Image Style Transfer

Neural Style

Overview

Combine style and content from different images to generate a new image.

Why does it work?

We choose multiple layers where we perform style loss and only a few layers where we do the content loss. The placement of these layers is essential. We want to capture small patterns as well as the overall style/structure of the image.

Convolutional deep neural networks used for object recognition inherently learn to separate the style and content from given images which we use here.

Implementation Overview

CNN Model Selection and Framework

We need to use a trained CNN model since training one of our own would require a lot of time and a lot of computing resources.

Possible options: VGG, AlexNet, GoogleNet

Torch - Lua JIT

Among the fastest and simplest frameworks to use for modelling networks and training them.

Style and Content Loss Modules

We would be using 3-5 style loss modules and 1-2 content loss modules along the 19 layered conv net (VGG) that we would be using initially.

We define these modules by ourselves which will be added in the network manually (Essentially we are adding loss modules to generate a loss in the network which can be back propagated as gradients)

Content Loss

1. Compute features of content image at input of selected modules and store it as target value at that module.
2. When we forward the input image x through the network, we compute loss at the above selected modules. A standard squared error loss is used.
3. While backpropagation, we add weighted gradient of this loss at this module.

Style Loss

1. Compute gram matrix of features of style image at input of selected modules. We use these as target for style at these particular modules.
2. Gram matrix is a dot product of pairs of features obtained at a module. Gram matrix is found to be good approximation for style as it gives similarity between various filters.
3. Similar to above, we add weighted gradient of this loss during backpropagation.

White noise to Image

- Using white noise as input(x), propagate through the network to get the output (y).
- Set dy = 0, and start back propagation.
- Initially the gradients would be zeros, but as soon as we reach our loss modules that we inserted in the network, the gradients start accumulating. These are now passed all the way till the input as dx.
- We update the input as x’ = x - dx*rate1 (Gradient descent on the input image)
- Perform this multiple times (1000s) and then we would end up minimizing the loss at the input image and our image x would start to have the content of our content image and the style of the style image we used as inputs initially.

Goals for evaluation 2

- Complete paper implementation.
- Apply various optimizations and experiment using different networks.
- Try using sketches for the content and artworks for the style to try come up with an automatic sketch colouring.

Work Division

Vishal Batchu

- Implement the loss modules
- Generate modified network adding in these modules
- Train initially using lbfgs, for basic outputs

Raghuram Vadapalli

- Various optimizations such as adam etc, to check which performs better.
- Image pre and post processing to improve results
- Try using different trained models for accuracy.

Ameya Prabhu

- Develop and formalize the theory which needs to be implemented.
- Define functions mathematically which would be implemented.
- Time based optimizations for the code to make it faster.
- Gram matrix extensions and other methods.