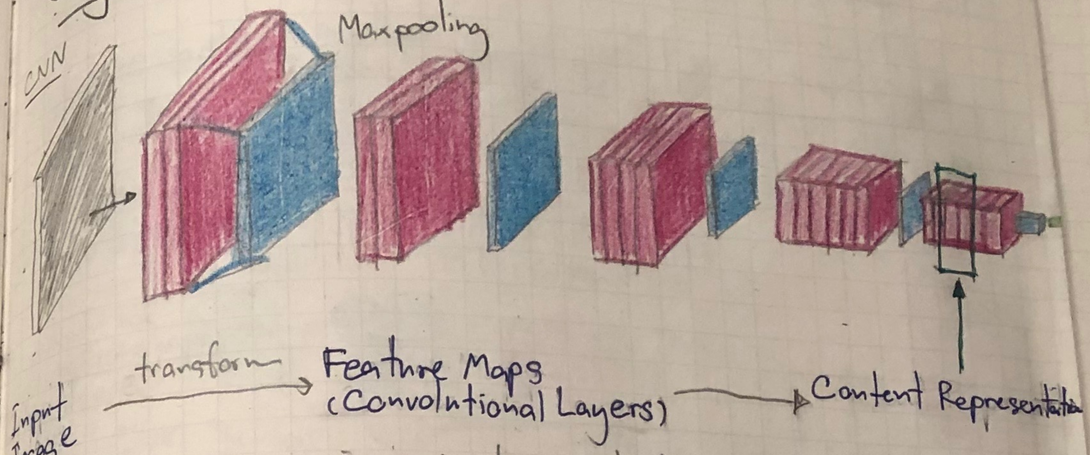
Style Transfer

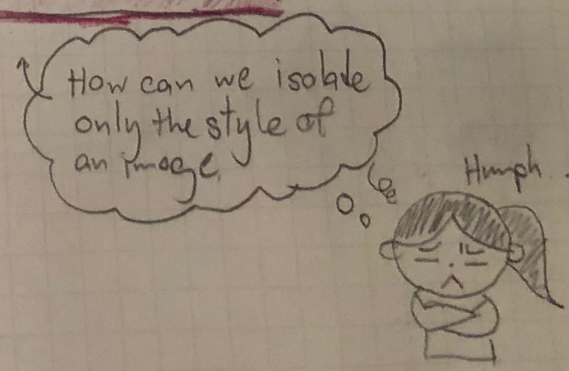


- Increasingly care about the CONTENT of the image
  - ↳ rather than any details about the texture & color pixels

- A trained CNN has already learned to represent the content of an image.

- How about the style
- Style transfer

CONTENT of one image + STYLE of Another image

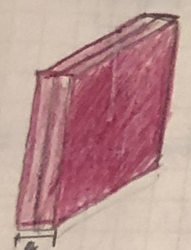


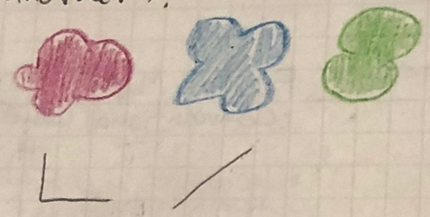
To represent the STYLE of an input image,

- Feature Space designed to capture texture and color information is used

↳ This space looks at spatial correlations within a layer of network

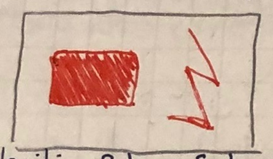
- How do the features in this layer relate to one another?





See which color & shapes in a set of feature maps are related & which are not

- For example, we detect that mini-feature maps in 1st conv layer have similar red edge features

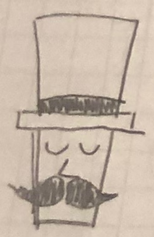


Similarities Between feature Maps

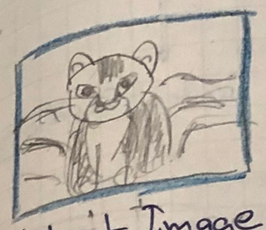
If there are COMMON colors and shapes among the feature maps

↳ Then, this can be thought of as part of that image style

- So the Similarities & Differences between features in a layer should give us some information about the texture and color information found in an image
- It SHOULD NOT tell us anything about the identity or placement of objects in an image

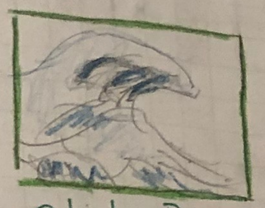


Style Transfer



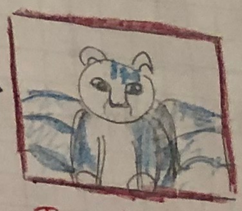
Content Image

object & shape arrangement



Style Image

Colors & textures



Target Image

From Paper: Image Style Transfer Using Convolutional Neural Networks

↳ VGG 19 network is used

↳ Accepts a 224 × 224 × 3 color image → Series of conv & pooling layers

↓

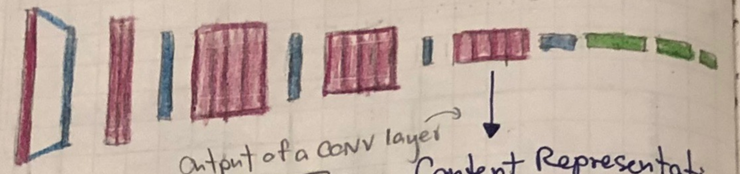
3 × Fully connected Layer to classify the passed image

↳ In between the pooling layers,

- there are stack of 2 or 4 convolutional layers
- Depth of these layers is standard within each stack
  - ↳ but increase after each pooling layer
  - ↳ conv1_1 → 1st CONV layer in the 1st stack
  - ↳ conv5_4 → Deepest CONV layer in network

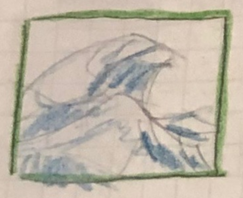
Feed Forward Process

Content Image

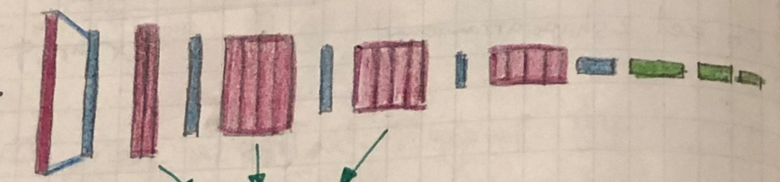


Output of a conv layer that is kept in the network

Content Representation



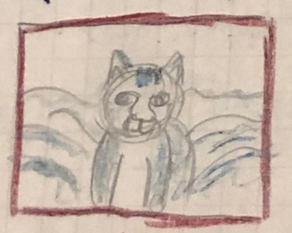
Style Image



Style Representation

- Extract different features from multiple layers that represent the STYLE of the image

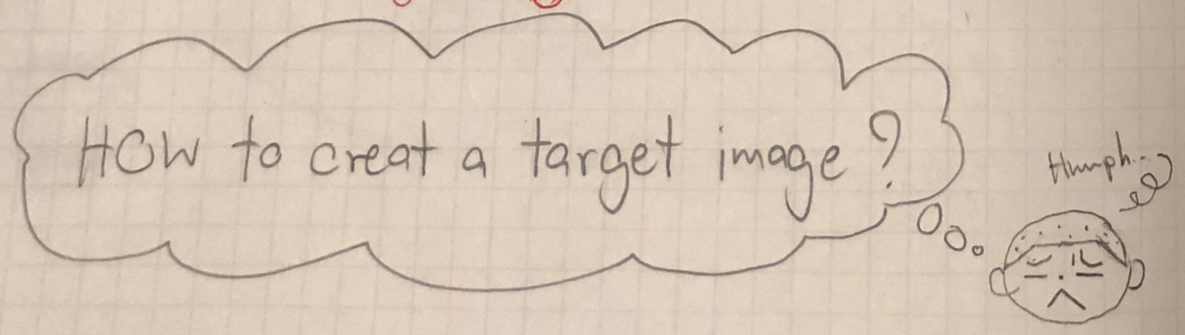
Finally,



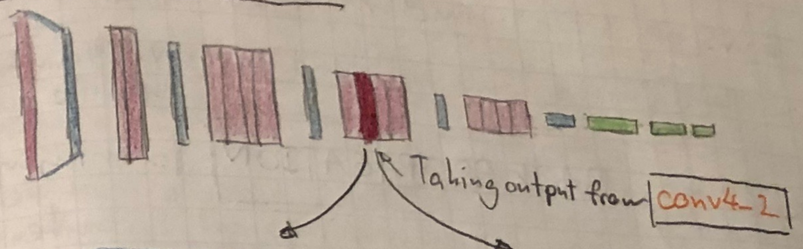
Content Representation

Style Representation

Target Image



Content Representation



Content Representation

from content image

≈= close

Content Representation

from target image

Content Loss

- A loss that calculates the difference between the content and target image representations

C_c

Content loss from content image

T_c

Content loss from target image

GOAL: Change ONLY target image
↳ update its appearance until it's content representation MATCHES that of our content image

Minimize

Mean squared difference between the two representations

Content Loss

$$L_{content} = \frac{1}{2}\sum(T_c - C_c)^2$$

- Measures how faraway these two representations are from one another

- To create the BEST target image

GOAL: Minimize Content Loss ($L_{content}$)

- Using a Pretrained network (VGG19) as a **FEATURE Extractor**
  - ↳ different from a traditional way → NOT training it to produce a specific output
- Using **BACKPROPAGATION** to minimize a defined loss function between our target & content images.

For Style

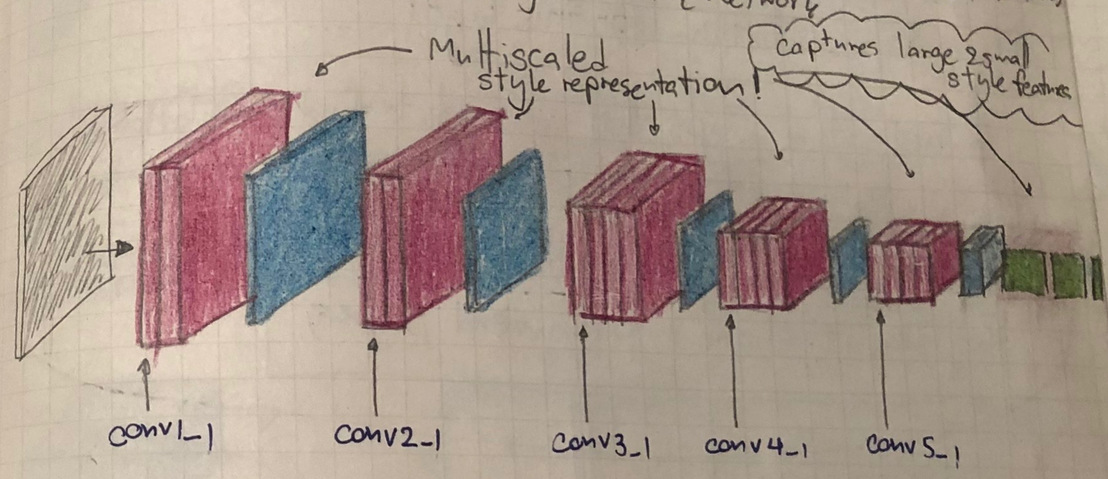
- We need to define a **Loss function** between our **TARGET** and **STYLE** images

↓

Produce an image w/ our DESIRED style.

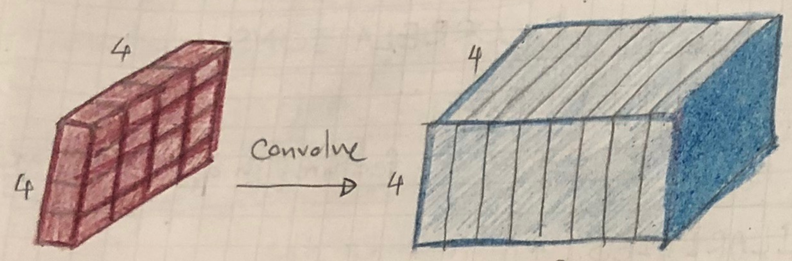
Style Representation

- Relies on looking at **CORRELATIONS** between the features in individual layers of VGG19
  - ↳ Looking at **how similar the features in a single layer are**
  - ↳ **SIMILARITIES** will include the general colors and textures found in that layer
    - ↳ we typically find the similarities between features in multiple layers in the network



- Style representation is calculated as an image passes through the network at 1st conv layer in all five stacks
- **Correlations** at each layer are given by a **GRAM MATRIX**

GRAM MATRIX

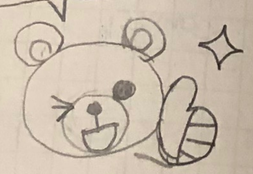


Input Image Convolutional Layer

Style Representation for this layer

↳ 8 Feature maps that we want to find the relationships between

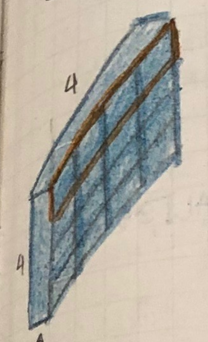
- GRAM Matrix is a mathematical way of representing the idea of share in prominent styles.



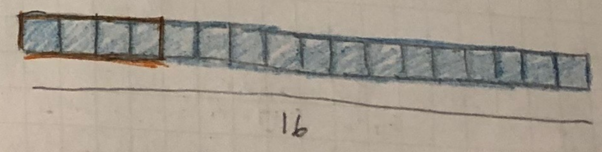
STEPS to calculate the Gram Matrix

① Vectorize the values in this layer

For One Feature Map



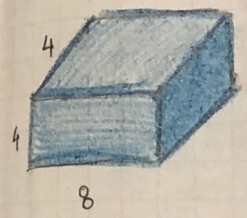
One Feature Map.



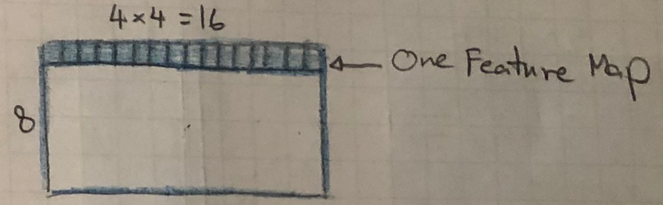
Vectorized Feature Map

For N Feature Map

- Converting 3D conv layer into a 2D matrix of values



8 Feature Maps



Vectorized Feature Maps

② Multiply this matrix (Vectorized Feature Maps) by its transpose

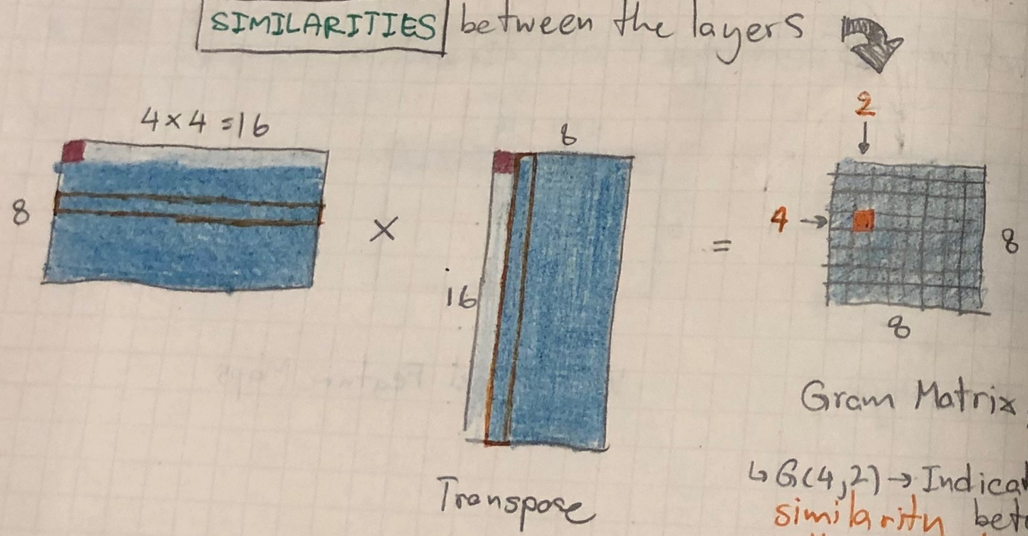
↳ Essentially, we multiply the features in each map to get the GRAM matrix

↳ This operation treats each value in the feature map as an individual sample, unrelated in space to other values

⇒ Resultant Gram Matrix contains NON-LOCALIZED information about the layer

Information that would still be there even if an image was shuffled around in space

⇒ Squared GRAM Matrix whose VALUES indicate the SIMILARITIES between the layers



↳ G(4,2) → Indicates the similarity between the 4th and 2nd feature maps in a layer

↳ Dimension will only related to ** feature maps in this conv layer

Style Loss

List of Gram Matrices from style image

List of Gram Matrices from target image

To compute style loss

- Find MEAN SQUARED Distance between the style and target Gram matrices
  - ↳ All 5 pairs that are computed at each layer in our predefined list conv1_1 up to conv5_1

S_s Style loss from Style image

T_s Style loss from Target image

The only value that changes!

Style Loss

$$L_{style} = a \sum_i w_i (T_{s,i} - S_{s,i})^2$$

where a: constant that accounts for ** values in each layer

w: style weight * give more or less weigh to the calculated style loss at each layer

Total Loss

$$L_{content} = \frac{1}{2}\sum(T_c - C_c)^2 \quad + \quad L_{style} = a\sum_i w_i(T_{s,i} - S_{s,i})^2$$

Steps:

- Compute TOTAL LOSS
- Then, use typical back propagation and optimization to reduce this loss
  - ↳ by iteratively changing the TARGET image to match our desired CONTENT and STYLE

We have values for the Content & style loss but, because they're calculated differently, These values will be pretty different

We want our TARGET IMAGE to take both into account fairly equally (total loss reflects an equal balance)

α $L_{content}$ + β L_{style}

Content weight × Style weight

IN PRACTICE ⇒ Often Much Larger!

$\frac{\alpha}{\beta}$

Bigger or smaller ratio have an impact to the target image

The more smaller $\frac{\alpha}{\beta}$, the more stylistic effect you will

Putting it all together.

- Extract features from the "content" image
- Calculate the "gram" matrices for each layer in our style presentation from the "STYLE" image
- Create a third "target" image
  - ↳ It is a good idea to start of w/ the target as a copy of our "content" image
  - ↳ Then, iteratively change its style

Loss and Weights

Individual Layer STYLE Weights

- Suggested that you use a range between 0-1 to weight these layers
- Weight EARLIER layers (conv1_1 and conv2_1) MORE

Get LARGER style artifacts

- If choose to weight LATER layers MORE

MORE Emphasis on SMALLER feature

Multi-scale style representation

Content (α) and Style (β) Weight

- The ratio $\frac{\alpha}{\beta}$ will affect how Stylized your final image is
- Recommended to
  - Leave $\alpha = 1$ (content weight
  - set the style weight β to achieve the ratio you want