Pattern lab3

## Done By: Ahmed Nasser – 8350 Baher Ahmed Hosny – 8165

**Data Loading and Splitting:**

* We specified the content path and the style path for the folders with their respective images.
* We used the path to gather all pictures and placed them in a content\_images folder and a style\_images folder.
* We had a maximum of 10,000 images so we split them to 9,000 content images and 1,000 style images.
* We split the data with 80% training, 10% validation and test.

**Data Preprocessing:**

* We resized all images to 256 \* 256 and used ToTensor to conver PIL image to pytorch tensor with shape [C,H,W].
* We stripped out extra channels to ensure RGB and applied imagenet normalization and then we applied preprocessing.
* We differentiate training and evaluation with a flag
* A computer screen shot of a program

  AI-generated content may be incorrect.The input contains a content image and a random style image and a style threshold

A screenshot of a computer program

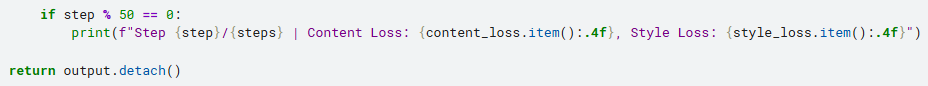
AI-generated content may be incorrect.

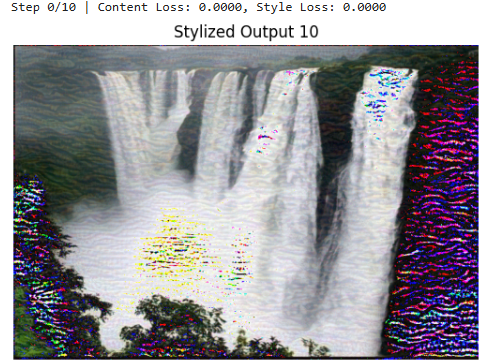
**Creating the Model:**

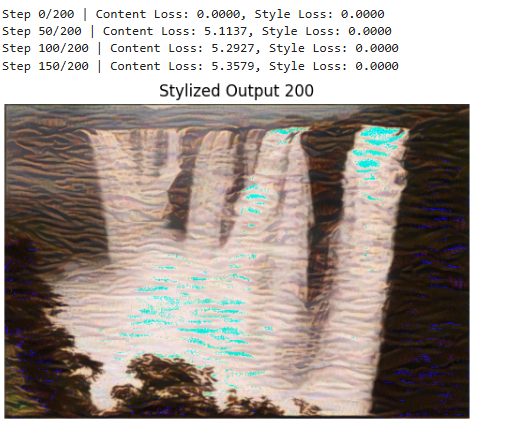
* We use a VGG-19 model which is convolutional network pretrained on imagenet. We remove fully connected layers used for classification and set it to evaluation mode and keep it frozen to ensure static feature extraction.
* We implemented the Gatys method which uses different layers for style and content losses by extracting specific layers for content (layer 21 which is conv4\_2) and the same thing for style (0,5,28).
* A screenshot of a computer program

  AI-generated content may be incorrect.We used gram matrices to compute style statistics which captures the correlations between feature channels.
* A screenshot of a computer program

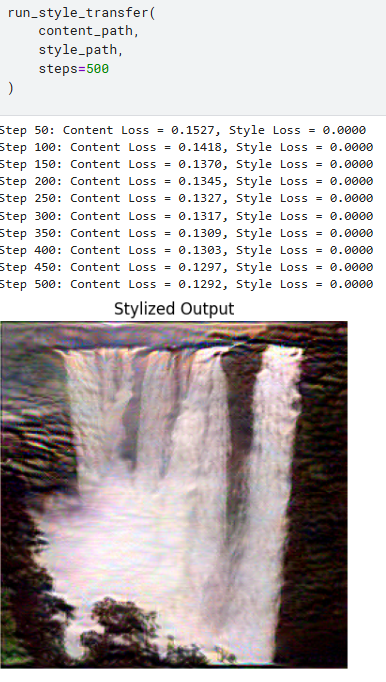
  AI-generated content may be incorrect.We optimize the image itself instead of updating the parameters, the gradient is taken with respect to the image pixels to blend content and style. Content\_weight and and style\_weight control how much style is transferred.

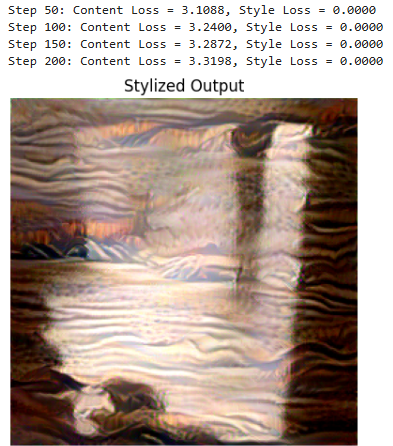




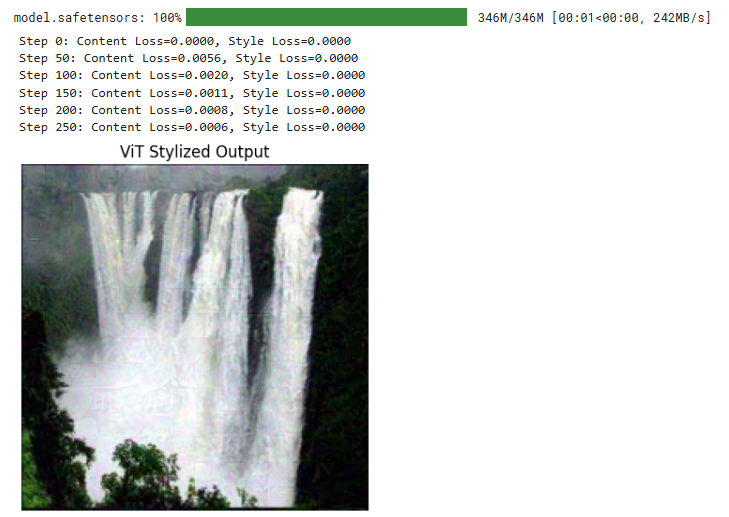


**Finetuning the model:**

* Like before, we use VGG model with the same layers for content and style. We load the image and calculate the losses (for content it’s the mean square error between the output and the content features, for style, it’s the mean square error between gram matrices of output and style features at each layer).
* We implemented batch training
* We added style\_weight to scale style loss.
* The main optimization loop uses L-BFGS which is a second order optimizer suited for image generation. We implemented things that help reduce loss like backpropagation and updating the weights so in the next loop, the loss can decrease.



**Bonus 1:**

* We resize the image to 224\*224 because that’s what ViT expects and normalized it with -1,1.
* We used a slightly different gram matrix where the patch tokens are used instead of feature maps. This computes global style correlations using self-attention token embeddings.
* We built the ViTfeatureextractor model using a pretrained ViT from timm. We removed the classification head (reset\_classifier(0)) and extracted token embeddings from the final transformer layer. [CLS token is used for content and the remaining patch tokens are used for style].
* The loss calculations are similar to the VGG model, the content loss is the mean square error between the CLS token of the output and content image. The style loss is the mean square error between gram matrices of patch tokens excluding CLS of output and style image.
* Here we used the Adam optimizer instead of L-BFGS and optimized the image pixels instead of the weights to minimize loss.