Generative Adversarial Networks

Why Generative Models?

- **We've only seen discriminative models so far**
  - Given an image **X**, predict a label **Y**
  - Estimates **P(Y|X)**

- **Discriminative models have several key limitations**
  - They can not estimate joint probability distribution P(X, Y)
  - A generative model learns the joint probability distribution P(X, Y)

Why Generative Models?

Here's a really simple example. Suppose you have the following data in the form (x,y):

`(1,0), (1,0), (2,0), (2, 1)`

`p(x,y)` is

```
      y=0   y=1
     -----------
x=1 | 1/2   0
x=2 | 1/4   1/4
```

`p(y|x)` is

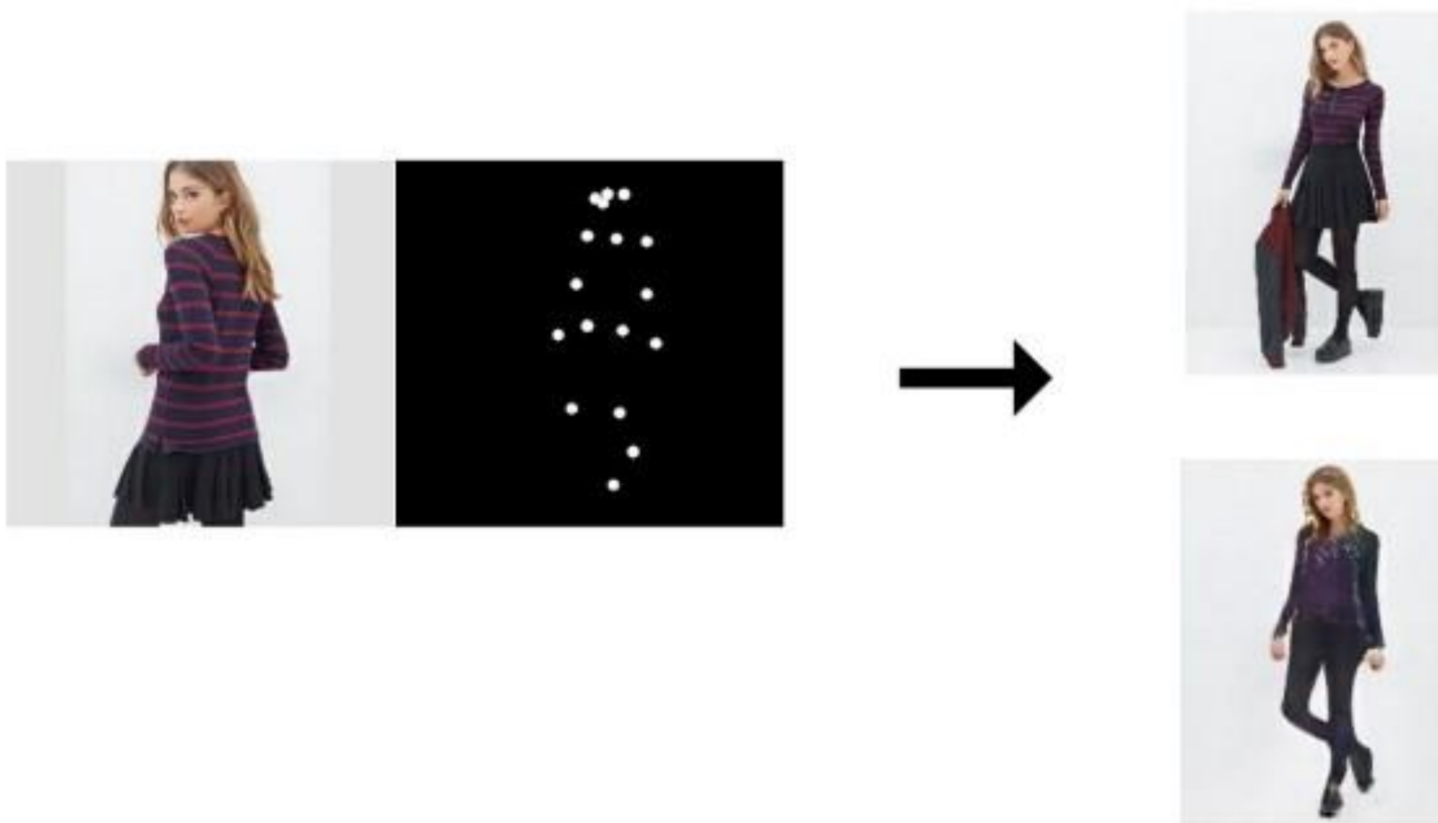
```
      y=0   y=1
     -----------
x=1 | 1     0
x=2 | 1/2   1/2
```

You could use p(x, y) to *generate* likely (x, y) pairs.

Celebrities [2]

Celebrities [2]

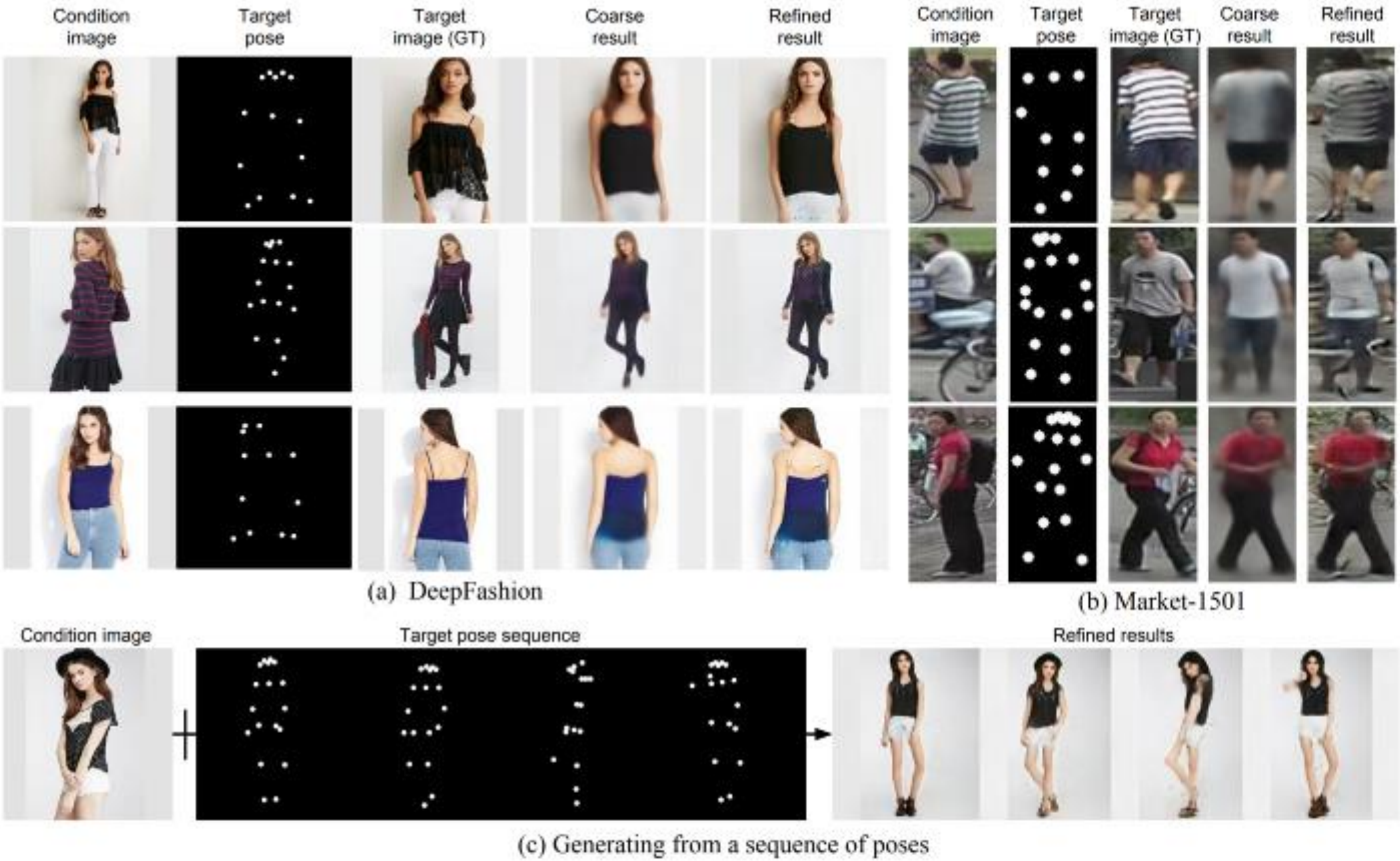
Pose Guided Person Image Generation [3]



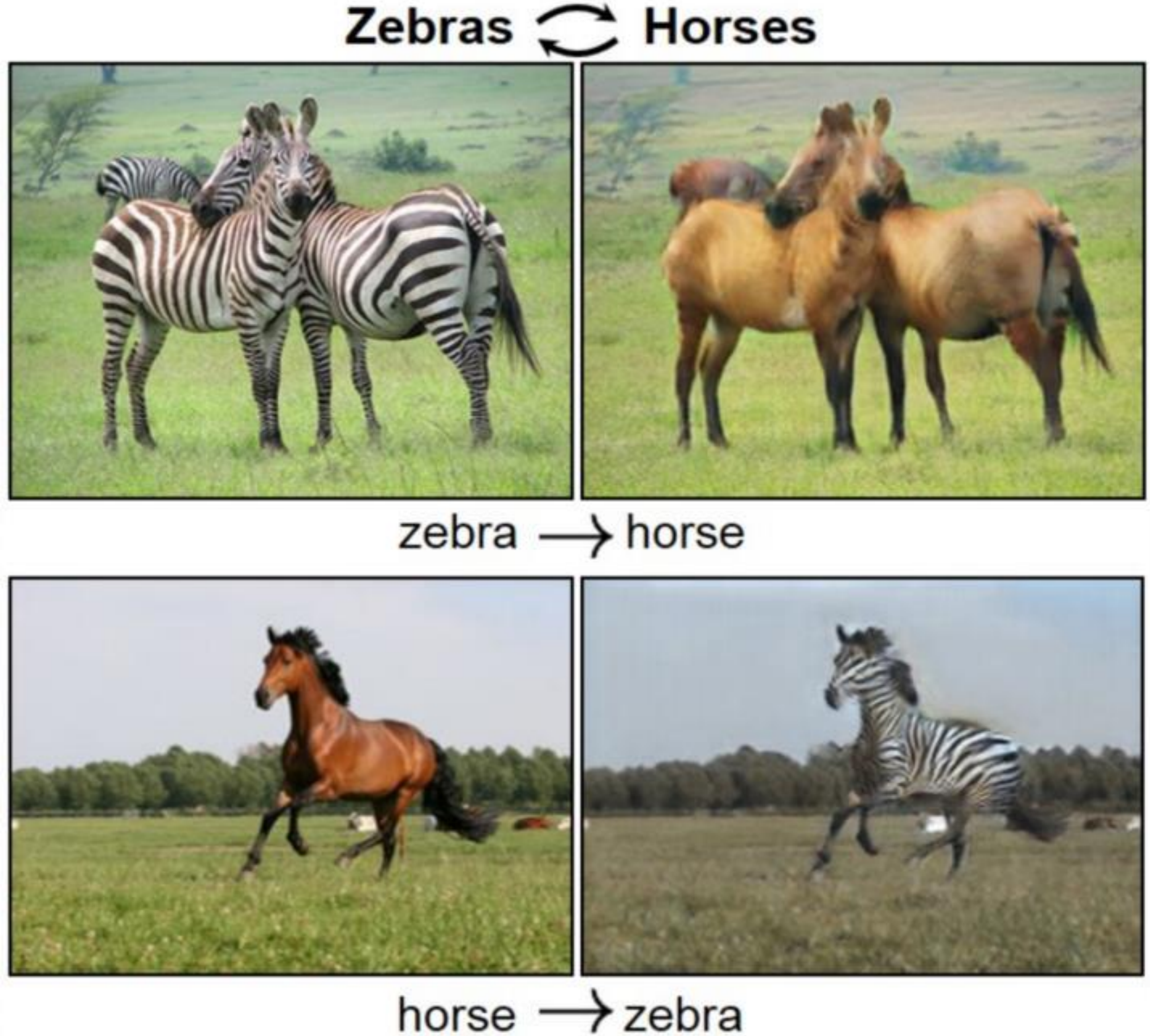
Ground truth

Generated

Pose Guided Person Image Generation [3]

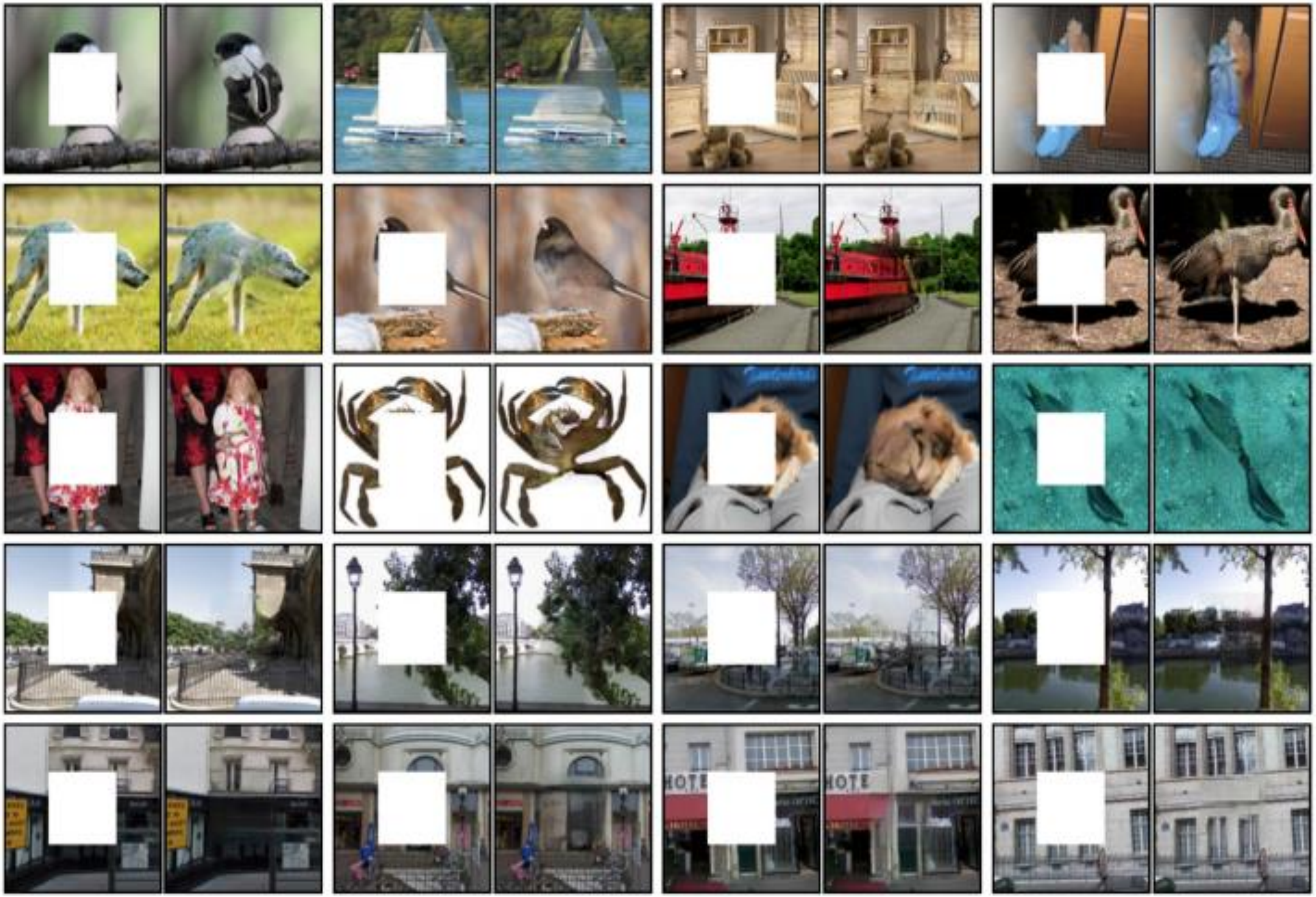


CycleGAN [4]



Text to Image [5]

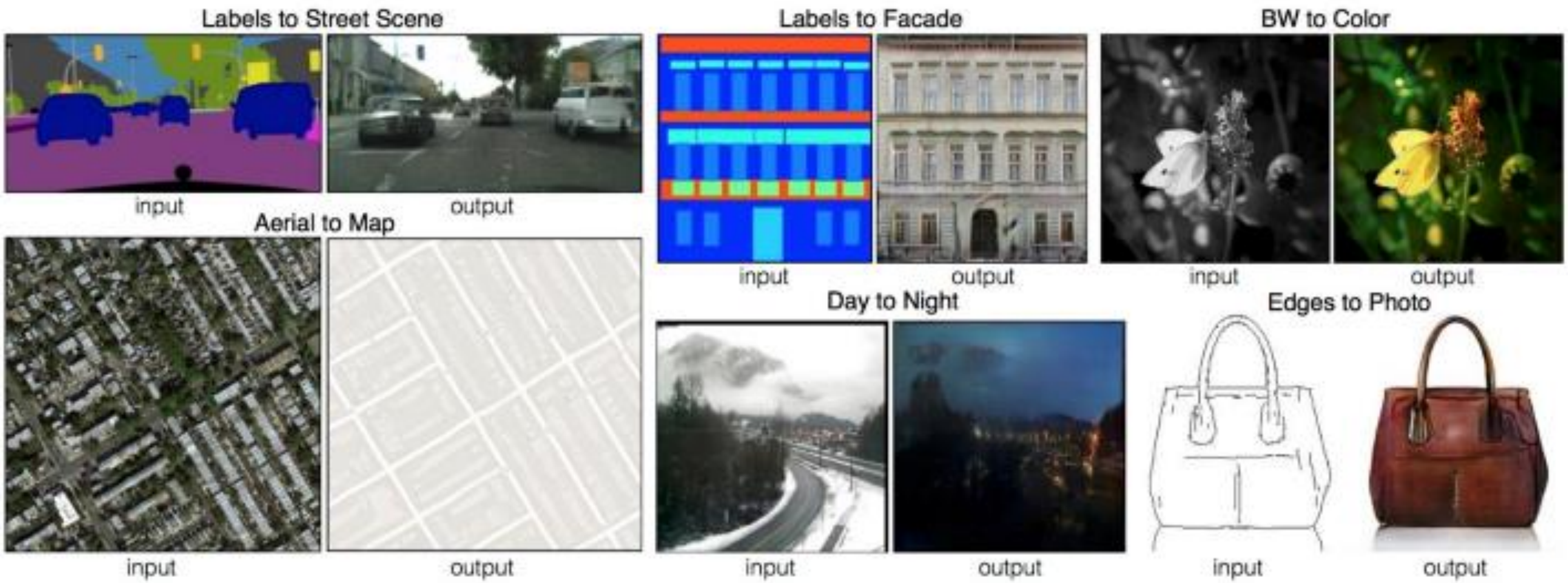
Image Inpainting [6]



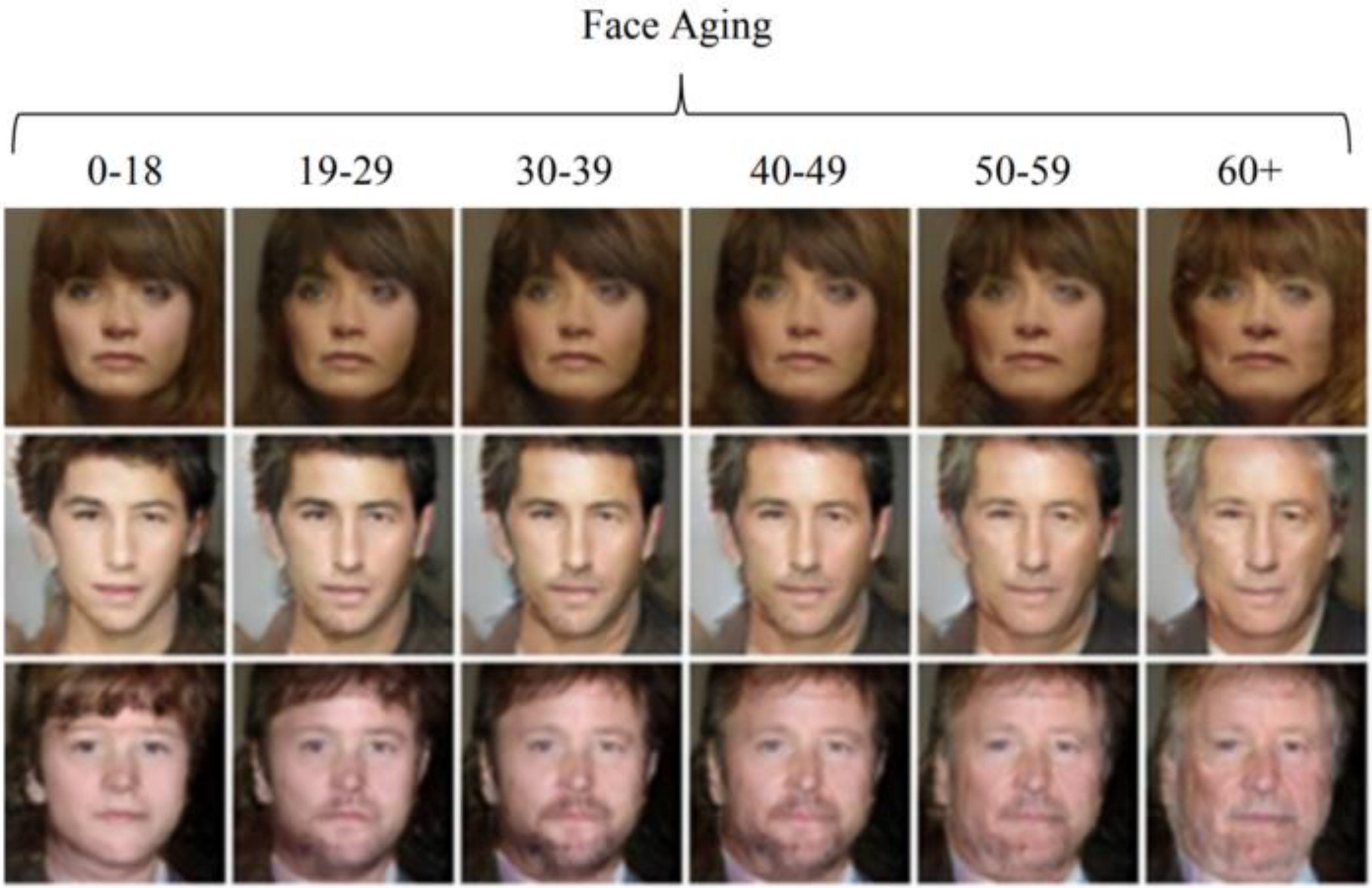
DiscoGAN [7]

(b) Handbag images (input) & **Generated** shoe images (output)

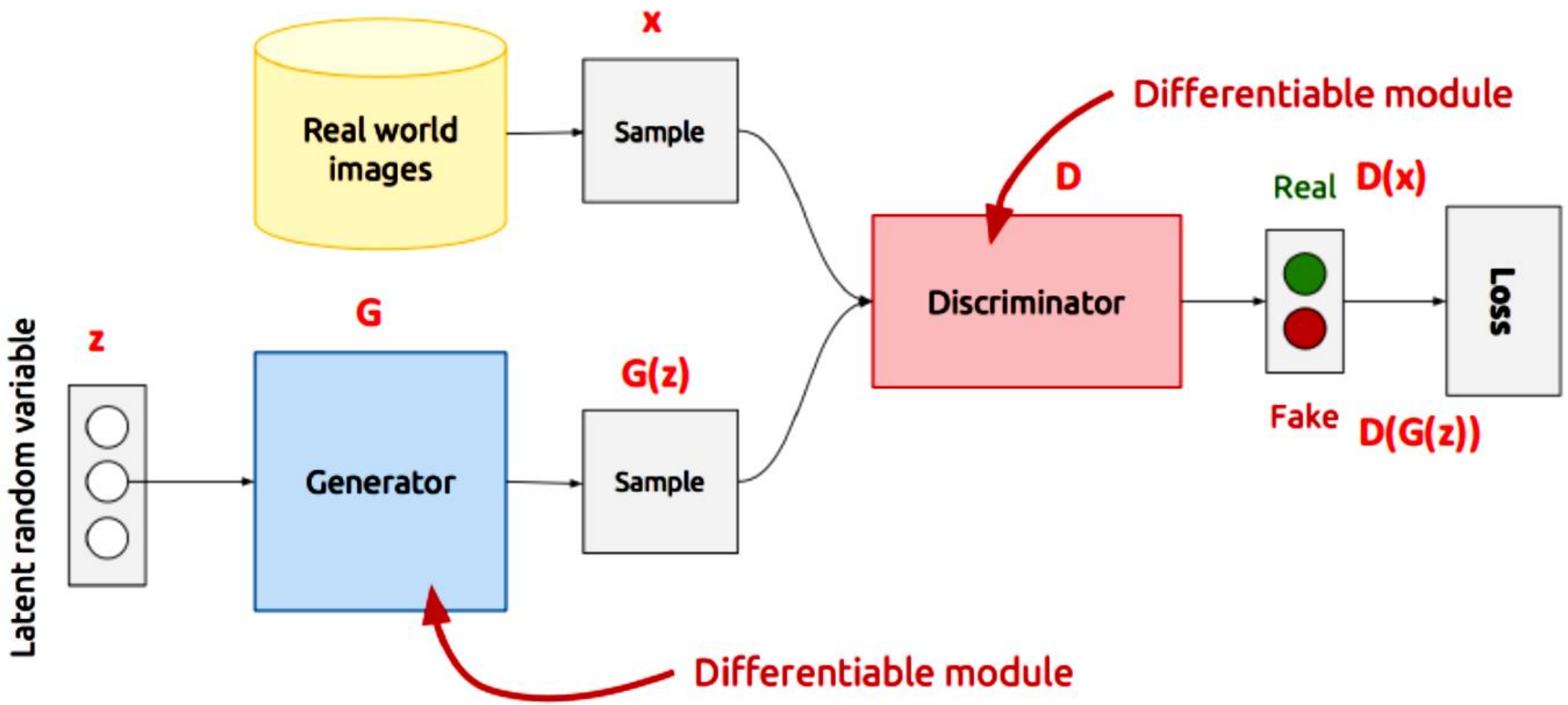
Pix2Pix [8]


Labels to Street Scene
input
output
Aerial to Map
input
output
Labels to Facade
input
output
BW to Color
input
output
Day to Night
input
output
Edges to Photo
input
output

Face aging (Age-cGAN) [9]



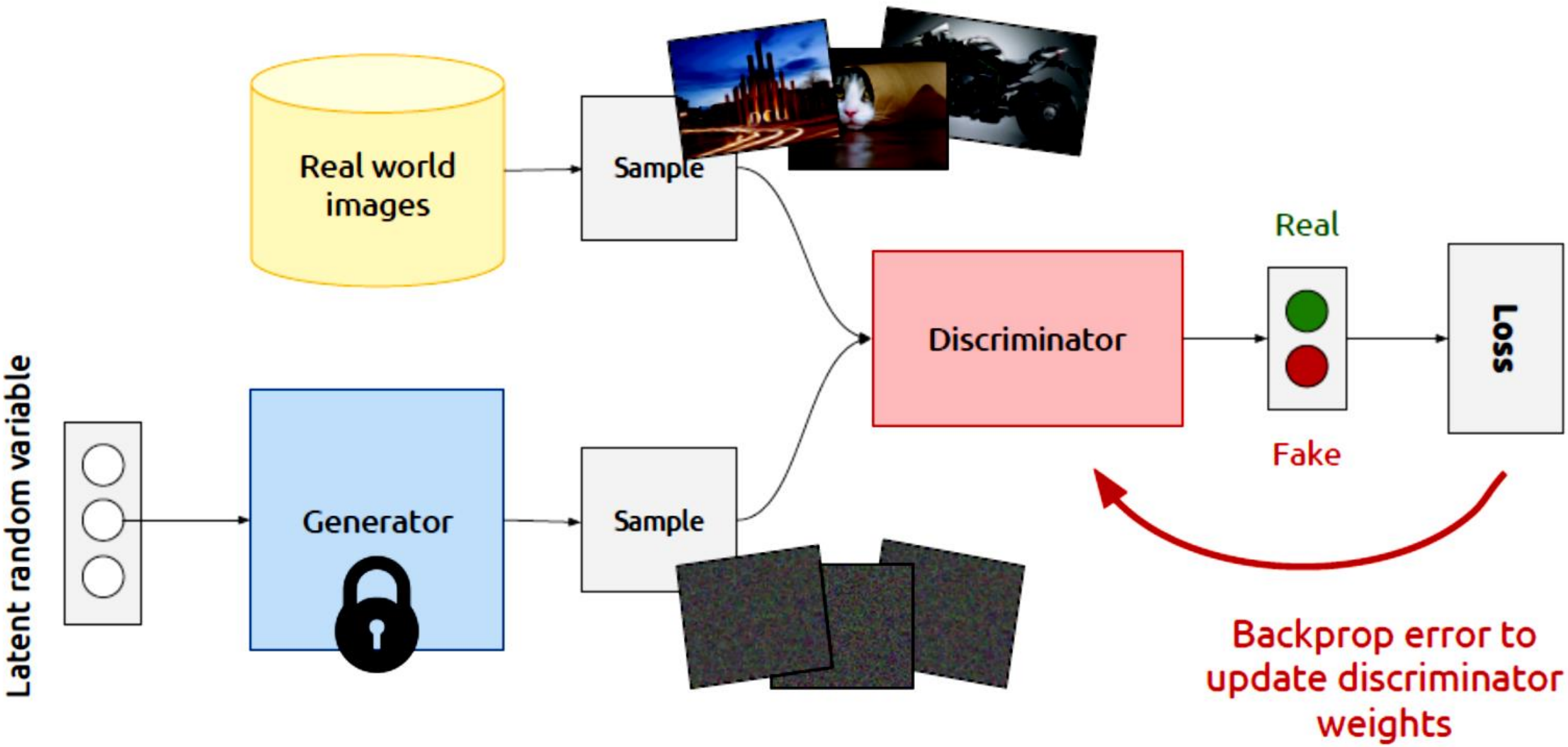
GAN's Architecture [1, 10]



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

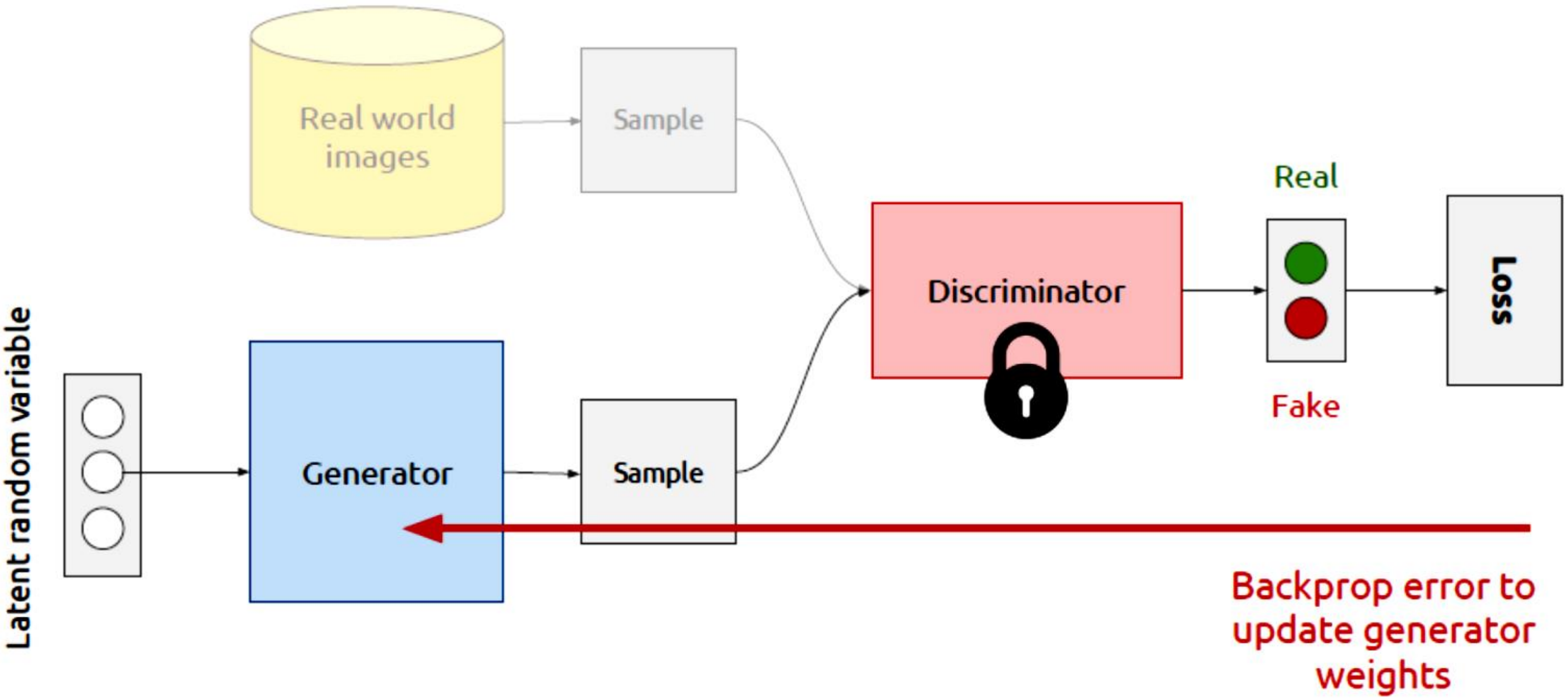
Training Discriminator [1, 10]

1. Fix generator weights, draw samples from both real world and generated images
2. Train discriminator to distinguish between real world and generated images



Training Generator [1, 10]

1. Fix discriminator weights
2. Sample from generator
3. Backprop error through discriminator to update generator weights



Training GANs: Two-player game [11]

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Training GANs: Two-player game [11]

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

x is sampled from real data

z is sampled from N(0, I)

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

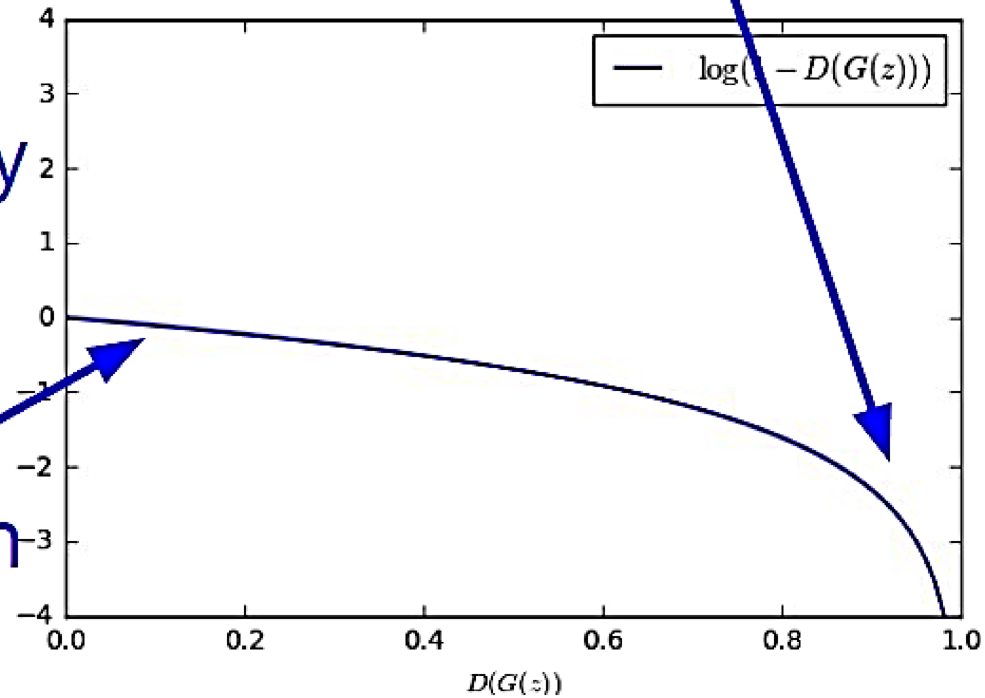
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Training GANs: Two-player game [11]

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



Training GANs: Two-player game [11]

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

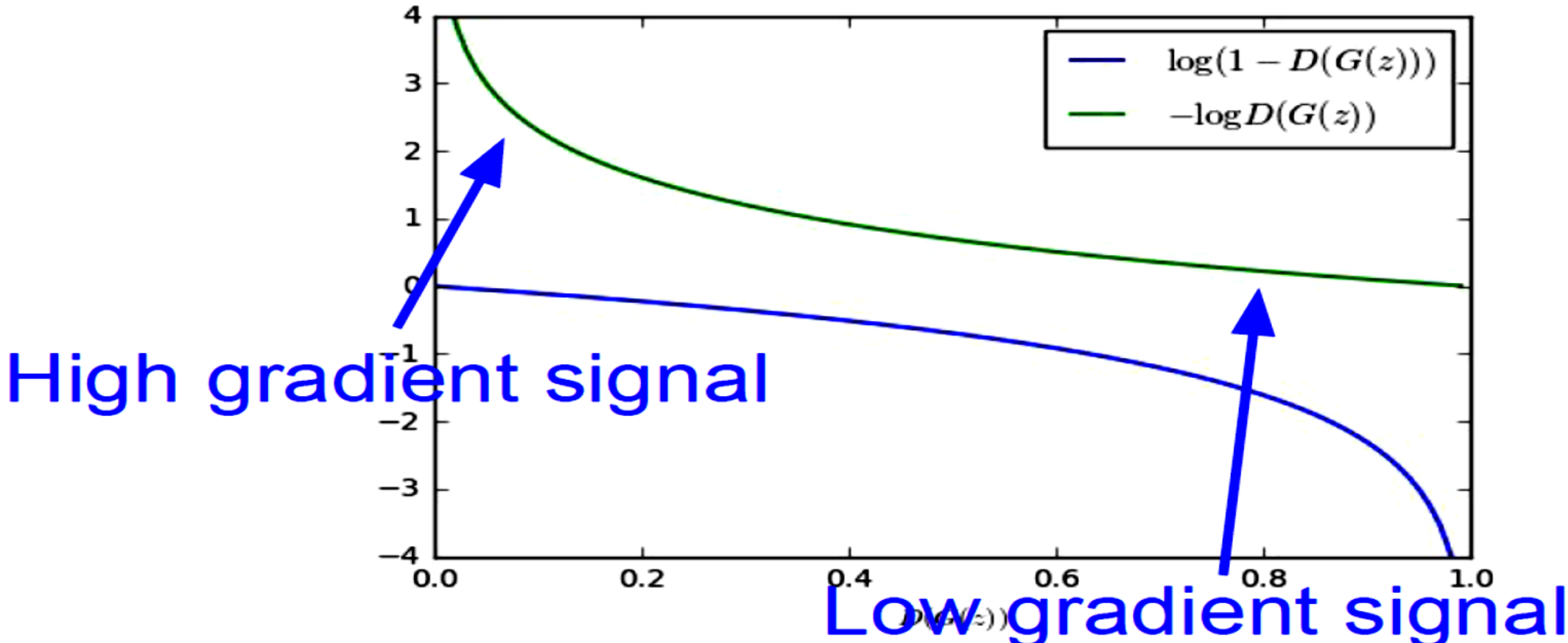
$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs: Two-player game [11]

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



Training GANs: Two-player game [11]

Putting it together: GAN training algorithm

Some find k=1 more stable, others use k > 1, no best rule.

for number of training iterations **do**

for $\boxed{k \text{ steps}}$ **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

end for

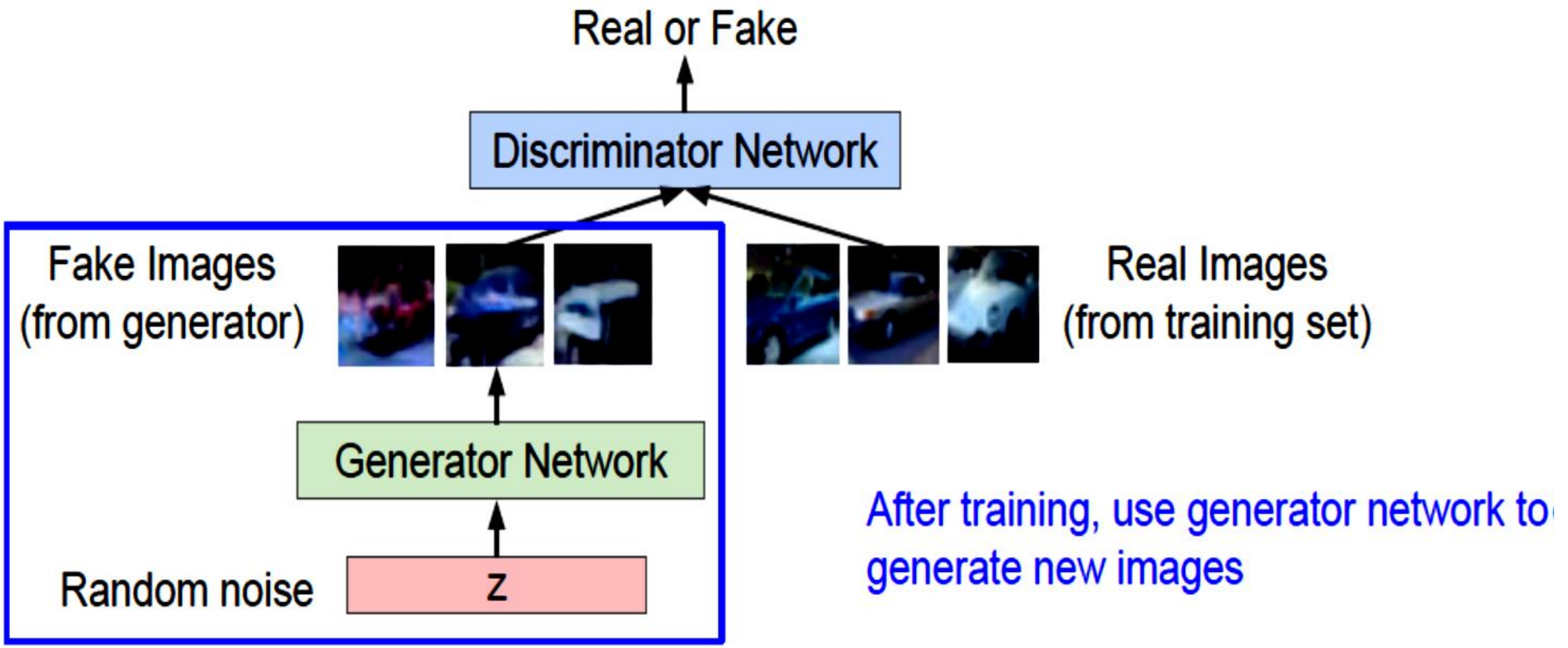
- Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Training GANs: Two-player game [11]

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



DCGAN [12]

Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

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

DCGAN [12, 13]

In-Network upsampling: "Unpooling"

Nearest Neighbor

1	2
3	4

→

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Input: 2 x 2

Output: 4 x 4

"Bed of Nails"

1	2
3	4

→

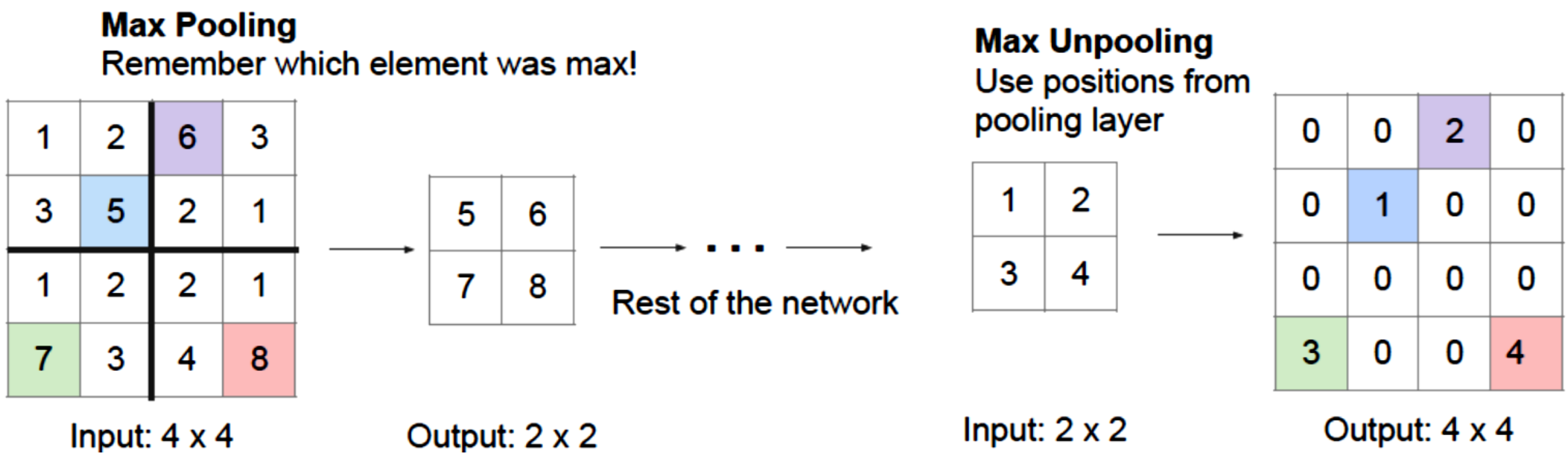
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2

Output: 4 x 4

DCGAN [12, 13]

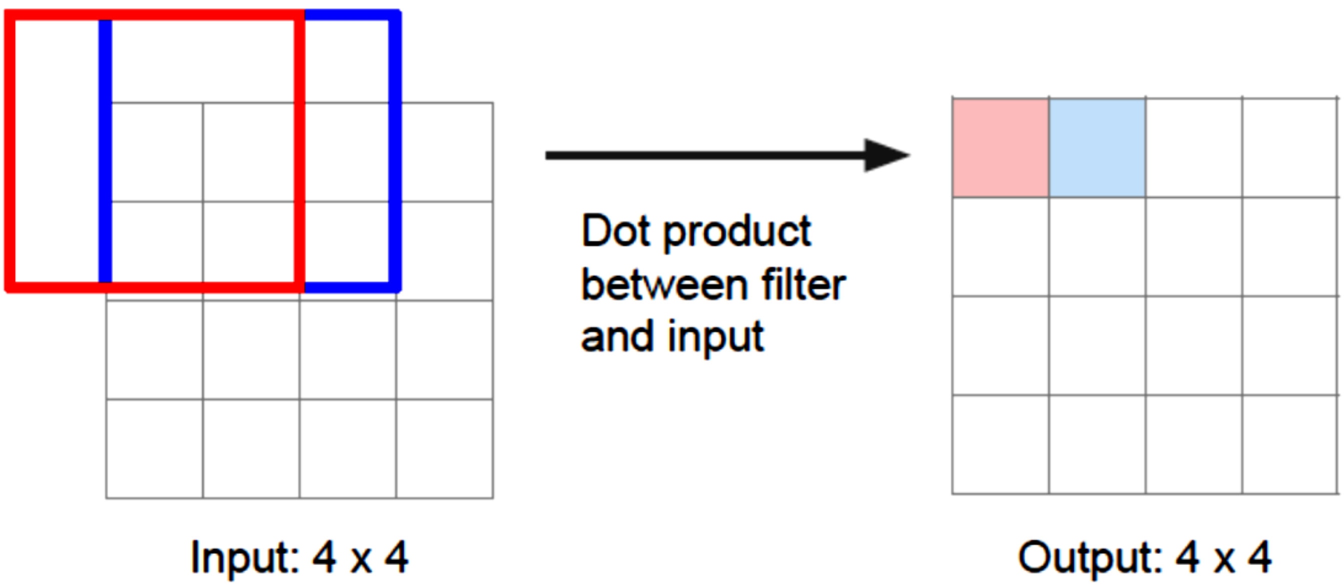
In-Network upsampling: “Max Unpooling”



DCGAN [12, 13]

Learnable Upsampling: Transpose Convolution

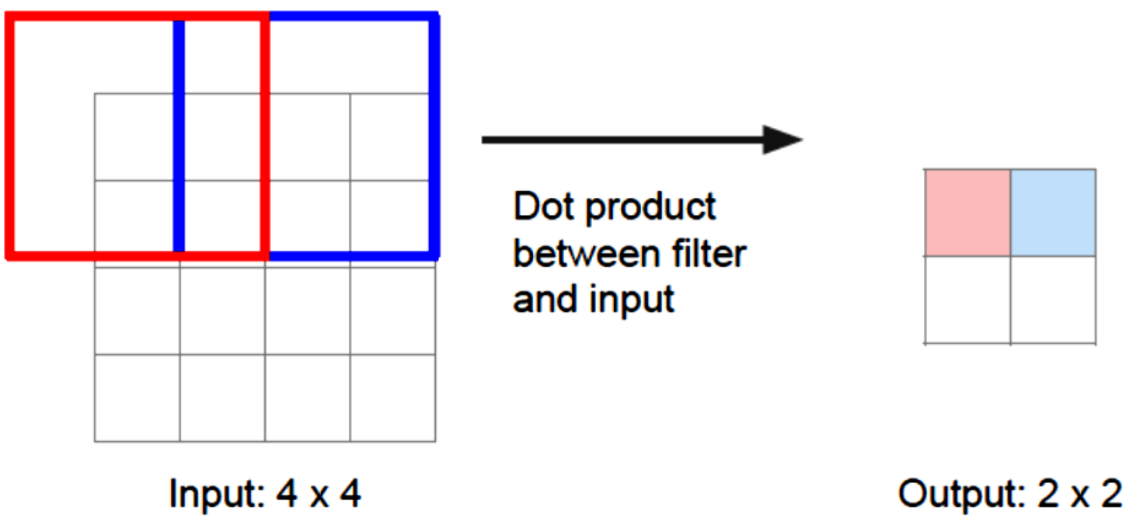
Recall: Normal 3 x 3 convolution, stride 1 pad 1



DCGAN [12, 13]

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, <u>stride 2</u> pad 1



Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

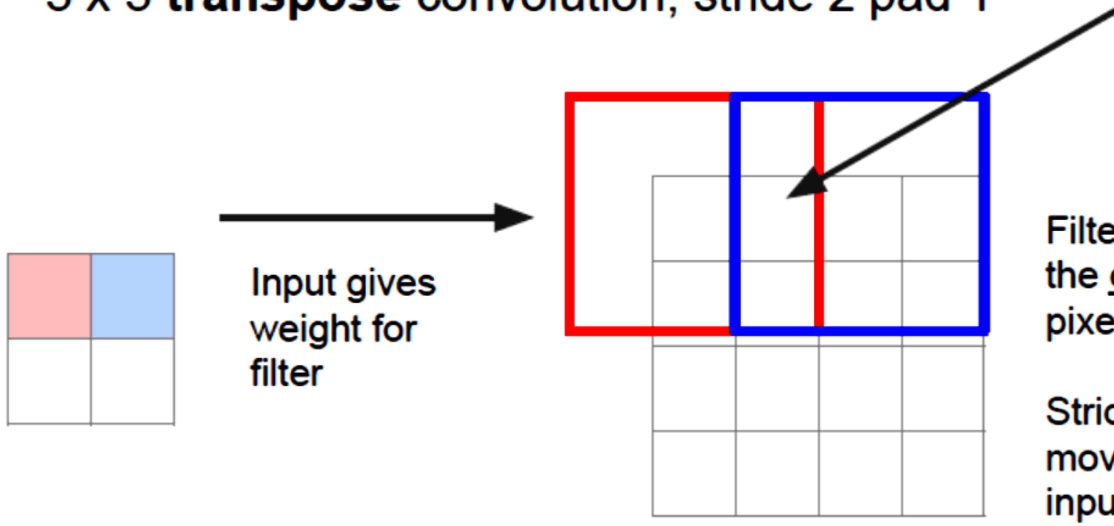
DCGAN [12, 13]

Learnable Upsampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1

Sum where output overlaps



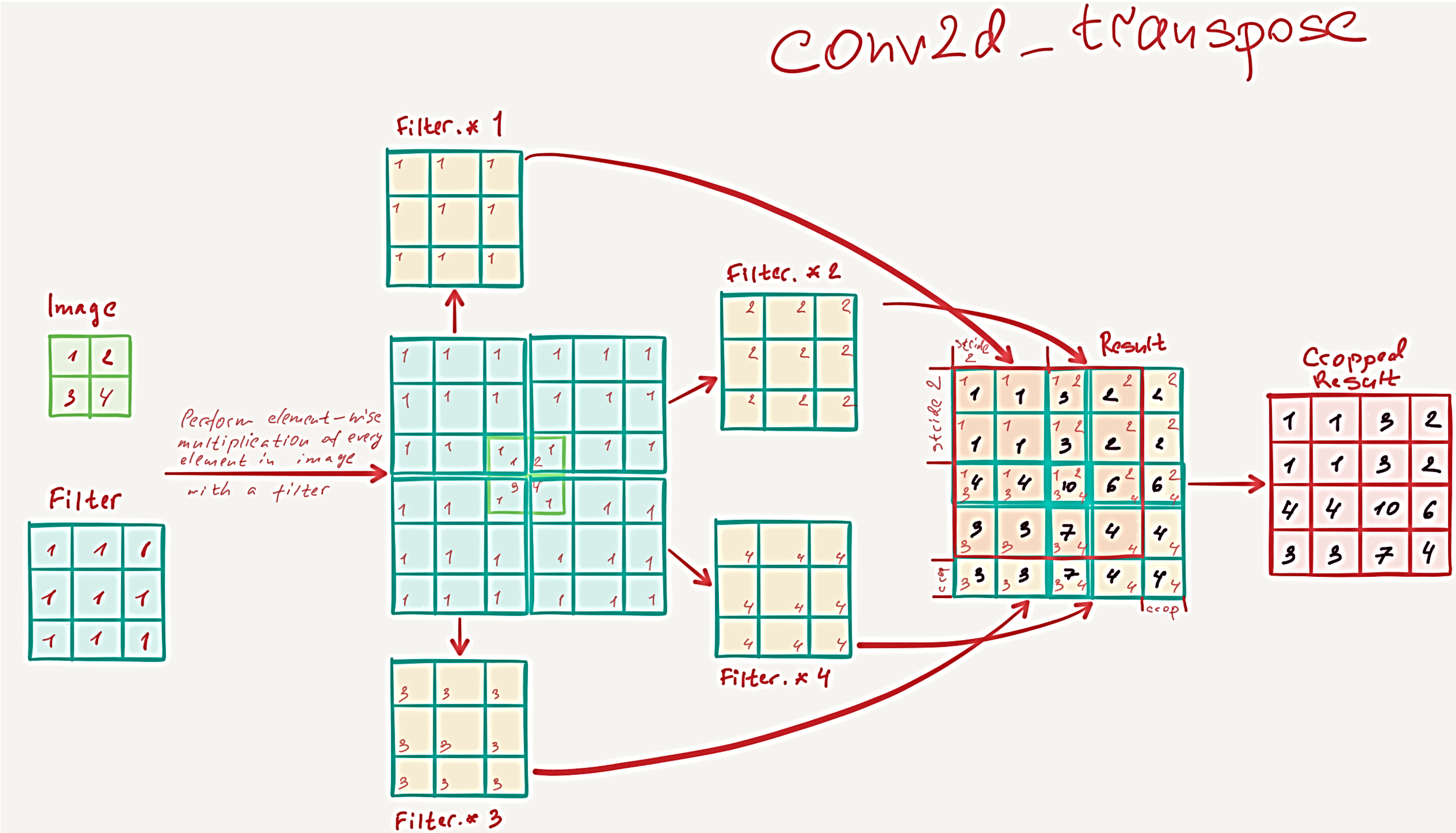
Input: 2 x 2

Output: 4 x 4

Filter moves 2 pixels in the output for every one pixel in the input

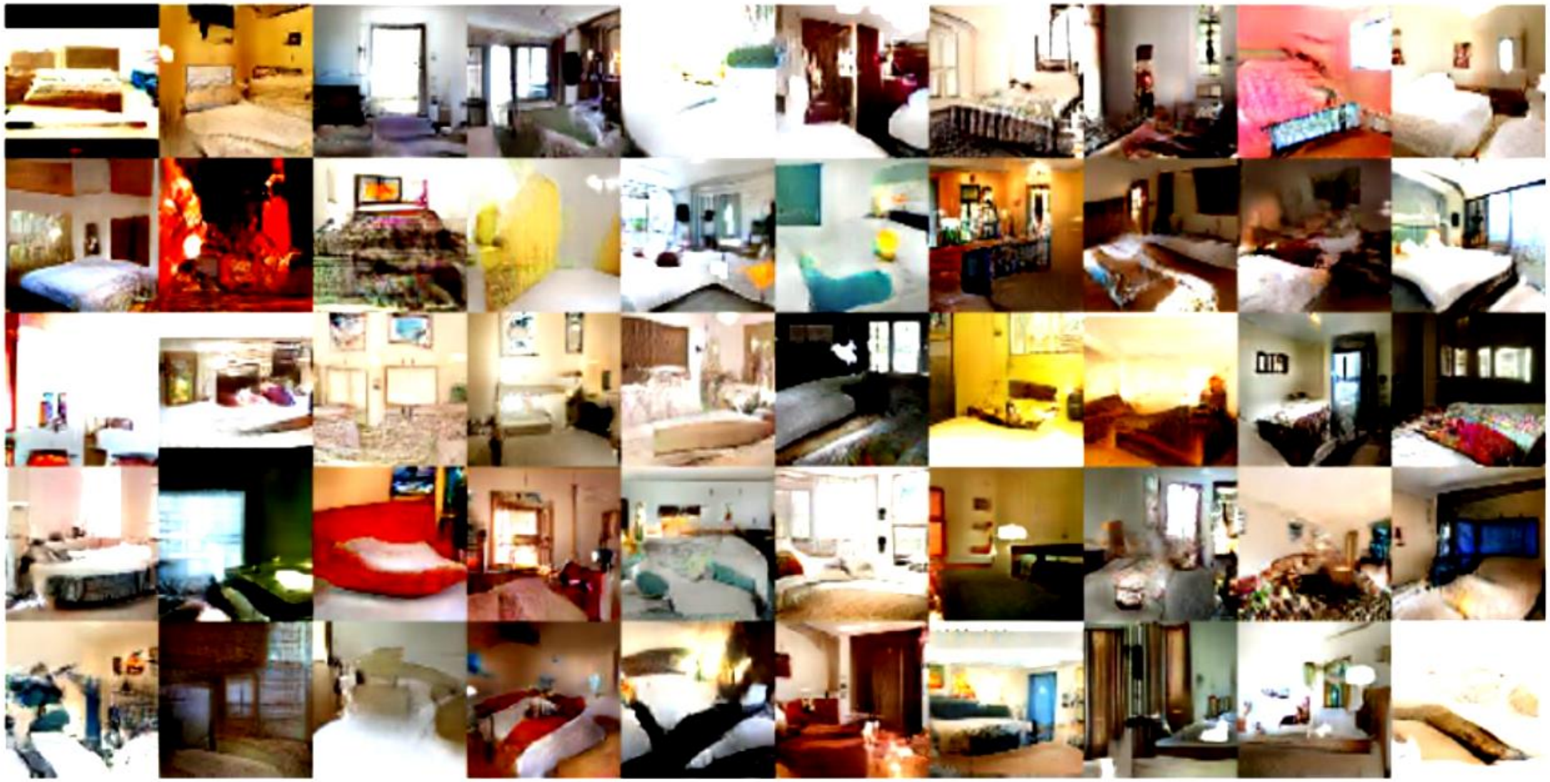
Stride gives ratio between movement in output and input

DCGAN [12, 13]


conv2d_transpose
Image
Filter
Perform element-wise multiplication of every element in image with a filter
Filter.* 1
Filter.* 2
Filter.* 3
Filter.* 4
Result
stride 2
crop
Cropped Result

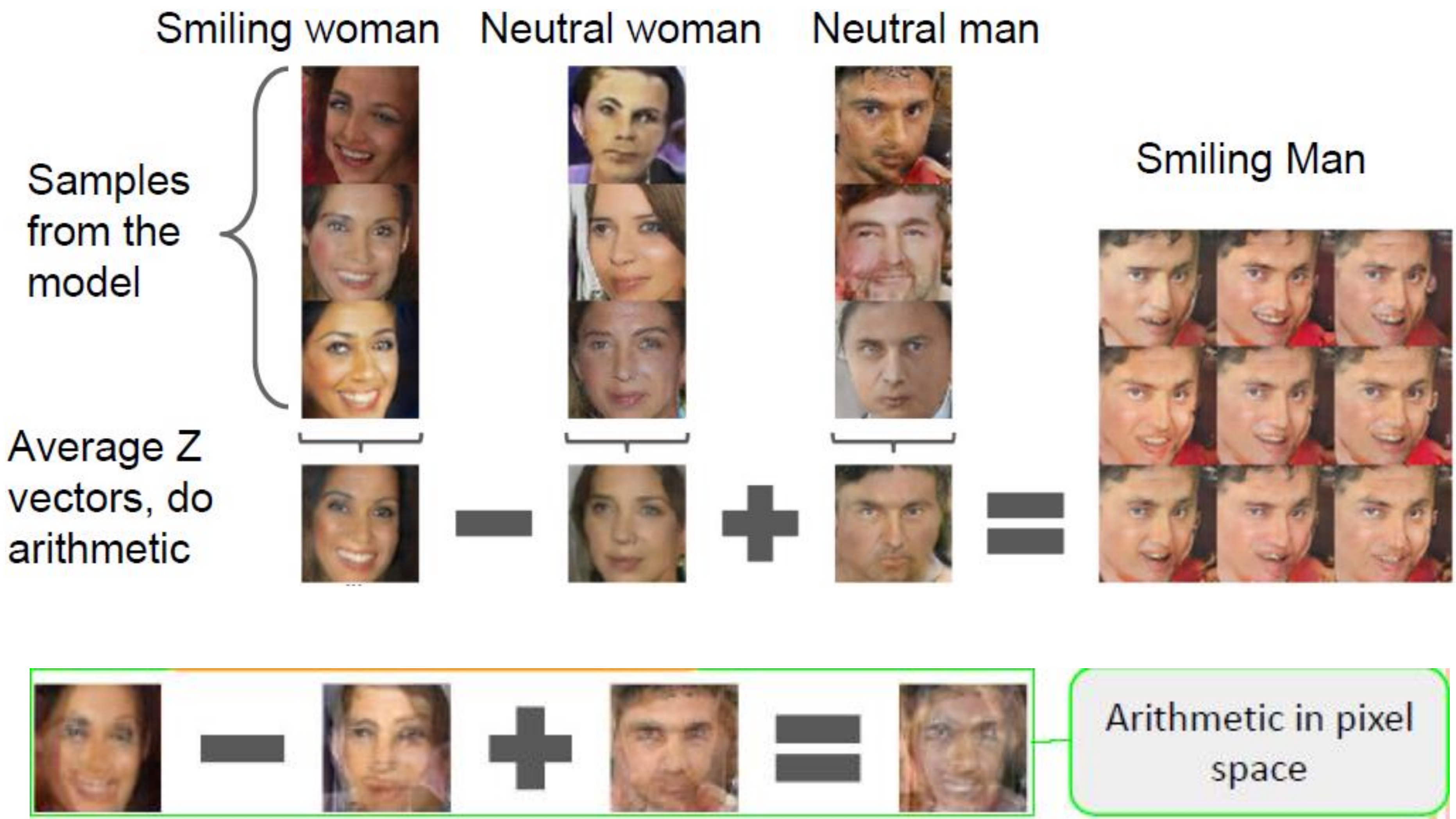
DCGAN [12]

Samples from the model look amazing!



Radford et al, ICLR 2016

DCGAN – Interpretable Vector Math [12]



DCGAN – Interpretable Vector Math [12]

References

1. Andrew Ng's Lecture on Sequence Models (https://www.coursera.org/learn/nlp-sequence-models)
2. Progressive Growing of GANs for Improved Quality, Stability, and Variation
3. Ma, Liqian, et al. "Pose guided person image generation." Advances in Neural Information Processing Systems. 2017.
4. https://github.com/junyanz/CycleGAN
5. https://github.com/hanzhanggit/StackGAN
6. https://github.com/pathak22/context-encoder
7. https://github.com/carpedm20/DiscoGAN-pytorch
8. https://github.com/phillipi/pix2pix

References

9. Antipov, Grigory, Moez Baccouche, and Jean-Luc Dugelay. "Face aging with conditional generative adversarial networks." Image Processing (ICIP), 2017 IEEE International Conference on. IEEE, 2017.

10. https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

11. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

12. Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

References

13. http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture11.pdf

Disclaimer

- These slides are not original and have been prepared from various sources for teaching purpose.