Style Transfer using Convolutional Neural Network

Ryan Chan , Last Updated: 20 October 2018

Motivation

Layers in neural network contains useful information. For example, one can use convolutional operation to reduce the dimension of the data, while embedding common information between each layer. Formerly known actviation maps, they contain useful presentations that can be processed for further purpose. Artistic Style Transfer is one of many examples that utilizes actvations in convolutional neural networks (VGG19) (Simonyan, K., & Zisserman, A. 2014) to produce useful results. This project sets to explore activation maps further.

Instruction for Testing and Producing Results

VGG weights

First download vgg weights from here. Put this in `/style_transfer/vgg/`. No change of file name needed.

Model Options

All options for training are located in `main.py`. The options you can fine tune are:

1. Dimension of the image
2. Layers for the style and content image activation maps
3. Weights for each layer
4. Trade-off between style and content (`alpha` for content and `beta` for style)
5. File path for content and style image
6. Initial image (content image, style image, white image, or random image)
7. Number of steps between each image save (`save_per_step = -1` if no saving wanted)

To run the model, run in command line

```
python3 main.py
```

Model Structure and the Flow of Information

Preprocess

1. style image is rescaled to be the same size as content image.
2. When images are loaded and turned into `(height, width, channel)` array, mean pixel values are subtracted from them such that their pixel values are centered at 0. This is due to the properties of the weights in our VGG Network, and computing the gram matrix requires values to be centered at 0.
3. Both image are passed into the VGG network, and activation maps from specific layers are extracted.
4. For activation maps from style image, we pre-compute each layer's gram matrix.
5. A random image is generated, ready to be updated at each iteration. This is our only variable that is being updated.

Generating result

1. Each iteration, we pass in the random image to obtain the same layers of activation maps we chose for content and style.

2. We then compute the content loss, which is the mean square error between the activation maps of the content image and that of the sythesized image. Content loss function can be described by the following equation:

$$L_{\text{content}} = \frac{1}{2}(\sum_{i,j} w_l^{(c)} \cdot (F_{ij}^l - P_{ij}^l))^2$$

   , where F^l is the activation map at layer l of the generated image, and P^l is that of the original content image. $w_l^{(c)}$ is the weight of each layer.

3. Similarily, the style loss is the mean square error between the gram matrix of the activation maps of the content image and that of the sythesized image. Gram matrix can be interpreted as computing the covariance between each pixel. Each layer's style loss is multipled by a style loss weight such that style loss from each layer is averaged out. Style loss can be described by the following equation:

$$L_{\text{style}} = \frac{1}{4M_l^2N_l^2} \sum_{i,j} w_l^{(s)} \cdot (G(F^l)_{ij} - G(P^l)_{ij})^2$$

   , where $G(\cdot)$ is the function that computes the grammain, M^l is the height times width of the activation map at layer l, and N^l is the number of channels of the activation map at layer l. $w_l^{(s)}$ is the weight of each layer.

4. The content loss and style loss are multipled by their respective tradeoffs, is then added up together, becoming the total loss. Our objective then is the minimize the following loss function:

$$L_{\text{total}} = \alpha \cdot L_{\text{content}} + \beta \cdot L_{\text{style}}$$

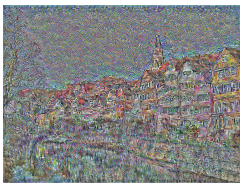
, where α is the weight trade-off of the content loss, and β is the trade-off of the style loss.

5. At each iteration, the random image is updated such that it converges to a synthesized image. Our model uses L-BFGS algorithm to mimize the loss.

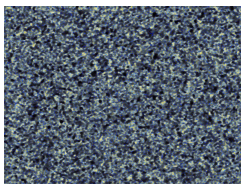
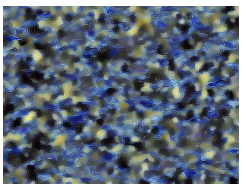
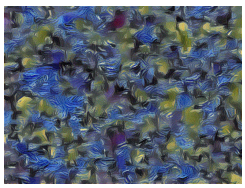
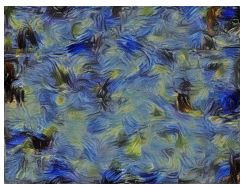
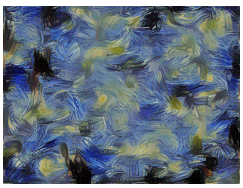
Replication of Figures in Paper

Figure 1 - Image Representations in a Convolutional Neural Network

Content Reconstruction. The following figures are created with `alpha = 1, beta = 0`.

				
`relu1_1`	`relu2_1`	`relu3_1`	`relu4_1`	`relu5_1`

Style Reconstruction. The following figures are created with `alpha = 0, beta = 1`.

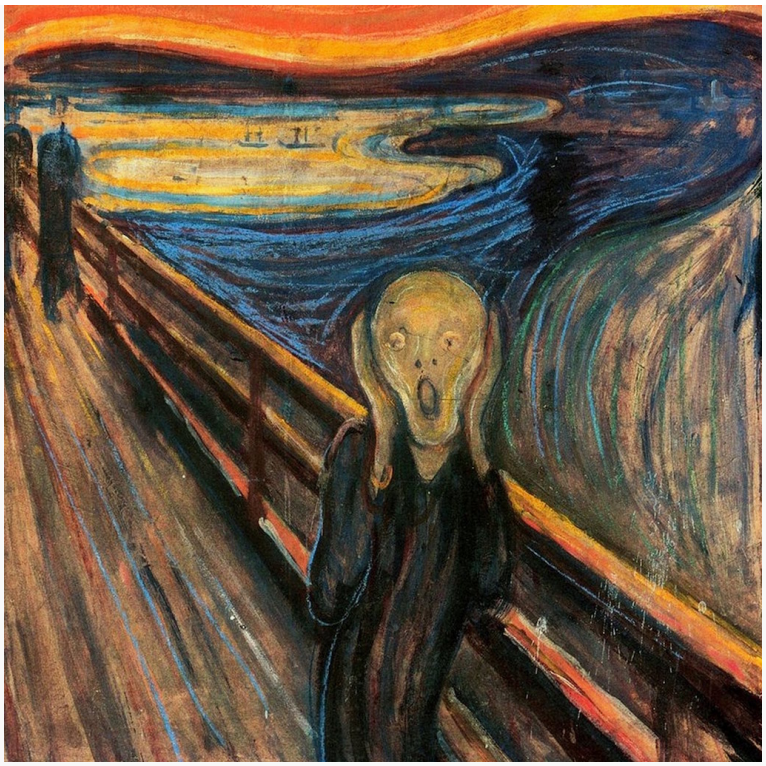
`relu1_1`	`relu1_1`
	`relu2_1`
		`relu1_1`
		`relu2_1`
		`relu3_1`
			`relu1_1`
			`relu2_1`
			`relu3_1`
			`relu4_1`
				`relu1_1`
				`relu2_1`
				`relu3_1`
				`relu4_1`
				`relu5_1`

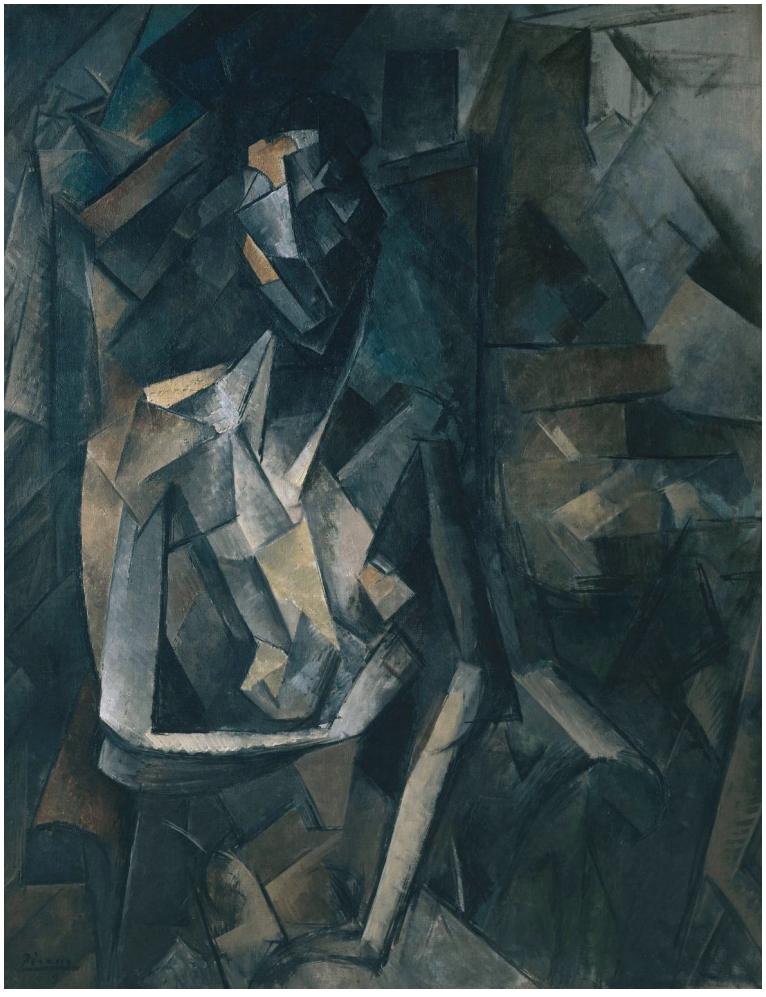
Figure 3 - Well-known Artwork examples

The following figures are created with:
Loss Weights: `alpha = 1e-6, beta = 1`

Style Weight:
`relu1_1 = 0.2 , relu2_1 = 0.2, relu3_1 = 0.2, relu4_1 = 0.2, relu5_1 = 0.2`
Style Layers: `relu1_1, relu2_1, relu3_1, relu4_1, relu5_1`
Content Layers: `relu4_2 = 1`
Content Weight:





Difference from original paper

A subtle difference between Leon's original implementation and this version is that the trade-off used to create the results are different. In the original paper, `alpha / beta = 1e-4`. Yet, I was unable to create the results with that loss trade-off. Hence, the figures about uses a `alpha / beta = 1e-6` trade-off. I was unable to find where the difference in implementation of model is.

Future Work

Definition of Representation. One advantanges of using neural networks on images is that there already exist perhaps the most useful and direct way to represent an image using numbers - pixel values. But this representation is not necessarily the only way to represent visual content. If there exist a different kind of "embedding" that encodes objects or relationship between pixels in a different way, content and style representation might change the way style transfer model defines the relationship between objects, or even color.

CNNs to Other Types of Neural Nets. One inspiration of Convolutional Neural Networks is the hierachical structure of simple cells and complex cells in the human visual cortex. Layer by layer, using convolution operation, an artifical neuron serves as a computing unit that summaries information from previous layer and compresses into a smaller space, which is then passsed onto the later layers. This type of model is one of many ways of compressing into a more meaningful and less redundant representation. Other models for compression includes autoencoders, which requires information to be passed down a smaller dimension and projected into a larger dimension again. Compression problems might shed insights on how information is embedded efficiently.

Losses and differences. The current style transfer model utilizes mean square error, which computes the

difference between pixel values from the content or style image and the synthsized image. From a mathematical point of view, this seems logical and reasonable. But, a difference in pixel value may not necessarily imply a difference in content or style. For instance, if we were to create a synthsized image that is more invariant to the position of objects in our synthesized image, calculate the exact difference in pixel at each coordinate would not be sensible. In other words, the definition of loss when considering objects may require a much more extensive function than computing losses.

Further Readings

1. Jing et al. 2018. Neural Style Transfer: A Review. Link to Paper Link to Github
   This github repository and paper provides a general overview of other posibilities of style transfer. There are now different branches of style transfer, while some focuses more on keeping the content and some focuses on keeping the style. There are also improvements in different aspects, such as training speed, or time-varying style transfers.

2. Johnson et at. 2016. Perceptual Loss for Real-Time Style Transfer and Super-Resolution. Link to Paper
   One potential change to Leon's model is to use the configurations that Johnson used in this paper. The similar result can be reproduced.

Other Github References

Throughout this project, I visited a few other implementations that provided me great insight to how to implement the style transfer model in a more efficient and neat way. The following is a list that I referenced.

1. https://github.com/hnarayanan/artistic-style-transfer
2. https://github.com/hwalsuklee/tensorflow-style-transfer
3. https://github.com/jcjohnson/neural-style
4. https://github.com/lengstrom/fast-style-transfer
5. https://github.com/fzliu/style-transfer
6. https://github.com/machrisaa/tensorflow-vgg
7. https://github.com/anishathalye/neural-style

As mentioned earlier, there is a slight difference in my implementation compared to the original implementation. I was trying to find one that exactly follows the original implementation, but most of them either also changes some settings on their own or implementations concurrently with other versions of style transfer.

Acknowledgement

I would like to devote my sincere gratitude to my mentor Dylan Paiton at UC Berkeley for the support he has

given. Much of this would not be possible without he continually mental and technical support. I have learned a great deal about neural networks and neuroscience through discussions and weekly meetings, and I look forward to the more research in the future.

Paper References

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2414-2423).

Personal Note

The artistic and imaginative side of human is known to be one of the most challenging perspective of life to model. Due to its free form and huamnly-cultivated experience, art is often appreciated not only because of its visual apperance, but also the history and motivations of the artist. In this project, I attempt to answer this question: "If we were to create a model that creates art, how would it do it, and what separates that from human life?"

This is my first project look in-depth into an academic paper and attempt to implement the model from scratch. Because it was widely used to illustrate what neural networks can do, artistic style transfer remains as one of the most interesting beginner projects. I am doing this to cultivate my extensive and critical thinking sills, and also understand the model thoroughly, to the extent where I have no doubt if asked to explain how it works from zero to a hundred.