Neural Image Processing:
Style Transfer, Image Transformation, Neural Compression

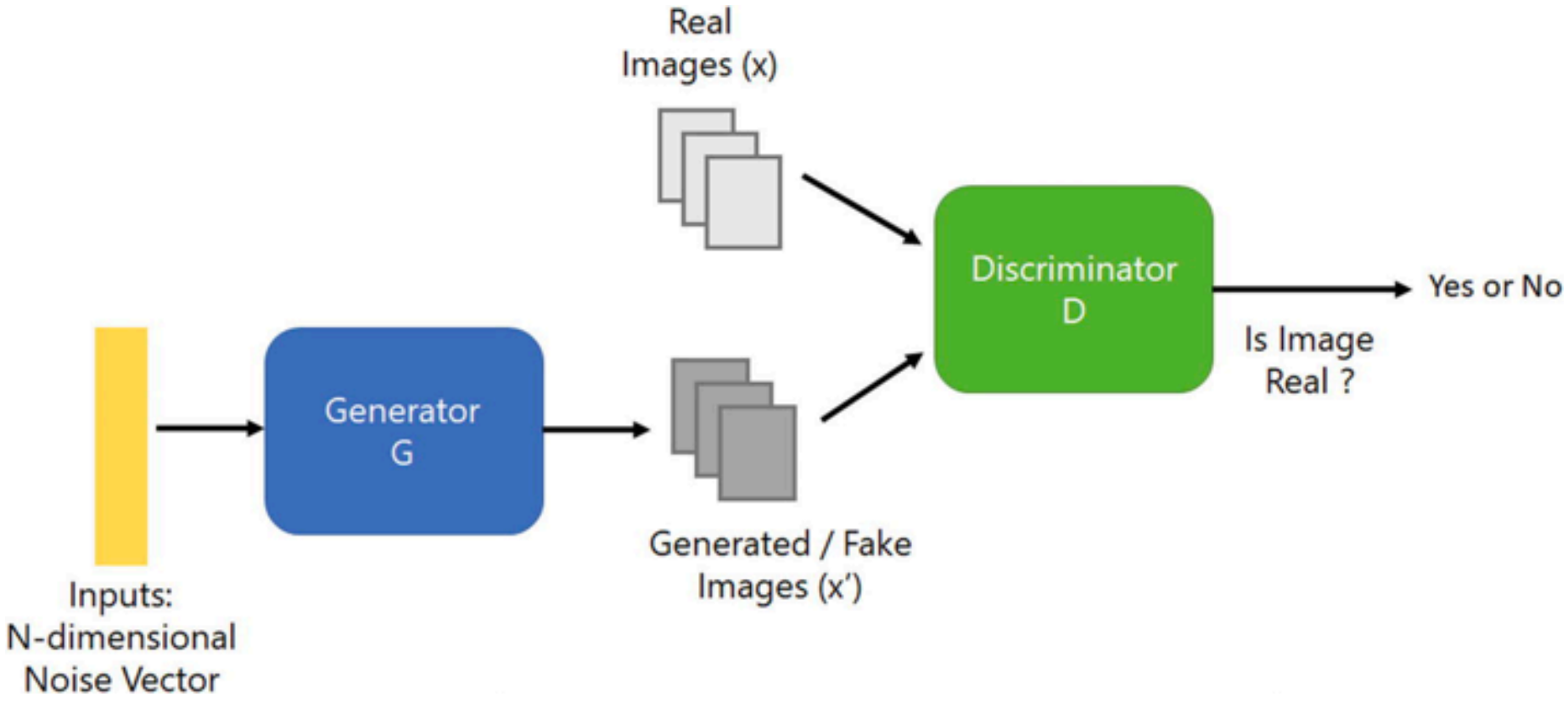
Bolei Zhou

IERG4190/IEMS5707

image credit to neuralstyle.art

Review: Generative Adversarial Networks

A two-player minimax game between a generator and a discriminator



Goodfellow et al. NIPS'14

Training objective for discriminator:

$$\max_D V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

Training objective for generator:

$$\min_G V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

$$\min_\theta \max_\phi V(G_\theta, D_\phi) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

https://github.com/pytorch/examples/blob/master/dcgan/main.py

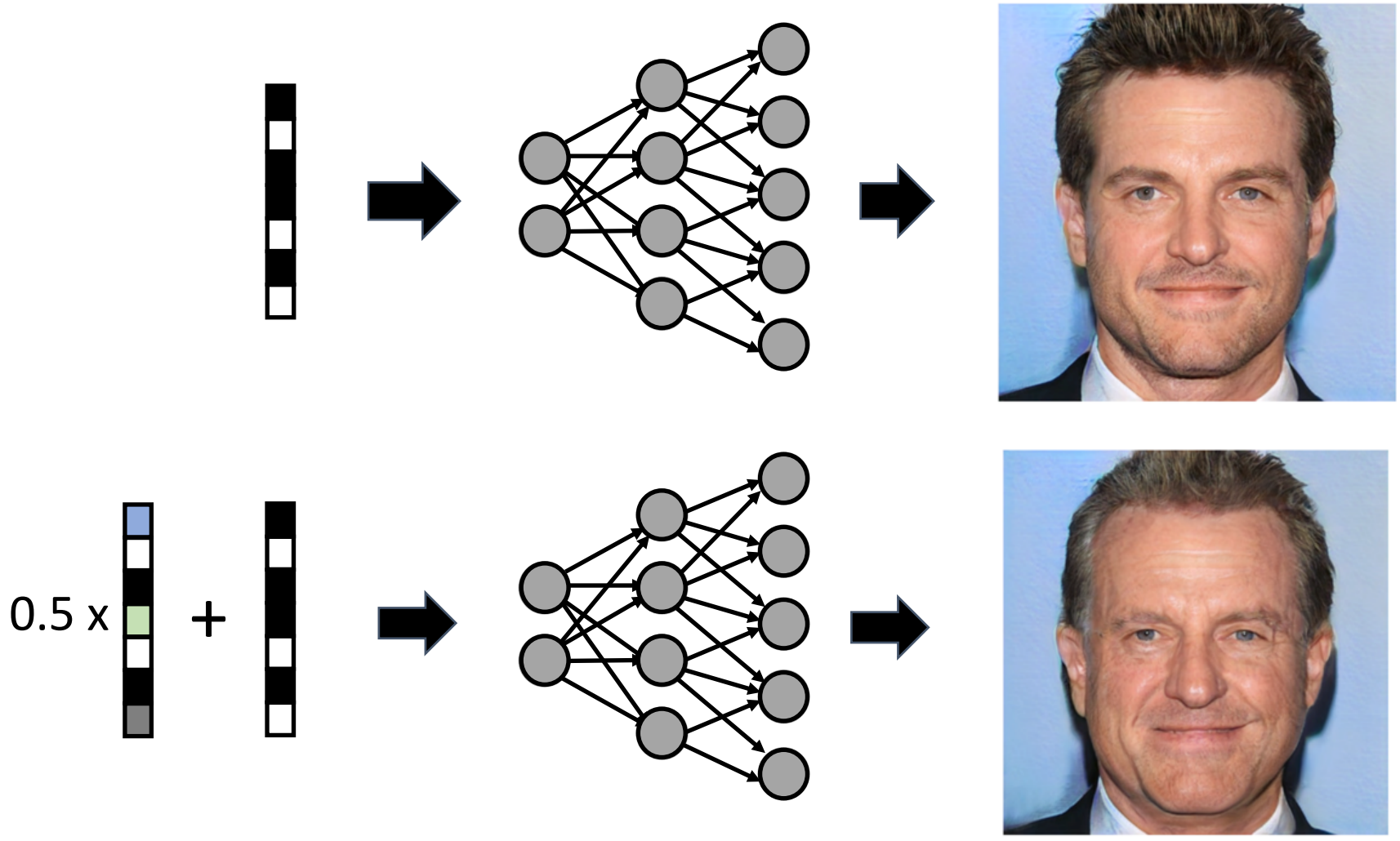
Example Code of Adversarial Training

```
for i, data in enumerate(dataloader, 0):
    ############################
    # (1) Update D network: maximize log(D(x)) + log(1 - D(G(z)))
    ###########################
    # train with real
    netD.zero_grad()
    real_cpu = data[0].to(device)
    batch_size = real_cpu.size(0)
    label = torch.full((batch_size,), real_label, device=device)

    output = netD(real_cpu)
    errD_real = criterion(output, label)
    errD_real.backward()
    D_x = output.mean().item()

    # train with fake
    noise = torch.randn(batch_size, nz, 1, 1, device=device)
    fake = netG(noise)
    label.fill_(fake_label)
    output = netD(fake.detach())
    errD_fake = criterion(output, label)
    errD_fake.backward()
    D_G_z1 = output.mean().item()
    errD = errD_real + errD_fake
    optimizerD.step()
```

```
    ############################
    # (2) Update G network: maximize log(D(G(z)))
    ###########################
    netG.zero_grad()
    label.fill_(real_label)  # fake labels are real for generator cost
    output = netD(fake)
    errG = criterion(output, label)
    errG.backward()
    D_G_z2 = output.mean().item()
    optimizerG.step()
```

https://github.com/pytorch/examples/blob/master/dcgan/main.py

Image Transformation through Latent Manipulation



InterFaceGAN, Shen, Gu, Tang, Zhou, CVPR'20

Image to Image Transformation

noise in
image out

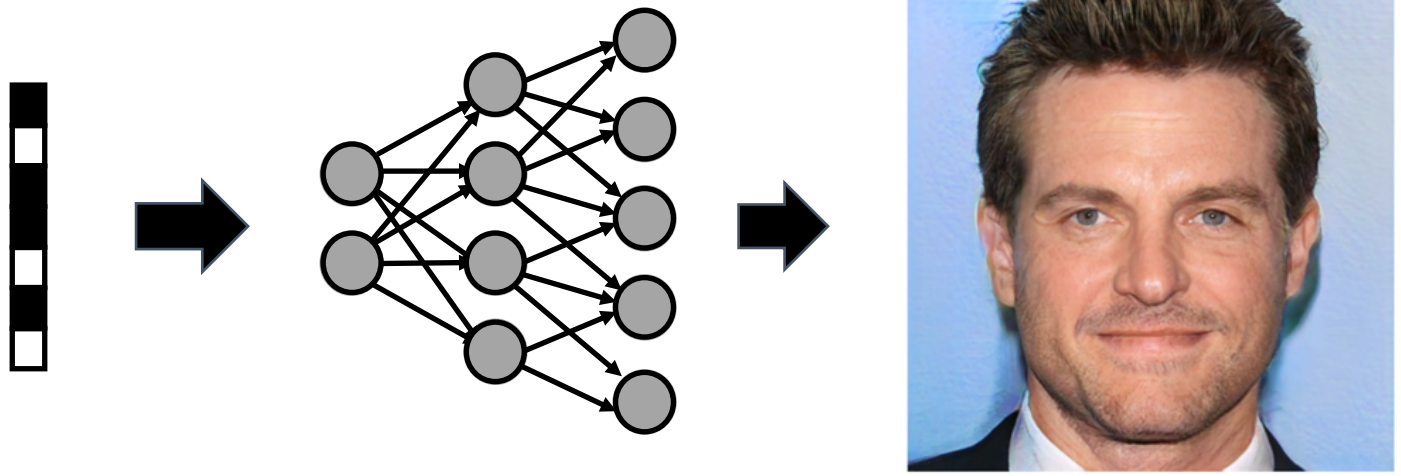
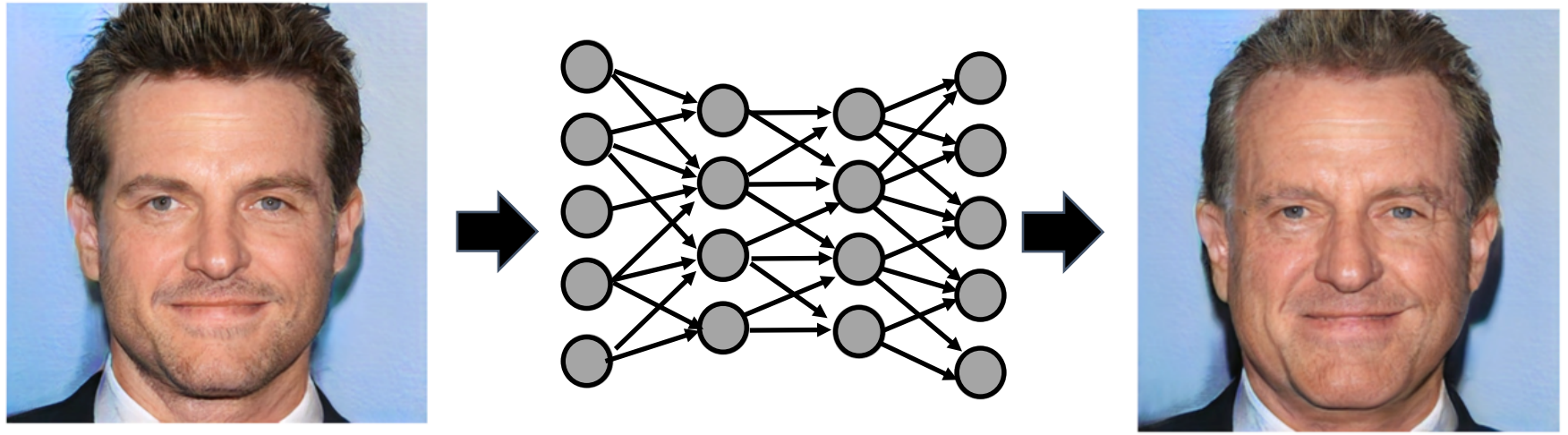
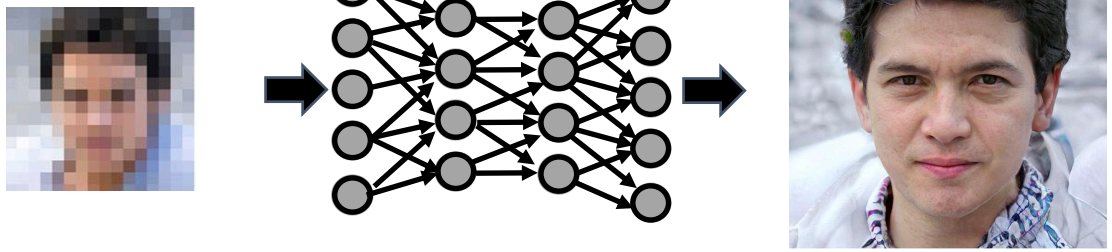


image in
image out

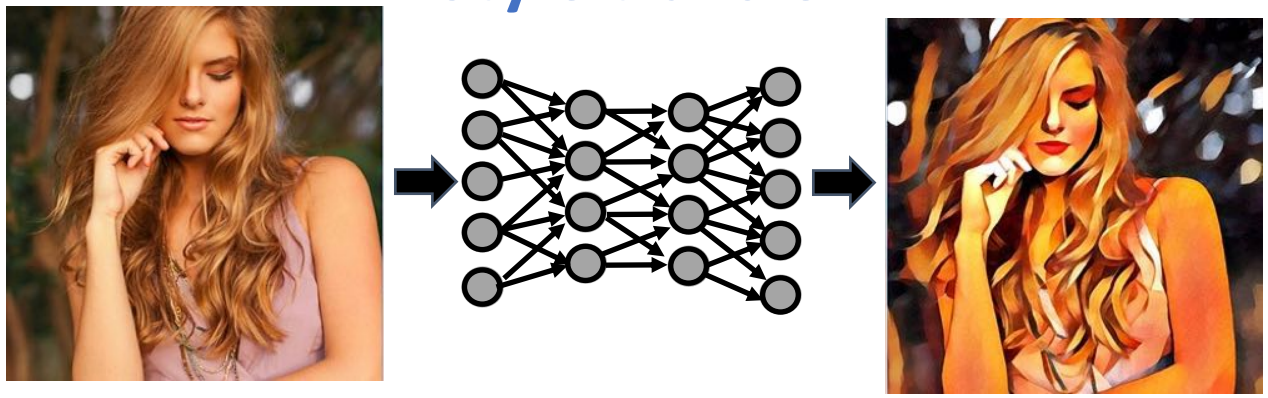


Neural Image Processing

Super-resolution



Style transfer



Colorization

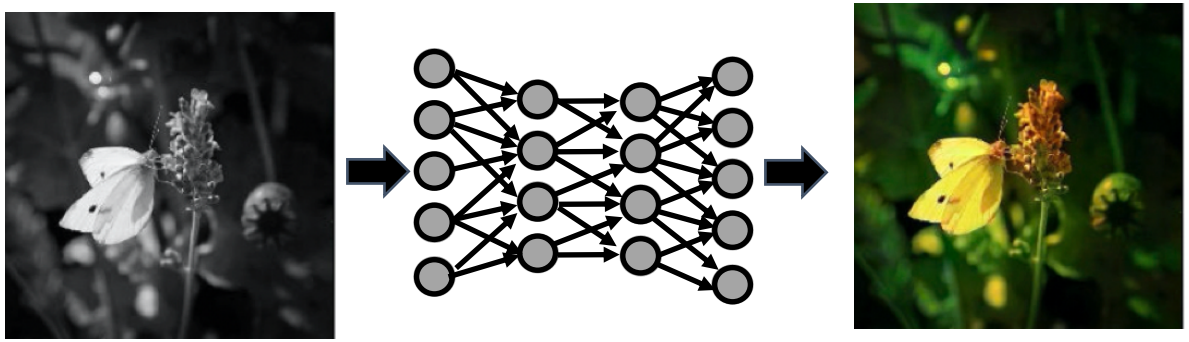
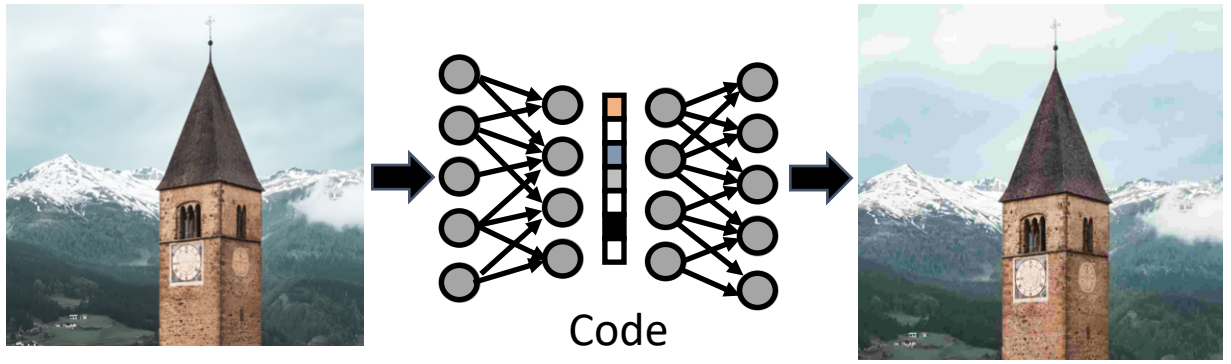
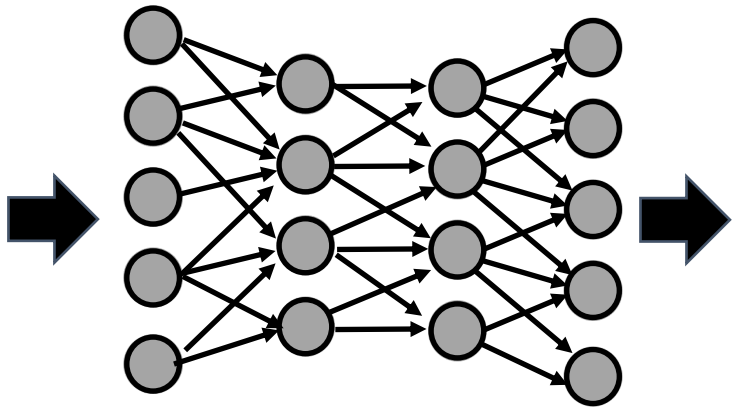


Image compression



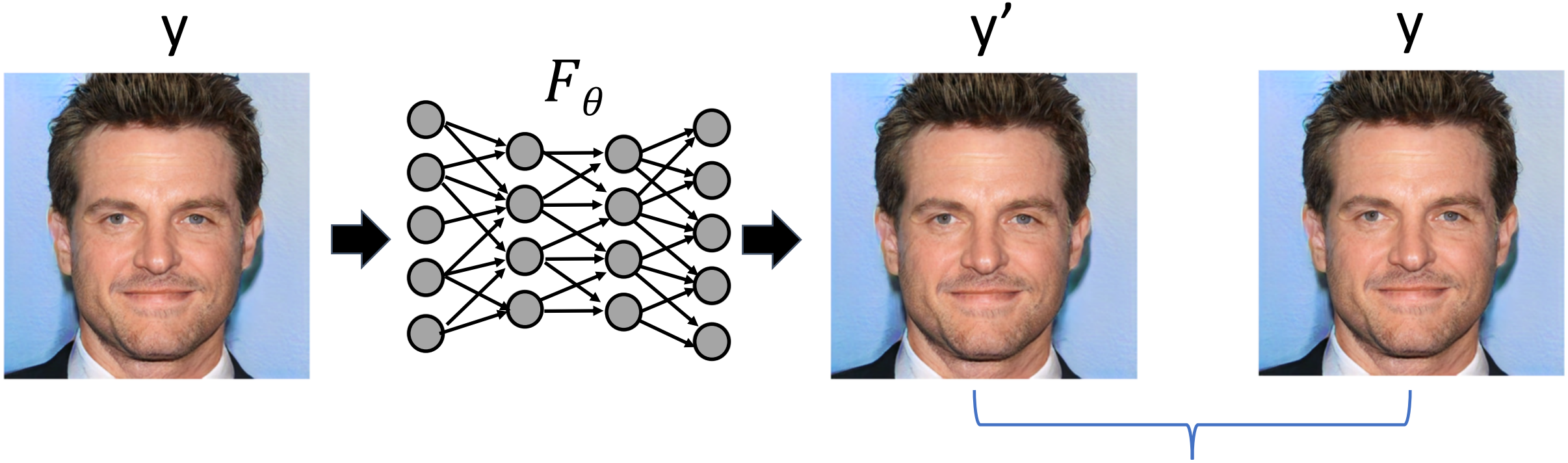
Basic model
Image Reconstruction through Auto-Encoder

y



y’

Reconstruction Loss d(y', y)



Per-pixel loss d(y', y) = || y' − y| |^2

$$\theta^* = \text{arg}min \, \|F_\theta(y) - y\|$$

Reconstruction Loss d(y', y)

- per-pixel loss does not capture perceptual differences between output and ground-truth images.
- Consider two identical images offset from each other by a few pixels; despite their perceptual similarity they would be very different as measured by per-pixel losses.

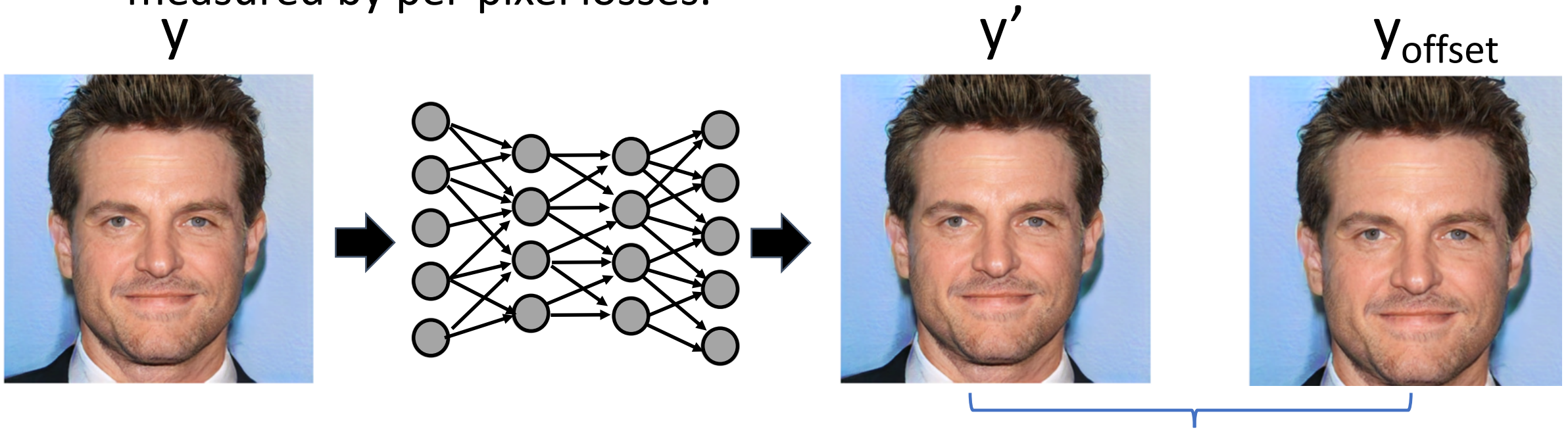
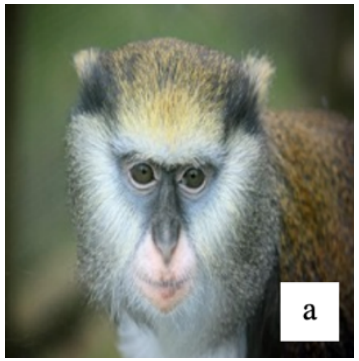
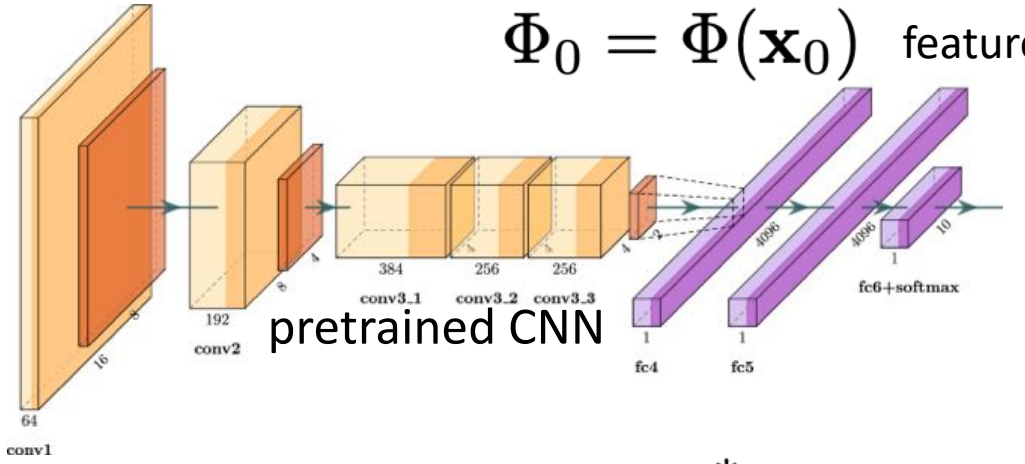


Image Reconstruction using CNN Feature Loss

x_0



$$\Phi_0 = \Phi(\mathbf{x}_0)$$ feature at conv5

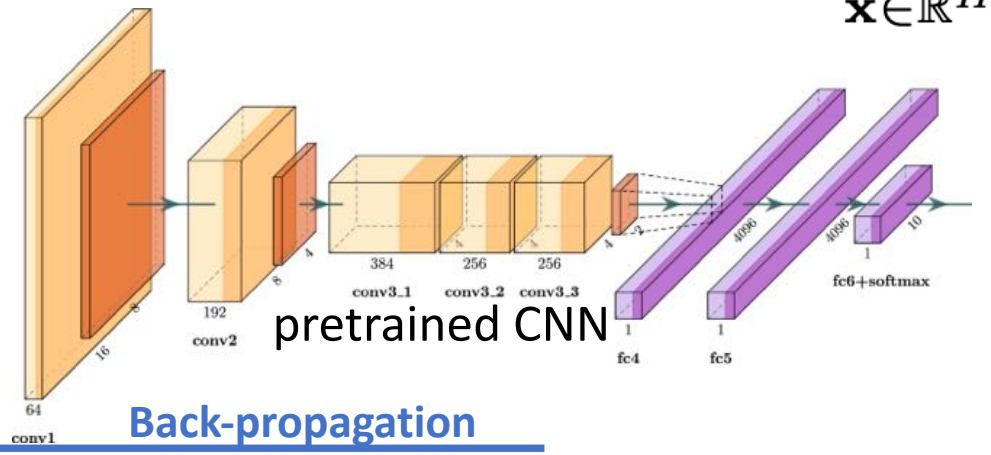


$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

conv5 feature loss

prior

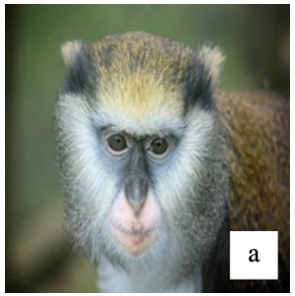
x

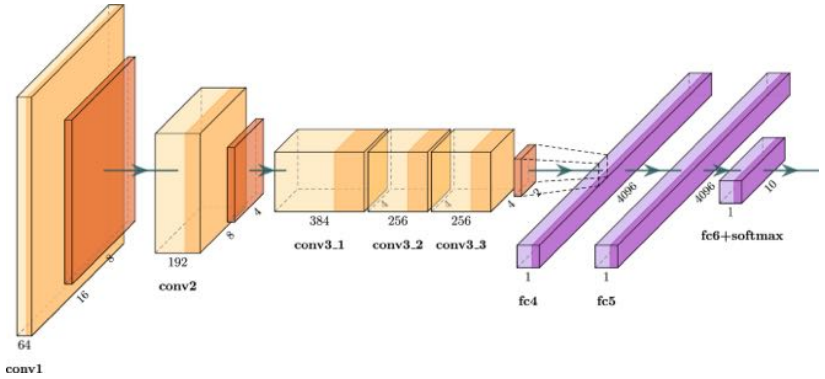


Back-propagation

Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)

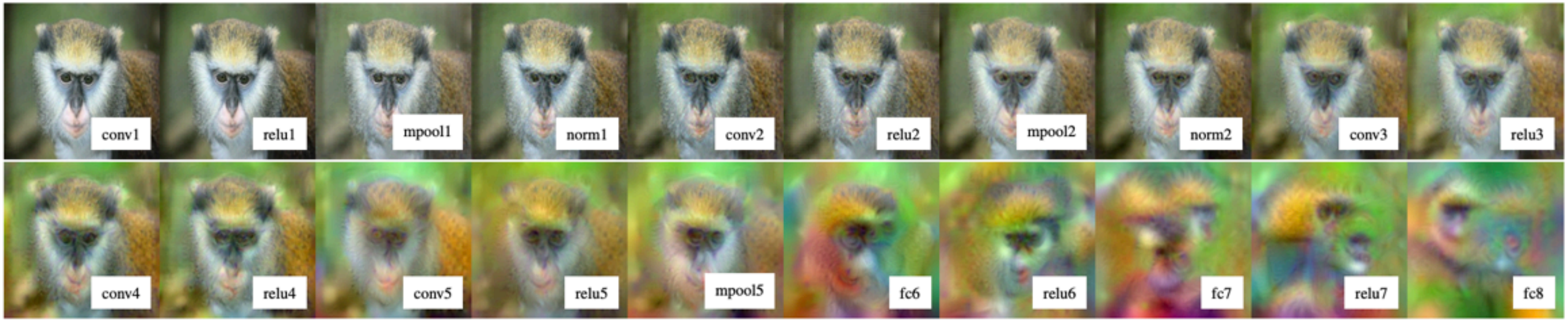
Image Reconstruction using CNN Feature Loss





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

feature loss could be at different layers



It means different layer's feature activation captures knowledge of the data

Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2015)

Neural Style Transfer

Content Image

Style Image

Generated image

https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f

Neural Style Transfer

Style
The Starry Night,
Vincent van Gogh,
1889

Style
The Muse,
Pablo Picasso,
1935

Neural Style Transfer: Live Demo

- https://reiinakano.com/arbitrary-image-stylization-tfjs/

How to Represent Content and Style?

Content Image

Style Image

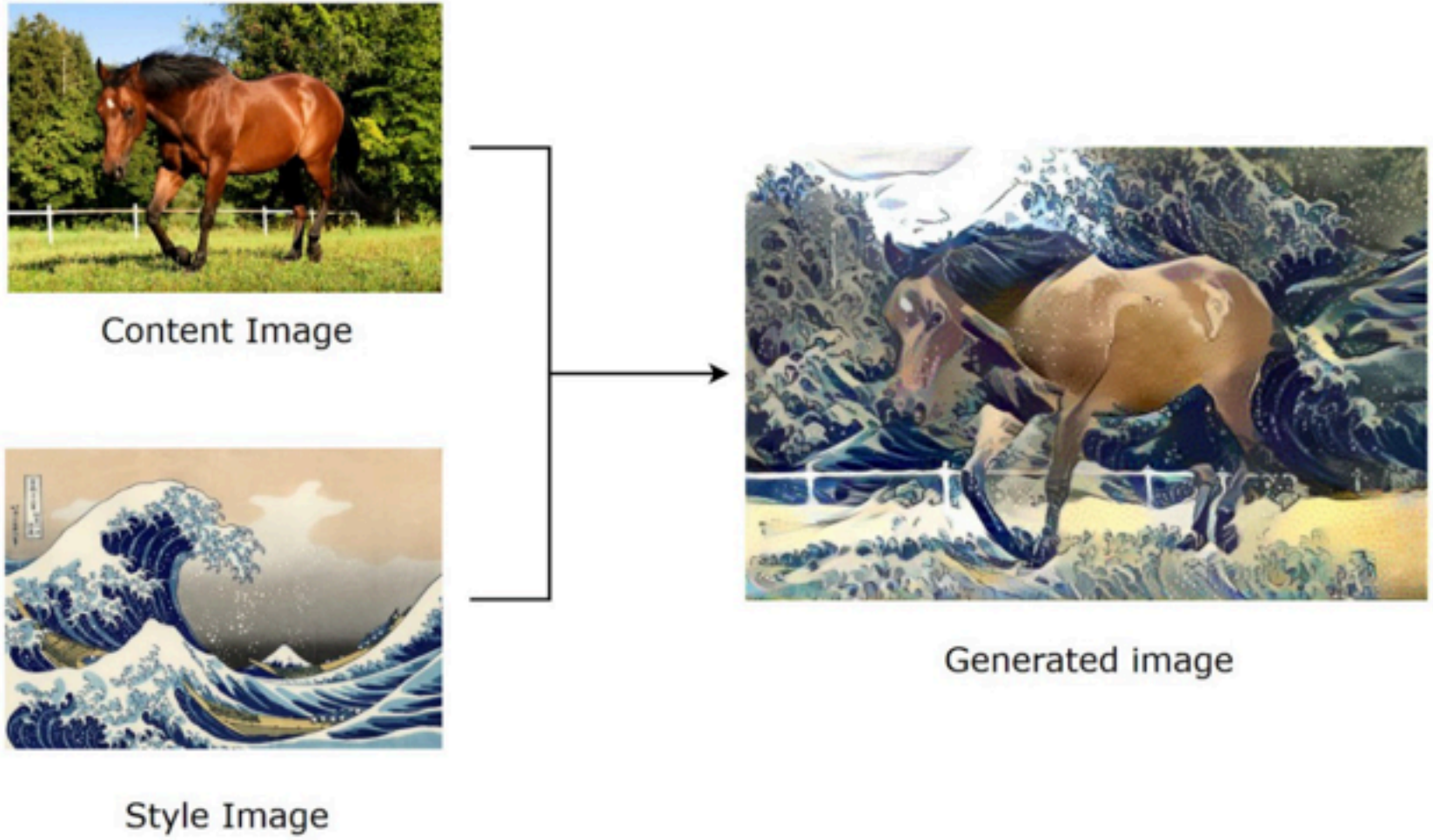
Generated image

https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f

How to Represent Content and Style?

Feature Loss

$$\|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$



https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f

How to Represent Content and Style?

Feature Loss

$$\|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

Style Loss???

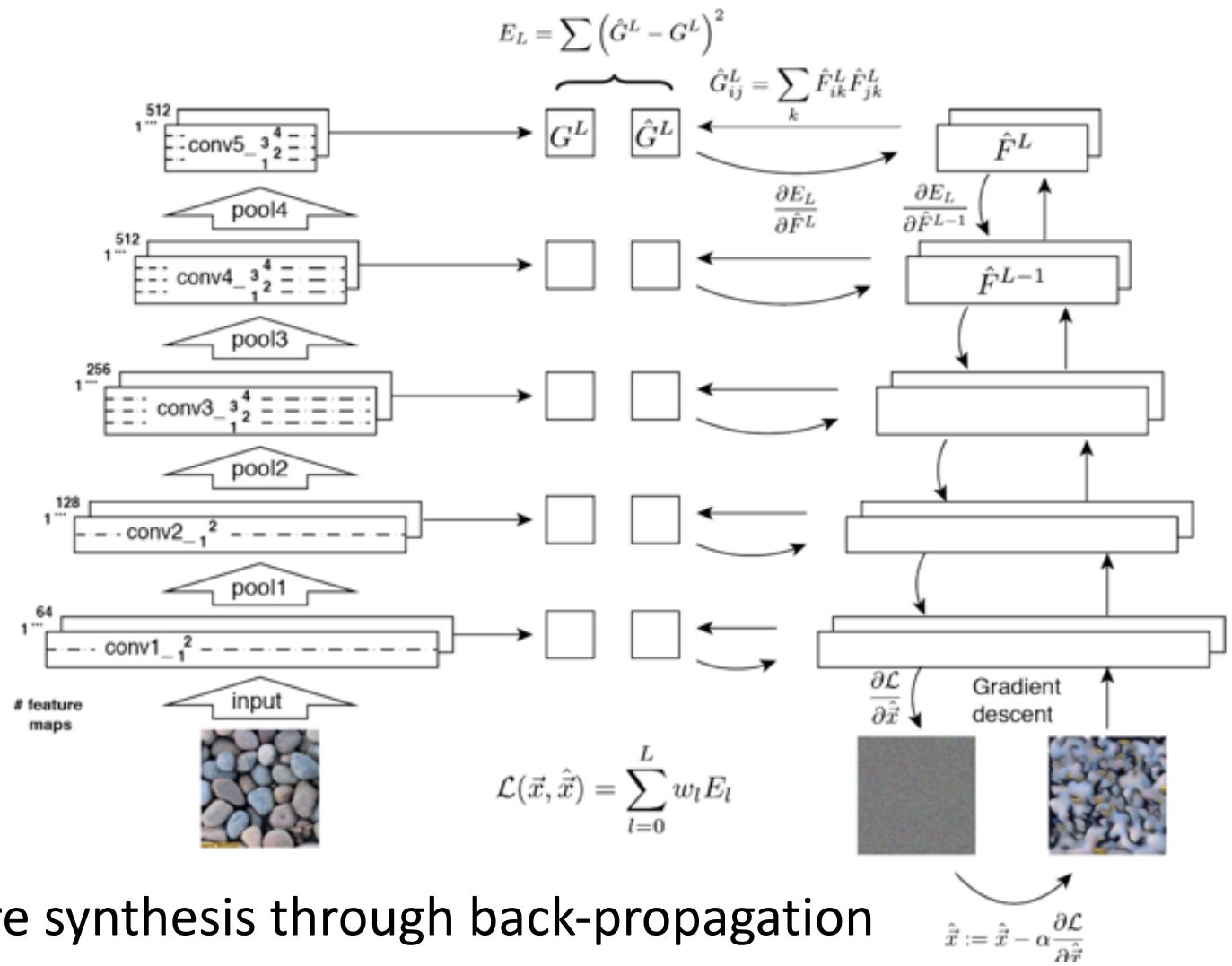
Content Image

Style Image

Generated image

Representing Style as Texture

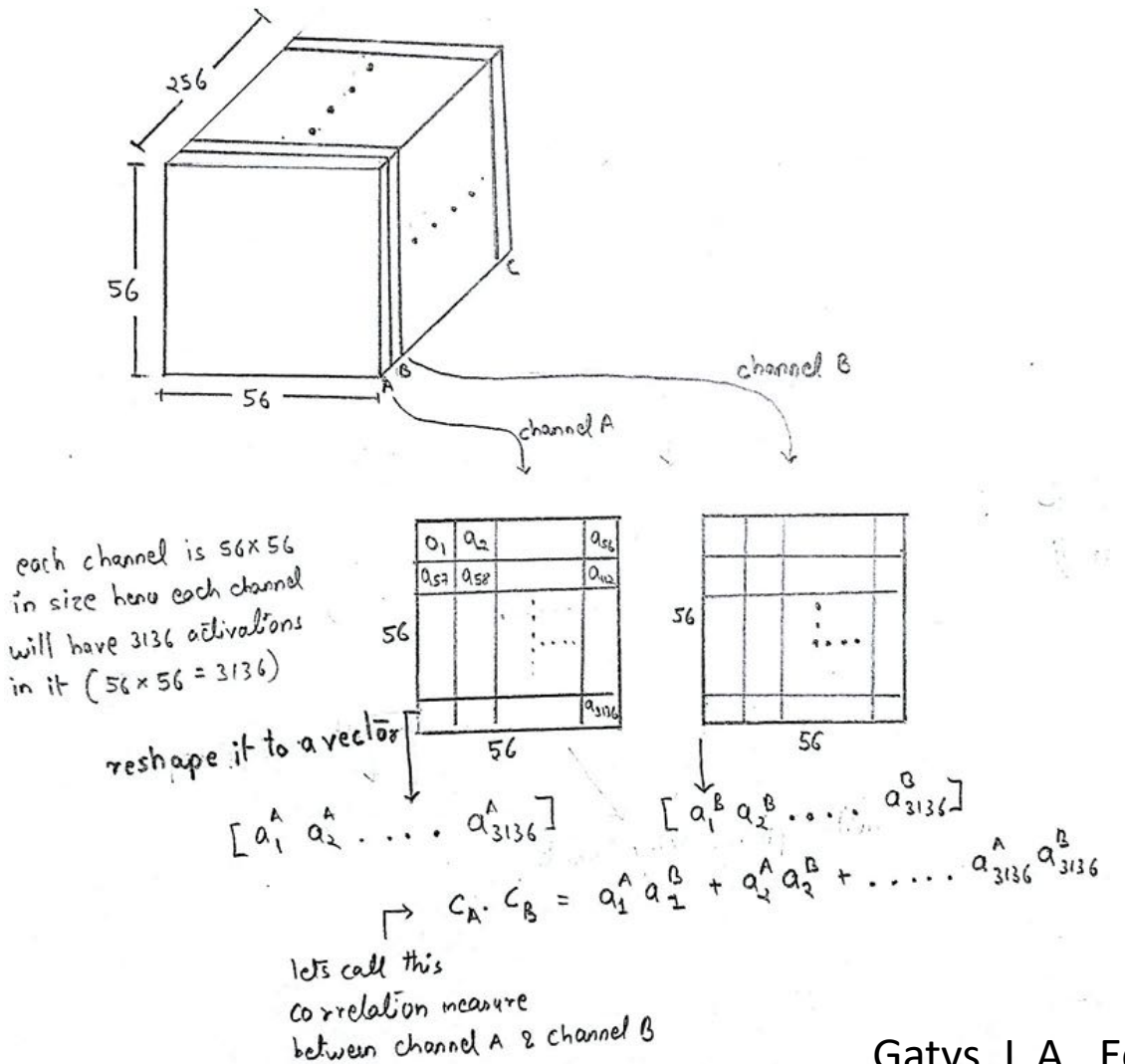
- Texture information is preserved as the correlation of features in CNNs

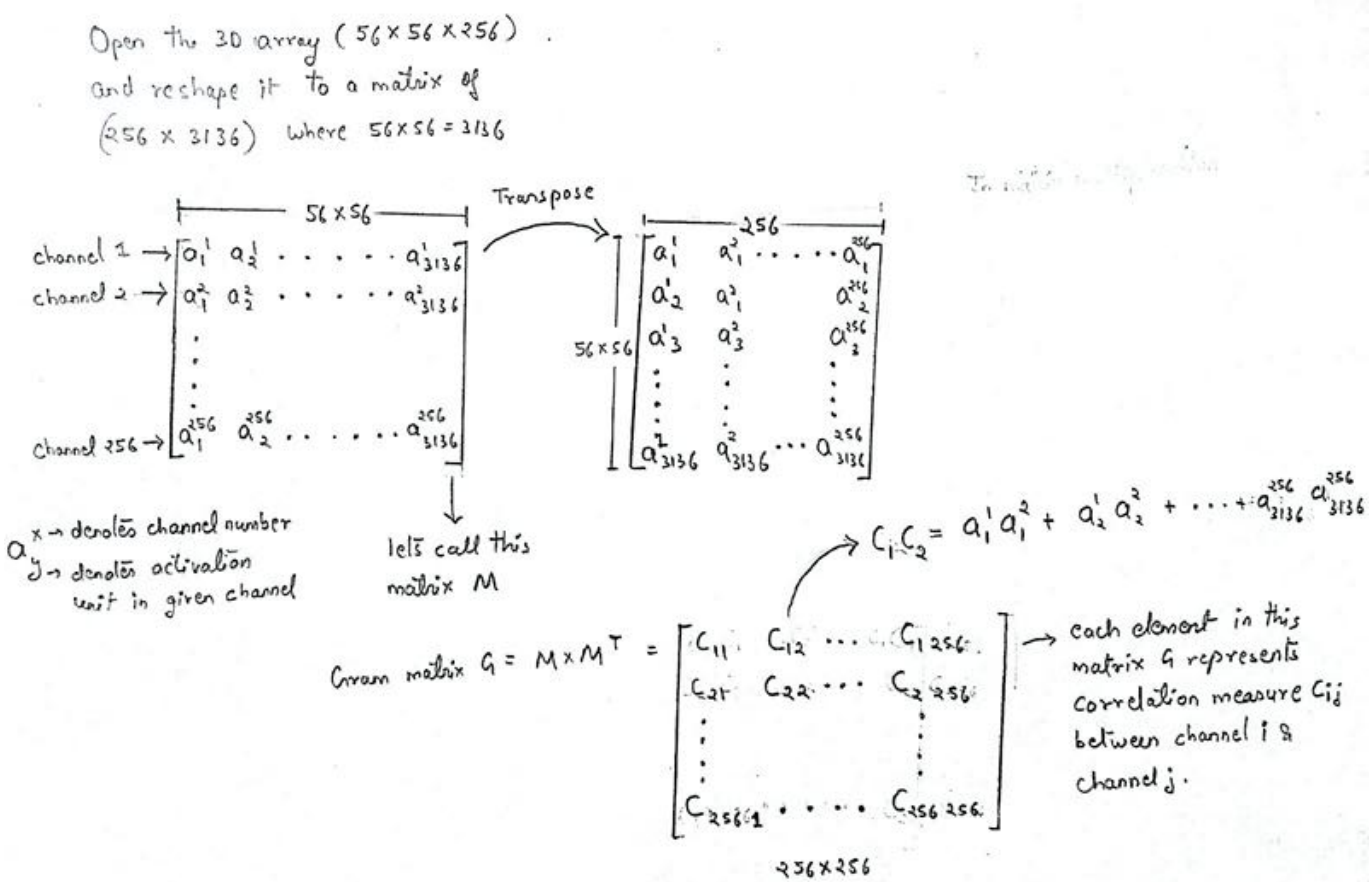


Texture synthesis through back-propagation

L. A. Gatys, A. S. Ecker, and M. Bethge. Texture Synthesis Using Convolutional Neural Networks. NeurIPS, 2015.

Representing Style with Gram Matrix

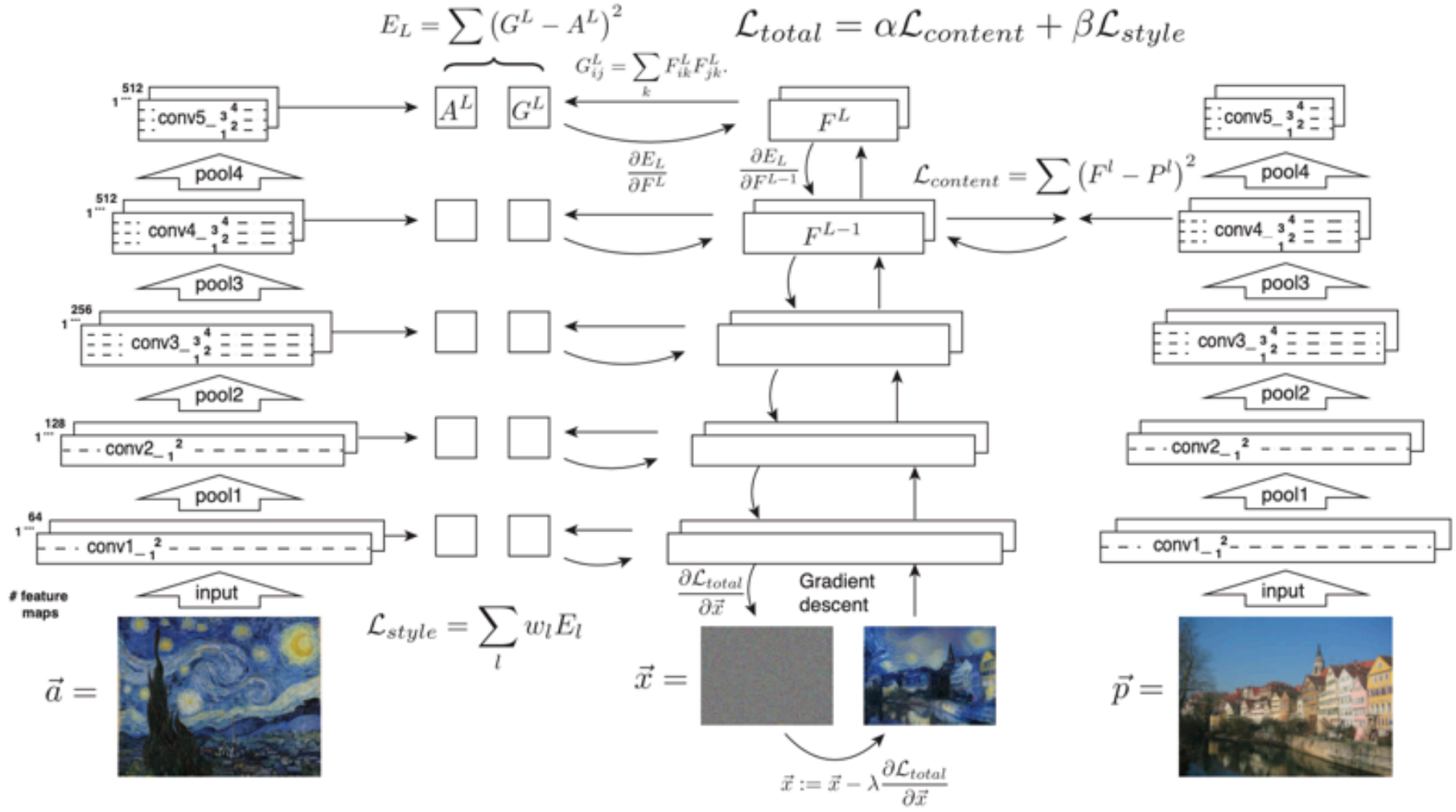




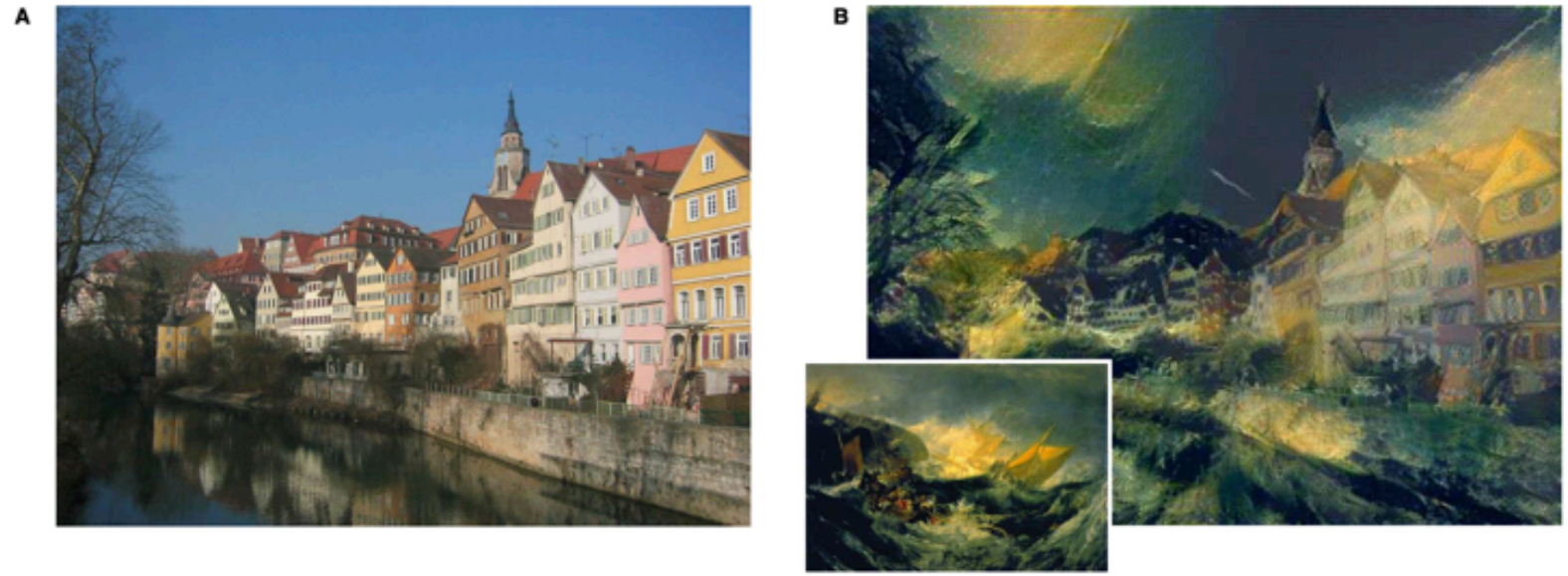
Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style

https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f

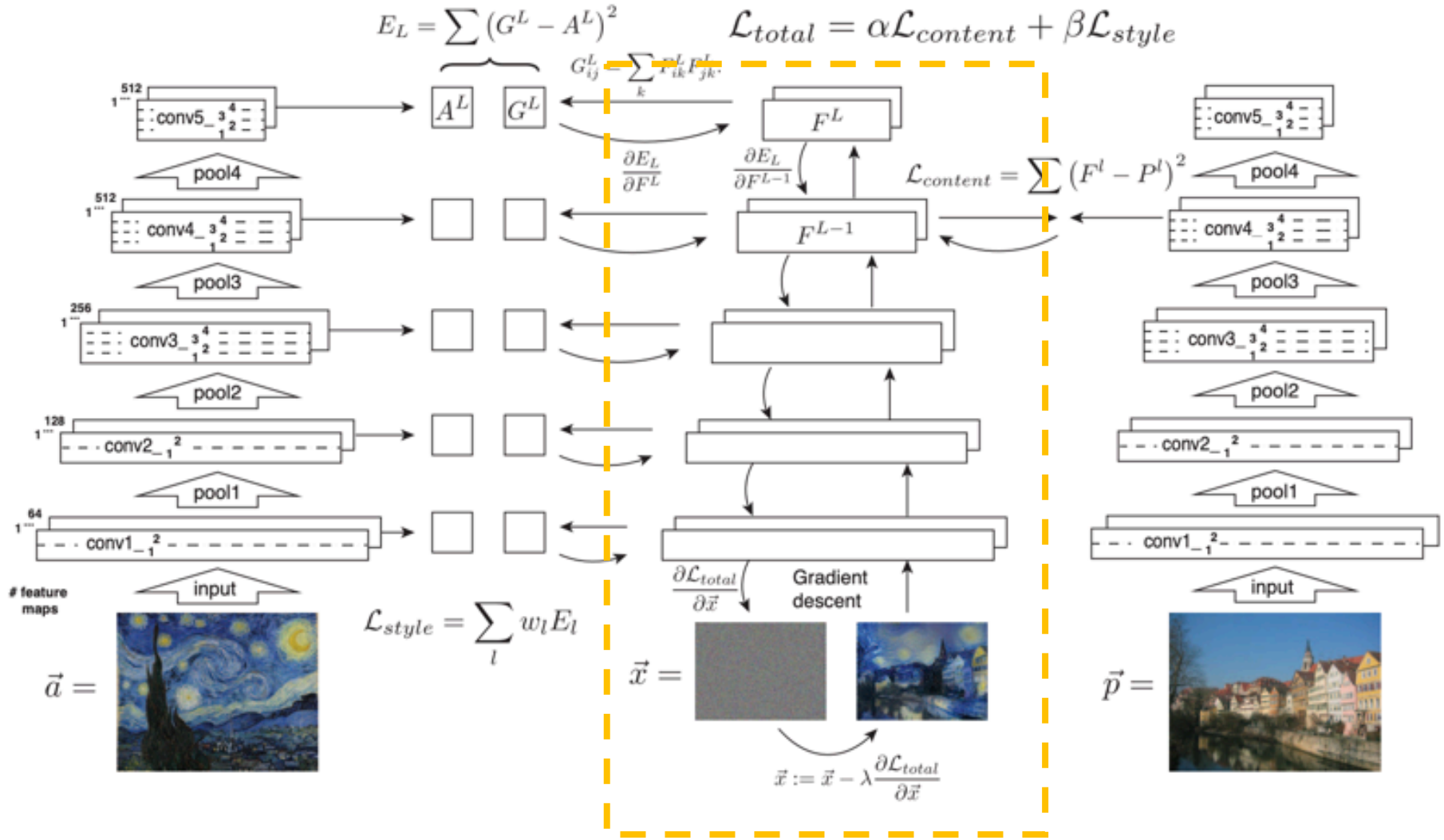
Transferring Style through back-propagation on both content and style losses



Gatys et al. Image Style Transfer Using Convolutional Neural Networks. CVPR'16

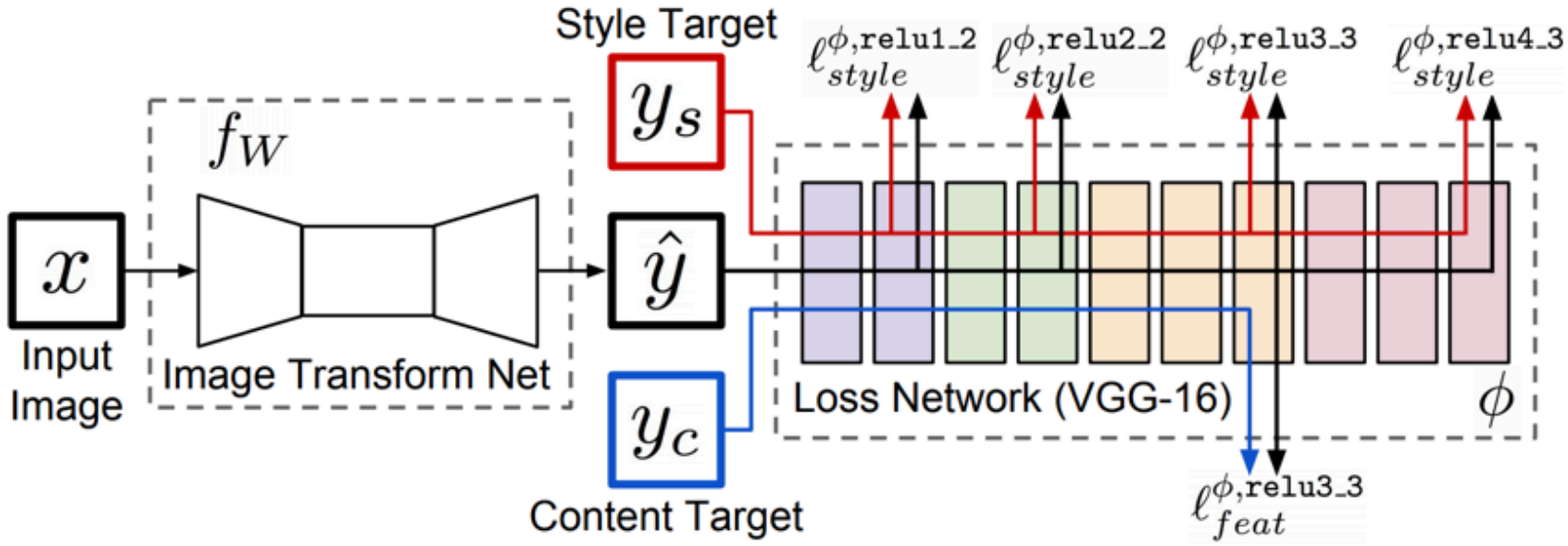

A
B
C
D

Drawback: back-propagation is slow and computationally expensive



Train an Image Transform Network

$$\ell^{\phi,j}_{style}(\hat{y}, y) = \|G^{\phi}_{j}(\hat{y}) - G^{\phi}_{j}(y)\|^2_F.$$



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

Perceptual loss:

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

Total variation regularizer:

$$V(y) = \sum_n |y_{n+1} - y_n|.$$

Johnson et al. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV'16

Result of Style Transfer

Johnson et al. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV'16

Result of Super-Resolution

Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [11]	Ours (ℓ_{feat})
This Image	21.69 / 0.5840	21.66 / 0.5881	22.53 / 0.6524	21.04 / 0.6116
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

What are the metrics PSNR/SSIM?

Johnson et al. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV'16

Metrics for Image Reconstruction

- Pairwise similarity:
  - PSNR
  - MS-SSIM
- Distribution similarity (commonly used in GANs):
  - FID

Metric: PSNR (Peak Signal-to-Noise Ratio)

- **Peak signal-to-noise ratio (PSNR)** is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.
- Commonly used to measure the quality of lossy compression reconstruction, an approximation to human perception (30dB to 50 dB)

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

The PSNR (in dB) is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \\ &= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \\ &= 20 \cdot \log_{10} (MAX_I) - 10 \cdot \log_{10} (MSE) \end{aligned}$$

Q=30, PSNR 36.81dB

Q=10, PSNR 31.45dB

Metric: MS-SSIM (Multi-scale structural similarity index measure)

- Perceptual image quality assessment

SSIM:

luminance: $$l(\mathbf{x},\mathbf{y}) = \frac{2\,\mu_x\,\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \quad (1)$$

contrast: $$c(\mathbf{x},\mathbf{y}) = \frac{2\,\sigma_x\,\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \quad (2)$$

structure: $$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\,\sigma_y + C_3}, \quad (3)$$

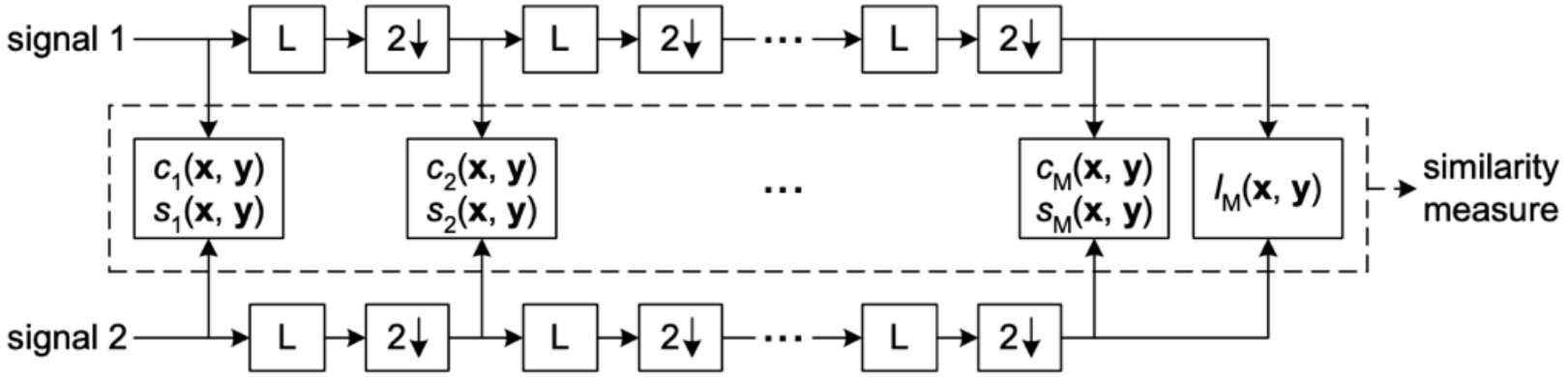
where C_1, C_2 and C_3 are small constants given by

$$C_1 = (K_1\,L)^2\,,\ C_2 = (K_2\,L)^2 \text{ and } C_3 = C_2/2, \quad (4)$$

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma}\,, \quad (5)$$

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\,\mu_x\,\mu_y + C_1)\,(2\,\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)\,(\sigma_x^2 + \sigma_y^2 + C_2)}\,, \quad (6)$$

Multi-scale SSIM

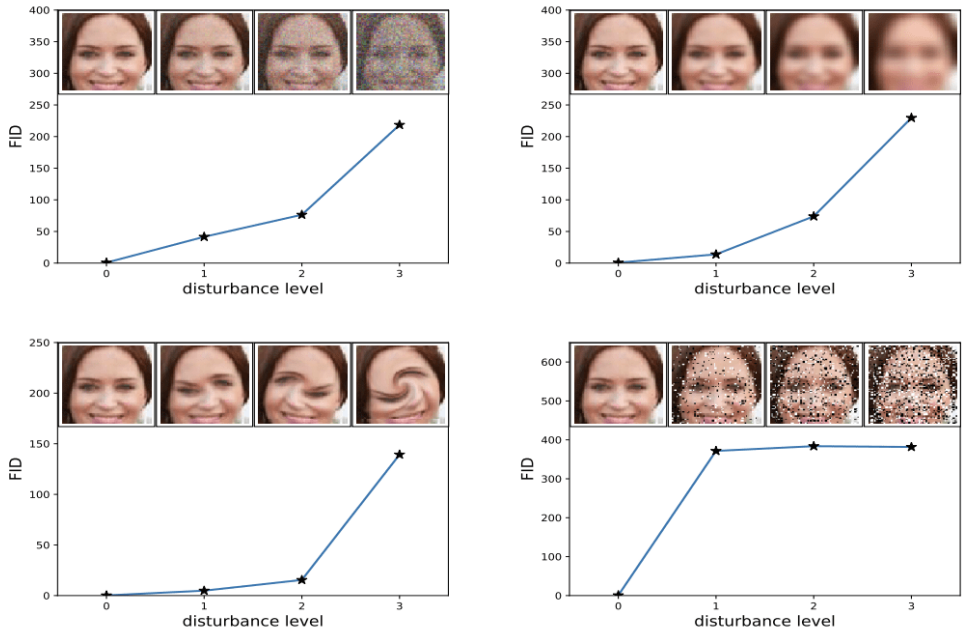


https://www.cns.nyu.edu/pub/eero/wang03b.pdf

Metric: Frechet Inception Distance (FID)

- FID is the Wasserstein distance between two multidimensional Gaussian distributions $N(u, \Sigma)$ and $N(u_w, \Sigma_w)$
- Rather than modeling the image pixels, it models the deep features from CNN called Inception v3
- Commonly used in GAN evaluation

$$\text{FID} = |\mu - \mu_w|^2 + \text{tr}(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{1/2}).$$



https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/

https://en.wikipedia.org/wiki/Fr%C3%A9chet_inception_distance

Result of Super-Resolution

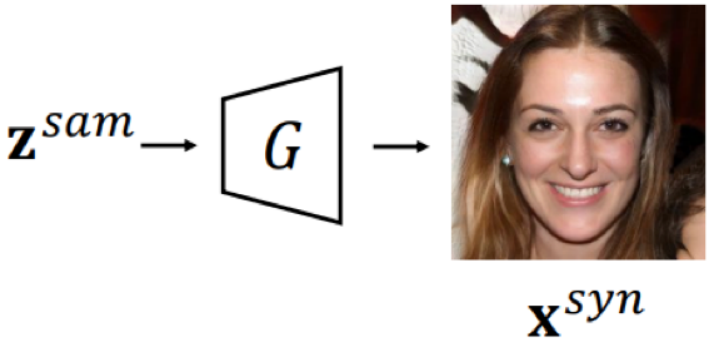
Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [11]	Ours (ℓ_{feat})
This Image	21.69 / 0.5840	21.66 / 0.5881	22.53 / 0.6524	21.04 / 0.6116
Set14 mean	25.99 / 0.7301	25.75 / 0.6994	27.49 / 0.7503	24.99 / 0.6731
BSD100 mean	25.96 / 0.682	25.91 / 0.6680	26.90 / 0.7101	24.95 / 63.17

PSNR/SSIM

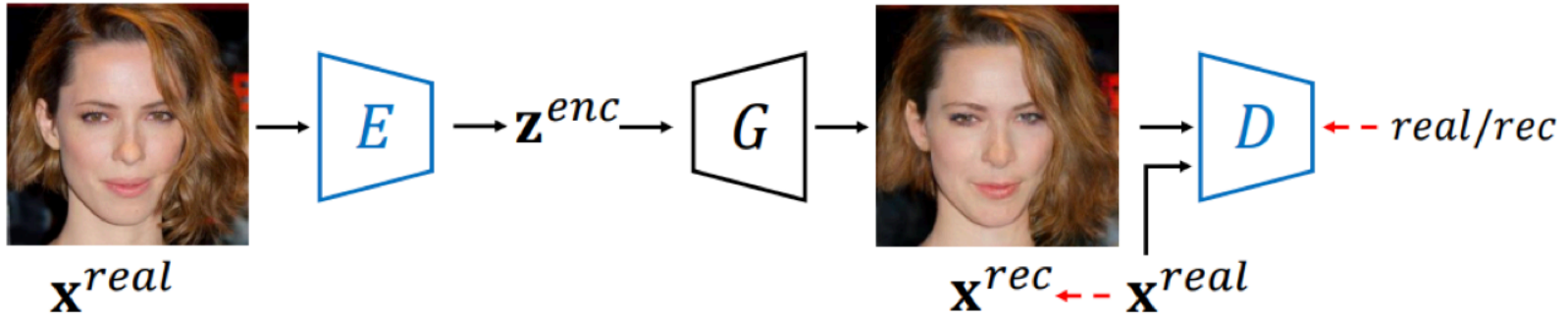
Johnson et al. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV'16

Example: In-Domain GAN inversion

GAN image generation



Train an encoder for GAN's generator

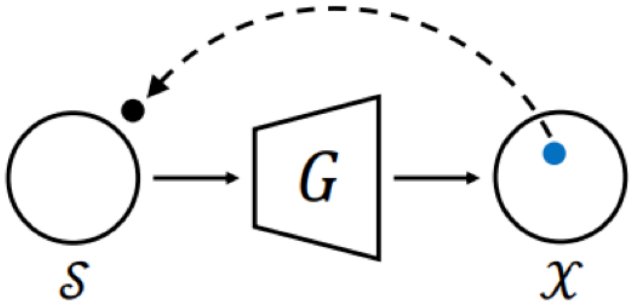


$$\min_{\Theta_E} \mathcal{L}_E = ||\mathbf{x}^{real} - G(E(\mathbf{x}^{real}))||_2 + \lambda_{vgg}||F(\mathbf{x}^{real}) - F(G(E(\mathbf{x}^{real})))||_2 - \lambda_{adv} \mathop{\mathbb{E}}_{\mathbf{x}^{real} \sim P_{data}} [D(G(E(\mathbf{x}^{real})))],$$

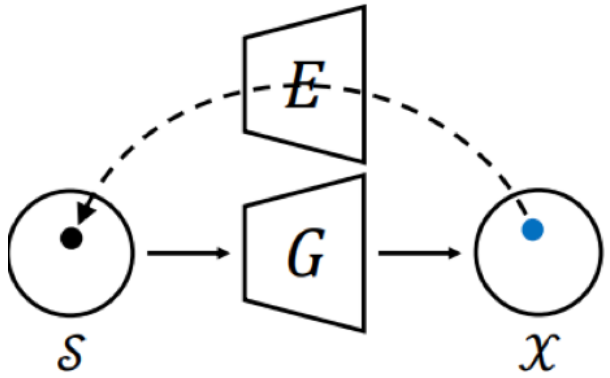
Zhu et al. In-Domain GAN Inversion for Real Image Editing. ECCV'21

Example: In-Domain GAN inversion

GAN inversion: optimize **z** such that **x** = G(**z**)



$$\mathbf{z}^{inv} = \arg\min_{\mathbf{z}} \; ||\mathbf{x} - G(\mathbf{z})||_2 \; + \lambda_{vgg}||F(\mathbf{x}) - F(G(\mathbf{z}))||_2$$

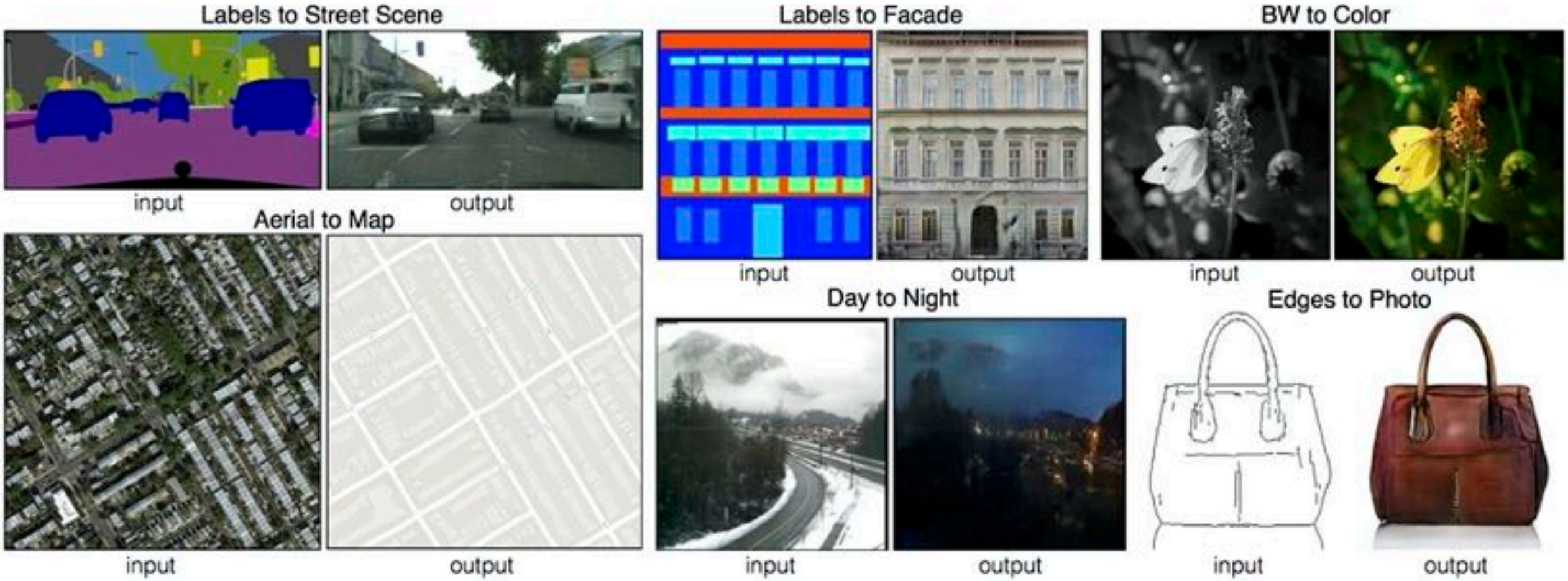


$$\mathbf{z}^{inv} = \arg\min_{\mathbf{z}} \; ||\mathbf{x} - G(\mathbf{z})||_2 \; + \lambda_{vgg}||F(\mathbf{x}) - F(G(\mathbf{z}))||_2 + \lambda_{dom}||\mathbf{z} - E(G(\mathbf{z}))||_2,$$

Zhu et al. In-Domain GAN Inversion for Real Image Editing. ECCV'21

Image to Image Translation with Conditional GAN

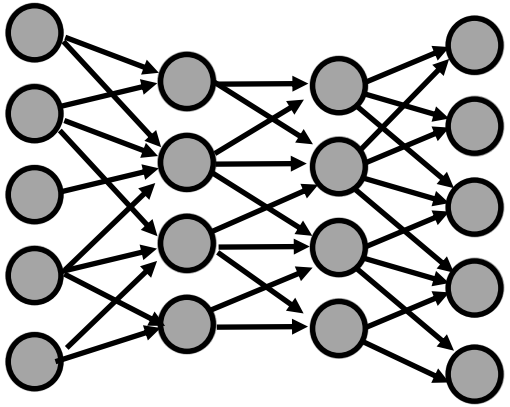
Image to Image Translation



Isola et al. Image-to-Image Translation with Conditional Adversarial Nets. CVPR'17

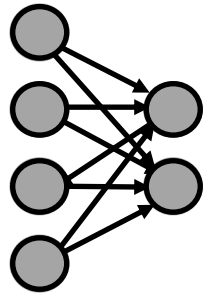
x

G



G(x)

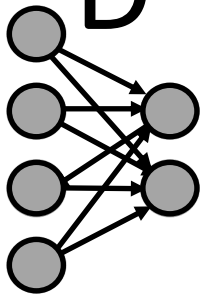
D



fake

y

D



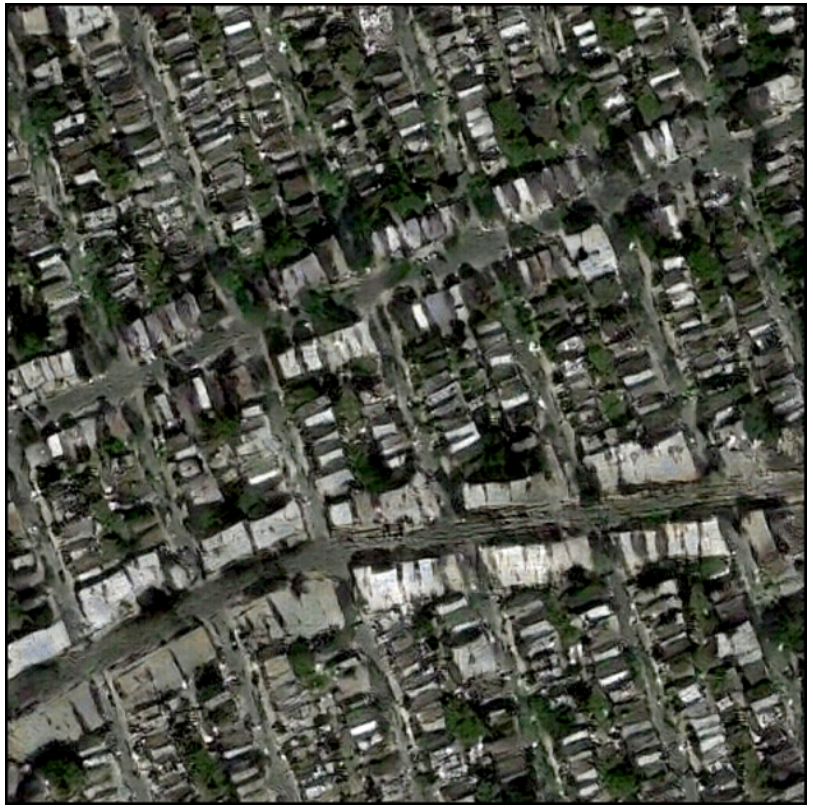
real

loss function:

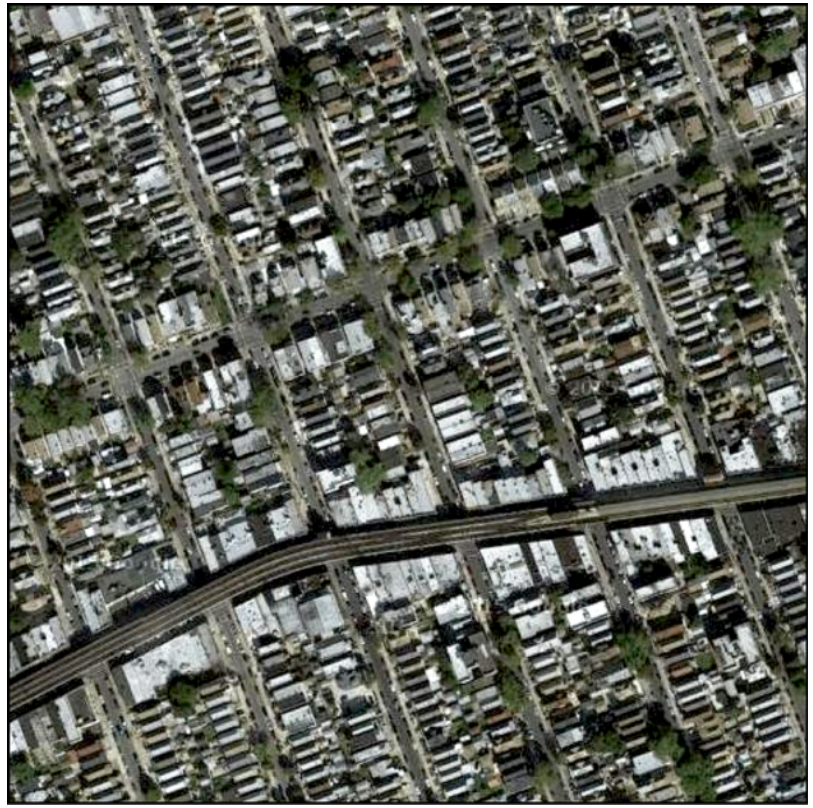
$$\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y})) \;\;]$$

Input

Output



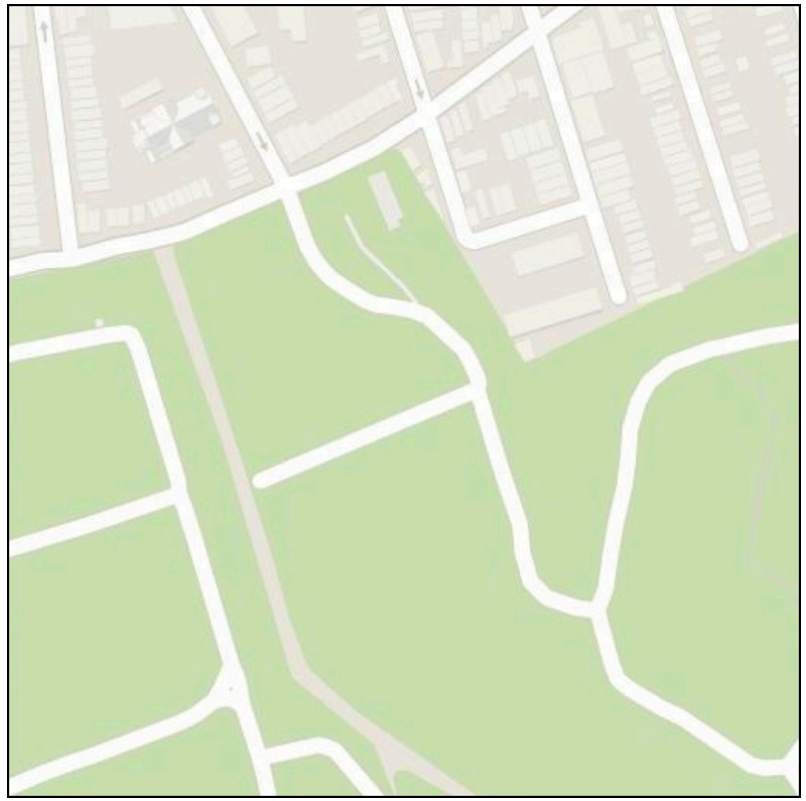
Ground-truth



data downloaded from the Google Map

Isola et al. Image-to-Image Translation with Conditional Adversarial Nets. CVPR'17

Input

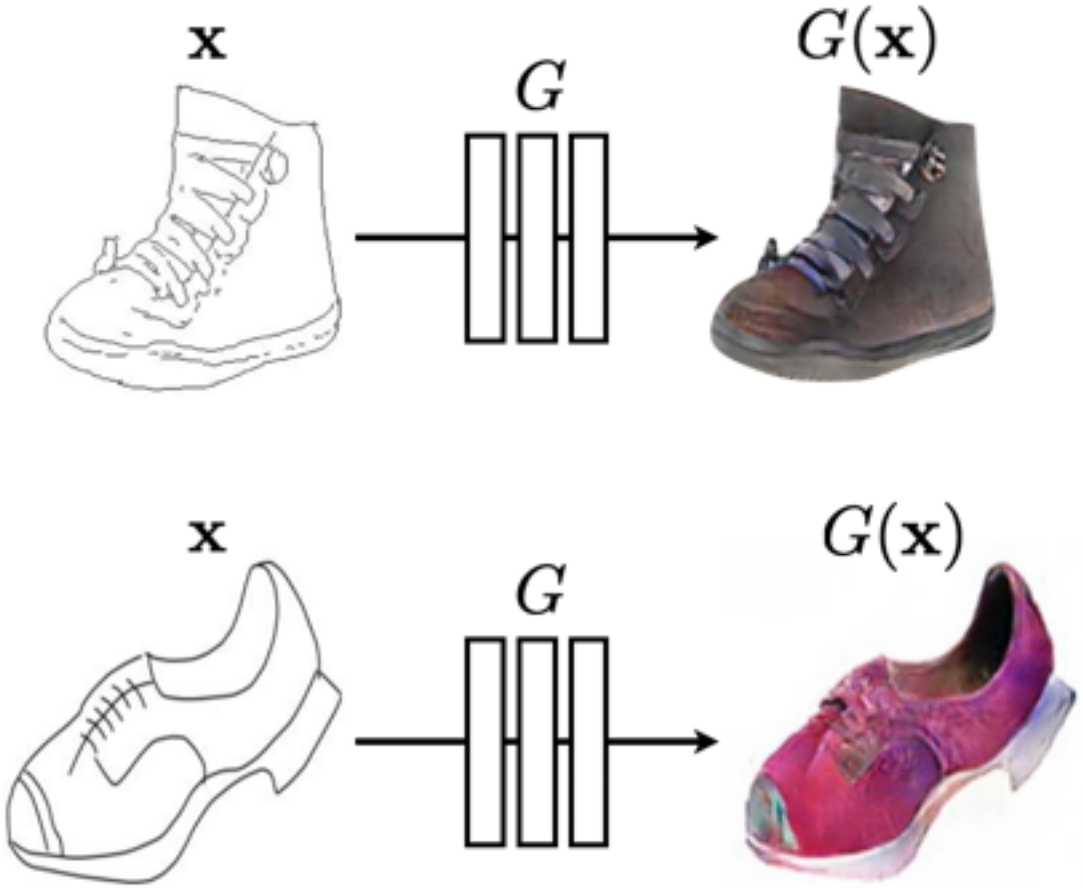


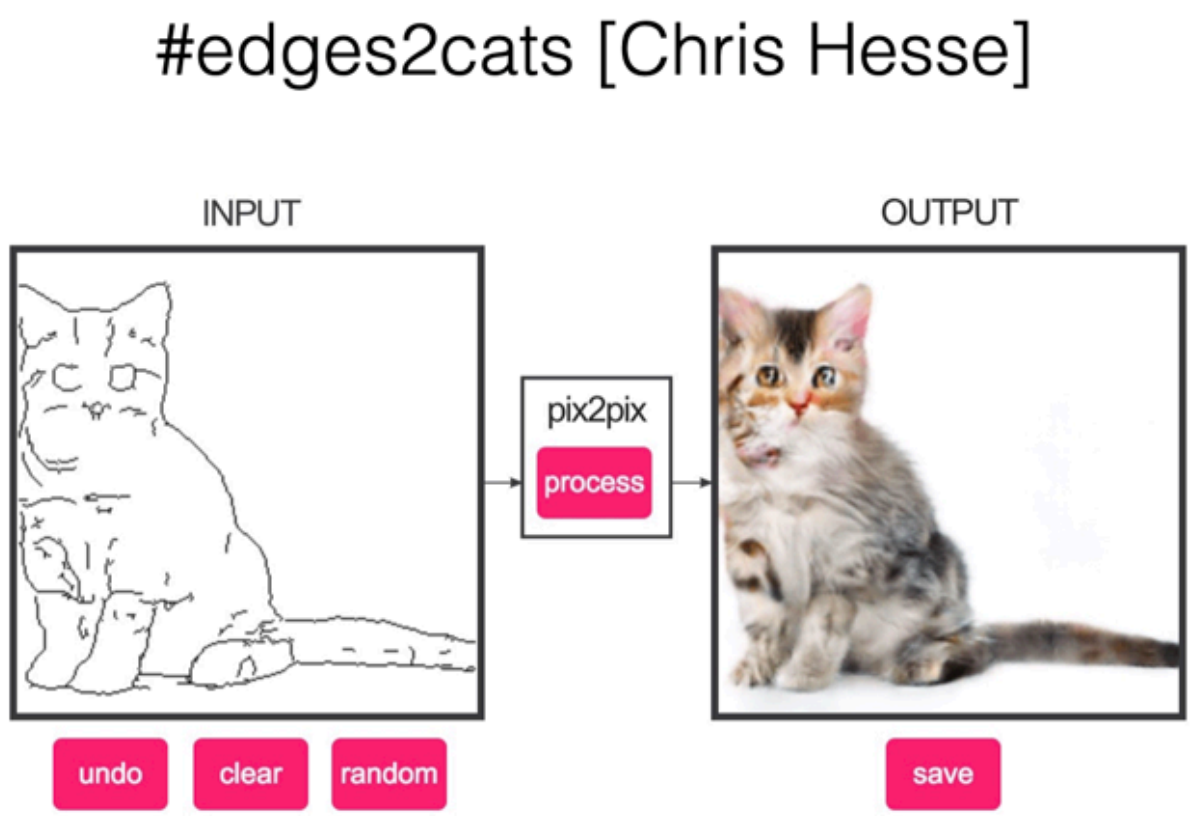
L1 loss

GAN loss

More structured loss

Amazing applications based on Pix2Pix





https://phillipi.github.io/pix2pix/

Amazing applications based on Pix2Pix

#edges2cats

Christopher Hesse trained our model on converting edge maps to photos of cats, and included this in his interactive demo. Apparently, this is what the Internet wanted most, and #edges2cats briefly went viral. The above cats were designed by Vitaly Vidmirov (@vvid).

Alternative Face

Mario Klingemann used our code to translate the appearance of French singer Francoise Hardy onto Kellyanne Conway's infamous "alternative facts" interview. Interesting articles about it can be read here and here.

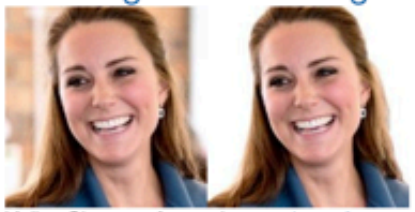
Person-to-Person

Brannon Dorsey recorded himself mimicking frames from a video of Ray Kurzweil giving a talk. He then used this data to train a Dorsey→Kurzweil translator, allowing him to become a kind of puppeter in control of Kurzweil's appearance.

Interactive Anime

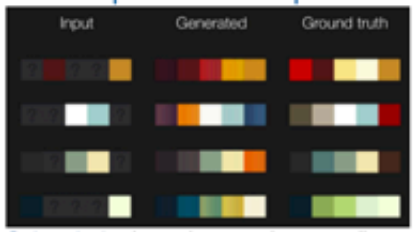
Bertrand Gondouin trained our method to translate sketches→Pokemon, resulting in an interactive drawing tool.

Background masking



Kaihu Chen performed a number of interesting experiments using our method, including getting it to mask out the background of a portrait as shown above.

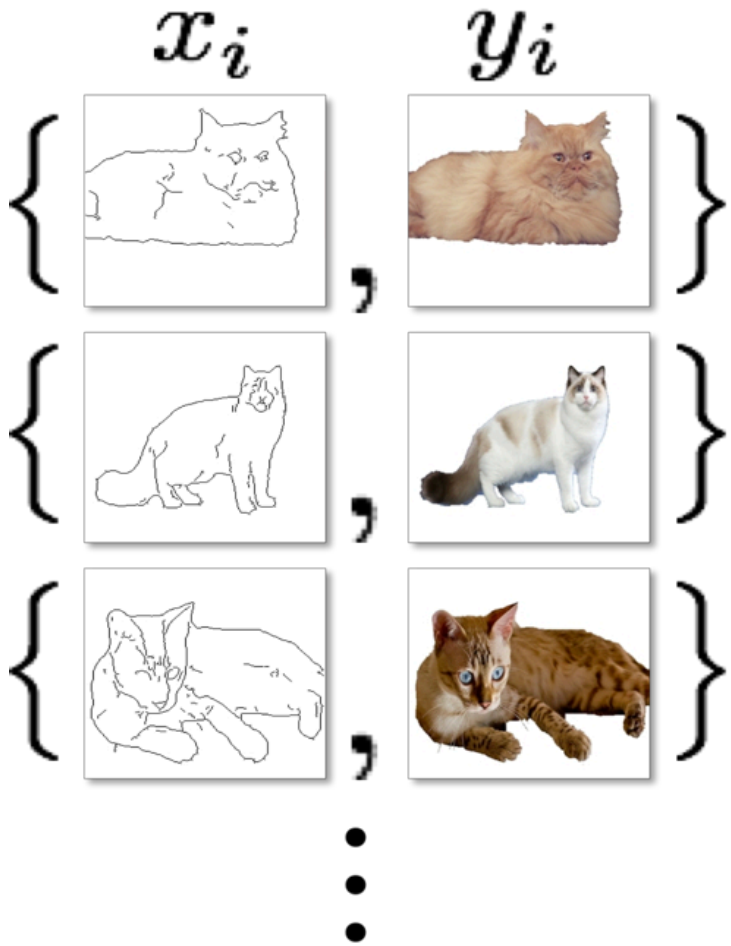
Color palette completion



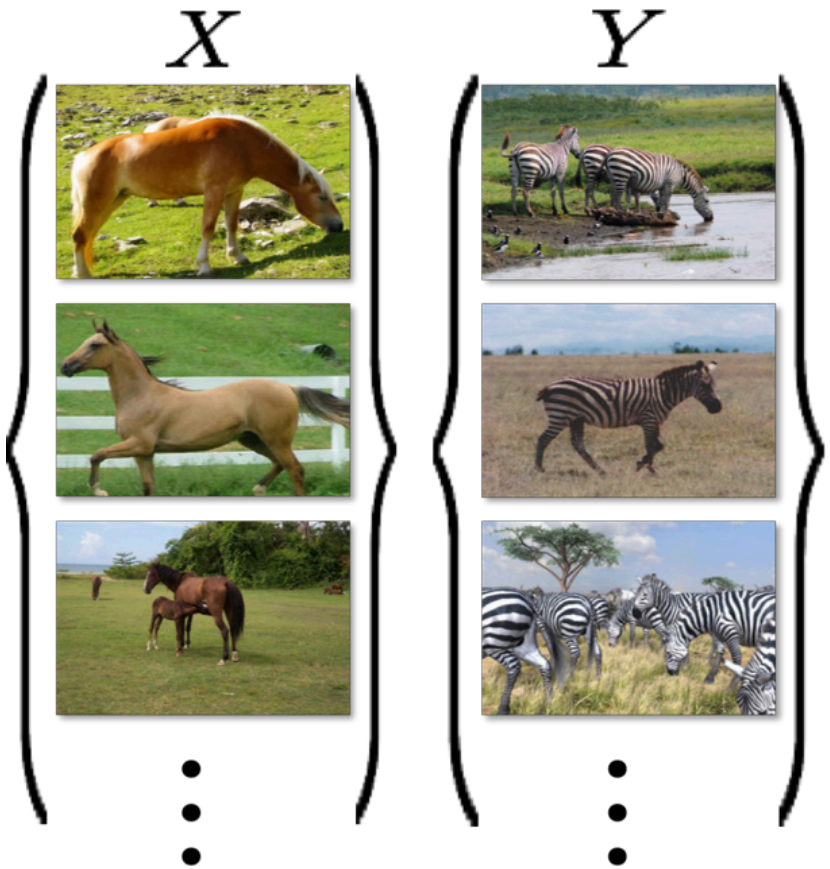
Colormind adapted our code to predict a complete 5-color palette given a subset of the palette as input. This application stretches the definition of what counts as "image-to-image translation" in an exciting way: if you can visualize your input/output data as images, then image-to-image methods are applicable! (not that this is necessarily the best choice of representation, just one to think about.)

https://phillipi.github.io/pix2pix/

Paired data



Unpaired data

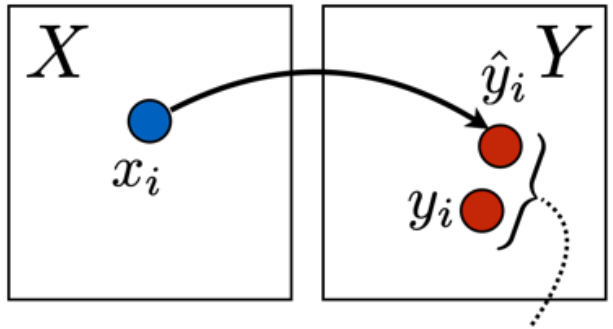


slides from Phillip Isola

Paired translation

Training data

Objective



regression error

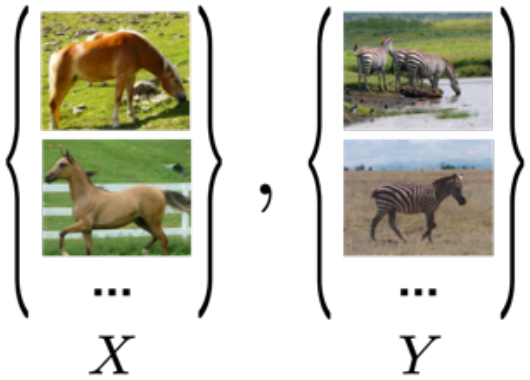
Input

Result

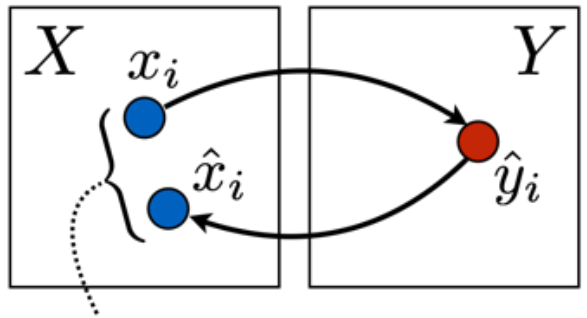
["pix2pix", Isola, Zhu, Zhou, Efros, 2017]

Unpaired translation

Training data



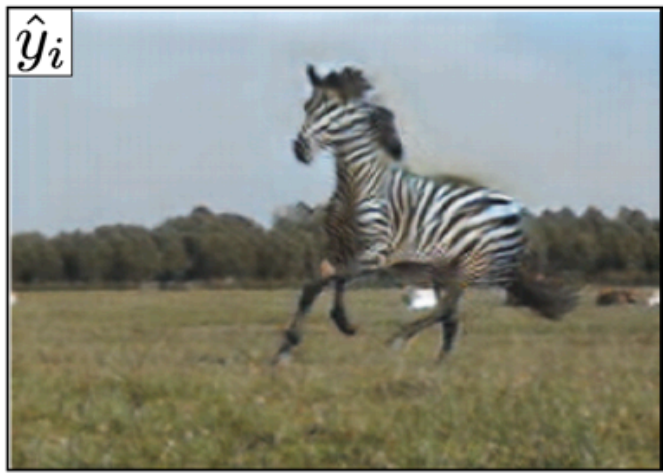
Objective



cycle-consistency error

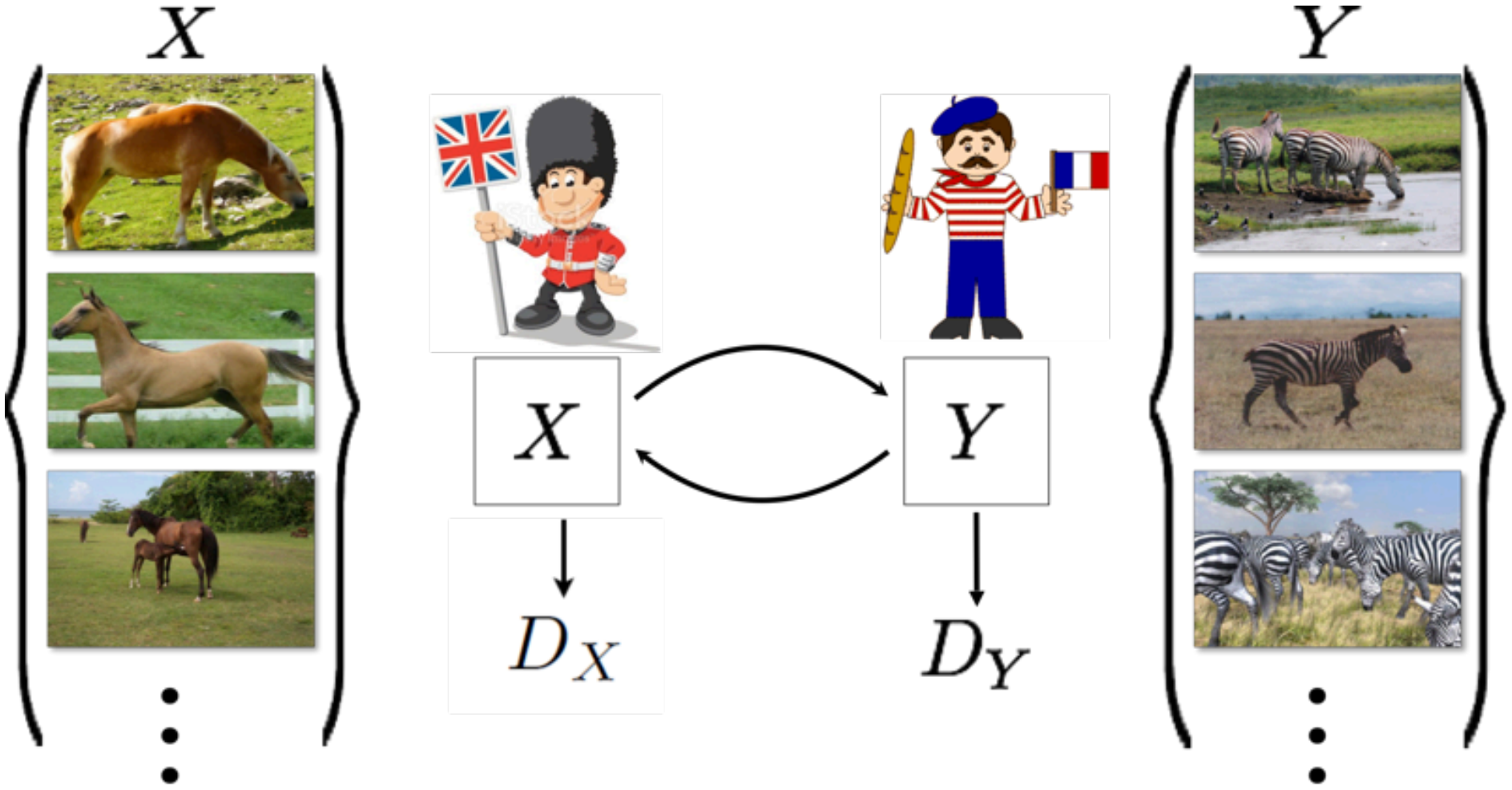
Input

Result



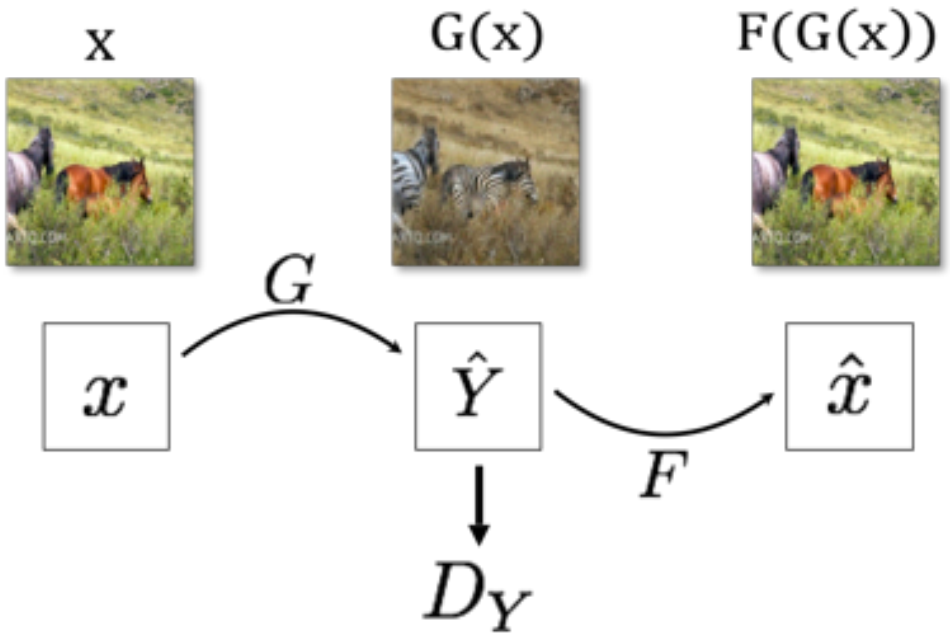
["CycleGAN", Zhu*, Park*, Isola, Efros, 2017]

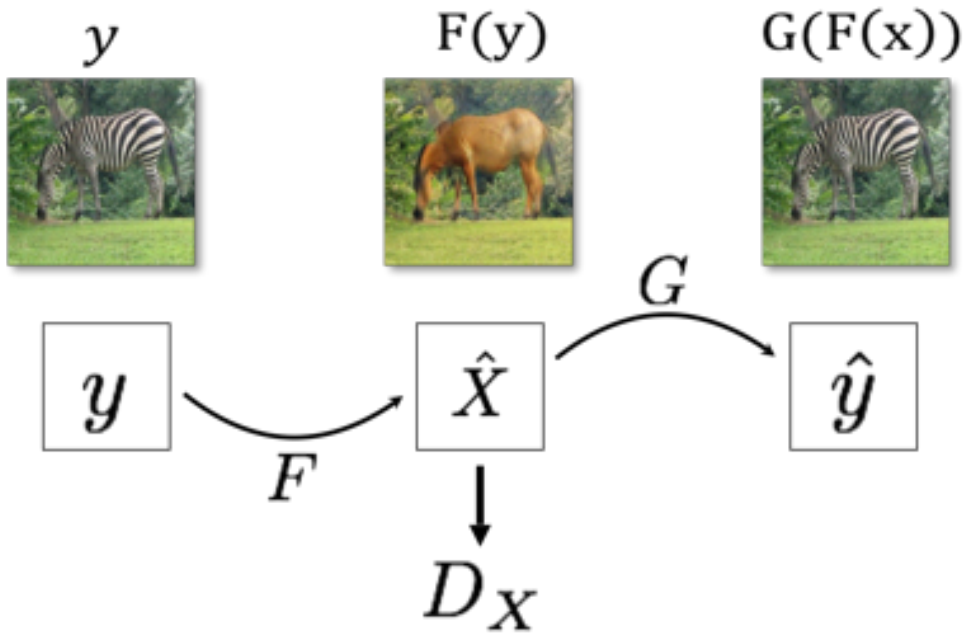
Cycle Consistency

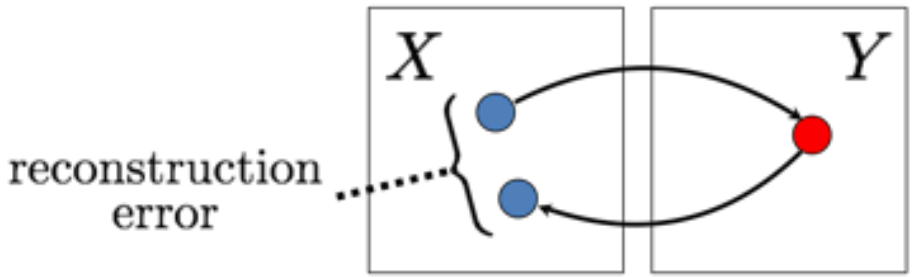


slides from Phillip Isola

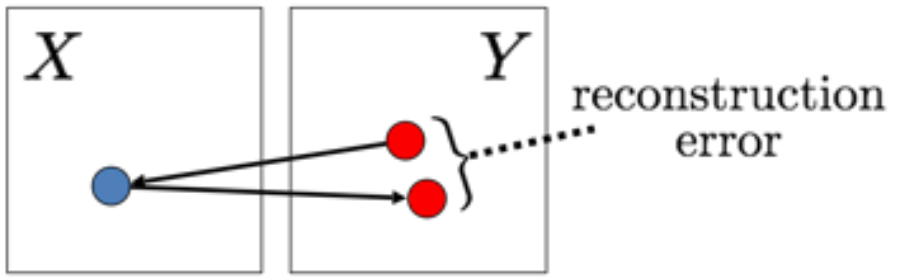
Cycle Consistency







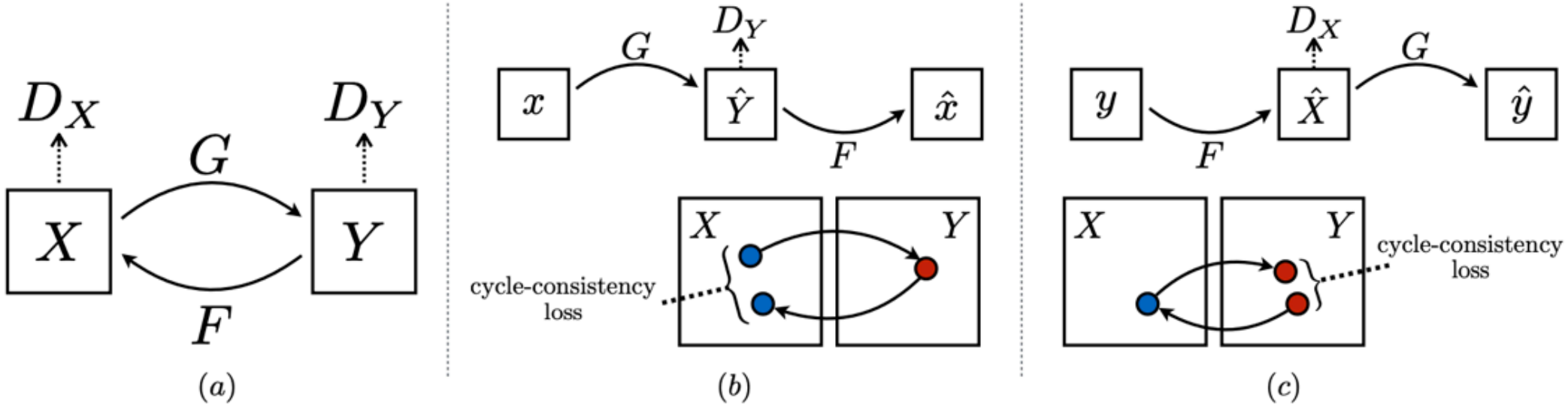
$$\|F(G(x)) - x\|_1$$



$$\|G(F(y)) - y\|_1$$

slides from Phillip Isola

Cycle Consistency



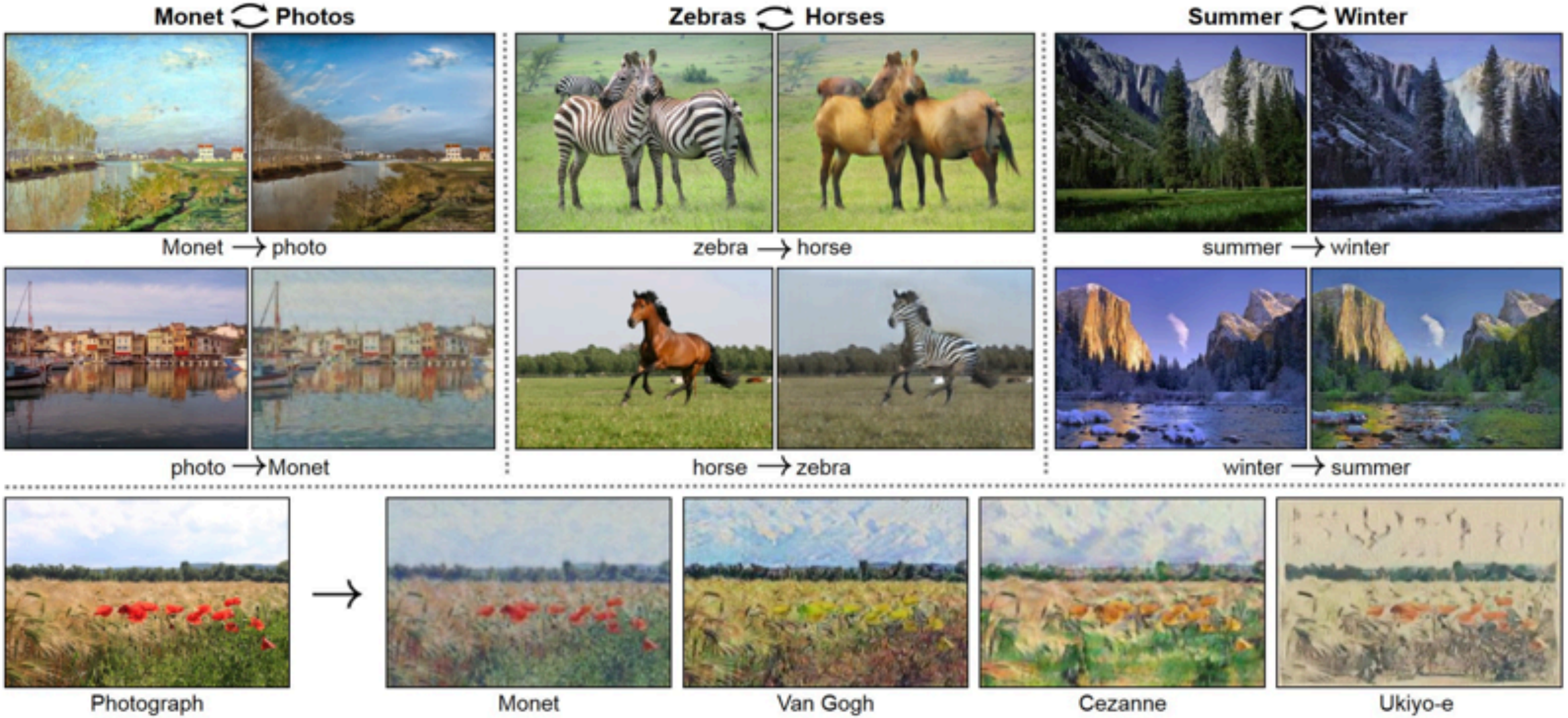
$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Our full objective is:

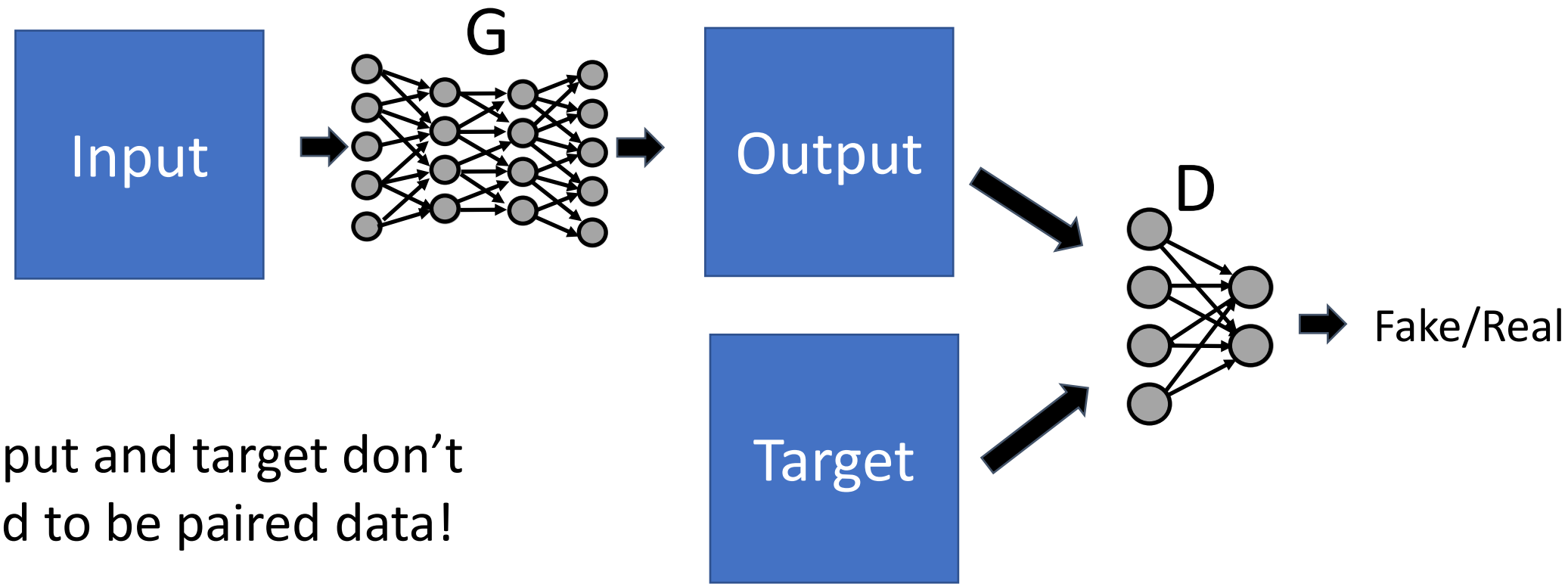
$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

Amazing applications based on CycleGAN



Check this out: https://junyanz.github.io/CycleGAN/

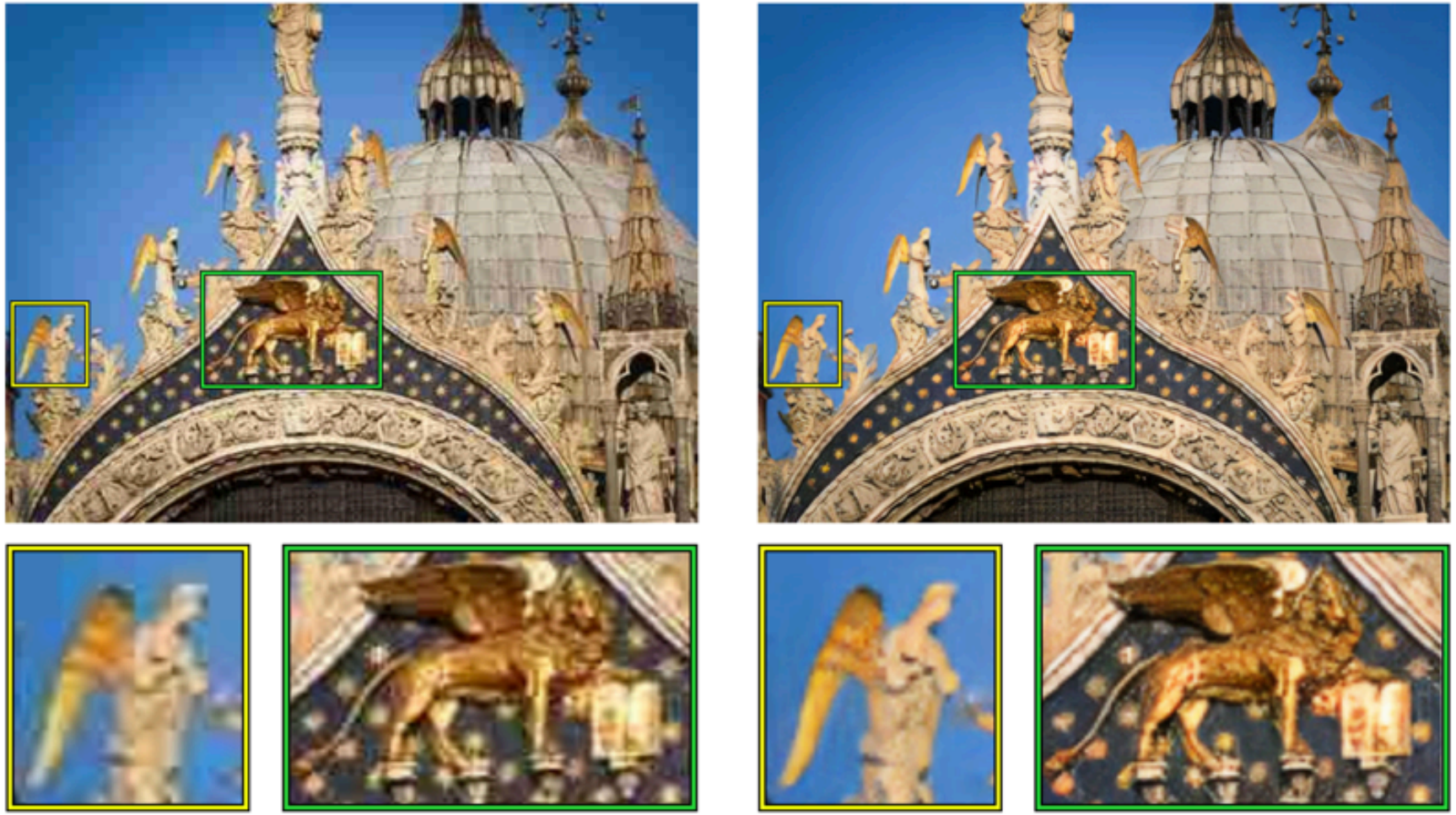
Wide Applications of Adversarial Loss



Output and target don't need to be paired data!

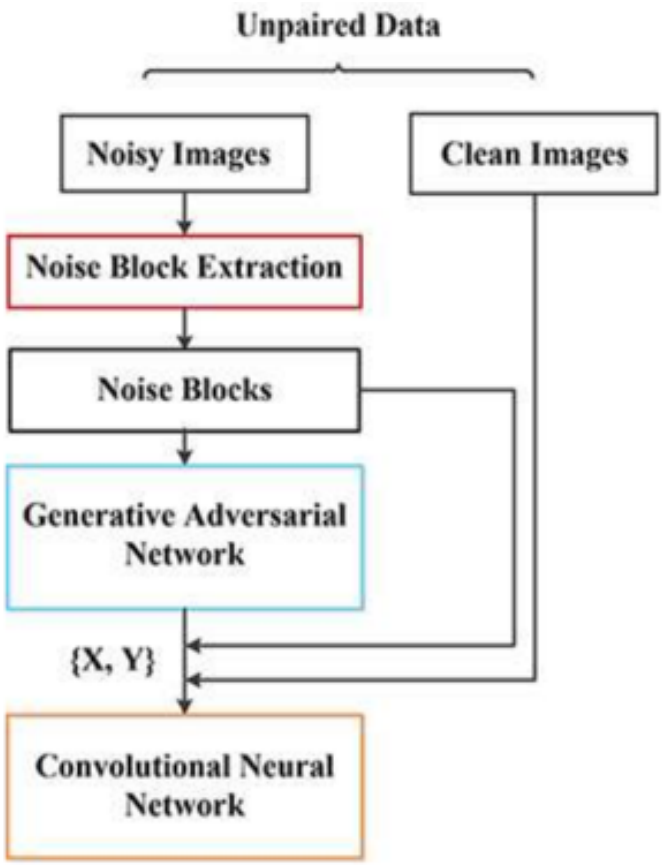
$$\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \;\; + \;\; \log(1 - D(\mathbf{y})) \;\;]$$

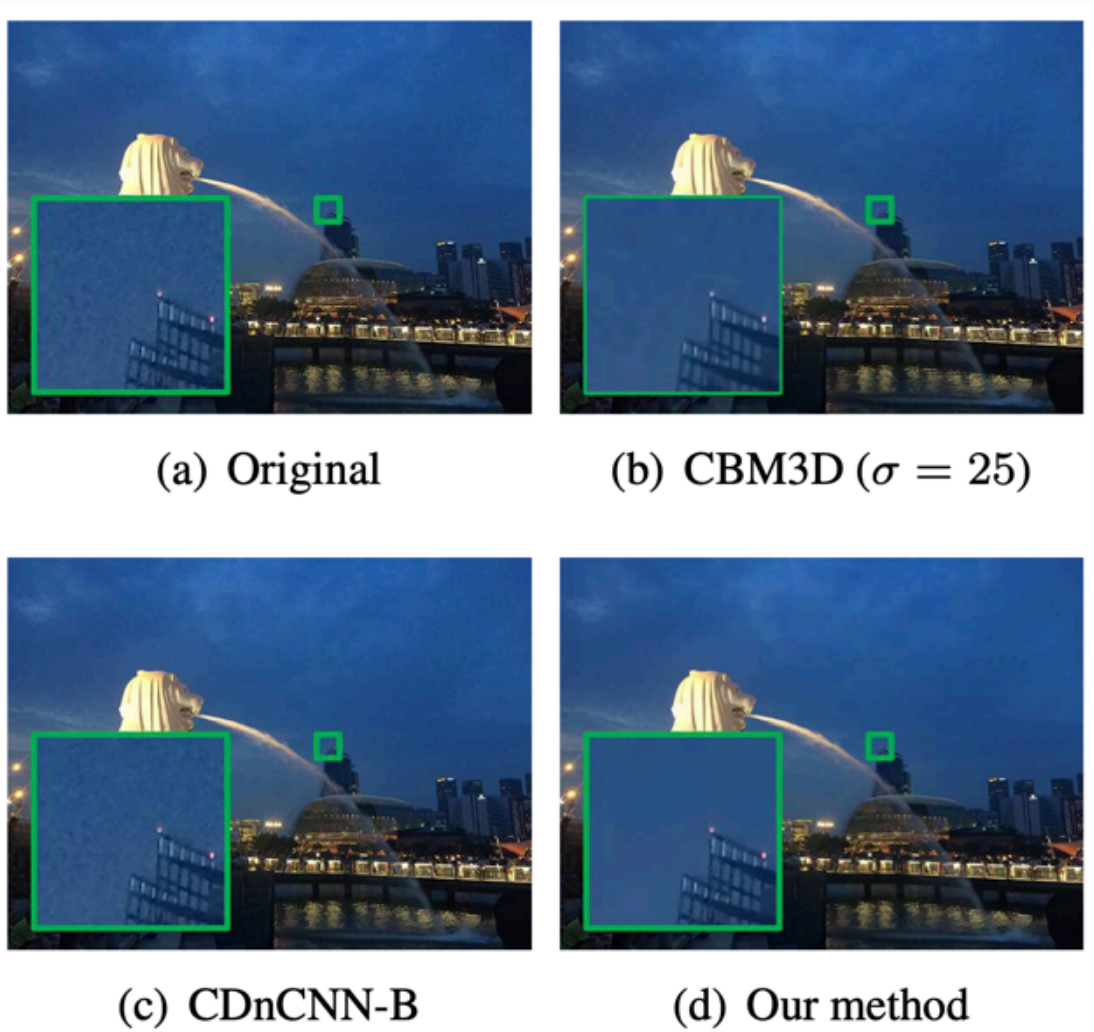
Application: Compression Artifact Removal



Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4826–4835, 2017.

Application: Denoising

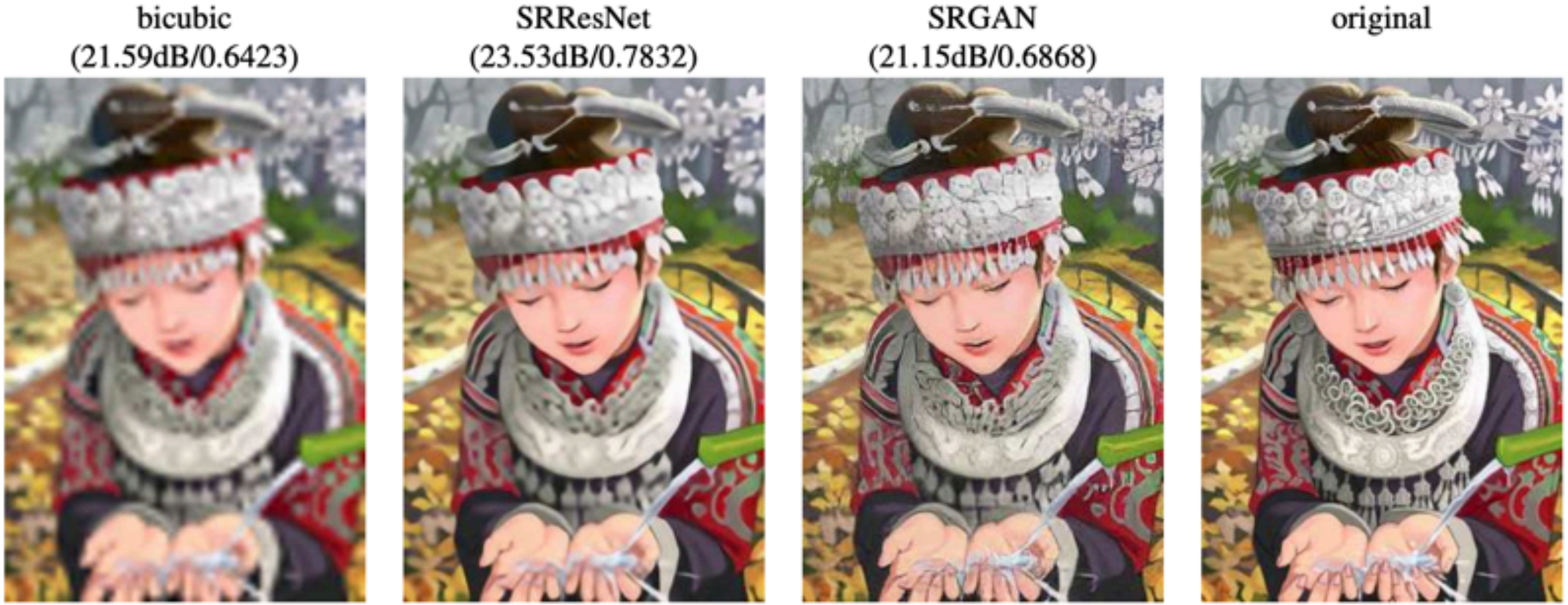




Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. Image blind denoising with generative adversarial network based noise modeling. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

Application: Super-resolution

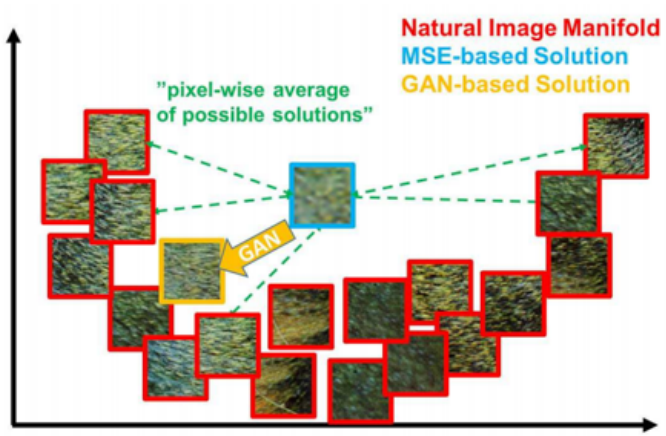
4 x Upsampling



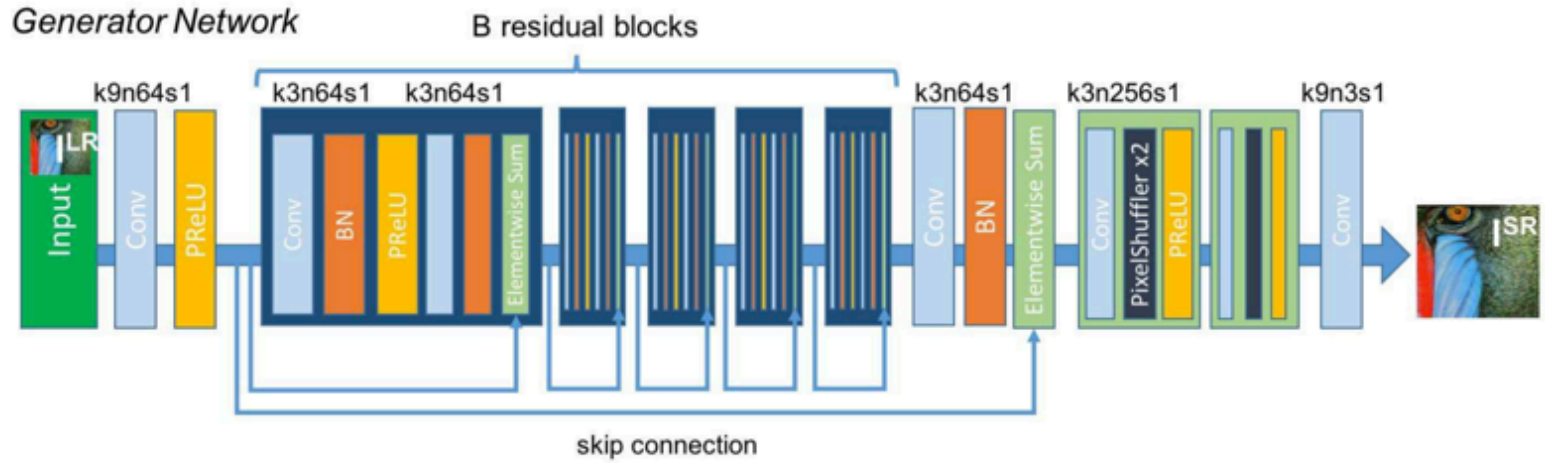
Ledig et al. Photo-realistic single image superresolution using a generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4681–4690, 2017.

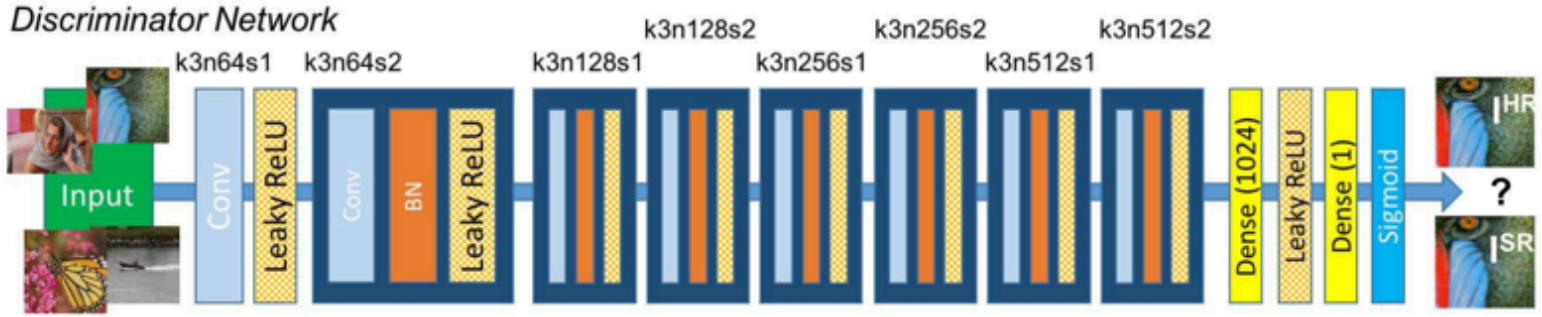
Application: Super-resolution

SRGAN



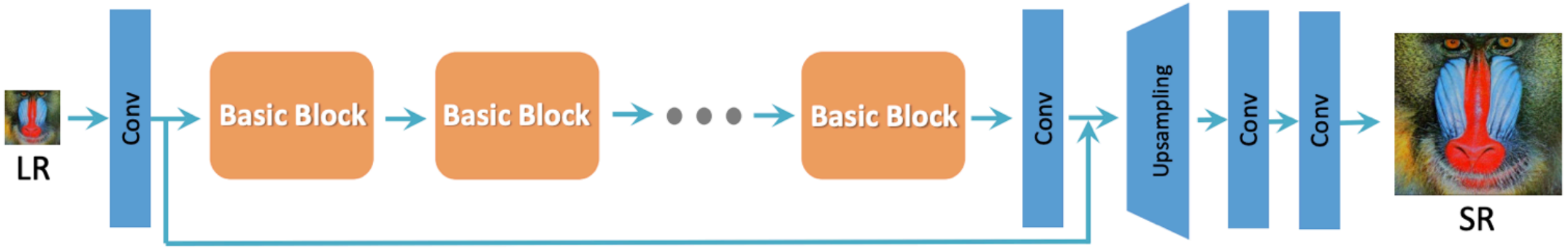
$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{\text{train}}(I^{HR})}[\log D_{\theta_D}(I^{HR})] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})}[\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))] \quad (2)$$

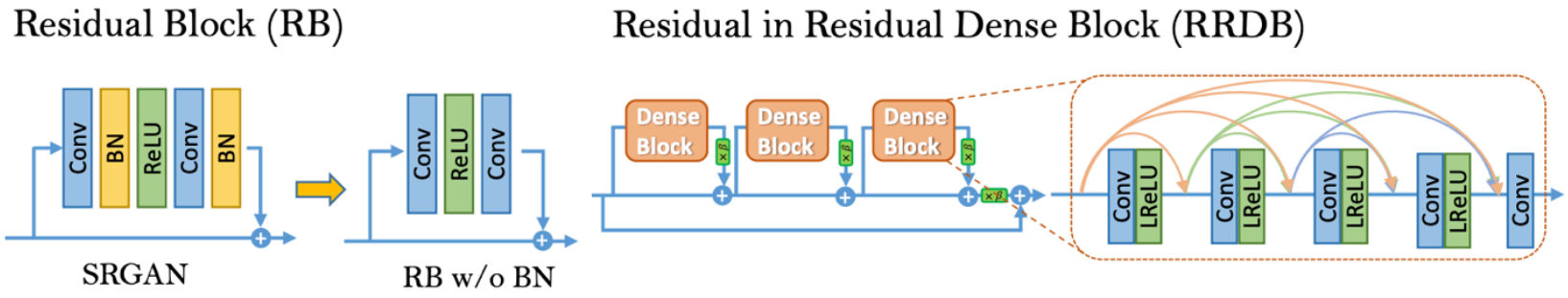




Ledig et al. Photo-realistic single image superresolution using a generative adversarial network. CVPR, 2017.

ESRGAN (Enhanced SRGAN)





$D(x_r) = \sigma(C(\text{Real})) \rightarrow 1$ Real?

$D(x_f) = \sigma(C(\text{Fake})) \rightarrow 0$ Fake?

a) Standard GAN

$D_{Ra}(x_r, x_f) = \sigma(C(\text{Real}) - \mathbb{E}[C(\text{Fake})]) \rightarrow 1$ More realistic than fake data?

$D_{Ra}(x_f, x_r) = \sigma(C(\text{Fake}) - \mathbb{E}[C(\text{Real})]) \rightarrow 0$ Less realistic than real data?

b) Relativistic GAN

https://arxiv.org/pdf/1809.00219.pdf

ESRGAN (Enhanced SRGAN)

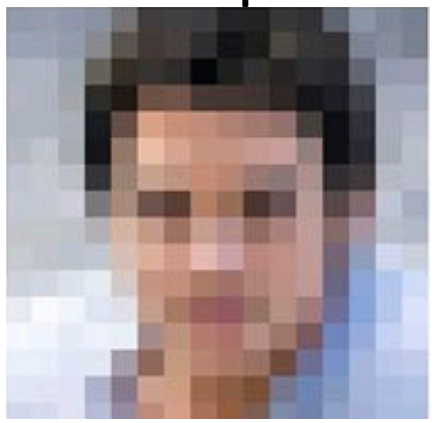
https://arxiv.org/pdf/1809.00219.pdf

https://github.com/xinntao/BasicSR

ESRGAN on Resident Evil 2 demo video: https://www.youtube.com/watch?v=DZZ2HiN5_zc

Face Hallucination versus Face Super-resolution

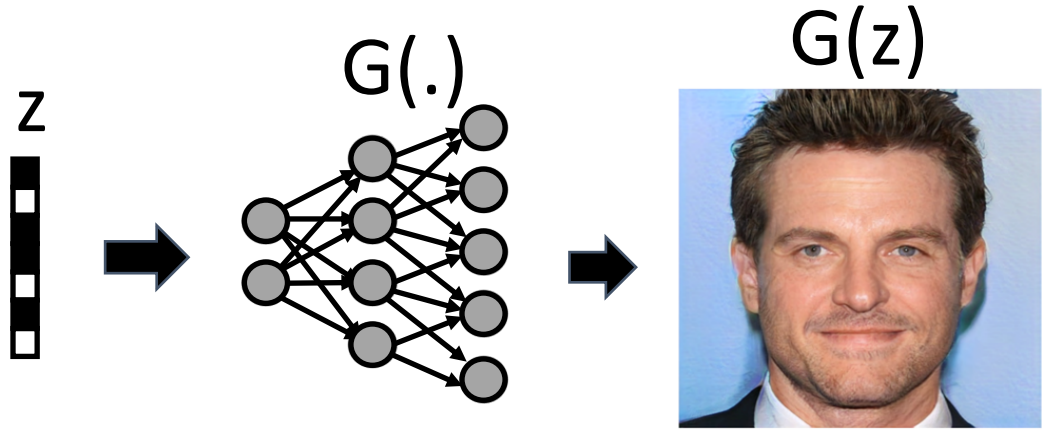
16x16 pixels

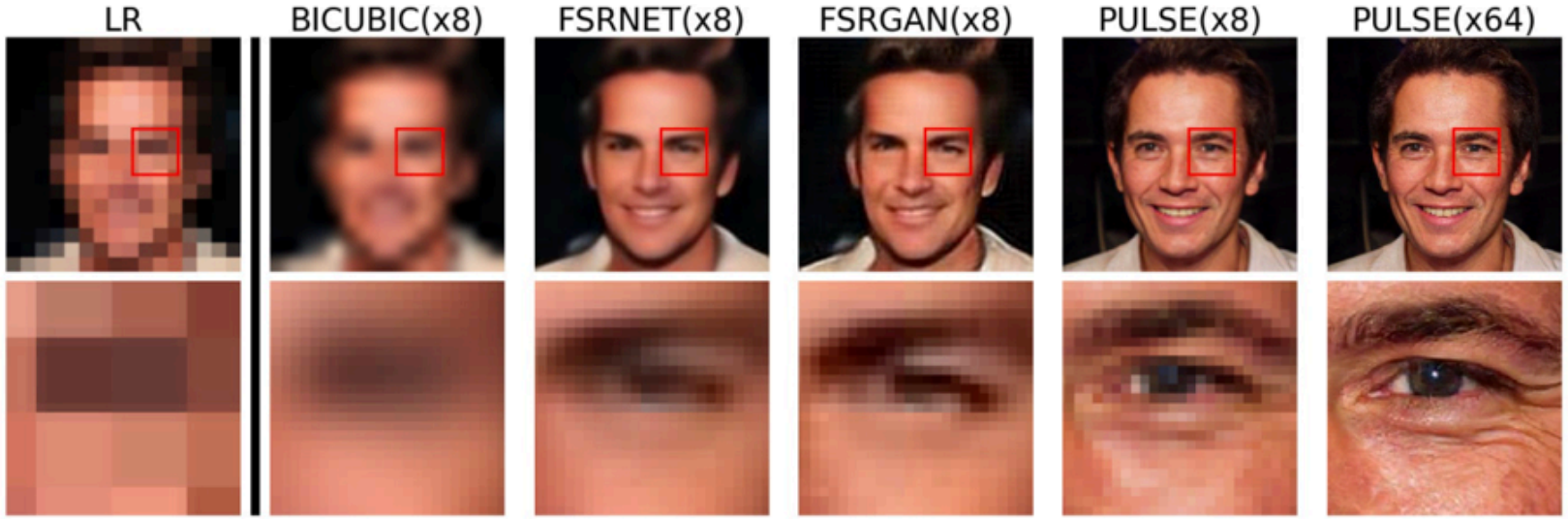


32 x upsampling

Face Hallucination as an Inversion Problem

$$\|DS(G(z)) - I_{LR}\|_p^p \leq \epsilon.$$





PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models. CVPR'20

theverge.com/21298762/face-depixelizer-ai-machine-learning-tool-pulse-stylegan-obama...

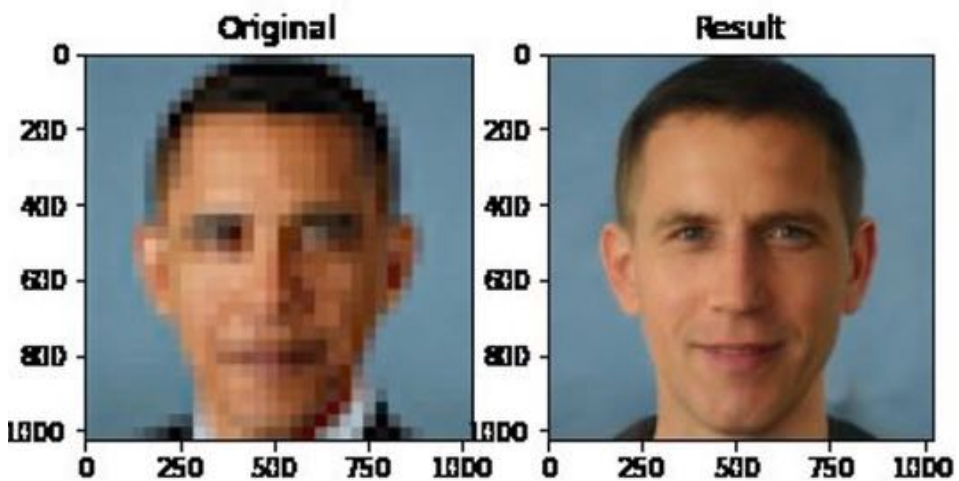
REPORT | TECH | ARTIFICIAL INTELLIGENCE

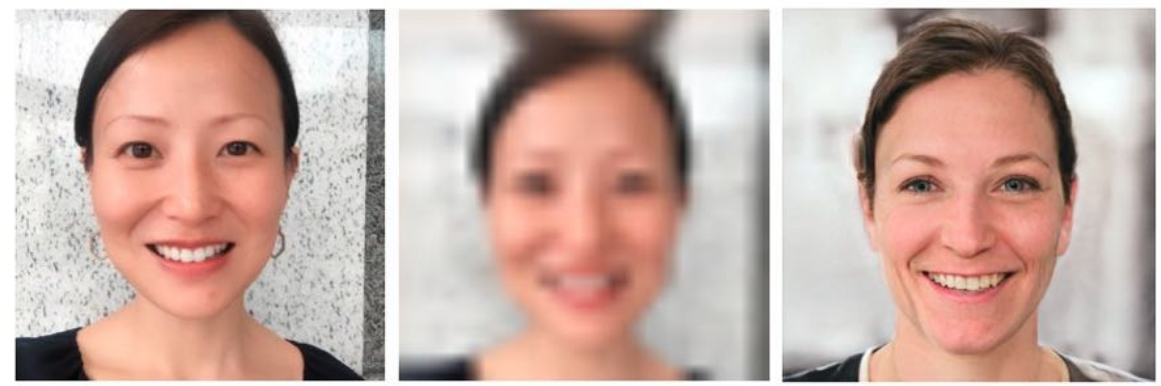
What a machine learning tool that turns Obama white can (and can't) tell us about AI bias

A striking image that only hints at a much bigger problem

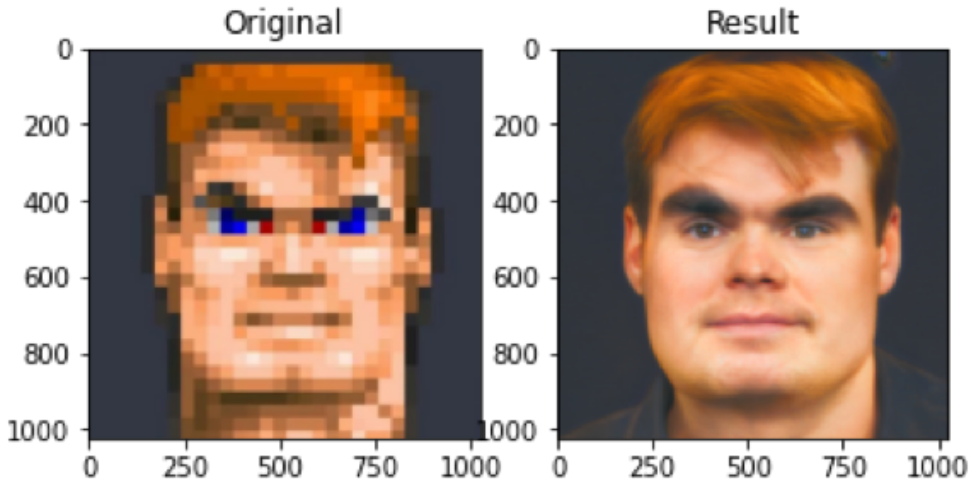
By James Vincent | Jun 23, 2020, 3:45pm EDT

SHARE

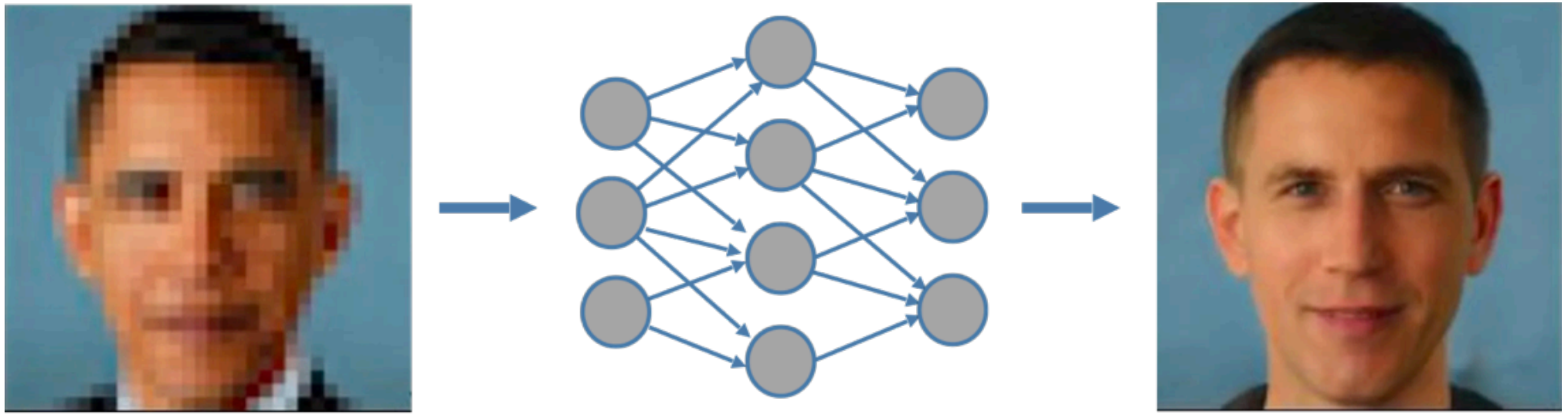




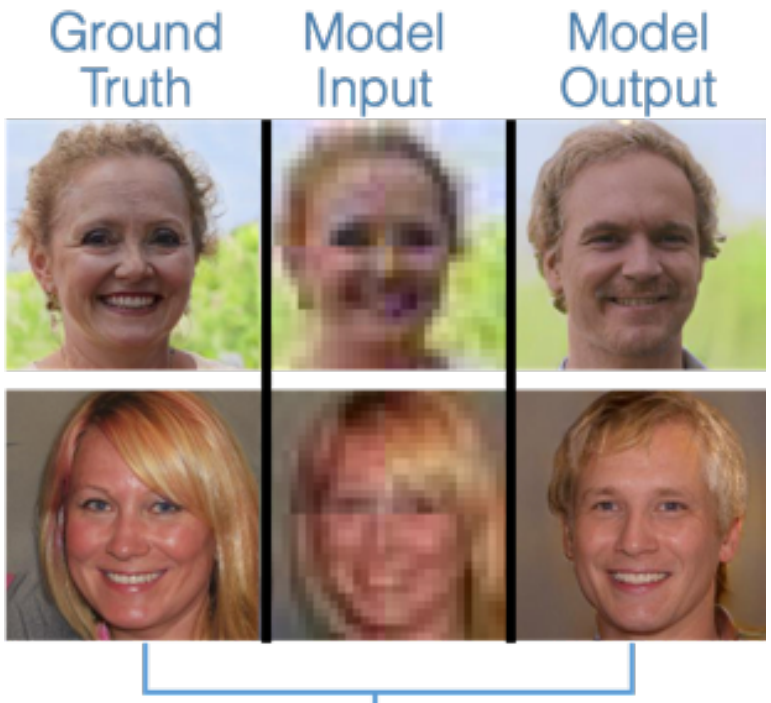

49
AMMO
100%
HEALTH
ARMS
0%
ARMOR
BULL 49 / 200
SHEL 0 / 50
RCKT 0 / 50
CELL 0 / 300


Original
Result
0
200
400
600
800
1000
0
250
500
750
1000

Quantifying the AI Biases

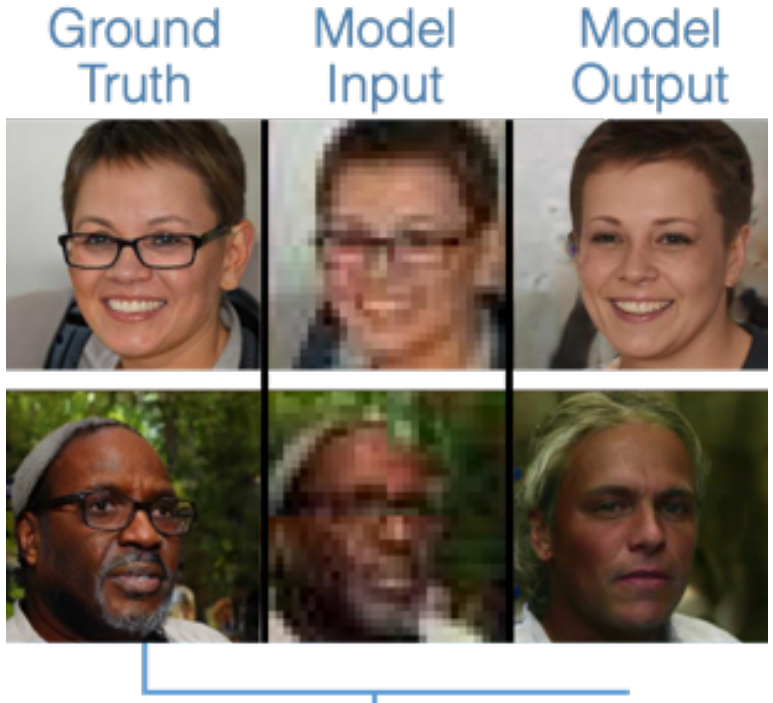


Quantifying the AI Biases



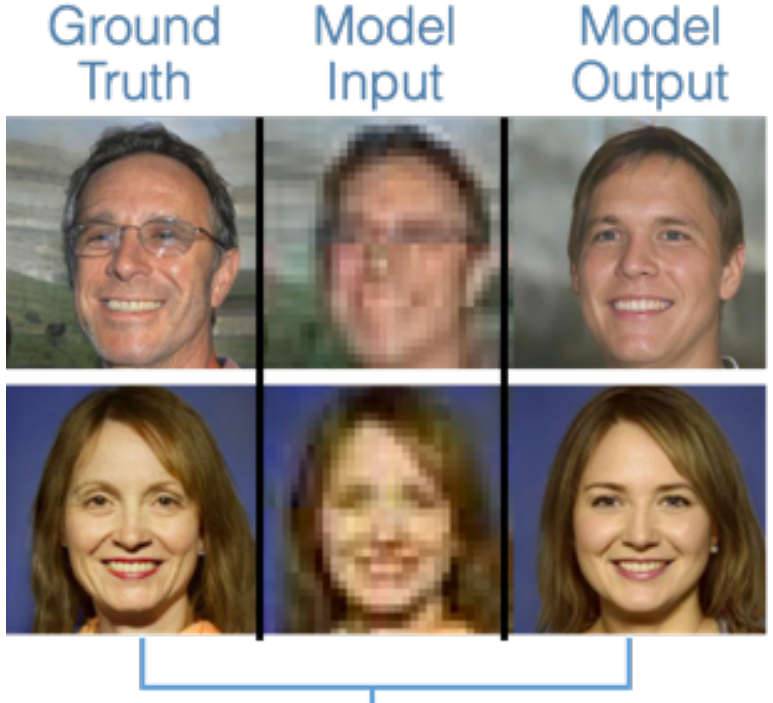
Gender changed

Male	Female
3.4%	7.8%



Glasses removed

With Glasses	Without Glasses
89.8%	0.3%



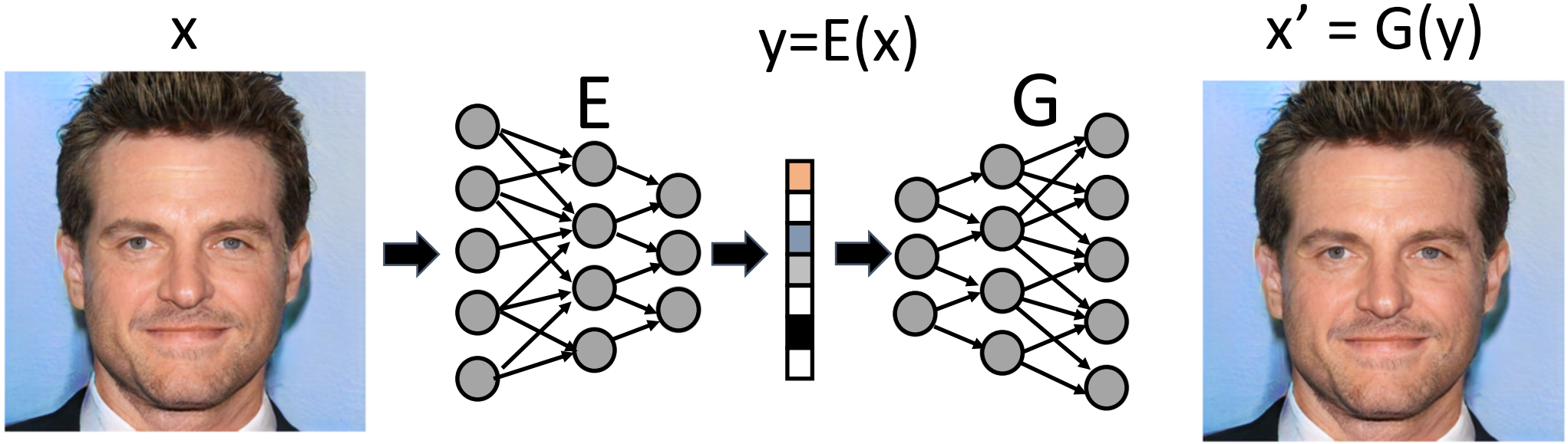
Age changed

Young	Elder
3.3%	29.5%

Improving the Fairness of Deep Generative Models without Retraining. Tan, Shen, Zhou. https://arxiv.org/pdf/2012.04842.pdf

Neural Compression

Neural Image Compression

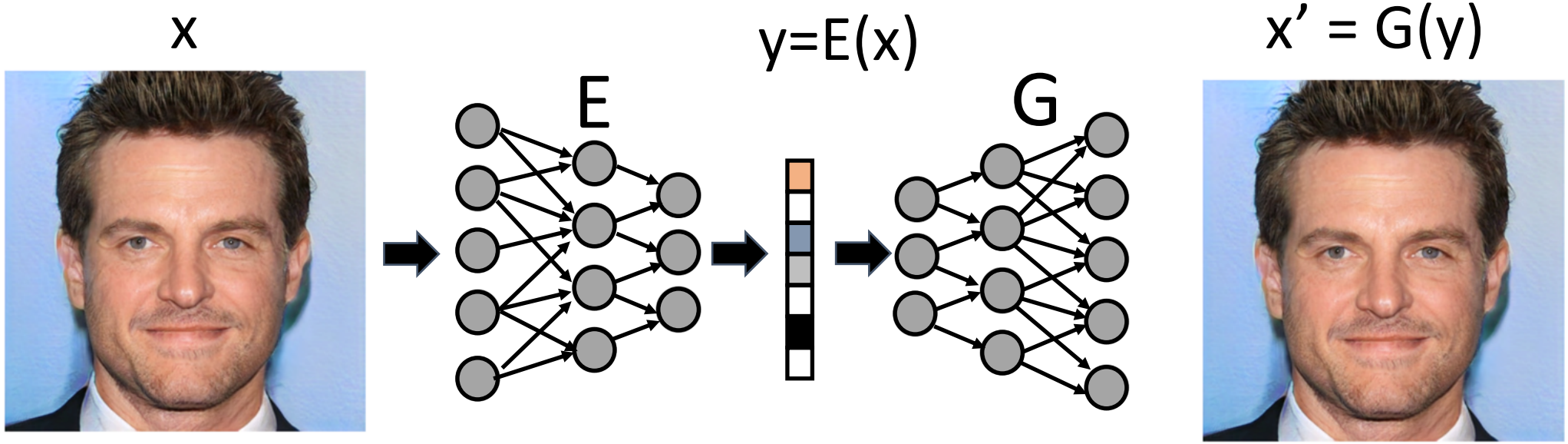


r(y) = -log(P(y))

Objective: $\mathcal{L}_{EG} = \mathbb{E}_{x \sim p_X}[\lambda\, r(y) + d(x, x')].$

* P is the distribution model of y, using P and an entropy coding algorithm (e.g., arithmetic coding [30]), we can store y losslessly using bitrate r(y)

Neural Image Compression with Adversarial Loss

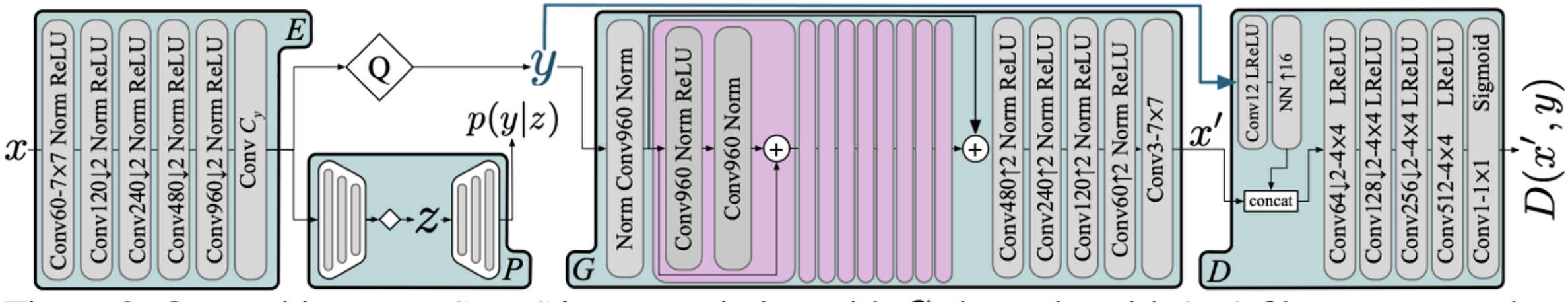


Objective:

$$\mathcal{L}_{EGP} = \mathbb{E}_{x\sim p_X}[\lambda r(y) + d(x, x') - \beta \log(D(x', y))],$$
$$\mathcal{L}_D = \mathbb{E}_{x\sim p_X}[-\log(1 - D(x', y))] + \mathbb{E}_{x\sim p_X}[-\log(D(x, y))].$$

* P is the distribution of code y, then an entropy coding algorithm on y

High-Fidelity Generative Image Compression (NeurIPS'20)



* Q is quantization, P is hyper-prior model

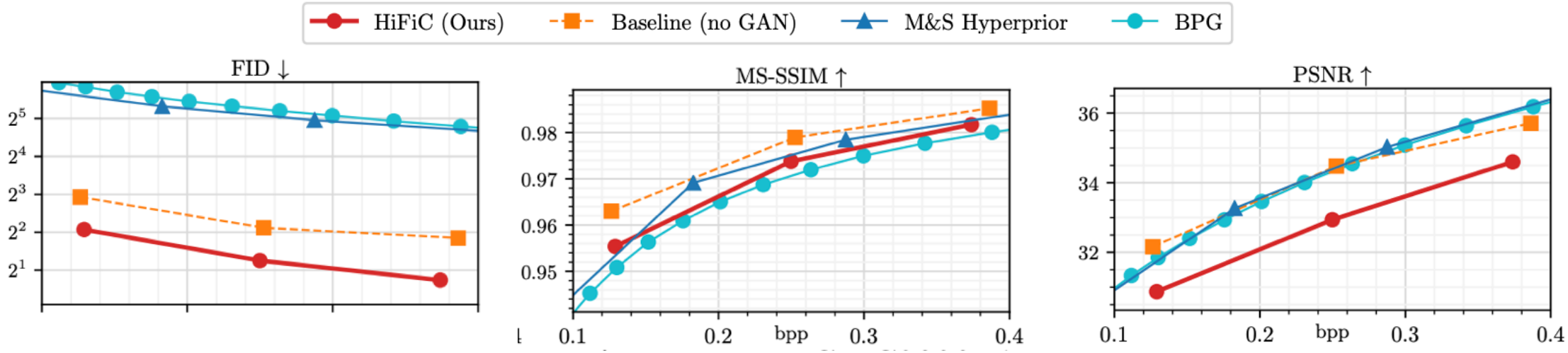
$$\mathcal{L}_{EGP} = \mathbb{E}_{x\sim p_X}[\lambda r(y) + d(x, x') - \beta \log(D(x', y))],$$
$$\mathcal{L}_D = \mathbb{E}_{x\sim p_X}[-\log(1 - D(x', y))] + \mathbb{E}_{x\sim p_X}[-\log(D(x, y))].$$

.https://hific.github.io/

Experiments of High Fidelity Compression (HiFiC)

- Training set: It consists of a large set of high-resolution images collected from the Internet
- Testing set:
  - Kodak [23] dataset (24 images),
  - CLIC2020 [46] testset (428 images)
  - DIV2K [2] validation set (100 images)

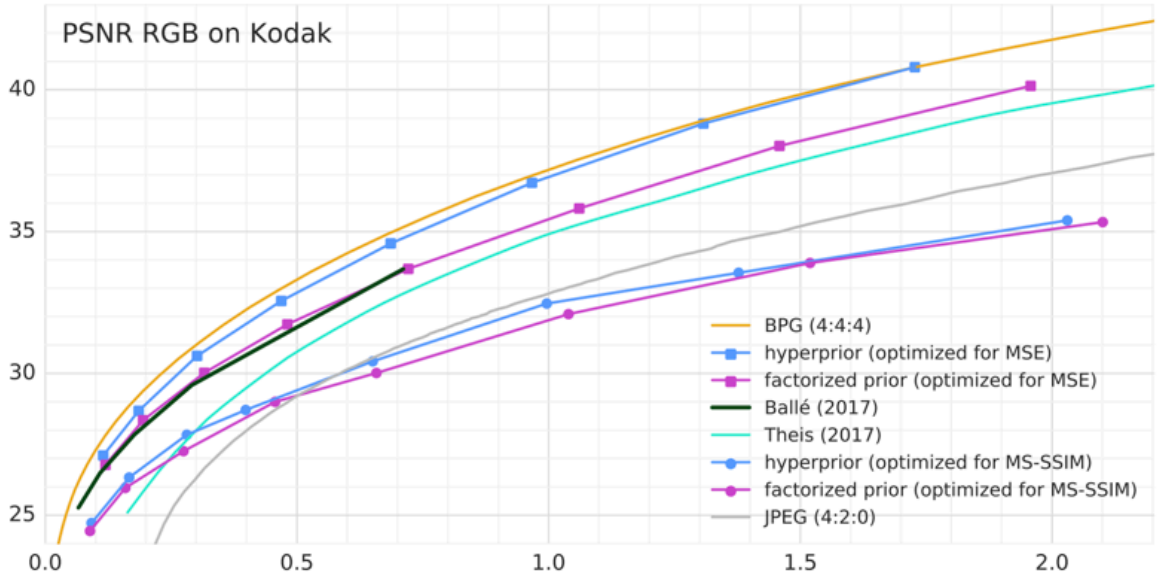
bpp: bit per pixel

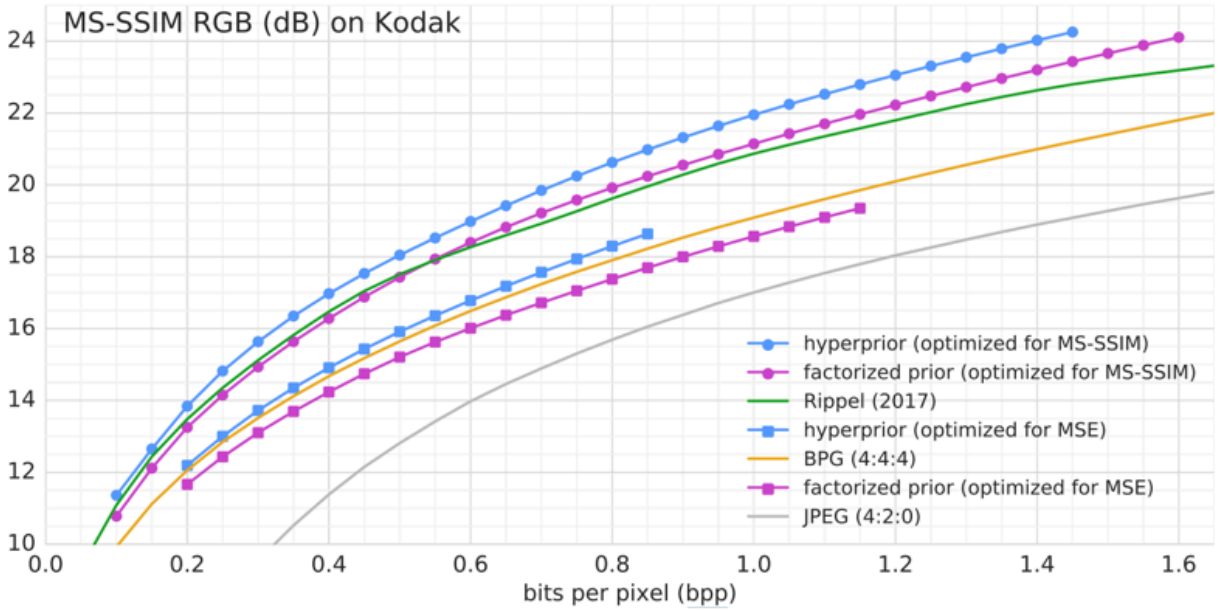


Calculating bpp for an image

- Say we have a jpg image with 420x920, its size is 20635 bytes. What is the bpp in this image?
- We know 1 byte = 8 bits, bpp (bits per pixel) = 20635 * 8 bits / (420*920)
- Other knowledge about units:
  - 1 Mb = 1024 kb
  - 1 kb = 1024 bytes

Experiments of BPG

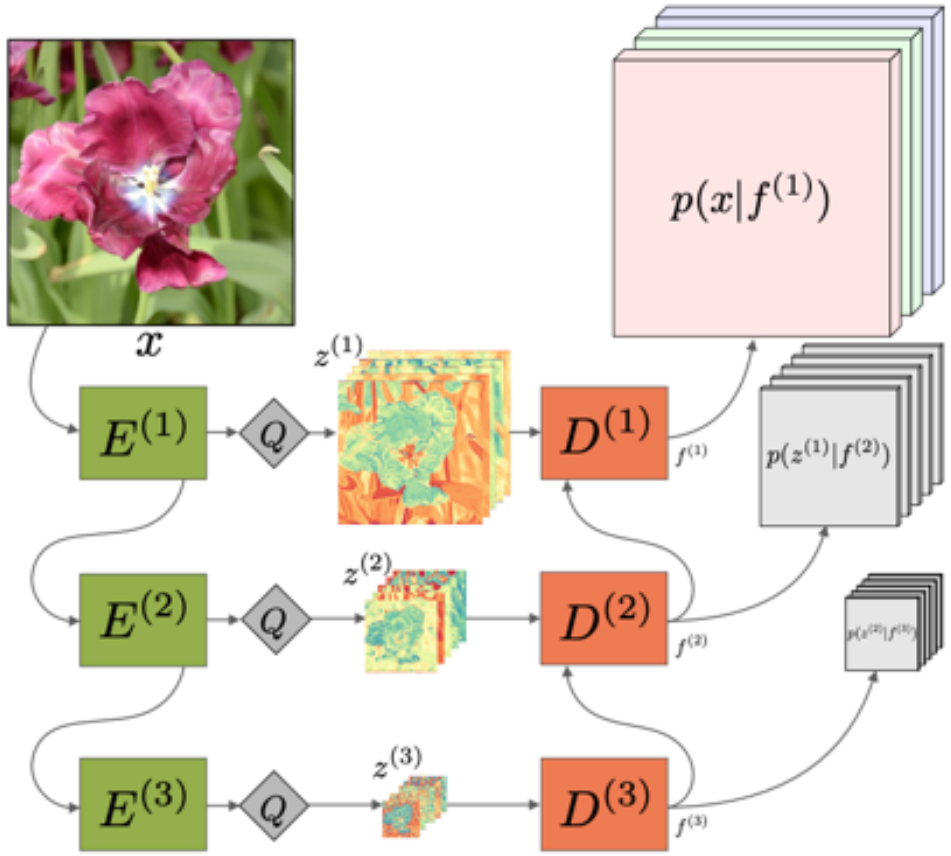


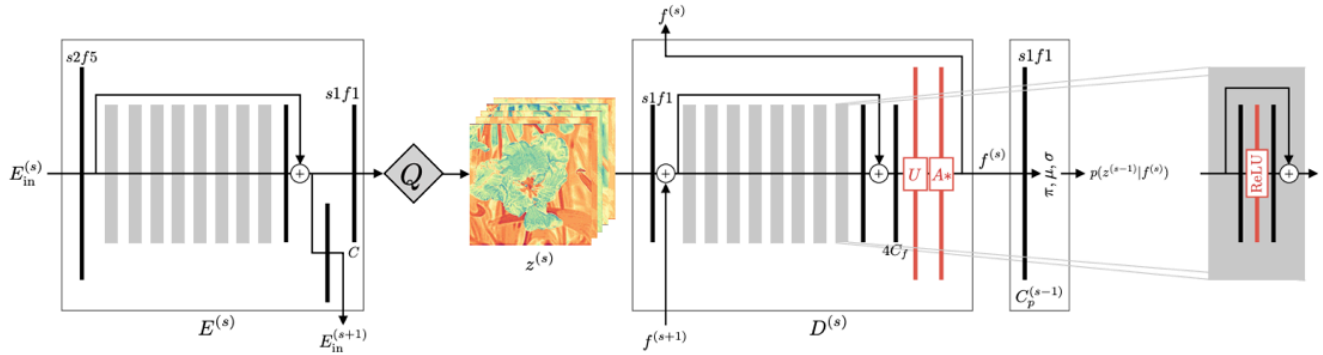


VARIATIONAL IMAGE COMPRESSION WITH A SCALE HYPERPRIOR. ICLR'18, https://arxiv.org/pdf/1802.01436.pdf

Interactive demo images: https://hific.github.io/

Learned Lossless Image Compression



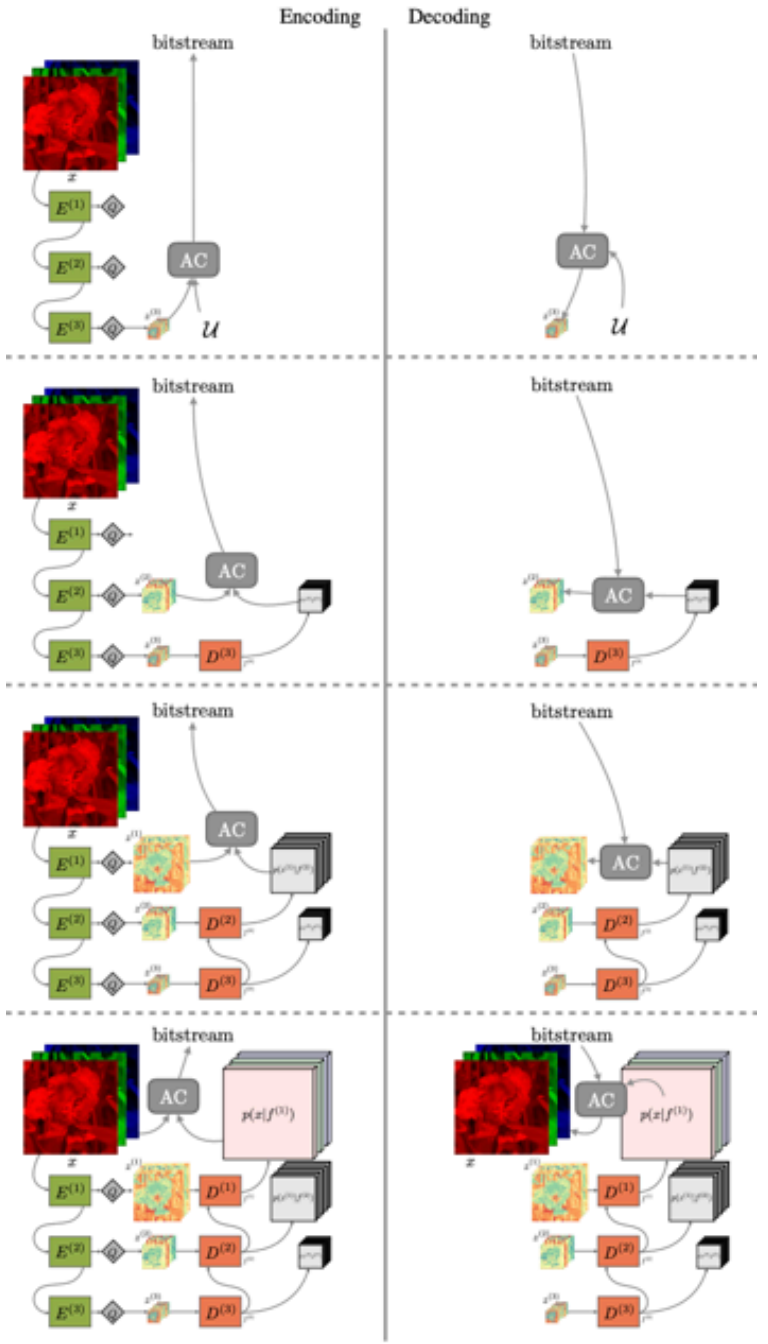


[bpsp]	ImageNet32	Learned
L3C (ours)	4.76	✓
PixelCNN [46]	3.83	✓
MS-PixelCNN [32]	3.95	✓
PNG	6.42	
JPEG2000	6.35	
WebP	5.28	
FLIF	5.08	

Table 3: Comparing bits per sub-pixel (bpsp) on the 32 × 32 im-

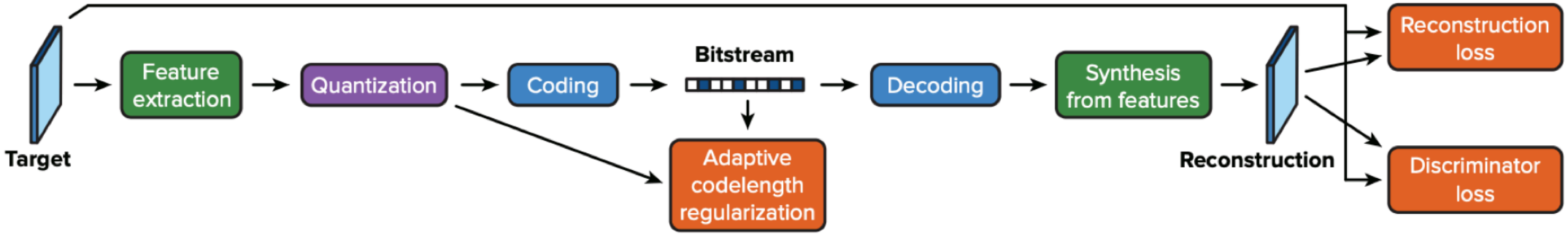
Practical Full Resolution Learned Lossless Image Compression. CVPR'20. https://arxiv.org/pdf/1811.12817.pdf

Learned Lossless Image Compression

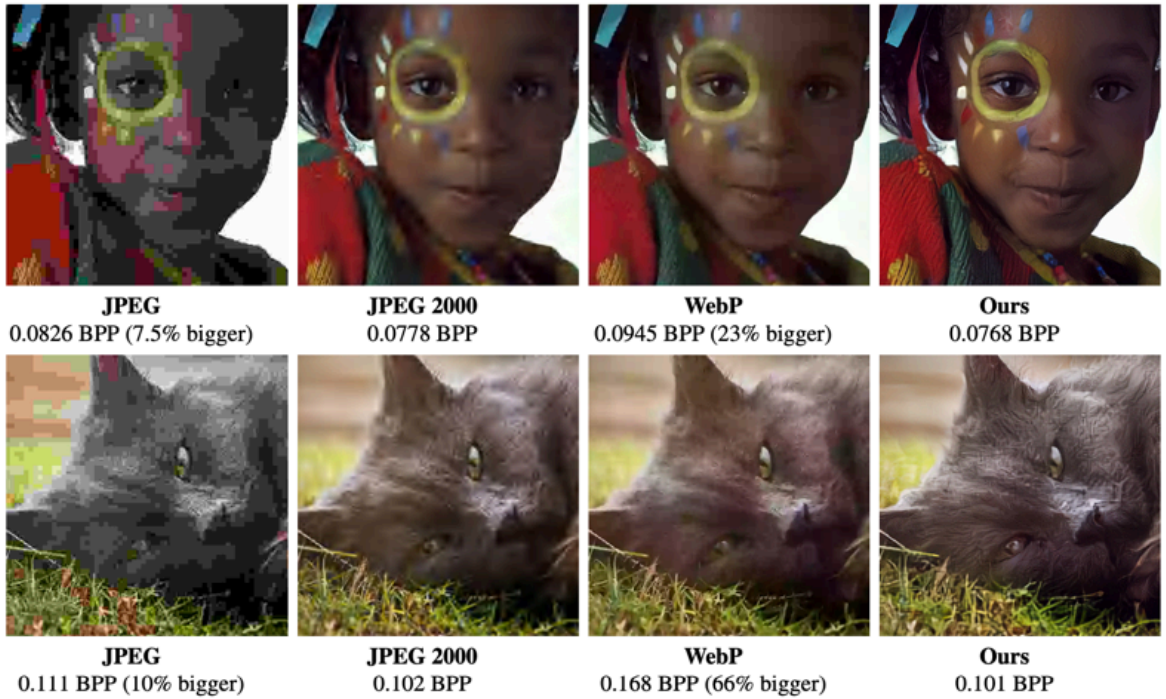


Practical Full Resolution Learned Lossless Image Compression. CVPR'20. https://arxiv.org/pdf/1811.12817.pdf

Real-Time Adaptive Image Compression

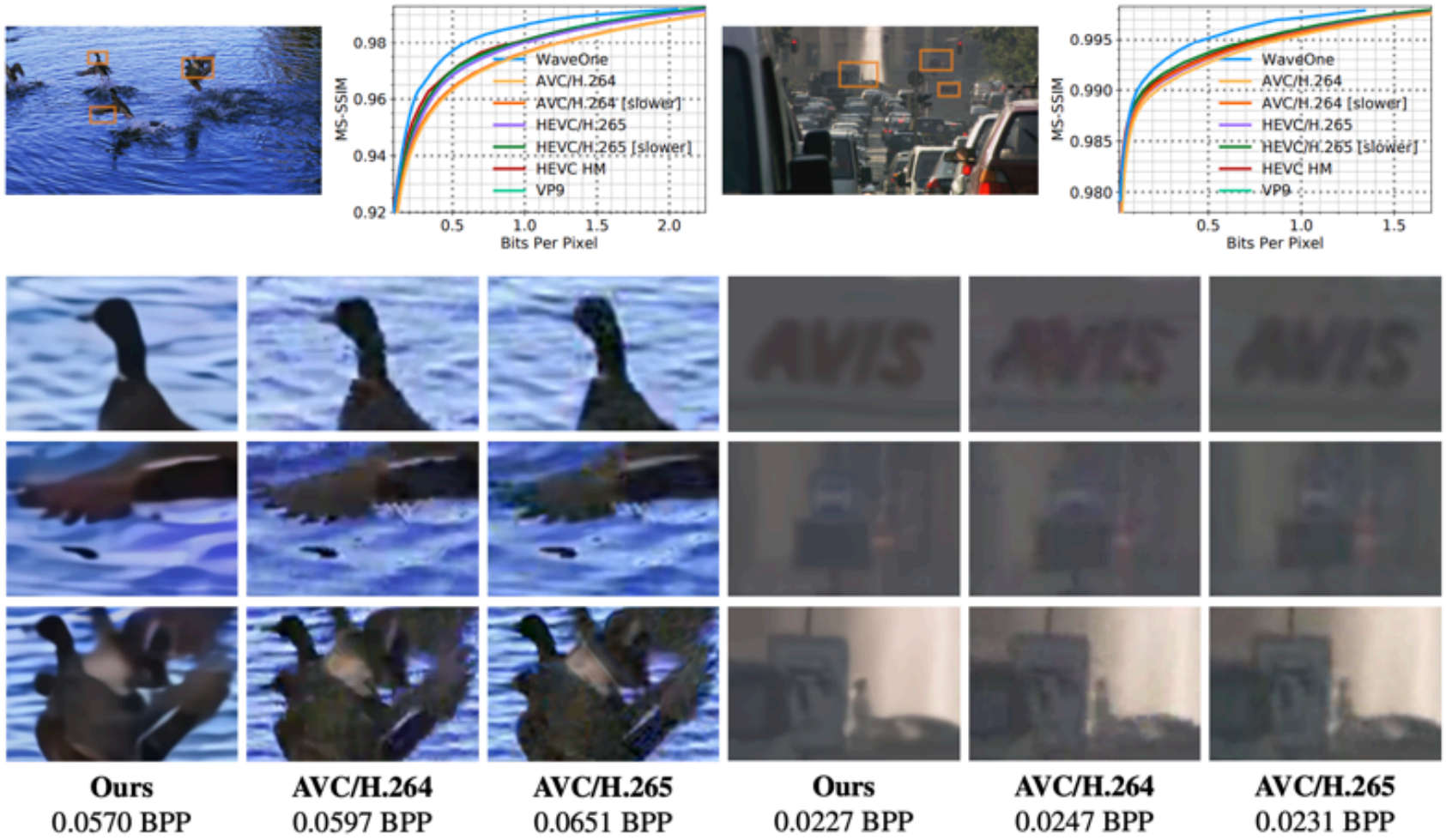


Codec	RGB file size (kb)	YCbCr file size (kb)	Encode time (ms)	Decode time (ms)
Ours	21.4 (100%)	17.4 (100%)	8.6*	9.9*
JPEG	65.3 (304%)	43.6 (250%)	18.6	13.0
JP2	54.4 (254%)	43.8 (252%)	367.4	80.4
WebP	49.7 (232%)	37.6 (216%)	67.0	83.7



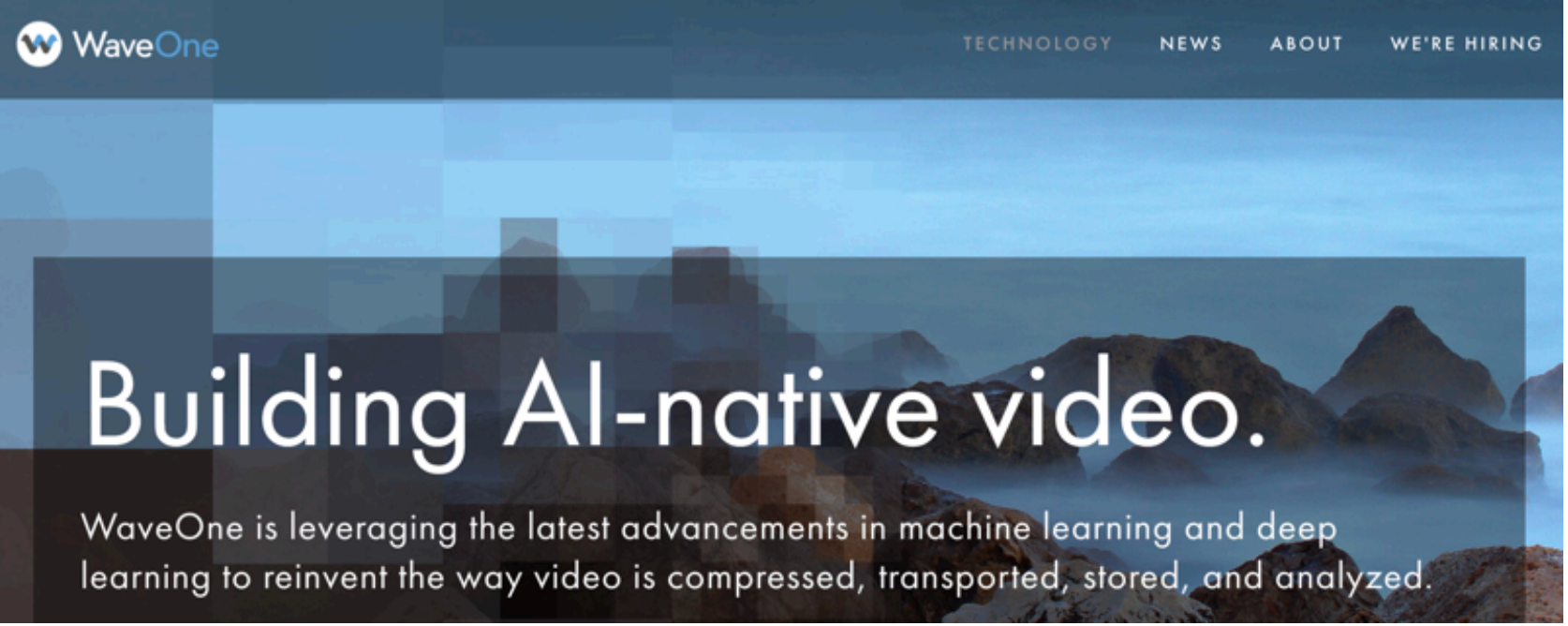
Rippel and Bourdev. Real-Time Adaptive Image Compression. ICML'17. https://arxiv.org/pdf/1705.05823.pdf

Learned Video Compression



https://www.wave.one/

Rippel et al. Learned Video Compression. ICCV'19. https://arxiv.org/pdf/1811.06981.pdf



https://www.wave.one/

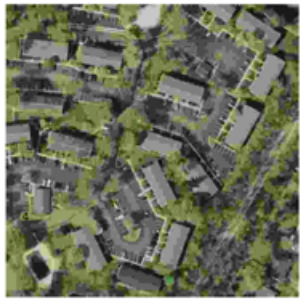
Domain-Specific

AI-native compression can be optimized for specific types of video, dramatically improving coding efficiency.

Video Conferencing

Street Views

Gaming

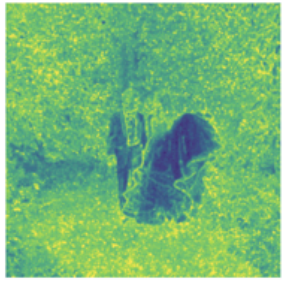


Satellite

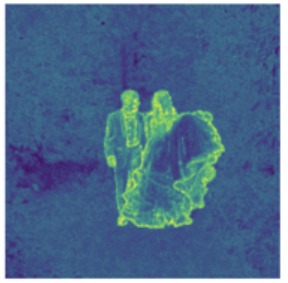
Semantics-Aware

AI-native compression can be trained to associate each pixel with its object type, prioritizing the pixels that matter most.

Source



Baseline Bit Allocation



Semantics-Aware Bit Allocation

Semantics-Aware Result

Other Resources on Neural Compression

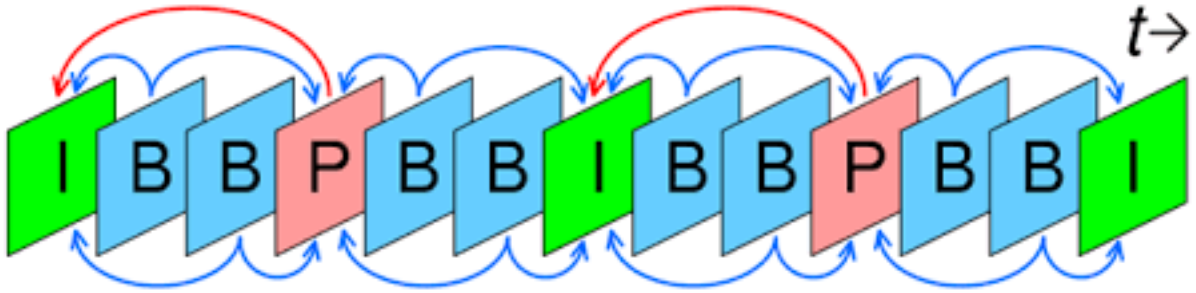
- Neural Compression Workshop: https://neuralcompression.github.io/
- Workshop and Challenge on Learned Image Compression: http://clic.compression.cc/

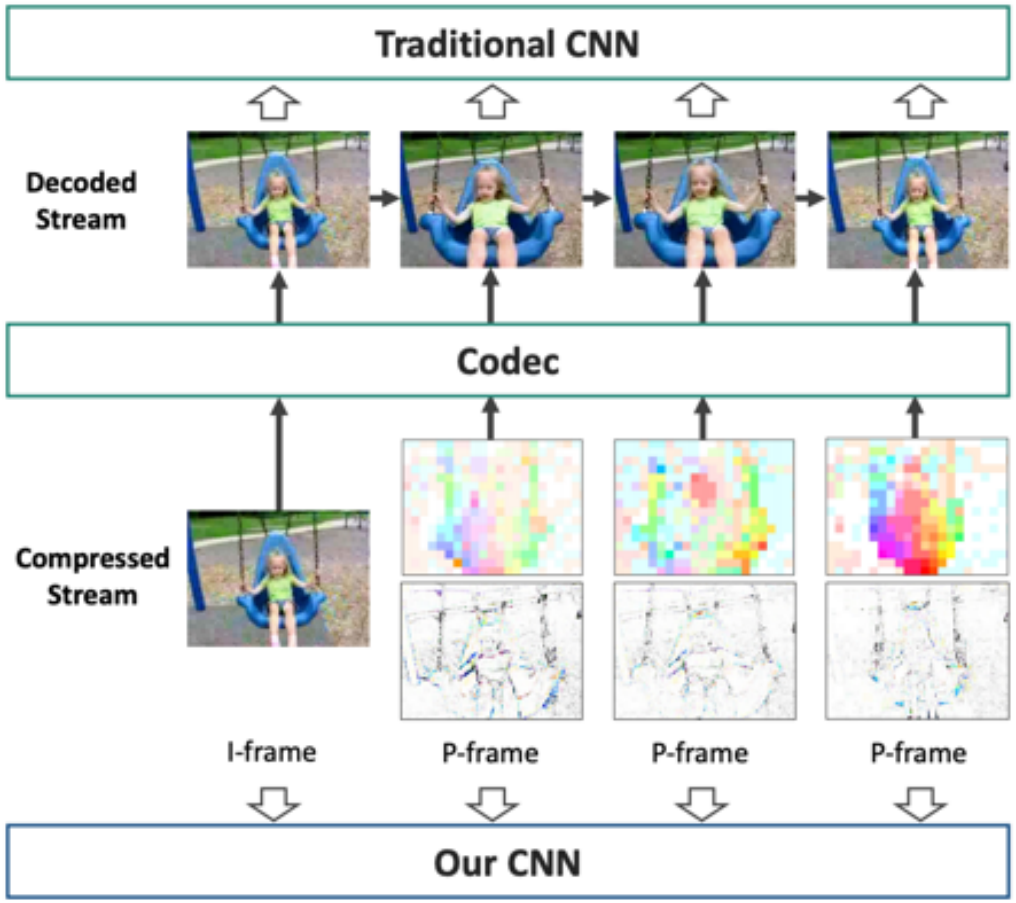
Big tech companies really care about this technology!

Other Application: Action Recognition on Compressed Video

- I-frames (intra-coded frames)
- P/B-frames (predictive frames)
  - Both B- and P- frames capture only what changes in the video

$$I_i^{(t)} = I_{i-\mathcal{T}_i^{(t)}}^{(t-1)} + \Delta_i^{(t)},$$



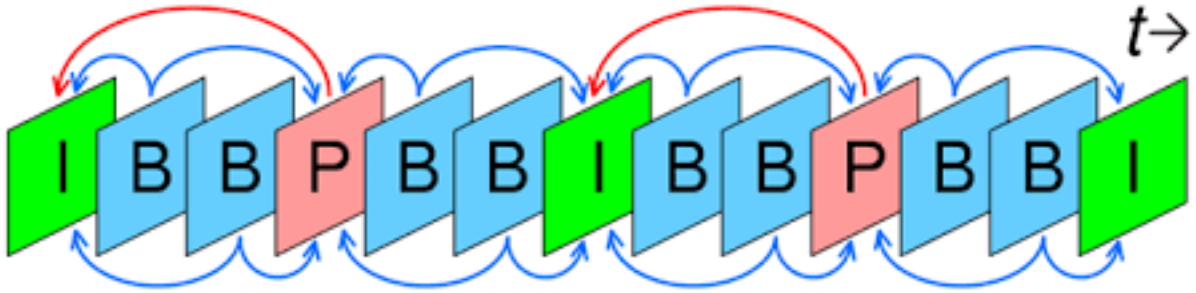


Wu et al. Compressed Video Action Recognition. CVPR'18 https://arxiv.org/pdf/1712.00636.pdf

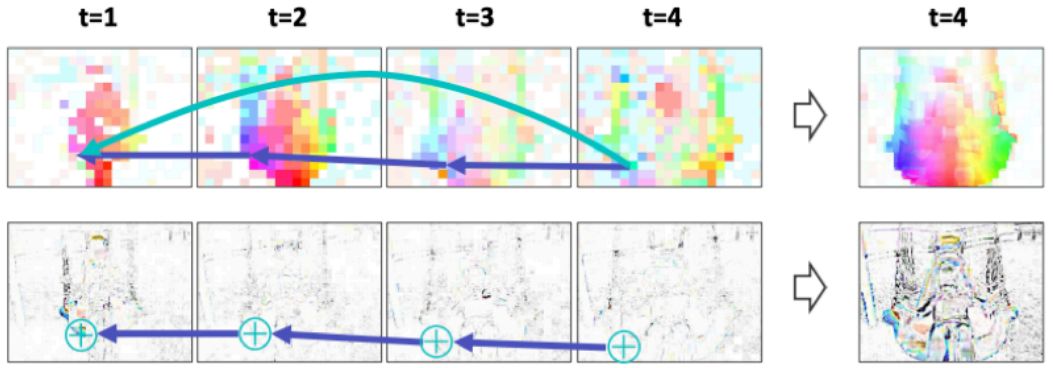
Other Application: Action Recognition on Compressed Video

- I-frames (intra-coded frames)
- P/B-frames (predictive frames)
  - Both B- and P- frames capture only what changes in the video

$$I_i^{(t)} = I_{i-\mathcal{T}_i^{(t)}}^{(t-1)} + \Delta_i^{(t)},$$



Trace-back from long-term dependency



$$\mathcal{J}_i^{(t,k)} := \mu_{\mathcal{T}^{(k+1)}} \circ \cdots \circ \mu_{\mathcal{T}^{(t)}}(i).$$

$$I_i^{(t)} = I_{i-\mathcal{D}_i^{(t)}}^{(0)} + \mathcal{R}_i^{(t)}, \quad t = 1, 2, \ldots,$$

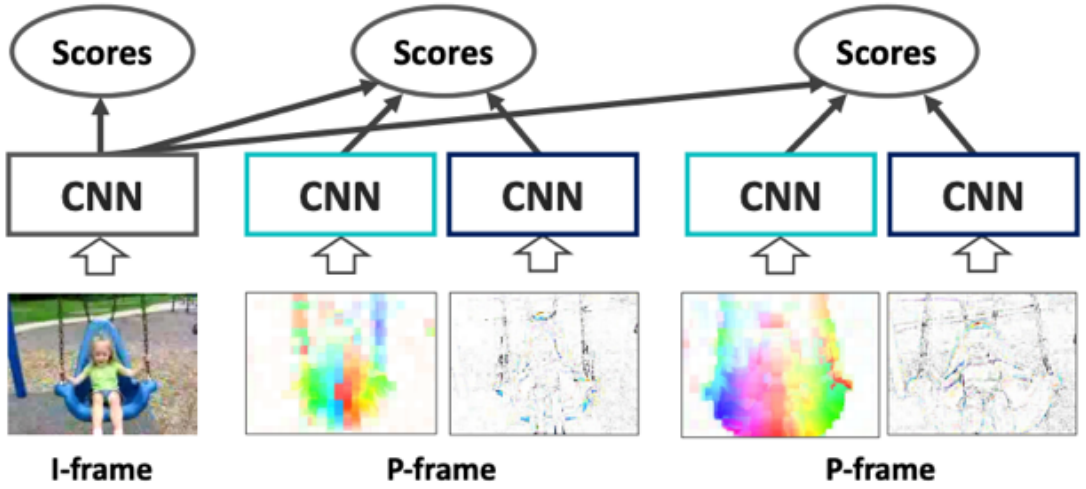
$$\mathcal{D}_i^{(t)} := i - \mathcal{J}_i^{(t,k)}, \text{ and}$$
$$\mathcal{R}_i^{(t)} := \Delta_{\mathcal{J}_i^{(t,k+1)}}^{(k+1)} + \cdots + \Delta_{\mathcal{J}_i^{(t,t-1)}}^{(t-1)} + \Delta_i^{(t)},$$

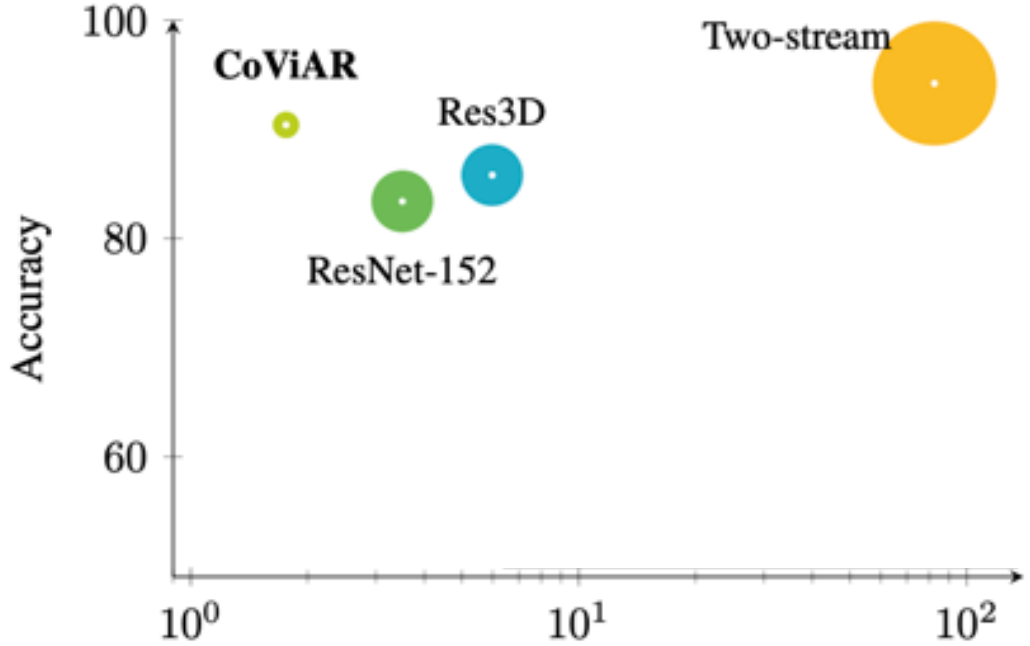
Wu et al. Compressed Video Action Recognition. CVPR'18 https://arxiv.org/pdf/1712.00636.pdf

Other Application: Action Recognition on Compressed Video

- Use ResNet-152 to model I-frames and ResNet-18 to model the motion vectors and residual

$$x^{(0)}_{\text{RGB}} := \phi_{\text{RGB}}\left(I^{(0)}\right)$$
$$x^{(t)}_{\text{motion}} := \phi_{\text{motion}}\left(\mathcal{D}^{(t)}\right)$$
$$x^{(t)}_{\text{residual}} := \phi_{\text{residual}}\left(\mathcal{R}^{(t)}\right)$$



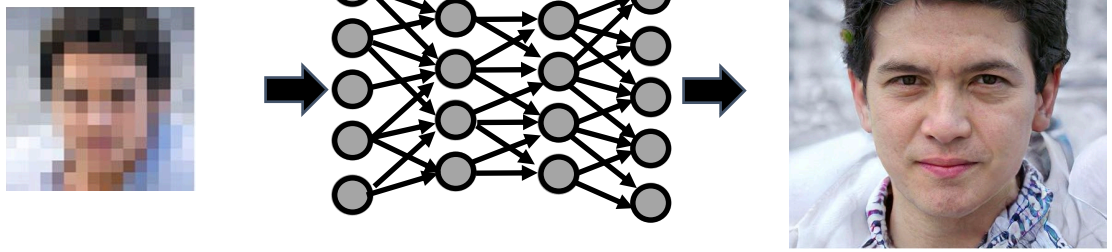


speed-accuracy on UCF101

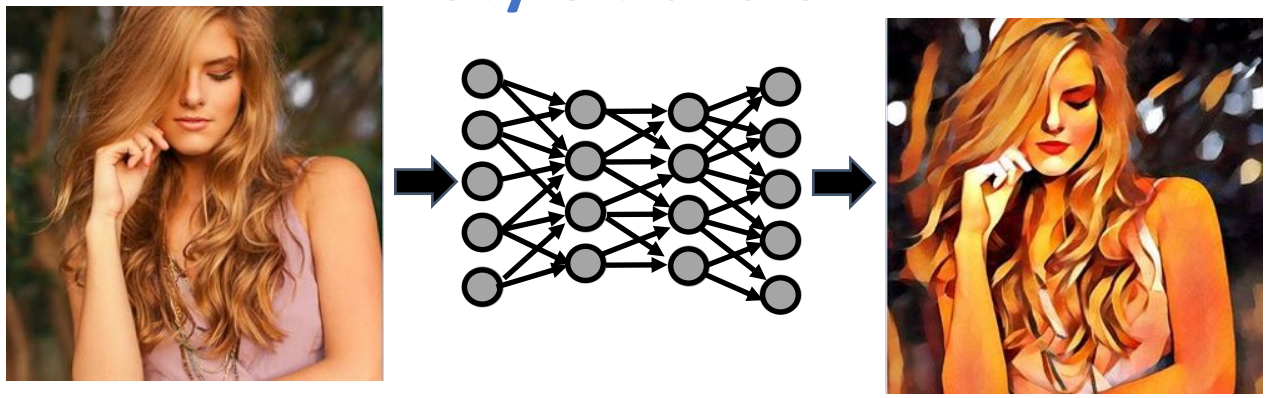
Wu et al. Compressed Video Action Recognition. CVPR'18 https://arxiv.org/pdf/1712.00636.pdf

New Frontier: Neural Image Processing

Super-resolution



Style transfer



Colorization

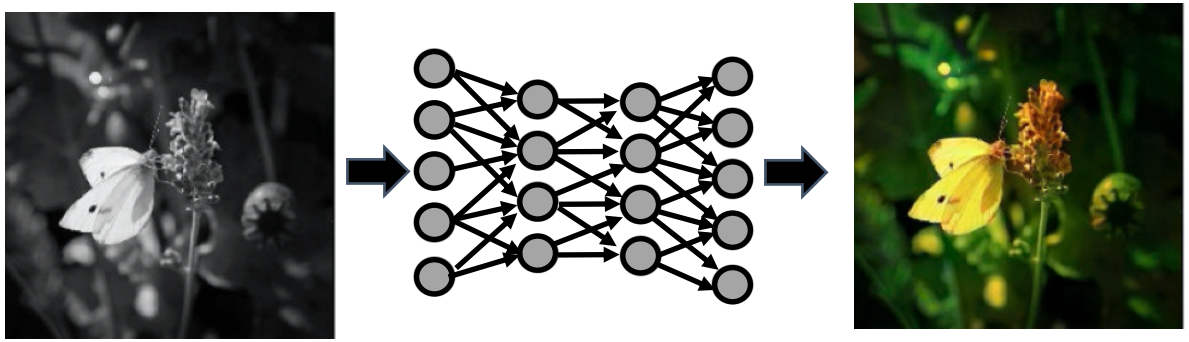


Image compression