETWORKS

3.1 Introduction

Recent years have witnessed tremendous success of deep neural networks (DNNs), especially the kind of bottom-up neural networks that are trained for discriminative tasks. In particular, Convolutional Neural Networks (CNNs) have achieved impressive accuracy on the challenging ImageNet classification benchmark [142, 238, 249, 106, 227]. Interestingly, it has been shown that CNNs trained on ImageNet for classification can learn representations that are transferable to other tasks [232], and even to other modalities [102]. However, bottom-up discriminative models are focused on learning useful representations from data, being incapable of capturing the data distribution.

Learning top-down generative models that can explain complex data distribution is a long-standing problem in machine learning research. The expressive power of deep neural networks makes them natural candidates for generative models, and several recent works have shown promising results [137, 91, 195, 157, 296, 178, 71]. While state-of-the-art DNNs can rival human performance in discriminative tasks such as object classification, current best deep generative models still fail when there are large variations in the data distribution.

A natural question therefore arises: can we leverage the hierarchical representations in a discriminatively trained model to help the learning of top-down generative models? In this paper, we propose a novel generative model named

Stacked Generative Adversarial Networks (SGAN). Our model consists of a top-down stack of GANs, each trained to generate "plausible" lower-level representations conditioned on higher-level representations. Similar to the image discriminator in the original GAN model which is trained to distinguish "fake" images from "real" ones, we introduce a set of *representation discriminators* that are trained to distinguish "fake" representations from "real" representations. The *adversarial loss* introduced by the representation discriminator forces the intermediate representations of the SGAN to lie on the manifold of the bottom-up DNN's representation space.In addition to the adversarial loss, we also introduce a *conditional loss* that imposes each generator to use the conditional information from the layer above, and a novel *entropy loss* that encourages each generator to generate diverse representations by maximizing a lower bound for the conditional entropy of the generator outputs. By stacking several GANs in a top-down way and using the top-most GAN to receive labels and the bottom-most GAN to generate images, SGAN can be trained to model the data distribution conditioned on class labels. Compared with a vanilla GAN, SGAN has three substantial advantages:

1. **Quality**. SGAN decomposes the difficult image generation task into several easier sub-tasks by leveraging the discriminative representations. As a result, SGAN can generate higher-quality samples than a vanilla GAN. Empirically, we also observe SGAN to be more stable than a vanilla GAN.
2. **Diversity**. With the proposed entropy loss, SGAN can avoid the (conditional) *model collapse* phenomenon that is common in (conditional) GAN training, thus being able to generate more diverse samples.
3. **Interpretability**. Our SGAN is more interpretable than a vanilla GAN in that it uses different levels of noise variables to represent different levels

of variations.

3.2 Related Work

Deep Generative Image Models. There has been a large body of work on generative image modeling with deep learning. Some early efforts include Restricted Boltzmann Machines (RBMs) [110] and Deep Belief Networks (DBNs) [111]. More recently, several successful paradigms of deep generative models have emerged, including the auto-regressive models [147, 86, 255, 195, 196, 95], Variational Auto-encoders (VAEs) [137, 135, 222, 283, 93], and Generative Adversarial Networks (GANs) [91, 64, 212, 220, 229, 149]. Our work builds upon the GAN framework, which uses a generator to transform a noise vector into an image and a discriminator to distinguish between ”real” and ”generated” images. The generator is trained to generate images that are real enough to “fool” the discriminator.

However, due to the vast variations in image content, it is still challenging for GANs to generate diverse images with sufficient details. To this end, several works have attempted to factorize a GAN into a series of GANs, decomposing the difficult task into several more tractable sub-tasks. [64] proposes a LAPGAN model that factorizes the generative process into multi-resolution GANs, with each GAN generating a higher-resolution residual conditioned on a lower-resolution image. Although both LAPGAN and SGAN consist of a sequence of GANs each working at one scale, LAPGAN focuses on generating *multi-resolution images* from coarse to fine while our SGAN aims at modeling *multi-level representations* from high-level to low-level. [270] proposes a S^2-GAN

model, using one GAN to generate surface normals and another GAN to generate images conditioned on surface normals. Surface normals can be viewed as a specific type of image representations, capturing the underlying 3D structure of an indoor scene. However, our framework can leverage the more general and powerful multi-level representations in a pre-trained discriminative DNN.

There are several works that use a pre-trained discriminative model to aid the training of a generative model. [146, 69] add a discriminative regularization term that encourages the image reconstructed by a VAE to be similar to the original image in the representation space defined by a pre-trained bottom-up DNN. [84, 83, 259, 123] use an additional "style loss" which minimizes the $L2$ distance between Gram matrices on some representation space. All the methods above only add loss terms to regularize the *output* of the generator, without regularizing its *internal representations*. Our SGAN adopts a very different approach, using adversarial training to encourage the internal representations of the generator to reside on the representation manifold defined by a pre-trained discriminative model.

Matching Intermediate Representations Between Two DNNs. There have been research directions attempting to "match" the intermediate representations between two DNNs. [224, 102] use the intermediate representations of one pre-trained DNN to guide another DNN in the context of knowledge transfer. Our method can be considered as a special kind of knowledge transfer. However, we aim at transferring the knowledge in a bottom-up DNN to a top-down generative model, instead of another bottom-up DNN. Another direction is the layer-wise reconstruction loss used in some auto-encoder architectures [260, 217, 295, 293]. The layer-wise loss is usually accompanied by lateral con-

nections from the encoder to the decoder at each level of the hierarchy. Our SGAN, however, is a fully independent generative model and does not require any information from the encoder once training completes. Another important difference is that we use adversarial loss instead of $L2$ reconstruction loss to match intermediate representations.

Visualizing Deep Representations. Our work is also related to the recent efforts in visualizing and understanding the internal representations of DNNs. One popular approach uses gradient-based optimization to find an image whose representation is close to the representation we want to visualize [236, 177, 192]. Other approaches, such as [70], train a top-down deconvolutional network to reconstruct the input image from a feature representation by minimizing the Euclidean reconstruction error in image space. However, there is inherent uncertainty in the reconstruction process, since the representations in higher layers of the DNN are trained to be invariant to irrelevant transformations and to ignore low-level details. With Euclidean training objective, the deconvolutional network learns to produce a blurry image when it is uncertain about the details.

To alleviate this problem, [69] further proposes a feature loss that encourages the reconstructed image to have similar high-level representations with the original image, and an adversarial loss that encourages the reconstructed image to lie on the natural image manifold. Their method is able to produce much sharper reconstructions. However, it still does not tackle the problem of uncertainty in reconstruction. Given a high-level feature representation, the deconvolutional network deterministically generates a single image, despite the fact that there exist many images having the same representation. Also, there

is no obvious way to sample images from their model because the distribution of the feature representations is unknown. Concurrent to our work, [190] incorporates the feature prior with a variant of denoising auto-encoder (DAE). Their sampling relies on an iterative optimization procedure, while we are focused on efficient feed-forward sampling.

3.3 Methods

In this section we introduce our model architecture. In Section 3.3.1 we briefly overview the framework of Generative Adversarial Networks. We then describe our proposal for Stacked Generative Adversarial Networks in Section 3.3.2. In Sections 3.3.3 and 3.3.4 we will focus on our two novel loss functions, conditional loss and entropy loss, respectively.

3.3.1 Background: Generative Adversarial Network

As shown in Figure 3.1 (a), the original GAN [91] is trained using a two-player min-max game: a discriminator D is trained to distinguish generated images from real images, and a generator G tries to fool the discriminator D. The discriminator loss $\mathcal{L}_D$ and the generator loss $\mathcal{L}_G$ are formally defined as follows:

$$\mathcal{L}_D = \mathbb{E}_{x \sim P_{data}}[-\log D(x)] + \mathbb{E}_{z \sim P_z}[-\log(1 - D(G(z)))] \tag{3.1}$$

$$\mathcal{L}_G = \mathbb{E}_{z \sim P_z}[-\log(D(G(z)))] \tag{3.2}$$

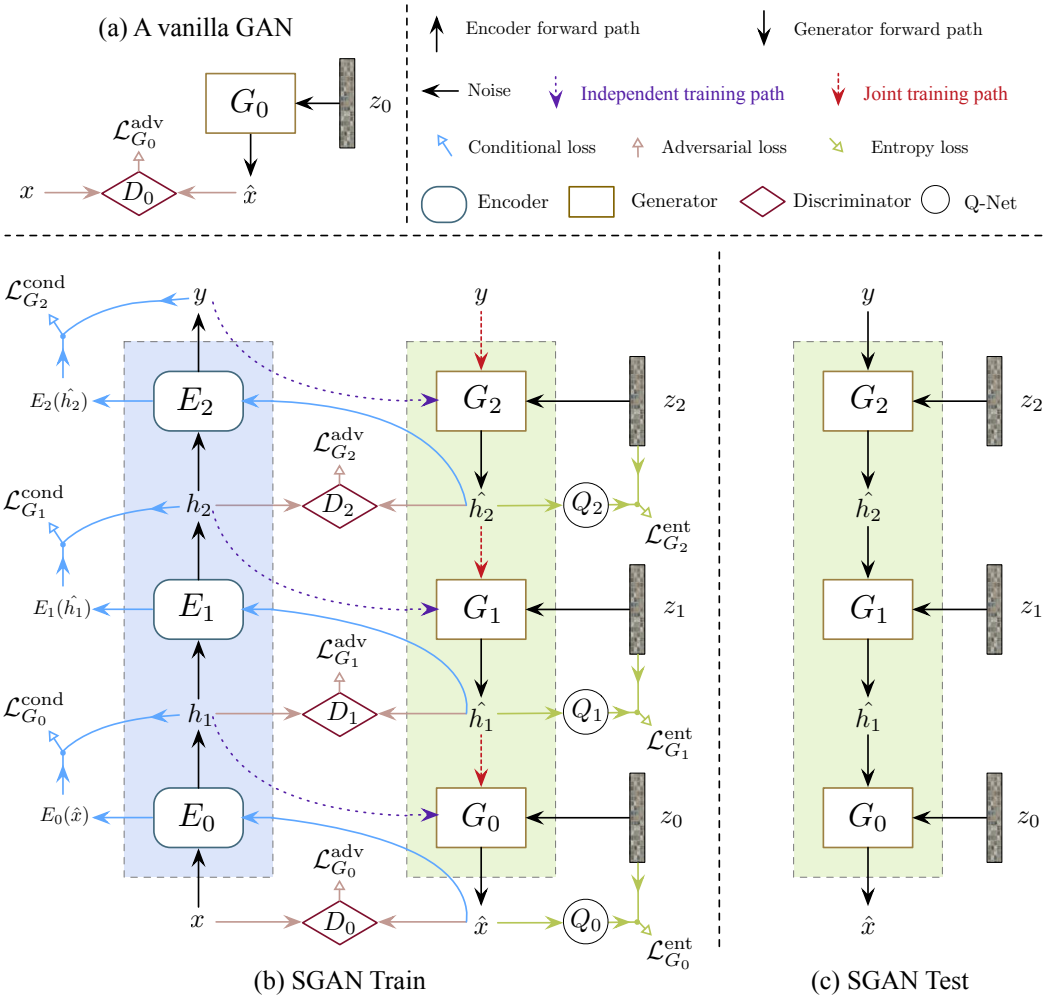


Figure 3.1: An overview of SGAN. (a) The original GAN in [91]. (b) The workflow of training SGAN, where each generator G_i tries to generate plausible features that can fool the corresponding representation discriminator D_i. Each generator receives conditional input from encoders in the independent training stage, and from the upper generators in the joint training stage. (c) New images can be sampled from SGAN (during test time) by feeding random noise to each generator G_i.

In practice, D and G are usually updated alternately. The training process implicitly matches the generated image distribution $P_G(x)$ with the real image distribution $P_{data}(x)$ in the training set. In other words, The adversarial training forces G to generate images that reside on the manifold of natural images.

In the original GAN framework, the single noise vector z has to encapsulate every detail about an image. As a result, z cannot focus on representing high-level invariant features because invariant features are by definition not sufficient to generate an image with enough details.

Intuitively, the total variations of images could be decomposed into multiple levels, with higher-level semantic variations (*e.g.*, attributes, object categories, rough shapes) and lower-level spatial variations (*e.g.*, detailed contours and textures, background clutters). A better approach would be to use different levels of noise variables to represent different levels of variations. Although [229, 296] have tried to feed multi-scale noise into different layers of the generator, there is nothing to teach their models to use noise variables at different levels to represent different levels of variations.

3.3.2 Stacked Generative Adversarial Networks

Pre-trained Encoder. We first consider a bottom-up DNN pre-trained for classification, which is referred to as the encoder E throughout. We define a stack of bottom-up deterministic nonlinear mappings: $h_{i+1} = E_i(h_i)$, where $i \in \{0, 1, ..., N-1\}$, E_i consists of a sequence of neural layers (*e.g.*, convolution, pooling), N is the number of hierarchies (stacks), $h_i (i \neq 0, N)$ are intermediate representations, $h_N = y$ is the classification result, and $h_0 = x$ is the input image.

Note that in our formulation, each E_i can contain multiple layers and the way of grouping layers together into E_i is determined by us. The number of stacks N is therefore less than the number of layers in E and is also determined by us.

Stacked Generators. Provided with a pre-trained encoder E, our goal is to train a top-down generator G that inverts E. Specifically, G consists of a top-down *stack* of generators G_i, each trained to invert a bottom-up mapping E_i. Each G_i takes in a noise prior z_i as input, and produces the corresponding representation $\hat{h}_i$.

Since E_i is usually a many-to-one mapping, there can be many h_is such that $h_{i+1} = E_i(h_i)$. To aid the uncertainty of generator G_i, we provide each G_i with the conditional information of the higher-level representation h_{i+1}, defined by the pre-trained encoder E. In other words, $\hat{h}_i = G_i(h_{i+1}, z_i)$.

G_i therefore implicitly defines a distribution of $p_{G_i}(\hat{h}_i|h_{i+1})$. To sample the generated representation at stack i, we first sample z_i from a simple distribution such as Gaussian, and then compute $\hat{h}_i$ accordingly.

Training Generator. Each G_i is trained using the GAN approach. The training workflow is shown in Figure 3.1 (b). Each generator G_i is trained with a linear combination of three loss functions: adversarial loss, conditional loss, and entropy loss.

$$\mathcal{L}_{G_i} = \lambda_1 \mathcal{L}_{G_i}^{adv} + \lambda_2 \mathcal{L}_{G_i}^{cond} + \lambda_3 \mathcal{L}_{G_i}^{ent}, \tag{3.3}$$

where $\mathcal{L}_{G_i}^{adv}$, $\mathcal{L}_{G_i}^{cond}$, $\mathcal{L}_{G_i}^{ent}$ denote adversarial loss, conditional loss, and entropy loss respectively. λ_1, λ_2, λ_3 are the weights associated with different loss terms. In

practice, we find it sufficient to set the weights such that the magnitude of different loss terms are of similar scales. No extensive hyper-parameter tuning is needed. In this section we will first introduce the adversarial loss $\mathcal{L}_{G_i}^{adv}$. We will then introduce $\mathcal{L}_{G_i}^{cond}$ and $\mathcal{L}_{G_i}^{ent}$ in Sections 3.3.3 and 3.3.4 respectively.

For each generator G_i, we introduce a *representation discriminator* D_i that distinguishes generated representations $\hat{h}_i$, from "real" representations h_i. Specifically, the discriminator D_i is trained with the loss function

$$\mathcal{L}_{D_i} = \mathbb{E}_{h_i \sim P_{data,E}}[-\log D_i(h_i)] + \mathbb{E}_{z_i \sim P_{z_i},\ h_{i+1} \sim P_{data,E}}[-\log(1 - D_i(G_i(h_{i+1}, z_i)))] \quad (3.4)$$

And G_i is trained to "fool" the representation discriminator D_i, with the adversarial loss defined by

$$\mathcal{L}_{G_i}^{adv} = \mathbb{E}_{h_{i+1} \sim P_{data,E},\ z_i \sim P_{z_i}}[-\log(D_i(G_i(h_{i+1}, z_i)))] \quad (3.5)$$

We first train each GAN independently and then train them jointly in an end-to-end manner, as shown in Figure 3.1. This is similar to the training process of S^2-GAN [270]. Each generator receives conditional input from encoders in the independent training stage, and from the upper generators in the joint training stage. During joint training, the adversarial loss provided by representational discriminators can also be viewed as a type of deep supervision [153], providing intermediate supervision signals to the generator. In our current formulation, E is a discriminative model, and G is a generative model conditioned on labels. However, it is also possible to train SGAN without using label information: E

can be trained with an unsupervised objective and G can be cast into an unconditional generative model by removing the label input from the top generator. We leave this for future exploration.

The idea of training a discriminator in some representation space has been explored in the context of domain adaptation [82, 48, 112], and regularization of auto-encoders [178]. However, our representation discriminators are designed to transfer the knowledge in the pre-trained bottom-up discriminative model to our top-down generative model.

Sampling Images. To sample images, all G_is are stacked together in a top-down manner, as shown in Figure 3.1 (c). Our SGAN is capable of modeling the data distribution conditioned on the class label: $p_G(\hat{x}|y) = p_G(\hat{h}_0|\hat{h}_N) \propto p_G(\hat{h}_0, \hat{h}_1, ..., \hat{h}_{N-1}|\hat{h}_N) = \prod_{0 \leq i \leq N-1} p_{G_i}(\hat{h}_i|\hat{h}_{i+1})$, where each $p_{G_i}(\hat{h}_i|\hat{h}_{i+1})$ is modeled by a generator G_i. From an information-theoretic perspective, SGAN factorizes the total entropy of the image distribution $H(x)$ into multiple (smaller) conditional entropy terms: $H(x) = H(h_0, h_1, ..., h_N) = \sum_{i=0}^{N-1} H(h_i|h_{i+1}) + H(y)$, thereby decomposing one difficult task into multiple easier tasks.

3.3.3 Conditional Loss

At each stack, a generator G_i is trained to capture the distribution of lower-level representations, conditioned on higher-level representations. However, in the above formulation, the generator might choose to ignore the high-level conditional information h_{i+1}, and generate plausible lower-level representations $\hat{h}_i$ from scratch. Some previous works [184, 85, 64, 220] tackle this problem by feeding the conditional information to both the generator and discriminator.

This approach, however, might introduce unnecessary complexity to the discriminator and increase model instability [201, 231].

Here we adopt a different approach: we regularize the generator by adding a loss term $\mathcal{L}_{G_i}^{cond}$ named *conditional loss*. We feed the generated lower-level representations $\hat{h}_i = G_i(h_{i+1}, z_i)$ back to the encoder E, and compute the recovered higher-level representations. We then enforce the recovered representations to be similar to the conditional representations. Formally:

$$\mathcal{L}_{G_i}^{cond} = \mathbb{E}_{h_{i+1} \sim P_{data,E},\ z_i \sim P_{z_i}}[f(E_i(G_i(h_{i+1}, z_i)), h_{i+1})] \tag{3.6}$$

where f is a distance measure. We define f to be the Euclidean distance for intermediate representations and the cross-entropy for labels. Our conditional loss $\mathcal{L}_{G_i}^{cond}$ is similar to the "feature loss" used by [69] and the "FCN loss" used by [270]. $\mathcal{L}_{G_i}^{cond}$ ensures that the generated lower-level representations are consistent with the higher-level conditional information.

3.3.4 Entropy Loss

Simply adding the conditional loss $\mathcal{L}_{G_i}^{cond}$ leads to another issue in our experiments: the generator G_i learns to ignore the noise z_i, and compute $\hat{h}_i$ deterministically from h_{i+1}. We refer to this phenomenon as *conditional model collapse*, *i.e.*, the conditional generative model ignores the noise input and deterministically generates the output from the conditional information. This problem has been encountered in a wide range of conditional GAN applications, *e.g.*, synthesizing future frames conditioned on previous frames [181], generating images conditioned on label maps [118], and most related to our work, synthesizing images conditioned on feature representations [69]. All the above works tried to gener-

ate *diverse* images/videos by feeding noise to the generator, but failed because the conditional generator simply ignores the noise. To our knowledge, there is still no principled way to deal with this issue.

It might be tempting to think that *minibatch discrimination* [229], which encourages sample diversity in each minibatch, could solve this problem. However, even if the generator generates $\hat{h}_i$ deterministically from h_{i+1}, the generated samples in each minibatch are still diverse since generators are conditioned on different h_{i+1}. Thus, there is no obvious way minibatch discrimination could penalize a collapsed conditional generator.

Variational Conditional Entropy Maximization. To alleviate this issue, we would like to encourage the generated representation $\hat{h}_i$ to be sufficiently diverse when conditioned on h_{i+1}, *i.e.*, the conditional entropy $H(\hat{h}_i|h_{i+1})$ should be as high as possible. Since directly maximizing $H(\hat{h}_i|h_{i+1})$ is intractable, we propose to maximize instead a *variational lower bound* on the conditional entropy. Specifically, we use an auxiliary distribution $Q_i(z_i|\hat{h}_i)$ to approximate the true posterior $P_i(z_i|\hat{h}_i)$, and augment the training objective with a loss term named *entropy loss*:

$$\mathcal{L}_{G_i}^{ent} = \mathbb{E}_{z_i \sim P_{z_i}}[\mathbb{E}_{\hat{h}_i \sim G_i(\hat{h}_i|z_i)}[-\log Q_i(z_i|\hat{h}_i)]] \tag{3.7}$$

Below we give a proof that minimizing $\mathcal{L}_{G_i}^{ent}$ is equivalent to maximizing a variational lower bound for $H(\hat{h}_i|h_{i+1})$.

$$
\begin{aligned}
H(\hat{h}_i|h_{i+1}) &= H(\hat{h}_i, z_i|h_{i+1}) - H(z_i|\hat{h}_i, h_{i+1}) \\
&\geq H(\hat{h}_i, z_i|h_{i+1}) - H(z_i|\hat{h}_i) \\
&= H(z_i|h_{i+1}) + \underbrace{H(\hat{h}_i|z_i, h_{i+1})}_{0} - H(z_i|\hat{h}_i) \\
&= H(z_i|h_{i+1}) - H(z_i|\hat{h}_i) \\
&= H(z_i) - H(z_i|\hat{h}_i) \\
&= \mathbb{E}_{\hat{h}_i \sim G_i}[\mathbb{E}_{z_i' \sim P_i(z_i'|\hat{h}_i)}[\log P_i(z_i'|\hat{h}_i)]] + H(z_i) \\
&= \mathbb{E}_{\hat{h}_i \sim G_i}[\mathbb{E}_{z_i' \sim P_i(z_i'|\hat{h}_i)}[\log Q_i(z_i'|\hat{h}_i)] \\
&+ \underbrace{KLD(P_i||Q_i)}_{\geq 0}] + H(z_i) \\
&\geq \mathbb{E}_{\hat{h}_i \sim G_i}[\mathbb{E}_{z_i' \sim P_i(z_i'|\hat{h}_i)}[\log Q_i(z_i'|\hat{h}_i)]] + H(z_i) \\
&= \mathbb{E}_{z_i' \sim P_{z_i'}}[\mathbb{E}_{\hat{h}_i \sim G_i(\hat{h}_i|z_i')}[\log Q_i(z_i'|\hat{h}_i)]] + H(z_i) \\
&\triangleq -\mathcal{L}_{G_i}^{ent} + H(z_i)
\end{aligned}
\tag{3.8}
$$

In practice, we parameterize Q_i with a deep network that predicts the posterior distribution of z_i given $\hat{h}_i$. Q_i shares most of the parameters with D_i. We treat the posterior as a diagonal Gaussian with fixed standard deviations, and use the network Q_i to only predict the posterior mean, making $\mathcal{L}_{G_i}^{ent}$ equivalent to the Euclidean reconstruction error. In each iteration we update both G_i and Q_i to minimize $\mathcal{L}_{G_i}^{ent}$.

Our method is similar to the variational mutual information maximization technique proposed by [47]. A key difference is that [47] uses the Q-network to predict only a small set of deliberately constructed "latent code", while our Q_i tries to predict *all* the noise variables z_i in each stack. The loss used in [47] therefore maximizes the *mutual information* between the output and the latent

code, while ours maximizes the *conditional entropy* of the output $\hat{h}_i$, provided with h_{i+1}. [67, 72] also train a separate network to map images back to noise space in the context of unsupervised feature learning. Independent of our work, [61] proposes to regularize EBGAN [296] with entropy maximization, in order to prevent the discriminator from degenerating to uniform prediction. However, our entropy loss is motivated from tackling the conditional model collapse phenomenon described above.

3.4 Experiments

In the following, we perform experiments on a variety of datasets including MNIST [151], SVHN [189], and CIFAR-10 [141].

Code and pre-trained models are available at: `https://github.com/xunhuang1995/SGAN`. Readers may refer to our code repository for more details about experimental setup, hyper-parameters, *etc.*

Encoder. For all datasets we use a small CNN with two convolutional layers as our encoder: `conv1-pool1-conv2-pool2-fc3-fc4`, where `fc3` is a fully connected layer and `fc4` outputs classification scores before softmax. We apply horizontal flipping on CIFAR-10. No data augmentation is used on other datasets.

Generator. We use generators with two stacks in our experiments. Note that our framework is generally applicable to the setting with multiple stacks, and we hypothesize that using more stacks would be helpful for large-scale and high-resolution datasets. For all datasets, our top GAN G_1 generates `fc3` fea-

tures from some random noise z_1, conditioned on label y. And the bottom GAN G_0 generates images from some noise z_0, conditioned on `fc3` features generated from GAN G_1. We set the loss coefficient parameters $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 10$.[1]

3.4.1 Datasets

We thoroughly evaluate SGAN on three widely adopted datasets: MNIST [151], SVHN [189], and CIFAR-10 [141]. The details of each dataset is described in the following.

MNIST The MNIST dataset contains $60,000$ labeled images of hand-written digits. Each image is sized by 28×28.

SVHN The Street View House Numbers (SVHN) dataset is composed of real-world color images of house numbers collected by Google Street View [189]. Each image is of size 32×32 and the task is to classify the digit at the center of the image. The dataset contains $73,257$ images in the training set and $26,032$ images in the test set.

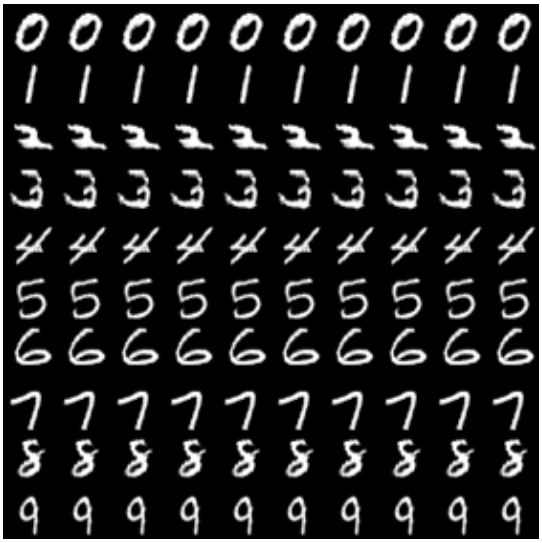
CIFAR-10 The CIFAR-10 dataset consists of colored natural scene images sized at 32×32 pixels. There are 50,000 training images and 10,000 test images in 10 classes.

[1]The choice of the parameters are made so that the magnitude of each loss term is of the same scale.

(a) SGAN samples (conditioned on labels)

(b) Real images (nearest neighbor)

(c) SGAN samples (conditioned on generated `fc3` features)



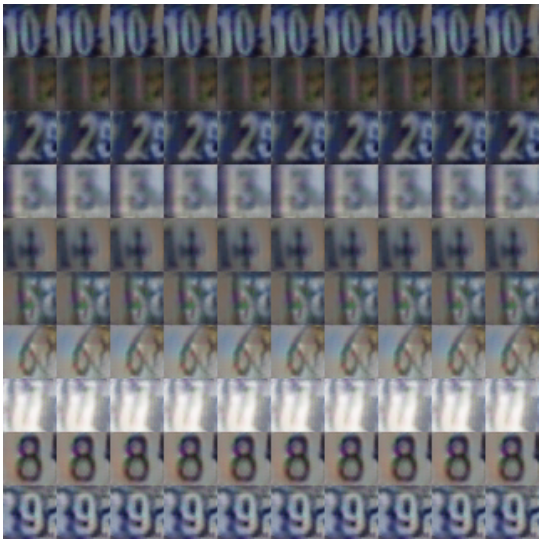
(d) SGAN samples (conditioned on generated `fc3` features, trained *without* entropy loss)

Figure 3.2: MNIST results. (a) Samples generated by SGAN when conditioned on class labels. Each row corresponds to a digit class. (b) Corresponding nearest neighbor images in the MNIST training set. (c) Each row corresponds to samples generated by the bottom GAN when conditioned on a fixed `fc3` feature activation, generated from the top GAN. (d) Same setting as (c), but the bottom GAN is trained without entropy loss.

(a) SGAN samples (conditioned on labels)

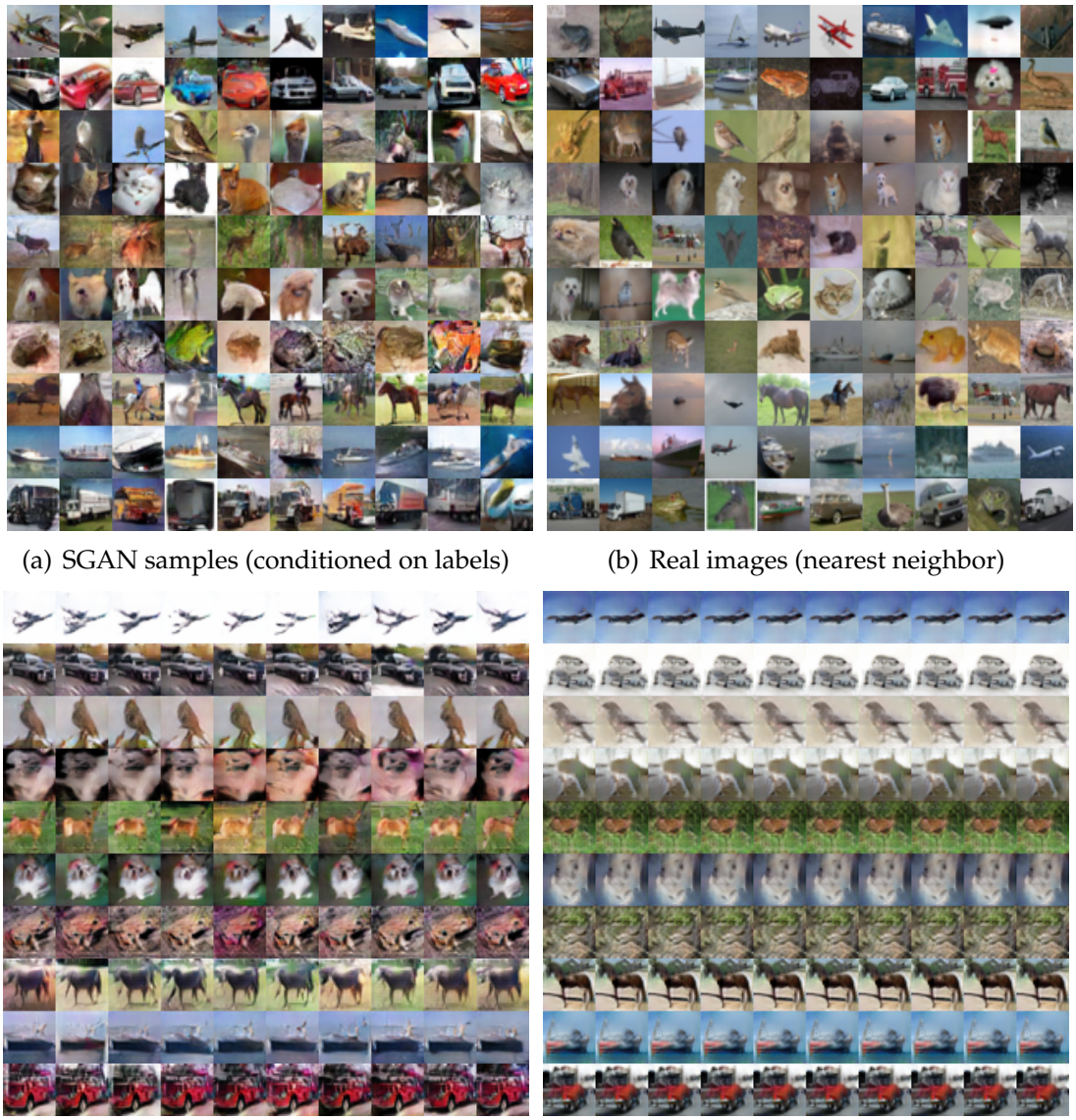
(b) Real images (nearest neighbor)

(c) SGAN samples (conditioned on generated `fc3` features)



(d) SGAN samples (conditioned on generated `fc3` features, trained *without* entropy loss)

Figure 3.3: SVHN results. (a) Samples generated by SGAN when conditioned on class labels. Each row corresponds to a digit class. (b) Corresponding nearest neighbor images in the SVHN training set. (c) Samples generated by the bottom GAN when conditioned on a fixed `fc3` feature activation, generated by the top GAN. (d) Same setting as (c), but the bottom GAN is trained without entropy loss.



(a) SGAN samples (conditioned on labels)

(b) Real images (nearest neighbor)

(c) SGAN samples (conditioned on generated `fc3` features)

(d) SGAN samples (conditioned on generated `fc3` features, trained *without* entropy loss)

Figure 3.4: CIFAR-10 results. (a) Samples generated by SGAN when conditioned on class labels. Each row corresponds to a object category. (b) Corresponding nearest neighbor images in the CIFAR-10 training set. (c) Samples generated by the bottom GAN when conditioned on a fixed `fc3` feature activation, generated by the top GAN. (d) Same setting as (c), but the bottom GAN is trained without entropy loss.

3.4.2 Samples

In Figure 3.2 (a), we show MNIST samples generated by SGAN. Each row corresponds to samples conditioned on a given digit class label. SGAN is able to generate diverse images with inherently different characteristics. The samples are visually indistinguishable from real MNIST images shown in Figure 3.2 (b), but still have differences compared with corresponding nearest neighbor training images.

We further examine the effect of entropy loss (Section 3.3.4). We show in Figure 3.2 (c) samples generated by bottom GAN when conditioned on a fixed `fc3` feature generated by the top GAN. The samples (per row) have sufficient low-level variations, which reassures that bottom GAN learns to generate images without ignoring the noise z_0. This indicates that each GAN has the potential to capture the full conditional distribution $p(\hat{h}_i|h_{i+1})$. In contrast, in Figure 3.2 (d), we show samples generated without using entropy loss for bottom generator, where we observe the conditional model collapse phenomenon: the bottom GAN ignores the noise and instead deterministically generates images from `fc3` features.

An advantage of SGAN compared with a vanilla GAN is its interpretability: it decomposes the total variations of an image into different levels. For example, in MNIST it decomposes the variations into y that represents the high-level digit label, z_1 that captures the mid-level coarse pose of the digit and z_0 that represents the low-level spatial details.

The samples generated on SVHN and CIFAR-10 datasets can also be seen in Figure 3.3 and Figure 3.4, respectively. Provided with the same `fc3` feature, we

see in each row of panel (c) that SGAN is able to generate samples with similar coarse outline but different lighting conditions and background clutters. Also, the nearest neighbor images in the training set indicate that SGAN is not simply memorizing training data, but can truly generate unseen images.

3.4.3 Comparison with the state of the art

In Table 3.1 we compare SGAN with other state-of-the-art generative models on CIFAR-10 dataset. The visual quality of generated images is measured by the widely used metric, Inception score [229]. Following [229], we sample $50,000$ images from our model and use the code provided by [229] to compute the score. Our SGAN obtains a score of 8.59 ± 0.12, outperforming AC-GAN [194] (8.25 ± 0.07) and Improved GAN [229] (8.09 ± 0.07). Also, note that the 5 techniques introduced in [229] are not used in our implementations. Incorporating these techniques might further boost the performance of our model.

3.4.4 More ablation studies

In Section 3.4.2 we have examined the effect of entropy loss on tackling the conditional model collapse problem. In order to further understand the effect of different model components, we conduct extensive ablation studies by evaluating several baseline methods on CIFAR-10 dataset. All models below use the same training hyper-parameters as the full SGAN model.

(a) SGAN: The full model, as described in Section 3.3.

Method	Score
Infusion training [32]	4.62 ± 0.06
ALI [72] (as reported in [272])	5.34 ± 0.05
GMAN [73] (best variant)	6.00 ± 0.19
EGAN-Ent-VI [61]	7.07 ± 0.10
LR-GAN [285]	7.17 ± 0.07
Denoising feature matching [272]	7.72 ± 0.13
DCGAN† (with labels, as reported in [268])	6.58
SteinGAN† [268]	6.35
Improved GAN† [229] (best variant)	8.09 ± 0.07
AC-GAN† [194]	8.25 ± 0.07
DCGAN ($\mathcal{L}^{adv}$)	6.16 ± 0.07
DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{ent}$)	5.40 ± 0.16
DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{cond}$)†	5.40 ± 0.08
DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{cond} + \mathcal{L}^{ent}$)†	7.16 ± 0.10
SGAN-no-joint†	**8.37** ± 0.08
SGAN†	**8.59** ± 0.12
Real data	11.24 ± 0.12

† Trained with labels.

Table 3.1: Inception Score on CIFAR-10. SGAN and SGAN-no-joint outperform state-of-the-art approaches.

(b) SGAN-no-joint: Same architecture as (a), but each GAN is trained *independently*, and there is no end-to-end joint training stage.

(c) DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{cond} + \mathcal{L}^{ent}$): This is a *single* GAN model with the same architecture as the bottom GAN in SGAN, except that the generator is conditioned on labels instead of `fc3` features. Note that other techniques proposed in this paper, including conditional loss $\mathcal{L}^{cond}$ and entropy loss $\mathcal{L}^{ent}$, are still employed. We also tried to use the full generator G in SGAN as the baseline, instead of only the bottom generator G_0. However, we failed to make it converge, possibly because G is too deep to be trained without intermediate supervision from representation discriminators.

(d) DCGAN ($\mathcal{L}^{adv}$ + $\mathcal{L}^{cond}$): Same architecture as (c), but trained without entropy loss $\mathcal{L}^{ent}$.

(e) DCGAN ($\mathcal{L}^{adv}$ + $\mathcal{L}^{ent}$): Same architecture as (c), but trained without conditional loss $\mathcal{L}^{cond}$. The model therefore does not use label information.

(f) DCGAN ($\mathcal{L}^{adv}$): Same architecture as (c), but trained with neither conditional loss $\mathcal{L}^{cond}$ nor entropy loss $\mathcal{L}^{ent}$. The model also does not use label information. It can be viewed as a plain unconditional DCGAN model [212] and serves as the ultimate baseline.

We show the samples and Inception scores [229] of the baseline methods in Figure 3.5 and Table 3.1, respectively. Below we summarize some of our findings:

1) SGAN obtains slightly higher Inception score than SGAN-no-joint. However SGAN-no-joint also generates very high quality samples and outperforms all previous state-of-the-art models in Inception scores. We also observe that joint training is more apt to (partial) model collapse and additional caution needs to be taken, such as adding Gaussian noise to the discriminators.

2) SGAN, either with or without joint training, achieves significantly higher Inception score and better sample quality than the baseline DCGANs. This demonstrates the effectiveness of the proposed stacked approach.

3) As shown in Figure 3.5 (d), DCGAN ($\mathcal{L}^{adv}$ + $\mathcal{L}^{cond}$) collapses to generate a single image per category, while adding the entropy loss $\mathcal{L}^{ent}$ enables it to generate diverse images (Figure 3.5 (c)). This further demonstrates that entropy loss is effective at preventing conditional model collapse.

(a) SGAN (b) SGAN-no-joint

(c) DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{cond} + \mathcal{L}^{ent}$) (d) DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{cond}$)

(e) DCGAN ($\mathcal{L}^{adv} + \mathcal{L}^{ent}$) (f) DCGAN ($\mathcal{L}^{adv}$)

Figure 3.5: Ablation studies on CIFAR-10. Samples from (a) full SGAN model (b) SGAN model without joint training step. (c) DCGAN model trained with $\mathcal{L}^{adv} + \mathcal{L}^{cond} + \mathcal{L}^{ent}$ (d) DCGAN model trained with $\mathcal{L}^{adv} + \mathcal{L}^{cond}$ (e) DCGAN model trained with $\mathcal{L}^{adv} + \mathcal{L}^{ent}$ (f) DCGAN model trained with $\mathcal{L}^{adv}$.

4) The plain unconditional DCGAN ($\mathcal{L}^{adv}$) does not collapse and obtains competitive Inception score compared with some previous models. In this case, adding entropy loss $\mathcal{L}^{ent}$ does not seem to provide benefits, at least in terms of the Inception score (Table 3.1).

5) The single DCGAN ($\mathcal{L}^{adv}$ + $\mathcal{L}^{cond}$ + $\mathcal{L}^{ent}$) model obtains higher Inception score than the conditional DCGAN reported in [268]. This suggests that $\mathcal{L}^{cond}$ + $\mathcal{L}^{ent}$ might offer some advantages compared to a plain conditional DCGAN, even without stacking.

6) In general, Inception score [229] correlates well with visual quality of images. However, it seems to be insensitive to diversity issues such as model collapse. For example, it gives the same score to Figure 3.5 (d) and (e) while (d) has clearly collapsed. This is consistent with results in [194, 268].

3.5 Discussion and Future Work

This paper introduces a top-down generative framework named SGAN, which effectively leverages the representational information from a pre-trained discriminative network. In contrast to GANs, our approach decomposes the hard problem of estimating image distribution into multiple relatively easier tasks – each generating plausible representations conditioned on higher-level representations. The key idea is to use representation discriminators at different training hierarchies to provide intermediate supervision. We also propose a novel entropy loss that is to our knowledge the first attempt to alleviate the conditional model collapse problem in GANs. Our entropy loss could be employed in other applications of conditional GANs, *e.g.*, synthesizing *different* future frames given

the same past frames [181], or generating a *diverse* set of images conditioned on the same label map [118]. We believe this is an interesting research direction in the future.

CHAPTER 4

NEURAL PUPPET: GENERATIVE LAYERED CARTOON CHARACTERS

4.1 Introduction

Traditional character animation is a tedious process that requires artists meticulously drawing every frame of a motion sequence. After observing a few such sequences, a human can easily imagine what a character might look in other poses, however, making these inferences is difficult for learning algorithms. The main challenge is that the input images commonly exhibit substantial variations in appearance due to articulations, artistic effects, and viewpoint changes, significantly complicating the extraction of the underlying character structure. In the space of natural images, one can rely on extensive annotations [9] or vast amount of data [235] to extract the common structure. Unfortunately, this approach is not feasible for cartoon characters, since their topology, geometry, and drawing style is far less consistent than that of natural images of human bodies or faces.

To tackle this challenge, we propose a method that learns to generate novel character appearances from a small number of examples by relying on additional user input: a deformable puppet template. We assume that all character poses can be generated by warping the deformable template, and thus develop a deformation network that encodes an image and decodes deformation parameters of the template. These parameters are further used in a differentiable rendering layer that is expected to render an image that matches the input frame. Reconstruction loss can be back-propagated through all stages, enabling us to learn how to register the template with all of the training frames. While

Figure 4.1: The user provides a template mesh and a set of unlabeled images (first row). The model learns to generate inbetween frames (row 2 and 3) and constrained deformations (row 4).

the resulting renderings already yield plausible poses, they fall short of artist-generated images since they only warp a single reference, and do not capture slight appearance variations due to shading and artistic effects. To further improve visual quality of our results, we use an image translation network that synthesizes the final appearance.

While our method is not constrained to a particular choice for the deformable puppet model, we chose a layered 2.5D deformable model that is commonly used in academic [57] and industrial [3] applications. This model matches many traditional hand-drawn animation styles, and makes it significantly easier for the user to produce the template relative to 3D modeling that requires extensive expertise. To generate the puppet, the user has to select a single frame and segment the foreground character into constituent body parts, which can be further converted into meshes using standard triangulation tools [233].

We evaluate our method on animations of six characters with 70%-30% train-test split. First, we evaluate how well our model can reconstruct the input frame and demonstrate that it produces more accurate results than state-of-the-art optical flow and auto-encoder techniques. Second, we evaluate the quality of correspondences estimated via the registered templates, and demonstrate improvement over image correspondence methods. Finally, we show that our model can be used for data-driven animation, where synthesized animation frames are guided by character appearances observed at training time. We build prototype applications for synthesizing in-between frames and animating by user-guided deformation where our model constrains new images to lie on a learned manifold of plausible character deformations. We show that the data-driven approach yields more realistic poses that better match to original artist drawings than traditional energy-based optimization techniques used in computer graphics.

4.2 Related Work

Deep Generative Models. Several successful paradigms of deep generative models have emerged recently, including the auto-regressive models [94, 195, 261], Variational Auto-encoders (VAEs) [136, 135, 222], and Generative Adversarial Networks (GANs) [91, 211, 229, 117, 12, 132]. Deep generative models have been applied to image-to-image translation [120, 298, 297], image superresolution [152], learning from synthetic data [33, 234], generating adversarial images [205, 204], and synthesizing 3D volumes [254, 242]. These techniques usually make no assumptions about the structure of the training data and synthesize pixels (or voxels) directly. This makes them very versatile and appealing when a large number of examples are available. Since these data might not be available in some domains, such as 3D modeling or character animation, several techniques leverage additional structural priors to train deep models with less training data.

Learning to Generate with Deformable Templates. Deformable templates have been used for decades to address analysis and synthesis problems [10, 30, 7, 8, 171, 299]. Synthesis algorithms typically assume that multiple deformations of the same template (e.g., a mesh of the same character in various poses) is provided during training. Generative models, such as variational auto-encoders directly operate on vertex coordinates to encode and generate plausible deformations from these examples [253, 139, 160]. Alternative representations, such as multi-resolution meshes [216], single-chart UV [19], or multi-chart UV [103] is used for higher resolution meshes. This approaches are limited to cases when all of the template parameters are known for all training

examples, and thus cannot be trained or make inferences over raw unlabeled data.

Some recent work suggests that neural networks can jointly learn the template parameterization and optimize for the alignment between the template and a 3D shape [98] or 2D images [124, 107, 262]. While these models can make inferences over unlabeled data, they are trained on a large number of examples with rich labels, such as dense or sparse correspondences defined for all pairs of training examples.

To address this limitation, a recent work on deforming auto-encoders provides an approach for unsupervised group-wise image alignment of related images (e.g. human faces) [235]. They disentangle shape and appearance in latent space, by predicting a warp of a learned template image as well as its texture transformations to match every target image. Since their warp is defined over the regular 2D lattice, their method is not suitable for strong articulations. Thus, to handle complex articulated characters and strong motions, we leverage an user-provided template and rely on regularization terms that leverage the rigging as well as mesh structure. Instead of synthesizing appearance in a common texture space, we do the final image translation pass that enables us to recover from warping artifacts and capture effects beyond texture variations, such as out-of-plane motions.

Mesh-based models for Character Animation. Many techniques have been developed to simplify the production of traditional hand-drawn animation using computers [42, 66]. Mesh deformation techniques enable novice users to create animations by manipulating a small set of control points of a single puppet [245, 121]. To avoid overly-synthetic appearance, one can further

stylize these deformations by leveraging multiple co-registered examples to guide the deformation [266], and final appearance synthesis [75, 76]. These methods, however, require artist to provide the input in a particular format, and if this input is not available rely on image-based correspondence techniques [35, 248, 50, 79, 247] to register the input. Our deformable puppet model relies on a layered mesh representation [76] and mesh regularization terms [245] used in these optimization-based workflows. Our method jointly learns the puppet deformation model as it registers the input data to the template, and thus yields more accurate correspondences than state-of-the-art flow-based approach [247] and state-of-the-art feature-based method trained on natural images [50].

4.3 Approach

Our goal is to learn a deformable model for a cartoon character given an unlabeled collection of images. First, the user creates a layered deformable template puppet by segmenting one reference frame. We then train a two-stage neural network that first fits this deformable puppet to every frame of the input sequence by learning how to warp a puppet to reconstruct the appearance of the input, and second, it refines the rendering of deformed puppet to account for texture variations and motions that cannot be expressed with a 2D warp.

4.3.1 A Layered Deformable Puppet

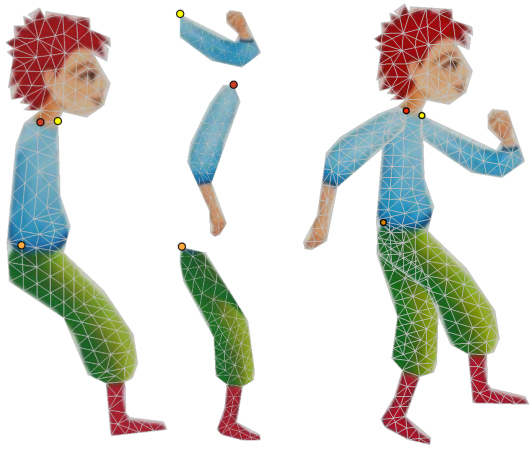
The puppet geometry is represented with a few triangular meshes ordered into layers and connected at hinge joints. For simplicity, we denote them as one mesh with vertices V and faces F, where every layer is a separate connected component of the mesh. Joints are represented as $\{(p_i, q_i)\}$ pairs, connecting vertices between some of these components. The puppet appearance is captured as texture image I^{uv}, which aligns to some rest pose $\hat{V}$. New character poses can be generated by modifying vertex coordinates, and the final appearance can be synthesized by warping the original texture according to mesh deformation.

Unlike 3D modeling, even inexperienced users can create the layered 2D puppet. First, one selects a reference frame and provides the outline for different body parts and prescribes the part ordering. We then use standard triangulation algorithm [233] to generate a mesh for each part, and create a hinge joint at the centroid of overlapping area of two parts. We can further run midpoint mesh subdivision to get a finer mesh that can model more subtle deformations. Figure 4.2 illustrates a deformable puppet.

4.3.2 Deformation Network

After obtaining the template, we aim to learn how to deform it to match a target image of the character in a new pose. Figure 4.3 illustrates our architecture. The inputs to the Deformation Network are the initial mesh and a target image of the character in a new pose. An encoder-decoder network takes the target image, encodes it to a 512-dimensional bottleneck via three convolutional filters[1] and

[1]Kernel size=5, padding=2, stride=2



(a) (b)

Figure 4.2: Deformable Puppet. a) For each body part a separate mesh is created, and joints (shown with circles) are specified. b) The meshes are combined into a single mesh. The UV-image of the final mesh consists of translated versions of separate texture maps.

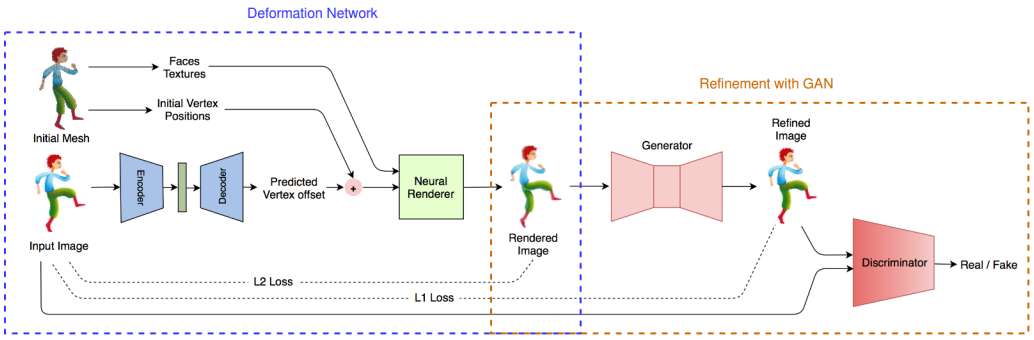


Figure 4.3: Training Architecture. An encoder-decoder network learns the mesh deformation and a conditional Generative Adversarial Network refines the rendered image to capture texture variations.

three fully connected layers, and then decodes it to per-vertex position offsets and a global bias via three fully connected layers. Deformation network learns to recognize the pose in the input image and then infer appropriate template deformation to reproduce the pose. We assume the connectivity of vertices and

the textures remain the same compared to the template. Hence, we pass the faces and textures of the initial mesh in tandem with the predicted vertex positions to a differentiable renderer R. The rendered image is then compared to the input image using L_2 reconstruction loss:

$$L_{rec} = \left\| x - R(V_{pred}, F, I^{uv}) \right\|^2 \tag{4.1}$$

in which x represents the input image. We use the Neural Mesh Renderer [131] as our differentiable renderer, since it can be easily integrated into neural network architectures. Note that the number of vertices can vary for different characters as we train a separate model for each character.

Regularization. The model trained with only the reconstruction loss does not preserve the structure of the initial mesh, and the network may generate large deformations to favor local consistency. In order to prevent this, we use the ARAP regularization energy [245] which penalizes deviations of per-cell transformations from rigidity:

$$L_{reg} = \sum_{i=1}^{|V|} \sum_{j \in \mathcal{N}_i} w_{ij} \left\| (\hat{v}_i - \hat{v}_j) - R_i(v_i - v_j) \right\|^2 \tag{4.2}$$

in which v_i and $\hat{v}_i$ are coordinates of vertex i before and after deformation, $\mathcal{N}_i$ denotes neighboring vertices of vertex i, w_{ij} are cotangent weights and R_i is the optimal rotation matrix as discussed in [245].

Joints Loss. If we do not constrain vertices of the layered mesh, different limbs can move away from each other, resulting in unrealistic outputs. In order to prevent this, we specify 'joints' $(p_i, q_i), i = 1, \ldots, n$ as pairs of vertices in different layers that must remain close to each other after deformation. We manually specify the joints, and penalize the sum of distances between vertices in each

joint:

$$L_{joints} = \sum_{i=1}^{n} \|p_i - q_i\|^2 \tag{4.3}$$

Our final loss for training the Deformation Network is a linear combination of the aforementioned losses:

$$L_{total} = L_{rec} + \lambda_1 \cdot L_{reg} + \lambda_2 \cdot L_{joints} \tag{4.4}$$

We use $\lambda_1 = 2500$ and $\lambda_2 = 10^4$ in the experiments.

4.3.3 Appearance Refinement Network

While articulations can be mostly captured by deformation network, some appearance variations such as artistic stylizations, shading effects, and out-of-plane motions cannot be generated by warping a single reference. To address this limitation, we propose an appearance refinement network that processes the image produced by rendering the deformed puppet. Our architecture and training procedure is similar to conditional Generative Adversarial Network (cGAN) approach that is widely used in various domains [184, 288, 120]. The corresponding architecture is shown in Figure 4.3. The generator refines the rendered image to look more natural and more similar to the input image. The discriminator tries to distinguish between input frames of character poses and generated images. These two networks are then trained in an adversarial setting [91], where we use pix2pix architecture [120] and Wasserstein GAN with Gradient Penalty for adversarial loss [12, 100]:

$$L_G = -\mathbb{E}\left[D(G(x_{\text{rend}}))\right] + \gamma_1 \left\| G(x_{\text{rend}}) - x_{\text{input}} \right\|_1 \tag{4.5}$$

And the discriminator's loss is:

$$L_D = \mathbb{E}[D(G(x_{\text{rend}}))] - \mathbb{E}[D(x_{\text{real}})] + \gamma_2 \, \mathbb{E}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \tag{4.6}$$

in which $D(\cdot)$ and $G(\cdot)$ are the discriminator and the generator, $\gamma_1, \gamma_2 \in \mathbb{R}$ are weights, x_{input} and x_{rend} are the input and rendered images, x_{real} is an image sampled from the training set, and $\hat{x} = \epsilon \, G(x_{\text{rend}}) + (1 - \epsilon) \, x_{\text{real}}$ with ϵ uniformly sampled from the $[0, 1]$ range. The cGAN is trained independently after training the Deformation Network as this results in more stable training. Note that one could also use deformed results directly without the refinement network.

4.4 Results and Applications

We evaluate how well our method captures (i.e., encodes and reconstructs) character appearance. We use six character animation sequences from various public sources. Figure 4.4 shows some qualitative results where for each input image we demonstrate output of the deformation network (*rendered*) and the final synthesized appearance (*generated*). The first three characters are from Dvoroznak et al. [76] with 1280/547, 230/92 and 60/23 train/test images respectively. The last character (robot) is obtained from Adobe Character Animator [3] with 22/9 train/test images. Other characters and their details are given in the supplementary material. The *rendered* result is produced by warping a reference puppet, and thus it has fewer degrees of freedom (e.g., it cannot change texture or capture out-of-plane motions). However, it still provides fairly accurate reconstruction even for very strong motions, suggesting that our layered puppet model makes it easier to account for significant character articulations, and that our image encoding can successfully predict these articulations even though it

Figure 4.4: Input images, our rendered and final results, followed by results obtained with PWC-Net [247] and DAE [235] (input images for the first three characters are drawn by ©Zuzana Studená. The fourth character ©Adobe Character Animator).

was trained without strong supervision. Our refinement network does not have any information from the original image other than the re-rendered pose, but, evidently, adversarial loss provides sufficient guidance to improve the final appearance and match the artist-drawn input. To confirm that these observations hold over all characters and frames, we report L_2 distance between the target and generated images in Table 4.1. See supplemental material for more examples.

We compare our method to alternative techniques for re-synthesizing novel frames (also in Figure 4.4). One can use optical flow method, such as PWC-Net [247] to predict a warping of a reference frame (we use the frame that was used to create the puppet) to the target image. This method was trained on real motions in videos of natural scenes and tends to introduce strong distortion artifacts when matching large articulations in our cartoon data. Various autoencoder approaches can also be used to encode and reconstruct appearance. We compare to a state-of-the-art approach that uses deforming auto-encoders [235] to disentangle deformation and appearance variations by separately predicting a warp field and a texture. This approach does not decode character-specific structure (the warp field is defined over a regular grid), and thus also tends to fail at larger articulations. Another limitation is that it controls appearance by altering a pre-warped texture, and thus cannot correct for any distortion artifacts introduced by the warp. Both methods have larger reconstruction loss in comparison to our rendered as well as final results (see Table 4.1).

One of the key advantages of our method is that it predicts deformation parameters, and thus can retain resolution of the artist-drawn image. To illustrate this, we render the output of our method as 1024×1024 images using vanilla

	Char1	Char2	Char3	Char4	Avg
Rendered	819.8	732.7	764.1	738.9	776.1
Generated	**710.0**	**670.5**	**691.7**	**659.2**	**695.3**
PWC-Net	1030.4	1016.1	918.3	734.6	937.1
DAE	1038.3	1007.2	974.8	795.1	981.6

Table 4.1: Average L_2 distance to the target images from the test set. Rendered and generated images from our method are compared with PWC-Net [247] and Deforming Auto-encoders [235]. The last column shows the average distance across six different characters.

OpenGL renderer in the supplementary material. The final conditional generation step can also be trained on high-resolution images.

4.4.1 Inbetweening

In traditional animation, a highly-skilled artist creates a few sparse keyframes and then in-betweening artists draw the other frames to create the entire sequence. Various computational tools have been proposed to automate the second step [138, 248, 26, 274], but these methods typically use hand-crafted energies to ensure that intermediate frames look plausible, and rely on the input data to be provided in a particular format (e.g., deformations of the same puppet). Our method can be directly used to interpolate between two raw images, and our interpolation falls on the manifold of deformations learned from training data, thus generating in-betweens that look similar to the input sequence.

Given two images x_1 and x_2 we use the encoder in deformation network to obtain their features, $z_i = E(x_i)$. We then linearly interpolate between z_1 and z_2 with uniform step size, and for each in-between feature z we apply the rest

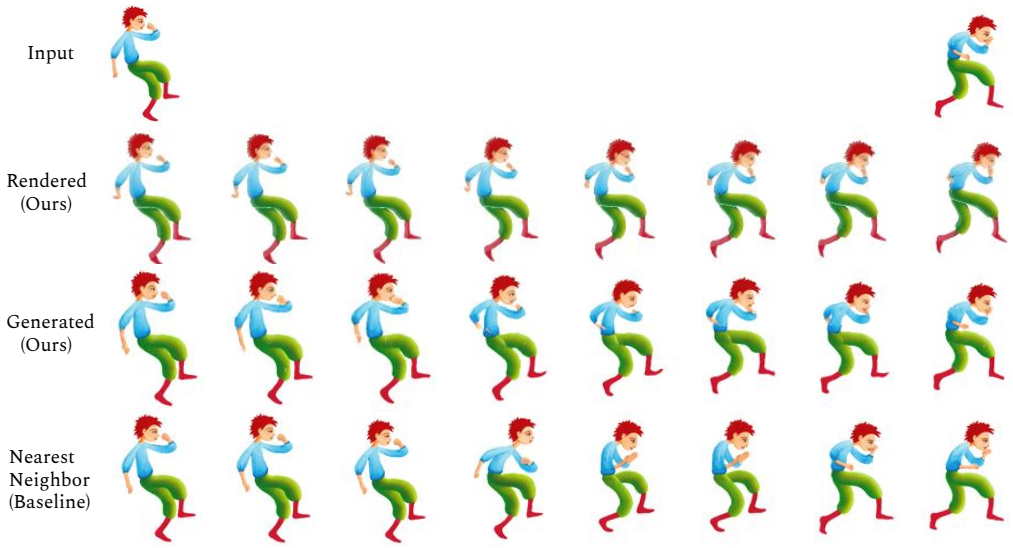


Figure 4.5: Inbetweening results. Two given images (first row) are encoded. The resulting latent vectors are linearly interpolated yielding the rendered and generated (final) images. For each generated image, the corresponding Nearest Neighbor image from the training set is retrieved.

of our network to synthesize the final appearance. The resulting interpolations are shown in Figure 4.5 and in supplemental video. The output images smoothly interpolate the motion, while mimicking poses observed in training data. This suggests that the learned manifold is smooth and can be used directly for example-driven animation. We further confirm that our method generalizes beyond training data by showing nearest training neighbor to the generated image (using Euclidean distance as the metric).

4.4.2 User-constrained Deformation

Animations created with software assistance commonly rely on deforming a puppet template to target poses. These deformations are typically defined by

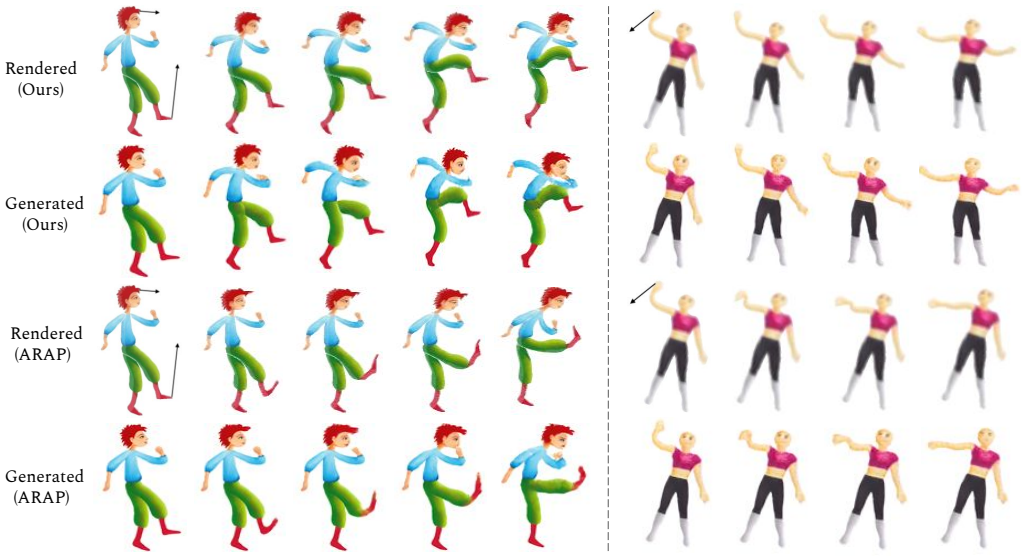


Figure 4.6: User-constrained deformation. Given the starting vertex and the desired location (shown with the arrow), the model learns a plausible deformation to satisfy the user constraint. Our approach of searching for an optimal latent vector achieves global shape consistency, while optimizing directly on vertex positions only preserves local rigidity.

local optima with respect to user-prescribed constraints (i.e., target motions) and some hand-crafted energies such as rigidity or elasticity [245, 154, 44]. This is equivalent to deciding on what kind of physical material the character is made of (e.g., rubber, paper), and then trying to mimic various deformations of that material without accounting for artistic stylizations and bio-mechanical priors used by professional animators. While some approaches allow transferring these effects from stylized animations [76], they require artist to provide consistently-segmented and densely annotated frames aligned to some reference skeleton motion. Our model does not rely on any annotations and we can directly use our learned manifold to find appropriate deformations that satisfy user constraints.

Figure 4.7: Characters in the wild. The model learns both the outline and the pose of the character (input frames and the character ©Soyuzmultfilm).

Given the input image of a character x, the user clicks on any number of control points $\{p_i\}$ and prescribes their desired target positions $\{p_i^{\text{trg}}\}$. Our system then produces the image x' that satisfies the user constraints, while adhering to the learned manifold of plausible deformations. First, we use the deformation network to estimate vertex parameters to match our puppet to the input image: $\mathbf{v} = D(E(x))$ (where $E(\cdot)$ is the encoder and $D(\cdot)$ is the decoder in Figure 4.3). We observe that each user-selected control point p_i can now be found on the surface of the puppet mesh. One can express its position as a linear combination of mesh vertices, $p_i(\mathbf{v})$, where we use barycentric coordinates of the triangle that encloses p_i. The user constrained can be expressed as an energy, penalizing distance to the target:

$$L_{user} = \sum_i \left\| p_i(\mathbf{v}) - p_i^{\text{trg}} \right\|^2 \tag{4.7}$$

For the deformation to look plausible, we also include the regularization terms:

$$L_{deform} = L_{user} + \alpha_1 \cdot L_{reg} + \alpha_2 \cdot L_{joints} \tag{4.8}$$

We use $\alpha_1 = 25$, $\alpha_2 = 50$ in the experiments.

Since our entire deformation network is differentiable, we can propagate the

gradient of this loss function to the embedding of the original image $z_0 = E(x)$ and use gradient descent to find z that optimizes L_{deform}:

$$z \longleftarrow z - \eta \nabla_z L_{deform} \tag{4.9}$$

where $\eta = 3 \times 10^{-4}$ is the step size. In practice, a few (2 to 5) iterations of gradient descent suffice to obtain satisfactory results, enabling fast, interactive manipulation of the mesh by the user (in the order of seconds).

The resulting latent vector is then passed to the decoder, the renderer and the refinement network. Figure 4.6 (left) illustrates the results of this user-constrained deformation. As we observe, deformations look plausible and satisfy the user constraints. They also show global consistency; for instance, as we move one of the legs to satisfy the user constraint, the torso and the other leg also move in a consistent manner. This is due to the fact that the latent space encodes high-level information about the character's poses, and it learns that specific poses of torso are likely to co-occur with specific poses of legs, as defined by the animator.

We compare our method with optimizing directly in the vertex space using the regularization terms only (Figure 4.6, right). This approach does not use the latent representation, and thus does not leverage the training data. It is similar to traditional energy-based approaches, where better energy models might yield smoother deformation, but would not enforce long-range relation between leg and torso motions.

	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.025$
Ours	**67.18**	**46.39**	**24.17**
UCN	67.07	43.84	21.50
PWC-Net	62.92	40.74	18.47

Table 4.2: Correspondence estimation results using PCK as the metric. Results are averaged across six characters.

4.4.3 Correspondence Estimation

Many video editing applications require inter-frame correspondence. While many algorithms have been proposed to address this goal for natural videos [50, 161, 133, 284, 221], they are typically not suitable for cartoon data, as it usually lacks texture and exhibits strong expressive articulations. Since our Deformation Network fits the same template to every frame, we can estimate correspondences between any pair of frames via the template. Table 4.2 compares correspondences from our method with those obtained from a recent flow-based approach (PWC-Net [247]), and with a supervised correspondence method, Universal Correspondence Networks (UCN) [50]. We use the Percentage of Correct Keypoints (PCK) as the evaluation metric. Given a threshold $\alpha \in (0, 1)$, the correspondence is classified as correct if the predicted point lies within Euclidean distance $\alpha \cdot L$ of the ground-truth, in which $L = \max(\text{width}, \text{height})$ is the image size. Results are averaged across pairs of images in the test set for six different characters. Our method outperforms UCN and PWC-Net in all cases, since it has a better model for underlying character structure. Note that our method requires a single segmented frame, while UCN is trained with ground truth key-point correspondences, and PWC-Net is supervised with ground truth flow.

4.4.4 Characters in the Wild

We also extend our approach to TV cartoon characters "in the wild". The main challenge posed by these data is that the character might only be a small element of a complex scene. Given a raw video[2], we first use the standard tracking tool in Adobe After Effects [2] to extract per-frame bounding boxes that encloses the character. Now we can use our architecture to only analyze the character appearance. However, since it still appears over a complex background, we modify our reconstruction loss to be computed only over the rendered character:

$$L_{\text{rec}}^{\text{masked}} = \frac{\left\| x \odot R(V_{pred}, F, I^{\mathbb{1}}) - R(V_{pred}, F, I^{uv}) \right\|^2}{\sum R(V_{pred}, F, I^{\mathbb{1}})} \tag{4.10}$$

where x is the input image, $\odot$ is the Hadamard product, and $R(V_{pred}, F, I^{\mathbb{1}})$ is a mask, produced by rendering a mesh with 1-valued (white) texture over a 0-valued (black) background. By applying this mask, we compare only the relevant regions of input and rendered images, i.e. the foreground character. The term in the denominator normalizes the loss by the total character area. To further avoid shrinking or expansion of the character (which could be driven by a partial match), we add an area loss penalty:

$$L_{area} = \left| \sum R(V_{pred}, F, I^{\mathbb{1}}) - \sum R(V_{init}, F, I^{\mathbb{1}}) \right|^2 \tag{4.11}$$

Our final loss is defined similarly to Equation 4.4 but uses the masked reconstruction loss $L_{\text{rec}}^{\text{masked}}$ and adds L_{area} loss with weight 2×10^{-3} (L_{reg} and L_{joints} are included with original weights). We present our results in Figure 4.7, demonstrating that our framework can be used to capture character appearance in the wild.

[2] e.g. https://www.youtube.com/watch?v=hYCbxtzOfL8

4.5 Conclusion and Future Work

We present novel neural network architectures for learning to register deformable mesh models of cartoon characters. Using a template-fitting approach, we learn how to adjust an initial mesh to images of a character in various poses. We demonstrate that our model successfully learns to deform the meshes based on the input images. Layering is introduced to handle occlusion and moving limbs. Varying motion and textures are captured with a Deformation Network and an Appearance Refinement Network respectively. We show applications of our model in inbetweening, user-constrained deformation and correspondence estimation. In the future, we consider using our model for applications such as motion re-targetting.

CHAPTER 5

INTERPOLATIVE AUTOENCODERS FOR UNSUPERVISED FEW-SHOT IMAGE GENERATION

5.1 Introduction

Recent work has produced powerful neural models for image generation. These models can synthesize high-quality [129, 219, 291], diverse [87, 179, 219], and high-resolution [37, 126, 129] images never before seen in the training data. We are approaching the point where we can train effective generative image models for *any* class of object, *given a large training dataset for these classes* [58].

However, this requirement of a large dataset is impractical in many scenarios. For example, an artist or designer might want to use these techniques to help create concept art of futuristic vehicles. Roboticists might want to create a diverse simulated environment that can be used to train agents to be robust in the real world. To enable these applications and others, we need generative modeling techniques capable of synthesizing images from a large, *ever-growing* list of object classes. Few of these classes will have large sets of labeled images easily available, and the vast majority will be *unknown* at the time of training.

We therefore need generative models that can train on one set of image classes, and then generalize to a new class using only a small quantity of new images. This is the problem of *few-shot image generation*. This problem is extremely challenging since it involves tackling two open research problems simultaneously. First, the generative model must be capable of even *representing* novel class images in its latent space (*cross-category generalization*). Second, we

must be able to *sample* diverse novel class generations given only a few examples of the novel class (*few-shot synthesis*).

While representations learned by *classification networks* are well known to generalize to new classes [80, 88, 218, 241, 265], we do not know if this is true for generative models. In fact, we find that the latest and greatest generative models *do not generalize*. This is clear in Figure 5.1, where a Progressive Growing GAN [127] trained on adult faces struggles to even *represent* baby faces. Perhaps because of this generalization challenge, recent attempts at few-shot image generation rely on simplifying assumptions and compromises that are ultimately undesirable. Most existing models tend to require *labeled* datasets of thousands of classes, which can be impractical to collect [77]. Others involve substantial computation at test time [53]. Yet others end up displaying highly domain-specific behavior, generalizing to a new object class reliably only if it is similar to the training classes [122].

In this paper, we introduce a strong, efficient, *unsupervised* baseline for few-shot image generation that avoids *all* of the above compromises. Our key insight derives from our finding that while the latent space of powerful generative models, such as VAEs and GANs, does not generalize to new classes (see Fig. 5.1), the representation learned by vanilla autoencoders generalizes extremely well. Unfortunately, autoencoders cannot *synthesize* new novel class images. To remedy this, we introduce a new training method to encourage a more interpretable and informative latent space, which allows for meaningful interpolation. We demonstrate on three different settings (handwritten characters, faces and general objects) that the resulting *Interpolative Autoencoder* achieves *simple, robust, highly general, and completely unsupervised* few-shot image generation.

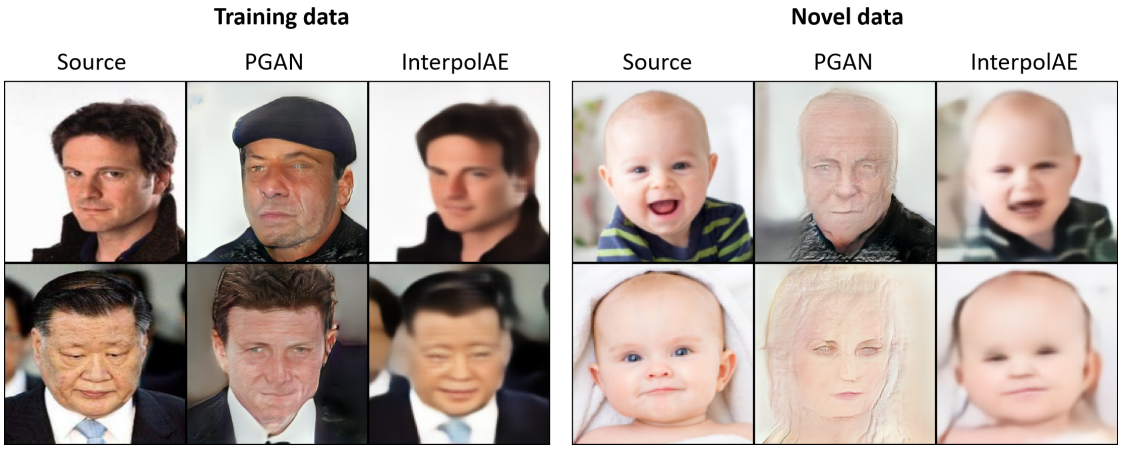


Figure 5.1: A Progressive Growing GAN - a powerful, high-resolution, state-of-the-art image generator - is trained on Celeb-A and reasonably recovers training images using a learned inversion network followed by latent space optimization [25]. However, when we attempt the same reconstruction procedure on images from a novel class (babies), the semantic content is lost: the reconstructed images are clearly not babies (right side, middle column). A simple, six-layer interpolative autoencoder trained on the same dataset, while slightly blurry, successfully preserves the semantic content in both cases (rightmost columns). Details are in appendix, images best viewed digitally.

5.2 Related Work

5.2.1 Generative Modeling

Neural generative modeling was initially a mere by-product of unsupervised learning. Autoencoders were originally proposed as a form of learned non-linear data compression, which could then be used for downstream tasks; the generator network was discarded completely [140]. This held true even when autoencoders were first applied to image data [111, 180]. VAEs do the opposite: by pushing the latent space toward a prior distribution during training,

the encoder network can be discarded at test time instead. New images are sampled directly from the prior distribution [136]. Subsequent models discard the encoder network entirely. GANs sample directly from a noise distribution and learn to generate images which fool a concurrently-trained real/fake image discriminator [91]. GLO [31] and related GLANN [114] architectures treat latent codes as learnable parameters directly, and train separate sampling procedures for synthesizing novel images.

While all of these approaches can generate new images, it is unclear if any of them can quickly generalize to novel object classes. Some results suggest the opposite: a VAE sufficiently powerful to model the training data becomes incapable of producing anything else [34].

5.2.2 Few-Shot Image Generation

Current attempts at few-shot image generation span a wide range of approaches and models. Neural Statistician, an early attempt, is similar to the early autoencoder in that it is built for few-shot classification, and largely discards the generative capability [77]. Generation-oriented iterations of the same idea exist, but likewise depend on a large and varied set of labelled data for training [109]. Other approaches based on few-shot classification algorithms include generative matching networks [24] and adversarial meta-learning [53]. These models also depend on heavy supervision from the training data, and as a general rule, are fairly complicated, involving multiple networks and training procedures working in tandem - making them potentially difficult to reliably train in practice.

Separate work has converged on the problem of few-shot image generation from the side of generative modeling. [271, 193] and [276] investigate the ability of GANs to handle domain adaptation via fine-tuning - thus requiring substantial computation, and more novel class examples than are likely available in the true few-shot setting. DAGAN [11] and FUNIT [163] use adversarial training to produce entirely feed-forward few-shot image generators. However, both algorithms still depend on labelled training data from a variety of classes, and risk exhibiting the same problematic behaviors as standard GANs: mode collapse and sensitivity to hyperparameters [14, 15].

[122] introduces an algorithm for class-conditioning an unconditioned generative model. By matching batch statistics with real images in a pretrained latent space, the algorithm produces novel images from the same, possibly novel class. However, the model is never evaluated on novel classes - it is unclear if it generalizes. It also requires heavy computation at test time.

In contrast to these prior works, interpolative autoencoders are simple and lightweight, train robustly, and generalize broadly from unlabelled data.

5.2.3 Interpolation

Ours is not the first attempt to improve latent space interpolation in autoencoders, or to recognize the need. [228] and [28] both use adversarial training to improve interpolation quality, while [167] uses interpolation under certain priors for data augmentation. None of these techniques, however, have been evaluated in the context of novel classes.

5.3 Problem Setup

Let X be a large collection of images depicting objects from a set of distinct classes C. We assume that image-level class labels exist, and are based on the semantic content of each image. We do not have access to these labels, however, or even know what or how many classes there are. Let X' be a very small collection of images - as few as two - all belonging to a single, novel class c' where $c' \notin C$. Our goal is to train an image generator on X which can rapidly adapt to X', such that subsequent generated images clearly belong to class c'.

This is an extremely difficult problem, as it encompasses two different challenges simultaneously. The first challenge is image synthesis: how to produce a novel image x such that $x \notin X$ while still belonging to some class $c \in C$. The second challenge is generalization: how to ensure that a model trained on X is still capable of producing images from some novel, unknown class $c' \notin C$.

We argue that the problem of generalization is much more difficult than synthesis. If X is highly constrained or domain-specific, then we would expect any network trained on X to model the data in an appropriately constrained and specific manner. This behavior is frequently observed in neural classifiers: it is common knowledge that fine-grained classifiers trained for a narrow, specific domain do not generalize well to different domains. Thus it is not clear that a model trained on X should generalize to novel class c' at all.

Therefore, we first examine how well existing generative models generalize. We train a VAE [136] and a WGAN-GP [99] on MNIST [151] handwritten digits (details in appendix). The networks converge nicely and achieve good synthesized image quality (see Fig. 5.2, third row). We then evaluate the ability

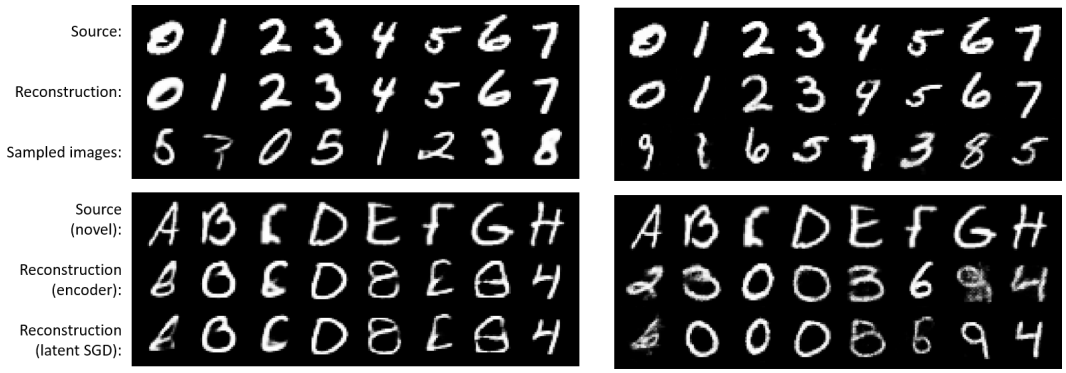


Figure 5.2: A VAE (left) and a WGAN-GP (right) are trained on MNIST (top row), and successfully recover the training images (second row) and synthesize new ones (third row). However, they struggle to represent images from novel classes (EMNIST letters, third and fourth rows). Even when latent codes are optimized directly, representing an oracle encoder, reconstruction quality is very poor. The generator itself is incapable of representing novel classes. Best viewed digitally.

of each generative model to recover particular images. We train a separate encoder network for the WGAN-GP which inverts the learned image/latent vector mapping of the generator. Using the built-in encoder of the VAE, and the new inversion network for the WGAN-GP, we find that both generative models can successfully autoencode and reconstruct a given set of training images (Fig. 5.2, top two rows). This is not surprising: the models are powerful and successfully learn the training data distribution. However, the same approach clearly fails on images coming from novel classes - in this case, handwritten letters from EMNIST [54]. The outputs do not resemble the inputs; crucial semantic information is lost (Fig. 5.2, fourth and fifth rows). It could still be the case that the generator networks are capable of producing letters, but require better latent codes. We test this by simulating an oracle encoder network, by refining the latent codes via SGD using a pixel reconstruction loss. The resulting reconstructions are not much better (Fig. 5.2, bottom row), indicating that it is the generators, not the

encoders, which fail to generalize to novel classes. This result confirms prior findings [34], and demonstrates the difficulty of few-shot image generation: despite their power and sophistication, current generative approaches simply *cannot* recover images from novel classes. Fig. 5.1 demonstrates a similar failure in a large, state-of-the-art GAN model.

Why do sophisticated generative models fail to generalize? We argue this is largely by design. Generative models such as VAEs and GANs are trained to minimize the divergence between a prior distribution and a learned posterior distribution. VAEs assign a prior over the space of latent embeddings $p(z)$, and push the posterior distribution toward it, where the posterior is a learned distribution over the training data $p(E(x))$, and E is the encoder network. GANs, on the other hand, minimize a divergence between prior and posterior distributions in image space, rather than latent space. The adversarial generator $G(z)$ parameterizes a learned distribution on images, which is trained to approach the true distribution of training data. The divergence between these distributions is parameterized and dynamically updated by the adversarial discriminator network. Regardless of the exact mechanism, however, training both models involves approximating a distribution via repeated sampling in latent space, and sending those samples through the generator network. Thus, by the time the generator is trained to convergence, and the posterior approaches the prior (or vice-versa), every region of the latent space feasible under the prior will have been mapped at some point to a training image - or, in the case of GANs, to an image indistinguishable from a training image. This means that a properly trained VAE or GAN cannot construct or reconstruct new object classes. If it could, then it would have been able to sample such images during training - which would mean that it had not been properly trained at all.

Figure 5.3: A random selection of autoencoder image reconstructions for unseen classes. From left to right: MNIST generalizes to EMNIST, Omniglot train generalizes to Omniglot test, and CIFAR10 generalizes to CIFAR100. Best viewed digitally.

5.4 Interpolative AutoEncoders

Minimizing the divergence between prior and posterior training data distributions ensures good performance in image synthesis, but also ensures poor generalization. We conjecture that the opposite might also be true: plain autoencoders do not enforce any particular latent distribution on the data posterior, and thus it is unclear how to sample new images. However, this could also mean that they generalize well. More formally, given a network architecture E which maps an input image x to latent vector z, and a generator architecture G which maps the latent vector z back to image space, we refer to the function composition $G(E(\cdot))$ as the autoencoder. E and G are trained jointly over X such that the pixel reconstruction error between $x \in X$ and $G(E(x))$ is minimized. The question of generalization becomes, to what degree does a trained autoencoder maintain close proximity between x' and $G(E(x'))$ for x' which lies far from the data manifold of X?

We find that our conjecture holds: autoencoders generalize *extremely* well. Examples are given in Fig. 5.3, demonstrating near-perfect generalization performance over three pairs of class-disjoint datasets: MNIST digits to EMNIST letters, Omniglot training alphabets to Omniglot testing alphabets [145], and

	MNIST	EMNIST	Omni (train)	Omni (test)
# training images:	*60,000*	*124,800*	*19,280*	*13,180*
MNIST	2.29	9.89	6.03	6.88
EMNIST	3.79	2.73	4.48	5.30
Omniglot (train)	8.36	14.6	1.92	4.62
Omniglot (test)	9.41	15.2	4.59	2.16
MNIST (VAE)	6.25	23.0	15.9	17.9

	CIFAR-10	CIFAR-100
# training images:	*50,000*	*50,000*
CIFAR-10	4.39	5.40
CIFAR-100	5.27	4.63

Table 5.1: Average autoencoder per-pixel L1 reconstruction error. Values are normalized to the color output range [0, 255]. Rows give the training dataset, columns give the evaluation dataset, and cells measure generalization from row to column. Generally, error is very low. The bottom row contains scores for a VAE, which generalizes poorly.

CIFAR10 to CIFAR100 [141]. A quantitative evaluation can be found in Table 5.4. In addition to the three dataset pairings in Fig. 5.3, we evaluate generalization in the other direction, from EMNIST to MNIST, Omniglot test to Omniglot train, and CIFAR100 to CIFAR10. Finally, we test the ability of MNIST/EMNIST and Omniglot networks to generalize to each others' datasets in the face of domain shift (stroke width in Omniglot is almost universally thinner than for MNIST/EMNIST). Reconstruction quality is high across the board - extremely high, accounting for the fact that the MNIST and CIFAR networks were trained on only ten distinct classes! Autoencoders appear to exhibit hardly any overfitting whatsoever with respect to object classes, no matter how constrained those classes may be. Instead, it appears that even in cases where the domain of the training data is narrow, autoencoders learn an extremely general mapping of pixels to latent embeddings and back.

Figure 5.4: Autoencoders do not generalize trivially to any input. Color inversion on the training data represents a clear failure mode.

It is worth asking to what degree the mapping extracts semantically meaningful information in the first place. If the intrinsic dimensionality of the image manifold is low enough, then perhaps the autoencoder is not compressing the data at all, in which case it will generalize trivially to *any* image input. Fortunately, we see that this is not the case: when the input data is too far from the training distribution, autoencoders do fail. Fig. 5.4 displays one such failure mode for the MNIST and Omniglot networks: simple color inversion. The mapping learned by an autoencoder, while general, is also nontrivial.

5.4.1 Interpolative Training

The fact that autoencoders generalize indicates that they are capable of acting as few-shot image generators for novel classes. The generator network G can construct a wide range of images. The challenge is in determining a reliable way to *sample* appropriate latent z vectors, given only a small set of real example images. A simple way to do this is to interpolate between data points in latent space: every sampled point should be a weighted sum of two or more real points. This allows us to produce novel combinations of our given reference images without changing the overall semantic content.

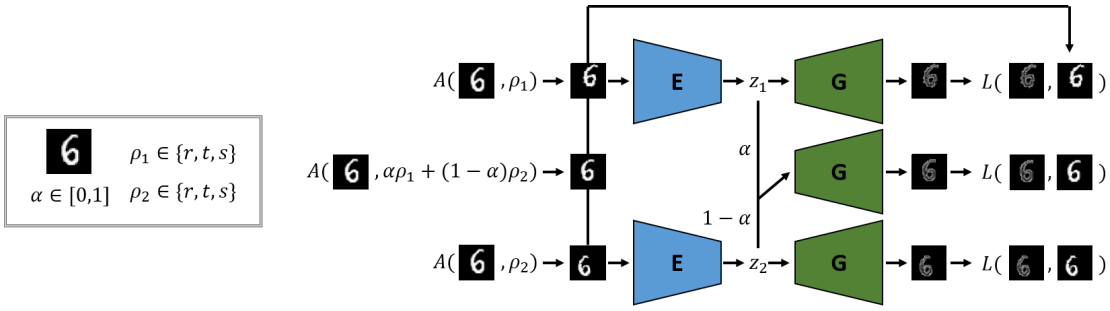


Figure 5.5: The training process for interpolative autoencoders. For every training image, two affine transformations ρ are sampled from predefined ranges of rotation, translation, and scale, as well as a mixing coefficient α. In addition to reconstructing the transformed images (top and bottom branches), the network is also trained to recover the mixed transformation image, by applying the same mixture in latent space. This forces the interpolation path in latent space to encode only spatially - and thus semantically - meaningful information.

Unfortunately, it is a known fact that autoencoders do not interpolate well [28]. Semantic information is sometimes only weakly encoded in the learned mapping from x to z and back. Thus image content is not always preserved when moving between the latent space embeddings for two images of the same object. Indeed, we observe that our autoencoders frequently introduce visual artifacts on interpolated handwritten characters (see Fig. 5.6 and 5.7, middle row). Interpolation on natural images in CIFAR largely reduces to pixel-level fading (see Fig. 5.11, second row).

We introduce a novel training procedure for autoencoders to remove these visual artifacts and improve the preservation of semantic content during latent space interpolation. In addition to reconstructing images, we train the interpolative autoencoder to also recover known, semantically interpolated images from the corresponding interpolated latent code. In particular, suppose that we have a triplet of images $A = f(\theta_1)$, $B = f(\theta_2)$ and $C = f(\alpha\theta_1 + (1-\alpha)\theta_2)$, where

f is some unknown image generating process, and θ is some semantic variable. Using this triplet, we train the interpolative autoencoder to reconstruct C by decoding the interpolated latent code of A and B. Formally, we minimize a loss L_{interp} such that

$$L_{interp} = \|C - G(\alpha E(A) + (1 - \alpha)E(B))\|_p \tag{5.1}$$

for some choice of norm p, where E and G are the encoder and generator network functions and α represents the ground-truth interpolation mixing weight.

In practice, finding appropriate image triplets A, B, C is difficult, as images in the real world generally cannot be described as straightforward combinations of two other images. Instead, we generate A, B, C *synthetically* using affine spatial transformations. For a given training image x, we randomly sample two sets of rotation, translation, and scale parameters. Applying each of these transformations independently to x yields A and B. C is constructed by random uniform sampling of $\alpha \in [0, 1]$, which is used to compute a weighted average of our paired rotation, translation, and scale parameters. This weighted average parameterizes a third affine transformation, which applied to x yields C.

The interpolative autoencoder is trained to faithfully reconstruct A and B using an image reconstruction loss. At the same time, the network receives the mixing parameter α for each image triplet, and attempts to recover C from the α-weighted latent space interpolation. This corresponds to Equation 5.1, and the network is trained to minimize L_{interp} with a choice of norm corresponding to the reconstruction loss for A and B. The full procedure is displayed in Fig. 5.5.

In practice, we weight the reconstruction and interpolation loss terms so that they contribute equally to the overall loss. Since we measure the reconstruction

loss on both A and B, the final interpolative autoencoder loss is

$$L = \frac{L_{reconstruct}}{2} + L_{interp} \tag{5.2}$$

5.5 Experiments

Through both qualitative and quantitative results, we demonstrate the effectiveness of interpolative autoencoders on the unsupervised few-shot image generation task. All encoder and decoder networks are simple 4- to 6-layer architectures. We employ such shallow networks to illustrate that the interpolative autoencoder approach, and not network power, is responsible for good performance. Training and evaluation details for all experiments can be found in the appendix.

5.5.1 Handwritten Characters

Image quality: We evaluate the performance of interpolative autoencoders on two handwritten character dataset pairs. We train one set of models on MNIST digits and evaluate on EMNIST letters, while the other generalizes from the Omniglot training alphabets to the testing alphabets. The autoencoder architecture in both cases consists of a 4-layer encoder and generator based on InfoGAN [47]. Input images have a 28×28 resolution, and the latent dimensionality of the autoencoder is 64, representing over an order of magnitude in dimensionality reduction.

Figures 5.6 and 5.7 display representative interpolation results for these two

Figure 5.6: A random selection of interpolations between image pairs from novel classes (EMNIST). Pixel interpolation is included as a simple baseline. Red squares indicate places where vanilla autoencoders break or overextend semantically important strokes. Interpolative autoencoders successfully remove these artifacts. Best viewed digitally.

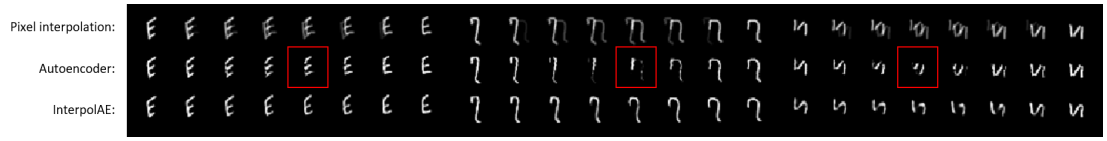


Figure 5.7: A random selection of interpolations between image pairs from novel classes (Omniglot). Pixel interpolation is included as a simple baseline. Red squares indicate places where vanilla autoencoders remove semantically important strokes. Interpolative autoencoders successfully address, or at least mitigate, these artifacts. Best viewed digitally.

contexts. It is clear that interpolative training produces better quality interpolations and removes visual artifacts present in the vanilla autoencoder images. These qualitative results are verified quantitatively in Table 5.5.1. Interpolative autoencoders, as measured by FID score [108], outperform both their vanilla counterparts and VAEs trained directly on the evaluation data, despite significant domain shift. This indicates that in terms of image quality, interpolative autoencoders are effective few-shot image generators.

Generalizability: We compare our approach to two prior few-shot generation methods: Neural Statistician [77] and Data Augmentation GANs (DAGAN) [11]. Both of these approaches require class labels on the training data, while

Model	EMNIST	Omniglot
AE	17.66	225.33
VAE	18.73	185.44
InterpolAE	17.32	169.88

Table 5.2: FID scores across models and datasets. The VAE model is trained directly on the evaluation data as a strong baseline. Interpolative autoencoders are superior in all contexts.

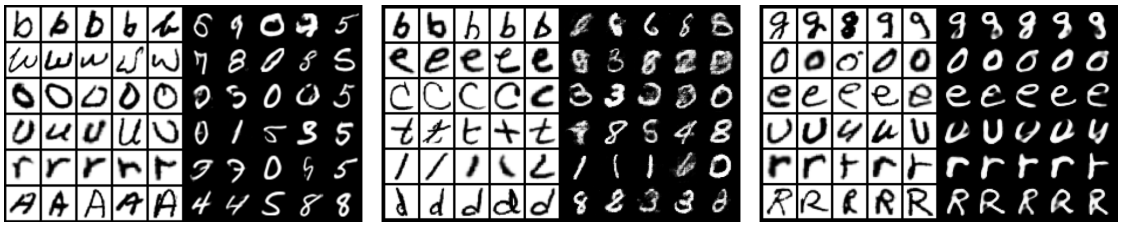


Figure 5.8: Comparison to prior work. Seed images are in color negative on the left, synthesized images on the right. Neural Statistician (left) and DAGAN (middle) overfit to the training data and fail to maintain class-consistency in the synthesized images. Interpolative autoencoders (right) successfully generate new letters.

ours does not. We train both models on MNIST and then attempt to synthesize new images from EMNIST. Fig. 5.8 makes clear that interpolative autoencoders generalize more broadly than prior approaches and are much less restricted by a narrow training data domain. Both neural statistician and DAGAN overfit to the training classes, and generate additional number images instead of the desired letters.

Data hallucination: To demonstrate the practical utility of interpolative autoencoders, we examine their performance as a form of data augmentation. A series of classifiers are trained on the "ByMerge" split of EMNIST, with the digit classes removed (i.e. letter classes only, lower- and upper-case separated). During training, half of each batch is dedicated to augmented data, with a mixing

Augmentation	Accuracy
None	90.33 ± .06
Mixup	90.66 ± .04
AE	91.27 ± .04
InterpolAE	**91.50 ± .05**

Table 5.3: Autoencoder interpolation as a form of data augmentation on EMNIST. Results are averaged over ten runs, with 95% confidence intervals. Interpolative autoencoders provide the most effective form of data augmentation.

coefficient of 0.5 in all cases.

We compare four classifiers, and find that accuracy directly improves with the sophistication of the data augmentation technique (see Table 5.5.1). Pixel-level interpolation, otherwise known as MixUp [292], provides a small boost to accuracy. Interpolating in the latent space of an autoencoder trained on MNIST yields a larger boost, and an interpolative autoencoder trained on MNIST improves further still.

5.5.2 Celeb-A

We extend our results to the more difficult domain of higher-resolution natural images. We use the Celeb-A dataset of faces, scaled to a resolution of 128×128. Our models are trained strictly on male faces, and attempt to generalize to female faces. Encoder and decoder networks are six layers deep and fully convolutional, with a latent dimensionality of 512. Since the input images have 3 color channels, the resulting dimensionality reduction is almost two orders of magnitude.

The increased output resolution, combined with the relative simplicity of our networks, makes this a much more difficult task than for handwritten characters. In practice, this difficulty manifests as blurriness in the outputs. Blur in generative models is a well-studied phenomenon and techniques exist to eliminate it. We augment our training by employing a perceptual loss [123], aligning not just pixel values to the ground truth but also activations inside a frozen feature extractor. In order to maintain the unsupervised nature of our algorithm, however, we cannot use a pretrained classifier as the feature extractor, as is common practice. Instead, we formulate an "autoperceptual loss" where the encoder of the current model is frozen and used to calculate the perceptual loss. We found that the autoperceptual loss, weighted equally with the reconstruction loss, improves convergence and removes some, though not all, of the blur in the outputs. This partial effectiveness is unsurprising, as our networks are only six layers deep. We predict that a more powerful, more sophisticated network would achieve better performance, but we focus on simple models here since we are primarily concerned with generalization, not image quality *per se*.

The results, displayed in Fig. 5.9, are similar to our findings for handwritten characters. Autoencoders generalize well, but produce visual artifacts during interpolation. In this case, the artifacts take the form of unrealistic, transparent regions around the head. Compared to the vanilla autoencoder, the interpolative autoencoder with autoperceptual loss removes these artifacts and restores semantic meaning to the interpolation path, though it sacrifices some image detail and quality to do so.

Diversity: a practical concern with using interpolation as a latent sampling mechanism is the accompanying restriction on image diversity. In the extreme

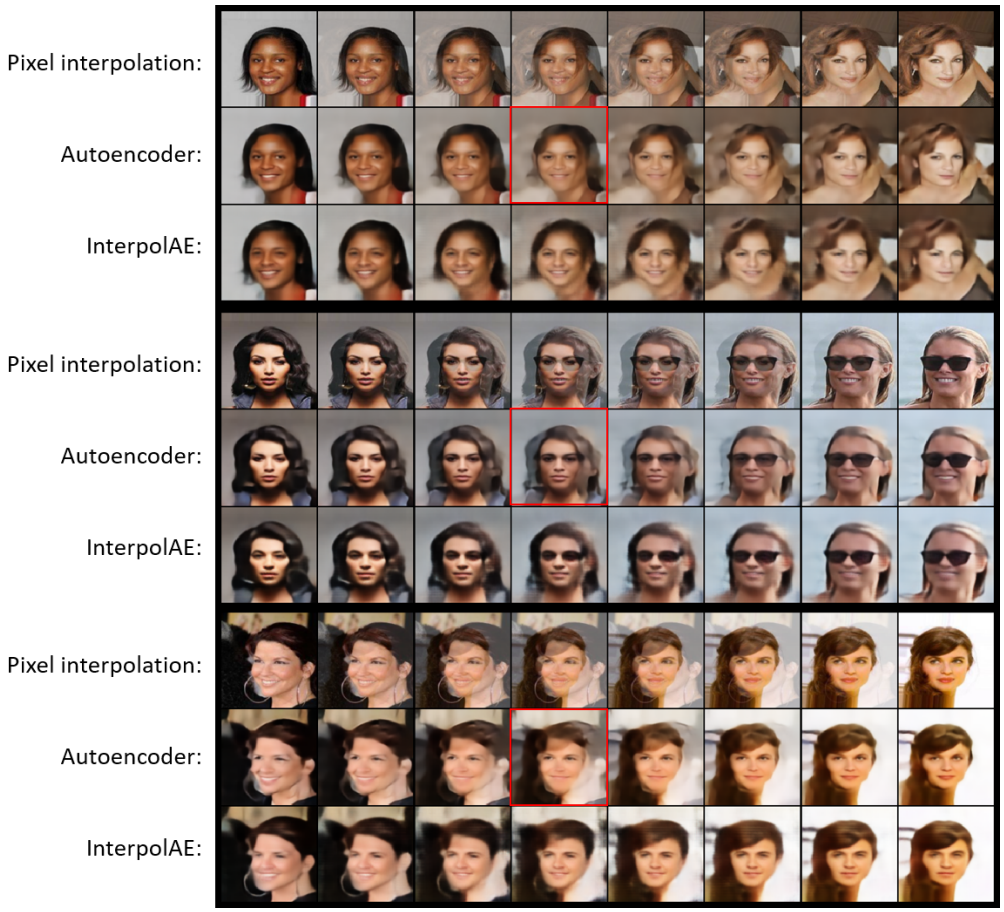


Figure 5.9: Examples of interpolations between image pairs from a novel class (Celeb-A). Pixel interpolation is included as a simple baseline. Red squares indicate places where vanilla autoencoders introduce transparency artifacts. Interpolative autoencoders successfully mitigate these artifacts. These examples are hand-picked, as transparency artifacts for the vanilla autoencoder do not appear as frequently as in the handwritten character datasets. Best viewed digitally.

case, where only two novel seed images are available, the model is forced to move back and forth between the two images and nowhere else, heavily restricting the semantic range of available outputs. We argue that this restriction seems worse than it is: the number of unique image pairs grows quadratically with the number of seed images, and Fig. 5.10 demonstrates the rich diversity

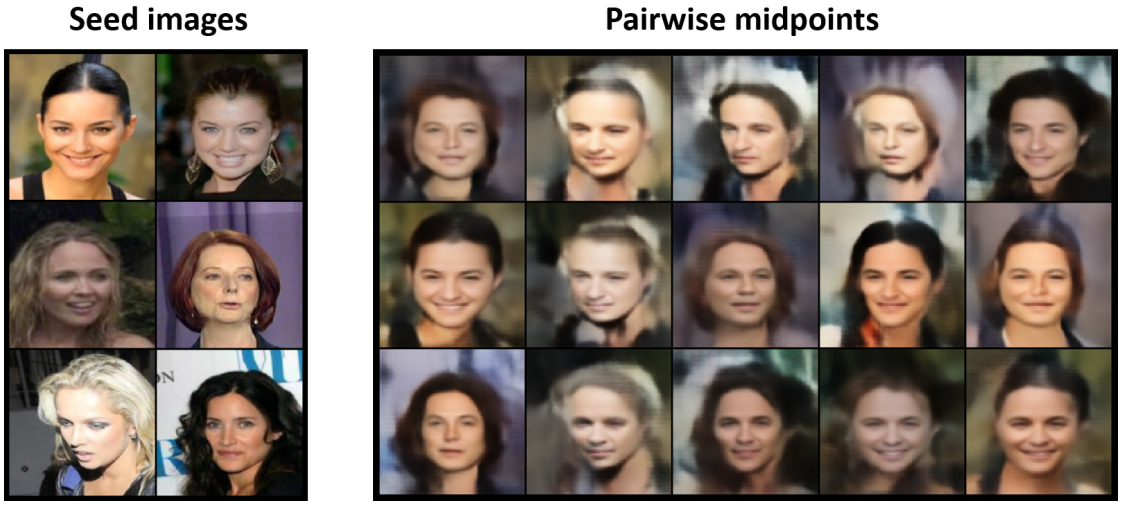


Figure 5.10: Six seed images (left) are sufficient to produce a diversity of novel images (right). The fifteen displayed here represent the midpoints of the fifteen unique pairwise interpolation paths. We re-emphasize that this network was trained only on male faces. Best viewed digitally.

of sampled images which can result from only six available seeds.

We conclude that interpolative autoencoders are effective not just on highly constrained handwritten characters, but also on higher-resolution color images.

5.5.3 CIFAR

Finally, we evaluate our model in an extremely challenging setting: natural images with no domain restriction. Unconstrained natural images represent a significant challenge for few-shot image generation, as intra-class variation is so much higher (images have very little consistent global structure). This also makes the natural image domain particularly difficult for interpolation-based models, as it is not always possible to interpolate smoothly and intuitively be-

Figure 5.11: Examples of interpolation between image pairs from novel classes (CIFAR100). Vanilla autoencoders are indistinguishable from pixel interpolation. Interpolative autoencoders produce more semantically meaningful transformations. **Left**: the interpolative autoencoder gradually spreads out an initially tight cluster of bottles. **Right**: the contour of the foliage morphs smoothly. These examples are hand-picked, as natural image interpolation is not always meaningful or interesting. Best viewed digitally.

tween two given images. If one image is a flower in a vase, and another is a flower in a field, how do you semantically interpolate between them?

We train our models on CIFAR10 and evaluate on novel CIFAR100 classes. Our network architectures are 5 layers deep and fully convolutional, with a latent dimensionality of 512. With a 32×32 image resolution and 3 color channels this amounts to a six times reduction in dimensionality. We again use an autoperceptual loss to aid convergence in the interpolative model and remove blur.

Fig. 5.11 displays example outputs from vanilla and interpolative autoencoders on novel CIFAR100 data. Again, the vanilla autoencoder produces transparency artifacts - in this case, the output is nearly indistinguishable from the pixel fade. Interpolative training is able to remove most of the transparency and restore semantic content along the interpolation path. We conclude that the combined techniques of interpolative autoencoders and autoperceptual loss are moderately effective at unsupervised few-shot image generation for natural images. It is likely that a more powerful neural architecture could achieve

better image quality - our lightweight autoencoder networks are sufficient for generalization, but cannot maintain high interpolation quality at the same time. However, we also point out that few-shot generation on natural images is inherently a very difficult problem, and many challenges remain.

5.6 Conclusion

We introduce a powerful, lightweight, and class-label-free method for few-shot image generation. Using the fact that autoencoders generalize very broadly from limited data, we incorporate a novel training procedure to produce interpolative autoencoders, which synthesize realistic images of novel classes from as few as two examples. Interpolative autoencoders are robust and generalize far more broadly than prior work, with outputs that are both good-quality and practically useful for downstream tasks. We are confident that with more powerful networks and sampling mechanisms, interpolative autoencoders can be further improved and serve as a valuable base model for unsupervised few-shot image generation.

CHAPTER 6

GENERATIVE ADVERSARIAL PERTURBATIONS

6.1 Introduction

In spite of their impressive performance on challenging tasks in computer vision such as image classification [142, 237, 249, 250, 106]and semantic segmentation [170, 18, 45, 289, 294], deep neural networks are shown to be highly vulnerable to adversarial examples, i.e. carefully crafted samples looking similar to natural images but designed to mislead a pre-trained model. This phenomenon was first studied in [251], and may hinder the applications of deep networks on visual tasks, or pose security concerns.

Two types of adversarial perturbations can be considered: Universal and Image-dependent. Image-dependent perturbations can vary for different images in the dataset. To generate these perturbations, we require a function which takes a natural image, and outputs an adversarial image. We approximate this function with a deep neural network. Universal perturbations are fixed perturbations which when added to natural images can significantly degrade the accuracy of the pre-trianed network. In this case, we seek a perturbation $\mathcal{U}$ with small magnitude such that for most natural images x, $x + \mathcal{U}$ can fool the pre-trained model. Unlike the iterative approaches proposed in the literature, we consider trainable networks for learning the universal perturbation.

From another viewpoint, adversarial attacks can be categorized as targeted and non-targeted. In targeted adversarial attacks, we seek adversarial images that can change the prediction of a model to a specific target label. In

non-targeted attacks we want to generate adversarial examples for which the model's prediction is any label other than the ground-truth label.

Considering all the possible combinations, we can have four types of adversarial examples: targeted universal, non-targeted universal, targeted image-dependent and non-targeted image-dependent. We elaborate on each of them in the following sections.

Our main contributions can be summarized as follows:

- We present a unifying framework for creating universal and image-dependent perturbations for both classification and semantic segmentation tasks, considering targeted and non-targeted attacks with L_∞ and L_2 norms as the metric.
- We improve the state-of-the-art performance in universal perturbations by leveraging generative models in lieu of current iterative methods.
- We are the first to present effective targeted universal perturbations. This is the most challenging task as we are constrained to have a single perturbation pattern and the prediction should match a specific target.
- Our attacks are considerably faster than iterative and optimization-based methods at inference time. We can generate perturbations in the order of milliseconds.

6.2 Related Work

6.2.1 Universal Perturbations

First introduced in [185], universal perturbations are fixed perturbations which after being added to natural images can mislead a pre-trained model for most of the images. The algorithm in [185] iterates over samples in a target set, and gradually builds the universal perturbation by aggregating image-dependent perturbations and normalizing the result. [188] presents a data independent approach for generating image-agnostic perturbations. Its objective is to maximize the product of mean activations at multiple layers of the network when the input is the universal perturbation. While this method obviates the need for training data, the results are not as strong as [185]. A method for generating targeted universal adversarial perturbations for semantic segmentation models is presented in [183]. Their approach is similar to [185] in that they also create the universal perturbation by adding image-dependent perturbations and clipping the result to limit the norm. [186] proposes a quantitative analysis of the robustness of classifiers to universal perturbations based on the geometric properties of decision boundaries. A defense method against universal adversarial perturbations is proposed in [4]. It learns a Perturbation Rectifying Network (PRN) from real and synthetic universal perturbations, without needing to modify the target model.

6.2.2 Image-dependent Perturbations

Various approaches have been proposed for creating image-dependent perturbations. Optimization-based methods such as [251] and [41] define a cost function based on the perturbation norm and the model's loss. Then they use gradient ascent in pixel space with optimizers such as L-BFGS or Adam [134] to create the perturbation. While these approaches yield better results than other methods, they are slow at inference time as they need to forward the input to the model several times.

[92] proposes a Fast Gradient Sign Method (FGSM) to generate adversarial examples. It computes the gradient of the loss function with respect to pixels, and moves a single step based on the sign of the gradient. While this method is fast, using only a single direction based on the linear approximation of the loss function often leads to sub-optimal results. Based on this work, [187] presents an iterative algorithm to compute the adversarial perturbation by assuming that the loss function can be linearized around the current data point at each iteration. [143] introduces the *Iterative Least-Likely Class* method, an iterative gradient-based method choosing the least-likely prediction as the desired class. This method is applied to ImageNet in [144]. It also discusses how to effectively include adversarial examples in training to increase model's robustness. [52] proposes a method for directly optimizing performance measures, even when they are combinatorial and non-decomposable. [191] generates images unrecognizable to humans but classified with high confidence as members of a recognizable class. It uses evolutionary algorithms and gradient ascent to fool deep neural networks. Our work bears a resemblance to [21] in that it also considers training a network for generating adversarial examples. However, [21]

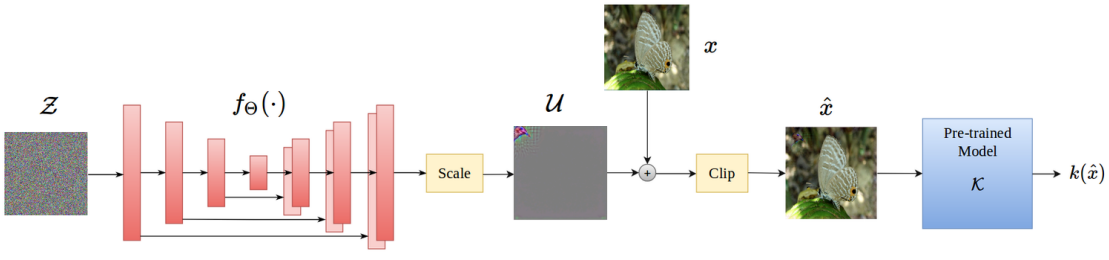


Figure 6.1: Training architecture for generating universal adversarial perturbations. A fixed pattern, sampled from a uniform distribution, is passed through the generator. The scaled result is the universal perturbation which, when added to natural images, can mislead the pre-trained model. We consider both U-Net (illustrated here) and ResNet Generator architectures.

does not provide a fixed bound on the perturbation magnitude, which might make perturbations detectable at inference time. It is also limited to targeted image-dependent perturbations. [280] extends adversarial examples from the task of image classification to semantic segmentation and object detection. For each image, it applies gradient ascent in an iterative procedure until the number of correctly predicted targets becomes zero or a maximum iteration is reached. Similar to [251] and [41], this method suffers from being slow at inference time. [13] evaluates the robustness of segmentation models against common attacks. [174] suggests that adversarial examples are sensitive to the angle and distance at which the perturbed picture is viewed. [17] presents a method for generating adversarial examples that are robust across various transformations.

Several methods have been proposed for defending against adversarial attacks. While our focus is on efficient attacks, we refer the reader to [176, 279, 101, 226, 175, 243, 230, 256, 16, 62, 213, 65, 264, 215, 209, 273] for recent works on defense.

6.3 Generative Adversarial Perturbations

Consider a classification network $\mathcal{K}$ trained on natural images from C different classes. It assigns a label $\mathcal{K}(x) \in \{1, \ldots, C\}$ to each input image x[1]. We assume that images are normalized to $[0, 1]$ range. Let $\mathfrak{N} \subset [0, 1]^n$ represent the space of natural images[2]. We assume that $\mathcal{K}$ achieves a high accuracy on natural images. Therefore, if we denote the correct class for image x by c_x, $\mathcal{K}(x) = c_x$ for most $x \in \mathfrak{N}$. Let $\mathcal{A}_\mathcal{K}$ stand for the space of adversarial examples for the network $\mathcal{K}$. Images in $\mathcal{A}_\mathcal{K}$ must resemble a natural image yet be able to fool the network $\mathcal{K}$. Hence, for each $a \in \mathcal{A}_\mathcal{K}$ there exists $x \in \mathfrak{N}$ such that $d(a, x)$ is small and $\mathcal{K}(a) \neq c_x$, where $d(\cdot, \cdot)$ is a distance metric.

This framework can be easily extended to the task of semantic segmentation in which the correct class for each pixel needs to be determined. In this case, the segmentation network $\mathcal{K}$ assigns a label map $\mathcal{K}(x) = (\mathcal{K}(x_1), \ldots, \mathcal{K}(x_n)) \in \{1, \ldots, C\}^n$ to each image $x = (x_1, \ldots, x_n)$. The ground-truth prediction for image x is $c_x = (c_{x_1}, \ldots, c_{x_n})$, and the set of adversarial examples is $\mathcal{A}_\mathcal{K} = \{a \in [0, 1]^n \backslash \mathfrak{N} \mid \exists\, x \in \mathfrak{N} : d(a, x) < \epsilon\,,\ \forall\, i \in \{1, \ldots, n\} : \mathcal{K}(a_i) \neq c_{x_i}\}$, where ϵ is a fixed threshold[3].

6.3.1 Universal Perturbations

Universal Perturbations were first proposed in the seminal work of Dezfooli *et al.* [185]. The paper proposes an iterative algorithm to generate the uni-

[1] Note that x may or may not belong to the space of natural images.

[2] For images of height h, width w and c channels: $n = h \times w \times c$.

[3] We can also relax the constraint, and require that for *most* pixels the prediction is different from the ground-truth.

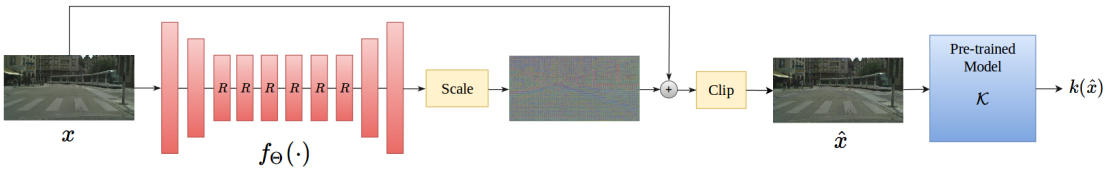


Figure 6.2: Architecture for generating image-dependent perturbations. The generator outputs a perturbation, which is scaled to satisfy a norm constraint. It is then added to the original image, and clipped to produce the perturbed image. We use the ResNet Generator architecture for most of the image-dependent tasks.

versal perturbation. It constructs the universal perturbation by adding image-dependent perturbations obtained from [187] and scaling the result. Unlike the iterative approach of [185], we seek an end-to-end trainable model for generating the universal perturbation. Let us denote the set of universal perturbations for the network $\mathcal{K}$ by $\mathcal{U}_\mathcal{K} = \{\mathcal{U} \in [0,1]^n \mid \text{for most } x \in \mathfrak{N} : x + \mathcal{U} \in \mathcal{A}_\mathcal{K}\}$. We do not want the perturbation to directly depend on any input image from the dataset. We seek a function $f : [0,1]^n \to \mathcal{U}_\mathcal{K}$ which can transform a random pattern to the universal perturbation. By changing the input pattern, we can obtain a diverse set of universal perturbations. In practice, we approximate $f(\cdot)$ with a deep neural network $f_\Theta(\cdot)$ with weights Θ. This setting resembles Generative Adversarial Networks (GANs) [91, 211, 64, 148, 116] in which a random vector is sampled from a latent space, and is transformed to a natural-looking image by a generator. In our case the range of the mapping is $\mathcal{U}_\mathcal{K}$ instead of $\mathfrak{N}$, and the generator is trained with a *fooling* loss instead of the discriminative loss used in GANs. We also tried using a combination of *fooling* and *discriminative* losses; however, it led to sub-optimal results.

There are several options for the architecture of the image transformation network $f_\Theta(\cdot)$. We consider two architectures used in recent image-to-image

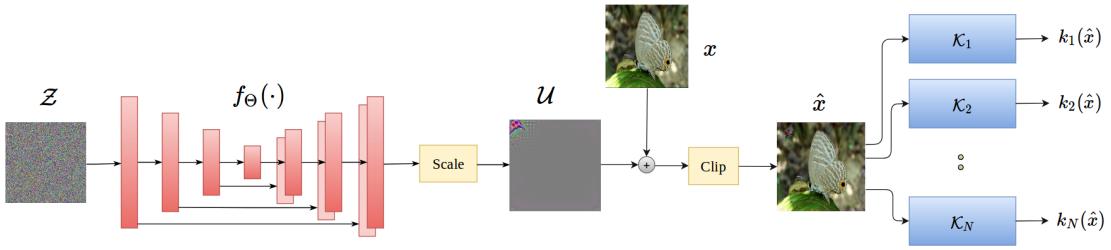


Figure 6.3: Architecture for training a model to fool multiple target networks. The fooling loss for training the generator is a linear combination of fooling losses of target models.

translation networks such as [119] and [298]. The U-Net architecture [225] is an encoder-decoder network with skip connections between the encoder and the decoder. The other architecture is ResNet Generator which was introduced in [123], and is also used in [298] for transforming images from one domain to another. It consists of several downsampling layers, residual blocks and upsampling layers. In most of our experiments, the ResNet Generator outperforms U-Net.

Figure 6.1 illustrates the architecture for generating universal perturbations. A fixed pattern $\mathcal{Z} \in [0,1]^n$, sampled from a uniform distribution $U[0,1]^n$, is fed to a generator f_Θ to create the perturbation. The output of the generator $f_\Theta(\mathcal{Z})$ is then scaled to have a fixed norm. More specifically, we multiply it by $\min\left(1, \frac{\epsilon}{\|f_\Theta(\mathcal{Z})\|_p}\right)$ in which ϵ is the maximum permissible L_p norm. Similar to related works in the literature, we consider $p = 2$ and $p = \infty$ in experiments. The resulting universal perturbation $\mathcal{U}$ is added to natural images to create the perturbed ones. Before feeding the perturbed image to the generator, we clip it to keep it in the valid range of images on which the network is trained. We feed the clipped image $\hat{x}$ to the network $\mathcal{K}$ to obtain the output probabilities $k(\hat{x})$[4].

[4]Note that $\mathcal{K}(\hat{x}) = \arg\max k(\hat{x})$.

Let $\mathbb{1}_{c_x}$ denote the one-hot encoding of the ground-truth for image x. In semantic segmentation, $c_x \in \{1, \ldots, C\}^n$ is the ground-truth label map, and $k(\hat{x})$ contains the class probabilities for each pixel in $\hat{x}$. For non-targeted attacks we want the prediction $k(\hat{x})$ to be different from $\mathbb{1}_{c_x}$, so we define the loss to be a decreasing function of the cross-entropy $\mathcal{H}(k(\hat{x}), \mathbb{1}_{c_x})$. We found that the following *fooling loss* gives good results in experiments:

$$l_{non-targeted} = l_{fool} = -\log(\mathcal{H}(k(\hat{x}), \mathbb{1}_{c_x})) \tag{6.1}$$

Alternatively, as proposed by [143] and [144], we can consider the least likely class $k_{ll}(x) = \arg\min k(x)$, and set it as the target for training the model:

$$l_{non-targeted} = l_{fool} = \log(\mathcal{H}(k(\hat{x}), \mathbb{1}_{k_{ll}(x)})) \tag{6.2}$$

In practice, the losses in equations 6.1 and 6.2 lead to competitive results. We also found that for the Inception model, the logit-based loss used in [29, 41] yields optimal results.

For targeted perturbations we consider the cross-entropy with the one-hot encoding of the target:

$$l_{targeted} = l_{fool} = \log(\mathcal{H}(k(\hat{x}), \mathbb{1}_t)) \tag{6.3}$$

where t represents the target. Note that for the classification task, $t \in \{1, \ldots, C\}$ is the target class while in semantic segmentation, $t \in \{1, \ldots, C\}^n$ is the target label map.

6.3.2 Image-dependent Perturbations

We consider the task of perturbing images as a transformation from the domain of natural images to the domain of adversarial images. In other words, we re-

quire a mapping $f : \mathfrak{N} \rightarrow \mathcal{A}_{\mathcal{K}}$ which generates a perturbed image $f(x) \in \mathcal{A}_{\mathcal{K}}$ for each natural image $x \in \mathfrak{N}$. A desirable function $f(\cdot)$ must result in a low accuracy and a high *fooling ratio*. Accuracy denotes the proportion of samples x for which $\mathcal{K}(f(x)) = c_x$, while fooling ratio represents the ratio of images x for which $\mathcal{K}(f(x)) \neq \mathcal{K}(x)$. Since we assume that the model achieves a high accuracy on natural images, these two metrics are highly correlated.

We consider two slightly different approaches for approximating $f(\cdot)$. The first approach is to parametrize it directly using a neural network $f_\Theta(\cdot)$. Hence, we seek Θ such that for most $x \in \mathfrak{N}$: $\mathcal{K}(f_\Theta(x)) \neq \mathcal{K}(x)$. We also require that the perturbed image $f_\Theta(x)$ look similar to the original image x. Hence, $d(x, f_\Theta(x))$ needs to be small for most $x \in \mathfrak{N}$, where $d(\cdot, \cdot)$ is a proper distance function. The second approach is to approximate the difference of natural and adversarial images with a neural network $f_\Theta(\cdot)$. We require that for most $x \in \mathfrak{N}$: $\mathcal{K}(x + f_\Theta(x)) \neq \mathcal{K}(x) \approx c_x$, and the L_p norm of the additive perturbation $\|f_\Theta(x)\|_p$ needs to be small in order for it to be quasi-imperceptible. The second approach gives us better control over the perturbation magnitude. Hence, we will focus on this approach hereafter.

Figure 6.2 shows the architecture for generating image-dependent perturbations. Input image x is passed through the generator to create the perturbation $f_\Theta(x)$. The perturbation is then scaled to constrain its norm. The result is the image-dependent perturbation which is added to the input image. We feed the clipped image $\hat{x}$ to the network to obtain the output probabilities $k(\hat{x})$. We use loss functions similar to the universal case as defined in equations 1–3. At inference time, we can discard the pre-trained model, and use only the generator to produce adversarial examples. This obviates the need for iterative gradient

computations, and allows us to generate perturbations fast.

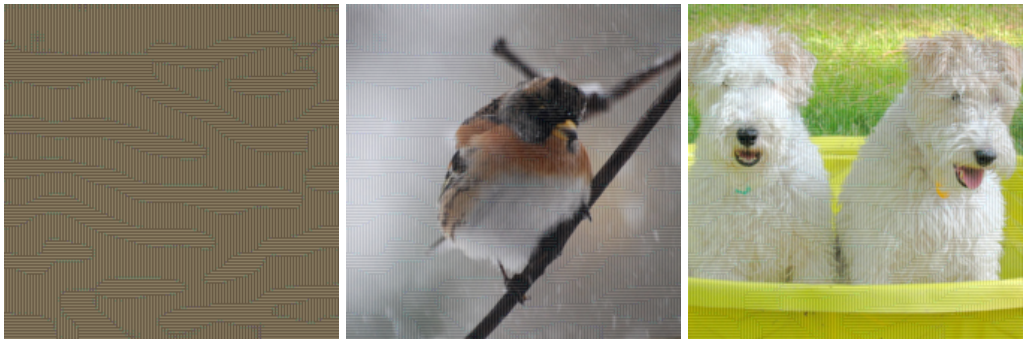
6.3.3 Fooling Multiple Networks

Using generative models for creating adversarial perturbations enables us to train sophisticated models. For instance, we can consider training a single model for misleading multiple networks simultaneously. Suppose we have models $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_m$ trained on natural images. Let $\mathcal{A}_{\mathbf{K}}$ denote the space of adversarial examples for these target models, i.e. $\mathcal{A}_{\mathbf{K}} = \{a \in [0,1]^n \backslash \Re \mid \exists\, x \in \Re : d(x,a) < \epsilon\,, \forall i \in \{1, \ldots, m\} : \mathcal{K}_i(a) \neq \mathcal{K}_i(x) \approx c_x\}$, in which $d(\cdot,\cdot)$ is a distance function, ϵ is a pre-specified threshold and c_x is the ground-truth for x. We can consider both universal and image-dependent perturbations. In the case of universal perturbations, we seek a mapping $\mathcal{F} : [0,1]^n \rightarrow \mathcal{A}_{\mathbf{K}}$ generating adversarial examples from input patterns. In practice, the function is approximated with a deep neural network $\mathcal{F}_\Theta$. Figure 6.3 depicts the corresponding architecture. It is similar to figure 6.1 other than that the resulting perturbed image $\hat{x}$ is fed to each of the pre-trained models. The loss function for training the generator is a linear combination of fooling losses of pre-trained models as defined in equations 1–3. Hence, we have:

$$l_{multi-fool} = \lambda_1 \cdot l_{fool_1} + \cdots + \lambda_m \cdot l_{fool_m} \tag{6.4}$$

in which $\{\lambda_1, \ldots, \lambda_m\} \subset \mathbb{R}$ is a set of weights chosen based on the difficulty of deceiving each target model. The architecture for image-dependent perturbations is similar except that inputs to the generator are natural images.

(a) Perturbation norm: $L_2 = 2000$, target model: VGG-16



(b) Perturbation norm: $L_\infty = 10$, target model: VGG-19

Figure 6.4: Non-targeted universal perturbations. Enhanced universal pattern is shown on the left, and two samples of perturbed images are given on the right.

6.4 Experiments on Classification

We generate adversarial examples for fooling classifiers pre-trained on the ImageNet dataset [63]. For the Euclidean distance as the metric, we scale the output of the generator to have a fixed L_2 norm. We can also scale the generator's output to constrain its maximum value when dealing with the L_∞ norm. All results are reported on the 50,000 images of the ImageNet [63] validation set. Note that the contrast of displayed perturbations is enhanced for better visualization.

[6]Since [185] does not report results on Inception-v3, we compare with their results on Inception-v1 (GoogLeNet).

[6]This result uses the logit-based loss [29, 41] as opposed to the least-likely class loss (equation 6.2), which is used for other results in the table.

		VGG16	VGG19	ResNet152
$L_2 = 2000$	**GAP**	**93.9%**	**94.9%**	79.5%
	UAP	90.3%	84.5%	**88.5%**

Table 6.1: Fooling rates of non-targeted universal perturbations for various classifiers pre-trained on ImageNet. Our method (GAP) is compared with Universal Adversarial Perturbations (UAP) [185] using L_2 norm as the metric.

		VGG16	VGG19	Inception[5]
$L_\infty = 10$	**GAP**	**83.7%**	**80.1%**	**82.7%**[6]
	UAP	78.8%	77.8%	78.9%

Table 6.2: Fooling rates of non-targeted universal perturbations using L_∞ norm as the metric.

6.4.1 Universal Perturbations

Non-targeted Universal Perturbations. This setting corresponds to the architecture in figure 6.1 with the loss functions defined in equations 6.1 and 6.2. Results are given in Tables 6.1 and 6.2 for L_2 and L_∞ norms respectively. For most cases our approach outperforms that of [185]. Similar to [185], a value of 2000 is set as the L_2-norm threshold of the universal perturbation, and a value of 10 is set for the L_∞-norm when images are considered in [0, 255] range[7]. We use U-Net and ResNet Generator for L_2 and L_∞ norms respectively. We visualize the results in figure 6.4. Notice that the L_2 perturbation consists of a bird-like pattern in the top left. Intuitively, the network has learned that in this constrained problem it can successfully fool the classifier for the largest number of images by converging to a bird perturbation. On the other hand, when we optimize the model based on L_∞ norm, it distributes the perturbation to make use of the

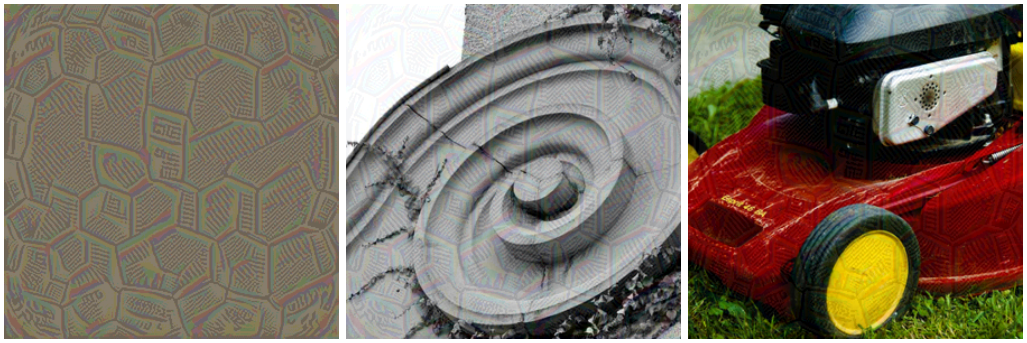
[7]The average L_2 and L_∞ norm of images in our validation set are consistent with those reported in [185].

maximum permissible magnitude at each pixel.

Targeted Universal Perturbations. In this case we seek a single pattern which can be added to any image in the dataset to mislead the model into predicting a specified target label. We perform experiments with fixed L_∞ norm of 10, and use the ResNet generator for fooling the Inception-v3 model. We use the loss function defined in equation 6.3 to train the generator. Figure 6.5 depicts the perturbations for various targets. It also shows the top-1 target accuracy on the validation set, i.e. the ratio of perturbed samples classified as the desired target. We observe the the universal perturbation contains patterns resembling the target class. While this task is more difficult than the non-targeted one, our model achieves high target accuracies. To the best of our knowledge, we are the first to present effective targeted universal perturbations on the ImageNet dataset. To make sure that the model performs well for any target, we train it on 10 randomly sampled classes. The resulting average target accuracy for $L_\infty = 10$ is 52.0%, demonstrating generalizability of the model across different targets.

6.4.2 Image-dependent Perturbations

[41] proposes a strong method for creating targeted image-dependent perturbations. However, its iterative algorithm is very slow at inference time. It reports attacks that take several minutes to run for each image, making it infeasible in real-time scenarios in which the input image changes constantly. FGSM [92] is a fast attack method but is not very accurate. In this work, we present adversarial attacks that are both fast and accurate.



(a) Target: Soccer Ball, Top-1 target accuracy: 74.1%

(b) Target: Knot, Top-1 target accuracy: 63.6%

(c) Target: Finch, Top-1 target accuracy: 61.8%

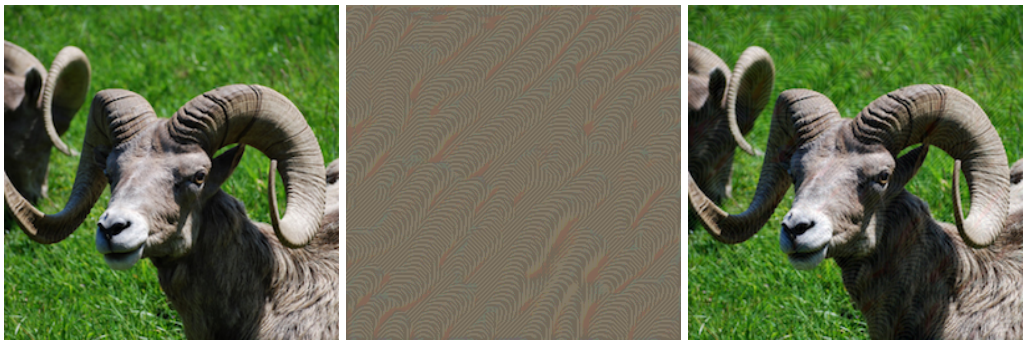
Figure 6.5: Targeted universal perturbations. Three different targets and the corresponding average target accuracy of perturbed images on Inception-v3 are given. Universal pattern is shown on the left and two sample perturbed images are depicted on the right. Perturbation norm is $L_\infty = 10$.

	$L_\infty = 7$	$L_\infty = 10$	$L_\infty = 13$
VGG16	66.9% (30.0%)	80.8% (17.7%)	88.5% (10.6%)
VGG19	68.4% (28.8%)	84.1% (14.6%)	90.7% (8.6%)
Inception-v3	85.3% (13.7%)	98.3% (1.7%)	99.5% (0.5%)

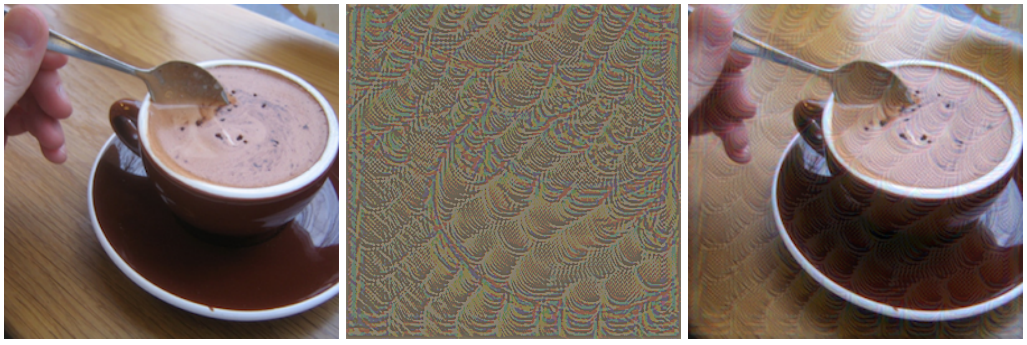
Table 6.3: Fooling ratios (pre-trained models' accuracies) for non-targeted image-dependent perturbations.

Non-targeted Image-dependent Perturbations. The corresponding architecture is given in figure 6.2 with the loss function defined in equations 6.1 and 6.2. We use ResNet generator with 6 blocks for generating the perturbations. Similar to related works on image-dependent perturbations, we focus on L_∞ norm as the metric. Results are shown for various perturbation norms and pre-trained classifiers in Table 6.3. Figure 6.6 illustrates the perturbed images. In this case the model converges to simple patterns which can change the prediction for most images. As we observe, the perturbations contain features from the corresponding input images.

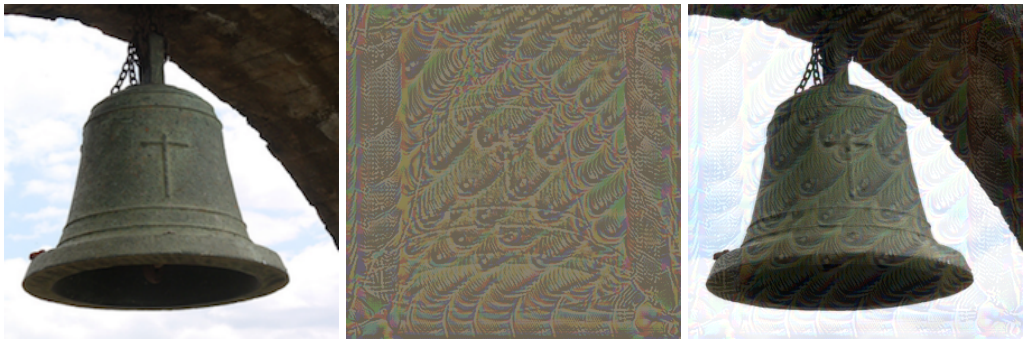
Targeted Image-dependent Perturbations. For this task we use the training scheme shown in figure 6.2 with the loss function in equation 6.3. Figure 6.7 shows samples of perturbed images for fooling the Inception-v3 model. The perturbations are barely perceptible, yet they can obtain high target accuracies. Moreover, the perturbation itself has features resembling the target class and the input image. See figure 6.7 for more examples. We also evaluate performance of the model on 10 randomly sampled classes. The average target accuracy for $L_\infty = 10$ is 89.1%, indicating generalizability of the proposed model across different target classes. The average inference time for generating a perturbation



(a) $L_\infty = 7$



(b) $L_\infty = 10$

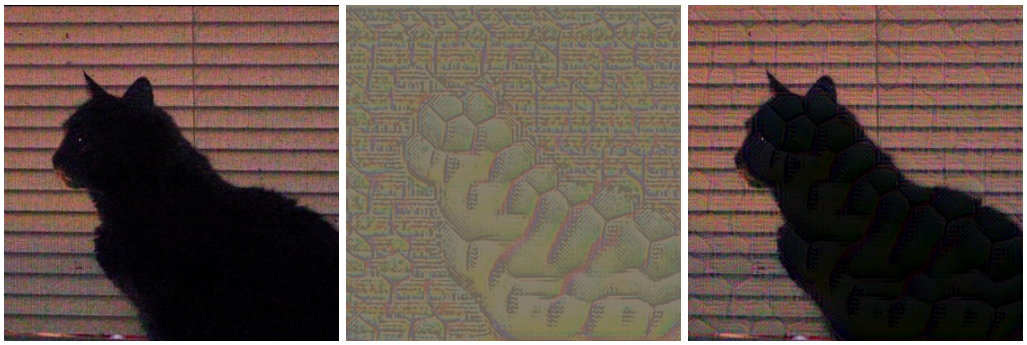


(c) $L_\infty = 13$

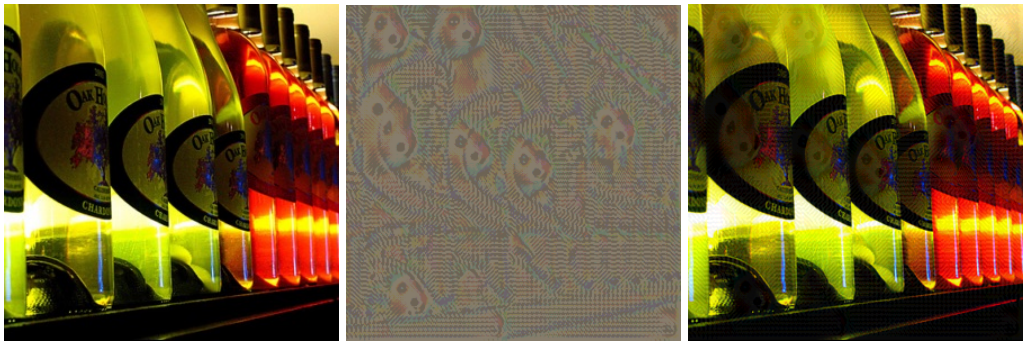
Figure 6.6: Non-targeted image-dependent perturbations. From left to right: original image, enhanced perturbation and perturbed image. Three different thresholds are considered with Inception-v3 as the target model.

to fool the Inception-v3 model is 0.28 *ms* per image, showing that our method is considerably faster than [41][8].

[8]The time is measured on Titan Xp GPUs.



(a) Target: Soccer Ball, Top-1 target accuracy: 91.3%



(b) Target: Hamster, Top-1 target accuracy: 87.4%

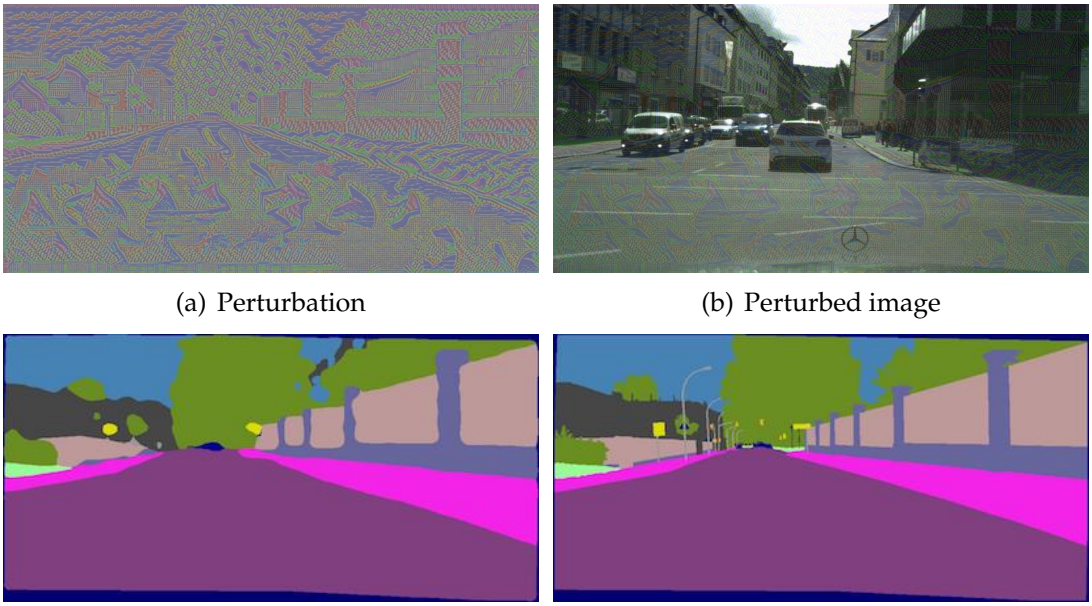
Figure 6.7: Targeted image-dependent perturbations. Two different targets and the corresponding average target accuracy of perturbed images on Inception-v3 are shown. From left to right: original image, enhanced perturbation and perturbed image. Perturbation magnitude is set to $L_\infty = 10$.

6.4.3 Transferability and Fooling Multiple Networks

Several works have demonstrated that adversarial examples generated for one model may also be misclassified by other models. This property is referred to as transferability, and can be leveraged to perform black-box attacks [251, 92, 197, 198, 168, 46, 29]. We show that our generated perturbations can be transferred across different models. Table 6.4 shows the fooling ratio of a non-targeted universal attack trained on one network and evaluated on others. Each row corresponds to the pre-trained model based on which the attack model is learned. The last row of the table corresponds to a model trained to jointly mis-

	VGG16	VGG19	ResNet152
VGG16	**93.9%**	89.6%	52.2%
VGG19	88.0%	**94.9%**	49.0%
ResNet152	31.9%	30.6%	**79.5%**
VGG16 + VGG19	90.5%	90.1%	**54.1%**

Table 6.4: Transferability of non-targeted universal perturbations. The network is trained to fool the pre-trained model shown in each row, and is tested on the model shown in each column. Perturbation magnitude is set to $L_2 = 2000$. The last row indicates joint training on VGG-16 and VGG-19.

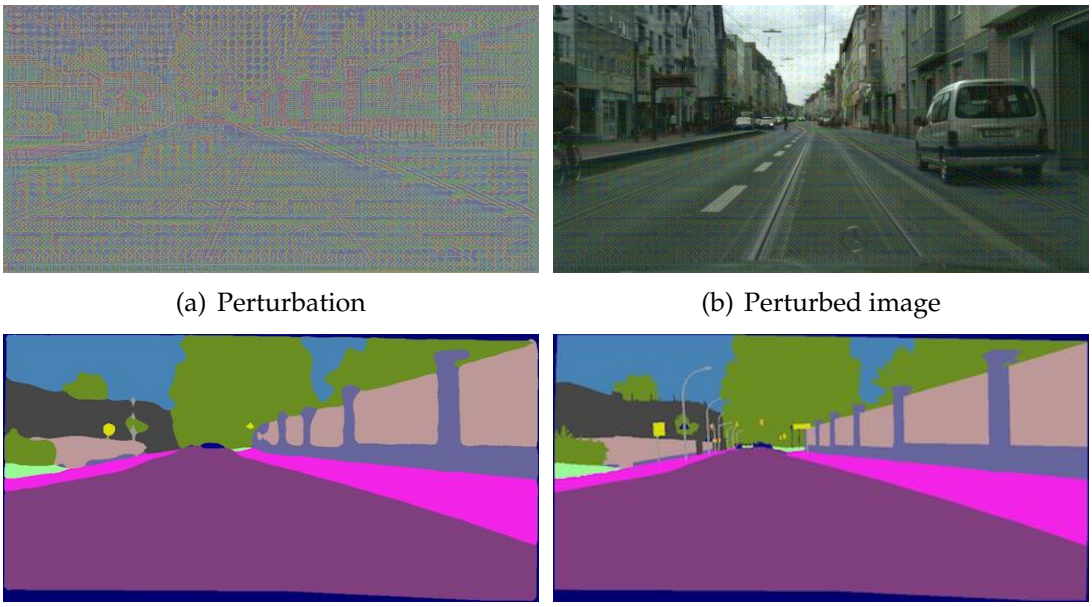


(a) Perturbation (b) Perturbed image

(c) Prediction for perturbed image (d) Target

Figure 6.8: Targeted universal perturbations with $L_\infty = 10$ for fooling the FCN-8s semantic segmentation model.

lead VGG-16 and VGG-19 models based on the architecture depicted in figure 6.3. We see that joint optimization results in better transferability than training on a single target network. This is expected as the network has seen more models during training, so it generalizes better to unseen models.



(a) Perturbation (b) Perturbed image

(c) Prediction for perturbed image (d) Target

Figure 6.9: Targeted image-dependent perturbations with $L_\infty = 10$ for fooling the FCN-8s model.

6.5 Experiments on Semantic Segmentation

Current methods for fooling semantic segmentation models such as [280] and [183] use iterative algorithms, which are hand-engineered for the specific task, and are slow at inference. We demonstrate that our proposed architectures are generalizable across different tasks. More specifically, we show that architectures similar to those used in the classification task yield strong results on fooling segmentation models. We leave extension to tasks other than classification and segmentation as future work. Experiments are performed on the Cityscapes dataset [56]. It contains 2975 training and 500 validation images with a resolution of 2048×1024 pixels. Similar to [183], we downsample images and label maps to 1024×512 pixels using bilinear and nearest-neighbor interpolation respectively.

	$L_\infty = 5$	$L_\infty = 10$	$L_\infty = 20$
GAP (Ours)	79.5%	**92.1%**	**97.2%**
UAP-Seg [183]	**80.3%**	91.0%	96.3%

Table 6.5: Success rate of targeted universal perturbations for fooling the FCN-8s segmentation model. Results are obtained on the validation set of the Cityscapes dataset.

6.5.1 Universal Perturbations

We first consider the more challenging case of targeted attacks in which a desired target label map is given. We use the same setting as in the classification task, i.e. the training architecture in figure 6.1 with the fooling loss defined in equation 6.3. In order for our results to be comparable with [183], we consider FCN-8s [170] as our segmentation model, and use L_∞ norm as the metric. Our setting corresponds to the *static target segmentation* in [183]. We use the same target as the paper, and consider our performance metric to be *success rate*, i.e. the categorical accuracy between the prediction $k(\hat{x})$ and the target t. Table 6.5 demonstrates our results. Our method outperforms the algorithm proposed in [183] for most of the perturbation norms. We also visualize the results in figure 6.8. We observe that the generator fools the segmentation model by creating a universal perturbation which resembles the target label map. We also demonstrates the resulting mean IoU for non-targeted attacks in Table 6.6.

Task	$L_\infty = 5$	$L_\infty = 10$	$L_\infty = 20$
Universal	12.8%	4.0%	2.1%
Image-dependent	6.9%	2.1%	0.4%

Table 6.6: Mean IoU of non-targeted perturbations for fooling the FCN-8s segmentation model on the Cityscapes dataset.

	$L_\infty = 5$	$L_\infty = 10$	$L_\infty = 20$
GAP	87.0%	96.3%	98.2%

Table 6.7: Success rate of targeted image-dependent perturbations for fooling FCN-8s on the Cityscapes dataset.

6.5.2 Image-dependent Perturbations

The targeted image-dependent task corresponds to the architecture in figure 6.2 with the loss function in equation 6.3. We use the same target as the universal case. Results for various norms are given in Table 6.7. As we expect, relaxing the constraint of universality leads to higher success rates. Figure 6.9 illustrates the perturbations for $L_\infty = 10$. By closely inspecting the perturbations, we can observe patterns from both the target and the input image. As shown in Table 6.6, image-dependent perturbations achieve smaller mean IoU by not having the universality constraint. The average inference time per image is 132.82 *ms* for the U-Net architecture and 335.73 *ms* for the ResNet generator[9].

6.6 Discussion and Future Work

We demonstrate the efficacy of generative models for creating adversarial examples. Four types of adversarial attacks are considered: targeted universal, non-targeted universal, targeted image-dependent and non-targeted image-dependent. We achieve high fooling rates on all tasks in the small perturbation norm regime. The perturbations can successfully transfer across different target models. Moreover, we demonstrate that similar architectures can be effec-

[9]The time is measured on Titan Xp GPUs.

tively used for fooling both classification and semantic segmentation models. This eliminates the need for designing task-specific attack methods, and paves the way for extending adversarial examples to other tasks. Future avenues of research include incorporating various properties such as transformation-invariance into the perturbations and extending the proposed framework to tasks other than classification and semantic segmentation.

CHAPTER 7

FINE-GRAINED SYNTHESIS OF UNRESTRICTED ADVERSARIAL EXAMPLES

7.1 Introduction

Adversarial examples, inputs resembling real samples but maliciously crafted to mislead machine learning models, have been studied extensively in the last few years. Most of the existing papers, however, focus on norm-constrained attacks and defenses, in which the adversarial input lies in the ϵ-neighborhood of a real sample using the L_p distance metric (commonly with $p = 0, 2, \infty$). For small ϵ, the adversarial input is quasi-indistinguishable from the natural sample. For an adversarial image to fool the human visual system, it is sufficient to be norm-constrained; but this condition is not necessary. Moreover, defenses tailored for norm-constrained attacks can fail on other subtle input modifications [78]. This has led to a recent surge of interest on unrestricted adversarial attacks in which the adversary is not restricted by a norm threshold. These methods typically hand-craft transformations to capture visual similarity. Spatial transformations [78, 278, 5], viewpoint or pose changes [6], inserting small patches [39], among other methods, have been proposed to generate unrestricted adversarial examples.

In this paper, we focus on fine-grained manipulation of images for unrestricted adversarial attacks. Building upon the Style-GAN model [128] which disentangles fine and coarse-grained variations of images, we manipulate stylistic and stochastic latent variables in order to mislead a classification model. Loss of the target classifier is used to learn subtle variations to create adver-

sarial examples. The pre-trained generative model constrains the search space to natural-looking images. We verify that we do not deviate from the space of realistic images with a user study using Amazon Mechanical Turk. Finally, we demonstrate that our proposed attack can break certified defenses, revealing new vulnerabilities of existing approaches. Adversarial training can be used as an effective defense, and unlike training on norm-bounded adversarial examples, it does not decrease accuracy on clean images. We elaborate on the proposed approach in Section 7.3.

7.2 Related Work

7.2.1 Norm-constrained Adversarial Examples

Most of the existing works on adversarial attacks and defenses focus on norm-constrained adversarial examples: for a given classifier $F : \mathbb{R}^n \to \{1, \ldots, K\}$ and an image $x \in \mathbb{R}^n$, the adversarial image $x' \in \mathbb{R}^n$ is created such that $\|x - x'\|_p < \epsilon$ and $F(x) \neq F(x')$. Common values for p are $0, 2, \infty$, and ϵ is chosen small enough so that the perturbation is imperceptible. Various algorithms have been proposed for creating x' from x. Optimization-based methods solve a surrogate optimization problem based on the classifier's loss and the perturbation norm. In their pioneering paper on adversarial examples, Szegedy *et al.* [251] use box-constrained L-BFGS [81] to minimize the surrogate loss function. Carlini and Wagner [41] propose stronger optimization-based attacks for L_0, L_2 and L_∞ norms using better objective functions and the Adam optimizer [134]. DeepFool is introduced in [187] as a non-targeted attack optimized for the L_2 distance.

It iteratively computes a minimal norm adversarial perturbation for a given image by linearly approximating the decision function. Gradient-based methods use gradient of the classifier's loss with respect to the input image. Fast Gradient Sign Method (FGSM) [92] uses a first-order approximation of the function for faster generation, and is optimized for the L_∞ norm. Projected Gradient Descent (PGD) [176] is an iterative variant of FGSM which provides a strong first-order attack by using multiple steps of gradient ascent and projecting perturbed images to an ϵ-ball centered at the input. Other variants of FGSM are proposed in [68, 143]. Jacobian-based Saliency Map Attack (JSMA) [199] is a greedy algorithm that modifies pixels one at a time. It uses the gradients to compute a saliency map, picks the most important pixel and modifies it to increase likelihood of the target class. Li *et al.* [156] introduce a gradient transformer module to generate regionally homogeneous perturbations. They claim state-of-the-art attack results, which are independent of input images and can be transferred to black-box models. Generative attack methods [22, 205, 277] use an auxiliary network to learn adversarial perturbations, which provides benefits such as faster inference and more diversity in the synthesized images.

Several methods have been proposed for defending against adversarial attacks. These approaches can be broadly categorized to *empirical* defenses which are empirically robust to adversarial examples, and *certified* defenses which are provably robust to a certain class of attacks. One of the most successful empirical defenses is *adversarial training* [92, 143, 176] which augments training data with adversarial examples generated as the training progresses. Adversarial logit pairing [125] is a form of adversarial training which constrains logit predictions of a clean image and its adversarial counterpart to be similar. Many empirical defenses attempt to defeat adversaries using a form of

input pre-processing or by manipulating intermediate features or gradients [155, 101, 279, 230, 158, 281]. Few approaches have been able to scale up to high-resolution datasets such as ImageNet [279, 158, 281, 209, 125]. Most of the proposed heuristic defenses were later broken by stronger adversaries [41, 258, 16]. Athalye *et al.* [16] show that many of these defenses fail due to an issue they term *obfuscated gradients*, which occurs when the defense method is designed to mask information about the model's gradients. They propose workarounds to obtain approximate gradients for adversarial attacks. Vulnerabilities of empirical defenses have led to increased interest in certified defenses, which provide a guarantee that the classifier's prediction is constant within a neighborhood of the input. Several certified defenses have been proposed [275, 214, 74, 257] which typically do not scale to ImageNet. Cohen *et al.* [55] use randomized smoothing with Gaussian noise to obtain provably L_2-robust classifiers on ImageNet.

7.2.2 Unrestricted Adversarial Examples

For an image to be adversarial, it needs to be visually indistinguishable from real images. One way to achieve this is by applying subtle geometric transformations to the input image. Spatially transformed adversarial examples are introduced in [278] in which a flow field is learned to displace pixels of the image. They use sum of spatial movement distance for adjacent pixels as a regularization loss to minimize the local distortion introduced by the flow field. Similarly, Alaifari *et al.* [5] iteratively apply small deformations, found through a gradient descent step, to the input in order to obtain the adversarial image. Engstrom *et al.* [78] show that simple translations and rotations are enough for fooling

deep neural networks. This remains to be the case even when the model has been trained using appropriate data augmentation. Alcorn *et al.* [6] manipulate pose of an object to fool deep neural networks. They estimate parameters of a 3D renderer that cause the target model to misbehave in response to the rendered image. Another approach for evading the norm constraint is to insert new objects in the image. Adversarial Patch [39] creates an adversarial image by completely replacing part of an image with a synthetic patch. The patch is image-agnostic, robust to transformations, and can be printed and used in real-world settings. Song *et al.* [244] search in the latent (z) space of AC-GAN [194] to find generated images that can fool a classifier, and show results on MNIST [150], SVHN [189] and CelebA [169] datasets. Since the z space is not interpretable, their method has no control over the generation process. On the other hand, our method can manipulate real or synthesized images in a fine-grained, controllable manner. Existence of on-the-manifold adversarial examples is also shown in [89], which considers the task of classifying between two concentric n-dimensional spheres. A challenge for creating unrestricted adversarial examples and defending against them is introduced in [38] using the simple task of classifying between birds and bicycles.

7.2.3 Fine-grained Image Generation

With recent improvements in generative models, they are able to generate high-resolution and realistic images. Moreover, these models can be used to learn and disentangle various latent factors for image synthesis. Style-GAN is proposed in [128] which disentangles high-level attributes and stochastic variations of generated images in an unsupervised manner. The model learns an intermediate la-

tent space from the input latent code, which is used to adjust style of the image. It also injects noise at each level of the generator to capture stochastic variations. Singh *et al.* introduce Fine-GAN [239], a generative model which disentangles the background, object shape, and object appearance to hierarchically generate images of fine-grained object categories. Layered Recursive GAN is proposed in [286], and generates image background and foreground separately and recursively without direct supervision. Conditional fine-grained generation has been explored in several papers. Bao *et al.* [23] introduce a Conditional VAE-GAN for synthesizing images in fine-grained categories. Modeling an image as a composition of label and latent attributes, they vary the fine-grained category label fed into the generative model, and randomly draw values of a latent attribute vector. AttnGAN [282] uses attention-driven, multi-stage refinement for fine-grained text-to-image generation. It synthesizes fine-grained details at different sub-regions of the image by paying attention to the relevant words in the natural language description. Hong *et al.* [113] present a hierarchical framework for semantic image manipulation. Their model first learns to generate the pixel-wise semantic label maps from the initial object bounding boxes, and then learns to generate the manipulated image from the predicted label maps. A multi-attribute conditional GAN is proposed in [267], and can generate fine-grained face images based on the specified attributes.

7.3 Approach

Most of the existing works on unrestricted adversarial attacks rely on geometric transformations and deformations [78, 278, 5], which are oblivious to latent factors of variation. In this paper, we leverage disentangled latent representa-

tions of images for unrestricted adversarial attacks. Style-GAN [128] is a state-of-the-art generative model which learns to disentangle high-level attributes and stochastic variations in an unsupervised manner. More specifically, stylistic variations are represented by *style* variables and stochastic details are captured by *noise* variables. Changing the noise only affects low-level details, leaving the overall composition and high-level aspects such as identity intact. This allows us to manipulate the noise variables such that variations are barely noticeable by the human eye, yet the synthesized image can fool a pre-trained classifier. The style variables affect higher level aspects of image generation. For instance, when the model is trained on bedrooms, style variables from the top layers control viewpoint of the camera, middle layers select the particular furniture, and bottom layers deal with colors and details of materials. This allows us to manipulate images in a controlled manner, providing an avenue for fine-grained unrestricted attacks.

Formally, we can represent Style-GAN with a non-linear mapping function f and a synthesis network g. The mapping function is an 8-layer MLP which takes a latent code $\mathbf{z}$, and produces an intermediate latent vector $\mathbf{w} = f(\mathbf{z})$. This vector is then specialized by learned affine transformations A to styles $\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_b)$. Style variables in turn control adaptive instance normalization operations [115] after each convolutional layer of the synthesis network g. Noise inputs are single-channel images consisting of un-correlated Gaussian noise that are fed to each layer of the synthesis network. Learned per-feature scaling factors B are then used to generate noise variables $\boldsymbol{\eta}$ which are added to the output of convolutional layers. The synthesis network takes style $\mathbf{y}$ and noise $\boldsymbol{\eta}$ as input, and generates an image $\mathbf{x} = g(\mathbf{y}, \boldsymbol{\eta})$. We then pass the generated image to a pre-trained classifier F. We seek to slightly modify $\mathbf{x}$ so that F can no longer classify

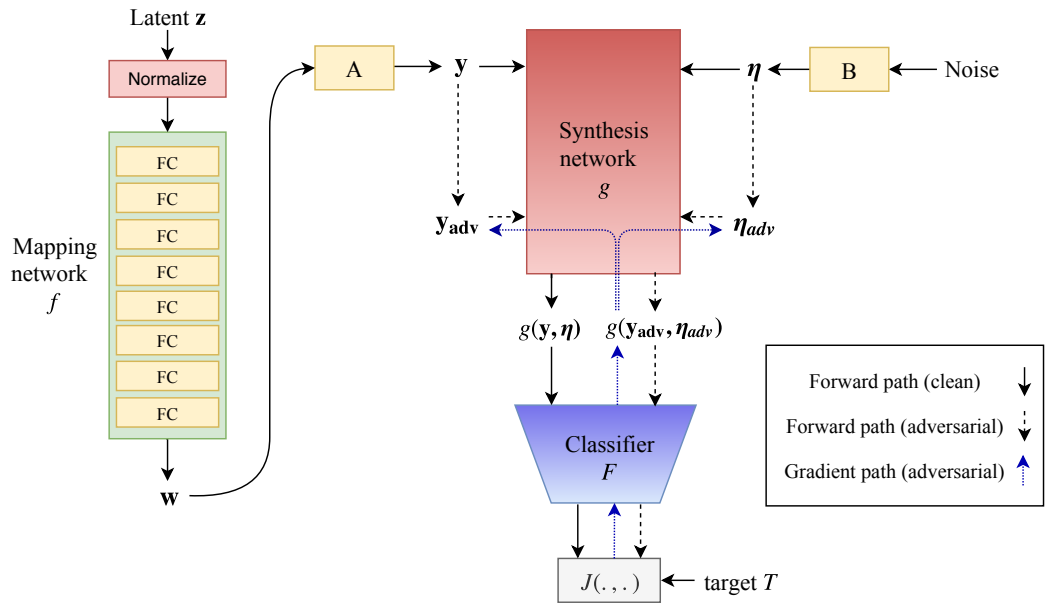


Figure 7.1: Model architecture. Style ($\mathbf{y}$) and noise ($\boldsymbol{\eta}$) variables are used to generate images $g(\mathbf{y}, \boldsymbol{\eta})$ which are fed to the classifier F. Adversarial style and noise tensors are initialized with $\mathbf{y}$ and $\boldsymbol{\eta}$ and iteratively updated using gradients of the loss function J.

it correctly. We achieve this through perturbing the style and noise tensors, which control different aspects of image generation in a fine-grained manner. More specifically, we initialize adversarial style and noise variables as $\mathbf{y}_{\mathbf{adv}}^{(\mathbf{0})} = \mathbf{y}$ and $\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{0})} = \boldsymbol{\eta}$ respectively. These adversarial tensors are then iteratively updated in order to fool the classifier. Loss of the classifier determines the update rule, which in turn depends on the type of attack. As common in the literature, we consider two types of attacks: non-targeted and targeted.

7.3.1 Non-targeted Attacks

In order to generate non-targeted adversarial examples, we need to change the model's original prediction. Starting from initial values $\mathbf{y}_{\mathbf{adv}}^{(\mathbf{0})} = \mathbf{y}$ and $\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{0})} = \boldsymbol{\eta}$,

we perform gradient ascent in the style and noise spaces of the generator to find values that maximize the classifier's loss. At time step t, the update rule for the style and noise variables is:

$$\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t+1})} = \mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}} J(F(g(\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}, \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})})), c_{\mathbf{x}})) \tag{7.1}$$

$$\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t+1})} = \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})} + \delta \cdot \text{sign}(\nabla_{\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})}} J(F(g(\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}, \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})})), c_{\mathbf{x}})) \tag{7.2}$$

in which $J(\cdot, \cdot)$ represents the classifier's loss function (e.g. cross-entropy), $c_{\mathbf{x}}$ is the ground-truth class for $\mathbf{x} = g(\mathbf{y}, \boldsymbol{\eta})$, and $\epsilon, \delta \in \mathbb{R}$ are step sizes. Note that $F(\cdot)$ gives the probability distribution over classes. This formulation resembles Iterative-FGSM [143]; however, the gradients are computed with respect to the noise and style variables of the synthesis network. Alternatively, as proposed in [143] we can use the least-likely predicted class $ll_{\mathbf{x}} = \arg\min(F(\mathbf{x}))$ as our target:

$$\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t+1})} = \mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})} - \epsilon \cdot \text{sign}(\nabla_{\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}} J(F(g(\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}, \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})})), ll_{\mathbf{x}})) \tag{7.3}$$

$$\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t+1})} = \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})} - \delta \cdot \text{sign}(\nabla_{\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})}} J(F(g(\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}, \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})})), ll_{\mathbf{x}})) \tag{7.4}$$

We found the latter approach more effective in practice. We use $(\epsilon, \delta) = (0.004, 0.2)$ and $(0.004, 0.1)$ in the experiments on LSUN and CelebA-HQ respectively. We perform multiple steps of gradient descent (usually 2 to 10) until the classifier is fooled. Unlike I-FGSM that generates high-frequency noisy perturbations in the pixel space, our pre-trained generative model constrains the space of generated images to realistic ones.

7.3.2 Targeted Attacks

Generating targeted adversarial examples is more challenging as we need to change the prediction to a specific class T. In this case, we perform gradient

descent to minimize the classifier's loss with respect to the target:

$$\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t+1})} = \mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})} - \epsilon \cdot \text{sign}(\nabla_{\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}} J(F(g(\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}, \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})})), T)) \tag{7.5}$$

$$\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t+1})} = \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})} - \delta \cdot \text{sign}(\nabla_{\boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})}} J(F(g(\mathbf{y}_{\mathbf{adv}}^{(\mathbf{t})}, \boldsymbol{\eta}_{\mathbf{adv}}^{(\mathbf{t})})), T)) \tag{7.6}$$

We use $(\epsilon, \delta) = (0.005, 0.2)$ and $(0.004, 0.1)$ in the experiments on LSUN and CelebA-HQ respectively. In practice 3 to 15 updates suffice to fool the classifier. Updating only the noise tensor results in finer adversarial changes, while only using the style variable creates coarser stylistic changes. We can also have a detailed control over the generation process by manipulating specific layers of the synthesis network. Note that we only control deviation from the initial latent variables, and do not impose any norm constraint on generated images.

7.3.3 Input-conditioned Generation

Generation can also be conditioned on real input images by embedding them into the latent space of Style-GAN. We first synthesize images similar to the given input image I by optimizing values of $\mathbf{y}$ and $\boldsymbol{\eta}$ such that $g(\mathbf{y}, \boldsymbol{\eta})$ is close to I. More specifically, we minimize the perceptual distance [123] between $g(\mathbf{y}, \boldsymbol{\eta})$ and I. We can then proceed similar to equations 3–6 to perturb these tensors and generate the adversarial image. Realism of synthesized images depends on inference properties of the generative model. In practice, generated images resemble input images with high fidelity especially for CelebA-HQ images.

7.4 Results and Discussion

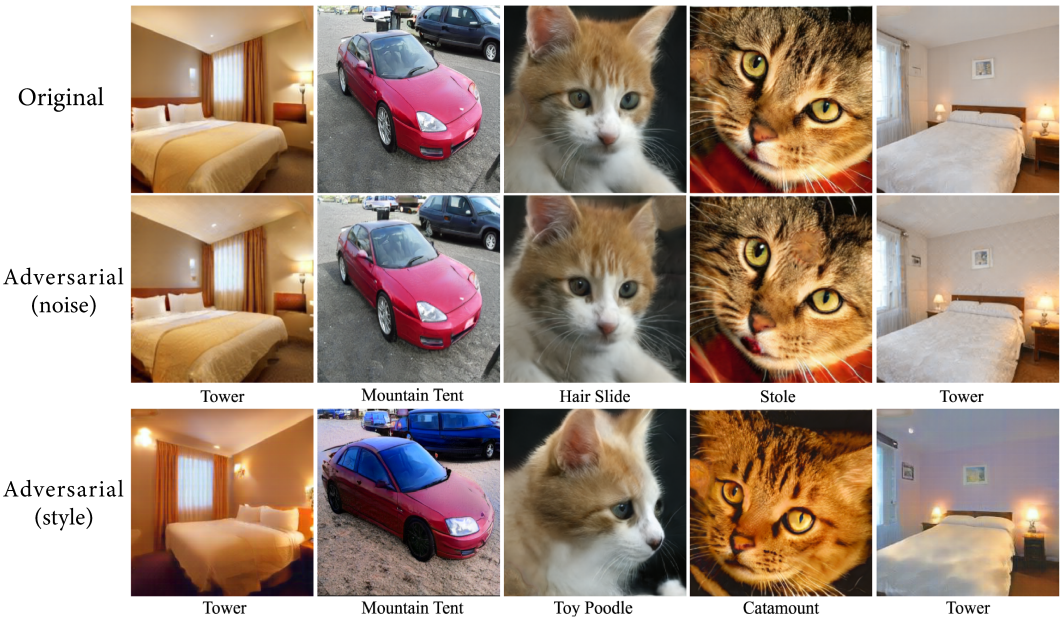
We provide qualitative and quantitative results demonstrating our proposed approach. Experiments are performed on LSUN [290] and CelebA-HQ [126]. LSUN contains 10 scene categories each with around one million labeled images and 20 object categories. We use all the 10 scene classes as well as two object classes: *cars* and *cats*. We consider this dataset since it is used in Style-GAN, and is well suited for a classification task. For the scene categories, a 10-way classifier is trained based on Inception-v3 [250] which achieves an accuracy of 87.7% on LSUN's test set. The two object classes also appear in ImageNet [63], a richer dataset containing 1000 categories. Therefore, for experiments on cars and cats we use an Inception-v3 model trained on ImageNet. This allows us to explore a broader set of categories in our attacks, and is particularly helpful for targeted adversarial examples. Note that there are multiple classes representing cars and cats in ImageNet, so we identify and group those classes. CelebA-HQ is a high-quality version of the CelebA dataset [169] consisting of 30,000 face images at 1024×1024 resolution. We consider the gender classification task, and use the classifier provided by Karras *et al.* [128]. This is a binary task for which targeted and non-targeted attacks are similar.

In order to synthesize a variety of adversarial examples, we use different random seeds in Style-GAN to obtain various values for $\mathbf{z}, \mathbf{w}, \mathbf{y}$ and $\boldsymbol{\eta}$. Style-based adversarial examples are generated by initializing $\mathbf{y}_{\mathbf{adv}}$ with the value of $\mathbf{y}$, and iteratively updating it as in equation 7.3 (or 7.5) until the resulting image $g(\mathbf{y}_{\mathbf{adv}}, \boldsymbol{\eta})$ fools the classifier F. Noise-based adversarial examples are created similarly using $\boldsymbol{\eta}_{\mathbf{adv}}$ and the update rule in equation 7.4 (or 7.6). We can also combine the effect of style and noise by simultaneously updating $\mathbf{y}_{\mathbf{adv}}$ and $\boldsymbol{\eta}_{\mathbf{adv}}$ in

each iteration, and feeding $g(\mathbf{y}_{\mathbf{adv}}, \boldsymbol{\eta}_{\mathbf{adv}})$ to the classifier. In this case, the effect of style usually dominates since it creates coarser modifications. To make sure the iterative process always converges in reasonable number of steps, we measure the number of updates required to fool the classifier on 1000 randomly-selected images. In the case of non-targeted attacks on LSUN, 3.7 ± 1.8 and 6.8 ± 3.6 (mean ± std) updates are required for noise-based and style-based examples respectively. For targeted attacks, we first randomly sample a target class different from the ground-truth label for each image. In this case, the number of updates required for noise-based and style-based attacks are 4.5 ± 1.7 and 9.1 ± 4.2 respectively. For the CelebA-HQ dataset, 6.2 ± 4.1 and 7.3 ± 3.0 updates are needed for noise and style manipulation respectively. While using different step sizes makes a fair comparison difficult, we generally found it easier to fool the model by manipulating the noise.

Figure 7.2 illustrates generated adversarial examples for non-targeted and targeted attacks on LSUN. Original image $g(\mathbf{y}, \boldsymbol{\eta})$, noise-based image $g(\mathbf{y}, \boldsymbol{\eta}_{\mathbf{adv}})$ and style-based image $g(\mathbf{y}_{\mathbf{adv}}, \boldsymbol{\eta})$ are shown. As we observe, adversarial images look almost indistinguishable from natural images. This also holds in targeted attacks even when original and target classes are very dissimilar. Manipulating the noise variable results in very subtle, imperceptible changes. Varying the style leads to coarser changes such as different colorization, pose changes, and even removing or inserting new objects in the scene. We can also control granularity of changes by selecting specific layers of the model. Manipulating top layers, corresponding to coarse spatial resolutions, results in high-level changes. Lower layers, on the other hand, modify finer details. In the first two columns of Figure 7.2, we only modify top 6 layers (out of 18) to generate adversarial images. The middle column corresponds to changing layers 7 to 12, and the last

(a) Non-targeted



(b) Targeted

Figure 7.2: Unrestricted adversarial examples on LSUN for a) non-targeted and b) targeted attacks. Predicted classes are shown under each image. First two columns correspond to manipulating top 6 layers of the synthesis network. The middle column manipulates layers 7 to 12, and the last two columns correspond to the bottom 6 layers.

Figure 7.3: Unrestricted adversarial examples on CelebA-HQ gender classification. From top to bottom: original, noise-based and style-based adversarial images. Males are classified as females and vice versa. First two columns correspond to manipulating top 6 layers of the synthesis network. The middle three columns manipulate layers 7 to 12, and the last two columns correspond to the bottom 6 layers.

Figure 7.4: Input-conditioned adversarial examples on CelebA-HQ gender classification. From top to bottom: input, generated and style-based images. Males are classified as females and vice versa.

two columns use the bottom 6 layers.

Figure 7.3 depicts adversarial examples on CelebA-HQ gender classification. Males are classified as females and vice versa. As we observe, various facial features are altered by the model yet the identity is preserved. Similar to LSUN images, noise-based changes are more subtle than style-based ones, and we observe a spectrum of high-level, mid-level and low-level changes. Figure 7.4 illustrates adversarial examples conditioned on real input images using the procedure described in Section 7.3.3. Synthesized images resemble inputs with high fidelity, and set the initial values in our optimization process. In some cases, we can notice how the model is altering masculine or feminine features. For instance, women's faces become more masculine in columns 2 and 4, and men's beard is removed in column 3 of Figure 7.3 and column 1 of Figure 7.4.

Unlike perturbation-based attacks, L_p distances between original and adversarial images are large, yet they are visually similar. Moreover, we do not observe high-frequency perturbations in the generated images. The model learns to modify the initial input without leaving the manifold of realistic images. Note that the classifiers are trained on millions of images, yet they are easily fooled by these subtle on-the-manifold changes. This poses serious concerns about robustness of deep neural networks, and reveals new vulnerabilities of them. Additional examples and higher-resolution images are provided in the supplementary material.

7.4.1 User Study

Norm-constrained attacks provide visual realism by L_p proximity to a real input. To verify that our unrestricted adversarial examples are realistic and correctly classified by an oracle, we perform human evaluation using Amazon Mechanical Turk. In the first experiment, each adversarial image is assigned to three workers, and their majority vote is considered as the label. The user interface for each worker contains nine images, and shows possible labels to choose from. We also include the label "Other" for images that workers think do not belong to any specific class. We use 2400 noise-based and 2400 style-based adversarial images from the LSUN dataset, containing 200 samples from each class (10 scene classes and 2 object classes). The results indicate that 99.2% of workers' majority votes match the ground-truth labels. This number is 98.7% for style-based adversarial examples and 99.7% for noise-based ones. As we observe in Figure 7.2, noise-based examples do not deviate much from the original image, resulting in easier prediction by a human observer. On the other hand, style-based images show coarser changes, which in a few cases result in unrecognizable images or false predictions by the workers.

We use a similar setup in the second experiment but for classifying real versus fake (generated). We also include 2400 real images as well as 2400 unperturbed images generated by Style-GAN. 74.7% of unperturbed images are labeled by workers as real. This number is 74.3% for noise-based adversarial examples and 70.8% for style-based ones, indicating less than 4% drop compared with unperturbed images generated by Style-GAN.

7.4.2 Evaluation on Certified Defenses

Several approaches have been proposed in the literature to defend against adversarial examples, which can be broadly divided into *empirical* and *certified* defenses. Empirical defenses are heuristic methods designed to mitigate effects of perturbations, and certified defenses provide provable guarantees on model's robustness. Almost all of these methods consider norm-constrained attacks. Most of the empirical defenses were later broken by stronger adversaries [40, 16]. This has led to a surge of interest in provable defenses. However, most certified defenses are not scalable to high-resolution datasets. Cohen *et al.* [55] propose the first certified defense at the scale of ImageNet. Using randomized smoothing with Gaussian noise, their defense guarantees a certain top-1 accuracy for perturbations with L_2 norm less than a specific threshold.

We demonstrate that our unrestricted attacks can break the state-of-the-art certified defense on ImageNet. We use 400 noise-based and 400 style-based adversarial images from the object categories of LSUN, and group all relevant ImageNet classes as the ground-truth. Our adversarial examples are evaluated against a randomized smoothing classifier based on ResNet-50 using Gaussian noise with standard deviation of 0.5 [55]. Table 7.1 shows accuracy of the model on clean and adversarial images. As we observe, the accuracy drops on adversarial inputs, and the certified defense is not effective against our attack. Note that we stop updating adversarial images as soon as the model is fooled. If we keep updating for more iterations afterwards, we can achieve even stronger attacks. Our adversarial examples are learned on Inception-v3, yet they can fool a defended model based on ResNet-50. This indicates that these inputs are transferable to other models, showing their potential for black-box attacks.

Considering the variety of methods used for creating unrestricted adversarial examples, designing effective defenses against them is a challenging task. We believe this can be an interesting direction for future research.

	Accuracy
Clean	63.1%
Adversarial (style)	21.7%
Adversarial (noise)	37.8%

Table 7.1: Accuracy of a certified classifier equipped with randomized smoothing on adversarial images.

7.4.3 Adversarial Training

Adversarial training increases robustness of models by injecting adversarial examples into training data [92, 176, 143]. This approach makes the classifier robust to perturbations similar to those used in training; however, it can still be vulnerable to black-box adversarial inputs transferred from other models [256]. To mitigate this issue, Ensemble Adversarial Training is proposed in [256] to augment training data with perturbations transferred from other pre-trained models. The main drawback of adversarial training is that it degrades performance of the classifier on clean images [176]. Various regularizers have been proposed to tackle this issue [292, 263].

We show that while adversarial training makes the model robust to our unrestricted adversarial inputs, it does not degrade accuracy on clean images. We perform adversarial training by incorporating generated images in training the LSUN classifier. 400k clean images as well as 50k noise-based and 50k style-based adversarial inputs are used to train the classifier. Same number of

samples are used across all scene categories. Table 7.2 shows accuracy of the strengthened and original classifiers on clean and adversarial test images. Similar to norm-constrained perturbations, adversarial training is an effective defense against our unrestricted attack. However, accuracy of the model on clean test images remains almost the same after adversarial training. This is in contrast to training with norm-bounded adversarial inputs, which hurts classifier's performance on clean images. This is due to the fact that unlike perturbation-based inputs, our generated images live on the manifold of realistic images as constrained by the generative model.

	Adv. Trained	Original
Clean	87.6%	87.7%
Adversarial (noise)	81.2%	0.0%
Adversarial (style)	76.9%	0.0%

Table 7.2: Accuracy of adversarially trained and original classifiers on clean and adversarial test images.

7.5 Conclusion and Future Work

We present a novel approach for creating unrestricted adversarial examples leveraging state-of-the-art generative models. Unlike existing works that rely on hand-crafted transformations, we learn stylistic and stochastic changes to mislead pre-trained models. Loss of the target classifier is used to perform gradient descent in the style and noise spaces of Style-GAN. Subtle adversarial changes can be crafted using noise variables, and coarser modifications can be created through style variables. We demonstrate results in both targeted and non-targeted cases, and validate visual realism of synthesized images through

human evaluation. We show that our attacks can break state-of-the-art defenses, revealing vulnerabilities of current norm-constrained defenses to unrestricted attacks. Moreover, while adversarial training can be used to make models robust against our adversarial inputs, it does not degrade accuracy on clean images.

The area of unrestricted adversarial examples is relatively under-explored. Not being bounded by a norm threshold provides its own pros and cons. It allows us to create a diverse set of attack mechanisms; however, fair comparison of relative strength of these attacks is challenging. It is also unclear how to even define provable defenses. While several papers have attempted to interpret norm-constrained attacks in terms of decision boundaries, there has been less effort in understanding the underlying reasons for models' vulnerabilities to unrestricted attacks. We believe these can be promising directions for future research. We also plan to further explore transferability of our approach for black-box attacks in the future.

CHAPTER 8

CONCLUSIONS

In this dissertation, we studied how generative models can benefit from hybrid representations in which a stronger representation guides a weaker one. We showed results on 3D synthesis conditioned on images or point clouds, label-conditioned 2D generation, 2.5D motion generation and few-shot synthesis. We also discussed how visual realism can be exploited to create adversarial examples to fool humans and machines.

Building realistic 3D generative models is a challenging task. 3D scenes contain multiple objects as well as texture information, which are difficult to learn from limited input data. Extending current 3D generative models from object-level to scene-level is a promising future research direction. Geometric priors and 2D-3D consistencies can be used to facilitate learning.

Few-shot generative models can be extended to the 3D domain. Current 3D generative models are trained on datasets such as ShapeNet which include a limited number of object classes, and these models do not generalize beyond these categories. Building 3D generative models which adapt to new classes from few examples is a venue for future research. Various techniques from computer graphics can be used to deform and augment the current set of examples for more efficient learning.

While adversarial examples are mainly considered as a threat to machine learning models, they can be harnessed as a form of data augmentation to improve the model's performance. This can be achieved by proper adversarial training which computes different statistics for real and adversarial samples.

This idea can be applied to both 2D and 3D computer vision models to improve their performance and generalization.

Deep neural networks can also be applied to several other problems. In [203] we propose a self-supervised learning framework on point clouds. We use the auxiliary task of rotation estimation to learn features that are useful for downstream tasks such as shape classification and keypoint prediction. In [208] we demonstrate how to estimate fundamental matrices from a pair of stereo images. We use a neural network comprised of a feature extractor and a regressor. Our model achieves competitive performance with traditional methods without the need for extracting keypoint correspondences. In [20] we demonstrate that in the neural networks trained using differentially private stochastic gradient descent (DP-SGD), accuracy drop is not borne equally: accuracy of DP models drops much more for the underrepresented classes and subgroups. Finally, in [207] we explore the impact of visual characteristics on real estate valuation. Using deep neural networks on a large dataset of photos of home interiors and exteriors, we develop a novel framework for automated value assessment using the photos in addition to other home characteristics.

BIBLIOGRAPHY

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative models for 3d point clouds. *ICML*, 2018.

[2] Adobe. Adobe after effects, 2019.

[3] Adobe. Adobe character animator, 2019.

[4] Naveed Akhtar, Jian Liu, and Ajmal Mian. Defense against universal adversarial perturbations. *arXiv preprint arXiv:1711.05929*, 2017.

[5] Rima Alaifari, Giovanni S Alberti, and Tandri Gauksson. Adef: An iterative algorithm to construct adversarial deformations. *arXiv preprint arXiv:1804.07729*, 2018.

[6] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. *arXiv preprint arXiv:1811.11553*, 2018.

[7] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM transactions on graphics (TOG)*, volume 22, pages 587–594. ACM, 2003.

[8] Brett Allen, Brian Curless, Zoran Popović, and Aaron Hertzmann. Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 147–156. Eurographics Association, 2006.

[9] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[10] Yali Amit, Ulf Grenander, and Mauro Piccioni. Structural image restoration through deformable templates. *Journal of the American Statistical Association*, 86(414):376–387, 1991.

[11] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[12] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[13] Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. *arXiv preprint arXiv:1711.09856*, 2017.

[14] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 224–232. JMLR. org, 2017.

[15] Sanjeev Arora, Andrej Risteski, and Yi Zhang. Do gans learn the distribution? some theory and empirics. 2018.

[16] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[17] Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

[18] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[19] Timur Bagautdinov, Chenglei Wu, Jason Saragih, Pascal Fua, and Yaser Sheikh. Modeling facial geometry using compositional vaes. *In practice*, 1:1, 2018.

[20] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*, pages 15453–15462, 2019.

[21] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[22] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *AAAI*, 2018.

[23] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2745–2754, 2017.

[24] Sergey Bartunov and Dmitry Vetrov. Few-shot generative modelling with generative matching networks. In *International Conference on Artificial Intelligence and Statistics*, pages 670–678, 2018.

[25] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *Proceedings of the International Conference Computer Vision (ICCV)*, 2019.

[26] William Baxter, Pascal Barla, and Ken Anjyo. N-way morphing for 2d animation. *Computer Animation and Virtual Worlds*, 20(2-3):79–87, 2009.

[27] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *SIGGRAPH Asia*, 2018.

[28] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018.

[29] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491*, 2017.

[30] Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003.

[31] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

[32] Florian Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. *ICLR submissions*, 2017.

[33] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[34] Alican Bozkurt, Babak Esmaeili, Dana H Brooks, Jennifer G Dy, and Jan-Willem van de Meent. Can vaes generate novel examples? *arXiv preprint arXiv:1812.09624*, 2018.

[35] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. Turning to the masters: Motion capturing cartoons. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 399–407, 2002.

[36] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR*, abs/1608.04236, 2016.

[37] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[38] Tom B Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Christiano, and Ian Goodfellow. Unrestricted adversarial examples. *arXiv preprint arXiv:1809.08352*, 2018.

[39] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[40] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[41] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[42] Edwin Catmull. The problems of computer-assisted animation. In *ACM SIGGRAPH Computer Graphics*, volume 12, pages 348–353. ACM, 1978.

[43] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song,

Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[44] Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. A simple geometric model for elastic deformations. In *ACM transactions on graphics (TOG)*, volume 29, page 38. ACM, 2010.

[45] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[46] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

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

[48] Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv*, 2016.

[49] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016.

[51] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[52] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.

[53] Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199*, 2019.

[54] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[55] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[56] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[57] Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture mapping for cel animation. pages 435–446, July 1998.

[58] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35, 10 2017.

[59] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[60] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[61] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *ICLR submissions*, 2017.

[62] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. *arXiv preprint arXiv:1802.06816*, 2018.

[63] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Im-

agenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[64] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[65] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

[66] Fabian Di Fiore, Philip Schaeken, Koen Elens, and Frank Van Reeth. Automatic in-betweening in computer assisted animation by exploiting 2.5 d modelling techniques. In *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No. 01TH8596)*, pages 192–200. IEEE, 2001.

[67] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *ICLR submissions*, 2017.

[68] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018.

[69] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016.

[70] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *CVPR*, 2016.

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

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

[73] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *ICLR submissions*, 2017.

[74] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *arXiv preprint arXiv:1803.06567*, 104, 2018.

[75] Marek Dvorožňák, Pierre Bénard, Pascal Barla, Oliver Wang, and Daniel Sỳkora. Example-based expressive animation of 2d rigid bodies. *ACM Transactions on Graphics (TOG)*, 36(4):127, 2017.

[76] Marek Dvorožnák, Wilmot Li, Vladimir G Kim, and Daniel Sỳkora. Toonsynth: example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics (TOG)*, 37(4):167, 2018.

[77] Harrison Edwards and Amos Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

[78] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[79] Xinyi Fan, Amit Bermano, Vladimir G. Kim, Jovan Popovic, and Szymon Rusinkiewicz. Tooncap: A layered deformable model for capturing poses from cartoon characters. *Expressive*, 2018.

[80] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[81] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[82] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016.

[83] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.

[84] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv*, 2015.

[85] Jon Gauthier. Conditional generative adversarial nets for convolutional

face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014, 2014.

[86] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: masked autoencoder for distribution estimation. In *ICML*, 2015.

[87] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8513–8521, 2018.

[88] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[89] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[90] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. *CoRR*, abs/1603.08637, 2016.

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

[92] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[93] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015.

[94] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*, 2013.

[95] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *ICML*, 2014.

[96] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and

Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 230–246, 2018.

[97] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mache approach to learning 3d surface generation. *arXiv preprint arXiv:1802.05384*, 2018.

[98] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Shape correspondences from learnt template-based parametrization. *arXiv preprint arXiv:1806.05228*, 2018.

[99] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5769–5779, 2017.

[100] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[101] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[102] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *CVPR*, 2016.

[103] Heli Ben Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *SIGGRAPH Asia*, 2018.

[104] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *CoRR*, abs/1906.06543, 2019.

[105] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, pages 412–420. IEEE, 2017.

[106] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[107] Paul Henderson and Vittorio Ferrari. Learning to generate and

reconstruct 3d meshes with only 2d supervision. *arXiv preprint arXiv:1807.09259*, 2018.

[108] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[109] Luke B Hewitt, Maxwell I Nye, Andreea Gane, Tommi Jaakkola, and Joshua B Tenenbaum. The variational homoencoder: Learning to learn high capacity generative models from few examples. *arXiv preprint arXiv:1807.08919*, 2018.

[110] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

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

[112] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arxiv*, 2016.

[113] Seunghoon Hong, Xinchen Yan, Thomas S Huang, and Honglak Lee. Learning hierarchical semantic image manipulation through structured representations. In *Advances in Neural Information Processing Systems*, pages 2708–2718, 2018.

[114] Yedid Hoshen, Ke Li, and Jitendra Malik. Non-adversarial image synthesis with generative latent nearest neighbors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5811–5819, 2019.

[115] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[116] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016.

[117] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pages 5077–5086, 2017.

[118] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[119] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[120] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[121] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 30(4):78:1–78:8, 2011.

[122] Wittawat Jitkrittum, Patsorn Sangkloy, Muhammad Waleed Gondal, Amit Raj, James Hays, and Bernhard Schölkopf. Kernel mean matching for content addressability of GANs. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3140–3151, 2019.

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

[124] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. *arXiv preprint arXiv:1803.07549*, 2018.

[125] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

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

[127] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *Proceedings of the sixth International Conference on Learning Representations*, 2018.

[128] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[129] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[130] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. 2018.

[131] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[132] Mohammadhadi Kiapour, Shuai Zheng, Robinson Piramuthu, and Omid Poursaeed. Generating a digital image using a generative adversarial network, September 19 2019. US Patent App. 15/923,347.

[133] Jaechul Kim, Ce Liu, Fei Sha, and Kristen Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2314, 2013.

[134] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[135] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

[136] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[137] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[138] Alexander Kort. Computer aided inbetweening. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 125–132. ACM, 2002.

[139] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Burna Joan. Surface networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018.

[140] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[141] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *technical report*, 2009.

[142] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[143] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[144] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[145] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[146] Alex Lamb, Vincent Dumoulin, and Aaron Courville. Discriminative regularization for generative models. In *ICML*, 2016.

[147] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS*, 2011.

[148] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

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

[150] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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

[152] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[153] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[154] Zohar Levi and Craig Gotsman. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE transactions on visualization and computer graphics*, 21(2):264–277, 2015.

[155] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5772, 2017.

[156] Yingwei Li, Song Bai, Cihang Xie, Zhenyu Liao, Xiaohui Shen, and Alan L Yuille. Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. *arXiv preprint arXiv:1904.00979*, 2019.

[157] Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. In *ICML*, 2015.

[158] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[159] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[160] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. *arXiv preprint arXiv:1712.00268*, 2017.

[161] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011.

[162] Jerry Liu, Fisher Yu, and Thomas Funkhouser. Interactive 3d modeling with a generative adversarial network. *International Conference on 3D Vision (3DV)*, 2017.

[163] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. *arXiv preprint arXiv:1905.01723*, 2019.

[164] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[165] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019.

[166] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. *Neural Information Processing Systems (NeurIPS)*, Oct 2019.

[167] Xiaofeng Liu, Yang Zou, Lingsheng Kong, Zhihui Diao, Junliang Yan, Jun Wang, Site Li, Ping Jia, and Jane You. Data augmentation via latent space interpolation for image classification. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 728–733. IEEE, 2018.

[168] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[169] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[170] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[171] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll,

and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015.

[172] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. pages 154–169, September 2014.

[173] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987.

[174] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

[175] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Michael E Houle, Grant Schoenebeck, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[176] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[177] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *IJCV*, pages 1–23, 2016.

[178] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *NIPS*, 2016.

[179] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1429–1437, 2019.

[180] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

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

[182] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin,

and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[183] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. *arXiv preprint arXiv:1704.05712*, 2017.

[184] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv*, 2014.

[185] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *arXiv preprint arXiv:1610.08401*, 2016.

[186] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of universal adversarial perturbations. *arXiv preprint arXiv:1705.09554*, 2017.

[187] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[188] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. *arXiv preprint arXiv:1707.05572*, 2017.

[189] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[190] Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv*, 2016.

[191] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[192] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visual-

ization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv*, 2016.

[193] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. *arXiv preprint arXiv:1904.01774*, 2019.

[194] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *ICLR submissions*, 2017.

[195] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

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

[197] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[198] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[199] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[200] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CVPR*, 2019.

[201] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[202] Omid Poursaeed, Matthew Fisher, Noam Aigerman, and Vladimir G Kim. Coupling explicit and implicit surface representations for generative 3d modeling. *arXiv preprint arXiv:2007.10294*, 2020.

[203] Omid Poursaeed, Tianxing Jiang, Quintessa Qiao, Nayun Xu, and Vladimir G. Kim. Self-supervised learning of point clouds via orientation estimation. *arXiv preprint arXiv:2008.00305*, 2020.

[204] Omid Poursaeed, Tianxing Jiang, Harry Yang, Serge Belongie, and Ser-Nam Lim. Fine-grained synthesis of unrestricted adversarial examples. *arXiv preprint arXiv:1911.09058*, 2019.

[205] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.

[206] Omid Poursaeed, Vladimir G Kim, Eli Shechtman, Jun Saito, and Serge Belongie. Neural puppet: Generative layered cartoon characters. *arXiv preprint arXiv:1910.02060*, 2019.

[207] Omid Poursaeed, Tomáš Matera, and Serge Belongie. Vision-based real estate price estimation. *Machine Vision and Applications*, 29(4):667–676, 2018.

[208] Omid Poursaeed, Guandao Yang, Aditya Prakash, Qiuren Fang, Hanqing Jiang, Bharath Hariharan, and Serge Belongie. Deep fundamental matrix estimation without correspondences. In *European Conference on Computer Vision*, pages 485–497. Springer, 2018.

[209] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. *arXiv preprint arXiv:1801.08926*, 2018.

[210] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[211] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[212] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[213] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[214] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10877–10887, 2018.

[215] Adnan Siraj Rakin, Zhezhi He, Boqing Gong, and Deliang Fan. Robust pre-processing: A robust defense method against adversary attack. *arXiv preprint arXiv:1802.01549*, 2018.

[216] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. *arXiv preprint arXiv:1807.10267*, 2018.

[217] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

[218] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[219] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[220] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[221] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 120(3):300–323, 2016.

[222] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[223] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[224] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

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

[226] Aurko Roy, Colin Raffel, Ian Goodfellow, and Jacob Buckman. Thermometer encoding: One hot way to resist adversarial examples. 2018.

[227] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[228] Tim Sainburg, Marvin Thielk, Brad Theilman, Benjamin Migliori, and Timothy Gentner. Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions. *arXiv preprint arXiv:1807.06650*, 2018.

[229] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[230] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. 2018.

[231] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. *arXiv*, 2016.

[232] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR*, 2014.

[233] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pages 203–222. Springer, 1996.

[234] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 2, page 5, 2017.

[235] Zhixin Shu, Mihir Sahasrabudhe, Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance. *arXiv preprint arXiv:1806.06503*, 2018.

[236] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv*, 2013.

[237] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[238] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[239] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. *arXiv preprint arXiv:1811.11155*, 2018.

[240] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

[241] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[242] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *The IEEE conference on computer vision and pattern recognition (CVPR)*, volume 3, page 4, 2017.

[243] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[244] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems*, pages 8312–8323, 2018.

[245] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[246] Hao Su, Haoqiang Fan, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. *CVPR*, 2017.

[247] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[248] Daniel Sỳkora, John Dingliana, and Steven Collins. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pages 25–33. ACM, 2009.

[249] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[250] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[251] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[252] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[253] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[254] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree gener-

ating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, volume 2, page 8, 2017.

[255] Lucas Theis and Matthias Bethge. Generative image modeling using spatial lstms. In *NIPS*, 2015.

[256] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[257] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 6541–6550, 2018.

[258] Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

[259] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[260] Harri Valpola. From neural pca to deep unsupervised learning. *Adv. in Independent Component Analysis and Learning Machines*, pages 143–171, 2015.

[261] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[262] Abbhinav Venkat, Sai Sagar Jinka, and Avinash Sharma. Deep textured 3d reconstruction of human bodies.

[263] Vikas Verma, Alex Lamb, Christopher Beckham, Aaron Courville, Ioannis Mitliagkis, and Yoshua Bengio. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *arXiv preprint arXiv:1806.05236*, 2018.

[264] Deepak Vijaykeerthy, Anshuman Suri, Sameep Mehta, and Ponnurangam

Kumaraguru. Hardening deep neural networks via adversarial model cascades. *arXiv preprint arXiv:1802.01448*, 2018.

[265] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[266] Kevin Wampler. Fast and reliable example-based mesh ik for stylized deformations. *ACM Transactions on Graphics (TOG)*, 35(6):235, 2016.

[267] Lipeng Wan, Jun Wan, Yi Jin, Zichang Tan, and Stan Z Li. Fine-grained multi-attribute adversarial learning for face generation of age, gender and ethnicity. In *2018 International Conference on Biometrics (ICB)*, pages 98–103. IEEE, 2018.

[268] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *ICLR submissions*, 2017.

[269] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. *arXiv preprint arXiv:1804.01654*, 2018.

[270] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016.

[271] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 218–234, 2018.

[272] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. *ICLR submissions*, 2017.

[273] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

[274] Brian Whited, Gioacchino Noris, Maryann Simmons, Robert W Sumner, Markus Gross, and Jarek Rossignac. Betweenit: An interactive tool for

tight inbetweening. In *Computer Graphics Forum*, volume 29, pages 605–614. Wiley Online Library, 2010.

[275] Eric Wong and J Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

[276] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. In *Advances In Neural Information Processing Systems*, pages 5962–5972, 2018.

[277] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[278] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018.

[279] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[280] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. *arXiv preprint arXiv:1703.08603*, 2017.

[281] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. *arXiv preprint arXiv:1812.03411*, 2018.

[282] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1316–1324, 2018.

[283] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016.

[284] H Yang, WY Lin, and J Lu. Daisy filter flow: A generalized approach to discrete dense correspondences. CVPR, 2014.

[285] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *ICLR submissions*, 2017.

[286] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *arXiv preprint arXiv:1703.01560*, 2017.

[287] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2018.

[288] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016.

[289] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[290] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[291] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[292] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[293] Yuting Zhang, Kibok Lee, and Honglak Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *ICML*, 2016.

[294] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *arXiv preprint arXiv:1612.01105*, 2016.

[295] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *ICLR Workshop*, 2016.

[296] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv*, 2016.

[297] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[298] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

[299] Silvia Zuffi and Michael J Black. The stitched puppet: A graphical model of 3d human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3537–3546, 2015.