Real World Scene To Anime Scene Translation

Abdul Jawad
University of California, Santa Cruz
abjawad@ucsc.edu

Hadiseh Gooran
University of California, Santa Cruz
hgoorano@ucsc.edu

Ishaan Paranjape
University of California, Santa Cruz
iparanja@ucsc.edu

Asiiah Song
University of California, Santa Cruz
julinas@ucsc.edu

Figure 1: The output of Chen et al.'s pretrained model of CartoonGAN.

ABSTRACT

We provide an end-to-end framework for translating a photograph into an anime-like image. Many of the current models do not translate images containing human faces. TwinGAN, which uses cycle-consistency loss and semantic loss, translates face images from the photo-realistic domain to the anime domain. In our pipeline, we connect TwinGAN with OpenCV scripts that do a face-swap for detected human faces in an input photograph. Next, a custom-trained CartoonGAN, which uses both a VGG semantic loss and an edge-promoting loss, is then run on the resulting image from the previous step to create the final output.

CCS CONCEPTS

• **Human-centered computing** → *Visualization systems and tools.*

KEYWORDS

anime style transfer, image processing, style transfer, PGGAN, neural networks, generative adversarial networks

ACM Reference Format:
Abdul Jawad, Hadiseh Gooran, Ishaan Paranjape, and Asiiah Song. 2019. Real World Scene To Anime Scene Translation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Since the original conception of generative adversarial networks (GANs)[9], they have yielded impressive results in many image generation tasks. Similarly, style transfer[8] using convolutional neural networks has captured the popular interest with its simple process and fascinating results. CycleGAN[27] further allowed image-to-image GANs to be trained without matching data sets. Even as such, neural style transfer is a method that applies well to texture-level transfers, but falters at higher level styles such as geometric structure or design.

In this project, we're interested in a sub-domain of image-to-image style transfer, from photo-realistic images to anime-like images. Chen et al.'s CartoonGAN[4] introduced the edge-promoting adversarial loss, pointing out that a significant identifying trait of anime and other cartoon styles is the prominence of clear, solid outlines. Neural style transfer using CartoonGAN produces satisfying results, generating images with distinct outlines and flattened textures that resemble cartoon-like styles much more than previous works. However, a particular challenge of photo-to-anime image translation comes when the source images contain human faces. Because of the inability of neural style transfer to translate styles of geometric structure, and a prominent difference between human faces and anime faces is the difference is geometric ratio of facial features, it is undesirable to simply use a variation of neural style transfer for this task. To this end, we bring in TwinGAN[12] to the pipeline to process human faces into anime-like faces.

In Section 2, we review some related works. In Section 3 and 4, we describe our methodology in detail. In Section 5, we go over

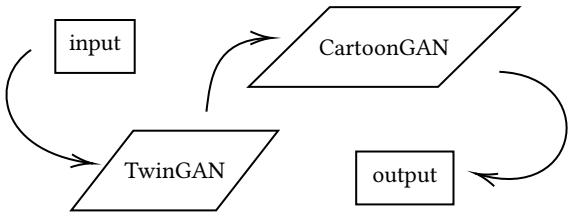


Figure 2: Our pipeline.

our results. In Section 6, we conclude and propose possible future work.

2 RELATED WORKS

2.1 Neural Style Transfer by Parts

Segmentation [26], feature separation [25] [14], and attentional networks [23] [16] have been used in style transfer beyond comic-specific applications. The additional granularity allows the neural style transfer to preserve geometric structure of the input, color the image by object rather than by texture, and do a true style "transfer" overwriting the style of the input image instead of blending with the style of the input image, just to name a few benefits.

2.2 Unpaired Cross-Domain Translation

CycleGAN[27] contributes an algorithm that allows for generation using unpaired, unlabeled data, greatly expanding the number of usable data sets. It has been applied in image synthesis [2] [5], pose detection[15], audio synthesis[10], question-answering image analysis[22], and many other projects.

2.3 Photo-to-Comic Translation

Recently there has been increased interest in using machine learning for style transfer between photos and comics. Chen, Lai, and Liu[3] use a convolutional neural network that, instead of calculating semantic loss using a standard image reconstruction model such as VGG19[21], uses an image reconstruction model trained on anime and comic content. Other approaches use segmentation or patch-wise style transfer to help maintain semantic structure while locally transferring to a comic-like style[26][20]. Overall, previous neural style transfer algorithms were non-comic-specific and did not work well[18]. CartoonGAN introduced the idea of an adversarial loss that promotes clear edges, enabling neural transferred style to look passable for a comic style. Extending the results of style transfer for comic style, Pkesko et al.'s work focused on extracting semantically salient and aesthetically pleasing frames from videos, then running comic stylizing on those frames to create output comic panels[17]. We base our pipeline (Figure 2) on CartoonGAN and TwinGAN, with more detail in the next section.

3 FACE TO ANIME CONVERSION

A big challenge in executing style transfer from the real world to anime domains is making the human beings, specifically the faces look like anime characters. Due to the challenges associated with this process alone, we chose to use a separate architecture for this purpose. The architecture we chose for this is TwinGAN. After getting the desired results, we swap the faces in the real world image using the OpenCV library and certain image processing techniques. After this step we proceed to using CartoonGAN for the remaining style transfer process as shown in figure 2. In this section, we describe each aforementioned step in detail.

Figure 3: Difference in features between human and anime faces.

3.1 TwinGAN

Currently there are two approaches for image style transfer, using neural style transfer[8] and using GANs[9]. In the neural style transfer approach, a trained object detection network like VGG19 is used for feature extraction. It gives unwanted results for objects that are not structurally similar to the data on which the network was trained. Neural style transfer also does not work well when changing styles involve changing the object's spatial proportions. This is especially required in the anime style transfer where the features in the target domain are quite different (large eyes, small faces, etc.) as shown in Figure 2.

Due to these reasons and promising results in the past, we use GANs for anime style transfer. A Generative Adversarial Network is a neural network which consists of a Generator(G) and a Discriminator(D). The Generator is responsible for synthesizing data points and the Discriminator tells the real and the synthesized points apart. A big reason why TwinGAN is used because it works with unpaired datasets. In other words, there is no labeling describing the common features.

3.1.1 Concept of latent embedding space. Many GANs assume that a latent embedding space exists. As explained in [12], given two related domains, S and T, for an item $a \in S$, there exists $b \in T$, which shares the same latent encoding. In other words, the high level features that are shared between two domains. An absence of this space can lead to a collapse of the model or unwanted results.

3.1.2 Structure. The TwinGAN architecture generates two functions which convert images from one domain to the next and vice versa. The architecture uses PGGAN [11]. In a progressive GAN, the G and the D stages progressively grow and the network alternates between growing and reinforcement stages. As shown in Figure 4, as the training stages progress, the resolution of the input data is progressively increased. The benefits of using this architecture is that the network gradually learns from the large

scale structures to the finer details of the images which improves stability tremendously. In addition to that the training process is 2-6 times faster. During reinforcement stages, unused stages from the lower resolution stages are discarded. The complete TwinGAN structure can be seen in Table 4.

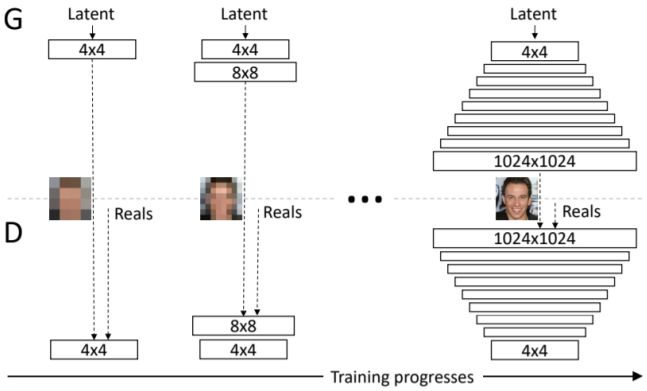


Figure 4: PGGAN training process.

Along with the generator, there also exists an encoder decoder structure. Since many important features of the images may be lost in the downsampling by the decoder. Due to this, we use skip connections as shown in the U-Net [19] architecture. Without skip connections, in [12], the network failed to find local semantic correspondences.

3.1.3 Batch Renormalization. Normalization is the process of converting the activations of neurons in one layer to values within a particular range as shown in Figure 5 [7]. The style difference in the two domains is captured using sets of batch renormalization parameters (γ, β) for each domain. The weights in all normalization layers is shared to imply that the same latent encoding space is being shared two different styles. To make the normalization consistent in the training and the inference stages, renormalization is performed. Batch renormalization allows us to use one network to perform multiple style transfers just by altering the sets of parameters as shown in Figure 6[7].

3.1.4 Losses. Various losses are used in TwinGAN. *Adversarial loss* compares the output and the input images. *Reconstruction loss* compares the synthesized input image with the original input image. *Cycle* and *Semantic* losses check if the output features are the same as the input in the same and the required style domain.

3.2 Precise cropping

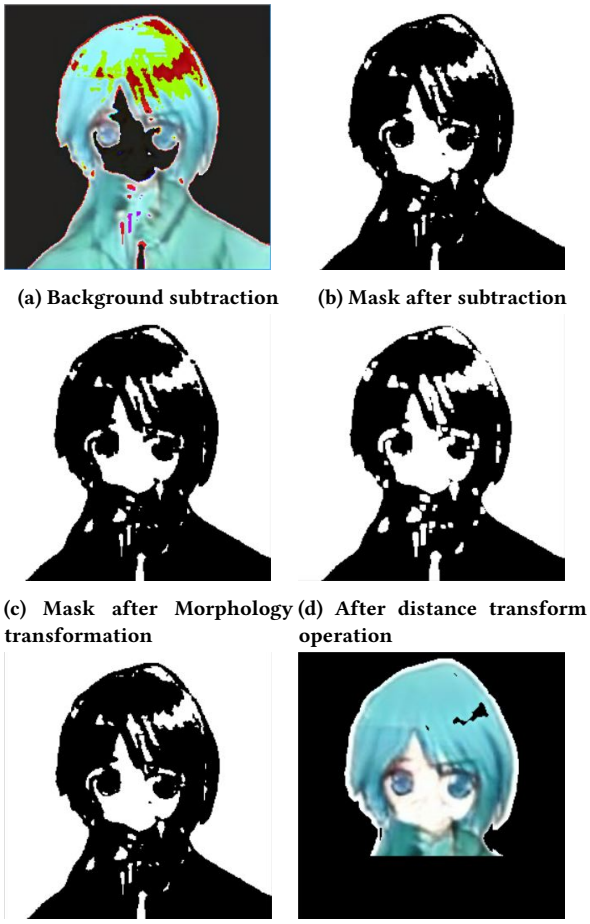
For translating real-world scene to the anime scene, the first thing to be done is to convert any faces present in the real world image. This conversion is because cartoon gan works fine in making the whole image cartoonish. As anime faces have different features like big eyes, pointy nose, spiked hair, cartoongan cannot make those changes in the real world images. Our main plan was to detect facial landmarks in both the real world and anime face and do one to one mapping. In theory, it worked fine. But the hurdles came into the scene when we discovered there was no existing facial landmarks detector for anime faces. We used dlib to detect facial landmarks in the real world faces. It gives 68 points in the human face, and with those points, we can identify eyes, eyebrows and jaw lines which is sufficient for delineating the exact area for the human face. For the anime face, there is an existing cascade classifier that gives the bounding box for anime face. But this bounding box is not enough for the one to one mapping we wanted. So we tried different image processing methods using OpenCV to get the exact crop from the anime images. While doing that we faced several other problems too. Here are all the issues we encountered during the exact face crop using OpenCV.

- From the twingan, we get output where the background is not entirely white or any other color.
- The head portion of the generated anime is not complete in the image.
- The face and neck portion are quite similar in some pictures.
- Twingan creates clothing in the output image.

We figured out the reasons for these problems. Apparently, in training, the images used for anime faces contains incomplete heads. That is the reason behind generating anime image with an unfinished head. Also when it creates a picture, it produces all the 256*256 pixels. Even though all the training images contains an entirely white background when it generates an image from the input face data it gives a white shade in the generated image. The similarity for neck and face comes because of the lower resolution of the generated images. The reason behind generating dresses is that all the anime training images contain some anime like dress in it. To overcome these, we used different features of the OpenCV. We elaborate those below:

3.2.1 Detecting human face: There are a lot of existing models for detecting a human face with all the facial features including landmarks. We used dlib - a python library for identifying a human face. For the facial landmarks, the library returns not only the bounding box but also 68 points for different facial landmarks. We extracted human faces from the real world images and used that as input for the twingan. While detecting the face, we also tracked the exact position of the face in the actual image. This positional coordinate is required to remake the main image after the face swap.

3.2.2 Detecting anime face: We are reading the converted anime images as a numpy array with three channels. The first thing we are doing is removing the white portion from the images. We are using an RGB color value of [200, 200, 200] as a threshold for the background. While removing the background, it sometimes removes some pixels from the anime face. So after removing the almost white background, we are pasting the bounding box again in that new image so that the face remain intake. This process gives us an image that contains only the colored part of the anime images. Due to the inability of detecting exact facial landmarks in the anime image, it becomes hard to distinguish between face, dress, hair. To overcome this, we used segmentation of images to get the hair part. And to make the face extraction easier, we removed the lower part of the face image after keeping some padding. This operation gives us the face and some parts of the hair with a lot of noise in it.



(a) Background subtraction (b) Mask after subtraction

(c) Mask after Morphology transformation (d) After distance transform operation

(e) After face reconstruction (f) Mask for composite image

Figure 5: intermediate results

For the segmentation, we used watershed segmentation algorithm and used the binary inverse threshold and OTSU threshold as they are providing good results. We also used distance transform to get the foreground colored images from the face image. After that, we created the forth channel and made all the parts except the colored parts transparent for making it ready for the replacement.

3.2.3 Image replace: After extracting anime face, we are making a composite image with the real world face image and extracted anime image. Making this composite image is comparatively an easy task. We just added pixels from the real-world image that are transparent in the anime image. This process gives us an image that contains background from the real image and foreground from the anime image. For smoothening the composite image, we used another anime like mask on it which removes the sharp lines from the converted image.

3.2.4 Remaking the real image. After we get the composite face image, we are copying that image in the main real world image. This image then will be used in the cartoon gan that will convert this image with a cartoon-like effect on it.

4 SCENE STYLE CONVERSION

4.1 Data

We created $\mathcal{P}hoto$, a random subset of the the Flickr30k data set[24] containing 6000 images. We created $\mathcal{A}nime$ using frames from a short clip of the "train scene" in *Spirited Away*[1], containing approximately 6000 images. We create $\mathcal{F}uzzy$ by processing $\mathcal{A}nime$, first dilating a mask of detected edges then using a Gaussian blur on the masked area.

We use $\mathcal{P}hoto$ to pretrain the generator of the scene-translating GAN as well as for the adversarial training. For the adversarial training, we use $\mathcal{A}nime$ as a positive data set and $\mathcal{P}hoto$ and $\mathcal{F}uzzy$ as a negative data set.

4.2 Training

We implement CartoonGAN in keras[6]. Following Chen et al., our generator G is a variational auto-encoder pre-trained with a semantic content loss using the VGG19[21] pre-trained model:

$$L_{content} = |VGG_l(G(\mathcal{P}hoto)) - VGG_l(\mathcal{P}hoto)|$$

where VGG_l is a higher level feature layer 'conv4_4' of VGG19. Although Chen et al. did not pre-train their discriminator, our discriminator D was pre-trained with $\mathcal{P}hoto$, $\mathcal{A}nime$, and $\mathcal{F}uzzy$ using standard binary cross entropy loss in order to help speed up convergence.

In the main training phase, the discriminator continues to run with binary cross entropy loss, using $\mathcal{A}nime$, $\mathcal{F}ake$, and $\mathcal{F}uzzy$ as input where $\mathcal{F}ake$ is the output of the generator each cycle. The generator runs using $\mathcal{P}hoto$ as input, with the following loss:

$$L_{generator} = L_{content} + L_{edge}$$

$$L_{edge} = log(D(G(\mathcal{P}hoto)))$$

Each pre-training phase is run for 5 epochs, while the main training phase is run for 200 epochs.

After many variations we managed to get a reasonable gradient. But after training for 200 epochs and more, the GAN still did not converge to a desirable result. We ended up using Chen et al.'s pretrained model[4][13] for our final pipeline.

5 RESULTS

We provided some of the results in Figure 6. Cartoon GAN does not work well in blending the face part with the other parts of the images. For getting a full head from the TwinGAN we added extra padding over the head of the cropped human images. We used pretrained cartoon GAN for the provided images. It retains the face orientation quite well. Twin GAN works well if the background is a light color in the real world images because if there are some structure or elements as the background in the real world images, those pixels adds lot more noise in the generated images which causes problems in the face-cropping part of the pipeline.

6 CONCLUSIONS

We encountered many problems while building TwinGAN, and ultimately was not able to train our versions of either TwinGAN or CartoonGAN to convergence, opting instead to use the pretrained

(a) A. before (b) A. after

(c) B. before (d) B. after

(e) C. before (f) C. after

Figure 6: Results

models opensourced by the paper authors. For CartoonGAN, because only the model weights were released, we used a reimplementation on github (using the original model weights).

It is unsatisfying that we were not able to improve the outputs of TwinGAN and CartoonGAN individually, and there are some visible flaws to the final output. However, some of our original goals were accomplished, namely the creation of an end-to-end pipeline that puts all the pieces together, including the TwinGAN and CartoonGAN architectures, and enables the conversion of a realistic photo to an anime-like photo by running a single script.

REFERENCES

[1] 2002. Spirited Away.

[2] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. 2018. Augmented cyclegan: Learning many-to-many mappings from unpaired data. *arXiv preprint arXiv:1802.10151* (2018).

[3] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. 2017. Transforming photos to comics using convolutional neural networks. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2010–2014.

[4] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. 2018. Cartoongan: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9465–9474.

[5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8789–8797.

[6] François Chollet et al. 2015. Keras. https://keras.io.

[7] Shlens Dumoulin and Kudlur. 2017. A learned representation for artistic style. *Proc. of ICLR* (2017).

[8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015).

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[10] Takuhiro Kaneko and Hirokazu Kameoka. 2017. Parallel-data-free voice conversion using cycle-consistent adversarial networks. *arXiv preprint arXiv:1711.11293* (2017).

[11] Laine Karras, Aila and Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

[12] Jerry Li. 2018. Twin-GAN–Unpaired Cross-Domain Image Translation with Weight-Sharing GANs. *arXiv preprint arXiv:1809.00946* (2018).

[13] Y. Li. 2018. CartoonGAN-Test-Pytorch-Torch. https://github.com/Yijunmaverick/CartoonGAN-Test-Pytorch-Torch.

[14] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. 2017. Universal style transfer via feature transforms. In *Advances in neural information processing systems*. 386–396.

[15] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2018. GANerated hands for real-time 3D hand tracking from monocular RGB. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 49–59.

[16] Dae Young Park and Kwang Hee Lee. 2018. Arbitrary Style Transfer with Style-Attentional Networks. *arXiv preprint arXiv:1812.02342* (2018).

[17] Maciej Pęśko, Adam Svystun, Paweł Andruszkiewicz, Przemysław Rokita, and Tomasz Trzciński. 2018. Comixify: Transform video into a comics. *arXiv preprint arXiv:1812.03473* (2018).

[18] Maciej Pęśko and Tomasz Trzciński. 2018. Neural comic style transfer: Case study. *arXiv preprint arXiv:1809.01726* (2018).

[19] Fischer Ronneberger and Brox. 2015. U-net: Convolutional networks for biomedical image segmentation.. In *n International Conference on Medical image computing and computer-assisted intervention, Springer*. 234âĂŞ241.

[20] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. 2018. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8242–8250.

[21] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[22] Gjorgji Strezoski and Marcel Worring. 2017. Omniart: multi-task deep learning for artistic data analysis. *arXiv preprint arXiv:1708.00684* (2017).

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[24] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2 (2014), 67–78.

[25] Chi Zhang, Yixin Zhu, and Song-Chun Zhu. 2018. MetaStyle: Three-Way Trade-Off Among Speed, Flexibility, and Quality in Neural Style Transfer. *arXiv preprint arXiv:1812.05233* (2018).

[26] Huihuang Zhao, Paul L Rosin, and Yu-Kun Lai. 2017. Automatic semantic style transfer using deep convolutional neural networks and soft masks. *arXiv preprint arXiv:1708.09641* (2017).

[27] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 2223–2232.

A APPENDIX

The data and code for this project are accessible on GitHub at: https://github.com/julinas/202-project3.