Generative Modeling

By Eng.Ahmed Hisham

Agenda

1-Neural style transfer

-neural style transfer project (2x)

2-autoenconders

-MNIST auto encoder project

3-deep auto enconder

-deep mnist project

4- convolutional autoencoder

-fashion mnist project CAE

-case study between CAE and AE

5-Noisy convolutional autoencoder

-fashion mnist project CAE

6-Varitional autoencoders moving from past to future

-anime face generation using VAEs

Agenda

8-Introduction to GANs

-mnist using GANs

9-DCGANs

-fashion mnist DCGANs

-CelebA face generation

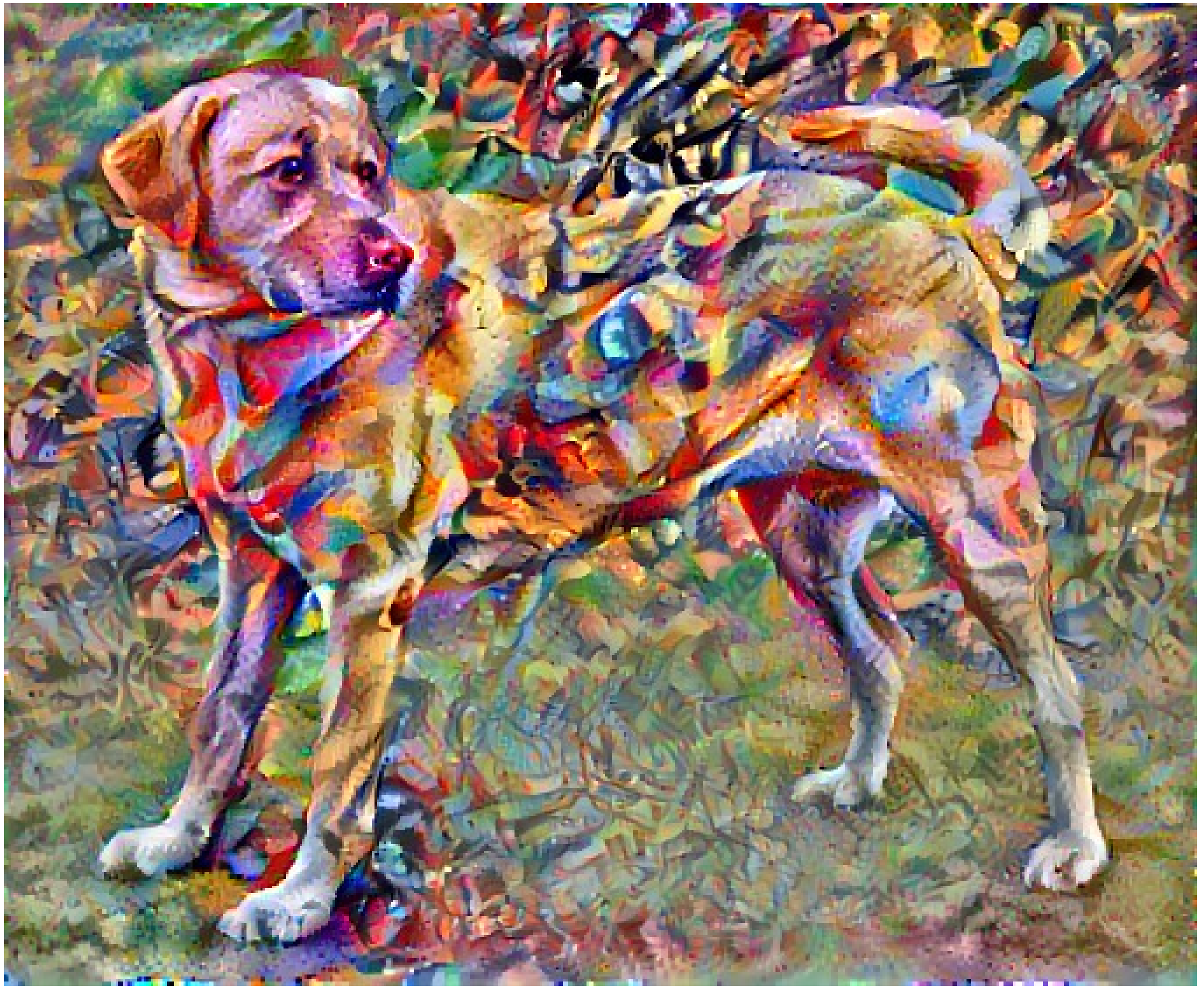
Neural style transfer

Neural style transfer is an optimization technique used to take two images—a content image and a style reference image (such as an artwork by a famous painter)—and blend them together so the output image looks like the content image, but “painted” in the style of the style reference image.

Neural style transfer (content image)

Neural style transfer style image

Blended image



How algorithm works

Step one

Pairs preparation

1-original image

2- style image

Step 2

Pretrained model for visual feature extraction

Ex VGG19, ReSNet50

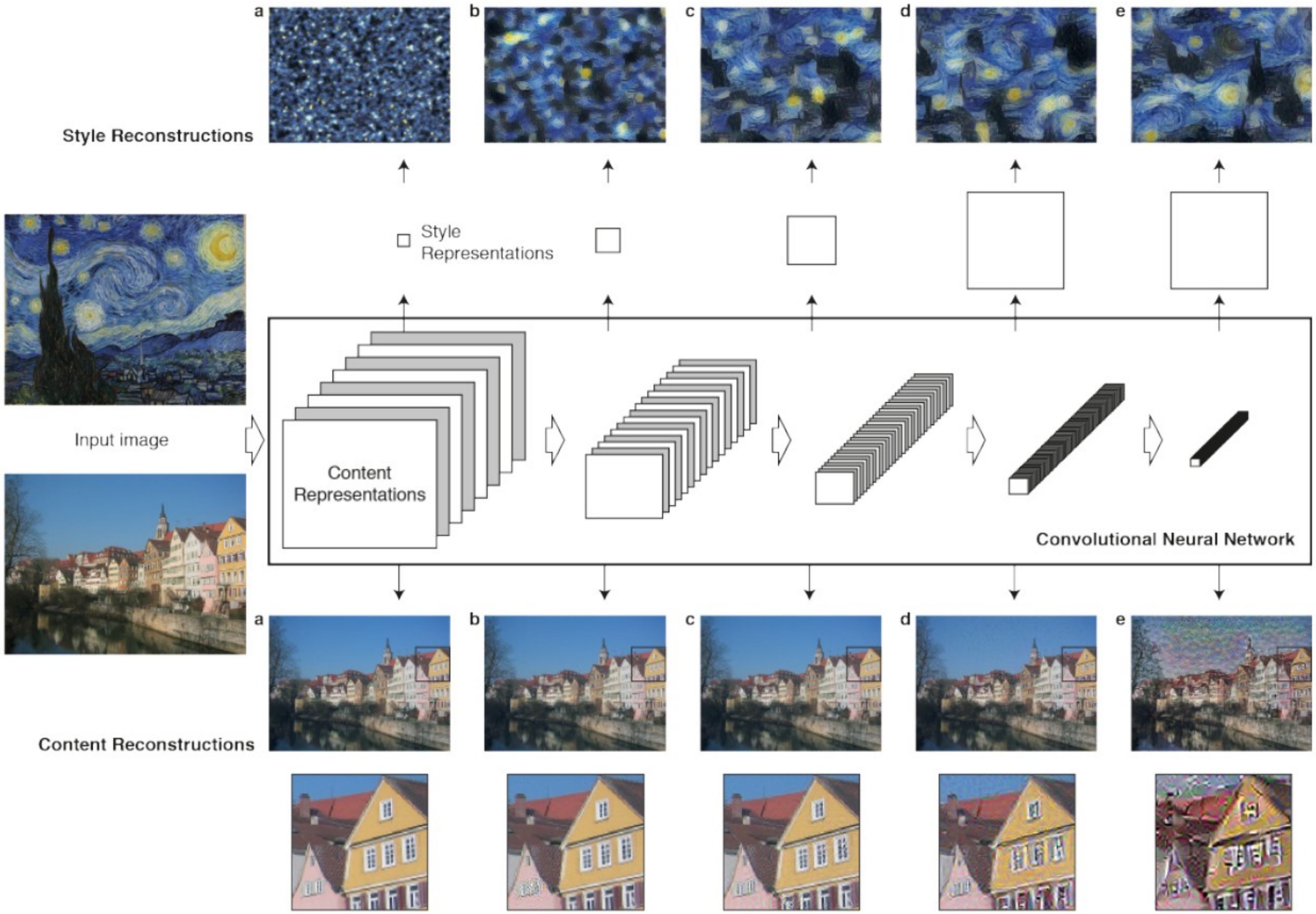
Step 3

Extract style from style image

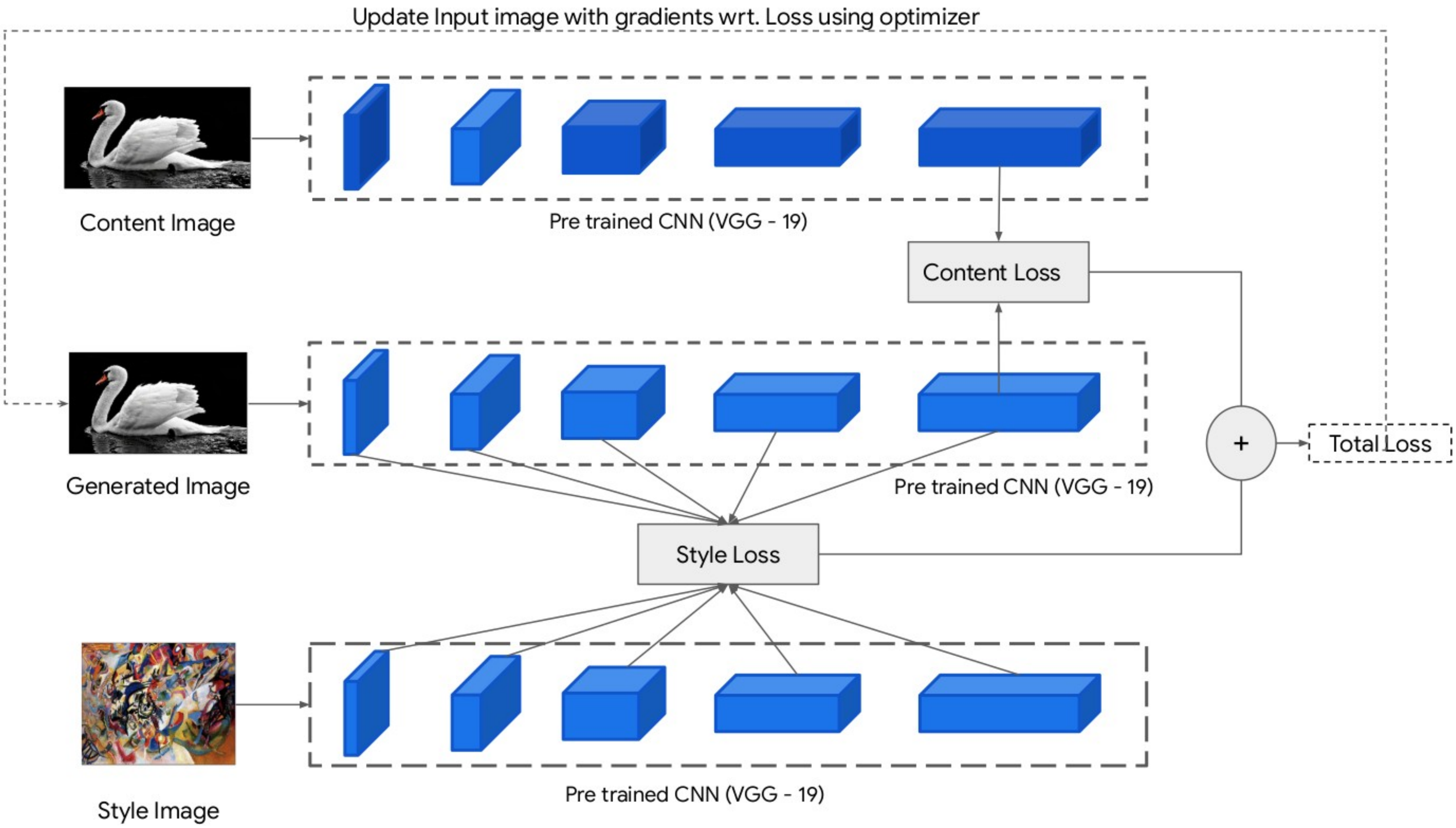
Extract content from content image

Step4

Generate an image to match the output


Style Reconstructions
a
b
c
d
e
Style
Representations
Input image
Content
Representations
Convolutional Neural Network
Content Reconstructions

architecture



Important notes

1-Content loss

Content loss

It helps to establish similarities between the content image and the generated image.

It is intuitive that higher layers of the model focus more on the features present in the image i.e. overall content of the image.

Content loss is calculated by Euclidean distance between the respective intermediate higher-level feature representation of input image (x) and content image (p) at layer l.

Important notes

Preprocessing stage

1-converts a tensor to an image

2-loads an image as a tensor and scales it to 512 pixels

3-loads the content and path images as tensors

4-Pretrained model to extract features VGG19

-style layers of interest
[conv1(block1),conv1(block2),conv1(block3),conv1(block4),conv1(block5)]

Important notes

the content layer of interest :

block5_conv2

output_layers = style_layers + content_layers

Important notes content loss

$$L^l_{content}(p, x) = \sum_{i,j} (F^l_{ij}(x) - P^l_{ij}(p))^2$$

Important notes content loss

f- content representation in the generated image

p- content representation in the content image

In code features will refer to generated image , while variable targets will refer to content image

Content loss

Generated image   Content image

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} - \begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix} = \begin{bmatrix}1-2 & 2-2\\ 3-2 & 4-2\end{bmatrix}$$

Element-wise subtraction

$$\begin{bmatrix}(1-2)^2 & (2-2)^2\\ (3-2)^2 & (4-2)^2\end{bmatrix}$$

Element-wise square

$$1^2 + 0^2 + 1^2 + 2^2 = \boxed{6}$$

Reduce sum

$$(\tfrac{1}{2})\,6 = \boxed{3}$$

weight

Content loss function as logic

Inputs /

features: tensor with shape: (height, width, channels)

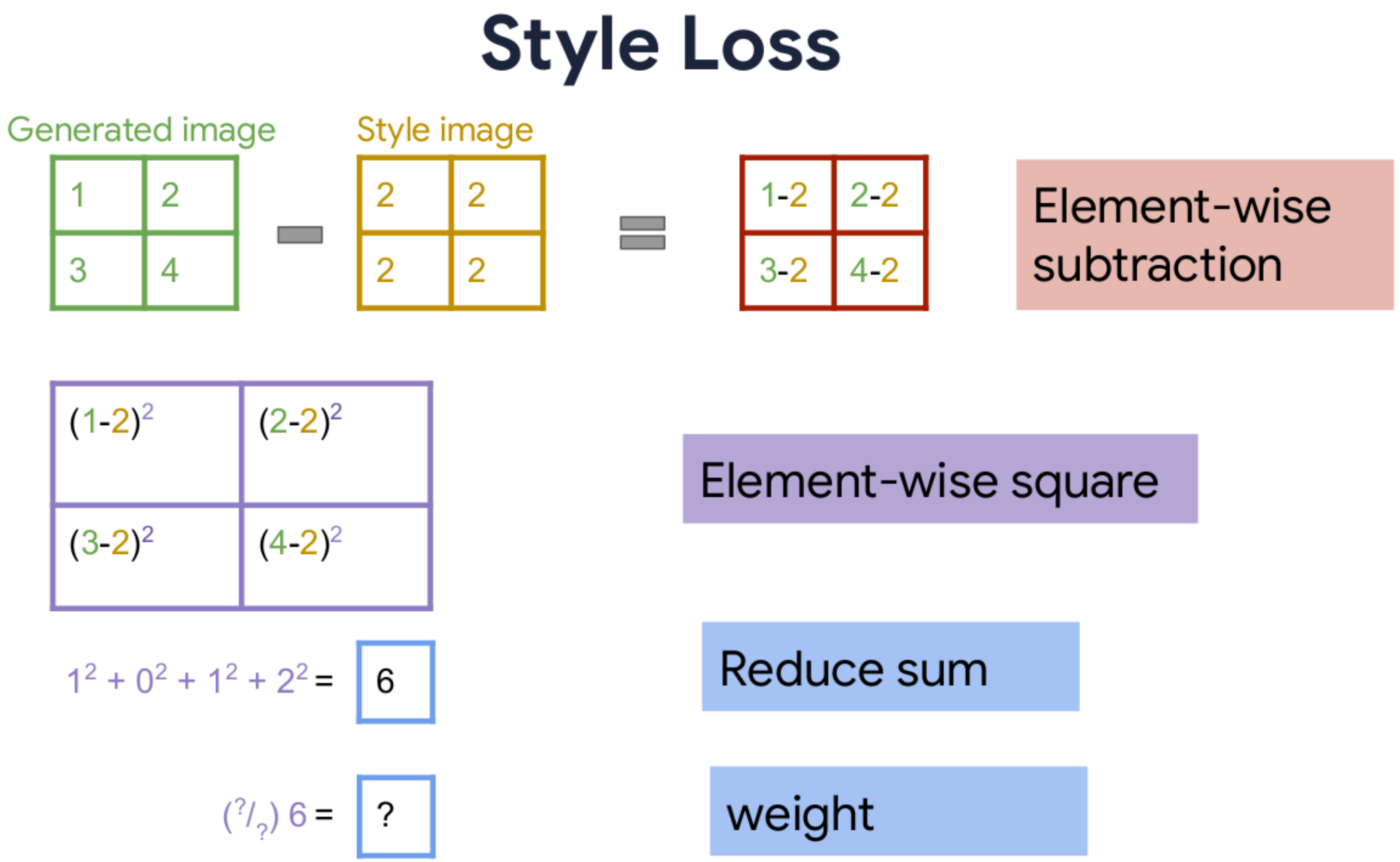
targets: tensor with shape: (height, width, channels)

Mathematical calculation

get the sum of the squared error multiplied by a scaling factor

Return content loss (scalar)

Get style loss

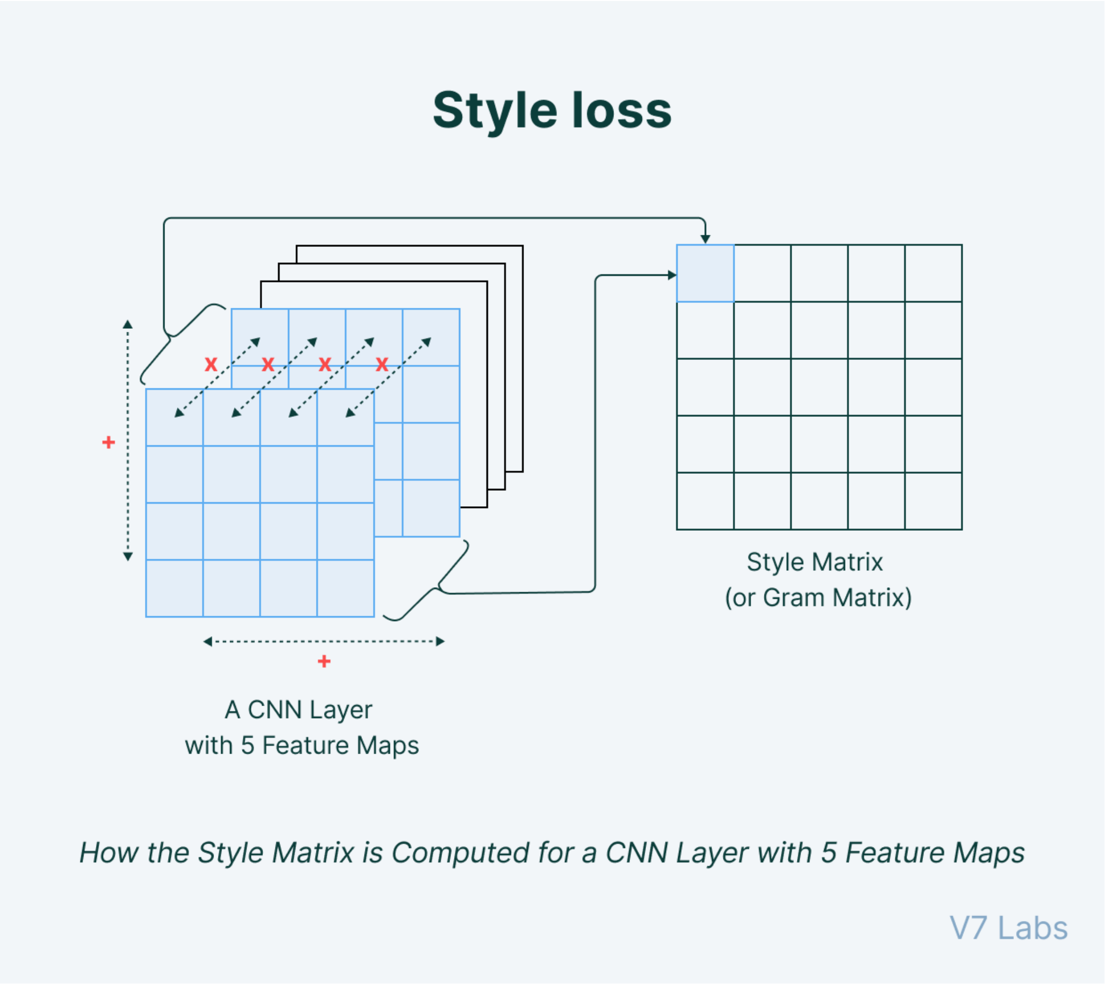


Style loss

Style loss is conceptually different from Content loss.

We cannot just compare the intermediate features of the two images and get the style loss

Style loss


Style loss
Style Matrix
(or Gram Matrix)
A CNN Layer
with 5 Feature Maps
How the Style Matrix is Computed for a CNN Layer with 5 Feature Maps
V7 Labs

GRAM matrix to solve style loss dimensionality

Style loss is calculated by the distance between the gram matrices (or, in other terms, style representation) of the generated image and the style reference image.

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

Thus, the total style loss across each layer is expressed as:

$$L_{style}(a,x) = \sum_{l \in L} w_l E_l$$

Style loss equation

Style Loss

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

l: layer *l*

A^l_{ij} : Style Representation(Gram Matrix) of style image a.

G^l_{ij} : Style Representation(Gram Matrix) of generated image x

Total loss

Total Loss

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

α : Content Weight

β : Style Weight

ImportantNotes gradient

Calculate the gradients of the loss with respect to the generated image

Arguments needed

- image: generated image
- style_targets: style features of the style image
- content_targets: content features of the content image
- style_weight: weight given to the style loss
- content_weight: weight given to the content loss
- var_weight: weight given to the total variation loss

ImportantNotes update image with style

image: generated image

style_targets: style features of the style image

content_targets: content features of the content image

style_weight: weight given to the style loss

content_weight: weight given to the content loss

var_weight: weight given to the total variation loss

optimizer: optimizer for updating the input image

Style fitting

style_image: image to get style features from

content_image: image to stylize

style_targets: style features of the style image

content_targets: content features of the content image

style_weight: weight given to the style loss

content_weight: weight given to the content loss

var_weight: weight given to the total variation loss

optimizer: optimizer for updating the input image

epochs: number of epochs

steps_per_epoch = steps per epoch

Neural style transfer ended

Check colab

Do the same colab neural style transfer using inception instead of vgg19