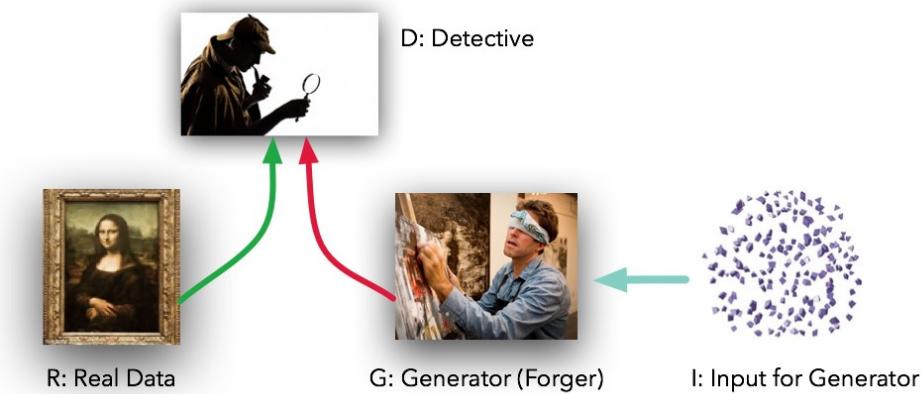


Generative Adversarial Network



Generative Adversarial Network

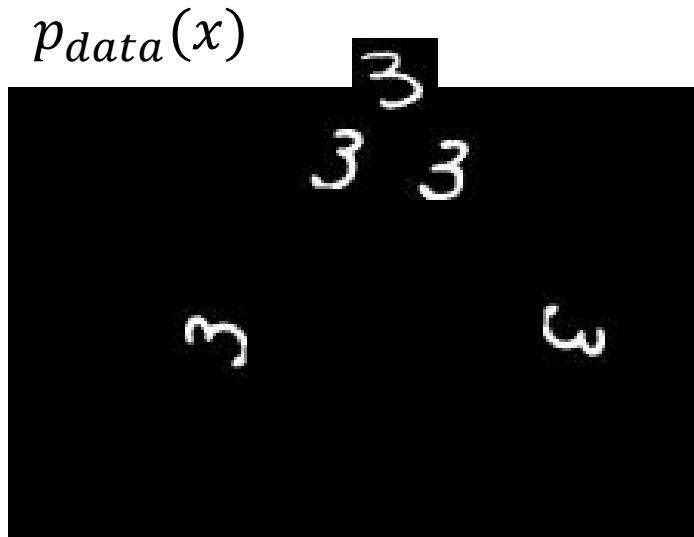
-NIPS 2014

Generative Model

- Data: 3 3 3 3 3 3 3 3 ...

- Generative model: $p_G(x; \theta_G)$

- True data distribution: $p_{data}(x)$
 - Train $p_G(x) \approx p_{data}(x)$

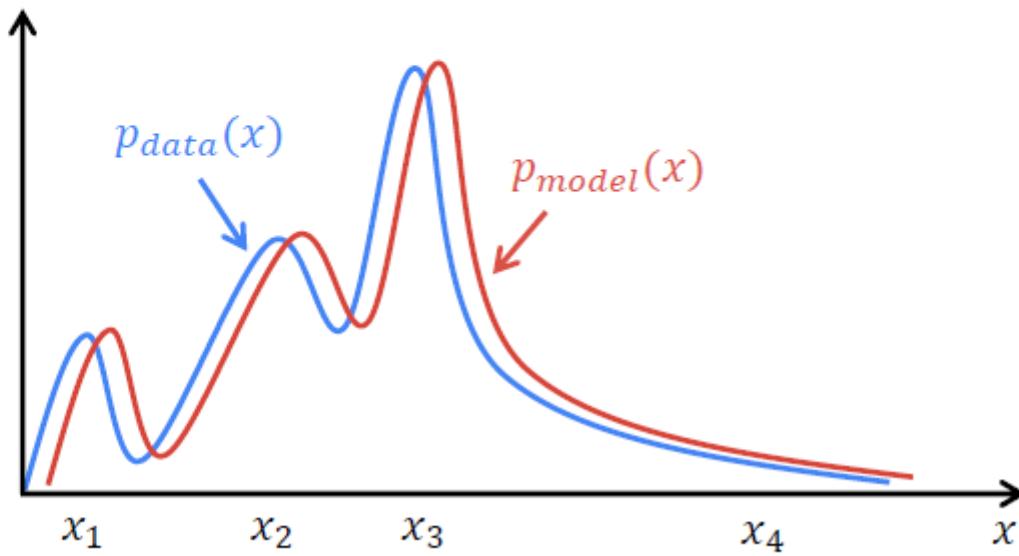


Generative Model

The goal of the generative model is to find a $p_{model}(x)$ that approximates $p_{data}(x)$ well.

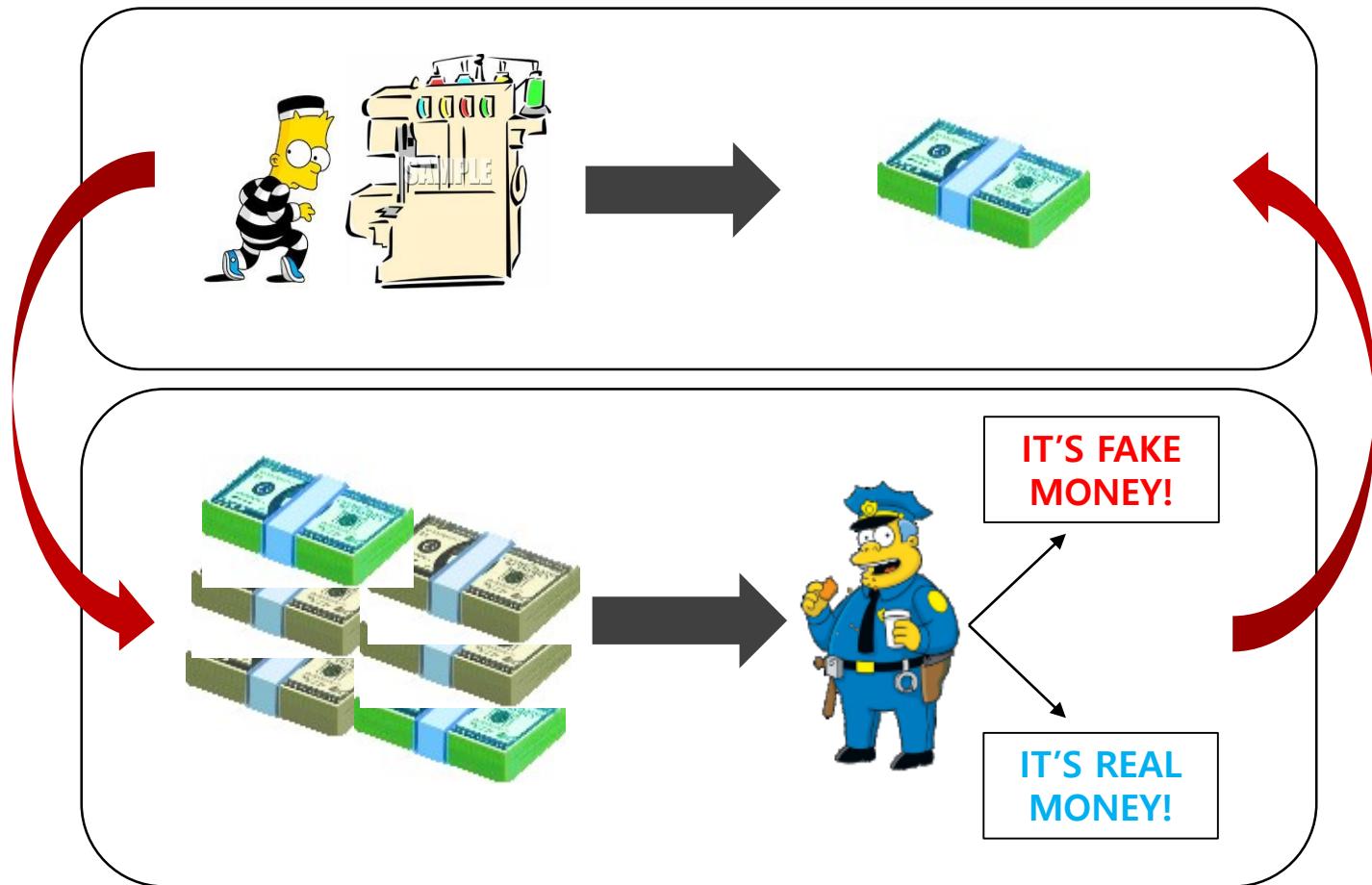
↗ Distribution of images generated by the model

↖ Distribution of actual images

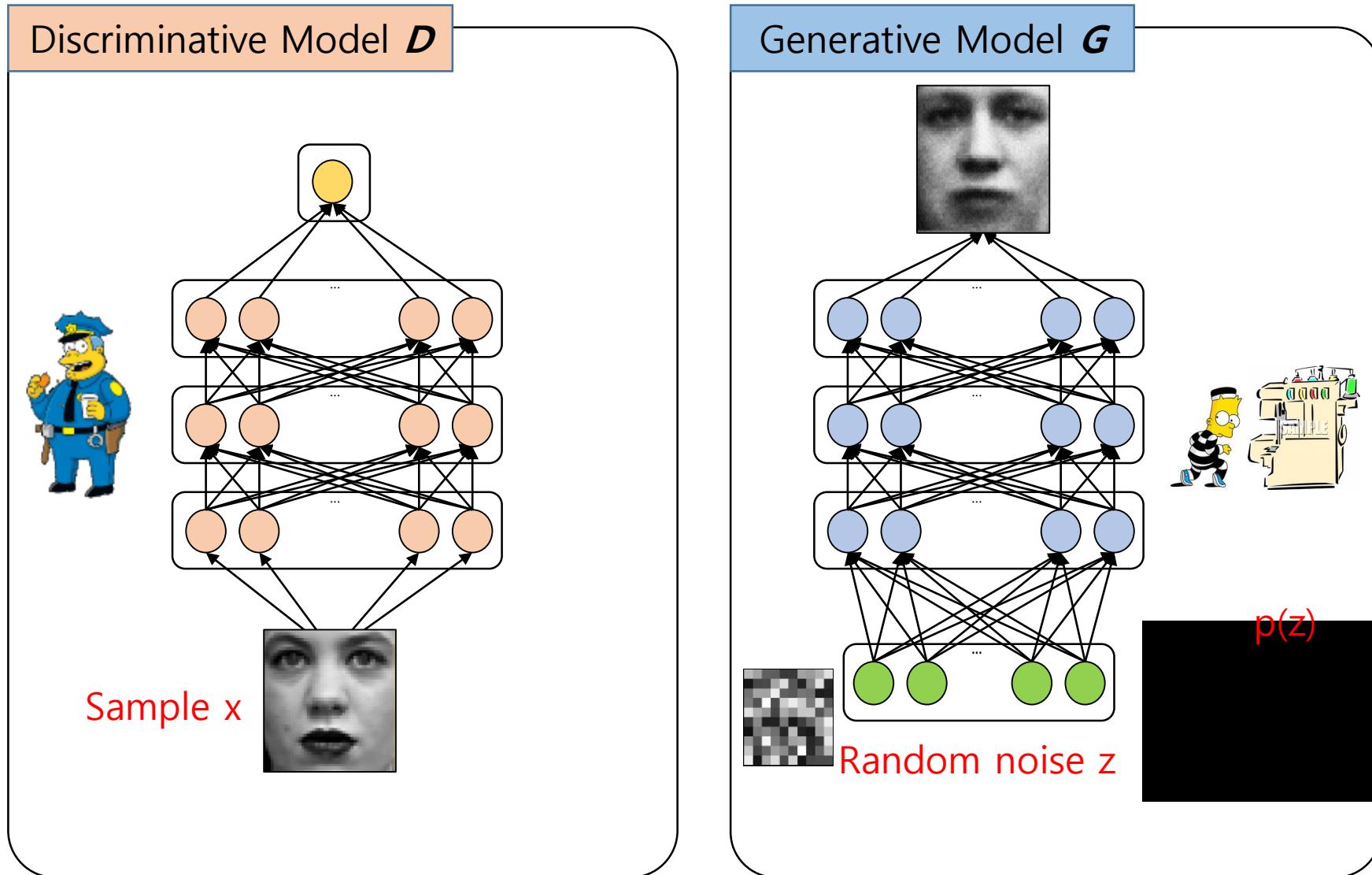


Generative Adversarial Network

- Counterfeitors vs Police Game



Generative Adversarial Network

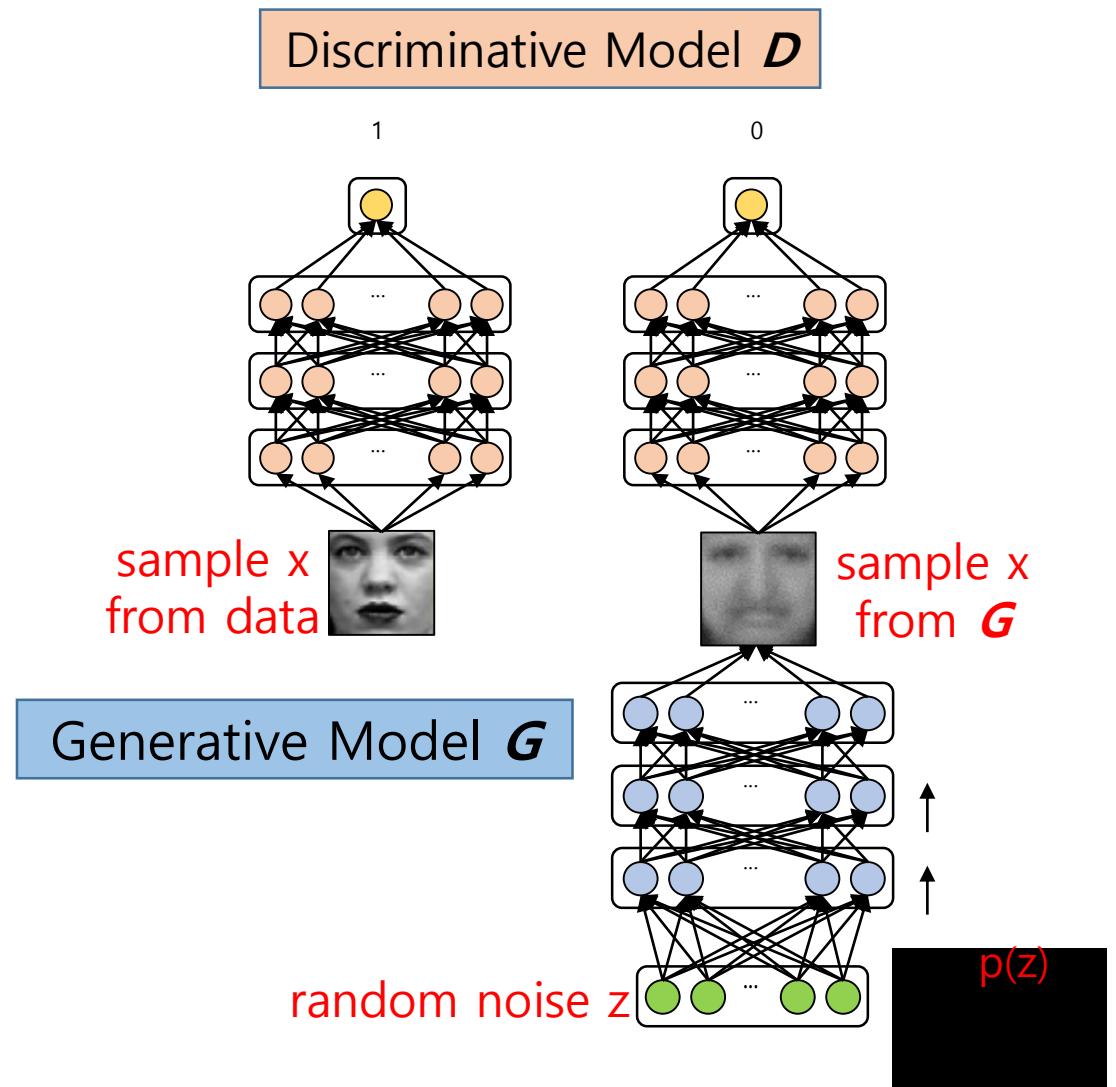


Generative Adversarial Network

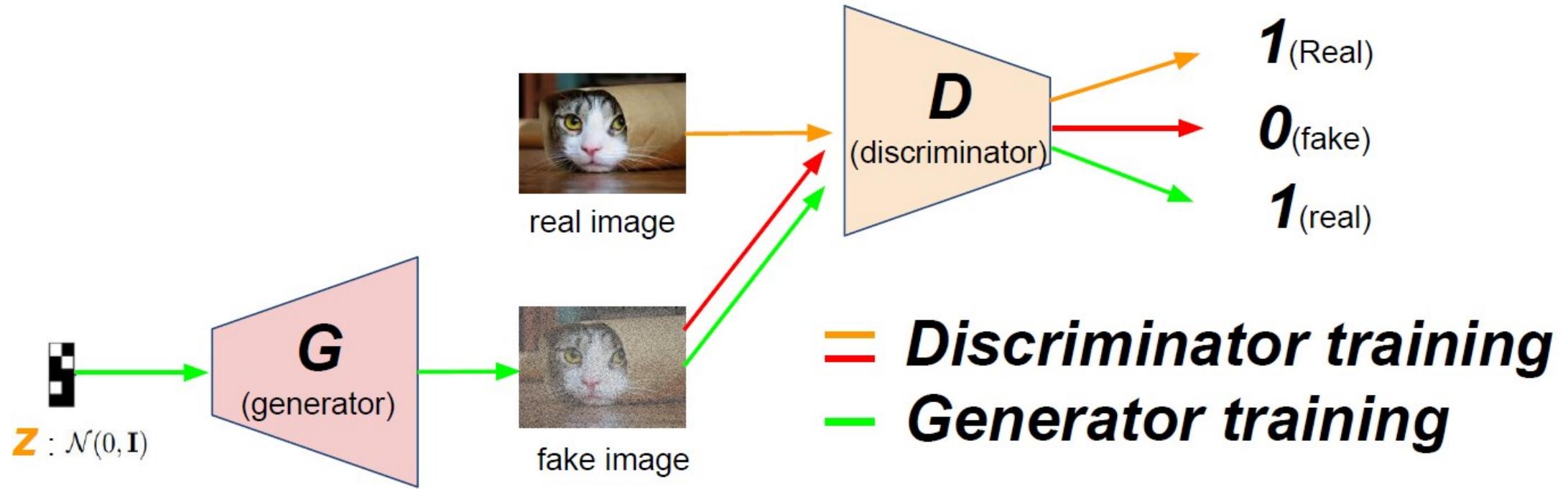
- Discriminative Model D
 - Try to classify the sample x
 - $D(x)=1$ when x from Data
 - $D(x)=0$ when x from G (generator)

Differentiable function
represented by a multilayer
perceptron with parameters

- Generative Model G
 - Try to generate sample x
 - As similar as the real data



Training GAN

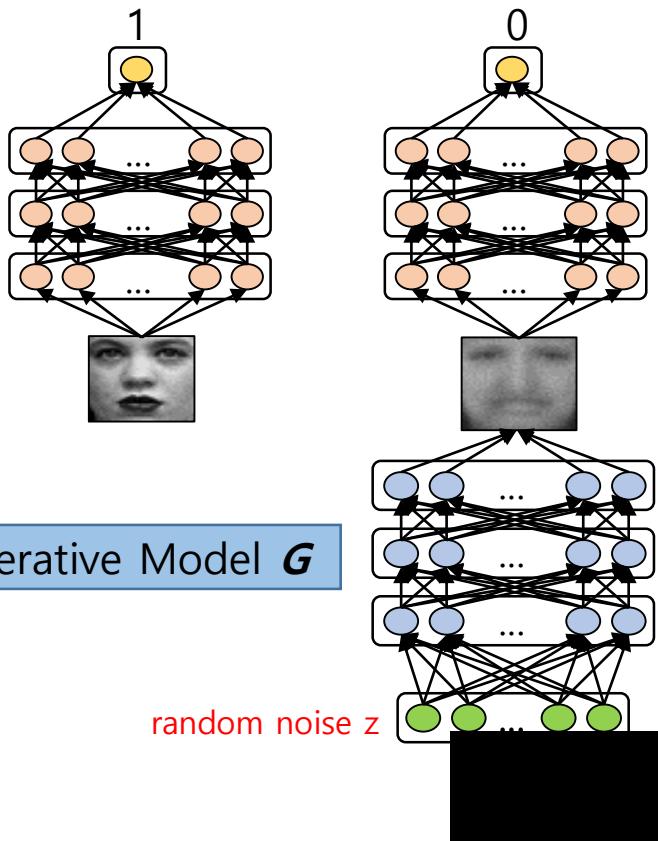


Generative Adversarial Network

$$\min_G \max_D V(D, G)$$

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Discriminative Model D



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
    for  $k$  steps do
        • Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
        • Sample minibatch of  $m$  examples  $\{x^{(1)}, \dots, x^{(m)}\}$  from data generating distribution  $p_{\text{data}}(x)$ .
        • Update the discriminator by ascending its stochastic gradient:
```

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

```
end for
• Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
• Update the generator by descending its stochastic gradient:
```

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)}))).$$

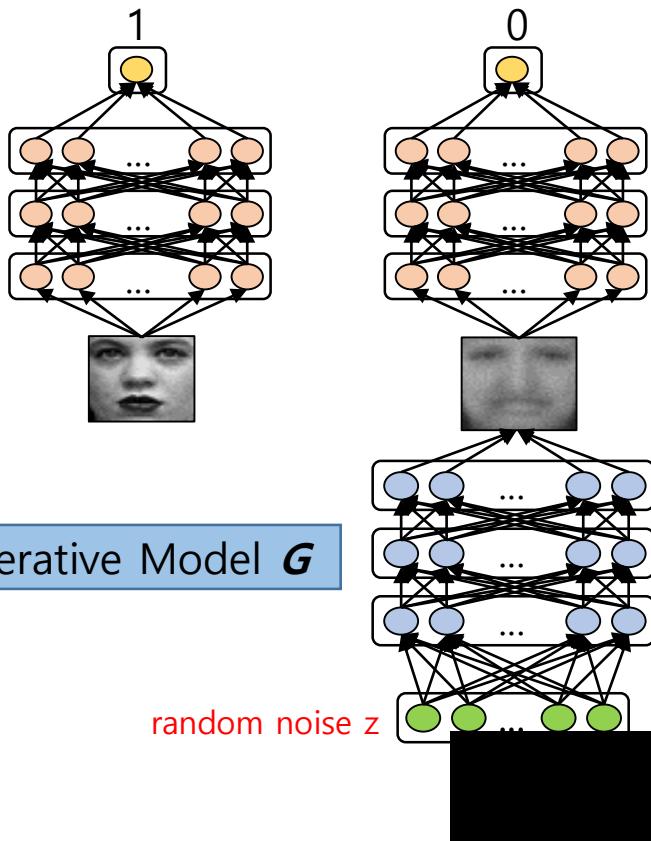
```
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```

Generative Adversarial Network

$$\min_G \max_D V(D, G)$$

$$V(D, G) = \mathbb{E}_{x \sim p_{data(x)}} [\log D(x)] + \mathbb{E}_{z \sim p_{z(z)}} [\log(1 - D(G(z)))]$$

Discriminative Model D



- Fixed G , maximize V :

$$\max_D V_G(D) = \max_D \left[\mathbb{E}_{x \sim p_{data(x)}} [\log D(x)] + \mathbb{E}_{z \sim p_{z(z)}} [\log(1 - D(G(z)))] \right]$$

- From sample $x^{(i)}, z^{(i)}$

$$\max_D \left[\sum_{i=1}^m \left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))) \right] \right]$$

- Binary Classification (logistic loss):

- Sample from data: label=1
- Sample from generator: label = 0

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

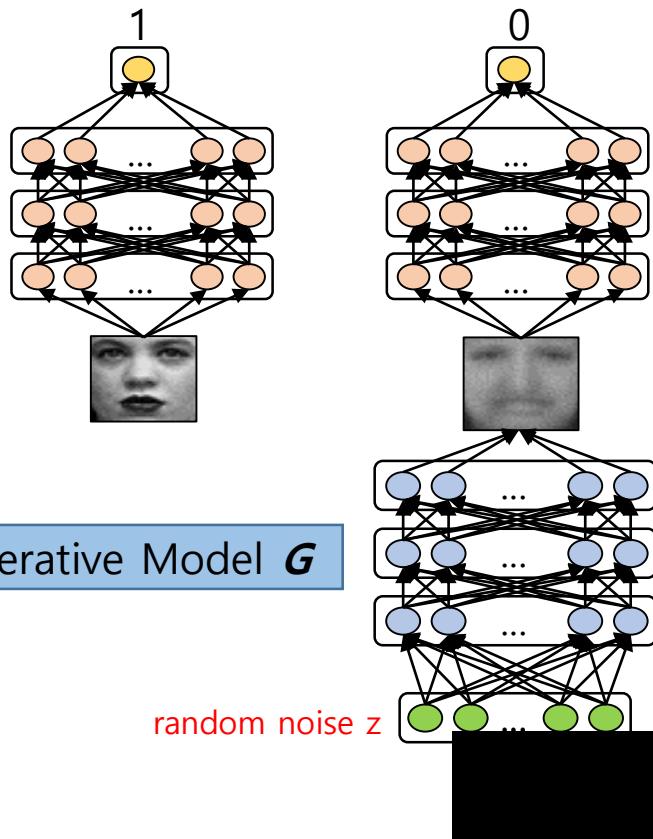
Stochastic
Gradient

Generative Adversarial Network

$$\min_G \max_D V(D, G)$$

$$V(D, G) = \mathbb{E}_{x \sim p_{data(x)}} [\log D(x)] + \mathbb{E}_{z \sim p_{z(z)}} [\log(1 - D(G(z)))]$$

Discriminative Model D



- Fixed D , minimize $V(G)$:

$$\min_G V_D(G) = \min_G \left[\mathbb{E}_{z \sim p_{z(z)}} [\log(1 - D(G(z)))] \right]$$

Try to make $D(G(z)) = 1$

Stochastic Gradient

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)}))) .$$

Generative Adversarial Network

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

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

Update D

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

end for

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

Update G

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)}))).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Meaning of GAN Loss

$$\min_G \max_D V(D, G) \quad \xrightarrow{\text{same}} \quad \min_{G, D} JSD(p_{\text{data}} || p_g)$$

Objective function of GANs

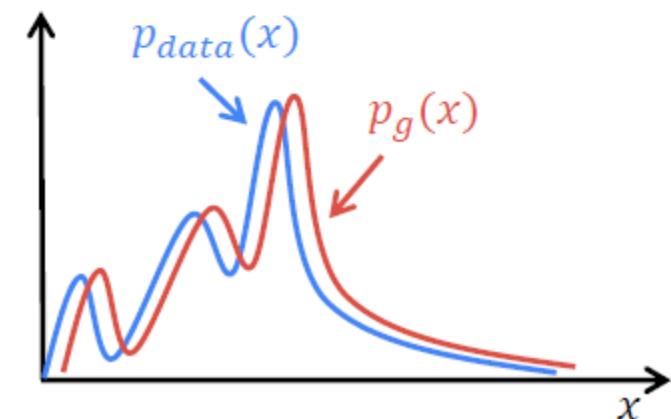
$$E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Jenson-Shannon divergence

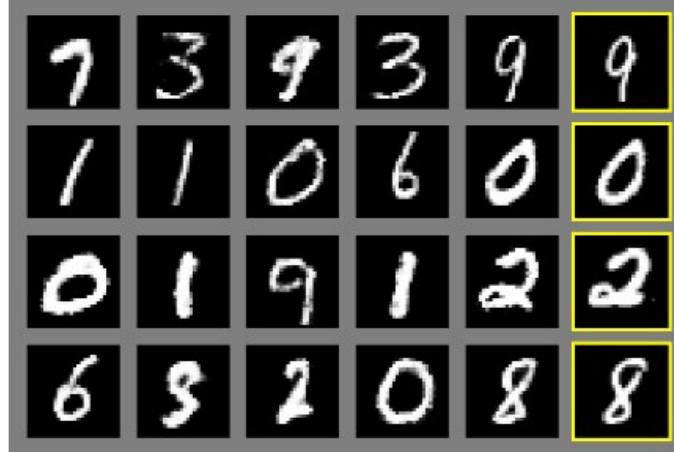
$$JSD(P || Q) = \frac{1}{2} KL(P || M) + \frac{1}{2} KL(Q || M)$$

where $M = \frac{1}{2}(P + Q)$

KL Divergence



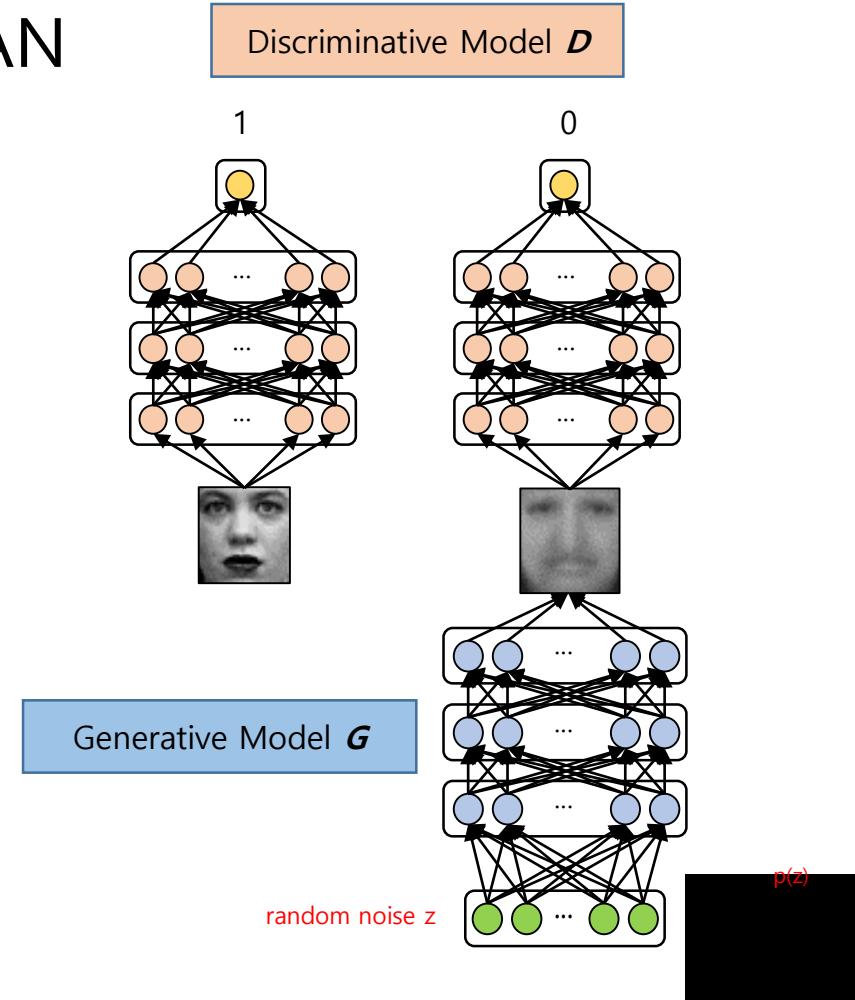
GAN Results



Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Network -ICLR 2016

DCGAN

- Deep Convolutional Network + GAN
- Tricks for stable training
- Experimental Analysis



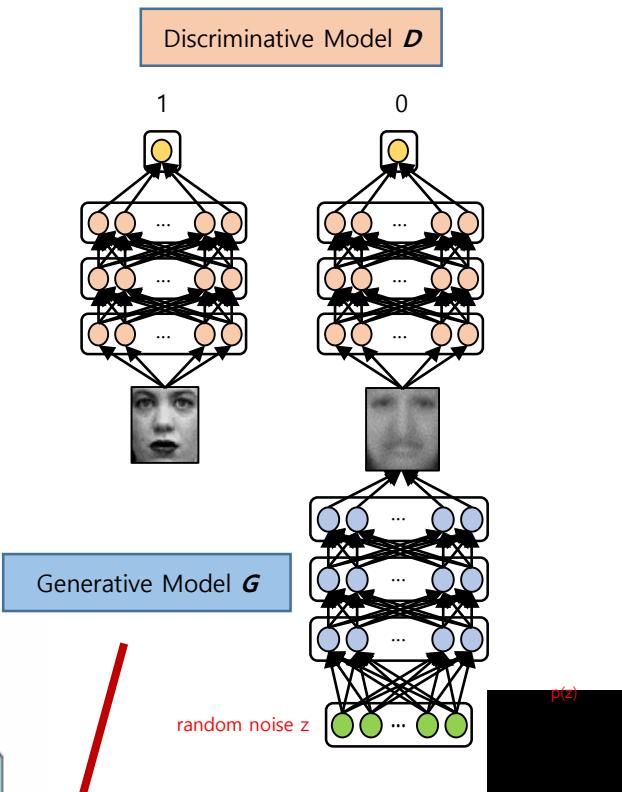
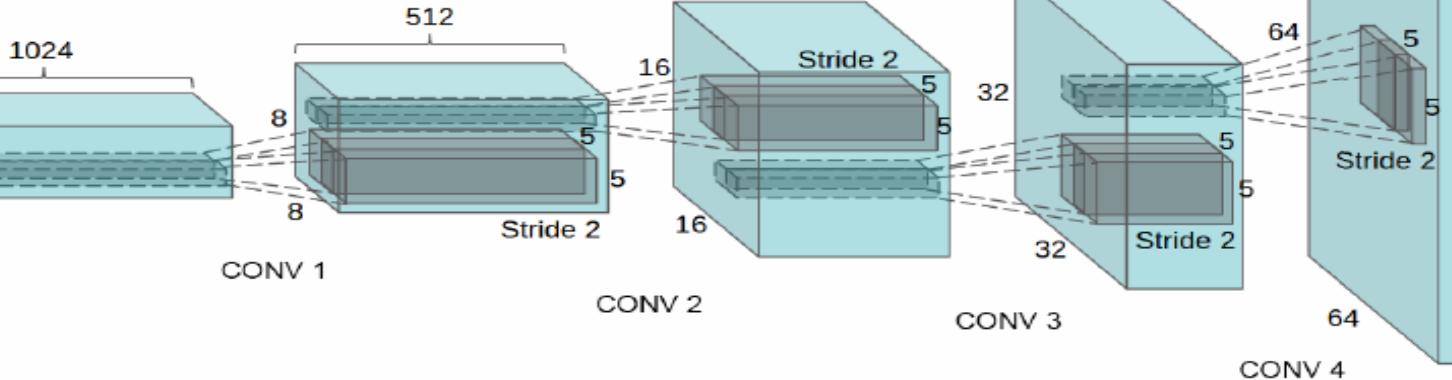
DCGAN

- Replace model's network to CNN
- Example of generator G
(same as D)

z :
uniform dist.



100 z → Project and reshape



Representation Learning

- Generator가 image를 외워서 보여주는 것이 아니어야 함
 - Memorization을 통한 1:1 mapping이 아니어야 함
- Generator의 input 공간인 latent space(z space)에서 움직일 때 급작스러운 변화(sharp transition)이 일어나지 않아야 함

Guidelines for Stable DCGAN

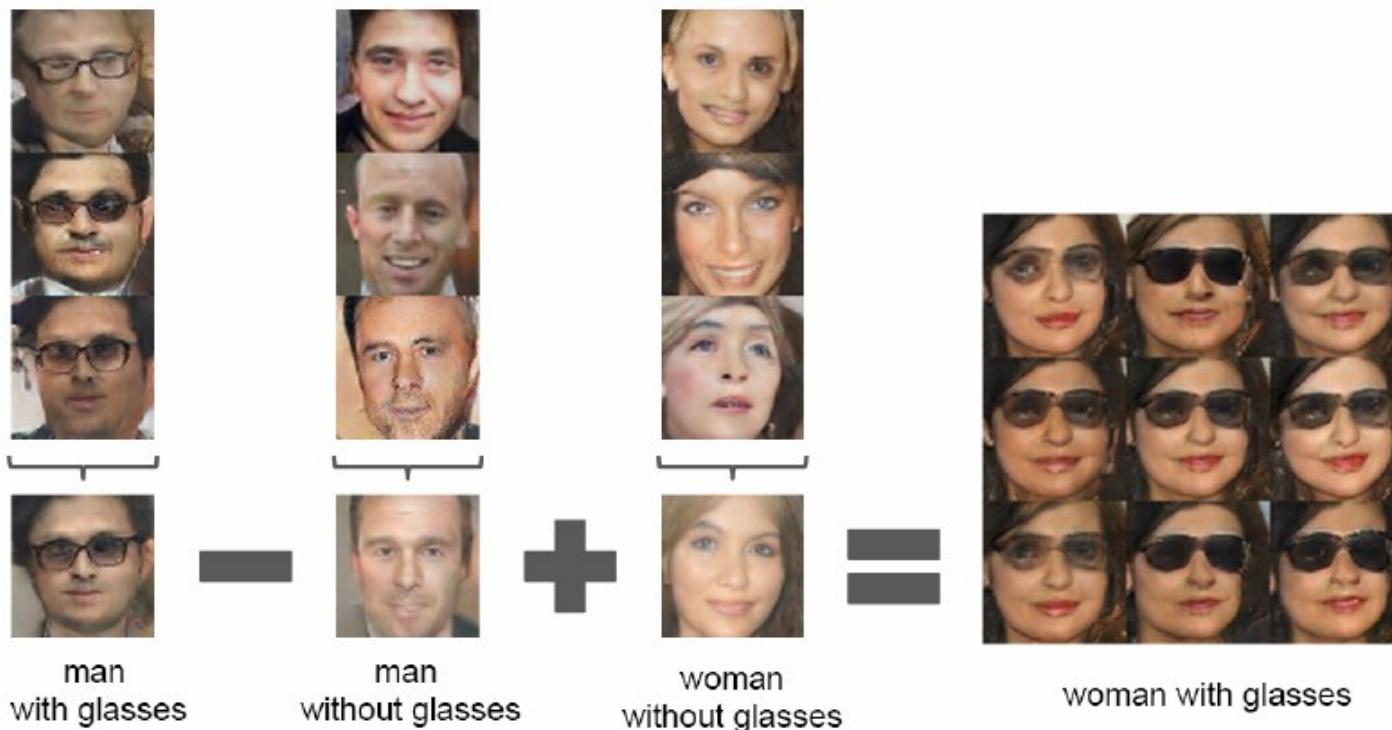
- Replace any pooling layers with strided convolutions
- Use batchnorm in both the generator and discriminator
- Remove fully connected hidden layers for deeper architectures
- Use ReLU activation in generator for all layers except for output, which uses Tanh
- Use LeakyReLU activation in the discriminator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

Experiments

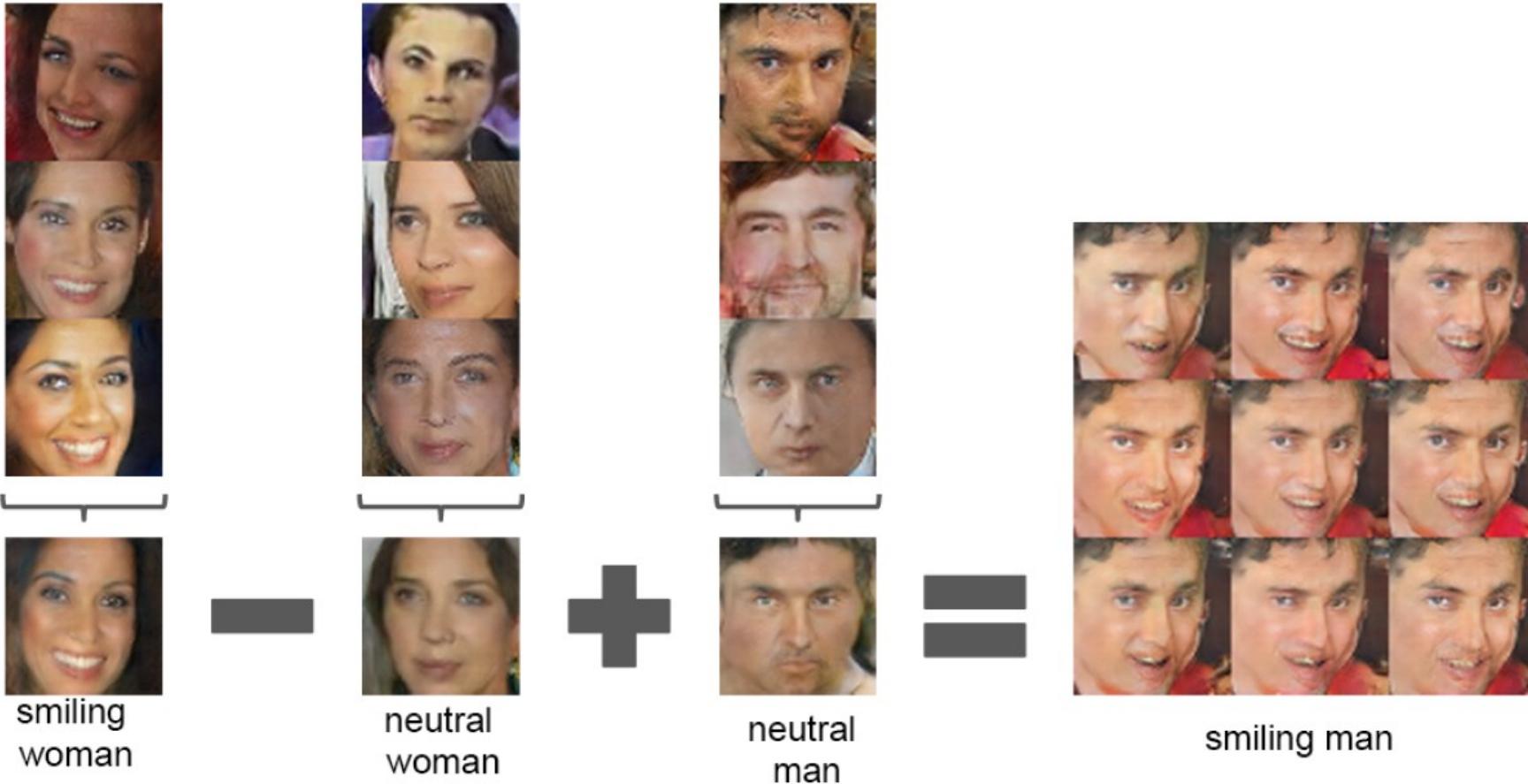


Experiments

- Faces – scraped human face image from web
 - 3 million images from 10,000 people.
- Vector arithmetic



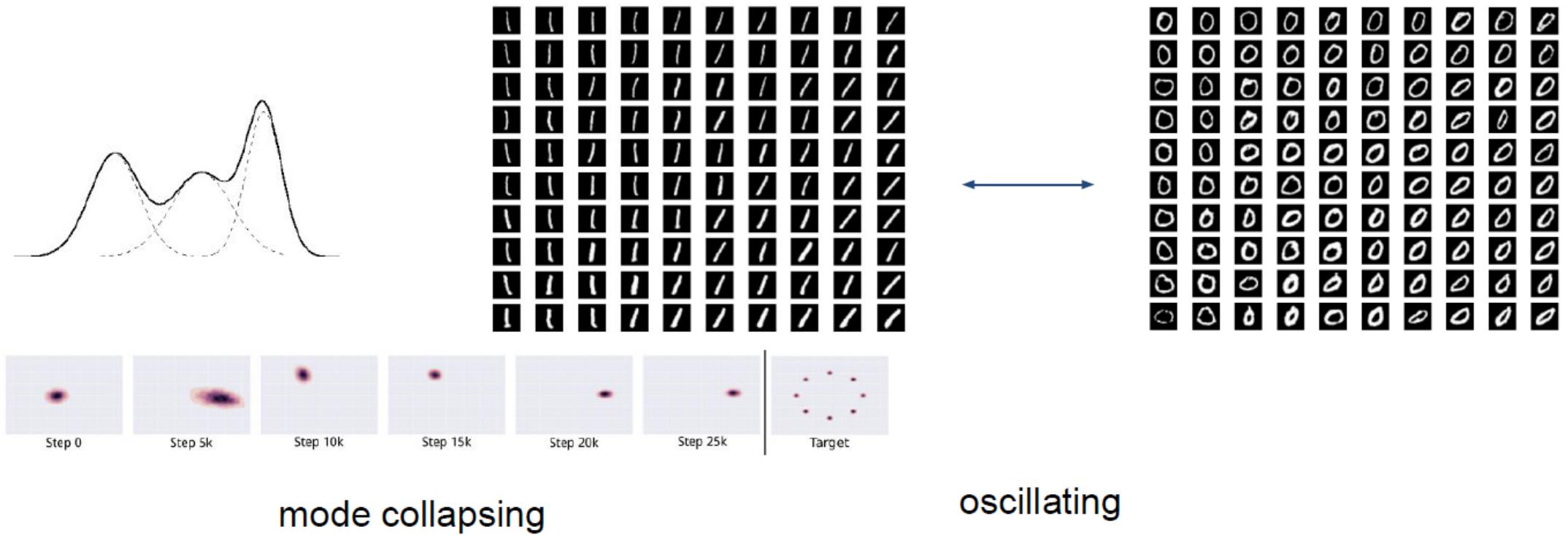
Experiments



Experiments



Pitfall of GAN



Conditional GAN

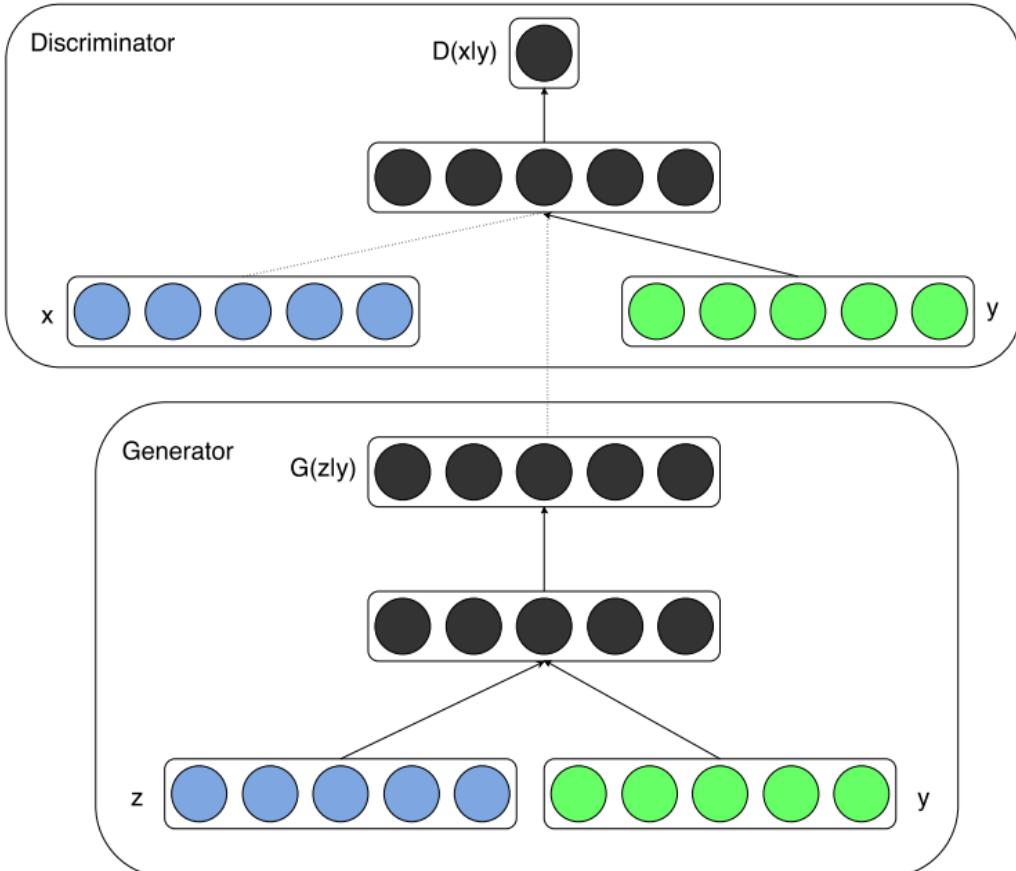


Figure 1: Conditional adversarial net

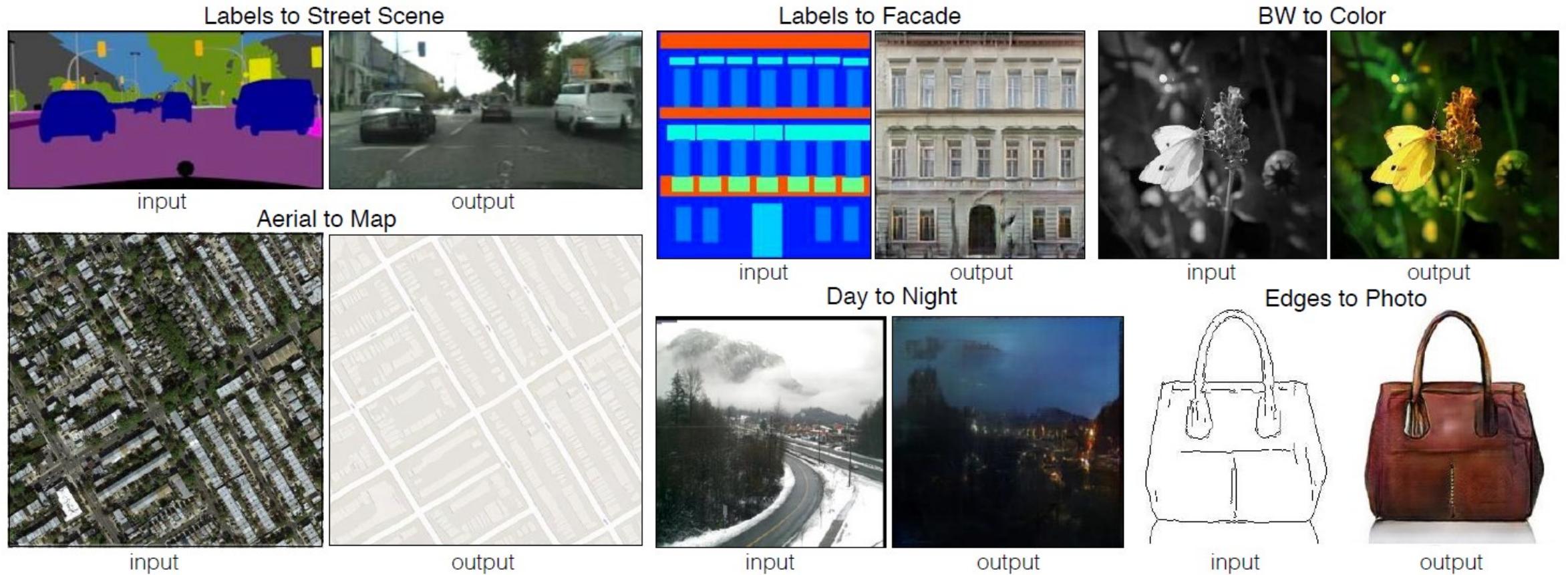
Fig 2 shows some of the generated samples. Each row is conditioned on one label and each column is a different generated sample.



Figure 2: Generated MNIST digits, each row conditioned on one label

Image-to-Image Translation with Conditional Adversarial Networks

Pix2Pix



Pix2Pix

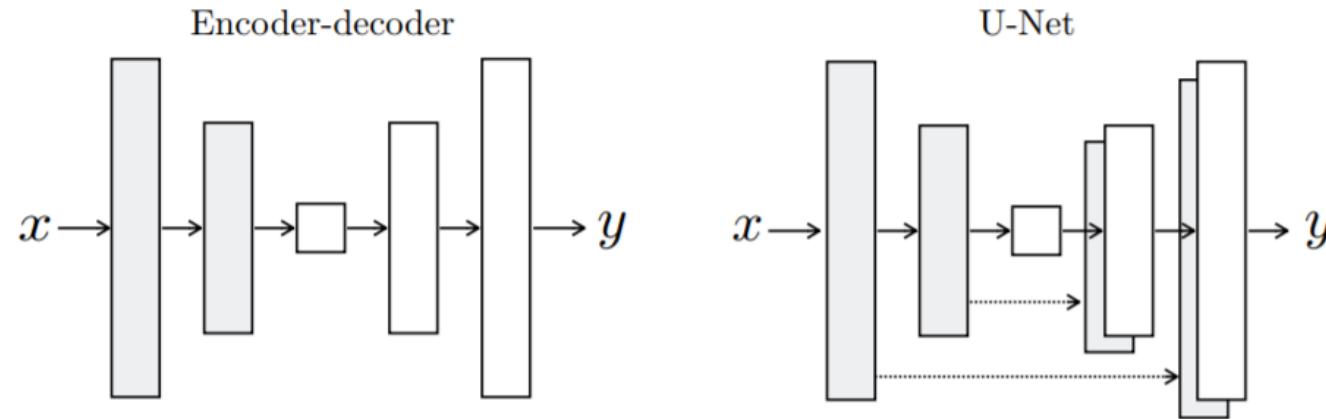


Figure 3: Two choices for the architecture of the generator. The “U-Net” [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

+ L1 loss function

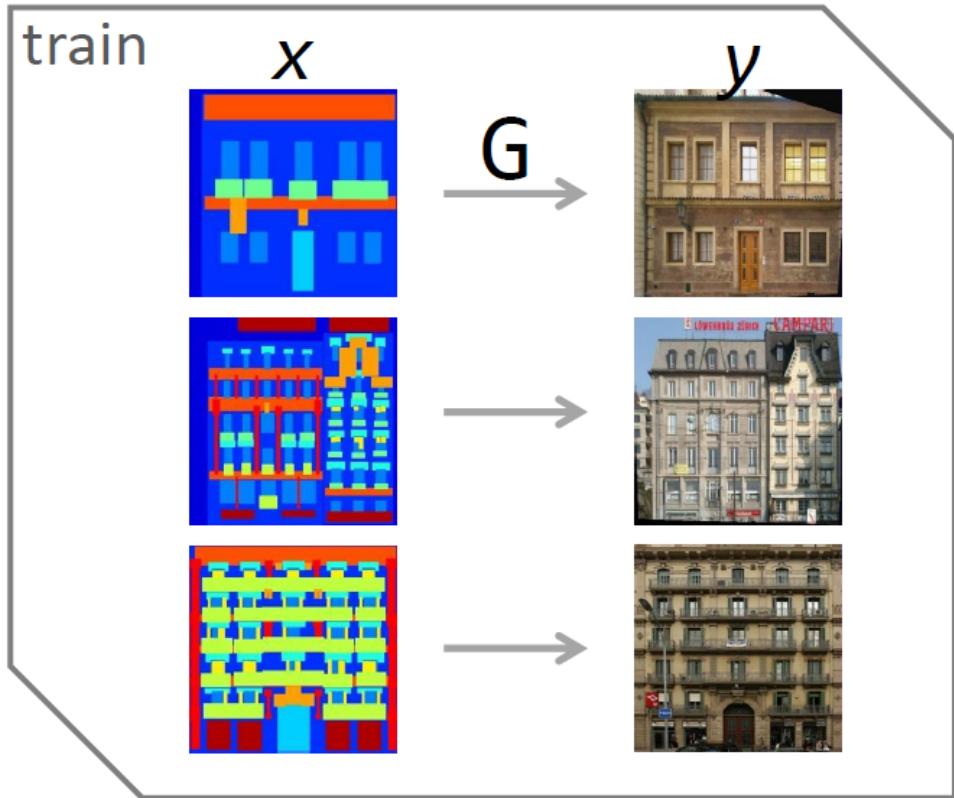
$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

Low-freq correctness

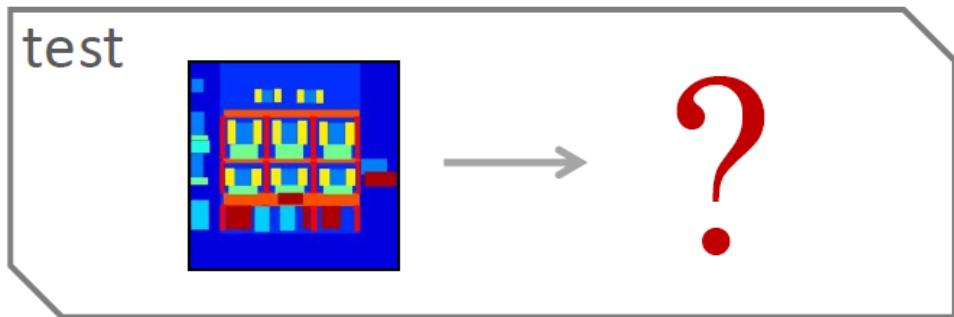
+ PatchGAN

High-freq correctness

Pix2Pix



- Supervised
- loss: Minimize the difference between output $G(x)$ and ground truth y



L1 Loss – Stable Guide Force

Loss: Minimize the difference between output $G(x)$ and the ground truth y

$$\sum_{(x,y)} \|y - G(x)\|_1$$



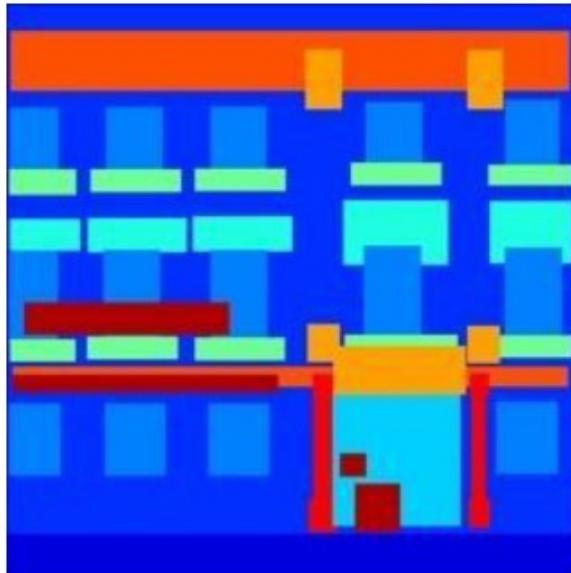
Input



Output



Ground Truth



Input



Output

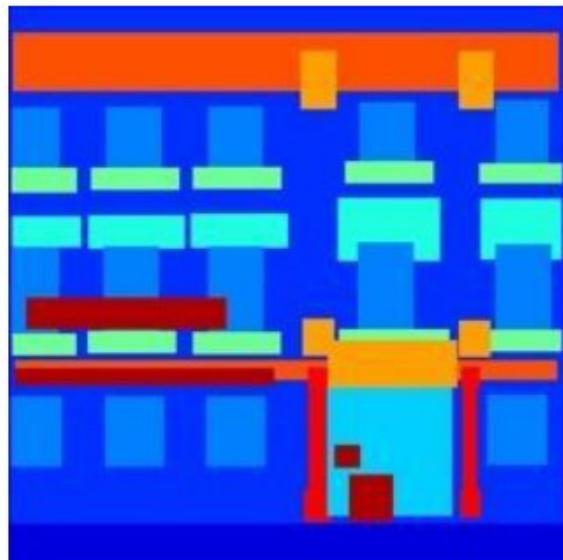


Ground Truth

Adding GAN Loss

Loss: Minimize the difference between and output $G(x)$ and ground truth y

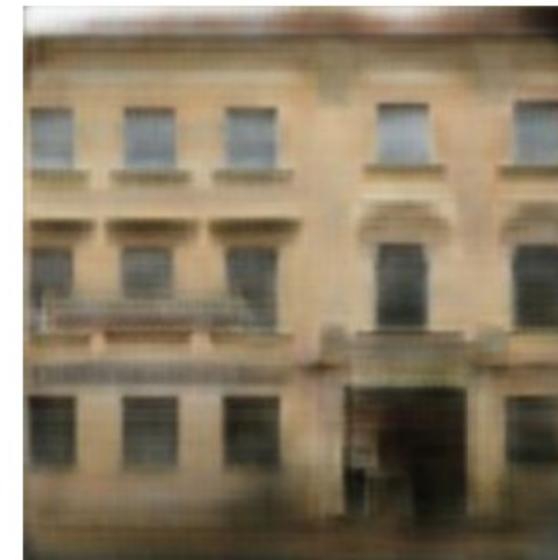
$$\sum_{(x,y)} \|y - G(x)\|_1 + L_{GAN}(G(x), y)$$



Input



Ground Truth



L1 loss only



L1+GAN loss

Image to Image Translation



Figure 8: Example results on Google Maps at 512x512 resolution (model was trained on images at 256x256 resolution, and run convolutionally on the larger images at test time). Contrast adjusted for clarity.

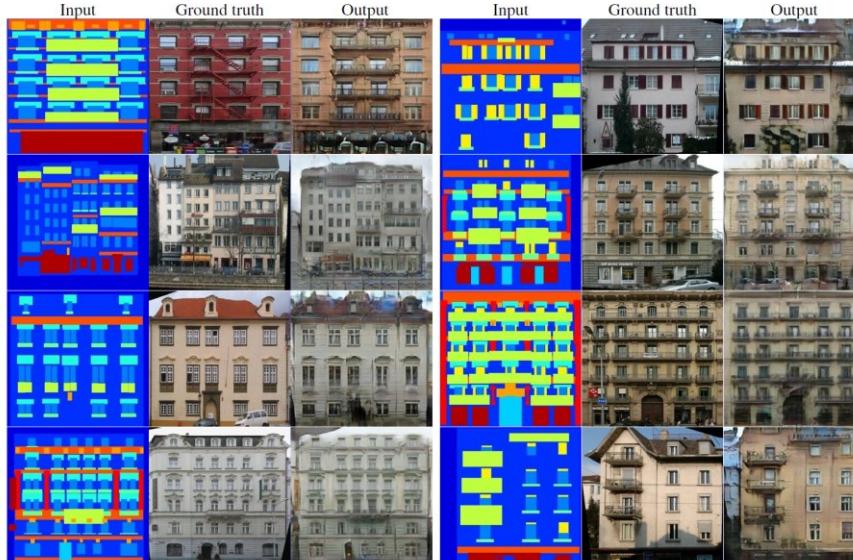


Figure 12: Example results of our method on facades labels→photo, compared to ground truth

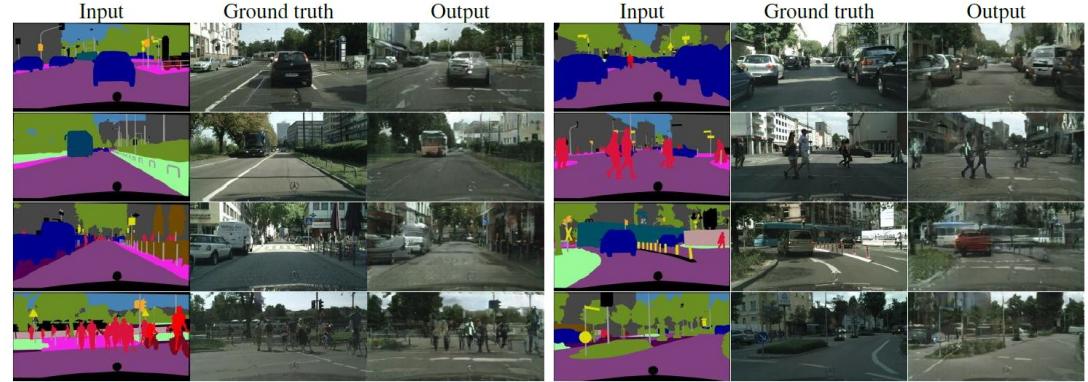


Figure 11: Example results of our method on Cityscapes labels→photo, compared to ground truth.



Figure 13: Example results of our method on day→night, compared to ground truth.

Image to Image Translation



Figure 15: Example results of our method on automatically detected edges→shoes, compared to ground truth.



Figure 14: Example results of our method on automatically detected edges→handbags, compared to ground truth.



Figure 16: Example results of the edges→photo models applied to human-drawn sketches from [10]. Note that the models were trained on automatically detected edges, but generalize to human drawings.

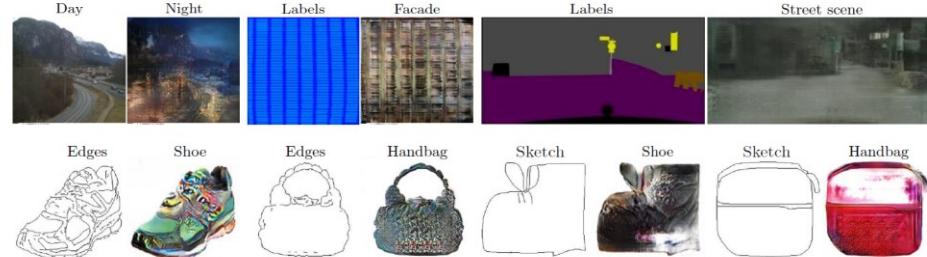


Figure 17: Example failure cases. Each pair of images shows input on the left and output on the right. These examples are selected as some of the worst results on our tasks. Common failures include artifacts in regions where the input image is sparse, and difficulty in handling unusual inputs. Please see <https://phillipi.github.io/pix2pix/> for more comprehensive results.

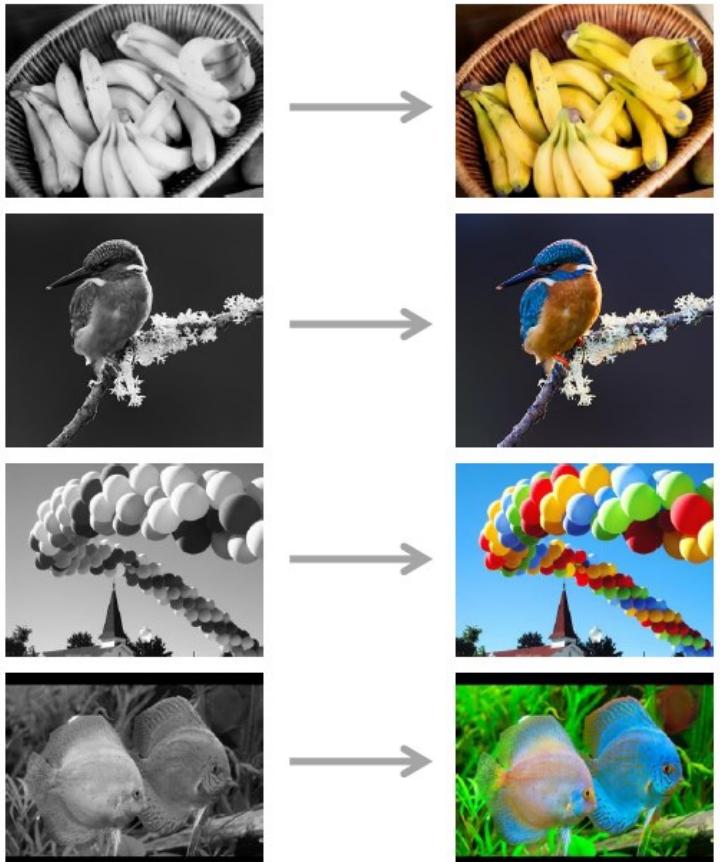
Let's Try

- <https://affinelayer.com/pixsrv/>
- <https://github.com/affinelayer/pix2pix-tensorflow/blob/master/pix2pix.py>

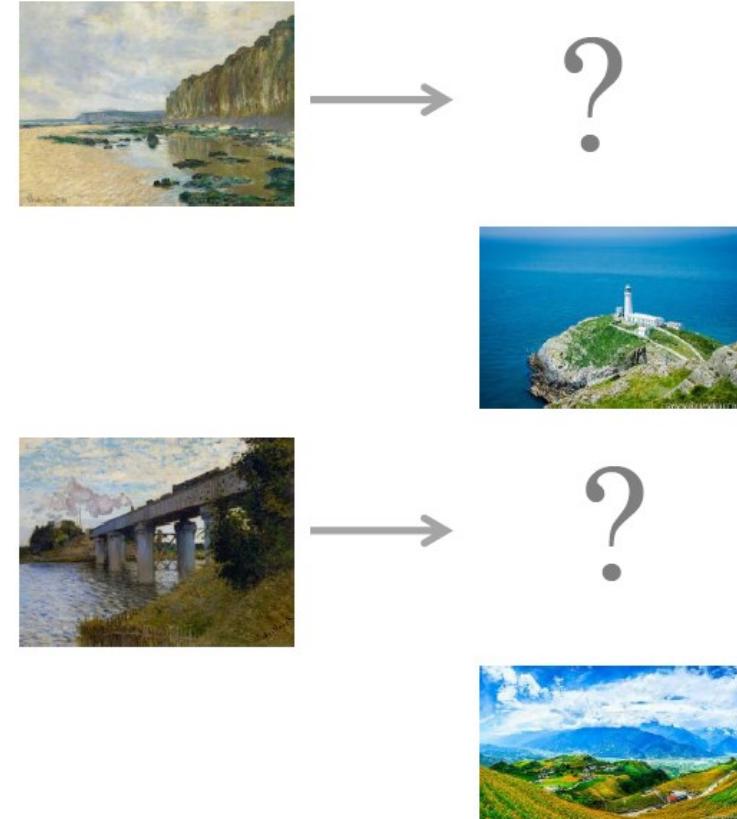
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN

pix2pix

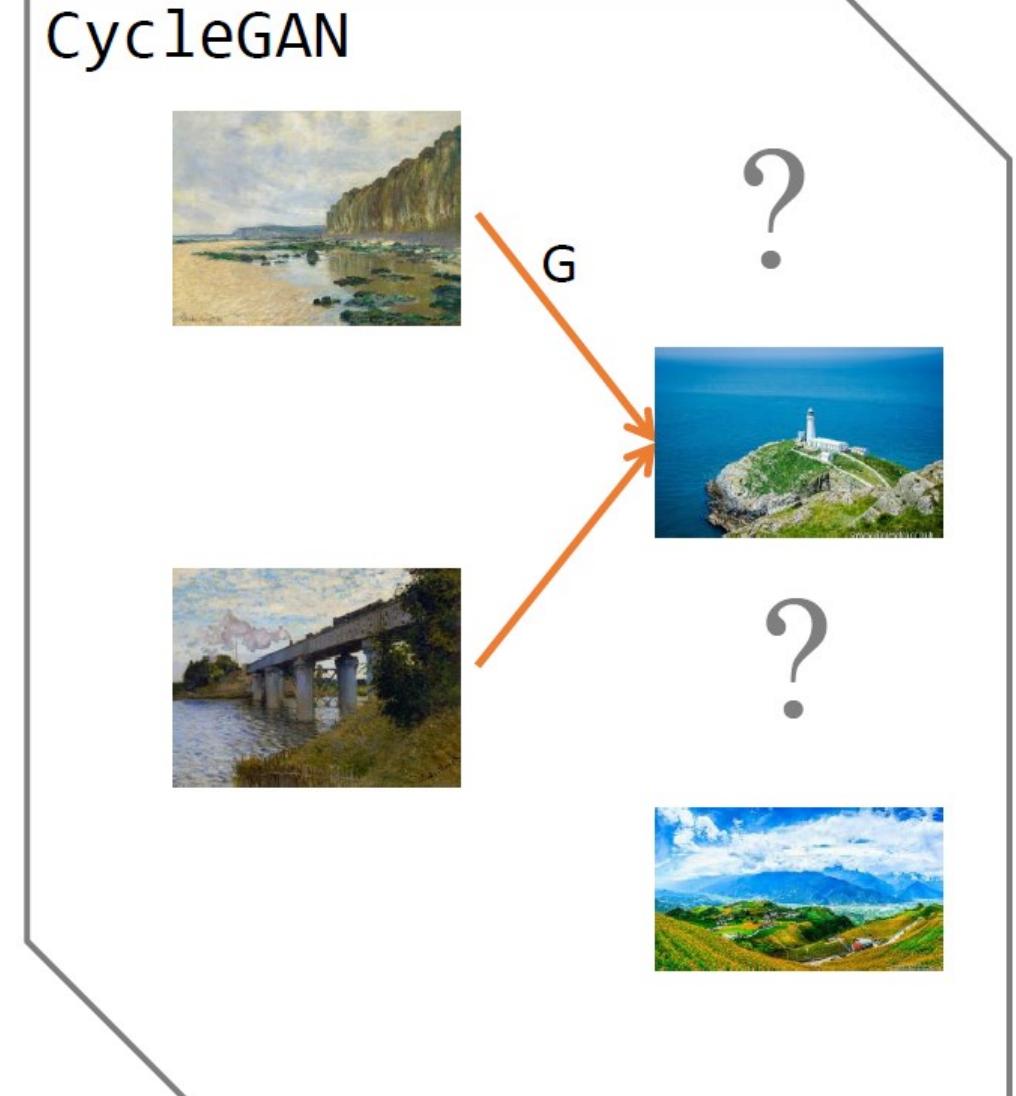


CycleGAN



CycleGAN – Mode Collapsing

Loss: $L_{GAN}(G(x), y)$
 $G(x)$ should just look photorealistic



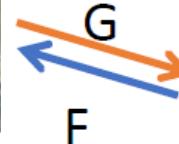
CycleGAN

LOSS

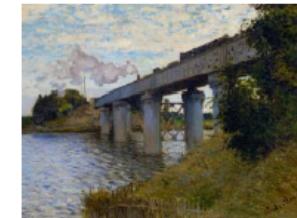
$$L_{GAN}(G(x), y) + \|F(G(x)) - x\|_1$$

$G(x)$ should just look photorealistic
and $F(G(x))$ should be $F(G(x)) = x$,
where F is the inverse deep network

CycleGAN



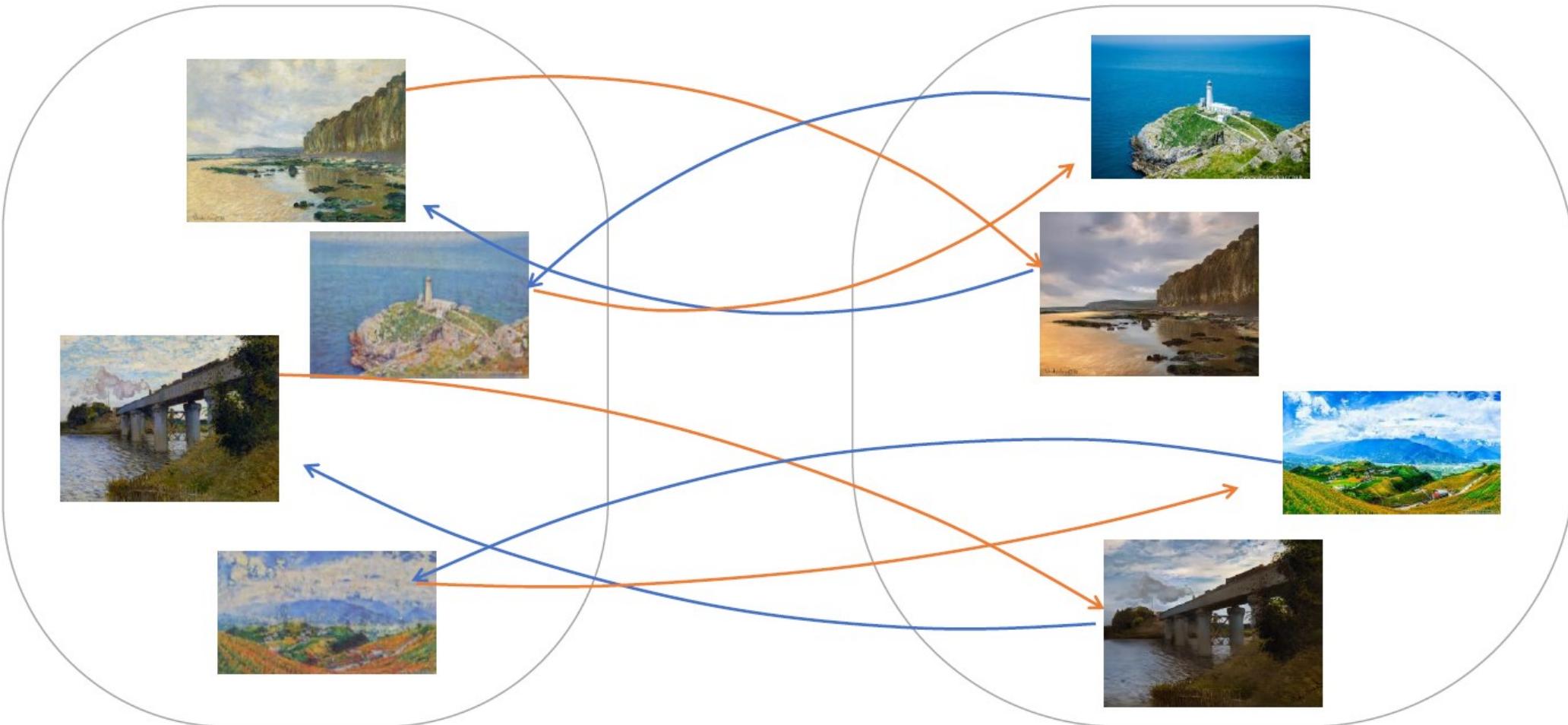
?



?

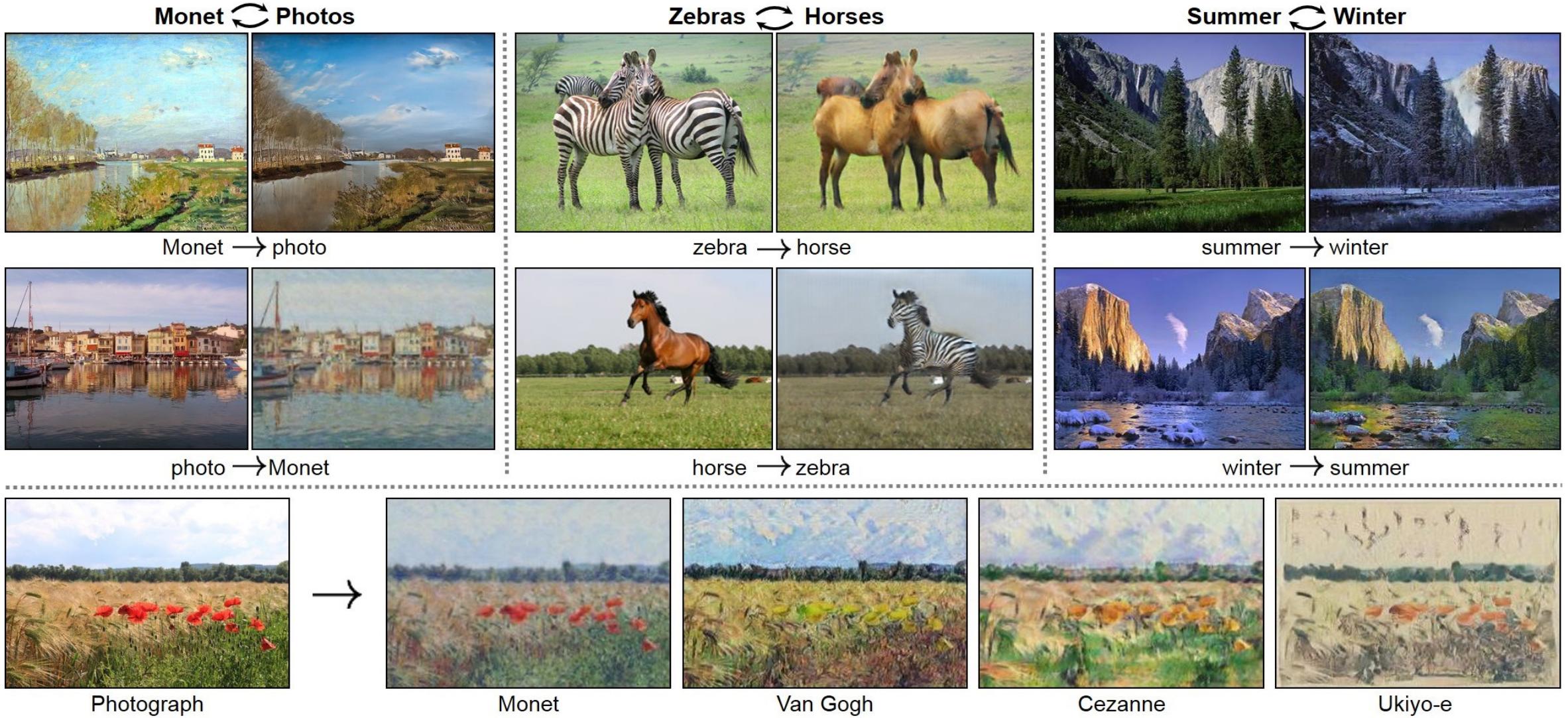


CycleGAN – Loss Function



$$L_{GAN}(G(x), y) + \|F(G(x)) - x\|_1 + L_{GAN}(F(y), x) + \|G(F(y)) - y\|_1$$

Results



Results



Results



Reconstructed Images

Original CG



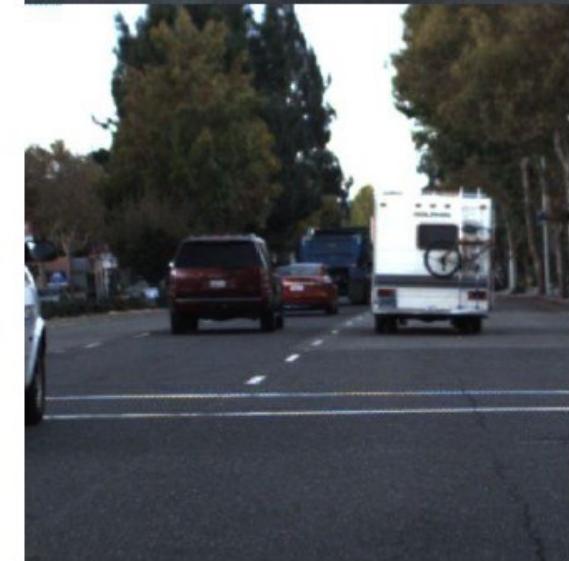
Fake Photo



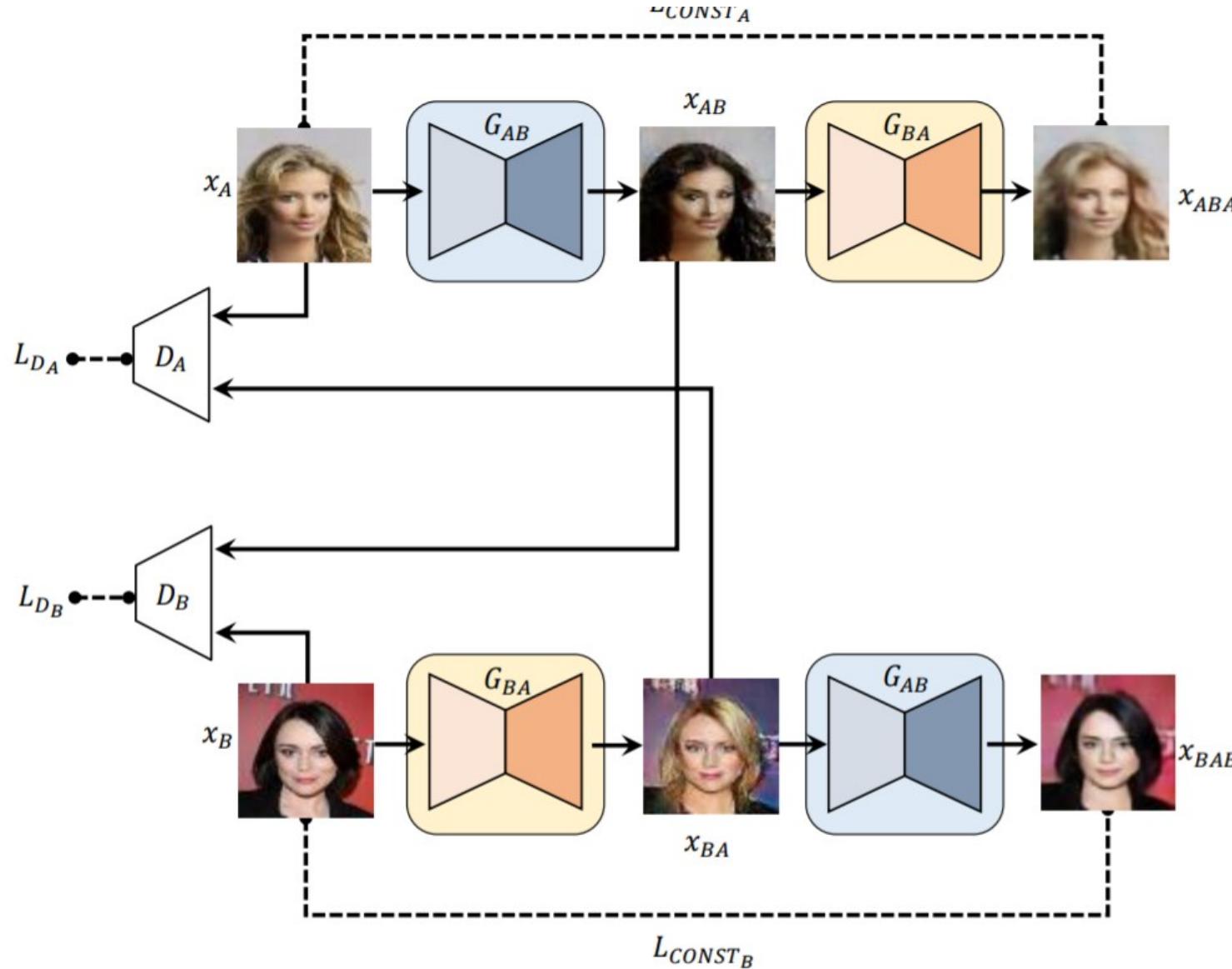
Reconstruction



Real Photo



DiscoGAN



Different Result

CycleGAN

Cat → Dog



Dog → Cat



DiscoGAN

Dog → Cat



Cat → Dog

