Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization

Huang et al., ICCV 2017

YANG Chengkai
Department of Electrical Engineering
and Information Systems, M1

Style Transfer

Content

Style

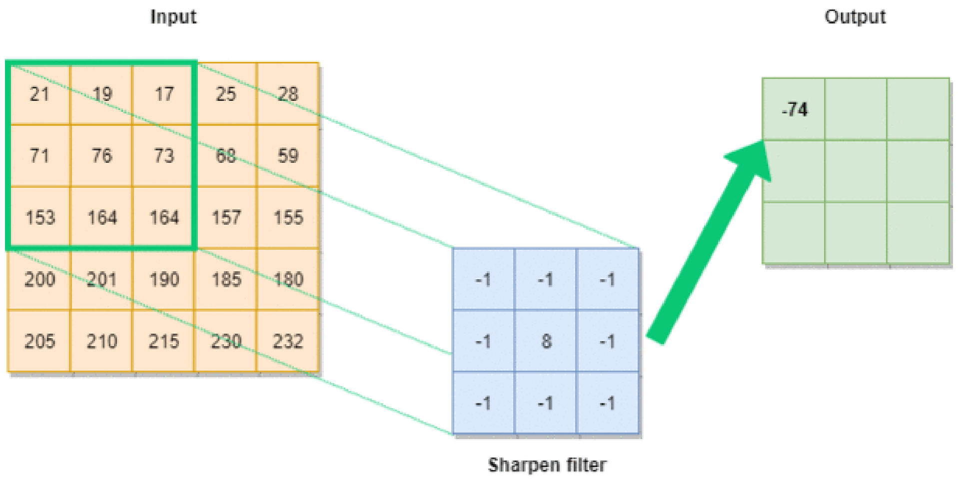
Output

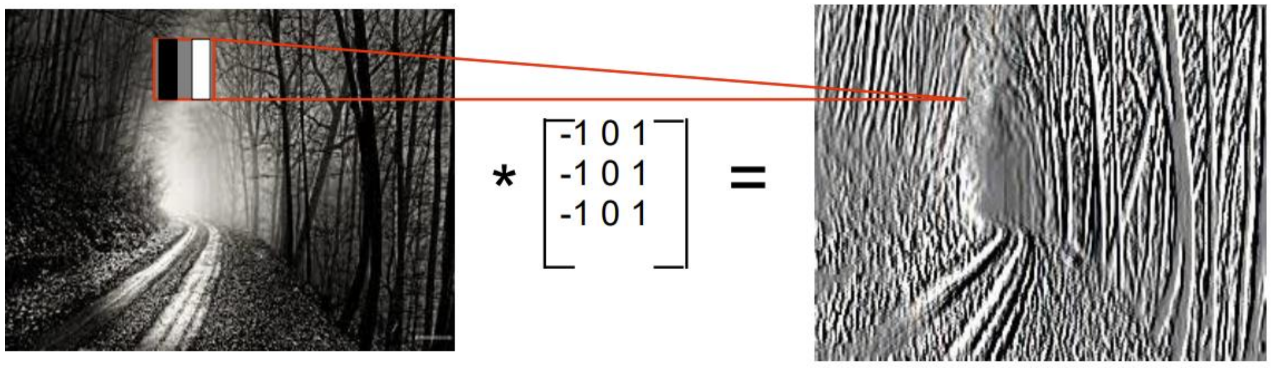
Source: http://stevydya.blogspot.com/2015/04/the-university-of-tokyo.html,
WikiArt

Convolutional Neural Networks

Convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$



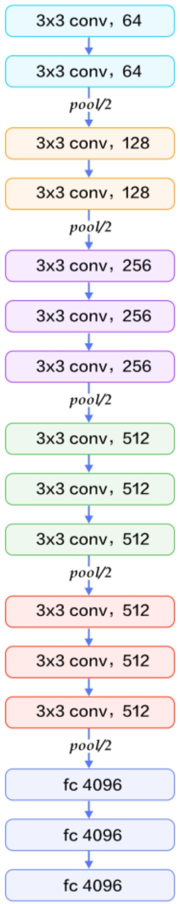


input

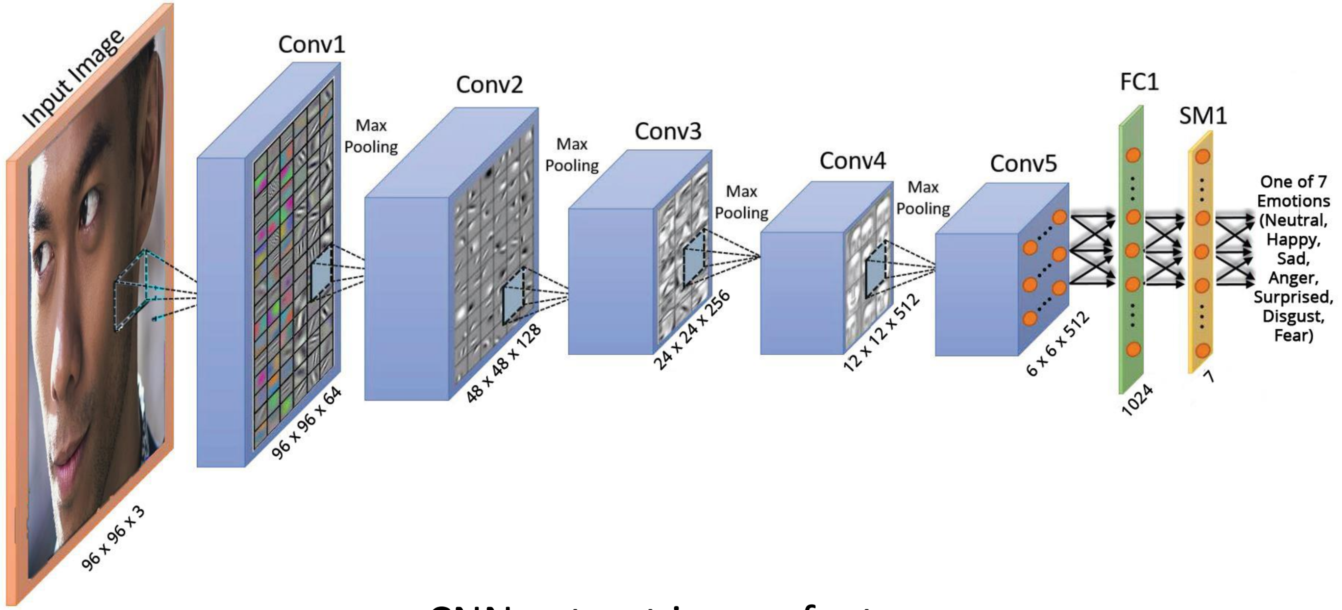
feature

Source: https://www.analyticsvidhya.com/blog/2021/07/convolution-neural-network-better-understanding/

Convolutional Neural Network



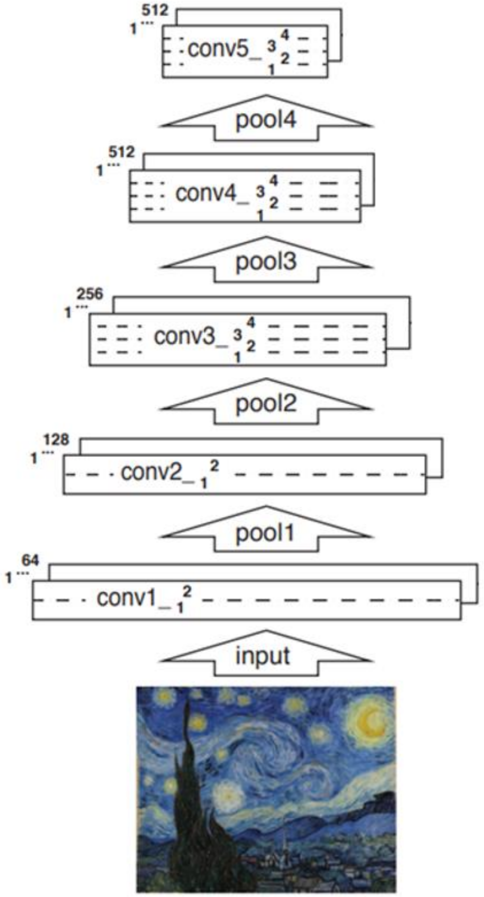
VGG Architecture

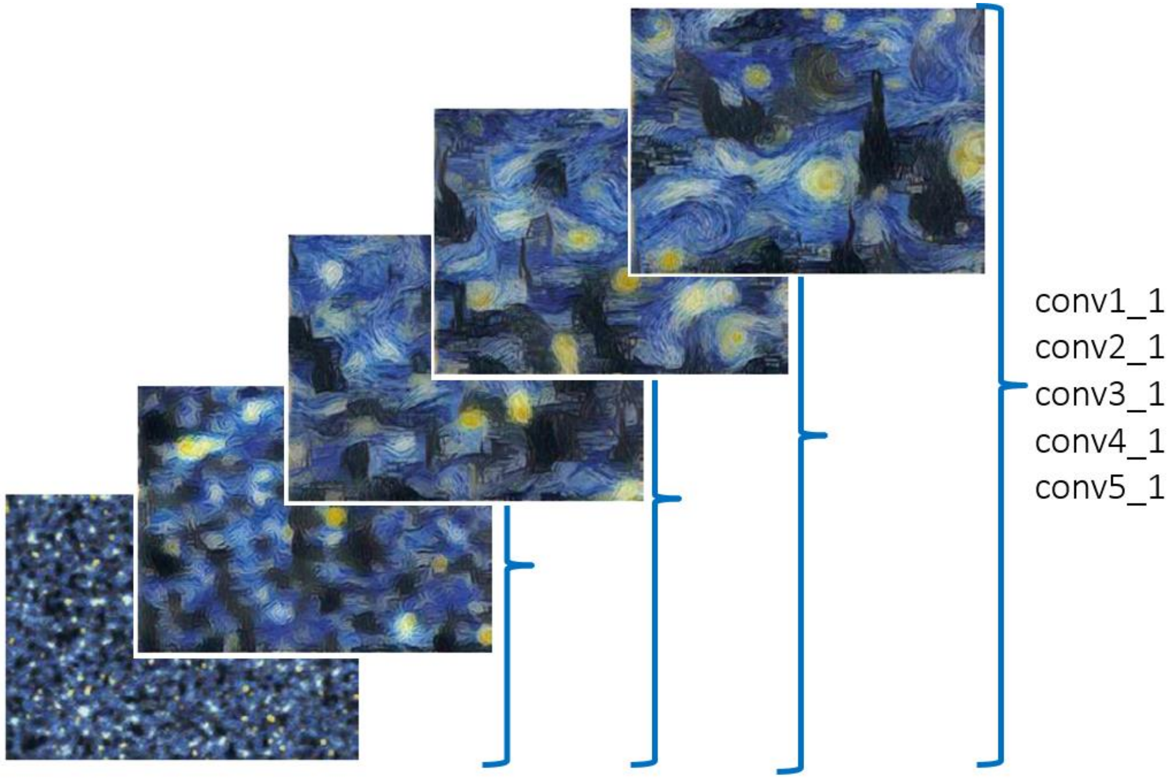


CNN extract image features

CNN Captures Style

Image Style Transfer using Convolutional Neural Networks, Gatys et al., CVPR 2016



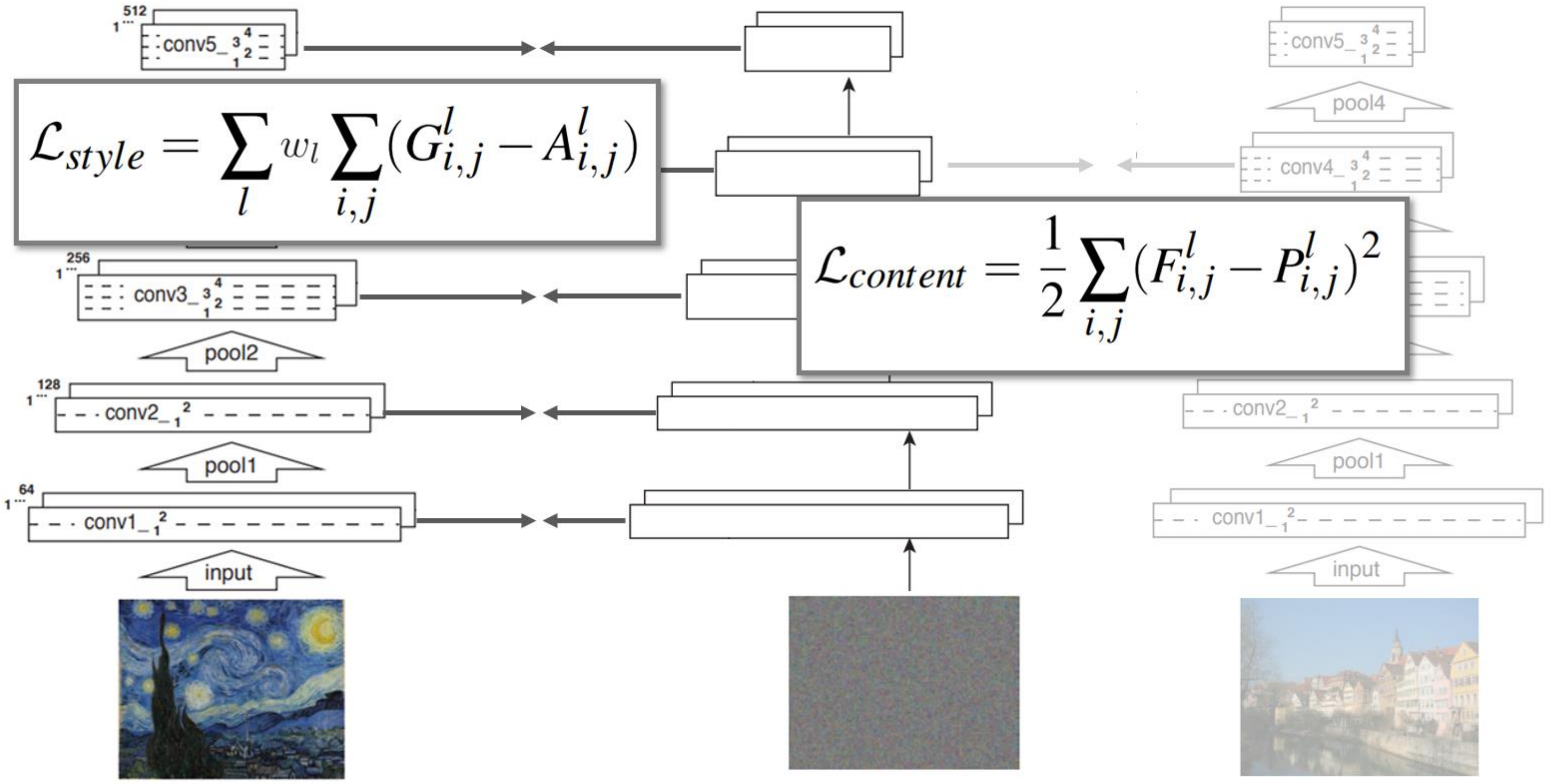


Content **representation**: Activations of the filter ***i*** at a position ***j*** in a layer ***l*** F_{ij}^{l}

Style **representation**: Inner product between vectorised feature maps $G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}.$

Arbitrary Style Transfer

Image Style Transfer using Convolutional Neural Networks, Gatys et al., CVPR 2016



Arbitrary: Works for any pair of inputs

Slow: Require network optimization to generate an output

AdaIN Style Transfer

Huang et al., ICCV 2017

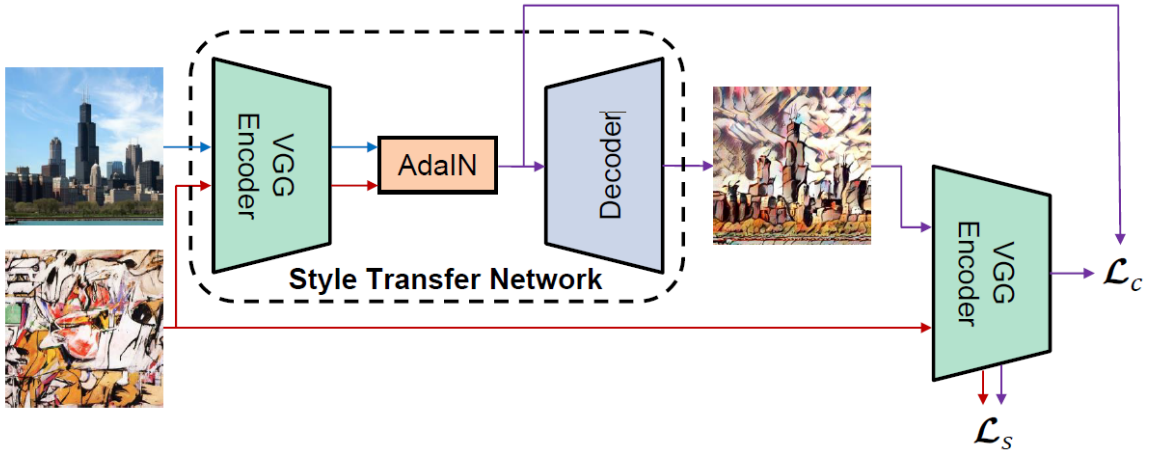


Figure 2. An overview of our style transfer algorithm. We use the first few layers of a fixed VGG-19 network to encode the content and style images. An AdaIN layer is used to perform style transfer in the feature space. A decoder is learned to invert the AdaIN output to the image spaces. We use the same VGG encoder to compute a content loss $\mathcal{L}_c$ (Equ. 12) and a style loss $\mathcal{L}_s$ (Equ. 13).

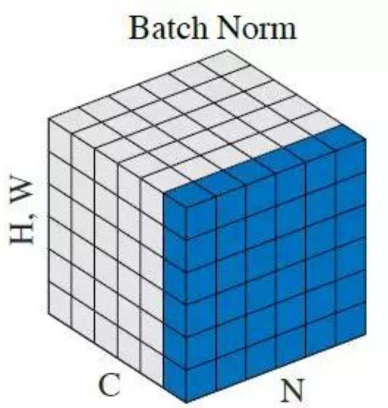
VGG Encoder – Extract features

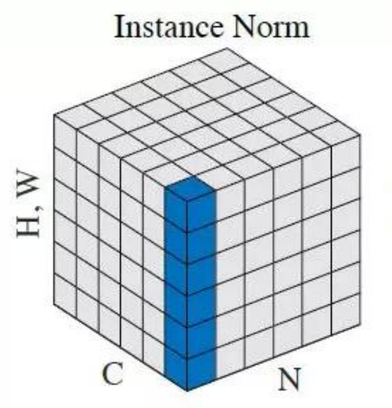
AdaIN – Perform style transfer on features

Decoder – Reconstruct image from features

Arbitrary: Works for any pair of inputs
Fast: No optimization required for testing

Batch Norm vs Instance Norm





shifting and ***scaling*** the activations
by ***mean*** and ***standard deviation***

$$BN(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \beta$$

$$\mu_c = \frac{1}{NHW}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W} x_{icjk}$$

$$\sigma_c^2 = \frac{1}{NHW}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W} (x_{icjk} - \mu_c)^2$$

$$IN(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \beta$$

$$\mu_{nc} = \frac{1}{HW}\sum_{j=1}^{H}\sum_{k=1}^{W} x_{ncjk}$$

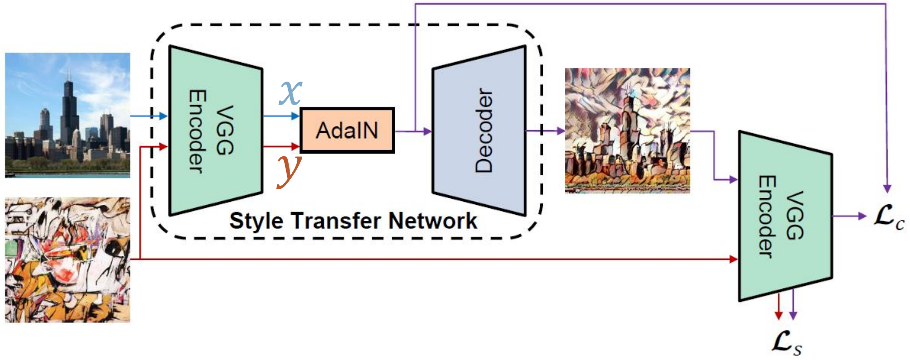
$$\sigma_{nc}^2 = \frac{1}{HW}\sum_{j=1}^{H}\sum_{k=1}^{W} (x_{ncjk} - \mu_{nc})^2$$

Affined parameters learned by gradient descent

Source: https://medium.com/syncedreview/facebook-ai-proposes-group-normalization-alternative-to-batch-normalization-fb0699bffae7

Adaptive Instance Normalization

Huang et al., ICCV 2017



x : content feature y : style feature

Style Representation is feature statistics: Feature-wise mean and variance

$$\mathrm{AdaIN}(x,y) = \sigma(y)\underbrace{\left(\frac{x-\mu(x)}{\sigma(x)}\right)}_{}+\mu(y)$$

Step 1 Normalize Content Image
Removes original style

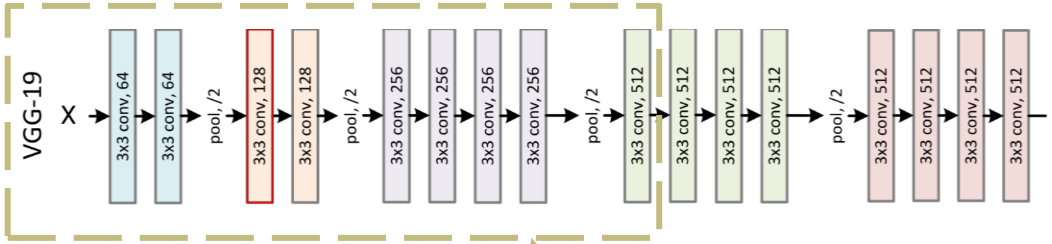
$$\mathrm{AdaIN}(x,y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\mu(y)$$

Step 2 Align style image statistics
Affine parameters adapted from style image

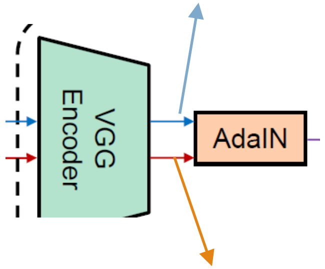
Architecture

Huang et al.,
ICCV 2017

VGG19 up
to ReLU4-1



f(c):content feature

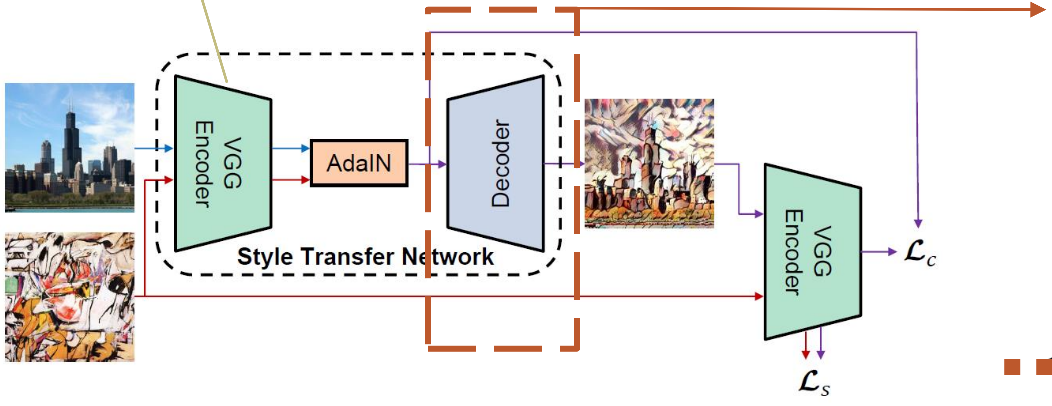


f(s):style feature

$$t = \mathrm{AdaIN}(f(c), f(s))$$

$$\mathrm{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

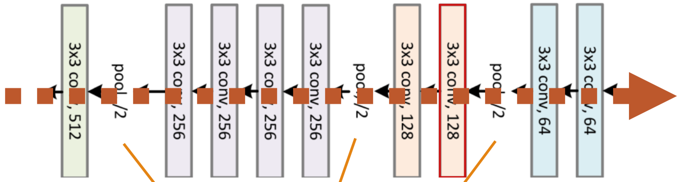
Align content statistics to style statistics
Output feature map t



Decoder generates stylized images from feature map t

$$T(c, s) = g(t)$$

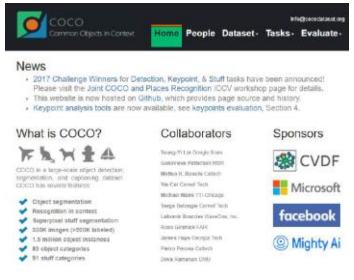
Stylized image
Decoder



Nearest Upsampling

Training

Huang et al.,
ICCV 2017

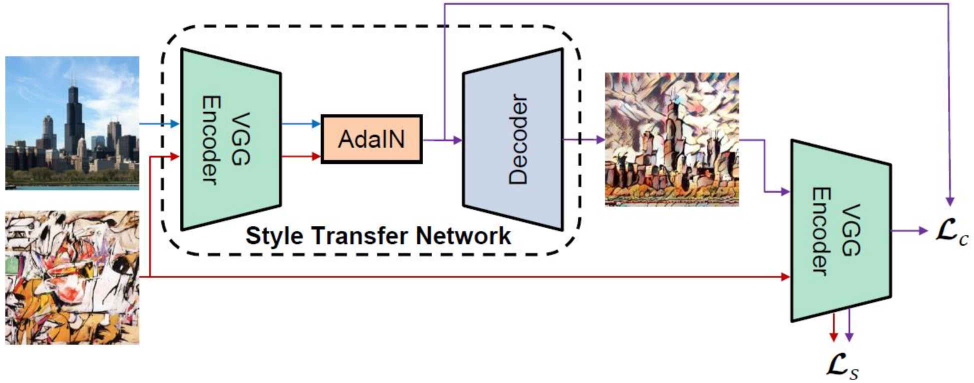


MS-COCO (80,000 images)

Content Image

WikiArt (80,000 images)

Content Image



Pre-process
- Resize (512)
- Ramdom Crop (256 x 256)

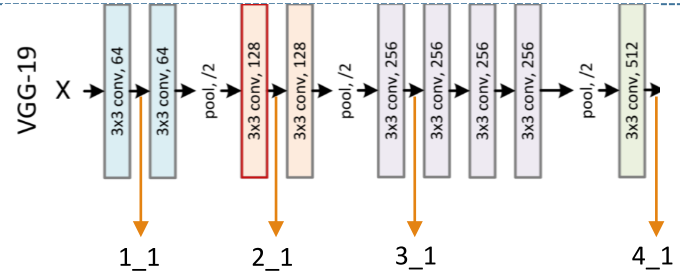
Content Loss

$$\mathcal{L}_c = \| f(g(t)) - t \|_2$$

- Euclidean distance between image features of t and output image
- t is AdaIN output -> better convergence

Style Loss

$$\mathcal{L}_s = \sum_{i=1}^{L} \| \mu(\phi_i(g(t))) - \mu(\phi_i(s)) \|_2 \ + \sum_{i=1}^{L} \| \sigma(\phi_i(g(t))) - \sigma(\phi_i(s)) \|_2$$



- Match mean, variance between features of style image and output image
- ϕ_i denotes a layer in VGG. Used relu1_1, relu2_1, relu3_1, relu4_1 with equal weights

Implementation

Unofficial Implementation using Python and PyTorch, available on github

- Built AdaIN architecture from scratch with PyTorch
- Code for training (Because of time, I did not finish training. Currently, I use pretrained weights)
- Code for testing
  - Basic Image Style Transfer
  - Style Interpolation
  - Video Style Transfer
  - Utility functions (grid image, …)

I referred to:

- Official Implementation in Torch7
- Pretrained weights of encoder and decoder from naoto0804

Content Images

Style Images from Wikiart, not seen by the model during training

MSCOCO

https://9gag.com/animals/amv2gOj?ref=fsidebar

https://inf.news/en/digital/cf25a5685894c53d45c9b7c6a6040564.html

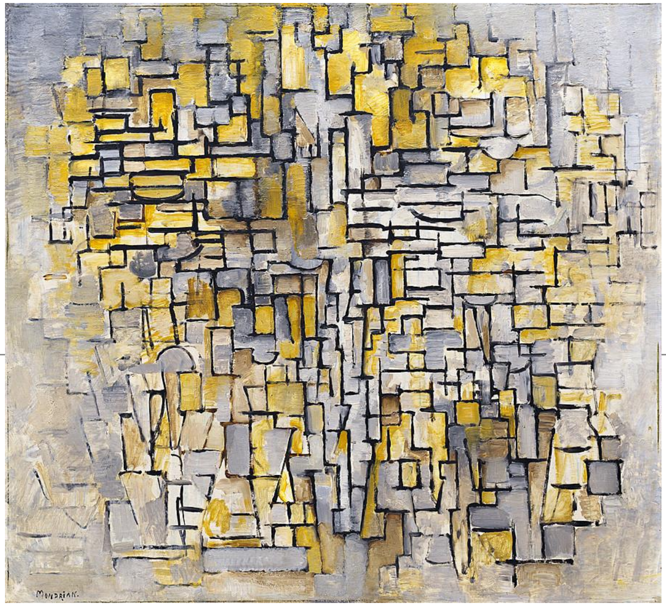
https://www.pinterest.com/pin/818318194786905146/

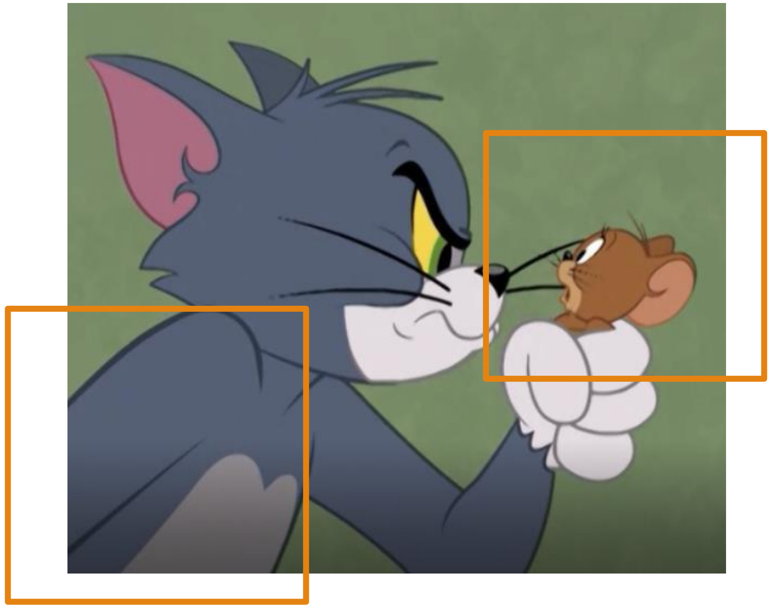
MSCOCO

https://www.cbr.com/spider-man-no-way-home-new-costume-gold-emblem-marvel/

https://www.indiatimes.com/entertainment/tom-and-jerry-birthday-facts-you-did-not-know-561641.html

http://stevydya.blogspot.com/2015/04/the-university-of-tokyo.html,

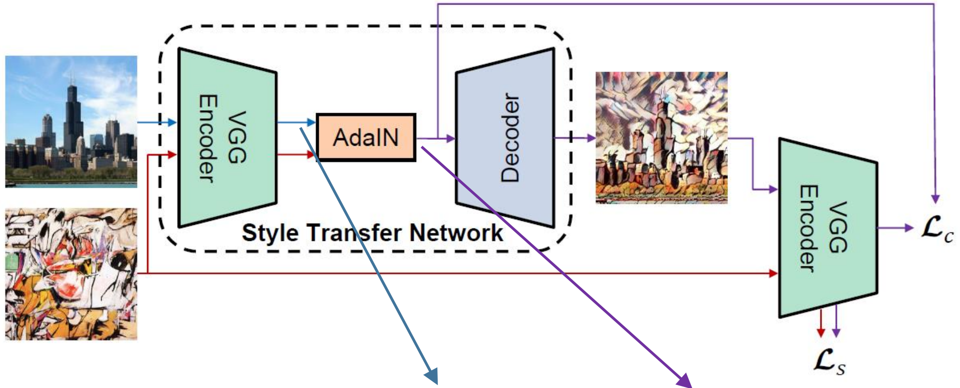




Texture

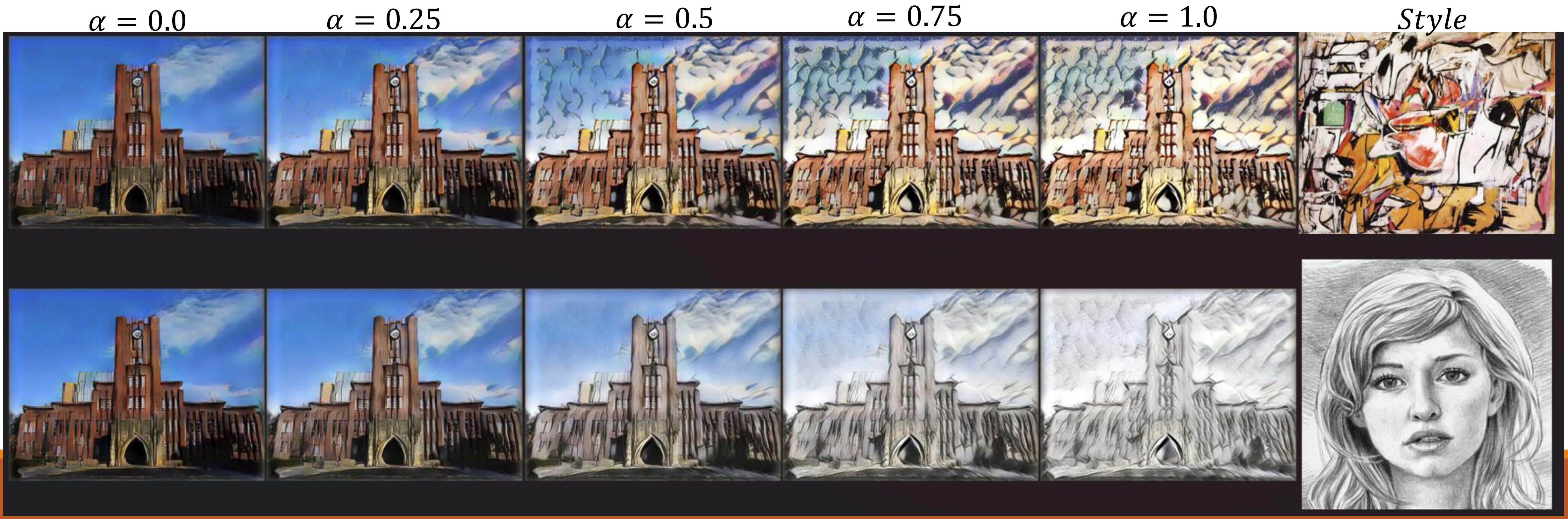
Color

Level of Style Transfer

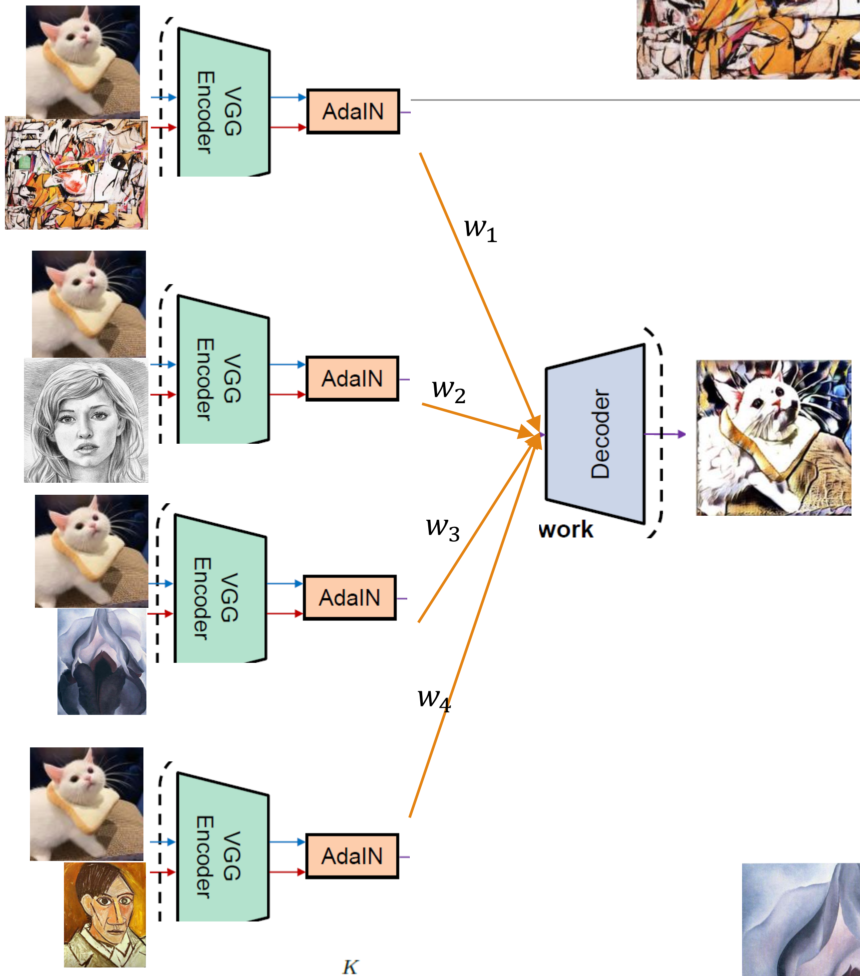


$$T(c,s,\alpha) = g((1-\alpha)\boxed{f(c)} + \alpha\boxed{\mathrm{AdaIN}(f(c), f(s))})$$

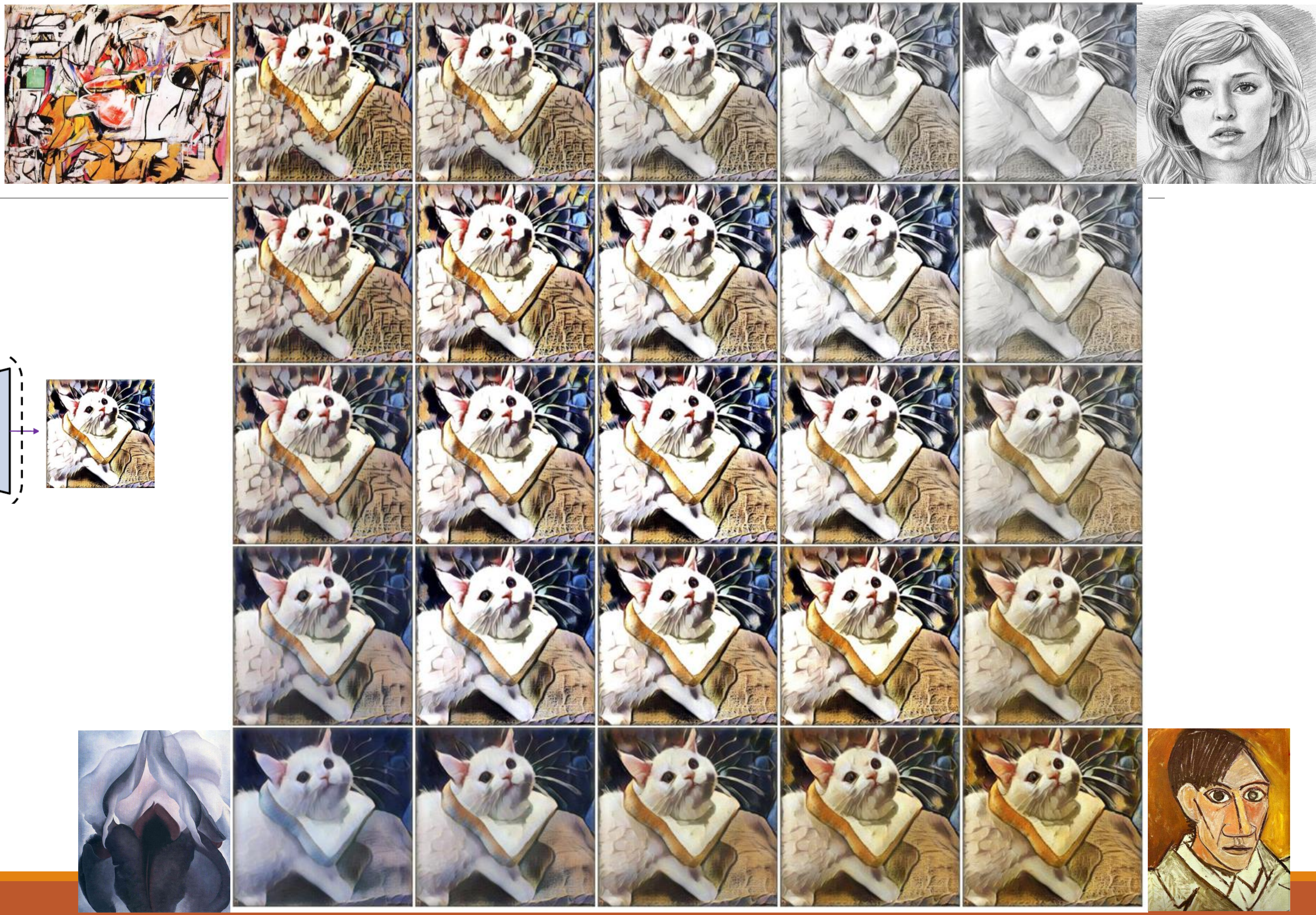
Content Features Stylized Features



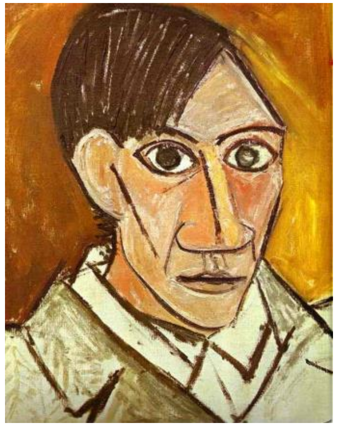
Style Interpolation



$$T(c, s_{1,2,\ldots K}, w_{1,2,\ldots K}) = g\left(\sum_{k=1}^{K} w_k \mathrm{AdaIN}(f(c), f(s_k))\right)$$



Video Style Transform



https://github.com/naoto0804/pytorch-AdaIN/tree/master/input/videos

Speed Analysis

Image Style Transfer time: 0.09213 s/image ≈10 images per second

- NVIDIA RTX 2060
- Image size 512 x 512px
- Averaged over 128 iterations

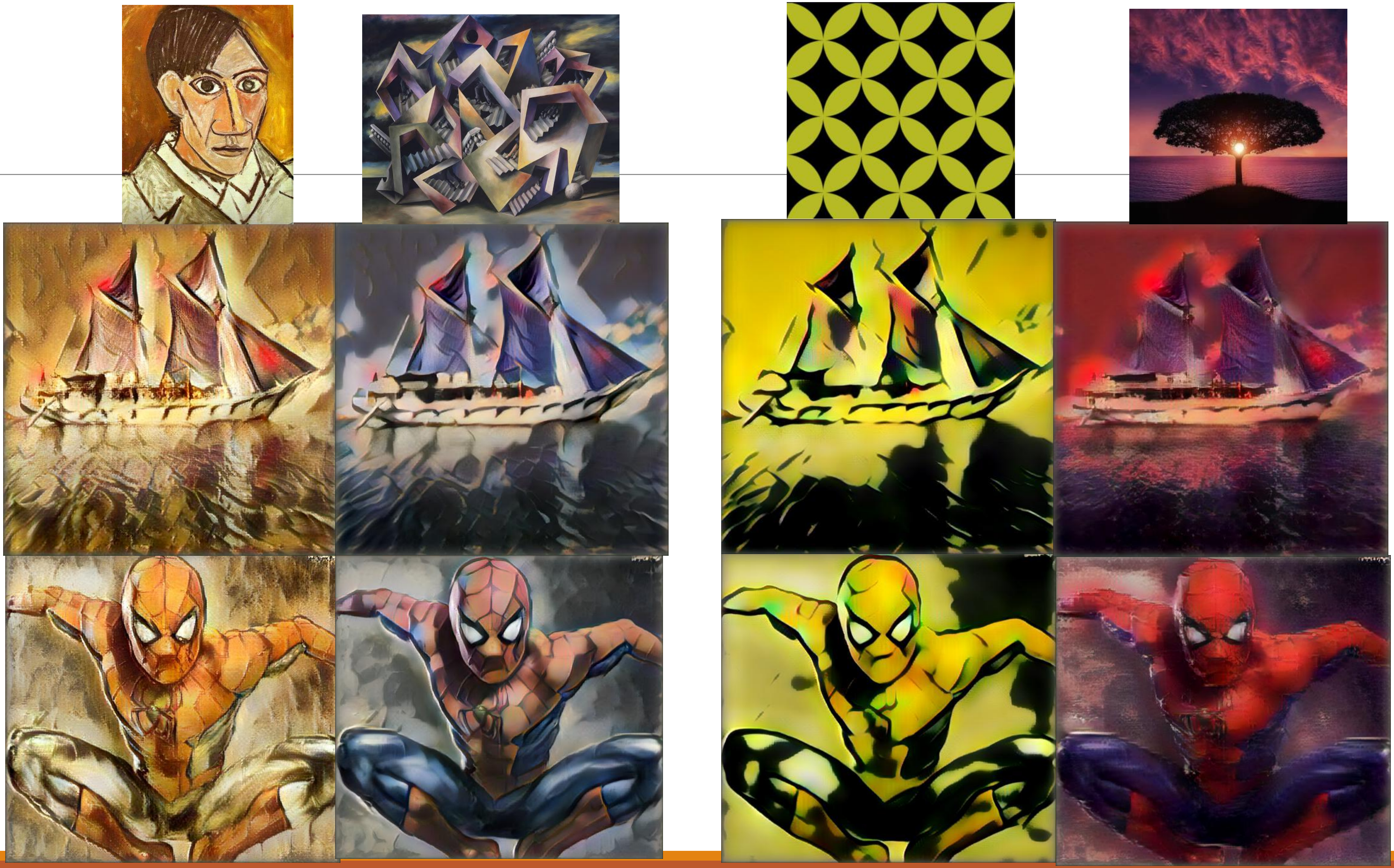
Quality Analysis

Lose more details
when style image not
selected from WikiArt

Decoder is biased against
training dataset.

WikiArt

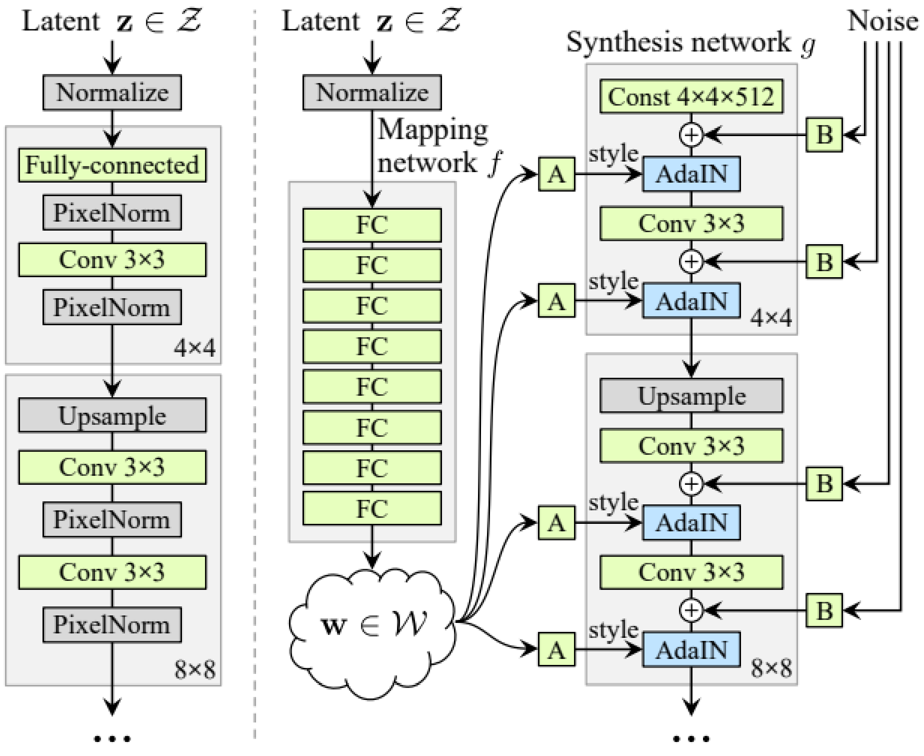
Others



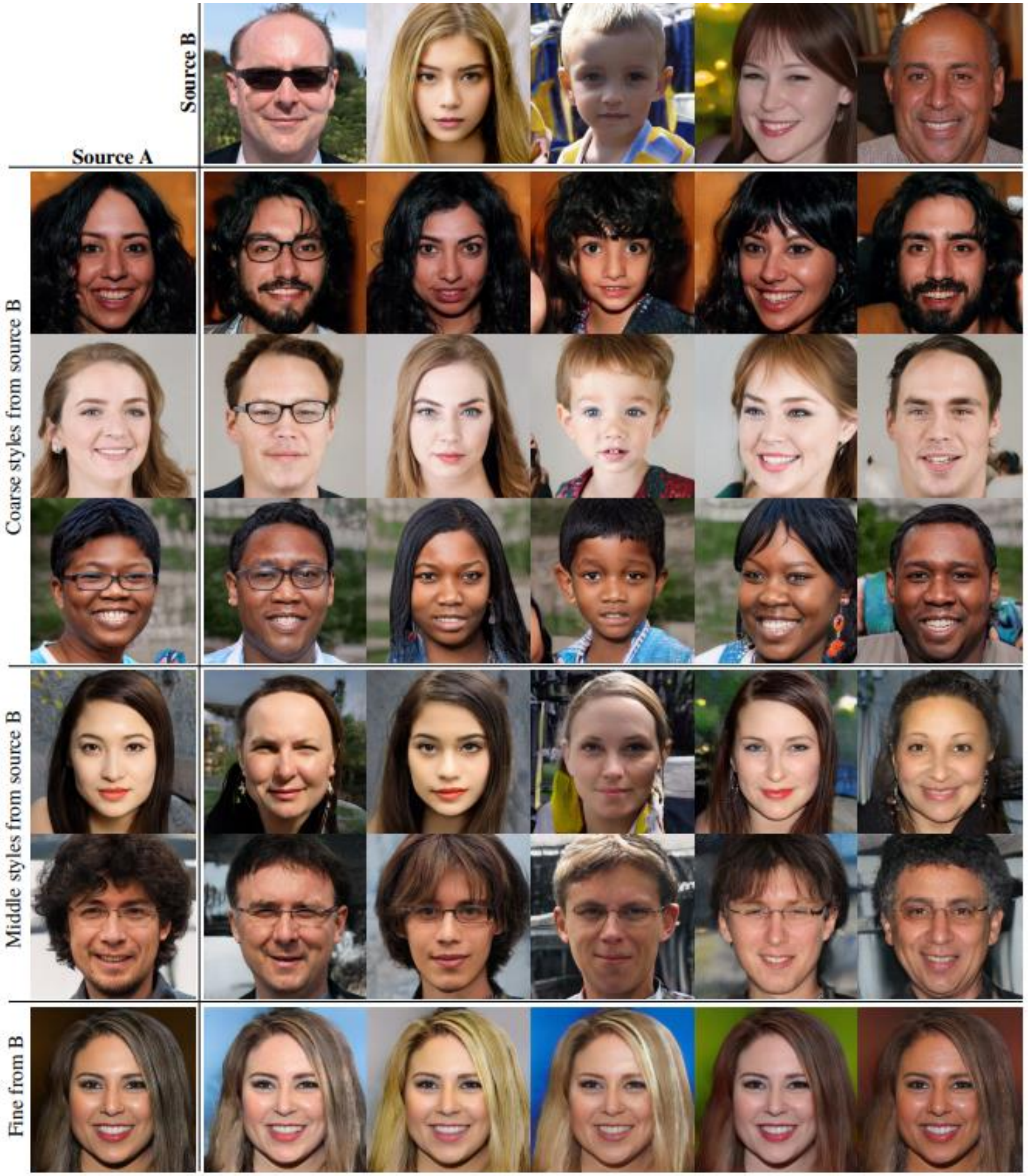
https://www.gruppodani.com/en/some-new-textures-designs-on-the-leather-that-speak-to-the-senses/
https://www.dreamstime.com/snake-skin-pattern-texture-repeating-seamless-fashionable-print-ready-textile-prints-image165760557

Applications

A Style-Based Generator Architecture for Generative Adversarial Networks (StyleGAN), CVPR 2019



(a) Traditional (b) Style-based generator



Pull Requests

Huang et al.,
ICCV 2017

Color Control

Figure 9. Color control. Left: content and style images. Right: color-preserved style transfer result.

Spatial Control

Figure 10. Spatial control. Left: content image. Middle: two style images with corresponding masks. Right: style transfer result.

Anything Else !