
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, usually 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 the most effective model that is able to produce high-quality and high-resolution images close to those created by humans.

1 Introduction

Colorization is an especially prominent bottleneck in manga and anime production. Given busy schedules of monthly releases, many artists opt for black and white images rather than high-quality colorized images due to the lack of personnel to colorize, the lower cost involved, and the faster production rate. Therefore, we seek to design a system that can enable artists to automate their colorization process and speed up production.

In this project, we propose a computer vision program for image-to-image translation between anime characters and their sketches. Specifically, the outcome is a neural network that takes as inputs sketch drawings of anime characters and produces high-resolution color images of these characters.

To attain this outcome, we develop three computer vision models, including Neural Style Transfer [1], Conditional Generative Adversarial Networks (C-GAN) [2] [3], and Cycle Generative Adversarial Networks (CycleGAN) [4]. These algorithms have demonstrated great success on a number of image translation tasks, so they are appropriate for our project. By employing several qualitative and quantitative evaluation metrics, we find that both the GAN models outperform the baseline. In particular, C-GAN with an added total variation loss performs the best with an average FID of 220.499 and an average SSIM of 0.7559.

2 Related Work

Image-to-image translation is an active area of research in computer vision. One popular algorithm is Neural Style Transfer [1], 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) [2], a variation of Generative Adversarial Networks [5] that generates new images based on inputs to the network. In particular, Pix2Pix [3] 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 [4] 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 [3] [6] [7]. 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 [8]). This is an interesting challenge because sketches contain less rich information than gray-scale inputs. In this paper, we scrutinize the effectiveness of different generative models on line art colorization.

3 Dataset and Input Pipeline

We used the Anime Sketch Colorization Pair dataset [9] 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 512×1024 .

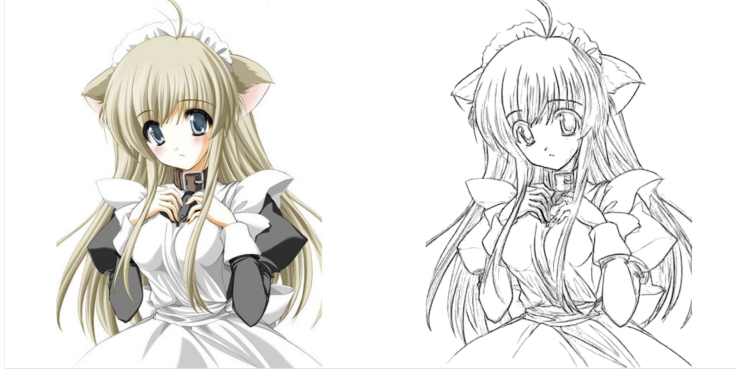


Figure 1: An image in the dataset.
Training example (right) and ground truth (left) in the same image.

In the dataset, the training example (a sketch drawing) and the ground truth (a colorized painting) are part of the same image (see Figure 1), so we ran a script to separate the data into inputs X and outputs Y . We then rescaled the image to 256×256 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 $[-1, 1]$, because we believe that normalization can help speed up learning.

For training GANs, 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 Implementation

We leveraged TensorFlow 2.1.0 [10] for implementing our program. In addition, we harnessed Amazon Web Services (AWS) [11] and Google Colab with GPU [12] for training the models.

Link to our project: <https://github.com/taivu1998/GANime>

5 Models

5.1 Baseline Model: Neural Style Transfer

We utilized the Neural Style Transfer [1] 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 [13], provided by Arbitrary Image Stylization on Tensorflow Hub [14]. 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.

5.2 C-GAN

Next, we built the Pix2Pix model [3]. 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 function 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 x is an input, y is the corresponding target, and z is a random noise. G tries to minimize this objective against D that tries to maximize it.

In addition, we added a L_1 reconstruction loss

$$\mathcal{L}_{L_1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

Hence, the total loss for Pix2Pix is

$$\mathcal{L}(G, D) = \mathcal{L}_{cGAN}(G, D) + \lambda_{L_1} \mathcal{L}_{L_1}(G)$$

Our objective is to seek $G^* = \arg \min_G \max_D \mathcal{L}(G, D)$.

Furthermore, we invented a new variant of Pix2Pix to enhance its performance. Particularly, we added a total variation loss

$$\mathcal{L}_{tv}(G) = \sum_{i,j} (|\hat{y}_{i+1,j} - \hat{y}_{i,j}| + |\hat{y}_{i,j+1} - \hat{y}_{i,j}|)$$

where $\hat{y} = G(x, z)$. This total variation loss acts as a regularization term that encourages adjacent pixels to have similar values and reduces high frequency artifacts of the generated image.

The modified loss function is

$$\mathcal{L}(G, D) = \mathcal{L}_{cGAN}(G, D) + \lambda_{L_1} \mathcal{L}_{L_1}(G) + \lambda_{tv} \mathcal{L}_{tv}(G)$$

We chose $\lambda_{L_1} = 100$, $\lambda_{tv} = 0.0001$ 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 $\epsilon = 10^{-7}$.

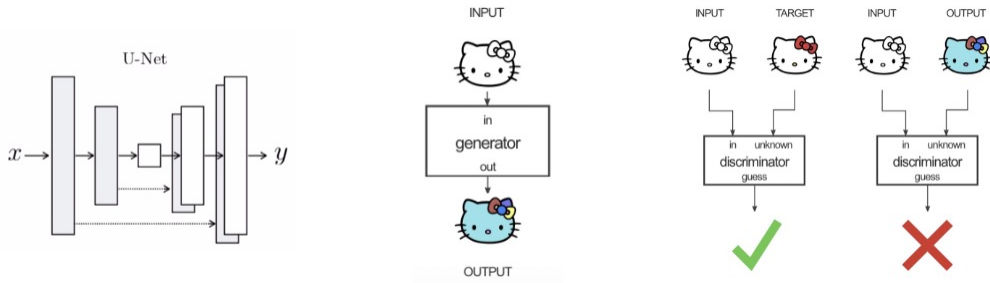


Figure 2: U-Net architecture (left), C-GAN generator (middle), C-GAN discriminator (right) [3], [15]

5.3 CycleGAN

Finally, we built a CycleGAN model based on the 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 function 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 x is an input and y is the corresponding target. G aims to minimize this objective against D_Y that tries to maximize it. Similarly, we have another loss $\mathcal{L}_{GAN}(F, D_X, Y, X)$.

Additionally, we added a cycle consistency loss

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_x[\|F(G(x)) - x\|_1] + \mathbb{E}_y[\|G(F(y)) - y\|_1]$$

Hence, the total loss for CycleGAN is

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda_{cyc}\mathcal{L}_{cyc}(G, F)$$

Our objective is to seek $G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y)$.

We chose $\lambda_{cyc} = 10$ 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 $\epsilon = 10^{-7}$.

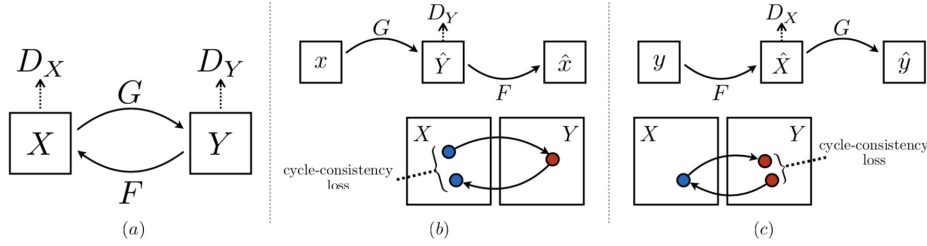


Figure 3: CycleGAN generators and discriminators [4]

6 Evaluation

We evaluated the performance of the three models 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 image we want to measure (in this case the generated image) and a good quality image (in this case the ground truth). Meanwhile, the FID [16] computes the covariance between real and fake distributions using a pre-trained Inception v3 network.

We calculated the average SSIM and the average FID scores 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 models. We also computed the SSIM and the FID for the two C-GAN versions across various epochs to examine the progress of the C-GAN.

7 Experiments and Discussion

7.1 Qualitative Results

7.1.1 Visual Results of Neural Style Transfer



Figure 4: Neural Style Transfer results. 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. More interestingly, the first implementation produces an output image with many distinct colors, while the second one focuses on transferring the main color of the style image to the generated outcome (as observed in Figure 4). We believe that Neural Style Transfer is a good baseline model that our final algorithm should outperform.

7.1.2 Visual Results of CycleGAN

Overall, we see that the CycleGAN performs slightly better than the baseline from a visual standpoint, but it is far from perfect. It focuses on one or two colors instead of learning to produce different colors, as shown in Figure 5. In addition, the colorization seems a bit inconsistent and unsmooth, with some portions colorized while others left with no colors.



Figure 5: CycleGAN results, epoch 150
Training example (first), ground truth (second), generated image (third).

7.1.3 Visual Results of C-GAN

In the first few epochs, the generated images are often low-quality (see Figure 6). Specifically, there is a grainy texture that makes the image look artificial, showed in the close-up. What is more, there can be two colors that are used to color one area which is supposed to be colored by one single color.



Figure 6: 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 has improved significantly, getting rid of the grainy texture. However, the color problem is still there (as shown in Figure 7), where green and purple compete for the same area. Afterwards, by epoch 150, the generated images are approaching the qualities of the ground truth images (as observed in Figure 8), with the texture problem gone entirely and the coloring problem mostly gone.



Figure 7: C-GAN results, epoch 18
Training example (first), ground truth (second), generated image (third).



Figure 8: C-GAN results, epoch 150
Training example (first), ground truth (second), generated image (third).

Qualitatively, the C-GAN (Pix2Pix) 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 and beautifully. However, Pix2Pix is still imperfect with sophisticated images with many details, and sometimes still smudges dark colors. Despite that, the C-GAN has very high performance overall.

Meanwhile, the addition of the total variation loss tends to improve the performance of Pix2Pix. By comparing Figure 8 and Figure 9, we can see that the outputs in the latter one are slightly better



Figure 9: Modified C-GAN results, epoch 150
Training example (first), ground truth (second), generated image (third).

looking. In fact, the total variation loss removes many high frequency components and encourages the network to color one region with one single color. Hence, the final outcomes become much smoother.

7.2 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.

In addition, the modified version of C-GAN has a slightly better performance than the original one. In particular, in comparison with the traditional Pix2Pix, the new one with a total variation loss has a lower FID score, a higher SSIM mean and a lower SSIM standard deviation. This is consistent with our above observation of visual results.

We then examine the learning process of the C-GAN model to analyze why C-GAN performs best out of the three models. We find that C-GAN is able to quickly overcome texture and color problems to focus on the harder part of learning detail. In Figure 11, 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 10, 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

Model	FID	SSIM (mean)	SSIM (standard deviation)
Neural Style Transfer	345.506	0.6547214	0.09885219
CycleGAN	272.619	0.7238495	0.08240251
C-GAN (Pix2Pix)	227.948	0.7468922	0.07413621
C-GAN (Pix2Pix modified)	220.499	0.75587333	0.07380538

Table 1: Performance comparison between all models.

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 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.

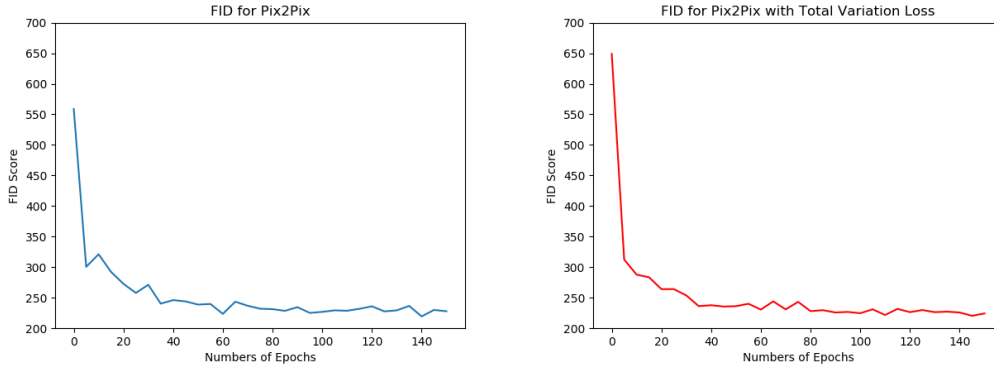


Figure 10: FID scores of C-GAN models trained to epochs 0-150 Pix2Pix (left), modified Pix2Pix (right).

8 Conclusion and Future Work

In summary, we implement the baseline algorithm of Neural Style Transfer and the generative models of CycleGAN and C-GAN. Our models produce decent outcomes on this anime colorization task, with C-GAN yielding the best performance as it improves past its texture and color problems. Additionally, the use of the total variation loss is of great help in removing noise and high frequency artifacts in the final outputs. Such a deep learning program has many practical applications, specifically in designing colorized images at an accelerated pace and bringing to life stunning manga and anime products.

The next step would be to fine-tune hyperparameters to improve our current models and to design a real-fake experiment (similar to the one in [3]) in order to test their performance based on human perception. Additionally, we want to utilize the higher resolution of 512×512 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 and harnessing different color spaces. 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.

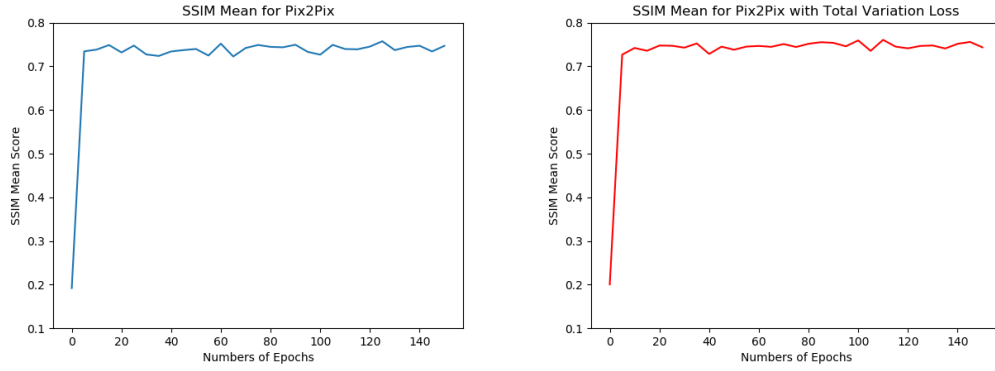


Figure 11: SSIM average scores of C-GAN models trained to epochs 0-150 Pix2Pix (left), modified Pix2Pix (right).

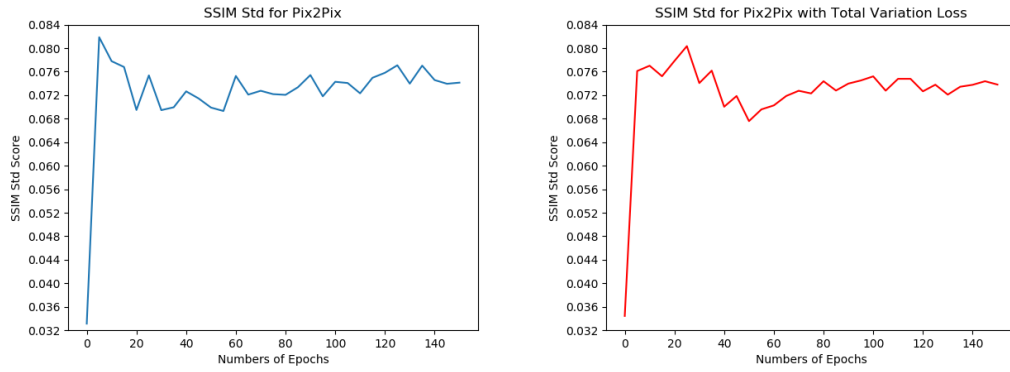


Figure 12: SSIM standard deviations of C-GAN models trained to epochs 0-150 Pix2Pix (left), modified Pix2Pix (right).

References

- [1] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” *arXiv preprint arXiv:1508.06576*, 2015.
- [2] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *arXiv preprint arXiv:1611.07004*, 2018.
- [4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *arXiv preprint arXiv:1703.10593*, 2018.
- [5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *arXiv preprint arXiv:1406.2661*, 2014.
- [6] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” *arXiv preprint arXiv:1603.08511*, 2016.
- [7] K. Nazeri, E. Ng, and M. Ebrahimi, “Image colorization with generative adversarial networks,” *arXiv preprint arXiv:1803.05400*, 2018.
- [8] K. Frans, “Outline colorization through tandem adversarial networks,” *arXiv preprint arXiv:1704.08834*, 2017.
- [9] T. Kim, “Anime sketch colorization pair,” 2019.
- [10] “Tensorflow,” 2019.
- [11] “Amazon web services,” 2006.
- [12] “Google colab,” 2018.
- [13] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, “Exploring the structure of a real-time, arbitrary neural artistic stylization network,” *arXiv preprint arXiv:1705.06830*, 2017.
- [14] “Tensorflow hub,” 2018.
- [15] A. Layer, “Image-to-image translation in tensorflow,” 2017.
- [16] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” *arXiv preprint arXiv:1706.08500*, 2018.
- [17] A. Athalye, “Neural style,” 2015.
- [18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *arXiv preprint arXiv:1512.00567*, 2015.
- [19] P. Isola, “pix2pix,” 2017.