Sketch Your Own GAN

Sheng-Yu Wang[1] David Bau[2] Jun-Yan Zhu[1]
[1]Carnegie Mellon University [2]MIT CSAIL

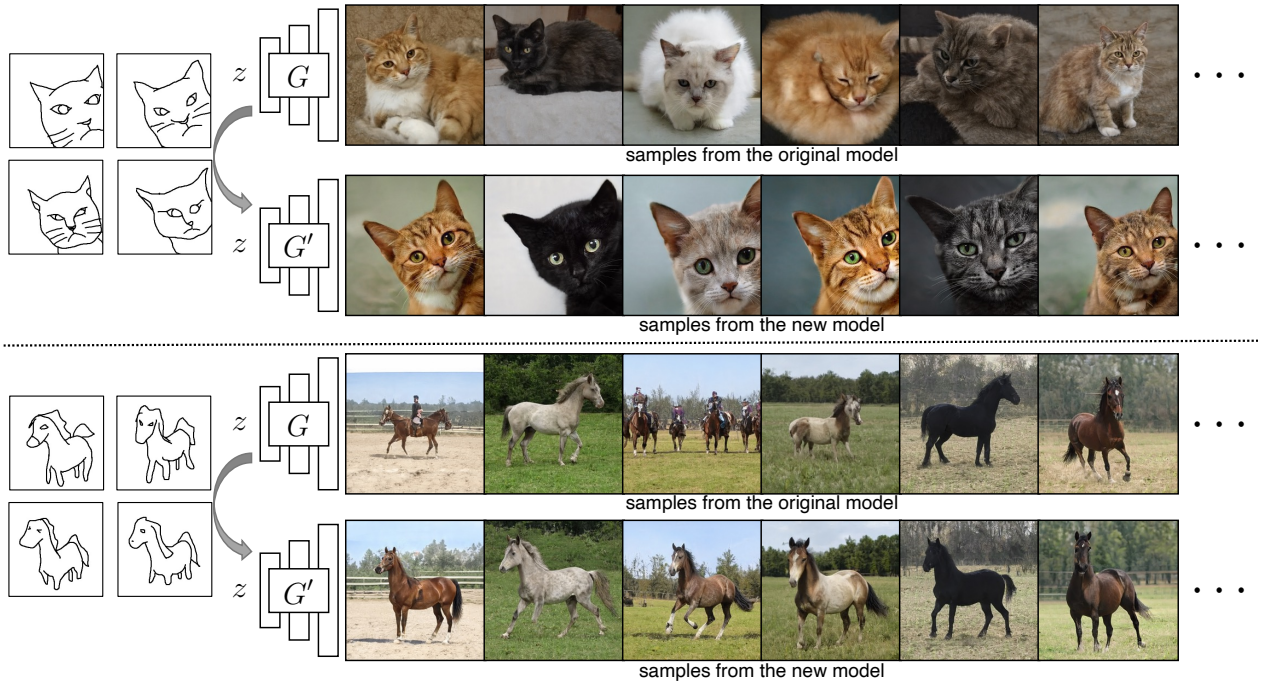


Figure 1. **Customizing a GAN with one or more human sketches.** Our method takes in one or a few hand-drawn sketches (a) and modifies an off-the-shelf GAN to match the input sketch (b). The same noise z is used for both the original model G and modified model G'. While our new model changes an object's shape and pose, other visual cues, such as color, texture, and background, are faithfully preserved after the modification.

Abstract

Can a user create a deep generative model by sketching a single example? Traditionally, creating a GAN model has required the collection of a large-scale dataset of exemplars and specialized knowledge in deep learning. In contrast, sketching is possibly the most universally accessible way to convey a visual concept. In this work, we present a method, GAN Sketching, for rewriting GANs with one or more sketches, to make GANs training easier for novice users. In particular, we change the weights of an original GAN model according to user sketches. We encourage the model's output to match the user sketches through a cross-domain adversarial loss. Furthermore, we explore different regularization methods to preserve the original model's diversity and image quality. Experiments have shown that our method can mold GANs to match shapes and poses specified by sketches while maintaining realism and diversity. Finally, we demonstrate a few applications of the resulting GAN, including latent space interpolation and image editing.

1. Introduction

The power and promise of deep generative models such as GANs [20] lie in their ability to synthesize endless realistic, diverse, and novel content with minimal user effort. The potential utility of these models continues to grow thanks to the increased quality and resolution of large-scale generative models [31, 7, 53, 51] in recent years.

Nonetheless, the training of high-quality generative models demands high-performance computing platforms, putting the process out of reach for most users. Furthermore, training

a high-quality model requires expensive large-scale data collection and careful pre-processing. Commonly used datasets such as ImageNet [13] and LSUN [69] require human annotation and manual filtering. The specialized FFHQ Face dataset [30] requires delicate face alignment and super-resolution pre-processing. Moreover, the technical effort is not trivial: developing an advanced generative model requires the domain knowledge [56, 31] of a team of experts, who often invest months or years into a single model on specific datasets.

Problem

This leads to the question: how can an ordinary user create their own generative model? A user creating artwork with cats might not want a generic model of cats, but a bespoke model of special cats in a particular desired pose: nearby, reclining, or all looking left. To obtain such a customized model, must the user curate thousands of reclining left-looking cat images and then find an expert to invest months of time in model training and parameter tuning?

In this paper, we propose the task of creating a generative model from just a handful of hand-drawn sketches. Ever since Ivan Sutherland's SketchPad [62], computer scientists have recognized the usefulness of guiding computer-generated content using sketching interface. This tradition has continued in the area of sketch-based image synthesis and 3D modeling [27, 11, 28]. But rather than creating a single image or a 3D shape from a sketch, we wish to understand if it is possible to create a generative model of realistic images from hand-drawn sketches. Unlike sketch-based content creation, where both the input and output are 2D or 3D visual data, in our case, the input is a 2D sketch and the output is a network with millions of opaque parameters that control algorithm behavior to make images. We ask: with such a different output domain, which parameters shall we update, and how? How do we know whether the model's output will resemble the user sketch?

In this paper, we aim to answer the above questions by developing a method to tailor a generative model to a small number of sketch exemplars provided by the user. To achieve this, we take advantage of off-the-shelf generative models pre-trained on large-scale data, and devise an approach to adjust a subset of the model weights to match the user sketches. We present a new cross-domain model fine-tuning method that encourages the new model to create images that resemble a user sketch, while preserving the color, texture, and background context of the original model. As shown in Figure 1, our method can change the object pose and zoom in cat faces with only four hand-drawn sketches.

We use our method to create several new customized GAN models, and we show that these modified models can be used for several applications such as generating new samples, interpolating between two generated images, as well as editing a natural photograph. Our method requires minimal user input. Instead of collecting a new dataset through manual filtering and image alignment, a user only needs to provide one or a few exemplar sketches for our method to work effectively. Finally, we benchmark our method to fully characterize its performance. Code and models are also available on our webpage.

2. Related Works

Sketch based image retrieval and synthesis. Retrieving images that resemble a human sketch has been extensively studied, including classic methods [16, 9, 59, 38] that rely on feature descriptors, as well as more recent deep learning methods [70, 57, 39, 49, 54]. The above sketch-based image retrieval (SBIR) techniques have powered sketch-based 3D modeling systems (e.g., Teddy [27]) as well as image synthesis systems, including Sketch2Photo [11] and PhotoSketcher [17]. These seminal works have further inspired deep learning solutions based on image-to-image translation [28, 78, 65] such as Scribbler [58], SketchyGAN [12], SketchyCOCO [19], and sketch-based face and hair editing [48, 44]. Other relevant work includes sketch recognition [15, 71] and sketch generation [21]. Collectively, the above methods have enabled a novice user to synthesize a *single* natural photograph. In this work, we would like to employ the same intuitive interface for rewriting a generative model. Once done, the resulting model can produce an infinite number of new samples that resemble the input sketch. The models can be further used for random sampling, latent space interpolation, as well as natural photo editing.

Generative models for content creation. After years of development, deep generative models are able to produce high-quality, high-resolution images [20, 30, 7, 32], powering a wide range of computer vision and graphics applications. Recent examples include image projection and editing with GANs [77, 6, 1, 76, 55, 47], image-to-image translation [28, 40, 45, 35], simulation-to-real [52, 60], and domain adaptation [18, 64, 24]. The advance of generative models comes at the cost of intensive computation [7, 31], the construction of large-scale high-quality datasets [13, 69, 30], and domain expertise on model training [56, 31]. As a result, recent advanced models are often developed in research labs with abundant computing and human resources. Different from prior works, we aim to help novice users quickly customize their own models without tedious data collection and domain knowledge. We achieve it using a sketching interface and our cross-domain fine-tuning method. Similar to our work, model rewriting [5] aims to change the rules of a pre-trained model through user interaction. Compared to the object copy-and-paste tool used in model rewriting, our sketching interface provides new capabilities. For example, it is much easier to describe the object shape, pose, and scene layout through quick sketching, compared to finding parts from different images and compositing them together.

Model fine-tuning. To train a GAN model of a new dataset,

researchers have fine-tuned the weights of a pre-trained generator and discriminator pair using transfer learning [14, 72]. The fine-tuning can improve upon training from scratch [68], but it can also easily overfit on the new training data. To avoid overfitting, several groups propose to limit the changes in model weights: Batch Statistic Adaptation preserves all weights except batch statistics [43]; Freeze-D freezes layers of the discriminator [42]; AdaFM preserves generator layers [73]; MineGAN keeps weights while altering latent-sampling [67]; and Elastic Weight Consolidation [33] preserves weights based on Fisher information [37]. The other technique is data augmentation, which has been proven effective for small-scale datasets [74, 29, 63, 75]. Different from the above works, our goal is *not* to fine-tune weights to learn a model of sketches. Instead, we aim to enable a user to create a new model of realistic images that builds upon the color, texture, and details of a pre-trained GAN, guided by the user-specified object shapes and poses from sketches.

3. Methods

Two constraints make the creation of a GAN model from user sketches challenging. First, as our goal is to simplify the user creation of a generative model, we must utilize only a very small amount of user-provided sketch data. It would be unreasonable to require the user to supply hundreds or thousands of sketches; instead, we aim to be able to create a model using as few as one single sketch.

Input-> Sketch
Output-> Image

Second, as we aim to synthesize realistic images without requiring the user to create realistic images, the user-provided sketches are not drawn from the target domain. This mismatch between training data (i.e., sketch) and model output (i.e., image) makes our problem setting drastically different from the traditional GAN training objective, which is to match the training data directly. In our setting, the goal is to create a model of realistic photographs where the shape and pose are guided by sketches — but where the outputs are realistic images, rather than sketches.

To overcome the above challenges, we introduce a cross-domain adversarial loss using a domain-translation network (Section 3.1). Unfortunately, simply using this loss dramatically changes the model's behavior and produces unrealistic results. To preserve the content of the original dataset as well as its diversity, we further train the model while applying image-space regularization (Section 3.2). Finally, to alleviate model overfitting, we limit the updates to specific layers and use data augmentation in Section 3.3.

3.1. Cross-Domain Adversarial Learning

Let $\mathcal{X}, \mathcal{Y}$ be the domains that consist of images and sketches, respectively. We have collected a large-scale set of training images $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ and a few human sketches $\mathbf{y} \sim p_{\text{data}}(\mathbf{y})$. We denote $G(\mathbf{z}; \theta)$ as a pre-trained GAN that produces an image $\mathbf{x}$ from a low-dimensional code $\mathbf{z}$, We

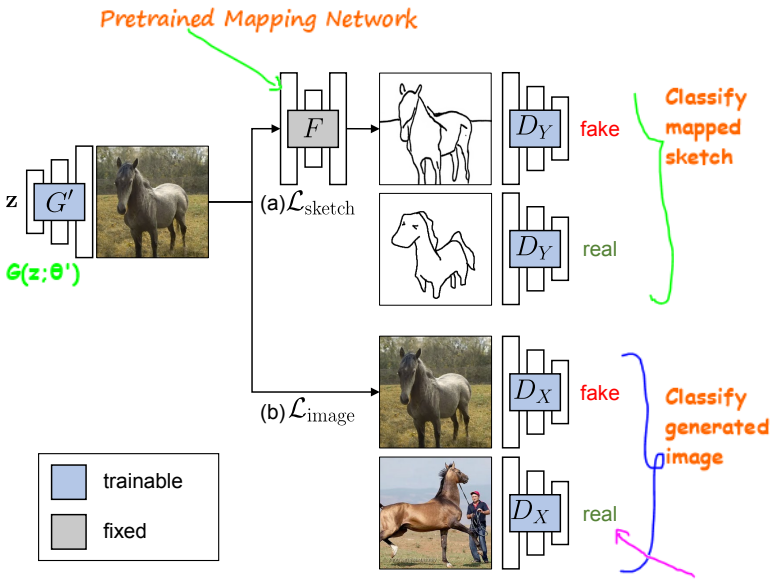


Figure 2. **Training procedure.** Our training consists of two major components. **(a)** $\mathcal{L}_{\text{sketch}}$**:** the sketch discriminator D_Y classifies between fake and user sketches. A pre-trained mapping network F [36] is used to translate the output of our model $G(\mathbf{z}; \theta')$ to a fake sketch. **(b)** $\mathcal{L}_{\text{image}}$**:** the image discriminator D_X classifies between fake and real images. Real images are sampled from the training set of the original model $G(\mathbf{z}; \theta)$.

wish to create a new GAN model $G(\mathbf{z}; \theta')$, whose output images still follow the same data distribution of $\mathcal{X}$, while the sketch version of the output images are similar to the data distribution of $\mathcal{Y}$.

Previous few-shot GAN algorithms fail to work in this setting as no ground truth images from new dataset are provided by the user. To address the challenge, we utilize a cross-domain image translation network from images to sketches $F : \mathcal{X} \rightarrow \mathcal{Y}$. This network can be trained using input-output pairs such as photos and their sketch version. Alternatively, it can be learned through unpaired learning [78, 25]. Once the mapping network is pre-trained, we do not need the sketch-image ground truth pairs during our model creation.

To bridge the gap between sketch training data and the image generative model, we introduce a cross-domain adversarial loss [20] to encourage the generated images to match the sketches $\mathcal{Y}$. Before passing into the discriminator, the output of the generator is transferred into a sketch by the pre-trained image-to-sketch network F.

Discriminator output on real sketch image

$$\mathcal{L}_{\text{sketch}} = \mathbb{E}_{\mathbf{y} \sim p_{\text{data}}(\mathbf{y})} \log(D_Y(\mathbf{y})) + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \log(1 - D_Y(F(G(\mathbf{z})))), \quad (1)$$

Discriminator Output of Generated Sketch Image. See Fig:2 (a).

Where we use Photosketch [36] as our image to sketch network F. Note that our method is able to generalize to sketch examples that are not from the original PhotoSketch training or test set (Section 4.1). Despite the difference in sketching styles, the network still helps capture the overall shape of the object.

3.2. Image Space Regularization

We observe that using the loss on sketches alone leads to a drastic degradation in image quality and diversity in generation, as this loss only enforces the shape of the generated images to match the sketch.

To resolve this, we add a second adversarial loss that compares outputs to the training set of the original model.

$$\mathcal{L}_{\text{image}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})} \log(D_X(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \log(1 - D_X(G(\mathbf{z}))). \quad (2)$$

A separate discriminator D_X is used to preserve the image quality and diversity of model outputs while matching the user sketch.

Weight regularization as an alternative. We also experiment with weight regularization, where we explicitly penalize large changes in the weights using the following loss:

$$\mathcal{L}_{\text{weight}} = ||\theta' - \theta||_1. \quad (3)$$

Although this regularization does not require the training set of the original model, we observe that applying weight regularization leads to slightly worse performance than image space regularization (Eqn. 2). See more detailed comparisons in Section 4.1.

A recent approach, Elastic Weight Consolidation (EWC) [33], aims to overcome catastrophic forgetting and can be used to regularize few-shot GANs training [37]. In our setting, we find that a simple L1-based weight regularization loss (Eqn. 3) performs on par with EWC. We report results of L1-based weight regularization for simplicity.

We experiment with models trained with both image and weight regularization methods combined, and find that they do not outperform models trained with only image regularization. However, we note that applying either weight or image regularization is critical for balancing image quality and shape matching.

3.3. Optimization

Our full objective is:

$$\mathcal{L} = \mathcal{L}_{\text{sketch}} + \lambda_{\text{image}}\mathcal{L}_{\text{image}}, \quad (4)$$

with $\lambda_{\text{image}} = 0.7$, controlling the importance of the image regularization term. We would like to learn a new set of weights $G(\mathbf{z};\theta')$ with the following minimax objective:

$$\theta' = \arg\min_{\theta'} \max_{D_X, D_Y} \mathcal{L}. \quad (5)$$

Which layers to edit. To prevent model overfitting and accelerate fine-tuning speed, we only modify the weights of the mapping network of StyleGAN2 [31], which essentially remaps $z \sim \mathcal{N}(0, I)$ to a different intermediate latent space ($\mathcal{W}$ space). We observe this to be effective, since modifying the mapping network is sufficient to obtain our target distribution, which is a subset of the original distribution. This choice has also been shown effective in previous few-shot GAN works [66]. We have also experimented with optimizing the entire generator, and observe that generated output contains severe artifacts.

Pre-trained weights. We use a pre-trained Photosketch network F and we fix the weights of F throughout the training. We optimize all the parameters for both D_X, D_Y during training. D_X, D_Y are initialized with the pre-trained weights from the original GAN. More training details are provided in Appendix A.

Data augmentation. We experiment with differentiable augmentation [74] applied to the sketches for training. We find that mild augmentation performs better in our scenario; in particular, we use translation for the augmentation. While augmentation does not necessarily improve the results when we use 30 input sketches generated from the Photosketch network F, we find that it is essential for model training with one or a few hand-drawn sketch inputs. More details are in Section 4.1.

4. Experiments

4.1. Evaluations

Datasets. To enable large-scale quantitative evaluation, we construct a dataset of model sketching scenarios with ground truth target distributions defined as follows. We transform the images from LSUN [69] horses, cats, and churches to sketches using PhotoSketch [36], and hand-select sets of 30 sketches with similar shapes and poses to designate as the user input, as shown in Figure 3. To define the target distribution, we hand-select additional 2,500 images that match the input sketches. We select them out of 10,000 candidate images with the smallest chamfer distances [4] to the designated inputs. (Candidate images retrieved by the SBIR method of Bui *et al.* [8] was also considered but did not match poses as faithfully.) Our method is given access only to the 30 designated sketches; the sets of 2,500 real images represent the real but unseen target distributions.

To test our method in a realistic scenario, we collect human sketches from the Quickdraw dataset [10]. It is challenging to curate an evaluation set with human sketches. Hence, we evaluate on the test cases qualitatively.

Performance metrics. We evaluate our models based on the Fréchet Inception Distance [23] (FID) between the generated images and the evaluation set. The FID measures the distribution similarity between the two sets, and serves as a metric for the diversity and quality of the generated images, as well as how well the images match the sketches. For a fair comparison, we evaluate each model by selecting the iteration with the best FID.

Discriminator output of real image
See Fig 2 (b)
Discriminator Ouput of Fake Image.
L1 loss
Main goal is to learn θ' and minimize it of generator model G(x;θ') such that it should generate realistic image with given low dimensional noise vector (z). And discriminator classify Generated Image and the Real Image (Sampled from the another pre-trained Generator G(x, θ)) Where Dx and Dy want to maximize it.

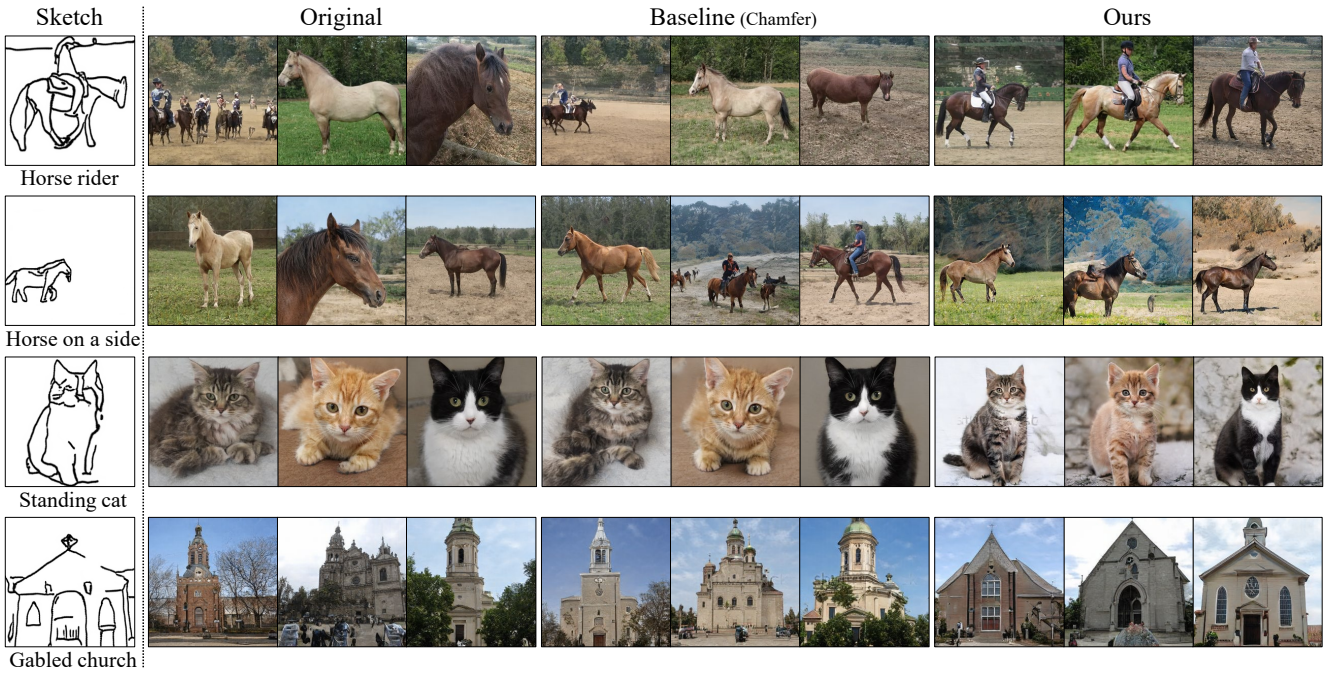


Figure 3. **Qualitative results on synthetic sketches.** Each row shows one model sketching task. For each of the four tasks, 30 PhotoSketch sketches are used for training; one training sketch is shown. We show samples generated from the original models, the models fine-tuned using **Baseline(Chamfer)** and the models customized using our method (**Full (w/o aug.)**), and the **Horse rider** model is **Full (w/ aug.)**). For each task, we show samples generated from the same three z. Following Karras *et al*. [31], truncation $\psi = 0.5$ is applied at all samples. We observe that the samples generated by our methods match the sketches better than the baseline.

Family	Name	Training settings: No. Samples	Training settings: Aug.	FID ↓: Horse rider	FID ↓: Horse on a side	FID ↓: Standing cat	FID ↓: Gabled church
Pre-trained	Original	N/A		50.43	42.24	58.71	32.64
Baseline	Bui *et al*. [8]	30		46.00	43.52	59.86	29.94
	Chamfer	30		47.04	48.18	54.04	19.71
Ours	1-sample	1		29.25	41.50	44.68	26.88
	5-sample	5		33.11	41.61	**31.20**	23.28
	D scratch‡	30		44.91	27.84	47.69	24.41
	W-shift‡	30		30.66	34.86	42.24	17.88
	Full (w/o aug.)	30		27.50	**29.62**	33.94	**16.70**
	Full (w/ aug.)	30	✓	**19.94**	30.39	36.73	21.35

Table 1. **Quantitative analysis.** We report the Fréchet Inception Distance (FID) of the original models, baselines and our methods on four different test cases with synthetic sketch inputs. The details of the baselines are in Section 4.1. We test on model variants trained on fewer training samples (No. Sample) and ablate our method by altering the training components. ✓ indicates translation augmentation is applied. ↑, ↓ indicate if higher or lower is better. Evaluations on the original models are in gray, and the best value is highlighted in **black**. (‡: "D scratch" indicates that the sketch discriminator D_Y is initialized randomly; "W-shift" indicates that a shift in $\mathcal{W}$ space is the only tunable parameter of the generator.)

Baselines. We compare our method to the following baselines. We evaluate the effect on the model output when it is customized by shifting the latent $w_{\text{new}} = w + \Delta w$ using a constant vector Δw derived by averaging samples that resemble the user sketch, similar to the vector arithmetic method proposed in Radford *et al*. [50]: $\Delta w = \mathbb{E}_{\text{match}}[w] - \mathbb{E}_{\text{orig}}[w]$.

	FID ↓: Horse rider	Horse on a side	Standing cat	Gabled church
Original model	50.43	42.24	58.71	32.64
$\mathcal{L}_{\text{sketch}}$	27.39	40.65	50.09	19.33
$\mathcal{L}_{\text{sketch}}$+aug.	28.28	39.03	49.52	21.60
$\mathcal{L}_{\text{sketch}}+\mathcal{L}_{\text{weight}}$	30.94	38.55	49.76	17.55
$\mathcal{L}_{\text{sketch}}+\mathcal{L}_{\text{weight}}$+aug.	21.99	35.44	48.84	22.41
$\mathcal{L}_{\text{sketch}}+\mathcal{L}_{\text{image}}$	27.50	**29.62**	**33.94**	**16.70**
$\mathcal{L}_{\text{sketch}}+\mathcal{L}_{\text{image}}$+aug.	**19.94**	30.39	36.73	21.35

Table 2. **Ablation study.** We evaluate the effect of each component of our losses and data augmentation on four test cases with synthetic sketch inputs, and report the Fréchet Inception Distance (FID) tested on our evaluation set. Lower value is better. "Original model" denotes the pre-trained models (labelled in gray).

Here $\mathbb{E}_{\text{orig}}[w]$ is the mean latent in the original unmodified model and $\mathbb{E}_{\text{match}}[w]$ is the mean latent over a subset of samples where the image resembles the user sketch according to a simple criterion. We implement two baseline variants, which use different methods to sample $\mathbb{E}_{\text{match}}[w]$. **(1) Baseline (SBIR):** selects best matched samples using the sketch-based image retrieval method by Bui *et al*. [8], **(2) Baseline (Chamfer):** matches samples using the symmetric Chamfer distance $d(x, y) + d(y, x)$ between the input sketch y and a sketch of the image x as computed by PhotoSketch [36]. To estimate $\mathbb{E}_{\text{match}}[w]$, we take the top-matched 10,000 images from 1 million samples, and we score each image using the minimum distance to all user sketches.

Model trained with regulaization, seems more realistic (Right), and viariation in generated output.

Figure 4. **Effect of regularization methods.** We compare models trained with (top) only $\mathcal{L}_{\text{sketch}}$, with (middle) weight regularization $\mathcal{L}_{\text{weight}}$, and with (bottom) image regularization $\mathcal{L}_{\text{image}}$. **(Left)** the snapshots at the same iteration. **(Right)** the best iterations selected based on our evaluation metric. We observe that the images look more realistic with either regularization applied, and the model trained with image regularization obtains better diversity than the one trained with weight regularization.

Table 1 shows the quantitative comparisons. We note that our method obtains significantly better FID than **Baseline (SBIR)** and **Baseline (Chamfer)**. The result agrees with our qualitative comparison in Figure 3, where the baseline does not match user sketches nearly as well as our method (**Baseline (Chamfer)** is chosen for visual comparison because it obtains better average FID than **Baseline (SBIR)**).

Ablation studies. We first investigate the effects of our regularization methods and data augmentation. The results are shown in Table 2.

Augmentation. We find that augmentation does not necessarily improve the performance for sketches generated from Photosketch. With image regularization applied, the **horse rider** model benefits from augmentation, whereas **horse on a side**, **standing cat** and **gabled church** models perform better without augmentation. However, we find that augmentation is critical when the training inputs are human-created sketches, as discussed later.

Comparing regularization methods. Either regularization method $\mathcal{L}_{\text{image}}$ or $\mathcal{L}_{\text{weight}}$ improves FID over models trained with only $\mathcal{L}_{\text{sketch}}$, while models trained with image regularization $\mathcal{L}_{\text{image}}$ outperform models trained with $\mathcal{L}_{\text{weight}}$. This agrees with our observation in Figure 4, which shows snapshots of models trained with and without regularization. At the same training iteration, both regularization methods preserve image quality, and image regularization obtains the most diverse outputs. When the best iterations are selected for each method, the comparison again reveals that the regularized models obtain a better balance of shape matching and image quality.

We also investigate the effects of other training components, as shown in Table 1. The following analysis focus on models trained without augmentation, since augmentation in general is not beneficial to synthetic sketch inputs. We refer $\mathcal{L}_{\text{sketch}}$+$\mathcal{L}_{\text{image}}$ as **Full (w/o aug.)** and $\mathcal{L}_{\text{sketch}}$+$\mathcal{L}_{\text{image}}$+**aug.** as **Full (w/ aug.)**.

D scratch. To test whether using pre-trained weights for the sketch discriminator D_Y is necessary, we evaluate a variant with D_Y initialized randomly. We observe a huge decrease in performance in most cases. This indicates that pre-training in discriminators is important even when switching the training domain from images to sketches. The finding is consistent with prior work on few-shot GAN finetuning [42, 37].

W-shift. We test on a variant where the only tunable parameter is a bias added to the mapping network, which effectively performs a shift in the $\mathcal{W}$ space. We observe this method leads to a reasonable performance, despite being worse than our full method by a margin. This shows that our training procedure can potentially serve as a flexible latent discovery method. However, in general, tuning the entire mapping network makes our method more effective.

Fewer sketch samples. We test if our method is able to work on a smaller number of sketches. For each task, we trained models with only 1 or 5 sketches, selected from the previous 30 sketches. The results are reported in Table 1.

We observe that the models trained with 1 or 5 sketches improve upon the original model. In most cases, training on 30 sketches still outperforms, which shows the degree to which results can be improved when the user provides more sketches. In the **standing cat** task, training with 5 sketches achieves slightly better results, and in the **horse on a side** task the improvement is small. The other two tasks improves significantly when 30 sketches are used. Training a model with very few sketches remains challenging, but our method improves with the number of user sketches.

Testing using real human sketches. To make GAN customization available to everyday users, we then test the effectiveness of our method on hand-drawn sketches from novice users. We collect cat and horse sketches from the Quickdraw [10] dataset as training images. We first train models on a single sketch, and show success and failure cases in Figure 5. We note that the image to sketch translation network Photosketch [36] used in our method is trained on a sketch style that is different from the Quickdraw dataset. Despite the difference in the styles, our method succeeds on sketches that are more contour-like and depict simple poses (e.g., cat head). However, there is headroom for improvement, to support a wider range of sketching styles and poses. In particular, the results are worse with sketches that have more abstract styles or complex poses. We also observe that

Seems that pretrained discriminator Dy is necessary for performance improvement in compare to initialize randomly

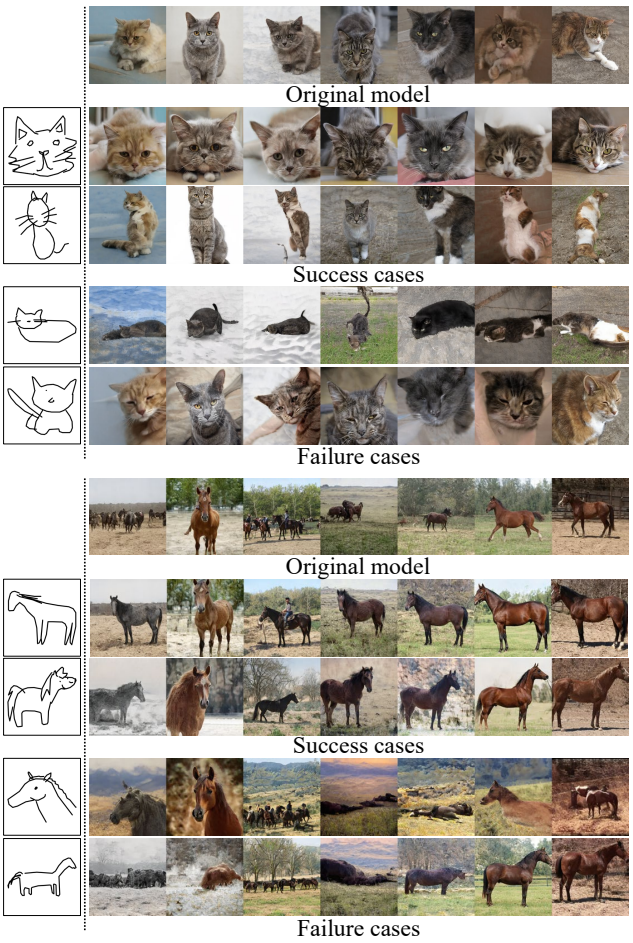
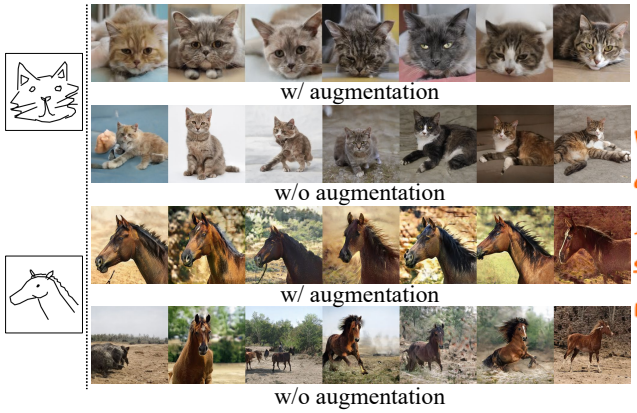


Figure 5. **Model creation using a single human-created sketch.** Each row shows uncurated samples generated from a model trained on a single sketch from Quickdraw [10]. Same noise z is used and truncation $\psi = 0.5$ is applied to each model. Results on cats **(top)** and horses **(bottom)** are shown. For each category, the 1st row is the original pre-traiend model, 2nd and 3rd rows represent the success cases, and the last two rows are the failure cases.

Figure 6. **Model creation using multiple human-created sketches.** Some failure cases in Figure 5 can be improved with more input sketches. 3 similar sketches are used in the cat model **(top)** and 4 are used in the horse model **(bottom)**. Samples are generated in the same fashion as in Figure 5

performance on difficult cases can be improved by increasing the number of input user sketches, as shown in Figure 6. Also, we find that augmentation is essential for our method to be successful for user sketches. As shown in Figure 7, given the same input sketches, only the model trained with



With augmentation, results seems realistic.

Figure 7. **Effect of augmentation on human-created sketches.** We take models from Figure 5 (top) and Figure 6 (bottom), and observe that only models trained with augmentation generate images that faithfully match the sketches.

Figure 8. **Customizing FFHQ models.** Each row shows samples on a customized FFHQ model, trained on 4 human-created sketches (1 is shown).

augmentation will faithfully generate images that match the sketches.

We apply our method to generative model of faces as well. We customize the StyleGAN2 FFHQ models [31] with 4 human-drawn sketches using our **Full (w/ aug.)** method, and we find that using $\lambda_{\text{image}} = 0.5$ produces better results in this case. Results are shown in Fig. 8, and one can observe the generated output resembles the input sketches.

4.2. Applications

In this section, we discuss several ways of applying our method to image editing and synthesis tasks. We show that with our customized models, one can perform latent space edits and manipulate natural images. We also demonstrate that it is possible to interpolate between the customized models.

Latent space edits. Interpretable user controls in a generative model are useful for many graphics applications. We investigate whether this property still holds for our customized models. More importantly, the edit operations should be identical to the original models, to avoid the need to run latent discovery algorithms again on each customized model. To investigate this, we apply the latent discovery method GANSpace [22] to the original models. By moving along the

Apply latent discovery method GANSpace to the original models by moving along latent direction.

Figure 9. **Editability of customized models.** We apply the edits reported in Härkönen *et al*. [22] to the customized models. We observe that the edit operations discovered from the original model gives the same effect to the customized models

Figure 10. **Interpolation using customized models.** Latent space interpolation is smooth with our customized models. Here we show **gabled church (top)** and the **horse on a side (bottom)** results.

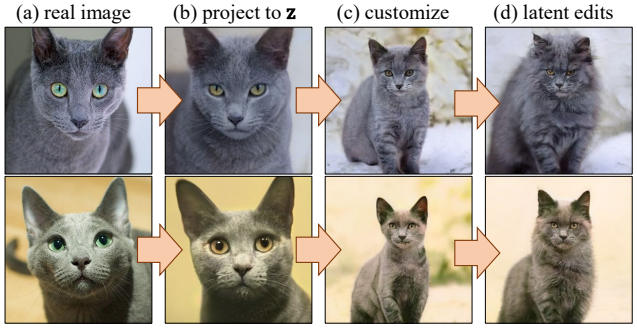


Figure 11. **Natural image editing with original and customized models.** Given a real image as input **(a)**, we project the image to the original model's z using Huh *et al*. [26] **(b)**. We then feed the projected z to the **standing cat** model trained on sketches, which effectively edits input to match the sketches **(c)**. Furthermore, we showed the image can be further edited using GANSpace [22] **(d)**. Similar to Figure 9, both images from the customized model can be applied with the "add fur" operation.

reported latent directions, we observe that our customized models can perform the exact same manipulations from the results in Härkönen *et al*., as shown in Figure 9. Since we are only tuning the mapping network of the generator, our method did not change how the model is processing the w-space latents. As a result, properties on latent edits are preserved. In addition, we observe that latent interpolation remains smooth in our models, as shown in Figure 10

Natural image editing with our models. Our method is capable of "editing" an original model to create a new model that matches user sketches, but is it possible to edit a single photograph using our new model? We show that natural image editing can be realized by image projection. To illustrate, we project the natural image to a noise z from the original model using Huh *et al*. [26]. Since our method modifies the shape and pose while preserving the texture of background and objects under the same z, we can feed the projected z to the customized model, and the output is effectively the transformed version of the input image with a new shape and pose matching the sketch. We also verified that the latent space edits (Figure 9) still apply to the "transformed" photograph. The results are shown in Figure 11.

Figure 12. **Failure cases.** Our method is not capable of generating images to match the Attneave's cat sketch [2] **(top)** or the horse sketch by Picasso **(bottom)**.

Interpolating between customized models. After models are customized, we demonstrate that it is possible to interpolate between these models in two ways, as shown in Figure 13. (1) First, we feed the same noise vector z to two different models to obtain two latents w_1, w_2 in the $\mathcal{W}$ space. We then feed $(1-\alpha)w_1 + \alpha w_2$ to the synthesis network to obtain images with interpolated shapes between the two models, where $\alpha \in [0, 1]$ controls the interpolation. (2) Also, we observe that the same effects can be achieved by directly interpolating the model weights θ_1, θ_2. In particular, the interpolated models have weights $(1-\alpha)\theta_1 + \alpha\theta_2$. We note that the similar task has been explored by concurrent work [3].

5. Discussion

In this work, we present a method that enables the user creation of customized generative models, by leveraging off-the-shelf pre-trained models and cross-domain training. The required input of our method is just one or more hand-drawn sketches, which makes model creation possible for novice users. Our method overcomes the large domain gap between user sketches and generator parameter space, and it is capable of generalizing to sketches in different styles.

However, plenty of improvement remains for our method. We have shown our method generalizes to other sketch styles and different poses in Section 4.2, but our method is not guaranteed to work for all sketches. For example, we tested our method on horse sketches from Picasso and the Attneave's sleeping cat [2]. As shown in Figure 12, our method cannot create a model that matches the pose faithfully. We note that Picasso's sketches are drawn with a distinctive style, and Attneave's cat depicts a complex pose, both of which

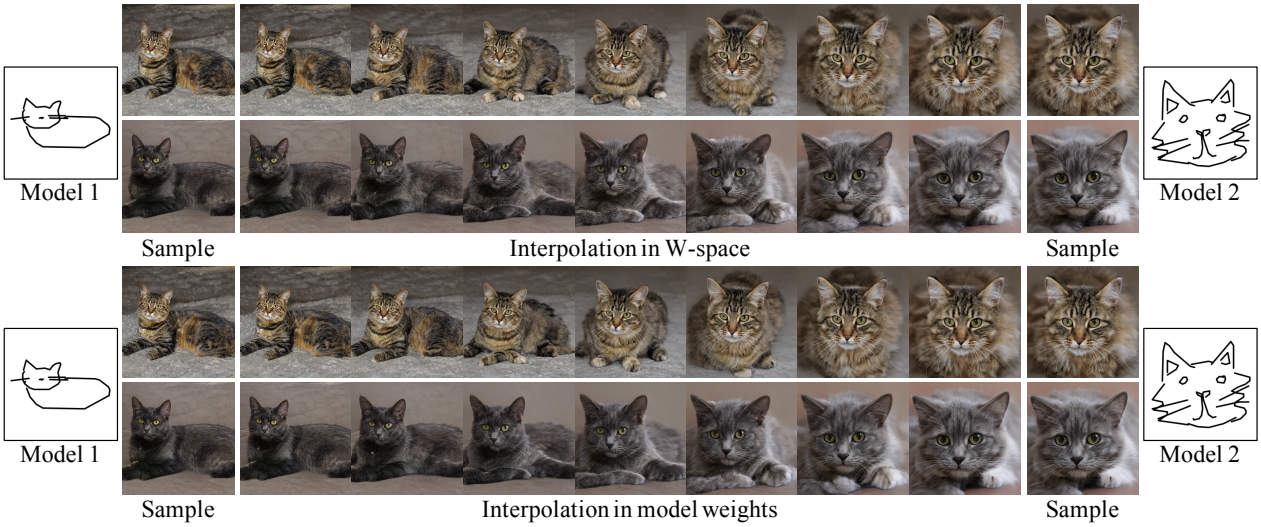


Figure 13. **Interpolating between customized models.** We can interpolate between the customized model by interpolating (top) the W-latents or (bottom) the model weights. We show image samples from model 1 and 2 on the left and right side, respectively, and the middle shows the interpolation results. Model 1 and 2 are from Figure 6 and Figure 5, respectively.

are potential reasons for the failure. Another limitation is that customizing a model in real-time is impossible with our current method, since our models take more than 30K iterations to train. Our method requires access to the training set of the original model, which may make it inappropriate for settings where this data is not available. Finally, while our method enables flexible control of shape and poses, it cannot customize other properties such as color and texture. To expand expressiveness, we note our cross-domain loss can be applied to other inputs such as VGG features, color scribbles, or semantic layouts.

Acknowledgment. We thank Nupur Kumari and Yufei Ye for proof-reading the drafts. We are also grateful for helpful discussions from Gaurav Parmar, Kangle Deng, Nupur Kumari, Andrew Liu, Richard Zhang, and Eli Shechtman. We are grateful for the support of Sony Corporation, Naver Corporation, DARPA SAIL-ON HR0011-20-C-0022 (to DB), and Signify Lighting Research.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[2] Fred Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193, 1954. 8

[3] Omri Avrahami, Dani Lischinski, and Ohad Fried. Gan cocktail: mixing gans without dataset access. *arXiv preprint arXiv:2106.03847*, 2021. 8

[4] 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 MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1977. 4, 13, 14

[5] 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. 2

[6] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM SIGGRAPH*, 38(4):1–11, 2019. 2

[7] 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. 1, 2

[8] Tu Bui, Leonardo Ribeiro, Moacir Ponti, and John Collomosse. Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network. *Computer Vision and Image Understanding*, 2017. 4, 5, 13, 14

[9] Yang Cao, Changhu Wang, Liqing Zhang, and Lei Zhang. Edgel index for large-scale sketch-based image search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2

[10] Salman Cheema, Sumit Gulwani, and Joseph LaViola. Quickdraw: Improving drawing experience for geometric diagrams. In *ACM SIGCHI*, 2012. 4, 6, 7

[11] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics (TOG)*, 28(5):124, 2009. 2

[12] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML)*, 2014. 3

[15] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on Graphics (TOG)*, 31(4):44–1, 2012. 2

[16] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE transactions on visualization and computer graphics*, 17(11):1624–1636, 2010. 2

[17] Mathias Eitz, Ronald Richter, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*, 31(6):56–66, 2011. 2

[18] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 17(1):2096–2030, 2016. 2

[19] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. Sketchycoco: Image generation from freehand scene sketches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[20] 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*, 2014. 1, 2, 3

[21] David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[22] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Advances in Neural Information Processing Systems*, 2020. 7, 8, 14

[23] 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*, 2017. 4

[24] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning (ICML)*, 2018. 2

[25] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *European Conference on Computer Vision (ECCV)*, 2018. 3

[26] 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. 8

[27] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *ACM SIGGRAPH*, 1999. 2

[28] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[29] 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. 3

[30] 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. 2, 13

[31] 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. 1, 2, 4, 5, 7, 13

[32] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018. 2

[33] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017. 3, 4

[34] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 13

[35] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[36] Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images. In *Winter Conference on Applications of Computer Vision*, 2019. 3, 4, 5, 6

[37] 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. 3, 4, 6

[38] Yen-Liang Lin, Cheng-Yu Huang, Hao-Jeng Wang, and Winston Hsu. 3d sub-query expansion for improving sketch-based multi-view image retrieval. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2

[39] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[40] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[41] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018. 13

[42] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. In *CVPR Workshop*, 2020. 3, 6

[43] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 3

[44] Kyle Olszewski, Duygu Ceylan, Jun Xing, Jose Echevarria, Zhili Chen, Weikai Chen, and Hao Li. Intuitive, interactive beard and hair synthesis with generative models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[45] 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. 2

[46] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On buggy resizing libraries and surprising subtleties in fid calculation. *arXiv preprint arXiv:2104.11222*, 2021. 13

[47] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. *arXiv preprint arXiv:2103.17249*, 2021. 2

[48] 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

[49] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. Deep shape matching. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[50] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016. 5

[51] 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

[52] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[53] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019. 1

[54] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[55] 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

[56] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016. 2

[57] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 2

[58] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[59] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Data-driven visual similarity for cross-domain image matching. In *ACM SIGGRAPH Asia*, 2011. 2

[60] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 13

[62] Ivan E. Sutherland. *Sketchpad: A Man-Machine Graphical Communication System*. PhD thesis, Massachusetts Institute of Technology, Lincoln Lab, 1963. 2

[63] 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. 3

[64] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[65] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[66] 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. 4

[67] 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. 3

[68] 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. 3

[69] 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. 2, 4, 13

[70] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe. In

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 2

[71] Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Sketch-a-net: A deep neural network that beats humans. *International journal of computer vision*, 122(3):411–425, 2017. 2

[72] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014. 3

[73] Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning (ICML)*, 2020. 3

[74] 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. 3, 4

[75] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for gan training. *arXiv preprint arXiv:2006.02595*, 2020. 3

[76] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[77] 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. 2

[78] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 3

Appendix

Below we include additional implementation details as well as extra results.

A. Implementation Details.

Training details We use the same training hyperparameters as [31]. In particular, we are using softplus for GAN loss, and R1 regularization [41] on both the sketch and image discriminator, D_Y and D_X. We do not use path length regularization, as it has no effect on the latent mapping network. Also, we set the batch size to 4 for all of our experiments, except when the sketch inputs are less than four, where we set the batch size to 1.

Hyperparameters. We use the same hyperparameters for our **full** method in all of our experiments. In particular, we use $\lambda_{\text{image}} = 0.7$

In Section 4.1, we compared several variants of our method in our ablation studies. To make the comparison fair, for each variant, we tuned the loss weights for optimal performance. In Table 3, we list the hyperparameters used for each variant. The only exception is that we use $\lambda_{\text{weight}} = 50$ for the $\mathcal{L}_{\text{sketch}}$ **+** $\mathcal{L}_{\text{weight}}$ and $\mathcal{L}_{\text{sketch}}$ **+** $\mathcal{L}_{\text{weight}}$ **+ aug.** variant model trained on the **standing cat** task. Also, if the variants are not listed in the table, the same loss weights as the full method are used. The search space of the λ_{image} is $[0.3, 0.5, 0.7, 1.0]$, and the search space of λ_{weight} is $[0.1, 1, 10, 50, 100, 1000]$.

Data collection. In Section 4.1, we selected sets of 30 sketches with similar shapes and poses to designate as the user input: examples of sketches from these sets are shown in Figure 14. To evaluate generation quality, we collected images that match the input sketches from LSUN [69]. To retrieve matching images, we experimented with two sketch-image cross-domain matching methods. We applied both the SBIR method of Bui *et al*. [8] and chamfer distance [4]. Both of these retrieval results are shown in Figure 15. We observe that with chamfer distance, the retrieved images match poses of the sketches more faithfully. As a result, we adopt this method to generate our evaluation sets. However, we notice that there still exists outliers after the retrieval; hence, we hand-selected 2,500 images out of top 10,000 matches to curate the evaluation sets. A comparison between curated dataset and top chamfer matches are shown in Figure 16.

Evaluation procedure. To evaluate each model, we sample 2,500 images without truncation and save them into png files. Likewise, the evaluation set described in Section 4.1 consists of 2,500 256×256 images stored in png. We evaluate the Fréchet Inception Distance values using the CleanFID code [46].

B. Additional results

Other evaluation metrics. We report Perceptual Path Length (PPL) [30] in Table 4. We find that our method improves the original models' PPL, and beats the baselines. We note that our model focuses on fewer modes than the original one, so interpolations are smoother on average, leading to smaller PPL.

	λ_{image}	λ_{weight}
$\mathcal{L}_{\text{sketch}}$	0	0
$\mathcal{L}_{\text{sketch}}$+aug.	0	0
$\mathcal{L}_{\text{sketch}}$+$\mathcal{L}_{\text{weight}}$	0	100
$\mathcal{L}_{\text{sketch}}$+$\mathcal{L}_{\text{weight}}$+aug.	0	100

Table 3. **Loss weights for each variant.** For a fair comparison, we use different loss weights for several variants. We find that using the above weights gives optimal performance. The variants not listed in this table is using the same hyperparameters as the **Full** method.

In addition, Precision, and Recall metrics [34] are reported in Table 4. The precision measures the proportion of generated samples that are close to the real dataset in VGG feature space [61], and the recall measures the proportion of real dataset that are close to generated samples in VGG feature space. We note that models with better results often have higher precision and lower recall. We expect our method to increase precision as it refines the generated distribution to better match the target distribution. But since our task aims at finding a subset of the source distribution, our method theoretically *cannot* increase the recall: increasing recall would require synthesizing new modes of real data without access to any new real examples. In our setting, the ideal maximizes precision while maintaining recall unchanged from the pretrained model. A drop in recall reveals some loss in diversity, and measures headroom for improving upon our method.

Additional qualitative results. In Figure 17, we show additional results on latent space editing with our customized models. Also, we show uncurated samples for our models in Figure 18 (horse rider), Figure 19 (horse on a side), Figure 20 (standing cat) and Figure 21 (gabled church).

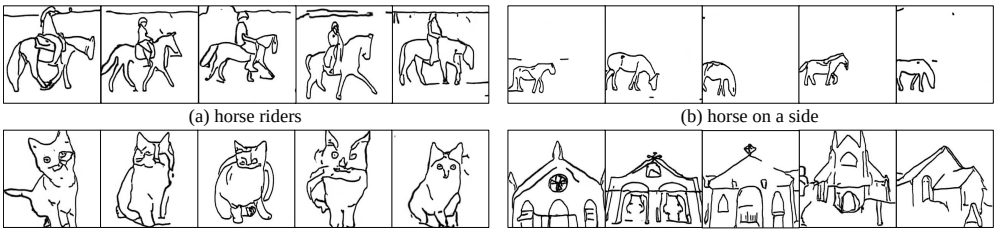
C. Changelog

v1 Initial preprint release.

v2 Updated customized FFHQ models and moved the corresponding figure to Figure 8. Corrected the information in Table 3. Added experiments on interpolating two customized models (Figure 13).

Family	Name	Training settings		Test cases											
				Horse rider			Horse on a side			Standing cat			Gabled church		
		No. Samples	Aug.	PPL ↓	Prec. ↑	Rec. ↑	PPL ↓	Prec. ↑	Rec. ↑	PPL ↓	Prec. ↑	Rec. ↑	PPL ↓	Prec. ↑	Rec. ↑
Pre-trained	Original	N/A		338.87	0.22	0.63	338.87	0.33	0.57	438.11	0.21	0.54	342.73	0.46	0.49
Baseline	Bui *et al*. [8]	30		356.56	0.24	0.53	343.48	0.26	**0.60**	433.05	0.22	**0.58**	346.48	0.49	0.48
	Chamfer	30		353.07	0.30	**0.56**	371.11	0.35	0.57	418.91	0.26	0.55	340.12	**0.50**	**0.52**
Ours	Full (w/o aug.)	30		353.71	0.42	0.52	266.69	0.42	0.49	**150.89**	**0.65**	0.20	344.24	0.48	0.48
	Full (w/ aug.)	30	✓	**306.81**	**0.50**	0.50	**232.95**	**0.44**	0.39	263.99	0.50	0.41	**336.67**	0.46	0.51

Table 4. **Other metrics.** We report the Perceptual Path Length (PPL), Precision (Prec.), and Recall (Rec.) of the original models, baselines and our methods on four different test cases. The details of the baselines are in Section 4.1. ✓ indicates translation augmentation is applied. ↑, ↓ indicate if higher or lower is better. Evaluations on the original models are in gray, and the best value is highlighted in **black**.



(a) horse riders (b) horse on a side (c) standing cat (d) gabled church

Figure 14. **Example of sketches used for training.** For each task (a, b, c, d), 30 sketches with similar shapes and layouts are hand-selected as training samples, where above shows subsets of 5 sketches.

Figure 15. **Comparison between retrieval methods.** We compare retrieval methods between chamfer distance [4] and SBIR method of Bui *et al*. [8]. We find that the retrievals using chamfer distance matches the input sketches better than those using Bui *et al*. Left shows the example query out of the 30 sketches used for the retrieval.

Figure 16. **Curated evaluation set.** We show random samples from the top 2,500 matches using chamfer distance [4] **(left)** and 2,500 hand-selected images **(right)**. The quality of the evaluation set is improved after curation.

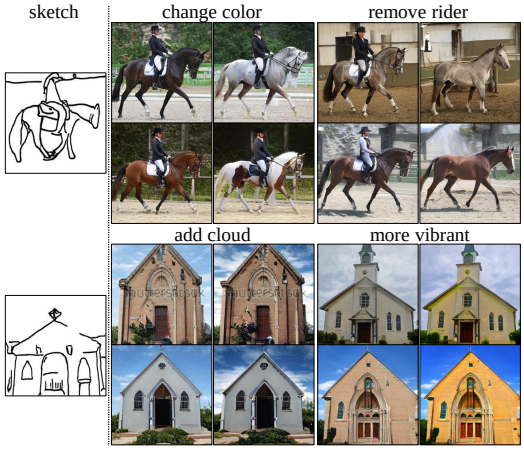


Figure 17. **Additional latent edit results.** Similar to Figure 7 of the main text, we show additional results of applying GANSpace [22] edits to our customized models, **horse rider** (top) and **gabled church** (bottom).

Figure 18. Uncurated samples of the **horse rider** model. Truncation $\psi = 0.5$ is applied to generate the images.

Figure 19. Uncurated samples of the **horse on a side** model. Truncation $\psi = 0.5$ is applied to generate the images.

Figure 20. Uncurated samples of the **standing cat** model. Truncation $\psi = 0.5$ is applied to generate the images.

Figure 21. Uncurated samples of the **gabled church** model. Truncation $\psi = 0.5$ is applied to generate the images.

Sketch Your Own GAN Summary

Prepared By: Sushant Gautam
Part of 30 Days Gan Paper Reading
https://github.com/sushant097/30-Days-GANs-Paper-Reading

Main Idea is to generate realistic image with different poses and orientaiton with GAN given a few sketch images.

See the output Below:

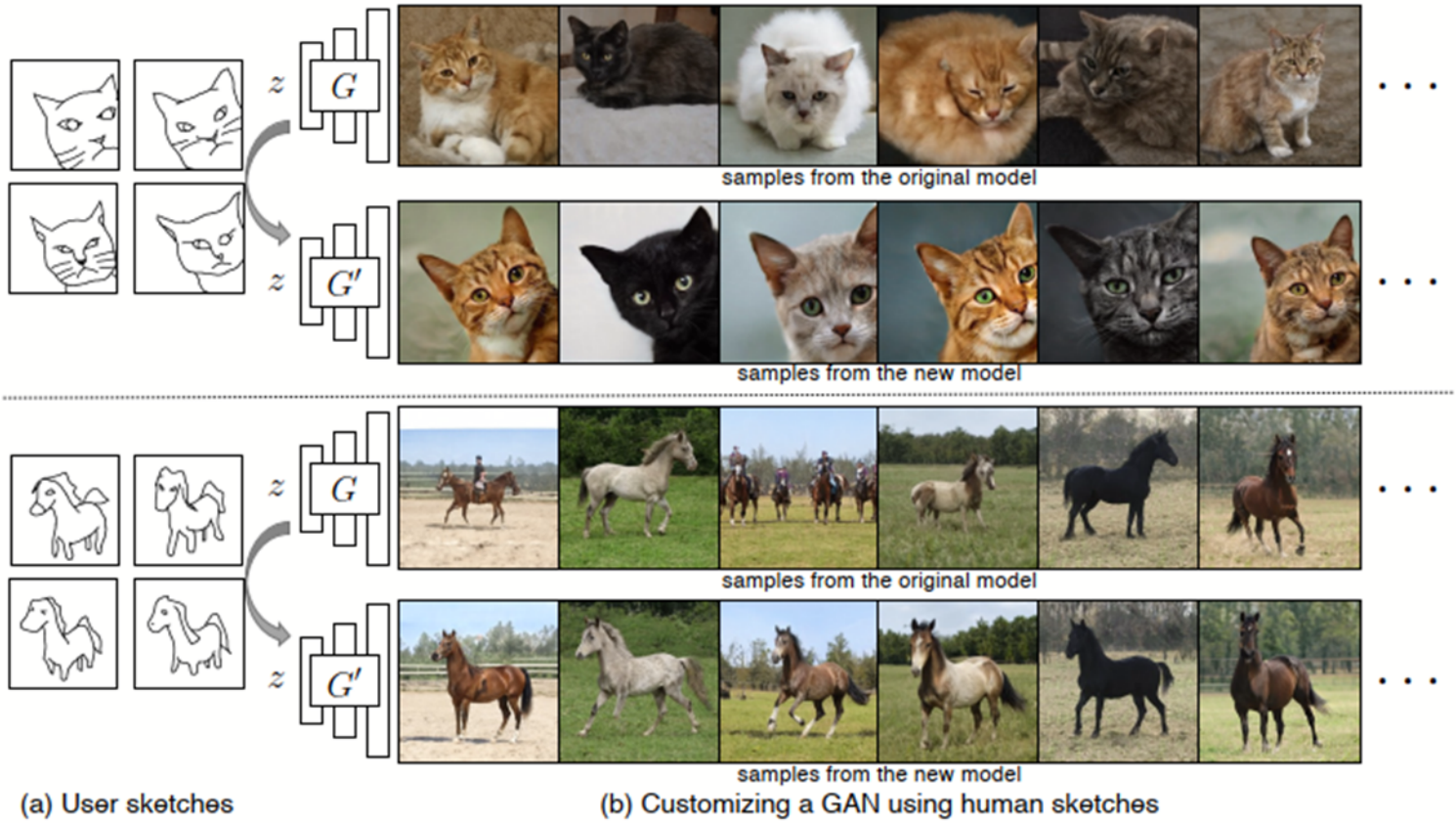


Figure 1. **Customizing a GAN with one or more human sketches.** Our method takes in one or a few hand-drawn sketches (a) and modifies an off-the-shelf GAN to match the input sketch (b). The same noise z is used for both the original model G and modified model G'. While our new model changes an object's shape and pose, other visual cues, such as color, texture, and background, are faithfully preserved after the modification.

They use two generators. One is Generator model G is a pretrained model that generates one output. But another model G' generates output of different object shapes and pose and also seems more realistic given with same noise vectors.

As input is sketch image and output should be realistic image which is translation on different domain. So, they use cross-domain adversarial loss that force the model's output to match the user sketches.

Model Architecture:

Main Goal is to learn the parameters θ' such that output of G' is close to the distribution of X.

There are two components:

(a): The output of G' is passed to the mapping network F:Image-> Sketch which is pretrained PhotoSketch (fixed) during training and new sketch is almost close to the original sketch data distribution Y. The discriminator Dy classify mapped sketch(fake) and real (input sketch).

(b): The output of G' is also passed to the Discriminator Dx where real samples are sampled from the training set of the original model G(z,θ).

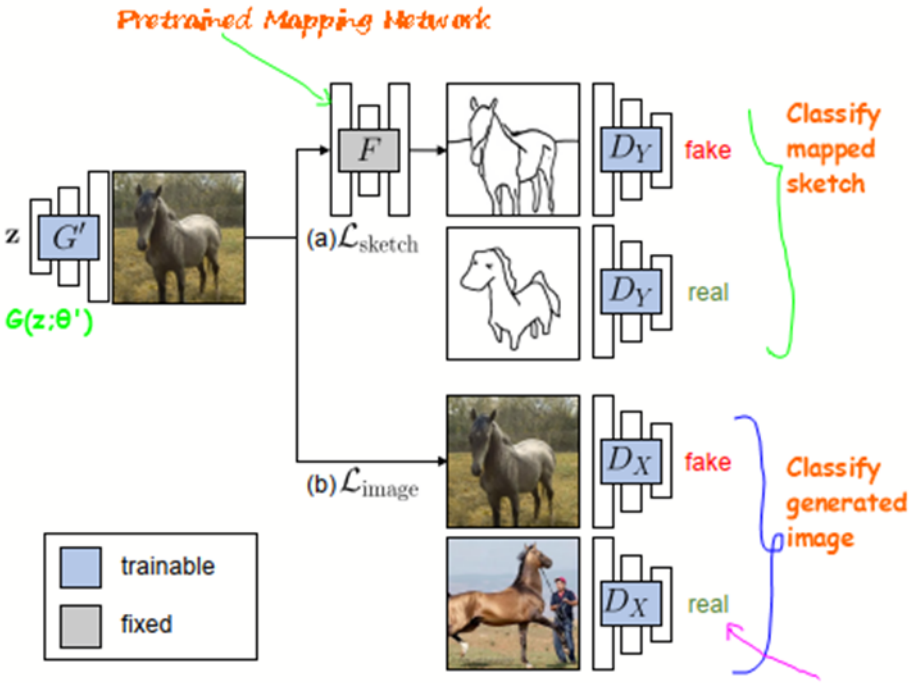


Figure 2. **Training procedure.** Our training consists of two major components. **(a)** $\mathcal{L}_{sketch}$: the sketch discriminator D_Y classifies between fake and user sketches. A pre-trained mapping network F [36] is used to translate the output of our model $G(\mathbf{z};\theta')$ to a fake sketch. **(b)** $\mathcal{L}_{image}$: the image discriminator D_X classifies between fake and real images. Real images are sampled from the training set of the original model $G(\mathbf{z};\theta)$.

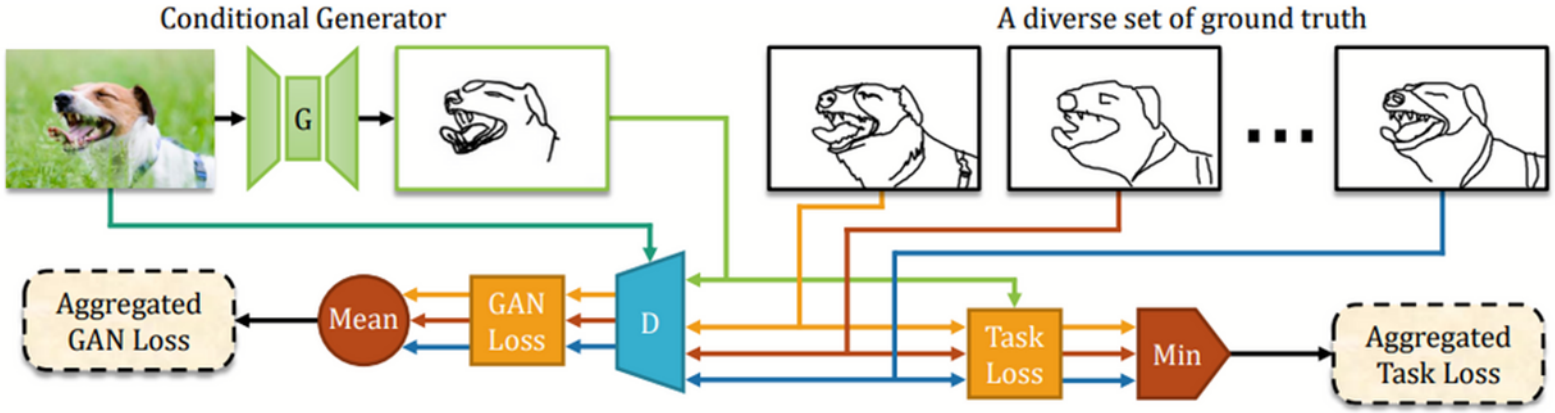


Fig: Architecture of PhotoSketch (F) used in Fig(2).

Cross Domain Adversarial Learning:

X, Y be the domain of image and sketches. Since Input sketch and output is realistic image, so it is more challenging to train the models. So, they introduce cross domain mapping network F:X->Y, which matches generated image to the sketch. Once the G'(z;θ') trained then, this mapping network is not used. After that G' should learn θ' that directly convert input sketch to realistic images.

Sketch Loss:

Discriminator output on real sketch image

$$\mathcal{L}_{\text{sketch}} = \mathbb{E}_{\mathbf{y}\sim p_{\text{data}}(\mathbf{y})} \log(D_Y(\mathbf{y})) + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \log(1 - D_Y(F(G(\mathbf{z})))), \quad (1)$$

Discriminator Output of Generated Sketch Image. See Fig-2 (a).

Image Space Loss & Regularization:

As stated in Fig(2) (b), this is the loss of realistic generated image by G'(z;θ') where discriminator Dx classify it as fake or real. Real image is sampled from original pretrained model G(z;θ).

See Fig 2 (b)

Discriminator output of real image

$$\mathcal{L}_{\text{image}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})} \log(D_X(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \log(1 - D_X(G(\mathbf{z}))). \quad (2)$$

Discriminator Ouput of Fake Image.

Also Regularization is simply the L1 norm between θ' and θ.

L1 loss

$$\mathcal{L}_{\text{weight}} = ||\theta' - \theta||_1.$$

Finally Full objective function as combination of Sketch Loss and Image Loss multiplied with regularizer.

Our full objective is:

$$\mathcal{L} = \mathcal{L}_{\text{sketch}} + \lambda_{\text{image}} \mathcal{L}_{\text{image}}, \quad (4)$$

The main goal is to learn θ' and minimize it of generator model G(x;θ') such that it should generate realistic image with given low dimensional noise vector (z). And discriminator classify Generated Image and the Real Image (Sampled from the another pre-trained Generator G(x, θ)) Where Dx and Dy want to maximize it.

$$\theta' = \arg\min_{\theta'} \max_{D_X, D_Y} \mathcal{L}. \quad (5)$$

Regularization Effects:

Model trained with regulaization, seems more realistic (Right), and viariation in qenerated output.

Figure 4. **Effect of regularization methods.** We compare models trained with (top) only $\mathcal{L}_{\text{sketch}}$, with (middle) weight regularization $\mathcal{L}_{\text{weight}}$, and with (bottom) image regularization $\mathcal{L}_{\text{image}}$. **(Left)** the snapshots at the same iteration. **(Right)** the best iterations selected based on our evaluation metric. We observe that the images look more realistic with either regularization applied, and the model trained with image regularization obtains better diversity than the one trained with weight regularization.