ELEC 475

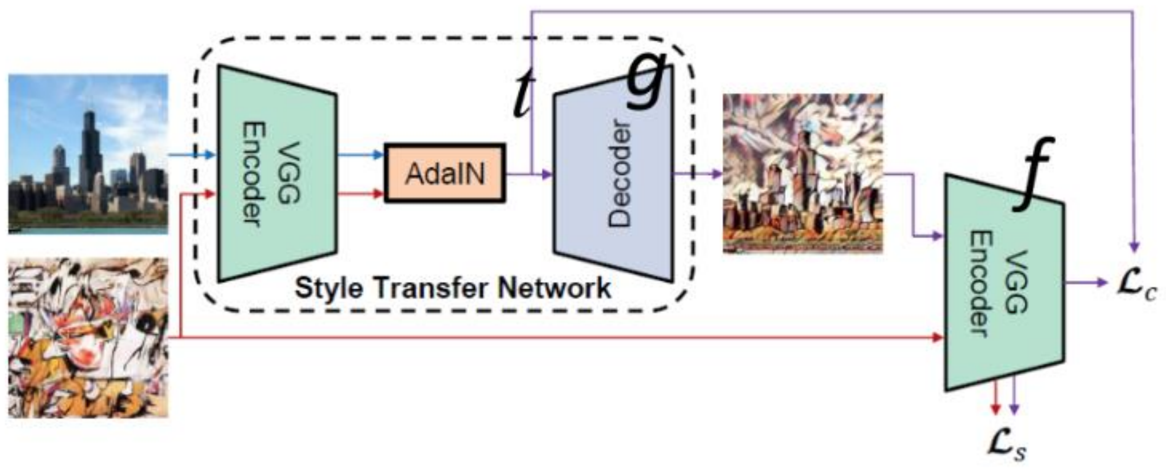
Lab2: Neural Style Transfer with AdaIN

Setareh Soltanieh

20370820

Introduction

In this lab we are focusing on style transfer with AdaIN network. Here we have some content images and style images. Our goal is to implement the following AdaIN network.



Where our encoder is a VGG network with pre-calculated weights and we are training the decoder with optimizing the calculated losses.

Hardware

We have used a GPU server that is: “Quadro RTX 6000”

Hyper Parameters

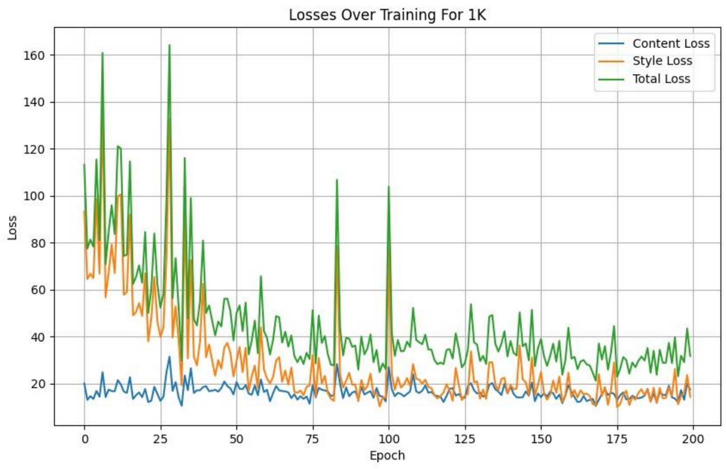
Learning rate = 1e-4 (we are adjusting the learning rate with a function inspired from the source code of AdaIN network) based on this function, the learning rate is slightly decreasing on each epoch

Batch_size = 20

Optimizer = Adam

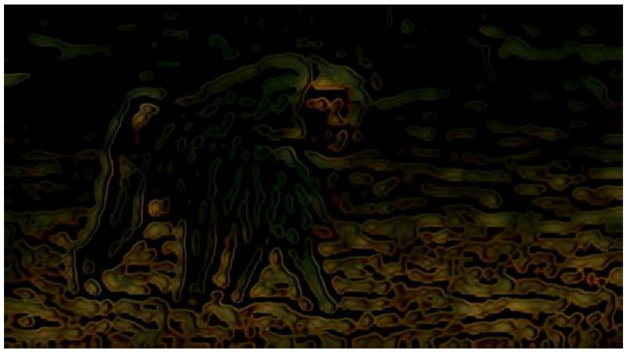
Training for 1K Dataset

At first, we have trained the network for 200 epochs for the COCO1K dataset and received the following losses.

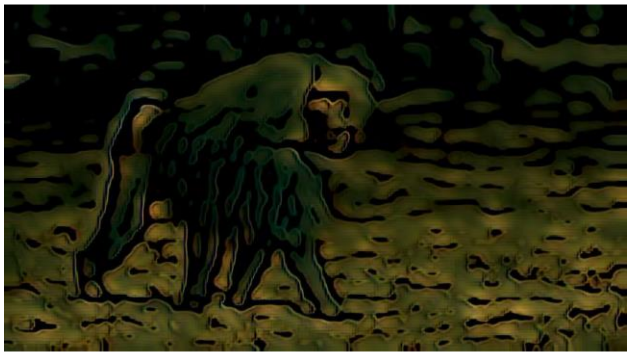


Based on this loss we can understand the model has been trained well and all the losses are decreasing. We have saved this model's weights to our folder "model_1k" for each time interval of 10 epochs. The total time for this training took 14 minutes. We have tested our final weights for testing and the results for the 'baboon.jpg' was as follows: (our alpha was the same for all of them and it was 0.9)

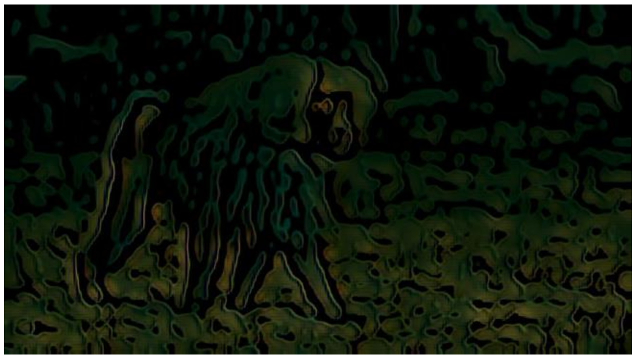
1. style_0a585acb9d7134c0b39656a588527385c



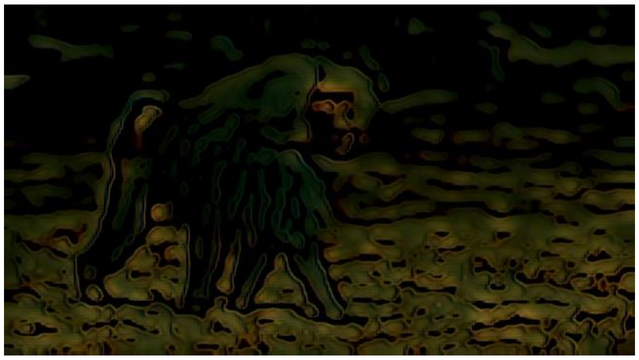
2. Andy_Warhol_97



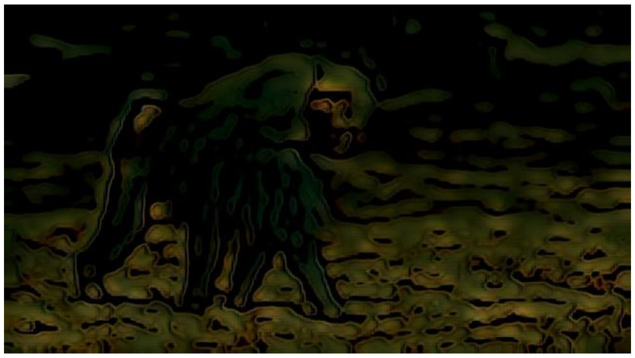
3. Brushstrokes



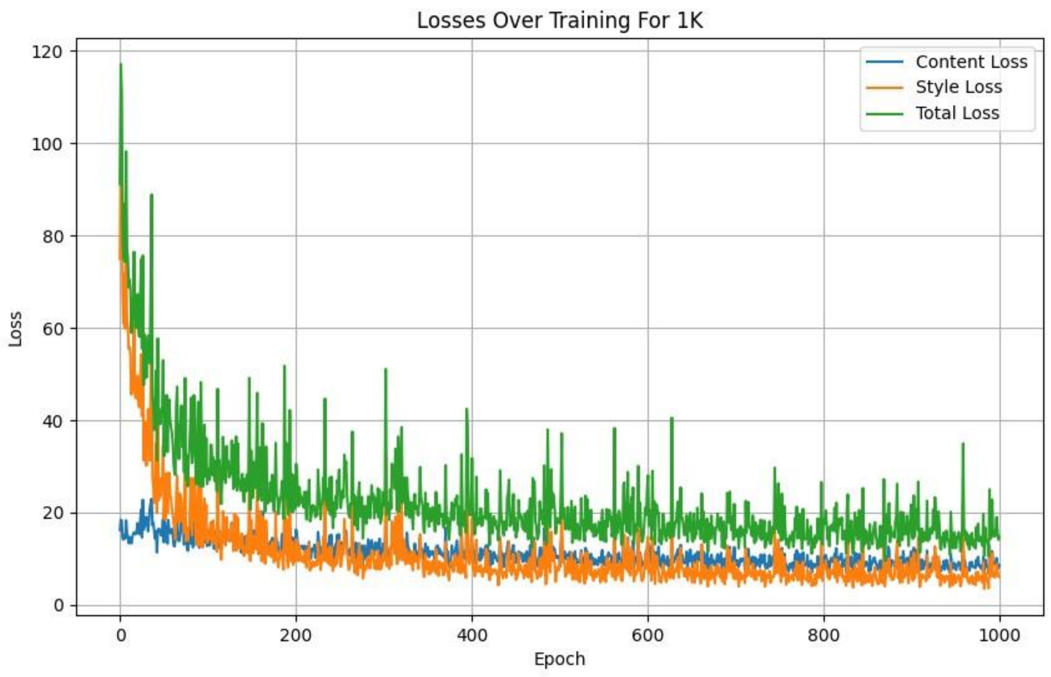
4. chagall_marc_1



5. the-persistence-of-memory-1931

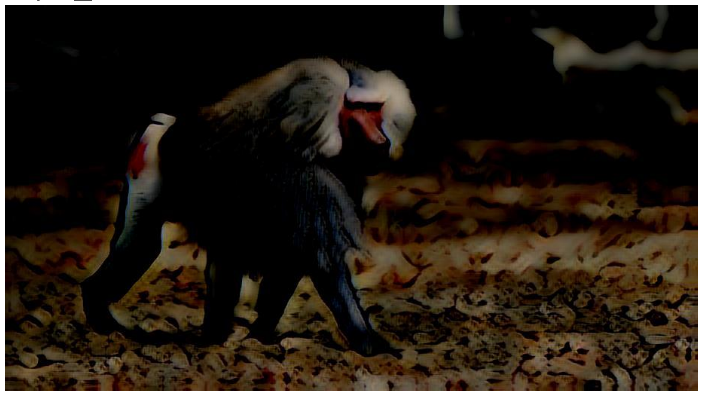


Here we can see that the style of this content is changed but this is not perfect so we have decided to continue our training for a longer epoch time like 1000 epochs, and here are our results (alpha=0.9). This training took 53 minutes. And the loss curve is as follows:

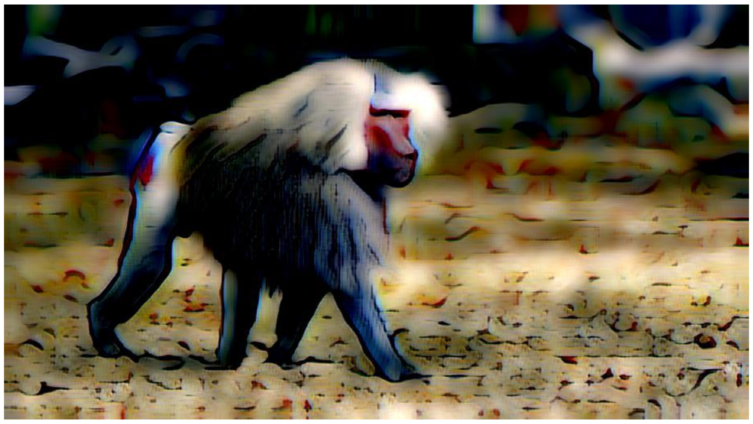


We have saved this model's weights to our folder "model_1k" for each time interval of 10 epochs. We have tested our final weights for testing and the results for the 'baboon.jpg' was as follows: (our alpha was the same for all of them and it was 0.9)

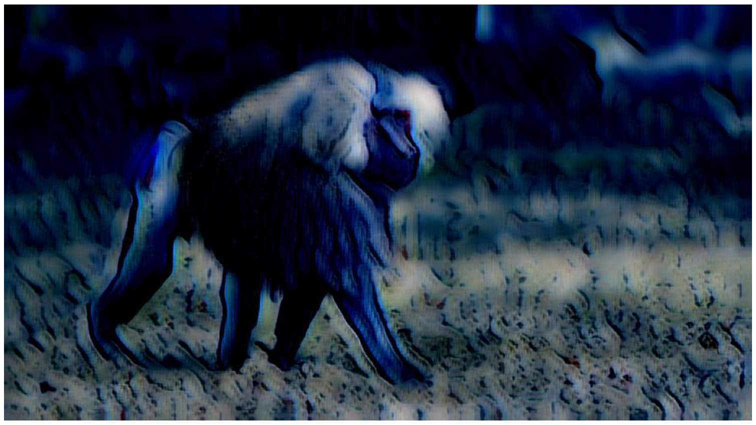
1. style_0a585acb9d7134c0b39656a588527385c



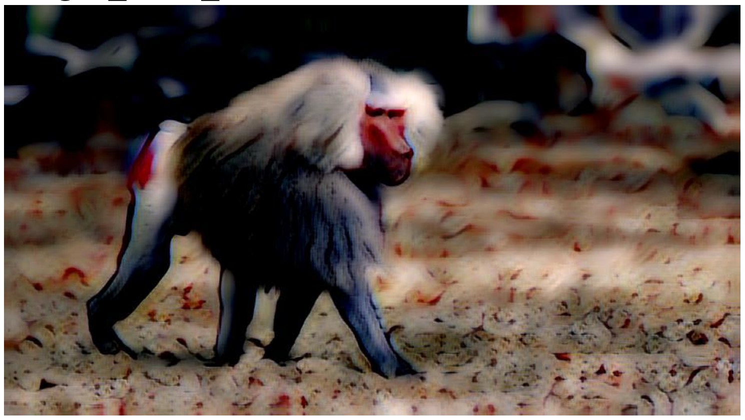
2. Andy_Warhol_97



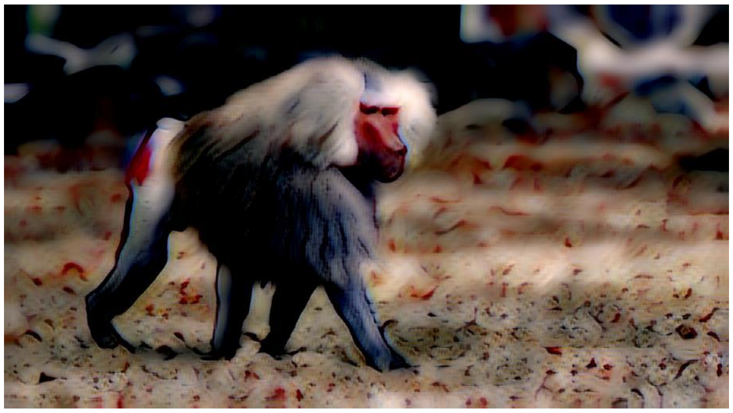
3. Brushstrokes



4. chagall_marc_1



5. the-persistence-of-memory-1931



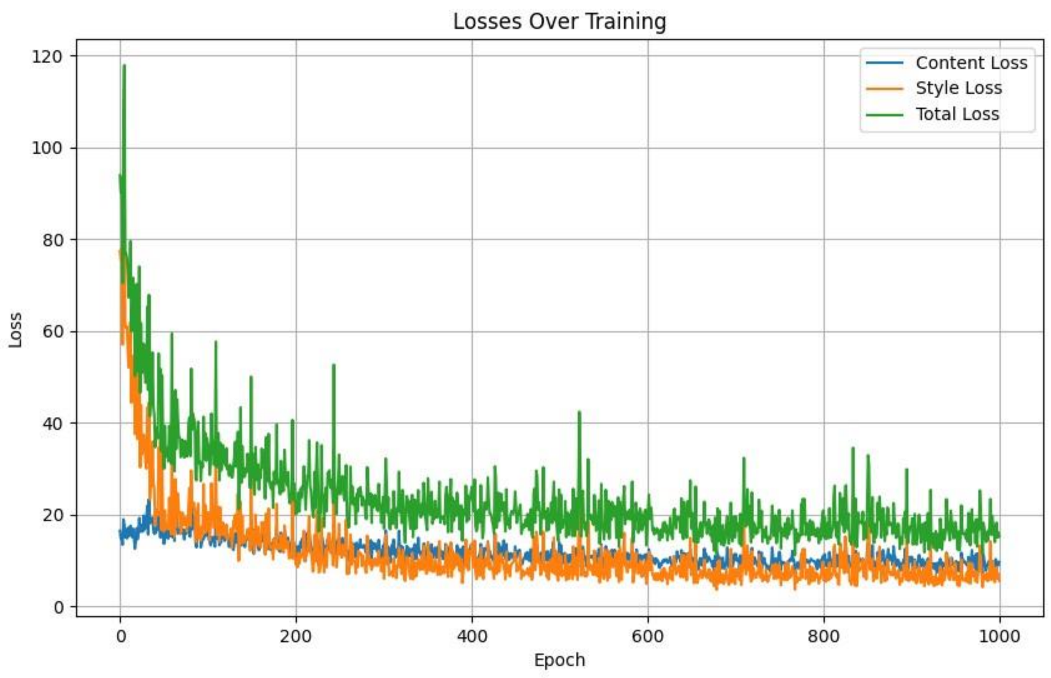
Based the previous results we can see that each content image has learnt the style of its style image.

Implementation Details for 1K Dataset

We have a ‘my_train_1k.py’ file that is the main python file for training. We run our models with the bash file ‘train_1k.sh’.

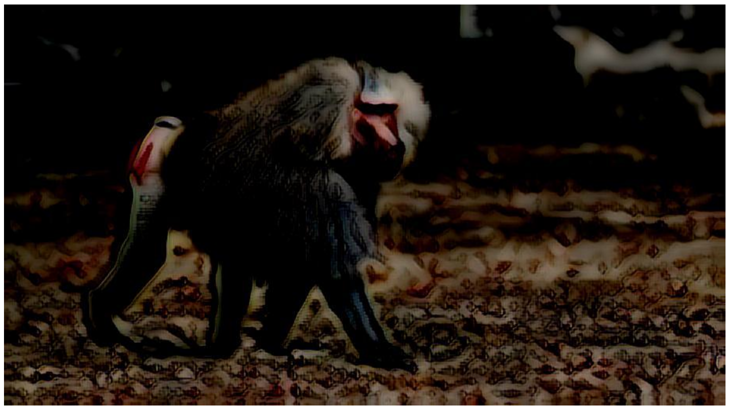
Training for 10K Dataset

We have trained this network for 1000 epochs for the COCO10K dataset and received the following losses.

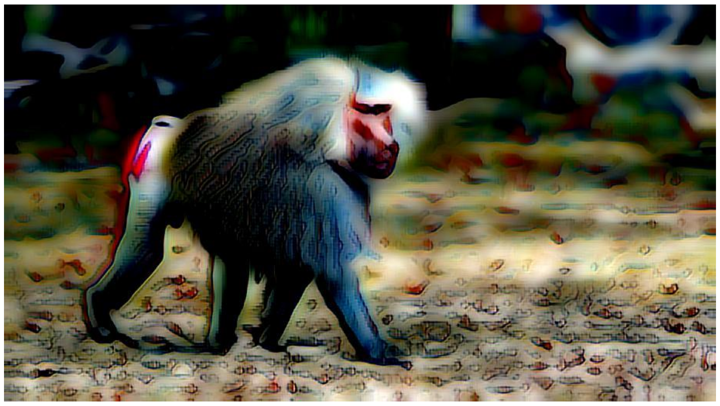


Based on this loss we can understand that our network has been trained well and the errors are low this training took 53 minutes. We have saved this model's weights to our folder "model_10k" for each time interval of 10 epochs. We have tested our final weights for testing and the results for the 'baboon.jpg' was as follows: (our alpha was the same for all of them and it was 0.9)

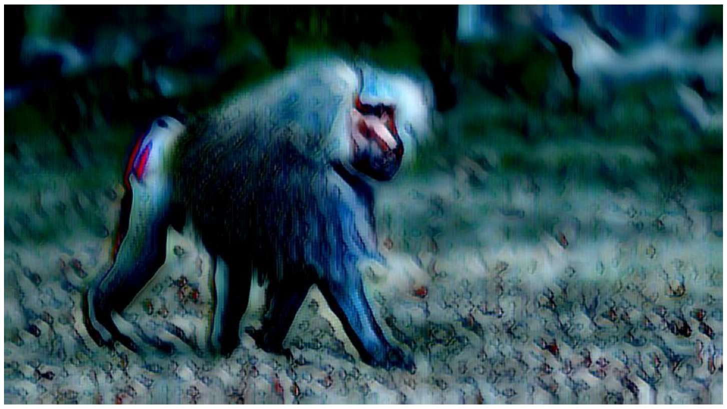
1.style_0a585acb9d7134c0b39656a588527385c



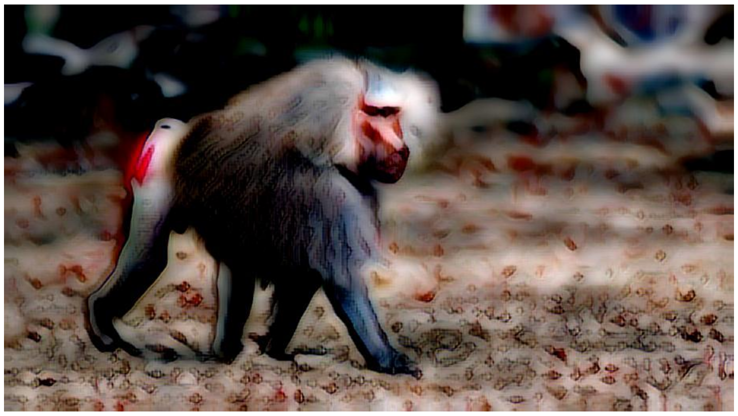
2.Andy_Warhol_97



3.Brushstrokes



4.chagall_marc_1



5.the-persistence-of-memory-1931

Based the previous results we can see that each content image has learnt the style of its style image.

Implementation Details for 10K Dataset

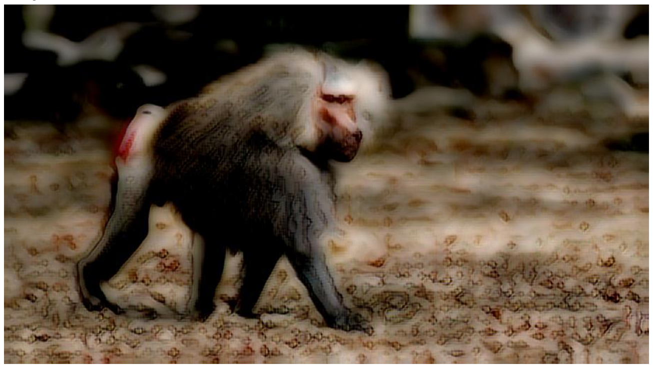
We have a 'my_train_10k.py' file that is the main python file for training. We run our models with the bash file 'train_10k.sh'.

Different Alpha Variable for Baboon.jpg

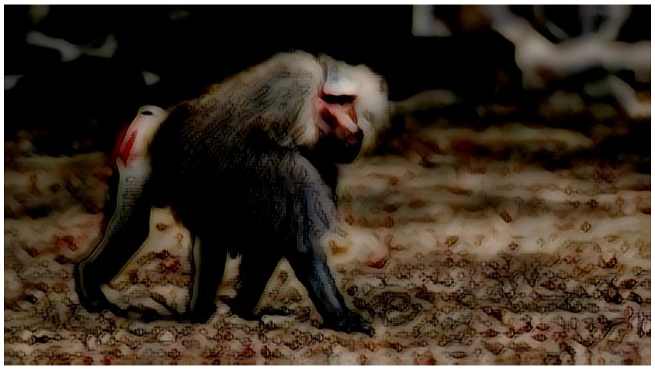
Here we wanted to see how alpha variable can effect our results in the test. So we set different alphas: 0.1, 0.5, 0.9

1.style_0a585acb9d7134c0b39656a588527385c

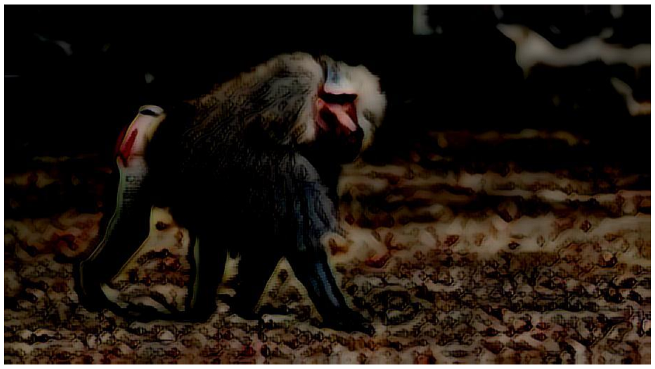
a. alpha = 0.1



b. alpha = 0.5

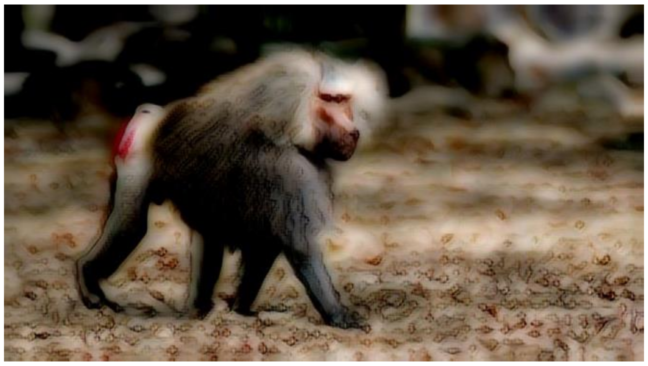


c. alpha = 0.9

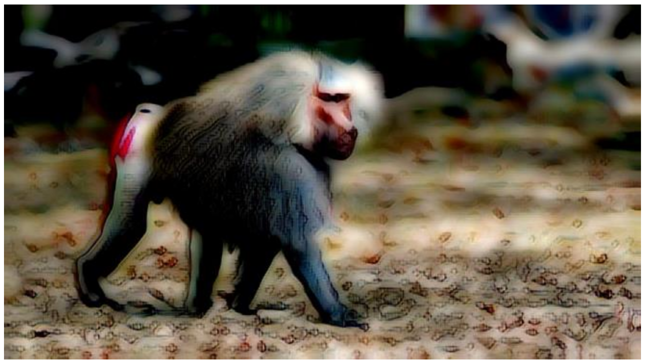


2. Andy_Warhol_97

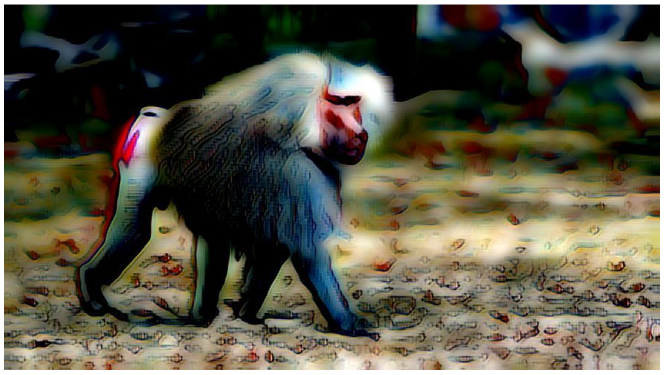
   a. alpha = 0.1



   b. alpha = 0.5

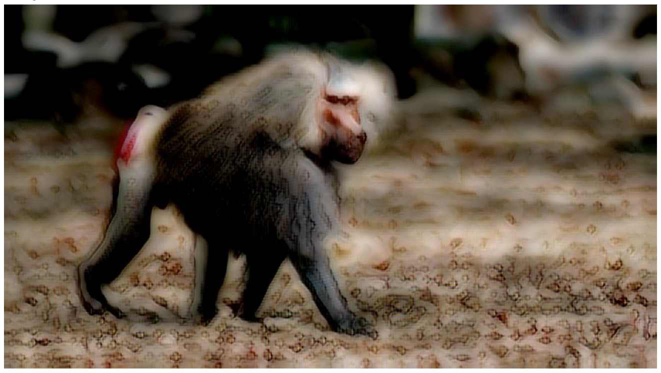


c. alpha = 0.9

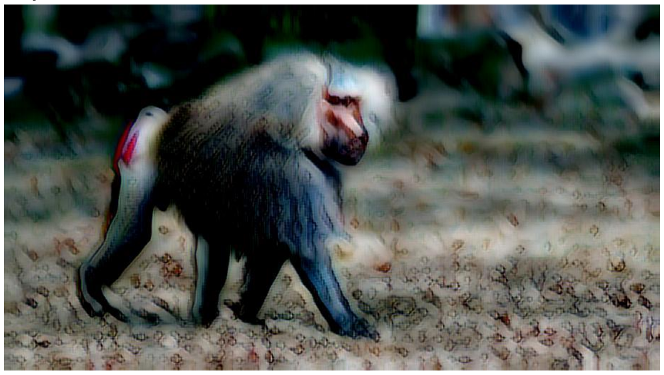


3. Brushstrokes

a. alpha = 0.1



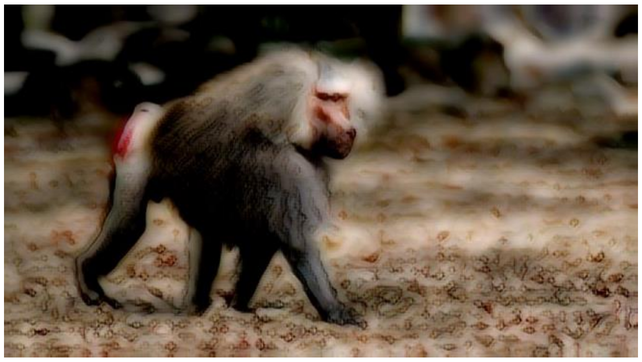
b. alpha = 0.5



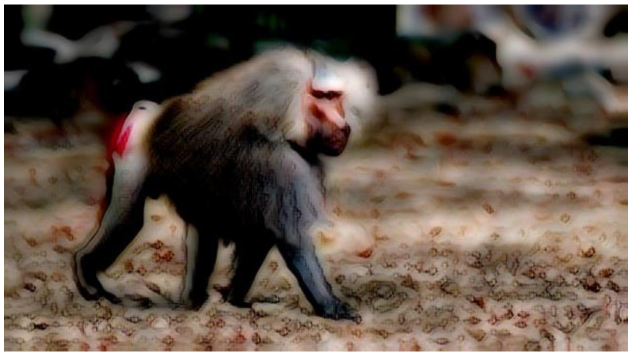
c. alpha = 0.9

4.chagall_marc_1

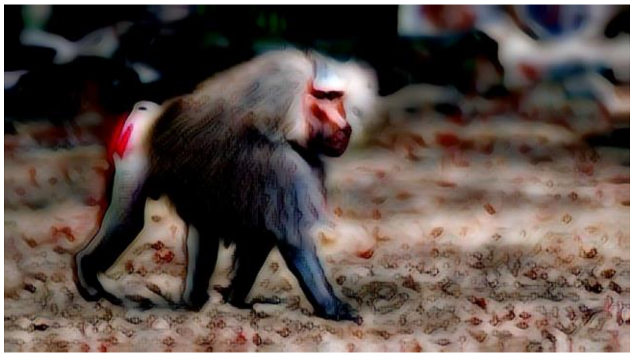
a. alpha = 0.1



b. alpha = 0.5

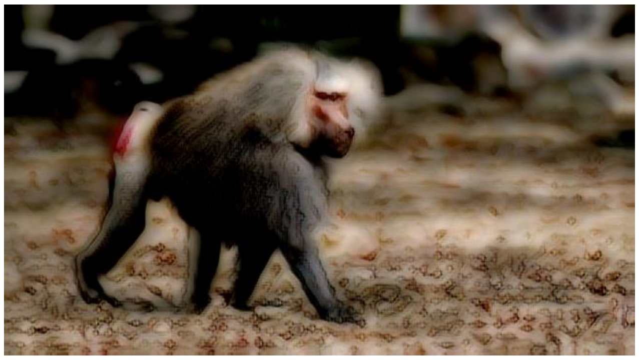


c. alpha = 0.9

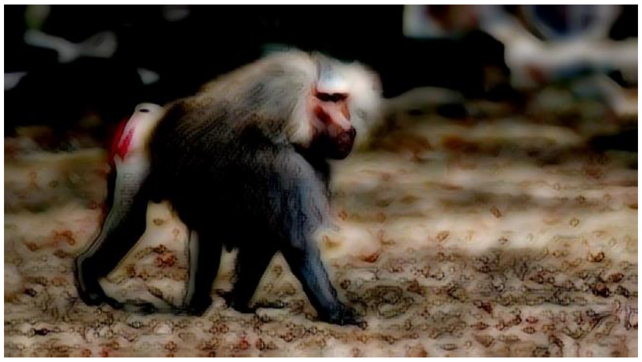


5.the-persistence-of-memory-1931

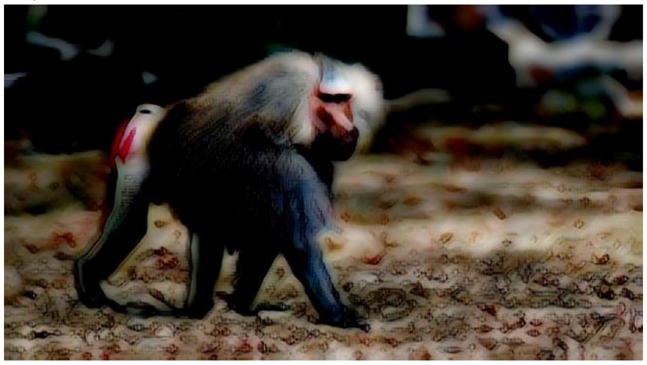
a. alpha = 0.1



b. alpha = 0.5



c. alpha = 0.9



Based the previous plots we can see how different alpha values can effect the results. In the alpha=0.9 we almost don't see any styles from the original content image. And it is also recognized that in alpha=0.1, the different styles look kind of the same and the style didn't have that much effect.

Gamma Value

Gamma value is the value that indicates the style loss's importance in the training process. We are calculating our total loss based on the following equation:

$$Loss = Loss_c + gamma * \ Loss_s$$

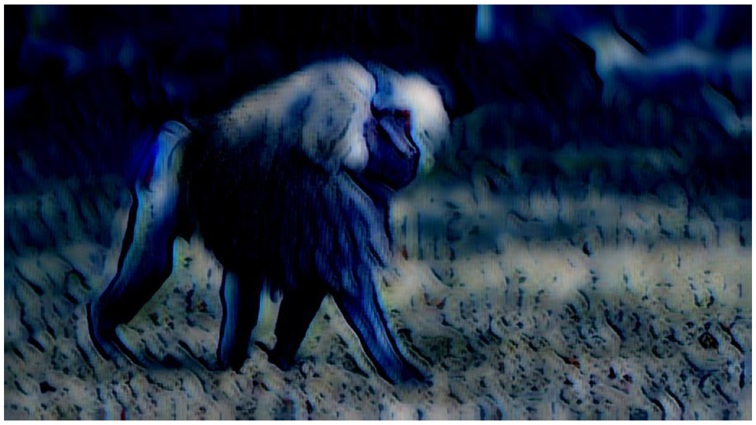
We have set this variable to 1 for both training files because we wanted our model to learn the style of the style images very well and lose the style of its own content.

Loss Curves Differences for 1K and 10K Datasets

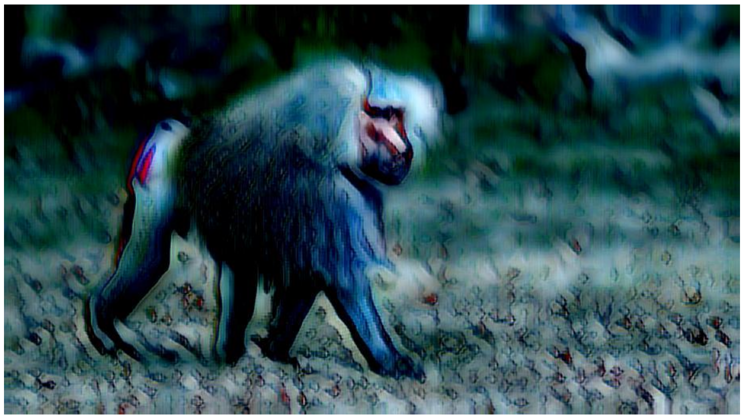
Based on the loss's curves, we can observe that the loss for the 10k images decreases faster than the 1K images. Also, it can be recognized that the loss curve for the 1K images, has more jumps which is caused by the number of images in that dataset. The more images we have in a dataset, these jumps are going to be lower.

Qualitative Differences for 1K and 10K Datasets

To understand the differences between the performance of the 2 different training results we are comparing the baboon.jpd style transfer with similar style (Brushstrokes) together. The following image is the result with the 1k dataset



And the following image is the result with the 10k dataset



Based these 2 images we can see how the style is accurate in the 10k dataset. In the first image, we can see some dark parts in the upper left side of the image which is fixed in the second image. Also, we can see more details of image in the second image than the first image. Also, the colors in the second image is more similar to the original style image of Brushstrokes.

TA Implementation of My Codes

I have provided the both 'my_train_1k.py' and 'my_train_10k.py' which are similar to each other but since I wanted to run them in the same time I have written it twice. Which one that you prefer, you can use.

I am saving the model's weights after each 20 epochs and saving them to 'model_1k' and 'model_10k' folders.

I have shared the saved decoder's weights for both 1k and 10k dataset.

Thank you!