Advanced Deep Learning Architectures
COMP 5214 & ELEC 5680

Instructor: Dr. Qifeng Chen
https://cqf.io

Credit: https://d2l.ai/

Image Augmentation

p=1.0

size_percent=0.30

p=0.50

cutoff=0.00

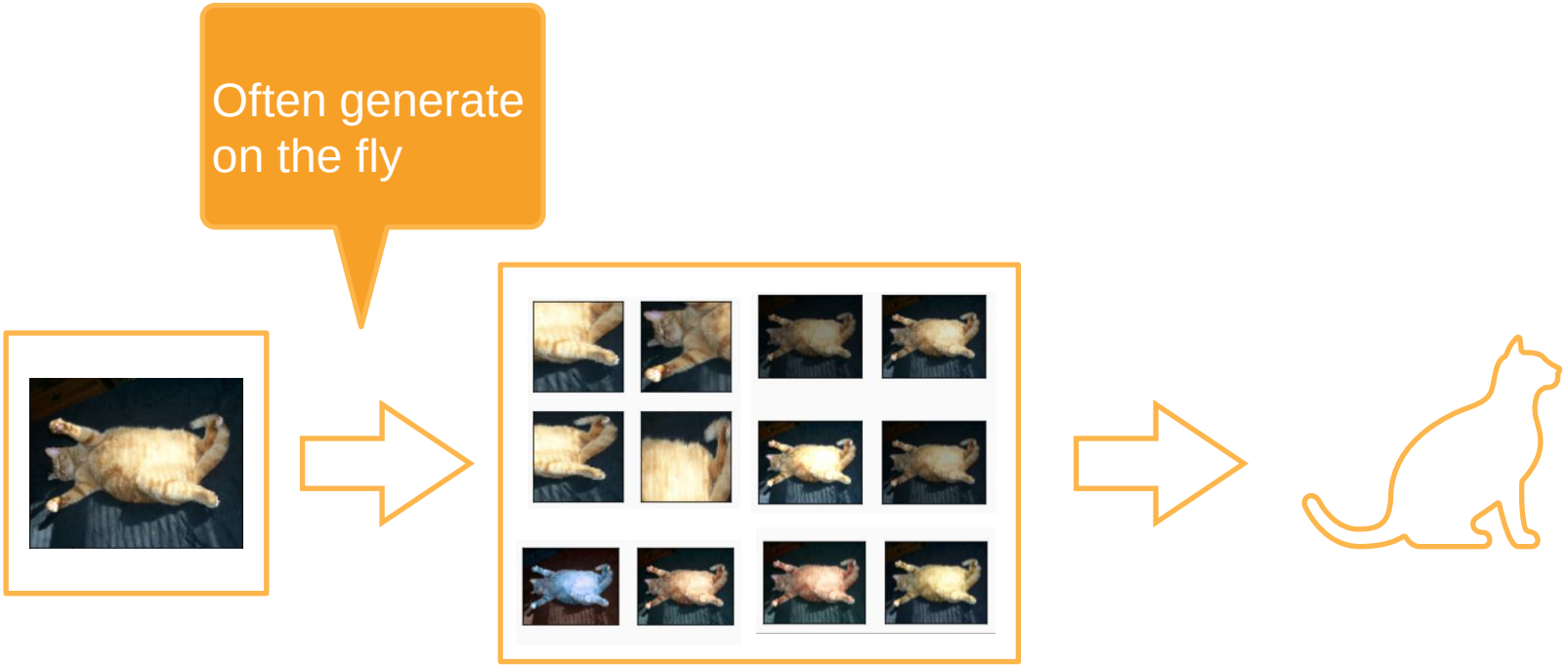
A Real Story at CES'19

- A startup found their demo, a smart vending machine that identifies what customers picked through a camera, didn't work because the showroom has
  - a different light temperature
  - light reflection from the desk
- They worked all night to re-collect data and train a new model, and ordered a tablecloth

Data Augmentation

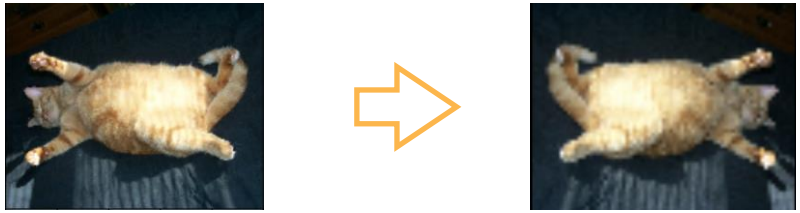
- Augment an existing dataset with more diversities
- Add various background noises into a speech
- Transform an image to several others by altering colors or/and changing shapes

Training with Augmented Data

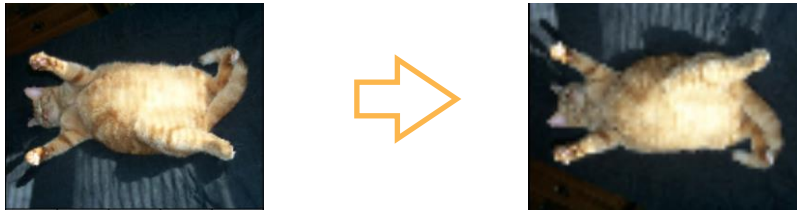


Flip

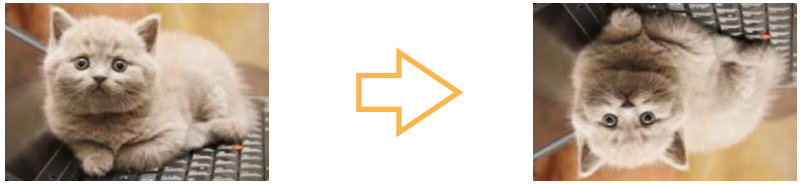
- Left-right flip



- Top-bottom flip

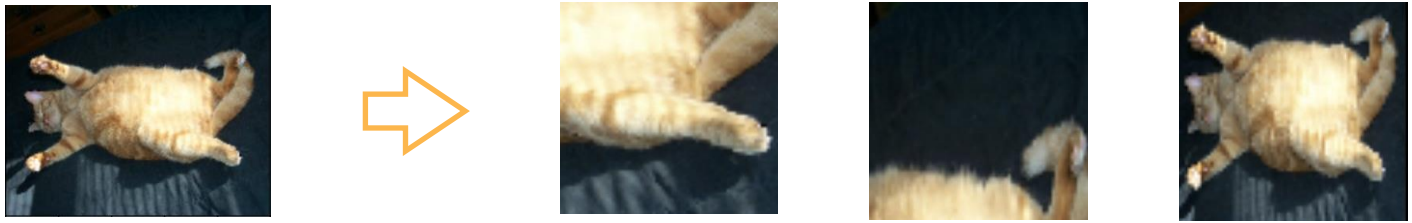


  - Not always makes sense



Crop

- Crop an area from the image and then resize it
  - A random width-height ratio (e.g. [3/4, 4/3])
  - A random area size (e.g. [8%, 100%])
  - A random position

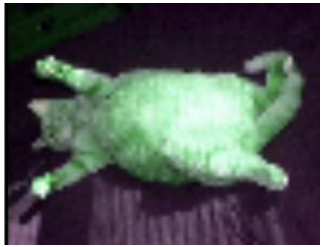


Color

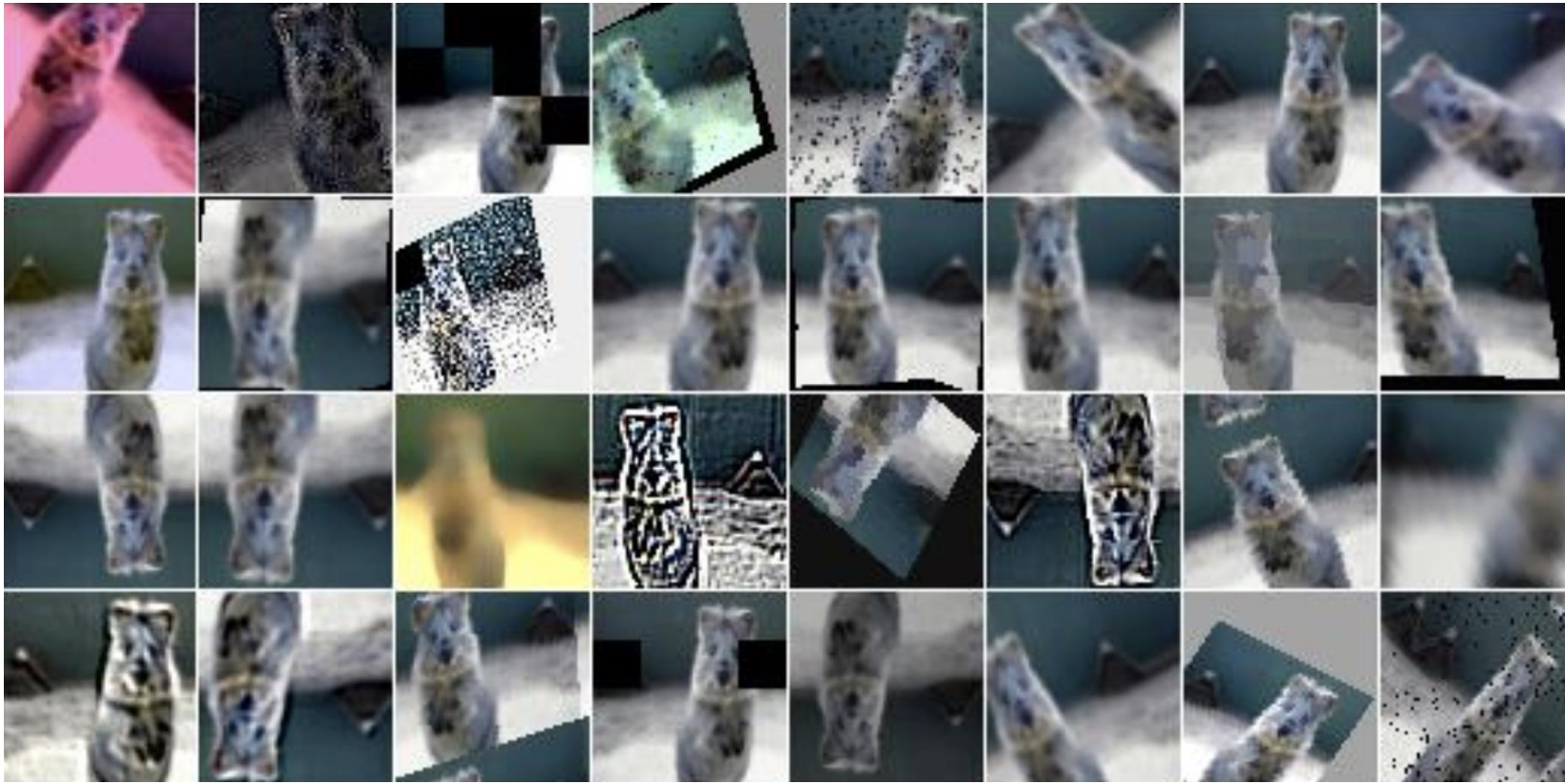
- Scale hue, saturation, and brightness (e.g. [0.5, 1.5])

Brightness

Hue



Tens of Other Ways to Augment



https://github.com/aleju/imgaug

Fine Tuning

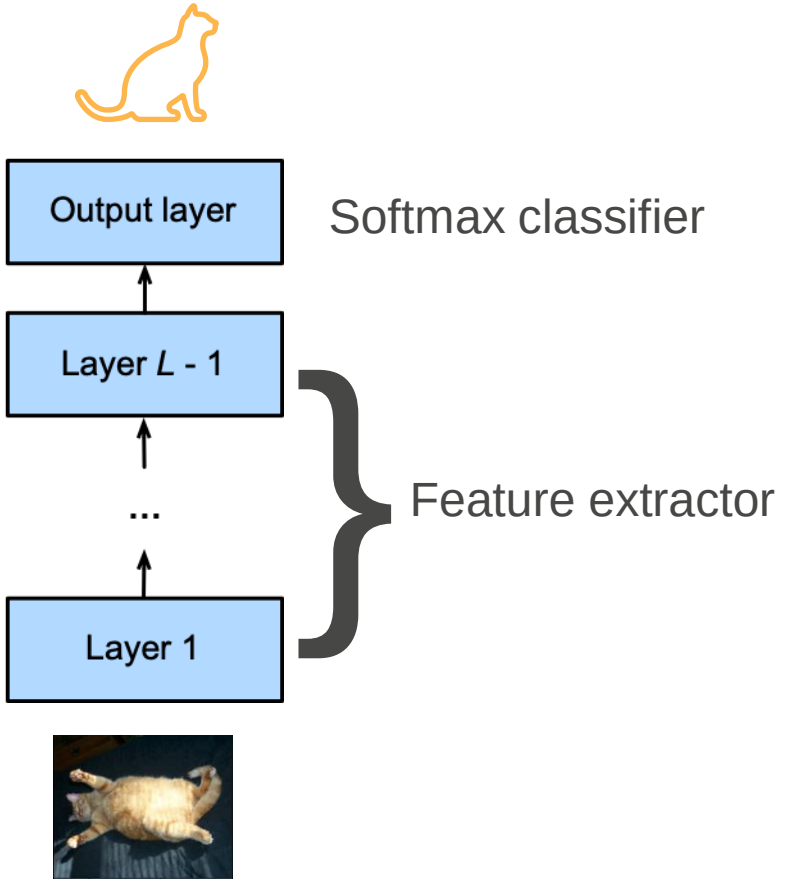
Labelling a Dataset is Expensive

My dataset

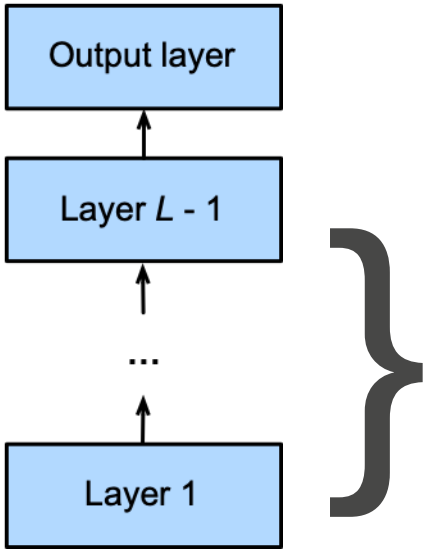
# examples	1.2 M	50K	60 K
# classes	1,000	100	10

Network Structure

- A neural network can be roughly partitioned into two parts
  - A feature extractor maps raw pixels into linearly separable features
  - A linear classify to make decisions



Fine Tuning



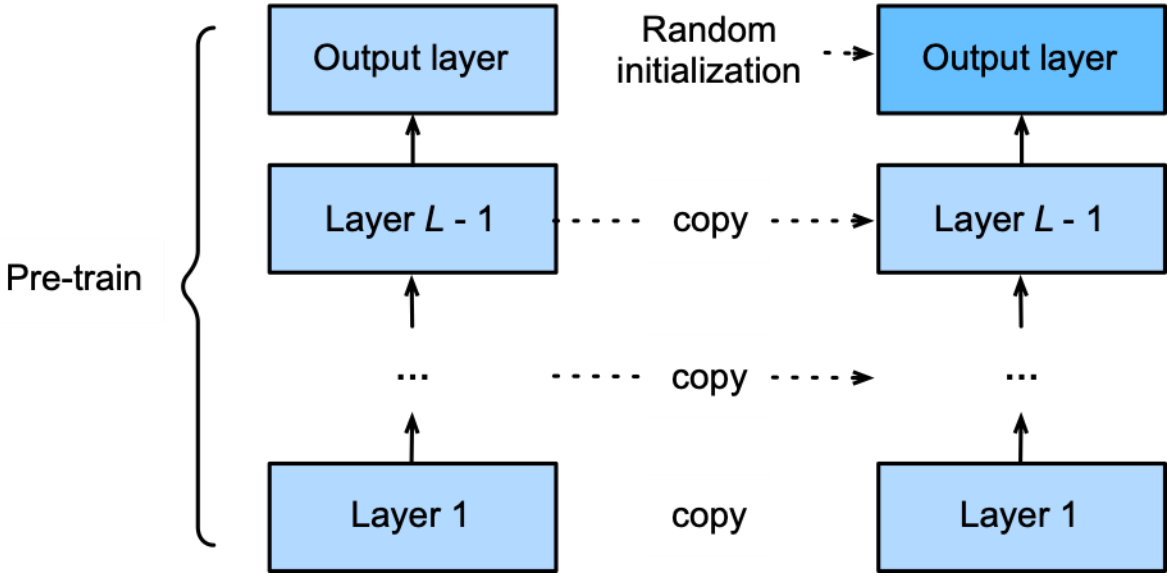
Maybe not use the classifier parameters directly because labels are changed

Maybe also a good extractor for my dataset

Source Dataset

Target Dataset

Weight Initialization for Fine Turning



Source Dataset

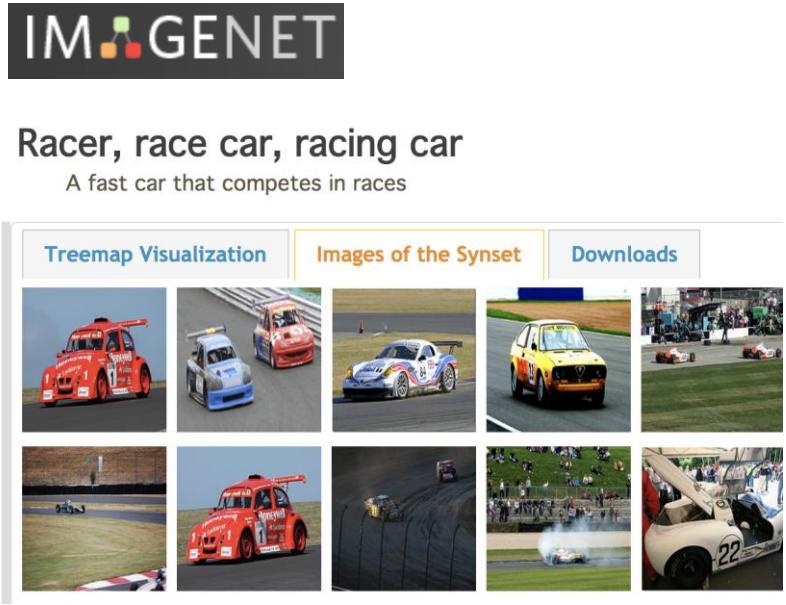
Target Dataset

Training

- Train on the target dataset as a normal training job, but with a strong regularization
  - Uses a small learning rate
  - Uses less epochs
- If source dataset is more complex than the target dataset, fine-tuning often leads to better quality models

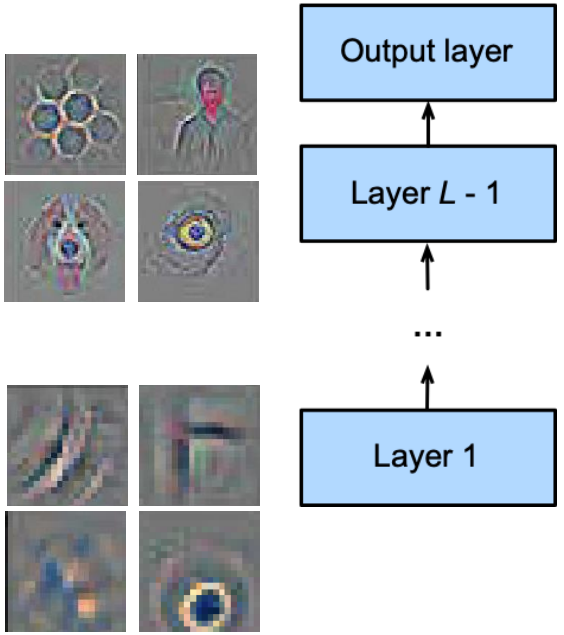
Re-use Classifier Parameters

- The source dataset may contain some categories from the target datasets
- Use the according weight vectors from the pre-trained model during initialization



Fix Some Layers

- Neural networks learn hierarchical feature representations
  - Low-level features are universal
  - High-level features are more related to objects in the dataset
- Fix the bottom layer parameters during fine tuning
  - Another strong regularizer



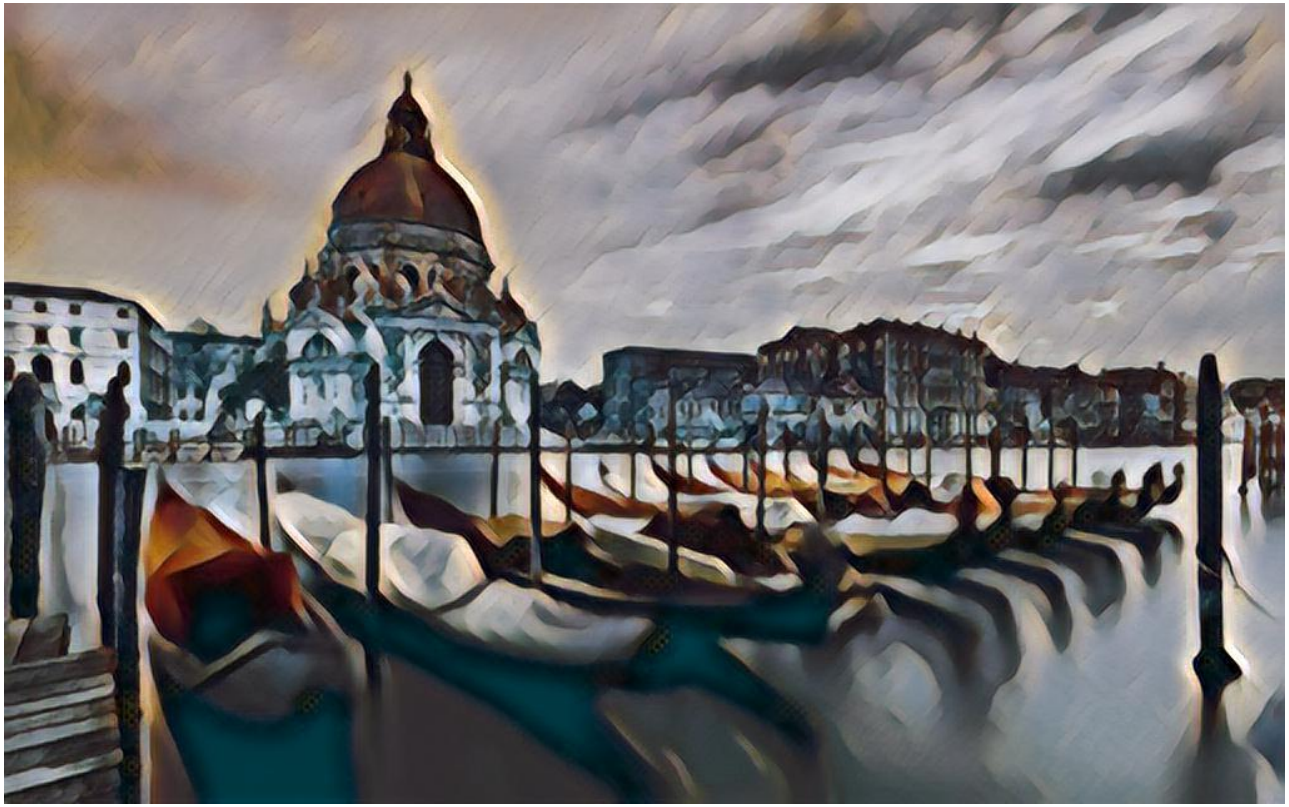
Style Transfer

+

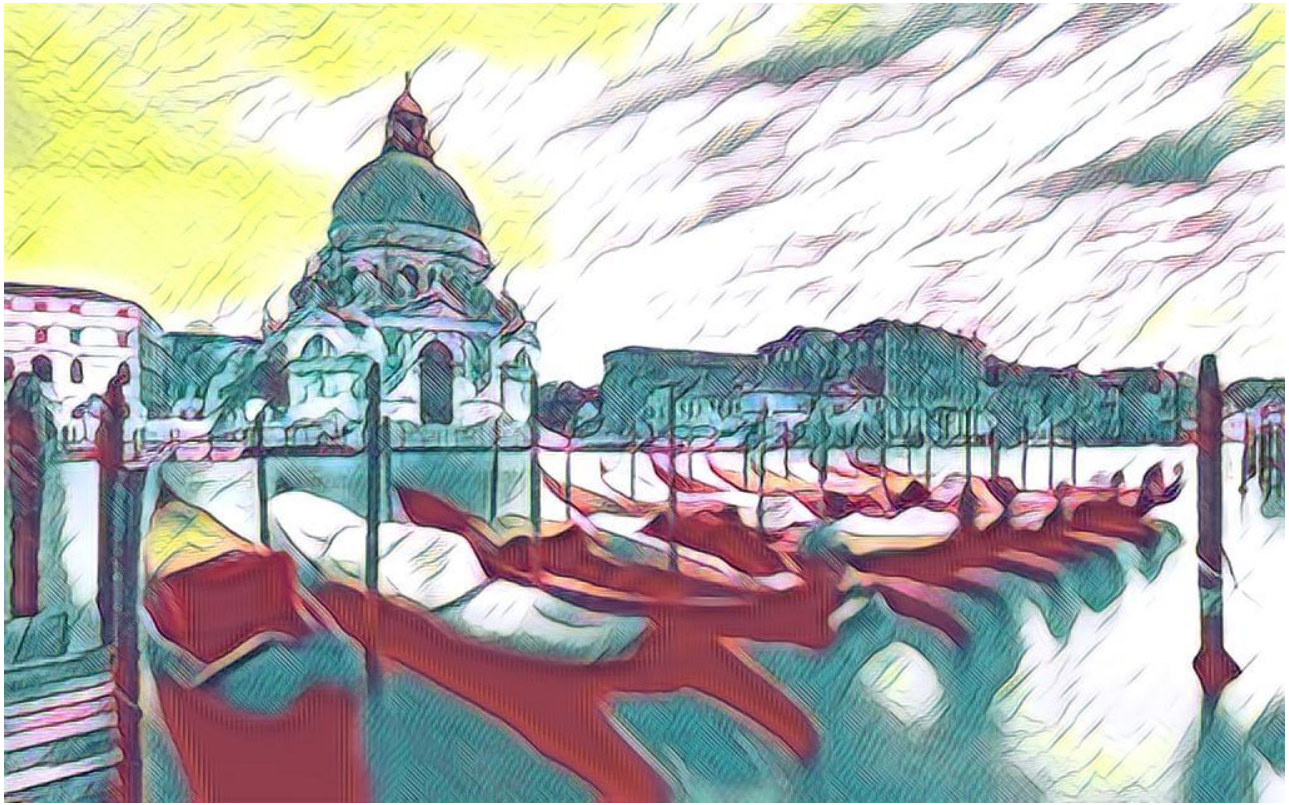
https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

+

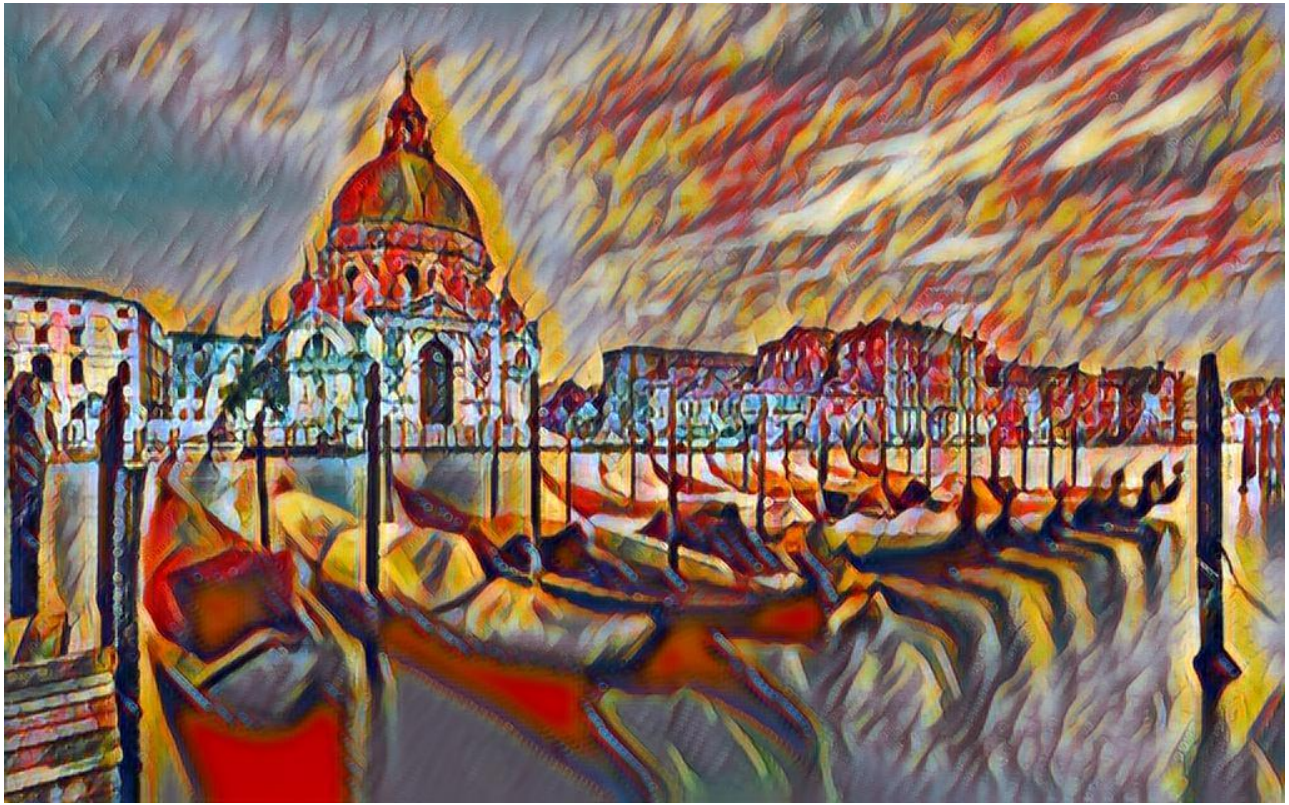

UDNIE



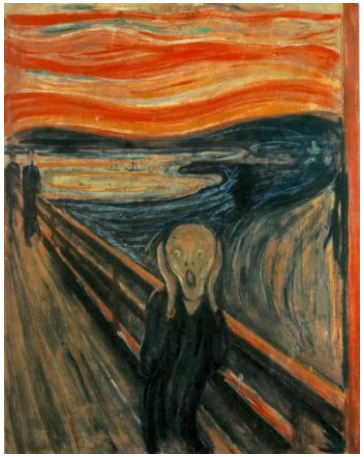
+

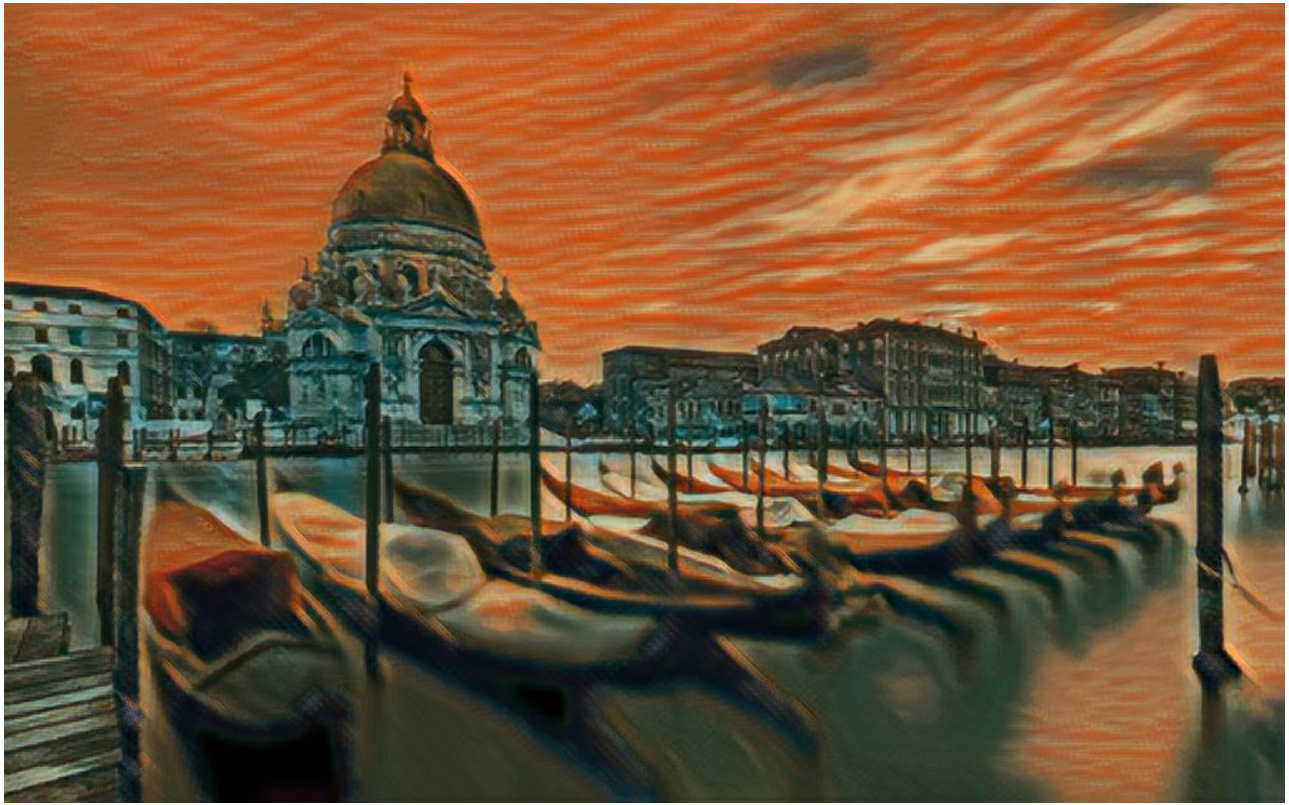


+



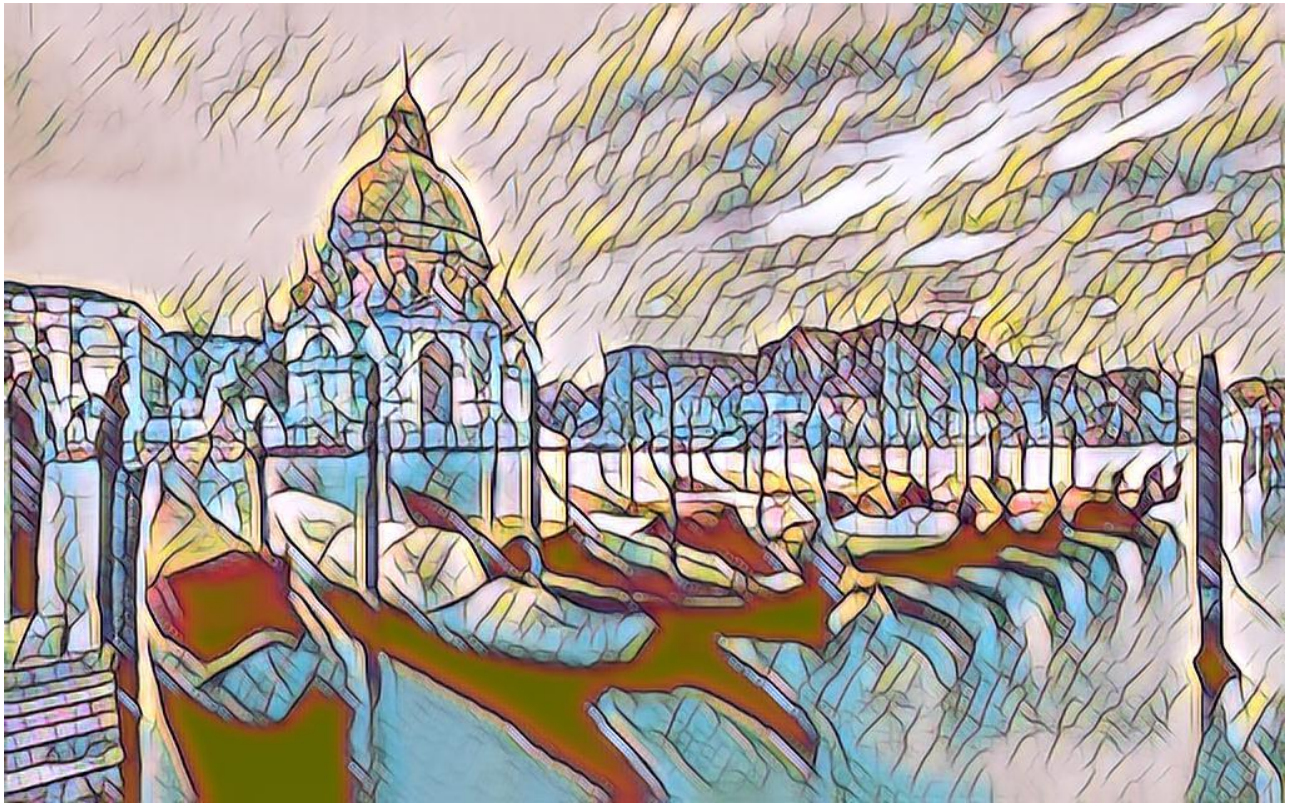
+



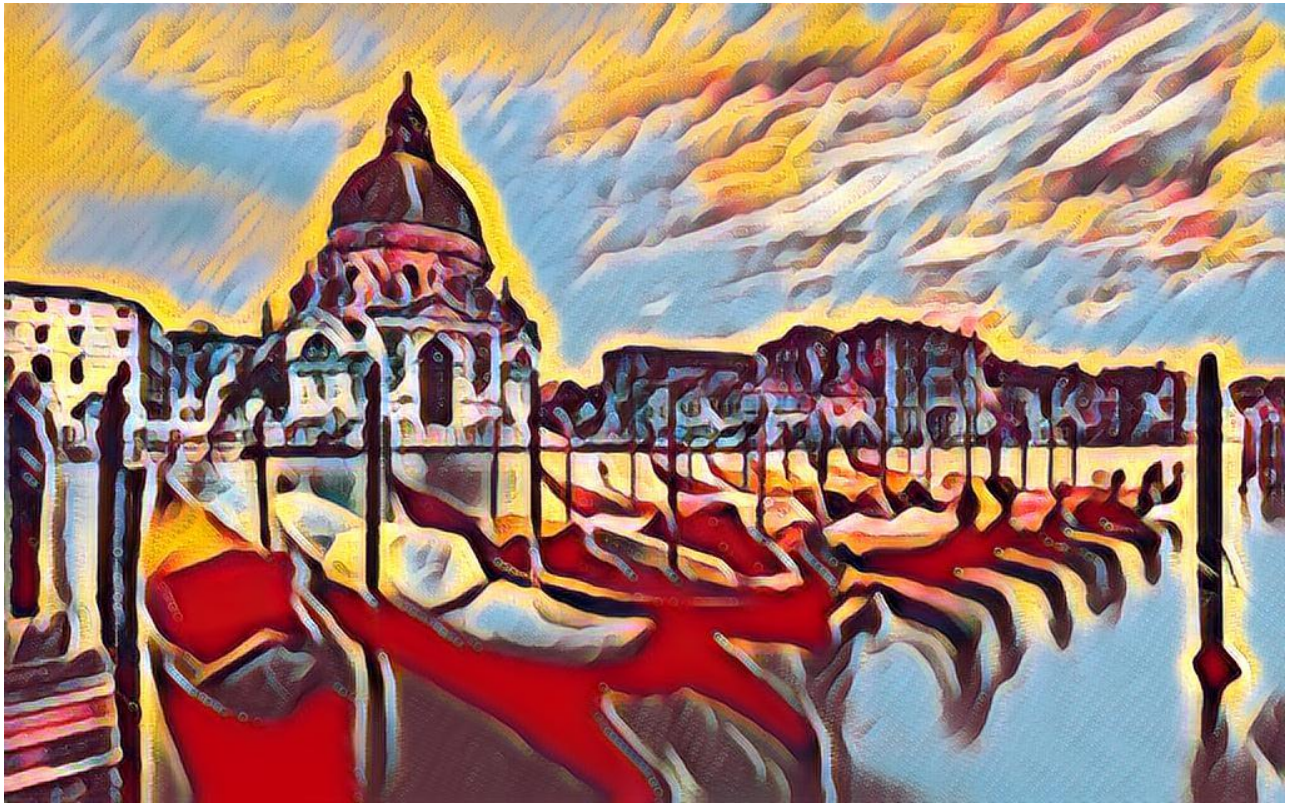


+

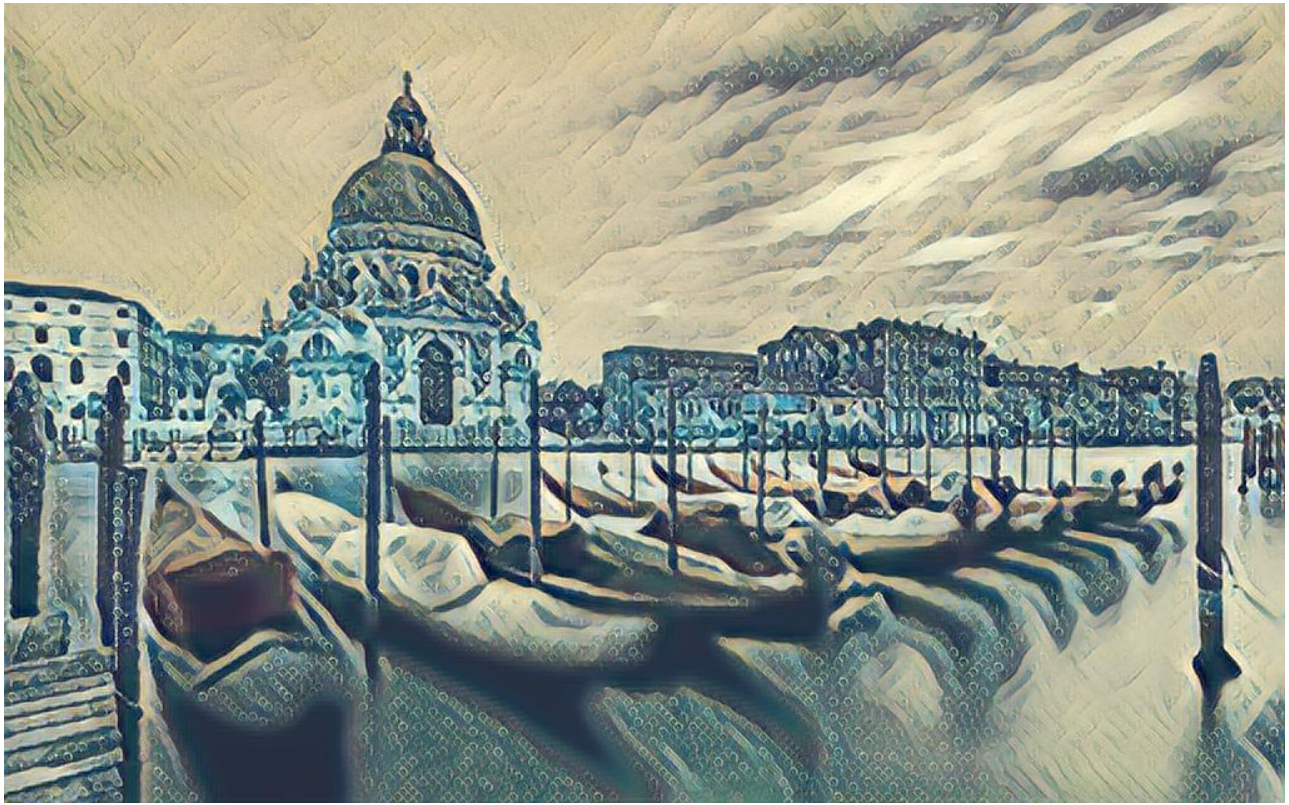
+



+



+



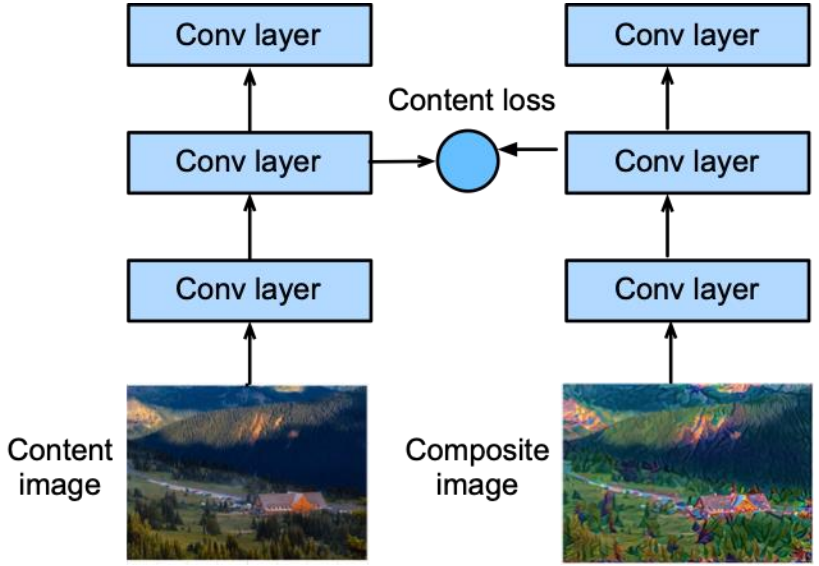
Neural Style (CVPR 2016)

- Learn a composite image to match the contents from a content image and the styles from the style image

$$\arg\min_{I} w_1 \ell_{\text{content}}(I_{\text{content}}, I) + w_2 \ell_{\text{style}}(I_{\text{style}}, I) + w_3 \ell_{\text{noise}}(I)$$

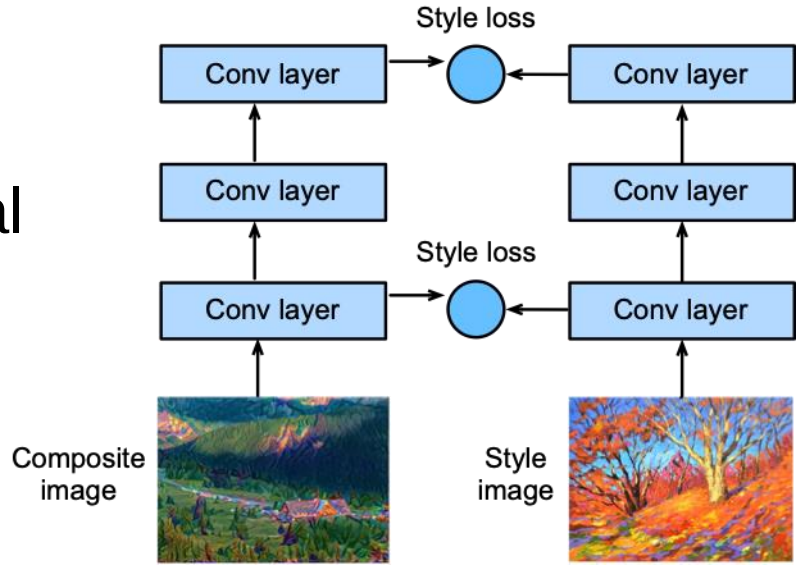
Content Loss (feature matching loss, perceptual loss)

- Feed both content and composite images to the a CNN
- Compare internal layer outputs with a squared loss
  - Bottom layers matches details
  - **Top layers matches global contents**



Style Loss

- Gram matrix G : $G_{i,j}$ is the inner product between channel *i* and *j*
- Compare gram matrices of internal layer outputs by a squared loss
  - Bottom layers matches local styles
  - Top layers matches global styles



Noise Loss

- The learned composite image may have a lot of high-frequency noise
- Use total variation to de-noise

$$\sum_{i,j} |x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|$$

content loss

$$\ell_{feat}^{\phi,j}(\hat{y},y) = \frac{1}{C_j H_j W_j}\|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

style loss

$$\ell_{style}^{\phi,j}(\hat{y},y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|_F^2.$$

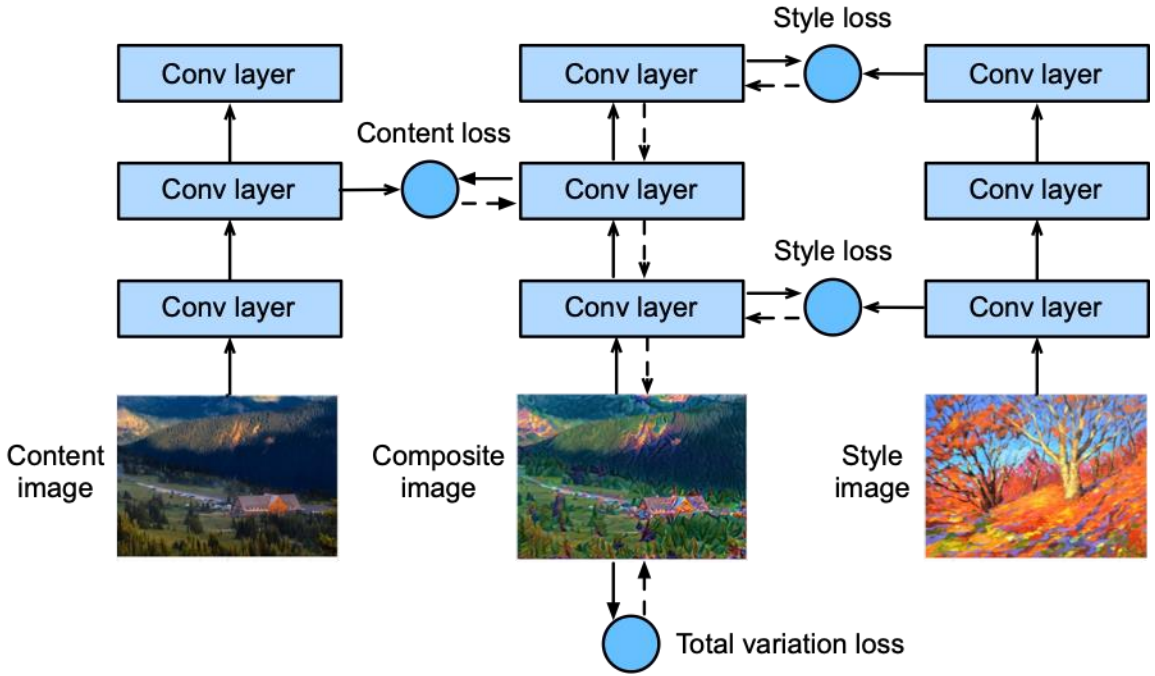
$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j}\sum_{h=1}^{H_j}\sum_{w=1}^{W_j}\phi_j(x)_{h,w,c}\phi_j(x)_{h,w,c'}.$$

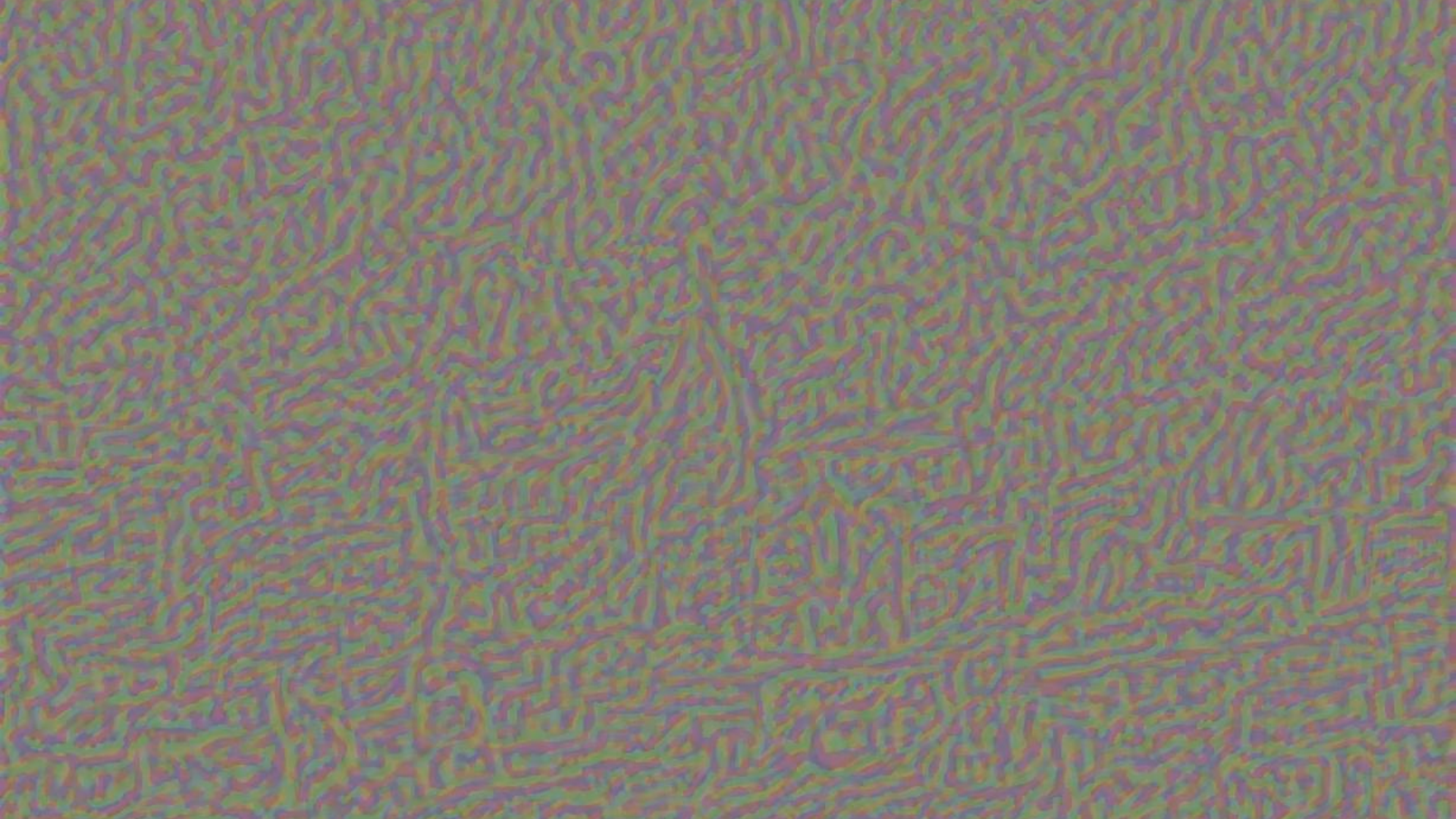
total variation loss

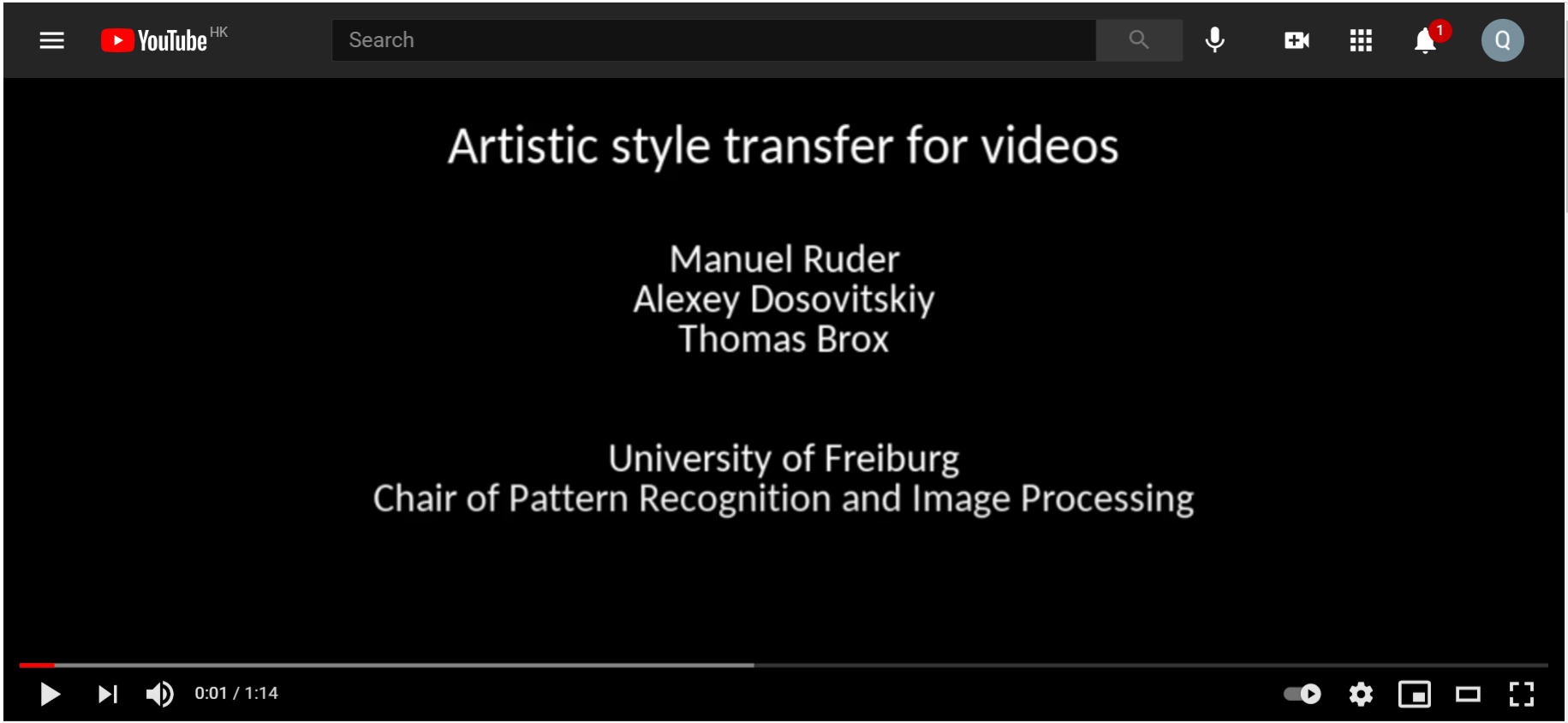
$$E_{\mathcal{U}}(\mathbf{w}) = \sum_{x_k \in \mathcal{I}} \Theta\left(\overline{f}(x_k;\mathbf{w})\right) = \sum_{x_k \in \mathcal{I}} |\nabla f(x_k;\mathbf{w})|$$
$$= \sum_{x_k \in \mathcal{I}} |\nabla_X f(x_k)| + |\nabla_Y f(x_k)|.$$

Put All Things Together (Optimization, takes minutes)

$$\arg\min_{I} w_1 \ell_{\text{content}}(I_{\text{content}}, I) + w_2 \ell_{\text{style}}(I_{\text{style}}, I) + w_3 \ell_{\text{noise}}(I)$$



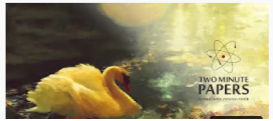




Artistic style transfer for videos

214,107 views • May 2, 2016

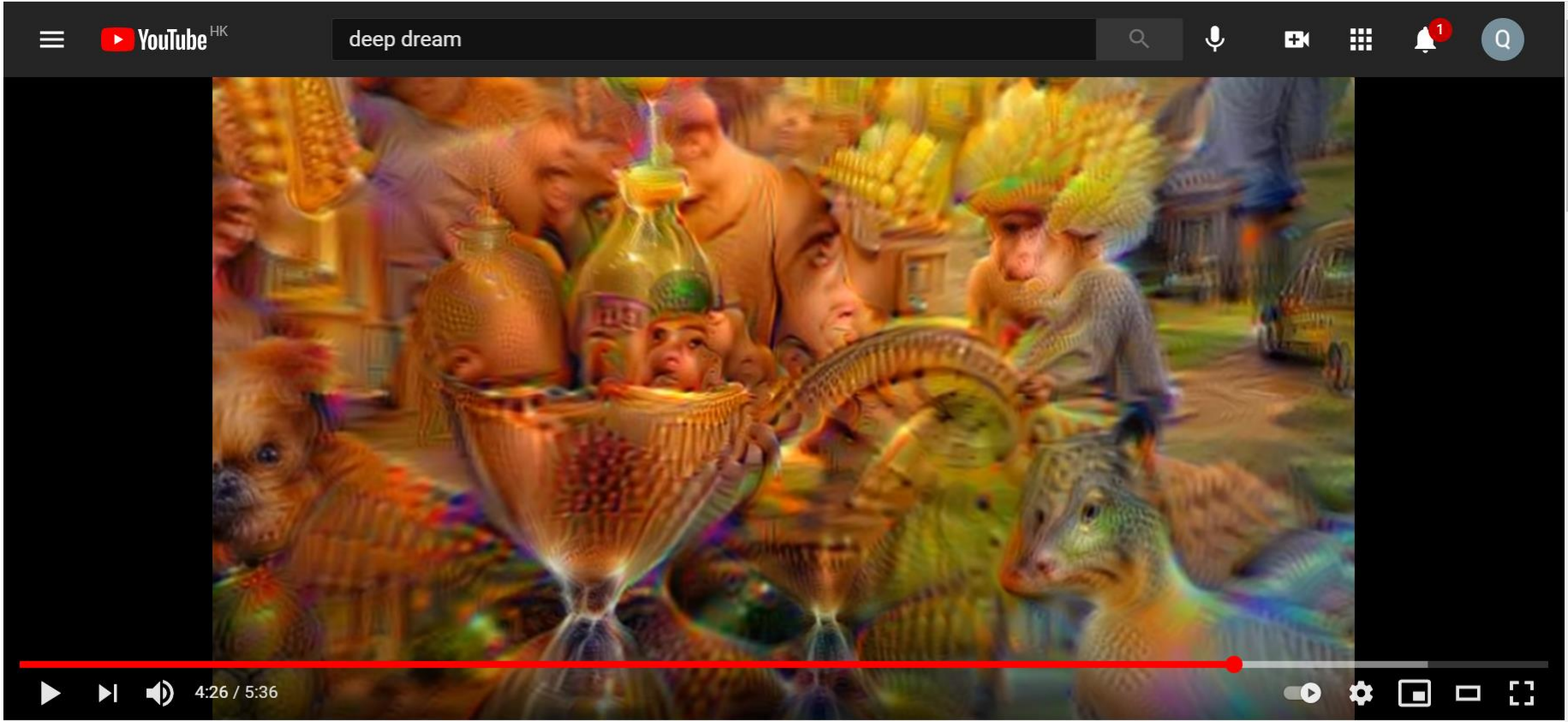
1.9K 16 SHARE SAVE



3D Style Transfer For Video is Now Possible!

Two Minute Papers

156K views • 1 year ago



Journey on the Deep Dream

455,747 views • Jul 8, 2015

6.9K 123 SHARE SAVE

DeepDream: Inside Google's 'Daydreaming' Computers

SciShow

371K views • 5 years ago

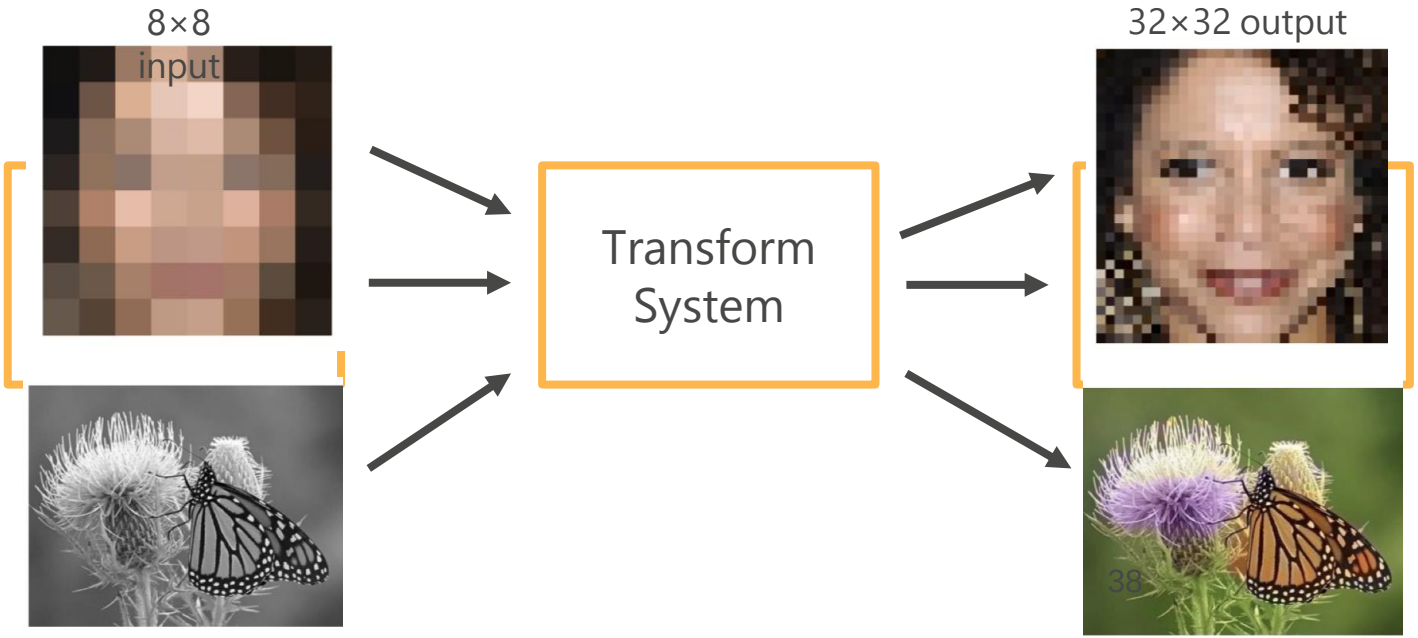
Perceptual losses for real-time style transfer and super-resolution (ECCV 2016)

Task: Image Transformation

A system receives an input image and transforms it to an output image

- super-resolution
- colorization
- denoising etc.

This paper focuses on style-transfer & super-resolution



Previous Approach

1. train a feed-forward transform network with per-pixel loss (for image translation super-resolution)

- 👍: **fast in inference**
- 👎: cannot measure perceptual differences

per-pixel loss

difference between images on pixel-lev

$$\mathrm{MSE}(x, y) = \frac{1}{n} \sum_i |x_i - y_i|^2$$

$$\mathrm{MAE}(x, y) = \frac{1}{n} \sum_i |x_i - y_i|$$

2. use perceptual loss function

- 👍: **high-quality images**
- 👎: slow in inference (need to solve an optimization problem)

perceptual loss

difference between high-level feature representations extracted from pretrained CNN

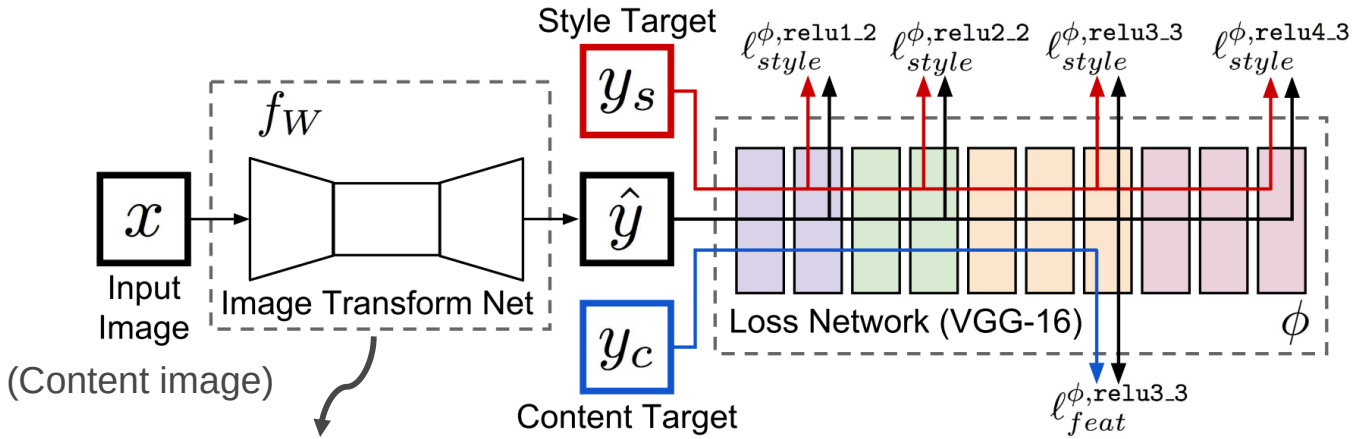
contents similarity ?

[Gatys+ 2016]

Proposed Method (1/2)

combine the benefits of the two approaches

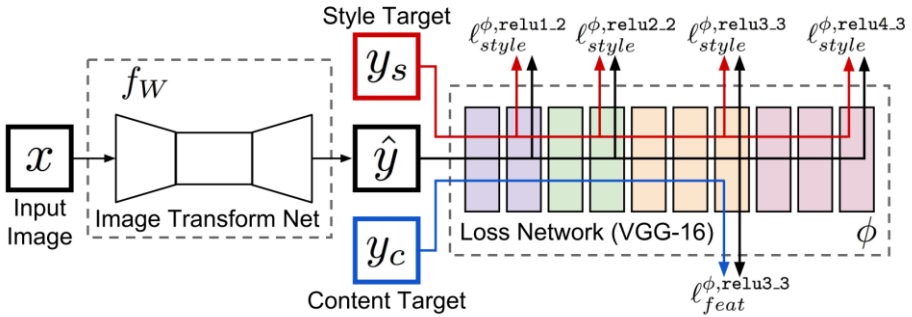
→ train **a feed-forward transform net** with **perceptual loss function**



Combination of

- Convolutional layers
- Residual blocks [He+ 2015]
- Transposed convolutional layers

Proposed Method (2/2) – loss function



After Transform Net, images are passed to Loss Network

- pretrained VGG16[Simonyan & Zisserman 2015] on ImageNet[Russakovsky+ 2015]
- compute feature reconstruction loss l_{feat} & style reconstruction loss l_{style}

$\boldsymbol{l_{feat}}$: Euclidean distances between feature representation

$$\ell^{\phi,j}_{feat}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

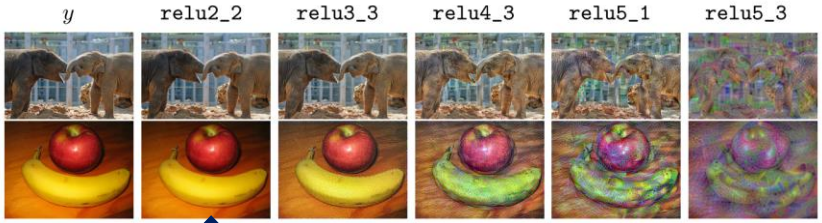


Fig. 3. Similar to [ ] use optimization to find an image $\hat{y}$ that minimizes the feature reconstructio[n] loss $\ell^{\phi,j}_{feat}(\hat{y}, y)$ for several layers j from the pretrained VGG-16 loss network ϕ. As we reconstruct from higher layers, image content and overall spatial structure are preserved, but color, texture, and exact shape are not.

$\boldsymbol{l_{style}}$: Frobenius norm of the difference between the Gram matrices of the output and targets $\ell^{\phi,j}_{style}(\hat{y}, y) = \|G^{\phi}_{j}(\hat{y}) - G^{\phi}_{j}(y)\|_F^2$

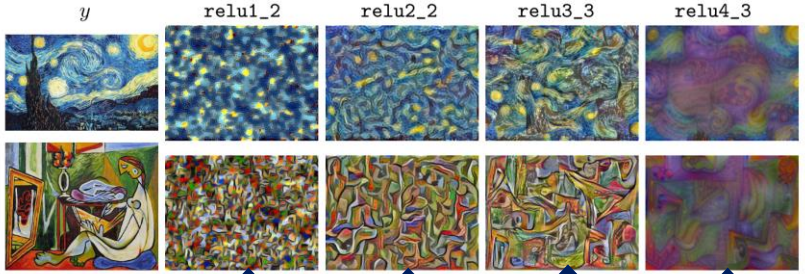


Fig. 4. Similar to [10], [ ] optimizati[on] find an ima[ge] at minimiz[es] style reconstruction loss $\ell^{\phi,j}_{style}$[$(\hat{y}, y)$] for several [l]ers j from t[he] pretrained V[GG]-16 loss network ϕ. The images [ ]

the sum of losses from 4 layers

Experiments – Style Transfer

Dataset: MS COCO 2014[Lin+ 2014] train (80k)
Batch size of 4 for 40,000 iterations (~2 epochs)
Adam[Kingma & Ba 2014] with the learning rate 1×10^{-3}

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

Image Size	Gatys *et al* [10] 100	300	500	Ours	Speedup 100	300	500
256 × 256	3.17	9.52s	15.86s	**0.015s**	212x	636x	**1060x**
512 × 512	10.97	32.91s	54.85s	**0.05s**	205x	615x	**1026x**
1024 × 1024	42.89	128.66s	214.44s	**0.21s**	208x	625x	**1042x**

- similar good quality image
- three orders of magnitude faster

Experiments – Super-Resolution

train

288 × 288 patches from MS COCO 2014[Lin+ 2014] train (10k)

→ prepare low-resolution inputs

- blurring with a Gaussian kernel of width $\sigma = 1.0$
- downsampling with bicubic interpolation

Batch size of 4 for 200k iterations

Adam[Kingma & Ba 2014] with the learning rate 1×10^{-3}

test

evaluated on the following datasets:

- × 4: Set5[Bevilacqua+ 2012]
- × 8: Set14[Zeyde+ 2010] & BSD100[Huang+ 2015]

measured by PSNR / SSIM

Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [11]	Ours (ℓ_{feat})
This image	31.78 / 0.8577	31.47 / 0.8573	32.99 / 0.8784	29.24 / 0.7841
Set5 mean	28.43 / 0.8114	28.40 / 0.8205	30.48 / 0.8628	27.09 / 0.7680
Set14 mean	25.99 / 0.7301	25.75 / 0.6994	27.49 / 0.7503	24.99 / 0.6731
BSD100 mean	25.96 / 0.682	25.91 / 0.6680	26.90 / 0.7101	24.95 / 63.17

☝ l_{feat} worked best

l_{pixel} and l_{feat} uses the same settings except the loss function

→ l_{pixel} gives higher PSNR ***BUT***
l_{feat} does a better job at the detail reconstruction

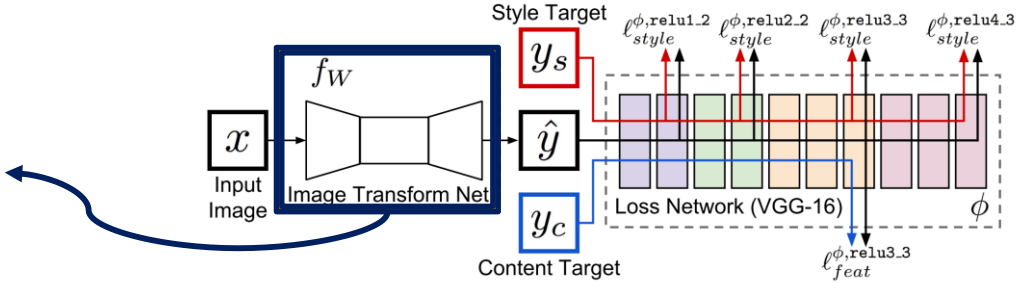
Conclusion

- combine the benefits of existing methods for image transformation tasks:
  - train a feed-forward network with per-pixel loss functions
  - optimization-based method with perceptual loss functions

  → **train a feed-forward network** with **perceptual loss** functions!
- style-transfer: comparable quality in significantly increased speed
- super-resolution: better reconstruction in details and edges

Transform Net Structure Details

Combination of
- Convolutional layers
- Residual blocks
- Transposed convolutional layers



Style Transfer

Layer	Activation size
Input	$3 \times 256 \times 256$
$32 \times 9 \times 9$ conv, stride 1	$32 \times 256 \times 256$
$64 \times 3 \times 3$ conv, stride 2	$64 \times 128 \times 128$
$128 \times 3 \times 3$ conv, stride 2	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride 1/2	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

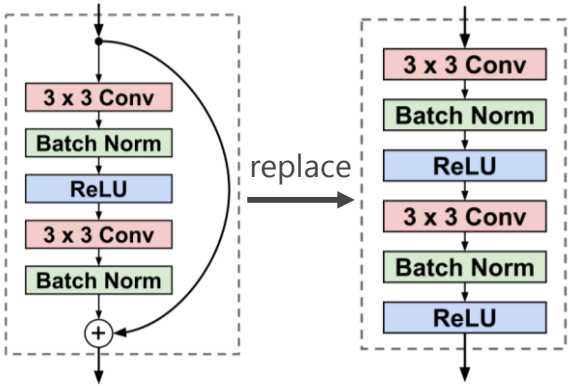
Table 1. Network architecture used for style transfer networks.

Super-Resolution

×4		×8	
Layer	**Activation size**	**Layer**	**Activation size**
Input	$3 \times 72 \times 72$	Input	$3 \times 36 \times 36$
$64 \times 9 \times 9$ conv, stride 1	$64 \times 72 \times 72$	$64 \times 9 \times 9$ conv, stride 1	$64 \times 36 \times 36$
Residual block, 64 filters	$64 \times 72 \times 72$	Residual block, 64 filters	$64 \times 36 \times 36$
Residual block, 64 filters	$64 \times 72 \times 72$	Residual block, 64 filters	$64 \times 36 \times 36$
Residual block, 64 filters	$64 \times 72 \times 72$	Residual block, 64 filters	$64 \times 36 \times 36$
Residual block, 64 filters	$64 \times 72 \times 72$	Residual block, 64 filters	$64 \times 36 \times 36$
$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 144 \times 144$	$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 72 \times 72$
$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 288 \times 288$	$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 144 \times 144$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 288 \times 288$	$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 288 \times 288$
-	-	$3 \times 9 \times 9$ conv, stride 1	$3 \times 288 \times 288$

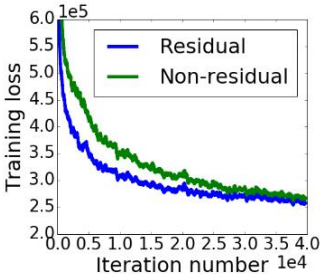
Table 2. Network architectures used for ×4 and ×8 super-resolution.

Residual vs non-residual



- trains faster but converges to similar loss
- no difference in learned transformation

16-layers of CNN in total is relatively shallow compared to [He+ 2015].

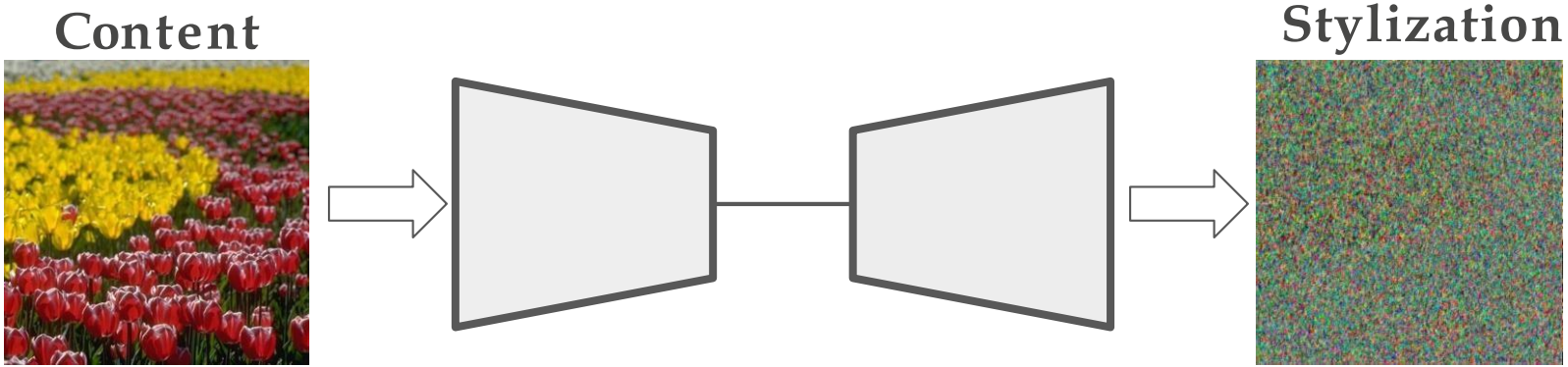


Style Content Residual Non-residual

Fig. 2. A comparison of residual vs non-residual networks for style transfer.

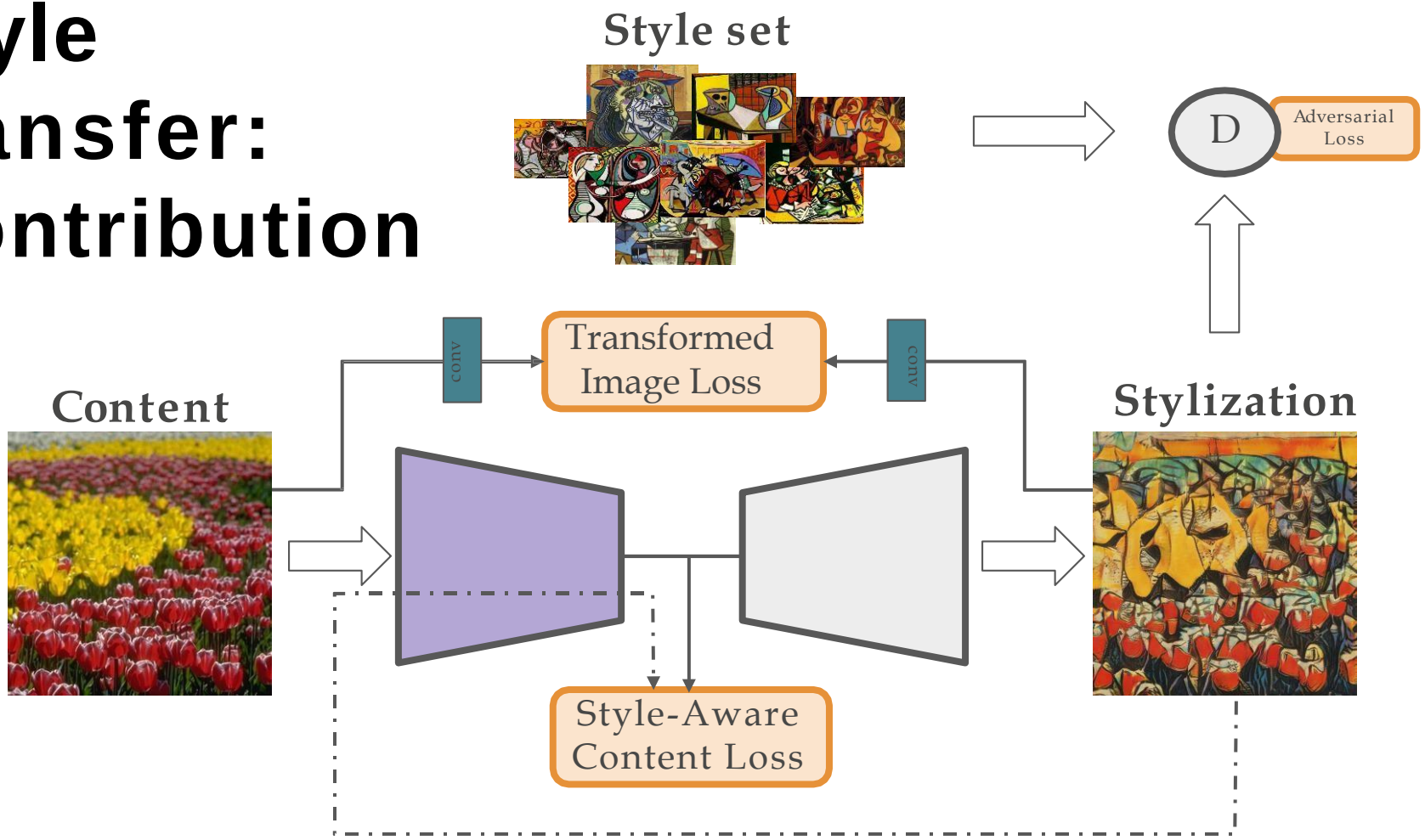
A Style-Aware Content Loss for Real-time HD Style Transfer (ECCV 2018)

Style transfer: Previous Approaches

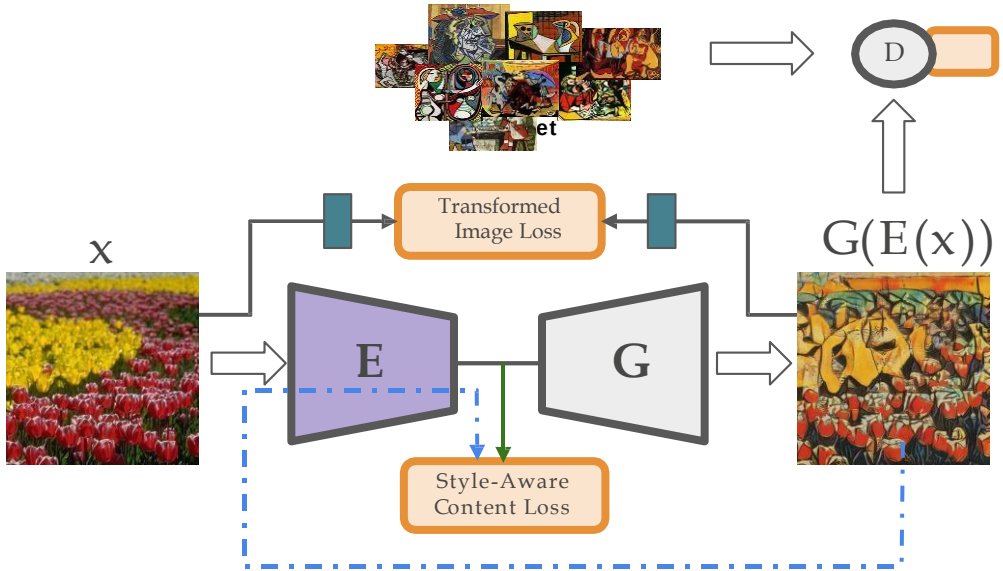


[1] Perceptual losses for real-time style transfer and super-resolution, Johnson et al., ECCV 2016

Style transfer: Contribution

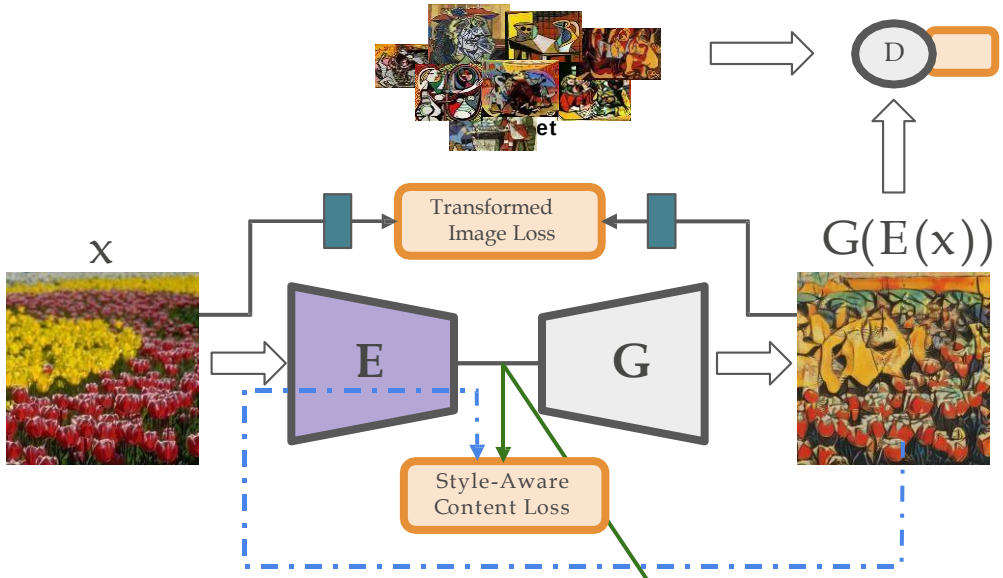


Style-Aware Content Loss



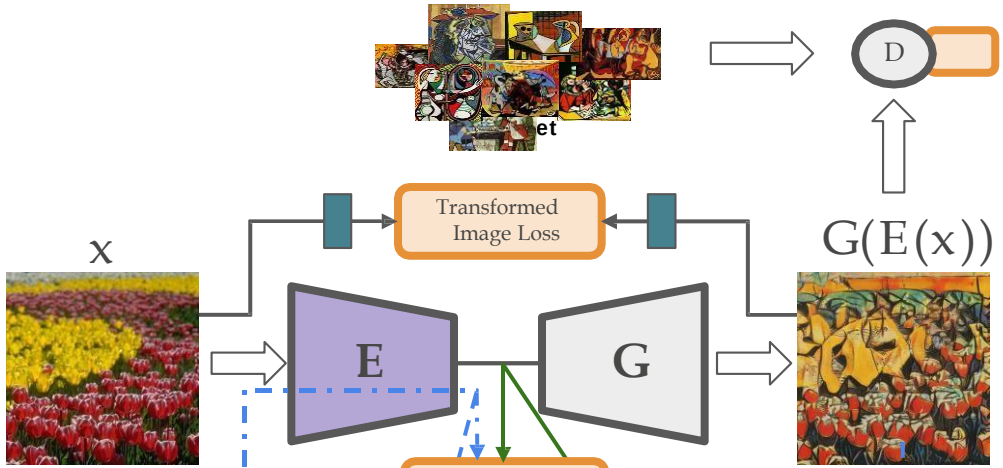
$$\mathcal{L}_c(E,G) = \mathop{\mathbb{E}}_{x\sim p_X(x)}\left[\frac{1}{d}\left\|E(G(E(x))) - E(x)\right\|_2^2\right]$$

Style-Aware Content Loss



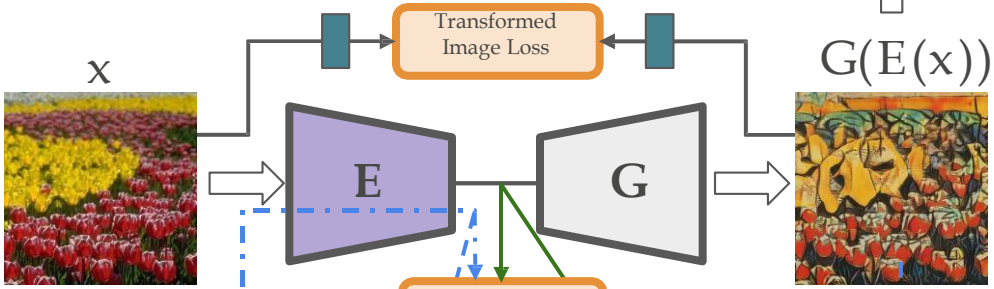
$$\mathcal{L}_c(E,G) = \mathop{\mathbb{E}}_{x\sim p_X(x)}\left[\frac{1}{d}\left\|E(G(E(x))) - \underbrace{E(x)}_{\text{Encoded content}}\right\|_2^2\right]$$

Style-Aware Content Loss



$$\mathcal{L}_c(E,G) = \mathop{\mathbb{E}}_{x\sim p_X(x)}\left[\frac{1}{d}\,\|\underbrace{E(G(E(x)))}_{\text{Encoded content of stylized image}} - \underbrace{E(x)}_{\text{Encoded content}}\|_2^2\right]$$

Style-Aware Content Loss

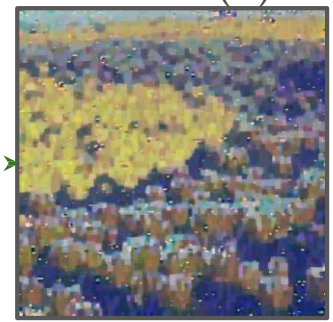


$$\mathcal{L}_c(E,G) = \mathop{\mathbb{E}}_{x\sim p_X(x)}\left[\frac{1}{d}\left\|E(G(E(x))) - E(x)\right\|_2^2\right]$$

Encoded content of stylized image

Encoded content

Reconstruction from E(x)



E(x) is a **style-dependent** content representation.

Full Objective

Style-Aware Content Loss

$$\mathcal{L}_c(E,G) = \mathop{\mathbb{E}}_{x\sim p_X(x)}\left[\frac{1}{d}\,\|\underbrace{E(x)}_{\text{Encoded content}} - \underbrace{E(G(E(x)))}_{\text{Encoded content of stylized image}}\|_2^2\right]$$

+

Transformed Image Loss

$$\mathcal{L}_{\mathbf{T}}(E,G) = \mathop{\mathbb{E}}_{x\sim p_X(x)}\left[\frac{1}{CHW}\|\underbrace{\mathbf{T}(x)}_{\text{Transf. content}} - \underbrace{\mathbf{T}(G(E(x)))}_{\text{Transformed stylized image}}\|_2^2\right]$$

+

T is one-layer fully convolutional network

Discriminator Loss

$$\mathcal{L}_D(E,G,D) = \mathop{\mathbb{E}}_{y\sim p_Y(y)}[\log D(y)] + \mathop{\mathbb{E}}_{x\sim p_X(x)}[\log(1 - D(G(E(x))))]$$

=

Full objective

$$\min_{E,G}\max_{D}\ \mathcal{L}_c(E,G) + \mathcal{L}_t(E,G) + \lambda\mathcal{L}_D(E,G,D)$$

Claude Monet: Real or Fake

Claude Monet: Real or Fake

real

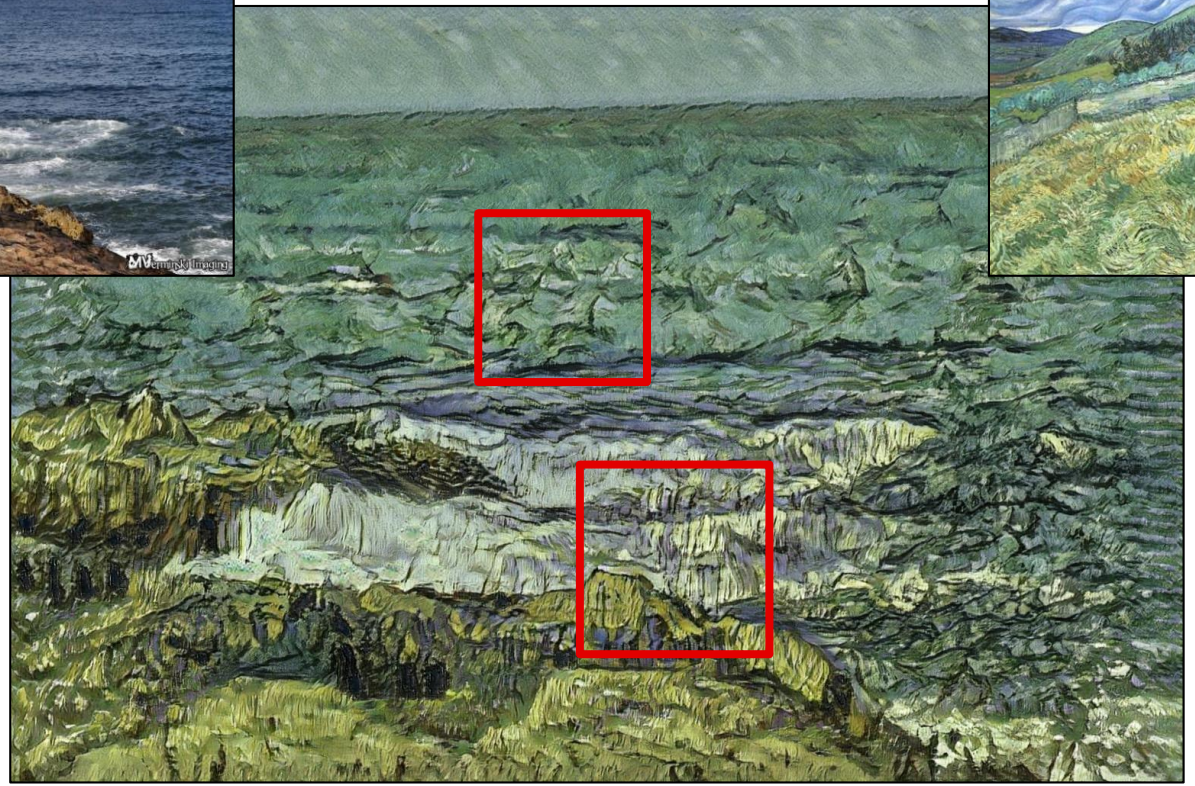
fake

real

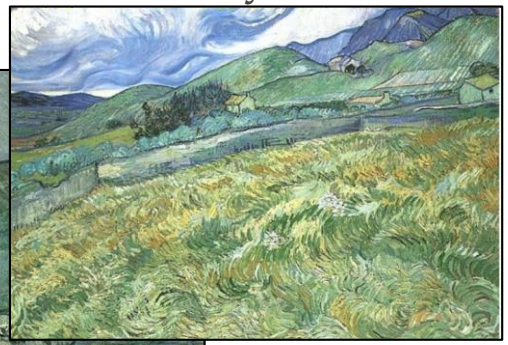
HD stylization

Content

Vincent van Gogh



Style

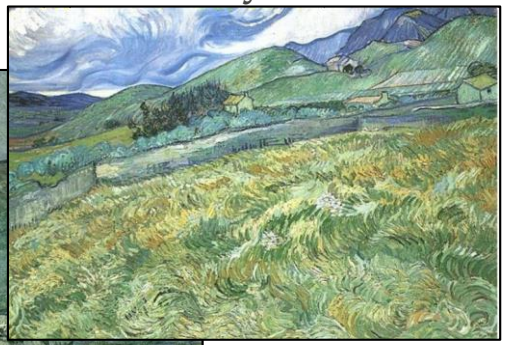


2.5 MP

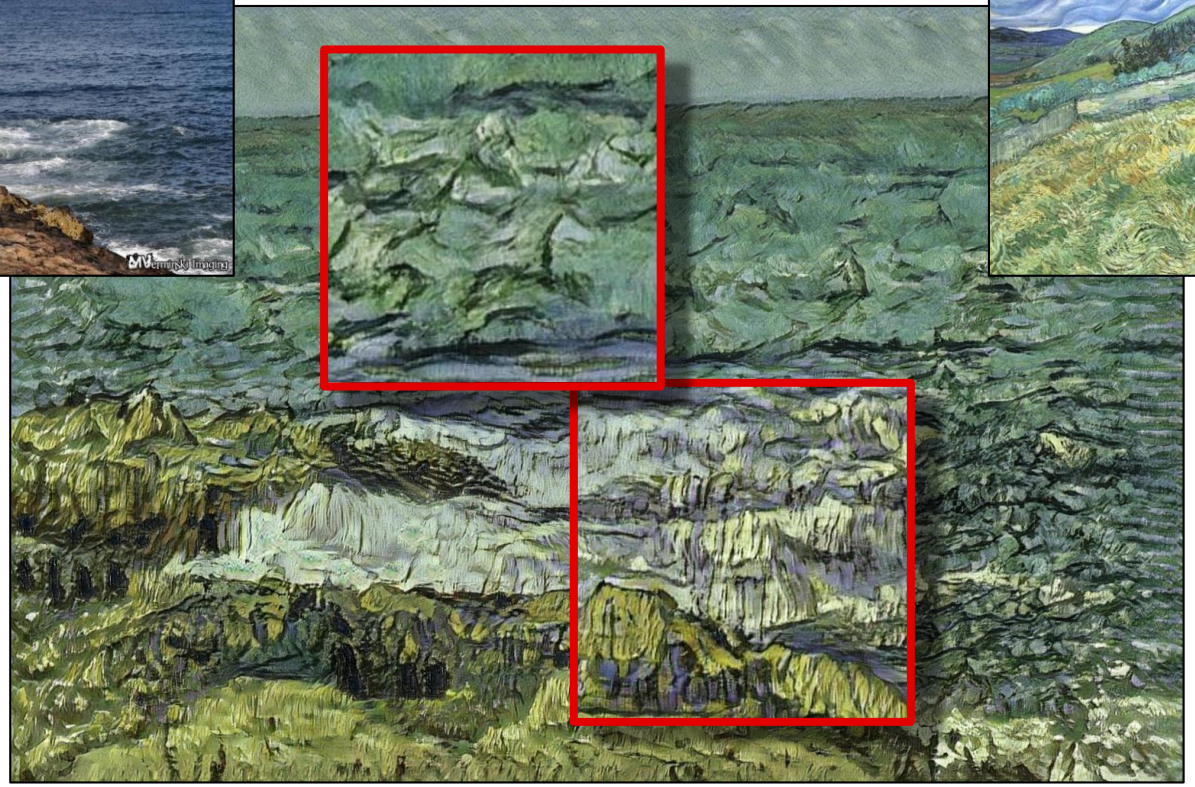
HD stylization

Content

Style



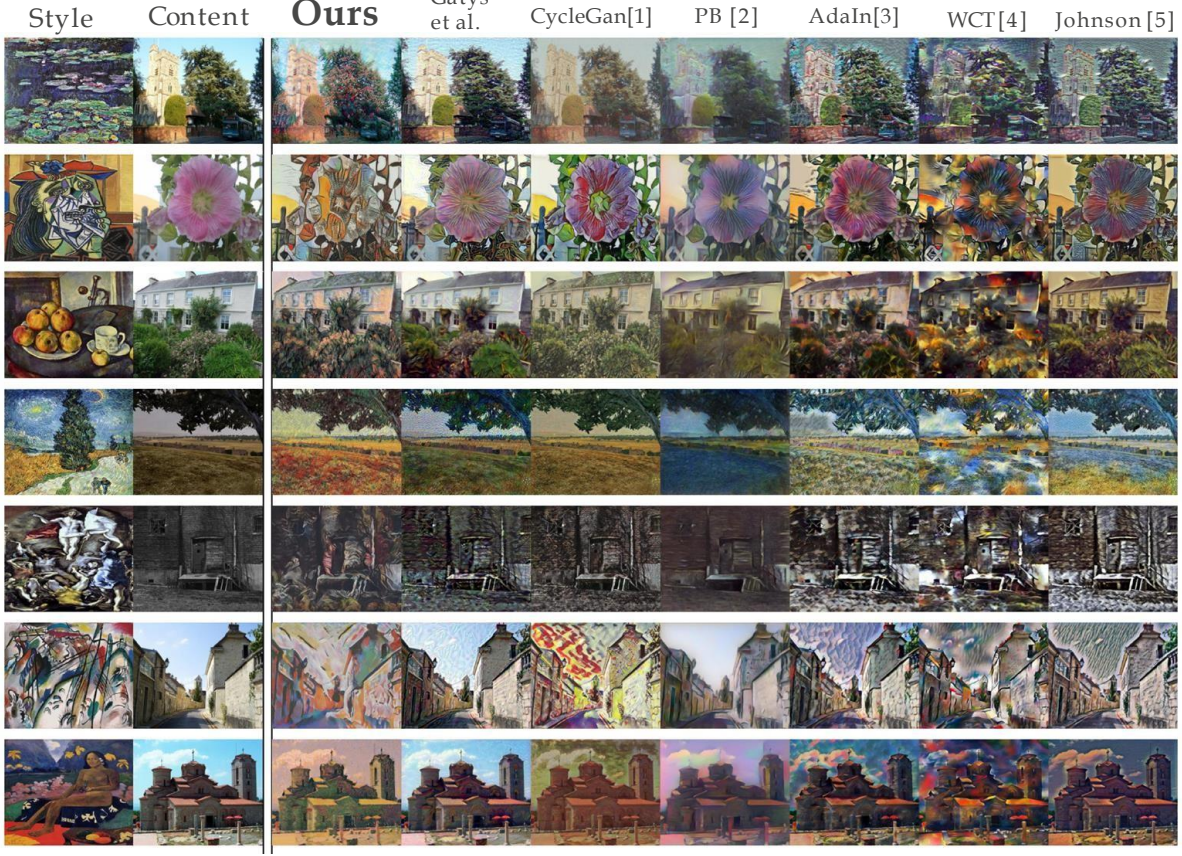
Vincent van Gogh



2.5 MP

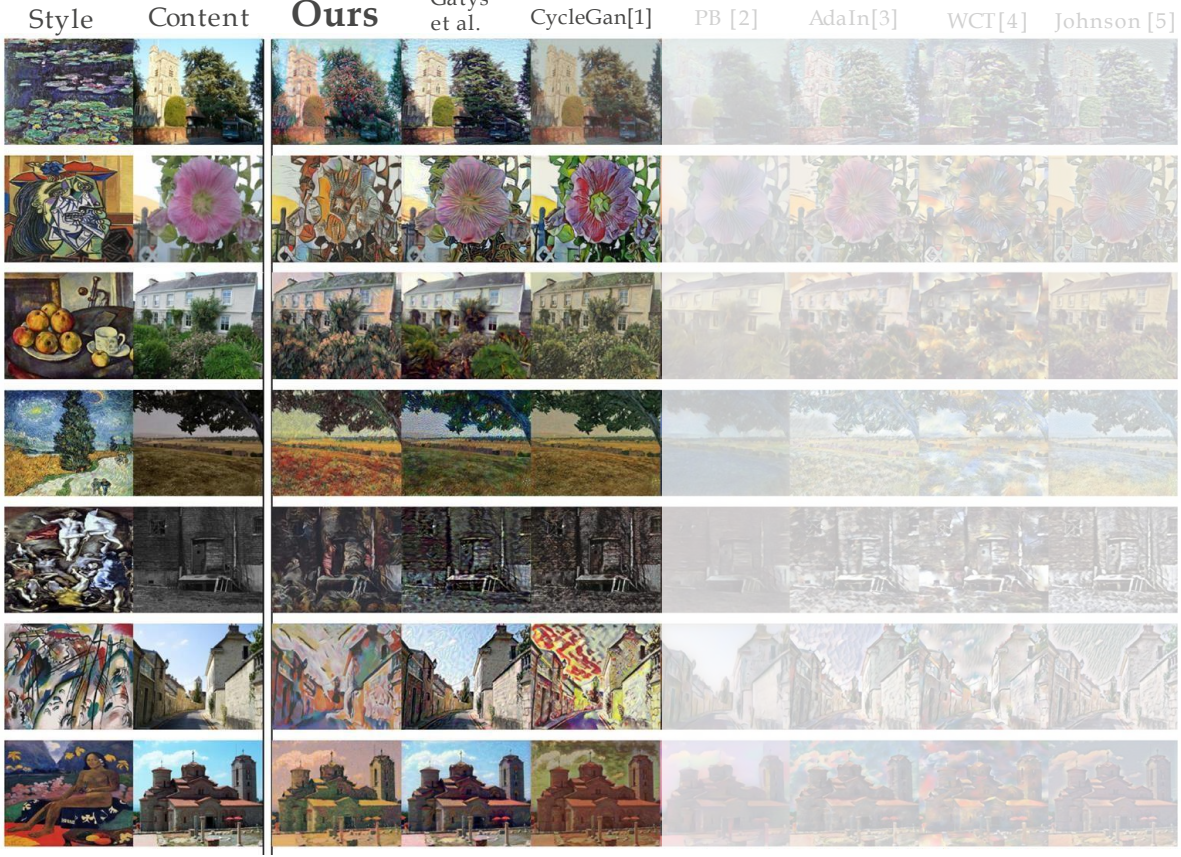
HD stylization

Comparison with other approaches



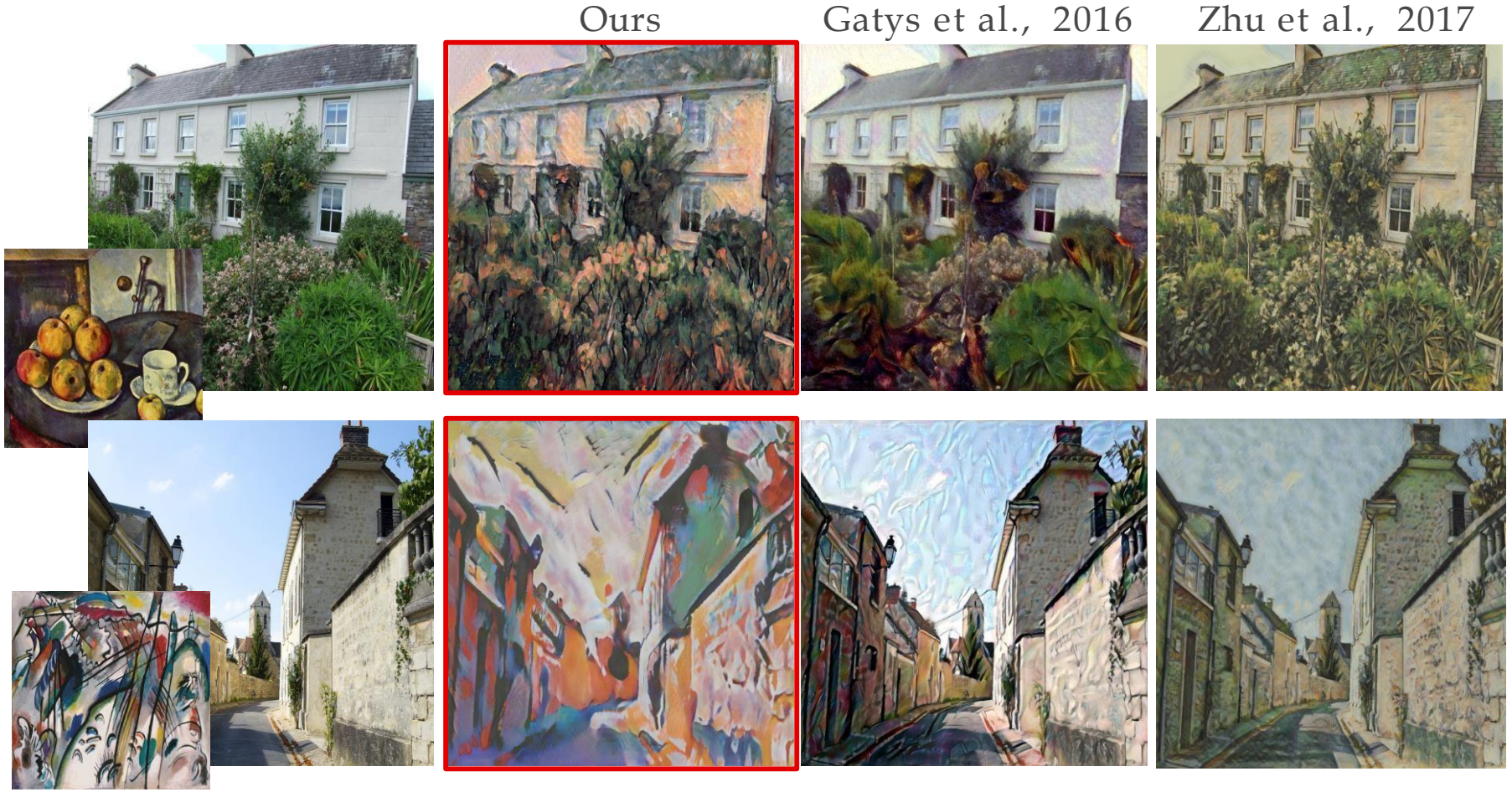
1 Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al., ICCV 2017
2 Fast patch-based style transfer of arbitrary style, Chen et al., 2016
3 Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization, Huang et al., ICCV 2017
4 Universal style transfer via feature transforms, Li et al., NIPS 2017
5 Perceptual losses for real-time style transfer and super-resolution, Johnson et al., ECCV 2016

Comparison with other approaches



1 Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al., ICCV 2017
2 Fast patch-based style transfer of arbitrary style, Chen et al., 2016
3 Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization, Huang et al., ICCV 2017
4 Universal style transfer via feature transforms, Li et al., NIPS 2017
5 Perceptual losses for real-time style transfer and super-resolution, Johnson et al., ECCV 2016

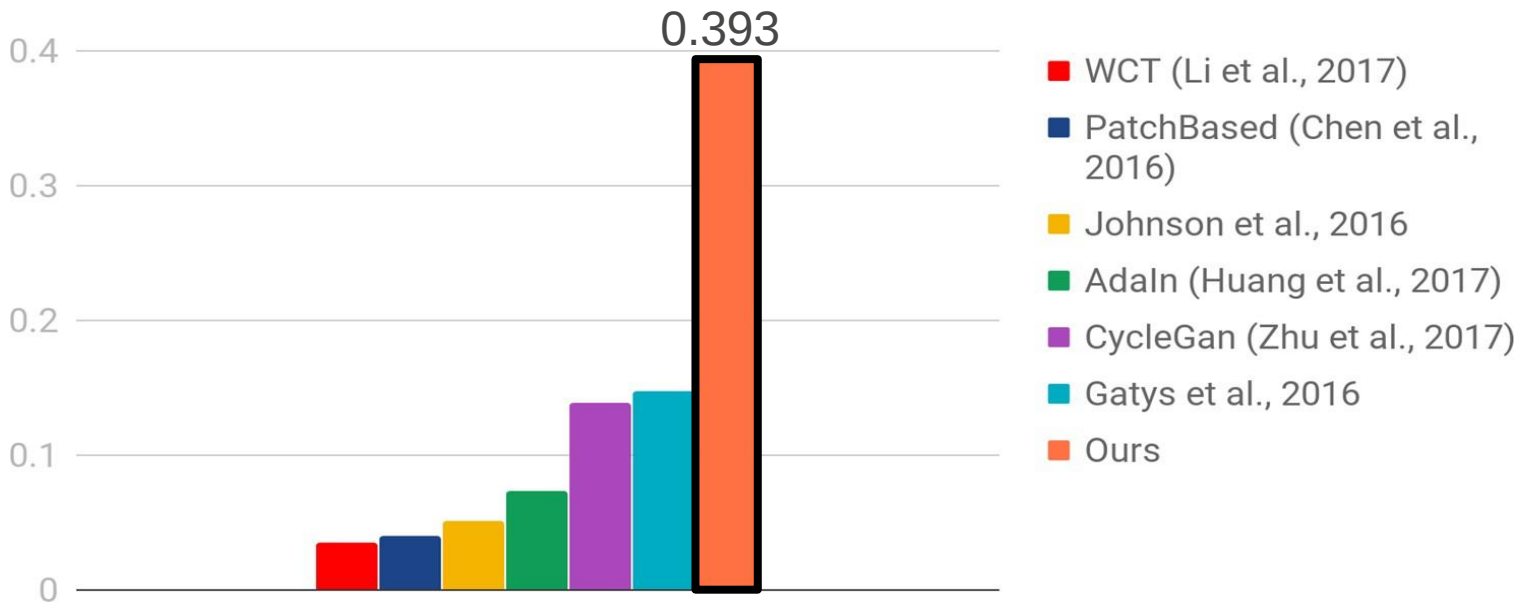
Comparison with other approaches



Deception Rate

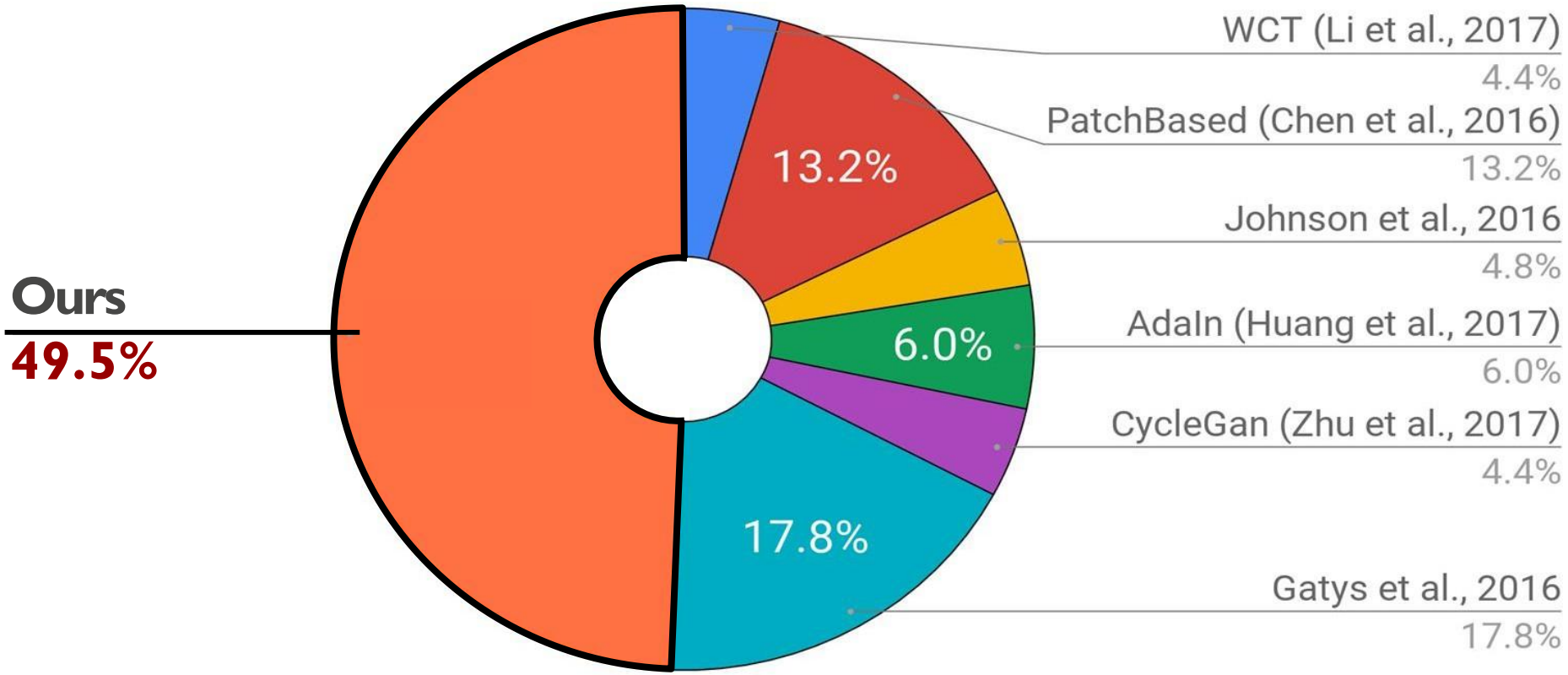
Stylizations → **Artist classifier (pretrained on real paintings)** → predict → Accuracy

Stylization **label** = **Author** of the style example

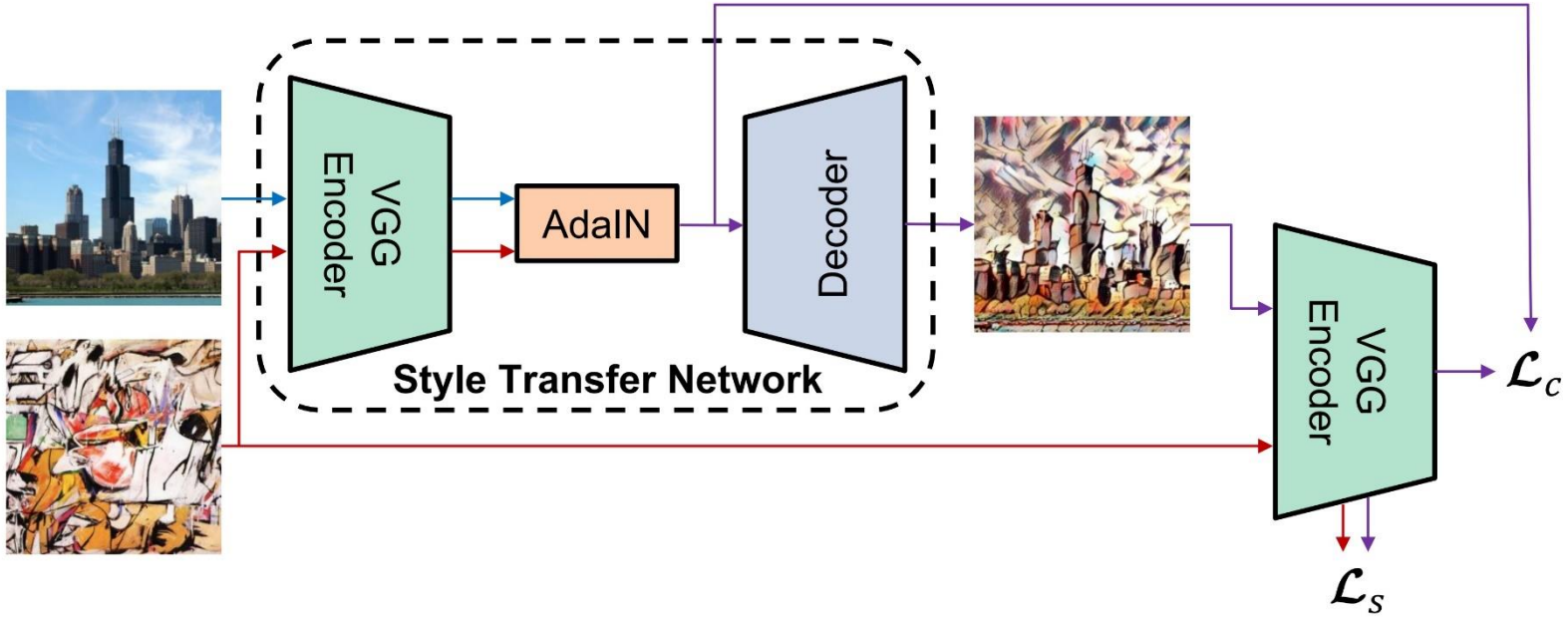


[*] Improved techniques for training gans, Salimans et al., NIPS 2016

Art History Expert Score

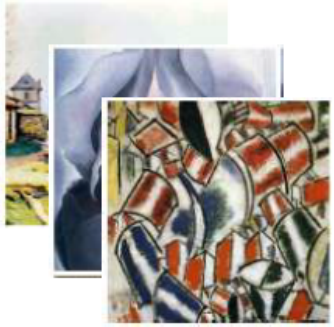


Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization (ICCV 2017)



Fast & Arbitrary Style Transfer

Combining the flexibility of the *optimization-based framework*
and the speed similar to the *fastest feed-forward approaches*



Style

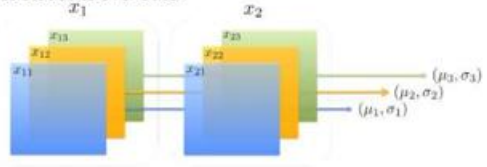
Content

Fast Feedforward Network

Output

Batch Normalization (BN)

- BN computes statistics *over batch*



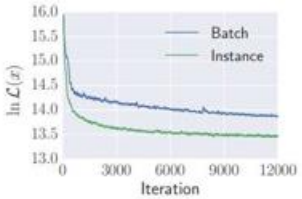
$$\mathrm{BN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

$$\mu_c(x) = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \qquad (2)$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw} - \mu_c(x))^2 + \epsilon} \qquad (3)$$

(c) Content. (d) StyleNet IN.

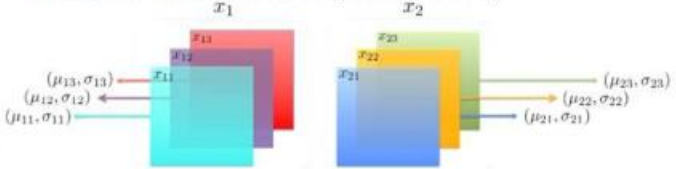
(f) Style. (g) StyleNet BN.



(a) Feed-forward history.

Instance Normalization (IN)

- **IN** computes statistics *for each object independently*



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

$$\mu_{nc}(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \qquad (5)$$

$$\sigma_{nc}(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw} - \mu_{nc}(x))^2 + \epsilon} \qquad (6)$$

Adaptive Instance Normalization

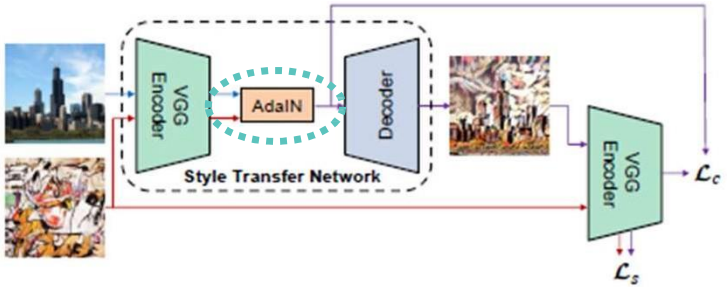


Figure 2. An overview of our style transfer algorithm. We use the first few layers of a fixed VGG-19 network to encode the content and style images. An AdaIN layer is used to perform style transfer in the feature space. A decoder is learned to invert the AdaIN output to the image spaces. We use the same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

$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \qquad (8)$$

content input

style input

normalized content image

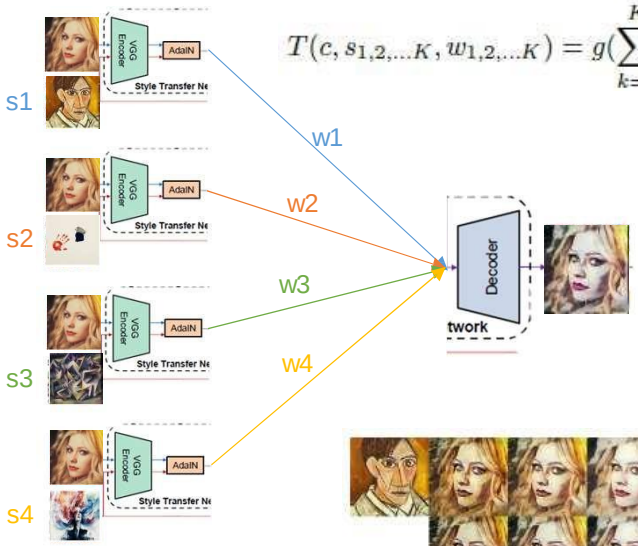
style input & affine parameter

Results

Figure 4. Example style transfer results. All the tested content and style images are never observed by our network during training.

Style Interpolation

$$T(c, s_{1,2,...K}, w_{1,2,...K}) = g(\sum_{k=1}^{K} w_k \text{AdaIN}(f(c), f(s_k)))$$



The End