Project Machine Learning
— Milestone 1 —

Hazem Lahoual, Shashi Durbha, Wai Tang Victor Chan

November 24, 2019

1 Outline

This report details our work in milestone 1. It begins in Section 2 with a brief introduction to the image transfer model and the project in general. Then we gave an overview on the dataset, followed by the pre-processing steps for the data and what we plan to do with the dataset over the course of the project in Section 3. In Section 4, we discuss two baseline models (SVD and AutoEncoder with WCT) after which we describe the prototype and try to compare the baseline models with the prototype in model evaluations. In the final section, we explain a few real-world applications for the model and how we plan to continue our work in future milestones.

2 Overview

2.1 An image domain transfer model

Zhu et al. (2017) proposed cycle-consistent adversarial networks, which is used for the task of image domain transfer, such as style transfer and object transfiguration, and is trained from unpaired images.

Such a model translate an image in a source domain (X) into an image in a target domain (Y) . An inverse model which translates images back into the source domain is trained concurrently, with a cyclic-consistency term introduced in the objective function.

In Mathematical terms, we attempt to find a mapping (G) between the Source domain (X) and the Target domain (Y) such that $G : X \to Y$. At the same time, We find an inverse translation F of the Target domain (Y) back to the Source domain (X) such that $F : Y \to X$. More will be explained and demonstrated in Section 4.3.

2.2 Our project

Our group aimed to study the properties of this network and experiment with different domain pairs to explore the possibilities achieved by this model.

During milestone 1, we acquired image data needed for the horse-to-zebra transfiguration model and some photos-to-painting style transfer models. We inspected the data and carried out data cleanup. We also implemented a horse-to-zebra prototype trained and performed basic model evaluation. All these will be beneficial for our future works with models of other categories or models of higher complexities.

3 Datasets

3.1 Datasets Overview

The images used in our work were downloaded from two sources. The landscape dataset was downloaded from Flickr[1] using the tags "landscape"(497 images), landscapephotography"(500 images) and "nature"(497 images), then, they are grouped together in one dataset called "landscape" for a total of 1494 images. The artworks were downloaded from wikiart.org[2]. The artists whose works were downloaded included Claude

[1] `https://www.flickr.com/`, an image hosting service, where people can share their photos and videos.
[2] `https://www.wikiart.org/`, a visual art encyclopedia.

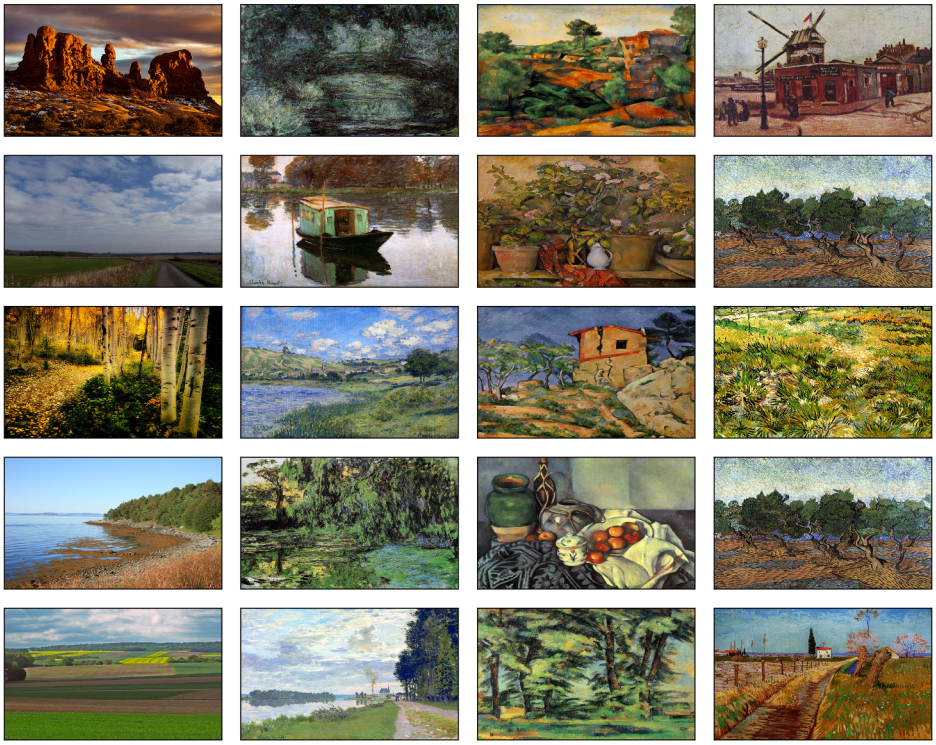


Figure 1: Samples from the downloaded datasets. Each column represents one class. Classes from left to right: landscape, Monet, Cezanne, Van Gogh.

Monet, Vincent van Gogh and Paul Cezanne (Cézanne). The training set size is 1348 for "Monet", 1931 "Van Gogh" and 583 for "Cezanne", respectively. Figure 1 shows samples from the four different classes: "landscape", "monet", "vangogh" and "cezanne". The artworks acquired include paintings of landscapes, buildings, indoor scenes and portrait of people.

We divided the images into 3 datasets: landscpae2monet, landscape2vangogh and landscape2cezanne. Each dataset is divided into two subfolders: subfolder A contains the dataset "landscape" , and subfolder B contains the images of artist Monet, Van Gogh and Cezanne respectively.

We also wanted to test the performance of object transfiguration, for example from horse to zebra or from apple to orange. We made an attempt to download images directly from ImageNet, as suggested by the original paper, but the website was under maintenance, and thus the needed dataset required special permission to download. The complete ImageNet dataset was available but due to its massive storage requirement (around 150 gb), it was not feasible to us. As an alternative, we downloaded the public dataset horse2zebra[3], which contains the same images of horses and zebras as ImageNet (using the keys "wild horse" and "zebra") but resized and cropped to 256×256 px. The "horse2zebra" dataset is divided into two subfolders: Folder A contains 1187 images of horses, and folder B contains 1474 images of zebras. Figure 2 shows some samples taken from it.

3.2 Data Preprocessing

The first step is data cleanup. In the case of image data set for the training of our model, this may require sophisticated tools for detection of unsuitable images, irrelevant images or duplicate images. Without those tools, such tasks need to be performed manually.

Our first task is to remove grayscale images. As for our case for style transfer, or for example in case day-to-night transformation, grayscale images will lead to bad results by the generators. As doing this manually will take a lot of time and effort, we tried to do it in a basic way: load an image, check number of channels, if it is less than three then

[3] `https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/dataset/`, to download horse2zebra dataset.

Figure 2: Samples from "horse2zebra" dataset.

discard and move to the next image in the dataset. But this method is a time-consuming, taking into consideration we are still using small number of samples. Despite that, it does not solve the problem completely, as the predominantly dark or predominantly bright images will not be pruned and will be used for the training.

Another way to do this first step of the preprocessing pipeline (not tried until now, but will be included later) is to look at the number of colors used and then define a threshold. Discard images with less than 20 colors used for example, as images too dark or too bright will fall in this category. We can also remove images that we think that they are not colorful enough to help to reproduce the art style. This can be done by increasing the threshold to 50. This method will be useful later to improve the performance of the generators, as we want to keep only the images that are important to obtain the desired art style.

The second step is to split the data into train set and test set. We randomly chose 90% of samples as train samples and the remaining 10% as test samples. We have a small test split, as a generative network requires a large training data set. On the other hand, the testing of the performance of the model will predominantly be performed manually, and therefore we do not need a huge test set to generate numerical statistics.

There is a different set of processing that needs to be applied to images as they are fed into our newwork during training. As our model is memory-intensive, a few transformations are needed before the training. The images are resized to 286×286 px using bicubic interpolation method. After that, a random cropping will be applied to get an image of size 256×256 px. We add a random horizontal flip with uniform probability $p = 0.5$. Finally, we normalize the image, so the pixels in each channel are transformed from $[0, 255]$ to $[0, 1]$ range. A Second normalization is applied then by subtracting by a mean of 0.5 and dividing by a standard deviation of 0.5, to get in the end images of mean 0 and values in [-1,1].

The normalization step is important in our pipeline. By doing this, each pixel in our input image will have a similar data distribution. Therefore, during the training phase, the gradients will act uniformly on every channel. This will help to improve the speed of convergence and the stability of the network (LeCun et al. (2012)). The second normalization can be explained by the fact that the generator has tanh function as activation function in the last layer, that means the values of the output of the generators/input of the discriminator will be in range of [-1,1]. Then, by scaling the image to the same range will give equal data for discriminator to be trained upon.

We resize the images to a common smaller size to speed up the training. The random cropping and the random horizontal flip will be applied differently at each epoch. This way, the generators will be be exposed to wide range of variations of the input data, which makes them less likely to overfit. This step of data augmentation is crucial, as we have a small datasets for the training.

3.3 Discussion

One problem encountered when searching for images to create the datasets is copyrights claims. Not all photos shared on Flickr are public and free to to download and use, as some need permission from the owner. So, we only looked for public and open source pictures which limited our access to a bigger and more diverse data. The same things goes for wikiart, where we wanted to extend our list of artists used in our work, but the artworks of many of them like Pablo Picasso or Salvador Dali are not available on copyright grounds. That's why we limit ourselves to the three artists mentioned above, that means to Monet, Van Gogh and Cezanne, and we will extend our data to other art

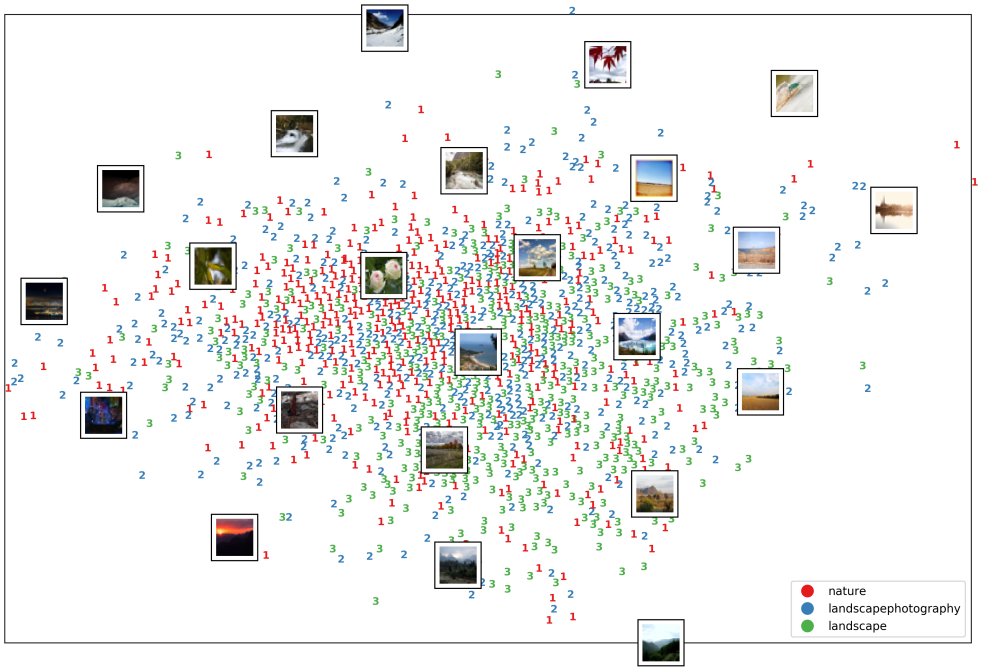


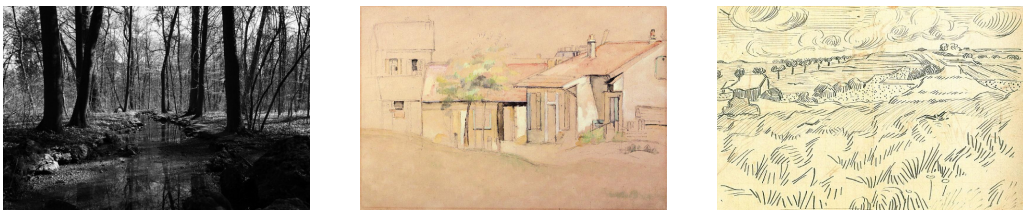
Figure 3: Visualization of "landscape" dataset using t-SNE. Here, the 1's represent the images downloaded using the tag "nature", 2's using the tag "landscapephotography" and 3's using the tag "landscape"

style later after making sure that our algorithm works as intended.

Another problem encountered is (near) duplicate images downloaded from Flickr. We used different tags to download landscape images. By doing so, we are not sure if there are duplicate or very similar images downloaded, and if there are any, how to detect and remove them. Our goal is to have data as different as possible, so that our model will not overfit. To overcome this, we needed a tool to visualize the datasets and have a general overview of its distribution. For this, we used t-distributed stochastic neighbor embedding (t-SNE) (see van der Maaten and Hinton (2008)), which is a dimension reduction and visualization tool where similar images are be modeled by nearby points and dissimilar images are modeled by distant points. This way, we can have a broad idea how good is our data, and then decide if we need to remove some images manually or download a new ones with different tags.

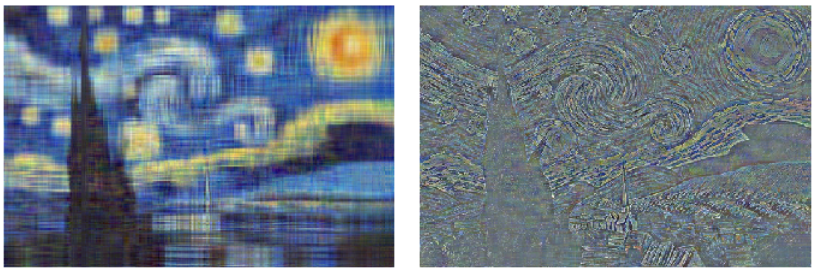
Figure 3 shows the visualization of the dataset "landscape" using t-SNE. We can see that there is no clusters, and most of the points are near each others. That means our data is very similar. A quick look at the images confirms this: we have a lot of images of flowers, trees and plants in general (a lot of green color). This explains why a lot of points in the middle of the visualization are located nearby. However, the points that represent mountains, hills, rivers, sea and other kind of landscapes are few and far from each other. As we don't want our generators to overfit, that means in our case will only work well on plants images, we need to balance our data and add more images of different types of landscapes, or delete images of plants in the case we have a big dataset.

One way to improve our datasets in the future is to remove sketches and incomplete works from the images, as keeping them will make the performance of the generators worsen. Moreover, only the late works of Van Gogh can be considered for the training, as the art style which made Van Gogh famous is only found in the paintings of the last few years of life of the Dutch artist. Also, we can limit the data to cover only paintings of landscapes, as in our case almost half of the artworks is about indoor scenes or portrait of persons. For now, we will keep the dataset as it is, as we don't have enough images for the training. Finally, to keep it safe for work, we need to go through all images and remove the ones that contain nudity.



(a) grayscale image (b) incomplete painting (c) sketch

Figure 4: Examples of images that can be removed from the datasets. Only grayscale images are discarded during the preprocessing phase



(a) compressed image (b) residual mage

Figure 5: SVD compression was performed on the image of a painting. The first 27 out of 347 layers were retained during compression.

4 Baseline Methods and Prototype Evaluation

In order to assess the difficulty of the task of image domain/style transfer for this task and the need for deep neural network learning, we first experimented with two simpler baselines. Then we compared the baseline results with our prototype and performed some simple evaluations.

4.1 Baseline 1: Singular Value Decomposition

The first baseline involves Singular Value Decomposition (SVD), which generates layers of eigenimages of descending luminance and frequency. While this technique is mostly adopted for image denoising or compression (see Sadek, 2012), we observed that residual layers, which are usually discarded after image compression, are associated with certain artistic styles of the images, such as edges with stylistic characters, as shown in Figure 5b.

To impose a style to a content image, we first separated each of content and style images into a compressed image and a residual image. Then we adapted the residual of the content to the residual of the style, using a classic signal whitening and coloring transformation (WCT), inspired by Li et al. (2017). The output was obtained by combining the content layers and the adapted residual layers.

The first experimental results were shown in Figure 6. We observed that the artistic style failed to be projected onto the output layers. The most visible change would be the watermark/noise effect caused by the high-frequency layer of the style image.

This naive baseline failed worse when we performed object transfiguration such as horse to zebra. In Figure 7c, although some subtle light edges of a zebra were introduced onto a horse, the result was far from convincing. The case of zebra-to-horse (Figure 7d) transfiguration was mostly a work of desaturation.

SVD is insufficient for two reasons. First, the features extracted by SVD carry little semantic information, which is yet crucial to the task of object transfiguration. Second, the transformed residual layer can only affect the compressed image by he act of superimposition, which is not effective to perform image domain transfer.

(a) content (b) style (c) output

Figure 6: The results of a stylization by WCT on SVD residual. 25 out of 256 layers were retained during compression.

(a) horse (b) zebra (c) horse to zebra (d) zebra to horse

Figure 7: The results of a stylization by WCT on SVD residual. 25 out of 256 layers were retained during compression.

4.2 Baseline 2: Autoencoder with Whitening and Coloring Transform

Although the use of the second baseline involves a neural network, it is simple in the sense of universality, where a single network can be used for all domain pairs. The only domain-specific information we need to extract from training data of a domain is the WCT matrix of the domain.

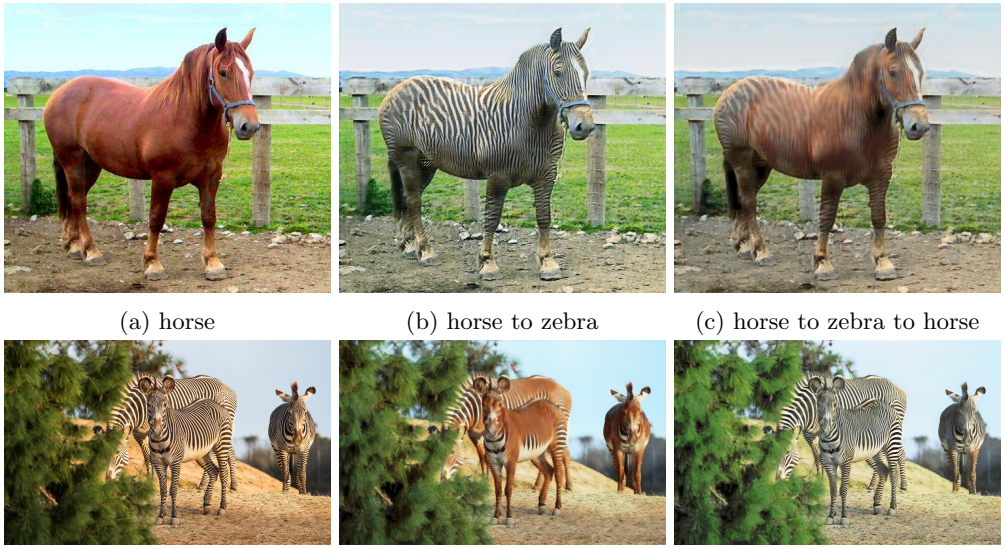
This approach was adapted from Li et al. (2017) and Li et al. (2018), where a pre-trained encoder (VGG-19) is used as feature extracter. The features are then adapted to the intended style using again WCT. The adapated features are then passed to a pre-trained decoder to reconstruct the image. We note that the training of the encoder and decoder is entirely independent of the stylization process.

We performed the same tests as we did in Figures 6 and 7. While this model seems to be better at artistic stylization, inconsistent stylizations in semantically uniform regions are visible throughout Figure 8a. The same problem can be observed in its performance in the task of transformation between horse and zebra, where the task of object transfiguration was interpreted as artistic stylization and thus the focus was put on blind colorization.

Overall, the failure of baseline 2 was expected as the model lacks the knowledge for

(a) stylized landscape (b) horse to zebra (c) zebra to horse

Figure 8: The results of stylization by an autoencoder with WCT in the deepest layer.



(a) horse (b) horse to zebra (c) horse to zebra to horse

(d) zebra (e) zebra to horse (f) zebra to horse to zebra

Figure 9: Some results from the prototype trained with 50 epochs.

the general domain, such as the ability to identify a genuine horse or zebra. The features extracted by an autoencoder trained from Imagenet may also not be suitable for the task of transfer between any domain pairs.

4.3 Prototype

Our model is entirely based on the methods propsed in Zhu et al. (2017) and the implementation was adapted from Zhu (2017) and Ruano (2017). There are two key main ideas.

First, the model is a Generative Adversarial Network (GAN) (see Goodfellow et al., 2014). We used a generative U-Net, which produces an image of an intended domain after receiving an original image. We also used a discriminative network to distinguish a generated stylized image from a genuine image in the intended domain. While the cost function used to train a discriminative network was used to maximize its ability to separate fake images from genuine ones, the cost function used to train a generative network was used to minimize the separation discerned by the discriminative network.

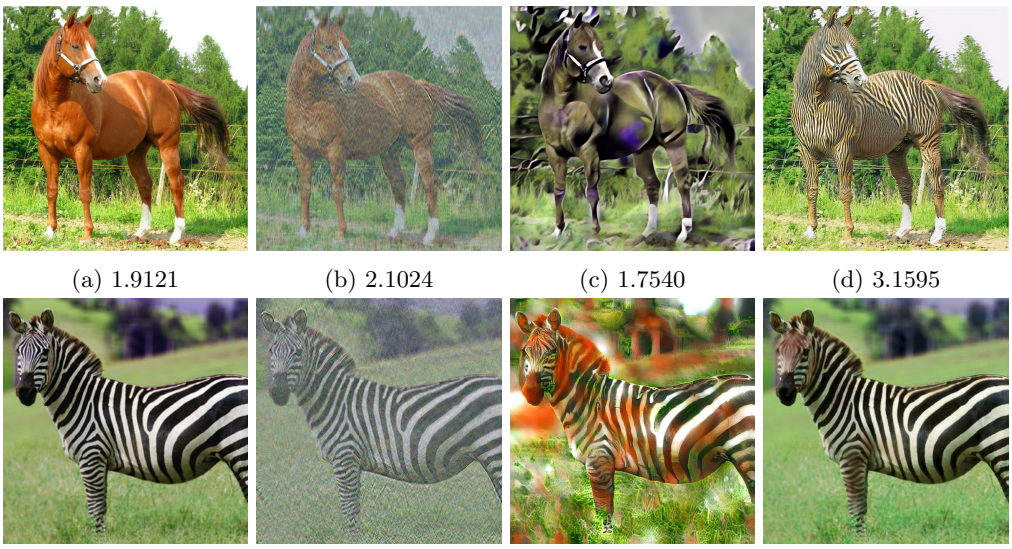
Second, the model strives to maintain cycle consistency. A generative network is always trained with another generative network that performs domain transfer in the opposite direction. With cycle consistency, we should ideally obtain the same image after passing through a forward generative network, followed by a backward generative network. A metric is used to evaluate the difference between the original and the output after a generative network cycle.

Some results from the prototype which help demonstrate the feature of cycle-consistency in the model are shown in Figure 9. Cycle-consistency is crucial for unpaired training of domain transfer model. Reversibility implies the transfigured image contains enough information from the original image. Such feature ensures that the transformed image actually adapts to the target domain from the original image, instead of generating an arbitrary image that suits the target domain.

4.4 Model Evaluations

Owing to the fact that the task of image domain transfer lacks a unique solution and the evaluation of the output quality is highly subjective, it is common practice for research teams in this field to conduct user study to obtain an average score. While we do not have the resources to conduct an experiment of similar scale, we will inspect the quality of the transformation done on the test data of sufficient size.

However, to facilitate possibly the automation of parameter tuning or input condition optimization, we also attempted the use of a trained discriminator, ideally on a different training set, to produce a numerical score which reflects how good the output fits the



(a) 1.9121 (b) 2.1024 (c) 1.7540 (d) 3.1595

(e) 4.2289 (f) 2.5032 (g) 2.5539 (h) 4.7957

Figure 10: Each row shows the score based on the discriminator from the image from original, Baseline 1, Baseline 2 and Prototype respectively. The first row shows results using the discriminator to identify genuine zebra, while the second row shows results using discriminator to identify genuine horse.

intended domain. We conducted an experiment to test the usefulness of this metric (see Figure 10). The score is based on a cost function where a low score means detection of a genuine image in the intended domain. Unfortunately, we observed that the resultant scores from the prototype are exceptionally high (an indicator of fake zebra/ fake horse), meaning that the discriminator is well-trained to identify fake images produced by our generative network, and thus fails to serve as a fair metric for image products generated from entirely different models.

5 Further Discussion and Future Exploration

5.1 Applications of Domain Transfer

5.1.1 Data Augmentation

Although deep neural networks have been performing brilliantly recently, their power depends heavily on the availability of large sets of data which are not always readily available. To solve this problem when image data are used, geometric transformations were typically applied on the image such as cropping, zooming, rotating, etc. to artificially generate more data. Style transfer would provide another way to generate image data. Introducing these variations can help to increase the robustness of the model Jackson et al. (2018).

5.1.2 Bring Pictures to Life

This concept can also be used to transfer painting into a photo realistic image. Such program will help interpolate and add details to the work of the most famous artists like Monet, Cézanne and Da Vinci. This provides supplementary information for historical paintings. Commercially, it may also help to enhance photorealism of near-photorealistic images.

5.1.3 Automatic Visual Effects

In general, cinematic visual effects require a lot of design and efforts. A domain transfer model may be helpful with such a task. For example, to apply aging effect to a movie actor, one can use a model trained between domains of young people and old people. One of the most famous apps that worked on this concept recently is called "FaceApp" where various beautification or aging effect can be applied to portraits.

5.2 Academic Values of Domain Transfer Model

We can learn to develop a better domain transfer model without resorting to just trial-and-error only if they are explainable (see Samek et al., 2017). Zeiler and Fergus (2014) and Yosinski et al. (2015) suggested techniques developed to reconstruct activated neural signals in different layers of a neural network. That way we can find ways to interpret or visualize the features extracted at deep levels in a network and do a better analysis on what modifications can be made to improve the performance of the domain transfer application.

Apart from the commercial values suggested in the previous section, with different choices of domain transfer, different knowledge can also be learned from a trained model. For example, we can gain some insights on fine-grained classification from a model that performs transfiguration between such categories. We can also perform a scientific analysis of the most important characteristic differences in artistic styles of painters from the deep layers of a model that applies style transfer.

5.3 Future Works

5.3.1 Tuning the Hyper-parameters

We are interested in seeing how the results vary when we change the hyper-parameters. It could give us some insights into how to improve the results we are currently able to achieve and analyze how badly-tuned parameters in cost functions could lead to sub-optimal outputs.

5.3.2 Varying the Depth of the Network

Increasing the number of pooling layers and making the model go deeper might improve or worsen the results and we are interested to see which way it goes. In general, people believe that deeper level features can better represent semantic information, rather than just pixel values. One can experiment and find an optimum number of layers to be used in a network.

5.3.3 Varying the Length of the Residual Blocks

We would also like to experiment the same thing we did with pooling layers to the number of residual blocks. Zhu et al. (2017) suggested using longer chain of residual blocks when one trains with images of higher resolution. We would like to see its effect experimentally.

5.3.4 Experimenting with Different Input Conditions

There are a lot more experiments to be performed to see how to best utilize a trained model. For example, as shown in Figure 11, we fed the same network with the same image of different sizes to find out which input size can best utilize a particular model.

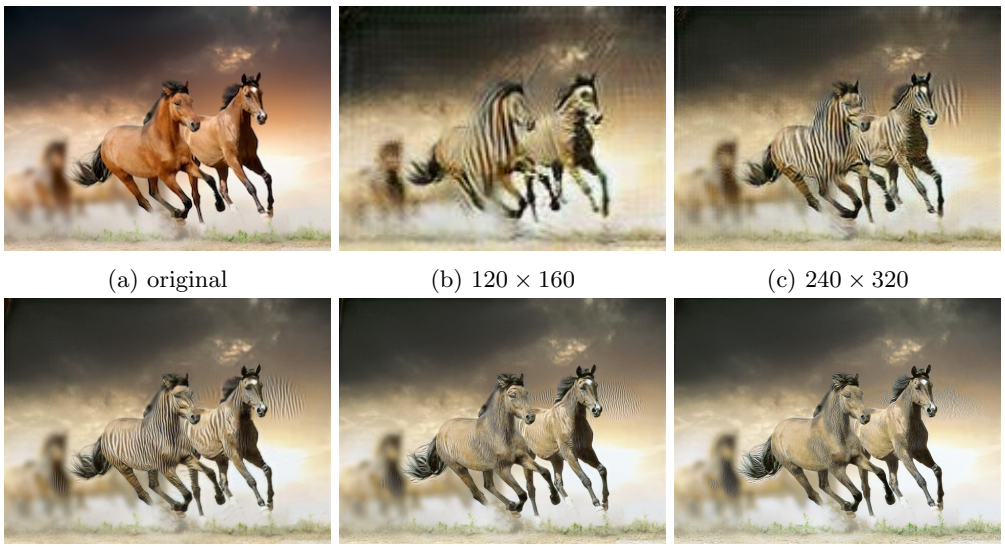
References

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

P. T. Jackson, A. Atapour-Abarghouei, S. Bonner, T. Breckon, and B. Obara. Style augmentation: Data augmentation via style randomization. *arXiv preprint arXiv:1809.05375*, 2018.

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Advances in neural information processing systems*, pages 386–396, 2017.



(a) original (b) 120×160 (c) 240×320

(d) 480×640 (e) 960×1280 (f) 1920×2560

Figure 11: The original image and the outputs received from inputs of various resolution.

Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz. A closed-form solution to photo-realistic image stylization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 453–468, 2018.

A. Ruano. Pytorch-cyclegan, 2017. URL `https://github.com/aitorzip/PyTorch-CycleGAN`.

R. A. Sadek. Svd based image processing applications: state of the art, contributions and research challenges. *arXiv preprint arXiv:1211.7102*, 2012.

W. Samek, T. Wiegand, and K.-R. Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9(Nov)*, pages 2579–2605, 2008.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

J.-Y. Zhu. Cyclegan and pix2pix in pytorch, 2017. URL `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.