ELEC 475 Lab 2
Neural Style Transfer with AdaIN

Noah Serhan

Description of the Hardware Used

I used a PC with a GTX 1080 GPU for this project. The GTX 1080 has 8GB of VRAM and was suitable for the 1K and 100 image training stages. However, for the 10K dataset training, the 8GB VRAM was insufficient to complete without issue, due to the larger dataset's size and complexity. So, the process was restarted using an AMD Ryzen 5 3600 CPU, at which point the training speeds were greatly reduced. A GPU with at least 12-16GB of VRAM would have been ideal to handle the 10K dataset training without issues.

Hyper-Parameters Used to Train the System

Optimization Method

- The optimization method used for this training is “Adam”, which is an adaptive learning rate optimization algorithm with efficient convergence properties that combines the advantages of both the AdaGrad and RMSProp gradient decent methods.
- The optimizer is initialized with a learning rate of 0.0005.

Optimization Hyperparameters

- The initial learning rate is set to 0.0005. This hyperparameter controls the step size during wight updates and is a crucial factor in determining the training convergence and stability.

Learning Rate Scheduling

- A learning rate scheduler is used in this training to adjust the learning rate during training based on the model’s performance. The specific scheduler employed is ReduceLROnPlateau.
- ReduceLROnPlateau monitors the training loss and dynamically reduces the learning rate when the loss plateaus. It operates in “min” mode, meaning it reduces the learning rate when the loss stops decreasing.

Loss Function

- The Mean Squared Error (MSE) loss function is used when computing the loss, performing backpropagation to update the model’s parameters.

Other Relevant Details

- The device used for training is determined dynamically. If a GPU is available (cuda), then it is prioritized as it is much more optimal for training models; otherwise, it runs on the CPU.
- Training loss values for each epoch are printed, and the training loss curve is saved as a plot.

- The training batch size is set to 20 for the 100-image dataset training, 32 for the 1K image dataset training, and 44 for the 10K image dataset training.

Existing AdaIN Implementations

My implementation drew inspiration from the AdaIN paper by Huang and Belongie that was provided.

I adopted some of the code from the GitHub that was provided for my AdaIN_net.py file, which was very useful in my design stage. My train.py was built from scratch, but I used another repository was consulted for directional guidance. Finally, my train.txt was based on what was provided in the lab handout but edited to fit my project.

Training Time

For the 100-image dataset, the training took 8 minutes of training. The 1k image dataset took 3 hours of training. The 10k image dataset, which is significantly larger, would have taken around 63 hours, but I did not have enough time to complete this training.

Gamma Value

I used a gamma value of 1.0 for training. The gamma factor determines the importance of style transfer relative to content preservation. My value of 1.0 signifies an equal emphasis on both content and style in the stylized image, which is why this value was used.

Loss Curves

To preface, unforeseen challenges arose, leading to time constraints that prevented us from completing the full 20 epochs of training for the 10K dataset. Training it in its entirety would have taken well over 2 days on the hardware available to the team. Descriptions of the loss plots will still be given as if they were completed in their entirety based upon how they theoretically should look like.

Both models' loss plots start high and quickly begin reducing until they converge around or slightly before 20 epochs.

The 10K image dataset allows the model to converge faster than the 1K dataset. More data helps the model learn better patterns, and as a result, it converges to a lower training loss faster. In fact, the initial training losses from the 100, 1K and 10K image datasets all approximately decrease by a factor of 10 or so per jump between dataset sizes.

The 1K loss plot shows signs of overfitting, as can be seen in Figure 2, where the style and content loss curves both spike quite abruptly around both the 6th and the 14th epochs. Since there was less data for the model to be trained on, the model learned to memorize the training data, rather than generalize from it. The 10K loss plot would show either very little or even no signs of overfitting as the model would learn to generalize better with more data.

The 10K dataset also contains more diverse examples of different environments, objects, subjects, lighting conditions, and more that all help the model learn, hence why the losses are, and would converge much lower than for the 1K dataset.

In summary, the main reason for the differences between the two loss plots is the quantity and diversity of the data. More data, and, consequently, more diverse data leads to a more efficient and stable learning process.

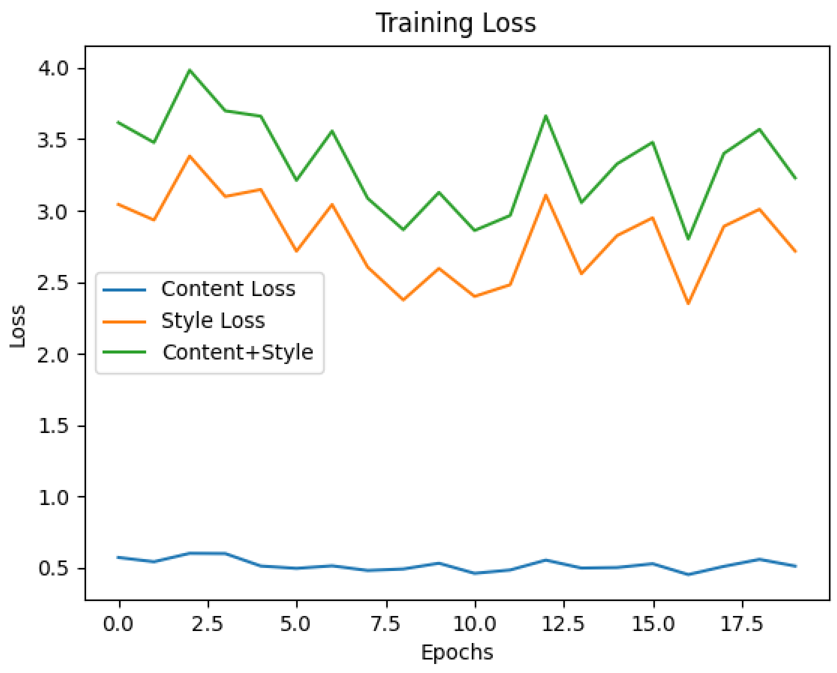

Figure 1: Loss plot for the training using the 100 image datasets.

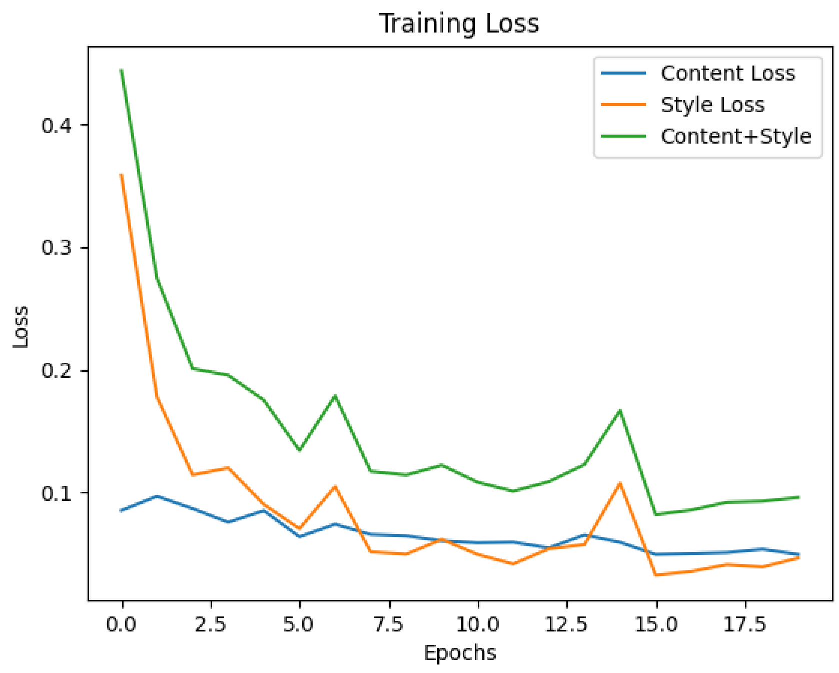

Figure 2: Loss plot for the training using the 1000 image datasets.

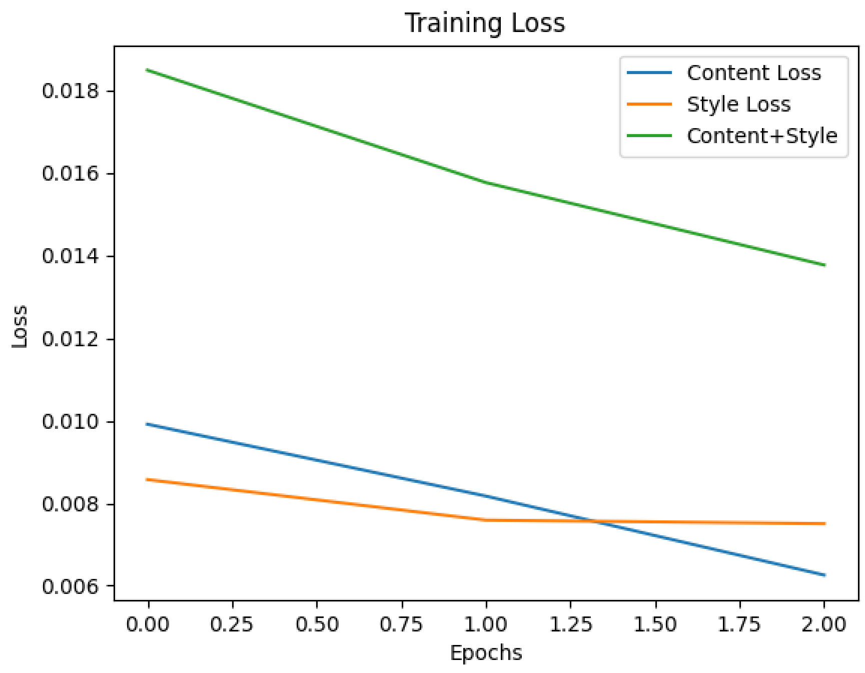

Figure 3: Loss plot for the training using the 10000 image datasets.

Qualitative Differences in Output

There are many qualitative differences in the output between the "1K" and "10K" tests that are to be noted. The "10K" dataset, being far more extensive and diverse, results in stylized images that capture a broader range of artistic styles. This dataset allows for more intricate style transfer. On the other hand, the "1K" dataset has limitations in representing various artistic styles, leading to differences in the quality and diversity of stylized output.

Although the 10K image was produced from 3 epochs, we can see below the difference it has from the 1K; the quality of the image is much greater, and the style of the image is more pronounced. The finer details in the baboon's face, for instance, are much more noticeable in the outputs produced using the 10K image dataset than they are in the 1K image dataset outputs. The style transfer does not alter the content of the image as much as it seems to add a stylistic layer on top of what is provided to the model, which enables the model to produce a much mor realistic output.

With the addition of more epochs on which the 10K image dataset can train, this difference to be even more drastic than it already is.

Figure 4: Example of the 100-image dataset model's output.

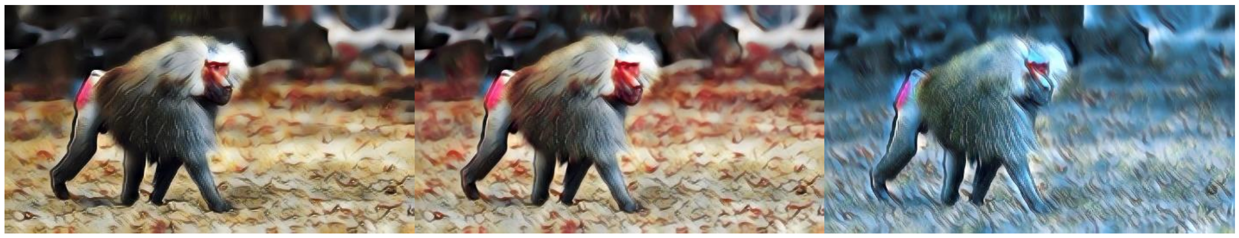

Figure 5: Example of the 1000-image dataset model's output.

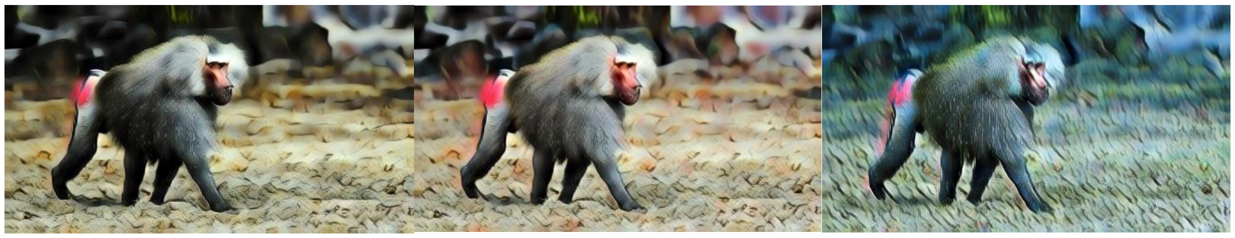

Figure 6: Example of the 10000-image dataset model's output.