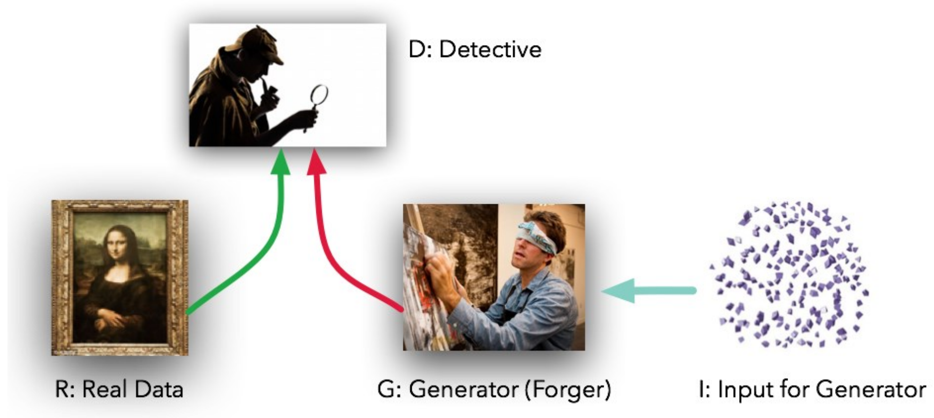
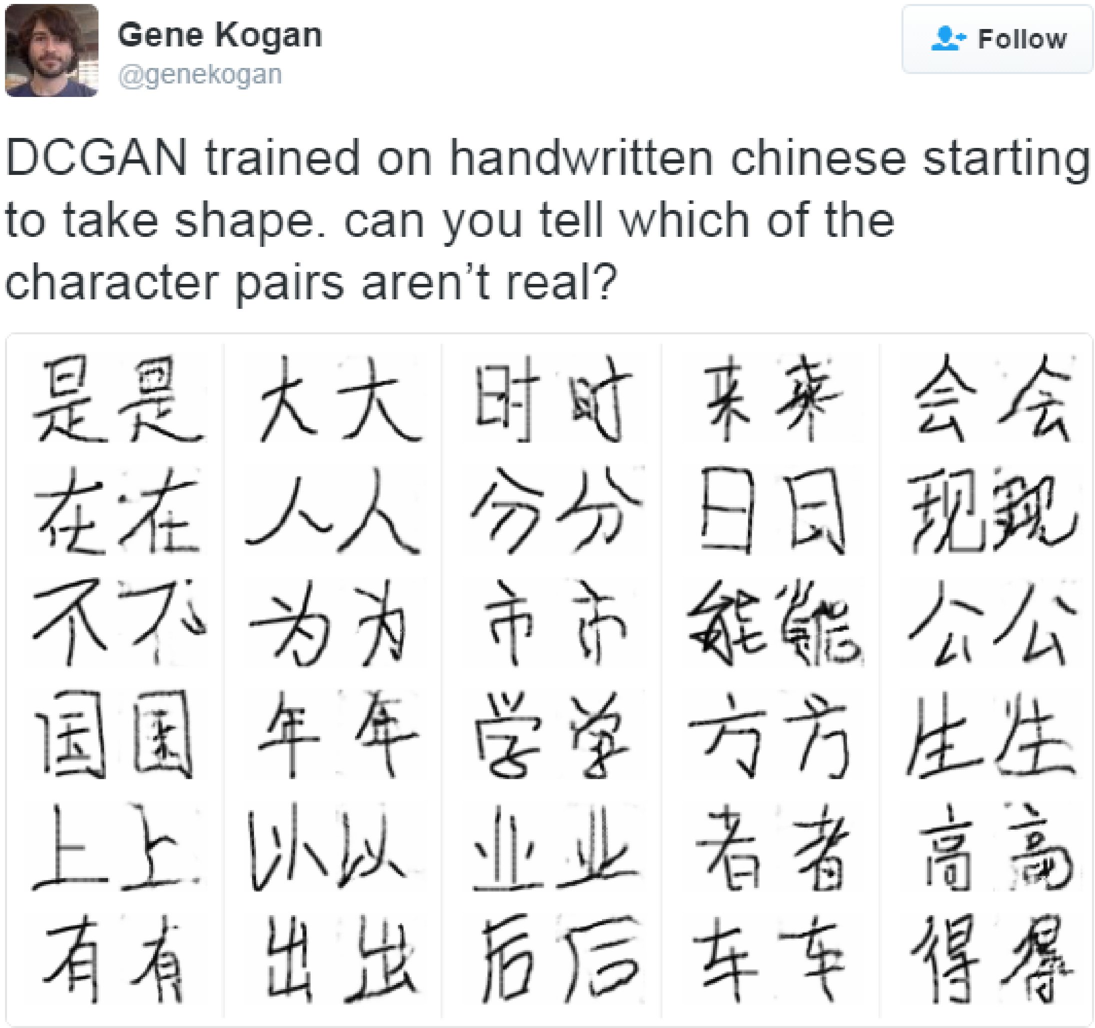
Generative Adversarial Network

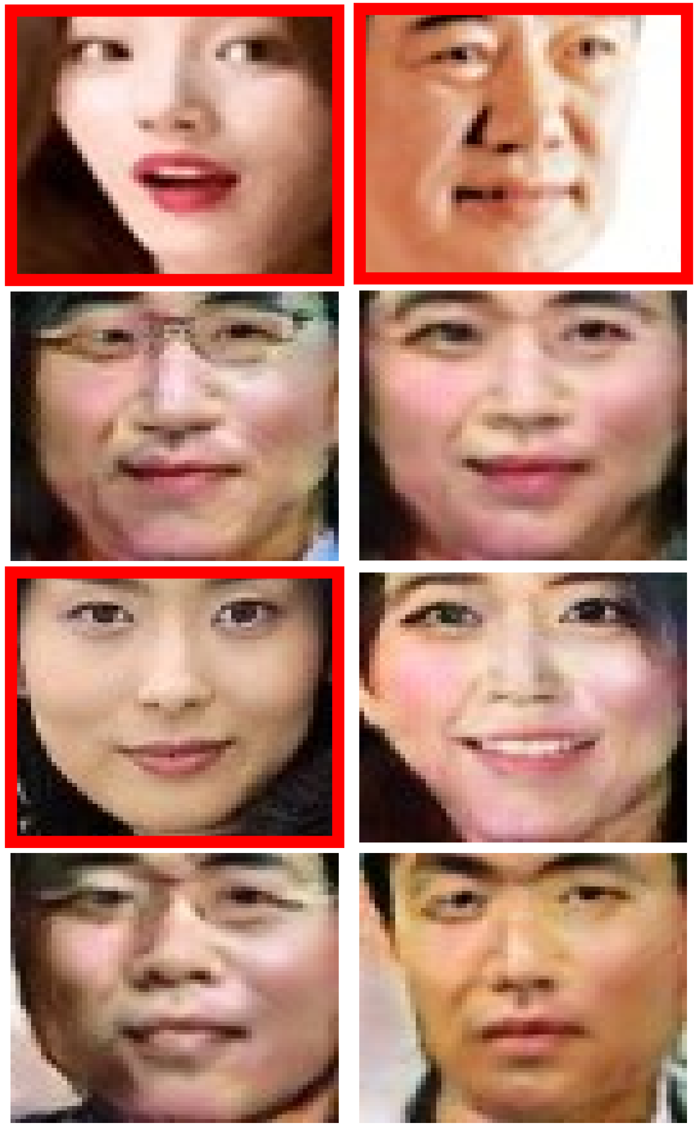


Fast Campus
Start Deep Learning with TensorFlow

Generative Adversarial Network
-NIPS 2014

Turing Test


Gene Kogan
@genekogan
Follow
DCGAN trained on handwritten chinese starting to take shape. can you tell which of the character pairs aren't real?

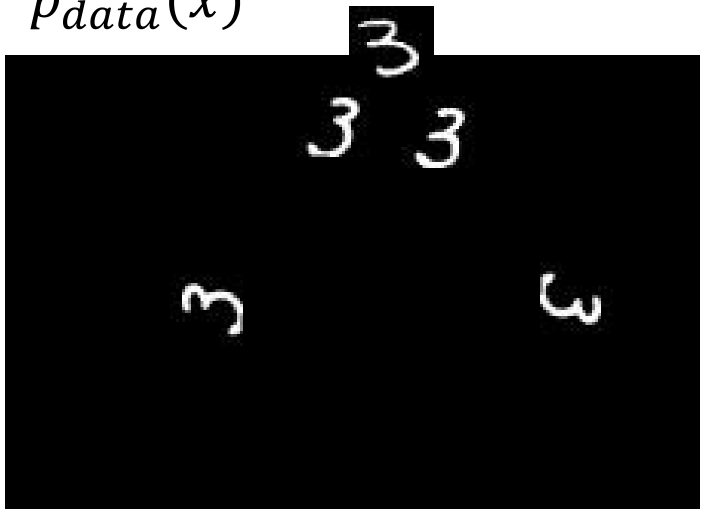


Generative Model

- Data:

…

$p_{data}(x)$



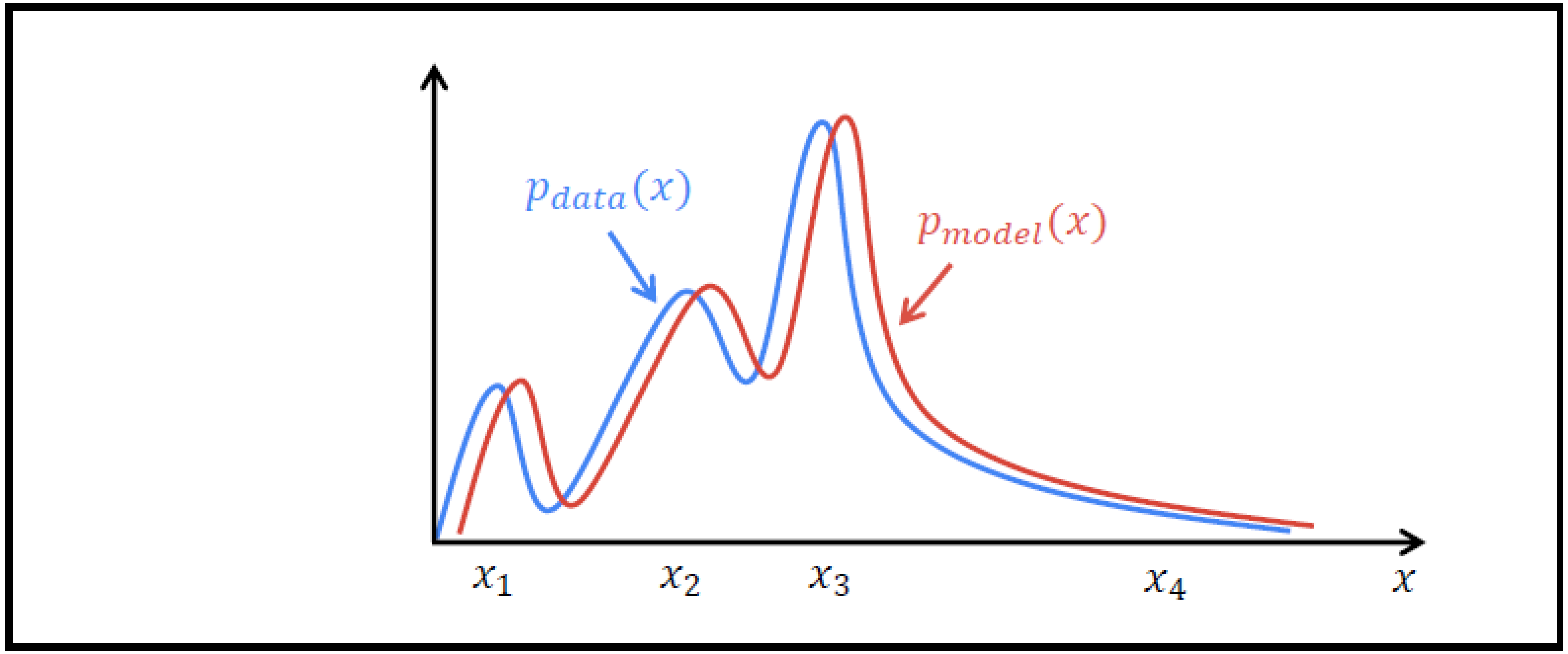
- Generative model: $p_G(x;\theta_G)$
  - True data distribution: $p_{data}(x)$
  - Train $p_G(x) \approx p_{data}(x)$

Generative Model

Distribution of images generated by the model

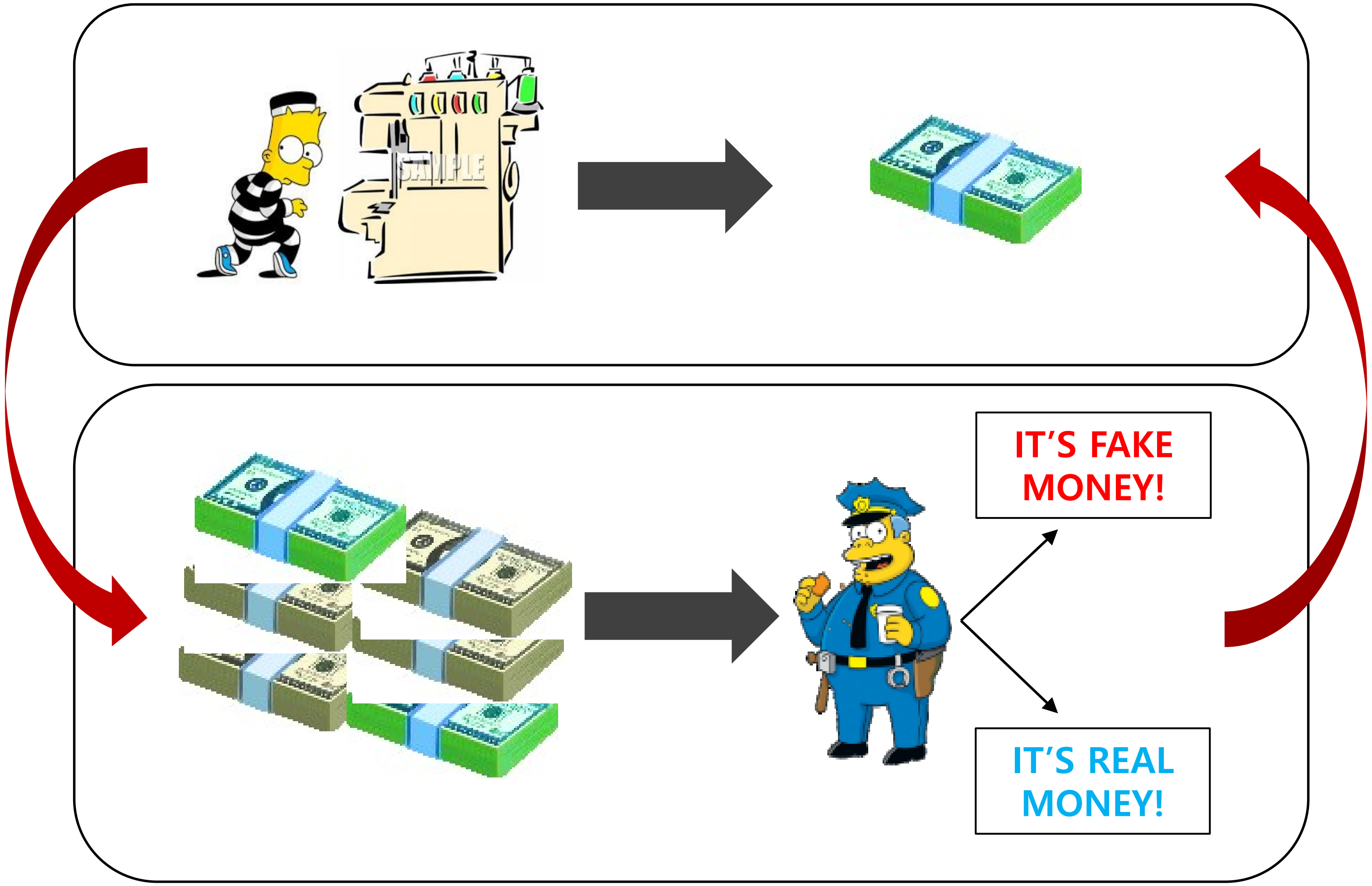
The goal of the generative model is to find a $p_{model}(x)$ that approximates $p_{data}(x)$ well.

Distribution of actual images

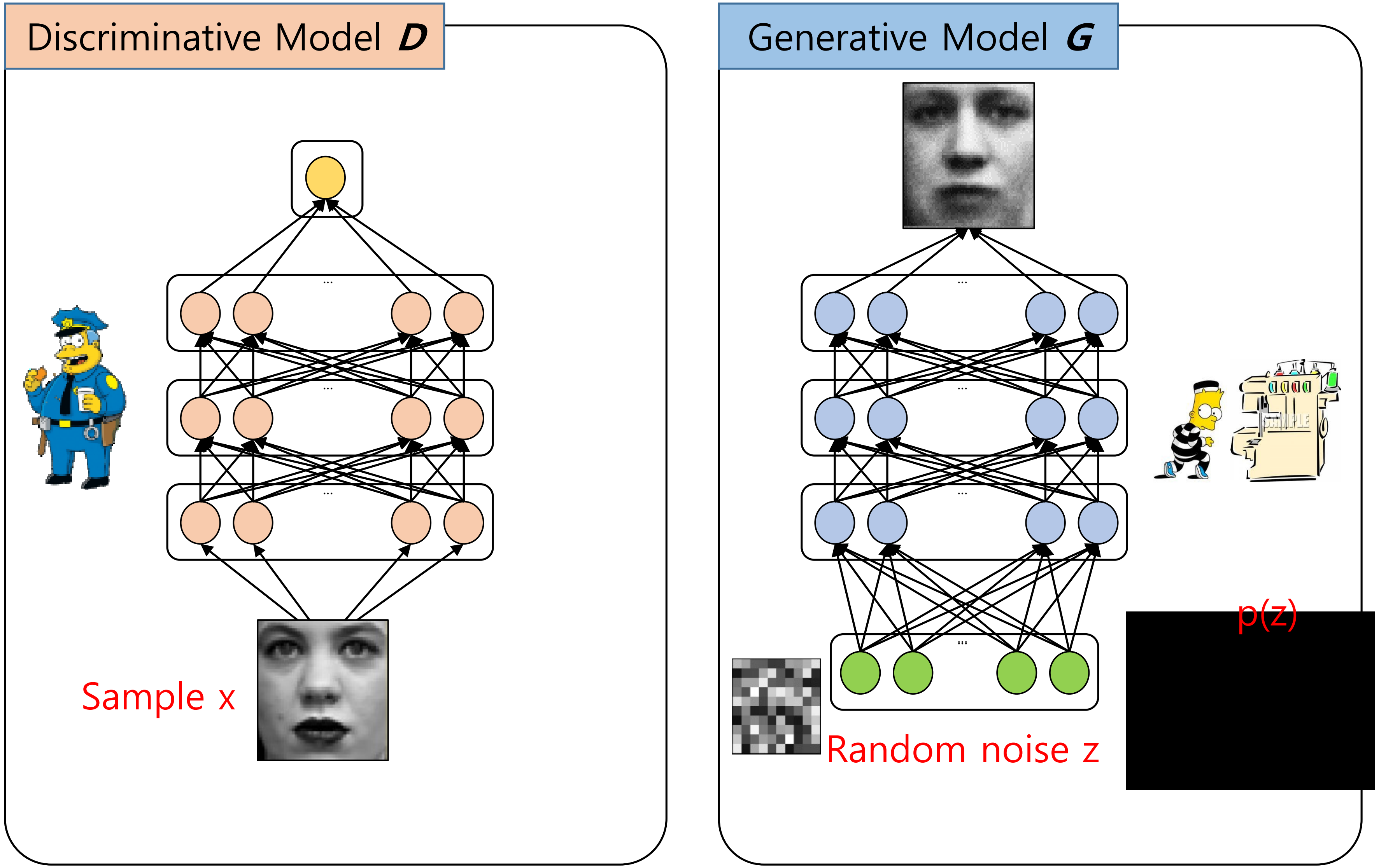


Generative Adversarial Network

- Counterfeiters vs Police Game



Generative Adversarial Network

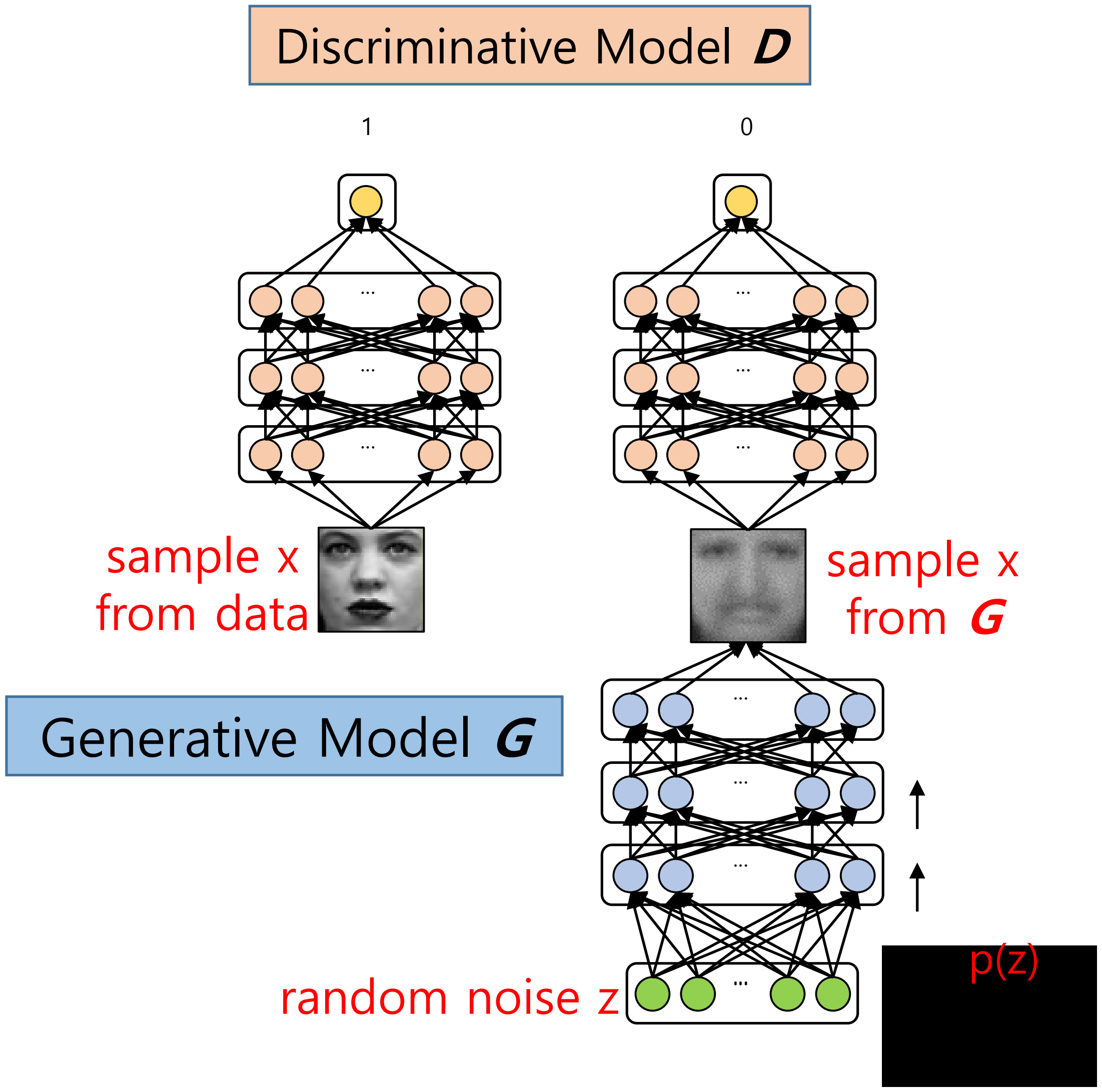


Generative Adversarial Network

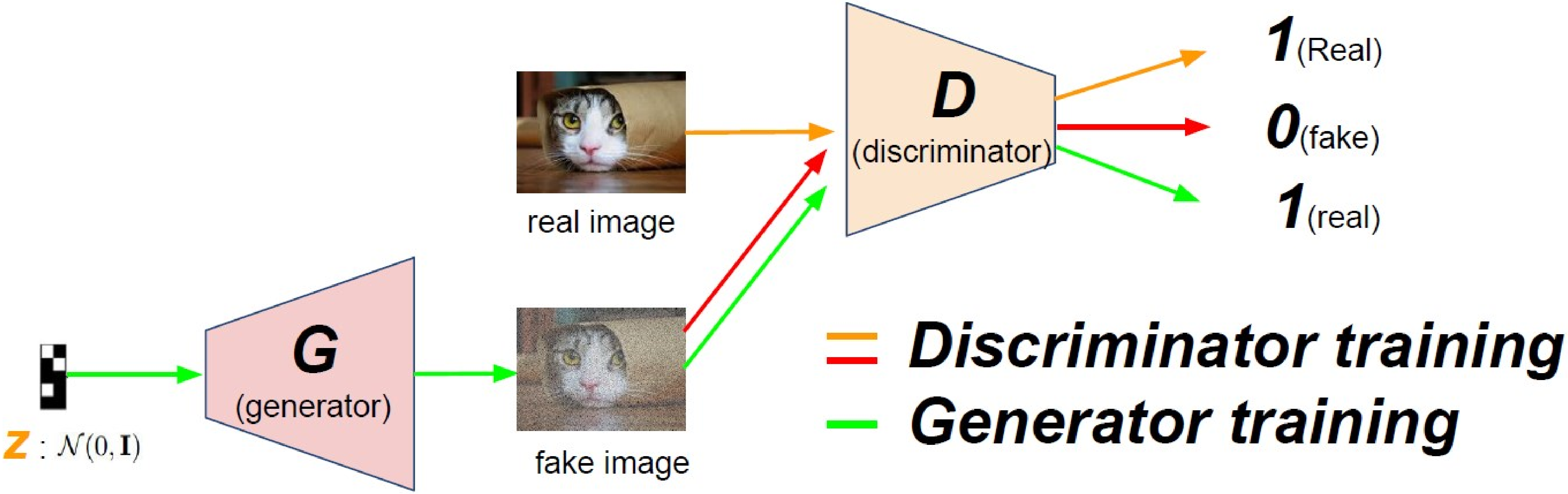
- Discriminative Model D
  - Try to classify the sample x
  - D(x)=1 when x from Data
  - D(x)=0 when x from G (generator)

Differentiable function represented by a multilayer perceptron with parameters

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



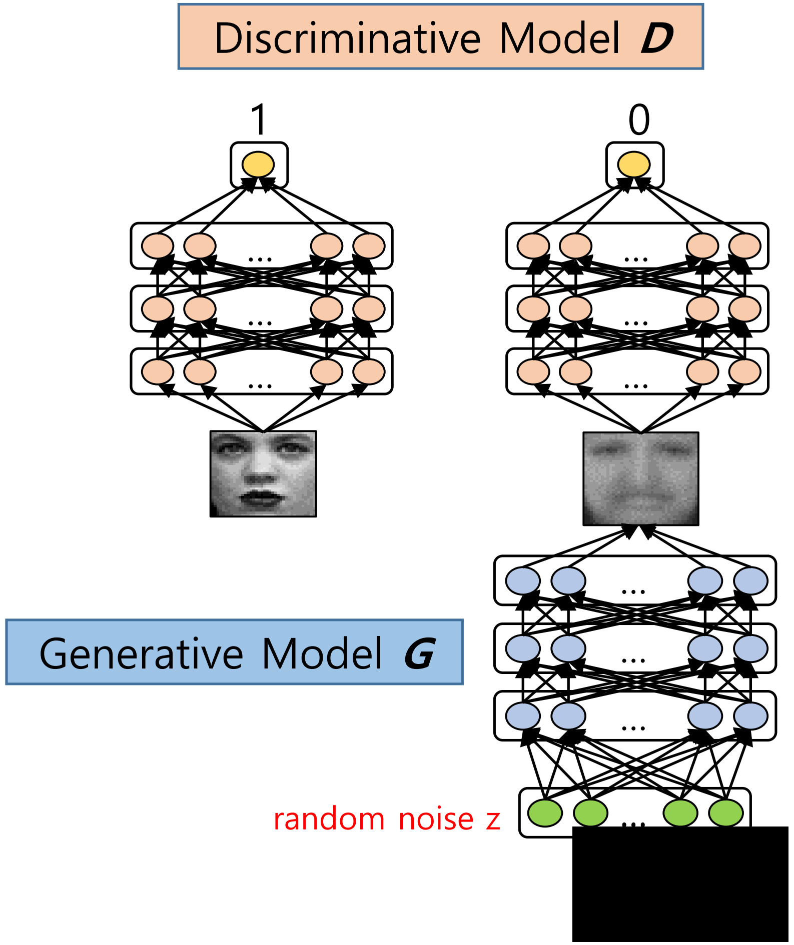
Training GAN



Generative Adversarial Network

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

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



[redacted]

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:

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

  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:

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

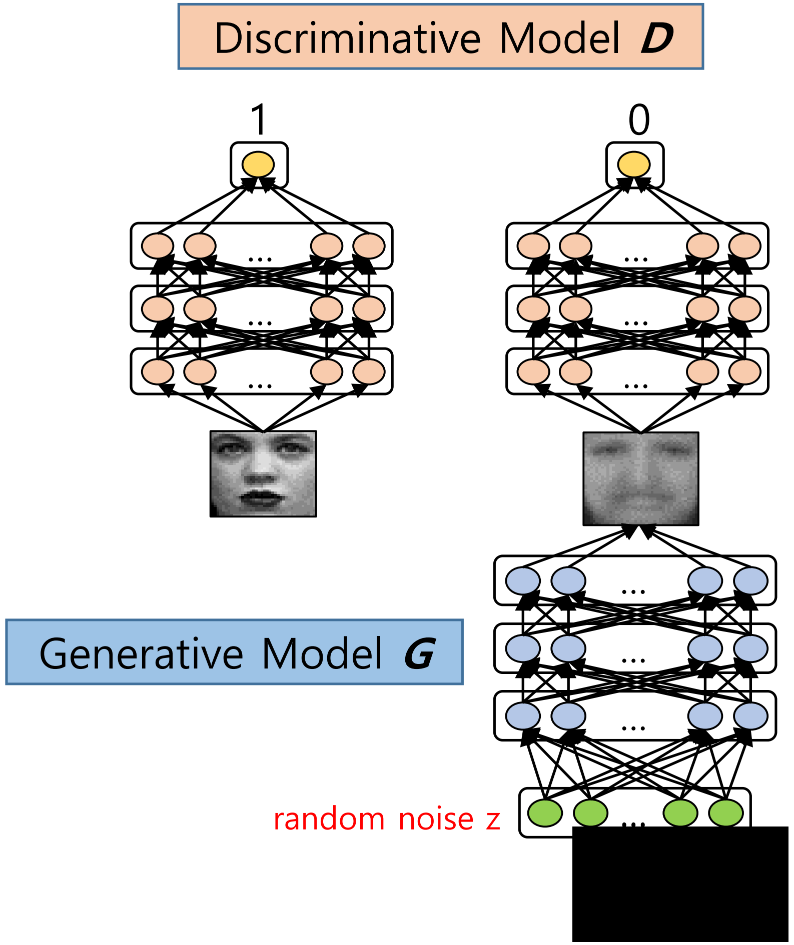
end for

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

Generative Adversarial Network

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

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



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

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

- Binary Classification (logistic loss):
  - Sample from data: label=1
  - Sample from generator: label = 0

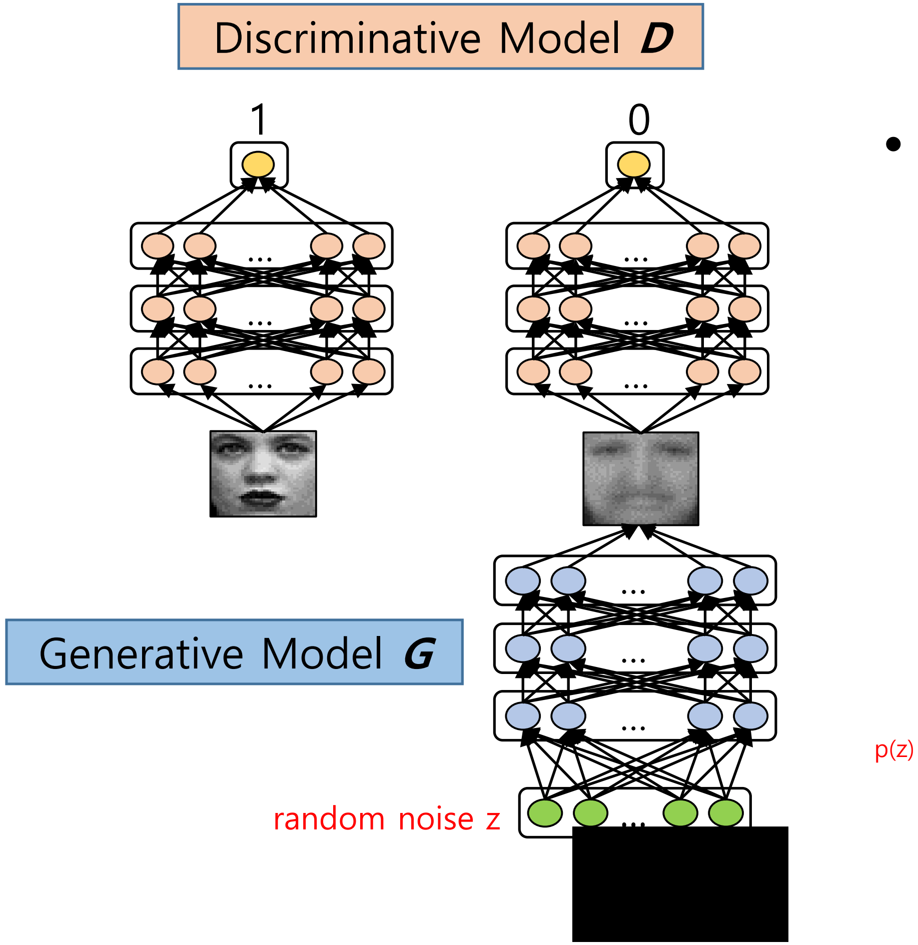
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

Stochastic Gradient

Generative Adversarial Network

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

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



- Fixed D, minimize V(G):

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

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

Stochastic Gradient

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

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

Update D

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

  end for

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

Update G

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

end for

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

Meaning of GAN Loss

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

Objective function of GANs

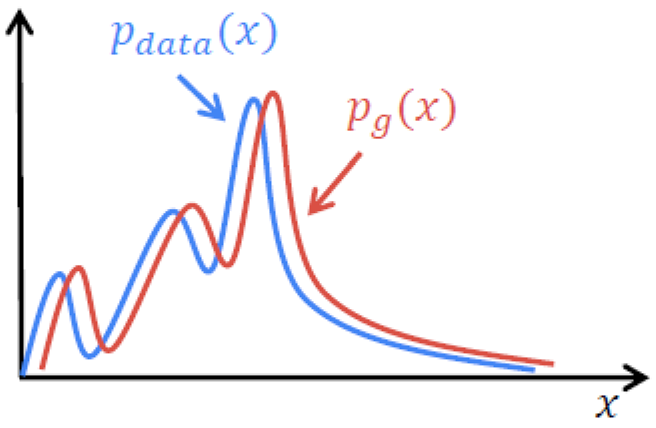
$$E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

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



Non-Saturating Game

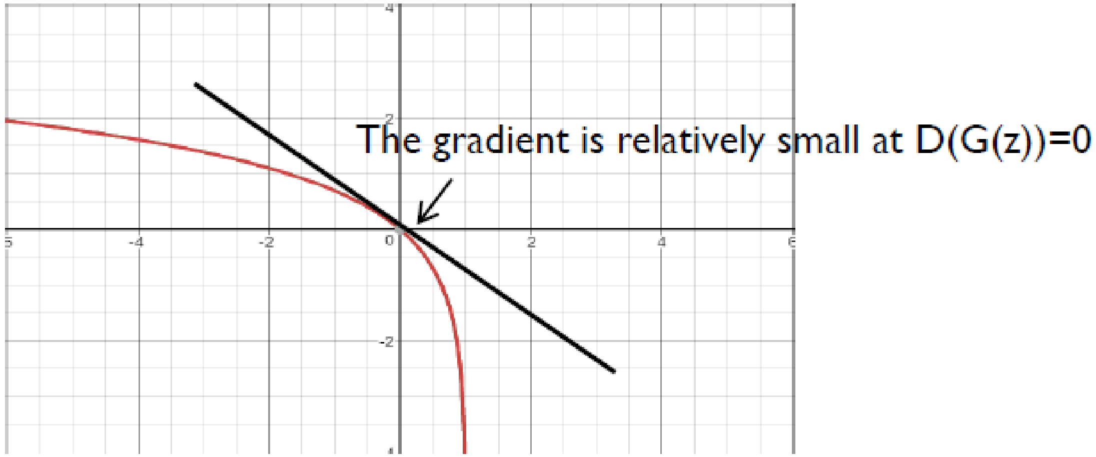
$$\min_G E_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

Objective function of G

At the beginning of training, the discriminator can clearly classify the generated image as fake because the quality of the image is very low.

This means that D(G(z)) is almost zero at early stages of training.

Images created by the generator at the beginning of training



$$y = \log(1 - x)$$

Non-Saturating Game

$$\cancel{\min_G E_{z\sim p_z(z)}[\log(1 - D(G(z))]}$$

↓ Modification (heuristically motivated)

$$\max_G E_{z\sim p_z(z)}[\log D(G(z))]$$

```
# tensorflow
tf.losses.sigmoid_cross_entropy()

# pytorch
nn.BCELoss()
```

- Practical Usage

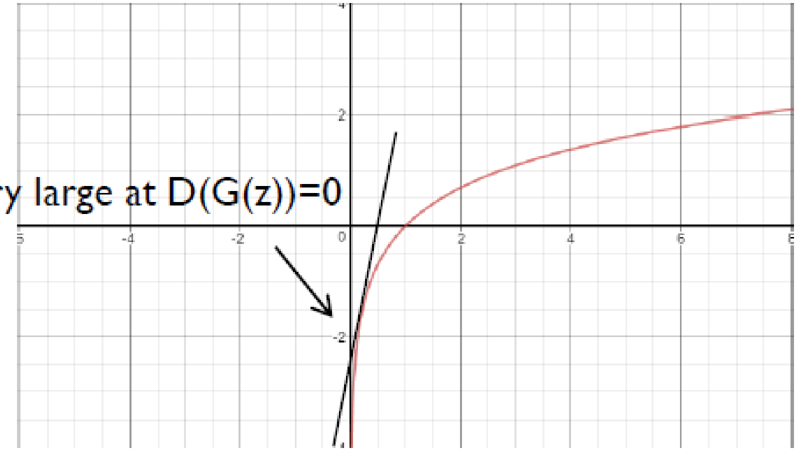
Use binary cross entropy loss function with fake label (1)

$$\min_G E_{z\sim p_z(z)}[-y\log D(G(z)) - (1-y)\log(1 - D(G(z))]$$

$$\Big\downarrow \quad y = 1$$

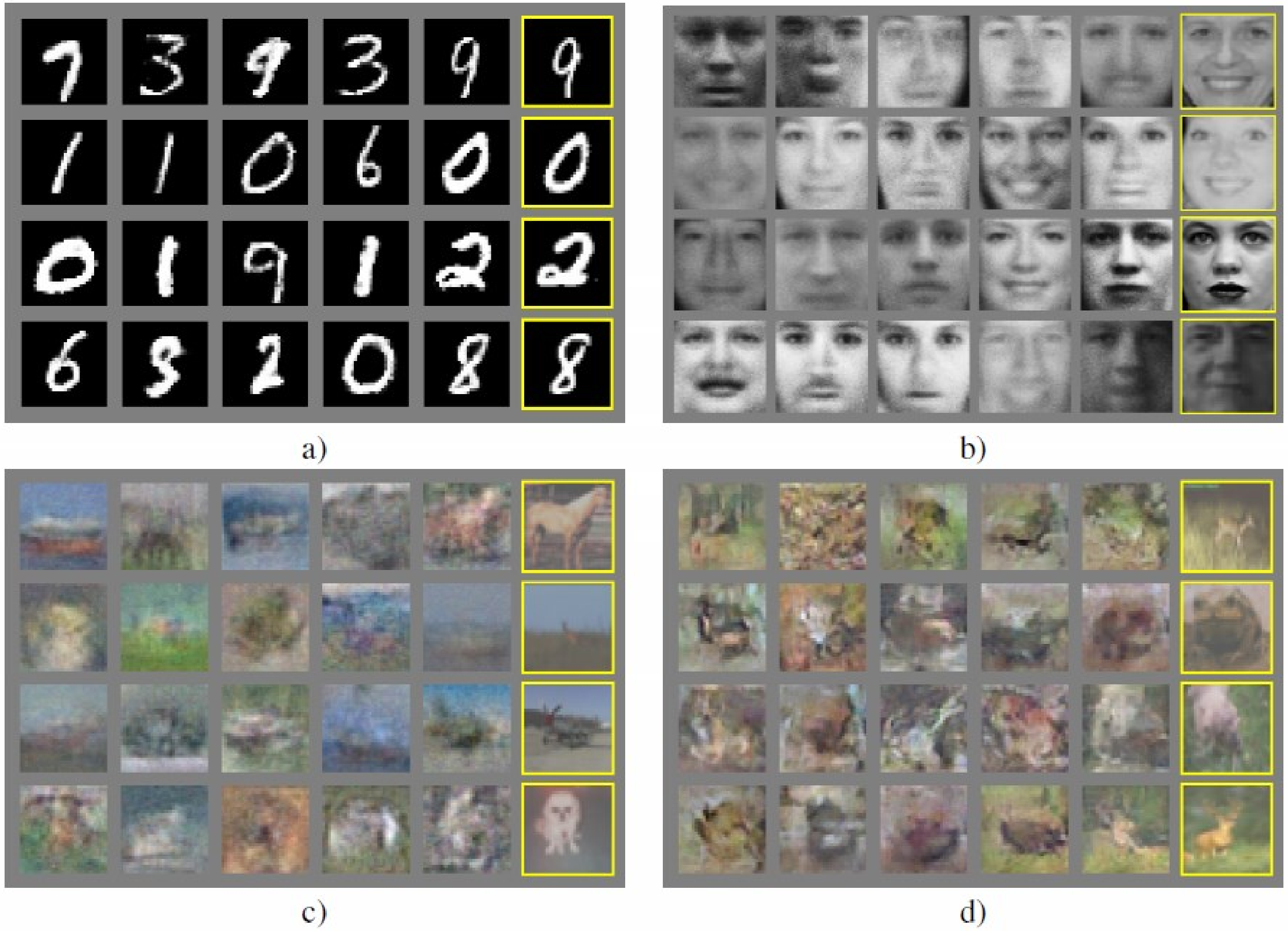
$$\min_G E_{z\sim p_z(z)}[-\log D\big(G(z)\big)]$$

The gradient is very large at D(G(z))=0



$$y = \log(x)$$

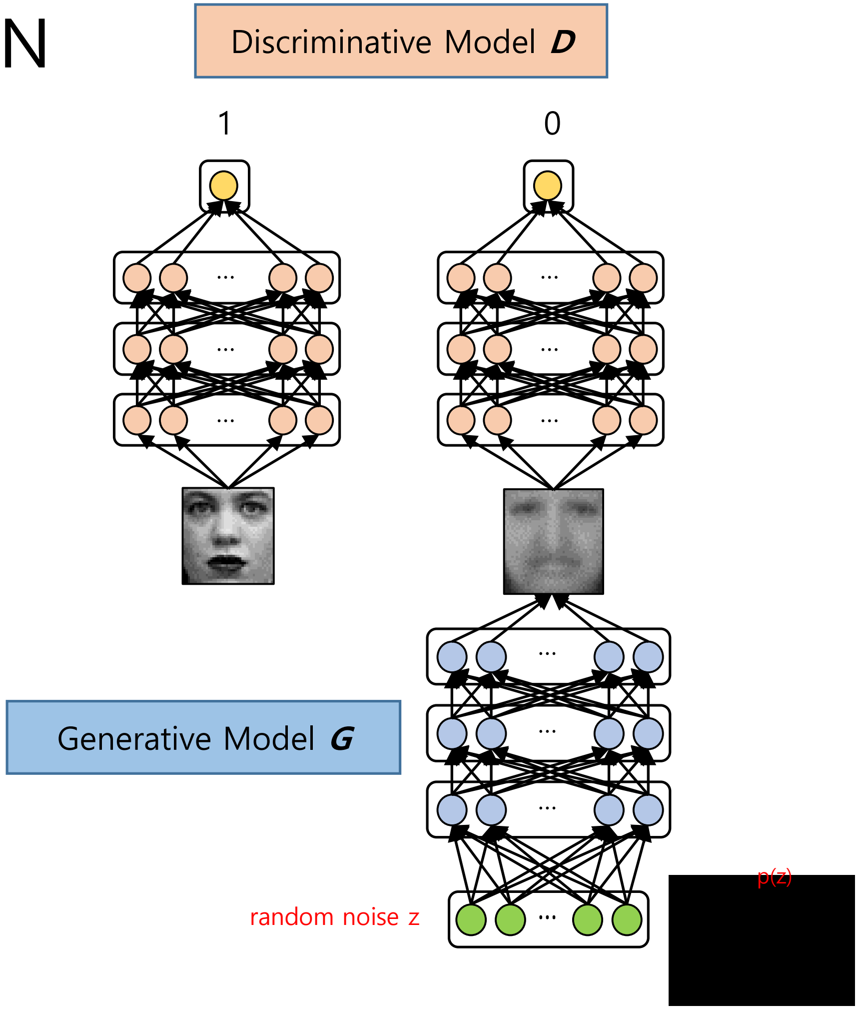
GAN Results



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

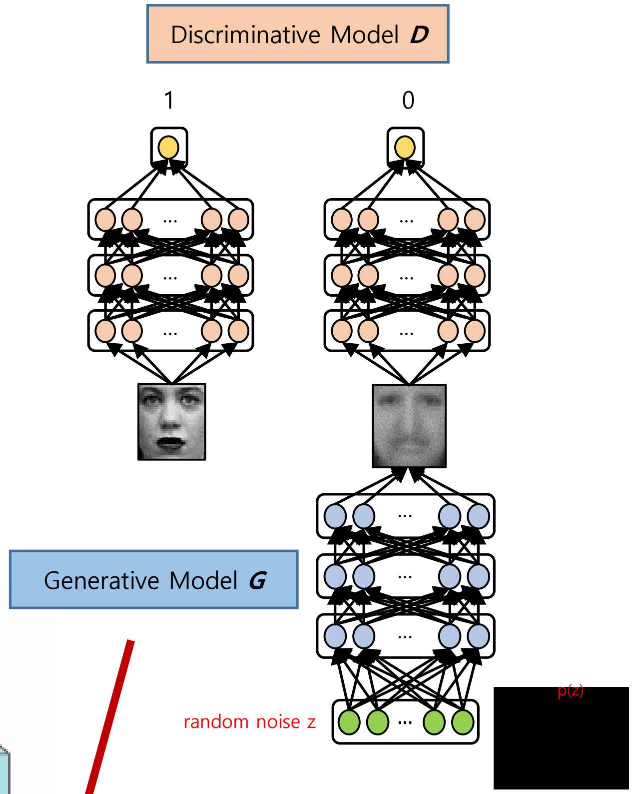
DCGAN

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

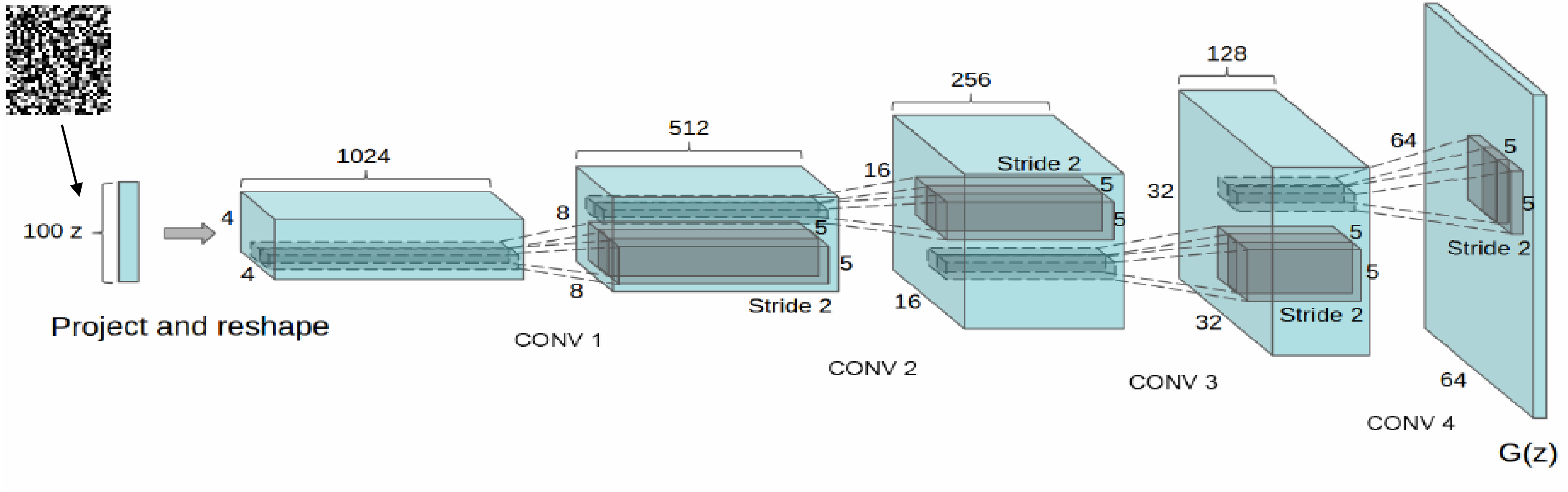


DCGAN

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



z:
unifor
m dist.



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

- LSUN – bedroom dataset
  - 3 million training examples

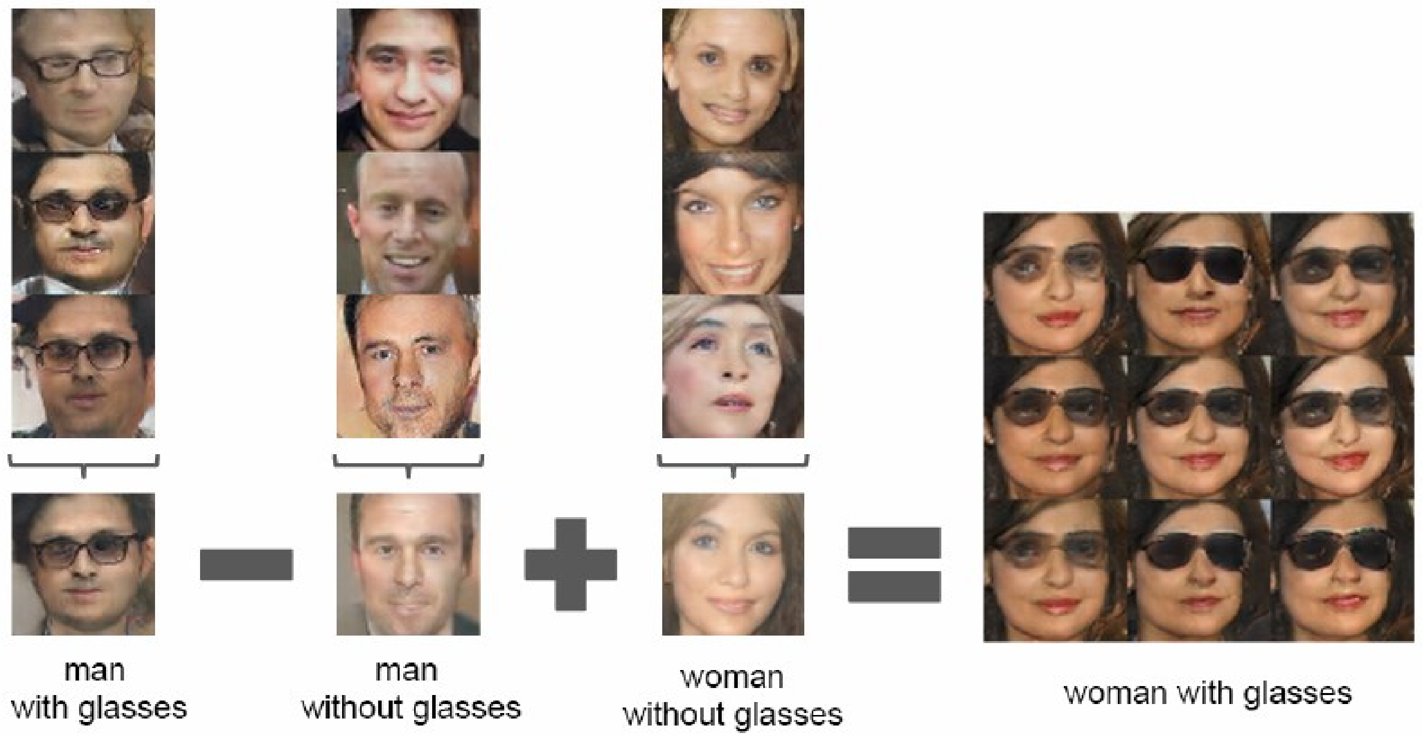
Epoch #1

Epoch #5

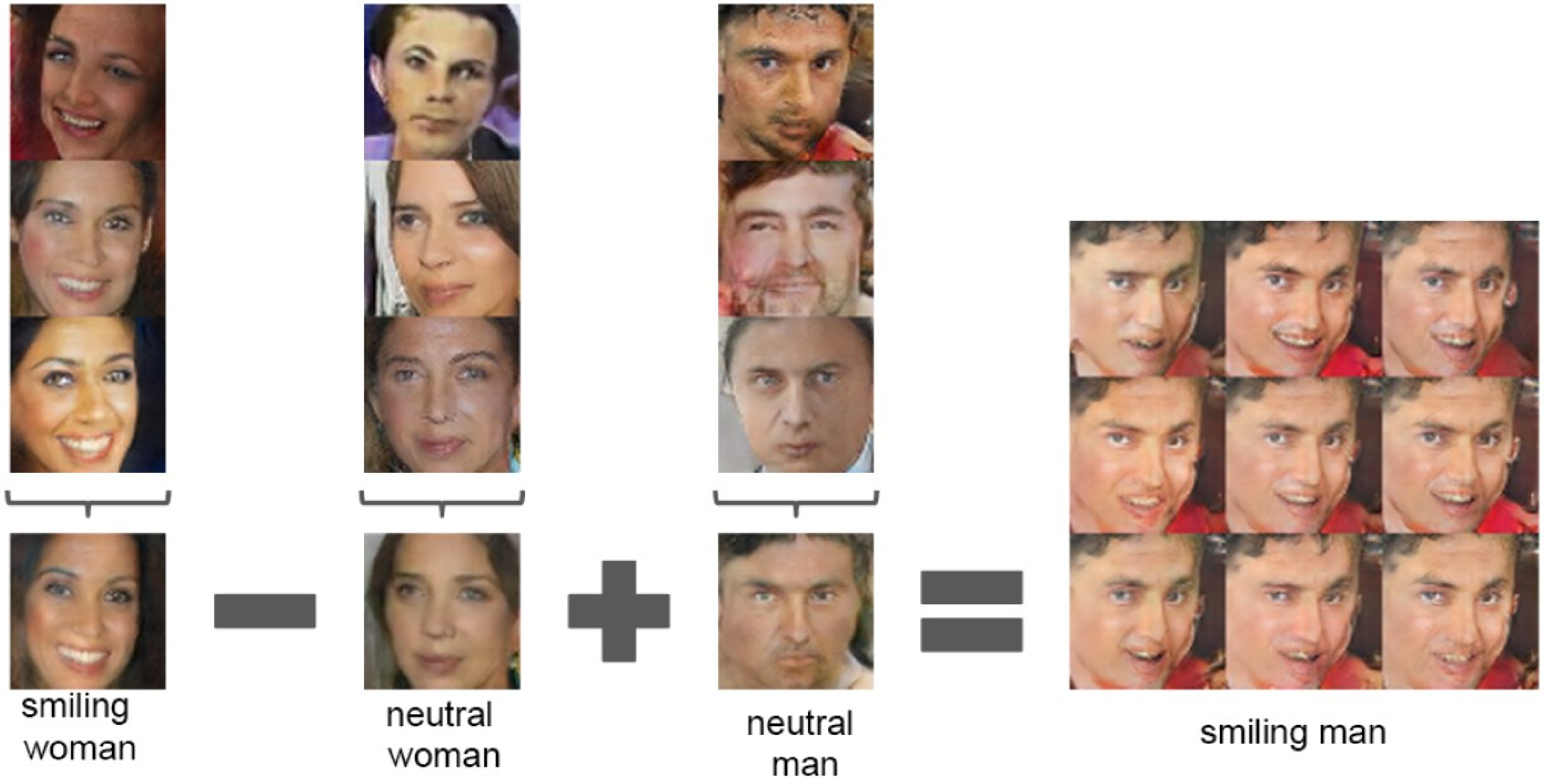
Experiments

Experiments

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

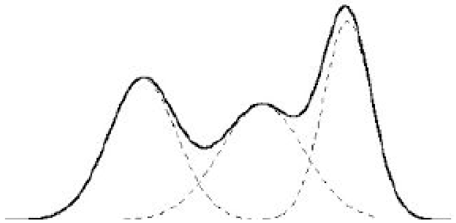


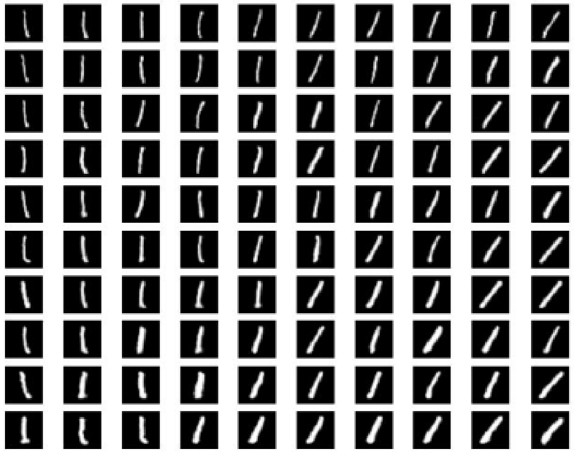
Experiments

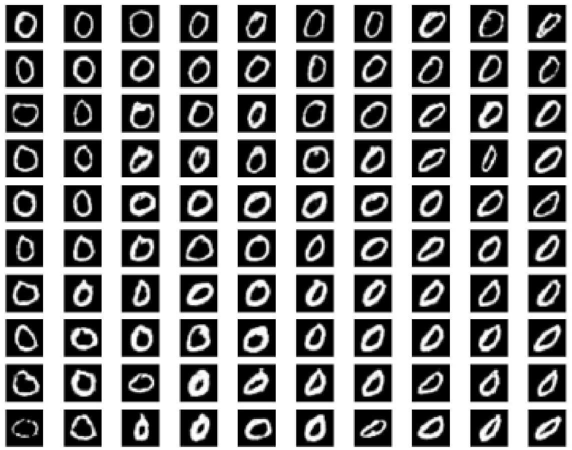


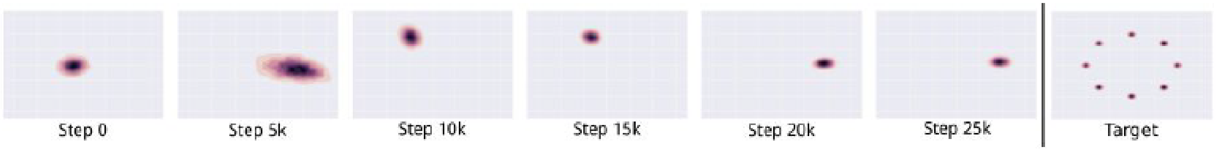
Experiments

Pitfall of GAN









mode collapsing

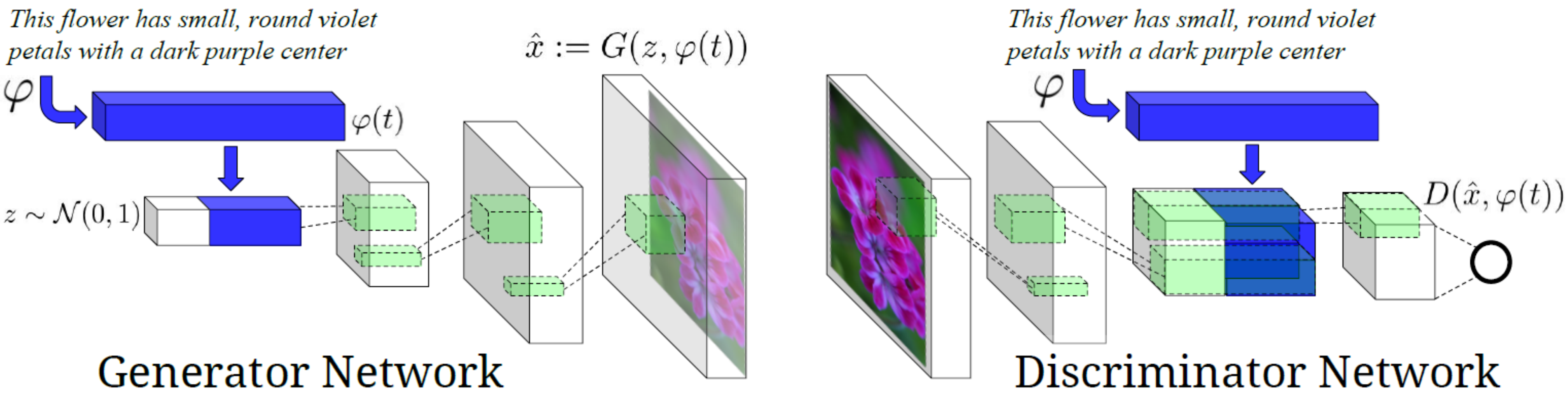
oscillating

Generative Adversarial Text to Image Synthesis
-ICML 2016.

GAN – text to image synthesis

- $\psi(t)$: text embedding function (map to 1024 dim)
  - → Fully-connected layer → 128 dim
  - Used pre-trained text encoder (can be done end-to-end manner)
- $z \sim N(0,1)$: 100 dim noise vector

Text-Conditional GAN



Conditional GAN

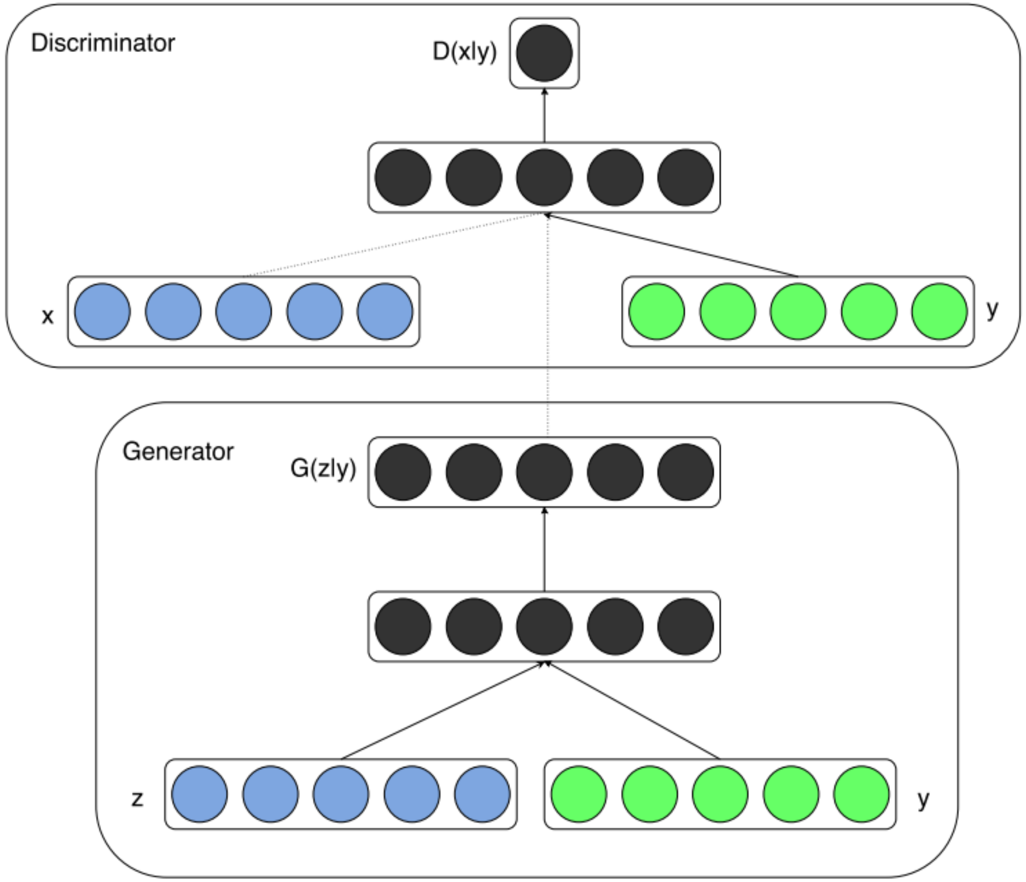


Figure 1: Conditional adversarial net

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

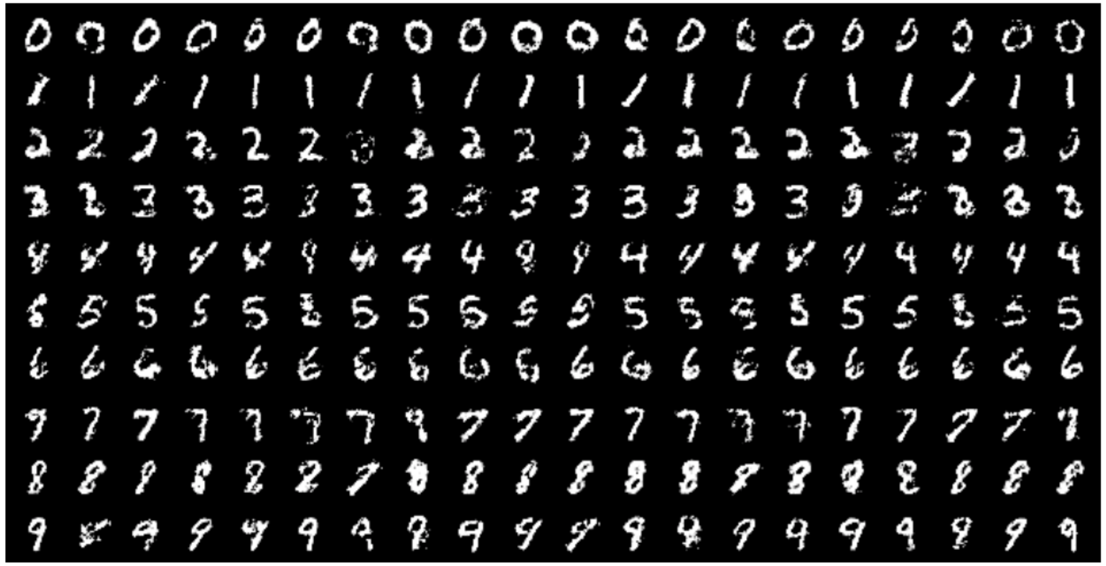
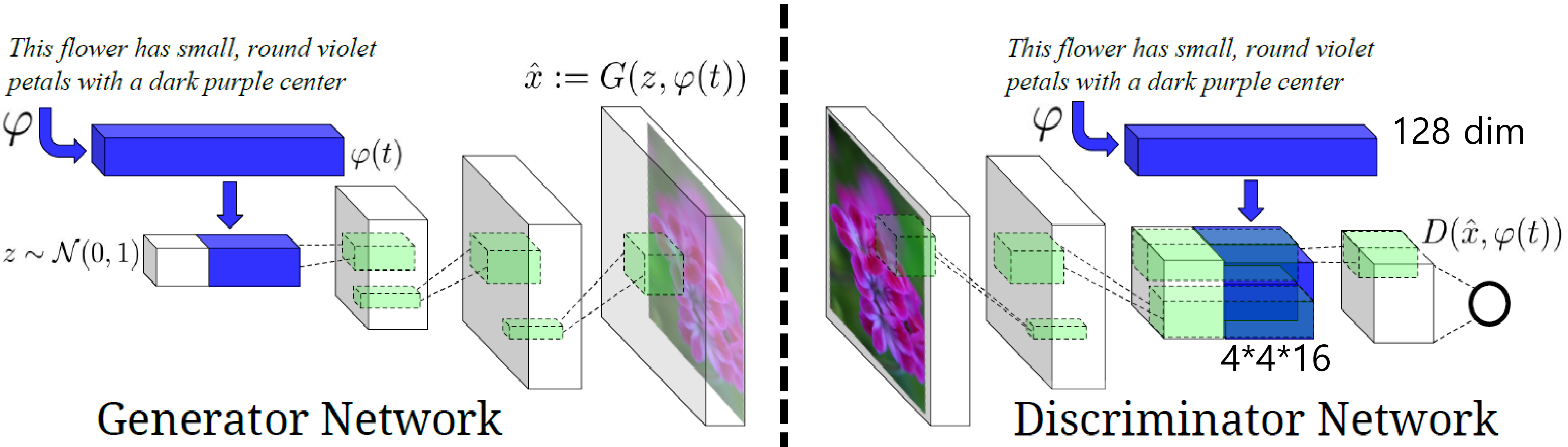


Figure 2: Generated MNIST digits, each row conditioned on one label

Text-conditional GAN (naïve)



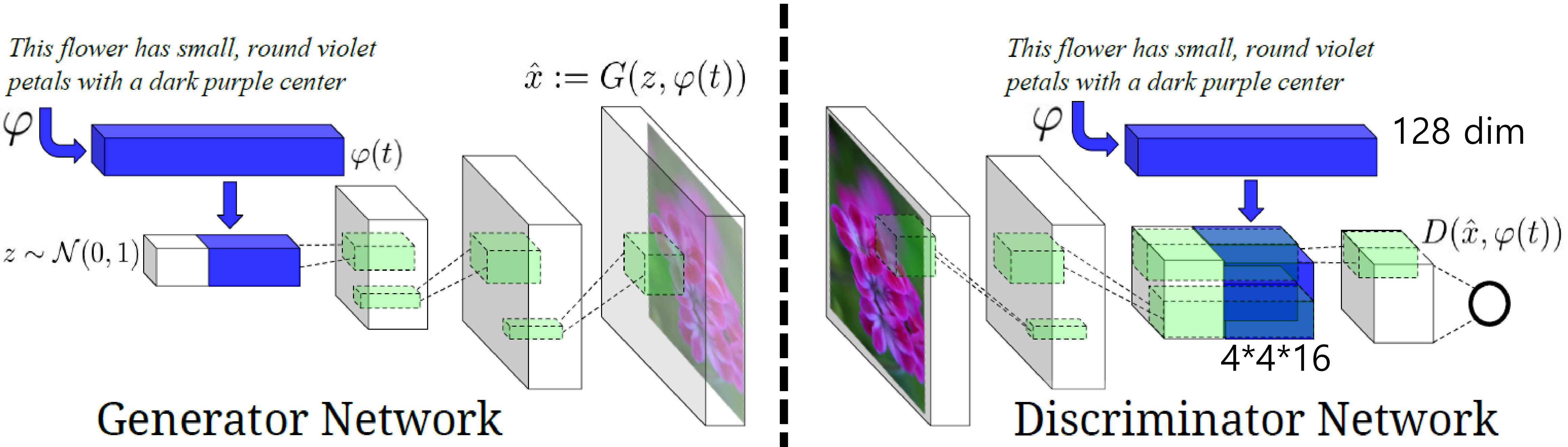
Real image & matched text

Fake image & arbitrary text

$$h = \psi(t)$$

$$\min_G \max_D E_{x\sim p_{data}}[\log(D(x,h))] + E_{z\sim p_z}\left[\log\left(1 - D\left(G\left(z,\hat{h}\right),\hat{h}\right)\right)\right]$$

Matching-aware Discriminator



Real image & matched text

Real image & mismatched text

$$h = \psi(t)$$

$$\min_G \max_D \; E_{x\sim p_{data}}[\log(D(x,h))] + E_{x\sim p_{data}}\left[\log\left(1 - D(x,\hat{h})\right)\right] + E_{z\sim p_z}[\log(1 - D(G(z,h),h))]$$

Fake image & matched text

Matching-aware Discriminator

$$h = \psi(t)$$

Real image & matched text

Real image & mismatched text

$$\min_G \max_D E_{x\sim p_{data}}\left[\log\left(D(x,h)\right)\right] + E_{x\sim p_{data}}\left[\log\left(1 - D(x,\hat{h})\right)\right] \\ + E_{z\sim p_z}\left[\log\left(1 - D(G(z,h),h)\right)\right]$$

Fake image & matched text

Algorithm 1 GAN-CLS training algorithm with step size α, using minibatch SGD for simplicity.

Input: minibatch images x, matching text t, mis-matching $\hat{t}$, number of training batch steps S
for $n = 1$ **to** S **do**
  $h \leftarrow \varphi(t)$ {Encode matching text description}
  $\hat{h} \leftarrow \varphi(\hat{t})$ {Encode mis-matching text description}
  $z \sim \mathcal{N}(0,1)^Z$ {Draw sample of random noise}
  $\hat{x} \leftarrow G(z,h)$ {Forward through generator}
  $s_r \leftarrow D(x,h)$ {real image, right text}
  $s_w \leftarrow D(x,\hat{h})$ {real image, wrong text}
  $s_f \leftarrow D(\hat{x},h)$ {fake image, right text}
  $\mathcal{L}_D \leftarrow \log(s_r) + (\log(1-s_w) + \log(1-s_f))/2$
  $D \leftarrow D - \alpha\partial\mathcal{L}_D/\partial D$ {Update discriminator}
  $\mathcal{L}_G \leftarrow \log(s_f)$
  $G \leftarrow G - \alpha\partial\mathcal{L}_G/\partial G$ {Update generator}
end for

Learning with Manifold Interpolation

Real image & matched text

Real image & mismatched text

$$h = \psi(t)$$

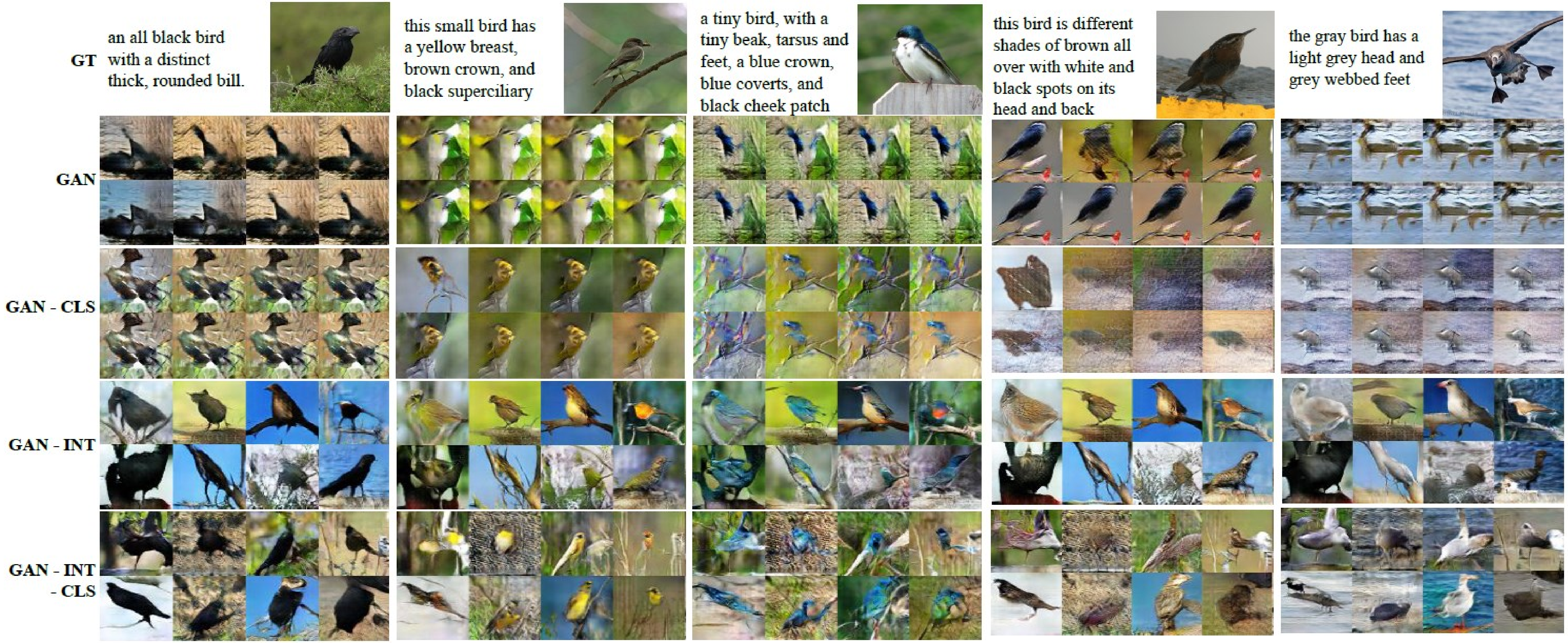
$$\min_G \max_D \; E_{x\sim p_{data}}[\log(D(x,h))] + E_{x\sim p_{data}}\left[\log\left(1 - D(x,\hat{h})\right)\right]$$
$$+ E_{z\sim p_z}[\log(1 - D(G(z,h),h))]$$

Fake image & matched text

Additional term to generator to minimize:

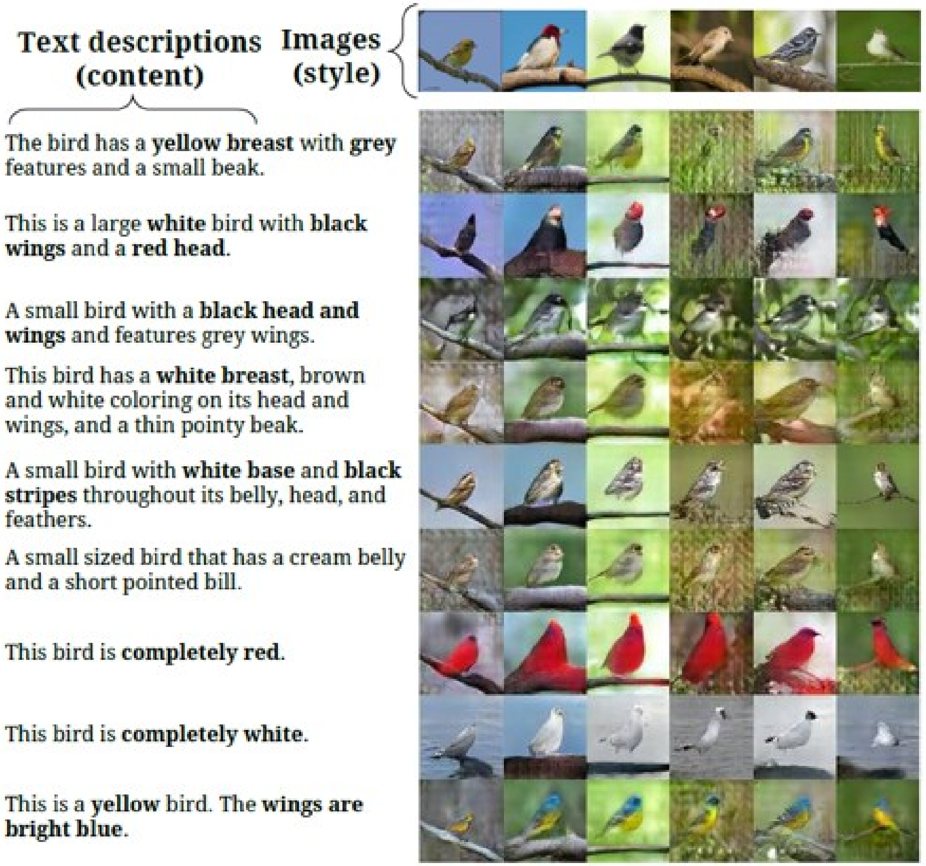
$$E_{t_1,t_2\sim p_{textdata}}\left[\log\left(1 - D\left(G\left(z,\bar{h}\right)\right),\bar{h}\right)\right],$$
$$\bar{h} = \beta h_1 + (1-\beta)h_2$$
$$h_1 = \psi(t_1), h_2 = \psi(t_2)$$

Experiments



Experiments

Style Transfer



$$L_{style} = E_{t,z\sim N(0,1)} \left\| z - S(G(z,h)) \right\|_2^2$$
$$S(x) \rightarrow z$$

Input image: x
Style: $z = S(x)$
Generated image: $G(z, h)$

Image ⇨ **Style vector** ⇨ Style transferred image

Text description

Sentence Interpolation

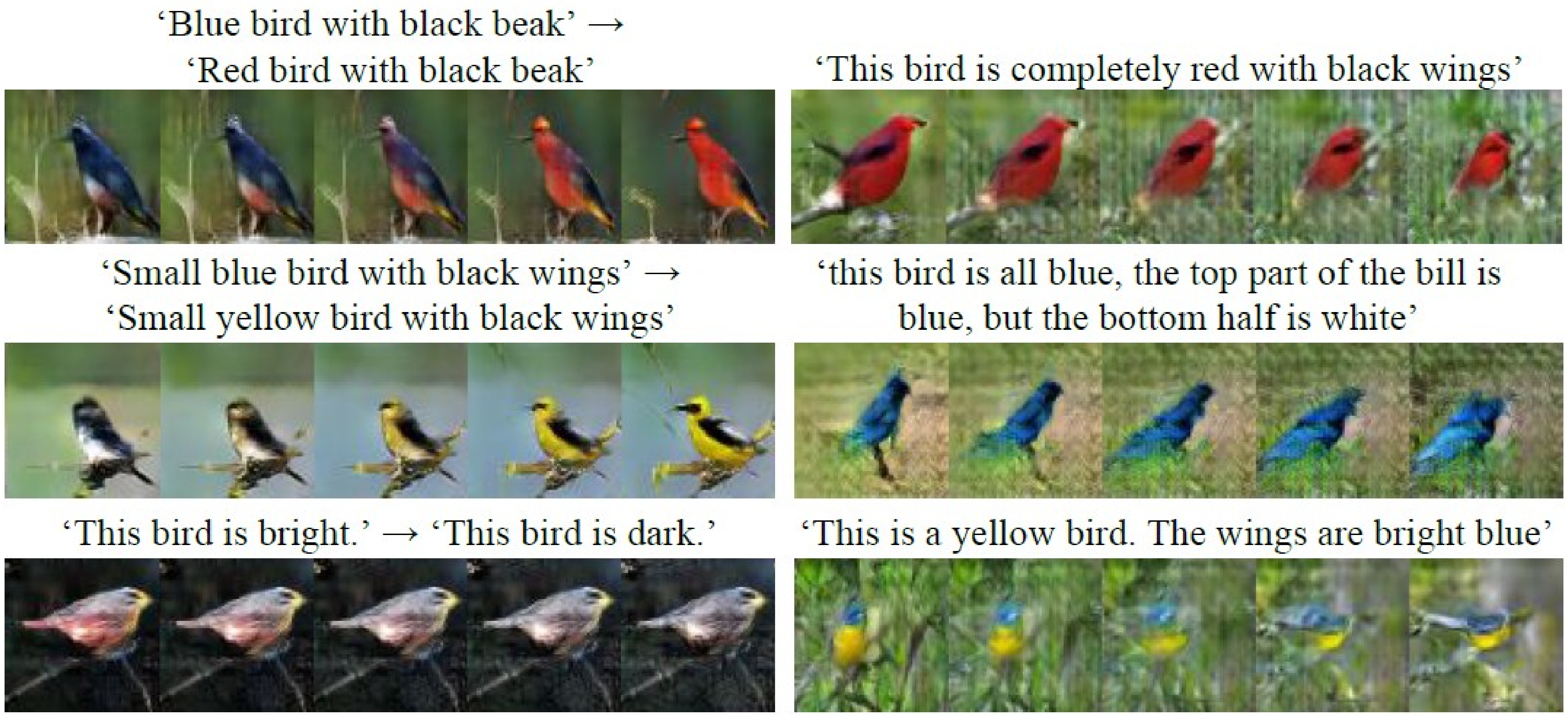


Figure 8. Left: Generated bird images by interpolating between two sentences (within a row the noise is fixed). Right: Interpolating between two randomly-sampled noise vectors.

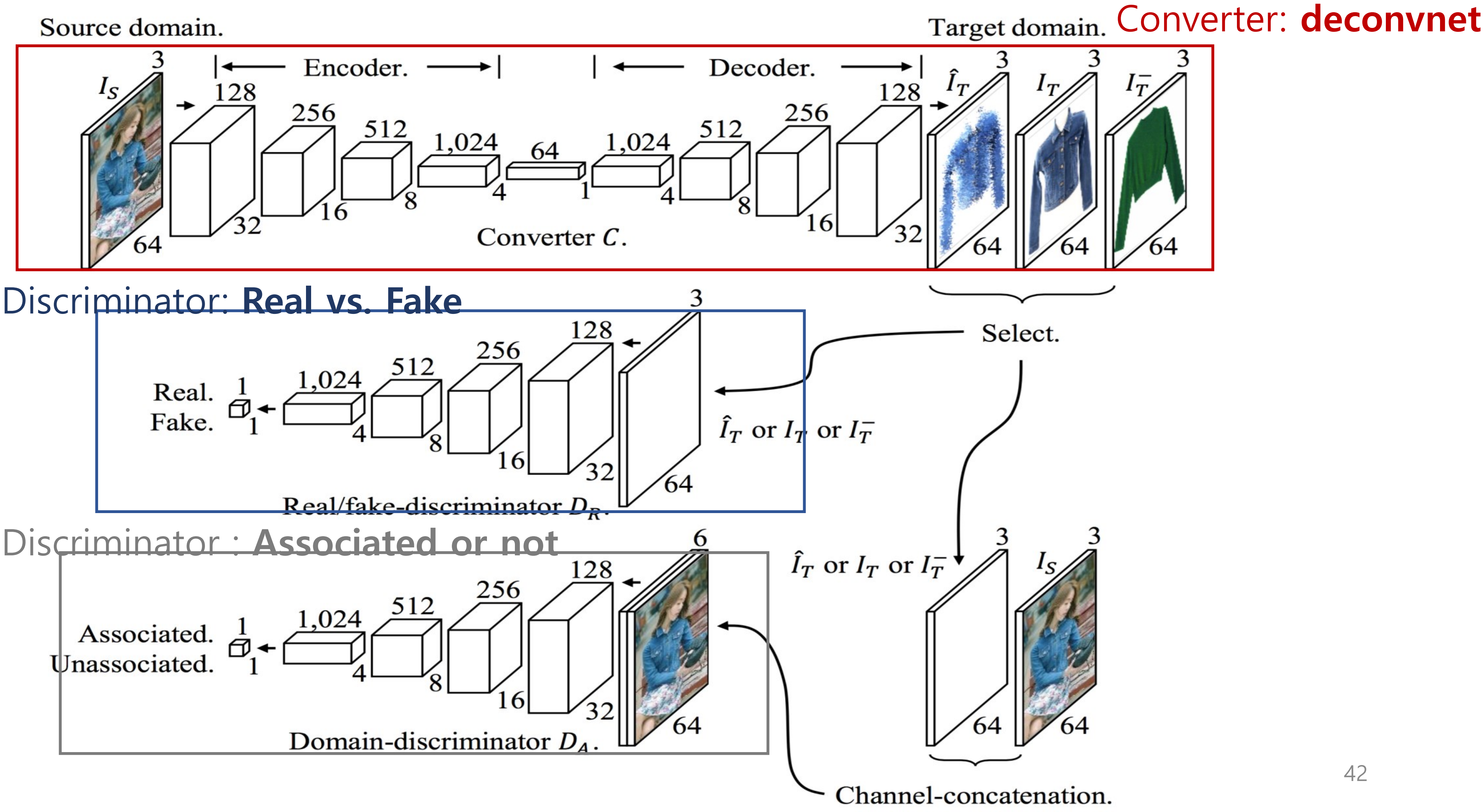
Pixel Level Domain Transfer
-ECCV 2016

Domain transfer

A source image. Possible target images.

Fig. 1. A real example showing non-deterministic property of target image in the pixel-level domain transfer problem.

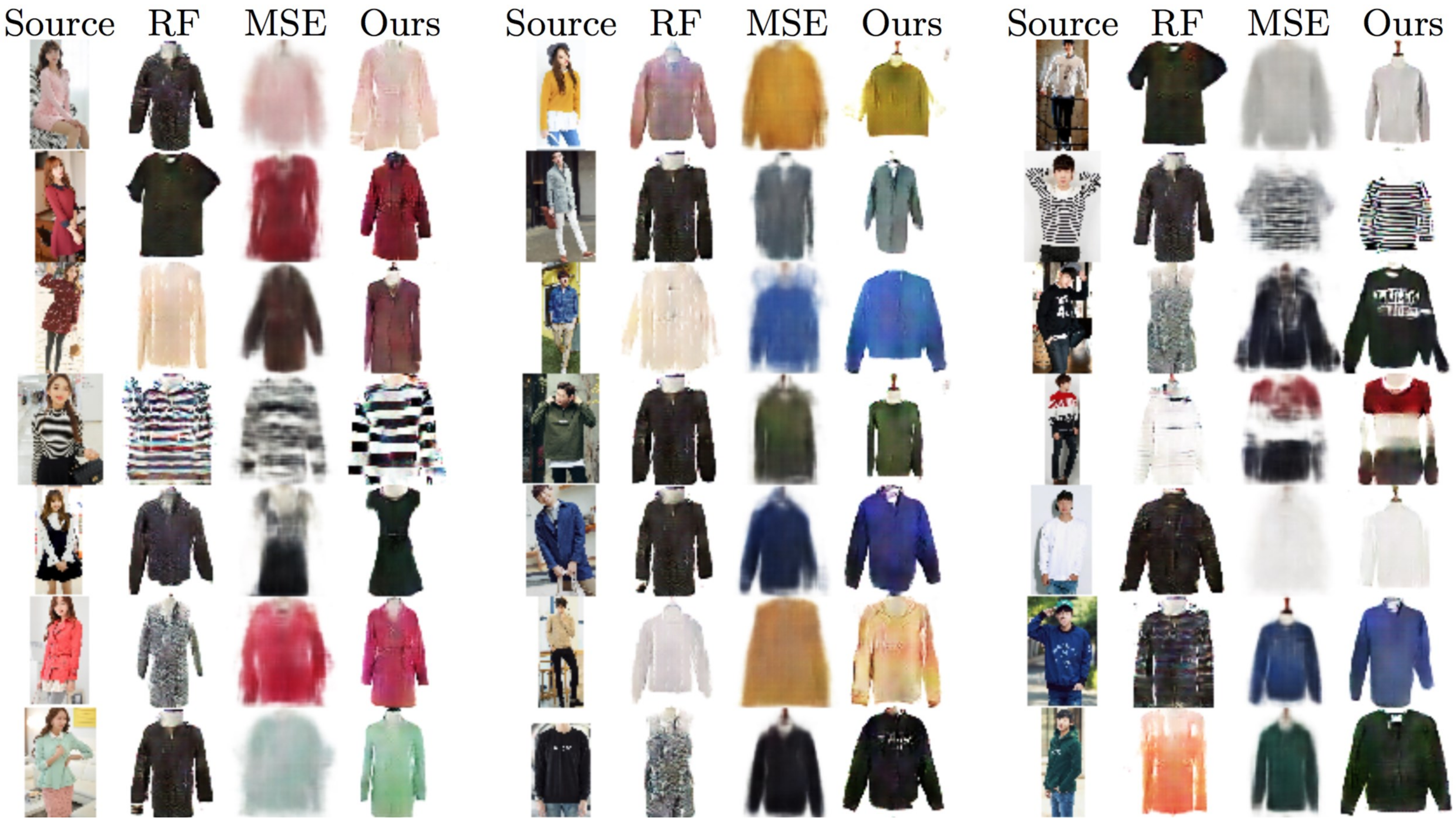
Whole architecture



Dataset

Fig. 3. Example images of LookBook. A product image is associated with multiple fashion model images.

Results

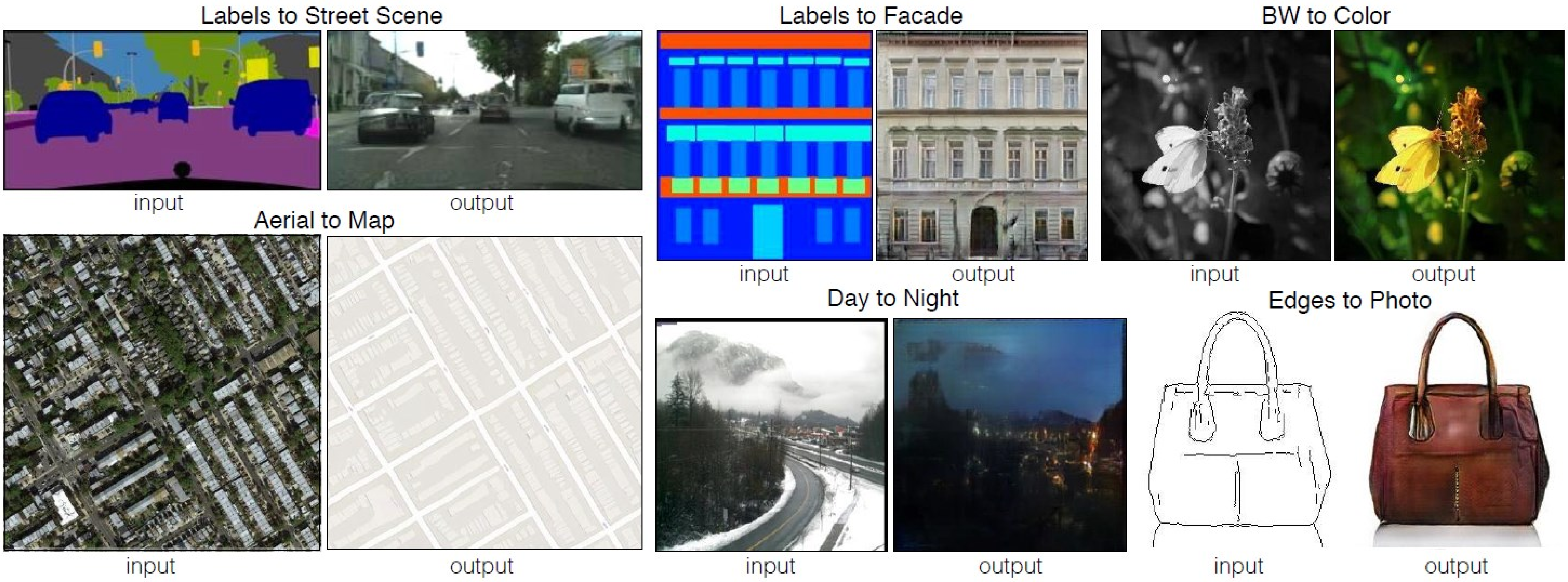


Results – varying input conditions

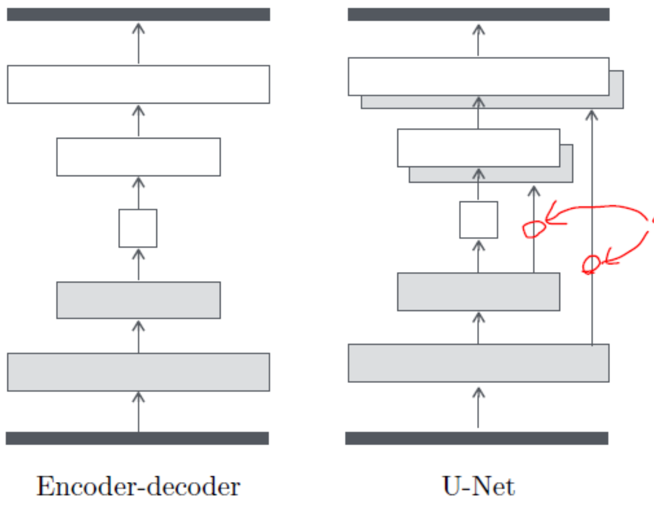
Results – inverse setting

Image-to-Image Translation with Conditional Adversarial Networks

Pix2Pix


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

Pix2Pix



얘가 U-net 이다!

Skip-connection 이다.

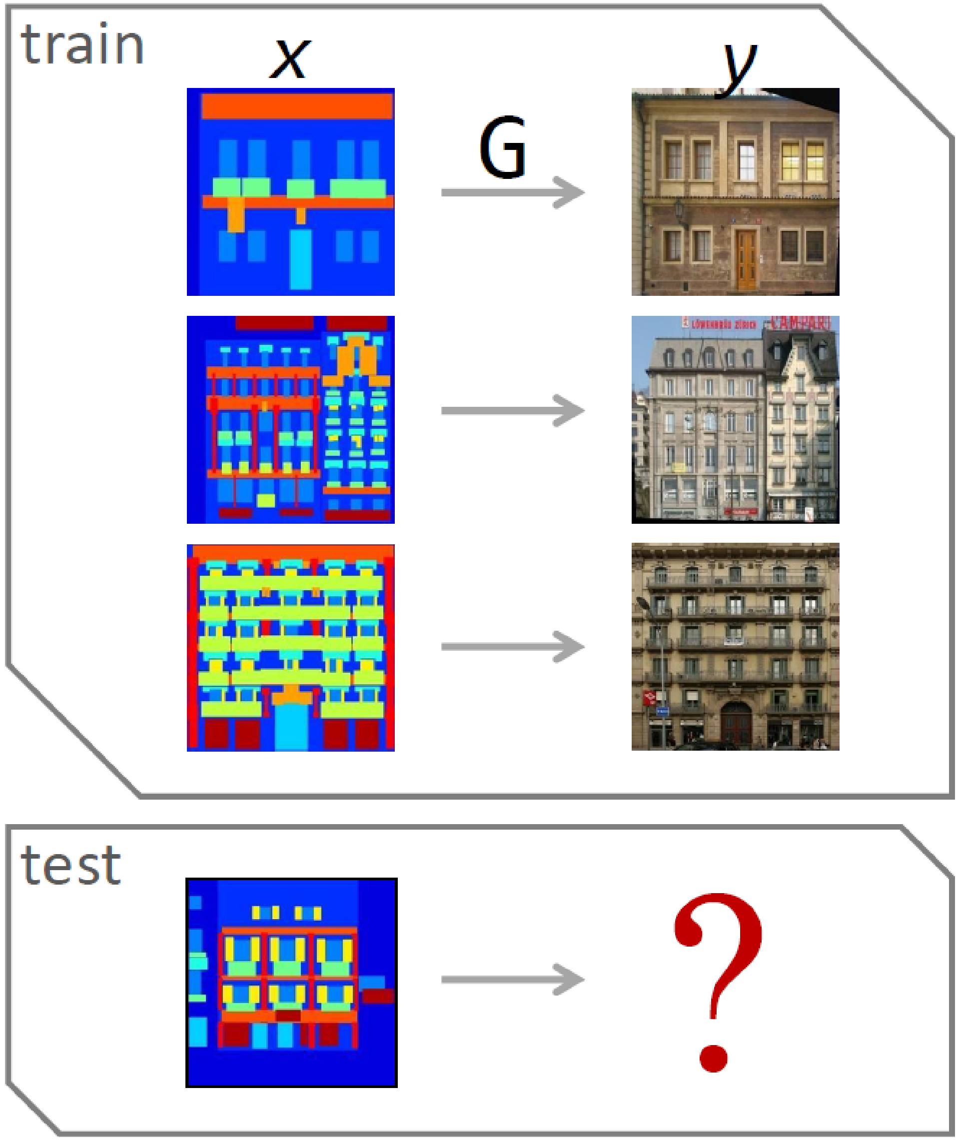
Figure 3: Two choices for the architecture of the generator. The "U-Net" [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

+ L1 loss function **Low-freq correctness**

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

+ PatchGAN **High-freq correctness**

Pix2Pix



- Supervised

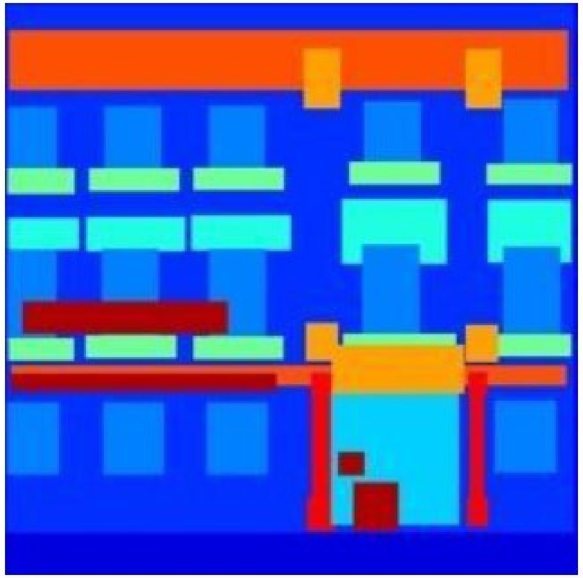
- loss: Minimize the difference between output *G(x)* and ground truth *y*

Data from [Tylecek, 2013]

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

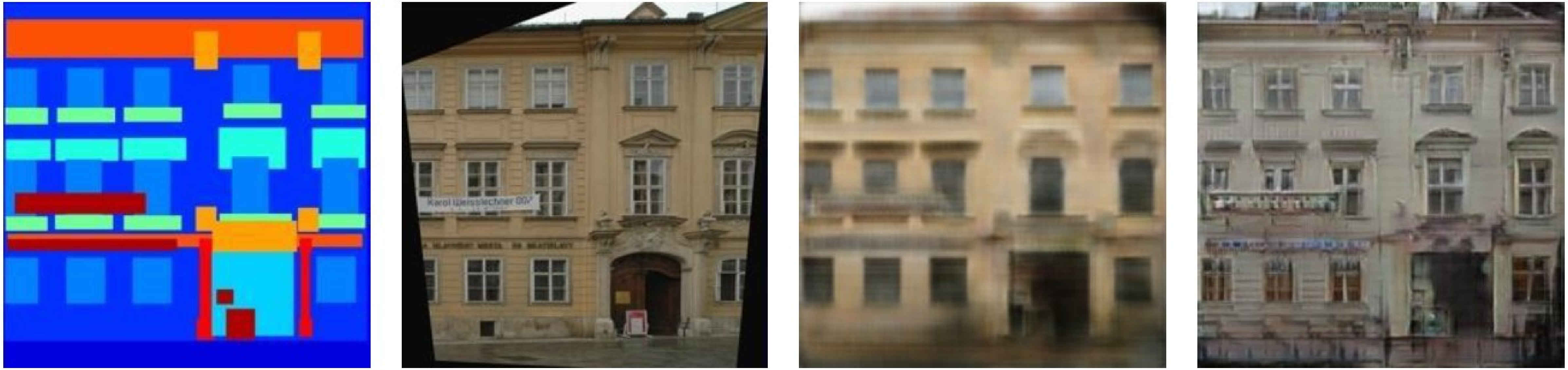
Output

Ground Truth

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.

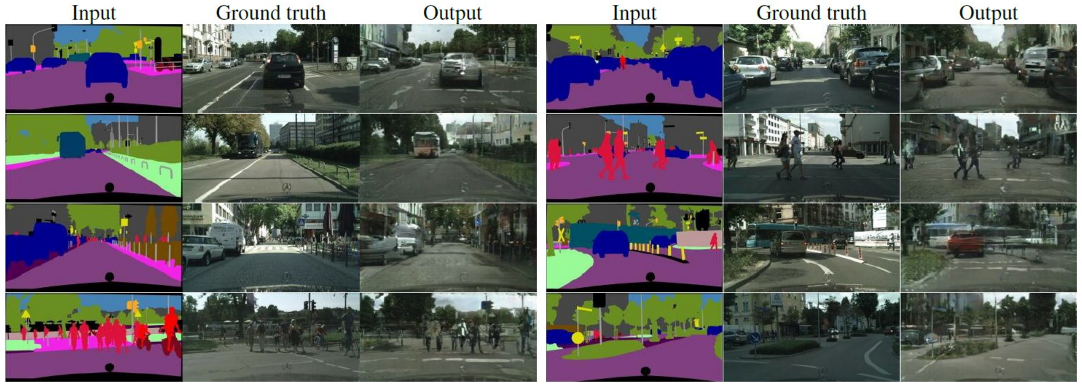


Figure 11: Example results of our method on Cityscapes labels→photo, compared to ground truth.

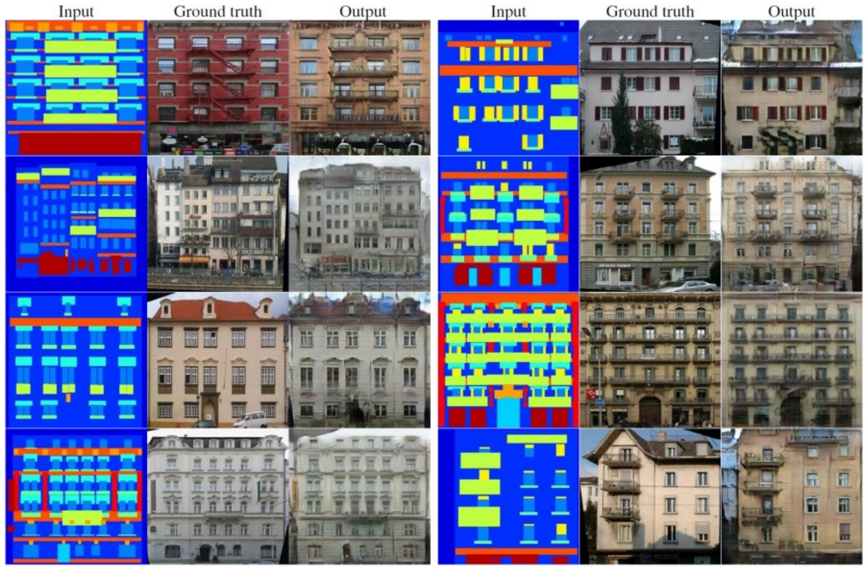


Figure 12: Example results of our method on facades labels→photo, compared to ground truth

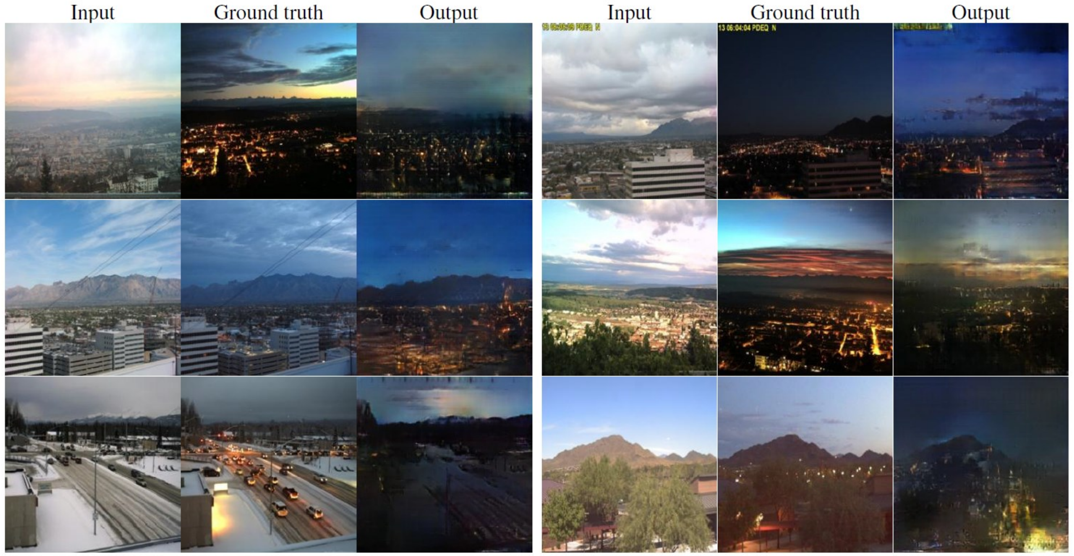


Figure 13: Example results of our method on day→night, compared to ground truth.

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

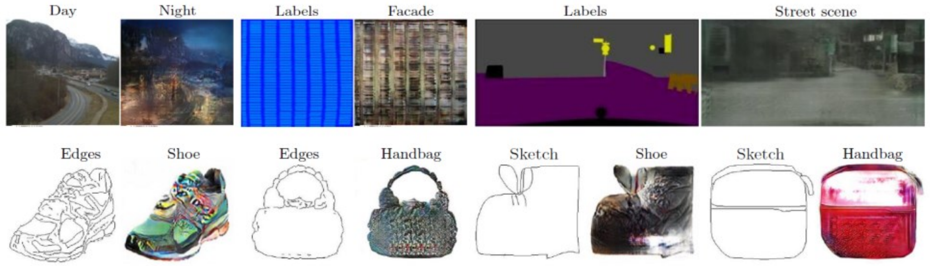


Figure 17: Example failure cases. Each pair of images shows input on the left and output on the right. These examples are selected as some of the worst results on our tasks. Common failures include artifacts in regions where the input image is sparse, and difficulty in handling unusual inputs. Please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

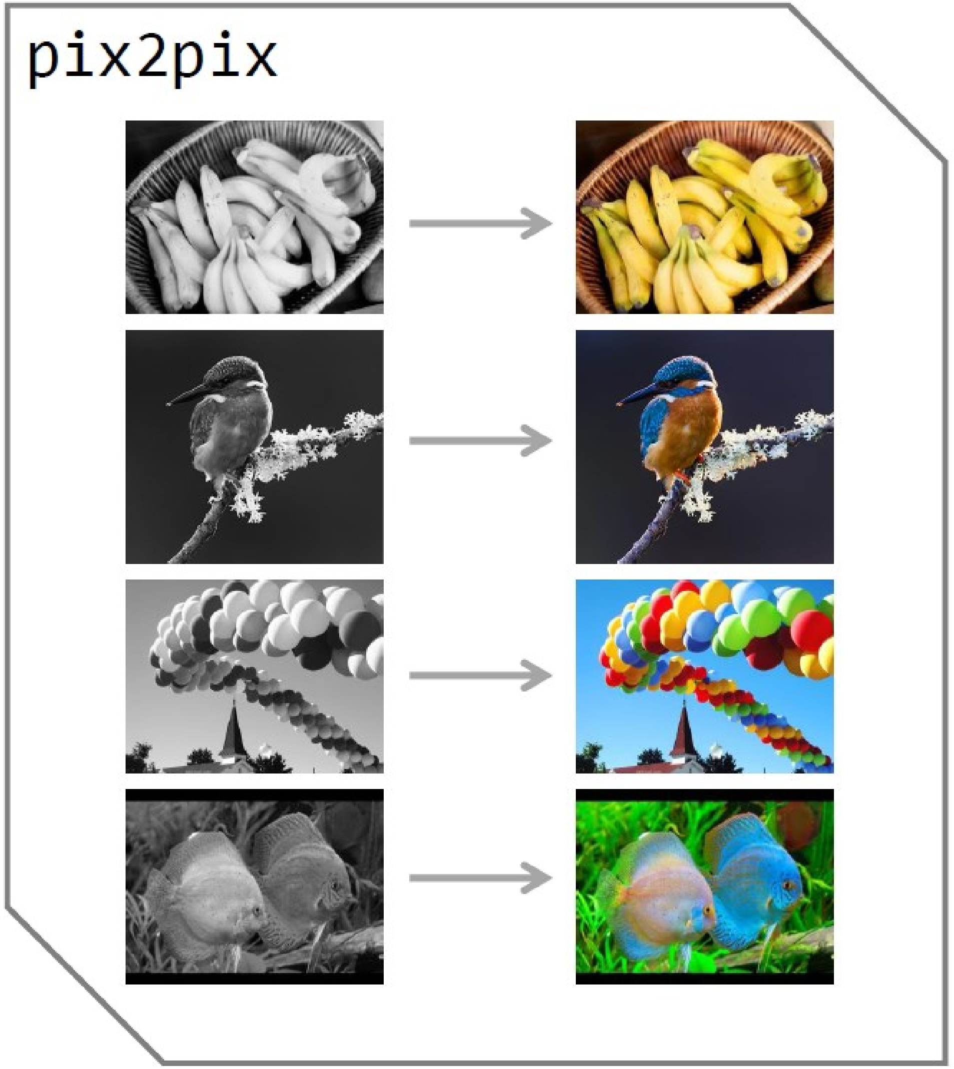
Let’s Try

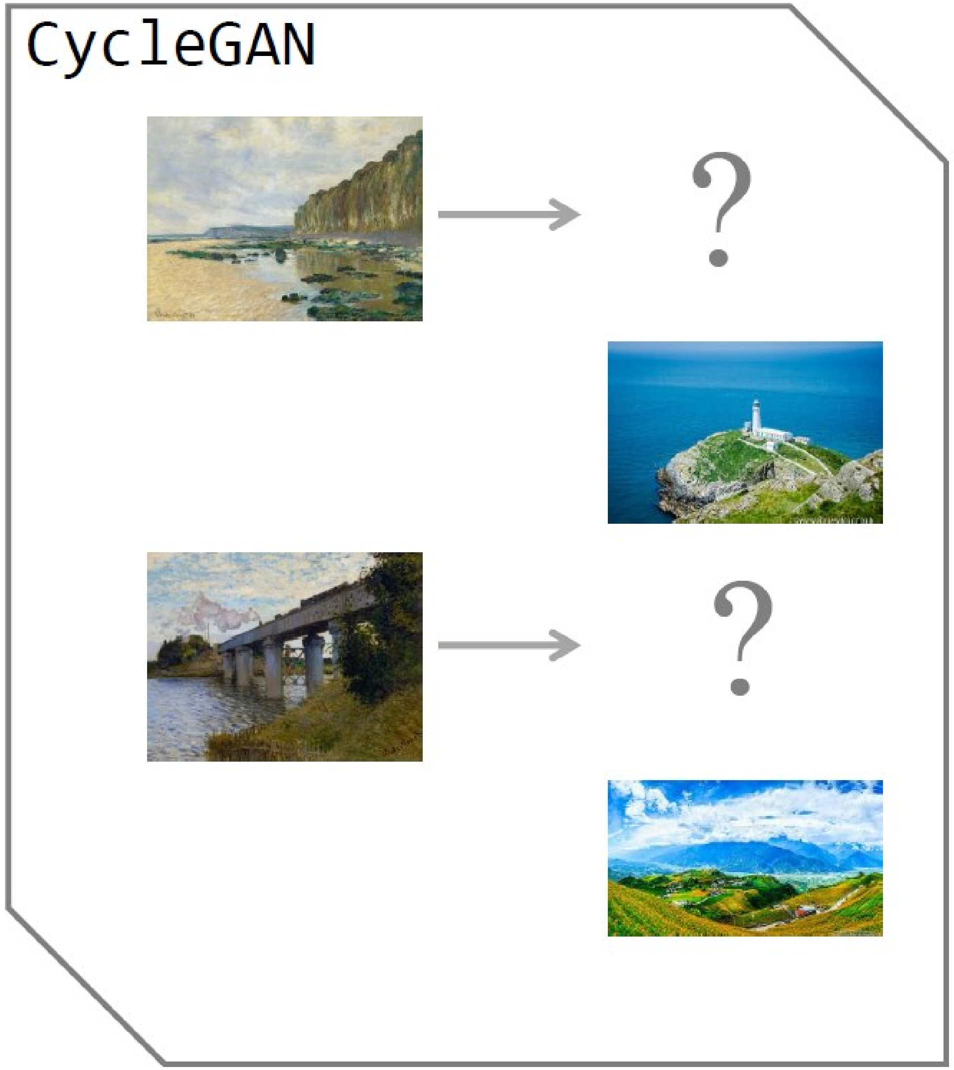
- https://affinelayer.com/pixsrv/

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

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

CycleGAN

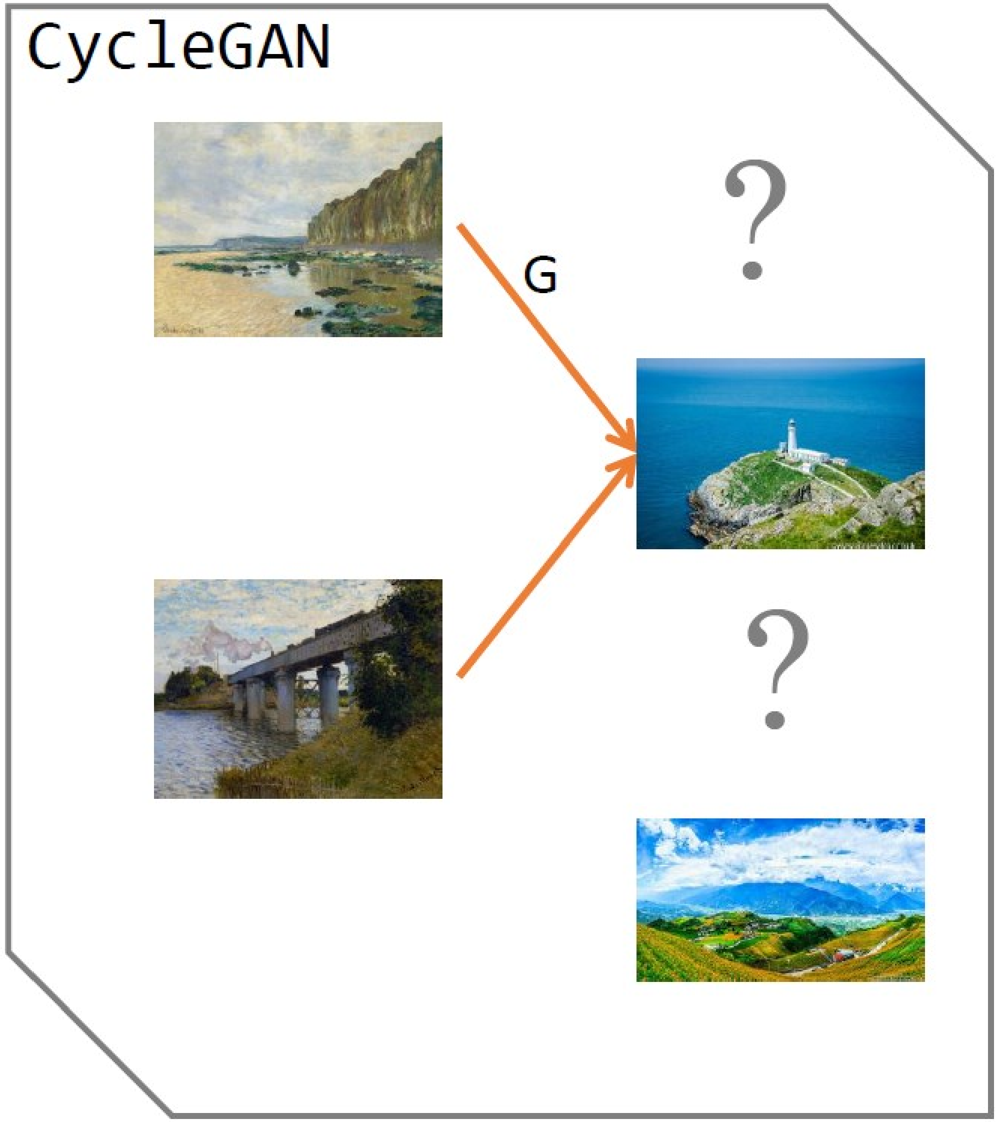




CycleGAN – Mode Collapsing

Loss: $L_{GAN}(G(x), y)$

G(x) should just look photorealistic



CycleGAN

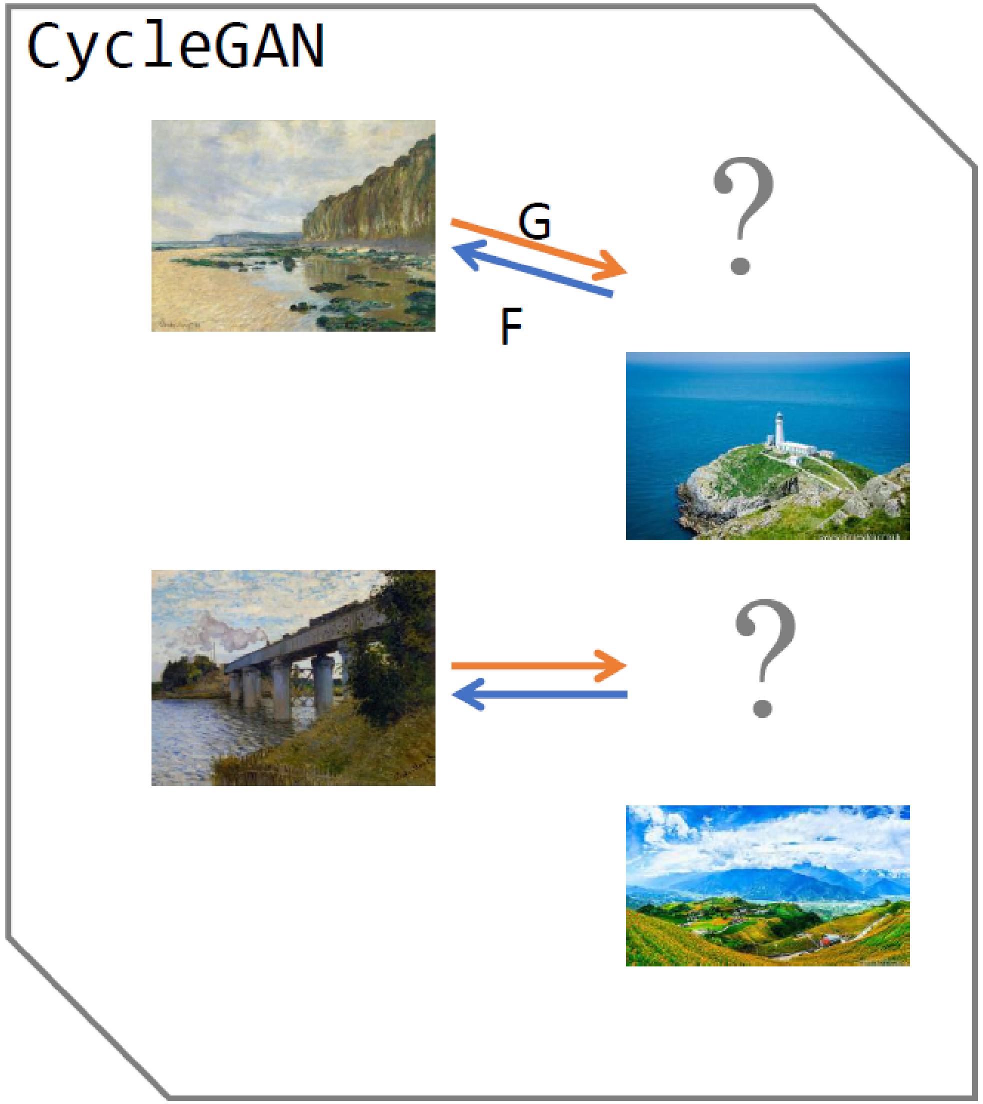
Loss

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

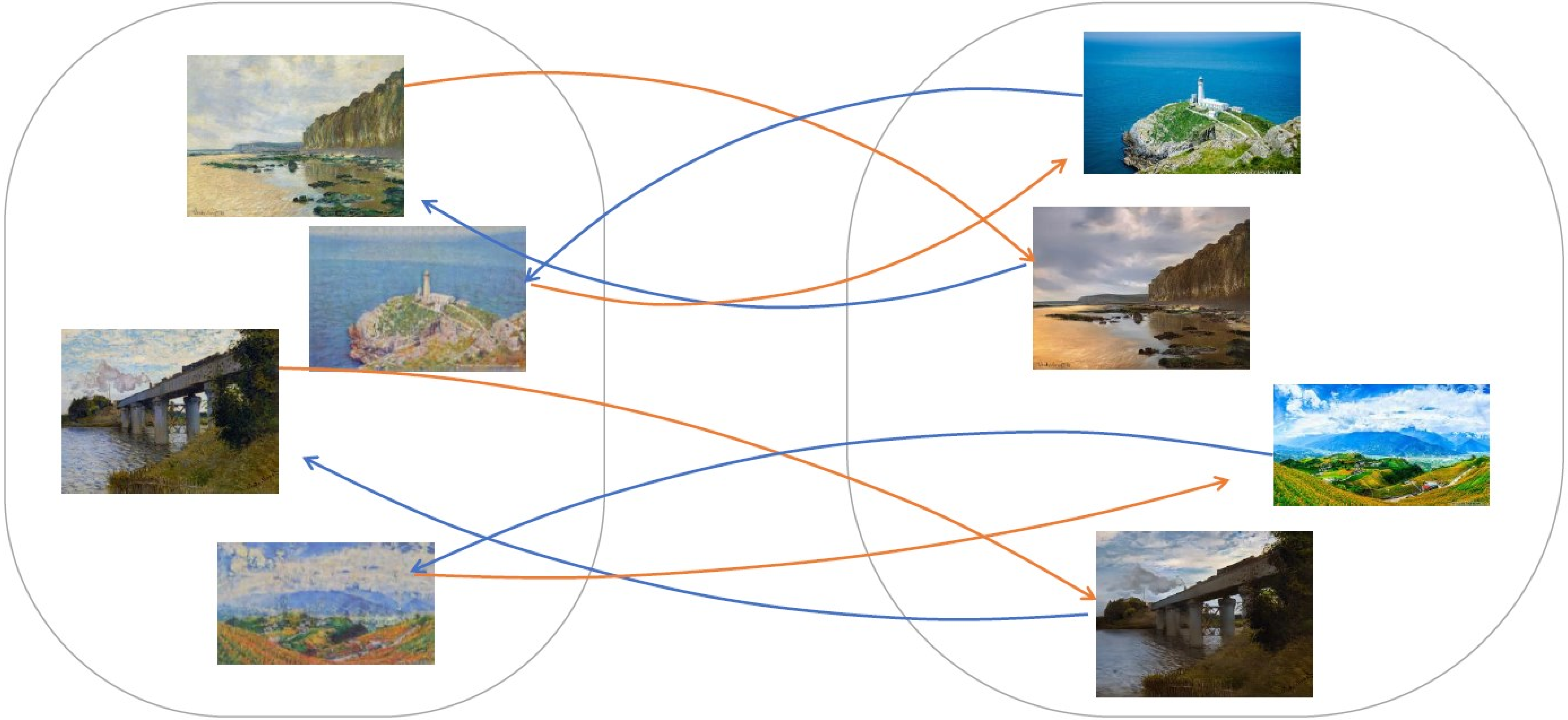
G(x) should just look photorealistic

and *F(G(x))* should be *F(G(x)) = x*,

where *F* is the inverse deep network

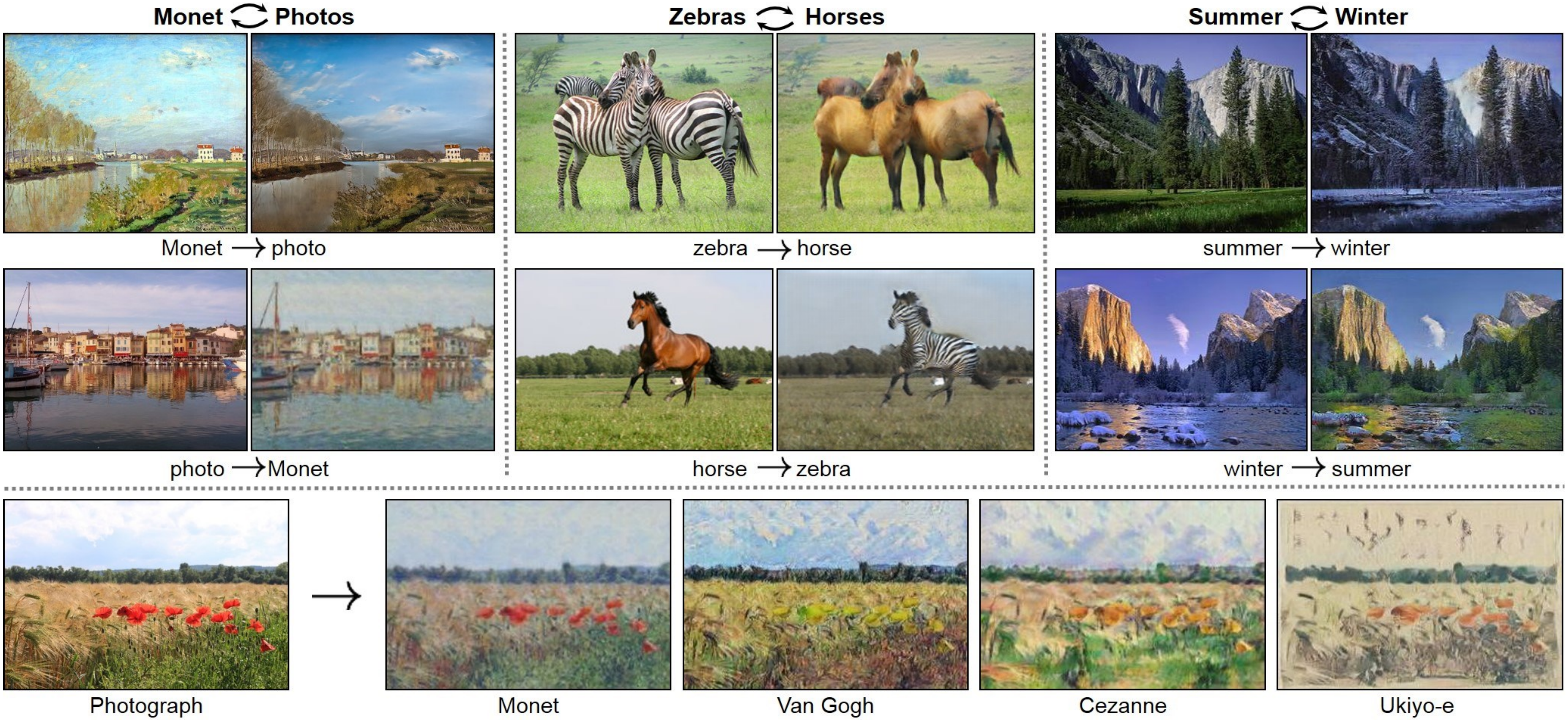


CycleGAN – Loss Function

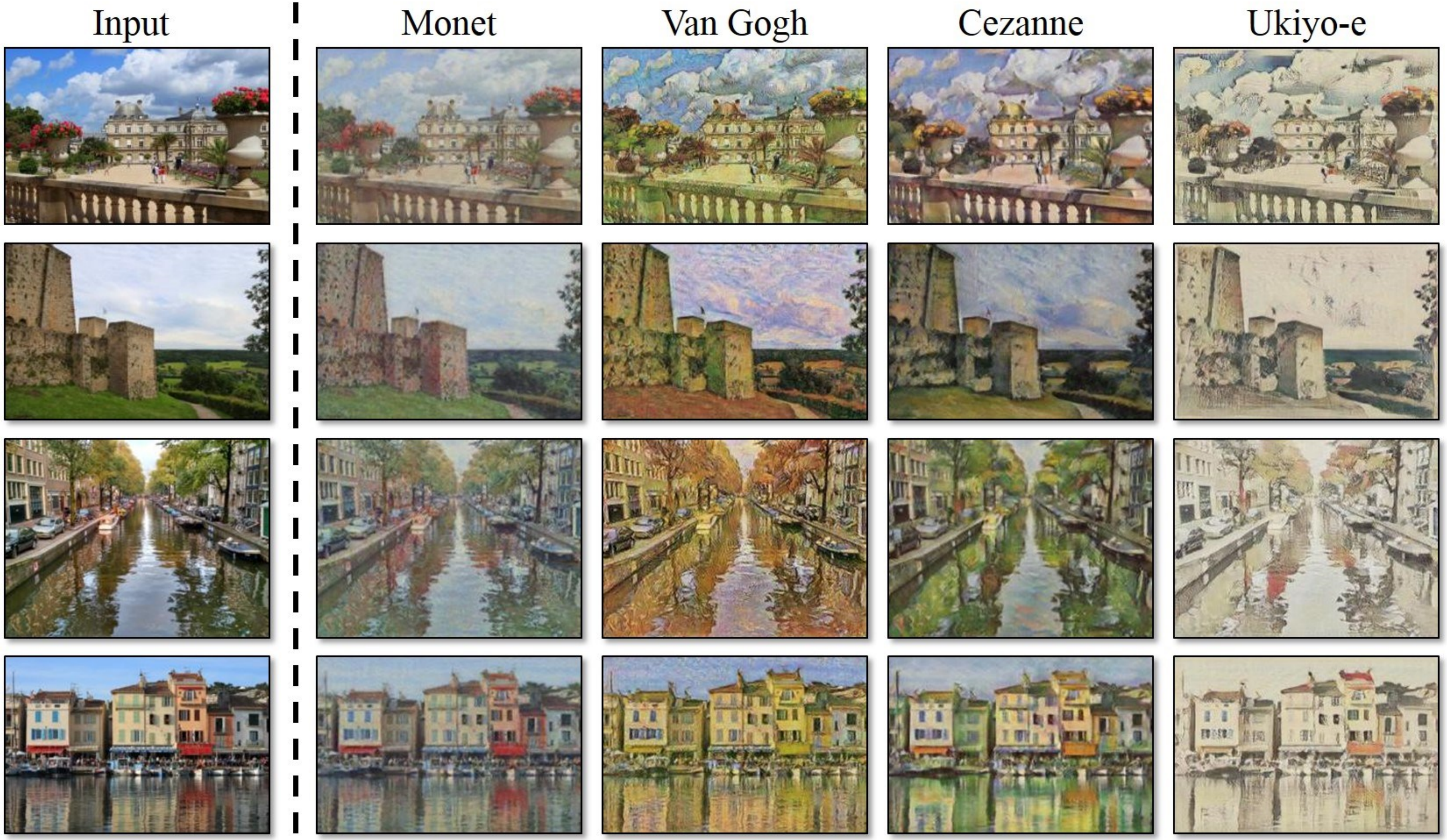


$$L_{GAN}(G(x), y) + \left\| F(G(x)) - x \right\|_1 \quad + \quad L_{GAN}(F(y), x) + \left\| G(F(y)) - y \right\|_1$$

Results


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

Results



Results

Input
Cycle alone
GAN alone
GAN+forward
GAN+backward
CycleGAN
Ground Truth

Reconstructed Images

Original CG | Fake Photo | Reconstruction

Real Photo