Technical Approach

December 30, 2024

0.1 Technical Approach

0.1.1 Feature Extraction

We have implemented Gatys-style and Lapstyle methods. For both methods, we use convolutional layers of VGG19 to extract features of images. To get better performance, we replace max pooling layers with average pooling layers.

Specifically, we use feature map of `conv4_2` output as the content feature. And **content loss** is defined as follows:

$$\mathcal{L}_c = \sum_{c,i,j} (F^c_{c,i,j} - F^x_{c,i,j})^2, \tag{1}$$

where F^x is the feature map of the target image, and F^c is the feature map of the content image , both of shape (C, H, W).

For style features, we implement calculation of Gram matrices on `conv1_1`, `conv2_1`, `conv3_1`, `conv4_1`, and `conv5_1` layers. Given a feature map F of size (C, H, W), the Gram matrix $G \in \mathbb{R}^{C \times C}$ computes the sum of products between channels. The entries k, l of the matrix are computed as:

$$G_{k,l} = \sum_{i,j} F_{k,i,j} F_{l,i,j}. \tag{2}$$

In practice, we choose to normalize the Gram matrix by dividing it by the product of the height and width of the feature map:

$$G_{k,l} = \frac{1}{HW} \sum_{i,j} F_{k,i,j} F_{l,i,j}. \tag{3}$$

Then the **style loss** can be computed as follows. Define the Gram matrix of input image feature map and style image feature map of at the l^{th} layer as $G^{x,l}$ and $G^{s,l}$, and the weight of the layer as w^l. Style loss at the l^{th} layer is

$$L^l_s = w^l \sum_{i,j} (G^{x,l}_{i,j} - G^{s,l}_{i,j})^2, \tag{4}$$

where w^l is the weight of layer l. The total style loss is a sum over all style layers:

$$\mathcal{L}_s = \sum_l L^l_s. \tag{5}$$

0.1.2 Lapstyle

For Lapstyle implementation, there are additional loss items.

We implement a Laplacian loss into neural style transfer. The Laplacian loss is defined as the mean-squared distance between the two Laplacians.

The Laplacian loss is computed by a small two-layer fixed CNN which includes an average pooling layer and a prespecified convolutional layer. The former layer smoothes the input image which can make the Laplacian loss better reflect its true detail strcutures. The latter layer combines a Laplacian operator to detect the edges of the content image.

We compute the Laplacian of an image on RGB channels, which means the Laplacian value consists of the three Laplacians:

$$D(x) = D(x^R) + D(x^G) + D(x^B). \tag{6}$$

Given the content image and the stylised image, we can compute the Laplacian loss to measure the difference between their Laplacians:

$$L_{Lap} = \sum_{i,j} (D(x_c) - D(x))^2_{i,j}. \tag{7}$$

In addition, we implement the depth loss. The depth loss function is used to measure the depth differences between the transformed image and the content target image. In order to preserve maximum depth information and potential structural features, we take the outputs of the depth estimation network and compute the distances as the depth loss.

We choose **MiDaS** as the depth estimation network. The depth loss is the (squared, normalized) Euclidean distance between feature representations:

$$\mathcal{L}_d = \sum_{i,j} (D^c_{i,j} - D^x_{i,j})^2. \tag{8}$$

The overall loss item will be computed as a weighted average of aforementioned loss items. The choice of weights will be discussed later.

0.1.3 Image Preprocessing and Initialization

In practice, we find the preprocessing of images crucial to final performance. Specifically, the input images of VGG19 must be normalized according to the mean and standard deviation of ImageNet dataset. And since the pretrained model provided by PyTorch was trained using images with 8-bit bit depth, we need to make sure that pixel values fall in 0-255 before normalization.

In our experiments, for the initialization of the target image to be optimized, both white noise and the content image are used. White noise initialization provides more scope for style transfer.