CS230

GANime: Generating Anime and Manga Character Drawings from Sketches with Deep Learning

Tai Vu
Department of Computer Science
Stanford University
taivu@stanford.edu

Robert Yang
Department of Computer Science
Stanford University
bobyang9@cs.stanford.edu

Abstract

The process of generating fully colorized drawings from sketches is a large, costly bottleneck in the manga and anime industry. In this study, we examine multiple models for image-to-image translation between anime characters and their sketches, including Neural Style Transfer, C-GAN, and CycleGAN. By assessing them qualitatively and quantitatively, we find that C-GAN is most effective in producing high-quality and high-resolution images close to those created by humans.

1 Introduction

In this project, we propose a computer vision model for Image-to-Image translation between anime characters and their sketches. Specifically, the outcome will be a neural network that takes as input sketch drawings of anime characters and produce high-resolution color images of these characters.

Colorization is an especially prominent bottleneck in manga, where many artists opt for black and white images rather than high-quality colorized images because the lack of personnel to colorize, the lower cost involved, and the faster production rate. With our program, artists can automate their colorization process and speed up production.

To attain this outcome, we develop three computer vision models: Neural Style Transfer [3], Conditional Generative Adversarial Networks (C-GAN) [5] [6], and Cycle Generative Adversarial Networks (CycleGAN) [8]. These algorithms have demonstrated great success on a number of image translation tasks, so they are appropriate for our project.

2 Related Work

Image-to-image translation is an active area of research in computer vision. One popular algorithm is Neural Style Transfer [3], which blends a content image and a style image. It aims to generate an image that is similar to the content image but follows the style of the style image.

Another common solution is Conditional GAN (C-GAN) [5], a variation of Generative Adversarial Networks that generates new images based on inputs to the network. In particular, Pix2Pix [6] is a C-GAN that provides a general-purpose solution to image-to-image translation problems, where we condition on an input image and generate a corresponding output image. These models not only learn the mapping from inputs to outputs, but also learn a loss function to train this mapping.

Additionally, CycleGAN [8] is a state-of-the-art model that learns to translate an image from a source domain to a target domain in the absence of aligned image pairs. It captures special characteristics of the source image collection and translates these characteristics into the other image collection.

There have been some attempts to apply GANs in image colorization problems [6] [14] [15]. However, these papers focused on producing real-life color images from their gray-scale versions. There is little work on handling sketch drawings and fictional cartoon characters (such as [16]). This is an interesting challenge because sketches contain less rich information than gray-scale inputs. In this paper, we examine the effectiveness of different generative models on line art colorization.

3 Dataset and Input Pipeline

We used the Anime Sketch Colorization Pair dataset [2] from Kaggle. This dataset contains 17769 pairs of sketch-color anime character images, which are separated into 14224 examples for training and 3545 instances for testing. Each of these images is an RGB image of size 512 x 1024. In the dataset, the training example (a sketch drawing) and the ground truth (a colorized painting) are part of the same image, so we ran a script to separate the data into inputs X and outputs Y. We rescaled the image to 256 x 256 resolution, which is a high-enough resolution to not obscure detail, but a low-enough resolution to save memory and expedite the running of the algorithm. In addition, we normalized the training examples to the range $[-1, 1]$ to speed up learning. Additionally, we utilized a batch size of 32 and shuffled the training instances in every epoch. We also implemented some data augmentation techniques, including random cropping and random mirroring.

4 Models

4.1 Baseline Model: Neural Style Transfer

We utilized the Neural Style Transfer [3] algorithm as a baseline model. The training example is inputted as the "content" image, and the ground truth is inputted as the "style" image. We observe if the style (colors) from the ground truth can be transferred onto the generated image. We designed two different implementations for this task. The first one is Fast Style Transfer [12], provided by Arbitrary Image Stylization on Tensorflow Hub [11]. This practice allows for fast real-time stylization with any content-style image pair. The second model is the original style-transfer algorithm with a pretrained VGG19 network. We leveraged its intermediate layers to get the content and style representations and matched our output image to these representations. We trained the model for 1000 epochs.

4.2 C-GAN

Next, we built the Pix2Pix model [6]. A generator G generates a colorized image from an input sketch image, while a discriminator D takes as inputs a sketch image and a color image and determines whether the color image is real or fake. The generator G is a U-Net architecture, which is an encoder-decoder with skip connections from encoder layers to decoder layers. The discriminator D is a PatchGAN architecture, which classifies $N \times N$ patches of an image as real or fake.

The loss for Pix2Pix is $\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))]$, where z is a random noise. G tries to minimize this objective against D that tries to maximize it. In addition, we add a L_1 reconstruction loss $\mathcal{L}_{L_1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$. The total loss is $\mathcal{L}(G, D) = \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{L_1}(G)$, and we seek $G^* = \arg\min_G \max_D \mathcal{L}(G, D)$.

We chose $\lambda = 100$ and patch size $N = 70$. We trained the model for 150 epochs using the Adam optimization algorithm with learning rate $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$ and $\epsilon = 10^{-7}$.

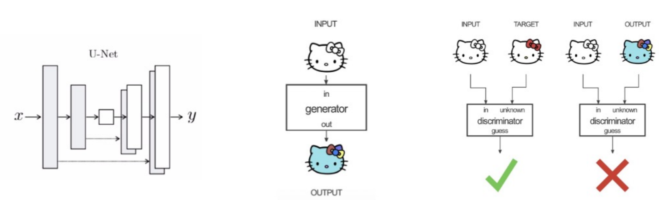

Figure 1: U-Net architecture (left), Generator (middle), Discriminator (right) [6], [7]

4.3 CycleGAN

Finally, we built a CycleGAN model based on Pix2Pix architecture design. A generator G generates a colorized image from a sketch image, while a generator F generates a sketch image from a colorized image. Two discriminators D_X and D_Y distinguish between real and fake images. The generators G and F are U-Nets, while the discriminators D_X and D_Y are PatchGANs.

The loss for CycleGAN is $\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_y[\log D_Y(y)] + \mathbb{E}_x[\log(1 - D_Y(G(x)))]$, where G aims to minimize this objective against D_Y that tries to maximize it. Similarly, we have $\mathcal{L}_{GAN}(F, D_X, Y, X)$. Additionally, we add a cycle consistency loss $\mathcal{L}_{cyc}(G, F) = \mathbb{E}_x[\|F(G(x)) - x\|]_1 + \mathbb{E}_y[\|G(F(x)) - y\|]_1$. The total loss is $\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda\mathcal{L}_{cyc}(G, F)$, and we seek $G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y)$.

We chose $\lambda = 10$ and patch size $N = 70$. We trained the model for 150 epochs using the Adam optimization algorithm with learning rate $\alpha = 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$.

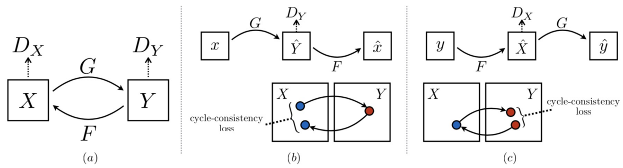

Figure 2: CycleGAN generators and discriminators [8]

5 Evaluation

We evaluate performance qualitatively through direct observation and quantitatively using the Structural Similarity (SSIM) Index and the Frechet Inception Distance (FID). The SSIM quantitatively determines the quality of generated outputs by evaluating the difference between the generated image and the ground truth image. Meanwhile, the FID [9] computes the covariance between real and fake distributions using a pre-trained Inception v3 network.

We calculated the average SSIM and the average FID for Neural Style Transfer, C-GAN, and CycleGAN, by generating a sample of 100 images from each model and finding the mean and standard deviation of each sample. This informed us on not only the overall quality of the images but also the consistency of the model. We also computed the SSIM and the FID for the C-GAN across various epochs to examine the progress of the C-GAN.

6 Experiments and Discussion

6.1 Qualitative Results

6.1.1 Visual Results of Neural Style Transfer

Figure 3: Results of Neural Style Transfer. Training example (first), ground truth (second), generated image - 1st model (third), generated image - 2nd model (fourth).

We can see that overall, the same colors from the style image have been transferred onto the generated image, but the location of those colors are different than expected. Also, the first implementation produced an output image with many distinct colors, while the second one focused on transferring the main color of the style image to the generated outcome (as observed in Figure 3). We believe that Neural Style Transfer is a good baseline model that our final algorithm should outperform.

6.1.2 Visual Results of CycleGAN

Overall, we see that the CycleGAN performs slightly better than baseline from a visual standpoint, but it is far from perfect. It focuses on one or two colors (black and brown) instead of learning to

Figure 4: Results of CycleGAN. Ground truth (left), generated image (right).

produce different colors, as shown in Figure 4. In addition, the colorization seems a bit inconsistent and unsmooth, with some portions colorized while others left with no colors.

6.1.3 Visual Results of C-GAN

In the first few epochs, the generated images were often low-quality (see Figure 5). Specifically, there was a grainy texture that made the image look artificial, showed in the close-up; furthermore, there can be two colors that are used to color one area which is supposed to be colored by one single color.

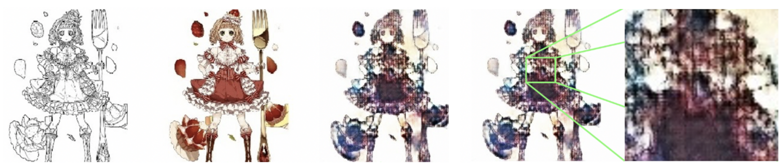

Figure 5: C-GAN results, epoch 2. Training example (first), ground truth (second), generated image (third & fourth) and close-up of generated image (fifth).

By epoch 18, the GAN had improved significantly, getting rid of the grainy texture. However, the color problem is still there (as shown in Figure 6), where green and purple compete for the same area. By epoch 146, the generated images were approaching the quality of the ground truth image (as observed in Figure 6), with the texture problem gone entirely and the coloring problem mostly gone.

Figure 6: C-GAN results, epoch 18 (left) & epoch 146 (right). Training example (first), ground truth (second), generated image (third).

Qualitatively, the C-GAN excels at coloring the hair of anime characters, often being quite detailed. Furthermore, C-GANs trained on more epochs grow increasingly good at coloring skin and eyes the right color (a rather difficult task for the computer). Lastly, the C-GAN can usually capture the nuances in the character's clothes and accessories by coloring them correctly. However, the C-GAN is still imperfect with sophisticated images with many details, and sometimes still smudges dark colors. However, it has very high performance overall.

6.2 Quantitative Results

We first compare the performance of Neural Style Transfer, C-GAN, and CycleGAN. In Table 1, we see that C-GAN has the best performance according to both the FID and SSIM metrics. Namely, C-GAN has the lowest FID, meaning that the distribution of C-GAN-generated images is closest to the ground truth distribution. It also has the highest average SSIM, meaning that it has the highest overall image quality compared to CycleGAN and Neural Style Transfer. Furthermore, it has the lowest SSIM standard deviation, which signifies superior robustness to input image complexity.

We then examine the learning process of the C-GAN model to analyze why C-GAN performed best out of the three models. We find that C-GAN is able to quickly overcome texture and color

Model	FID	SSIM (mean)	SSIM (standard deviation)
Neural Style Transfer	345.506	0.6547214	0.09885219
C-GAN (Pix2Pix)	**227.948**	**0.7468922**	**0.07413621**
CycleGAN	272.619	0.7238495	0.08240251

Table 1: Performance comparison between three models

problems to focus on the harder part of learning detail. In Figure 7 (middle), we see that the SSIM index improves quickly until around epoch 10 and then gradually levels out. The SSIM, which mainly focuses on the texture of the image, is only good at distinguishing bad texture (i.e. grainy or grid-shaped coloring). Thus, C-GAN focuses on learning to output the right texture from the start to epoch 10. Then, in Figure 7 (left), the FID improves quickly until approximately epoch 35 and then gradually levels out. The FID is able to detect both "texture" and "color" problems, so from epoch 10 to epoch 35, the C-GAN mainly focuses on improving its skills at coloring one area with one single color (as opposed to two colors smudged together). Lastly, the FID and SSIM scores improve slightly from epoch 35 onwards, even though qualitatively, the images show improvement until approximately epoch 100. This is because the GAN focuses on learning small details after epoch 35 that the two metrics do not weight heavily. As mistakes in the small details are not as blatant as large mistakes such the as "texture" and "color" problems, improving in those areas might seem substantial qualitatively but may be obscured by noise in the plot of the metrics.

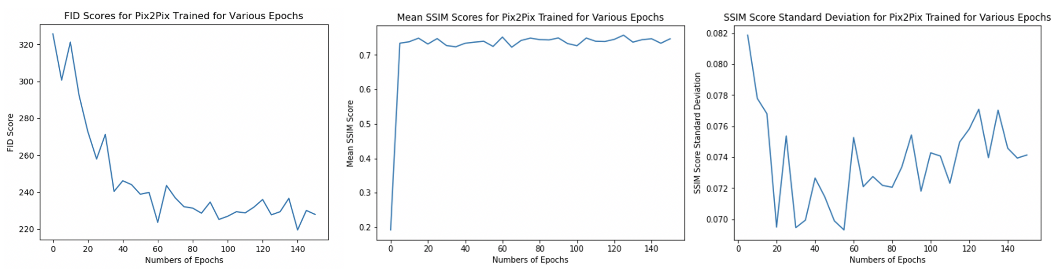

Figure 7: FID scores (left), Mean SSIM Index (middle), and SSIM Index Standard Deviation (right) of a sample size of 100 images over C-GAN models trained to epochs ranging from 0 to 150

7 Conclusion and Future Work

In summary, we implemented the baseline algorithm of Neural Style Transfer and the models of CycleGAN and C-GAN. Our models produced decent outcomes on this anime colorization task, with C-GAN yielding the best performance as it improved past its texture and color problems.

The next step would be to fine-tune hyperparameters to improve our current models and design a real-fake experiment to test their performance based on human perception. Additionally, we want to utilize the higher resolution of 512 x 512 to generate high-quality outputs that can be deployed in the real industry. We would also experiment with various generative models such as other GANs and conditional variational autoencoders as well as modify the network architectures, such as employing ResNet, ImageGAN and so on. Other options include doing transfer learning with pre-trained weights for Pix2Pix, utilizing different color spaces and adding total variation loss to remove high frequency artifacts. Finally, one feature we are excited to implement is to condition on certain colors for the image in order to give the user more control.

8 Contributions

Tai Vu and Robert Yang brainstormed together and came up with the project idea together. Tai Vu was responsible for researching Neural Style Transfer, C-GAN and CycleGAN, while Robert Yang was responsible for researching SSIM and FID metrics. Afterwards, both Tai and Robert contributed to building the data pipeline, implementing the program, training the models and evaluating the model performance.

References

[1] Goodfellow, Ian J., et al. “Generative Adversarial Networks.” ArXiv:1406.2661 [Cs, Stat], June 2014. arXiv.org, http://arxiv.org/abs/1406.2661.

[2] Anime Sketch Colorization Pair. https://kaggle.com/ktaebum/anime-sketch-colorization-pair. Accessed 22 Jan. 2020.

[3] Gatys, Leon A., et al. “A Neural Algorithm of Artistic Style.” ArXiv:1508.06576 [Cs, q-Bio], Sept. 2015. arXiv.org, http://arxiv.org/abs/1508.06576.

[4] Athalye, Anish. Anishathalye/Neural-Style. 2015. 2020. GitHub, https://github.com/anishathalye/neural-style.

[5] Mirza, Mehdi, and Simon Osindero. “Conditional Generative Adversarial Nets.” ArXiv:1411.1784 [Cs, Stat], Nov. 2014. arXiv.org, http://arxiv.org/abs/1411.1784.

[6] Isola, Phillip, et al. “Image-to-Image Translation with Conditional Adversarial Networks.” ArXiv:1611.07004 [Cs], Nov. 2018. arXiv.org, http://arxiv.org/abs/1611.07004.

[7] Image-to-Image Translation in Tensorflow - Affine Layer. https://affinelayer.com/pix2pix/. Accessed 14 Mar. 2020.

[8] Zhu, Jun-Yan, et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks.” ArXiv:1703.10593 [Cs], Nov. 2018. arXiv.org, http://arxiv.org/abs/1703.10593.

[9] Heusel, Martin, et al. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium.” ArXiv:1706.08500 [Cs, Stat], Jan. 2018. arXiv.org, http://arxiv.org/abs/1706.08500.

[10] Szegedy, Christian, et al. “Rethinking the Inception Architecture for Computer Vision.” ArXiv:1512.00567 [Cs], Dec. 2015. arXiv.org, http://arxiv.org/abs/1512.00567.

[11] TensorFlow Hub. https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/2. Accessed 21 Feb. 2020.

[12] Ghiasi, Golnaz, et al. “Exploring the Structure of a Real-Time, Arbitrary Neural Artistic Stylization Network.” ArXiv:1705.06830 [Cs], Aug. 2017. arXiv.org, http://arxiv.org/abs/1705.06830.

[13] Isola, Phillip. Phillipi/Pix2pix. 2016. 2020. GitHub, https://github.com/phillipi/pix2pix.

[14] Zhang, Richard, et al. “Colorful Image Colorization.” ArXiv:1603.08511 [Cs], Oct. 2016. arXiv.org, http://arxiv.org/abs/1603.08511.

[15] Nazeri, Kamyar, et al. “Image Colorization with Generative Adversarial Networks.” ArXiv:1803.05400 [Cs], vol. 10945, 2018, pp. 85–94. arXiv.org, doi:10.1007/978-3-319-94544-6_9.

[16] Frans, Kevin. “Outline Colorization through Tandem Adversarial Networks.” ArXiv:1704.08834 [Cs], Apr. 2017. arXiv.org, http://arxiv.org/abs/1704.08834.