Neural Style Transfer

Neural style transfer is a computer vision technique consisting on applying the *artistic style* from one image into the content of another image. This way, the output image will have the intention of the artist who draw the content image, as if the artist who draw the style image painted it.



# How does it work?

The publication where neural style transfer was introduced is: <https://arxiv.org/pdf/1508.06576.pdf>

It is pretty much as applying Convolutional Neural Networks to an image. The trick resides on how the loss of the model is defined. In fact, the ***weights in NST are fixed***, which means we could use a pretrained model and never update its weights.

**So, what do we maximize?** In NST we maximize the output with respect to the input itself!

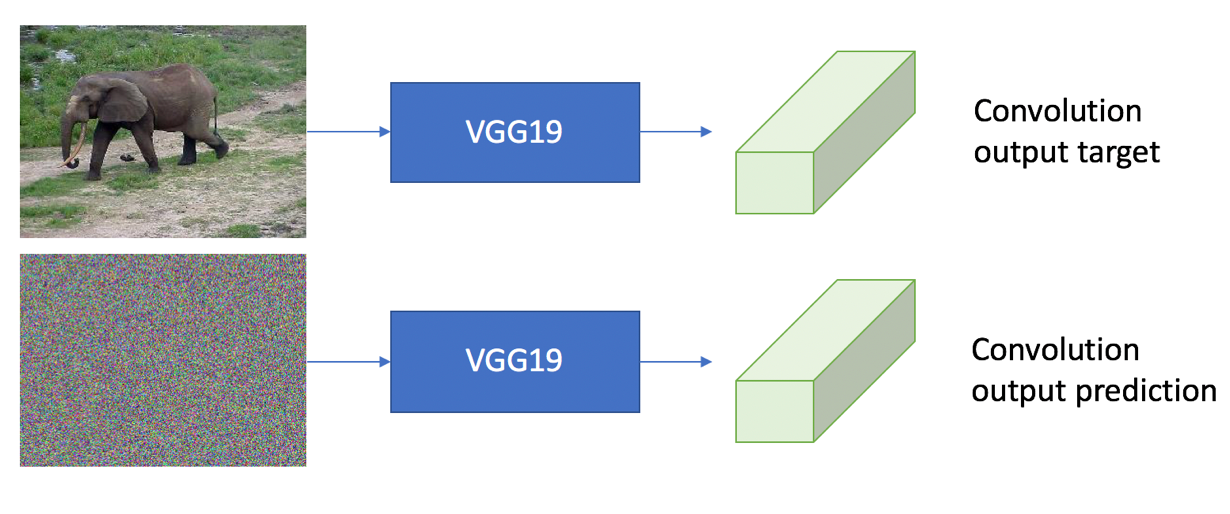
We know that a CNN allow us to have a representation of the image into what we call feature maps, that can actually be the size we desire. Therefore, what we are looking is for another image that have the convolved outputs as the original images.

|  |
| --- |
|  |

This would be equivalent to answering the question: what image maximizes these activations?

# Reproduce content

How could we reproduce the content of an image?



As mention, we are looking for the result of the convolution to be equivalent. In the figure above, our content image (the elephant) will produce an output after the forward pass on the CNN. We will then ***change the pixels in out input image*** (the noisy image) to ***produce the same output volume.***

And compute the loss of this aspect is trivial. We want to know “how far” both volumes are, so we could simple calculate the MSE for the vector resulting of flatten those volumes.

|  |
| --- |
|  |
|  |
|  |

### NOTE

We now could think about how CNN works to take more advantage of using it in this task. As we know, CNN propagate representations of the image over the depth dimension, layer-by-layer. This means, that the further we go from the input image, more detail from the image is represented by the output of that particular layer.

Since right now we want to capture the content of the image, we are ***not actually interested in using the entire network***, since we will lose precision in the content encoding. Therefore, the above equations will be more accurate by changing:

|  |
| --- |
|  |
|  |
|  |

The next picture shows how out prediction image looks if we convolve the content image until the last convolutional layer. We can see how the main features are still kept, and we have *“made room”* for the style matching.

Note that if we recreate after passing the content image only until the first convolution, it will be very sharp, close to the actual content image. So, the content will be kept, but there wouldn’t be that space for the style.



# Reproduce Style

We have already cover how are we going to attempt to replicate the content of a target image. But, at the same time we keep track of replicating the style of the style image. In fact, if we were only to replicate the output using the input image, why did we bother to use a CNN in the first place?

As we saw before, the deeper we go into the CNN the more detail we lose. However, this is the ***key point for extracting the style of an image***! We want to capture the style, we are not looking at the details at pixel level, we are looking for more general representations of the image.

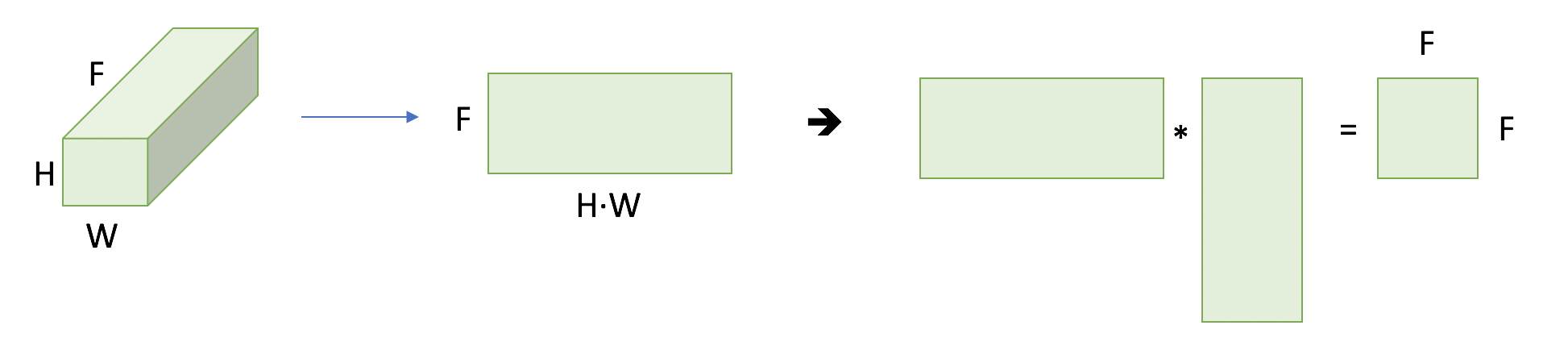
### Gram Matrix

We need to explain the concept of the gram matrix in order to understand how to reproduce the style.

|  |
| --- |
|  |

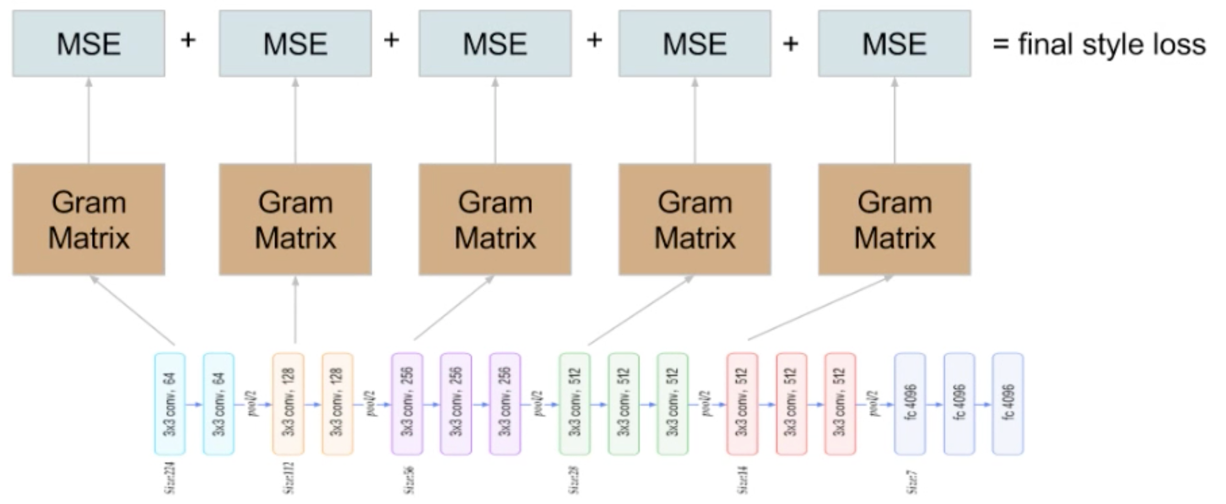
The gram matrix G is computed by multiplying the input image by its transpose. The intuition behind this matrix is that it contains non-localized information about the images, such as textures and other features that conform the style of an artist. The gram matrix formulation suggest that it is computing the ***autocorrelation***. This means, how correlated are the input with itself.

But our convolution output is a 3D volume. We need to do a transformation to be able to convert it to a 2D matrix. Therefore, we flatten the volume along the spatial dimension.



The resulting gran matrix becomes size (Features x Features). Interestingly, the image size dimensions disappear. This agrees with the statement we mention before. When it comes to texture, we don’t care about the localization on the image, it must be consistent at every location on the image.

In the original paper, they took the output at 5 different locations of the VGG:



By doing this, it allows us to capture patterns of different sizes, since they are at different depth of the CNN. So, to compute the final loss of the style, we could simply make a weighted average of the errors at every defined output convolution.

### NOTE

These are one of the hyperparameters we have to decide for the architecture of our style transfer model. The other ones are regarding how much importance we give to the content being well reproduced against the style, to compute the final global loss function.

|  |
| --- |
|  |
|  |

# Final Picture

