ELEC 475 Lab 2: Neural Style Transfer with AdaIN

Jordan Hunter – 20178182,

Nicholas Krsikapa – 20102039

# Training Details

The training of our 1k dataset started on our own computers using our laptops and desktop computers to test different hyperparameters. On average the 1k dataset took about 8 hours running to complete 20 epochs. After settling on a set of hyperparameters, we felt ready to test against the 10k dataset. For the 10k dataset, we used Google Colab to do the training, the entire training took about 6 hours running at different times of the evening when we could get access to the faster gpus. Our code was setup in such a way that we were able to stop save the model at each epoch and restart the training at those points. After further tweaking with the 10k results, the hyper parameters used for both the 10k and 1k dataset are as follows:

* Gamma – 1.0
* Initial learning rate – 0.0005
* Optimizer – Adam
  + Weight\_Decay – 0.00001
* Scheduler – ReduceLROnPlateau
  + Factor – 0.1
  + Min\_lr – 0.0001
* Batch size – 20

When testing different hyperparameters using the 1k dataset, gamma was found to affect the output too much whenever the value was greater than 1. With a value of 0.5 produced images barely effected by a style image. We settled with a gamma value of 1.0 and found kept the content image details while adding in significant changes based on the style images.

For our AdaIN model implementation, we used both the paper and github provided in the Lab 2 outline pdf as resources. Mainly to understand how the losses are to be calculated and the difference in the inference vs training portion.

# Results

## Are there differences in the loss curves for training the “1K” vs. “10K” datasets? If so, describe these differences, and explain what accounted for them.

The total loss curve of the “10K” dataset converges at a faster rate and at a lower loss value than that of the “1K” dataset. The reason this occurs is because during each epoch, the 10K decoder is learning features from 10K images, meaning that it learns exponentially more information regarding each feature, per epoch. Allowing the loss curve to converge both at a faster rate at a lower error amount.

As the total loss curve is the sum of both the content and style curve, with the main change in loss originating from the style loss. The style curve for the “1k” dataset has a similar issue to it’s total loss curve.

While the content loss curves for both datasets are much shallower in their convergence rate as opposed the style loss curves, due to the content images acting as the base image for the overlay. The content loss curve that is generated for the 1k dataset is more linear than that of the loss curve from the 10k dataset. This is due to the lack of images in the 1k dataset when compared to the curve form the 10k dataset. Due to lack of learned features, the data does not have the opportunity to converge as the 10k loss function has.

## Are there qualitative differences in output between the “1K” vs. “10K” tests? If so, what accounted for these differences?

Yes, overall, the output of the 1k decoder is of much lesser quality than that of the 10k decoder. With regards to the content image, there is visibly less detail in picture, causing the image to look more like it was formed by an assembly of polygons. 10k decoder reconstruction on the other hand includes a lot more of the finer details from the initial content image. Furthermore, when it comes to the style image’s implementation, it appears to look a lot rougher for the 1k decoder (less blending of colours/styles) than that of 10k decoder. In addition, the outlines in the image appear to look more defined in the 1k decoder images than that of the 10k, further contributing to the rougher appearance of the 1k image. These differences in quality are due to better trained parameters in the within the 10k decoder. Because the network has significantly more data to use during the 10k dataset training session, the decoder had the opportunity to further refine its internal weights (parameters), as opposed to the decoder using the 1k dataset. As a result, it was able to deconstruct and reconstruct, the combined input images more precisely, while incorporating the minor details from both the content and style image.