# Rewriting Geometric Rules of a GAN

Sheng-Yu Wang

CMU-RI-TR-22-77

December 5, 2022


The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

**Thesis Committee:**
Prof. Jun-Yan Zhu, *chair*
Prof. Shubham Tulsiani
Yufei Ye

*Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.*

# Abstract

Deep generative models make visual content creation more accessible to novice users by automating the synthesis of diverse, realistic content based on a collected dataset. However, the current machine learning approaches miss a key element of the creative process – the ability to synthesize things that go far beyond the data distribution and everyday experience. To begin to address this issue, we enable a user to "warp" a given model by editing just a handful of original model outputs with desired geometric changes. Our method applies a low-rank update to a single model layer to reconstruct edited examples. Furthermore, to combat overfitting, we propose a latent space augmentation method based on style-mixing. Our method allows a user to create a model that synthesizes endless objects with defined geometric changes, enabling the creation of a new generative model without the burden of curating a large-scale dataset. We also demonstrate that edited models can be composed to achieve aggregated effects, and we present an interactive interface to enable users to create new models through composition. Empirical measurements on multiple test cases suggest the advantage of our method against recent GAN fine-tuning methods. Finally, we showcase several applications using the edited models, including latent space interpolation and image editing.

# Acknowledgments

I would like to thank my advisor, Prof. Jun-Yan Zhu, for the support and advice throughout my study. Jun-Yan taught me many important nuances in research, especially how important it is for researchers to be methodological. I'm extremely grateful that Jun-Yan always patiently guide me towards becoming a better researcher.

Next, I would like to thank my collaborator David Bau for providing me new perspectives in research. I cannot stress how inspiring it is to summarize the whole course of academia as teaching. After all, what we are excited about is educating ourselves as well as other people.

I would also like to thank the other members in the lab – Gaurav Parmar, Kangle Deng, Nupur Kumari, George Cazenavette, Muyang Li, Ruihan Gao, Chonghyuk (Andrew) Song, Daohan (Fred) Lu, Rohan Agarwal, Bingliang Zhang, Maxwell Jones, Mia Tang, and Aniruddha Mahapatra for the research conversations, casual chats, and fun.

Last but not least, I would like to thank my family for feeding me during the COVID years, where I started my CMU studies back in Taiwan, and of course, for the endless support.

# Contents



*When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.*

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Deep generative models have reduced the technical barriers to creating visual content. They can free a person from the need to develop all the skills to create the fine details of a realistic image or an art piece. Instead, by taking simple user instructions, generative models apply statistical prowess to synthesize realistic, diverse content and fill in details that imitate observed patterns in our visual world. Using a generative model, a novice user can create a magnificent landscape scene with a few brushstrokes [54], synthesize a video of themselves performing expert dance moves [15], or even generate and manipulate a photo based on a simple text prompt [56, 59].

These applications all use generative models as a data-driven source, so that newly-created content will resemble previous content in the training data distribution. However, we humans do not limit our creativity to imitations of things that already exist. For instance, the creators of movies such as Avatar [13] or Star Wars [47] imagined fantastical creatures unlike any in the real world, inspiring wonder and excitement about a vivid universe beyond everyday experience. Currently, to create a generative model of such virtual creatures, an artist could fabricate a number of examples from scratch and apply few-shot GAN training methods [33, 42, 50, 88]. However, the process of assembling enough data to apply these methods is still time-consuming, costly, and not an option for everyday users.

This leads to the question: can a novice user apply changes to a generative model to synthesize new content that differs from existing examples? We draw inspiration from the creative process of visual effects artists who imagine new worlds [19]. Designing a new creature can begin by warping an existing species into a completely new shape to

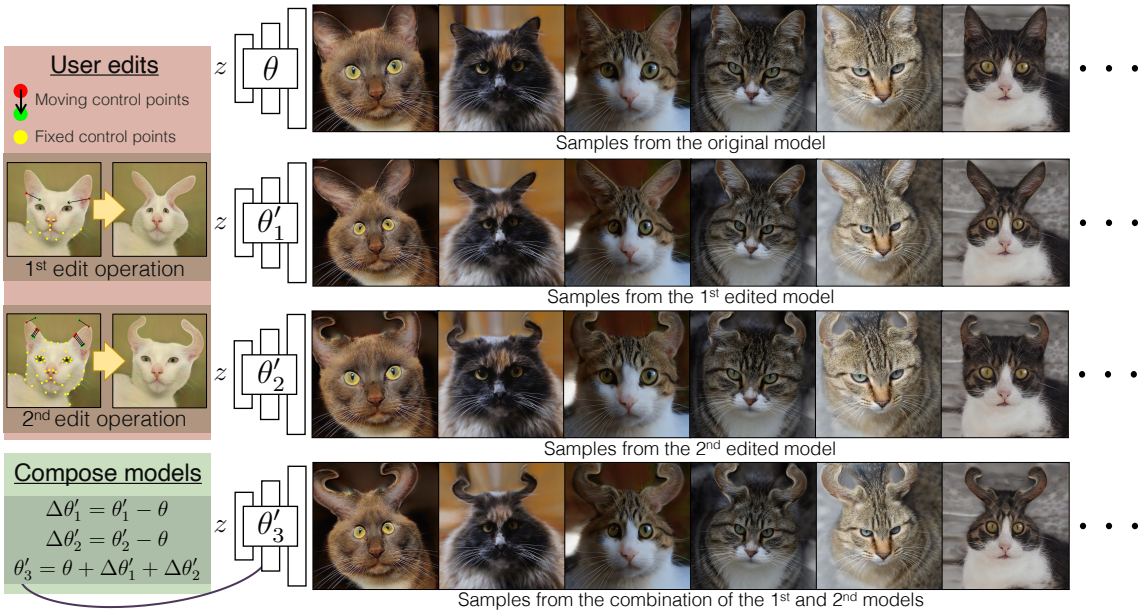


Figure 1.1: With our method, a user can edit a GAN model to synthesize many unseen objects with the desired shape. The user is asked to warp just a handful of generated images by defining several control points **(User edits)**, to obtain the customized models **(1st and 2nd edited models)**. Furthermore, a user can compose the edited models into a new model with aggregated geometric changes **(last row)**. For each row, the same noise $z$ is used for all models. While the edited models change an object's shape, other visual cues, such as pose, color, texture, and background, are faithfully preserved after the modification.

achieve the desired effect. The artist redefines the geometric rules of a creature, for example, re-imagining ordinary shapes of animal eyes into expressive new shapes that portray a particular personality. We ask if such a creative process can be applied to manipulate a generative model, using simple user inputs to change the geometric rules within a learned model. A manipulated generative model would be able to synthesize a universe of creatures with diversity and variation, all following new geometric rules defined by the artist.

We present a new problem setting: user-guided modification of the geometric rules encoded in a pre-trained generative model. To edit a model, we ask the users to move a handful of control points as warping examples. Exisiting GAN inversion [91] and few-shot GAN adaptation [50] methods fail to tackle this new problem: the geometric changes can go beyond the learned distribution, placing the goal out of reach of GAN inversion; and given extremely few user inputs, it is difficult to adapt a GAN to the new domain. To address these issues, we directly optimize the model weights with a reconstruction loss between original samples and their warped versions. We introduce an augmentation scheme based on style-mixing [32] to improve our model's generalization to unseen samples, and we show that we can update only a small subset of networks weights to rewrite the geometric rules. Moreover, by a simple linear combination of edited model's weights, we observe that the edited models can be composed to build a model with combined shape-changing effects. Based on this finding, we present an interactive interface that allows users to build new models by composing different models.

As shown in Figure 1.1, our method enables a user to edit a whole generative model that synthesizes an endless collection of creatures, such as the long-eared rabbit cats, or the devilish cats with curly-shaped ears. We can also combine two warped models to create a new model of rabbit cats with curly ears.

We use our method to create several edited generative models, and we demonstrate several applications of these models, including smooth transitions between generated images, GAN-based image manipulations, and real photo editing. We also show how to generalize our formulation to apply coloring changes to a generative model. Furthermore, we evaluate our method on multiple warping operations on different classes to fully characterize its performance. We show that our method outperforms recent few-shot GAN fine-tuning methods. Finally, we include an extensive ablation study regarding several algorithmic choices and hyperparameters. Our code and models are available at our website: https://peterwang512.github.io/GANWarping.

# Chapter 2

# Related Works

## 2.1 Generative Models for Content Creation

Deep generative models [18, 25, 39, 51] and its recent advances [11, 23, 27, 32, 35, 38, 60] have enabled a wide range of image and video synthesis applications. Recent applications include text-based image synthesis and editing [56, 59], semantic photo synthesis [54], motion transfer [15, 78], image manipulation [62, 73, 91], virtual Try-on [5, 41], and face editing [2, 4, 52, 57, 72]. Generative models excel at synthesizing realistic samples as they learn the prior of natural image statistics from training data. Unfortunately, they are also confined to imitating the training data, making synthesizing out-of-the-distribution samples incredibly difficult. Different from the above works, our goal is *not* to synthesize or edit an image via a generative model. Instead, we aim to change the geometric rules of the model without curating a large-scale training set.

## 2.2 Model Fine-tuning and Rewriting

Various methods have been proposed to adapt the pre-trained generative model to a small unseen domain. Fine-tuning from pre-trained weights can improve upon training from scratch [79] but is also prone to overfitting. To alleviate this issue, several data augmentation methods [33, 74, 88, 89] and regularization methods [42, 50, 75] are proposed. Some also suggest limiting the changes in model weights [48, 49, 80, 87], and some use models

pre-trained on other vision tasks to make the discriminator more robust [22, 40, 65, 71]. Cross-domain methods fine-tune a generative model to match user-specified sketches [77] or text prompts [24]. Our work differs from these previous approaches since our task does *not* adapt a model to an existing target distribution, and our rewritten rules can apply out-of-distribution shape changes that cannot be achieved by latent manipulations [44, 55, 86]. We enable the user to define a novel target directly through manipulation of the model. Our approach is inspired by work that rewrites object association rules of a deep network [9, 64], but while those works apply to copy-and-paste changes, our method addresses the new problem of rewriting the geometric rules that define shapes in a generative model.

## 2.3 Image Warping

Image warping is a classic image editing problem in computer graphics and computational photography. There are mainly three types of warping operations. Global warping aims to transform every single pixel via the same transformation matrix and has been widely used in panorama image stitching[12] and registration [20]. However, global transformation cannot model complex local geometric changes. In contrast, we can use a per-pixel dense flow field to warp an input image into an arbitrary shape, in which each vector in the flow field describes the pixel movement between two images. Thanks to its expressiveness, they have been used in image editing and style transfer [7, 43, 68, 69] as well as image alignment [45, 46]. The third approach is to achieve local deformation through sparse correspondence. This approach allows a user to specify a small number of source and target control points or lines, and the algorithm then constructs a dense flow field using various interpolation techniques [6, 29, 36, 66, 82]. The reconstructed flow field needs to satisfy the sparse user controls while staying smooth in the spatial domain. In this work, we leverage Moving Least Squares [66] to create editing examples. Recently, conditional generative models such as WarpGAN [67], CariGAN [14], and StyleCariGAN [30] warp and stylize faces to create caricatures. However, unlike previous methods, which aim to warp a single image, we aim to change the weights of a neural network so that the resulting model can consistently produce samples that reflect the desired geometric changes without additional user inputs.

# Chapter 3

# Methods

We aim to modify the geometric rules of a generative model according to a few user edits. Given a pre-trained generator $G(\mathbf{z}; \theta)$, a user is asked to edit a small number of generated samples. For generative models that use an intermediate latent space (e.g., StyleGAN3 [35]), we denote the intermediate latent code by $\mathbf{z}$ to make notations consistent. $\{\mathbf{z}_{(i)}, \mathcal{T}_{(i)}(G(\mathbf{z}_{(i)}; \theta))\}_{i=1}^{N}$ denotes the sampled latent codes and corresponding edits, where $N$ is the number of samples, and $\mathcal{T}_{(i)}$ is the user-defined local warping function for the $i$-th each generated image. We often refer to them as training examples as we train our model with these edits. Our goal is to learn a new model $\theta'$, whose samples resemble the visual effect of user edits.

We introduce a simple method to edit the GAN model. In Section 3.1, we formulate our training objective and discuss several alternative formulations. While directly applying this objective leads to severe overfitting, we drastically improve our model's generalization via a latent space augmentation method based on the disentanglement structure of the generative model (Section 3.2). Furthermore, we perform model updates on a smaller set of parameters to significantly reduce model storage (Section 3.3). We then demonstrate how to compose multiple geometric changes into a single model and present an interactive interface for users to create their own models (Section 3.4). Finally, in Section 3.5, we extend our method to achieve color changes.

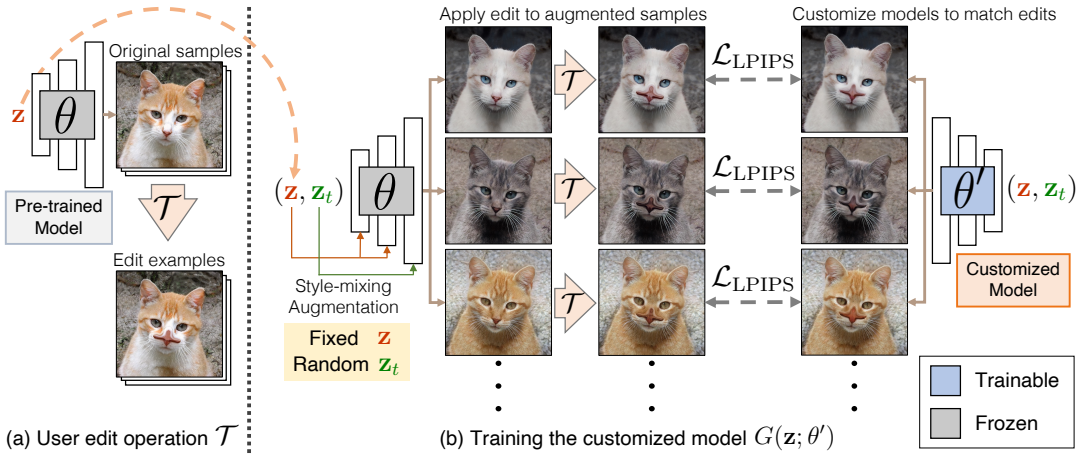


Figure 3.1: **Overview.** (a) A user first edits a handful of samples from the pre-trained generative model. Each user edit $\mathcal{T}_{(i)}$ warps the input $G(\mathbf{z}_{(i)};\theta)$ to output $\mathcal{T}_{(i)}(G(\mathbf{z}_{(i)};\theta))$. (b) We train a customized model $\theta'$ so that it can synthesize new samples with a similar visual effect specified by the user edits $\mathcal{T}_{(i)}$. To prevent overfitting, we apply style-mixing augmentation to the edited samples (Section 3.2). For each sample, we mix the original latent code $\mathbf{z}_{(i)}$ with a new randomly sampled texture latent code $\mathbf{z}_t$. Since the augmented samples still preserve shapes and poses, we can apply the same user edit $\mathcal{T}_{(i)}$ to obtain a training set with diverse texture variations. We learn the customized model $\theta'$ on the augmented training set using the LPIPS loss [84]. In the figure, we denote $\mathbf{z}_{(i)}$ by $\mathbf{z}$ and $\mathcal{T}_{(i)}$ by $\mathcal{T}$ for brevity.

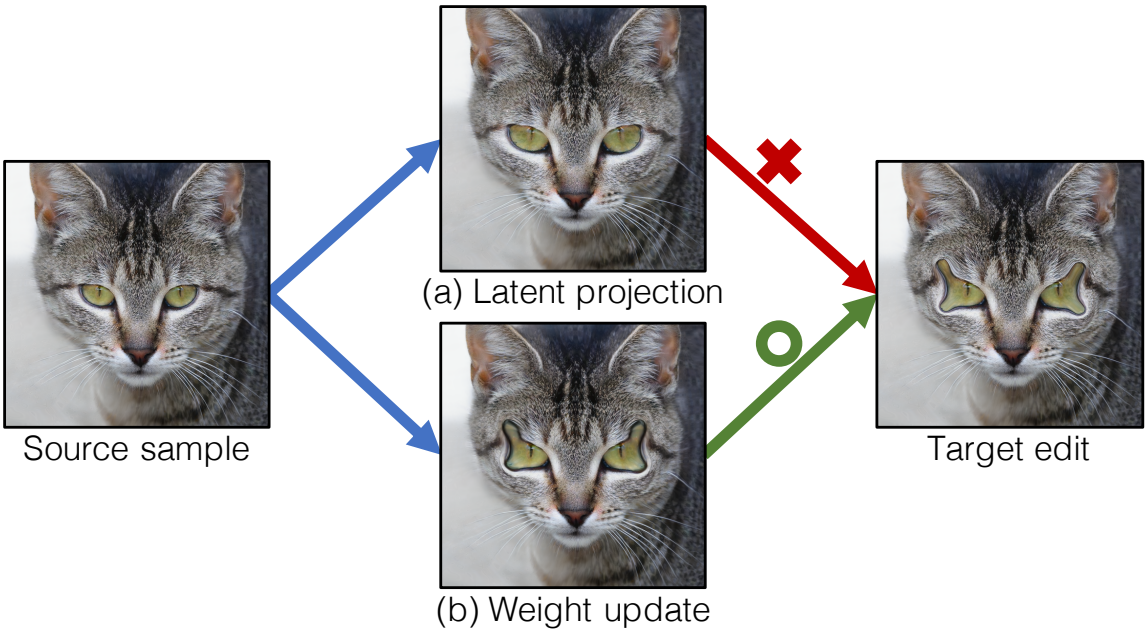


Figure 3.2: **Reconstructing an image with a drastic warp edit.** We reconstruct a model-generated sample that is warped drastically by (a) projecting the edited image into the extended latent space proposed by Abdal *et al*. abdal2020image2stylegan or (b) updating the generative model weights directly. The latent projection method can only produce an image with bigger eyes, whereas the weight update method correctly matches the target edits. The latent code from the source sample is used to initialize GAN projection.

## 3.1 Adapting Generator Weights to User Edits

**Warping interface for collecting training examples.** To make model creation accessible to a novice user, we present a warping interface wherein the user can define the warping function $\mathcal{T}_{(i)}$ intuitively. We create training edit examples using random sampling, and the user can manually reject samples when editing is not feasible (e.g., parts to edit are occluded). For each sample, a user clicks to define a control point p, then drags and releases the click to define the new position q. The control points indicate where and how each part of the generated image should move. Given user control points, we compute the image deformation grid and apply inverse warping using Moving Least Squares [66] to obtain the edited examples $\mathcal{T}_{(i)}(G(\mathbf{z}_{(i)};\theta))$. As shown in Figure 1.1, a user can warp a cat face into a rabbit face or deform a cat ear into a curly shape within a few clicks. Thanks to the flexibility of the control points, our users can quickly introduce a wide range of geometric

changes. Our method is not tied to a particular local deformation algorithm and can be used with other methods [6, 29, 82] and software (e.g., Photoshop Liquify).

**Challenges and earlier attempts.** Altering the geometric rules of a model is a challenging task. To create a local geometric change, such as changing the shape of a nose, the new model needs to decide how to move the correct parts to the right place. To achieve this, the model has to detect object parts and understand geometric relations between different parts. However, given a few training examples, it might be difficult to learn these concepts and generalize well to unseen samples.

In the early stage, we experimented with two problem formulations. Our first attempt was to formulate it as a few-shot GAN training problem [33, 50, 79, 88] and treated the edited images as training data for GANs. Unfortunately, direct GAN training, even with pre-trained generators, regularization, and data augmentation, still fails to work due to the limited number of training samples (i.e., less than 10). We then cast the problem as an image projection problem [1, 28, 61, 91]. Image projection aims to obtain the optimal latent vector such that the generated sample reconstructs the target image. One can further traverse the latent space to manipulate images. However, as demonstrated by Figure 3.2, it is difficult to even project a single warped output to the latent space.

**Our learning objective.** As it is difficult to manipulate shapes by only updating the latent vector, we propose updating the *network weights* to change the geometric rules inside the models directly. This brings two potential benefits. First, we can reconstruct training samples more easily due to the high-dimensional nature of weight space. Second, we observe that it is easier to introduce out-of-the-distribution geometric changes (e.g., deform a cat ear into a curly shape) compared to using latent directions [26].

More concretely, we optimize the model weights to reconstruct the edited examples using the following loss:

$$\underset{\theta'}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{LPIPS}}(G(\mathbf{z}_{(i)}; \theta'), \mathcal{T}_{(i)}(G(\mathbf{z}_{(i)}; \theta))), \tag{3.1}$$

where $\mathcal{L}_{\text{LPIPS}}$ is the LPIPS loss [84] that measures the similarity between two images based on deep feature embeddings. The objective above encourages the new model $\theta'$ to mimic the user edits $\mathcal{T}_{(i)}$ for each generated example $G(\mathbf{z}_{(i)}; \theta)$ over N examples. We initialize the

weights as $\theta' = \theta$ for faster convergence. Regarding the number of training examples $N$, we have included an ablation study for $N \in \{1, 2, 4, 6, 8, 10\}$. We observe that our method works well for 5 to 10 user edits.

## 3.2 Style-mixing Augmentation

However, updating weights also introduce a new issue: overfitting. While this method accurately reconstructs the training examples, the resulting model weights do not generalize to unseen samples, as shown in Figure 4.6. Collecting more user edits (e.g., hundreds or thousands) can potentially alleviate the overfitting issue. Unfortunately, it requires a significant amount of manual efforts.

To resolve this, we apply a latent space augmentation scheme that utilizes the disentanglement of the generative model. In state-of-the-art GANs [11, 35], the layer weights are conditioned on the latent vectors, and StyleGAN [32] observes that the latent codes in the earlier layers control the shape of objects, while those in the later layers control color and texture. Furthermore, one can perform "style-mixing," where two different latent codes are applied to the earlier and later layers respectively. The two different latent codes control different aspects of the sampling process.

Building on this concept, we can obtain infinitely many new data points from each warping edit by randomly sampling the latent codes of the color and texture. Since the original latent code controlling the shape does not change, the same warping function can be applied to the new samples and still achieve the desired edits. Mathematically speaking, the objective becomes:

$$\underset{\theta'}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{z}_\mathrm{t}} \left[ \mathcal{L}_{\text{LPIPS}}(G(\mathbf{z}_{(i)}, \mathbf{z}_\mathrm{t}; \theta'), \mathcal{T}_{(i)}(G(\mathbf{z}_{(i)}, \mathbf{z}_\mathrm{t}; \theta))) \right]. \tag{3.2}$$

As shown in Figure 3.1 (b), $\mathbf{z}_{(i)}$ is the original latent code, which we used to create the input example. During training time, we fix $\mathbf{z}_{(i)}$ to preserve the shape of the input image. We then mix it with a randomly sampled latent code $\mathbf{z}_\mathrm{t}$ to introduce new texture variations and create *free* training examples. Figure 4.6 shows the comparison between models trained with and without the style-mixing augmentation, and it is evident that the models trained with our augmentation generalize much better.

## 3.3 Updating Models with Fewer Parameters

Inspired by the recent works on fine-tuning generative models [9, 24, 53, 77], we explore optimizing only a subset of model weights during training. Surprisingly, we obtain similar performances by tuning either all weights or just the weight of a single layer. Moreover, for single-layer updates, we find that the weight update has a low-rank structure, agreeing with the findings of Bau *et al*. bau2020rewriting. Taking advantage of the low-rank structure, we can train a model by updating the weight at layer $j$, denoted by $W_j$, where we enforce the weight update to be low-rank as follows:

$$\begin{aligned} &\underset{U,V}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{z}_\mathrm{t}} \left[ \mathcal{L}_{\text{LPIPS}}(G(\mathbf{z}_{(i)}, \mathbf{z}_\mathrm{t}; W'_j), \mathcal{T}_{(i)}(G(\mathbf{z}_{(i)}, \mathbf{z}_\mathrm{t}; W_j))) \right] \\ &\text{subject to} \qquad W'_j = W_j + UV \end{aligned} \tag{3.3}$$

Following the formulation of [9], we view $W_j \in \mathbb{R}^{m \times n}$ as a $m$ by $n$ matrix, where $n$ and $m$ denote the number of input and output channels, respectively for the $1 \times 1$ Conv layers used in StyleGAN3 [35]. We define $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$, where $r$ is the desired rank. As a design choice, we choose a small $r$ such that $r \ll m, n$. Since the optimized parameters $U$ and $V$ are restricted to rank $r$ by their shape, the change in the weight $W'_j - W_j = UV$ has a rank not greater than $r$. The update scheme further reduces the storage requirement for each created model, since we can store just the difference of the weights as $U$ and $V$. At inference time, we can add the difference to the pre-trained model to perform the model edits.

In our setting, storing the full weight takes around 60 MB, and storing a single layer takes around 4 MB. We observe that $r = 50$ performs well, and storing the rank-50 matrices takes around $0.6$MB, reducing the storage by $100\times$. Efficient storage is critical for real-world applications as we want to avoid storing the weight differences at full scale, each time we create a new effect.

We observe that our proposed style-mixing augmentation serves as a strong regularization for the above three variants: full-model updates, single-layer updates, and low-rank updates. We provide a detailed analysis in Section 4.1.

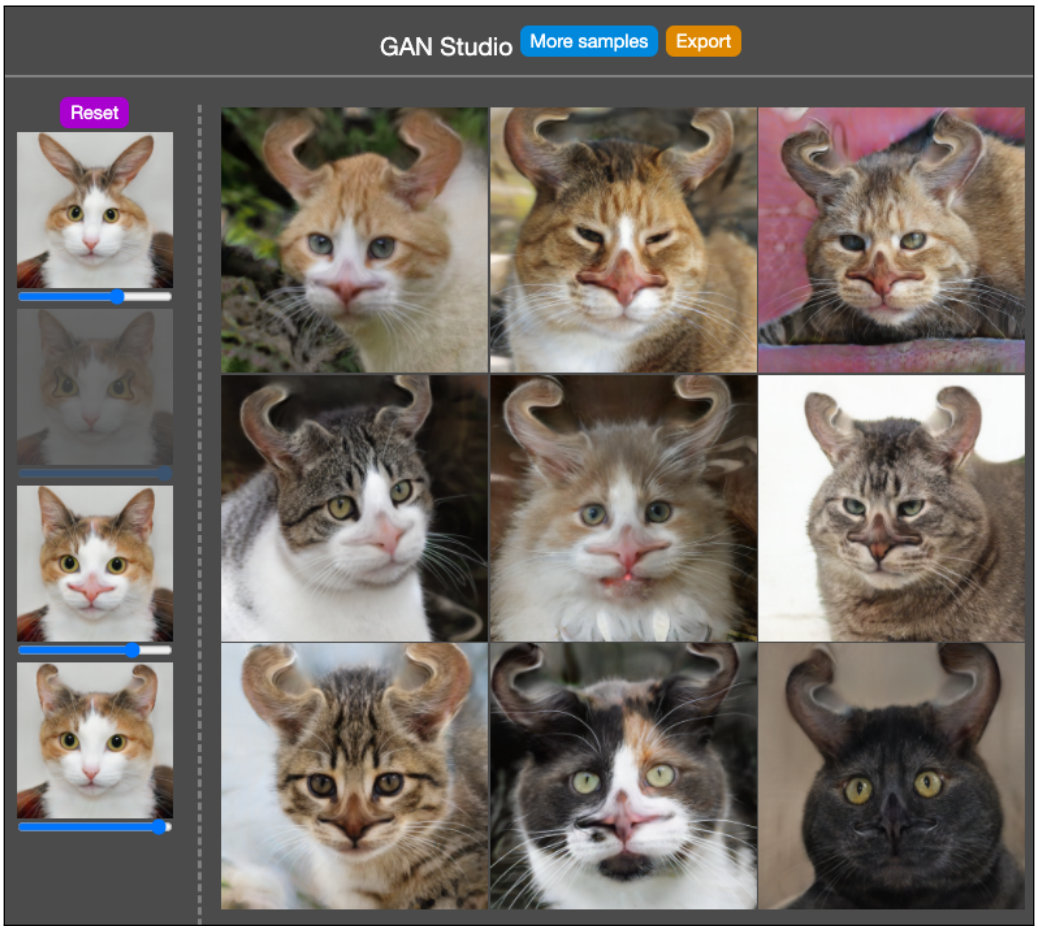


Figure 3.3: **User interface for model composition.** We present an interface for users to easily create a new model by composing the edited models made beforehand. To get the desired model, the user can toggle on and off each edited model on the left and move the slider bars to adjust the blending strength of each model. The user can also click a button to view more samples from the new model and export the created model weights for future applications. Please view the supplemental video for more details.

## 3.4 Composing Multiple Edited Models

One of our key applications is to enable users to compose different models created beforehand to achieve aggregated effects. The composition operation is powerful since it caters to the group of users who wish to have simple, interactive access to the creation of a new model. For example, users can download models with preset effects shared by other model creators, and personalize the combination of models.

Surprisingly, the composition can be achieved by a simple linear operation in the weight space. Given a set of models created from the same pre-trained generator, we denote the weights of the pre-trained model as $\theta$ and previously created models as $\{\theta'_1, \theta'_2, \ldots, \theta'_K\}$. We can compose a new model of weights $\theta'_{\text{new}}$ as follows:

$$\theta'_{\text{new}} = \theta + \sum_{k=1}^{K} \alpha_k (\theta'_k - \theta) \tag{3.4}$$

where $\alpha_k$ is a parameter that controls the strength of the edits from each model. Based on the finding, we present an interactive user interface to create new models from a set of predefined models, as shown in Figure 3.3. A user can easily control the contribution of each preset to produce aggregated edits within seconds, and export their favorite combinations as a new, customized model.

## 3.5 Extending to Color Changes

Our method can also be applied to color edits. To achieve this, a user is asked to paint certain image regions with selected colors and specify regions where the color needs to be preserved. For each user edit, we have the specified color $C$, the colored region $M_{\text{color}}$, and the preserved region $M_{\text{fixed}}$. We use this input to design a loss as follows:

$$\begin{aligned} \underset{\theta'}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{z}_\text{t}} \big[ \lambda_{\text{color}} || M_{\text{color}} \odot (C - G(\mathbf{z}_{(i)}, \mathbf{z}_\text{t}; \theta')) ||_1 \\ + || M_{\text{fixed}} \odot (G(\mathbf{z}_{(i)}, \mathbf{z}_\text{t}; \theta) - G(\mathbf{z}_{(i)}, \mathbf{z}_\text{t}; \theta')) ||_1 \big]. \end{aligned} \tag{3.5}$$

We apply the loss to the masks $M_{\text{color}}$ and $M_{\text{fixed}}$, which specify the regions to update or preserve the color, respectively. We find that our method works well with $\lambda_{\text{color}} = 8$. We

also apply style-mixing augmentation to this task. Since the augmentation does not alter shape, the same color mask can be applied to the augmented samples to achieve similar effects. Our edited models produce realistic, local color changes, thanks to the disentangled representation learned in the source models. For example, we can change the color of the entire sky without precisely painting the whole region. More details are in Section 4.3.

# Chapter 4

# Experiments

We evaluate our method, along with the baseline methods, on a number of warping instructions in Section 4.1. In Section 4.2, we compare different variants of our full method. We also present several applications of our method and the qualitative results in Section 4.3.

**Implementation details.** We adopt StyleGAN3-R [35], as the model learns better geometric representations along with translation and rotation invariance. The style-mixing operation is performed in the intermediate latent space produced by the mapping network. Also, we fix the weights for the mapping network for full-weight updates.

We train and test our models on a single NVIDIA A5000 GPU. We train all models with 2000 iterations using an Adam optimizer [37]. We adopt the learning rate schedule from the image projection method proposed by Karras et al. karras2020analyzing, where the learning rate is ramped up from zero linearly during the first 100 iterations and ramped down to zero using a cosine schedule during the last 1000 iterations. We use a learning rate of 0.05 and a batch size of 8 for single-layer updates. For full-model updates, we use a learning rate of 0.001 and a batch size of 6. It takes 20 minutes to train a model of $256 \times 256$ resolution and 40 minutes to train a model of $512 \times 512$ resolution. We find that updating either the full weights or a single layer results in a similar training time.

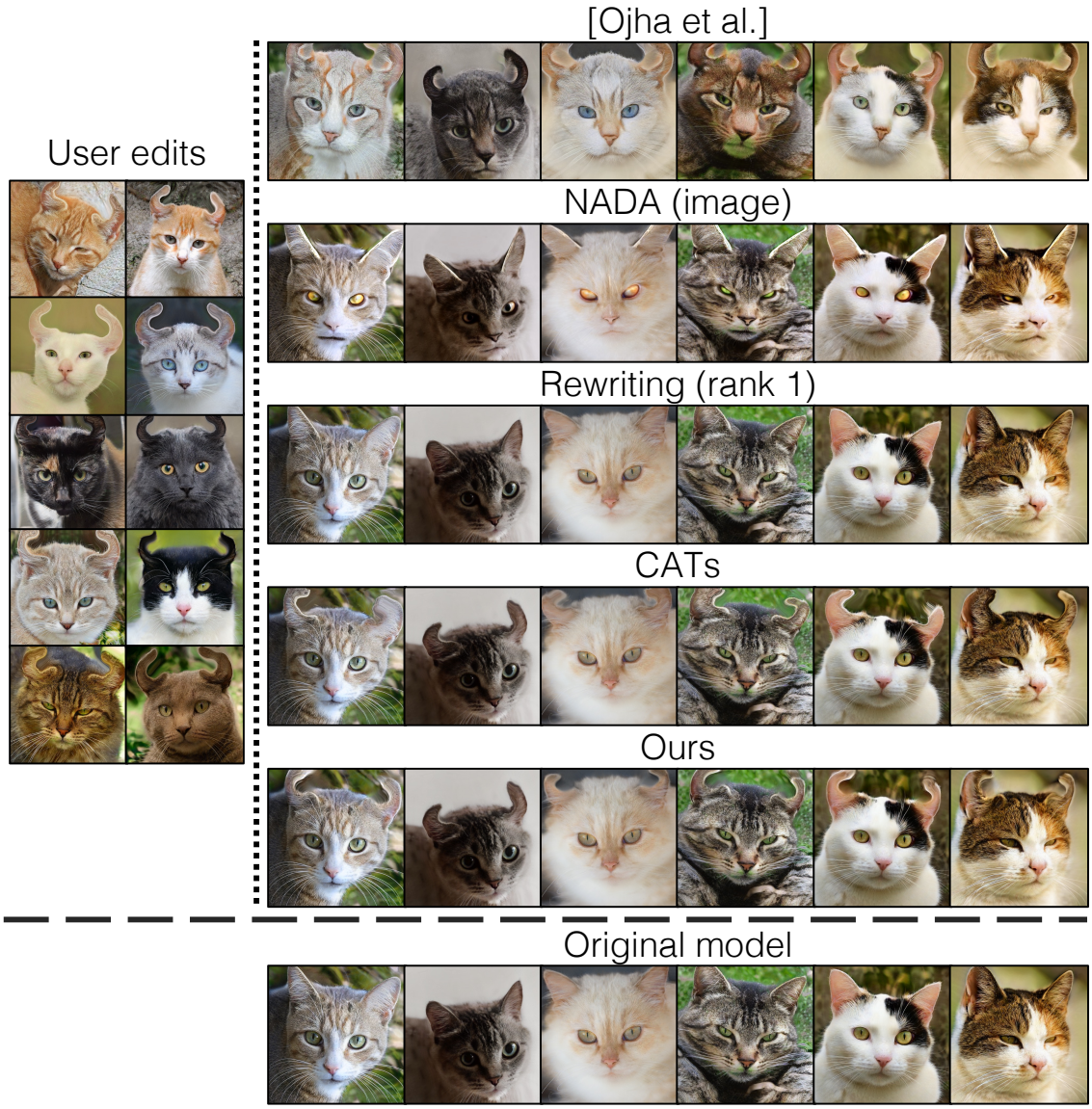


Figure 4.1: **Qualitative baseline comparisons.** We compare our method against several baselines. For each method, we update the original model (bottom) using 10 user edit examples (left). We use the same noise $z$ to generate the images. We find that our method creates a model that matches the specified shape changes, while preserving the color and texture of cat faces and backgrounds. On the other hand, the few-shot GAN fine-tuning method by Ojha *et al*. ojha2021few-shot-gan alters the color and textures, and both StyleGAN-NADA [24] and Model Rewriting [9] fails to apply accurate shape changes. Applying CATs, a supervised dense correspondence model, can achieve similar results but with much higher inference costs in both run-time and memory. More details can be found in Section 4.1.

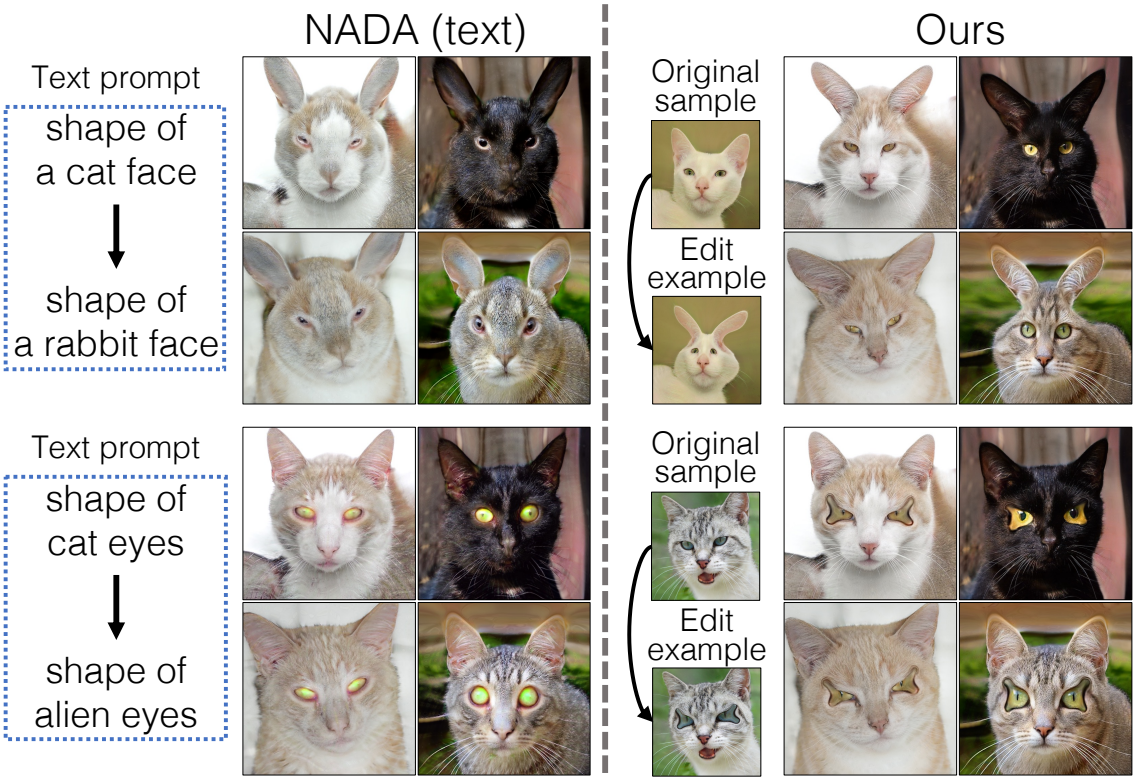


Figure 4.2: **Comparison with text-based model editing.** We test if the text-based model editing proposed by StyleGAN-NADA [24] can achieve the warping edits. It is unnatural to describe the warping edits precisely using text, but we attempted to provide the text prompts (left) that best match the warping edits (right). Despite this, we observe that training with text inputs leads to unintended color and texture changes.

## 4.1 Evaluations

**Test cases.** We demonstrate our model editing method in various ways. We evaluate our method on three source models, which are pre-trained on AFHQv2 cats [17], LSUN horses [83], and Places houses [90]. We perform four different edit operations on the cat model and two on the horse and house model. For every operation, we create ten edit examples for training and five hold-out edit examples for validation and testing. Additionally, we would like to evaluate the edited and unaltered regions separately. To achieve this, we manually mask the edited regions in the edit examples and only use the masks for validation and testing.

**Performance metrics.** We evaluate our method in two ways. First, we expect a decent model to successfully transfer user edits to other unseen samples, wherever the edits are applicable. Also, we hope the model to preserve the content of unaltered regions. In particular, given the edit example and the corresponding output from the edited model, we apply similarity measures separately to the edited region and the untouched region as defined above. The similarity is measured by three standard metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [81], and LPIPS [84].

To directly assess the correctness of shape changes, we adopt the procedure from Wang et al. wang2021sketch, where the symmetric Chamfer Distance (CD) [8] $d(x, y) + d(y, x)$ is calculated between the contour of the two images $x, y$. The contours are computed by DexiNed [70] and further processed to 1-pixel-wide edges as described in Isola et al. isola2017image. We report the Chamfer Distance in the edited region.

We average the numbers over all the test cases from the same source model and report the results for each object category.

**Baseline comparisons.** We compare our method against several recent few-shot GAN fine-tuning methods: TGAN [79], TGAN+ADA [33], FreezeD [48], Ojha et al. ojha2021few-shot-gan, and StyleGAN-NADA [24]. The first four methods are directly trained on the edit examples, and StyleGAN-NADA is trained on CLIP embeddings [58]. The CLIP embeddings are generated either from text inputs or image inputs, and we denote the two baselines by *NADA (text)* and *NADA (image)*, respectively. We acknowledge that it is unnatural to describe the warping edits precisely using text. However, to demonstrate the

inherent differences between the use of text prompts and warping interfaces in our task, we have attempted to provide text prompts that best match the desired warping effect. For *NADA (image)*, we extract the target CLIP embeddings from the same edit examples.

In addition, we compare our work with Model Rewriting [9], a method that applies low-rank updates to a model layer for copy-and-paste edits. We apply Model Rewriting to our warping tasks with rank-1 and rank-50 updates. Also, we compare against a baseline based on the state-of-the-art supervised dense correspondence algorithm CATs [16]. We estimate the correspondences between the edit examples and every unseen sample, which we use to transfer the user edits. More implementation details of baselines are in the supplemental material.

Table 4.1 shows the quantitative comparisons. We find that our method outperforms few-shot GANs, StyleGAN-NADA, and Model Rewriting by far, and this result agrees with the qualitative comparisons presented in Figure 4.1. Few-shot GAN fine-tuning methods, such as the state-of-the-art by Ojha et al. ojha2021few-shot-gan, are not designed to preserve the color and texture of the original model. Instead, these methods alter the textures to match a small number of training samples. Notably, our problem setting differs from few-shot adaptation in two ways. First, we aim to enable a user to directly control the behavior of a pre-trained model, instead of adapting a model to a new domain, Second, our method learns to mimic before-and-after user edits, instead of matching the output distribution.

NADA (image) preserves the color and textures better but fails to apply accurate warping edits. We speculate that CLIP embeddings may not capture sufficient fine-grained information of the image, or may not generalize to out-of-distribution shapes. We also provide a qualitative comparison against NADA (text) in Figure 4.2. Despite our effort to explicitly specify shape via text alone, training with text inputs leads to unwanted color and texture changes. We find that Model Rewriting does not change the object shapes, since the method's training objective is specialized for copy-and-paste edits. On the other hand, the CATs baseline works on par with ours. Trained with dense correspondence supervision, this baseline outperforms our method for cat models, whereas our methods works better for horse and house models. We note that our method is only trained with extremely few samples and takes only one forward pass to create new samples. In contrast, CATs-based warping transfer requires us to align a new sample to all the existing training samples via the dense correspondence algorithm. Also, directly transferring warping fields does not accommodate dramatic changes in object sizes, which hurts the baseline's performance for

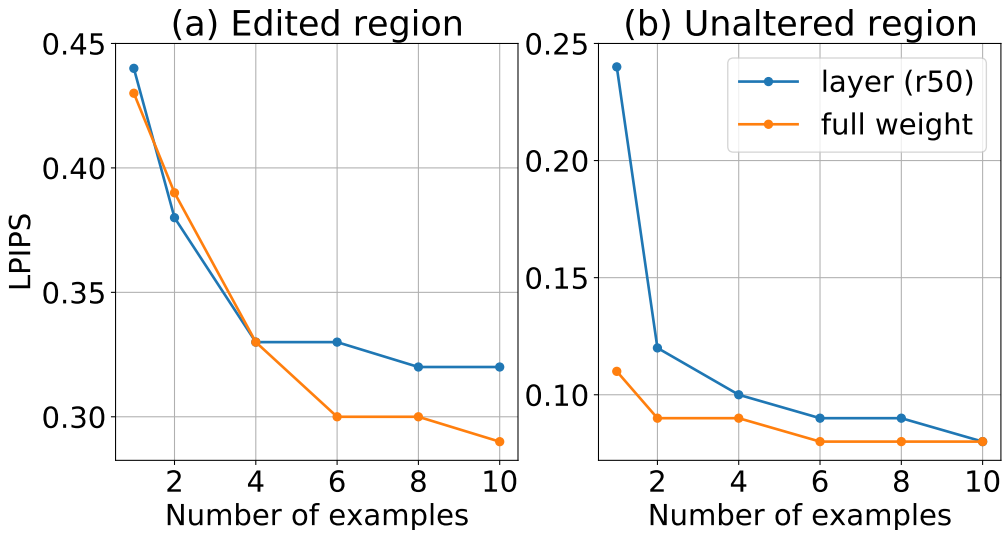


Figure 4.3: **Training with fewer user edit examples.** We study how the number of user edit examples affects our model's performance. For both full-weight updates and rank 50 single-layer updates ("layer (r50)"), the performance improves when more examples are added, and the improvement quickly saturates after training with four or more examples. Both models are trained with style-mixing augmentation, and we report average LPIPS over all test cases.

models with large geometric variations (e.g., horse and house model).

## 4.2 Ablation Study

We investigate our method's performance regarding several algorithmic choices and hyper-parameters. The quantitative analysis is summarized in Table 4.2.

**Style-mixing augmentation.** Applying style-mixing augmentation improves performance consistently in all cases, showing that the augmentation largely increases model generalization. This finding is also consistent with the qualitative results as shown in Figure 4.6, where the augmented models generate more promising ears and noses shape changes in the cat model and produce crisper and more accurate edits on the horse backs and house roofs. As such, the augmented models obtain better similarity scores and Chamfer Distances in the edited region. Table 4.2 also shows that augmented models barely alter the untouched region, since they are trained to preserve samples with a wide range of color and texture

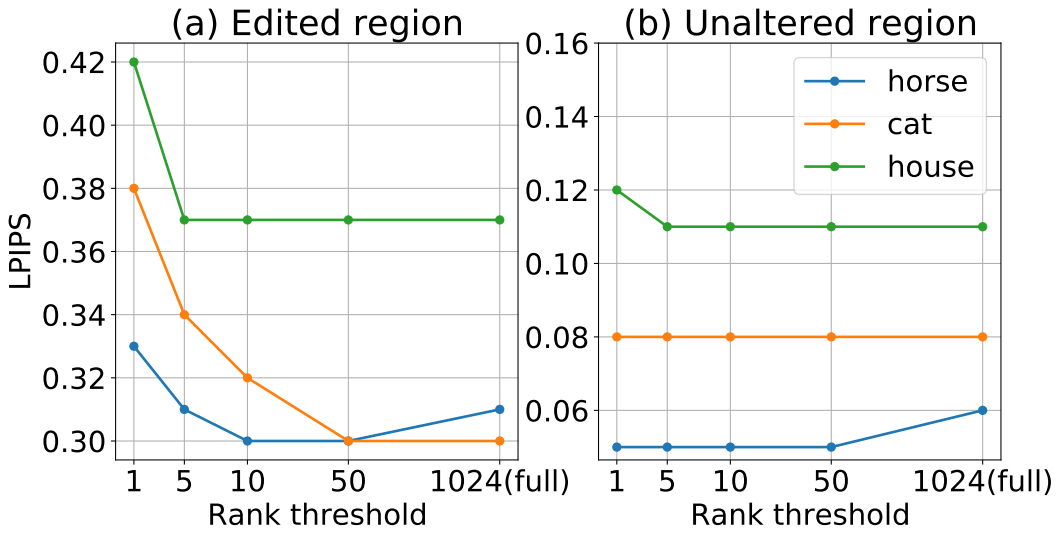


Figure 4.4: **Quantitative analysis on different rank thresholds.** We analyze the low-rank structure of our single-layer update methods by evaluating models trained on different rank thresholds. In all classes of models, we observe that the performance saturates quickly as the rank threshold increases. All models are trained with style-mixing augmentation, and we report performance with LPIPS averaged over each class.


Figure 4.5: **Qualitative analysis on different rank thresholds.** We visually compare the models trained with different rank thresholds. When the rank threshold is too small (rank 1, 5), the model fails to produce sharp changes to the cat ears. However, the results improve quickly as the rank threshold increases, and a desirable effect can be achieved with a rank-50 update.

variations.

**Updating fewer parameters.** We test the effects of updating the full model, a single layer, or a single layer with rank-50 update as described in Section 3.3. Table 4.2 shows that the performances of the three strategies are similar, with the best-performing models being tuned with full weights. Interestingly, models updated with full weights improve more than other models in the edited regions after applying style-mixing augmentation. This suggests that style-mixing augmentation serves as a strong regularization to prevent model overfitting even when updating the full weights, and it avoids the need to update fewer parameters, which is vital for generalization as suggested by previous works [9, 24]. Nonetheless, models updated with fewer parameters enjoy smaller storage of weights, and with a similar performance, smaller storage is favorable when a user creates and stores many models. Hence, we generate qualitative results from the models with low-rank updates.

**Which layer to update.** For single-layer update methods, we select the optimal layer to tune by empirically evaluating on the validation set. We observe that there is a consistent pattern for the optimal layers. In all three source models we tested, we can tune the 2nd convolution layer for coarse geometry changes and the 9th layer for finer, part-based deformations.

Our method offers options for coarse or fine shape changes, which correspond to updating the 2nd or 9th layer, respectively. The user can select the option that achieves the most desirable effect.

**Fewer user edits for training.** Next, we investigate how many user edits are sufficient for each weight update method, as we would like to collect as few user edits as possible. As shown in Figure 4.3, for both full-weight updates and low-rank updates trained with style-mixing augmentation, the performance improves when more examples are added, and the improvement quickly saturates after training with 4 or more examples. This shows that style-mixing augmentation is effective with fewer training samples as well. As a result, we can achieve similar performance with just 4 to 6 user edit examples per warping task.

**Different rank thresholds.** We experimented with different rank thresholds for the low-rank update method, and the results are shown in Figure 4.4. Applying different rank

thresholds mainly affects the edited region, where the performance improves but saturates quickly as the rank threshold increases. This observation shows that we can model the warping edits with a small but sufficient number of parameters. Also, the performance saturates at a different rate for different object categories, but we can pick the rank threshold to be 50 for all models to achieve a good balance. Interestingly, updating with full-rank can sometimes lead to slightly worse performance, as shown in the horse category results. We note that updating extra parameters in a single layer sometimes leads to undesirable changes.

We demonstrate the visual effects of different rank thresholds in Figure 4.5. When the rank threshold is too low, the model cannot model the geometric changes precisely. But the model can quickly achieve the desirable edit effects as the rank threshold increases.

**Sensitivity to user edit variance.** We study how much performance varies if the same user performs editing multiple times. For each task, we evaluate sensitivity by manually creating another set of edits, producing a similar geometry change on the same training set. Models trained on the new edits obtain similar performance to the original ones. Our edited cat models scored 11.83 and 21.09 in PSNR for edited and unaltered regions, respectively; training on the replicated edits gives a similar score (11.83, 20.75). Furthermore, as shown in Table 4.3, we train 10 models on different combinations of user edits by randomly sampling which edit version to use for each sample, and results are consistent and robust: PSNR of the 10 models' performances is 11.83+/-0.16 and 20.9+/-0.15 (mean+/-stdev) for edited and unaltered regions, respectively.

**Sensitivity to random sample selections.** We study our model's sensitivity to different random samples. For each task, we train 10 models to produce the same edits, where each is trained on a different random subset of N samples. As shown in Table 4.4, the variance is small. For cat models, N=5 yields 11.71+/-0.23 and 20.55+/-0.35 (mean+/-stdev) in PSNR for edited and unaltered regions, respectively. N=8 yields 11.82+/-0.15 and 20.87+/-0.17.

## 4.3 Applications

In this section, we discuss several image editing and synthesis applications based on our method. We show that a user can compose multiple models into a new model, generate

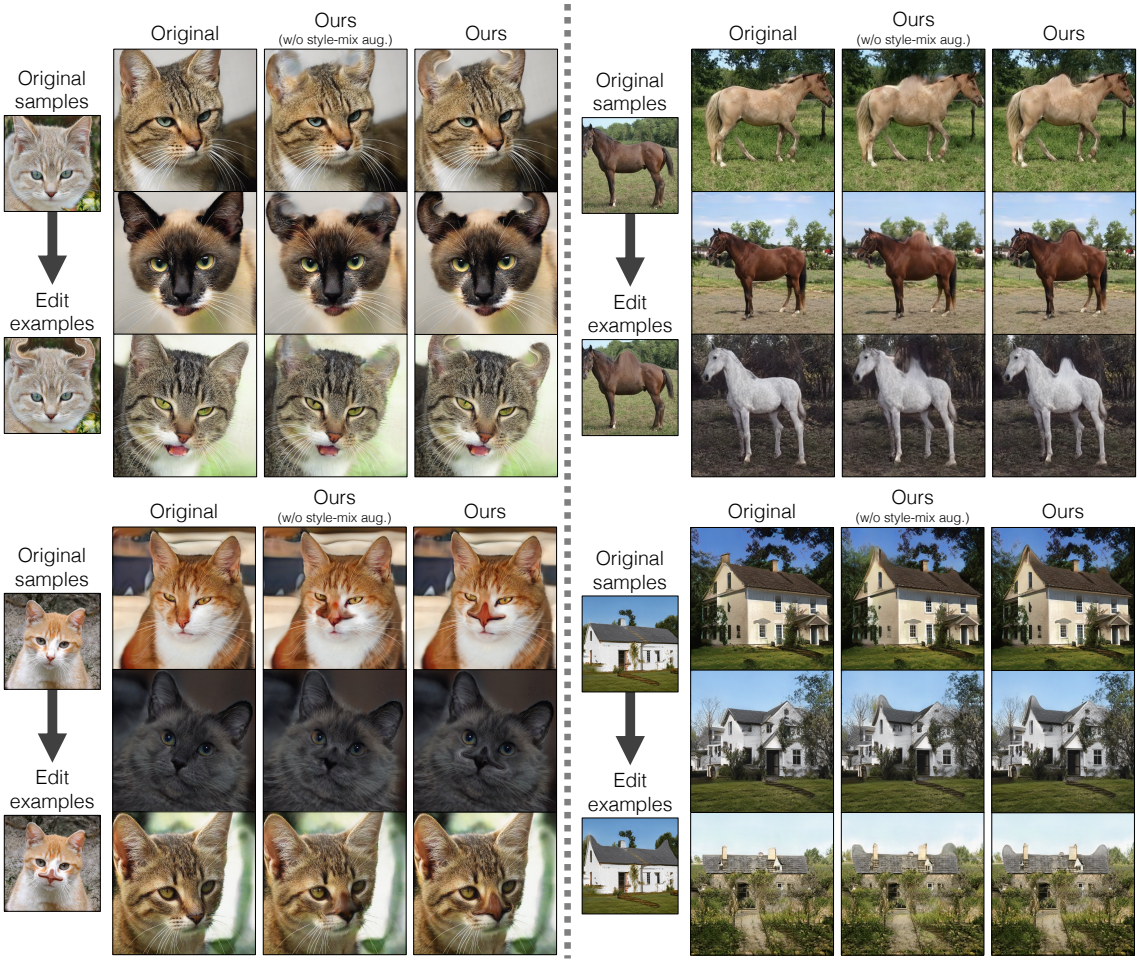


Figure 4.6: **Style-mixing augmentation improves generalization.** We show the effects of style-mixing augmentation in 4 editing tasks. Each block shows one model editing task. For each task, 10 user edit examples are used for training, and one example is shown. We show samples generated from the original models and the edited models trained with or without style-mixing augmentation. The edited models are all trained with the rank-50, single-layer update method, as explained in Section 3.3. For each task, we show samples generated from the same three $z$. We note that models trained without augmentation often generate blurry artifacts in the edited region. In contrast, the models trained with augmentation consistently generate crisper shape changes. The readers are encouraged to zoom in and view the results in more detail.

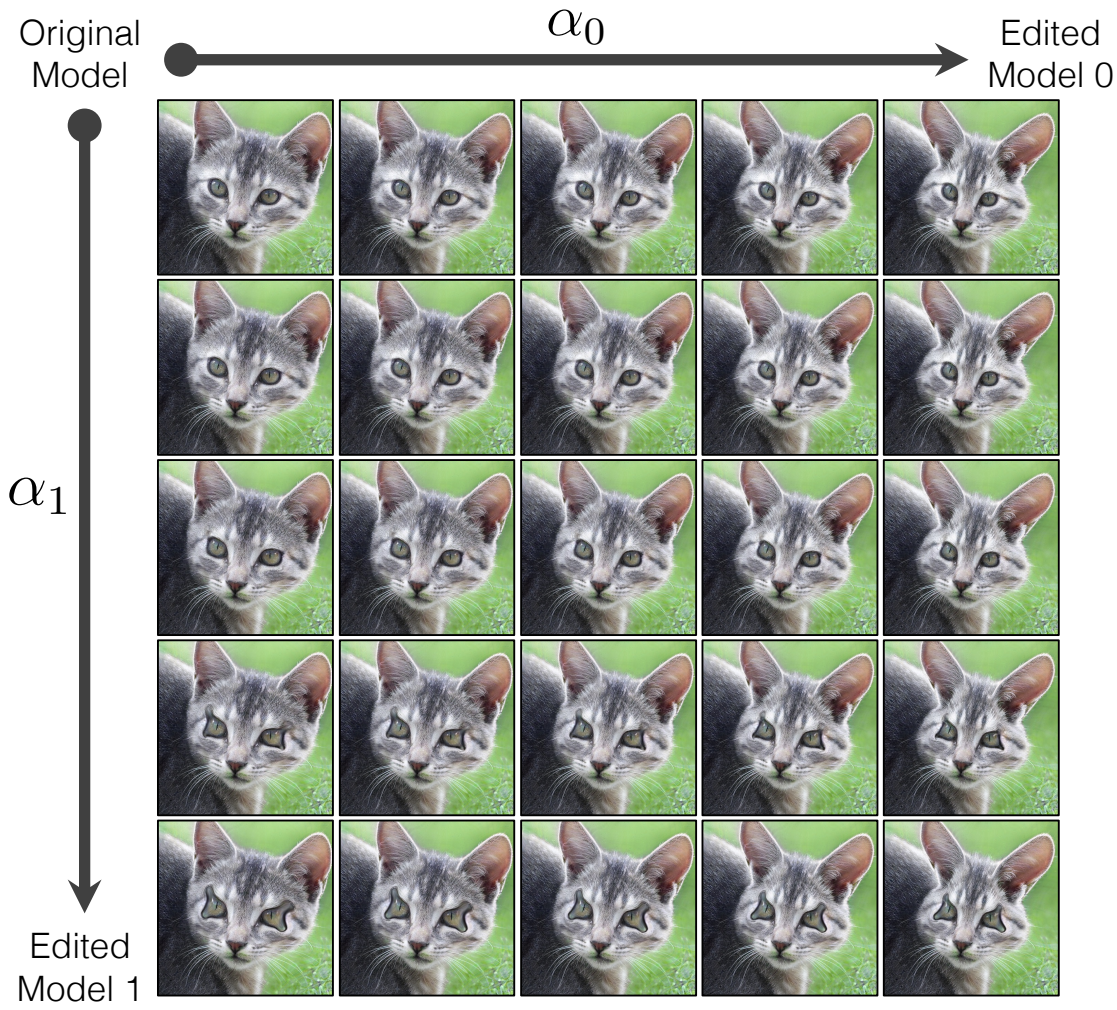


Figure 4.7: **Composing and interpolating two edited models.** We compose two edited models: one model changes the face shape (top right), and the other changes the eye shape (bottom left). The combined model (bottom right) changes the shape of both faces and eyes. One can adjust the blending strength of each model ($\alpha_0, \alpha_1$) to acquire a smooth shape transition.

smooth transitions between images generated by our models, manipulate real photos, and edit models with color strokes.

**Compose edited models.** We can compose different models to achieve aggregated edit effects, based on the approach explained in Section 3.4. Figure 4.7 shows the result of interpolating and composing two models by linearly combining the weights. It is to our surprise that we can achieve smooth shape transitions, even though the newly introduced geometry is not within the original distribution of the source model. Moreover, by applying simple arithmetic to the parameters, we can naturally compose two different out-of-distribution shape changes and synthesize high-quality samples.

Gal et al. gal2021stylegan and Wang et al. wang2021sketch have also observed smooth transitions by interpolating the weights between generative models trained by their method. Our results concur with their findings and further suggest the potential to compose out-of-distribution geometric changes for different models.

**Smooth latent traversals.** Various useful image manipulations can be done with GANs thanks to its smooth latent space representation. As shown in Figure 4.8, our edited models preserve such properties. First, the edited models can generate smooth transitions between two images by interpolating two random latent codes. Moreover, we can apply interpretable controls to the generated samples. In particular, we obtain the controls from the pre-trained source models using the latent discovery method GANSpace [26]. The same controls can be applied to the edited models to achieve the same effects such as changing poses or colors. The results indicate that our edited models preserve many desirable properties in the original latent space.

**Real photo manipulation** We can apply our edited models to manipulate a real photo from the source domain. Specifically, we can transfer the warping edits to the photo and further apply latent manipulations, as shown in Figure 4.9. To achieve this, we project a real photo to a latent code in the source model using StyleGAN2 projection code [34]. We feed the projected latent code to the edited model, which effectively transfers the warping effects to the photo. In addition, we can apply latent manipulations to further edit other aspects of the transformed photo.

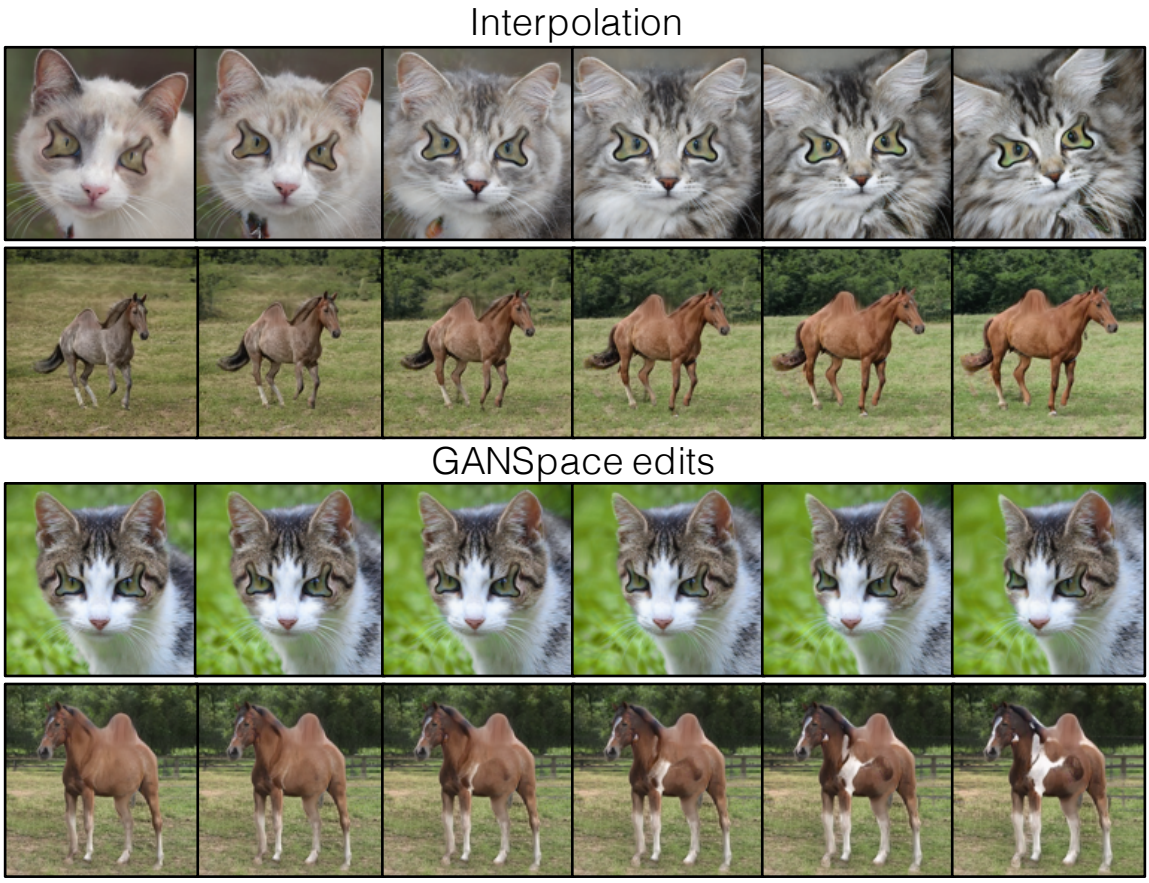


Figure 4.8: **Smooth latent traversal.** Our edited models can generate smooth transitions between two random samples by interpolating the latent space. We can also apply GANSpace edits [26] to our models to change the object attributes such as poses or colors.

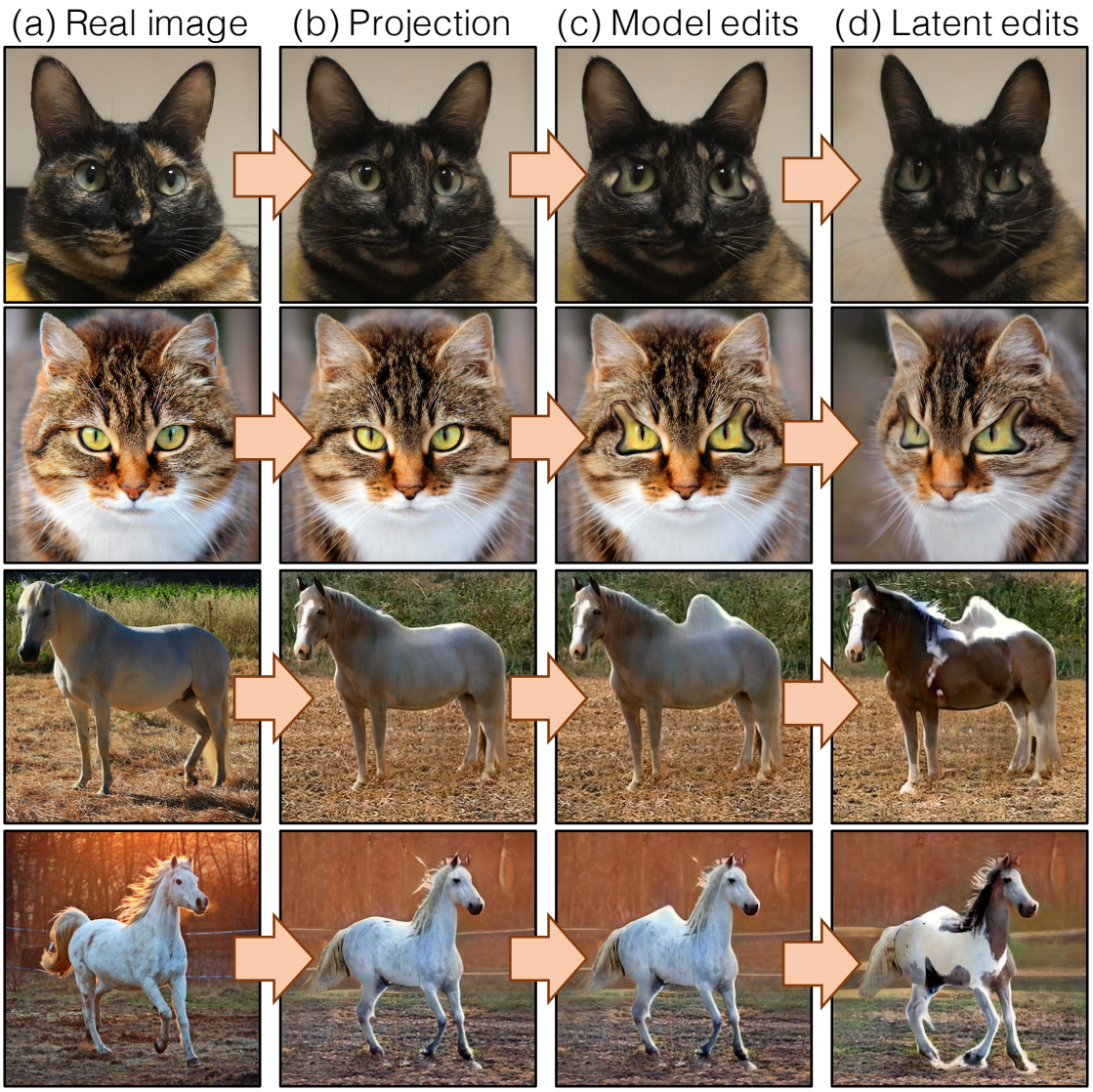


Figure 4.9: **Editing a real photo.** (a) Given a real photo as input, we can edit it in the following steps: (b) we project the image to the latent space of the source model. (c) We feed the projected latent code to an edited model to effectively transfer the warping edits to the photo. (d) We can further apply the GANSpace edits shown in Figure 4.8 to further manipulate the photo. Real images in the 2nd, 3rd and 4th row courtesy of Pixabay users "Ben_Kerckx", "rihaij", and "rauschenberger", respectively.

In Figure 4.9, we project photos of cat faces into the $W+$ space, an extended latent space introduced by Image2StyleGAN [1]. For horses, we project photos into the $W$ space.

**Edit models with color strokes.** As discussed in Section 3.5, we can also edit a model to match the user-specified color edits. As shown in Figure 4.10, the color edits are faithfully transferred to the corresponding parts in the unseen samples. Interestingly, when we change the color of cat eyes to red, the pupil color and the flashes of light on the eye remain intact. Furthermore, color edits can be applied to different models, such as horses and houses. This effect can be attributed to the disentangled representation learned in the source models.

**Additional results.** We show qualitative results on AFHQ dog and wild (Figure 4.11), FFHQ faces (Figure 4.12), and StyleGAN2 [34] (Figure 4.13). All models are trained with ten user edits.


Figure 4.10: **Color edits.** We show results of models edited with coloring operations. The first column shows the user edits. The colored strokes specify the locations to perform coloring changes, while the darker region defines the region to be preserved. The edited models produce precise coloring changes in the specified parts. In the first example (red eye), the pupil color and the flashes of light on the eye are preserved after the edits.

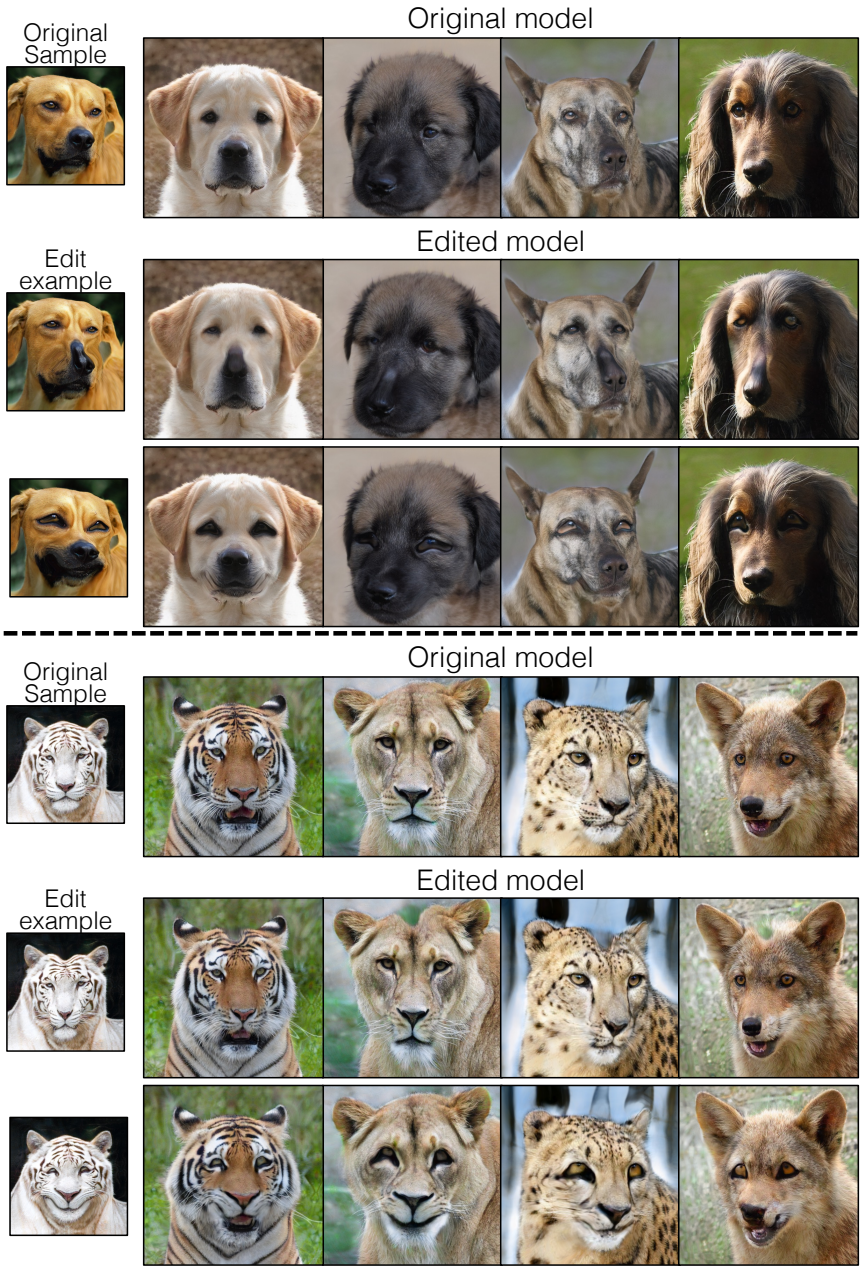


Figure 4.11: **Qualitative results on AFHQ Dogs and Wilds.** Our method can be applied effectively on the AFHQ dog and wild classes.

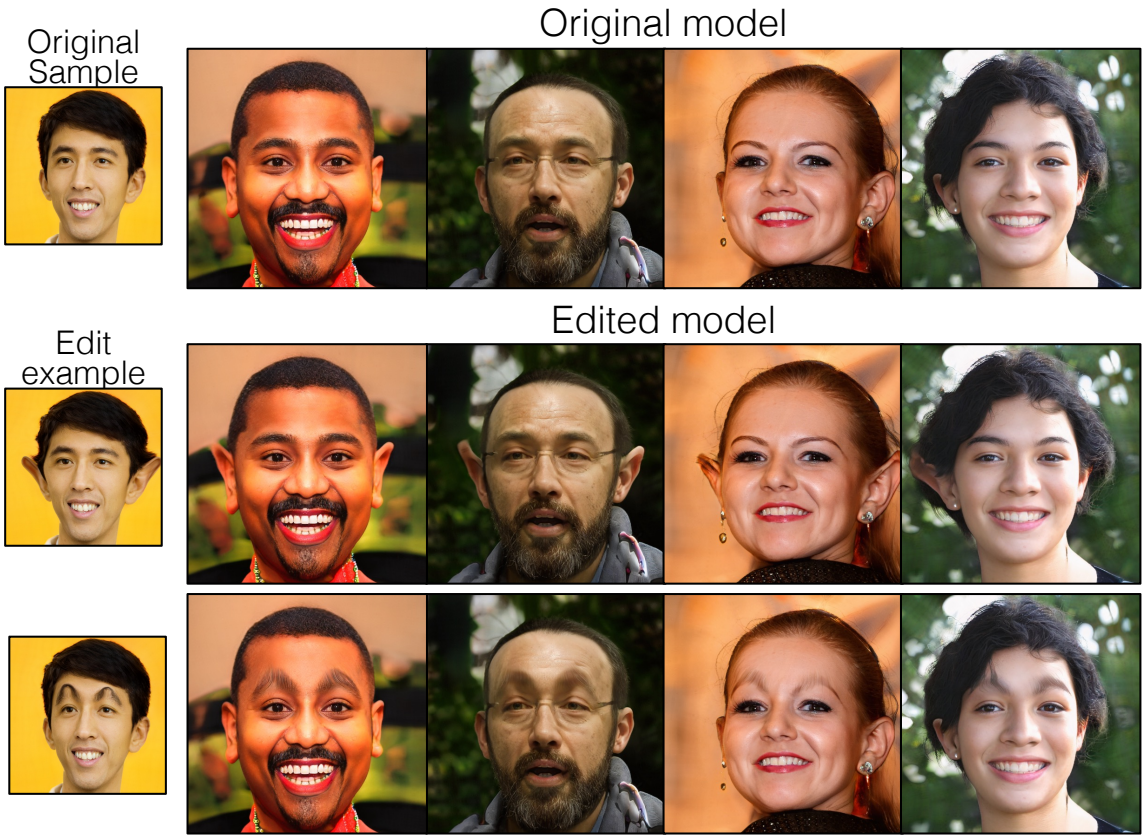


Figure 4.12: **Qualitative results on FFHQ.** Our method can be applied effectively on the 1024px FFHQ faces.

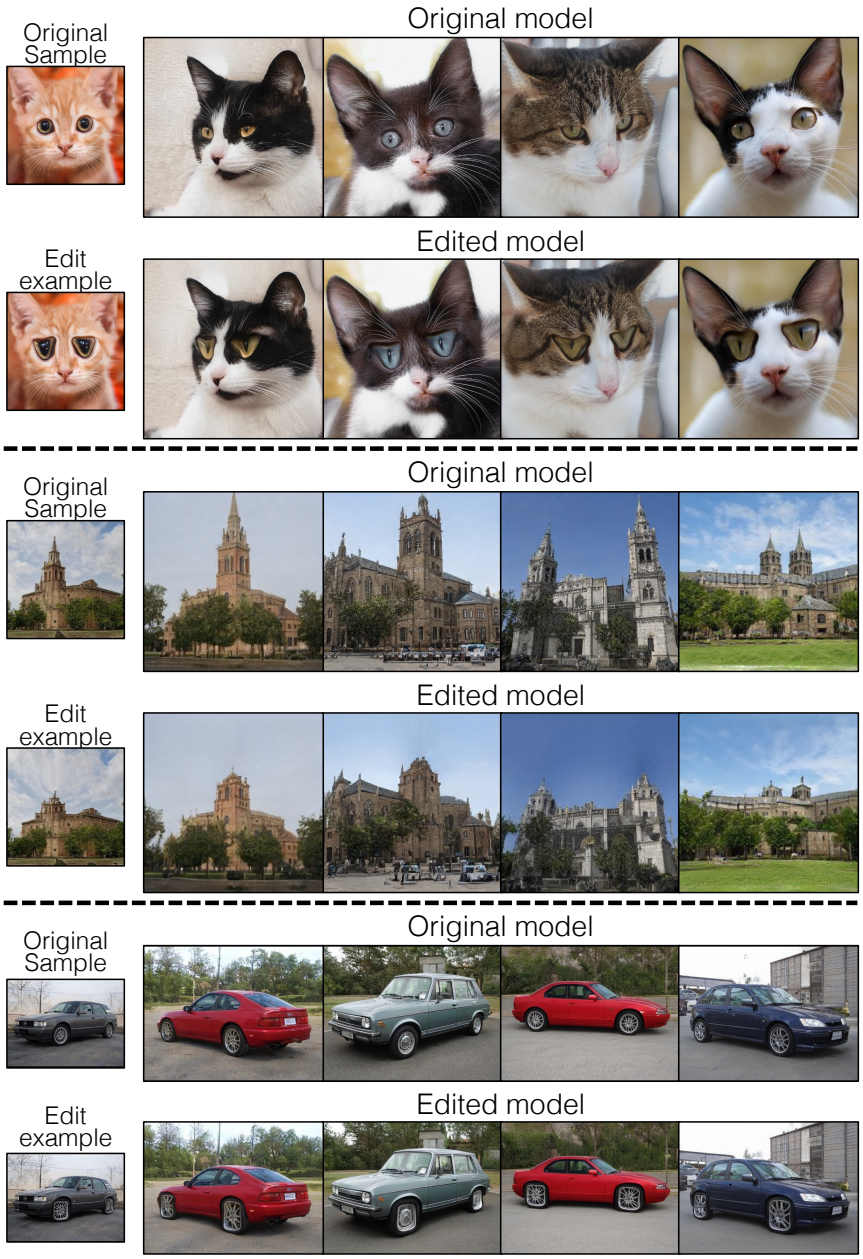


Figure 4.13: **Qualitative results on StyleGAN2 models.** Our method can be applied effectively on StyleGAN2 [34]. From top to bottom, we alter the eye shapes for a AFHQ cat model, lower the tower height for a LSUN church model, and turn the tires into squares for a LSUN car model.

Table 4.1: **Quantitative baseline comparisons.** We compare our rank-50 single layer update method against several baselines. our method outperforms most baselines in both the edited and unaltered regions. While training with extremely few samples, our method performs on par with CATs [16], a supervised dense correspondence method. More details of the baselines and metrics can be found in Section 4.1.

| Class | Name | Edited region | | | | Unaltered Region | | |
|---|---|---|---|---|---|---|---|---|
| | | PSNR (↑) | SSIM (↑) | LPIPS (↓) | CD (↓) | PSNR (↑) | SSIM (↑) | LPIPS (↓) |
| Cat | TGAN | 6.67 | 0.21 | 0.60 | 6.41 | 6.03 | 0.34 | 0.67 |
| | TGAN+ADA | 6.84 | 0.22 | 0.52 | 7.01 | 7.98 | 0.41 | 0.56 |
| | FreezeD | 6.34 | 0.20 | 0.58 | 6.92 | 6.47 | 0.34 | 0.64 |
| | [50] | 7.25 | 0.22 | 0.51 | 7.73 | 7.96 | 0.40 | 0.53 |
| | NADA (text) | 8.67 | 0.24 | 0.62 | 10.98 | 14.32 | 0.50 | 0.28 |
| | NADA (image) | 8.13 | 0.25 | 0.55 | 9.06 | 14.20 | 0.56 | 0.25 |
| | Rewriting (rank 1) | 9.73 | 0.26 | 0.53 | 9.75 | 19.91 | 0.76 | 0.10 |
| | Rewriting (rank 50) | 9.81 | 0.26 | 0.53 | 8.98 | 19.89 | 0.75 | 0.11 |
| | CATs | **12.98** | **0.41** | **0.24** | **3.05** | **23.14** | **0.83** | **0.06** |
| | Ours | 11.83 | 0.36 | 0.30 | 5.07 | 21.09 | 0.77 | 0.08 |
| Horse | TGAN | 8.53 | 0.21 | 0.49 | 8.44 | 8.44 | 0.19 | 0.48 |
| | TGAN+ADA | 8.15 | 0.19 | 0.48 | 6.23 | 8.43 | 0.18 | 0.46 |
| | FreezeD | 7.86 | 0.18 | 0.47 | 7.28 | 8.18 | 0.18 | 0.46 |
| | [50] | 8.98 | 0.17 | 0.45 | 8.88 | 8.84 | 0.19 | 0.41 |
| | NADA (text) | 7.92 | 0.12 | 0.56 | 8.08 | 9.52 | 0.21 | 0.32 |
| | NADA (image) | 9.23 | 0.14 | 0.54 | 8.07 | 11.05 | 0.27 | 0.25 |
| | Rewriting (rank 1) | 8.57 | 0.16 | 0.52 | 7.51 | 20.33 | 0.79 | 0.06 |
| | Rewriting (rank 50) | 8.73 | 0.17 | 0.51 | 9.27 | 19.94 | 0.77 | 0.06 |
| | CATs | 10.24 | 0.24 | 0.41 | 4.46 | 19.75 | 0.78 | 0.07 |
| | Ours | **11.98** | **0.28** | **0.30** | **3.87** | **20.41** | **0.79** | **0.05** |
| House | TGAN | 5.98 | 0.25 | 0.64 | 9.99 | 7.13 | 0.30 | 0.61 |
| | TGAN+ADA | 6.64 | 0.28 | 0.58 | 8.05 | 8.07 | 0.32 | 0.56 |
| | FreezeD | 5.89 | 0.25 | 0.68 | 12.11 | 7.28 | 0.27 | 0.68 |
| | [50] | 6.80 | 0.32 | 0.66 | 13.89 | 8.94 | 0.34 | 0.54 |
| | NADA (text) | 6.75 | 0.28 | 0.60 | 9.88 | 9.21 | 0.27 | 0.45 |
| | NADA (image) | 6.89 | 0.31 | 0.64 | 12.17 | 10.53 | 0.34 | 0.38 |
| | Rewriting (rank 1) | 7.24 | 0.31 | 0.61 | 11.81 | 15.29 | 0.56 | 0.16 |
| | Rewriting (rank 50) | 7.64 | 0.31 | 0.58 | 10.84 | 14.74 | 0.56 | 0.20 |
| | CATs | 9.14 | 0.36 | 0.47 | 5.39 | 15.29 | **0.61** | 0.19 |
| | Ours | **10.04** | **0.37** | **0.37** | **5.19** | **16.45** | **0.60** | **0.11** |

Table 4.2: **Ablation studies.** We evaluate the effects of style-mixing augmentation **(style-mix aug.)** and the differences between full-weight updates **(Full)**, single-layer updates **(Layer)**, and single-layer updates with a rank-50 threshold **(Layer (r50))**. Note that style-mixing augmentation consistently improves all cases. The full-weight update method achieves the best performance, while single-layer methods obtain comparable performance with less demanding storage requirements.

| Class | Name | Edited region | | | | Unaltered Region | | |
|---|---|---|---|---|---|---|---|---|
| | | PSNR (↑) | SSIM (↑) | LPIPS (↓) | CD (↓) | PSNR (↑) | SSIM (↑) | LPIPS (↓) |
| Cat | Full | 11.60 | 0.37 | 0.40 | 7.61 | 21.89 | **0.80** | 0.08 |
| | Full + style-mix aug. | **12.56** | **0.40** | **0.28** | **4.49** | **22.07** | **0.80** | **0.07** |
| | Layer | 11.56 | 0.35 | 0.39 | 6.40 | 19.46 | 0.73 | 0.11 |
| | Layer + style-mix aug. | 11.90 | 0.36 | 0.30 | 5.18 | 21.04 | 0.77 | 0.08 |
| | Layer (r50) | 11.50 | 0.36 | 0.35 | 6.18 | 19.01 | 0.70 | 0.12 |
| | Layer (r50) + style-mix aug. | 11.83 | 0.36 | 0.30 | 5.07 | 21.09 | 0.77 | 0.08 |
| Horse | Full | 10.09 | 0.26 | 0.43 | 5.28 | 19.45 | 0.74 | 0.08 |
| | Full + style-mix aug. | **12.55** | **0.31** | **0.25** | **3.23** | 19.05 | 0.71 | 0.06 |
| | Layer | 10.33 | 0.26 | 0.39 | 4.09 | 18.09 | 0.69 | 0.10 |
| | Layer + style-mix aug. | 11.81 | 0.27 | 0.31 | 4.06 | 19.96 | 0.77 | 0.06 |
| | Layer (r50) | 10.36 | 0.26 | 0.37 | 4.86 | 16.99 | 0.63 | 0.12 |
| | Layer (r50) + style-mix aug. | 11.98 | 0.28 | 0.30 | 3.87 | **20.41** | **0.79** | **0.05** |
| House | Full | 8.99 | 0.37 | 0.50 | 6.71 | 17.23 | **0.64** | 0.15 |
| | Full + style-mix aug. | **10.55** | **0.39** | **0.36** | **4.63** | **17.27** | **0.64** | **0.11** |
| | Layer | 9.64 | 0.37 | 0.41 | 5.08 | 16.01 | 0.59 | 0.13 |
| | Layer + style-mix aug. | 10.00 | 0.37 | 0.37 | 5.16 | 16.51 | 0.60 | **0.11** |
| | Layer (r50) | 10.04 | 0.37 | 0.39 | 5.13 | 15.40 | 0.56 | 0.14 |
| | Layer (r50) + style-mix aug. | 10.04 | 0.37 | 0.37 | 5.19 | 16.45 | 0.60 | **0.11** |

Table 4.3: **Sensitivity to user edit variance.** For every task, we manually create another set of edits, producing a similar geometry change on the same training set. We train 10 models on different combinations of user edits by randomly sampling which edit version to use for each sample. We report mean ± stdev over the 10 models. We find that the performance is not sensitive to the user edit variance.

| Class | Edited region | | | | Unaltered region | | |
|---|---|---|---|---|---|---|---|
| | PSNR | SSIM | LPIPS | Chamfer | PSNR | SSIM | LPIPS |
| Cat | 11.83±0.16 | 0.36±0.01 | 0.31±0.01 | 5.19±0.30 | 20.91±0.15 | 0.77±0.00 | 0.08±0.00 |
| Horse | 11.90±0.15 | 0.28±0.01 | 0.30±0.01 | 3.99±0.43 | 20.62±0.10 | 0.79±0.00 | 0.05±0.00 |
| House | 10.17±0.09 | 0.37±0.00 | 0.36±0.01 | 4.93±0.26 | 16.37±0.09 | 0.60±0.00 | 0.11±0.00 |

Table 4.4: **Sensitivity to random sample selections.** For every task, we evaluate sensitivity to sample selection on 10 models, where each is trained on a different random subset of $N$ samples. We report mean $\pm$ stdev over 10 models. We find that the performance does not vary much with different random samples.

| Class | $N$ | Edited region | | | | Unaltered region | | |
|---|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS | Chamfer | PSNR | SSIM | LPIPS |
| Cat | 5 | 11.71±0.23 | 0.36±0.01 | 0.31±0.01 | 5.22±0.40 | 20.55±0.35 | 0.75±0.01 | 0.09±0.01 |
| | 8 | 11.82±0.15 | 0.36±0.01 | 0.31±0.01 | 5.29±0.34 | 20.87±0.17 | 0.77±0.00 | 0.09±0.00 |
| Horse | 5 | 11.75±0.43 | 0.28±0.02 | 0.31±0.02 | 3.75±0.80 | 19.92±0.35 | 0.77±0.02 | 0.06±0.00 |
| | 8 | 11.95±0.28 | 0.28±0.01 | 0.30±0.01 | 3.91±0.79 | 20.26±0.18 | 0.78±0.01 | 0.06±0.00 |
| House | 5 | 9.65±0.36 | 0.35±0.01 | 0.39±0.02 | 5.43±0.81 | 16.06±0.30 | 0.59±0.01 | 0.13±0.01 |
| | 8 | 9.89±0.12 | 0.36±0.01 | 0.38±0.01 | 5.14±0.39 | 16.40±0.18 | 0.61±0.01 | 0.12±0.00 |

# Chapter 5

# Discussion and Limitations

We have presented a method for altering the local geometric rules of a pre-trained GAN model. With our method, a user can provide just a handful of control points as warping instructions to edit a pre-trained model. The edited model can faithfully synthesize endless samples that follow the desired geometric changes. Surprisingly, the geometric changes do not need to be within the original data distribution, so users have the freedom to customize their models and introduce a wide range of shapes. In addition, our models with different shape edits can be composed together to achieve the new effects, and we present an interactive interface for users to create new models by composing a preset of edited models.

## 5.1 Limitations

One natural question is to understand the range of geometric changes where we can successfully apply our method. In Figure 5.1, we present several test cases to study this. In the first example, we train a model on ten warping examples where only the left ear is warped, and we find that the resulting model cannot produce the desired shape changes. In the second example, we train a model on the same set of examples, but this time the supervision only comes from the left halves of the images. This is achieved by masking out the right halves for both the model output and training images when calculating the loss. Interestingly, without seeing any examples of right ears, the resulting model applies the geometric change to the right ears as well. This suggests that the pre-trained cat model has a symmetry bias, so it is difficult to perform asymmetric shape changes to the edited

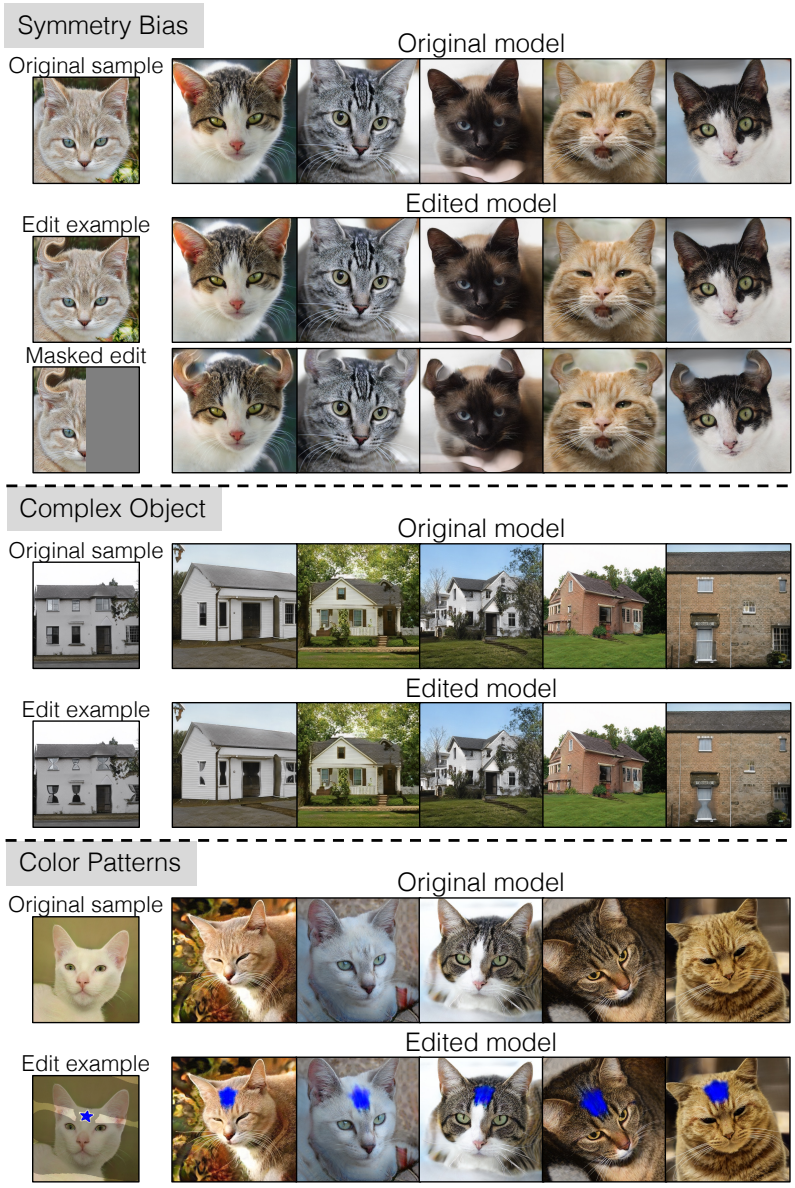


Figure 5.1: **Failure cases.** (Top) Our method cannot apply asymmetric shape edits to a cat model, such as warping just a cat's left ear. Next, we mask out the right halves of the same images to train the model. The second model does change the shape of both ears, without seeing any right ears. (Middle) Our method cannot edit a house model to produce consistent shape changes to the windows. We observe that not every windows in the generated samples undergo the desired shape changes. (Bottom) Our method does not work well with inserting new objects, such as a "star sticker" onto cats' foreheads. Although the stars in the edited models do appear at the correct location, the results are blurry.

models.

Moreover, in the house example, we test if we can warp the windows into the shape of hourglasses in the house model, and we observe that only a small number of windows change their shape accordingly in the edited model, potentially due to the large style and shape variations of windows. Also, we find that our method fails to add new objects. We experimented with adding a "star sticker" onto cats' foreheads. The stars appear at the correct locations in the edited model, but the results are blurry.

Additionally, our work focuses on keypoint-based warping. It would be great to support other types of warping algorithms such as mesh-based Liquify [36] and line-based methods [10], as well as semi-automatic methods [3]. Finally, we would like to further reduce our current training time (20-40 minutes). One possible solution is to pre-compute a shared warping basis in a certain layer that could work for different types of geometric changes.

## 5.2 Societal Impact

While our work makes it easier for users to create generative models and image content, potential misuse of this method unfortunately exists. For instance, a malicious user could apply facial expression changes to a face generative model to spread misinformation. These manipulations could change the narrative, for example editing a frowning political figure to be smiling, and it is important to develop methods to counter such abuses. One potential solution is to watermark the content whenever it is generated using this method. Another way is to train a classifier to detect whether an image is generated by a generative model or not [21, 63, 76, 85]. We test a standard CNN-detection method [76] on our warped cat models along with real AFHQv2 cat images. The classifier gets 98.3% average precision but 59.58% accuracy. As suggested by [76], the classifier trained only with ProGAN [31] images obtains correct rankings of real-verse-fake, but it would require calibration using a few samples. A classifier that achieves high fake-detection accuracy for new, unseen models remains an open challenge.

# Appendix A

# Additional Analysis

**Few-shot GANs trained with style-mixing augmentation.** We test several few-shot GAN methods trained on the edited examples with style-mixing augmentation applied, and results are shown in Table A.1. Comparing to Table 1 in the main text, we find that style-mixing augmentation improves most baselines, but the performance remains significantly inferior to our full method. This is because that few-shot GANs do not explicitly enforce correspondence between original samples and edited samples, even if style-mixing augmentation improves the diversity of the training set.

**Runtime comparison to StyleGAN-NADA.** According to Gal *et al*. gal2021stylegan, StyleGAN-NADA's training time depends on the edit types. While texture-based changes take 1-3 minutes on a V100, complex shape edits can take up to 6 hours.

Further, we observe that training *NADA (image)* and *NADA (text)* with longer iterations does not improve the performance in all of our tasks. We train StyleGAN-NADA with 10x more iterations than the default iteration counts used in our paper, 30 minutes on an A5000. For cat models, when trained longer, *NADA (image)* gets 6.60 and 9.11 in PSNR for edited and unaltered regions, respectively. This is worse than *NADA (image)*'s performance reported in the paper (8.13, 14.20), and our method's performance (11.83, 21.09).

**Additional qualitative results.** Our geometric edits generalize well to new latent codes with color, texture, shape, and pose variations. As shown in Figure A.1, we tested our method on latent codes that represent new shapes – a sample projected from a real Sphynx cat image. Our method succeeded in changing the eyes and ears of the Sphynx cat and producing smooth interpolations between the Sphynx cat latent code and other latent codes.

Table A.1: **Few-shot GANs trained with style-mixing augmentation.** We compare our rank-50 single layer update method against few-shot GANs trained with style-mixing augmentation. We find that our method still outperforms significantly.

| Class | Name | Edited Region | | | | Unaltered region | | |
|---|---|---|---|---|---|---|---|---|
| | | PSNR (↑) | SSIM (↑) | LPIPS (↓) | Chamfer (↓) | PSNR (↑) | SSIM (↑) | LPIPS (↓) |
| Cat | TGAN | 9.06 | 0.25 | 0.47 | 6.69 | 9.81 | 0.47 | 0.49 |
| | TGAN+ADA | 9.58 | 0.26 | 0.41 | 5.84 | 10.81 | 0.46 | 0.42 |
| | FreezeD | 9.29 | 0.26 | 0.45 | 6.73 | 9.99 | 0.48 | 0.48 |
| | [50] | 8.73 | 0.24 | 0.47 | 7.32 | 10.27 | 0.44 | 0.43 |
| | Ours | **11.83** | **0.36** | **0.30** | **5.07** | **21.09** | **0.77** | **0.08** |
| Horse | TGAN | 9.14 | 0.20 | 0.41 | 10.19 | 9.36 | 0.19 | 0.40 |
| | TGAN+ADA | 9.30 | 0.18 | 0.41 | 10.49 | 9.54 | 0.19 | 0.39 |
| | FreezeD | 9.78 | 0.21 | 0.40 | 7.48 | 9.57 | 0.19 | 0.40 |
| | [50] | 7.95 | 0.16 | 0.48 | 5.86 | 9.44 | 0.20 | 0.39 |
| | Ours | **11.98** | **0.28** | **0.30** | **3.87** | **20.41** | **0.79** | **0.05** |
| House | TGAN | 6.60 | 0.28 | 0.59 | 10.05 | 7.73 | 0.32 | 0.54 |
| | TGAN+ADA | 6.86 | 0.29 | 0.58 | 9.68 | 8.37 | 0.33 | 0.51 |
| | FreezeD | 6.78 | 0.26 | 0.57 | 8.79 | 7.41 | 0.29 | 0.52 |
| | [50] | 5.44 | 0.30 | 0.70 | 11.71 | 8.45 | 0.34 | 0.53 |
| | Ours | **10.04** | **0.37** | **0.37** | **5.19** | **16.45** | **0.60** | **0.11** |

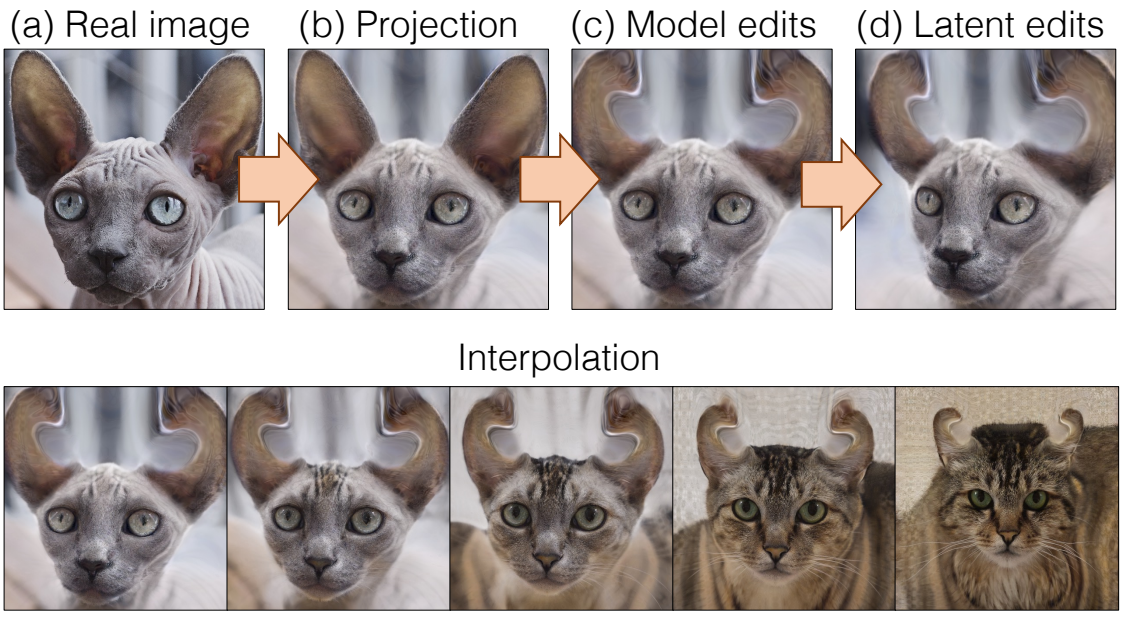


Figure A.1: **Sphynx cat example.** We show that our method can be applied to a Sphynx cat, which has a unique color and shape. (Top) the cat image can be edited the same way as Figure 13 in the main text. (Bottom) The projected cat sample interpolates smoothly to a random sample using our model.

We also include additional model warping results in Figure B.1.

# Appendix B

# Implementation Details

**Details of the Chamfer Distance metric.** For each pair of hold out ground truth edit example and edited model's output, we predict the contour of both images using Dexined pre-trained on BIPEDv2 [70]. The model can predict detailed contours, which helps us better evaluate fine-grained parts like eyes or noses. We then post-process the contour following Isola et al. isola2017image to obtain a single-pixel-wide contour map. Finally, we calculate the symmetric Chamfer distance on the edited region.

**Text prompts used for StyleGAN-NADA.** Table B.1 shows the text prompts used to evaluate the baseline *NADA (text)*. StyleGAN-NADA [24] finetunes a GAN using a source-to-target text prompt. The GAN is fine-tuned using the difference in the CLIP embedding between the text pair.

**Model Rewriting implementation.** To apply Model Rewriting [9] to our warping tasks, for each training sample, we manually mask the region that will be warped. The masks are used to compute and fix the context key for the proposed rank-1 update method. We optimize for the reconstruction loss between the original and edited samples to obtain the edited model. We also test a variation of the baseline using rank-50 updates.

**CATs implementation.** We apply CATs [16] to predict the correspondence between a given test sample and the 10 training samples. For each training sample, we transform the warping field using the estimated correspondence. In total, we get 10 transformed warping fields for each test sample, and we use the averaged warping field to warp the test sample.

**Hyperparameter details** We show the hyperparameters of warped StyleGAN3 models, color-edited StyleGAN3 models, and warped StyleGAN2 models in Table B.2, Table B.3,

Table B.1: **StyleGAN-NADA text prompts.** The prompts are used to evaluate one of the baselines (**NADA(Text)**). We note that StyleGAN-NADA [24] uses a source-to-target text prompt pair to update a generative model.

| Source | Target |
|---|---|
| shape of a cat face | shape of a rabbit face |
| shape of cat ears | shape of devil ears |
| shape of cat eyes | shape of alien eyes |
| normal-shaped cat nose | cross-shaped cat nose |
| shape of a horse back | shape of a camel back |
| a horse standing | a horse with a lifted front leg |
| shape of a house roof | shape of a curly house roof |
| shape of a normal ground | shape of a risen ground |

and Table B.4, respectively.

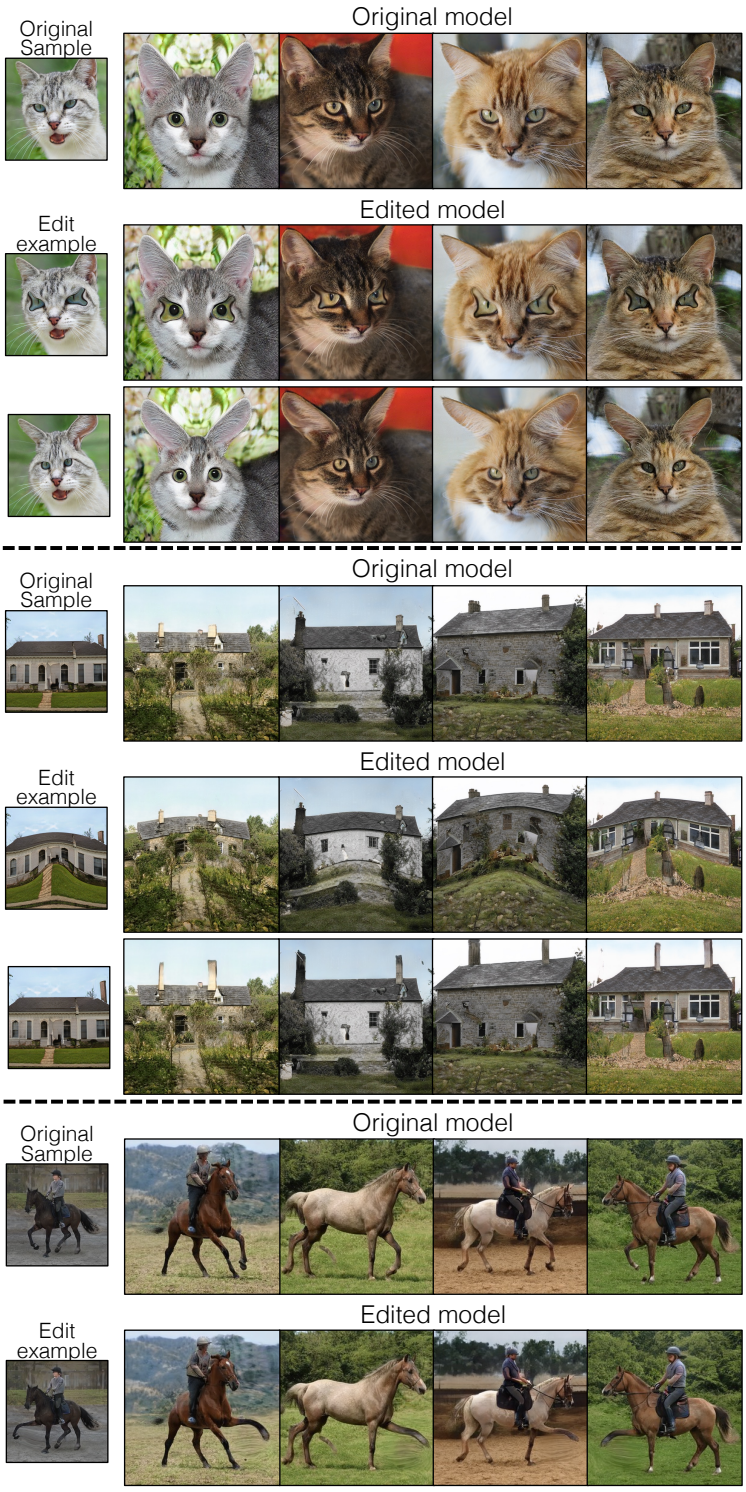


Figure B.1: **Extra qualitative results.** We show more model warping results on StyleGAN3 cat, house, and horse models.

Table B.2: **Warped StyleGAN3 hyperparameters.** We show the layer used for training (Layer), learning rate (Lr), and whether it is evaluated quantitatively in the paper (Evaluated). Default hyperparameters are used if not mentioned.

| Class | Warp type | Details | | |
|---|---|---|---|---|
| | | Layer | Lr | Evaluated |
| Cat | Devilish ears | 8 | 0.05 | ✓ |
| | Cross-shaped noses | 8 | 0.05 | ✓ |
| | Alien eyes | 8 | 0.05 | ✓ |
| | Rabbit faces | 1 | 0.05 | ✓ |
| Horse | Camel backs | 8 | 0.05 | ✓ |
| | Lifted front leg | 8 | 0.05 | ✓ |
| House | Curly roofs | 8 | 0.05 | ✓ |
| | Heaved ground | 1 | 0.05 | ✓ |
| | Taller chimneys | 8 | 0.05 | |
| FFHQ | Elf ears | 5 | 0.01 | |
| | Arched eyebrows | 6 | 0.01 | |

Table B.3: **Color-edited StyleGAN3 hyperparameters.** We show the layer used for training (Layer) and the weight of the color loss ($\lambda_{\text{color}}$). Default hyperparameters are used if not mentioned.

| Class | Color-edit type | Details | |
|---|---|---|---|
| | | Layer | $\lambda_{\text{color}}$ |
| Cat | Red eyes | 13 | 8 |
| | Golden noses | 13 | 8 |
| | Blue star | 10 | 8 |
| Horse | Purple fur | 14 | 1 |
| House | Yellow sky | 13 | 8 |

Table B.4: **Warped StyleGAN2 hyperparameters.** We show the layer used for training (Layer) and the learning rate (Lr). Default hyperparameters are used if not mentioned.

| Class | Warp type | Details | |
|---|---|---|---|
| | | Layer | Lr |
| AFHQ Cat | Big triangular eyes | 8 | 0.05 |
| LSUN Church | Shorter towers | 4 | 0.05 |
| LSUN Car | Square-shaped tires | 8 | 0.05 |

# Bibliography


[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3.1, 4.3

[2] Rameen Abdal, Peihao Zhu, Niloy J Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (TOG)*, 2021. 2.1

[3] Kfir Aberman, Jing Liao, Mingyi Shi, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Neural best-buddies: Sparse cross-domain correspondence. *ACM Transactions on Graphics (TOG)*, 2018. 5.1

[4] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Only a matter of style: Age transformation using a style-based regression model. *ACM Transactions on Graphics (TOG)*, 2021. 2.1

[5] Badour Albahar, Jingwan Lu, Jimei Yang, Zhixin Shu, Eli Shechtman, and Jia-Bin Huang. Pose with Style: Detail-preserving pose-guided image synthesis with conditional stylegan. *ACM Transactions on Graphics (TOG)*, 2021. 2.1

[6] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. pages 157–164, 2000. 2.3, 3.1

[7] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM SIGGRAPH*, 2009. 2.3

[8] Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen C Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical report, SRI International Artificial Intelligence Center, 1977. 4.1

[9] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *European Conference on Computer Vision (ECCV)*, 2020. (document), 2.2, 3.3, 3.3, 4.1, 4.1, 4.2, B

[10] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *ACM Transactions on Graphics (TOG)*, 1992. 5.1

[11] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 2.1, 3.2

[12] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision (IJCV)*, 2007. 2.3

[13] James Cameron and Jon Landau. Avatar, 2009. 1

[14] Kaidi Cao, Jing Liao, and Lu Yuan. Carigans: Unpaired photo-to-caricature translation. *ACM Transactions on Graphics (TOG)*, 2018. 2.3

[15] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 2.1

[16] Seokju Cho, Sunghwan Hong, Sangryul Jeon, Yunsung Lee, Kwanghoon Sohn, and Seungryong Kim. Cats: Cost aggregation transformers for visual correspondence. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (document), 4.1, 4.1, B

[17] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 4.1

[18] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations (ICLR)*, 2017. 2.1

[19] Ian Failes. *Masters of FX: Behind the Scenes with Geniuses of Visual and Special Effects*. CRC Press, 2016. 1

[20] Martin A Fischler and Robert C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 2.3

[21] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *International Conference on Machine Learning (ICML)*, 2020. 5.2

[22] Raghudeep Gadde, Qianli Feng, and Aleix M. Martinez. Detail me more: Improving gan's photo-realism of complex scenes. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2.2

[23] Rinon Gal, Dana Cohen, Amit Bermano, and Daniel Cohen-Or. Swagan: A style-based wavelet-driven generative model. *ACM Transactions on Graphics (TOG)*, 2021. 2.1

[24] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 2022. (document), 2.2, 3.3, 4.1, 4.2, 4.1, 4.2, B, B.1

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley,

Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 2.1

[26] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. (document), 3.1, 4.3, 4.8

[27] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2.1

[28] Minyoung Huh, Jun-Yan Zhu Richard Zhang, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images to class-conditional generative networks. In *European Conference on Computer Vision (ECCV)*, 2020. 3.1

[29] Takeo Igarashi, Tomer Moscovich, and John F Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (TOG)*, 2005. 2.3, 3.1

[30] Wonjong Jang, Gwangjin Ju, Yucheol Jung, Jiaolong Yang, Xin Tong, and Seungyong Lee. Stylecarigan: Caricature generation via stylegan feature map modulation. *ACM Transactions on Graphics (TOG)*, 2021. 2.3

[31] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 5.2

[32] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2.1, 3.2

[33] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2.2, 3.1, 4.1

[34] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (document), 4.3, 4.3, 4.13

[35] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2.1, 3, 3.2, 3.3, 4

[36] Byungmoon Kim, Daichi Ito, and Gahye Park. Facial feature liquifying using face mesh, March 5 2019. US Patent 10,223,767. 2.3, 5.1

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 4

[38] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2.1

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014. 2.1

[40] Nupur Kumari, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Ensembling off-the-shelf models for gan training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2.2

[41] Kathleen M Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. Try-on-gan: Body-aware try-on via layered interpolation. *ACM Transactions on Graphics (TOG)*, 2021. 2.1

[42] Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2.2

[43] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Transactions on Graphics (TOG)*, 36(4), July 2017. 2.3

[44] Huan Ling, Karsten Kreis, Daiqing Li, Seung Wook Kim, Antonio Torralba, and Sanja Fidler. Editgan: High-precision semantic image editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2.2

[45] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010. 2.3

[46] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1981. 2.3

[47] George Lucas and Gary Kurtz. Star wars, 1977. 1

[48] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. In *CVPR Workshop*, 2020. 2.2, 4.1

[49] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2.2

[50] Utkarsh Ojha, Yijun Li, Cynthia Lu, Alexei A. Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2.2, 3.1, 4.1, A.1

[51] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 2.1

[52] Roy Or-El, Soumyadip Sengupta, Ohad Fried, Eli Shechtman, and Ira Kemelmacher-

Shlizerman. Lifespan age transformation synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. 2.1

[53] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. In *European Conference on Computer Vision (ECCV)*, 2020. 3.3

[54] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2.1

[55] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2.2

[56] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 1, 2.1

[57] Tiziano Portenier, Qiyang Hu, Attila Szabó, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing. *ACM Transactions on Graphics (TOG)*, 37(4), 2018. 2.1

[58] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. 4.1

[59] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning (ICML)*, 2021. 1, 2.1

[60] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2.1

[61] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3.1

[62] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2.1

[63] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies,

and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 5.2

[64] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2.2

[65] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2.2

[66] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Transactions on Graphics (TOG)*, 25(3), 2006. 2.3, 3.1

[67] Deb Debayan Shi, Yichun and Anil K. Jain. Warpgan: Automatic caricature generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2.3

[68] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):200, 2013. 2.3

[69] YiChang Shih, Sylvain Paris, Connelly Barnes, William T Freeman, and Frédo Durand. Style transfer for headshot portraits. *ACM Transactions on Graphics (TOG)*, 2014. 2.3

[70] X. Soria, E. Riba, and A. Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *Winter Conference on Applications of Computer Vision*, 2020. 4.1, B

[71] Diana Sungatullina, Egor Zakharov, Dmitry Ulyanov, and Victor S. Lempitsky. Image manipulation with perceptual discriminators. In *European Conference on Computer Vision (ECCV)*, 2021. 2.2

[72] Zhentao Tan, Menglei Chai, Dongdong Chen, Jing Liao, Qi Chu, Lu Yuan, Sergey Tulyakov, and Nenghai Yu. Michigan: Multi-input-conditioned hair image generation for portrait editing. *arXiv preprint arXiv:2010.16417*, 2020. 2.1

[73] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 2021. 2.1

[74] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. Towards good practices for data augmentation in gan training. *arXiv preprint arXiv:2006.05338*, 2, 2020. 2.2

[75] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularing generative adversarial networks under limited data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2.2

[76] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. Cnn-generated images are surprisingly easy to spot... for now. In *IEEE Conference on*

*Computer Vision and Pattern Recognition (CVPR)*, 2020. 5.2

[77] Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Sketch your own gan. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2.2, 3.3

[78] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2.1

[79] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *European Conference on Computer Vision (ECCV)*, 2018. 2.2, 3.1, 4.1

[80] Yaxing Wang, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. Minegan: Effective knowledge transfer from gans to target domains with few images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2.2

[81] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004. 4.1

[82] Simon N Wood. Thin plate regression splines. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2003. 2.3, 3.1

[83] 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. 4.1

[84] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (document), 3.1, 3.1, 4.1

[85] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2019. 5.2

[86] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image {gan}s meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. In *International Conference on Learning Representations (ICLR)*, 2021. 2.2

[87] Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning (ICML)*, 2020. 2.2

[88] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *Advances in Neural Information*

*Processing Systems (NeurIPS)*, 2020. 1, 2.2, 3.1

[89] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for gan training. *arXiv preprint arXiv:2006.02595*, 2020. 2.2

[90] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 4.1

[91] 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 (ECCV)*, 2016. 1, 2.1, 3.1