Image-to-Image Translation
(CSE 803 Project Report)

Asadullah Hill Galib (Single Project)
CSE, Michigan State University
galibasa@msu.edu

Abstract

Image-to-image translation is a popular and growing field in computer vision that deals with many sorts of mapping between an input image and an output image. This field is booming currently thanks to the incorporation of deep learning techniques, particularly generative modeling techniques. It has a variety of applications, such as image synthesis, segmentation, style transfer, restoration, and pose estimation, etc. In this project, edge-to-image - an application of image-to-image translation will be explored. The goal of this project is to generate colored images from sketches using a generative model. Conditional GAN-based architecture is incorporated to accomplish the goal.

Figure 1. Examples of Image-to-Image Translation (image source: [8])

1. Problem Definition

The goal of image-to-image translation is to transfer images from one domain to another while keeping the content representations intact. Due to the introduction of deep learning techniques, notably generative modeling approaches, this subject is now thriving. It has a wide range of applications, including translation of images from day to night, translation of images from season to different seasons, translation of edge to image, translation of semantic segmentation to image, translation of satellite photographs to Google Maps, translation of black and white photographs to color, translation of sketches to color photographs, etc. Figure 1 refers some examples of image-to-image translation.

Generically, all of the aforementioned applications have tried to generate output images condition on the input images. This conditional theme for the image to image translation is promulgated by Isola et al. [5] in their benchmark study in this field. They employed a model called the conditional Generative Adversarial Network [6], or cGAN for image-to-image translation in general, where the generation of the output image is conditional on the input image. This cGAN is a type of GAN [3] architecture. Followed by this benchmark work, a lot of variations of the GAN architecture are proposed for image-to-image translation, such as CycleGAN [11], StarGAN [2], AttGAN [4], etc.

This project aims at generating images from edges/sketches. Given, edge image representation and the corresponding original image, a cGAN model will be trained as described in [5] for generating and discriminating produced/output image. The discriminator is given a source image and a target image and is asked to decide whether the target is a realistic transformation of the source image. Adversarial loss is used to train the generator, which encourages it to generate believable images in the target domain. L1 loss between the generated image and the intended output image is also used to update the generator. The generator model is encouraged to construct credible translations of the original image as a result of the additional loss.

So, any dataset containing edge image - original image pair can be used here. This kind of datasets exists, such as Danbooru [10], Pokemon [1], etc. Also, automatic edge detected (using edge detector techniques) from sample images can also be used as the dataset. In terms of evaluation, mean absolute error and cross-entropy are used. According to [5], the underlying U-net-based (with skip connection) generator has been implemented first. On top of that, the cGAN architecture has been implemented combining gen-

erator and discriminator. Finally, the U-net-based generator and the full cGAN architecture are evaluated and compared.

Section 2 describes background, Section 3 describes Methodology, Section 4 describes Experimental Evaluation, Section 5 describes challenges, and Section 6 concludes the report.

2. Background

This section describes the GAN and cGAN algorithms.

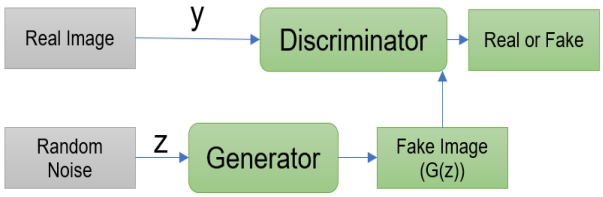


Figure 2. GAN architecture

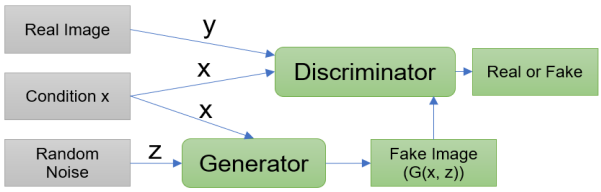


Figure 3. cGAN architecture

2.1. GAN

GANs (generative adversarial networks) [3] are a type of artificial intelligence algorithm created to tackle the challenge of generative modeling. In order to perform unsupervised learning, the GAN creates fake / random-looking data and attempts to distinguish whether a sample is created fake data or actual data. Here, the two components - Generator and Discriminator compete with each other to become more accurate in their task. In an image generation setting, the generator starts with a random noise image and attempts to create fake images that are similar to those in the training set. Both the original and produced images are sent into the discriminator. It attempts to distinguish between genuine and fake images. This cycle is continued as long as the discriminator should be unable to distinguish between genuine and fake images. In other words, it is minmax game between Generator and Discriminator, and the algorithm tries to reach the Nash equilibrium where both components are in a win-win situation. The overall architecture cGAN is described in Fig. 2

2.2. Conditional GAN - cGAN

Despite the fact that GAN models may produce new random fake images for a given dataset, there is no method to regulate the sorts of pictures that are generated other than attempting to decipher the non-linear relationship between the latent space supplied to the generator and the generated images. Conditional GAN aka. cGAN [7] closes this gap by introducing conditions on the generator as well as descriptor. In short, cGAN uses a generator model to conditionally generate pictures. If a class label is supplied, picture generation can be conditional on it, allowing for the targeted production of images of a specific type. The overall architecture cGAN is described in Fig. 3

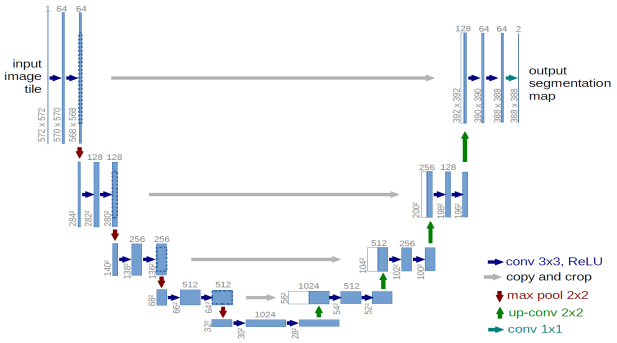


Figure 4. Generator: U-net architecture (image source: [9])

3. Methodology

This section describes the overall methodology of the project, i.e., the underlying architecture, details, and loss functions.

3.1. Generator

The first part of the GAN architecture: Generator is implemented using the underlying U-net convolutional architecture [9]. This U-net-based Generator consists of an encoder-decoder pair with skip connections. The encoder down-samples the original image sequentially using Conv2D, ReLU, Batch Normalization layers with strides. The decoder takes the down-sampled version of the original image and starts up-sampling it in a reversed way and generates the output. To guard against spatial information losing and to avoid vanishing/exploding gradient problems, skip connections are used from the individual layer of the encoder to the corresponding layer of the decoder. The overview of the U-net architecture is shown in Fig. 4

3.2. Discriminator

In comparison to the generator, the discriminator merely has the encoder unit. Its goal is to determine whether the sketch-image combination in the input is real or fake. The network has been trained to achieve the highest classification accuracy possible.

So, any image classifier algorithm can be used as the Dis-

criminator. A convolutional neural network similar to last part (down-sampling) of the U-net architecture is used in this project. The details of the architecture is described below:

- number of sequential layers: 3
- batch size: 64
- number of epochs: 100
- kernel size: 5x5
- padding: 0
- stride: 2
- Batch Normalization after Conv2D
- LeakyReLU after Batch Normalization

3.3. Loss Function

Let, G denotes the generator, D denotes the discriminator, x denotes the input sketches, y denotes the real images, z denotes random noise.

Then, the objective function of the original GAN algorithm can be expressed as

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_z[\log(1 - D(G(z)))] \quad (1)$$

As the image generation is conditioned on input sketches, the loss function of the cGAN also reflects that. Let, sketch inputs are denoted by x. Then the objective function of the cGAN wil be:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \quad (2)$$

As mentioned in the original cGAN study [7], a traditional loss is mixed with the GAN objective function. To train the Generator U-net, L1 loss (mean absolute error) is used in the loss function. L1 loss is calculated pixel-wise. The model optimizes using the loss function to generate images as close to the original images. The following equation denotes the L1 loss of the Generator:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \quad (3)$$

The overall objective function is

$$\mathcal{L}_{Total}(G, D) = \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (4)$$

4. Experimental Evaluation

This section describes the dataset, data-preprocessing, and result analysis.

4.1. Dataset

According to the project aim of generating colored images from sketches/edges, a relevant large-scale dataset is selected. That is Danbooru Sketch Pair [10]: a large collection of 128x128 anime pictures and sketch pair dataset converted from Danbooru2017. First, the image-sketch pairs are extracted and then processed. Images are converted into RGB and normalized to 0 means. Training, validation, and test sets are created from the dataset. Due to resource limitation, a portion of the dataset is used in this project. That protion contains 7600 sketch-image pairs.

4.2. Result Analysis

First, the U-net-based generator is evaluated without using any discriminator. Then, the full cGAN architecture is evaluated where both the generator and discriminator are used. Finally, a qualitative comparison between these two is shown.

4.2.1 Using only Generator component (U-net)

A U-net generator is trained on input sketches using pixel-wise L1 loss function (Equation 3). The model is trained using the loss function to generate images as close to the original images. The learning curve of the training is depicted in Fig. 5. Both the training an validation loss do not minimized notably after 30 epochs.

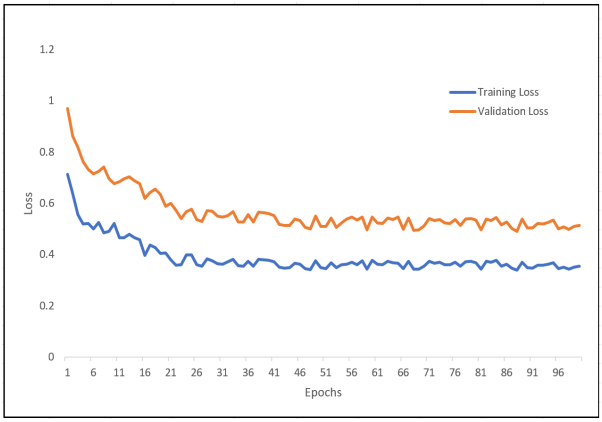
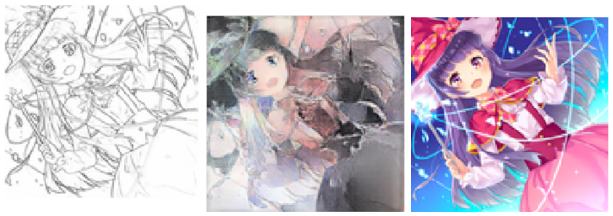
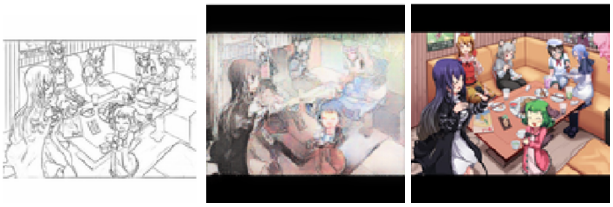


Figure 5. Learning curve of the U-net Generator

Sample generated images using U-net Generator are shown in Fig. 6



(a) Sketch (b) Generated (c) Ground Truth



(d) Sketch (e) Generated (f) Ground Truth

Figure 6. Sketch to Image using U-net-based Generator

4.2.2 Using the full cGAN architecture

The full cGAN architecture is trained on input sketches using the main objective function (Equation 4). The model is trained using the loss function to generate images as close to the original images.

The learning curve of the generator is depicted in Fig. 7. The generator loss is decreased through epochs. It does not diverges much and quickly getting converged as sketches are used as conditional input rather using only random noise as input.

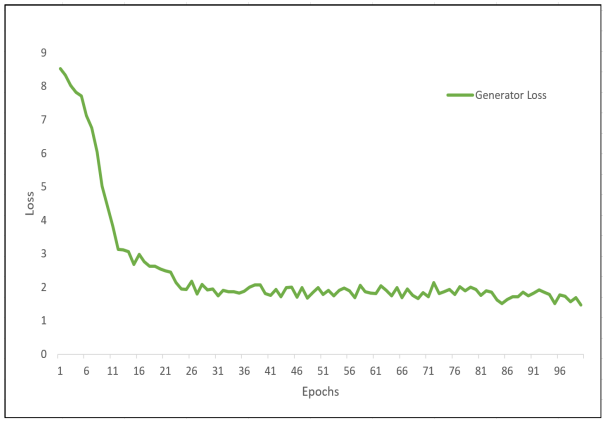


Figure 7. Generator Loss

The learning curve of the discriminator is depicted in Fig. 8. Initially, the discriminator quickly learns to distinguish between real and fake images. But, eventually, the discriminator loss increases as the generator is gradually learning to fool the discriminator.

Sample generated images using U-net Generator are shown in Fig. 9

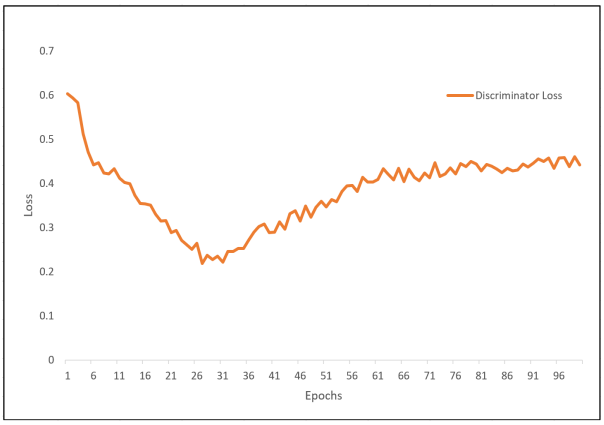
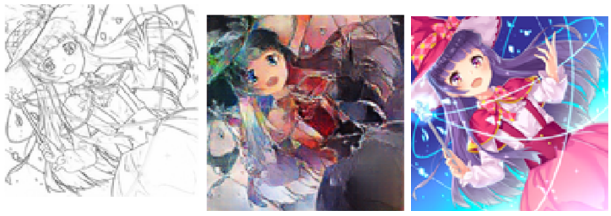
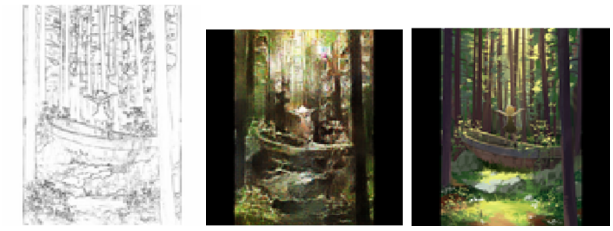


Figure 8. Discriminator Loss

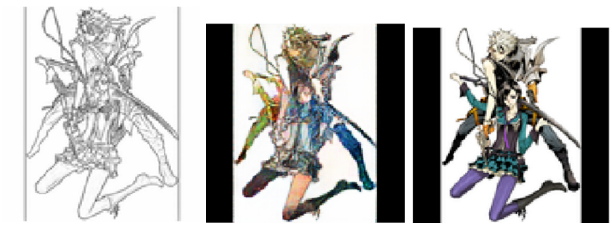


(a) Sketch (b) Generated (c) Ground Truth



(d) Sketch (e) Generated (f) Ground Truth

(g) Sketch (h) Generated (i) Ground Truth

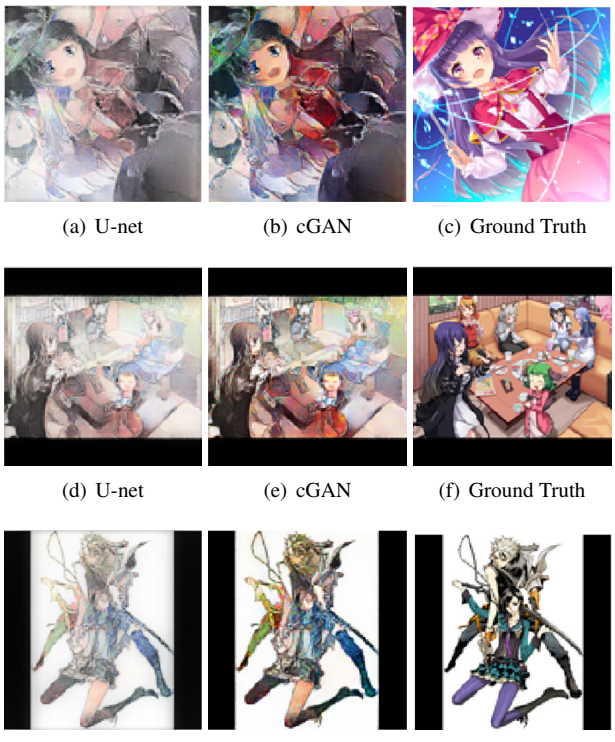


(j) Sketch (k) Generated (l) Ground Truth

Figure 9. Sketch to Image using cGAN architecture

4.2.3 Comparison between U-net Generator (without discriminator) and cGAN architecture (with discriminator)

Finally a qualitative comparison between the U-net Generator (without discriminator) and the cGAN architecture (with discriminator) is depicted in Fig. 10



(a) U-net (b) cGAN (c) Ground Truth

(d) U-net (e) cGAN (f) Ground Truth

(g) U-net (h) cGAN (i) Ground Truth

Figure 10. Comparison between U-net Generator (without discriminator) and cGAN architecture (with discriminator)

5. Challenges

Training the cGAN seems quite difficult due to instability. It is not always the case when model generates reasonable output. It might occur due the mode collapse issue, complexity, and other factors. Also, Also it was difficult to manage the large dataset (10.8 GB) and train it without an easily accessible GPU. So, the experiments are carried out by using a small fraction of the full dataset and a conclusive hyper-parameter tuning is not carried out.

6. Conclusion

In this project, a image-to-image translation algorithm - cGAN is implemented and evaluated. Experimental results show that it can produce reasonable colored images from sketches which are quite close to the original image. But, there are a lot of scopes to improve with: using more data samples, comprehensive hyper-parameter tuning, etc. In future, those things will be considered and re-evaluated. Also, running this on various image-to-image translation applications would be interesting, i.e., translation of images from day to night, translation of images from season to different seasons, translation of edge to image, translation of semantic segmentation to image, translation of satellite photographs to Google Maps, translation of black and white photographs to color, etc.

References

[1] Doron Adler. Sketch2pokemon, Oct 2019.

[2] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[4] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE transactions on image processing*, 28(11):5464–5478, 2019.

[5] 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.

[6] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[7] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[8] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen. Image-to-image translation: Methods and applications. *arXiv preprint arXiv:2101.08629*, 2021.

[9] 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.

[10] Wuhecong. Danbooru sketch pair 128x, Nov 2019.

[11] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.