Neural Style Transfer

Yuejiao Qiu, Ruoting Shen, Wenxuan(Ivy) Tang, Yuening Wang, Sirui Wang

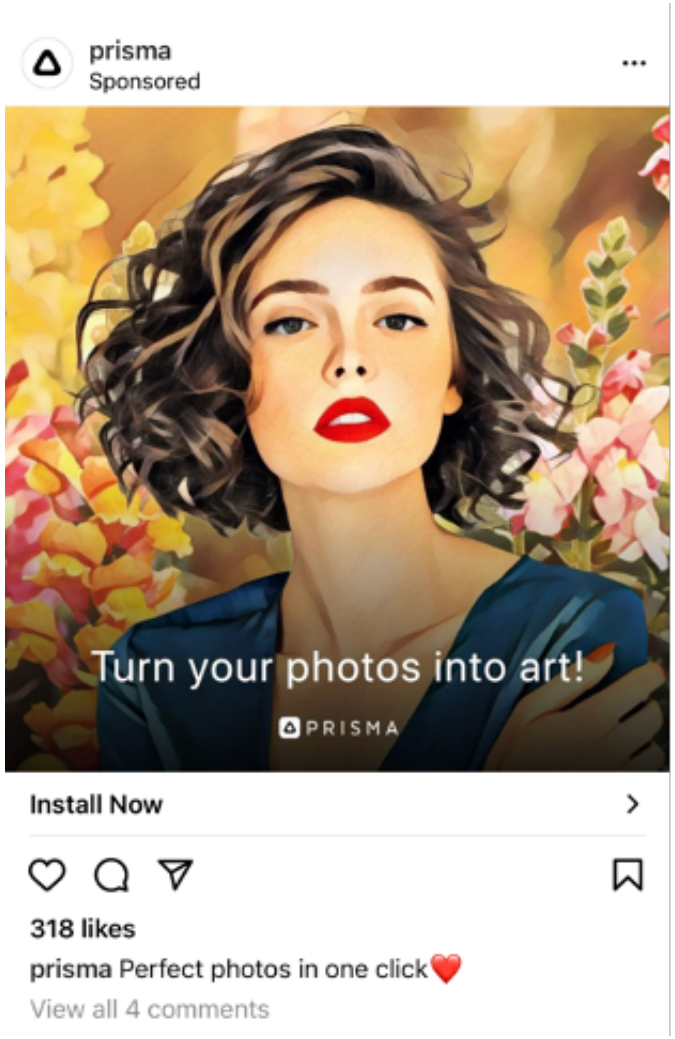
GEORGETOWN UNIVERSITY

Agenda

- Problem description
- Models and algorithms
  - Image representation
  - Basic style transfer
  - Fast transfer using Transform Net
- Experiments

Introduction

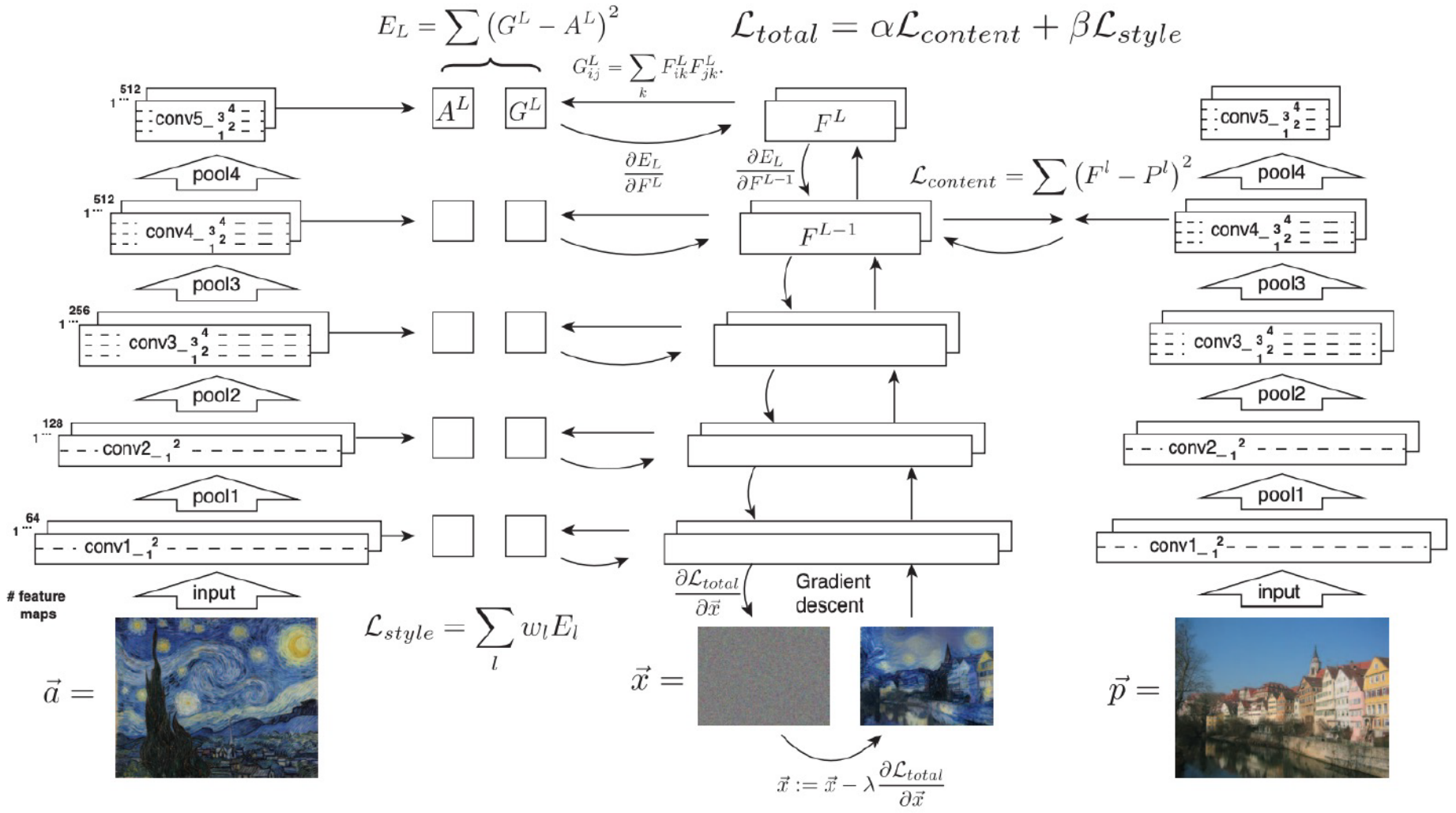
- Use case: trendy filter apps
  - Transform the original image into another image
    - Keeps the same content
    - Change the artistic styles & textures
- But filters are NOT satisfying
  - Limited to shading, brightness, contrast



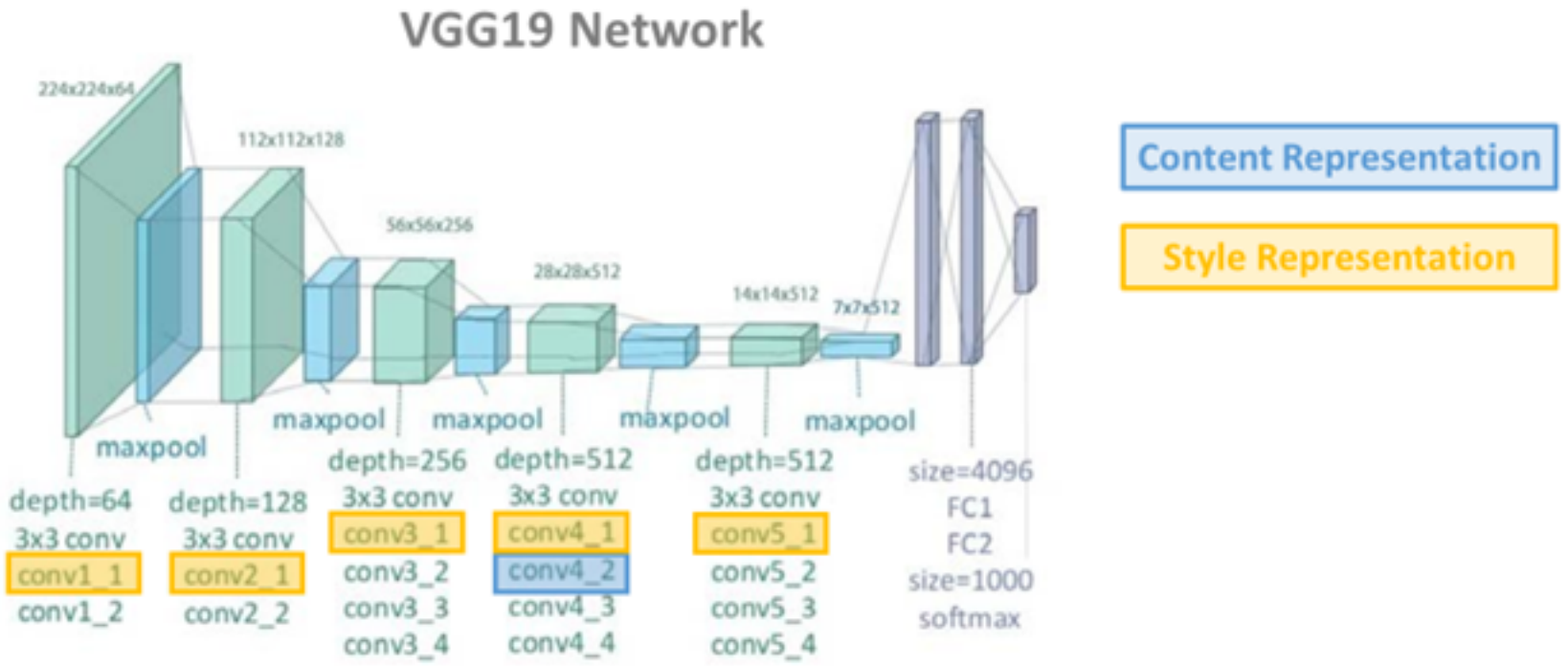
About Neural Style Transfer

- Previous research
  - AlexNet (2012), Transform Net (2016-17), Meta Network (2018)
- Goal
  - Synthesize a texture from a source image while constraining the texture synthesis in order to preserve the content of a target image.
- Category
  - Image-optimization-based (IOB) online neural methods
  - model-optimization-based (MOB) offline neural methods

Image Representations

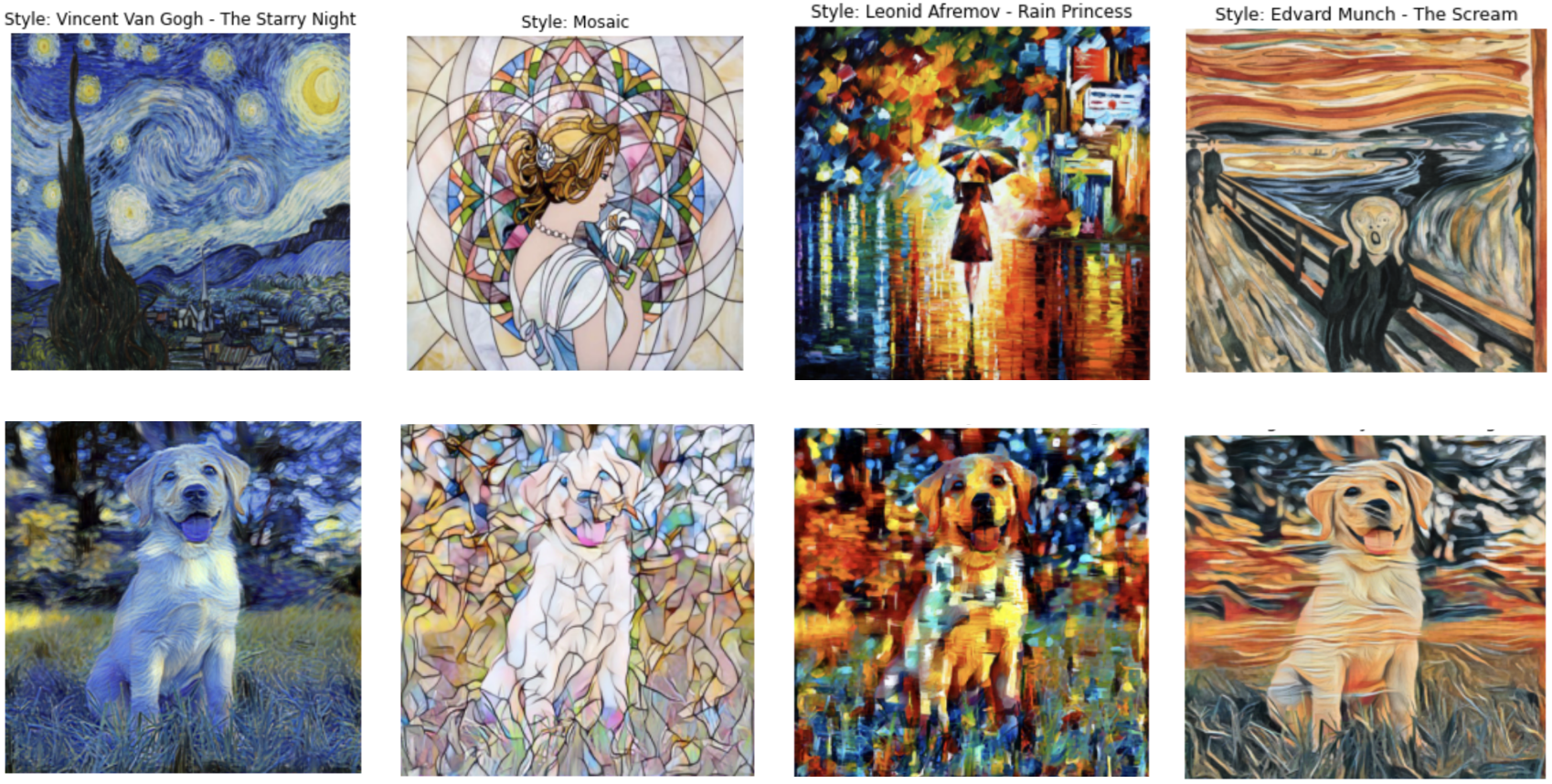


Basic style transfer - VGG19




```
# desired depth layers to compute style/content losses :
content_layers_default = ['conv_4']
style_layers_default = ['conv_1', 'conv_2', 'conv_3', 'conv_4', 'conv_5']
```

Basic style transfer performance

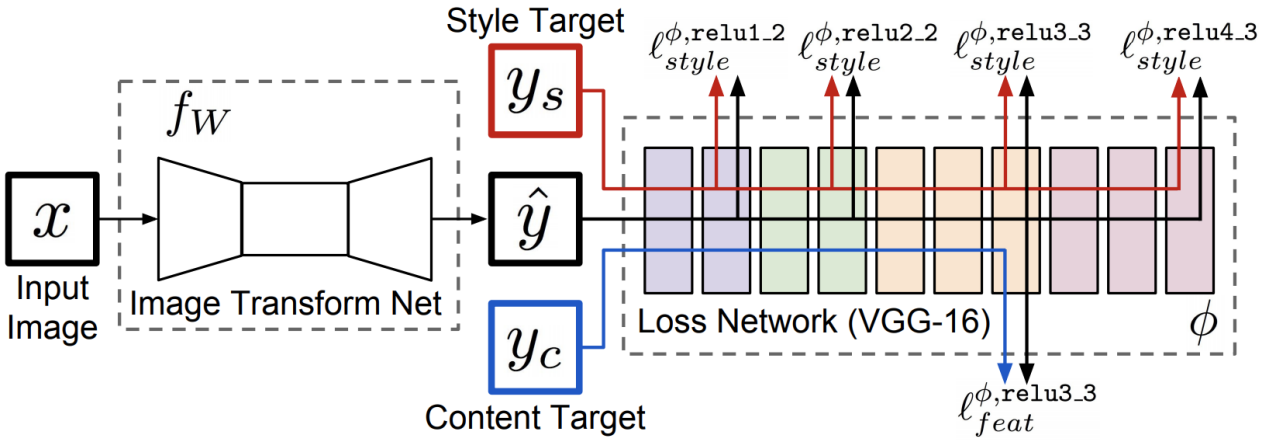


- Hue imitation
- Brushstroke imitation
- Avg runtime: 63.27s

Fast Style Transfer

Model-Optimisation-Based Offline Neural Methods (MOB-NST)

- Per-Style-Per-Model (PSPM) MOB-NST methods
  - The first two MOB-NST algorithms a proposed by Johnson et al. and Ulyanov et al.
- Multiple-Style-Per-Model (MSPM) MOB-NST Methods
- Arbitrary-Style-Per-Model (ASPM) MOB-NST Methods



Perceptual Loss Function

- Content loss $\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$
  - The (squared, normalized) Euclidean distance between feature representations (Johnson et al., 2016).
- Style loss $\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|_F^2.$
  - The squared Frobenius norm of the difference between the Gram matrices of the output and target images (Johnson et al., 2016).
- Minimize total loss
  - *total variation regularize*

$$\hat{y} = \arg\min_y \lambda_c \ell_{feat}^{\phi,j}(y, y_c) + \lambda_s \ell_{style}^{\phi,J}(y, y_s) + \lambda_{TV} \ell_{TV}(y)$$

$\ell_{TV}(\hat{y})$.

Difference

Basic Style Transfer:

➔ Output feature maps of the provided pair of images.

Fast Style Transfer:

➔ Train the image transform network and deal with **any** content and style of images.

Dataset

LabelMe

- A large dataset created by the MIT Computer Science and Artificial Intelligence Laboratory.
  - Total: 187,240 images, 62,197 annotated images, 658,992 labeled objects
  - We used 20,000 images

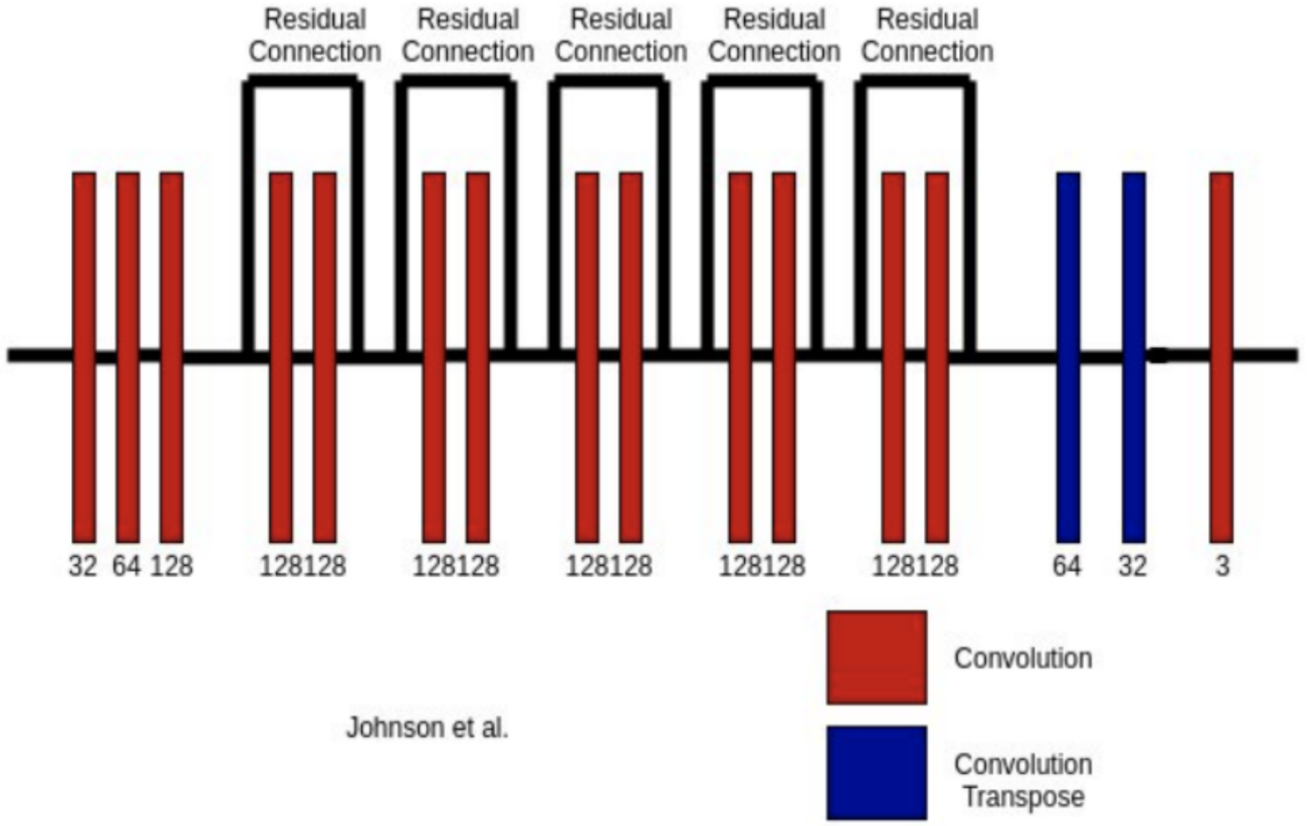
Implementation - Modified VGG19

```
# create 'VGG' class to train on VGG19 loss network
class VGG(nn.Module):
    # initialize
    def __init__(self, features):
        super(VGG, self).__init__()
        self.features = features
        # specify layer need to save in output results
        self.layer_name = {
            '3': 'relu1_2',
            '8': 'relu2_2',
            '17': 'relu3_4',
            '22': 'relu4_2',
            '26': 'relu4_4',
            '35': 'relu5_4'
        }
        # Turn-off Gradient History
        for p in self.features.parameters():
            p.requires_grad = False #frozen parameters of VGG
```

- Outputs of certain layers
- Gram matrix function
- Calculate style loss

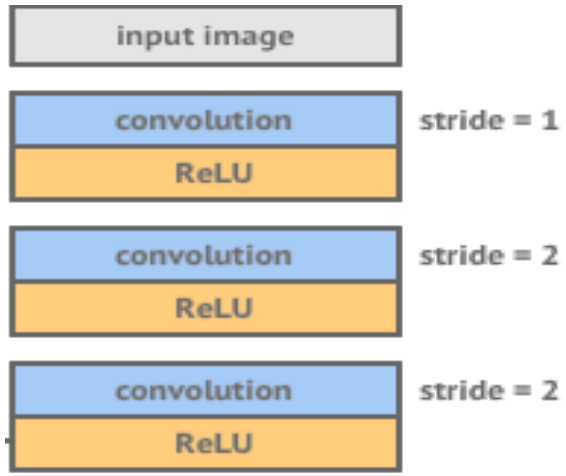
Implementation - Transform Network

- Architecture overview
- Instance normalization
- Activation: ReLU & Tanh



Implementation - Convolutional block

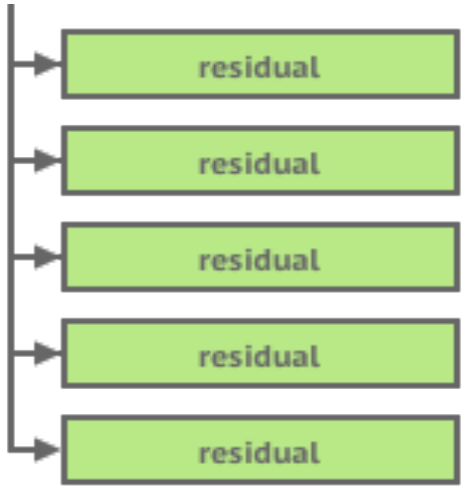
- 3 Convolutional layers
- Downsample
- Higher dimension




```
# Conv layer block
        self.ConvBlock = nn.Sequential(
            ConvLayer(3, base, kernel_size=9, stride=1),
            ConvLayer(base, base*2, kernel_size=3, stride=2),
            ConvLayer(base*2, base*4, kernel_size=3, stride=2),
        )
```

Implementation - Residual blocks

- 5 residual blocks
- 2 convolutional layers
- Adding features

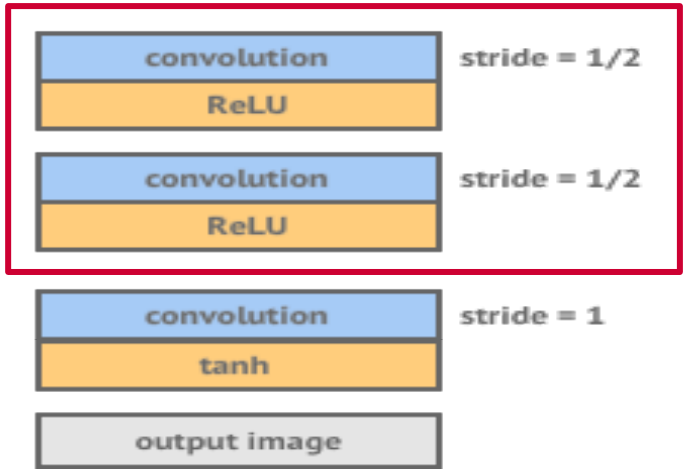



```
# residual class
     self.conv1 = ConvLayer(channels, channels, kernel_size=3,
stride=1)
     self.conv2 = ConvLayer(channels, channels, kernel_size=3,
stride=1, relu=False)

# res layer block
       self.ResidualBlock = nn.Sequential(
           *[ResidualLayer(base*4) for i in range(5)]
           )
```

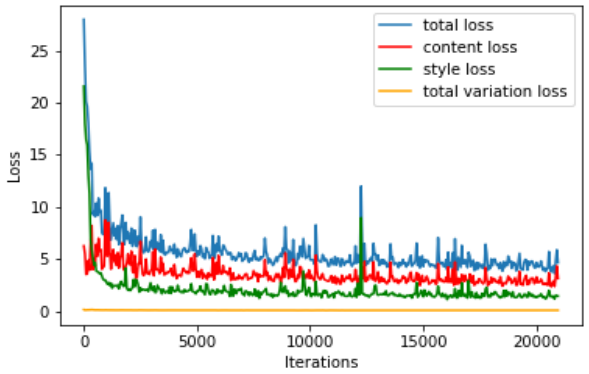
Implementation - Convolutional Transpose

- 2 conv transpose layers
- Upsample
- 1 convolutional layer

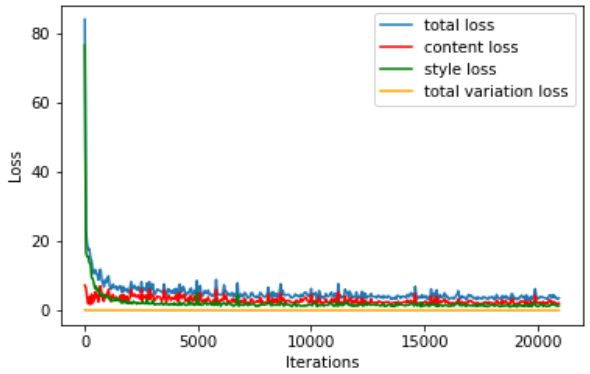



```
# conv transpose layer block
        self.ConvTBlock = nn.Sequential(
            ConvTLayer(base*4, base*2, kernel_size=3, stride=2,
output_padding=1),
            ConvTLayer(base*2, base, kernel_size=3, stride=2,
output_padding=1),
            ConvLayer(base, 3, kernel_size=9, stride=1,
norm=False, relu=False)
        )
```

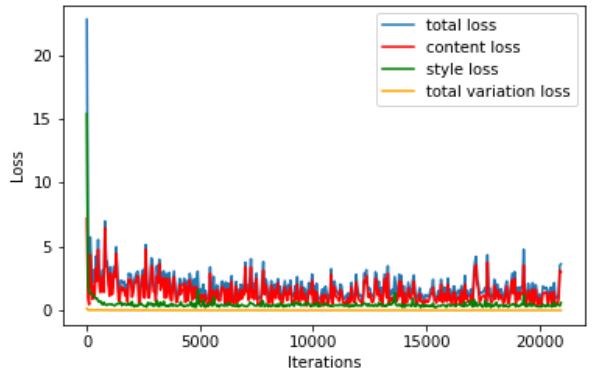
Loss Descent Plots



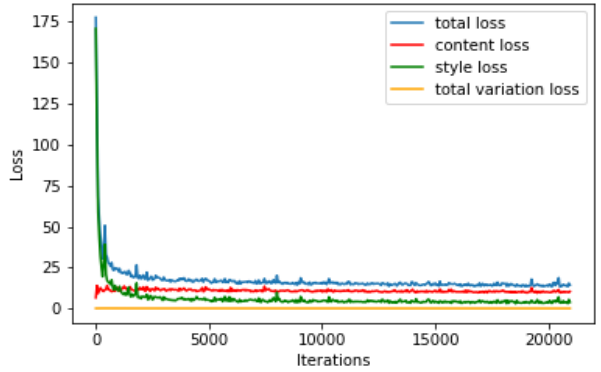
The Great Wave off Kanagawa
Katsushika Hokusai



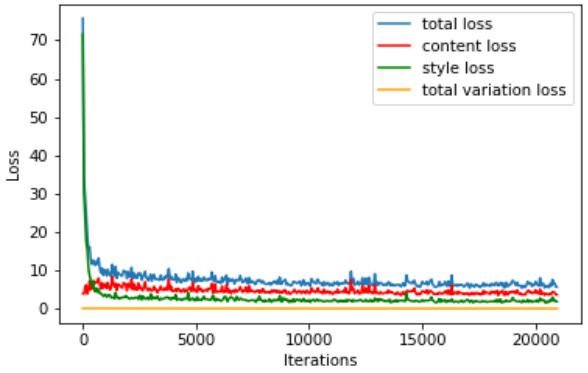
Old Canal Port
Oscar Florianus Bluemner



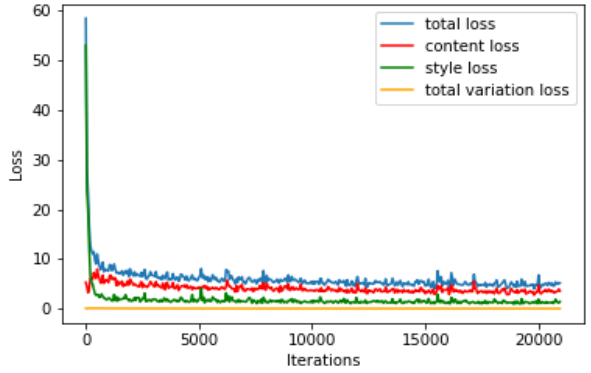
Portrait of Lisa Gherardini
Leonardo Da Vinci



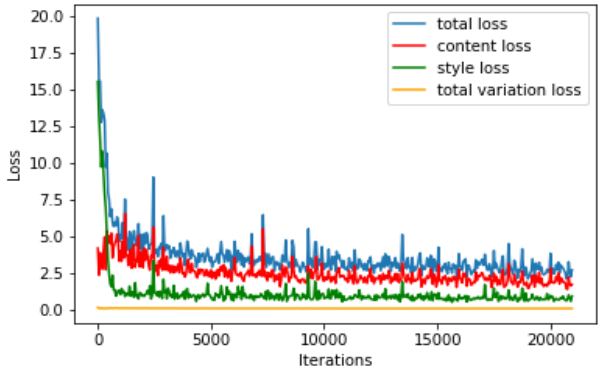
Rain Princess
Leonid Afremov



The Scream
Edvard Munch



The Starry Night
Vincent Van Gogh

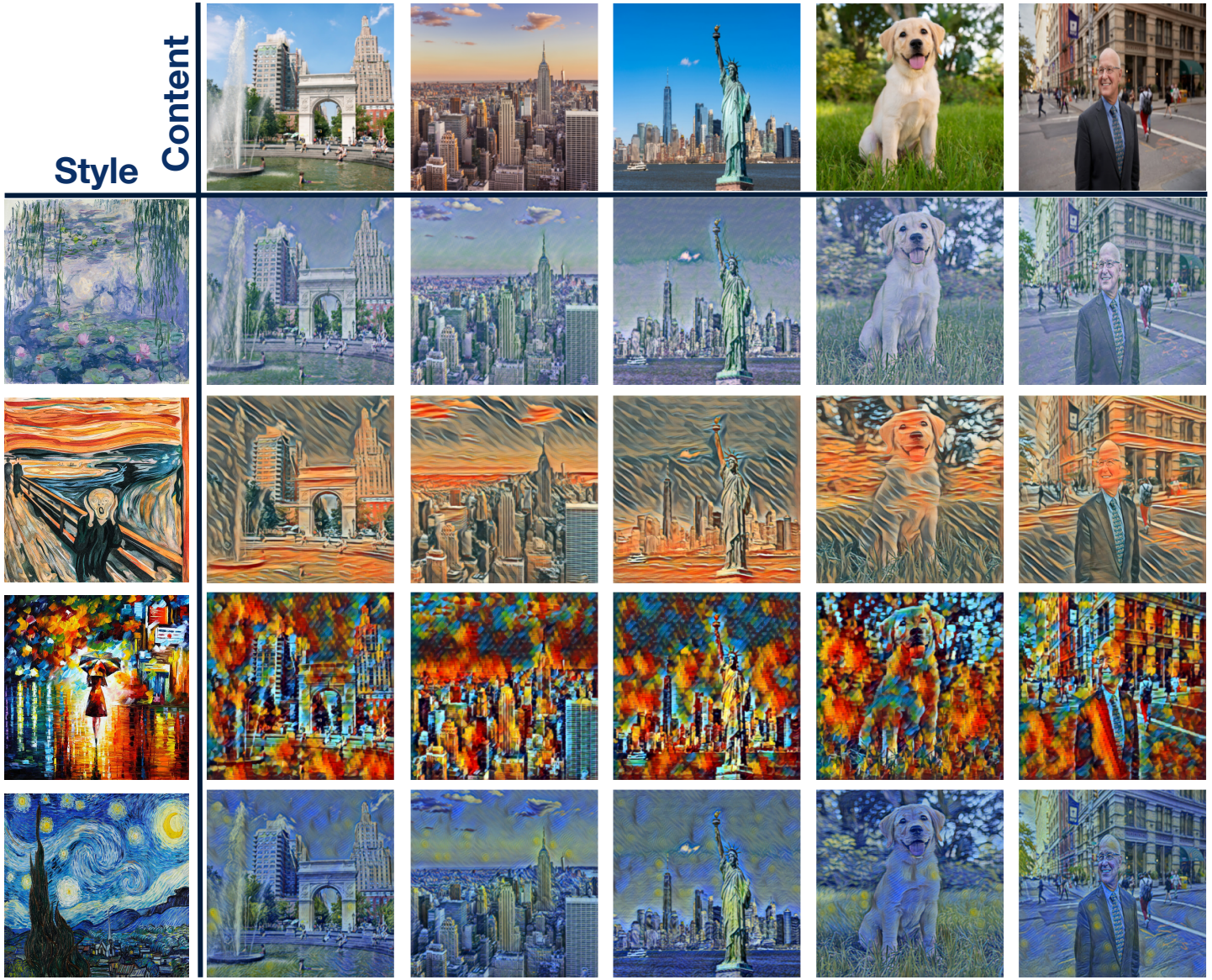


Water-Lilies (1916)
Claude Monet

Output-1

- **Average Running time: 0.17s**

Output-2


Content
Style
Water-Lilies (1916)
Claude Monet
The Scream
Edvard Munch
Rain Princess
Leonid Afremov
The Starry Night
Vincent Van Gogh

Comparison and Conclusion

Further Explorations

- Use basic transfer output to fine-tune faster transfer model.
- Real-time style transfer regardless the style.
- Blend multiple styles into one content image.

References

- Johnson, J., Alahi, A., & Fei-Fei, L. (2016, March 27). *Perceptual losses for real-time style transfer and Super-Resolution*. arXiv.org. Retrieved December 6, 2022, from https://arxiv.org/abs/1603.08155

- Jing,Y., Yang, Y., Feng,Z., Ye, J., Yu, Y., and Song, M., “Neural style transfer: A review”, IEEE Transactions on Visualization and Computer Graphics, pp. 1–1, 2019.

- Shen, F., Yan, S., Zeng, G., “Neural Style Transfer via Meta Networks”

- Ulyanov, D., Lebedev V., Vedaldi, A., and Lem-pitsky, V., “Texture Networks: Feed-forward Synthesis of Textures and Stylized Images”, arXiv e-prints, p.arXiv:1603.03417, 2016.

- Gatys,L. A., Ecker,A. S., and Bethge, M., “Image style transfer using convolutional neural networks”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

Q & A