NEURAL IMAGE STYLE TRANSFER

Group 16 - Mingqian Liu, Xin Xiang, Tianqi Zhou, Yanfeng Zhang

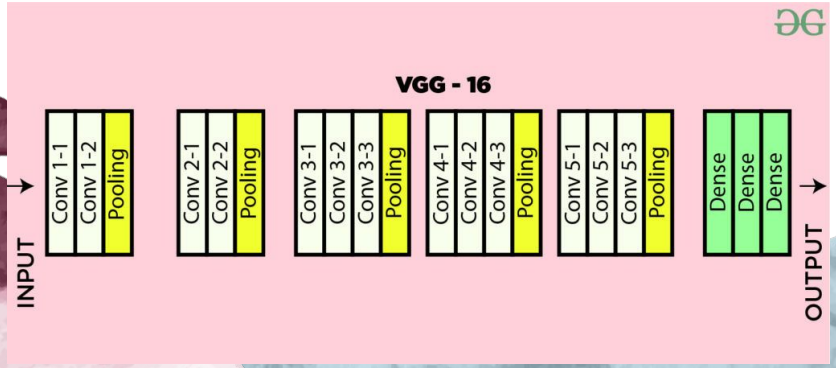
TABLE OF CONTENTS

PROJECT INTENTION

WHAT IS VGGNET?

VISUAL GEOMETRY GROUP

It is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The **"deep"** refers to the number of layers with **VGG-16 or VGG-19** consisting of 16 and 19 convolutional + fully connected layers.



IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC)

- ★ **Object localization**
- ★ **Image classification**

VGG 16 was proposed by Karen Simonyan and Andrew Zisserman. This model won **1st** and **2nd** place in the above categories in the **2014 ILSVRC** challenge.

WHAT IS TRANSFER LEARNING?

TRANSFER LEARNING

Transfer learning is the reuse of a **pre-trained** model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another.
Example: use the knowledge of detecting food to recognize drinks.

How to implement VGGs to ...

- ★ **Utilize the ability to detect items**
- ★ **Exploit the strengths of condensing image information**

VGG + **Transfer Learning** = **Neural Image Style Transfer**

WHAT IS NEURAL IMAGE STYLE TRANSFER?

Learn the **painting style** of the **style image** and apply it to the **content image** as filters.

- ★ Content image
- ★ Style image

The output should maintain the information of the content image, but painted in the style of the style image!

CONTENT IMAGE

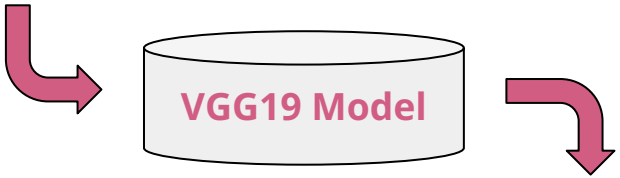
STYLE IMAGE

IMAGE PREPROCESSING

Image Preprocessing Steps for VGG19

Normalization steps:

- Resizing to 224x224 pixels
- Normalization (mean RGB values: [123.68, 116.779, 103.939])
- BGR Conversion



Why do we need to preprocess data in the same way?

Denormalization steps:

- Adding the Mean Values
- Converting BGR back to RGB
- Rescaling Pixel Values

Image Preprocessing Steps for VGG19

Why do we need to preprocess data in the same way?

- Ensuring **Model Compatibility**
- Importance of **Feature Scaling**
- Replicability of **Model Performance**

VGG19 Style layers choice:

- ‘block1_conv1' → captures **most basic and intricate** details
- 'block2_conv1' → captures **basic forms and structures**
- 'block3_conv1' → ...
- 'block4_conv1' → captures the **broad strokes and larger patterns** that constitute the style
- 'block5_conv1'

★ **micro-level** details
★ **macro-level** abstract patterns


LOSS FUNCTION

$$L_{total}(i,j,k) = \alpha L_{content}(i,k) + \beta L_{style}(j,k)$$

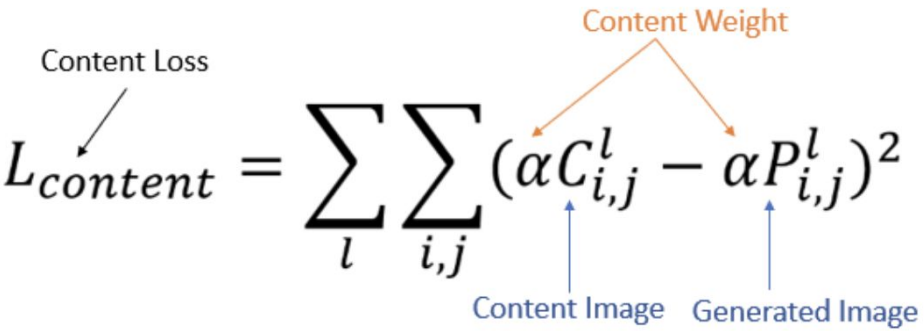
Content Weight

Style Weight

Hyperparameters

CONTENT LOSS

This function helps to check how similar the stylized image is to the content image. It gives the measure of how far (different) are the features of the content image and stylized image. The Euclidean distance is calculated.



EXAMPLE OF LARGE CONTENT LOSS

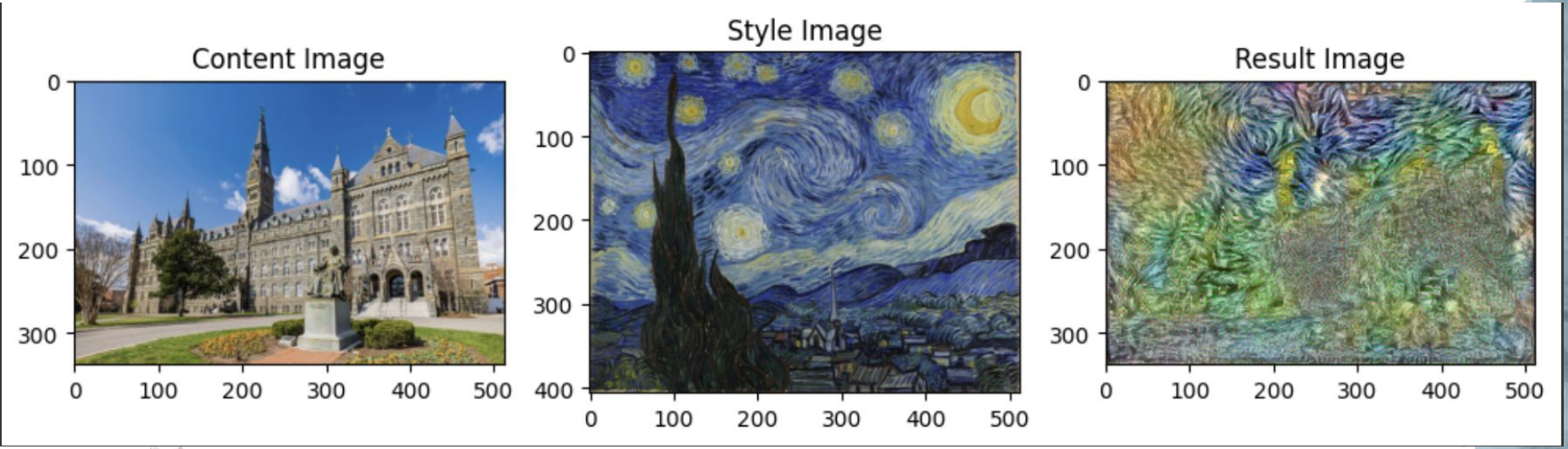
Content Loss

$$L_{content} = \sum_{l} \sum_{i,j} (\alpha C_{i,j}^{l} - \alpha P_{i,j}^{l})^2$$

Content Weight

Content Image

Generated Image



EXAMPLE OF SMALL CONTENT LOSS

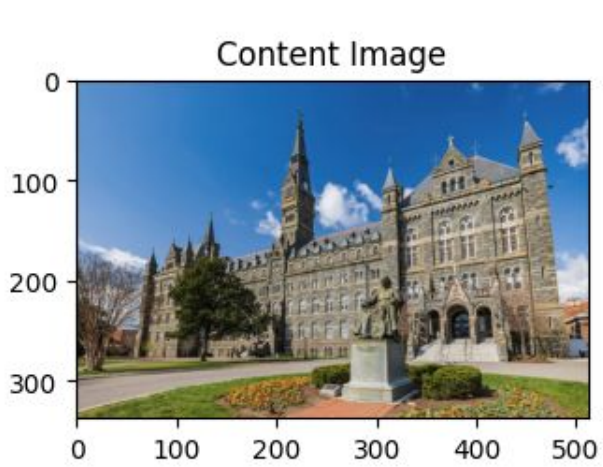
Content Loss

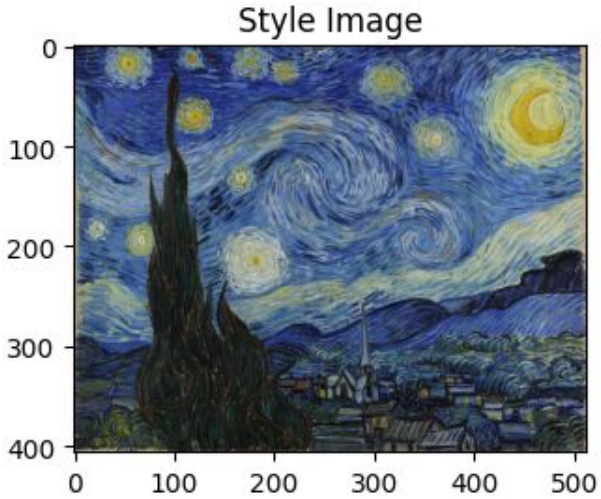
$$L_{content} = \sum_{l} \sum_{i,j} (\alpha C_{i,j}^{l} - \alpha P_{i,j}^{l})^2$$

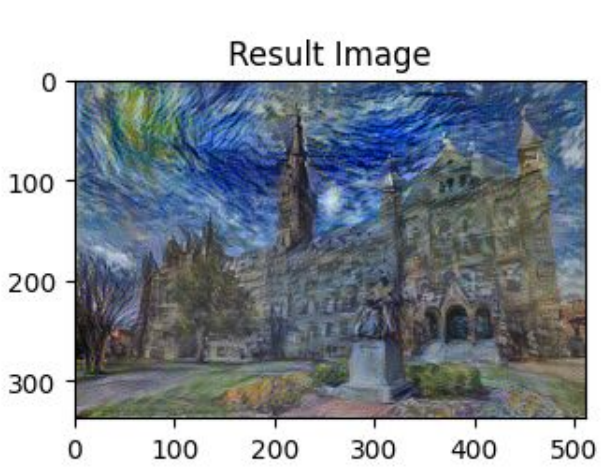
Content Weight

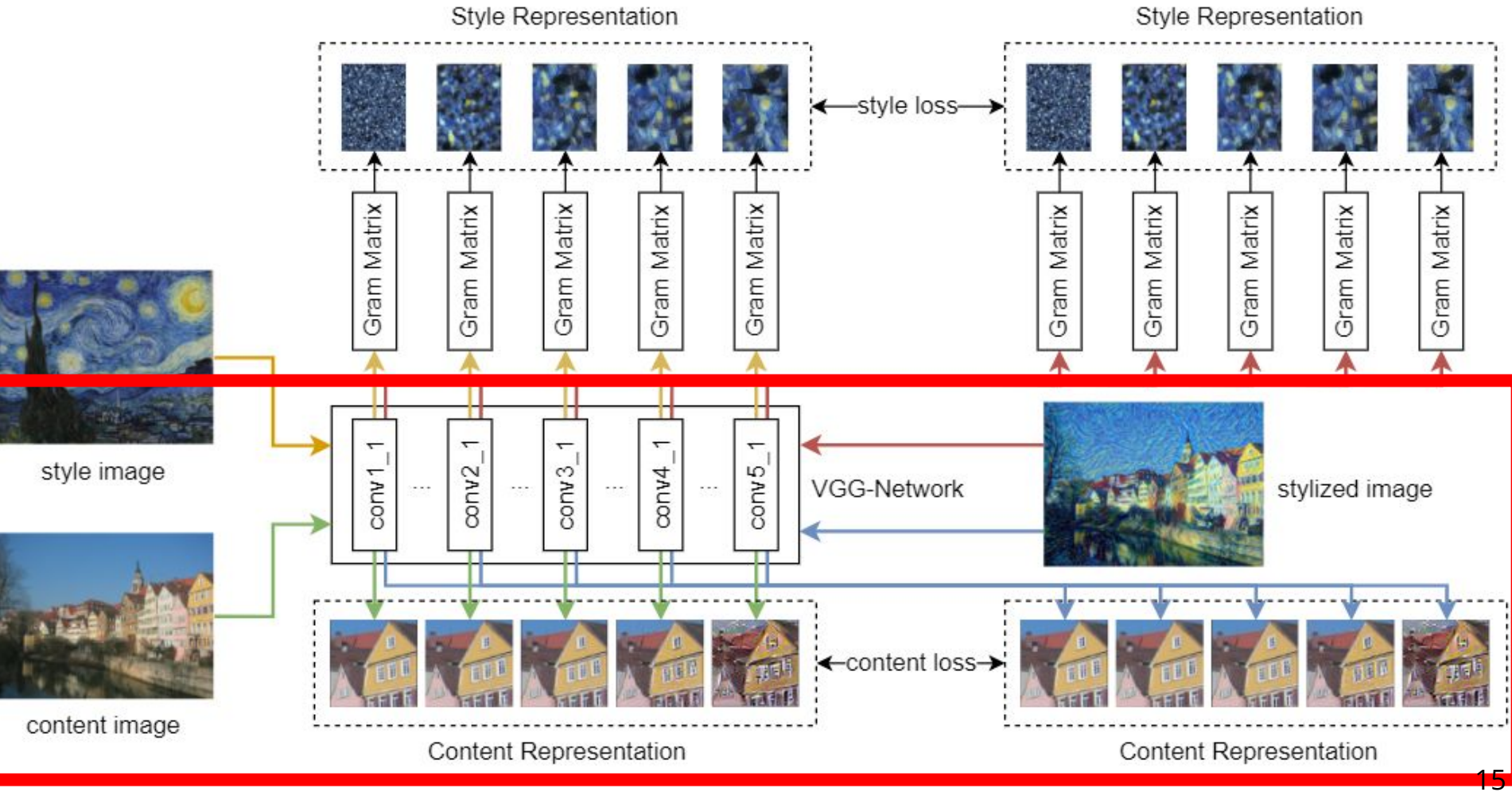
Content Image

Generated Image








Style Representation
Style Representation
style loss
Gram Matrix
Gram Matrix
Gram Matrix
Gram Matrix
Gram Matrix
Gram Matrix
Gram Matrix
Gram Matrix
Gram Matrix
Gram Matrix
style image
conv1_1
conv2_1
conv3_1
conv4_1
conv5_1
VGG-Network
stylized image
content image
content loss
Content Representation
Content Representation

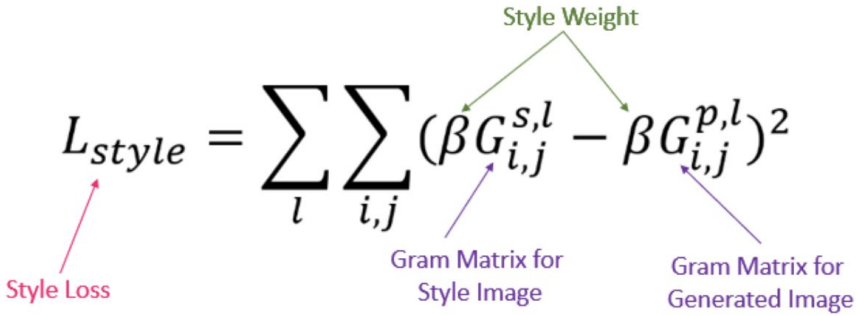
$$L_{total}(i,j,k) = \alpha L_{content}(i,k) + \beta L_{style}(j,k)$$

Content Weight

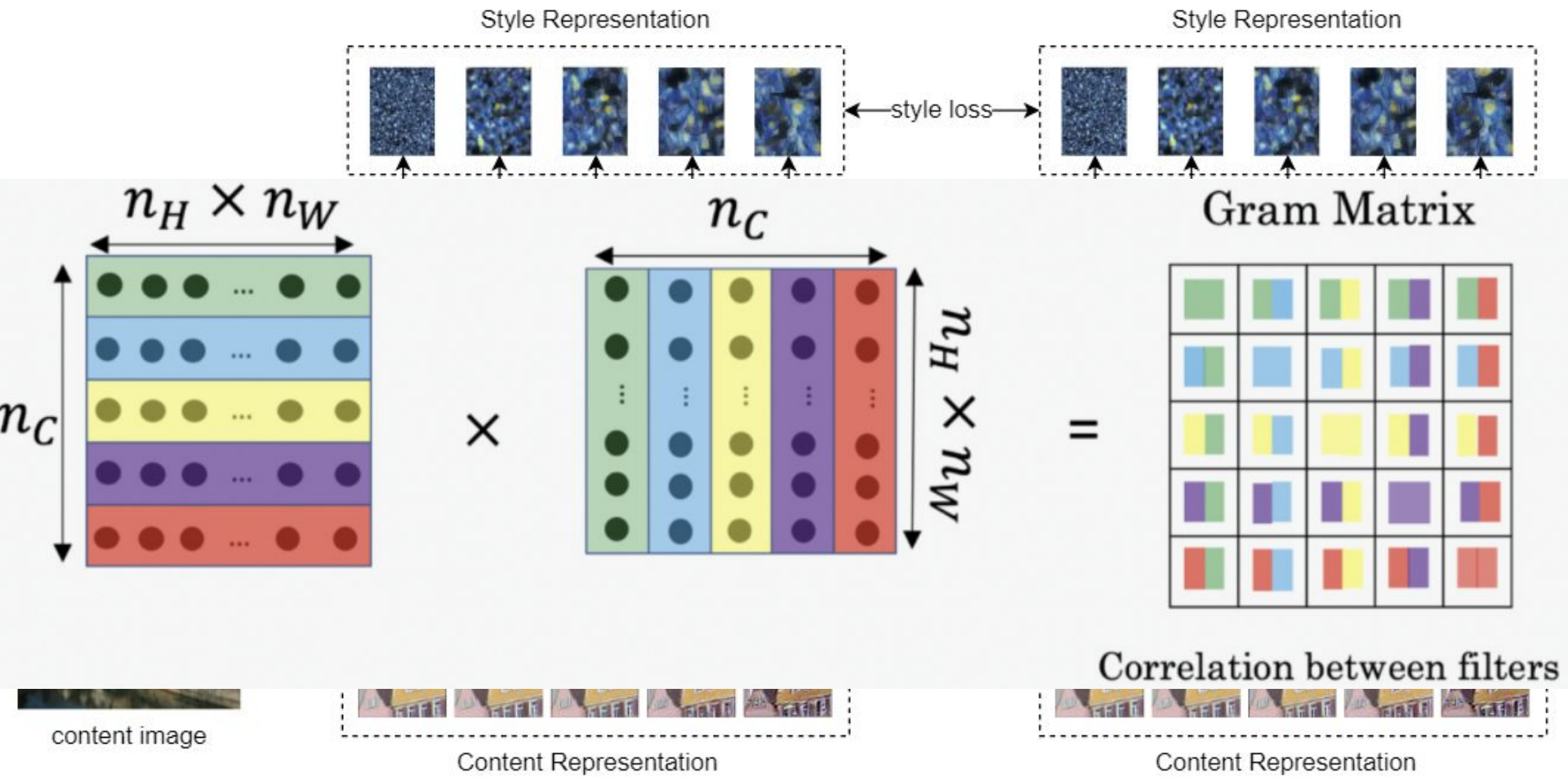
Style Weight

Hyperparameters

STYLE LOSS



Style Loss measures how different the generated image, in terms of style features, is from your style image. But it's not as straightforward as content loss. The style representation of an image is given by Gram Matrix.


Style Representation
Style Representation
style loss
$n_H \times n_W$
n_C
n_C
$n_W \times n_H$
Gram Matrix
Correlation between filters
content image
Content Representation
Content Representation

EXAMPLE OF LARGE STYLE LOSS

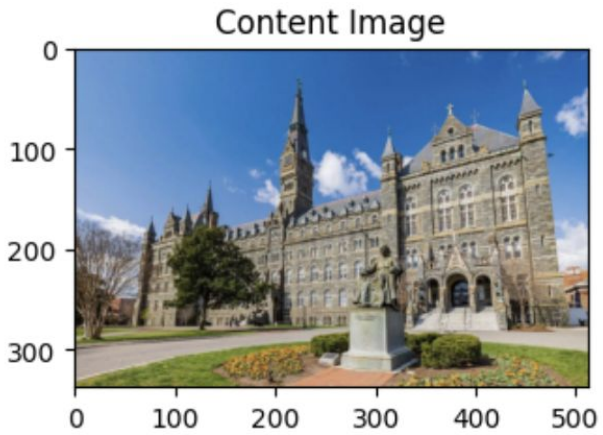
$$L_{style} = \sum_{l} \sum_{i,j} (\beta G_{i,j}^{s,l} - \beta G_{i,j}^{p,l})^2$$

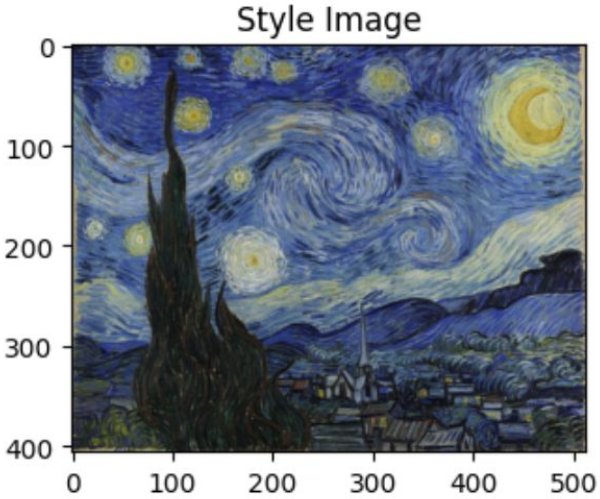
Style Weight

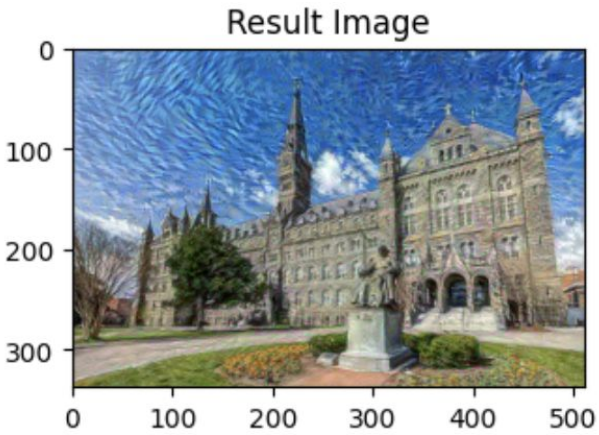
Style Loss

Gram Matrix for Style Image

Gram Matrix for Generated Image







EXAMPLE OF SMALL STYLE LOSS

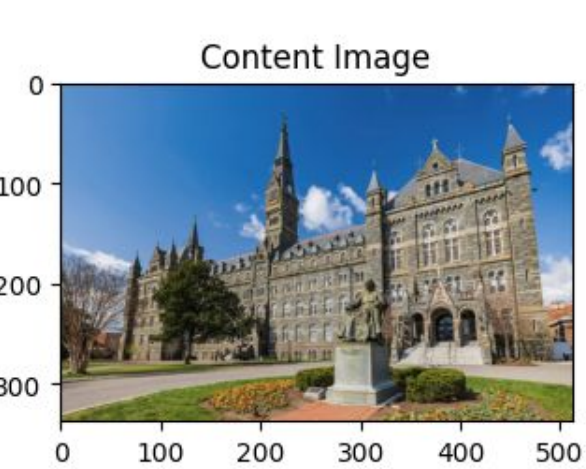
$$L_{style} = \sum_{l} \sum_{i,j} (\beta G_{i,j}^{s,l} - \beta G_{i,j}^{p,l})^2$$

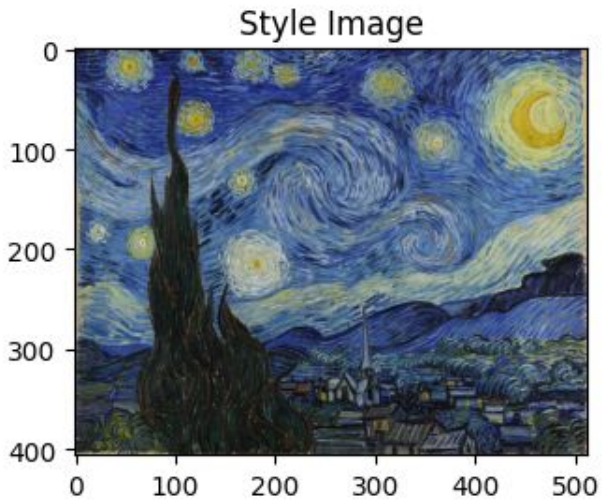
Style Weight

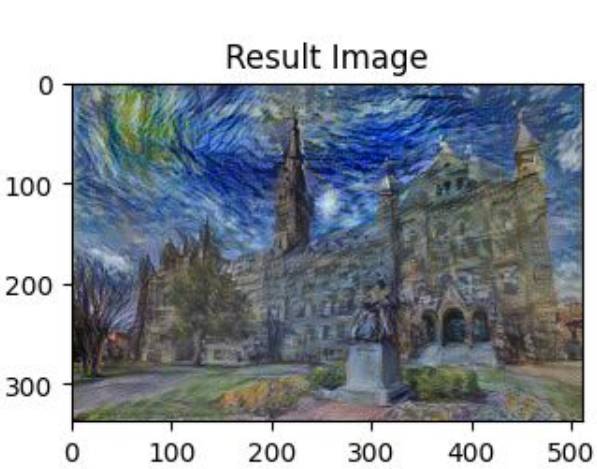
Style Loss

Gram Matrix for Style Image

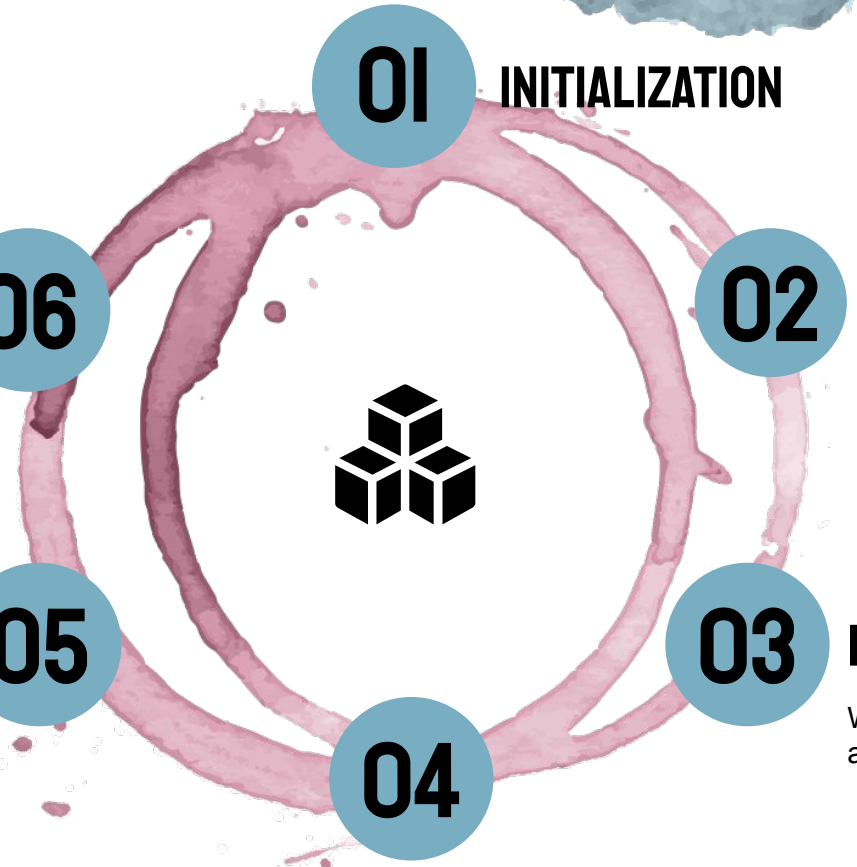
Gram Matrix for Generated Image







HOW THE STYLIZED IMAGE GETS UPDATED ?



01 INITIALIZATION

02 FEATURE EXTRACTION

- content extraction
- style extraction

03 LOSS COMPUTATION

Weighted sum of the content loss and style loss.

04 BACKPROPAGATION

05 IMAGE UPDATE

Update pixels, not weights.

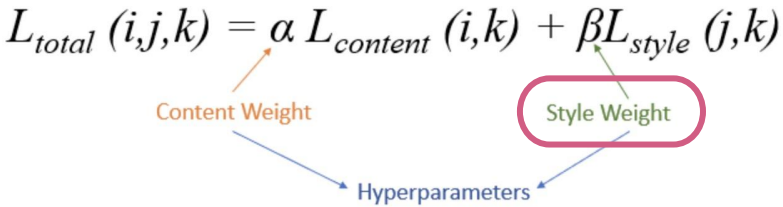
06 ITERATION

The stylized image becomes more stylized in the direction that minimizes the total loss.

1000 EPOCHS OF TRAINING

THE BEST MODEL

HYPERPARAMETER TUNING



STYLE-WEIGHT

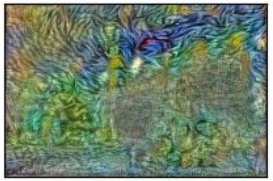
1. Interpretable Parameter
2. Balance Between Content and Style
3. Flexibility in Aesthetic Outcomes
4. Model Responsiveness
5. Optimization Stability

TUNING PROCESS

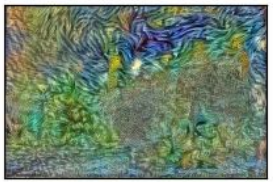
RESULTS EXAMPLE

	style_weight	epoch	total_loss	style_loss	content_loss
0	1.0	1	1.076152e+10	1.076152e+10	0.00
1	1.0	2	1.479752e+10	1.479416e+10	3367905.50
2	1.0	3	1.023803e+10	1.023430e+10	3731106.50
3	1.0	4	1.140819e+10	1.140451e+10	3678191.75
4	1.0	5	5.867498e+09	5.863605e+09	3893433.00

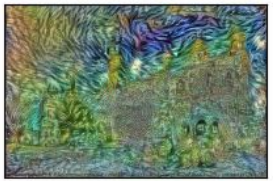
`style_weights = 1.0`

`style_weights = 0.1`

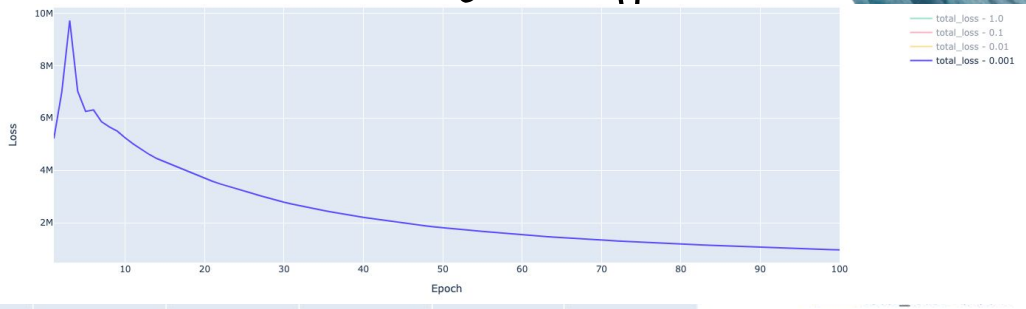



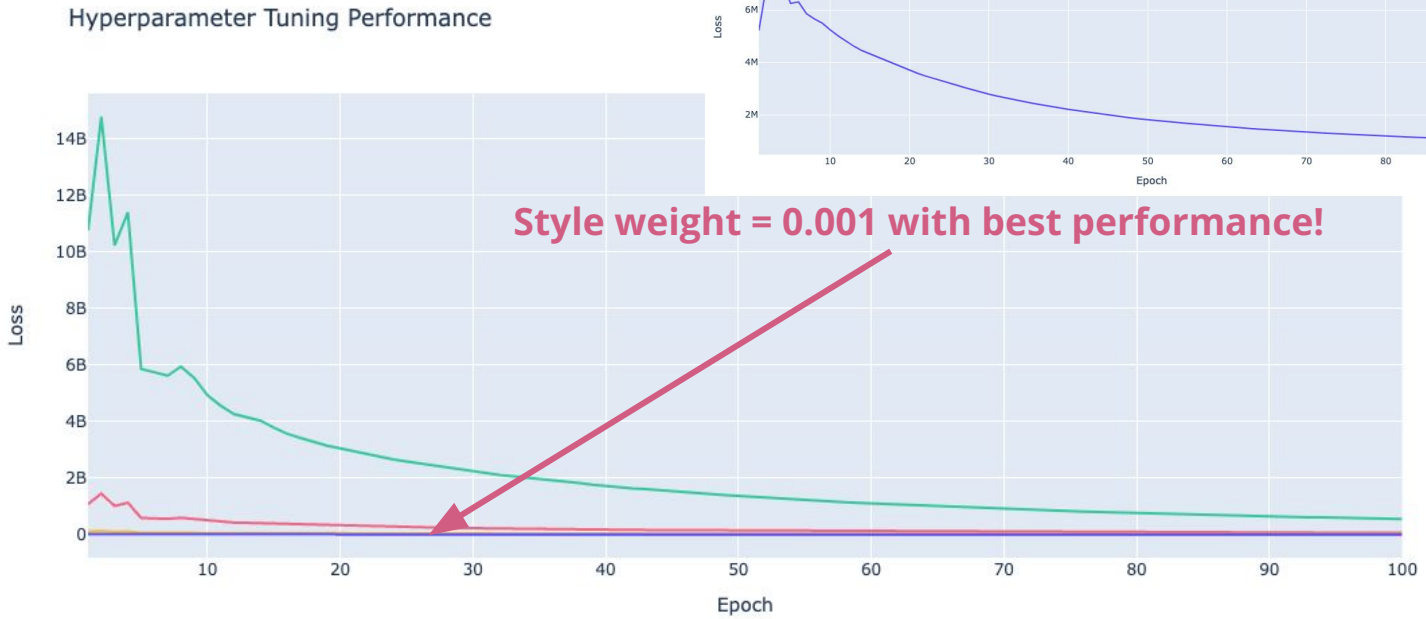
`style_weights = 0.01`

`style_weights = 0.001`


HYPERPARAMETER TUNING

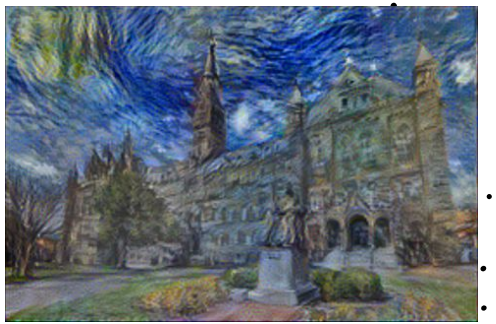




BEST MODEL PERFORMANCE

CONTENT IMAGE

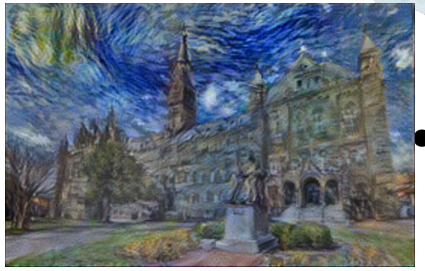
STYLE IMAGE

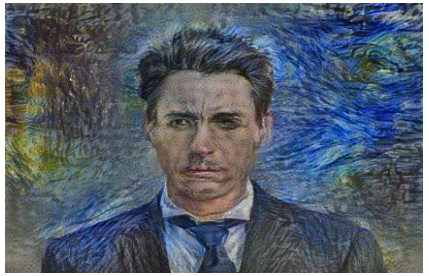


RESULT IMAGE

MODEL PERFORMANCE

THE STARRY NIGHT
Vincent Van Gogh



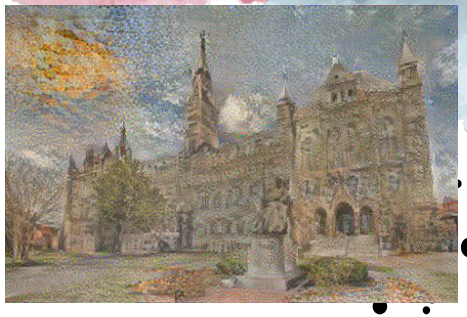


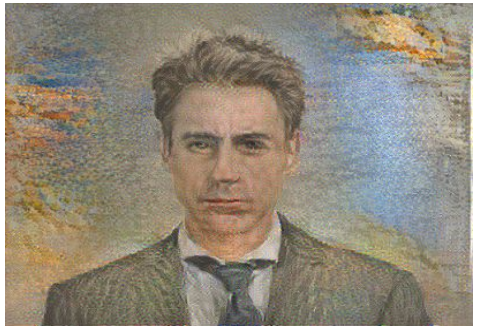
MODEL PERFORMANCE

BICENTENNIAL PRINT
Roy Lichtenstein

MODEL PERFORMANCE

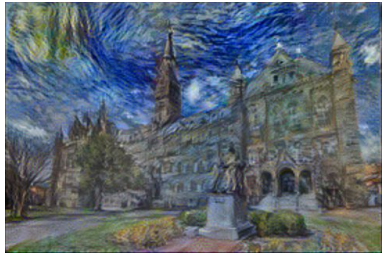
CASSIS,
CAP LOMBARD, OPUS 196
Paul Signac

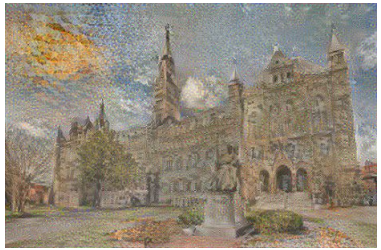
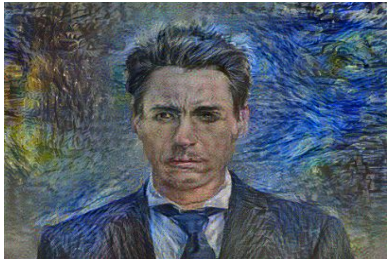


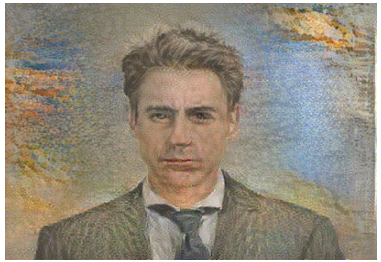


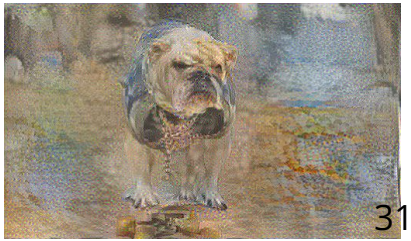
CONCLUSION





REFERENCE

- Singhal, Gaurav. “Gaurav Singhal.” Pluralsight, 3 June 2020, www.pluralsight.com/guides/implementing-artistic-neural-style-transfer-with-tensorflow-2.0.
- Zhang, Aston. “14.12. Neural Style Transfer¶ Colab [Pytorch] Open the Notebook in Colab Colab [Mxnet] Open the Notebook in Colab Colab [Jax] Open the Notebook in Colab Colab [Tensorflow] Open the Notebook in Colab Sagemaker Studio Lab Open the Notebook in SageMaker Studio Lab.” 14.12. Neural Style Transfer - Dive into Deep Learning 1.0.3 Documentation, Oct. 2020, d2l.ai/chapter_computer-vision/neural-style.html.
- Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- Dumoulin, Vincent, Jonathon Shlens, and Manjunath Kudlur. "A learned representation for artistic style." *arXiv preprint arXiv:1610.07629* (2016).

THANKS FOR LISTENING!

Q&A