Lecture 10: CNNs – Visualizing and Understanding

Lan Xu
SIST, ShanghaiTech
Fall, 2022

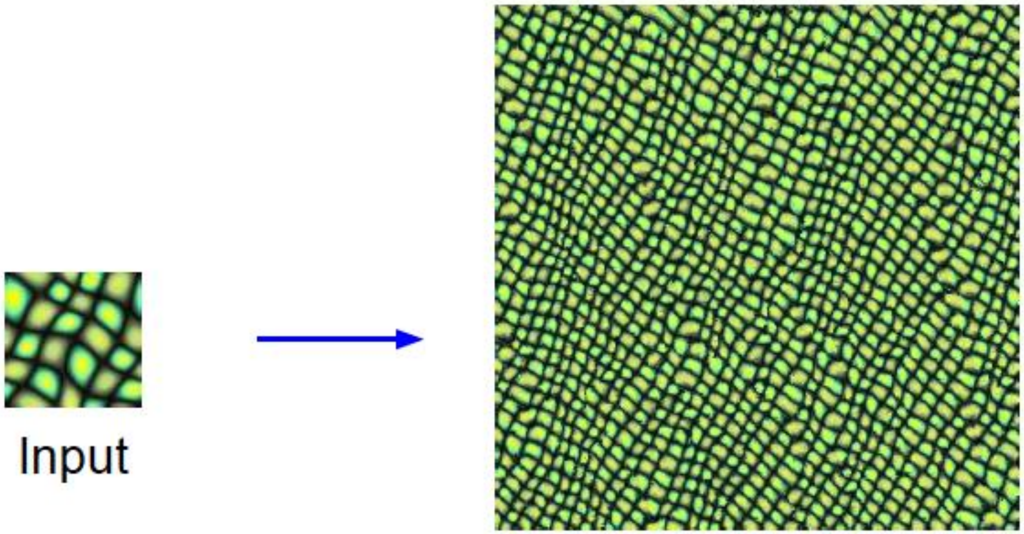
Outline

- Understanding CNN through visualization
  - Visualizing filters: Network weights
  - Visualizing neural activations: Network outputs
  - Visualizing sensitivities: Network inputs
- Case studies
  - Adversarial examples
  - DeepDreams
  - Neural texture synthesis and style transfer

Texture Synthesis

- Problem setup

Given a sample patch of some texture, can we generate a bigger image of the same texture?

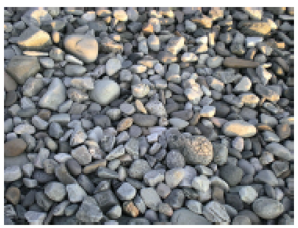


Input

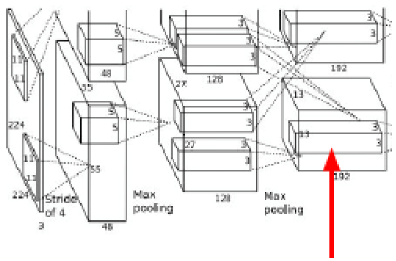
Output

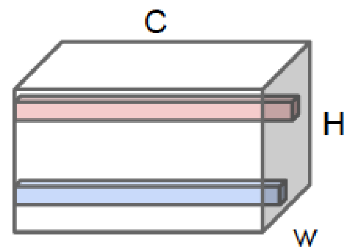
Texture Synthesis

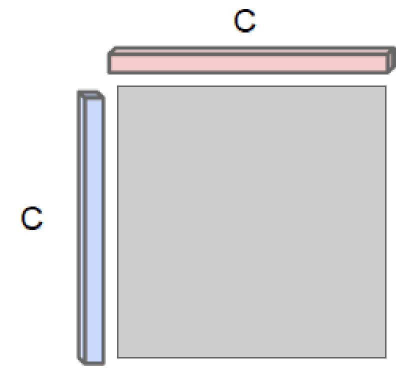
- CNN-based modeling of image statistics



This image is in the public domain.





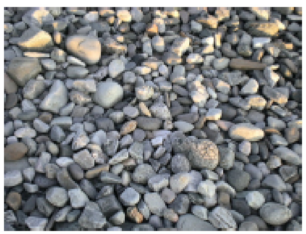


Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

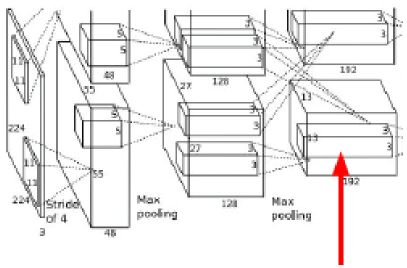
Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

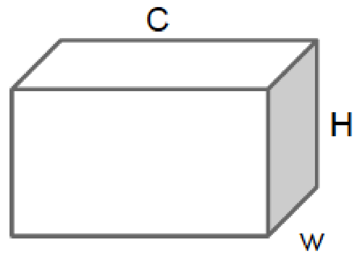
Texture Synthesis

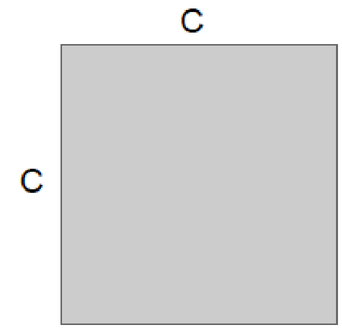
- CNN-based modeling of image statistics



This image is in the public domain.







Gram Matrix

Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

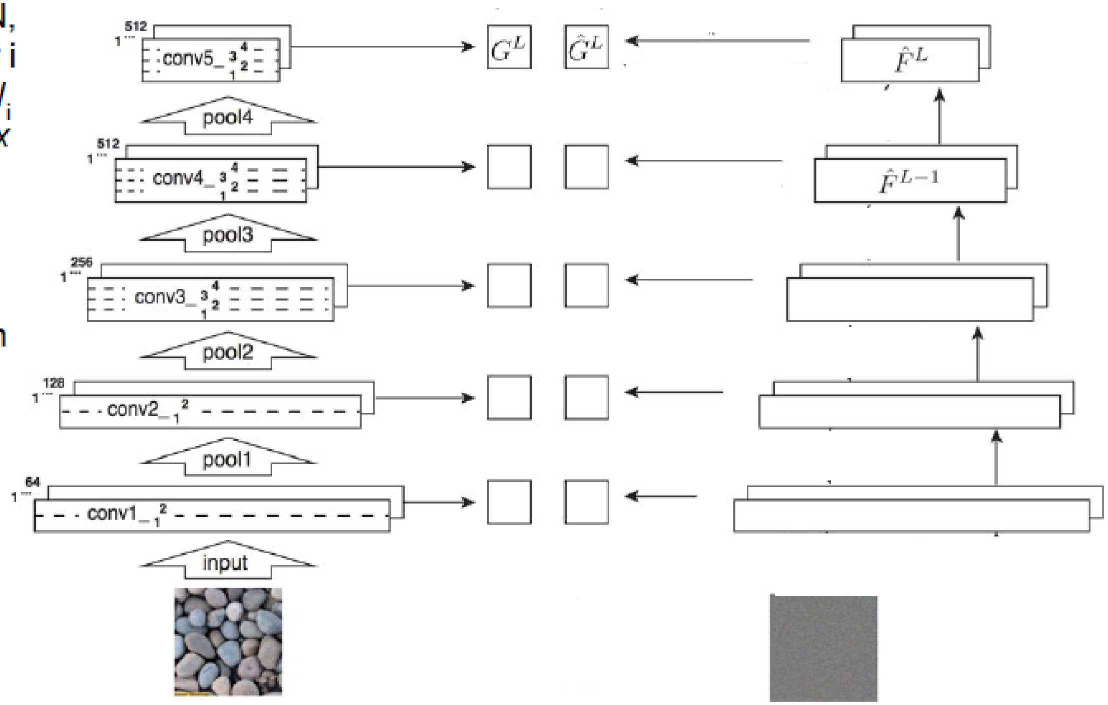
Average over all HW pairs of vectors, giving **Gram matrix** of shape C x C

- Neural texture synthesis

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l} \text{ (shape } C_i \times C_i\text{)}$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Texture Synthesis

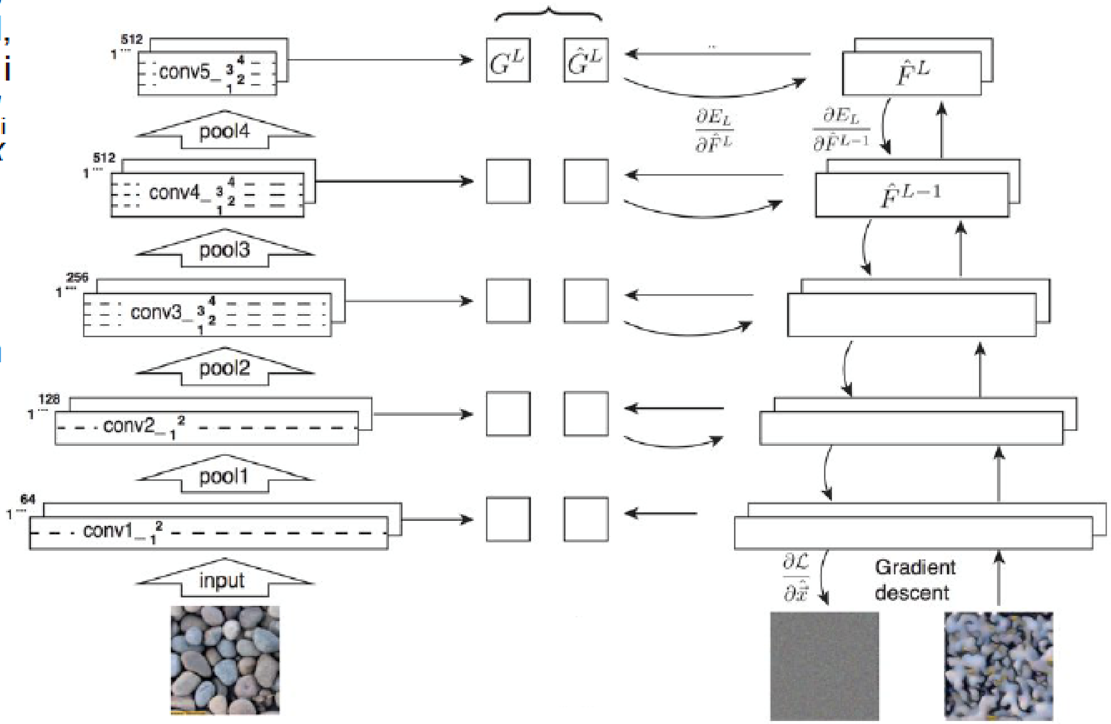
- Neural texture synthesis

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \text{ (shape } C_i \times C_i)$$

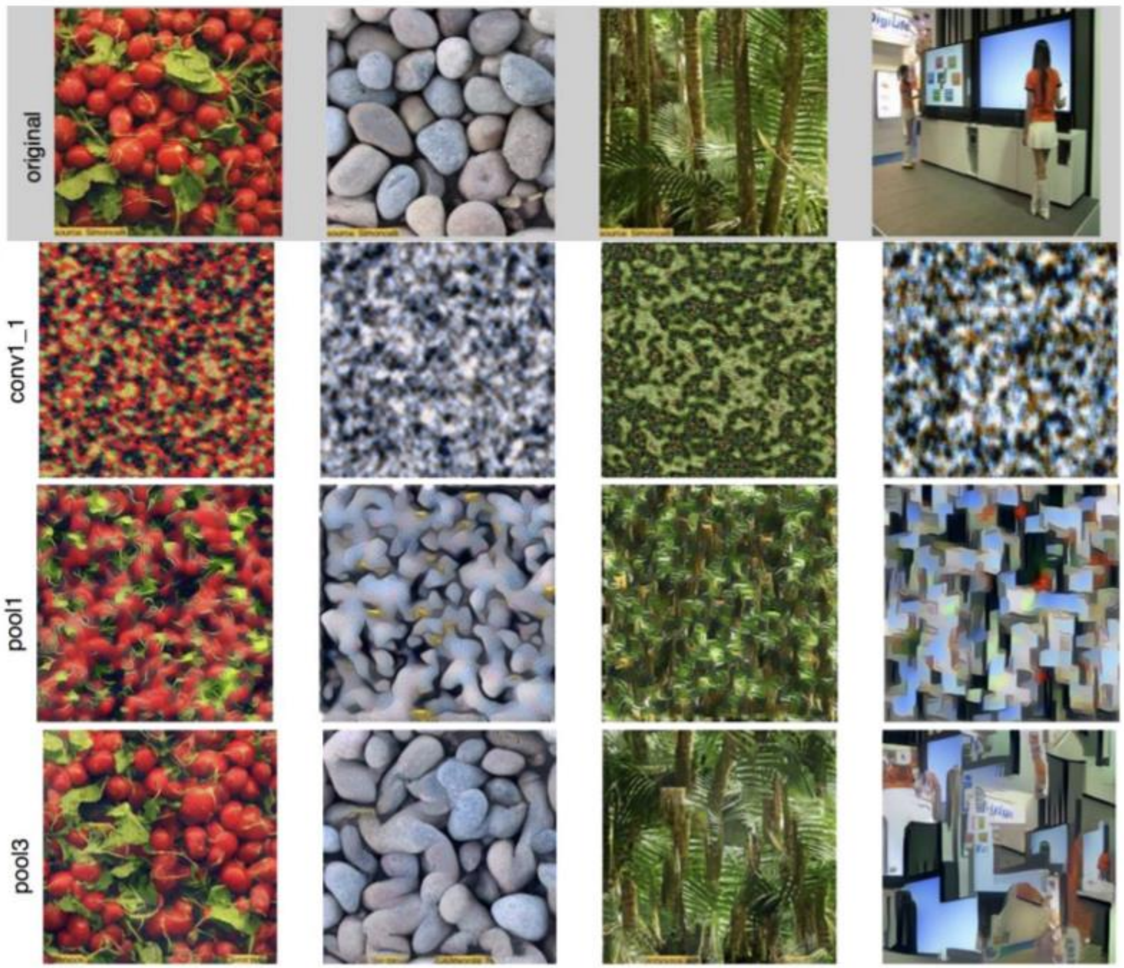
4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices
7. Backprop to get gradient on image
8. Make gradient step on image
9. GOTO 5

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - \hat{G}^l_{ij} \right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$



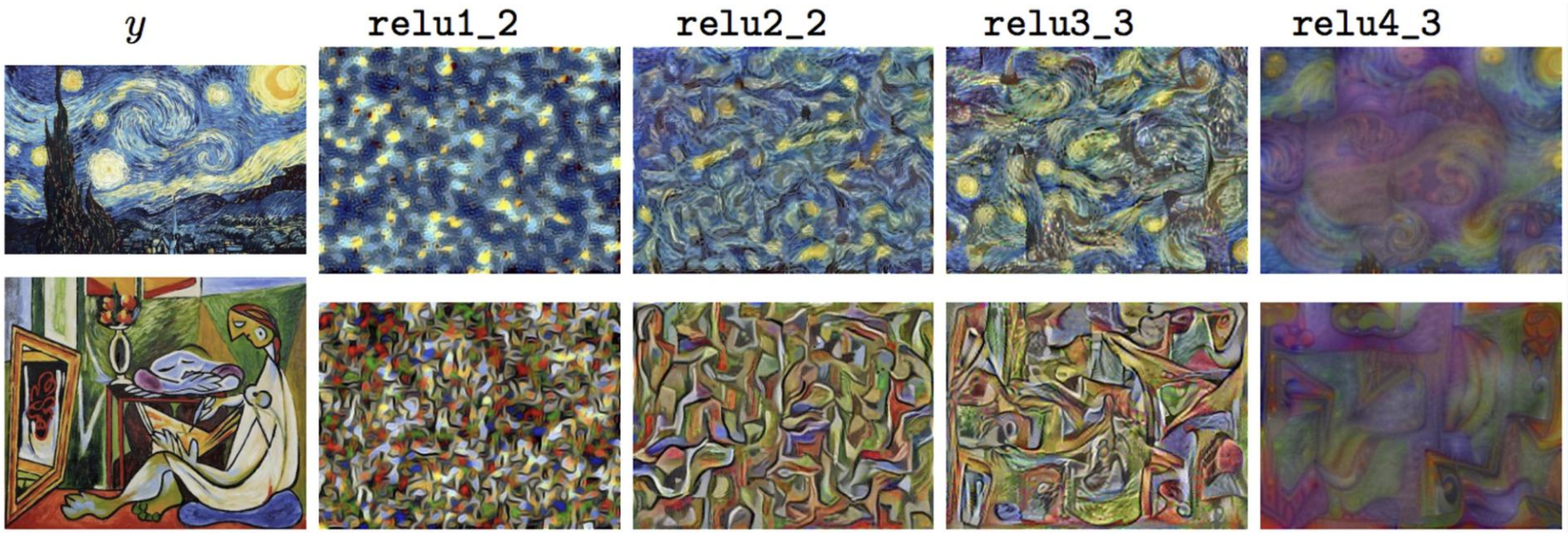
Texture Synthesis

- Neural texture synthesis



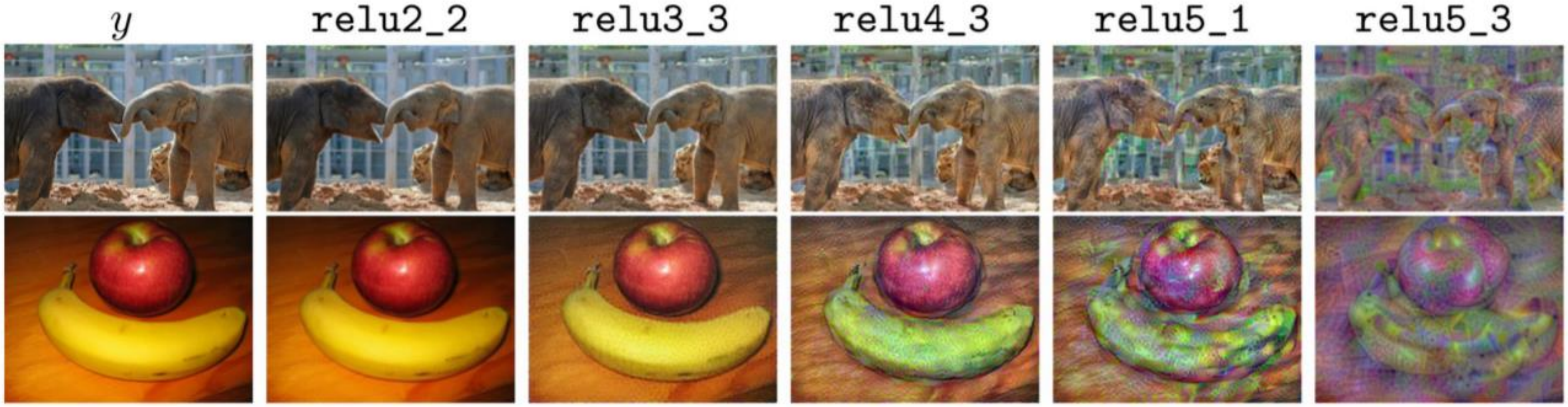
Texture Synthesis

- In terms of Gram Reconstruction



Recall Feature inversion

Reconstructing from different layers of VGG-16



Given a CNN feature vector for an image, find a new image that:

- Matches the given feature vector
- "looks natural" (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^{H\times W\times C}}{\operatorname{argmin}} \ \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda\mathcal{R}(\mathbf{x})$$

Given feature vector

Features of new image

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j}\left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2\right)^{\frac{\beta}{2}}$$

Total Variation regularizer (encourages spatial smoothness)

Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015

Neural Style Transfer

- Problem setup

Content Image

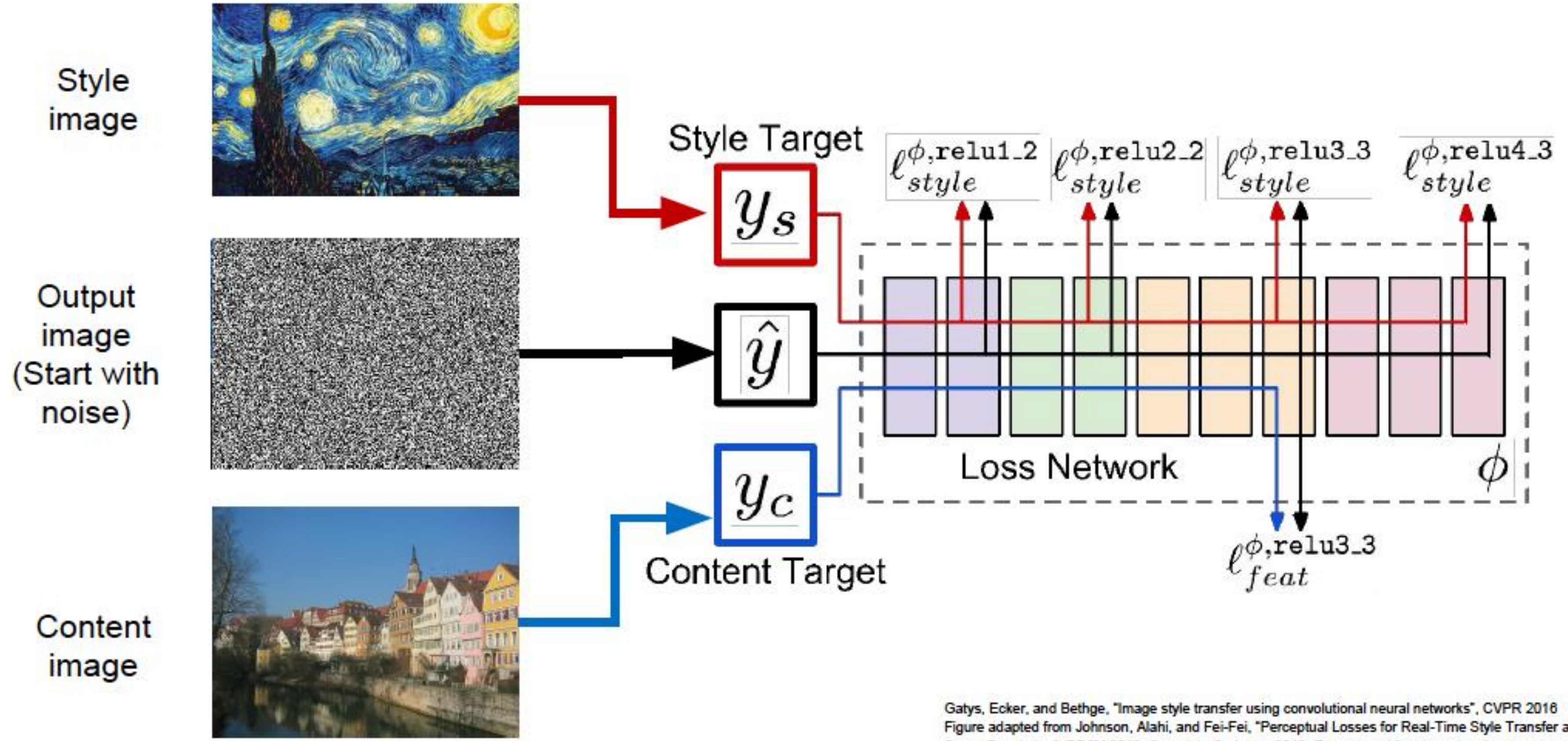
+

Style Image

=

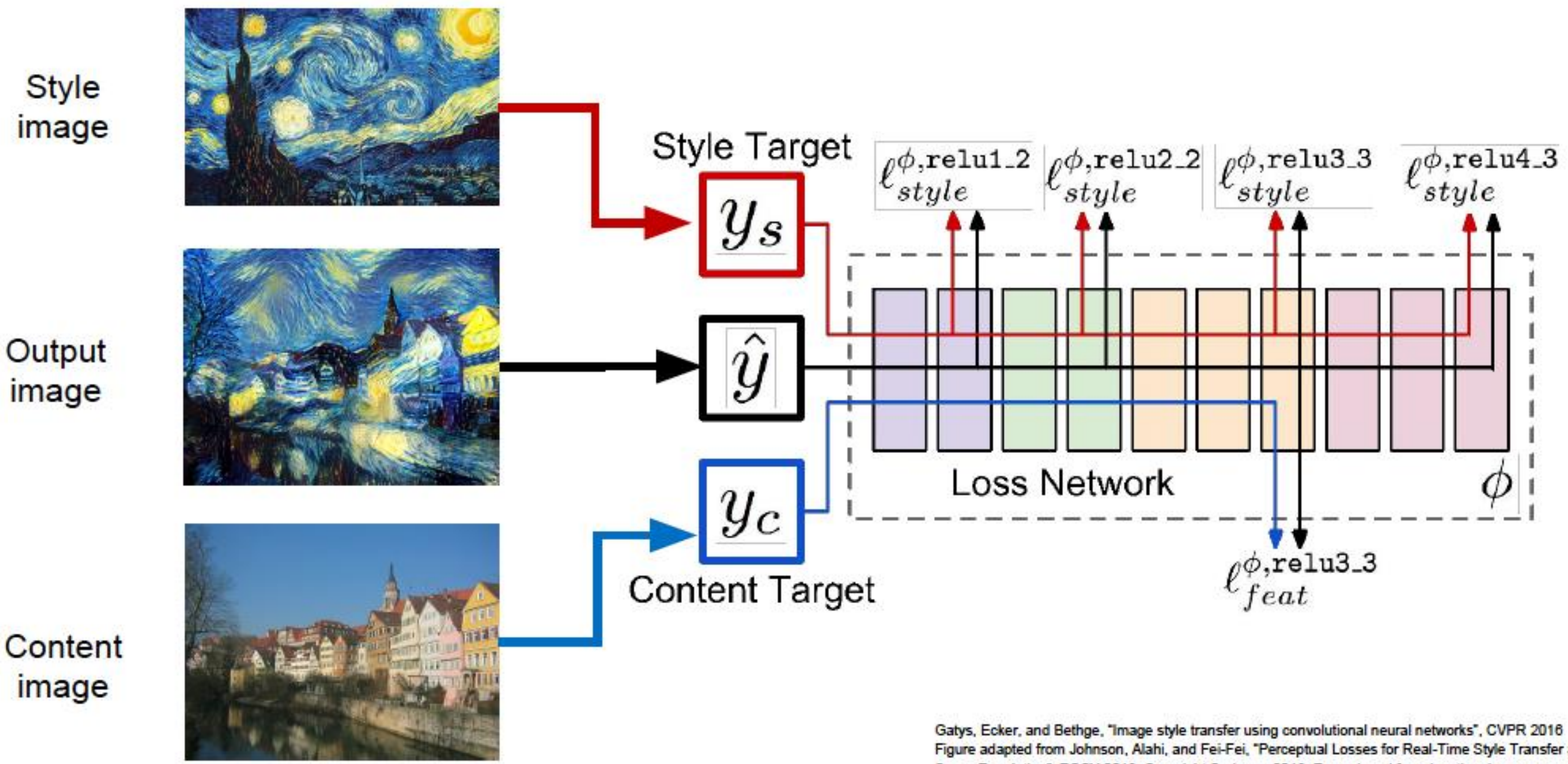
Style Transfer!

Neural Style Transfer



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

Neural Style Transfer

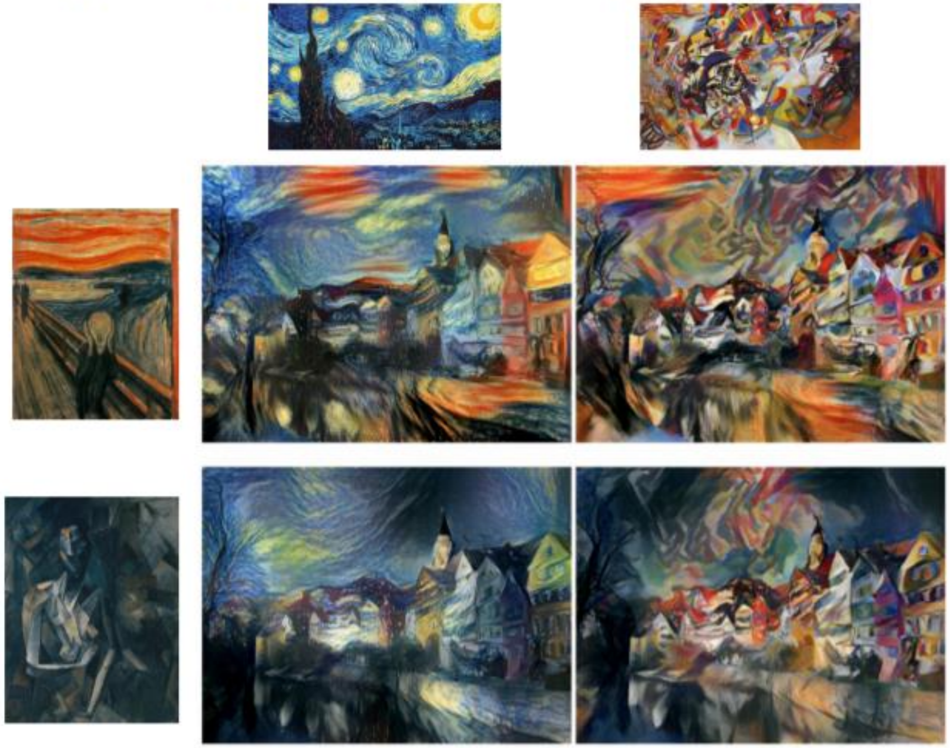


Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

Neural Style Transfer

Neural Style Transfer

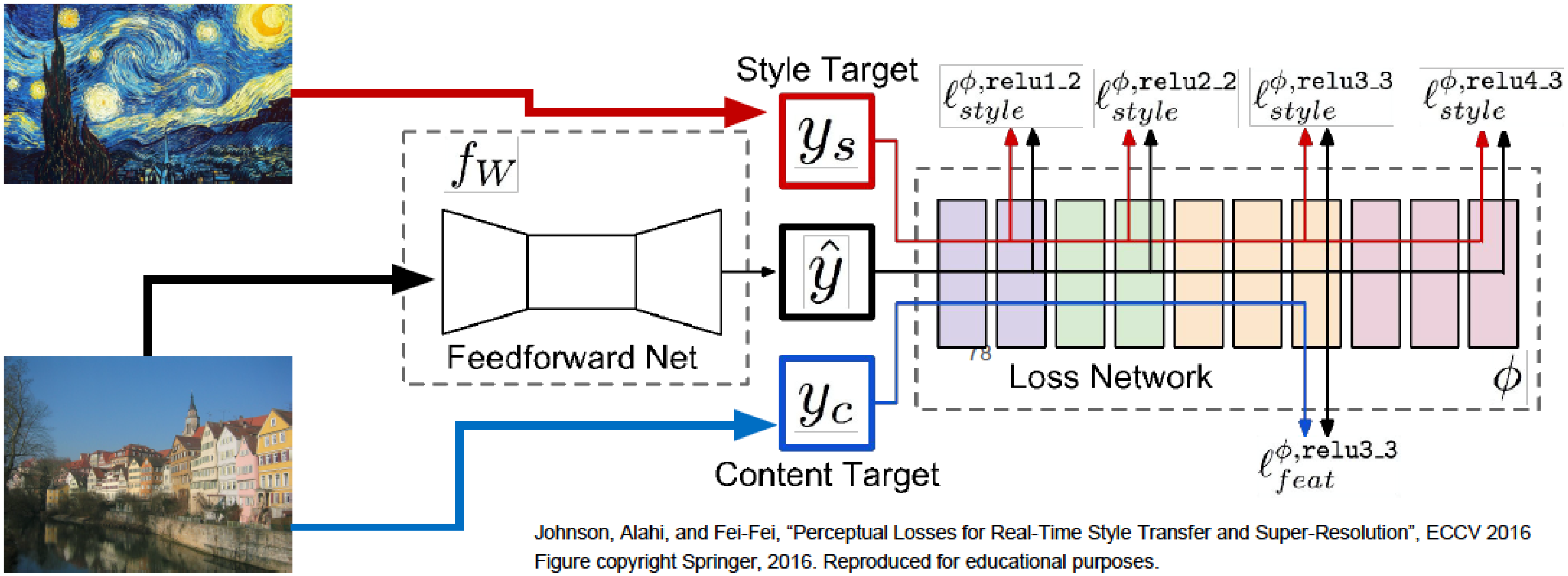
Mix style from multiple images by taking a weighted average of Gram matrices



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure copyright Justin Johnson, 2015.

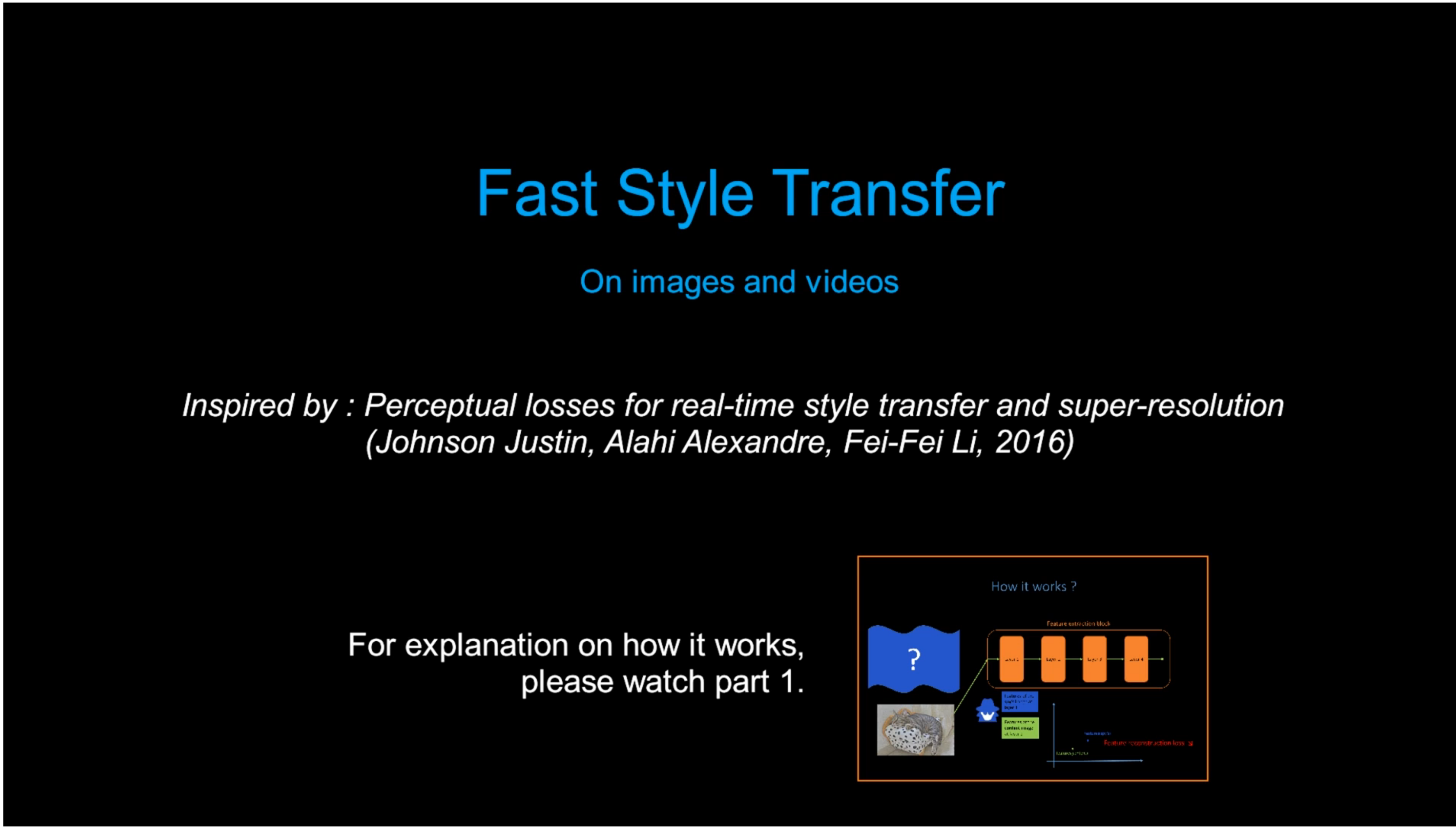
Fast Style Transfer

(1) Train a feedforward network for each style
(2) Use pretrained CNN to compute same losses as before
(3) After training, stylize images using a single forward pass



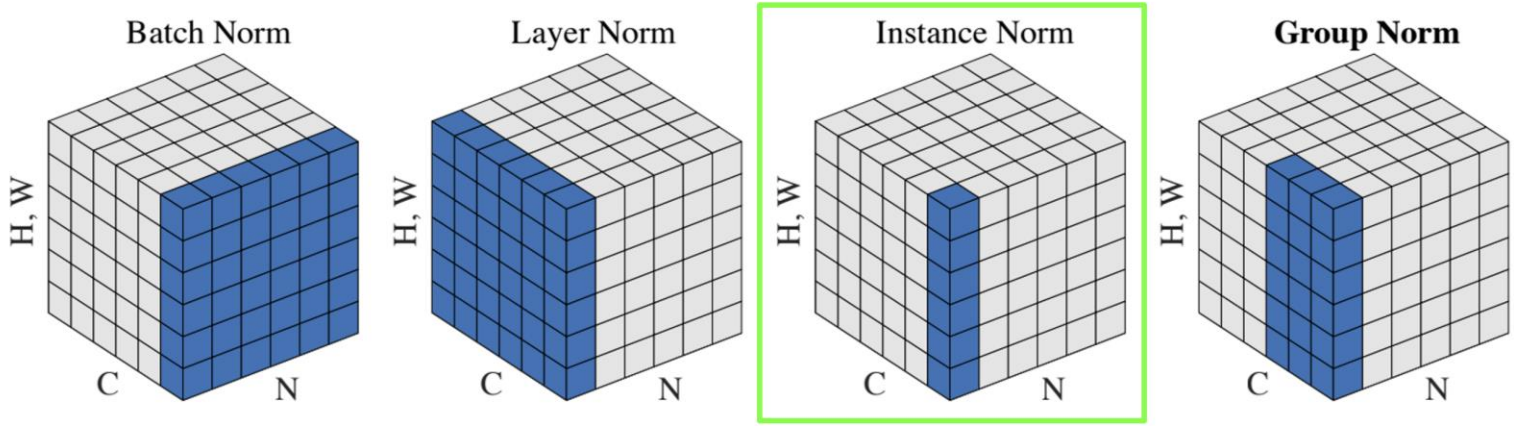
Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016

Fast Style Transfer



Recall Instance Normalization

- Instance Normalization was developed for style transfer!



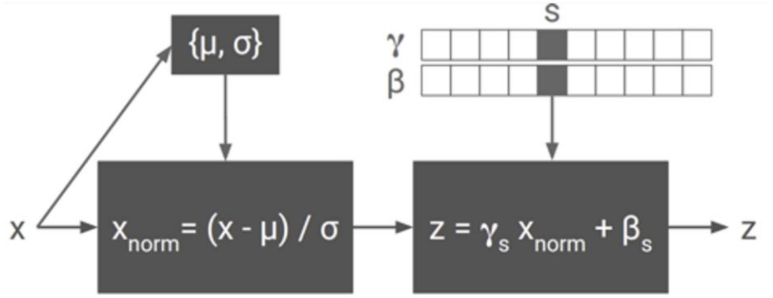
Ulyanov et al, “Texture Networks: Feed-forward Synthesis of Textures and Stylized Images”, ICML 2016
Ulyanov et al, “Instance Normalization: The Missing Ingredient for Fast Stylization”, arXiv 2016

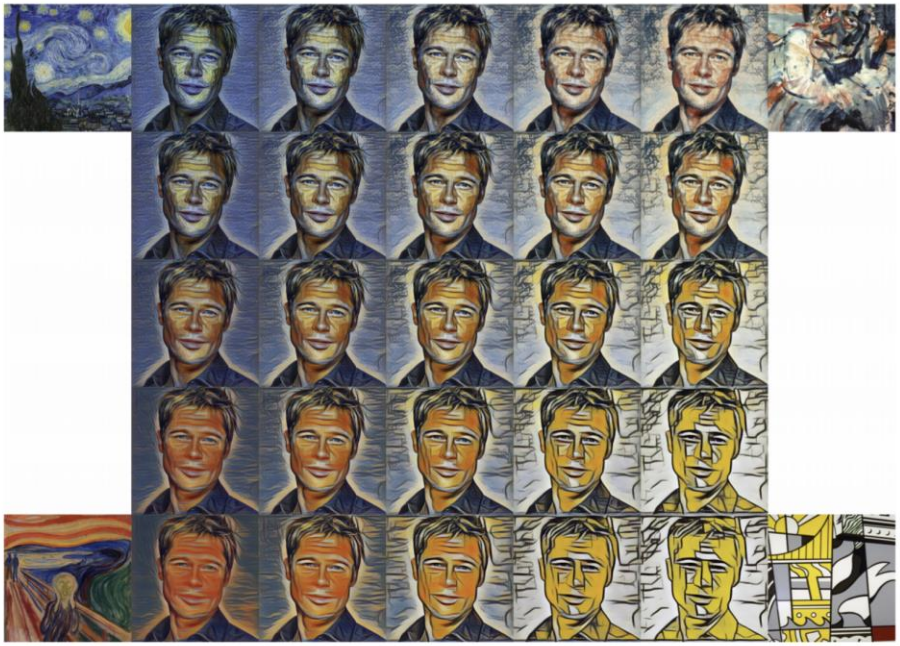
Fast Style Transfer

- Replacing BN with IN improves results!

One Network, Many Styles

- Same network for multiple styles
- Conditional Instance Normalization: learn separate scale and shift parameters per style

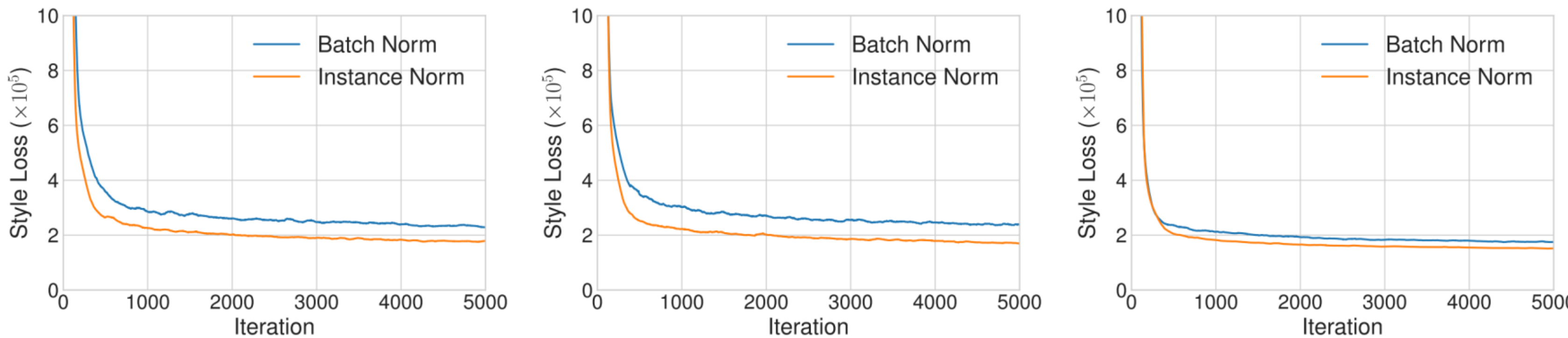




Dumoulin, Shlens, and Kudlur, "A Learned Representation for Artistic Style", ICLR 2017

Adaptive Instance Normalization

- Why IN is better than BN?
- Why CIN can model various styles?



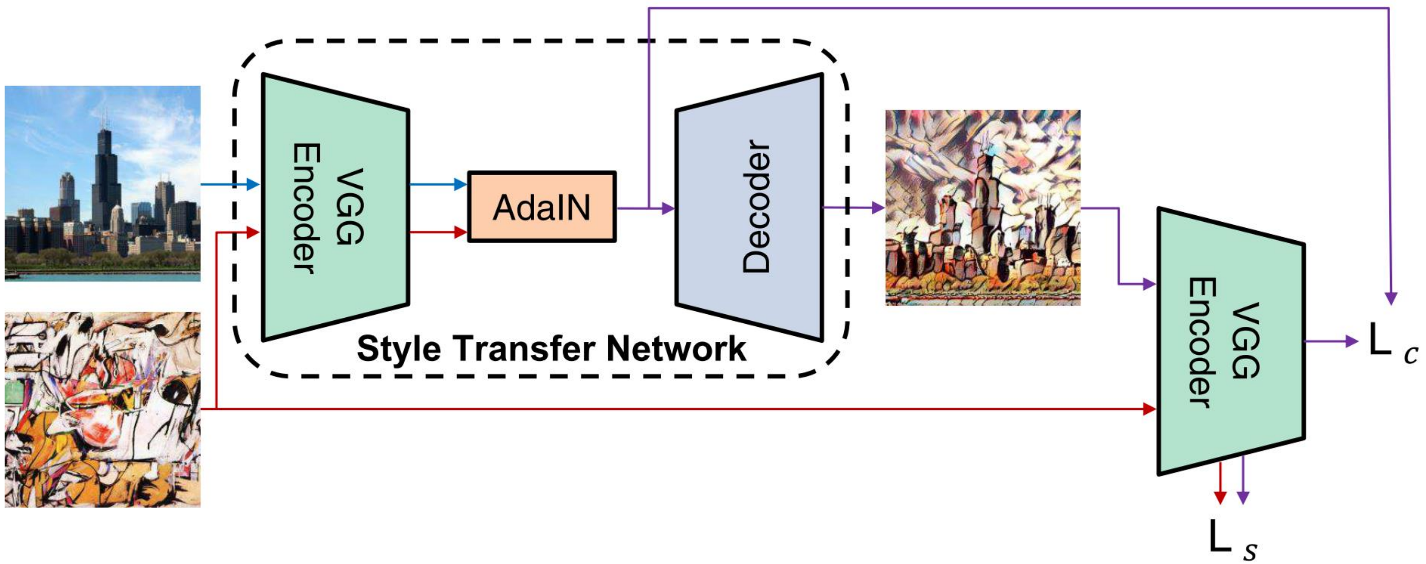
(a) Trained with original images. (b) Trained with contrast normalized images. (c) Trained with style normalized images.

Figure 1. To understand the reason for IN's effectiveness in style transfer, we train an IN model and a BN model with (a) original images in MS-COCO [36], (b) contrast normalized images, and (c) style normalized images using a pre-trained style transfer network [24]. The improvement brought by IN remains significant even when all training images are normalized to the same contrast, but are much smaller when all images are (approximately) normalized to the same style. Our results suggest that IN performs a kind of style normalization.

Huang et al, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization", ICCV 2017

Adaptive Instance Normalization

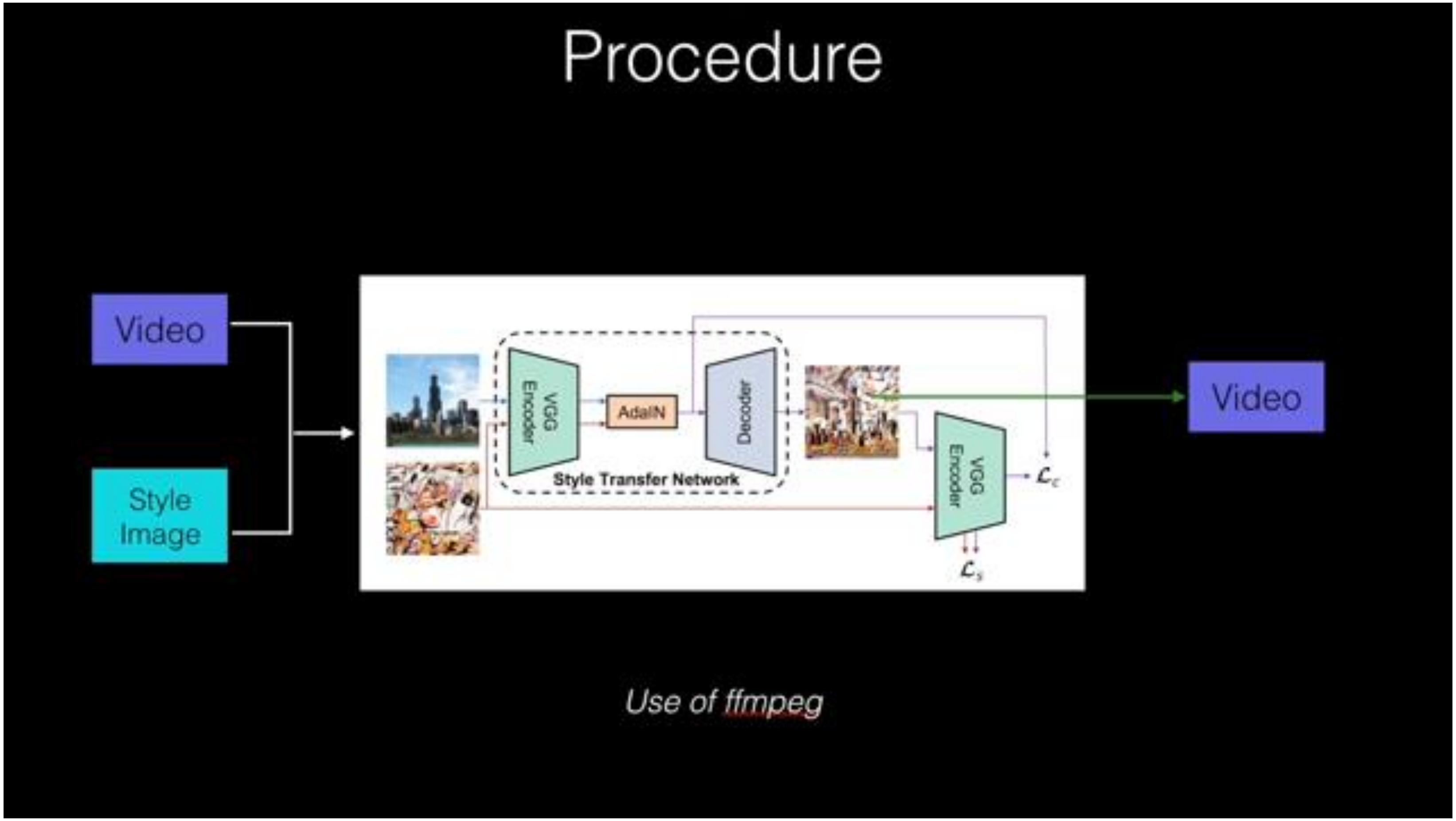
- x: content image; y: style image



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

Huang et al, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization", ICCV 2017

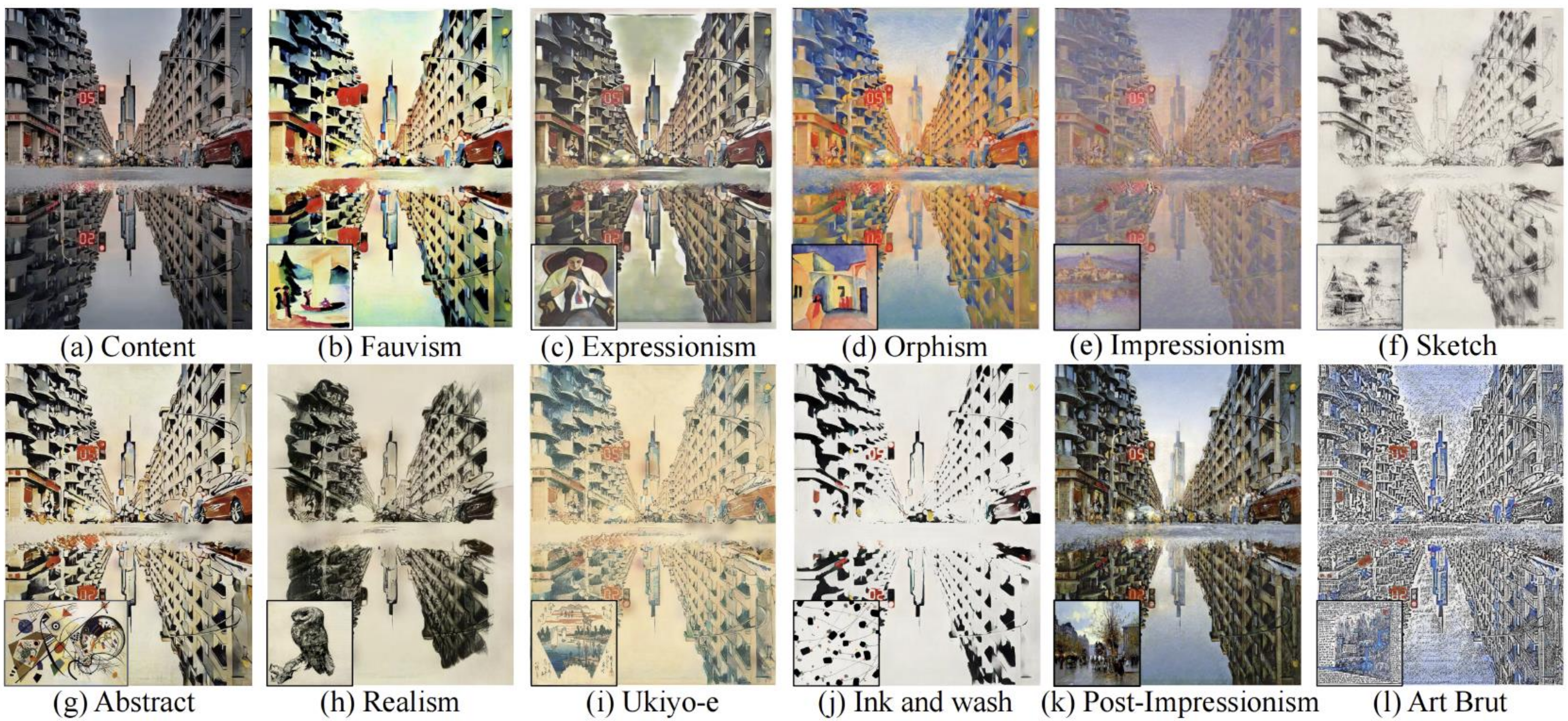
Adaptive Instance Normalization



Huang et al, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization", ICCV 2017

Recent advances

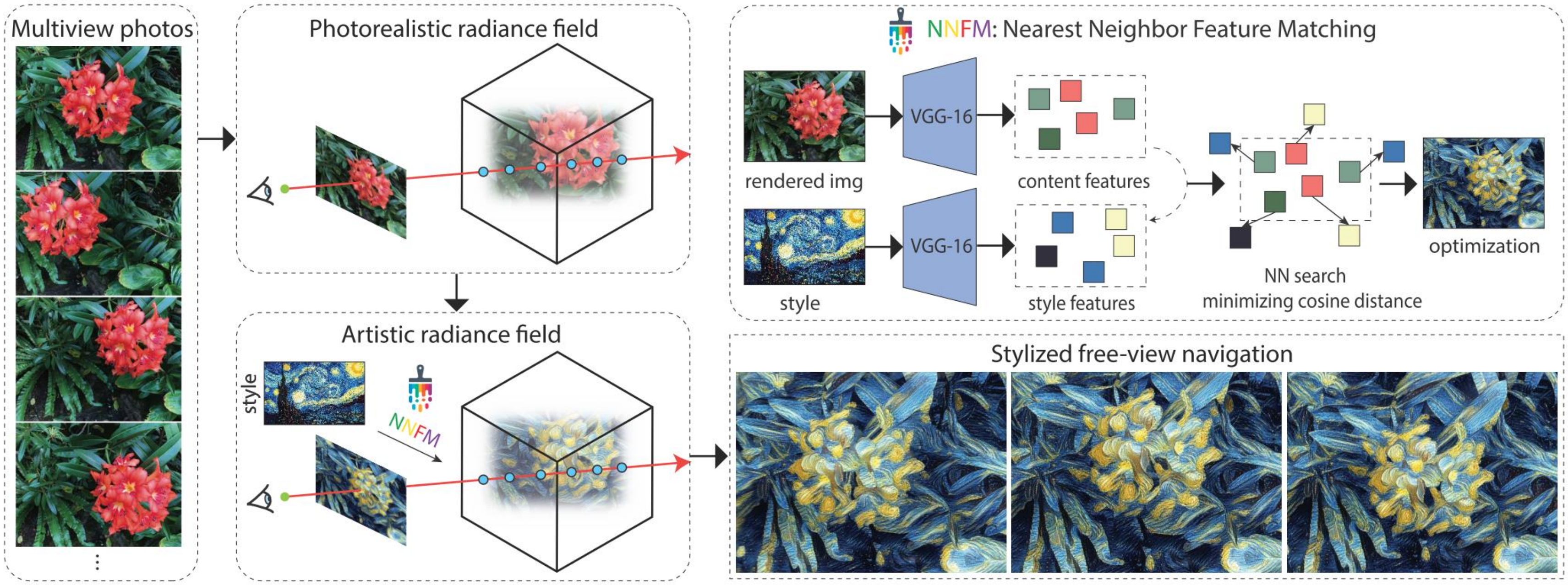
- More complex style representation than second-order statistics



Zhang et al, “Domain Enhanced Arbitrary Image Style Transfer via Contrastive Learning (CAST)”, SIGGRAPH 2022

Recent advances

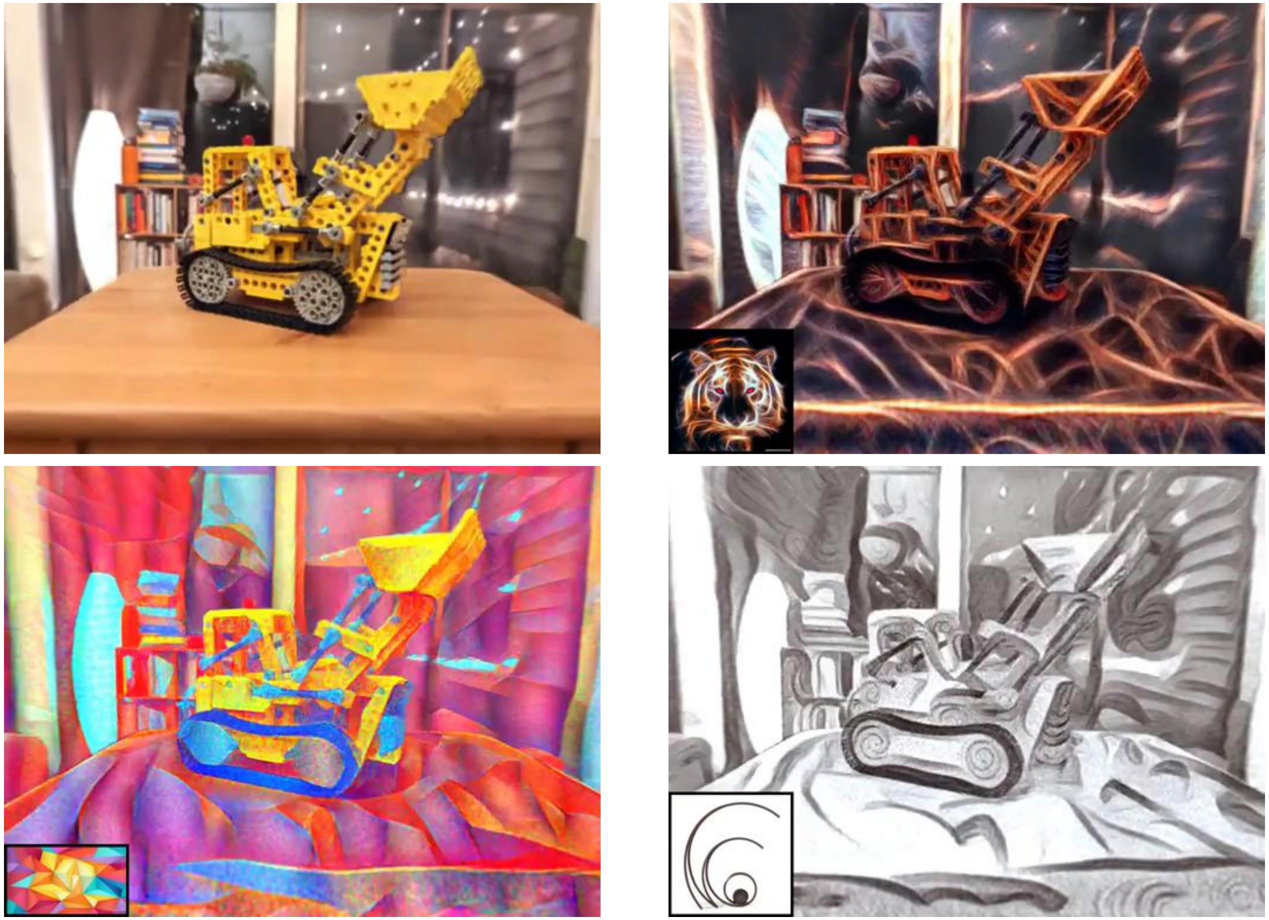
- From 2D to 3D using Neural Radiance Field (NeRF)



Zhang et al, “ARF: Artistic Radiance Fields”, ECCV 2022

Recent advances

- From 2D to 3D using Neural Radiance Field (NeRF)



Zhang et al, “ARF: Artistic Radiance Fields”, ECCV 2022

Summary

- CNNs in computer vision
  - Many and more applications
  - Still lack of deep understanding

- Quiz-4: totally online! Please send the result through gradscope (refer to PIAZZA)!

- Next time:
  - Recurrent Neural Networks