Deep Learning HomeWork3 Task 3

Nikulina Kseniia
Department of Computer Science
University of Innsbruck
Innsbruck, 2022

February 11, 2022

Abstract

Changed the architecture, parameters, optimization, regularizer in GAN network. Applied a different reconstruction to origin image.

1 Introduction and background

In this work I want to explore how hyperparameters affect the output of the work. Implement another reconstruction and see how the origin image is changing and what new parameters are used to build reconstruction.

GAN Network - is a class of machine learning frameworks. Two neural networks contest with each other in a game. Given a training set, this technique learns to generate new data with the same statistics as the training set.

Reconstruction which parameters will be changed: Style transfer

Neural style transfer is an optimization technique used to take two images—a content image and a style reference image and blend them together so the output image looks like the content image, but "painted" in the style of the style reference image. This is implemented by optimizing the output image to match the content statistics of the content image and the style statistics of the style reference image. These statistics are extracted from the images using a convolutional network.

New reconstruction: Style Reconstruction

Style Reconstruction - is a methodology involving the encoding of an image into tensors using typical image replication networks, then applying a VGG framework before the decoding.

2 Methodology

Style Transfer describes the style transfer process between a white noise image x, a content image p, and a style representation a. Performing gradient descent of the content loss and style loss with respect to x impressions the content of p into x, bearing local styles, and colors from a.

The results presented in the main text were generated on the basis of the VGG-Network. We used the feature space provided by the 13 convolutional and 5 pooling layers of the 16 layer VGGNetwork. We do not use any of the fully connected layers. For image synthesis we found that replacing the max-pooling operation by average pooling improves the gradient flow and one obtains slightly more appealing results, which is why the images shown were generated with average pooling. We used SGD optimizer, instead of Adam.

Generally each layer in the network defines a non-linear filter bank whose complexity increases with the position of the layer in the network.

Hence a given input image $\vec{x}$ is encoded in each layer of the CNN by the filter responses to that image. To visualise the image information that is encoded at different layers of the hierarchy we perform gradient descent on a white noise image to find another image that matches the feature responses of the original image. So let $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated and P^l and F^l their respective feature representation in layer l. We then define the squared-error loss between the two feature representations

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

From the derivative of this loss with respect to the activations in layer l the gradient with respect to the image $\vec{x}$ can be computed using standard error back-propagation. On top of the CNN responses in each layer of the network we built a style representation that computes the correlations between the different filter responses, where the expectation is taken over the spatial extend of the input image. These feature correlations are given by the Gram matrix G^l To generate a texture that matches the style of a given image, we use gradient descent from a white noise image to find another image that matches the style representation of the original image. This is done by minimising the mean-squared distance between the entries of the Gram matrix from the original image and the Gram matrix of the image to be generated.

To generate the images that mix the content of a photograph with the style of a painting we jointly minimise the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN. So let be the photograph and be the artwork. The loss function we minimise is

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}\vec{x}) + \beta \mathcal{L}_{style}(\vec{a}\vec{x})$$

where α and β are the weighting factors for content and style reconstruction respectively.

Style Reconstruction describes the style reconstruction process from white noise. Performing gradient descent of the style loss on a white noise input x for a given artwork a yields a representation of the networks activation for a given set of layers L. To extract the representation of the style content, we build a feature space on the top of the filter responses in each network layer. It consists of the correlations between the different filter responses over the spatial

extent of the feature maps. The filter correlation of different layers captures the texture information of the input image. This creates images that match a given image's style on an increasing scale while discarding information of the global arrangement. Steps for the algorithm:

1. Compute the Gram matrix G by $V^T V$ where the columns of V are the vectorized feature maps for a particular layer.
2. Compute the loss between the gram matrix of the styling image and the gram matrix of the image undergoing optimization.
3. Set up input node and feature extractor
4. Set up restoring feature extractor weights
5. Load image and create to-be optimised image
6. Set up gram matrix nodes
7. Obtain gram matrices for real image
8. Calculate the loss for each layer Let and be the original images and the image that is generated, and A^l and G^l their respective style representation in layer l. The contribution of layer l to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum (G_{ij}^l - A_{ij}^l)^2$$

and the total style loss is

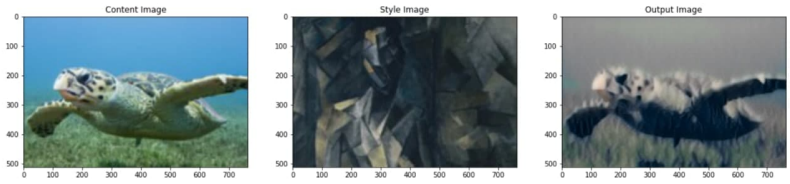
$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{i=0}^{L} w_l E_l$$

where w_l are weighting factors of the contribution of each layer to the total loss

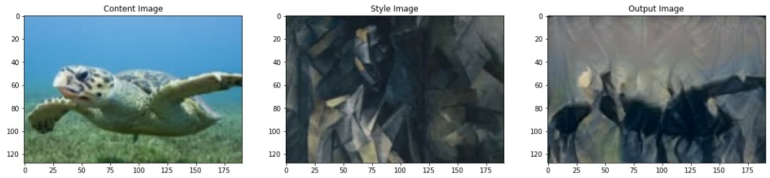
9. Set up optimizer, loss function

3 Results

Style Transfer If we use Adam optimizer then the final output looks more realistic to the art of artist whose style we tried to copywrite.



SGD

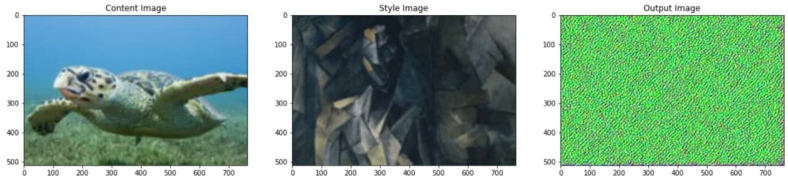


Adam

Adding more layers do not give a better result. It becomes much worse. (Here I added 3 more leayers)

```
Building the style transfer model..
Optimizing..
run [100]: Style Loss : 24997116.000000 Content Loss: 439.914246
run [200]: Style Loss : 12943774.000000 Content Loss: 330.756958
run [300]: Style Loss : 11101772.000000 Content Loss: 313.066162
run [400]: Style Loss : 1368549504.000000 Content Loss: 2120.258057
run [500]: Style Loss : 32696660.000000 Content Loss: 500.322113
run [600]: Style Loss : 15160137.000000 Content Loss: 354.615967
run [700]: Style Loss : 10735396.000000 Content Loss: 306.472412
run [800]: Style Loss : 161575648.000000 Content Loss: 973.164124
run [900]: Style Loss : 39659068.000000 Content Loss: 544.752441
run [1000]: Style Loss : 17375792.000000 Content Loss: 376.941162
```

Performance

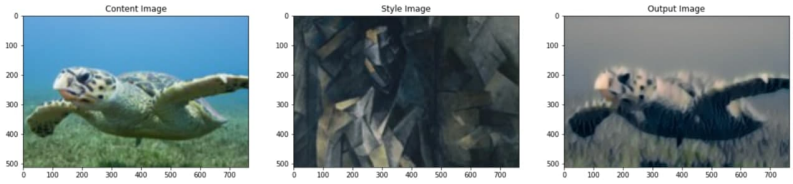


Output

Decreasing layers would show almost the same result as origin number of layers (Removed 2 layers)

```
Building the style transfer model..
Optimizing..
run [100]: Style Loss : 335.635254 Content Loss: 1.353561
run [200]: Style Loss : 266.210236 Content Loss: 1.478559
run [300]: Style Loss : 225.366150 Content Loss: 1.581075
run [400]: Style Loss : 197.908630 Content Loss: 1.653480
run [500]: Style Loss : 178.411972 Content Loss: 1.703268
run [600]: Style Loss : 163.338669 Content Loss: 1.736927
run [700]: Style Loss : 150.930756 Content Loss: 1.760454
run [800]: Style Loss : 140.314728 Content Loss: 1.777241
run [900]: Style Loss : 131.044891 Content Loss: 1.789942
run [1000]: Style Loss : 122.823410 Content Loss: 1.800234
```

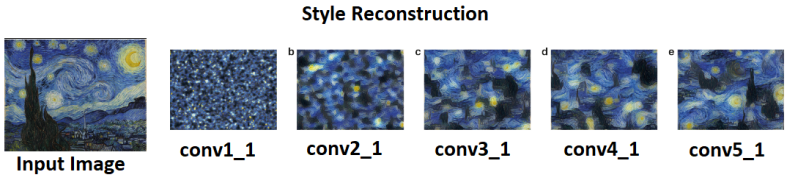
Output



Output

Also I changed VGG19 to VGG16 to see if there will be any difference and results were surprising. NN performed better on VGG16 even tho it has less convolutional layers.

Style Reconstruction



Output

Picture above shows the feature space above each convolutional layer representing the correlation between different features in different layers of CNN. As we move deeper into the network, we can see that the global arrangement or the structural features are discarded.

4 Discussion

Style Transfer To transfer the style of an artwork onto a photograph we synthesise a new image that simultaneously matches the content representation of and the style representation of . Thus we jointly minimise the distance of the feature representations of a white noise image from the content representation of the photograph in one layer and the style representation of the painting defined on a number of layers of the Convolutional Neural Network. To extract image information on comparable scales, we always resized the style image to the same size as the content image before computing its feature representations.

Style Reconstruction To obtain a representation of the style of an input image, we use a feature space designed to capture texture information. This feature space can be built on top of the filter responses in any layer of the network. It consists of the correlations between the different filter responses, where the expectation is taken over the spatial extent of the feature maps. These feature correlations are given by the Gram matrix G^l. By including the feature correlations of multiple layers, we obtain a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement. For visualizing the information captured by these style feature

spaces use gradient descent from a white noise image to minimise the mean-squared distance between the entries of the Gram matrices from the original image and the Gram matrices of the image to be generated/

References

[1] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge1 "A Neural Algorithm of Artistic Style"

[2] Style Reconstruction Info

[3] Simonyan, K. Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition.(2014).

[4] Jia, Y. et al. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia, (ACM, 2014).

[5] Guc¸lu, U. Gerven, M. A. J. v. Deep Neural Networks Reveal a Gradient in the Complexity of Neural Representations across the Ventral Stream.

[6] Krizhevsky, A., Sutskever, I. Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems