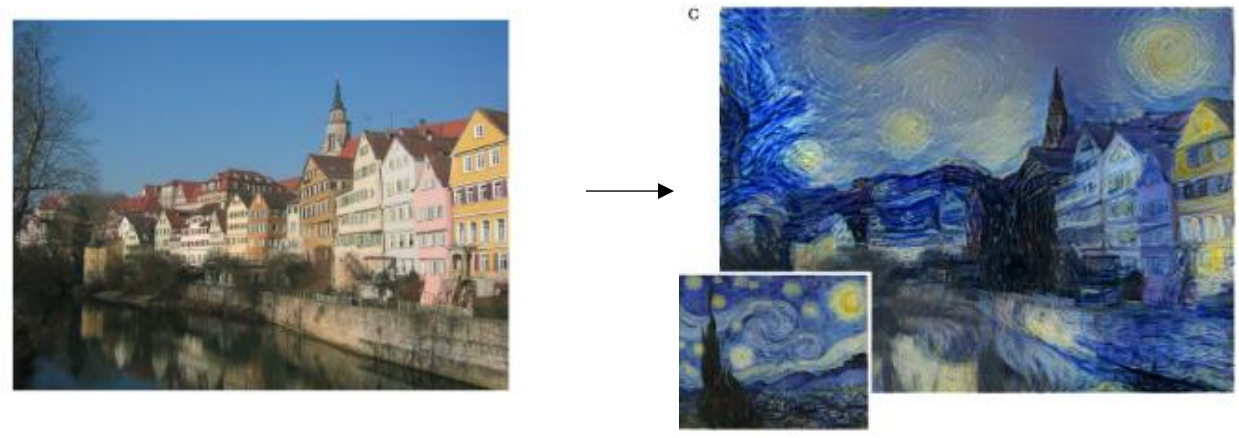
Neural Style Transfer

Student: Surdoiu Tudor Marian

This paper is a presentation of a computer vision application that I find rather interesting called Neural Style Transfer. The algorithm that stands at the base of this concept was published in the paper "A Neural Algorithm of Artistic Style" [1] in 2015. The practical usage of the technique presented in the aforementioned document is very limited but the results can seem quite fascinating, or at least hey did to me the first time I saw them.

From an abstract point of view, this Neural Style Transfer allows us to apply the style of a picture to another one, in the case presented in the paper, to apply the style of the work of some well known painters to ordinary pictures. For example:



This technique mainly uses a Convolutional Network to accomplish this transformation, not by training one from zero but by using almost any network that has decent number of hidden layers so that it can extract the desired features.

Due to the way that a Convolutional Network works, an image that is used as an input is convolved with several filters at every layer, plus other transformations (max pooling, fully connected layer) and thus resulting the

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

corresponding feature maps. In the lower layers the maps look for simpler and low level features such as lines, edges, circles and others. The upper layers combine the extracted features from the lower ones to detect increasingly more complicated features in a hierarchical manner. These types of low level and high level feature detection is the basis of style transfer.

As I mentioned earlier there is no need to train a Neural Network by changing the weights and values of its layers to optimize a cost function, in exchange we optimize the cost function by changing the pixels of a random generated image. The key is to define a cost function that can induce the transformation we target.

The loss function has two components:

- **Content loss**

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

P - representation of the original image
F - representation of the generated image
l – the layer at which we use the feature map

- **Style loss**

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l}\right)^2$$

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

A – representation of the artwork image
G – representation of the generated image
N – the number of feature maps
M – size of the flattened feature map

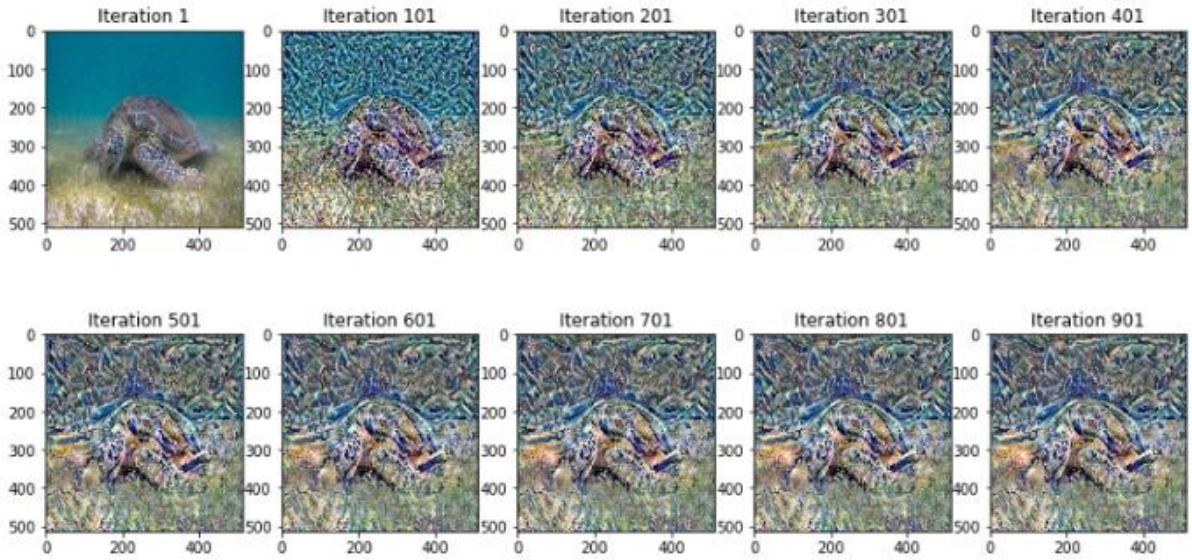
The G and A in the style loss are Gram matrices that represent the correlations between different filter responses. Calculated with: $G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$.

And the total loss 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 alpha and beta are weights for content and style. An easy way to alter the final result is to change those values.

The problem that we are trying to solve is to minimize the difference between the content of the final image and the input image and the difference between the style of final image and the style image. From the research that I have made any gradient descent style optimizer can be used, like: ADAM, Adagrad etc.

As we can see in the example below, multiple iterations are required to get a batter style transfer.



References:

- "A Neural Algorithm of Artistic Style", Leon A. Gatys, Alexander S. Ecker, Matthias Bethge (https://arxiv.org/pdf/1508.06576.pdf)
- Article: https://towardsdatascience.com/a-brief-introduction-to-neural-style-transfer-d05d0403901d
- Article: https://towardsdatascience.com/neural-style-transfer-applications-data-augmentation-43d1dc1aeecc