Image Generation and GANs

CSE P576

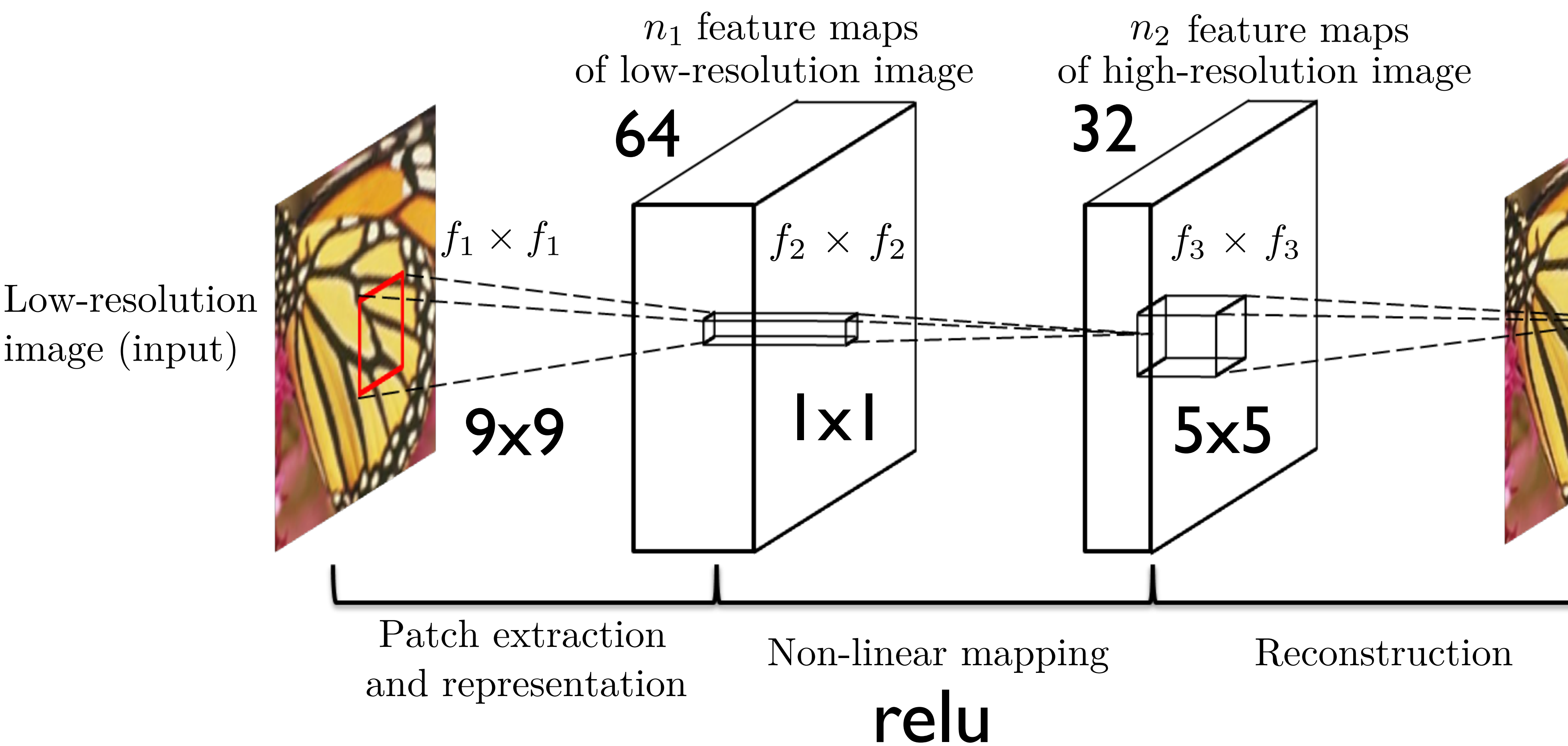
Dr. Matthew Brown

Image Generation + GANs

- Can a neural net define the loss function?
- Loss functions for Super-Resolution: L2, VGG, Adversarial
- Generative Adversarial Nets and Image Generation
- Conditional GANs, Image Translation, pix2pix

Super-Resolution: SRCNN

- Small network (3 layers) generates reasonable results

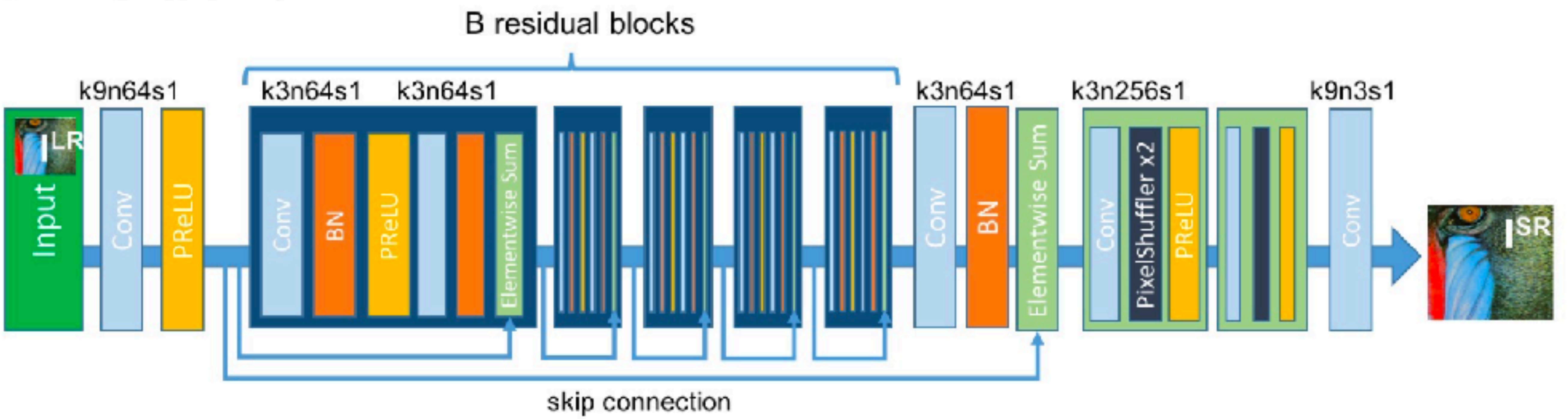


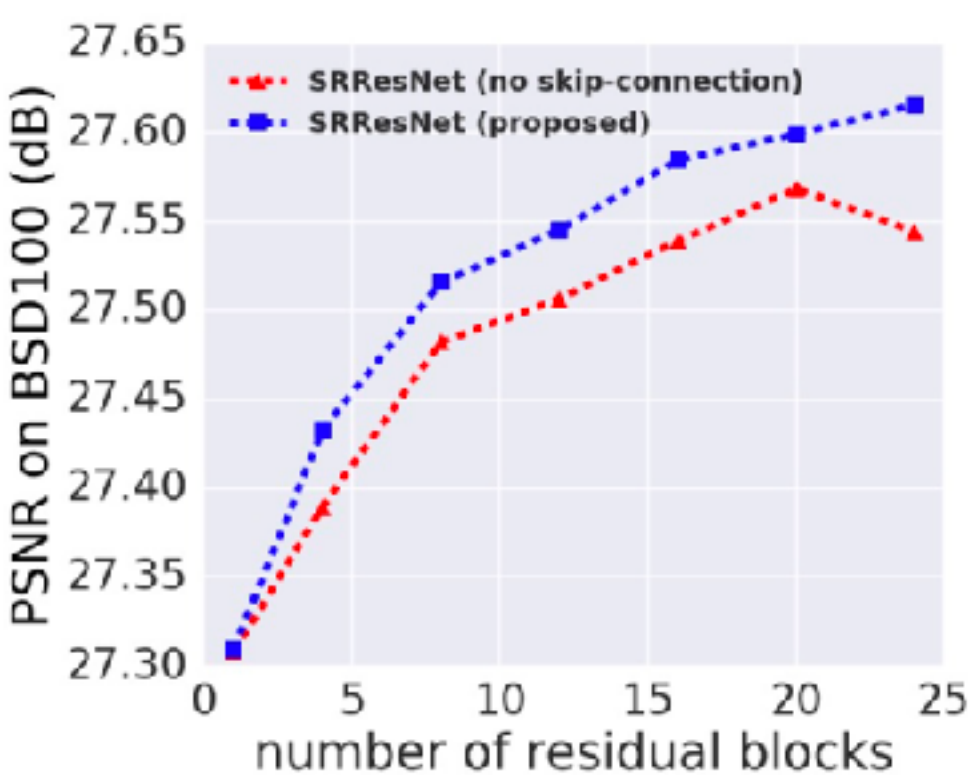
Train using
L2 loss vs
ground truth

[SRCNN, Dong et al 2014]

Super-Resolution: SRResNet

- Deeper networks generate better results, e.g., SRResNet





Trained with L2 loss, state-of-the-art PSNR in 2017

10.1

[Ledig et al 2017]

SRResNet: L2 Loss

bicubic
(21.59dB/0.6423)

SRResNet
(23.53dB/0.7832)

original

A state of the art super-res network trained with L2 loss is good at sharpening edges, but results lack realistic texture

[Ledig et al 2017]

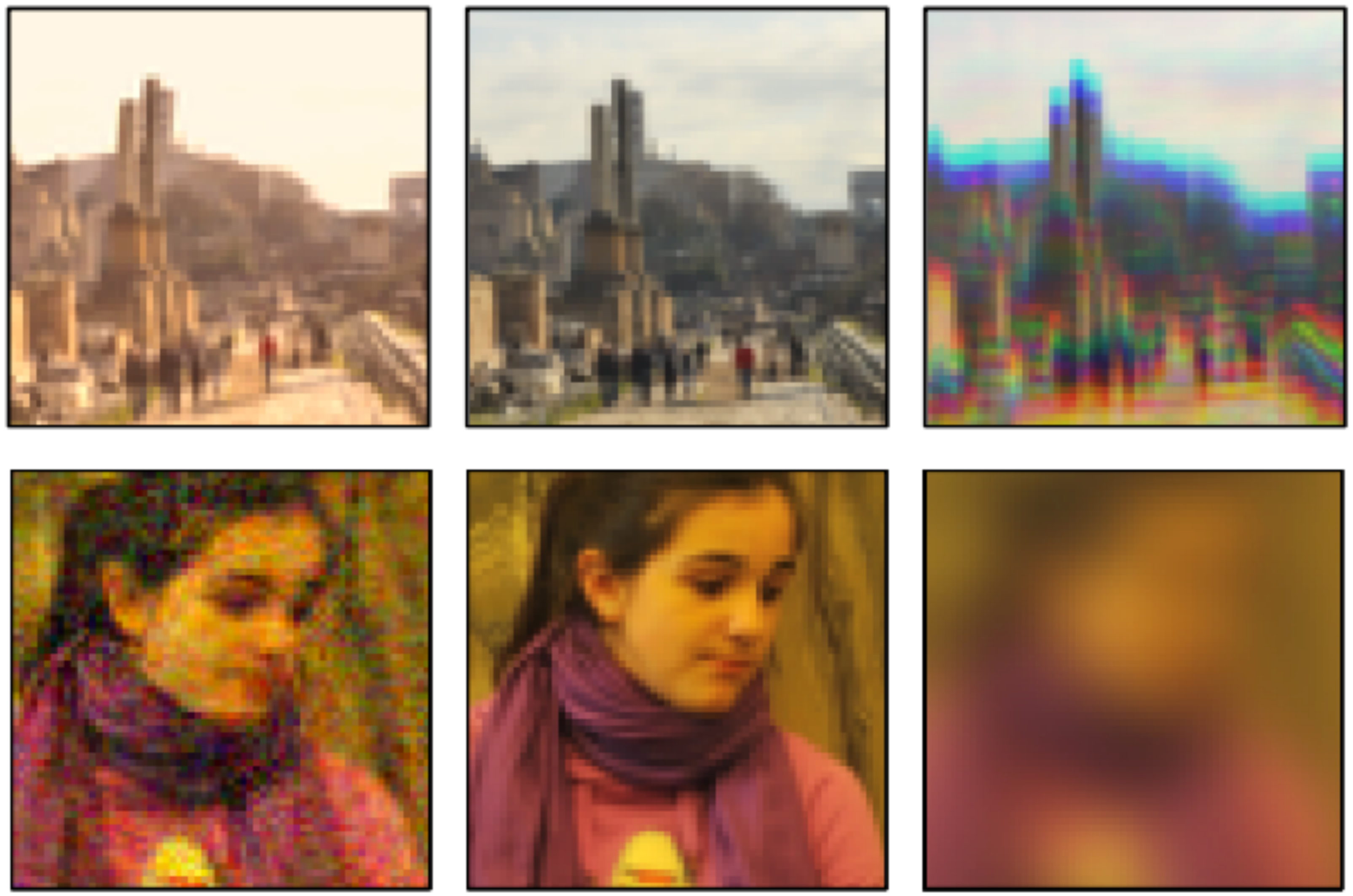
SRResNet
(23.53dB/0.7832)

SRResNet
(23.53dB/0.7832)

More realistic, but L2
loss is worse
(21.15dB/0.6868)

Perceptual Metrics

- L2 loss does not match human perception in general



Human preferred | Reference | L2 preferred

[Zhang et al 2018]

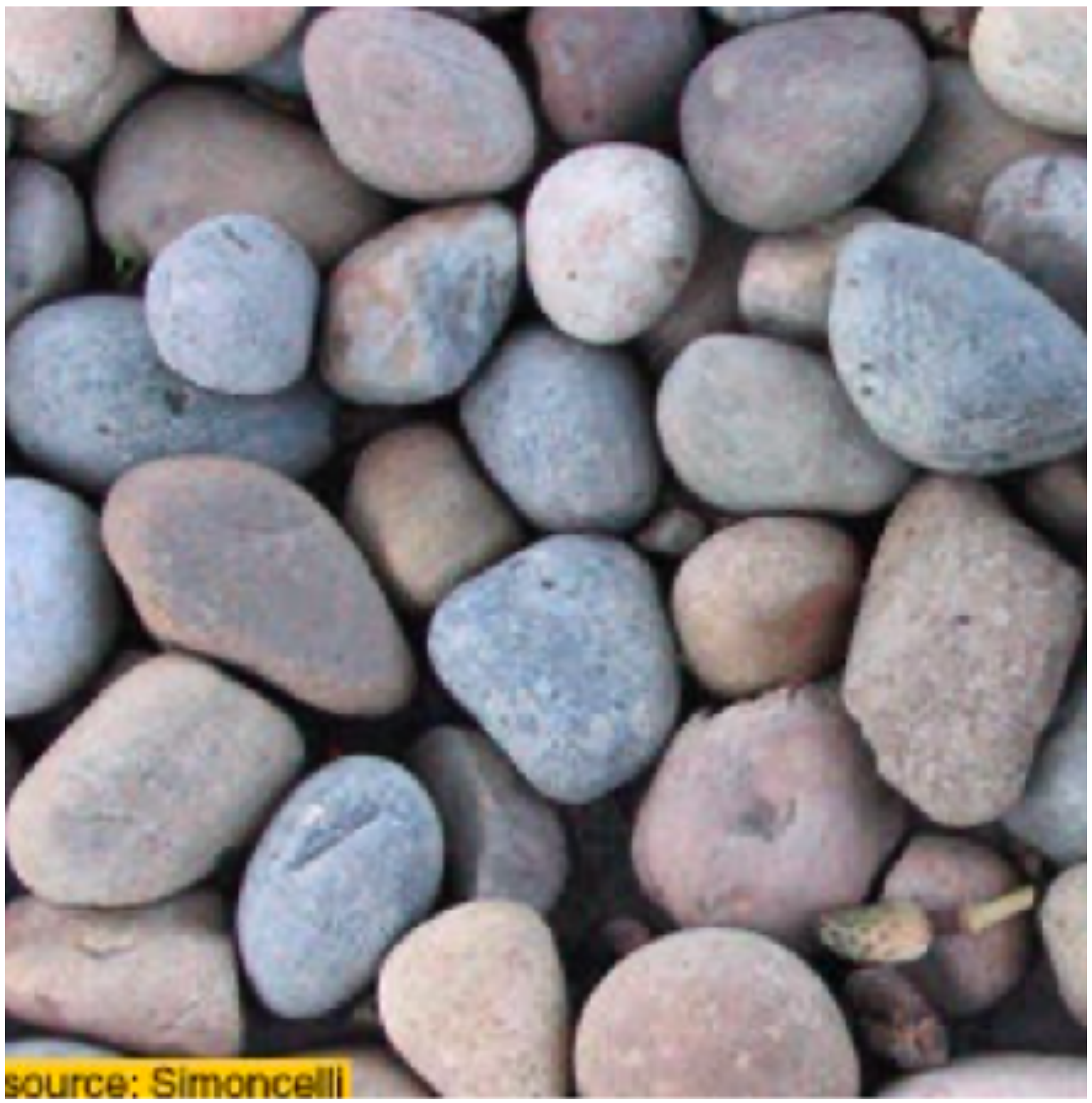
Texture Synthesis

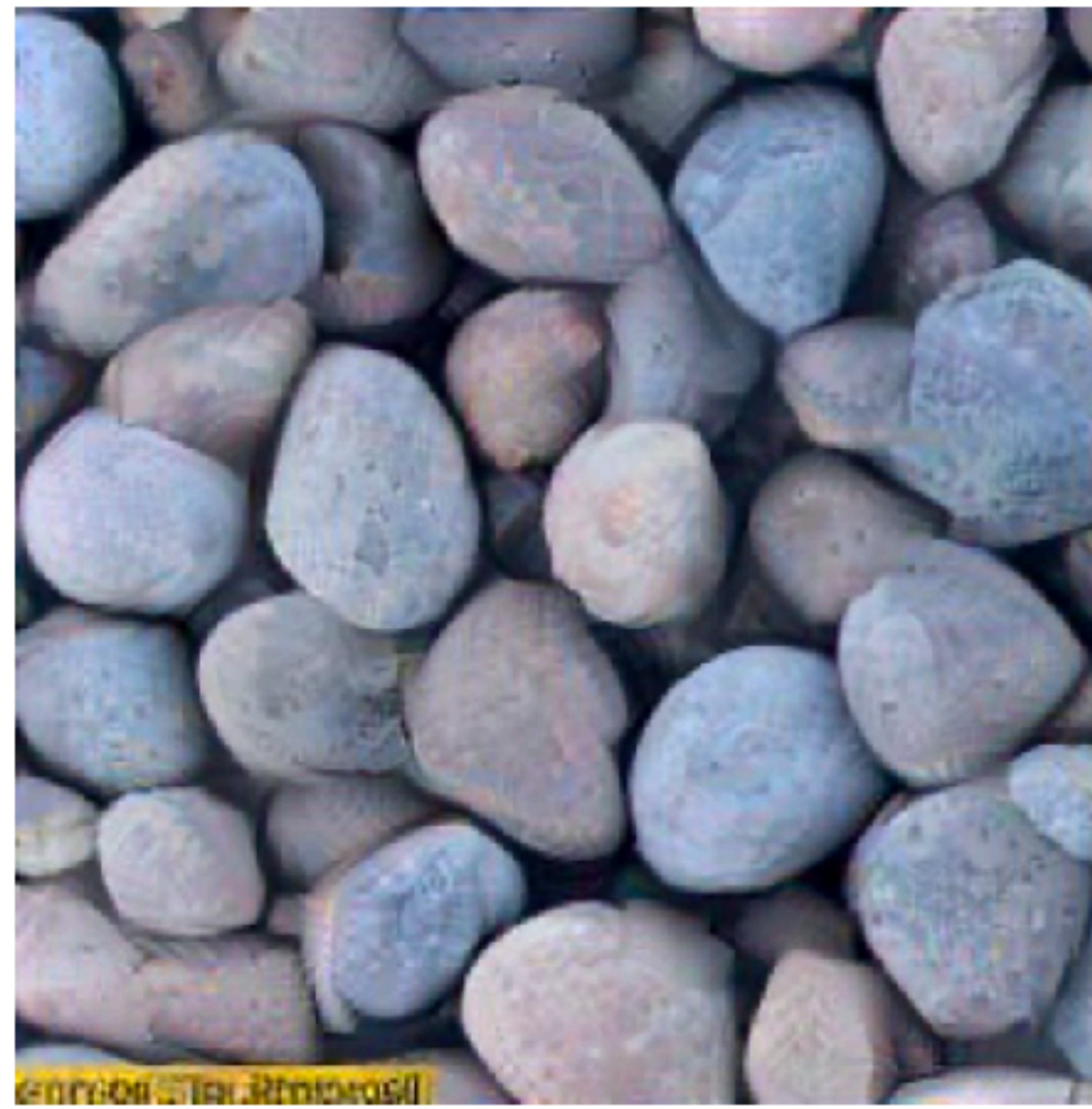
- Which are the real radishes?

[Gatys et al 2015]

Texture Synthesis

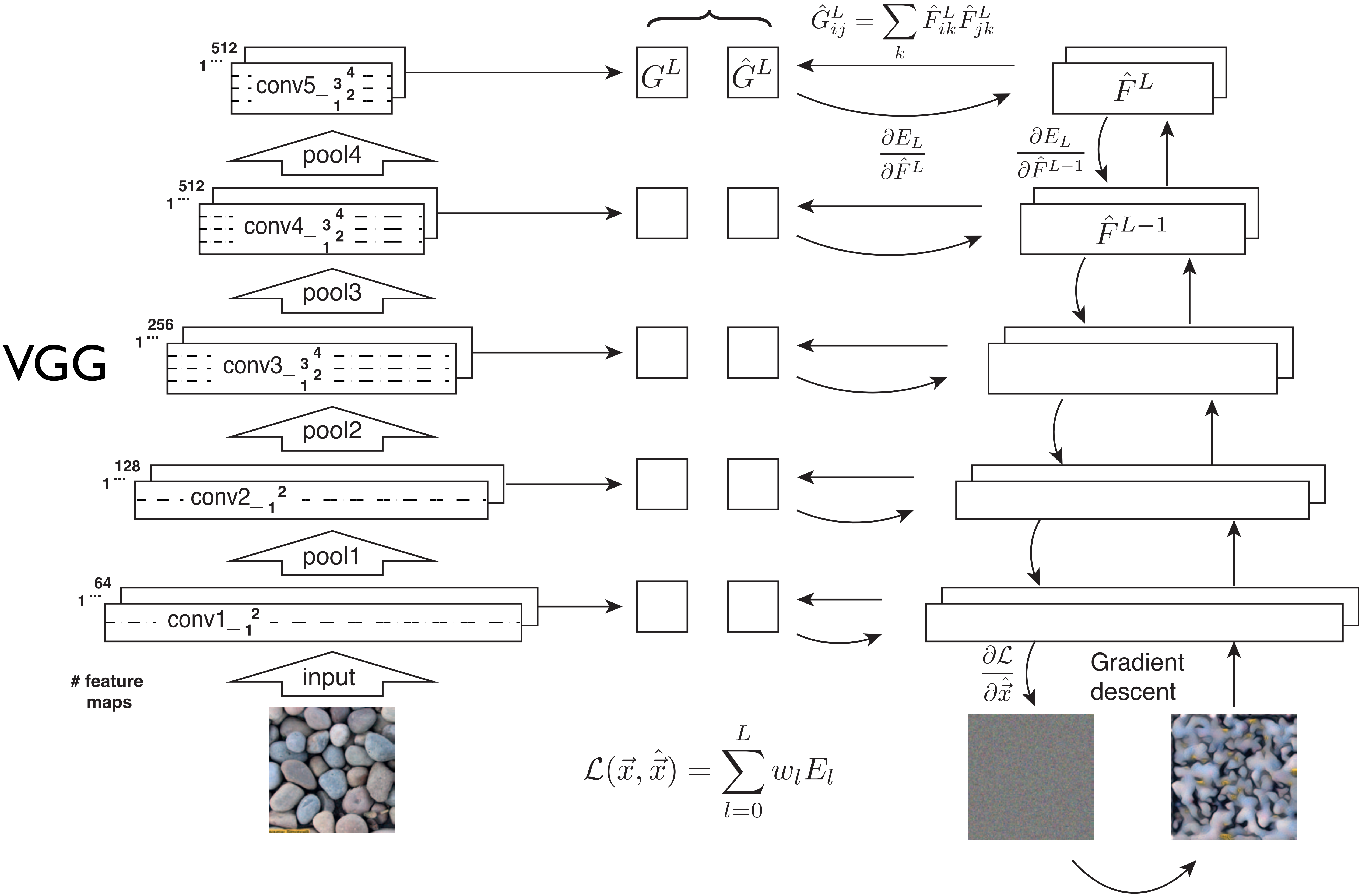
- Which are the real rocks?





[Gatys et al 2015]

loss = $E_L = \sum \left(\hat{G}^L - G^L\right)^2$ G=Gram-matrix



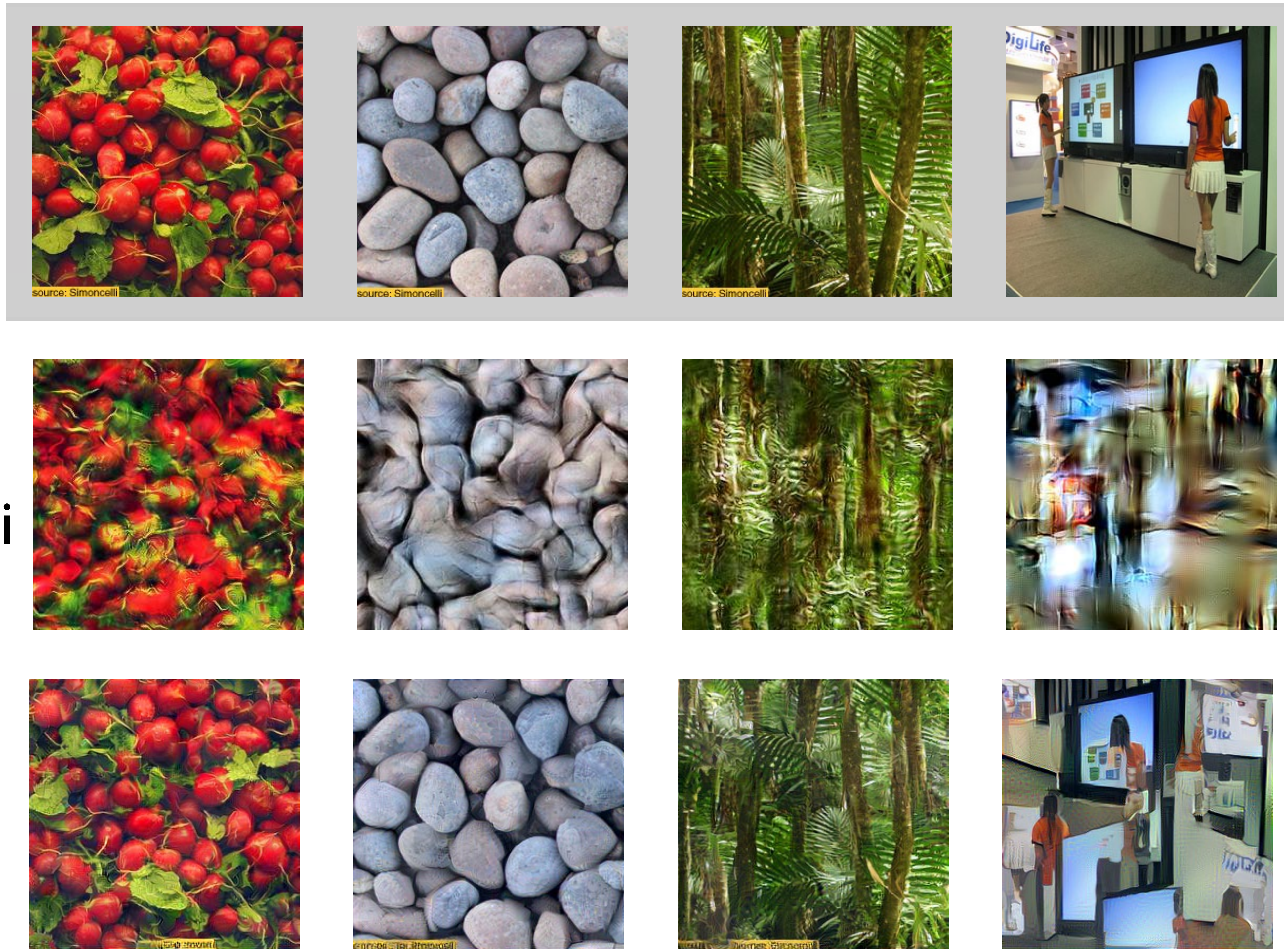
Match Σ of outer products of feature activations $\hat{\vec{x}} := \hat{\vec{x}} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$

Texture Synthesis

Original

Portilla Simoncelli

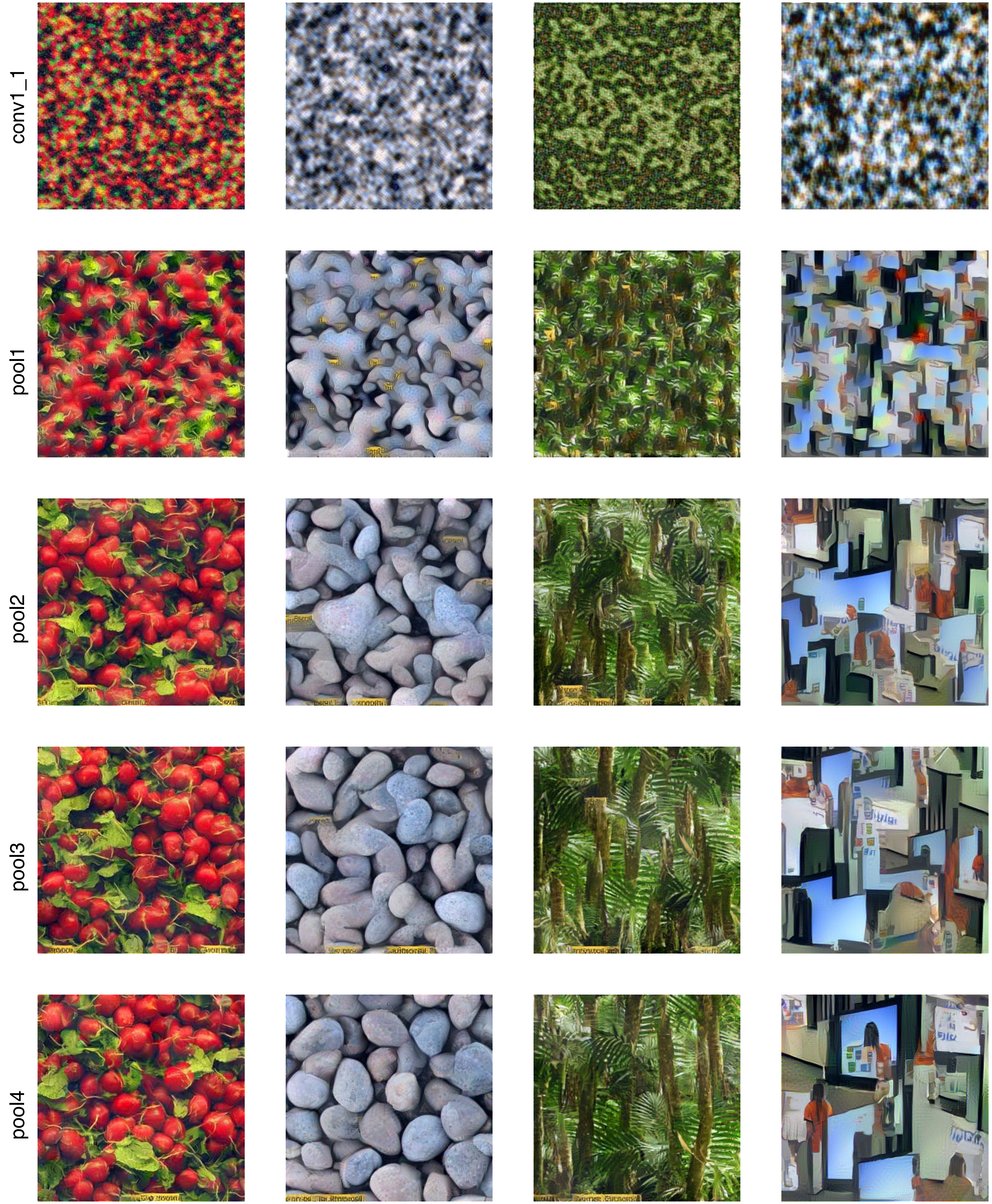
Gatys



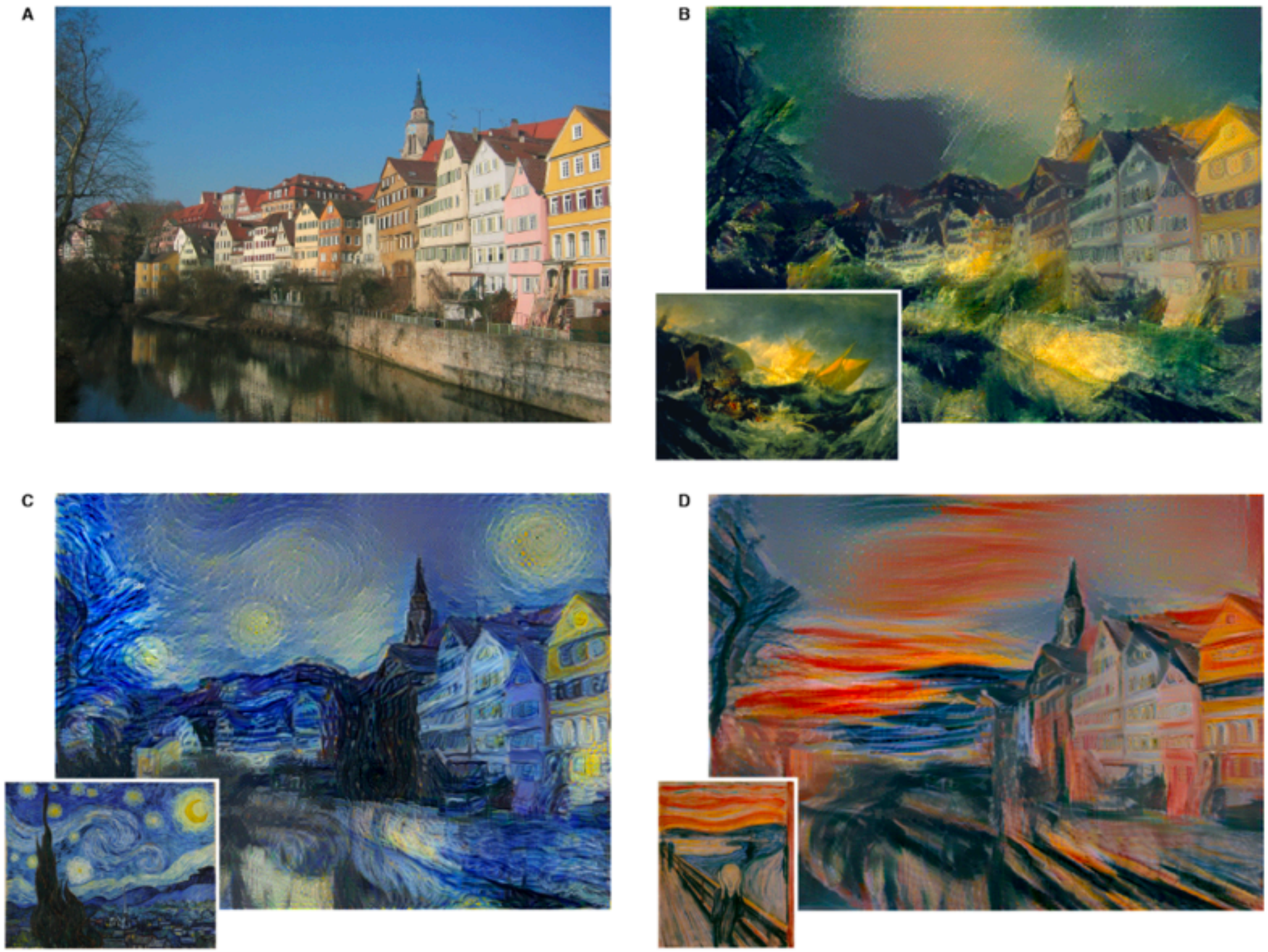
- Portilla Simoncelli 1999 texture model also used correlation of filter responses (though shallow features / hand tuned) simple features

Match texture statistics up to level =

more complex features



Style Transfer

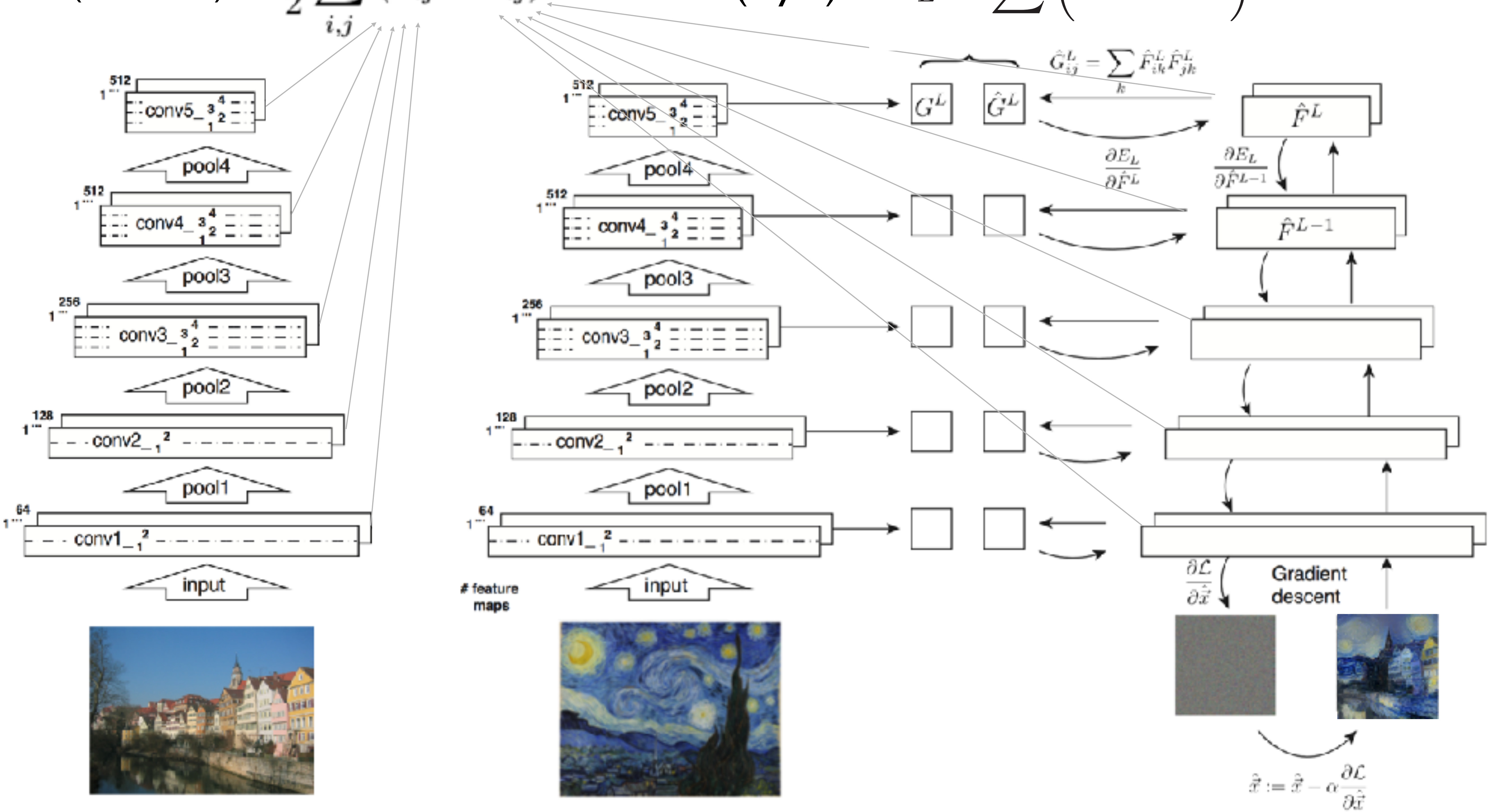


Re-render an image given the “style” of an artist [Gatys et al 2015]

Content and Style Losses

E(content) = $\frac{1}{2}\sum_{i,j}(F_{ij}^l - P_{ij}^l)^2$

E(style) = $E_L = \sum\left(\hat{G}^L - G^L\right)^2$

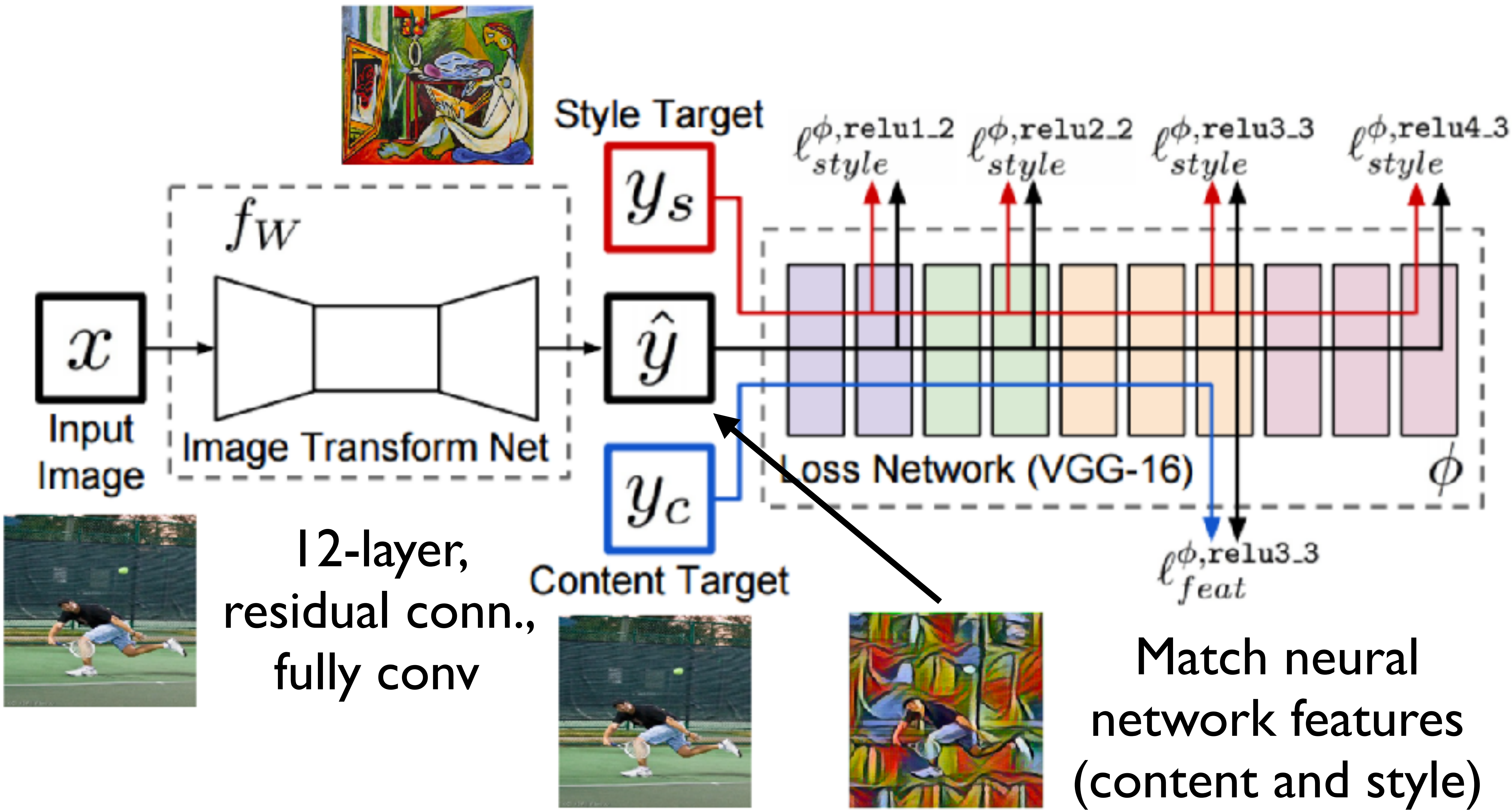


Match content

Match style

[Gatys et al 2015]

Feedforward Style Transfer



12-layer, residual conn., fully conv

Match neural network features (content and style)

[Johnson et al. 2016]

Feedforward Style Transfer

Content Gatys Johnson Content Gatys Johnson

Comparable results to with much lower computational costs (single feedforward pass vs 1000s of backprop iterations)

Super-Resolution

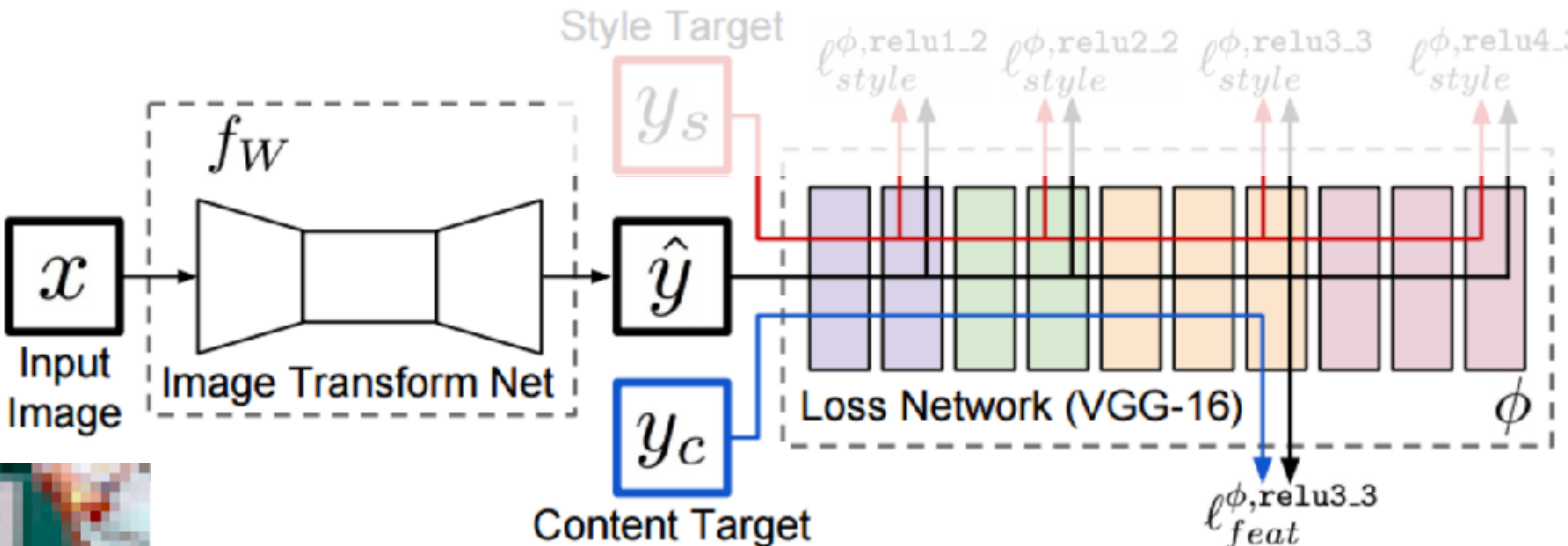
- Small networks are generally good at sharpening edges and can work well for small factor (e.g., 2) super-resolution
- Better results can be achieved by using deeper networks, + more sophisticated loss functions (perceptual loss, GANs)

Original   Bicubic   SRCNN   Johnson et al*

*12-layer, residual conn., fully conv, VGG loss   [Johnson et al. 2016]

Super-Resolution with VGG loss

Note: style loss not used for super-res



Match neural net features (content loss)

Low-res input

High-res target

[Johnson et al. 2016]

Super-Resolution with VGG loss

- Results

Ground Truth	Bicubic	SRCNN	Johnson et al. VGG loss

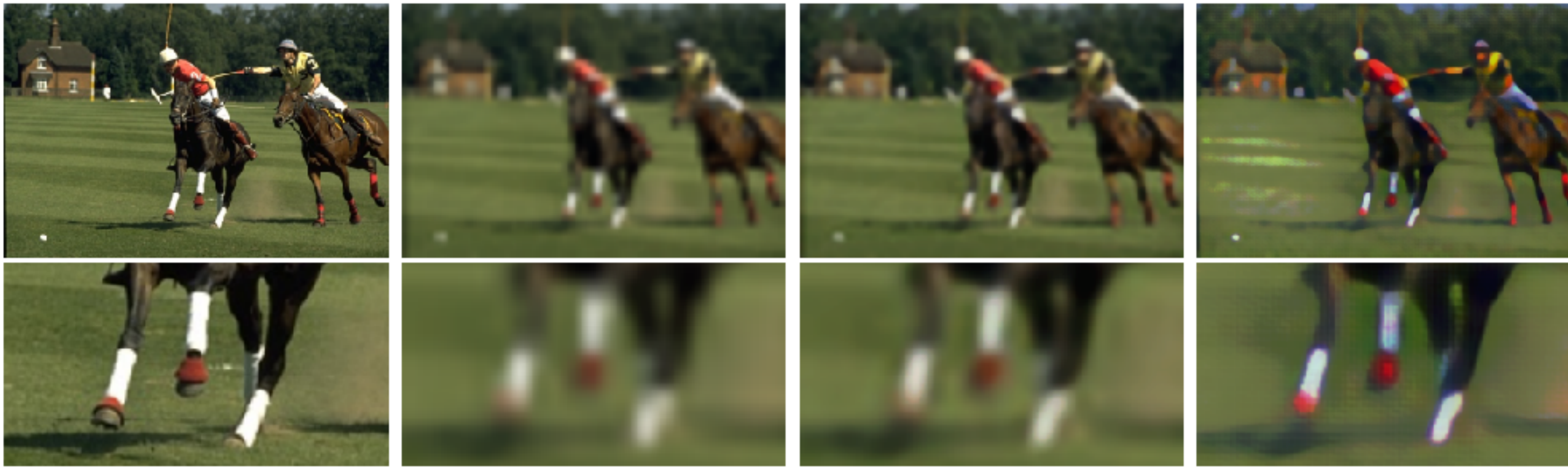
[Johnson et al. 2016]

Super-Resolution with VGG loss

SRCNN

Johnson

Super-Resolution with VGG loss

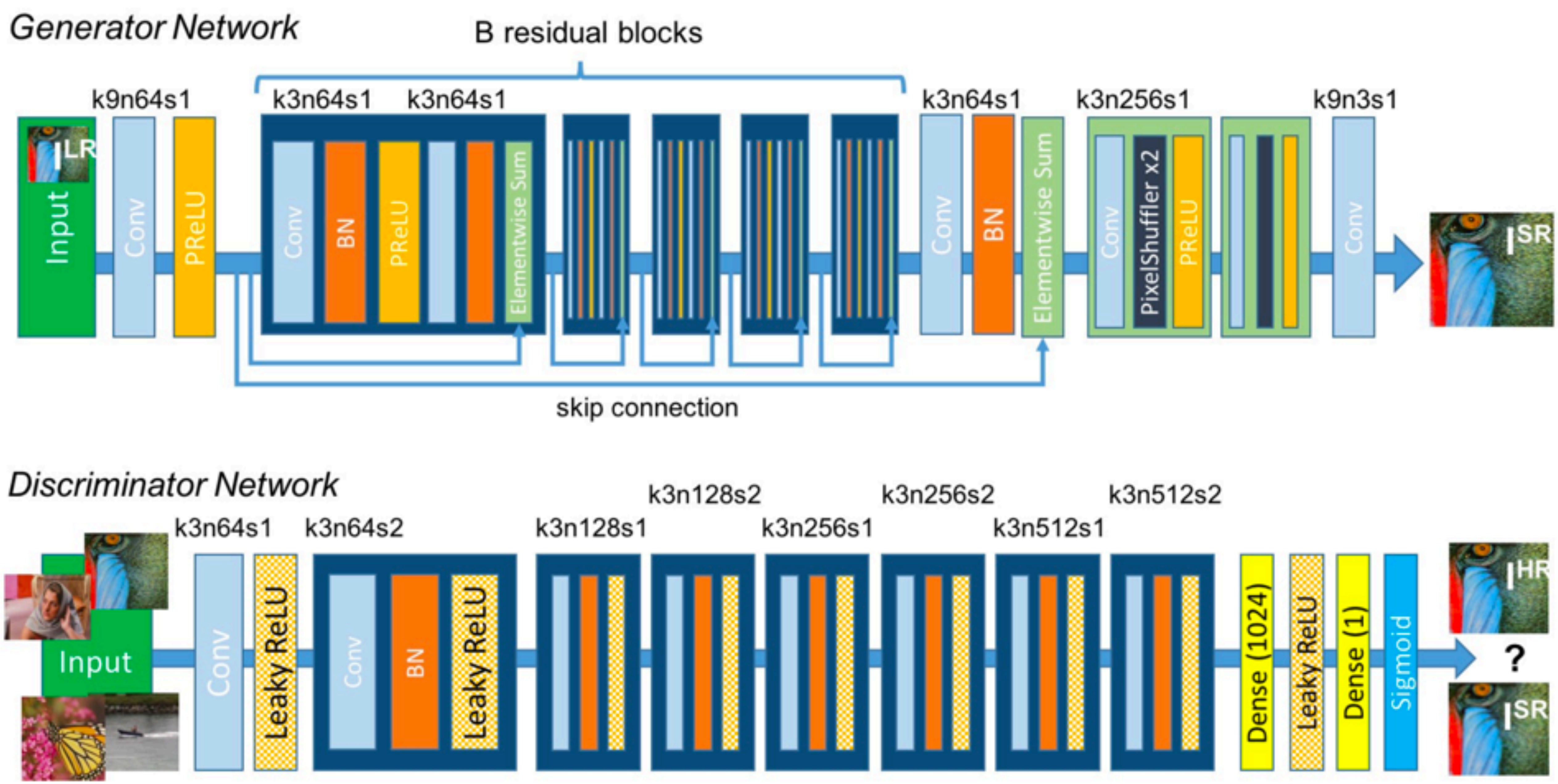


Ground Truth	Bicubic	Ours (ℓ_{pixel})	Ours (ℓ_{feat})
This image	22.75 / 0.5946	23.42 / 0.6168	21.90 / 0.6083
Set5 mean	23.80 / 0.6455	24.77 / 0.6864	23.26 / 0.7058
Set14 mean	22.37 / 0.5518	23.02 / 0.5787	21.64 / 0.5837
BSD100 mean	22.11 / 0.5322	22.54 / 0.5526	21.35 / 0.5474

[Johnson et al. 2016]

SRGAN

- Can we train a neural network to define the loss?



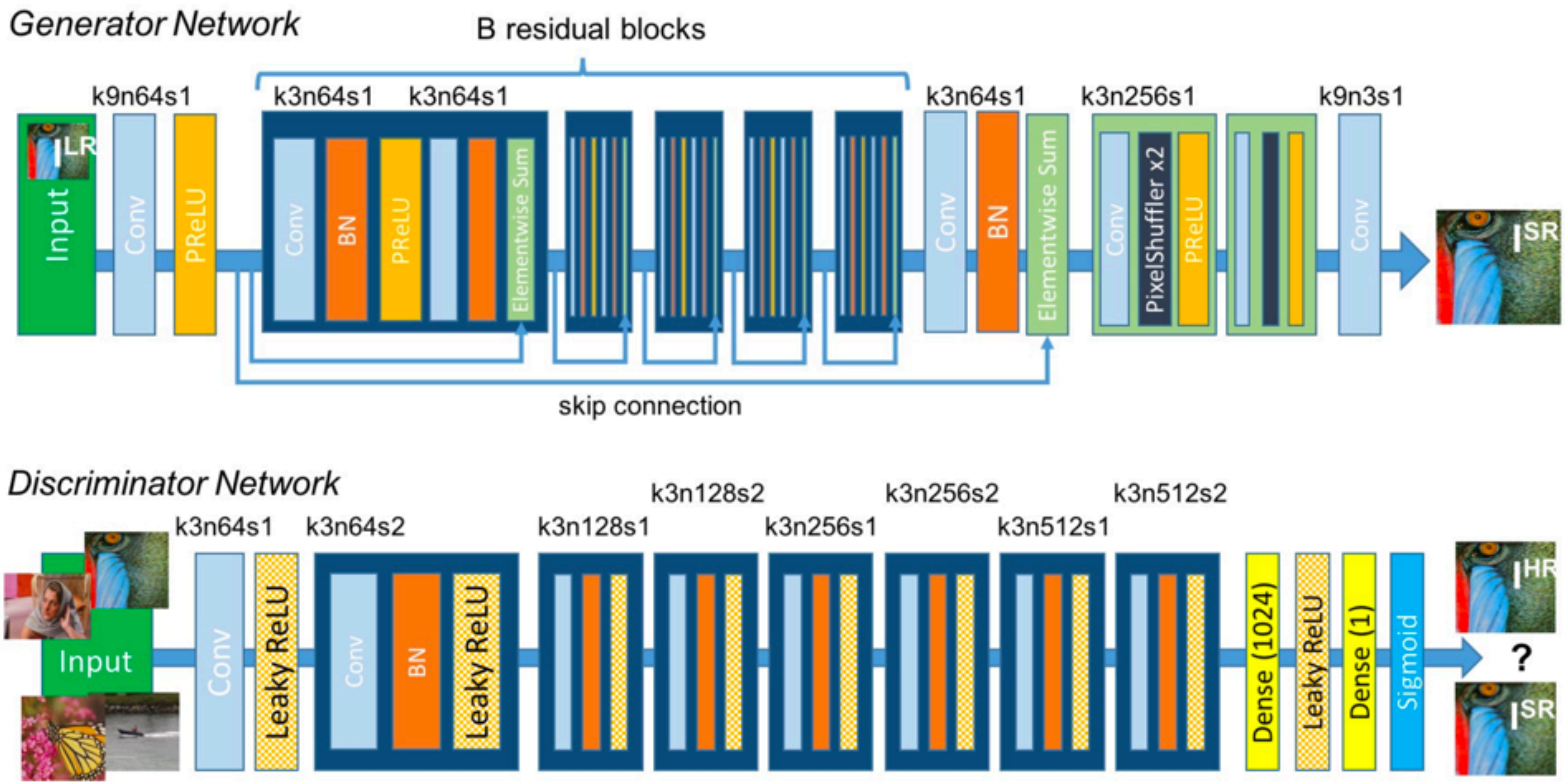
[Ledig et al. 2017]

SRResNet
(23.53dB/0.7832)

SRGAN
(21.15dB/0.6868)

SRGAN

Generator performs super-resolution, tries to fool discriminator



Discriminator tries to detect real vs super-resolved images

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.

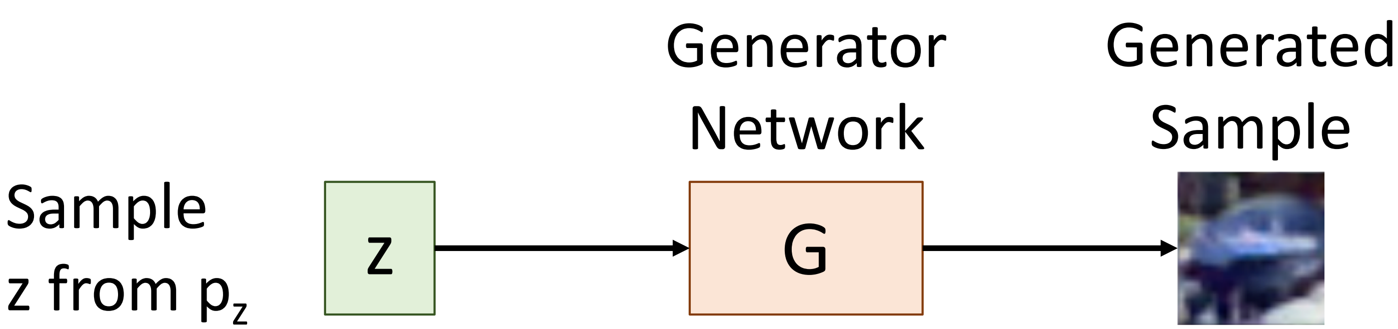
Idea: Introduce a latent variable z with simple prior p(z).
Sample $z \sim p(z)$ and pass to a **Generator Network** x = G(z)
Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.

Idea: Introduce a latent variable z with simple prior p(z).
Sample $z \sim p(z)$ and pass to a **Generator Network** x = G(z)
Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!



Train **Generator Network** G to convert z into fake data x sampled from p_G

Goodfellow et al, “Generative Adversarial Nets”, NeurIPS 2014

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.

Idea: Introduce a latent variable z with simple prior p(z).
Sample $z \sim p(z)$ and pass to a **Generator Network** x = G(z)
Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!

Jointly train G and D. Hopefully p_G converges to p_{data}!

Train **Generator Network** G to convert z into fake data x sampled from p_G **by "fooling" the discriminator D**

Train **Discriminator Network** D to classify data as real or fake (1/0)

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

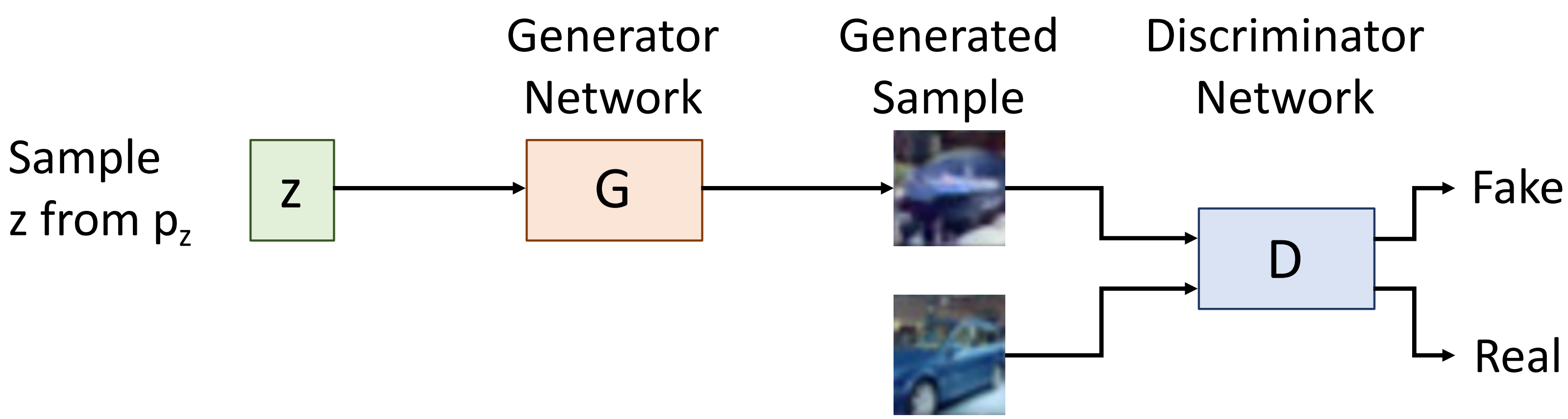
$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D\big(G(z)\big) \right) \right] \right)$$

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

$$\min_{G} \max_{D} \left(\overbrace{E_{x \sim p_{data}} [\log D(x)]}^{\text{Discriminator wants } D(x) = 1 \text{ for real data}} + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

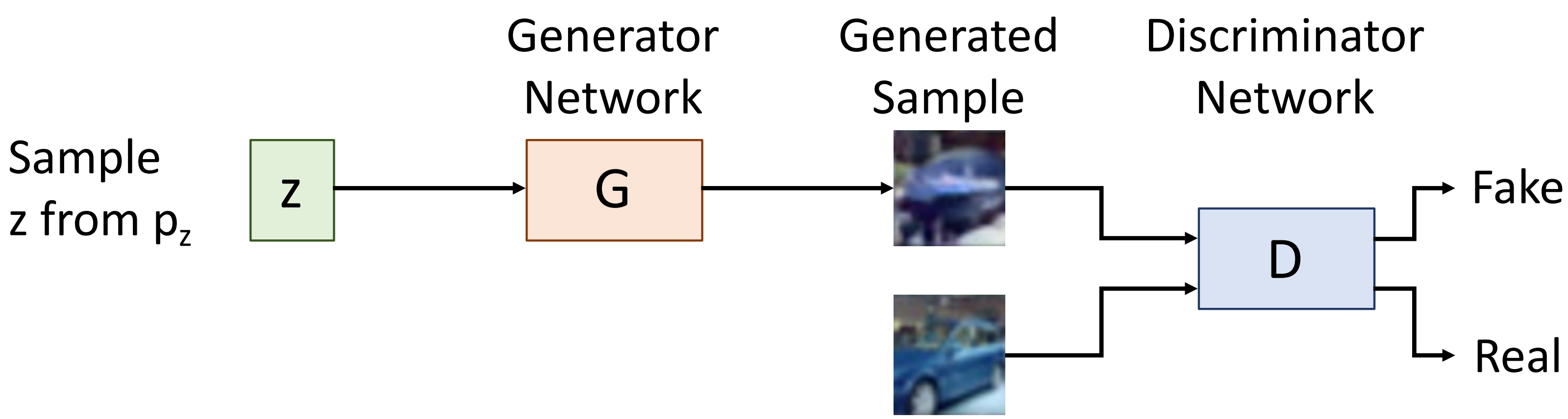


Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

$$\min_{G} \max_{D} \left(\overbrace{E_{x \sim p_{data}} [\log D(x)]}^{\text{Discriminator wants } D(x) = 1 \text{ for real data}} + E_{z \sim p(z)} \overbrace{\left[\log \left(1 - D(G(z)) \right) \right]}^{\text{Discriminator wants } D(x) = 0 \text{ for fake data}} \right)$$

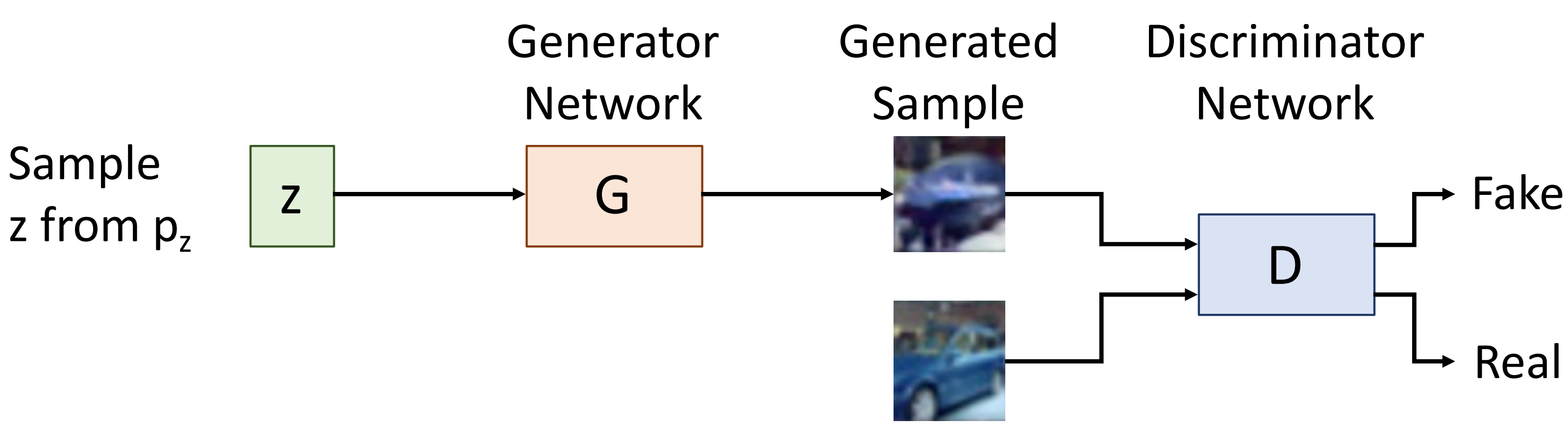


Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

$$\min_{G} \max_{D} \left(\overbrace{E_{x\sim p_{data}}[\log D(x)]}^{\substack{\text{Discriminator wants} \\ \text{D(x) = 1 for real data}}} + E_{z\sim p(z)} \underbrace{\overbrace{\Big[\log\big(1 - D(G(z))\big)\Big]}^{\substack{\text{Discriminator wants} \\ \text{D(x) = 0 for fake data}}}}_{\substack{\text{Generator wants} \\ \text{D(x) = 1 for fake data}}} \right)$$



Goodfellow et al, “Generative Adversarial Nets”, NeurIPS 2014

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

Train G and D using alternating gradient updates

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_{G} \max_{D} V(G, D)$$

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

Train G and D using alternating gradient updates

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_{G} \max_{D} V(G, D)$$

We are not minimizing any overall loss! No training curves to look at!

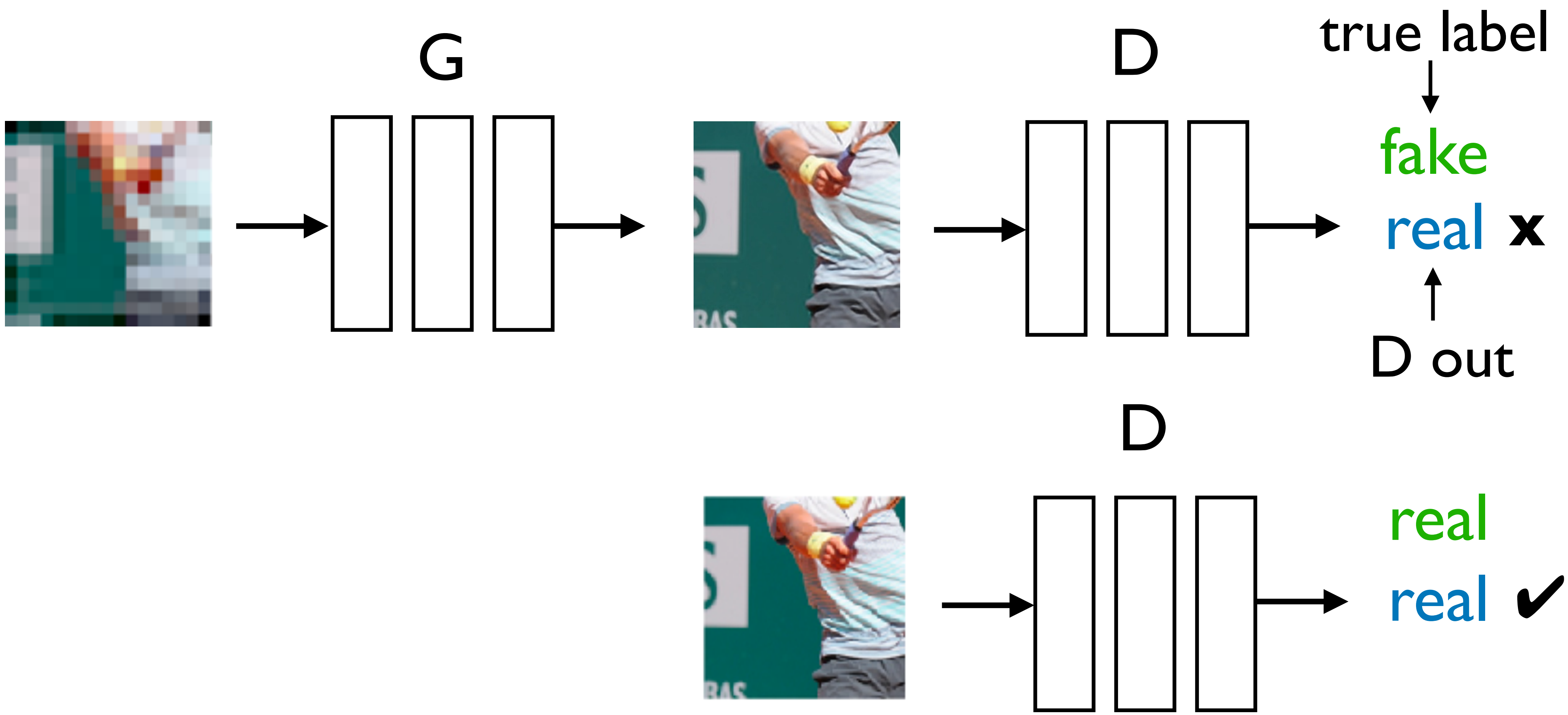
For t in 1, … T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

SRGAN

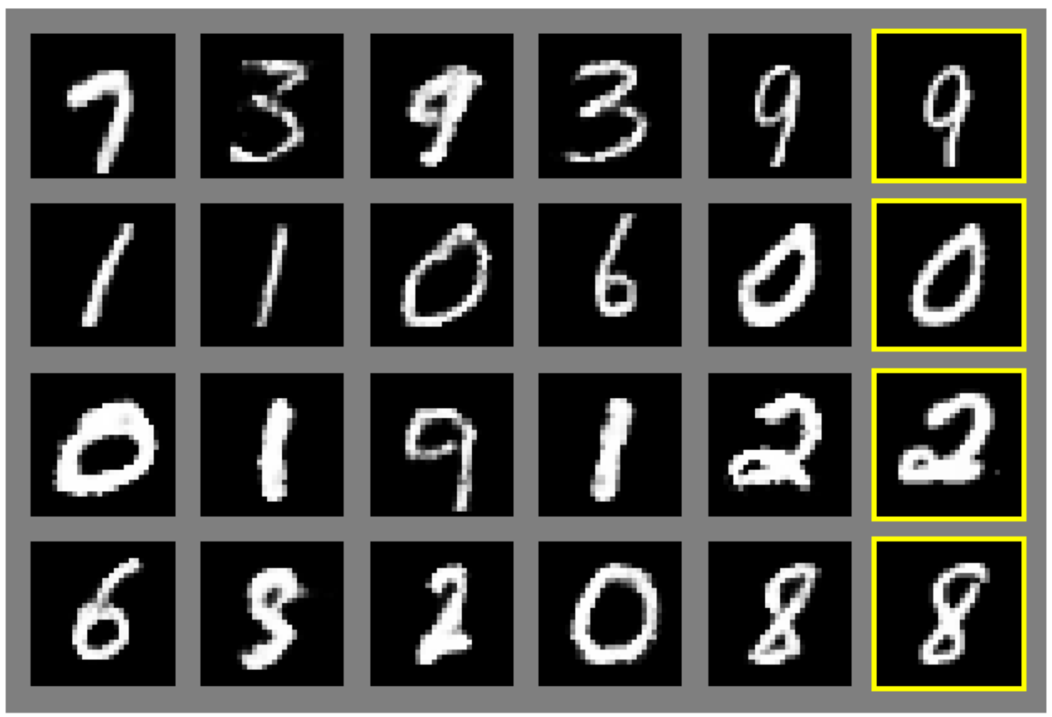
- Generator performs superres, discriminator attempts to detect real images vs super-resolved low res images



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

Generative Adversarial Networks: Results

Generated samples

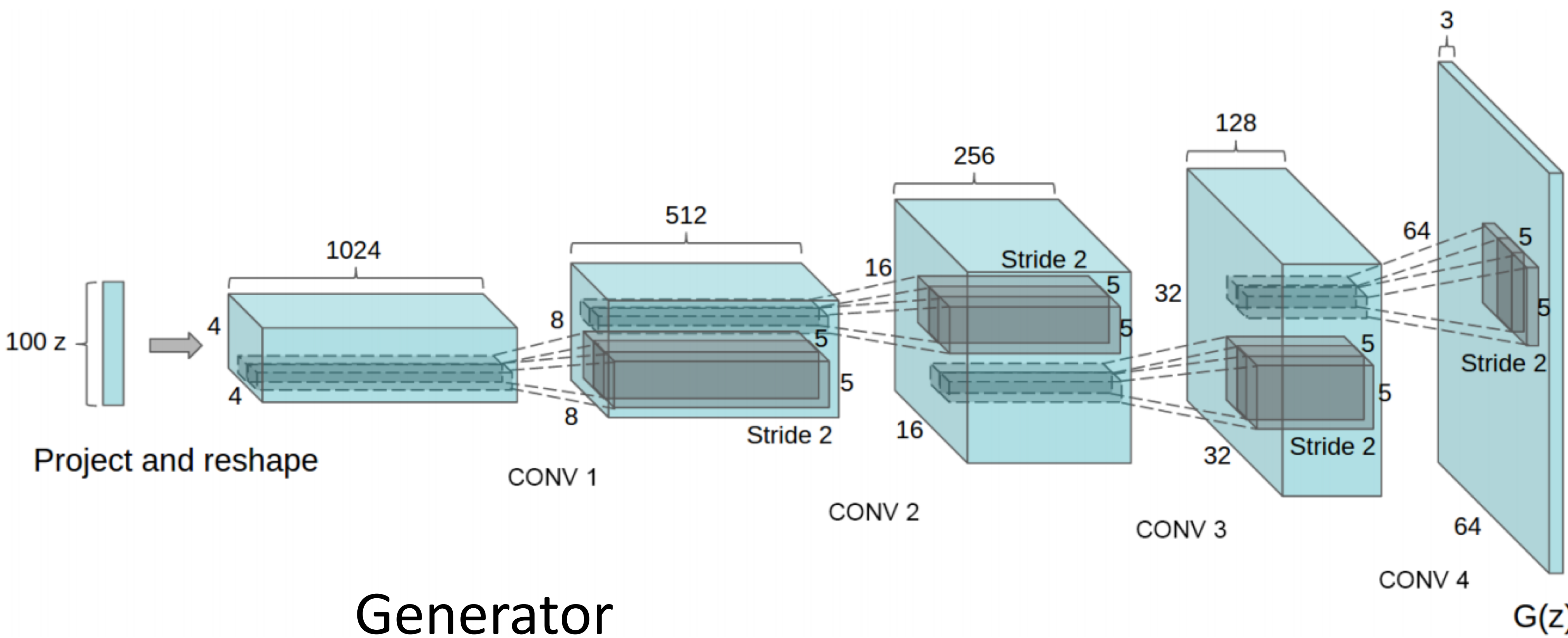


Nearest neighbor from training set

Goodfellow et al, “Generative Adversarial Nets”, NeurIPS 2014

Original GAN paper, image generation, driven by noise input

Generative Adversarial Networks: DC-GAN



Generator

Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

Generative Adversarial Networks: DC-GAN

Samples from the model look much better!

Radford et al, ICLR 2016

Generative Adversarial Networks: Interpolation

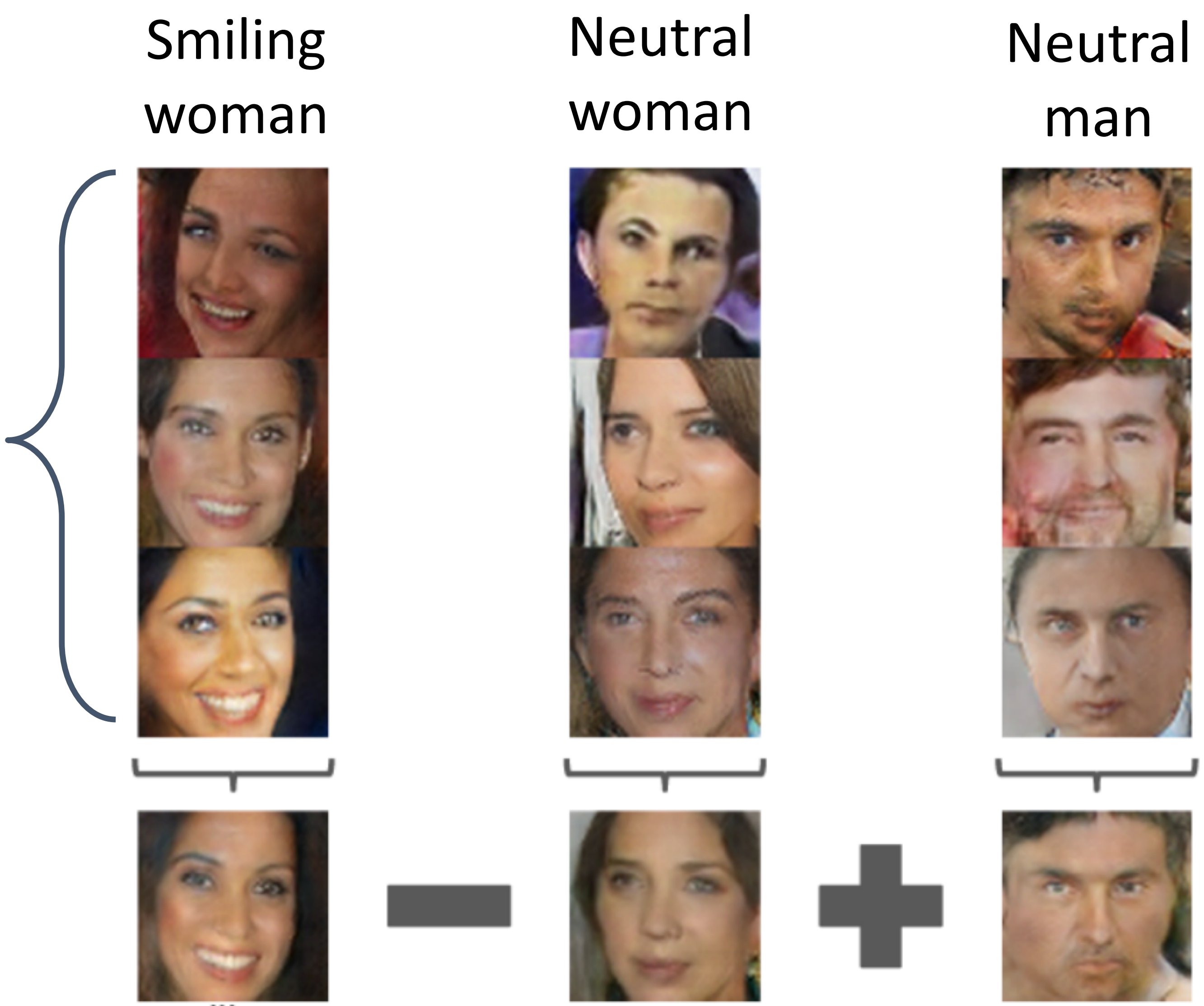
Interpolating between points in latent z space

Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

Samples from the model

Average Z vectors, do arithmetic

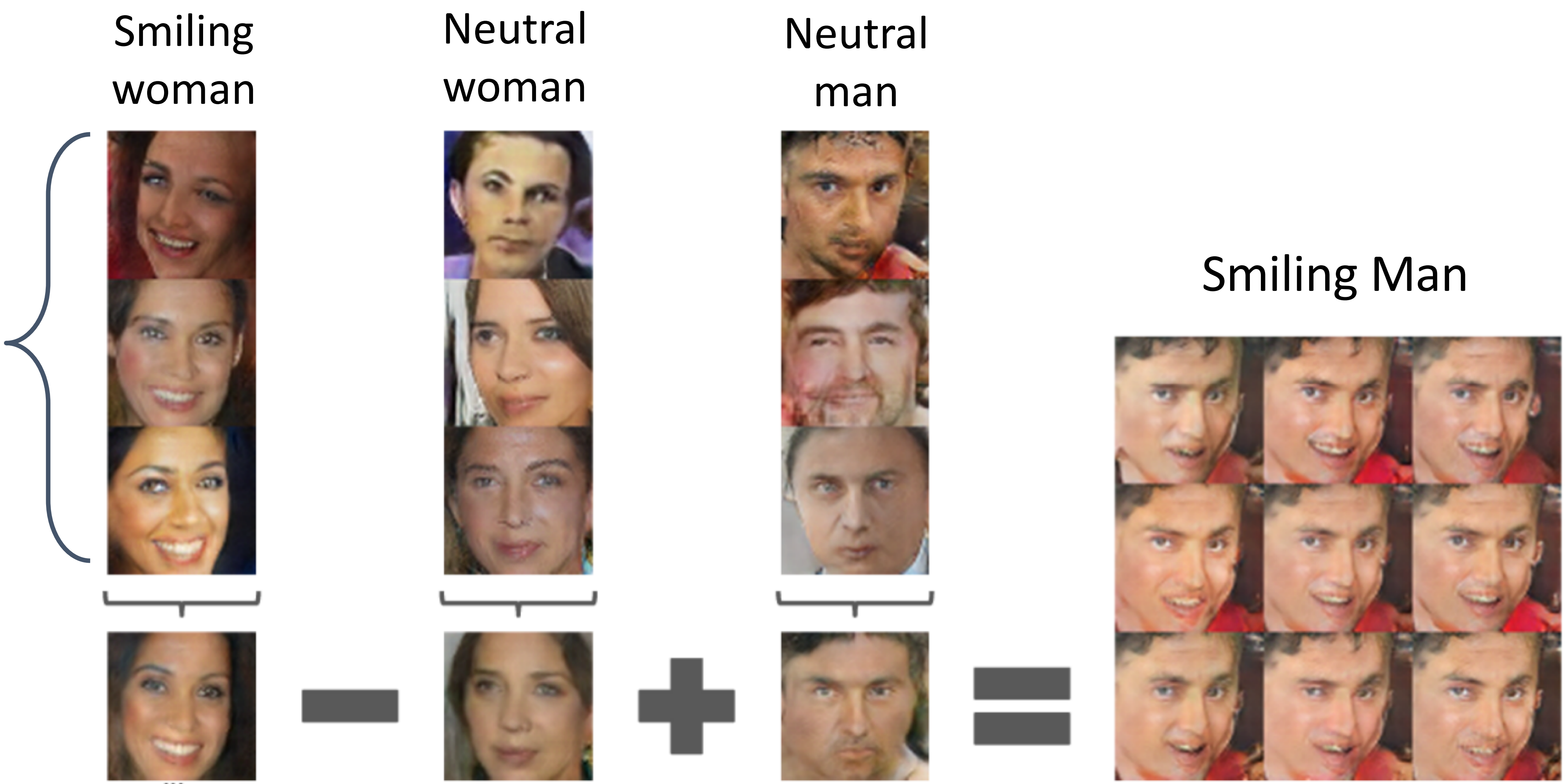


Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

Samples from the model

Average Z vectors, do arithmetic

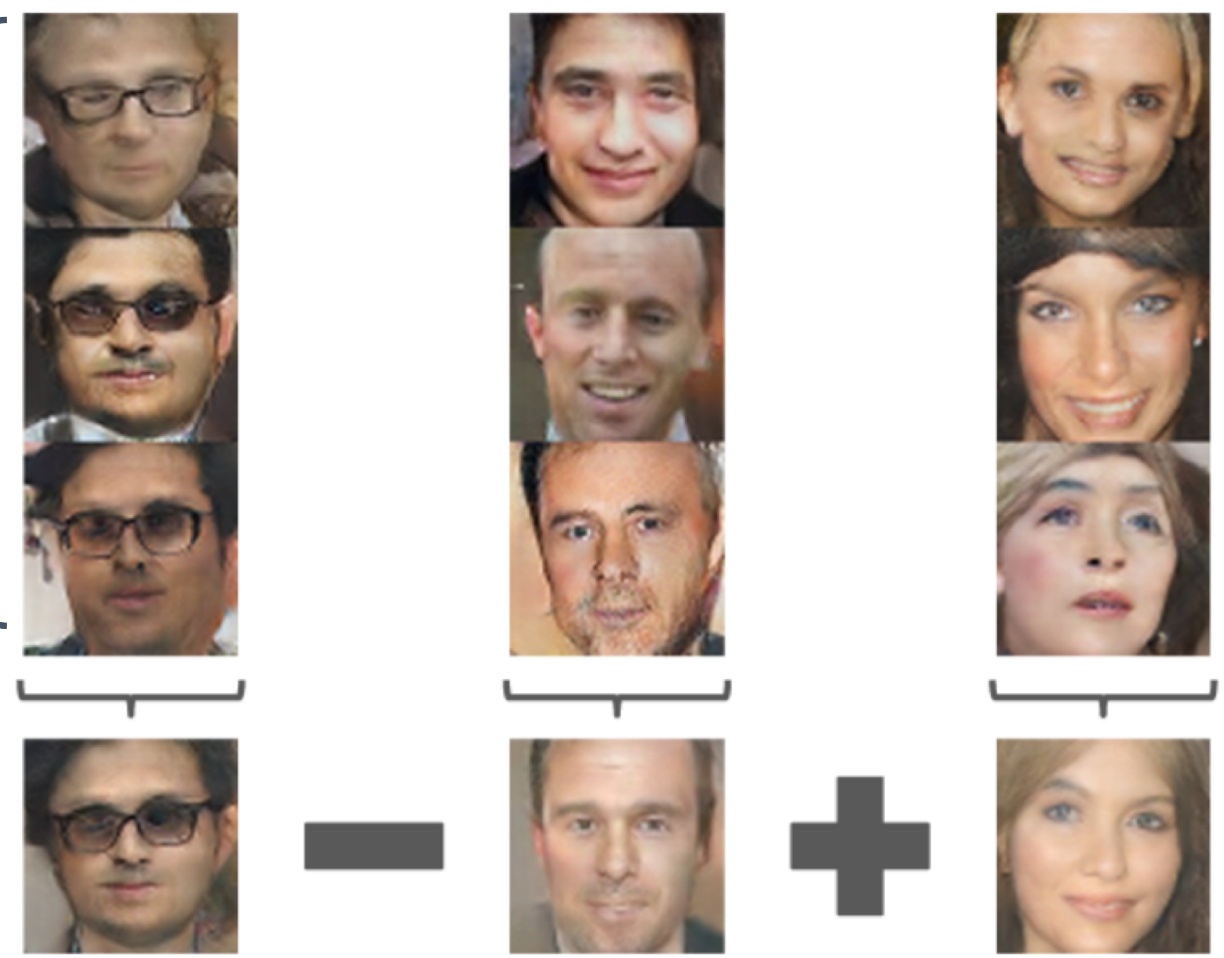


Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

Man with glasses | Man w/o glasses | Woman w/o glasses

Samples from the model



Average Z vectors, do arithmetic

Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

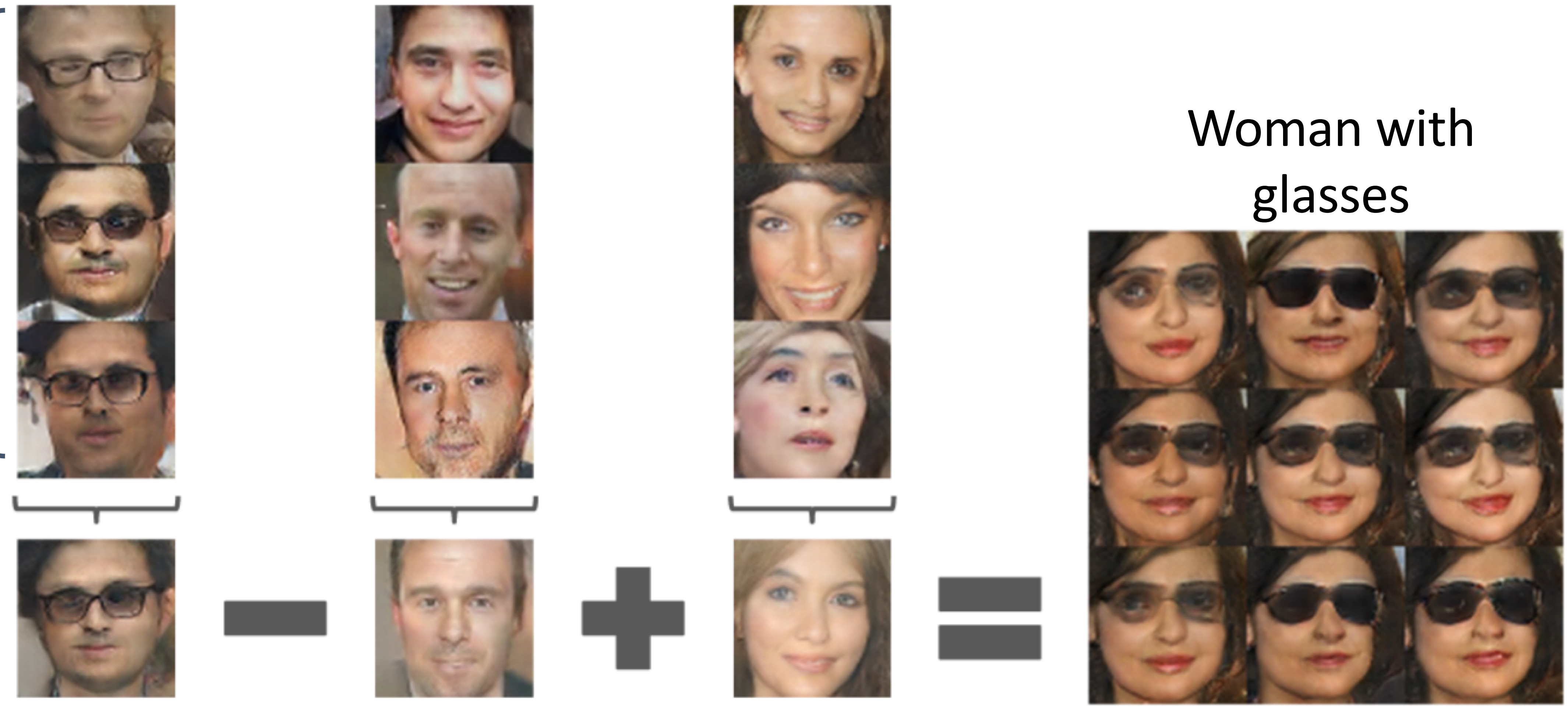
Man with glasses

Man w/o glasses

Woman w/o glasses

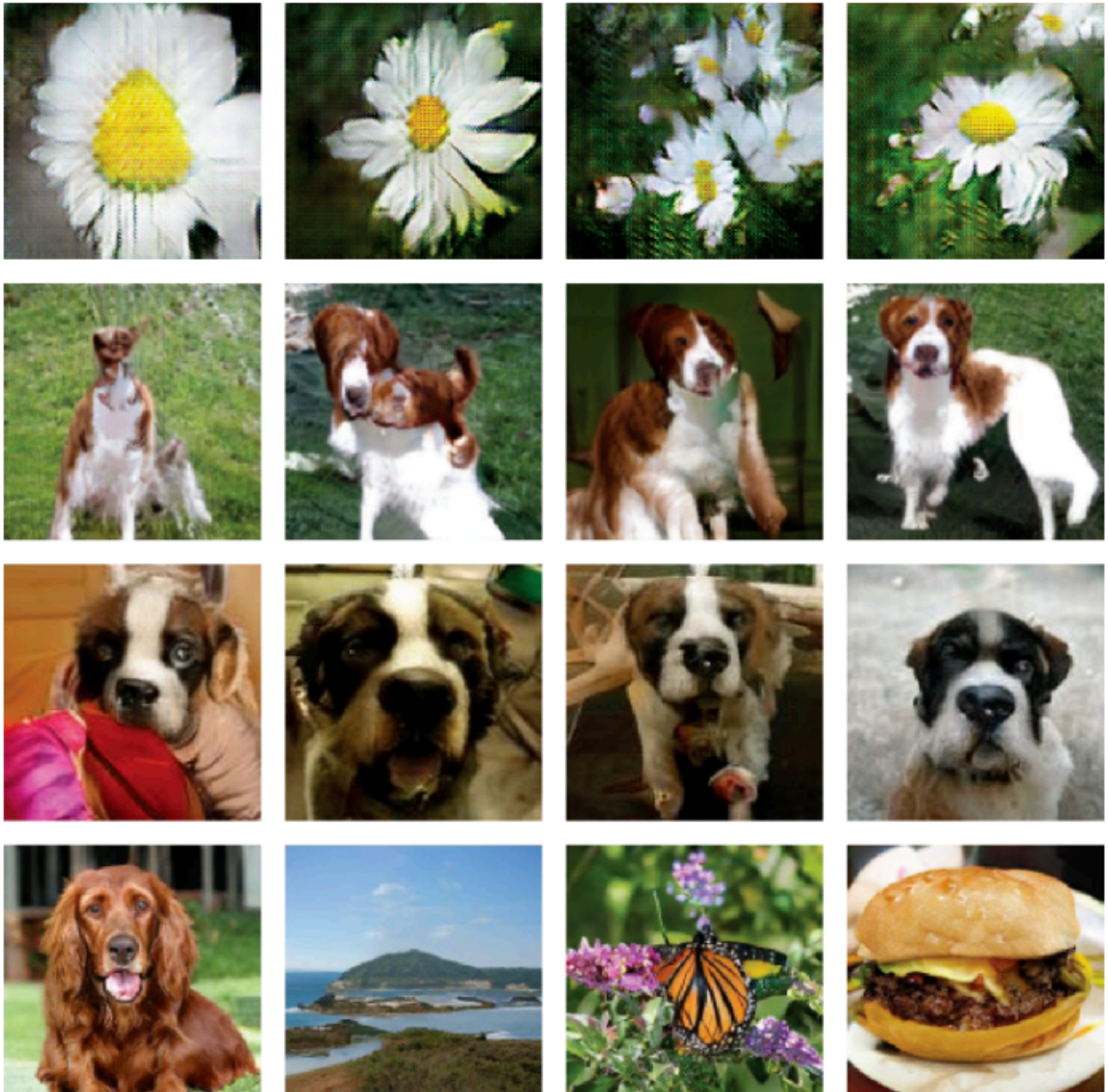
Samples from the model

Average Z vectors, do arithmetic



Woman with glasses

Radford et al, ICLR 2016



Odena et al., 2016 [1]

Miyato et al., 2017 [3]

Zhang et al., 2018 [2]

Brock et al., 2018 [4]

BigGAN

[Odena 2019]

[StyleGAN, Karras et al 2019]

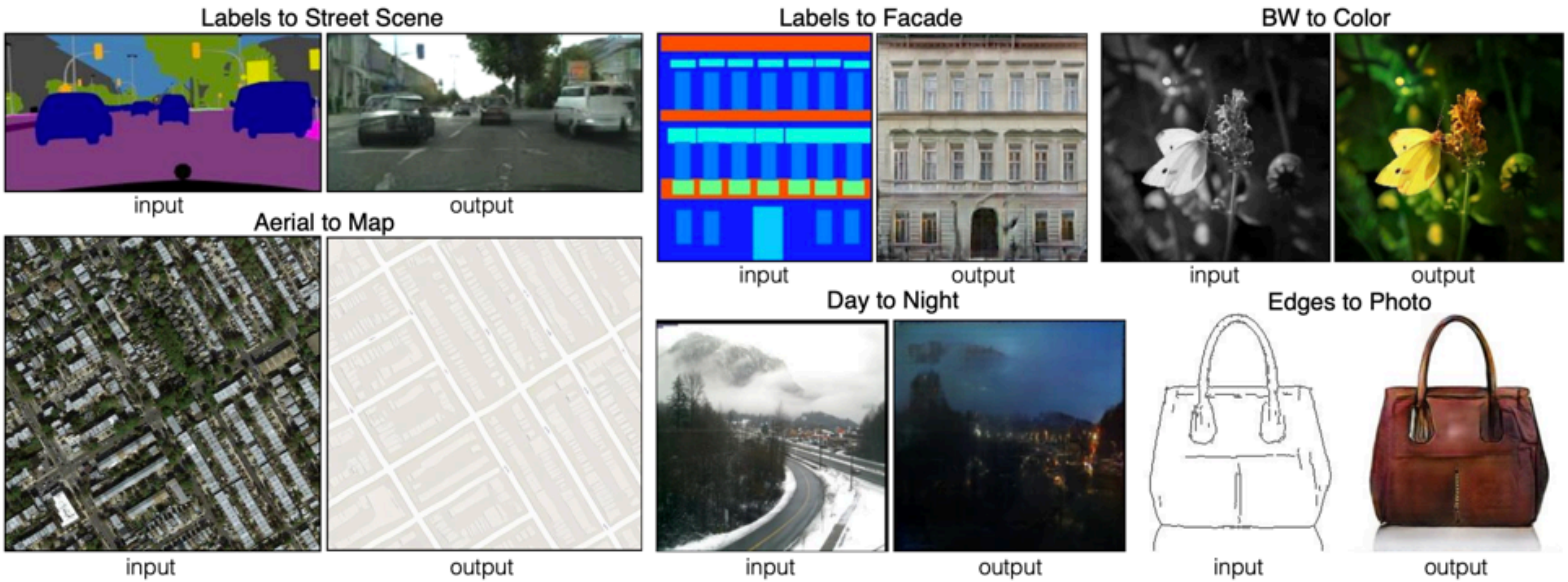
GAN Questions

Problem 1	What are the trade-offs between GANs and other generative models?
Problem 2	What sorts of distributions can GANs model?
Problem 3	How can we Scale GANs beyond image synthesis?
Problem 4	What can we say about the global convergence of the training dynamics?
Problem 5	How should we evaluate GANs and when should we use them?
Problem 6	How does GAN training scale with batch size?
Problem 7	What is the relationship between GANs and adversarial examples?

https://distill.pub/2019/gan-open-problems/

Image Translation

- Many problems in vision/graphics can be viewed as image translation problems

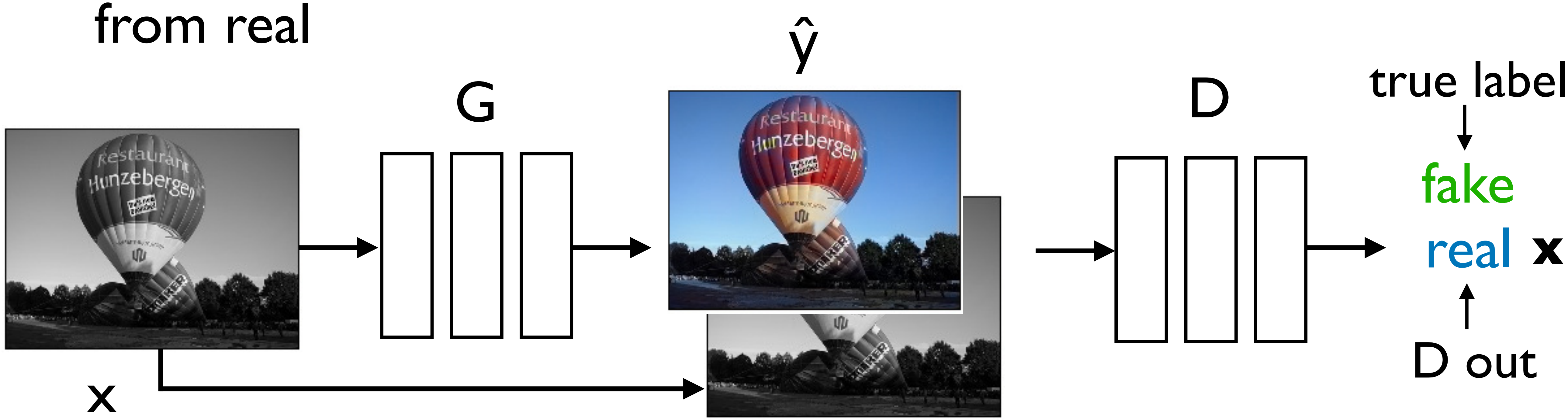


Can we build a general machine to translate images?

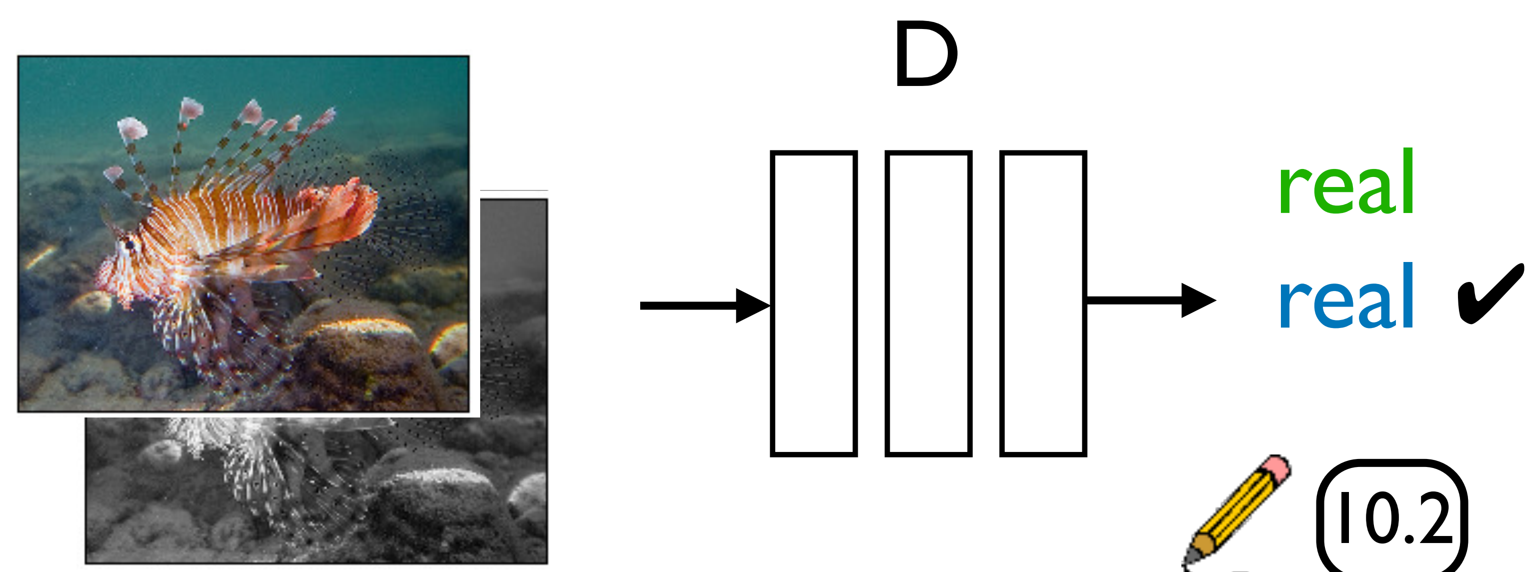
[pix2pix, Isola et al. 2018]

Image Translation

- e.g., translation from grey to color should be indistinguishable from real

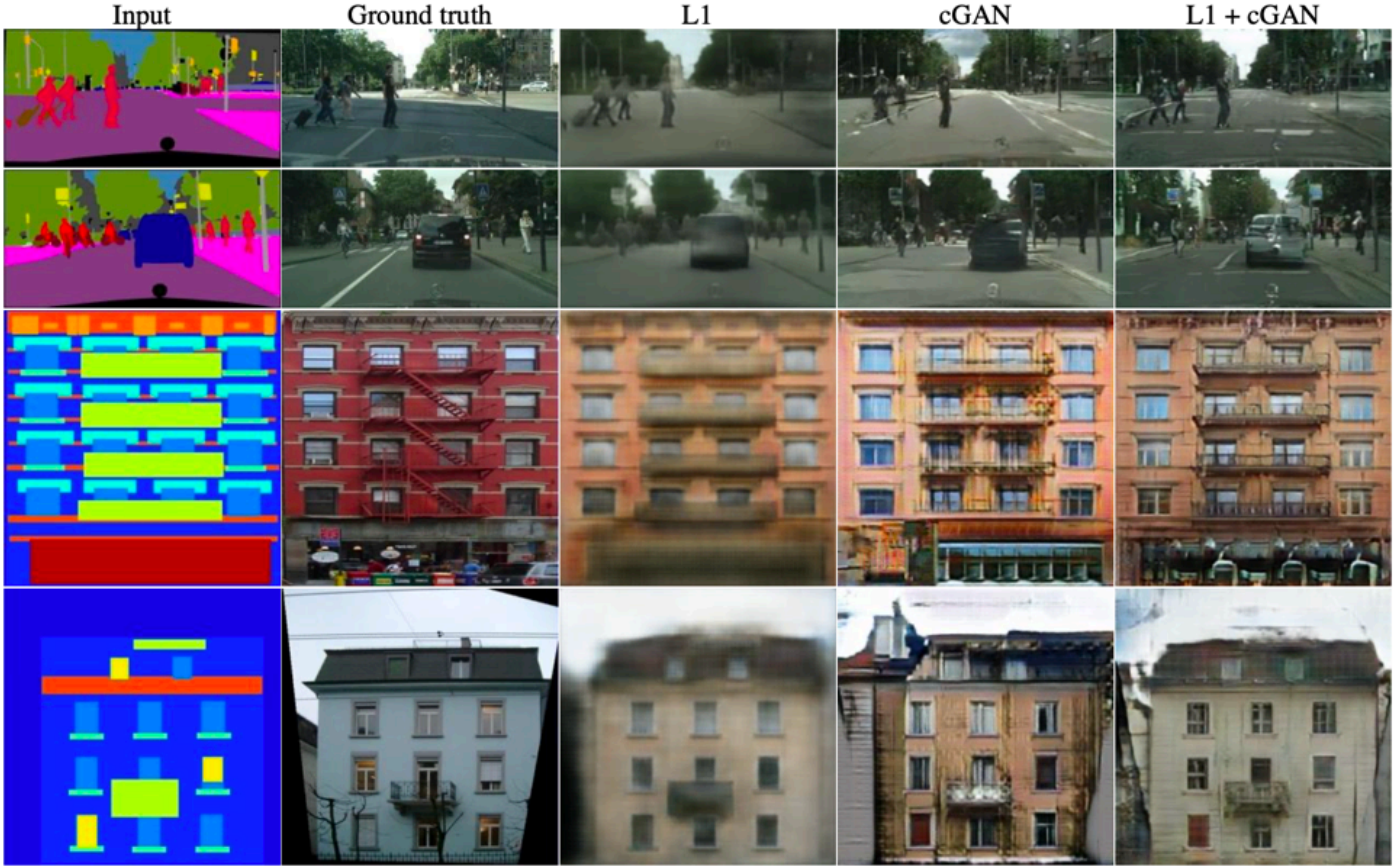


Note: pix2pix has an additional supervisory L1 loss = |y-ŷ|



This is a (conditional) Generative Adversarial Network

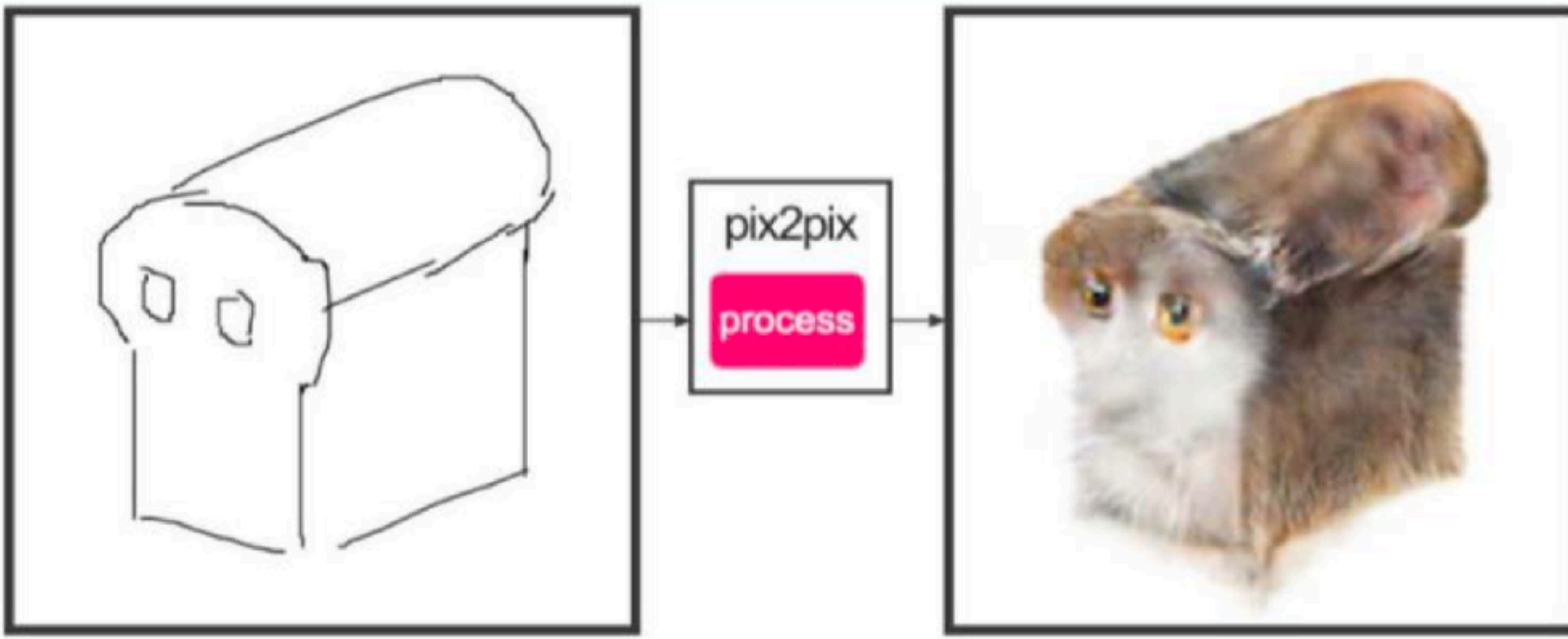
pix2pix: Segmentation→Image



pix2pix: Edges→Image

Model trained using edge detection, but works for hand drawings:

#edges2cats *by Christopher Hesse*



sketch by Ivy Tsai

https://affinelayer.com/pixsrv/


THANKS