EE 4780 Project Breakdown

* VGG Network
  + In pyTorch, the way you create a network is overloading nn.Module. there are two parts to it always, first is it \_\_init\_\_ method which initializes all yours layers, and then there is the forward method, the forward method is used to actually pass your data through the network and get the outputs.
* Transform input image to a torch tensor
  + Transform the input image, no matter what original resolution to convert it into something that can be fed into the VGG network.
* Load images into program with CUDA
  + Usually, we have our data on the CPU, but the training of NN models for efficiency is trained on GPUs, hence we use the .cuda() function to transfer objects like our style and content image onto the GPU.
* The Loss Function
  + First we have the gram matrix class. We use the Gram Matrix to extract feature layers, the .bmm() stands for batch matrix multiplication. In batch matrix multiplication, the last 2 dimension are used for the matrix multiplication while the rest of the dimension are treated as batches. The style loss is computed by first passing the layer to get its gram matrix representation and then taking a mean squared error between the representation and the target. Going through the rest step by step, we detach the layers from the VGG network and add them to the respective target layers. Then we go onto define the losses, targets and the loss layers.
* Training Loop
  + To outline a general framework, we do three things in any training loop, We decide on the optimizer, compute the loss and backprop after that. In the cases I’ve seen, LBFGS is used. We can call the .zero\_grad() function because, pytorch by default accumulates gradients, so we need to set the gradients to zero each time before computing them. The total loss is then accumulated by going through all the style and content layers, after which the .backward() function is used for computing the gradients and .step() is used for updating the weights.

Reference for these steps:

<https://towardsdatascience.com/neural-style-transfer-series-part-2-91baad306b24>