Deep learning – Final Course Project

Submitted as final project report for the DL course, BIU, 2023

Introduction

During the semester we have learnt about generative models, and their abilities to generate a new instance from a defined distribution. In this project, we will try to define the distribution of Claude Monet paintings, by creating a generator that transforms instances from the distribution of real live photos to Monet, which can be defined as applying the style of Monet to a photo.

For this task, we were asked to pick only 30 Monet paintings, and by them to define a Monet Generator for photos, which can change the style of the image to those of Monet.

Related Works

Some of the works that discuss about style generative models are the GAN models, especially Cycle-Gan, which learn 2 domains, their generators, and their discriminators by unpaired images.

Another approach to this problem is the neural style transfer, which uses a pre trained image classifier and then improve the content image according to a single style image.

Solution

General approach

For selecting wisely 30 paintings of Monet from a dataset of 300 paintings, we tried to find the most "centered" instances in related to a distribution that a convolution neuron network might find for this data. So, we built an image-reconstruction auto-encoder, and then we picked the most closed images to the mean of data, represented by an embedding layer of the auto encoder. We could find a linear representation in PCA, or traditional image processing techniques like color histograms, but we choose auto encoder, since it come from the same concept.

For the Cycle-Gan, we adopted the same architecture as we learned, with minor adjustments for the task, like the size of the output, and respective field of the patch-Gan discriminator.

We tried also generating paintings using neural style transfer, using the pre trained VGG-19 image classifier.

Design

For picking the 30 most suitable paintings from the 300, we build a simple image reconstruction auto encoder, trained by the Monet dataset. The auto encoder has 16 layers, when the 8th layer is the an embedding layer, creating a vector with size of 1024 ,(the input and output are 256*256). We trained the auto encoder with binary cross entropy loss for 30 epochs with Adam optimizer, which took about 2 minutes.

For the Cycle Gan, we used the UNET architecture for the generators, with minor changes for input and output shape of 320*320, and the patch-Gan architecture for the discriminators with respective field of 40*40. We defined the losses for the generator outputs (identity loss and cycle-loss) using mean absolute error, and for the discriminator we used binary cross entropy loss. We used Adam optimizer as well in all the models. During our experiments we change some of the hyperparameter of the architecture, such as trying another activation functions for the final layer of each model and the dropout rate for the relevant layers.

Neural style transfer – as we mentioned, we used the pre trained VGG19 classifier as was trained on the image-net dataset. We defined the style loss by the gram matrix defined from middle outputs of the 5 first layers, one output from each layer. For the content loss, we use a single output from the 6th layer. We used the SGD optimizer with scheduled decay, and run it for 150 epochs.

Since neural style transfer demands just one style image, we tried two approaches:

- Picking the most suitable style image from the 30 paintings, by the comparing the chi-square of the histograms for the style and the content image, and then trained by it.
- Run the model with the 30 paintings together, when an epoch contains 30 steps, each time the content image is changed by another style image.

For the whole code we used the libraries of TensorFlow and Keras for creating and training the model, matplotlib for showing results, and the Pillow library for image preprocessing. We use the interactive platforms of Google colab and Kaggle, using their GPU accelerators. Training this models took a long time, related to interactive code, such as 20 minutes to an hour. This challenge makes us prefer a small number of epochs, for finding the best generator result in the shortest time we can get.

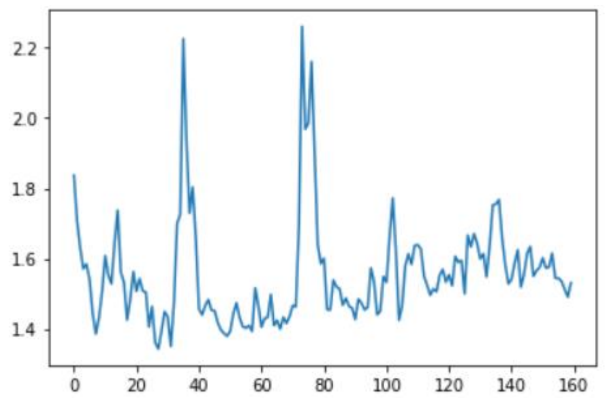
Experimental results

Provide information about your experimental settings. What alternatives did you measure? Make sure this part is clear to understand, provide as much details as possible. Provide results with tables and figures.

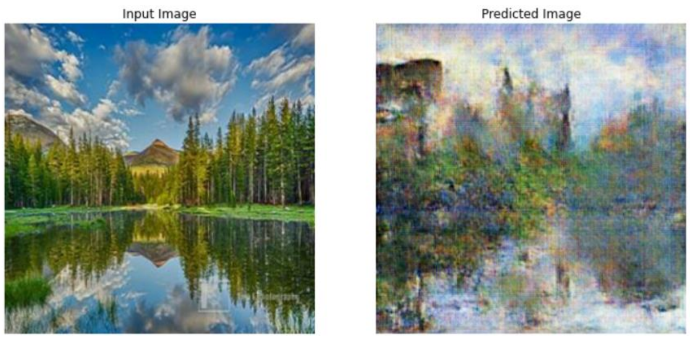
As we say, we tried different hyperparameters for the Cycle-Gan. We used the *tanh* in the final layer of the generator models and *LeakyRelu* instead the common *Relu* in middle layers and the final layer of the discriminator model.

We tried to play with the epochs number, the λ (to give different weight to the identity loss and to cycle loss), and the dropout rate. All of them are hyperparameters we decide at the beginning of each experiment.

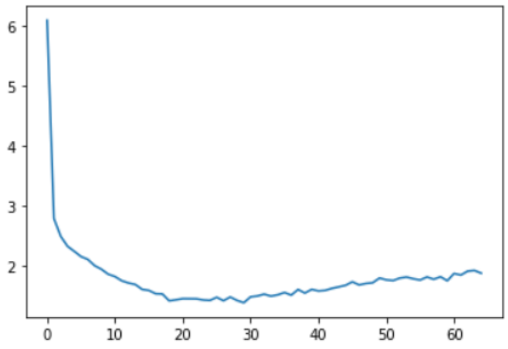
For the first training, we define 160 epochs, dropout rate of 0.6 and $\lambda = 8$. The loss was:



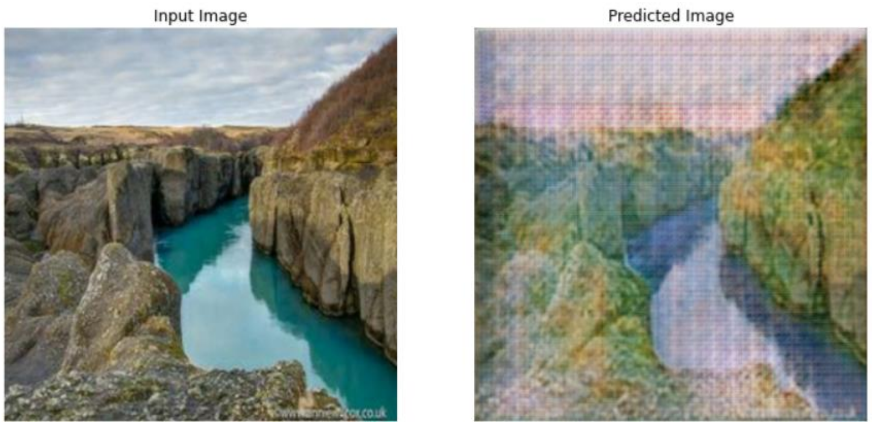
We found the loss diagram too noisy. The generative results were accordingly:

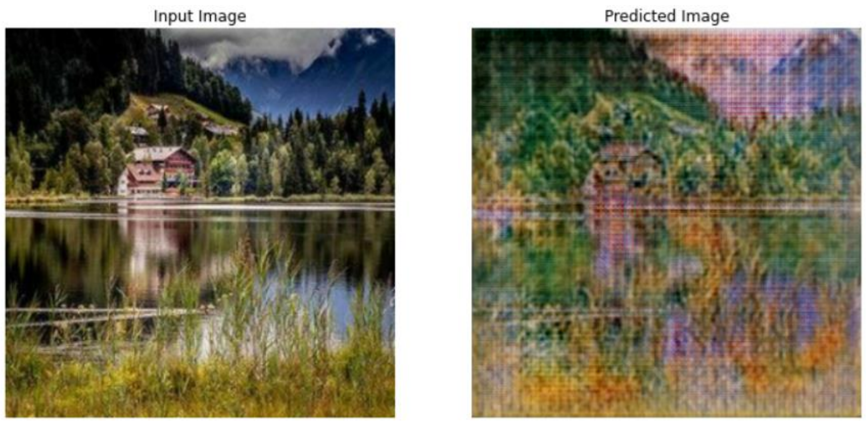


So, then we applied some changes, A cycle loss of CYCLE_GAN with $\lambda = 10$, 65 epochs and droptout rate of 0.65.

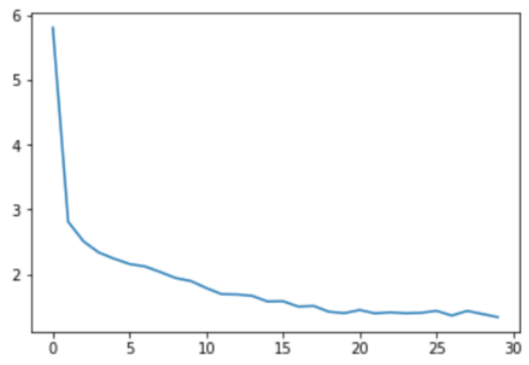


Some of the results for this model are:

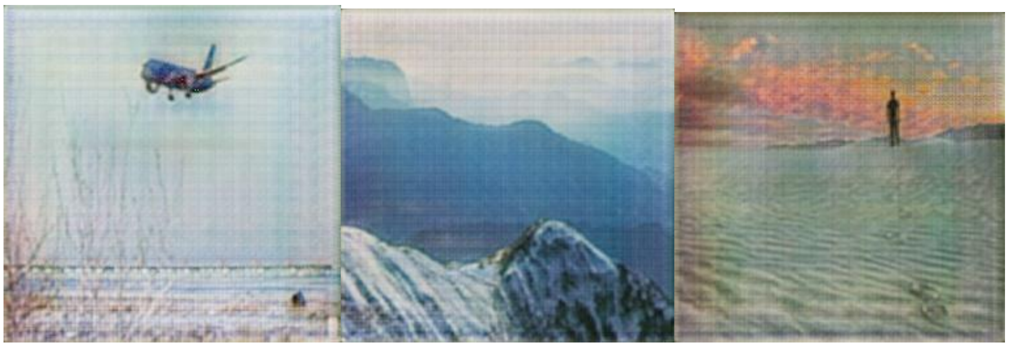




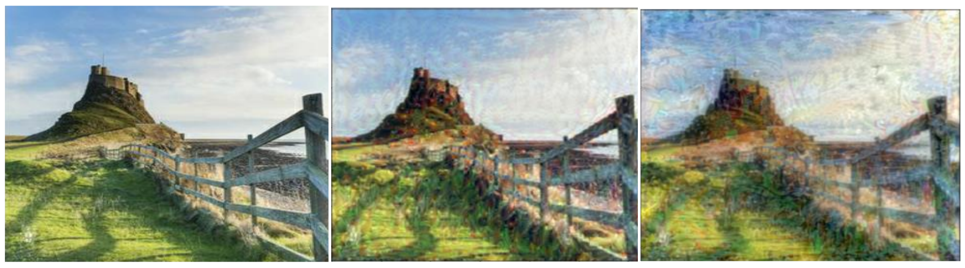
Then we tried another training, with 30 epochs, dropout rate of 0.65, and $\lambda = 10$. The cycle loss is:



The loss is seen converging well, some of the visual results are:

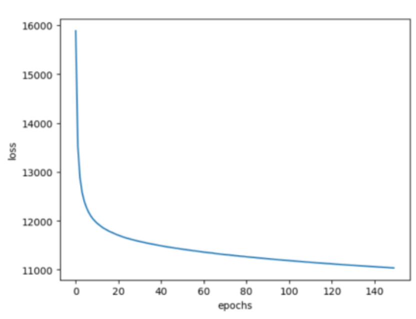
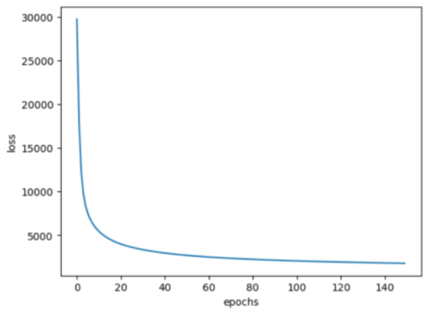


For neural style transfer, we train the content image in the first time by only single style image for 150 epochs, and then train the content image by the 30 images styles. The result:



From left to the right: the original image, single style transfer, 30 styles tranfer.

Here the loss function for the two approaches:

Loss for 30 styles combined, Loss for single style.

Discussion

First and foremost, we should say that performance evaluation of each method might be challenging. Since we don't have metrics that can define the artistic qualities of our results (and maybe those metrics are not existed in the real world), the group of Monet paintings is defined by simple question: did Monet paint this image or not.

As a team, we discussed what are the properties we look for in the results – the color palette, brush texture, composition, and even the clarity of the content. In the Kaggle competition the results are measured by Fréchet inception distance, assuming the real domain and the generative domain are maid by multi gaussian distribution, which might be mistaken in the real world.

We saw that in the neural style transfer methods the content is still very clear, maybe too much, while the brush textures are generated well. However, the composition has not been changed. Maybe if we selected more inner layers output as the style loss, the composition would change.

Training the Cycle-Gan was a challenge. Since we used patch GAN discriminator, we could not use it results to evaluate its performance well like a simple binary classifier model. So, we tried to focus in hyperparameters that relevant to all of model's parts.

Though the Cycle-Gan did not change significantly the texture of the photos, it change their color palette, and in some versions it tried to adapt the composition of the photo. We saw that the model can be overfitted easily – for some photos the output was almost perfectly one of the more abstract paintings in our data set. Hence, we tried to fine-tune the number of the epochs, knowing that the model can not be converge to a hypothetic good generator, and as we noted above, maybe this generator isn't exists at all.

Code

Selecting the 30 Monet paintings:

https://colab.research.google.com/drive/13GZNzn-7tec8wx5ZJAuykUYq3F53knfj?usp=sharing

Cycle-GAN training:

https://colab.research.google.com/drive/1h_OyMAdnhy_Y4Vw7ddr2BiLmjimQGq7x?usp=sharing

Cycle Gan Testing:

https://colab.research.google.com/drive/1P96xD6qskegfEicixUx-ImgX4pf-jRab?usp=sharing

Neural Style Transfer testing:

https://colab.research.google.com/drive/1Or7ywdIEvm6GGZoR5I99Kvb2BMakTbpP?usp=sharing