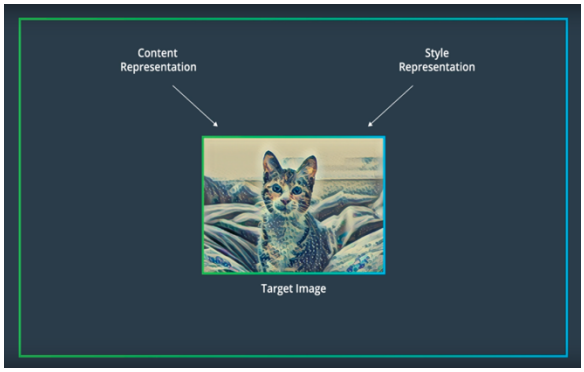
## Chapter 1. Style transfer

*CNN process visual information in a feedforward manner. Each image is passed through a collection of filters, which extract different low-level and high-level features. This process can be used not only in image classification but in image construction as well. It is a core component of applications such as Deep Dream and Style Transfer.*

*The idea behind Style Transfer is quite simple. It involves taking a pretrained CNN and using its convolutional layers (pretrained filters) to separate content from style. Then it is possible to merge the content of one picture with the style of another to create a new picture.*

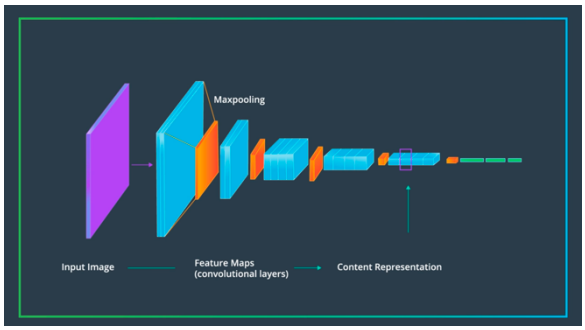




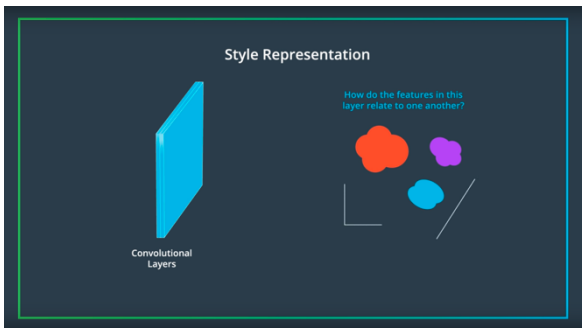
## Chapter 2. Content and style

*If the pretrained CNN has been trained to classify images, then the network already knows how to extract the content information of an image, because its original goal was to recognize some common object and name it. The details about how exactly that object is represented (colors, textures) are irrelevant for this task.*

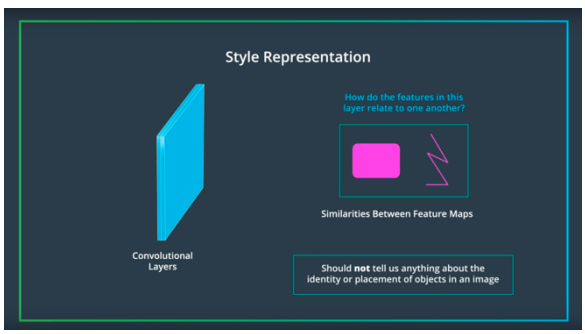
*As we go deeper into the network, we see that it cares more and more about content of the image rather than stylistic details. It assembles an **invariant** representation of an object, that means that if we take two stylistically-different images of one object and visualize their feature maps produced by later layers, those feature maps will be quite similar. Later layers are sometimes referred to as **content representation**.*




*First few convolutional layers of a CNN are usually trained to detect simple features such as edges, simple shapes, texture and colors. Hence, those layers are used to extract the stylistic information from an image. We can use the feature maps produced by these layers to detect common colors and shapes which can be perceived as a part of image style. This is done via Gram matrix, which is explained in Chapter 5.*




*Style representation should leave out any content information.*

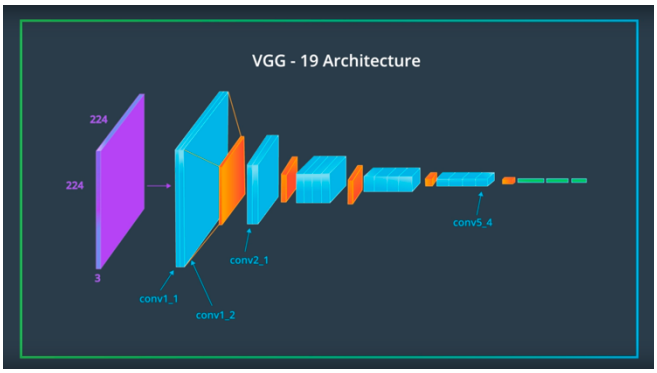



## Chapter 3. VGG architecture

*VGG-19 is a CNN that was trained on ImageNet dataset and showed a good performance. It has been made publicly available, so it is convenient to use this network in the style transfer applications. It consists of several stacked convolutional layers with maxpooling operation in between.*

*In the original paper four layers were used to extract the style: **conv1_1**, **conv1_2**, **conv2_1**, **conv5_4**; and one layer to extract the content: **conv4_2**. This particular choice seems arbitrary because there is no strict rule in regards to which layers to choose. We can try to use any layers to do either job*

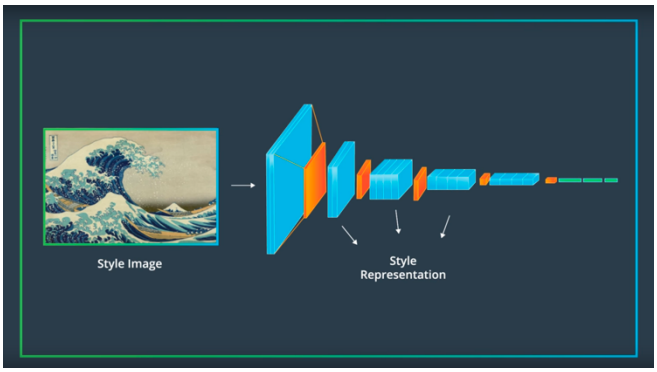
*(style or content extraction) with the rule of thumb presented in the chapter 2 in mind.*

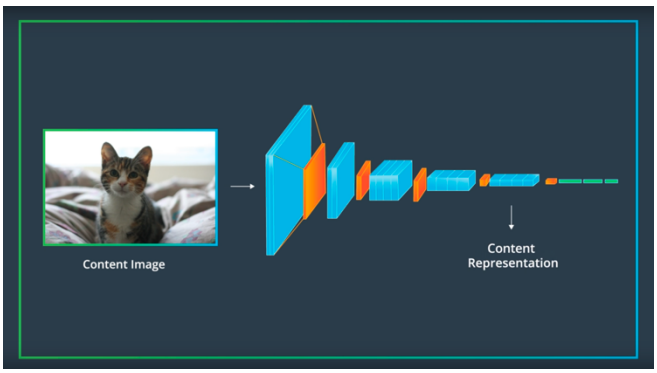



*To perform the style transfer both source images are passed through the network.*

*In order to get the style representation, we collect all the feature maps produced by **conv1_1**, **conv1_2**, **conv2_1**, **conv5_4**. Later those feature maps are used to construct Gram matrices (chapter 5) that will detect specific stylistic information.*

*The content representation is just our feature maps produced by the **conv4_2** without any modifications.*



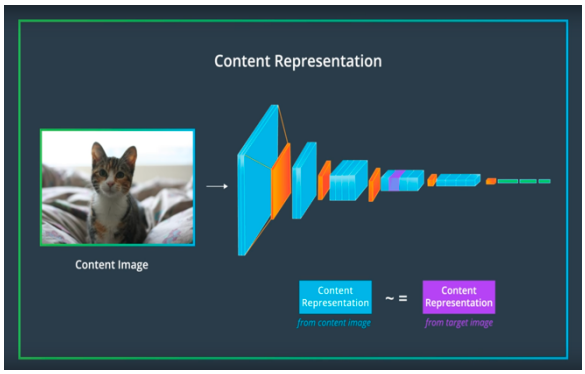




## Chapter 4. Content loss

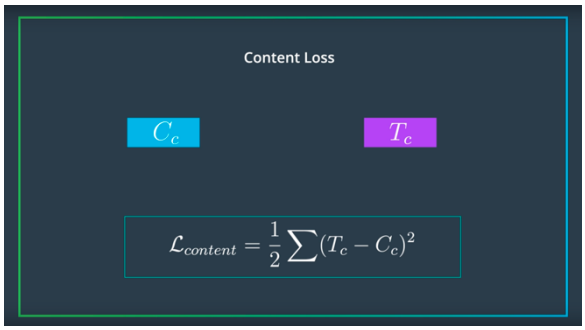
*In the style transfer we have two objectives:*

1. *(1) adjust the content of a new (target) image, so it is close to the content image*
2. *(2) adjust the style of a new (target) image, so it is close to the style image*

*So, the first objective is to minimize the content loss:*




*The content loss is basically a squared loss between the two content representations computed pairwise:*

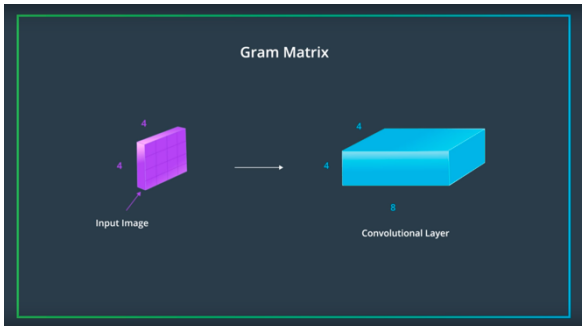



## Chapter 5. Style representation and Gram matrix

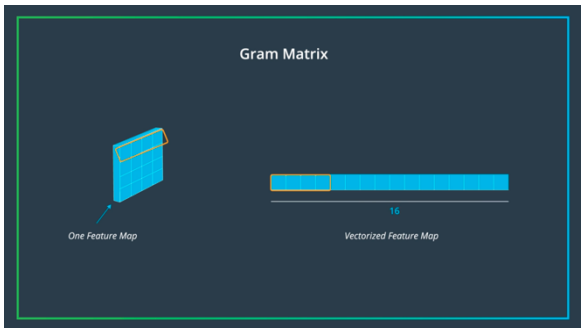
*How to ensure that the target image has desired style? To rephrase the question, how to measure how close are two images in their style? The first step is to understand how exactly the style is represented.*

*The style is represented via the Gram matrix which captures the correlations between different feature maps in one layer. As was said in the chapter 2, repetitive colors, shapes and textures can be seen as an image style.*

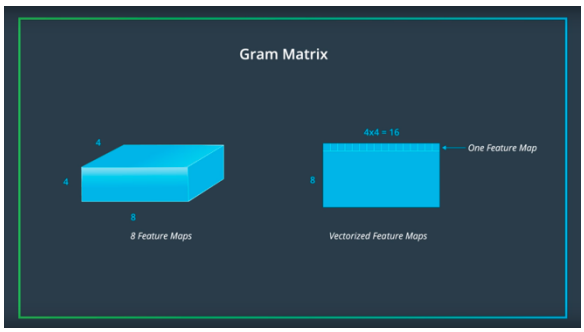
*One layer will have its own Gram matrix. This matrix is computed using all feature maps in that layer (number of feature maps is equal to the number of layer filters).*



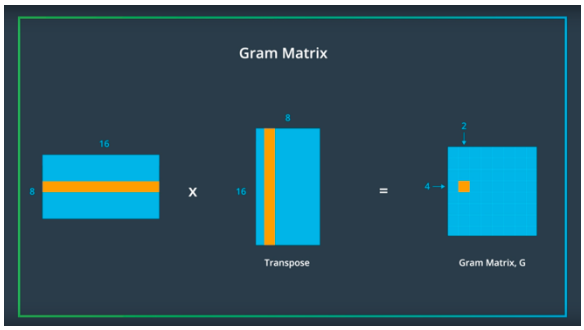
*The first step is to flatten each feature map into a vector:*




*Then stack all flattened feature maps together:*




*Gram matrix is obtained by multiplying the resulting matrix by its transpose:*



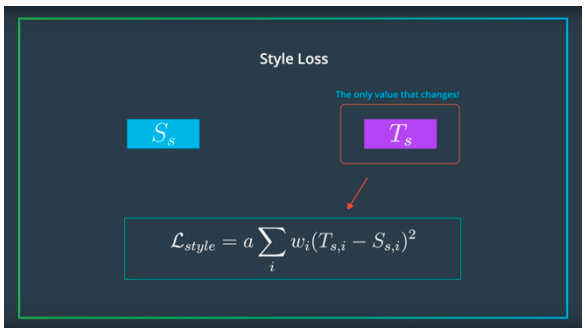

*Entries of the Gram matrix indicate correlations between feature maps. For example, the entry in the fourth row and the second column will indicate the similarities between fourth and second feature maps.*

*Gram matrix contains non-localized information about a layer, this information would still be there if image is shuffled (e.g. prominent colors and shapes will still be detected).*

## *Chapter 6. Style loss*

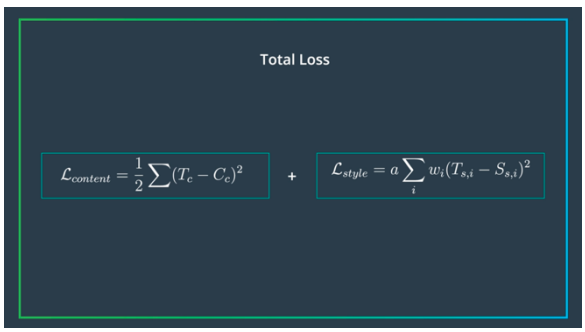
*After computing Gram matrices, we can compute the style loss, which indicates the stylistic similarity between two images. Returning to the VGG-19 example, we will have 4 different pairs (one for the style image and one for the target image) of Gram matrices for each layer chosen for style representation.*

*The loss is again a squared difference between those Gram matrices pairs. The formula for the style loss is given below, where **a** is a constant that accounts for the number of values in each layer, and **w** is a style weight, which helps us to control how much influence each style layer will have on the target picture.*

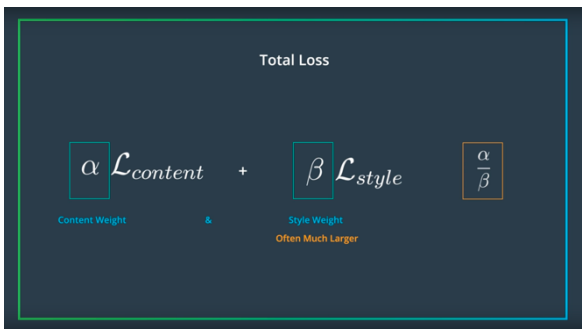



## *Chapter 7. Total loss*

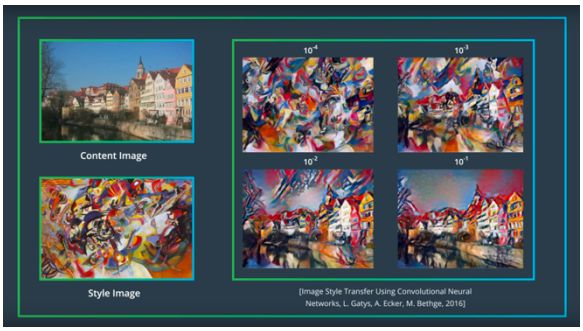
*The total loss is calculated by summing up the style loss and the content loss. The total loss will be optimized using backpropagation and gradient descent. It is important to understand, that in the case of the style transfer we are not training the network. The actual goal is reversed: we are 'training' the input, by optimizing its values.*




*However, it is important to notice that those losses are calculated very differently, but we need to take both into account fairly equally. This is done through $\alpha$ and $\beta$ constants, which control the tradeoff between content and style.*



*Different $\alpha$ and $\beta$ ratios can be tried experimentally:*




***Author***. *Tatiana Gaponova, tatiana.gaponova@gmail.com*

***Course.*** *Intro to DL with PyTorch (Udacity)*