Abstract:

The goal of the project is to do the style transfer. Our first step was to read the paper that introduced NST to the research community. The authors of the paper present an algorithm to create artwork through neural networks by transferring the style from one image onto the content of another.

Humans are good at recognizing the overall style of an image, and those skilled at art can even reproduce art pieces in the style of other, more famous works of art. The algorithm attempts to model this capability of humans to produce new pieces of art. Using the activations of intermediate layers in VGG19 pretrained weights to extract the information from the content and style images, the output image was optimized to minimize 2 types of loss: Content loss and style loss.

We mainly referred to two resources of the code part, one is the PyTorch Tutorial of Style Transfer, another is the code from Professor's lectures. We dove into the details of the two different models and made the modification to those models and found several interesting aspects. Our project achieved a relatively good result of transfer effects.

Team: Liying Li, Jinghui Zhao

Description of your problem and dataset:

Problem:

The goal of the project is to do the style transfer. The Neural Style Transfer is an algorithm that takes as input a content-image and a style-image and returns the content of the content-image as if it was 'painted' using the artistic style of the style-image. We used deep learning to separate the content representation of an image from the style representation.

Data:

We used the pretrained VGG19 model. Therefore, we can directly extract the features of the input images generated from different Convolutional layers. The training process works for extraction the textures from style image input and object content from content image input. Hence, we only need to provide the style images and content images. Basically, for each training process, we only need one content image and one style image.

Links to code:

https://github.com/DeepLearningWithPytorch/dl-final-project-jinghui-liying

Explain in which ways you have modified the original code:

1. Loss function: We used MSE losses for both content loss and style loss. The content loss represents a weighted version of a content distance for an individual layer. And the style loss is similar to the content loss.
2. We changed the method to preprocess the input images. The OpenCV package always has conflicts in many Anaconda environment. The AWS instance we use is p2xlarge of deep learning instance. The cv2 can't work normally. Hence we changed the methods to preprocess the input images.
3. To combine the loss functions and the pretrained model, we added the content loss and style loss immediately after the convolutional layer. We used a Sequential module to achieve this goal. According to the paper, we used L-BFGS algorithm to run the gradient descent. In the last step, we defined a function to do the neural transfer.
4. We changed the layers in the pretrained model to extract the features of input images.
5. We added a normalization layer to normalize the input images.

Details of your machine learning methods and experimental results:

The convolutional neural networks are applied here. It consists of a batch of layers. Each layer of units can be considered as a collection of image filters. Every filter would extract certain features from the input image.

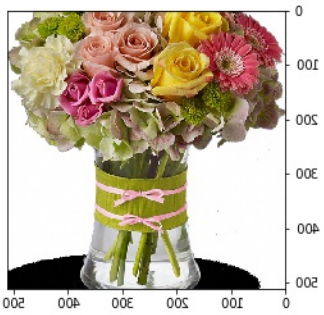
Higher layers in the network capture the high-level content, like the objects and their arrangement, but not the exact pixel values. We, therefore, refer to the feature responses in higher layers of the network as content representation.

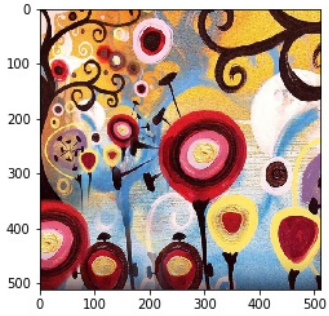
To obtain the representation of the style of the input image, we can use a feature space to capture texture information. This feature space is built on top of the filter responses in each layer of the network. It consists of the correlations between the different filter ng the correlations of multiple layers, we can obtain a stationary, multi-scale representation of the input image, that is the style representation.

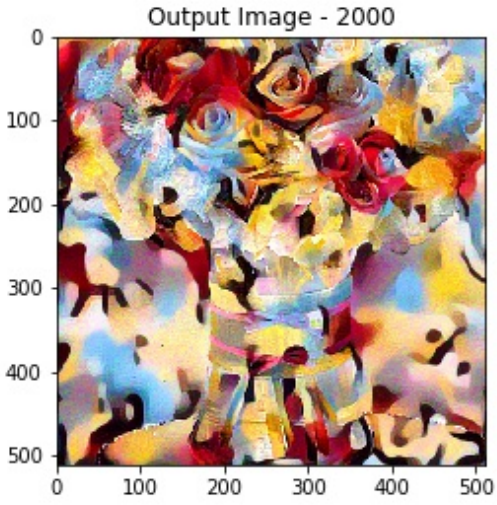
And during the training process, we minimize the loss function containing two parts separately. One part is content loss and another is style loss. Thus, the model can smoothly regulate the emphasis on either reconstructing the content or the style.

Problems and challenges

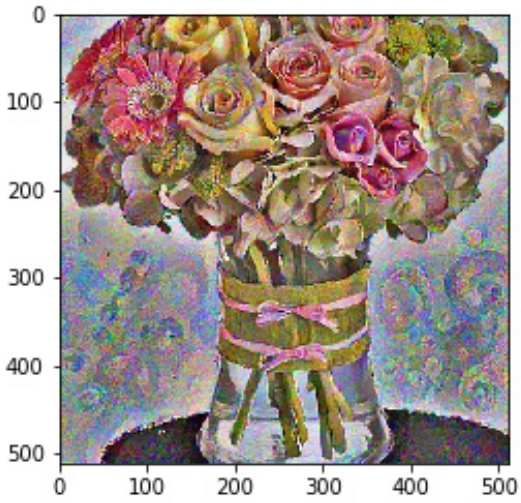
Using the models from Toturial, we changed the input images to transfer the styles. However, we get the results below:







We then adapted the models from Yannet. Then we get the following results:



We looked into the details of the models and found several points of difference of these two models:

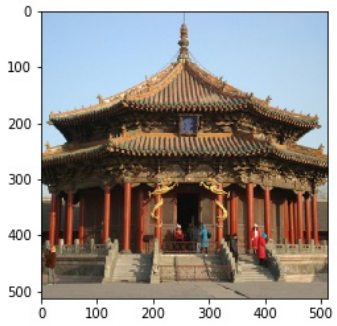
1) The model adapted PyTorch Tutorial uses one model to train both the style image and the content image. In this model, we inserted the style loss layer after every selected convolutional layers for extraction style textures. Also, we inserted the content layer after the content convolutional layer. However, the models adapted from lectures use two independent models to extract the features of style image and the features of the content image. And we wrote a training function to combine the content loss and style loss to calculate the final total loss.
2) As we can see above, the model from Tutorial captures more style textures and ignores much of the content object, while the model adapted from lectures mainly captures the content and fails to extract much of the style textures. This is because the layers we selected are different. The first model uses five lower layers to extract features of style images, while the second model uses three lower layers and two higher layers to extract the features of style image. At the same time, the first model uses the pretty low layer to represent the content image, while the second model uses a much higher layer to represent the content image. Those make the second model does well in capturing the content object while the first model can only capture the styles well.

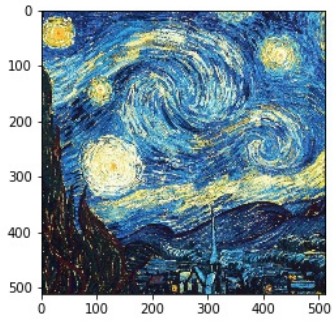
The training time could be another problem for us. We setup a p2xlarge AWS instance with GPU to run the transfer training process. Basically, a 2000 iteration training process would take 15 minutes to run. After referring more materials, we find some methods to resolve this problem. And we can try to implement those methods in the future.

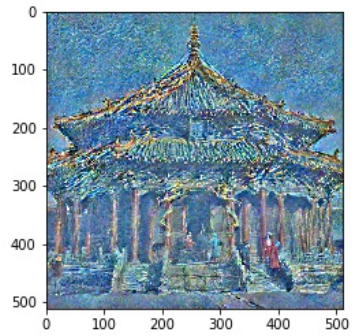
Results

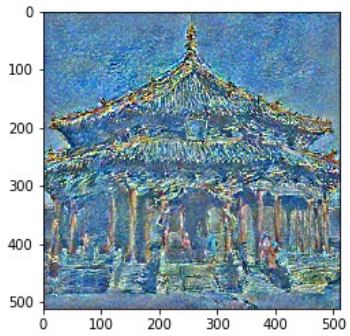
We found that the number of iterations would affect the transfer output. The input images that we selected would affect the difficulty level of transferring the styles of the content image.

After running the model on many different images and styles, and after we modified some details of the initial model, we finally get a model could transfer the styles of many of the content images.









Conclusions and key lessons learned:

Conclusions:

This is a great example of the representative powers of neural networks, and how seemingly enigmatic layers and algorithms can produce such beauty. The algorithm proves how similar neural networks are to our own understanding of this world.

However, there are still many challenges of building the models. And there are many spaces that we can improve.

Lessons learned:

Regarding improving the quality of the generated image, we did several experiments to find optimal hyper-parameters. And style transfer involves various hyperparameters like which content layers to use for calculating content loss, which style layers to use for calculating style loss. Instead of using a neural network-based method for training and finding optimal hyperparameters, we used the optimization based method on our test images to create their stylized versions and cross-validation.

If we a style image of smaller size, the network is unable to figure out enough style information and the images generated are of poor quality. The cropped region of the style image also matters, and this should be chosen on the basis of the region containing the brushstrokes we want in our stylized image.

List the responsibilities:

Liying Li: Modeling
Jinghui: Summary, Report, Presentation