ELEC 475

Lab 2 – Neural Style Transfer with AdaIN

Lab 2 – Neural Style Transfer with AdaIN

Michael Greenspan

Contents

1. Introduction

In this lab you will be implementing a version of AdaIN, which allows transferring the style of one image onto another.

The distribution includes the modules **`AdaIN_net.py`**, **`custom_dataset.py`**, and **`test.py`**. There is also an **`encoder.pth`** file that contains the pre-trained weights (trained from ImageNet) of the **`AdaIN_net`** encoder, and example scripts for training and testing.

Your task is as follows:

- Complete the **`forward`** method in the **`AdaIN_net.py`** module
- Create a train.py module, that follows the command-line syntax of the **`train.txt`** script
- Train and test your system on the COCO1k and wikiart1k, and COCO10k and wikiart10k datasets

2. Resources

2.1 Paper, and Code

To understand the method, a valuable resource is the AdaIN original paper:

https://arxiv.org/abs/1703.06868

There also exist a number of existing pytorch implementations of AdaIN which you can make use of, such as:

https://github.com/naoto0804/pytorch-AdaIN

2.2 Datasets and Dataloaders

When training your system, for each epoch, you'll need to loop through two datasets simultaneously: one for the content images, and one for the style images. One way to do this is to wrap each dataloader in an iterator, as follows:

```
for epoch in range(n_epochs):
    for batch in range(n_batches):
        content_images = next(iter(content_dataloader)).to(device)
        style_images = next(iter(style_dataloader)).to(device)
        # ...
```

2.3 GPU vs. Colab vs. CPU

If you don't have access to your own GPU, then Google colab is one way to access GPU resources to accelerate training. I found that colab was particularly useful when training using the "1K" datasets, but that it timed out when training on the "10k" datasets.

One strategy then is to debug your code using colab on the "1k" datasets, and then run your final tests using a (local) CPU the "10k" datasets. My final "10k" tests on my desktop workstation CPU took ~20 hours of runtime, so make sure that you leave yourself enough time to execute.

3. Deliverables

The deliverable as described below comprises the following

- The completed code;
- An example output loss plot;
- Example output images;
- A report.

3.1 Completed Code

The **`forward`** method shall be completed in the **`AdaIN_net.py`** module. No code within the **`AdaIN_net.py`** module shall be altered, other than the **`your code here`** sections as indicated within the **`forward`** method itself.

3.2 Command Line Scripts

You shall provide an example **`my_train.txt`** and **`my_test.txt`** script, to execute your training and testing processes. These shall follow the syntax specified in the provided **`train.txt`** and **`test.txt`** scripts. Your **`my_train.txt`** script should generate the weight files for the "1K" and "10K" training processes, and your **`my_train.txt`** script should generate the output images described in Section 4.4 below.

3.3 Loss Plot

The loss plot shall include both the (scaled) content and the style loss elements, as well as the total loss, for the "1K" and "10K" training processes. An example loss plot for the 1K dataset is as follows:

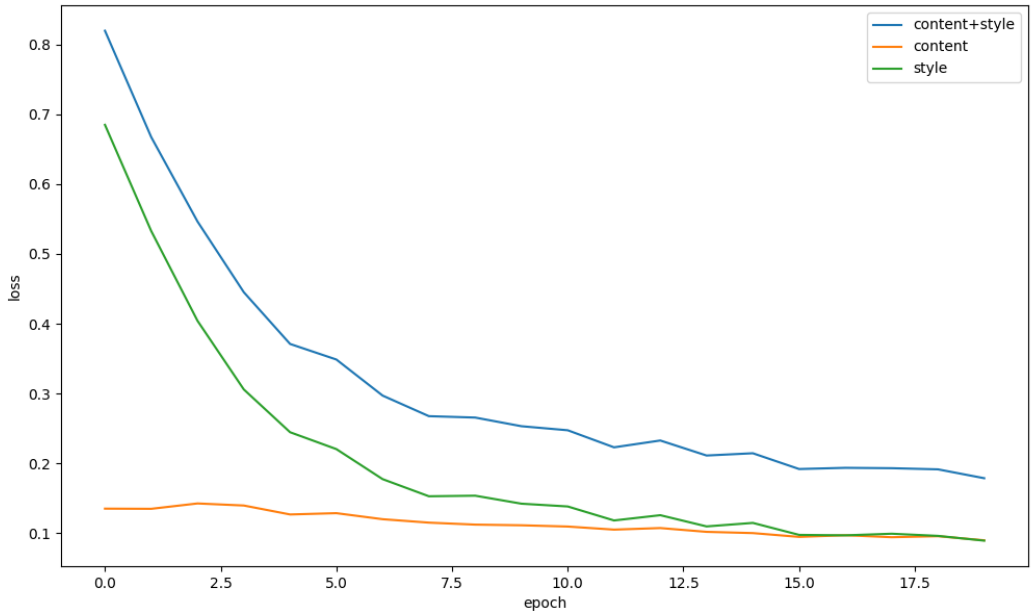

Figure 1: Loss Plot, 1K Dataset

3.4 Output Images

The output images shall be generated using the **`test.py`** module, using syntax specified in the **`test.txt`** script. Generate output images for a selection of 2 content images and 3 corresponding style images for each content image (i.e. 6 combinations total). For each content-style image pair, execute the test script using the three alpha values of 0.1, 0.5 and 0.9. This will result in 18 output images total.

An example of some output images for content image **`baboon.jpg`** and style image **`brushstrokes.jpg`** is follows:

alpha = 0.1 alpha = 0.5 alpha = 0.9

Figure 2: AdaIN Style Transfer Example

3.5 Report

You report should include the following information:

- A description of the hardware used (CPU, GPU, colab, etc.) for both "1K" and "10K" training stages.
- A description of the hyper-parameters used to train your system (initial learning rate, optimizer, batch size, etc.) for both "1K" and "10K" training stages. This should be complete so that your process could be reproduced.
- A statement of which (if any) existing AdaIN implementations you drew from, or which helped you in your implementation.
- A statement of (approximately) how much time it took to train each dataset.
- What value of **`gamma`** did you use for training, and why?
- 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.
- Are there qualitative differences in output between the "1K" vs. "10K" tests? If so, what accounted for these differences?

4. Submission

The submission should all of the elements described in Section 4.

All deliverables should be compressed into a single **`<zzz>.zip`** `file,` where filename **`<zzz>`** is replaced with your student number. If you are in a team of 2, then concatenate both student numbers, separated by an underscore. The zipped directory should include your code and scripts, your trained parameter files, and all output plots and images.

The report should include a title (e.g. minimally ELEC 475 Lab 2), your name and student number. If you are working in a team of two, then include the information for both partners in the report (but only make one submission in OnQ).

Your code should train by entering the syntax provided in your included **`my_train.txt`** on a PyCharm terminal. This should generate the decoder **`*.pth`** files used in testing, and the loss plots specified in Section 3.3.

Your code should then test by entering the syntax provided in your included **`my_test.txt`** on a PyCharm terminal. This should generate your output images specified in Section 3.4.

The marking rubric is as follows:

Item	mark
Code modifications	2
1K training and testing	1
10K training and testing	1.5
Report	1.5
Correct submission format	1
Total:	**7**