Style Transfer Using Deep Neural Networks

CS 182 - Designing, Visualizing & Understanding Deep Neural Networks

Final Project Proposal

Team Name: The PhotoShop

Ze Li (3033242422): Senior, majoring in EECS
Zhanyuan Zhang (3033208206): Senior, double majoring in Statistics & Computer Science
Bofan Xue: 3031925157: Junior, majoring in CS
Guowei Yang (3033254902): Senior, majoring in EECS

Problem Statement and Background

Deep neural networks have been really powerful nowadays, and one extremely use of deep neural nets is in the field of image processing. In some cases, such as tuning the tone of an image so that it matches another given image, or even changing the image so that it has a more artistic style, it would be handy if there exists a tool that takes in two images as input, and transfers the style from one to the other. This is where deep neural nets come into play. The Generative adversarial network (GAN) does this job perfectly. We plan to use a GAN with two seperate loss functions as our model and use it to achieve the style transfer task. We could transfer painting style to real world images, and potentially convert a painting to a real world image.

Algorithm overview:

The inputs are two images namely the content image and the style image. Our goal is to generate an output image that has the same content as the content image with the style of the style image. The fact that we have two sets of trainable parameters leads the algorithm to be divided into two parts: 1) training the cnn and 2) training the output image. In this project, we won't train cnn from the sketch but instead take the pre-trained vgg-16 model with fine-tuning and then mainly focus on the second part.

In order to train the output image, we have defined a loss function that captures both the content loss and the style loss which has the formula $loss = \alpha \cdot content\ loss + \beta \cdot style\ loss$, where α and β are hyper-parameters. In each

epoch, we feed the two input images and the generated image(randomly generated at the very beginning) into our cnn and then perform gradient descendent to tune the generated image.

Here is the main idea of how both losses are computed:

Content loss: This is relatively easy, compute the element-wise squared difference between the outputs of content image and generated image after each activation layer and then sum them together.

Style loss: Vgg-16 is a well-trained model that can effectively capture the features in the image, in other words, it separates different features. The style in the image is an attribute that has little variation across features. So, we define the style of an image by the correlation between each pair of its features, then the style loss of a layer's output would be the element-wise squared difference between the sum of the generated image's correlations and the sum of the style image's correlations.

The Data Source You Intend to Use

We have downloaded the pre-process image libraries including animals, buildings etc as content images and paintings to be used as style images. Also, both datasets will be used to fine-tune vgg-16.

Description of the Tools You Plan to Use

Based on what reference model and what data set we found, what computation resources we have access, what libraries we are familiar with and other the practical considerations, we may use the following tools in this project:

- Libraries and frameworks:
  Tensorflow: not quite familiar with, but many models use for implementations.
  Keras with Tensorflow backend. PyThorch. MxNet.
- Computation resources: AWS
- Model evaluation: Besides the evaluation method coming along with the reference models, we may use other evaluation tool (for example, PRISMA).

Evaluation

Since we are working with unsupervised learning, there is no canonical way of testing our model. However, given the development of style-transfer techniques through the past few years, there is a commercial app called PRISMA (https://prisma-ai.com/), with which we can compare our model with. We will focus on the following criteria when judging the model performance

- Inference time: how fast is our model?
- Style: How complete is how style transfer?
- Content: Is the original object still recognizable in the output picture?
- Color: How vivid is our output picture?
- Sharpness: what is the level of resolution our model can achieve?
- Robustness: What is the success rate of our model performing style transfer?

References and Resources

https://www.kaggle.com/c/painter-by-numbers/data
https://bam-dataset.org
https://github.com/metmuseum/openaccess
https://github.com/trevorfiez/The-Metropolitan-Museum-of-Art-Image-Downloader

https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f