Lecture 11: Neural networks for Prediction - II

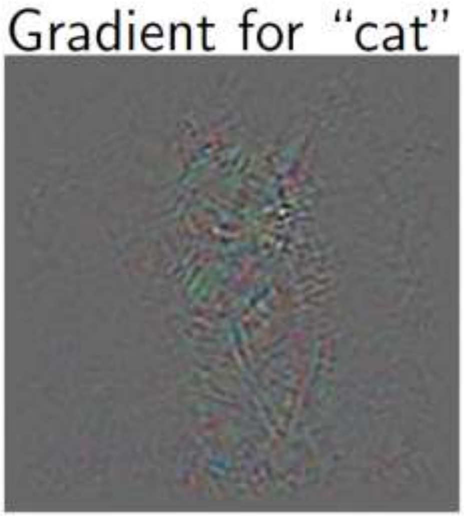
Lan Xu
SIST, ShanghaiTech
Fall, 2023

Outline

- Visualizing sensitivities: Network inputs
- Case Study
  - Adversarial examples
  - DeepDreams
  - Neural texture synthesis and style transfer

Visualizing input gradient

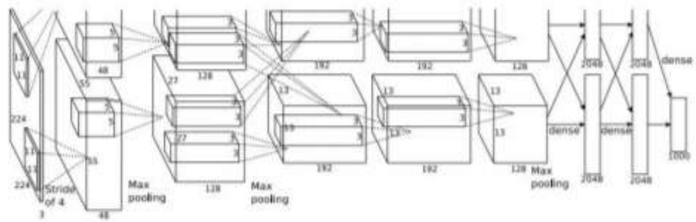
- Take a trained object classification network (AlexNet) and compute the gradient of $\log P(y = \text{“}cat\text{”} | \mathbf{x})$



Visualizing input gradient

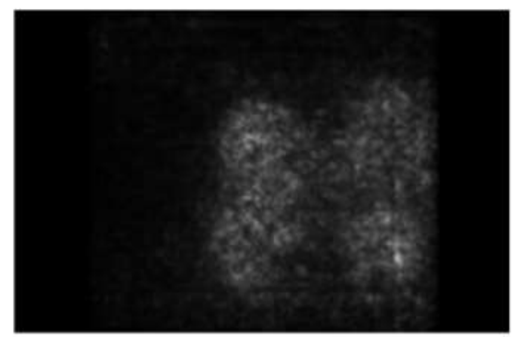
- Take a trained object classification network (AlexNet) and compute the gradient of class score

Forward pass: Compute probabilities



Dog

Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Synthesizing input images

- Gradient ascent on an image to maximize the activation of a given neuron

(Guided) backprop:
Find the part of an image that a neuron responds to

Gradient ascent:
Generate a synthetic image that maximally activates a neuron

$$I^* = \arg\max_I \underbrace{\boxed{f(I)}}_{\text{Neuron value}} + \underbrace{\boxed{R(I)}}_{\text{Natural image regularizer}}$$

Synthesizing input images

- Dataset examples vs. optimized input

Baseball—or stripes? *mixed4a, Unit 6*

Animal faces—or snouts? *mixed4a, Unit 240*

Clouds—or fluffiness? *mixed4a, Unit 453*

Buildings—or sky? *mixed4a, Unit 492*

Synthesizing input images

- Feature inversion

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

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

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

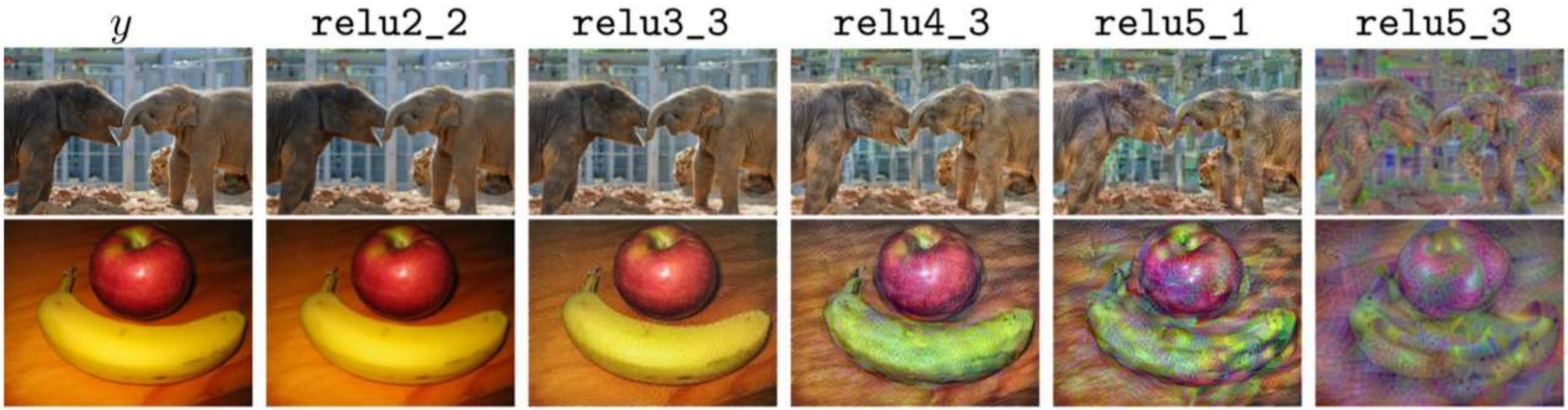
Total Variation regularizer
(encourages spatial smoothness)

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

Synthesizing input images

- Feature inversion

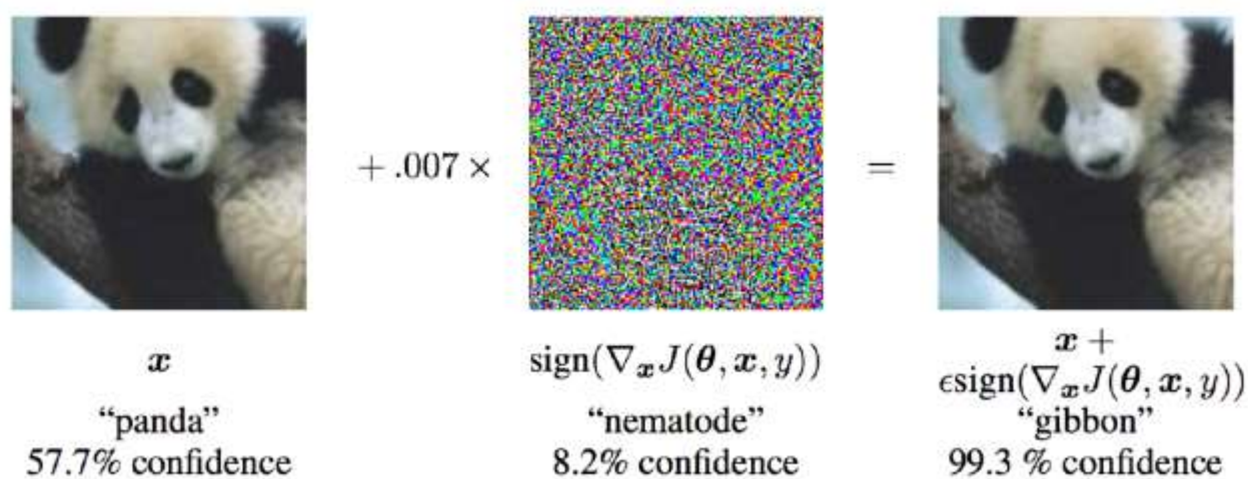
Reconstructing from different layers of VGG-16



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

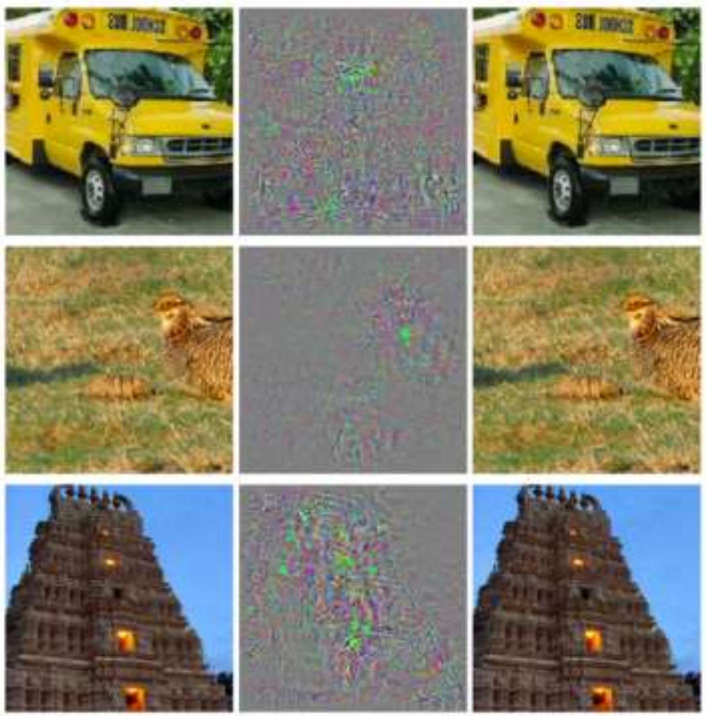
Adversarial Examples

- Surprising findings: adversarial inputs
  - Inputs optimized to fool an algorithm
- Given an image for one category (e.g., “cat”), compute the input gradient to maximize the network’s output for a different category (e.g., “dog”)
  - Perturb the image very slightly in this direction and the network will change its prediction
  - Fast gradient sign method: take the sign of the entries in the gradient



Adversarial Examples

- The following adversarial examples are misclassified as ostriches (Middle = perturbation x 10)



Adversarial Examples

- 2013: ha ha, how cute!
  - "Intriguing Properties of Neural Networks"
- 2018: serious security threat
  - Nobody has found a reliable method yet to defend against them
    - 7 of 8 proposed defenses accepted to ICLR 2018 were cracked within days
  - Adversarial examples transfer to different networks trained on a totally separate training set
  - You don't need access to the original network; you can train up a new network to match its predictions, and then construct adversarial examples for that
    - Attack carried out against proprietary classification networks accessed using prediction APIs (MetaMind, Amazon, Google)

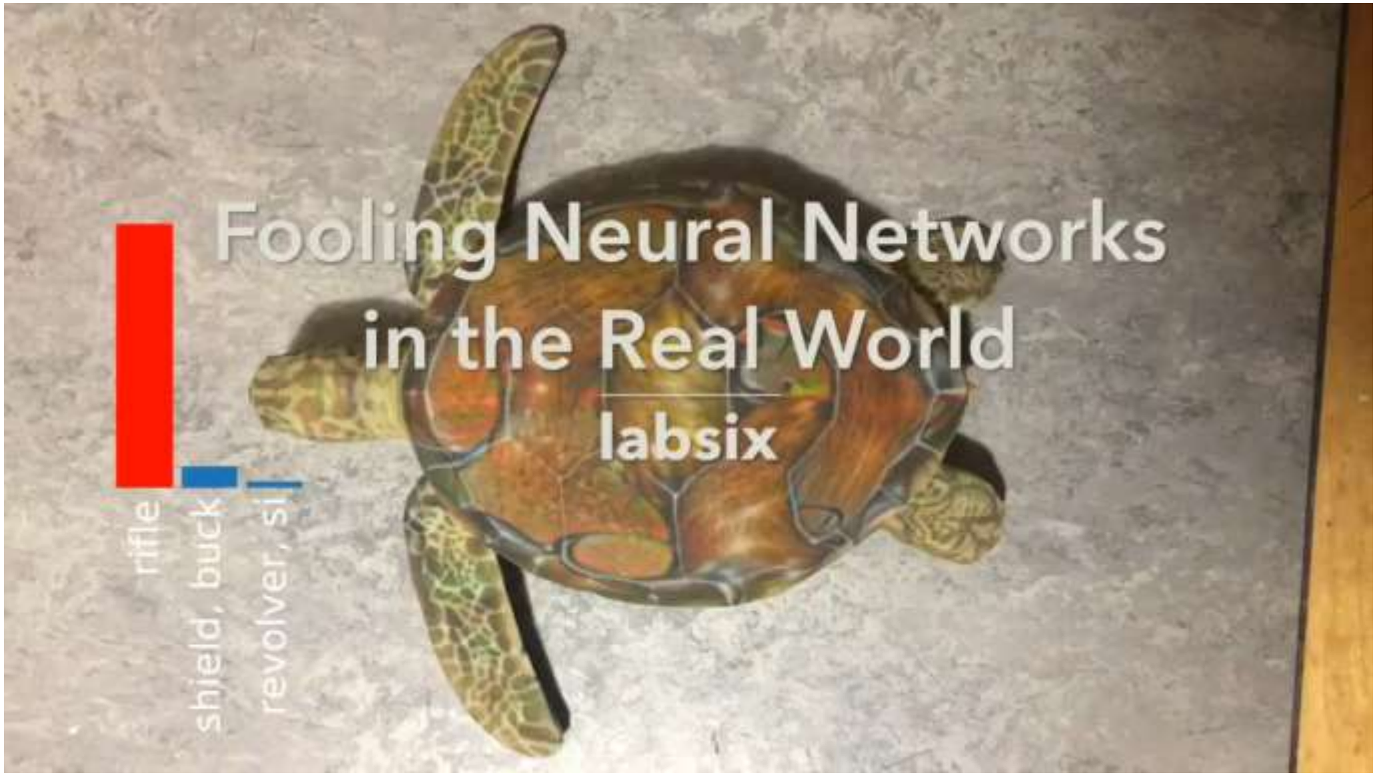
Adversarial Examples

- You can print out an adversarial image and take a picture of it, and it still works!

(a) Printout (b) Photo of printout (c) Cropped image

Adversarial Examples

- An adversarial example in the physical world
  - Network thinks it is a gun from a variety of viewing angles



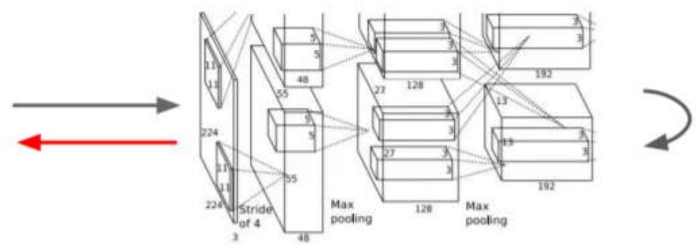
Deep Dream

- Start with an image, and run a ConvNet on it.
- Pick a layer in the network.
- Change the image such that units which were already highly activated get activated even more strongly. "Rich get richer."
- Repeat
- This will accentuate whatever features of an image already kind of resemble the object.

Deep Dream

- Method

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network



Choose an image and a layer in a CNN; repeat:
1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

Equivalent to:
$$I^* = \arg\max_I \sum_i f_i(I)^2$$

Mordvintsev, Olah, and Tyka, "Inceptionism: Going Deeper into Neural Networks", Google Research Blog.

Deep Dream

"Admiral Dog!" "The Pig-Snail" "The Camel-Bird" "The Dog-Fish"

Deep Dream

Recent Attempt

- Plug-In Inversion: https://arxiv.org/pdf/2201.12961.pdf

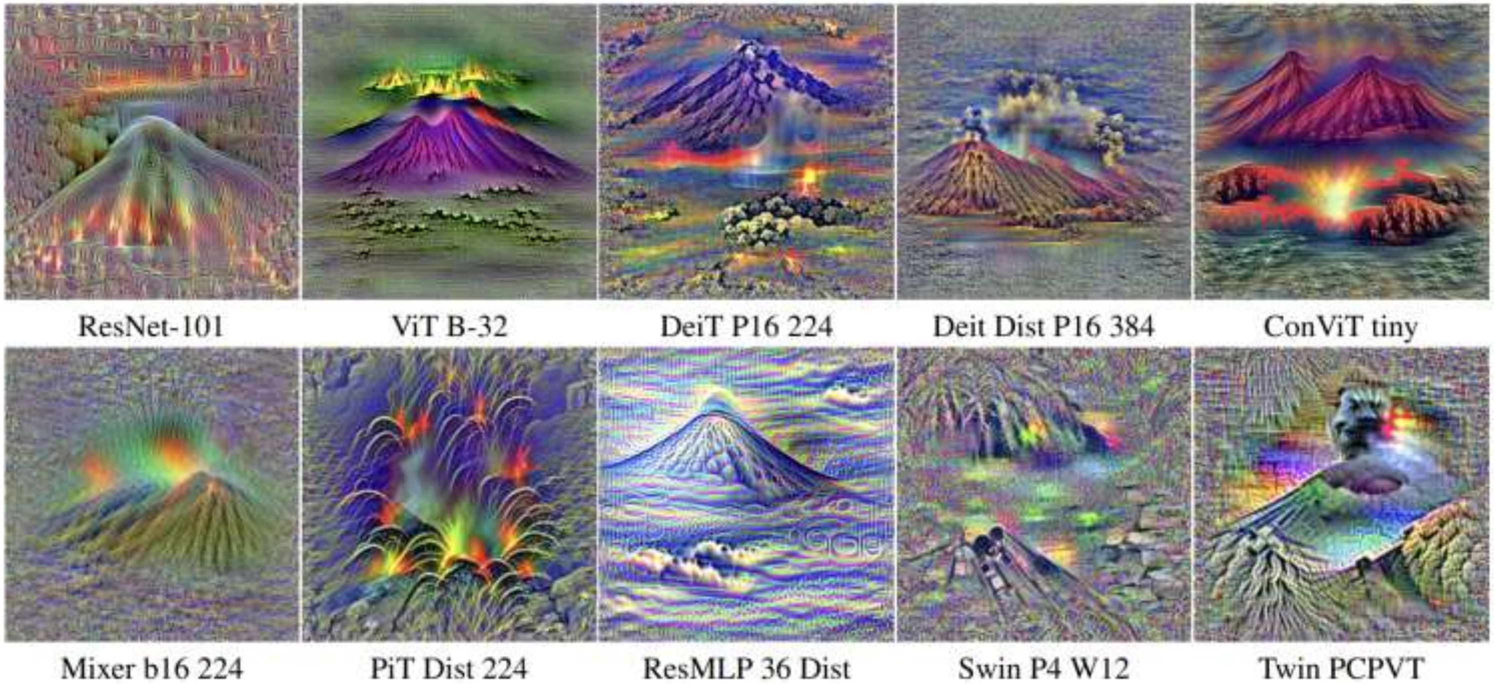
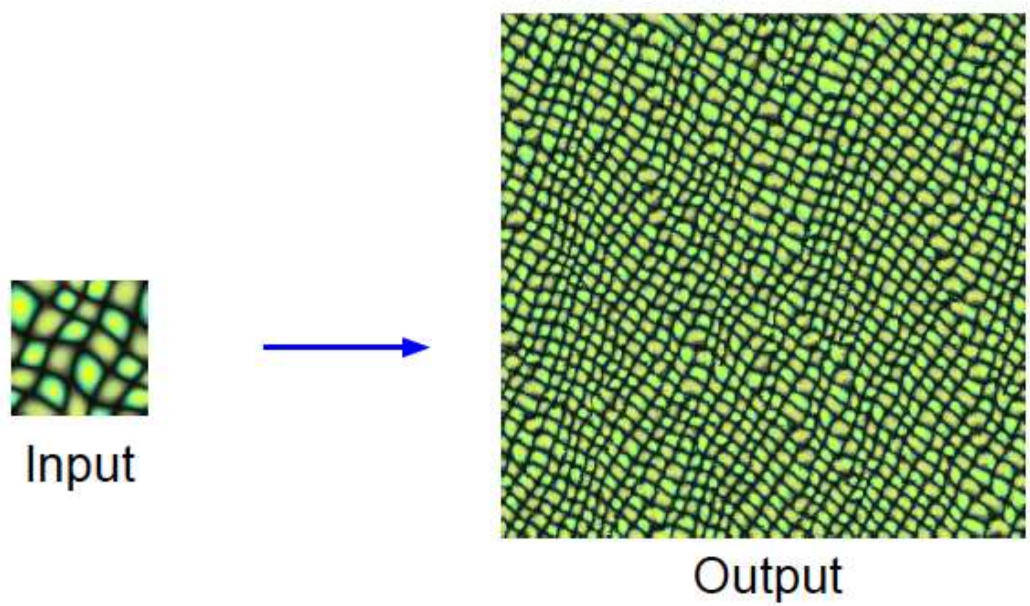


Figure 6. Images inverted from the ImageNet Volcano class for various Convolutional, Transformer, and MLP-based networks using *PII*. See figure 17 for further examples. For more details about networks, refer to Appendix B.

Texture Synthesis

- Problem setup

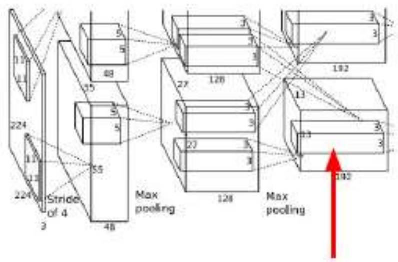
Given a sample patch of some texture, can we generate a bigger image of the same texture?

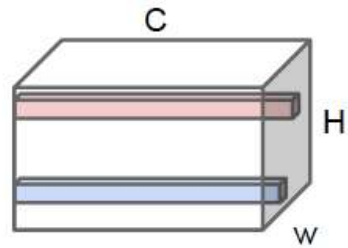


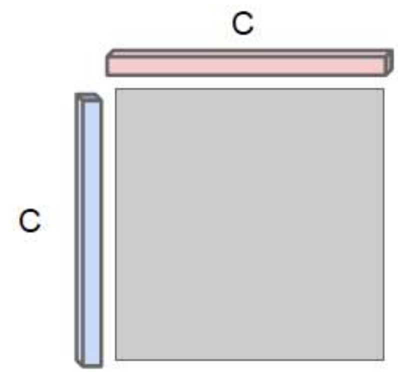
Texture Synthesis

- CNN-based modeling of image statistics

This image is in the public domain.







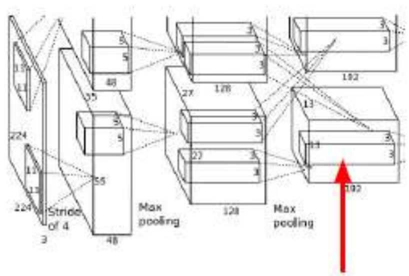
Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

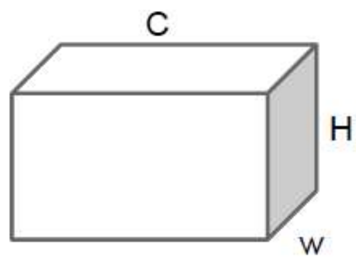
Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

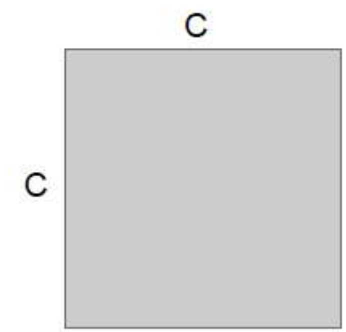
Texture Synthesis

- 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

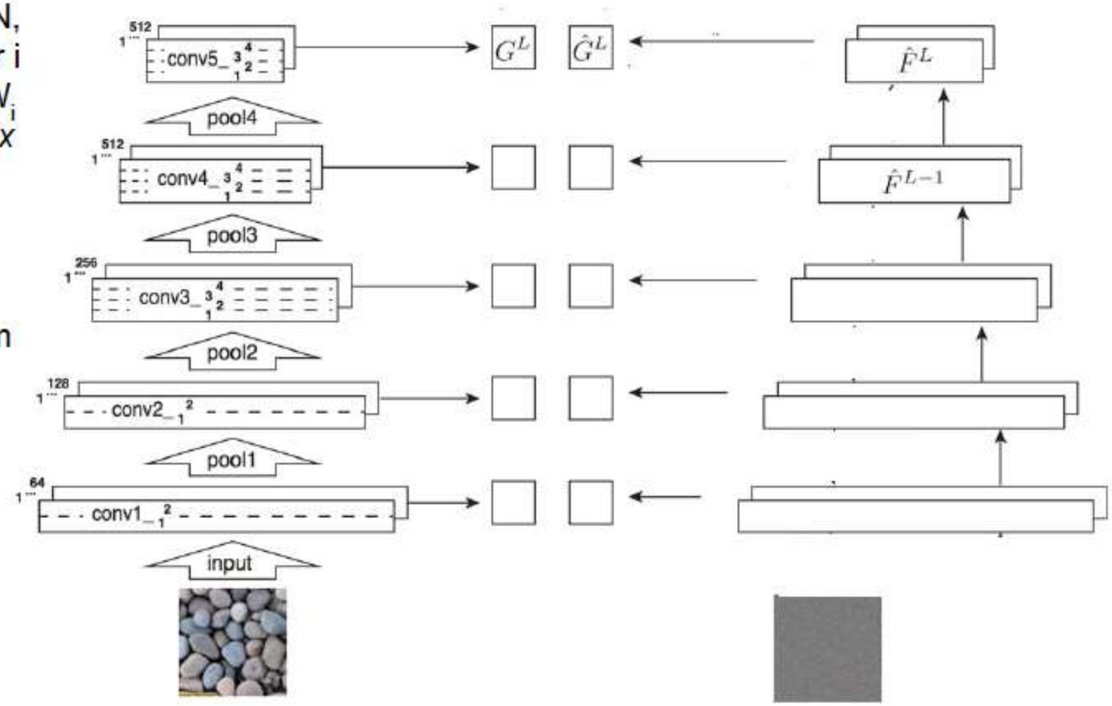
Texture Synthesis

- 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:

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

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

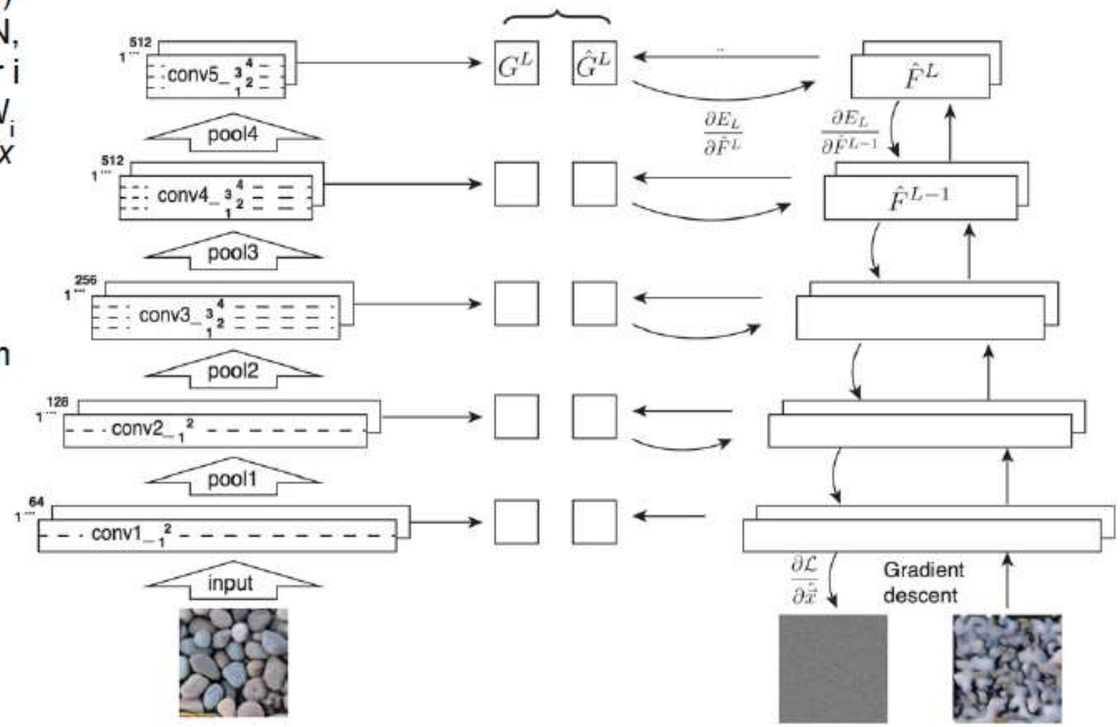
- Neural texture synthesis

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:

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

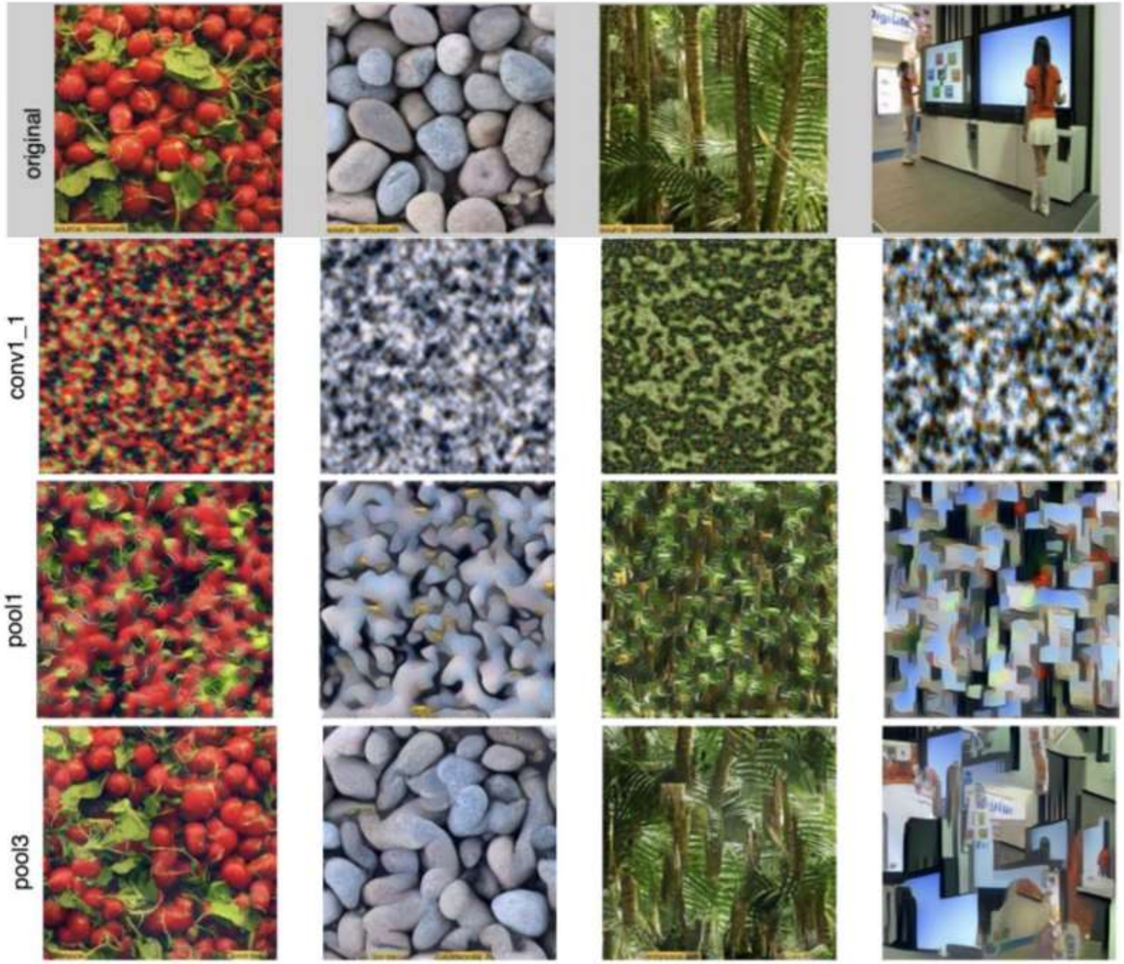
4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices
7. Backprop to get gradient on image
8. Make gradient step on image
9. GOTO 5

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



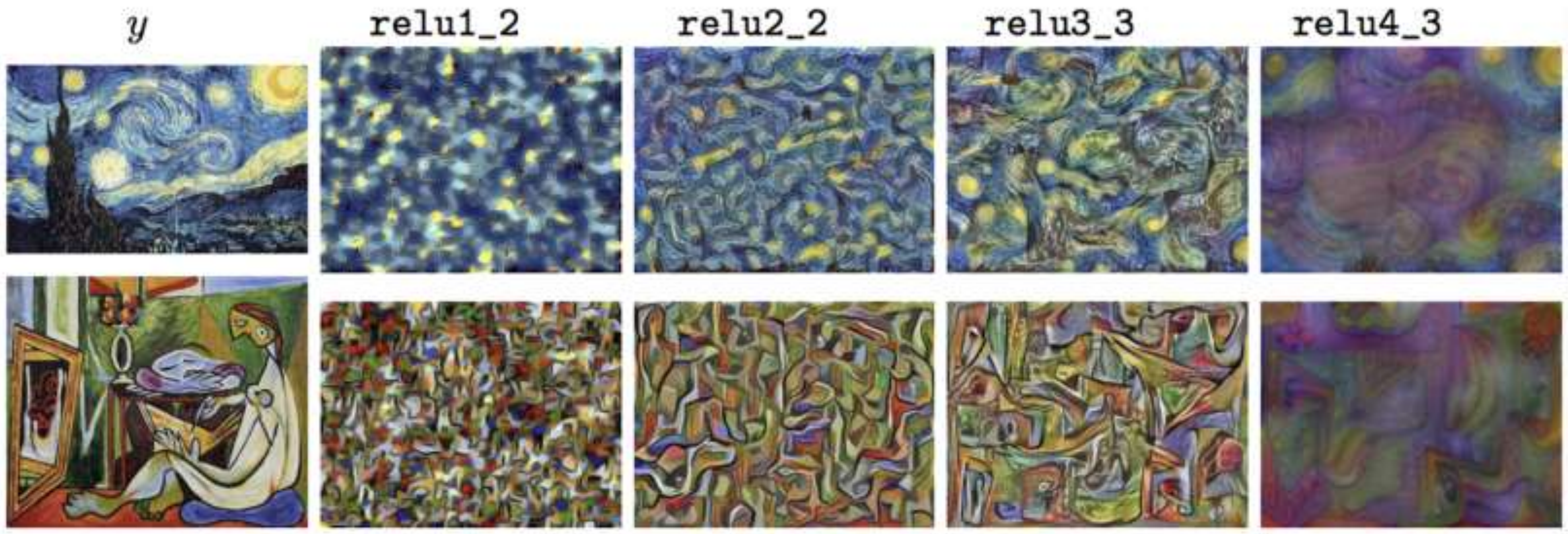
Texture Synthesis

- Neural texture synthesis

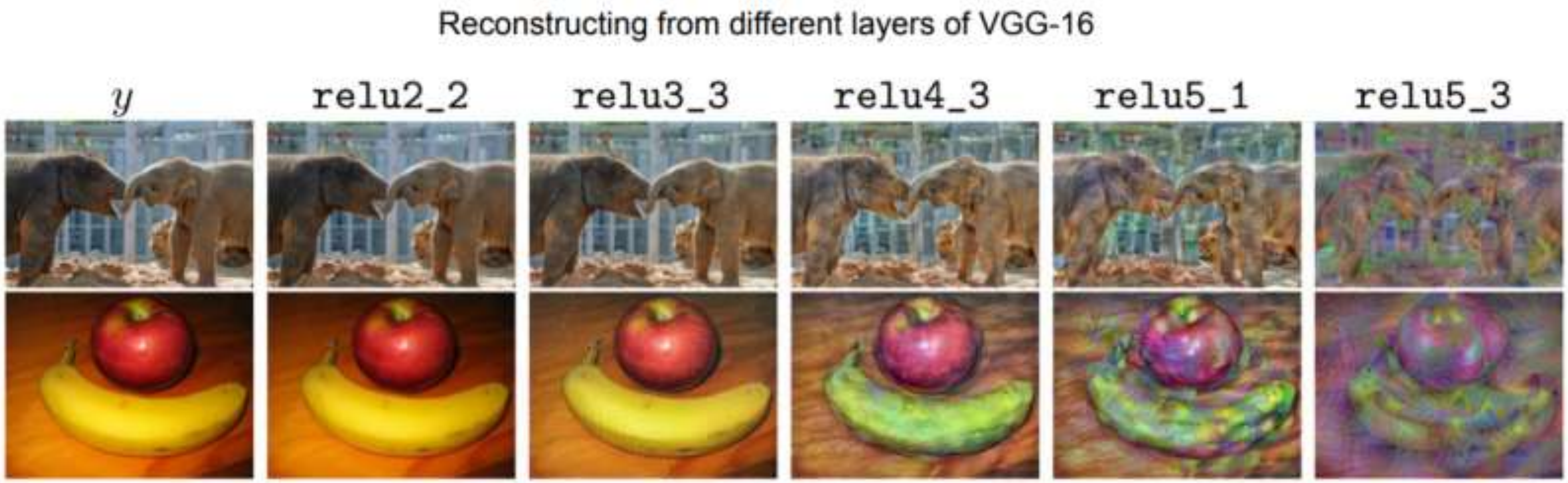


Texture Synthesis

- In terms of Gram Reconstruction
- Texture = artwork



Recall Feature inversion



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

This image is licensed under CC-BY 3.0

+

Style Image

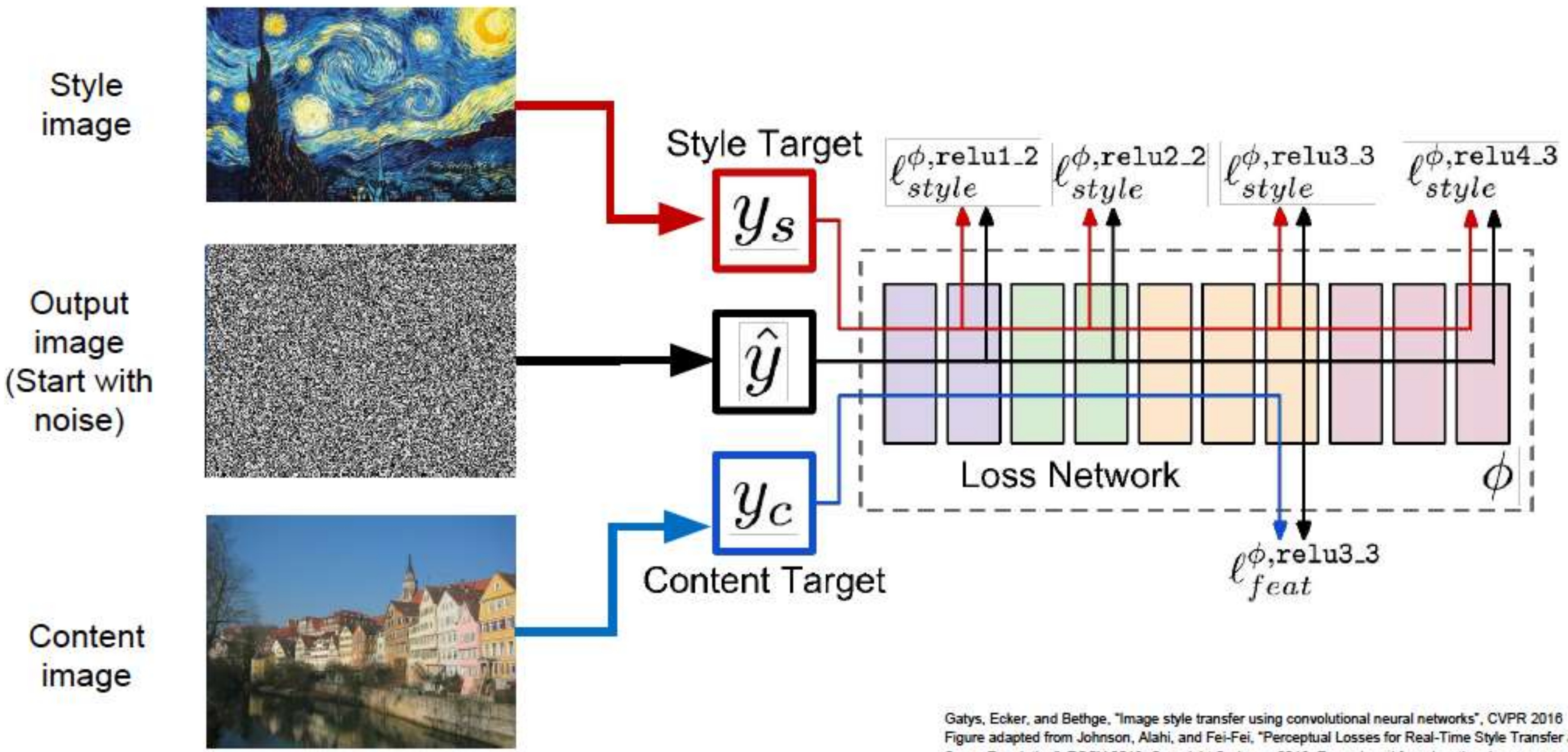
Starry Night by Van Gogh is in the public domain

=

Style Transfer!

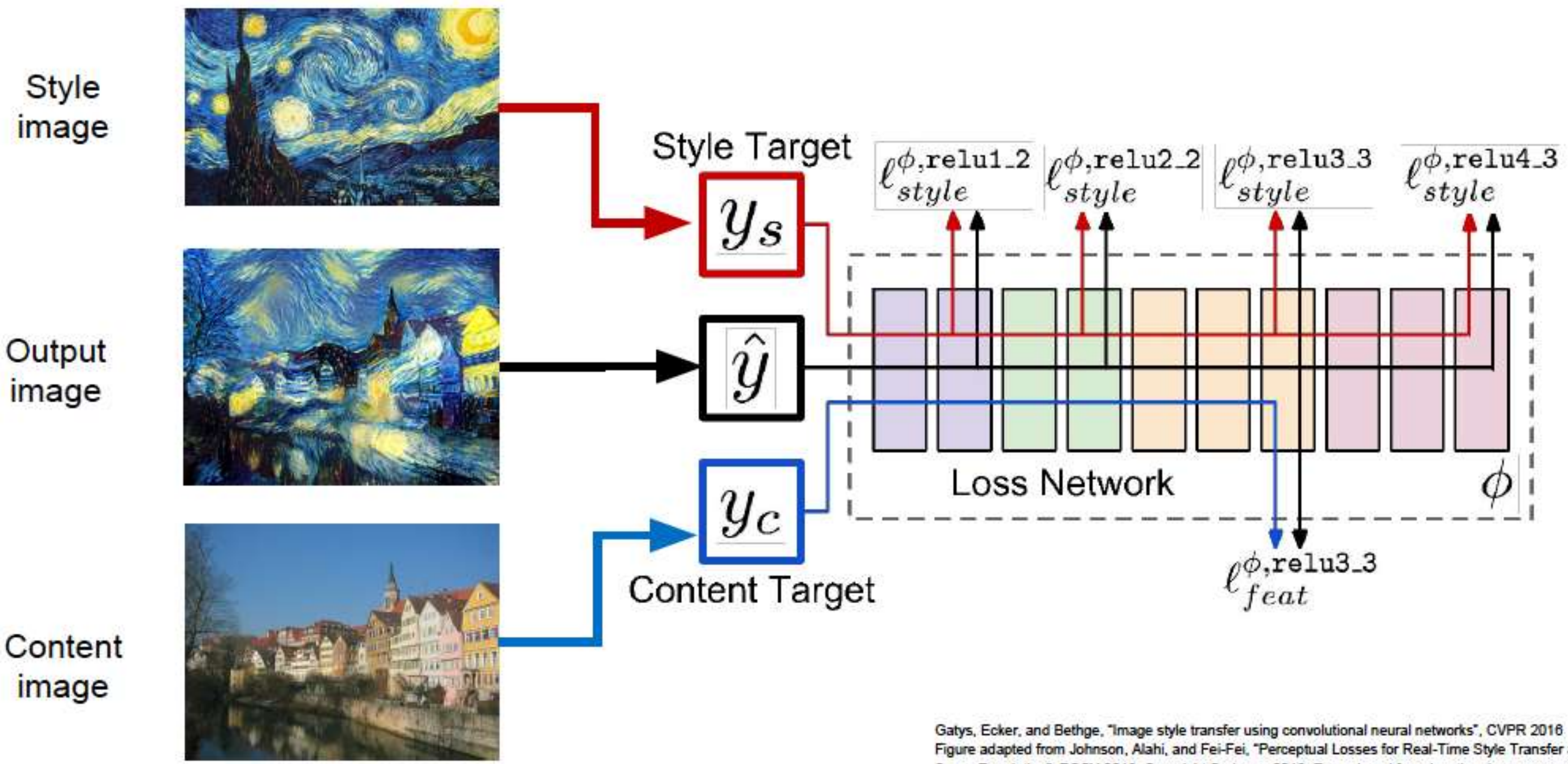
This image copyright Justin Johnson, 2015. Reproduced with permission.

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

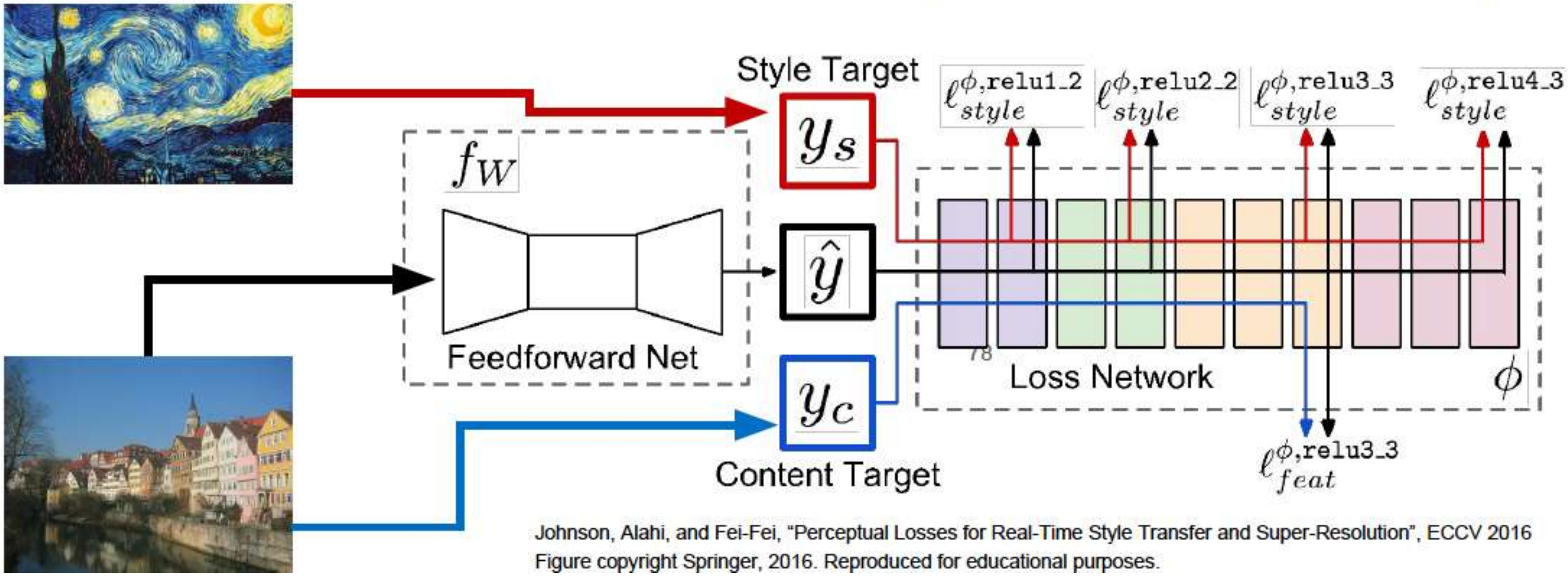
Neural Style Transfer

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.

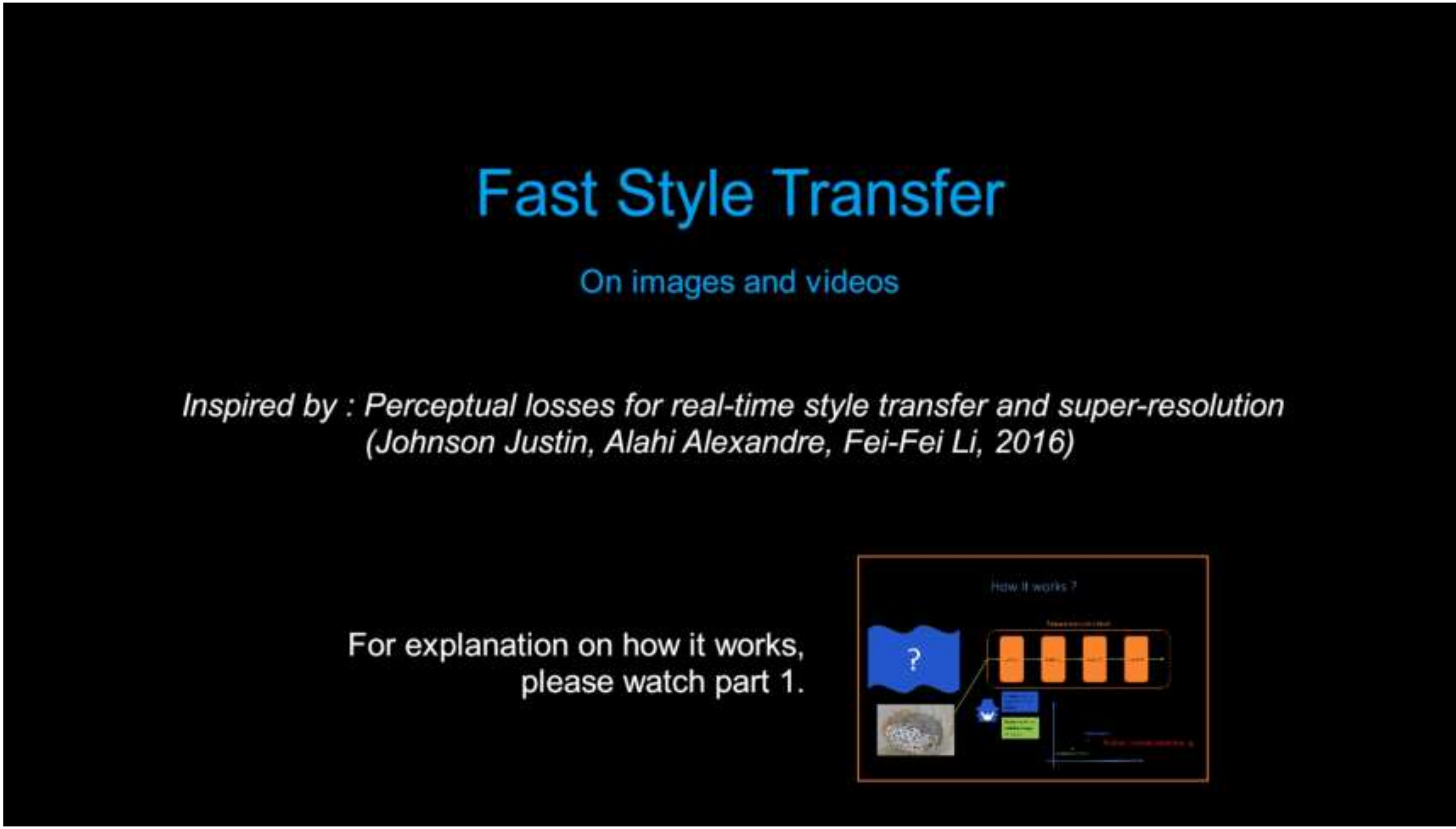
Fast Style Transfer

(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



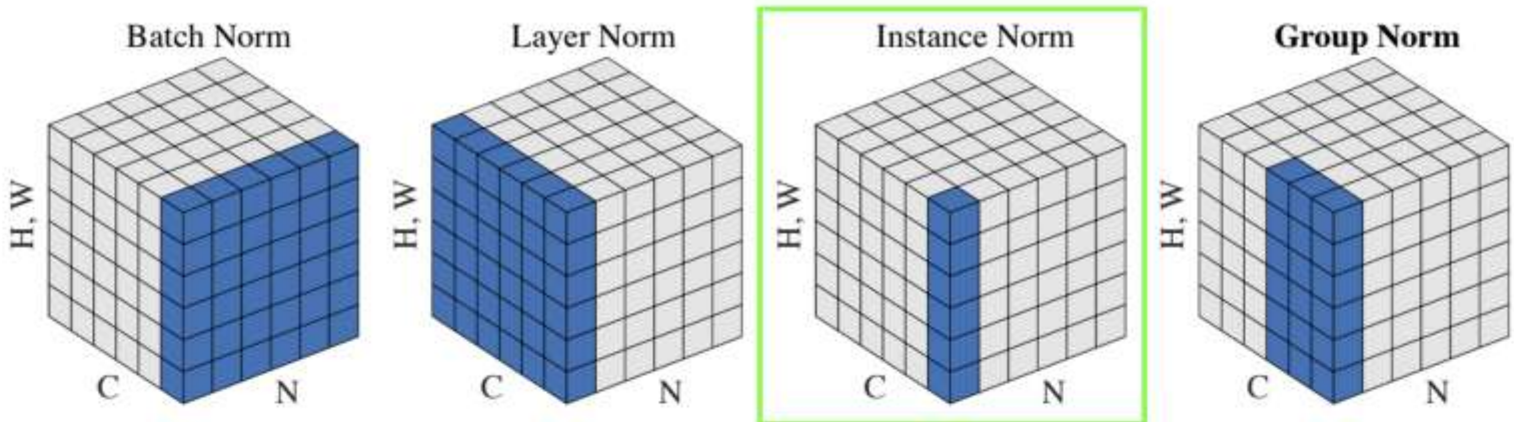
Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016

Fast Style Transfer



Recall Instance Normalization

- Instance Normalization was developed for style transfer!



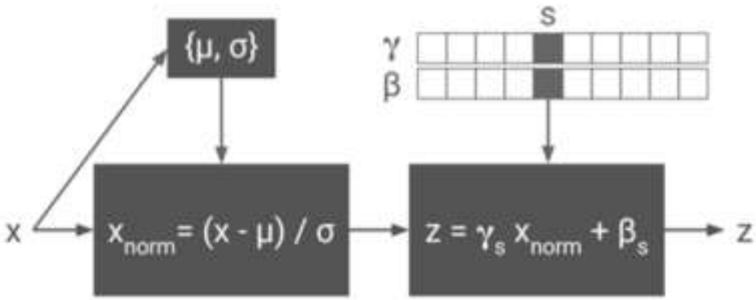
Ulyanov et al, "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images", ICML 2016
Ulyanov et al, "Instance Normalization: The Missing Ingredient for Fast Stylization", arXiv 2016

Fast Style Transfer

- Replacing BN with IN improves results!

One Network, Many Styles

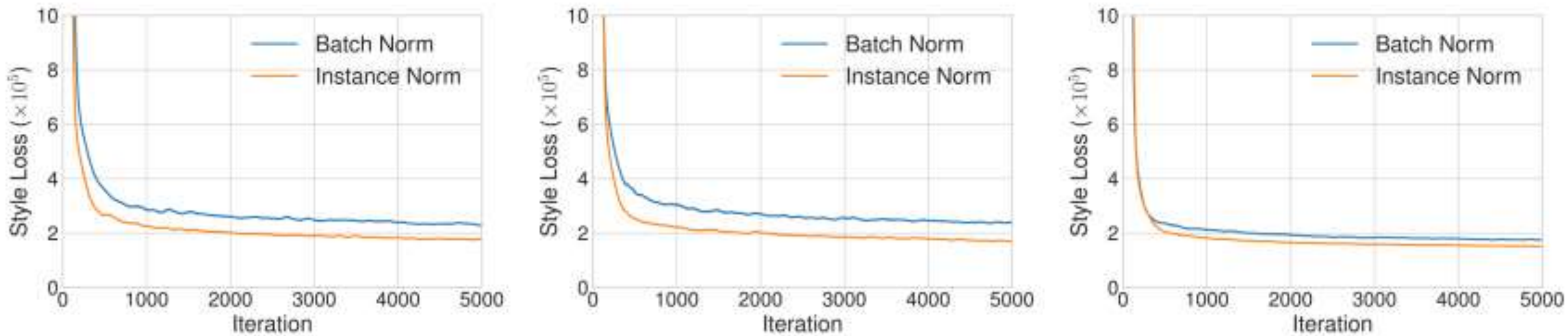
- Same network for multiple styles
- Conditional Instance Normalization: learn separate scale and shift parameters per style



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?



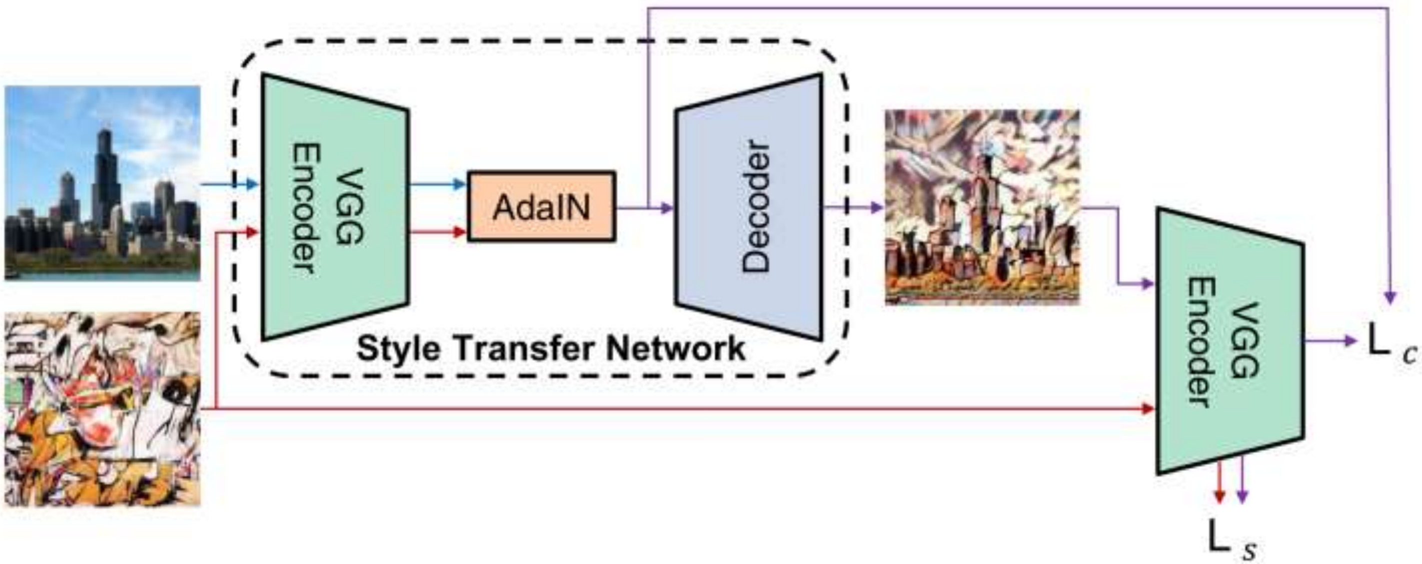
(a) Trained with original images. (b) Trained with contrast normalized images. (c) Trained with style normalized images.

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

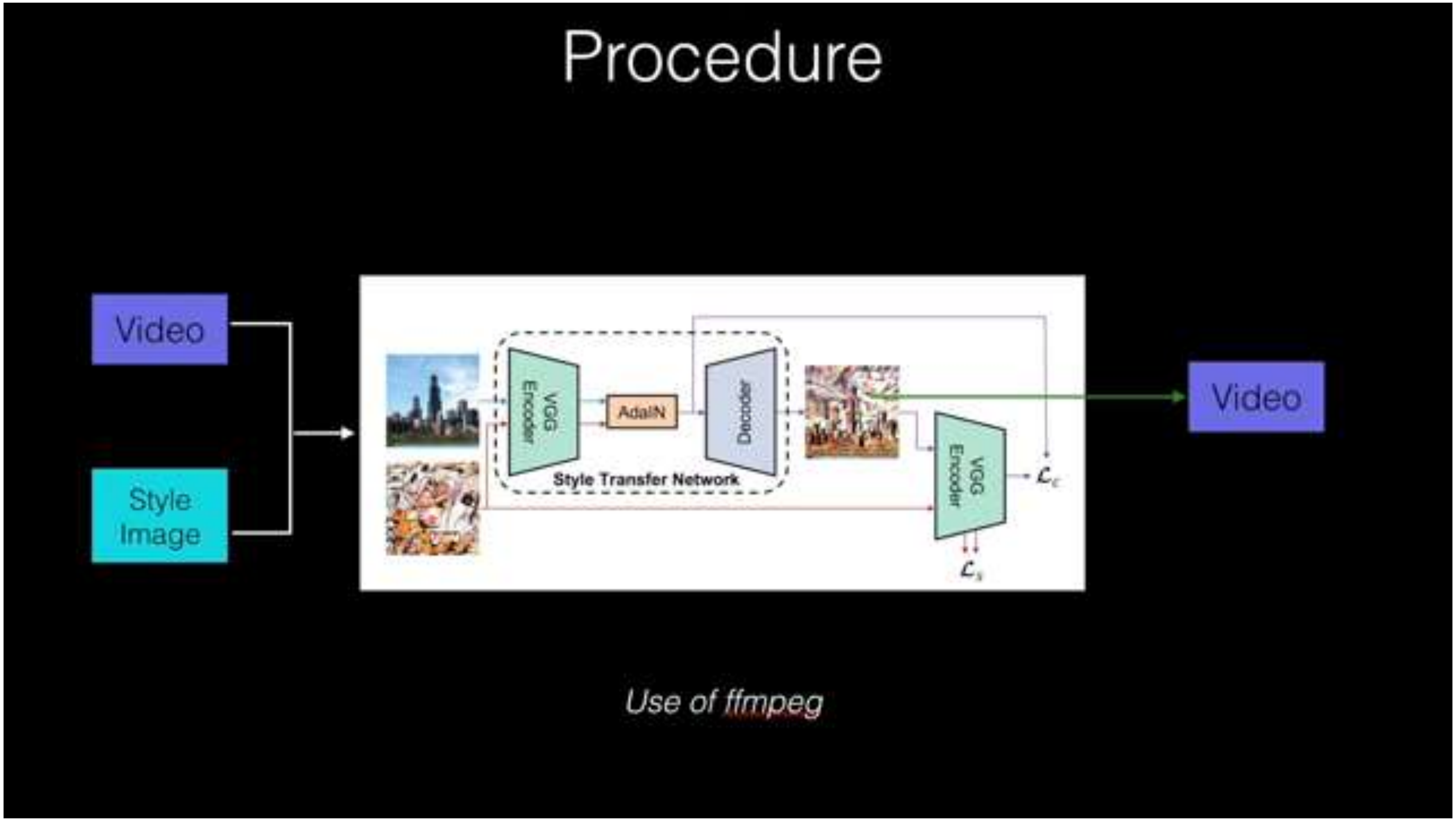
- x: content image; y: style image



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

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

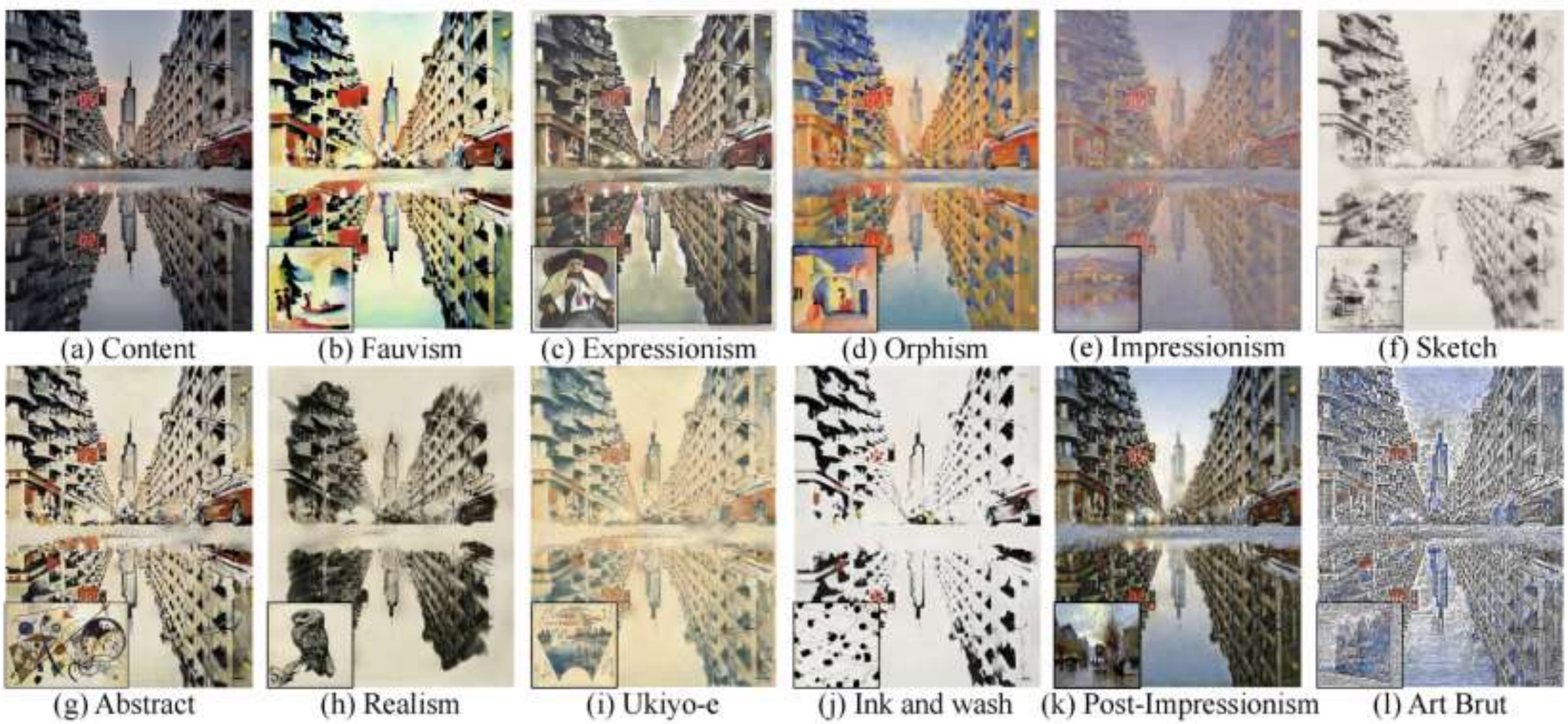
Adaptive Instance Normalization



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

Recent advances

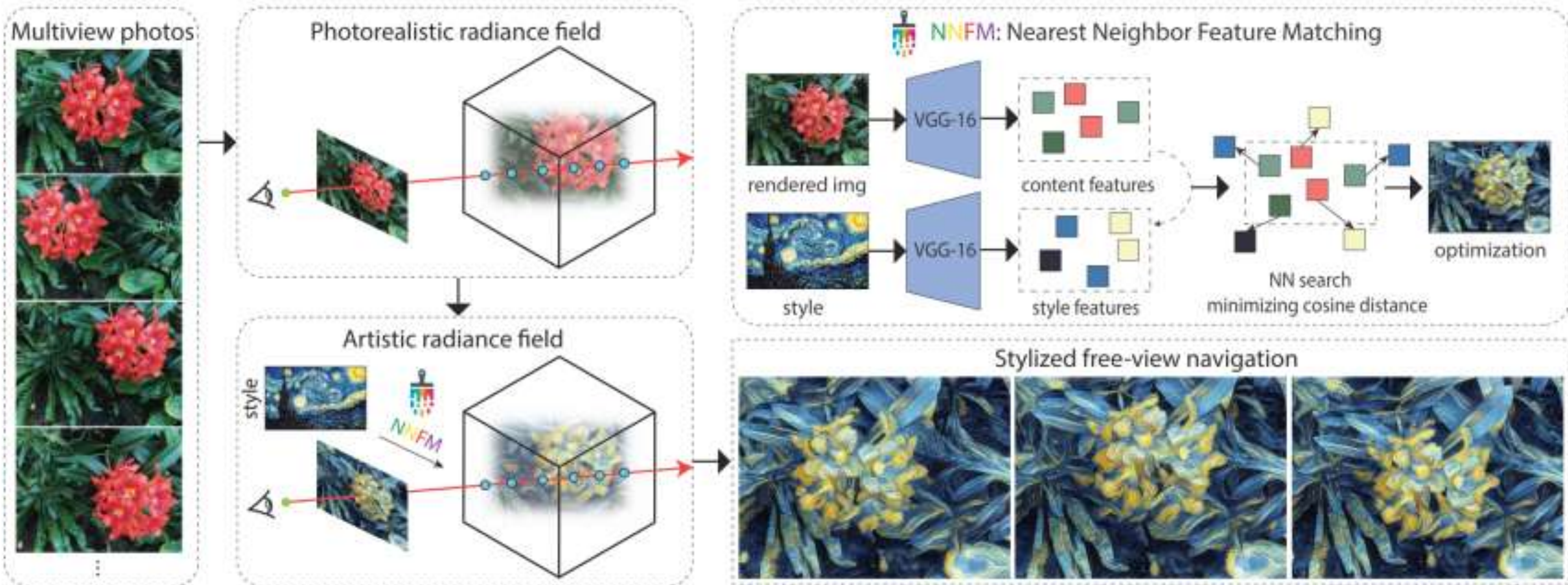
- More complex style representation than second-order statistics



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

Recent advances

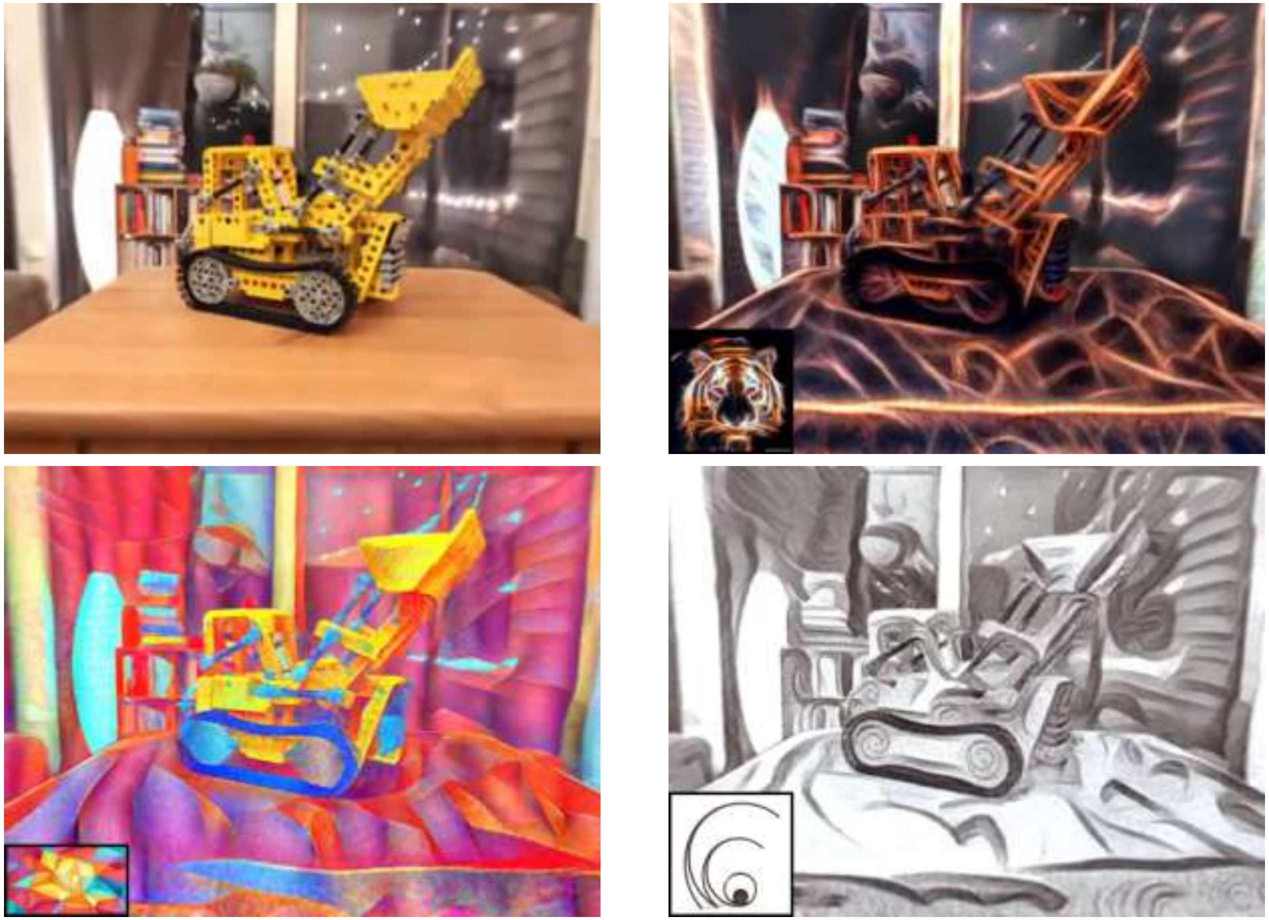
- From 2D to 3D using Neural Radiance Field (NeRF)



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