- **Title**

  Style Transform Based On GAN

- **Who**

  Yuyang Wang, Zihan Miao, Chi Zhang

- **Introduction**

  We are implementing an existing paper called Generative Adversarial Networks for Image to Illustration Translation[1], which solved the problem that existing image-to-image style translation models cannot transfer both style and content at the same time. There are actually existing style transfer functions on some streaming websites and applications. Many live streaming software has a filter function on it. In fact, lots of these filter functions are based on GAN. But it is even more interesting to implement a style transfer model on our own. So we chose this paper.

- **Methodology**

  We decided to use the monet2photo dataset, the paper used to test their model, to produce our own results. Monet refers to Monet style painting while photo refers to real natural photos. This dataset is designed for style transfer and we also decreased the size of the dataset in order to decrease the training time.

  The edited monet training set includes 100 images in the trainA folder, the test set includes 121 images in the testA folder, the photo training set includes 400 images in the trainB folder, and the test set includes 751 images in the testB folder, where the images are all jpg files with size of 256x256.

  GAN is an unsupervised model. So we just need to feed the data into the model and let the generator contest with the discriminator. Model collapse is a major issue when training GAN models. When a model collapse happens, whatever the input is, the model produces the same result. Training of GAN is also highly unstable, the losses of generator and discriminator are quite bouncing. It is really important to find proper hyper parameters for the model.

  Here are the steps a GAN takes:

  ▪ The generator takes in random numbers and returns an image.

  ▪ This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.

  ▪ The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

So we have a double feedback loop:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.

When training the discriminator, we will hold the generator values constant; and when training the generator, we will hold the discriminator constant. Each should train against a static adversary.

GANILLA designed a new generator network in such a way that it preserves the content and transfers the style at the same time. A high-level architectural description of our generator network, GANILLA, is presented in Figure 2 along with the current state-of-the-art models and our two ablation models. GANILLA utilizes low-level features to preserve content while transferring the style. This model consists of two stages (Figure 1): the downsampling stage and the upsampling stage. The downsampling stage is a modified ResNet-18 network with the following modifiifications: In order to integrate low-level features at the down sampling stage, we concatenate features from previous layers at each layer of down sampling. Since low-level layers incorporate information like morphological features, edge and shape, they ensure that transferred image has the substructure of the input content.

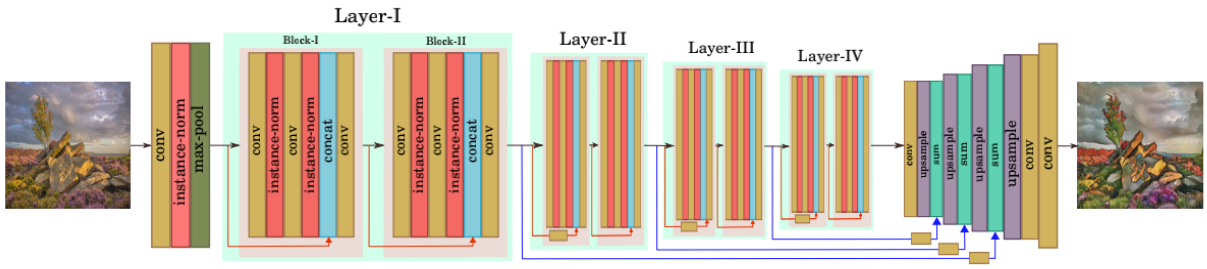


Figure 1: GANILLA generator network. The model uses concatenative skip connections for downsampling (from image up to Layer-IV). Then, the output of Layer-IV is sequentially upsampled while lower-level features from the downsampling stage are added to it via long, skip connections (blue arrows). The final output is a 3-channel stylized image.

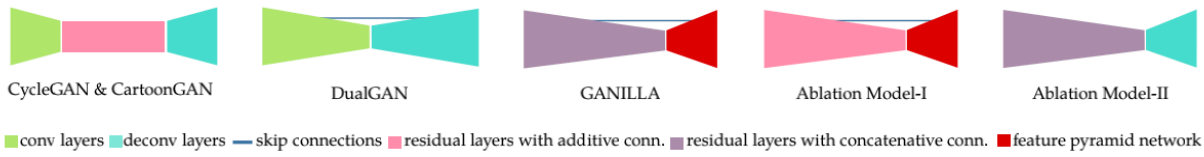


Figure 2: High-level architectural description of state-of-the-art models (CycleGAN, CartoonGAN, DualGAN), our model (GANILLA) and the two ablation models we experimented with. We follow the idea of cycle-consistency to train our GANILLA model. Specifically, there are two couples of generator-discriminator models. The first set(G) tries to map source images to the target domain, while the second set (F) takes input as the target domain images and tries to generate source images in a cyclic fashion.

Due to the limited computing power of our equipment, we simplified the model to the following figure3:

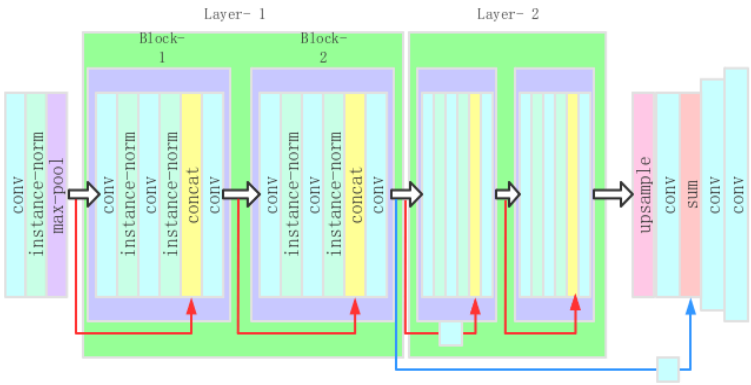


Figure 3: simplified model

Loss function:

Our loss function consists of two Minimax losses[2] for each Generator and Discriminator pair, and one cycle consistency loss[3]. Cycle consistency loss tries to ensure that a generated sample could be mapped back to source domain. We use L1

distance for cycle consistency loss. When we give source domain images to generator F , we expect no change on them since they already correspond to source domain. A similar situation applies when we feed the generator G with target domain images.

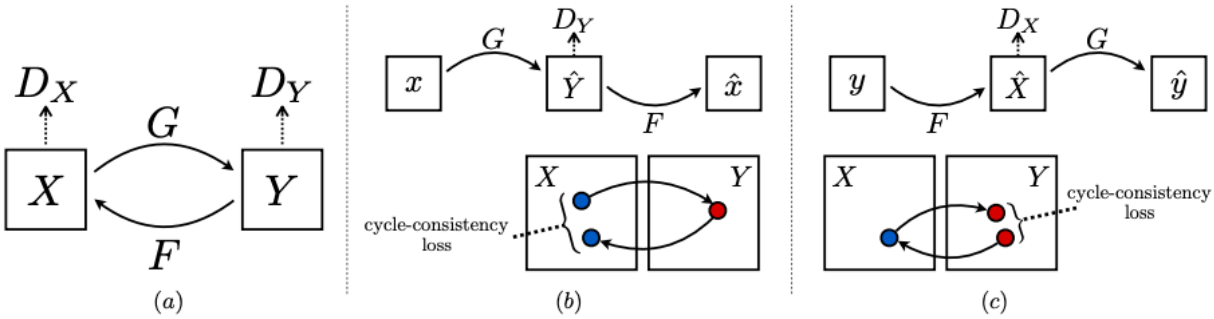
(1). Two Minimax losses:

The generator tries to minimize the following function while the discriminator tries to maximize it:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

In this function: D(x) is the discriminator's estimate of the probability that real data instance x is real. Ex is the expected value over all real data instances. G(z) is the generator's output when given noise z. D(G(z)) is the discriminator's estimate of the probability that a fake instance is real. Ez is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)). The formula derives from the cross-entropy between the real and generated distributions. The generator can't directly affect the log(D(x)) term in the function, so, for the generator, minimizing the loss is equivalent to minimizing log(1 - D(G(z))).

(2). Cycle consistency loss:



(a) The model contains two mapping functions G : X → Y and F : Y → X, and associated adversarial discriminators DY and DX. DY encourages G to translate X into outputs indistinguishable from domain Y , and vice versa for DX and F. To further regularize the mappings, we introduce two cycle consistency losses that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: x → G(x) → F(G(x)) ≈ x, and (c) backward cycle-consistency loss: y → F(y) → G(F(y)) ≈ y Cycle Consistency Loss is a type of loss used for generative adversarial networks that performs unpaired image-to-image translation. It was introduced with the CycleGAN architecture. For two domains X and Y, we want to learn a mapping G:X→Y and F:Y→X. We want to enforce the intuition that these mappings should be reverses of each other and that both mappings should be bijections. Cycle Consistency Loss encourages F(G(x))≈x and G(F(y))≈y. It reduces the space of possible mapping functions by enforcing forward and backwards consistency:

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

(3). Adversarial Loss We apply adversarial losses to both mapping functions. For the mapping function G : X → Y and its discriminator DY , we express the objective as:

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

where G tries to generate images G(x) that look similar to images from domain Y , while DY aims to distinguish between translated samples G(x) and real samples y. G aims to minimize this objective against an adversary D that tries to maximize it. We introduce a similar adversarial loss for the mapping function F : Y → X and its discriminator DX as well.

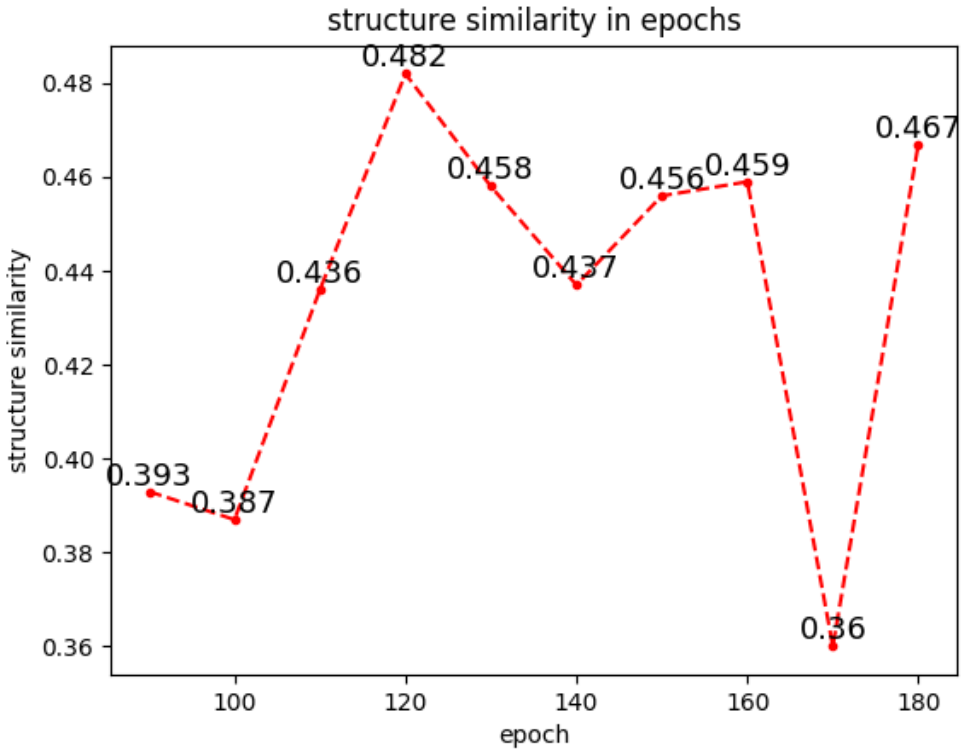
(4) The Full object of cycle consistency loss:

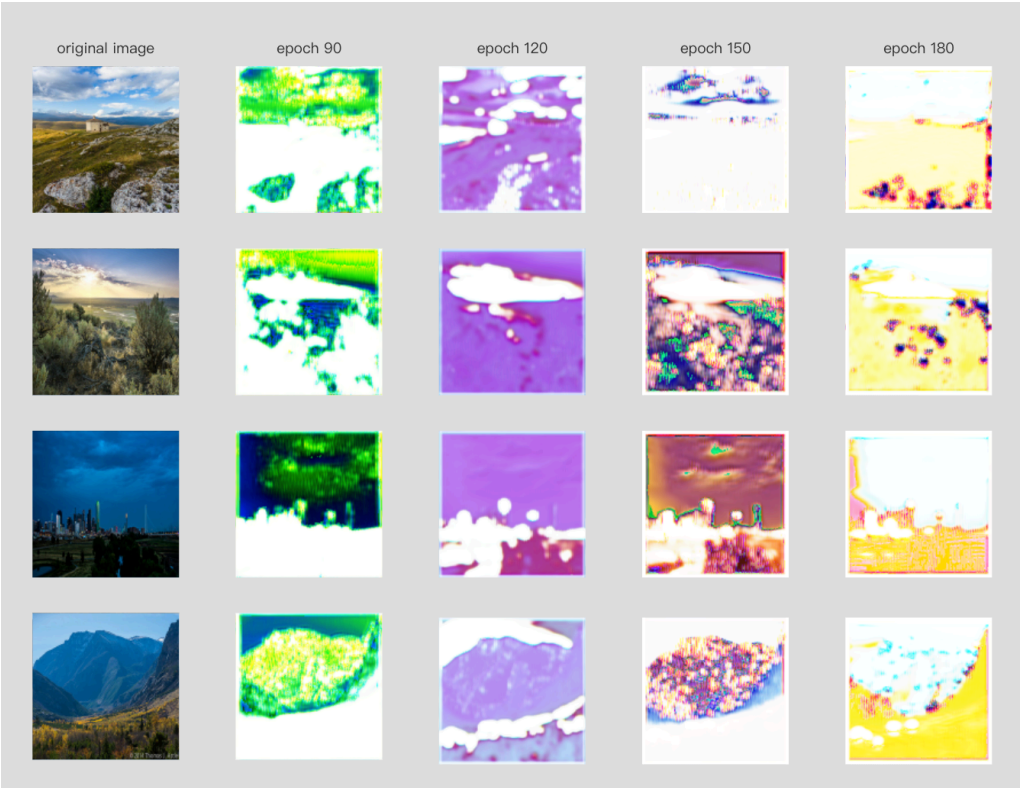
$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

where λ controls the relative importance of the two objectives. Our full objective function is to minimize the sum of the two Minimax losses function for each Generator and Discriminator pair, and one cycle consistency loss function.

- **Results**

  The model output did not meet our expectations. As you can see, the structure similarity increases steadily at first, but then starts to fluctuate up and down. Based on our analysis of the training process, we believe there are two possible reasons for this result: 1. Insufficient training data, as we only used a small portion of the data set due to machine limitations. 2. the learning rate was not changed during the training process, which caused the network to fail to converge in the backward epoch, and we should reduce the learning rate and retrain after a period of training.


structure similarity in epochs
structure similarity
epoch
0.48
0.46
0.44
0.42
0.40
0.38
0.36
100
120
140
160
180
0.393
0.387
0.436
0.482
0.458
0.437
0.456
0.459
0.36
0.467


original image
epoch 90
epoch 120
epoch 150
epoch 180

- **Challenges**

  After thoroughly viewing the code provided with the paper, we found that the code is hard to read due to the architecture of the model. In this case, we tried to build the model following the instructions given by the paper instead of following the code. After roughly building the model, during the training and optimizing process, we found that it is difficult to set up the appropriate hyperparameters due to the high time cost per training and the frequent collapse of the GAN model.

- **Reflection**
- How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?

  Our model is able to switch styles, but due to the omission of the training strategy, our model did not achieve the best state that we expected.
- How did your approach change over time? What kind of pivots did you make, if any? Would you have done differently if you could do your project over again?

  At first, we decided to re-implement the paper completely, then we found it is impossible because the requirement of computation and time is beyond the resources we can find. So we reduce the depth of the original network. If we did it over again, we may deepen our network and try to use pre-trained weights as our start to enhance the result.
- What do you think you can further improve on if you had more time?

  we can train the network with different merged dataset to produce better results.
- What are your biggest takeaways from this project/what did you learn?

  We have a deeper understanding of the GANILLA. We know how to construct a GANILLA generator network, how to downsample and upsample in the network. We also know how to compute the loss function and the meaning of the loss function. We also improved our debugging skills during coding. This project also cultivated our ability to cooperate with team members. We have learned a lot from each other.

Reference:
[1] Hicsonmez, S., Samet, N., Akbas, E. and Duygulu, P., 2020. GANILLA: Generative adversarial networks for image to illustration translation. Image and Vision Computing, 95, p.103886.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[3] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, IEEE International Conference on Computer Vision.