Sketch-to-Photo, Image-to-Image Translation with GAN

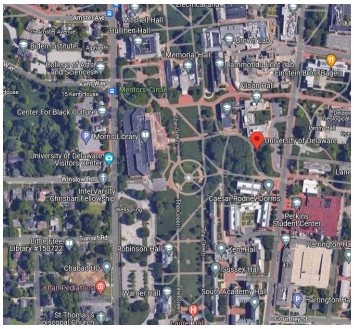
Michael Bocelli

Abstract - This paper serves to investigate the capabilities of generative adversarial networks (GANs) in regards to image-to-image translation. Many prior works have experimented with image translation as a task for GANs, and this paper intends to add to the growing field by specifically investigating the problem of translating an image from an abstract domain to one of higher fidelity. Through testing different GAN architectures, we hope to find what makes a given model a good fit for this particular type of translation. Specifically, the paper will compare the results of a model that relies on paired training data, and one that uses unpaired generation. It will be demonstrated that paired training dominates when translating an image from an abstract domain, however, this does not mean that the unpaired model is not without its use cases. For instance, the reader will see that the unpaired model produces two generators for bidirectional translation, and perhaps could be trained further to better match the results of the paired model.

I. INTRODUCTION

Interpretations of the abstract are subjective. One may see a face in a cluster of brush strokes whereas another may visualize an animal, all unique translations from what we see to what we imagine. However, if we constrain these translations to specific domains, we may be able to materialize them via machine learning. Specifically, if we present a machine with an image, to what extent can it translate that image into something else? The task of this paper is to leverage this idea of machine generation and investigate how well a machine can translate extremely abstract data into another domain.

On the topic of image-to-image translation, consider Google Maps. At any location in the world, users are able to view the map from two different lenses, “Satellite” view and “Default” view, as seen in Figure 1.



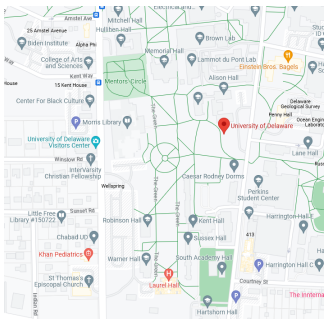


Fig. 1. An example of a one-to-one mapping between “Satellite” (a) and “Default” (b) view. Adapted from [1].

To achieve the simplistic “Default” view for every area in the map, without having to meticulously draw a one-to-one mapping from satellite imagery of the entire world, you could train a machine learning model, known as a generative adversarial network (GAN), to translate satellite images to the simplistic, map images seen in Fig. 1 (a). In this example, we are translating from an extremely detailed image domain to a more abstract one, so the model used here would have more information than it needs, making it an easier task. However, as aforementioned, we want to investigate the reverse. What if the input image had very little information, could Fig. 1 (b) be used to generate Fig. 1 (a)? This is the more challenging task, as sometimes we lack more information than we need, for instance, forensic sketches to profile suspects. If we could translate an abstract sketch to a more realistic image, then the public may have a better chance of identification. Even in creative areas such as independent game development, who lack the resources to produce photorealistic textures, this technology could assist in the concept art to texture pipeline. This paper will investigate various GAN architectures’ abilities to perform image-to-image translation when provided with abstract data.

II. RELATED WORK

Generative Adversarial Networks (GANs) gained prominence following Goodfellow et al.’s paper of the same name [2]. Like the title implies, the framework of a GAN consists of an adversarial game, in which two models compete. The game is as follows: three probability distributions are created, one for the original data (set of images), one for the generator model (G), and the other for the discriminator model (D). Both G and D are multilayer perceptrons, where D is trained to discriminate between images from the original data and images from G by producing a scalar value, which relates the probability of the image being from the original data [2, p. 2]. G’s task is to produce images that maximize D’s error in its labeling of real (from original data) and fake (from G), in other words, to fool D. After sufficient training time and data quantity, G will produce images that perfectly match the original data’s distribution, and D will not be able to discriminate between the two, generating a value of 0.5 (meaning the data has a 50% chance of being real, indicating that D has little confidence in differentiating the distributions) [2, p. 3]. The game is visualized in Figure 2.

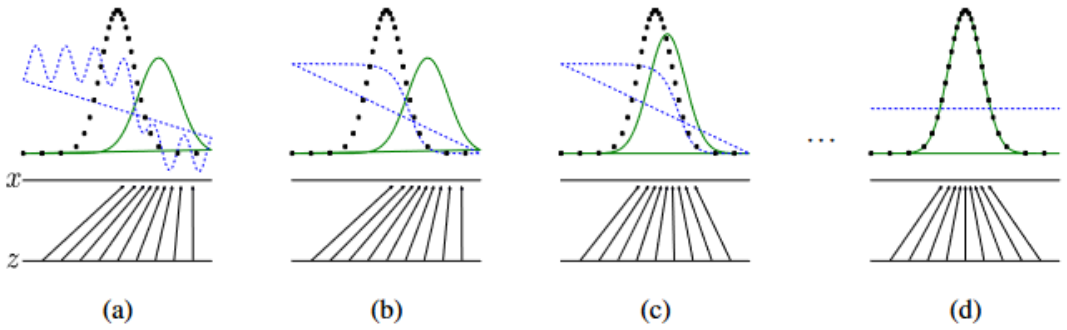


Fig. 2. Distribution of original data is represented by the black, dotted curve. Generated image distribution is in green, and the discriminative distribution is in blue. As the model progresses from a to d, G learns to fool D by fitting perfectly with the original data distribution. Adapted from [2, p. 3].

This is the theoretical framework for a GAN model, and Goodfellow et al. [2, p. 7] acknowledges that there are disadvantages in training, such that G and D must be synchronized well so that one does not dominate the other and thwart the overall training of G.

With the success of GAN came variations of the framework, such as **conditional GANs (cGANs)**. This method of image generation is outlined in Isola et al. [3], which is responsible for the Pix2Pix architecture used in the experiments of this project. Essentially, the output of the generator is conditioned on an input image, which means both G and D are fed the input image. In contrast, the unconditional architecture only provides a random noise vector to G, which is then forced to condition on the output of D and its loss function [3, p. 2]. This makes cGANs ideal for image-to-image translations tasks, as the problem has an inherent mapping that needs to be solved, and one can provide instances of this mapping via a paired dataset for the cGAN to learn. Since a cGAN's loss is also learned, outputs that differ structurally from inputs will be penalized more than in a traditional GAN architecture, which contributes to cGAN being an idealistic approach to image translation [3, p. 2].

Another GAN framework, known as **Cycle-Consistent Adversarial Network (CycleGAN)**, as described in Jun-Yan Zhu et al. [4], follows an unpaired image approach, in which the authors propose that to translate an image from one domain to another, a dataset of paired images is not required. CycleGAN relies on having two generators and two discriminators which are trained via a cycle consistency and identity loss [4, p. 3]. The evaluation of both the cGAN Pix2Pix model and CycleGAN adopted use of an FCN score, which generates a "label map" [4, p. 6] for a generated photo to be compared against the ground truths of the data. On average, CycleGAN reported a lower per-pixel accuracy than Pix2Pix's cGAN framework, however still outperformed most other models in its evaluation [4, p. 7].

III. METHOD

A. Model Choice And Dataset

As aforementioned, this paper employs both the Pix2Pix cGAN architecture, and the CycleGAN architecture to investigate abstract image-to-image translation with GAN models. This task is approached as a generation task, particularly conditional generation, as our outputs are conditioned on the image domain in our input space. With this in mind, a dataset was chosen that consists of paired images for its training and testing data, where one image in the pair represents the abstract data we want to translate from, and the second image represents what we want to translate to. The dataset "Person Face Sketches" [5] was chosen, which consists of roughly 20,000 pairs of black-and-white sketch portraits X and their corresponding photographic images Y (which are pulled from the CelebA dataset [6]). An example pair is shown in Figure 3.

Fig. 3. Image on the left (X) represents a sketch and the image on the right (Y) represents the photo which the sketch is based on.

Given the limited scope of this project due to available computing power, only a subset of this data was used, starting at 1,000 image pairs for training data. Furthermore, the sketches are not human drawn, rather the result of filtering out the color channels of the photos and eliminating various details, while retaining the identifying characteristics [5].

To carry out the experiments, the official implementations of the Pix2Pix and CycleGAN architectures were used, which are provided online by the original authors. Pix2Pix uses what's known as "U-Net" for its generator. The structure of U-Net follows an encoder-decoder architecture, where the input image is repeatedly downsampled so that lower level features are observed/extracted. Then, the decoder reconstructs the image with the provided parameters learned from previous losses so that the output is structurally similar to the input (as we are performing translation), but visually different. The U-Net generator is unique in the fact that it provides "skips" in the encoding-decoding process, where information that remains the same between input and output is shuttled directly to the output layer, rather than repeatedly broken down during the encoding process (such as edges) [3, p. 3]. This adds a layer of efficiency to the Pix2Pix architecture, since less lower level information is needed to be fully fed to the output layer with each translation. The discriminator, PatchGAN, is pioneered by the authors of Pix2Pix, and is run convolutionally across the whole image. Instead of classifying the global structure as real or fake, it uses a square kernel and LeakyReLU activation function to produce its output of whether or not the given patch of the image it is looking at is real or fake. It then averages the output scalars of all these patches for its final decision. This allows the model to capture local features better, as image generation trained with mean absolute error/L1 loss often results in blurry output, due to the discriminator scrutinizing globally [3, p. 4].

The CycleGAN architecture, as aforementioned, uses 2 generators and 2 discriminators. If generator G translates X (a sketch) → Y (a photo), then generator F translates Y → X. Then, discriminator A will judge the output of G and discriminator B will do the same for F [4, p. 4]. The implementation of CycleGAN differs slightly from the paper, as the generator and discriminator models use the same model architectures as those from the Pix2Pix implementation for simplicity, while the original paper uses a modified ResNET for its generators [7].

B. Model Training

In a GAN model, both the generator and discriminator must be trained. Therefore, for both Pix2Pix and CycleGAN we have multiple loss functions. In Pix2Pix, we must minimize the sigmoid cross entropy loss between the generated image's score from the discriminator and the label for a real photo (array of ones). We also need to minimize the L1 loss between the generated image and the input sketch's corresponding label (photo image). These two losses are combined along with a constant for a total loss, and then the calculated gradients are applied to an Adam optimizer. The discriminator is trained with two loss functions as well, however, both are sigmoid cross entropy. The first is between the discriminator's assessment of a real photo and an array of ones (since a value of 1 indicates 100% chance that the image came from the dataset), and the second is between the assessment of a generated image and an array of zeros. Similarly, these losses are combined for a total loss which is then used to update the gradients, and subsequently the model [8].

For CycleGAN, since there is no paired input data, it would not make sense to perform cross-entropy losses to establish a mapping between the domains. Instead, a cycle consistency loss is calculated, where cycle consistency can be thought of as transitivity. For example, if generator G translates image X to Y, and generator F translates Y to X, then a loss is measured by passing X to G, then the resulting Y' to F, and finally comparing the L1 loss between the resulting X' and X [7]. The second loss function is identity loss. After passing X to F, L1 loss is calculated between X and X'. Theoretically if F was a perfect photo to sketch translator, then X' should be identical to X, so the loss should be minimized [4, p. 9].

Various adjustments needed to be made in order to process this particular dataset. For instance, the original models are built on 256x256 images, so convolutions needed to be added to account for the 512x512 size of *Person Face Sketches*. Additionally, the data input pipeline needed to be reworked to fit the structure of this particular dataset, which had each sketch-photo pair as separate images in different directories, whereas Pix2Pix assumes both were in the same image. This constituted a majority of the implementation time for both architectures.

IV. EXPERIMENTS

A. Pix2pix Model 1

Three models were trained using the Pix2Pix architecture. The first consisted of 1000 image pairs, which were resized to 64x64. Data augmentation was performed during the input pipeline, in which random jittering was applied to each pair (this included random crop and horizontal flip). Furthermore, the values of the tensor arrays for each image were normalized between -1 to 1. The training loop is very simple and consists of four main steps:

1. Input given sketch to generator and receive output
2. Input sketch, generated image, and sketch's target photo to the discriminator
3. Calculate losses

4. Calculate gradients of losses w.r.t. generator and discriminator values and apply them to Adam optimizer [8]

Figure 4 portrays some results after 40,000 iterations of the training loop (20 epochs).

Fig. 4. Two examples of testing instances being fed to the generator after training for 40,000 iterations.

Evidently, there was a problem with the input pipeline which resulted in overexposure in both the sketch and the photo. Subsequently, the generator had less detail to learn from, which resulted in unsatisfactory results which will be analyzed in the next section.

B. Pix2pix Model 2

For the second model trained with the Pix2Pix architecture, the training size was reduced to 400 image pairs and the full 512x512 resolution was utilized. However, the exposure issue persisted, resulting in the following example in Figure 5 from running the testing data through the generator.

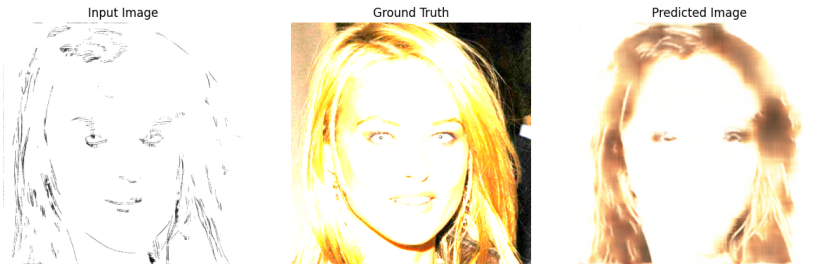


Fig. 5. One example test instance being fed to the generator after 40,000 iterations.

C. Pix2pix Model 3

Fortunately, the bug resulting in the overexposure was discovered to be in the normalization process, in which the constant used for dividing the dimensions of the image tensors was incorrect. This resulted in much cleaner results, even after just the first 1,000 iterations as seen in Figure 6.

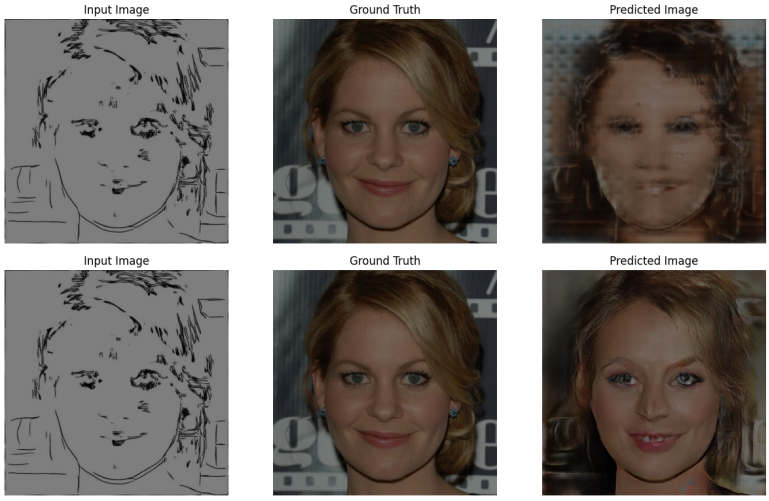


Fig. 6. The same training instance after 1,000 after 40,000 iterations respectively from top to bottom.

D. CycleGAN Model 1

Only one model was trained using the CycleGAN architecture. This model was trained for 20 epochs with 400 256x256 image pairs in the following training loop:

1. Getting predicted image
2. Calculating the loos
3. Calculating the gradients via backpropagation
4. Applying the gradients to the Adam optimizer [7]

Figure 7 demonstrates the results after all 20 epochs of training:

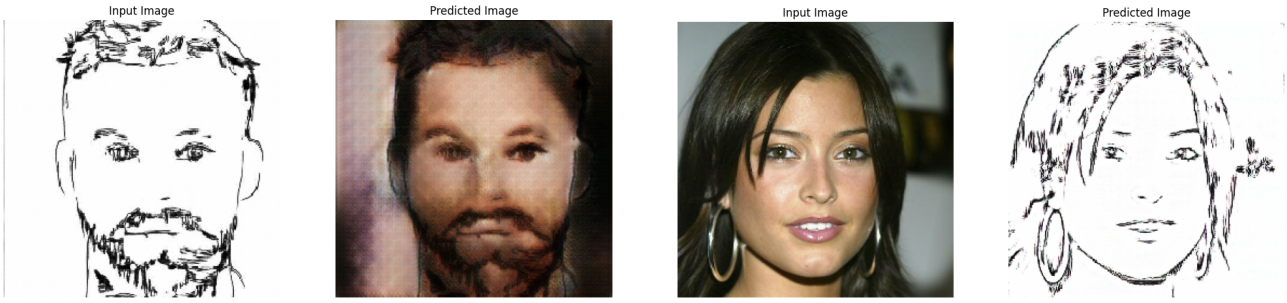


Fig. 7. Image on the left (a) demonstrates a training example being passed through the first generator which handles sketch to photo translation. The right image (b) demonstrates a test example being passed through the second generator which handles photo to sketch translation.

As expected, the generator which conditions on input with more detail was able to generate better output as seen in Fig. 7 (b).

V. EVALUATION

The loss of the Pix2Pix models was graphed during training in Figure 8.

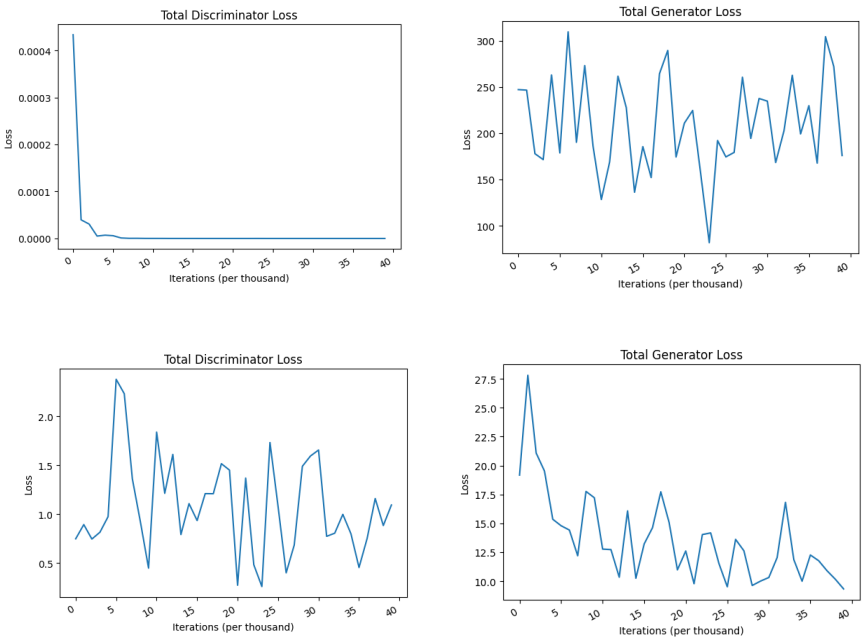


Fig. 10. Total discriminator and generator losses for Pix2Pix models 2 and 3 respectively from top down.

It is clear that model 2 with the normalization bug resulted in the discriminator dominating the adversarial game, resulting in the generator being unable to learn given the low detail of the input. In contrast, model 3 had much better performance, with neither submodel dominating the other too early on, illustrating the synchronization during training, as described in [2].

Measuring accuracy of GAN models is not an easy problem. Luckily, since we have target images to reference with the translation task, we are able to measure the similarity between the distribution of generated images to the distribution of the target images via Fréchet Inception Distance (FID), as implemented in [9].

Architecture	**Model**	**FID Score**
Pix2Pix	2	~290.15
Pix2Pix	3	~67.23
CycleGAN	1	~145.72

Table 1: FID scores calculated between ground truths in testing dataset and generated outputs given inputs from testing dataset.

When comparing these scores, it is clear that the distance between the distribution of generated images to ground truths is smallest for Pix2Pix model 3. This aligns with CycleGAN's original analysis where pixel to pixel accuracy was higher for the Pix2Pix model in comparison to their own.

VI. CONCLUSION

A valid interpretation of the results would surmise that image-to-image translation from an abstract input is, in fact, possible with GAN models. It is also clear that the translation performs better when given a paired dataset, however, the quality of the unpaired model could be improved given more data and epochs to run. Quantity of training data was a primary impediment on the quality of the models trained in this experiment. [9] references how a FID score is more meaningful once data is above 1,000 in quantity. Unfortunately, time limited training in this case to 800 total training images for the models with higher resolution images. Despite this, the experiments with the Pix2Pix architecture in particular have positive implications for the future of image-to-image translation where the input domain is more abstract than the target.

REFERENCES

[1] Google maps, https://www.google.com/maps (accessed Aug. 11, 2023).

[2] I. J. Goodfellow et al., "Generative Adversarial Networks," arXiv (Cornell University), June. 2014, doi: https://doi.org/10.48550/arxiv.1406.2661. Available: https://arxiv.org/pdf/1406.2661.pdf (accessed: Aug. 10, 2023).

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros. Research, "Image-to-Image Translation with Conditional Adversarial Networks," Berkeley AI Research (BAIR) Laboratory, UC Berkeley, Nov. 2018. Available: https://arxiv.org/pdf/1611.07004.pdf (accessed: Aug. 10, 2023).

[4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017 IEEE International Conference on Computer Vision (ICCV), Mar. 2017. doi:10.1109/iccv.2017.244 (accessed: Aug. 11, 2023).

[5] J. Xu, "Person Face Sketches," www.kaggle.com, Jul. 11, 2022. Available: https://www.kaggle.com/datasets/almightyj/person-face-sketches (accessed: Aug. 10, 2023).

[6] Switchable Normalizations, "CelebAMask-HQ," GitHub, Jul. 10, 2020. Available: https://github.com/switchablenorms/CelebAMask-HQ (accessed: Aug. 10, 2023).

[7] "CycleGAN : Tensorflow Core," TensorFlow, https://www.tensorflow.org/tutorials/generative/cyclegan (accessed Aug. 11, 2023).

[8] "Pix2pix: Image-to-image translation with a conditional Gan : Tensorflow Core," TensorFlow, https://www.tensorflow.org/tutorials/generative/pix2pix (accessed Aug. 11, 2023).

[9] Bioinf-Jku, "Bioinf-JKU/TTUR: Two time-scale update rule for training gans," GitHub, https://github.com/bioinf-jku/TTUR (accessed Aug. 11, 2023).