Life2Anime: Unpaired Image-to-Image Translation with ShenaniGAN

Anh Tuan Tran
Dept. of Comp. Sci.
University of Michigan
alantran@umich.edu

Justin Chang
Dept. of Comp. Sci.
University of Michigan
changjus@umich.edu

Raymond Ku
Dept. of Comp. Sci.
University of Michigan
rayku@umich.edu

Zhizhuo Zhou
Dept. of Comp. Sci.
University of Michigan
zhizhuo@umich.edu

Abstract

This paper compares three different approaches for unpaired image-to-image translation from real-life domain to anime domain: style transfer [2], CycleGAN [13], and our ShenaniGAN. While ShenaniGAN improved upon style transfer, CycleGAN is still superior in image-to-image translation.

1. Introduction

For decades, anime lovers around the word have preferred the dynamic anime world over the dull reality. They would do anything to be one step closer to an alternate anime reality where they can be the protagonist and defeat the demon lord. Unfortunately, time traveling and world line hopping technology are still a few millenniums away. Therefore, we do the next best thing, convert real life images to anime.

At the simplest instance, image-to-image translation is similar to style transfer [2] where we use intermediate layers from SqueezeNet [5] to transfer styles from a source image to a target image. However, style transfer introduces a lot of artifacts and degrades the content of the original image, resulting in suboptimal results.

In contrast to style transfer, paired image-to-image translation deep neural networks such as pix2pix [7] offers amazing visual results while preserving the content of the original image. However, pix2pix requires a paired dataset where image A is paired to image B. Such paired dataset is impractical to obtain for real life and anime images.

CycleGAN [13] enables image-to-image translation with unpaired training images using cycle consistency loss. More recently, the authors of CycleGAN released a new contrastive architecture for unspaired image-to-image translation, CUT [9]. The downside of CycleGAN is that it requires a lot of data and compute to achieve good results.

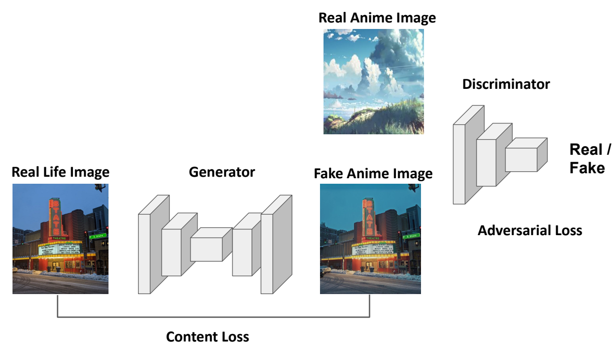

Figure 1. Overview of ShenaniGAN.

We create ShenaniGAN, a custom GAN [3] designed to offer good image-to-image translation performance with limited data and compute. ShenaniGAN follows the adversarial learning setup of a normal GAN but with an additional content loss to enforce consistency between input and output images. The generator of ShenaniGAN follows a U-Net [10] encoder-decoder architecture while the discriminator is simply a U-Net encoder with an additional pooling and fully connected layer.

We find that CycleGAN [13] achieves the best qualitative and quantitative performance for the Life2Anime image-to-image translation task; however, ShenaniGAN offers competitive performance.

2. Approach

We modify a GAN for the task of unpaired image-to-image translation by adding a content loss.

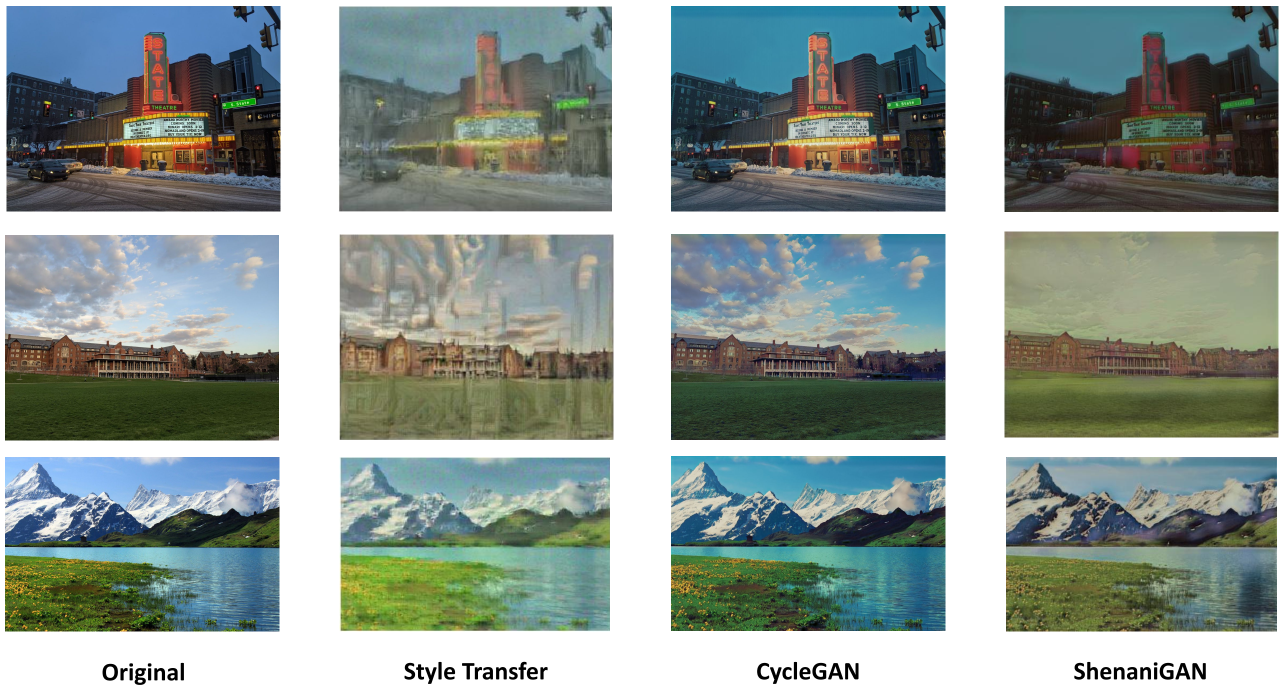

Figure 2. Comparison of Life2Anime using style transfer, CycleGAN [13], and ShenaniGAN.

2.1. Model architecture

Our Generator model closely follows U-Net [10] and is organized into blocks with similar layers but different input and output sizes. Each block contains a convolutional layer with kernel size 3 and padding 1; a 2D batch norm [6] layer; a ReLU layer; a dropout [12] layer with probability 0.2; another convolutional layer with the same parameters; a 2D batch norm layer; and a ReLU layer. The first convolutional layer of each block increases the number of channels of the input tensor. We use 5 of these blocks to gradually change the number of channels from 3 to 32, 64, 128, 256, and finally 512 channels, while the image size decreases from 64 to 32, 16, 8, and finally 4. Between each block is a 2D max pooling layer with kernel size 2. We follow this with an additional 4 blocks. Each of these 4 blocks take as input the concatenation of the output of the previous layer and the corresponding block in the first 5 blocks. For example, the 6th block takes as input the concatenation of the output of the 5th block and the 4th block, while the 7th block takes the 6th block and 3rd block outputs. These 4 blocks take the number of channels from 512 to 256, 128, 64, 32, and 3 again, while the image size increases from 4 to 8, 16, 32, and then 64, as in the original image.

Our Discriminator model uses the same block design as our Generator model. It has 4 blocks which takes the number of channels from 3 to 64, 128, 256, and finally 512, while the image size decreases from 64 to 32, 16, and finally 8. We then use a fully connected layer to produce a single scalar output, which is passed through a sigmoid function.

2.2. Loss Function

We used the GAN losses (on the Discriminator and Generator) and Content loss [1] as the training objective.

$$
\begin{aligned}
\mathcal{L}_D &= \mathrm{BCE}(D(G(\text{Real images})), \mathbf{0}) \\
&\quad + \mathrm{BCE}(D(\text{Anime images})), \mathbf{1}) \\
\mathcal{L}_G &= \mathrm{BCE}(D(G(\text{Real images})), \mathbf{1}) \\
&\quad + \|\mathrm{VGG}(\text{Real images}) - \mathrm{VGG}(G(\text{Real images}))\|_1
\end{aligned}
$$

The Discriminator GAN loss L_D is measured by calculating the binary cross entropy loss between the discriminator outputs when given real or generated anime images and the actual labels (real or generated). This loss is smaller when the discriminator successfully distinguishes between real and generated images. The Generator GAN loss L_G is measured by calculating the binary cross entropy loss between the discriminator outputs on generated images and the real label. This loss is smaller when the generator successfully causes the discriminator to mistake generated images as real ones. In addition, a content loss is added to the

Method	FID 2nd pool	FID 3rd pool
Style Transfer	76.5	2.55
CycleGAN	**19.8**	2.17
ShenaniGAN	23.0	**2.15**

Table 1. We evaluate FID [4] on the 2nd and 3rd pooling layers of VGG [11] between output images and a set of real anime images. A smaller FID indicates that the set of images are more similar to real anime images.

loss of the generator. This content loss is calculated by taking the L1 difference between the output of the 25th layer of the VGG16 [11] network on the generated anime images and the original real life images. This content loss aims to penalize differences in the subject of the generated images versus the original images, thus training our network to generate images which depicts the same subjects or scenes in the original.

2.3. Training procedure

We train ShenaniGAN for 20 epochs, which takes about an hour on a single NVidia V100 GPU. In each epoch, we generate anime images from minibatches of 32 real life images. We then run the discriminator on the these generator images, as well as 32 real anime images. The discriminator outputs are used to calculate the discriminator loss and update the discriminator parameters. We then run the discriminator on the generated images again. These outputs are used to calculate the GAN loss for the generator. We calculate the content loss and add it to the generator loss before taking the gradient and updating the generator parameters. We use the Adam optimizer with learning rate 0.0002 and betas of (0.5, 0.999).

3. Experiments

We compare the image-to-image translation quality of stlye transfer, CycleGAN [13], and ShenaniGAN both qualitatively and quantitatively. For all experiments, we use the same set of real life images at resolution of 384 by 512 pixels.

Style Transfer: We perform a simple style transfer approach based on techniques from [8] by extracting image features with a SqueezeNet [5] and updating the target image with a content loss, style loss, and a total variation loss. We use the 3rd layer as content layer, 1st, 4th, 6th, and 7th layer as style layers. We use content weight of $5e-2$, style weights of $3e4$, $1e3$, 15, 2, and a total variation weight of $5e-2$. We update the target image for 300 epochs with a landscape image from an anime.

CycleGAN: We use the default CycleGAN from [13] and train it on our unpaired dataset for 10 epochs. The training took several hours on a NVidia V100 GPU. We use default hyperparameters and only modify the dataset.

3.1. Qualitative Assessment

Figure 2 shows sample image outputs from style transfer, CycleGAN [13] and ShenaniGAN. Style transfer produces significant artifacts. ShenaniGAN is better at preserving original image details but CycleGAN appears the best at preserving original image details. Evaluation on the style is subjective: CycleGAN results are more true to life with some minor color shifts while ShenaniGAN performs more aggressive smoothing.

3.2. Quantitative Assessment

We use Frechet Inception Distance (FID) [4] to measure the similarities between the distribution of generated images and real anime images. Lower FID corresponds to more similar image distributions. We take the FID at the 2nd and 3rd intermediate pooling layers of VGG [11]. The 2nd layers focuses on more local details while the 3rd layer focuses on larger details. Both CycleGAN [13] and ShenaniGAN outperforms style transfer. ShenaniGAN achieves a lower FID at the 3rd pooling layer while CycleGAN achieves a lower FID at the 2nd pooling layer. This suggests that the CycleGAN is better at preserving low level details, which is reflected qualitatively, while ShenaniGAN sacrifices low level details for a smoother image similar to anime.

4. Implementation

We used the Pytorch library as well as other standard Python libraries to implement our code. We used an unpaired image dataloader from https://github.com/wonbeomjang/cyclegan-pytorch/ to read images from two different folders and put them together in a convenient way. We designed and implemented our own Generator and Discriminator models using Pytorch. We implemented our objective function and training loop. We used code from this PyTorch tutorial which prints training log messages and visualizes our images. For experimentation, we used the CycleGAN GitHub repository to generate anime images trained on our dataset and compare with the results of our model.

4.1. Data

We downloaded anime real real life movies and extracted individual frames from them to generate our dataset. We used 10 Ghibli movies, 7 Shinkai movies, and 10 other anime movies to extract 9781 anime frames. We extracted 6540 frames from Youtube videos and 10 real life movies. We used FFmpeg for frame extraction. These movies and videos were chosen arbitrarily, and thus are not paired.

References

[1] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9465–9474, 2018. 2

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 1

[3] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 1

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017. 3

[5] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 1, 3

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 2

[7] 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. 1

[8] Andrej Karpathy et al. Cs231n convolutional neural networks for visual recognition. *Neural networks*, 1(1), 2016. 3

[9] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer, 2020. 1

[10] 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. 1, 2

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 2

[13] 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. 1, 2, 3