Image-to-Image Translation using CNN and Cycle-Consistent Adversarial Networks

Xuejiao Li xjli1013@stanford.edu
Wenxin Wei wxwei@stanford.edu
Ip Chun Chan kaichun2@stanford.edu

***Abstract*—This project aims to make image-to-image translation for different tasks using both neural style transfer technique and cycle-consistent adversarial networks. Using our own dataset generated by a combination of online images, ImageNet dataset, and Flickr dataset and Wikiart.org, we have successfully generated images for different tasks. Our innovation is we train and test the cycle-consistent adversarial networks using our own dataset. We have also tried to fine-tune hyperparameters such as batch size, learning rate, lambda in loss function and trying to add dropout to our network. The experiment results show that our method is able to successfully transfer for different tasks while preserving the original content.**

***Keywords*—Image-to-Image Translation, Neural Style Transfer, Cycle-Consistent Adversarial Networks**

I. INTRODUCTION

Image style transfer has been popular these days for producing samples for visualizing interior design, computer games items and create drawings. Lots of people use filters on camera to get funny pictures (ex: snapchat filters, Instagram filters). However, style transfer between two images is a less studied problem. Given both a content image and a style image, photographic style transfer will recreate the content image in the style of the second image, which can be used to transfer effects such as time of day, season, and illumination. Existing techniques are not enough to produce accurate transferred images, so we want to explore more and try to improve the existing algorithms. For example, if we transfer the style of a painting to a photograph, the final output tends to have a very generalized style, because a painting commonly has a consistent style, but a photographs tend to have a more localized style. In addition, for many tasks, paired training data will not be available.

We present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples using CycleGAN (Cycle-consistent Generative Adversarial Networks). Our project is mainly about transferring the look of one photo onto another, while still looking like a photo, and especially focusing on implementing CycleGAN for unpaired examples. For CycleGAN algorithm, our objective contains two types of terms: *adversarial losses* for matching the distribution of generated images to the data distribution in the target domain; and *cycle consistency losses* to prevent the learned mappings G and F from contradicting each other. These two make CycleGAN more powerful than traditional methods.

With this technique, we can not only transfer input images into artistic styles of Monet, Van Gogh, Ukiyo-e, and Cezanne, but also make object transfiguration such as changing between a horse and a zebra, an apple and orange and even changing the background environment into a snowing scene.

We can imagine all these despite neither having seen a side by side example of a Monet painting next to a photo of the scene he painted, nor snow in California. Instead, we have knowledge of the set of Monet paintings or snow images and of the set of landscape photographs. We can reason about the stylistic differences between these two sets, and thereby imagine what a scene might look like if we were to "translate" it from one set into the other.

We apply our method to a wide range of applications, including collection style transfer, object transfiguration, season transfer and photo enhancement. These applications can be adapted by companies like Instagram or Snapchat to create interesting filters, or even by autonomous driving companies to improve their performance based on the snow images we generated.

II. RELATED WORK

CNN has been proved to have powerful ability for image style transformation. Our work with CNN is based on the paper *Deep Photo Style Transfer* by Luan et al [1]. This paper builds upon the well known work Neural Style Transfer by Gatys et al. [2]. Throughout this paper we will give an in-depth explanation of image style transfer as described in [1] and discuss specifics of how to improve the original algorithm to generalize well to two unpaired images. The largest improvements in this method are gained through CycleGAN. This newly developed algorithm is more powerful than the traditional CNN to do image style transfer. Our work with CycleGAN is based on the paper *Unpaired Image-to-Image*

Translation using Cycle-Consistent Adversarial Networks by Zhu et al [3]. Building on the model introduced in the paper, we use combined datasets to do different image style transferring tasks such as let there be snow in California.

Generative Adversarial Networks (GANs) [4] have achieved impressive results in image generation, image editing , and representation learning. The key to GANs' success is the idea of an *adversarial loss* that forces the generated images to be, in principle, indistinguishable from real photos. This loss is particularly powerful for image generation tasks, as this is exactly the objective that much of computer graphics aims to optimize. We adopt an adversarial loss to learn the mapping such that the translated images cannot be distinguished from images in the target domain.

Cycle-Consistency uses transitivity as a way to regularize structured data. In visual tracking, enforcing simple forward-backward consistency has been a standard trick for decades [5]. More recently, higher-order cycle consistency has been used in structure from motion, 3D shape matching, co-segmentation, and depth estimation. Our work is very similar to that of Zhu et al [3], Zhou et al. [6] and Go- dard et al. [7], as they use a *cycle consistency loss* as a way of using transitivity to supervise CNN training.

III. DATASET AND FEATURES

A. *Dataset preparation*

We gather our own unique dataset for both neural style transfer and cycle-consistent adversarial networks from google internet, ImageNet and Flickr.

Neural style transfer task. We choose to use images directly from the internet, and use some painting images as our style image. We then scale our images to 400*300 pixels.

For cycle-consistent adversarial networks, depending on different applications, we choose to use different dataset.

Making snow on the image task. We choose to use dataset provided by Flickr with the tag yosemite. The images are scaled to 256*256 pixels. And our training set contains 1200 original images and 850 snowy images. For test set, we randomly download some images from the internet to gather the result.

Horse to Zebra and Apple to Orange task. We choose to use dataset provided by ImageNet that contains horse, zebra, apple and orange as our training set. We scale the images to 256*256 pixels. For training set, we use 800 images for horse, 1000 images for zebra, 1000 images for apple and 1000 images for orange. For test set, we randomly download some images from the internet and gather the result.

Monet painting to photo task. We gather Monet paintings both from the internet and from Wikiart.org. The photo images are obtained from both the internet and Flickr. We also scale the images to 256*256 pixels. Again, for test set, we randomly download some images from the internet and gather the result.

IV. METHODS

A. *Neural Style Transfer*

We first used Neural Style Transfer to perform image-to-image translation, which synthesizes a novel image by combining the content of one image with the style of another image. We used VGG convolutional neural network and we defined a cost function for both content and style pictures and then use gradient descent to minimize the combined cost. The overall cost function is the following

$$J(G) = \alpha\, J_{content}(C, G) + \beta\, J_{style}(S, G)$$

where and are the hyper-parameter where we tuned to get the minimum loss. For the input of the cost functions, C means the content image, S means the style image and G means the generated image. In particular, we used $J_{content}(C, G) = 0.5\times(a^{[l](c)} - a^{[l](G)})2$ where $a^{[l]}$is the activation of layer on the images. For style loss, we used Gram matrix (ex: $G_{kk\prime}^{[l]} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a_{ijk}^{[l]} a_{ijk\prime}^{[l]}$) to get the cost function $J_{Style}(C, G) = (G^{[l](S)} - G^{[l](G)})\ {}_F^2$ where i and j corresponds to the vertical and horizontal position of the image and k and k' corresponds to the channel index.

Rather than mapping one specific image to another one, we trained the network to learn the mapping from one collection to another collection (ex: from a background environment to a snowing background). This learning can be applied to other tasks and change all the new input images to have the same feature.

B. *Cycle-Consistent Adversarial Networks*

Our second method is to combine adversarial losses and cycle consistency losses to learn mapping functions between two domains.

B1a) Adversarial Loss

We used the following formula as the adversarial loss $L_{GAN} = E[logD_Y(y)] + E[log(1 - D_Y(G(x)))]$ where $D_Y$is a discriminator that encourages G which is a mapping function that maps from domain X to Y to translate image X into output that is indistinguishable from domain Y. The first E is the expected value taken from the data distribution x~p(x) and the second E is the expected value taken from the data distribution y~p(y).

In the above formula, the purpose of the mapping function G is to generate images G(x) that look similar to images from domain Y. We used a similar function for the mapping function F which maps from domain Y to X and the adversarial loss is as follows:
$L_{GAN} = E[logD_X(x)] + E[log(1 - D_X(F(y)))]$ with the first E taken from the data distribution y~p(y) and the second E taken from the data distribution x~p(x).

B1b) Cycle Consistency Loss

Using adversarial loss alone cannot guarantee every input x mapped to a desire output y, because with large capacity, a network can map the same set of input images to any random permutation of images in the target domain. To reduce the mapping function's dimensions, we need to make sure that the image translation cycle will bring the x back to the original image. So we used the following Cycle Consistency Loss formula:

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

where F is a mapping from Y to X and G is a mapping from domain X to Y. In order to be cycle-consistent, F(G(x)) should map back to x and G(F(y)) should map back to y.

B2) Network Architecture

We use the the architecture from Johnson et al. [8] as the starting point of our network, because it showed impressive results for neural style transfer. This network contains two stride-2 convolutions, several residual blocks [9], and two fractionally strided convolutions with stride 1/2 . We use 6 blocks for 128 × 128 images and 9 blocks for 256 × 256 and higher resolution training images. Similar to Johnson et al. [8], we use instance normalization [10]. We use 70 × 70 PatchGANs [11] for the discriminator networks, which aim to classify whether 70 × 70 overlapping image patches are real or fake. Such a patch-level discriminator architecture has fewer parameters than a full-image discriminator and can work on arbitrarily-sized images in a fully convolutional fashion [11].

V. EXPERIMENTS AND RESULTS

A. Training Details

Training Environment

For neural style transfer, we simply use MacOS CPU to generate the result. Our training environment is shown in table 1.

Table 1: Training Environment

Language	Python3.5
Framework	Google TensorFlow 1.4.0

For cycle-consistent adversarial networks, we choose to use a GPU to train our network from scratch, since the required computation power of neural network is huge. Our training environment is shown in table 2.

Table 2: Training Environment

Language	Python3.5
Framework	PyTorch 0.4
GPU	Nvidia GeForce GTX 960M
GPU Memory	4044MB

Initialization

Weights are initialized randomly from a truncated normal distribution with zero mean and 0.001 standard deviation for symmetry breaking.

With proper data normalization, it is reasonable to assume that approximately half of the weights will be positive and half of them will be negative. Therefore, we want the weights to be very close to zero, but not identically zero, because if every neuron in the network computes the same output, then they will also all compute the same gradients during back-propagation and makes the exact same parameter updates [12].

Learning rate

The learning rate is 0.00025 for the first 150 epochs and we use linear decay every 30 epochs towards the end. The learning rate for the latter is relatively smaller, since we are fine-tuning the model for the latter steps. We tried several combinations and find this one can obtain both high efficiency and good convergence.

Batch size

Unlike the original paper, we use batch size for gradient descent equal to 5.

We choose batch size not equal to 1 to avoid overfitting. And we observe a better trade-off between performance and hardware limitation.

Update Method

We applied Adam update, which in practice works slightly better than standard momentum.

Regularization

We have also applied dropout method as regularization technique, which can reduce overfitting in neural networks by preventing complex co-adaptations on training data. In the experiment, drop probability is equal to 0.5, as it can maximize number of randomly-generated network structures [12].

Loss Function

Our full objective is:

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

where the first two are adversarial losses aiming to minimize G(x) and y; F(y) and x. The last part is the cycle consistency loss, aiming to reconstruct images F(G(x)) to match the input images x. We have tried to fine-tune lambda by a number of combinations and find the original 10 to be the best choice.

B. Result

Our tasks include: *Neural style transfer task, Making snow on the image task, Horse to Zebra task, Apple to Orange task and Monet painting to photo task.*

For neural style transfer task, we perform 200 iterations for a given content and style image, and use a pre-trained VGG model to generate the artistic image.

For other tasks we perform 250 epochs for training, and after training we perform a single generation on our test-set to generate images.

Neural style transfer task

We load the content image, style image and VGG16 model, and randomly initialize the image to be generated, and then run 200 iterations and updates the generated image at every step.

We plot Fig.1, Fig. 2 and Fig.3 to illustrate one of the generated image, where we successfully transferred a real photo into a oil-painting.

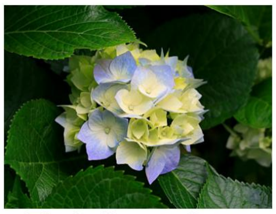
Fig.1. Content Image

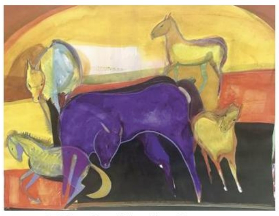
Fig.2. Style Image

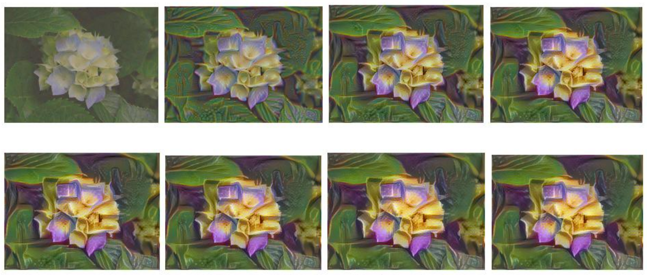
Fig.3. Generated Image

Making snow on the image task

We perform 250 epochs for training, and after training we perform a single generation on our test-set to generate images.

We plot the some of the generated images illustrated in Fig. 4, 5, 6 and 7.

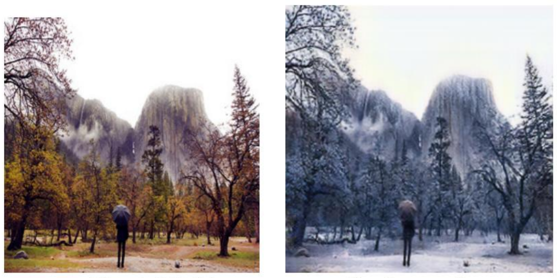
Fig.4. Making snow on the image 1

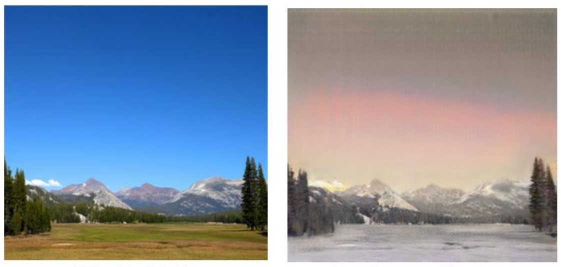
Fig.5. Making snow on the image 2

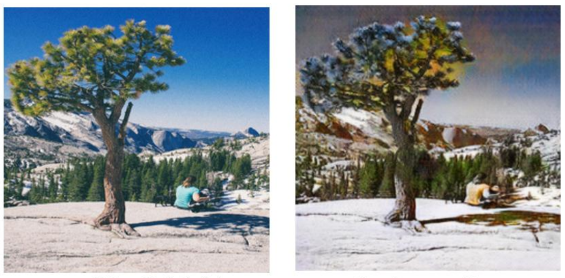
Fig.6. Making snow on the image 3

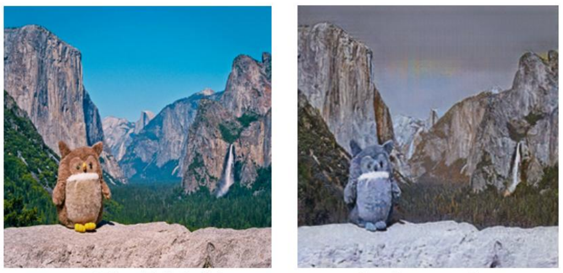
Fig.7. Making snow on the image 4

Horse to Zebra and Apple to Orange task

We also trained for Horse to Zebra and Apple to Orange task using the same technique described above. We plot some of the generated images illustrated in Fig. 8, 9, 10.

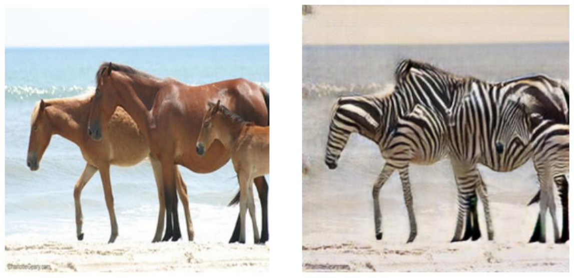
Fig.8. Horse to Zebra 1

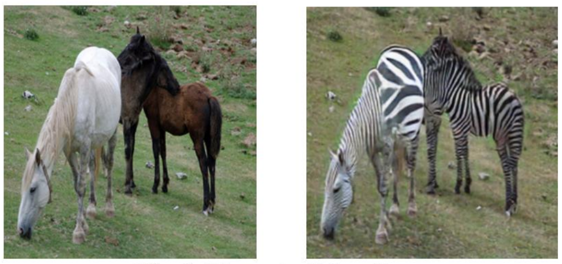
Fig.9. Horse to Zebra 2

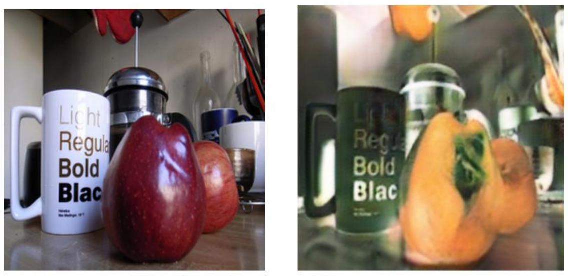

Fig.10. Apple to Orange

Monet painting to photo task

Again, we trained for Monet painting to photo task using the same technique described above. We plot some of the generated images illustrated in Fig. 11 and 12.

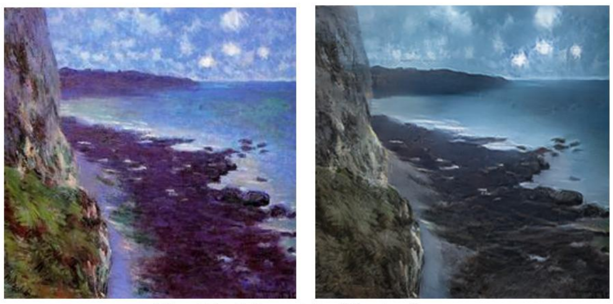

Fig.11. Monet to Photo 1

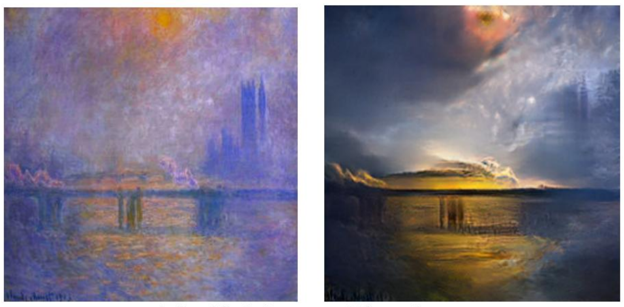

Fig.12. Monet to Photo2

C. Analysis and discussion

We show some generated images from the test set for the snow transfer task in Fig. 4-7. As shown in the examples, our method is able to successfully transfer the snow to the test image while preserving the original content. In Fig.6, note the bright green shirt is transferred into a dark yellow shirt. This is because trees become bare in the winter and our network has learned to transfer green image regions into dark yellow.

In Fig.7-8, we show successful examples of horse to zebra transfer. In Fig.10, we show a failure case for apple to orange transfer. First, note a red lamp from the source image is transferred into an orange lamp. This is because our method mistakenly treats the red lamp as an apple and transfers it into an orange. In addition, a white cup is also transferred into a green cup. We hypothesize that the network has learned to transfer white image regions into green because the training images of oranges contain many green leaves.

VI. CONCLUSION AND FUTURE WORK

In this project, we tried two different methods to do image-to-image translation, namely, neural style transfer and cycle-consistent adversarial network. Our innovation is we train and test the cycle gan using our own dataset. We also tried to fine-tune hyperparameters such as batch size, lambda in loss function and trying to add dropout to our network. The experiment results show that our method is able to successfully transfer for different tasks while preserving the original content.

CONTRIBUTIONS

The team worked together to run the training, process result data, plot curves and tables, as well as make poster and write the final report. We share ideas through frequent discussions and cooperate each other.

CODE [13, 14, 15]

Github: https://github.com/kaichun2/cs230

REFERENCES

[1] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. *arXiv preprint arXiv:1703.07511*, 2017.

[2] L. Gatys, A. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[3] J. Zhu, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, 2017.

[4] .I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.

[5] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward- backward error: Automatic detection of tracking failures. In ICPR, 2010.

[6] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d- guided cycle consistency. In CVPR, 2016

[7] C. Godard, O. Mac Aodha, and G. J. Brostow. Un- supervised monocular depth estimation with left-right consistency. In CVPR, 2017

[8] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In ECCV, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[10] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016.

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image- to-image translation with conditional adversarial networks. In CVPR, 2017.

[12] xjli, Xuejiao Li and Zixuan Zhou zixuan. "Speech Command Recognition with Convolutional Neural Network." (2017).

[13] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[14] CS230, coursera, course 4, Art generation with Neural Style Transfer

[15] CS231N, assignment 3, Q4, Style Transfer