TroGAN Horse: Disguising Contours to Look Like Sketches

Akshay Dharmavaram, Mayank Mali
16-824 Final Report Fall 2021
{adharmav, mmali}@andrew.cmu.edu

Abstract

There has been an increased interest in employing sketches as a user-friendly representation to control the output of a generative model. Sketch Your Own GAN focuses on rewriting a generative model to output realistic images that are semantically similar to a handful of fixed, user-defined sketches. However, there are failure cases – GANSketching cannot replicate images from complex poses or distinctive, minimalist art styles (e.g. "the Picasso horse"). This is believed to occur because the user-sketch distribution is considerably different from the generated sketch distribution, given the underlying dataset. To rectify these failure cases, we propose a method that builds upon the GAN Sketching architecture by introducing a translation model that shifts the distribution of fake sketches to be more similar to that of the user-sketches while also retaining the essence of the initially generated image. Such a formulation avoids overfitting by the discriminator, thus reducing the discriminability and increasing gradient propagation. We also experiment with translating the user-sketches instead and discuss how the choice in the direction of translation affects generator performance. Finally, we show that our method reduces the number of training steps required when compared with GAN Sketching. Our code is available here: https://github.com/Aks-Dmv/TroGAN.git.

1. Introduction

Prior to the advent of the camera, creating realistic pictures based on a particular stylistic penchant required artistic skill and meticulous attention to detail, which resulted in exorbitant costs and man-hours. Nowadays, with a smartphone in most people's pockets we have a plethora of realistic image generation methods. However, we still have not overcome the barrier of editing images to match a particular style, at scale. There are applications such as Photoshop [9] that can edit images; however, they require dozens of man-hours and cannot be scaled to millions of images. In this paper we intend on investigating a neural approach to editing realistic images and mass producing variations using relatively simple sketches as standard archetypes.

GANs have recently been shown to be powerful tools in creating generative models for image synthesis, and there has been interest in being able to control the generated outputs [6, 11, 13]. In the GAN Sketching model [13], a generator is trained to create fake images similar to a human-provided drawing (the "user-sketch"). The goal of their architecture is to allow users to create their own generative network from only a single drawing or a handful of sketches.

As is discussed later, our hypothesis is that the ability for the generator to learn involves fooling the discriminator by mimicking the semantics of the sketch, which becomes increasing difficult the farther apart the distributions between the user-sketch and the generated-contours are. If the distributions are considerably far apart, the discriminator's job becomes "easy" and the generator cannot learn as well due to low gradient magnitudes. This becomes the case for user-sketches that are far from the fake sketch distribution, e.g. for user-sketches of cats in difficult poses, or minimalist, over-simplified drawings with rare sketch styles.

Our goal therefore becomes to understand how to rectify failure cases (discussed in Section 2) by bringing these two distributions closer together. In a sense, we want to make it harder for the sketch discriminator to discern between the two distributions so that gradients can better help the generator replicate the user-sketch styles or poses. It is important to note; however, that making the discriminator's inputs indiscernible from the get-go also is not desired as the generator will get little to no training signal.

2. Related Works

Being able to control the output of a generative model would enable engineers and scientists to create realistic images in a matter of seconds, which would have otherwise taken weeks or months to synthesize [3]. Moreover, there has recently been an increased interest in employing sketches as a user-friendly representation to control the output of a generative model [13]. Rewriting generative models [1, 14] is one such approach that enables fast adaptation of pre-trained models for customized image synthesis.

The framework presented in this paper builds upon the GAN rewriting work by Wang *et al*. [13].

2.1. GAN Sketching

The GAN sketching framework involves initializing a generative model with a set of pre-trained weights, followed by a training regiment that encourages the model to match a set of user defined sketches. The architecture is illustrated in Fig 1. The type of object to mimic (e.g. horse, cat, etc.) is either known beforehand or identified by a human-drawn user-sketch. A generator and two discriminators are trained end-to-end. The framework employs two soft constraints that are defined by two adversarial losses – a style loss and a sketch loss. One discriminator provides a style loss of the generated image using a dataset of real images of the object, and the other one (the main focus of our paper) provides a loss of the sketch outlines of the generated image against one or a handful (single-sketch and multi-sketch paradigm, respectfully) human-drawn sketches. The sketch of the generated image are provided by an image to sketch translation network (a pre-trained pix2pix [7] network). Note that the network must train end-to-end for every new sketch(es) provided. In this paper, "sketches" or "fake sketches" will refer to the output of fake images under this translation network, "human/user-sketch" refers to the human-drawn sketch, and "contours" (not yet introduced) will refer to the sketch-translation of real images under some image translation network.

The flow of the algorithm involves first sampling a generated image $G(z)$ from the generator. Next, this image is passed into a mapping or encoding network represented by $\mathcal{F}$. In GAN Sketching [13], the mapping network is initialized using a pre-trained network similar to Photosketch [8] and identifies the contours in the image. The output at this stage $F(G(z))$ would look similar to a sketch drawn by a human, and is thus used as the fake images for training a sketch discriminator. The real user-sketches used to train the sketch discriminator would come from a subset of the QuickDraw dataset [2]. Finally, the original sampled image $G(z)$ is passed into a style discriminator along with other real images to ensure that the style remains intact.

2.2. Failure Cases

In GAN Sketching [13], the authors identify a few failure-cases for their architecture, one such case being user-sketches drawn with a distinctive style. They illustrate this issue with an example of a horse drawn to imitate the style of Picasso. This example demonstrates the sketch-space-mismatch issue with this framework, namely the problem arising from a distributional shift in the user-sketches and the generated sketches. We will build upon the issues arising from this mismatch shortly, while also presenting some empirical analysis in the Experiments 4 section.

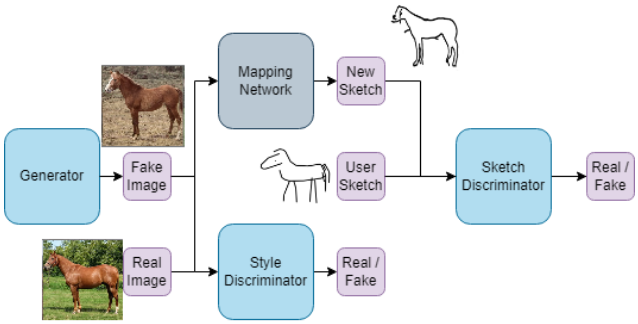


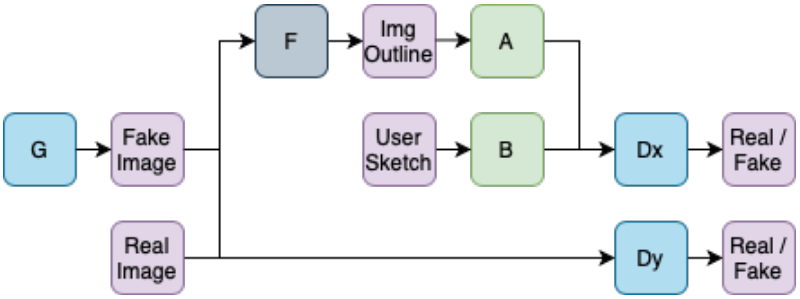
Figure 1. GAN Sketching Architecture: The main forward pass involves sampling an image from the generator, passing it through an encoding network or mapping network, followed by a sketch discriminator and a style discriminator.

The Picasso Horse [13] and other rare drawing styles example sheds light on the interesting problems surrounding sketch generation. This discussion motivates our choice to use an unpaired image-2-image translation model [10] as a method to bridge the gap arising from the distributional shifts between the sketch space of the model and the user-sketches.

2.3. Distributional Shifts

The central objective behind the adversarial min-max problem [4] is to converge to a Nash equilibrium point which describes a state where the generator's outputs are indistinguishable from the ground-truth samples. However, if there is still a considerably distinct deviation between the two distributions when the model converges, then a discriminator might overfit to the classification objective, thereby resulting in vanishing gradients passing back to the generator. This can be fixed with approaches such as increasing the number of generator update steps; however, this doesn't solve the underlying issue of the current GAN Sketching [13] framework on rare drawing styles. The issue with the current framework is that the fake sketches arising from the generated samples cannot match the user-sketches as they are from entirely different distributions even after the model has converged.

This leads us back to the issues discussed in the previous sub-section and motivates the design choices that we will present in the following sections. We aim to solve the distributional mismatch issue with an unpaired image-2-image translation model [10], thus resulting in the possibility for the generator to potentially converge to a Nash equilibrium. Furthermore, this avoids the overfitting issue that we would otherwise see when the discriminator optimizes over semantically inappropriate features, which would not result in meaningful weight updates for the generator.



(a) Generalized experimental template.

Variant	A	B
v1	SketchRNN	Identity
v2	Identity	SketchRNN
v3	Identity	U-Net
v4	Identity	CUT
v5	CUT	Identity

(b) A table of architectural variants (which networks go in A,B in 2a).

Figure 2. (Left) the general template for our experimental architectures. Rectifying the distribution between user-sketches and fake-sketches involve putting image-translation models (ResNet18) in slots A or B. (Right) a summary of experimental variants in terms of what networks go in slots A,B in 2a

3. Method

In order to rectify the distributional shift, we introduce a translation model that modifies one sketch distribution as described in Figure 2a. Our goal is to modify either sketch distributions in a way that closes the gap between the two distributions, while also retaining the original meaning of the user-sketch. We acknowledge that incorporating a translation model between the output of the mapping network F and the sketch discriminator could cause sparse gradients due to the depth of the pipeline.

3.1. Architectural Variants

While we initially planned to use the translation model to modify ground-truth user-sketches, we also experiment with rectifying the distribution shift by transforming the user-sketches as shown in Figure 2. The general template of our modified GAN Sketching model is shown in Figure 2a and the variants listed in 2b. We also try experimenting with biasing the training dataset for GAN Sketching by identifying nearest-neighbor real images to the user-sketch (in the sketch domain). Note that we did not implement v1, v2, or v3, but discuss observations or anticipations of each variant below.

3.2. Differences between A and B

We have a choice of inserting an image translation model in either A or B, while leaving the other slot empty. If a model is inserted into A, then it is intended to serve as fake sketch to user-sketch image translation. The general idea is to pre-train a translation network to make fake sketches look more like user-sketch domain, and these variants are v1 & v5, as shown in 2b. On the other hand, if a model is inserted into B, it is intended to translate sketches from the user-sketch to the fake sketch distribution. Ideally a pre-trained model is used, and these variants are v2,v3, and v4, as shown in 2b.

3.3. SketchRNN Variants (v1, v2)

Inspired by the observation that fake sketches usually have more numerous and smooth strokes than user-sketches, we hypothesized that a recurrent model such as SketchRNN [5] will sequentially add sketches to the user-sketch to look more like fake-sketches (v2). Conversely, we might imagine a scheme where strokes are sequentially removed from the fake-sketch to look more like the user-sketch (v1). Although recurrent networks would theoretically add/remove strokes one by one in this hypothesized manner, v1 inserts a RNN in an already deep pipeline and training may suffer from vanishing gradients. Both variants need to train with multi-step rollouts and the number of steps may add additional complexity as an unknown hyperparameter. Furthermore, pre-training these network requires training with image pairs, which requires human labeling or some semi-supervised paradigm with nearest neighbor search. In any case, we decided to employ non-recurrent image-to-image translation models.

3.4. U-NET Variants (v3)

U-nets are shown to be powerful tools for image-to-image translation [12], so our v3 variant proposes to insert a U-net model in slot B, where the network is pre-trained to translate user-sketches to fake sketches. The main difficulty with this model is that it still requires image pairs to translate. Unlike SketchRNN, U-net is not a recurrent model and requires no rollouts, but the image-pair creation problem still persists and we would like to proceed with unsupervised translation models.

3.5. CUT Variants (v4, v5)

The contrastive unpaired translation model (CUT) [10] is a good model candidate since it does not require supervised pair of images. It works by training an encoder-decoder generation model to translate image styles while learning to keep corresponding patches of the input and output se-

mantically similar. The use of CUT will eliminate multi-step rollouts as in SketchRNN and supervised image pairing during pre-training. v4 introduces a CUT model in B, and v5 introduces a CUT model in A. For v4 since we are transforming user-sketch to look like fake sketches (which often have non-horse related strokes), the translated image may not be semantically correct, although having the style of these fake sketches. We implement v4 and v5 and identify some good results with v5 later, which still may need more samples or training.

4. Experiments

We hypothesize that introducing an image-2-image translation model to the GAN Sketching [13] model will rectify the distributional shift between the generated sketch and user-sketch, which will thereby result in improved performance – both qualitatively and quantitatively. In this section we present experiments that justify our claim, along with a few failure cases as well.

4.1. Evaluations

Datasets. To compare our model against the GAN Sketching baseline we needed to first construct an unpaired image-2-image translation dataset, consisting of horse contours and sketches. We collected 200 horse contours by sampling 400 pictures of horses from the LSUN Dataset [15], passing them through the QuickDraw model [2], and selecting the top 200 qualitatively appropriate horse contours.

Next, we sampled a subset of 25,000 horse images from the LSUN Dataset for evaluating our image loss. We have included instructions on how to download this dataset in the repository released along with this paper. Finally, we obtained a set of 200 sketches from the QuickDraw dataset [2] by obtaining JSON files with stroke information and rendering the vector images. Using this process, we qualitatively evaluate the sketches and select the best 200 for our task.

Performance Metrics. Apart from a qualitative evaluation of our models, we also evaluate the number of horses that were initially in an incorrect configuration, and the number that remained incorrect after 20,000 epochs.

We also plot the Generator Sketch Loss, which is described as: $\mathrm{softplus}(-D(G(z)))$, where $G(z)$ is the generated sketch and D is the sketch discriminator. This loss quantitatively evaluates the degree to which the generator is able to fool the sketch discriminator. In other words, the generator sketch loss evaluates the number of false positives by this discriminator.

4.2. Generator Sketch Loss

In Fig. 3, we plot the generator sketch loss for the GAN Sketching Baseline, the TroGAN sketch-to-contour (s2c, v4) model, and the TroGAN contour-to-sketch (c2s, v5) model. The loss is defined as $\mathrm{softplus}(-D(G(z)))$, where G is the generated sketch and D is the discriminator output given a sketch. This loss quantitatively evaluates the degree to which the generator is able to fool the sketch discriminator. This metric was chosen as a quantitative approach to validate our hypothesis, namely that introducing an image-2-image translation model would allow for better gradient propagation, and lower generator losses. We notice that the baseline has considerably greater mean and variance compared to the both TroGAN models. Though these results must be taken in conjunction with the qualitative analysis, we show that the introduction of a CUT [10] model improves training dynamics.

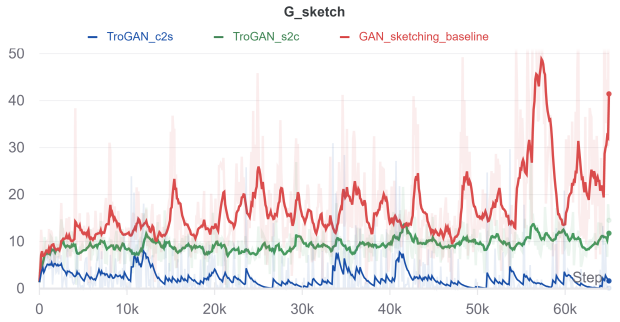


Figure 3. The Generator sketch loss defined as: $\mathrm{softplus}(-D(G(z)))$. A lower loss indicates an increased predilection by the discriminator into believing the generated images are indeed real. We observe higher mean and variance in the baseline (red) compared to the two TroGAN architectures (blue and green).

4.3. Faster Convergence

As described in Fig. 3, the TroGAN architectures have richer gradients, thereby resulting in faster convergence. We can see the results of this in Fig. 4. We notice that the baseline and TroGAN s2c have still not converged even after 20,000 epochs. However, the TroGAN c2s model has indeed converged. We have added more samples in the Appendix for further reference. We measure convergence based on the qualitative likeliness to the sketch – illustrated to the left of the dashed line in Fig. 4.

4.4. Interpolations

In this experiment, we show some interpolated examples between the horse samples. As expected, the TroGAN architectures fare better than the baseline. However, we can see that not all interpolations are semantically similar to the left-facing horse sketch. There could be various explanations, and one is overfitting by the architecture over exclusively the sampled images. This could mean that the bulk of the architecture is in fact the same, while only portions of

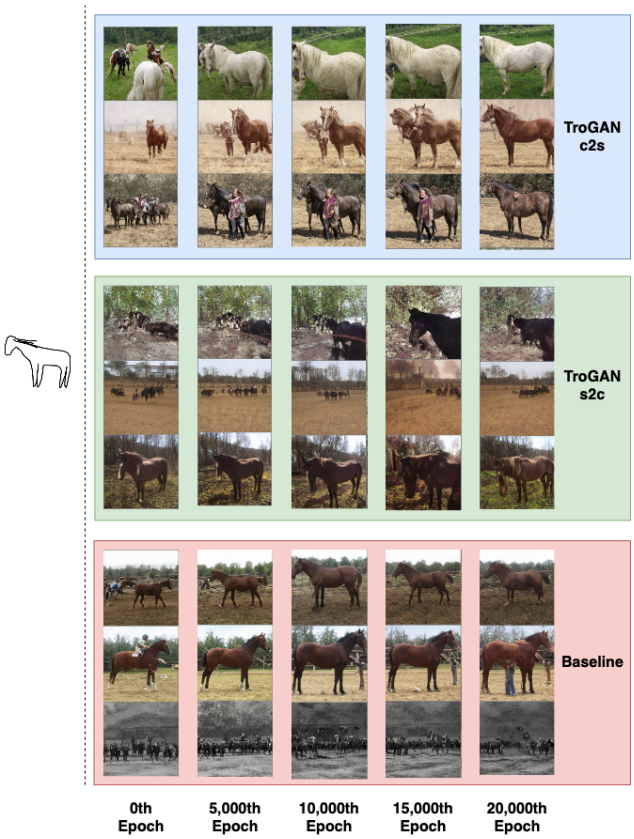


Figure 4. We illustrate randomly sampled outputs from the generator for time-steps at multiples of 5,000 for the baseline (red), the TroGAN s2c (green), and TroGAN c2s (blue). We show the sketch used for GAN rewriting to the left of the dotted line.

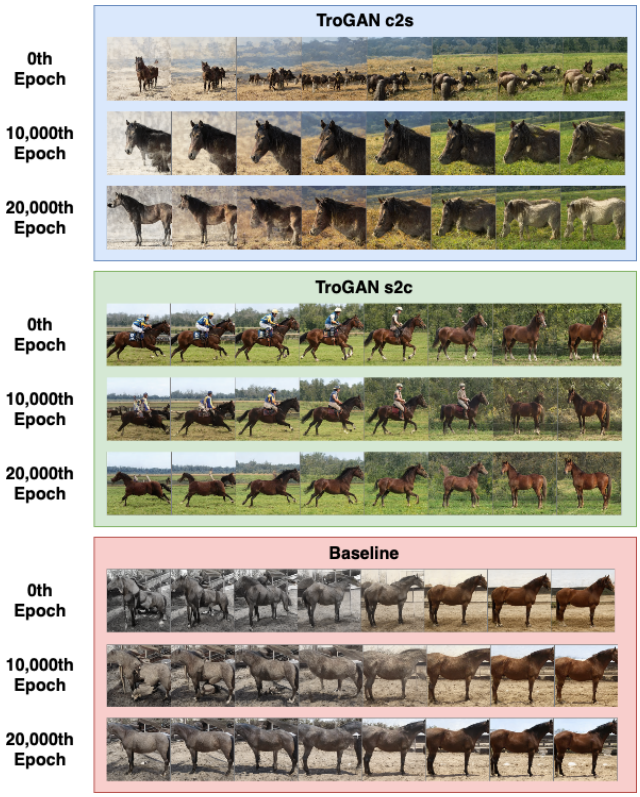


Figure 5. We illustrate randomly sampled interpolations between outputs from the generator for time-steps at multiples of 10,000 for the baseline (red), the TroGAN s2c (green), and TroGAN c2s (blue). The sketch used for GAN rewriting is the same as in Fig 4.

the state space have been modified to look like the sketch.

4.5. Semantically Incorrect Lines

We can train the translation model to transform sketches to contours as well contours to sketches. However, we observe that the TroGAN c2s model vastly outperforms TroGAN s2c. This can be attributed to the fact that the translation model from sketches-to-contours adds spurious details to the sketches, which need not be semantically correct. We show a few examples in Fig. 6. The addition of these lines that may not be semantically appropriate would be picked up as causal features by the Discriminator, and would be used for classification. However, these features were just stylistic renditions introduced by the CUT model to mimic the style of a contour. This significantly hampers the performance of the overall model, as seen in the results present previously. Thus, translating from contours-to-sketches is the optimal process, as the spurious features are removed, thus resulting in only the important features being taken into account by the Discriminator. This conclusion is bolstered by the improved results seen by TroGAN c2s over its s2c counterpart.

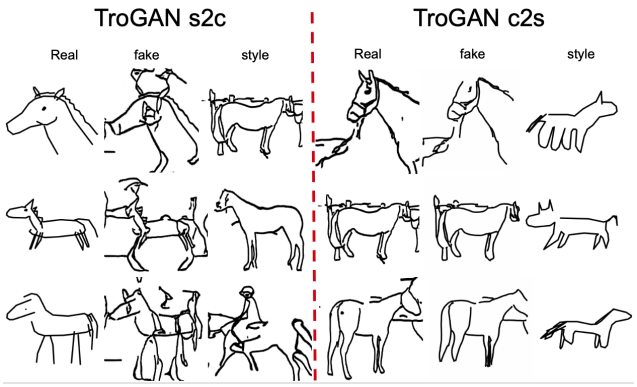


Figure 6. We show samples of contour and sketch translations after training, along with the style the model is trying to imitate.

4.6. Nearest Neighbors

Another idea we implemented was biasing the training dataset for our v4 model (CUT translates user-sketch to fake sketches). The hypothesis is that if the training images look more like the user-sketch semantically, then there will be

less generation diversity, and the fake distribution will be semantically closer to the desired target in the user-sketch domain.

We start with a pre-trained CUT model that translates user-sketches to fake sketches and keeps the translated user-sketch as a template. Then we translate the entire image dataset to the contour domain by using the F translation network used in [13]. Finally, we find the contours that have the smallest Chamfer Distance to the template in the sketch domain. Note that for simplicity, we find the contours of the sketches using OpenCV and compare these points to find the closest corresponding images. Using 1/5 nearest neighbors of the original $\tilde{2}5$K dataset to the template, training v4 using the picasso horse did not give results better than v4 on the original dataset (TroGAN_s2c in 3).

5. Conclusion

In general, we outline five methods 2 to rectify the distributional shifts between the sketch discriminator inputs, in order to assist in the generation of images that are semantically similar to hard/rare user-sketches. We implement and test two methods involving using CUT [10] to translate either distribution. Our results show that for the user-sketches that we tested upon, the output of the TroGAN c2s generator converges to the desired pose faster than the GAN Sketching model [13]. However, the picasso-horse sketch still proves to be difficult to emulate, and persists as a failure case in our experiments. This can be attributed to the issue of identifying *why* a failure case is a failure case, which is rather difficult. Our modified architecture seems to converge faster on normal input but does not verify our hypothesis on user-sketches with rare or distinctive styles. Without further testing, it is unclear whether modifying the sketch discriminator inputs in this way converges at different speeds for user-sketches closer or farther from distribution. We leave this to future work.

6. Acknowledgements

We would like to thank Sheng-Yu Wang and Prof. Jun-Yan Zhu for their invaluable insight and feedback without which we wouldn't have been able to complete this project.

References

[1] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *European Conference on Computer Vision*, pages 351–369. Springer, 2020. 1

[2] Salman Cheema, Sumit Gulwani, and Joseph LaViola. Quickdraw: improving drawing experience for geometric diagrams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1037–1064, 2012. 2, 4

[3] Andrew S Glassner. *Principles of digital image synthesis*. Elsevier, 2014. 1

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[5] David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. 3

[6] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 1

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[8] Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1403–1412. IEEE, 2019. 2

[9] Lev Manovich. Inside photoshop. *Computational Culture*, 2011. 1

[10] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer, 2020. 2, 3, 4, 6

[11] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *CoRR*, abs/2007.00653, 2020. 1

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[13] Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Sketch your own GAN. *CoRR*, abs/2108.02774, 2021. 1, 2, 4, 6

[14] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 218–234, 2018. 1

[15] 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

7. Appendix A

The figures in the paper only included a few samples, due to the space constraints. In this appendix, we provide a larger sampling of fake-images and interpolations for the reader to investigate.

(a) TroGAN C2S iteration=0

(b) TroGAN C2S iteration=10K

(c) TroGAN C2S iteration=20K

Figure 7. TroGAN C2S Samples Psi0.5

(a) TroGAN S2C iteration=0

(b) TroGAN S2C iteration=10K

(c) TroGAN S2C iteration=20K

Figure 8. TroGAN S2C Samples Psi0.5

(a) GAN Sketching iteration=0

(b) GAN Sketching iteration=10K

(c) GAN Sketching iteration=20K

Figure 9. GAN Sketching Baseline Samples Psi0.5

(a) TroGAN C2S iteration=0

(b) TroGAN C2S iteration=10K

(c) TroGAN C2S iteration=20K

Figure 10. TroGAN C2S Interp Psi0.5

(a) TroGAN S2C iteration=0

(b) TroGAN S2C iteration=10K

(c) TroGAN S2C iteration=20K

Figure 11. TroGAN S2C Interp Psi0.5

(a) GAN Sketching iteration=0

(b) GAN Sketching iteration=10K

(c) GAN Sketching iteration=20K

Figure 12. GAN Sketching Baseline Interp Psi0.5