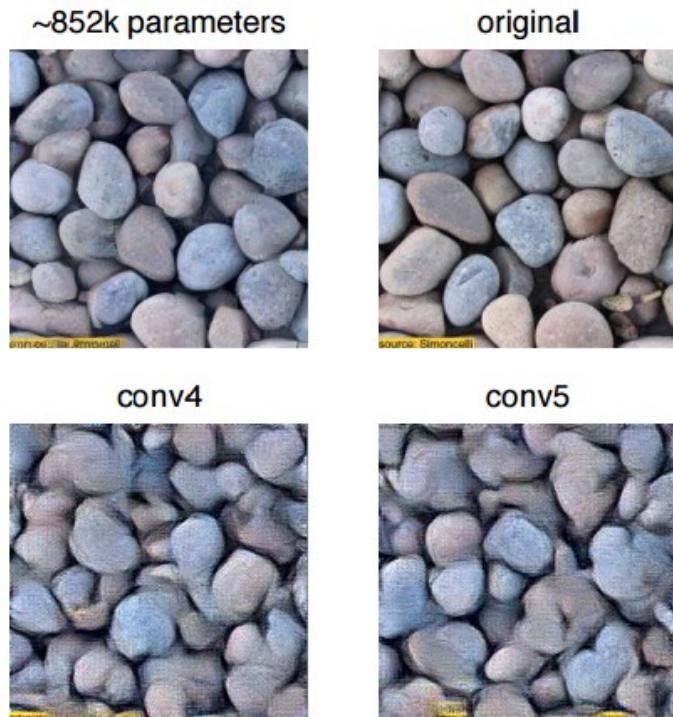


# Neural Style & Generative Adversarial Network

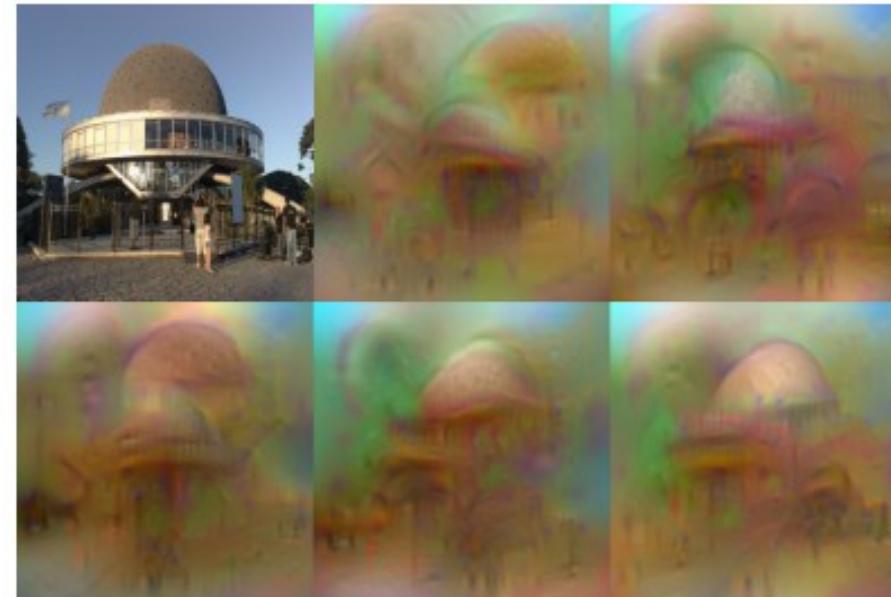
Fast Campus  
Start Deep Learning with Tensorflow

# Preliminaries

Texture Synthesis Using  
Convolutional Neural Networks  
**NIPS2015**



Understanding Deep Image Representations by  
Inverting Them  
**CVPR2015**



# A Neural Algorithm of Artistic Style



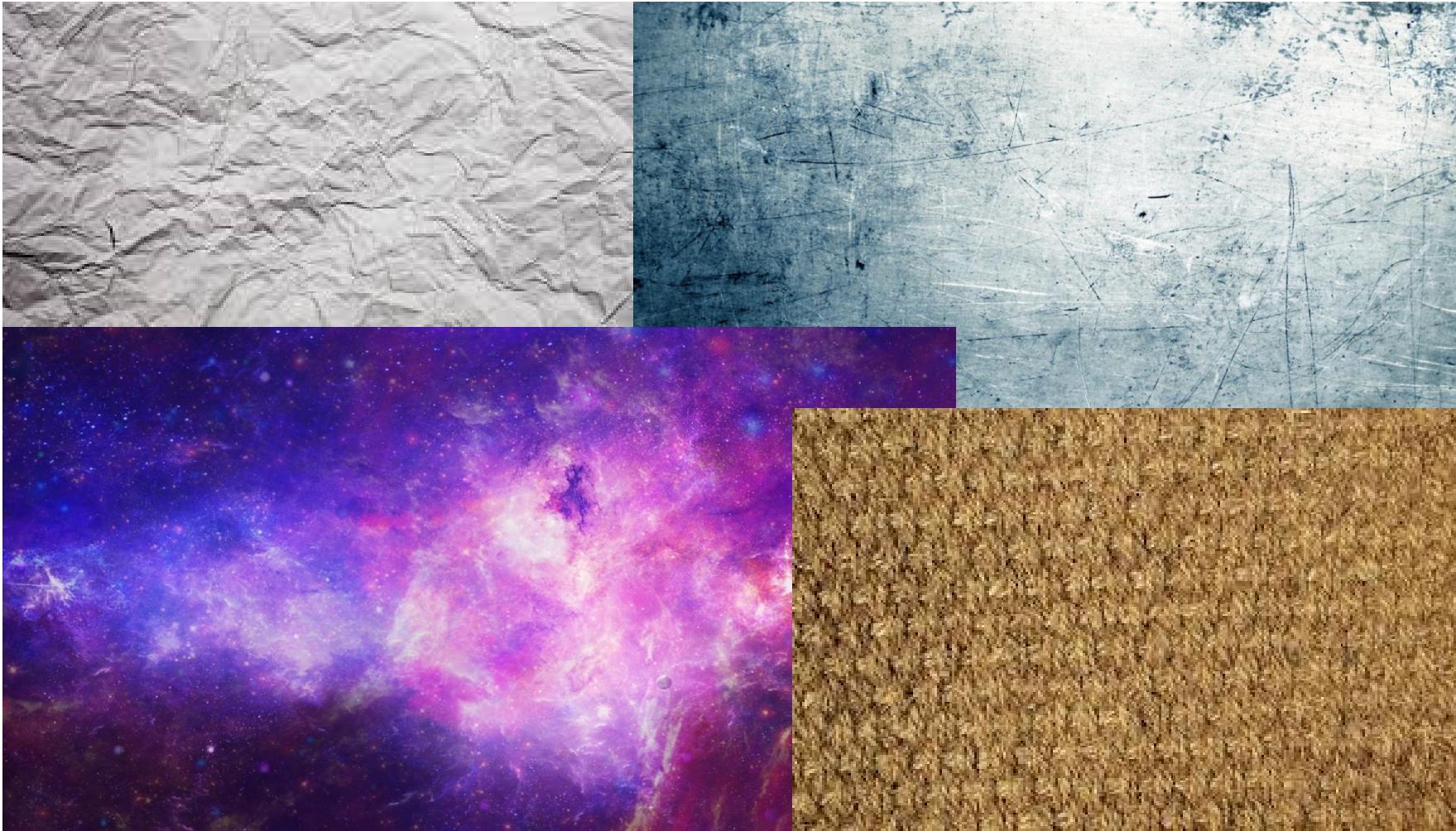
# **Texture Synthesis Using Convolutional Neural Networks**

## **-NIPS2015**

# A Neural Algorithm of Artistic Style



# Texture

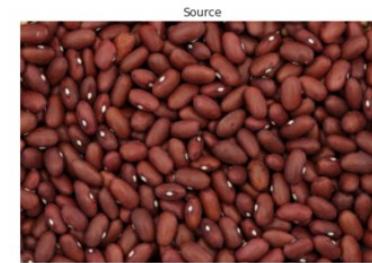
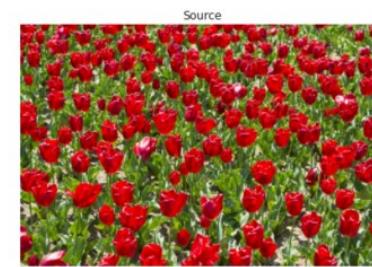
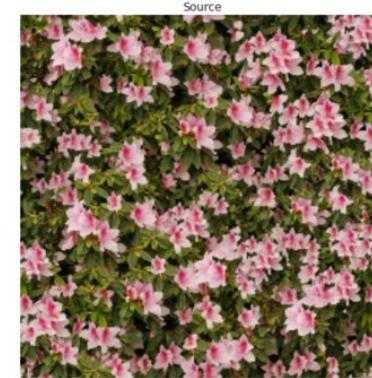


# Visual texture synthesis

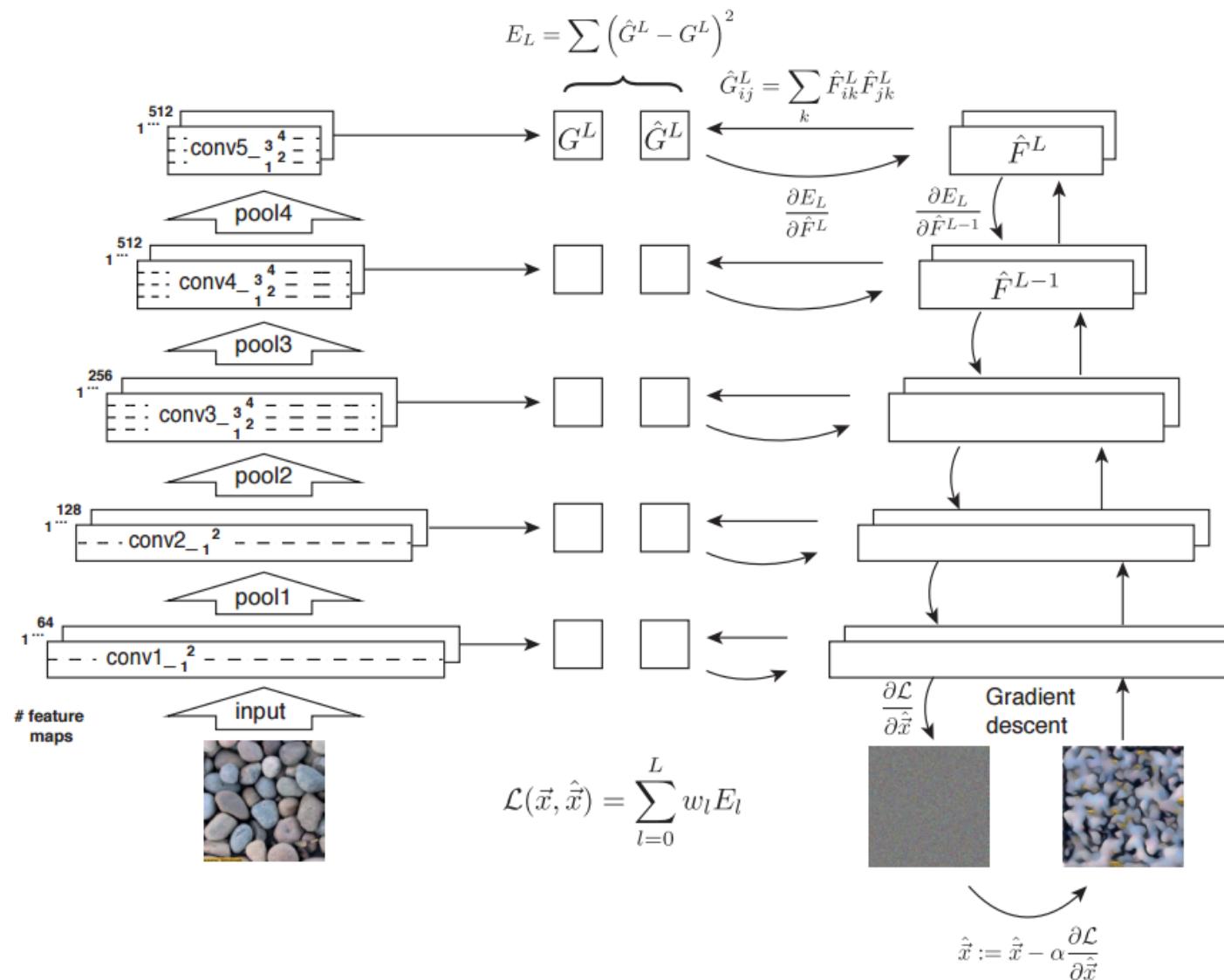


- Which one do you think is real?
- Right one is real.
- Goal of **texture synthesis** is to produce (arbitrarily many) new samples from an example texture.

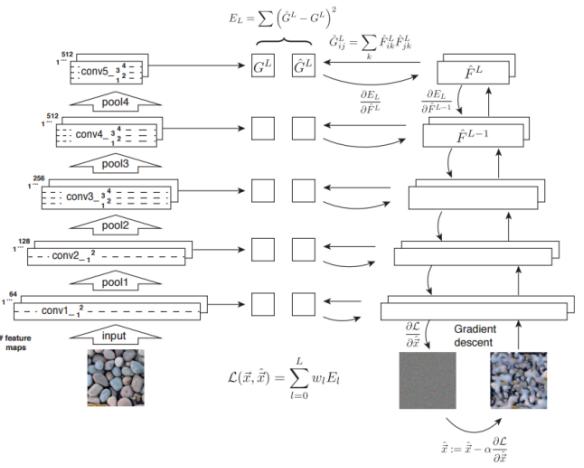
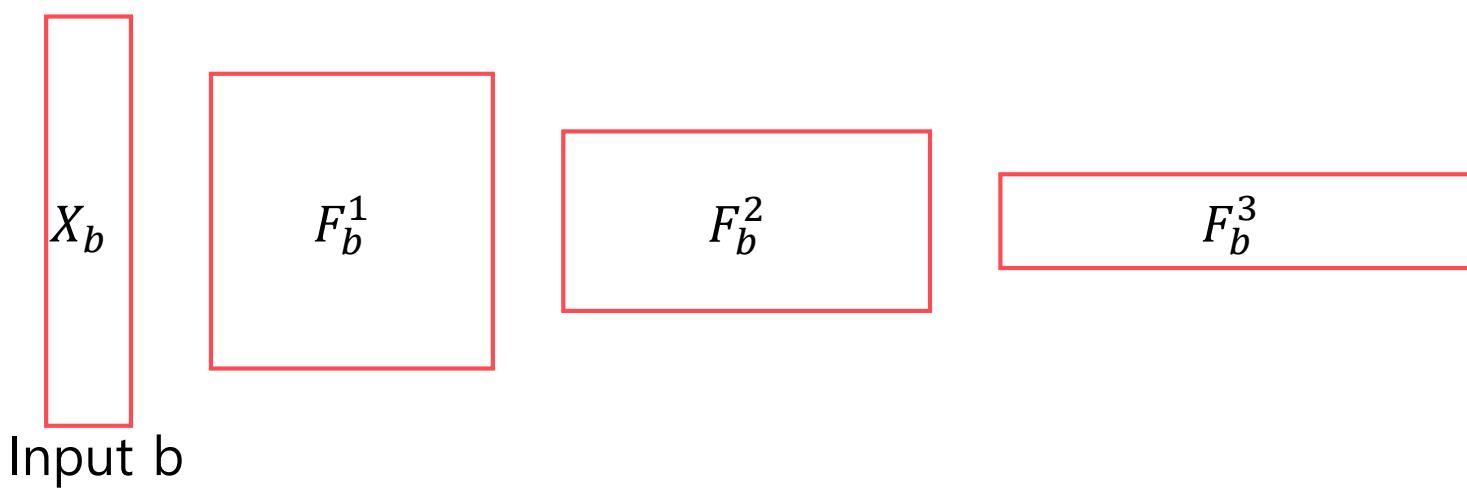
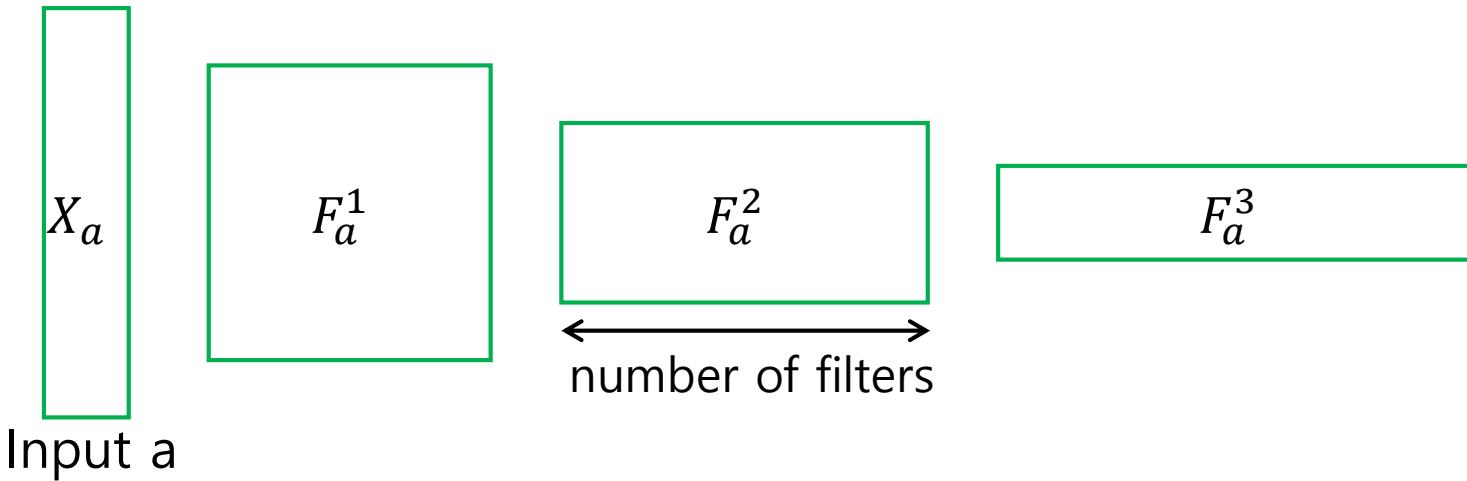
# Results of this work



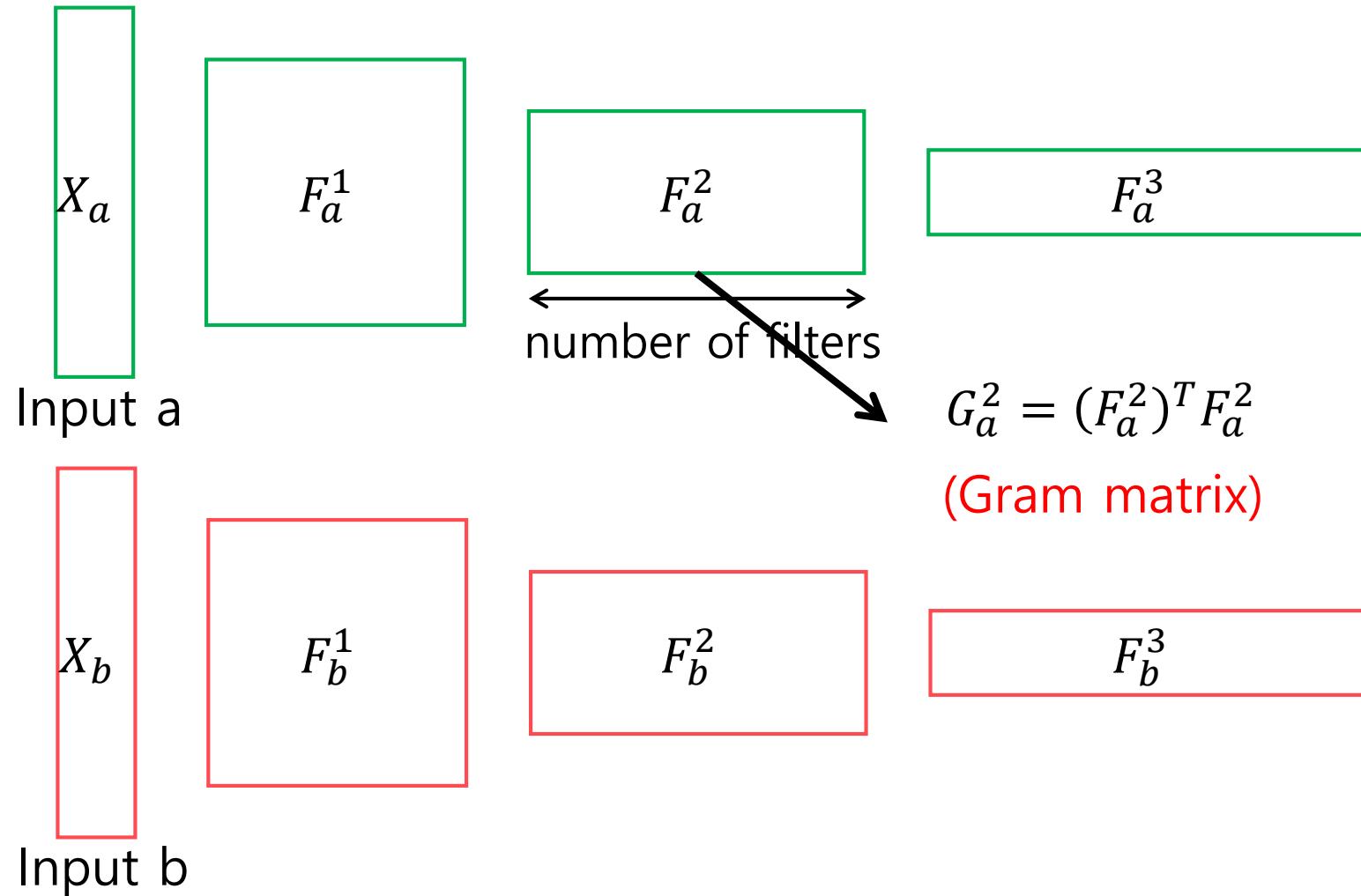
# How?



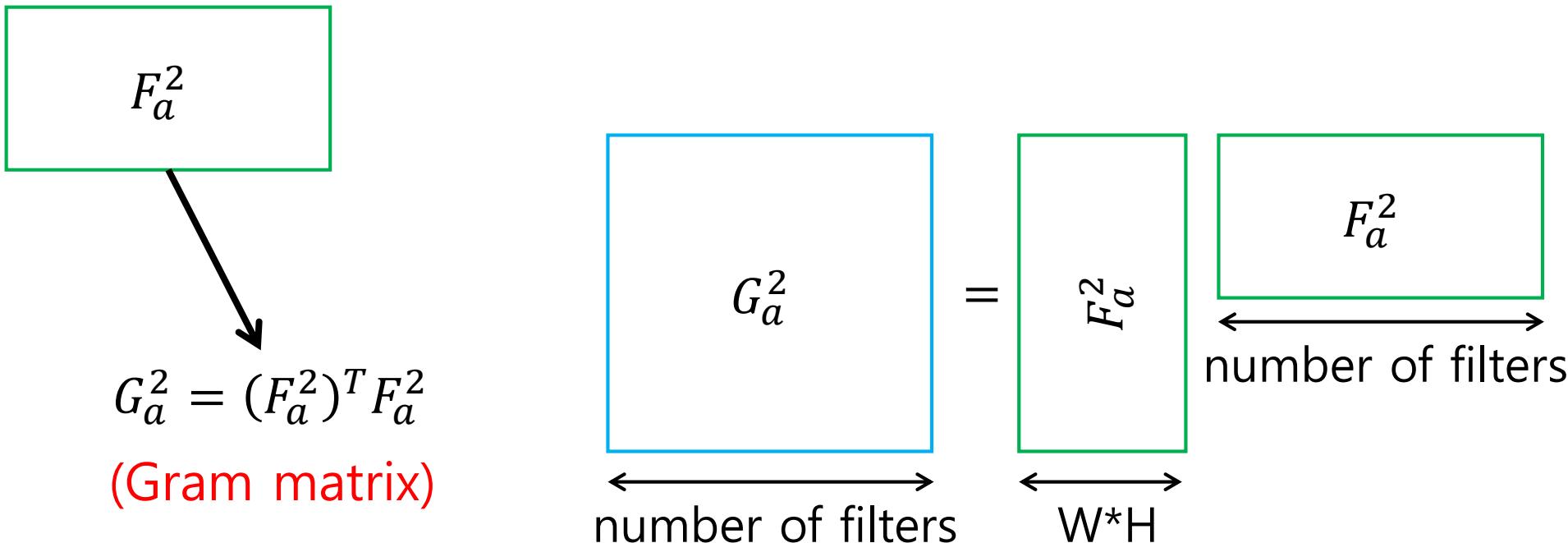
# Texture Model



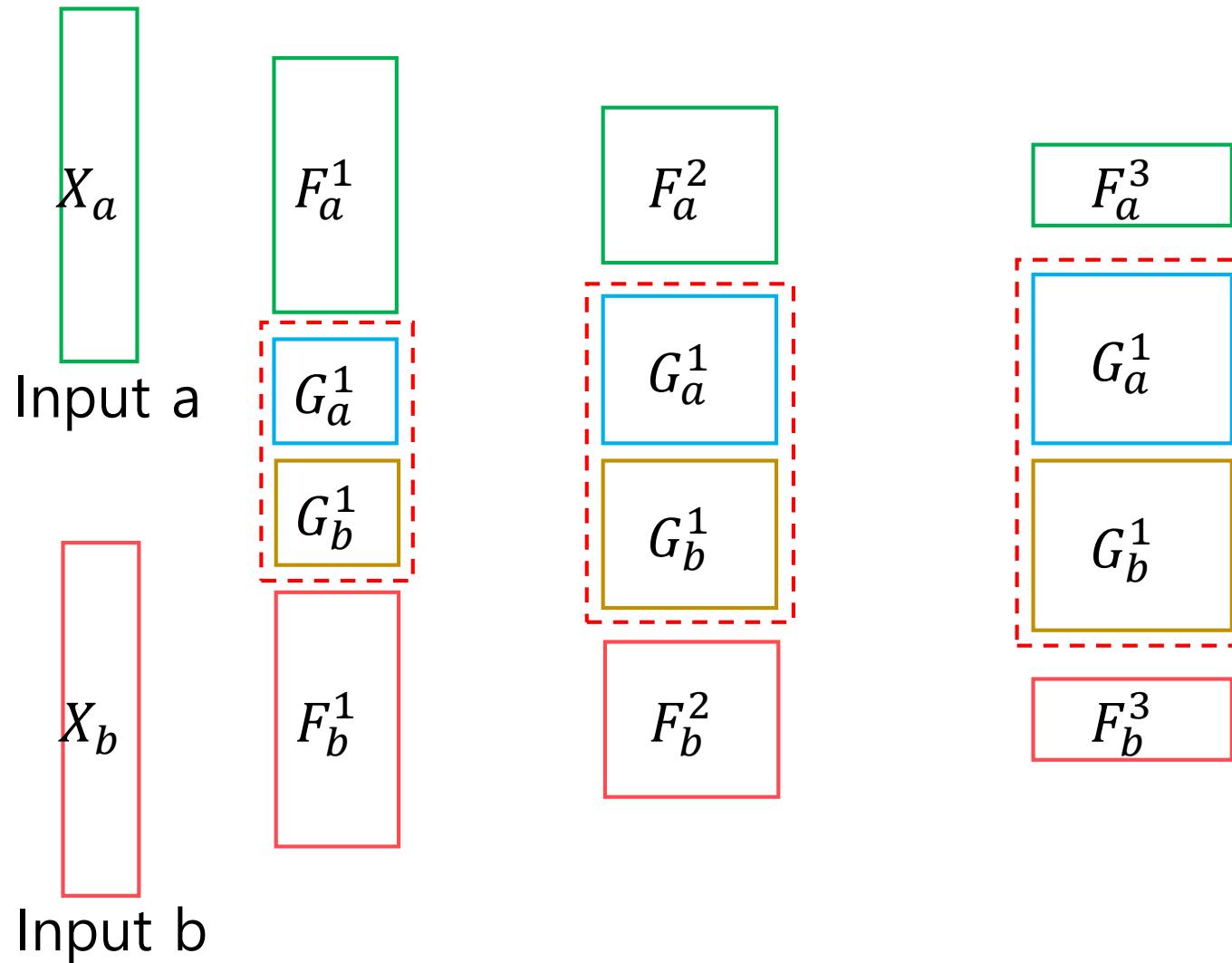
# Feature Correlations



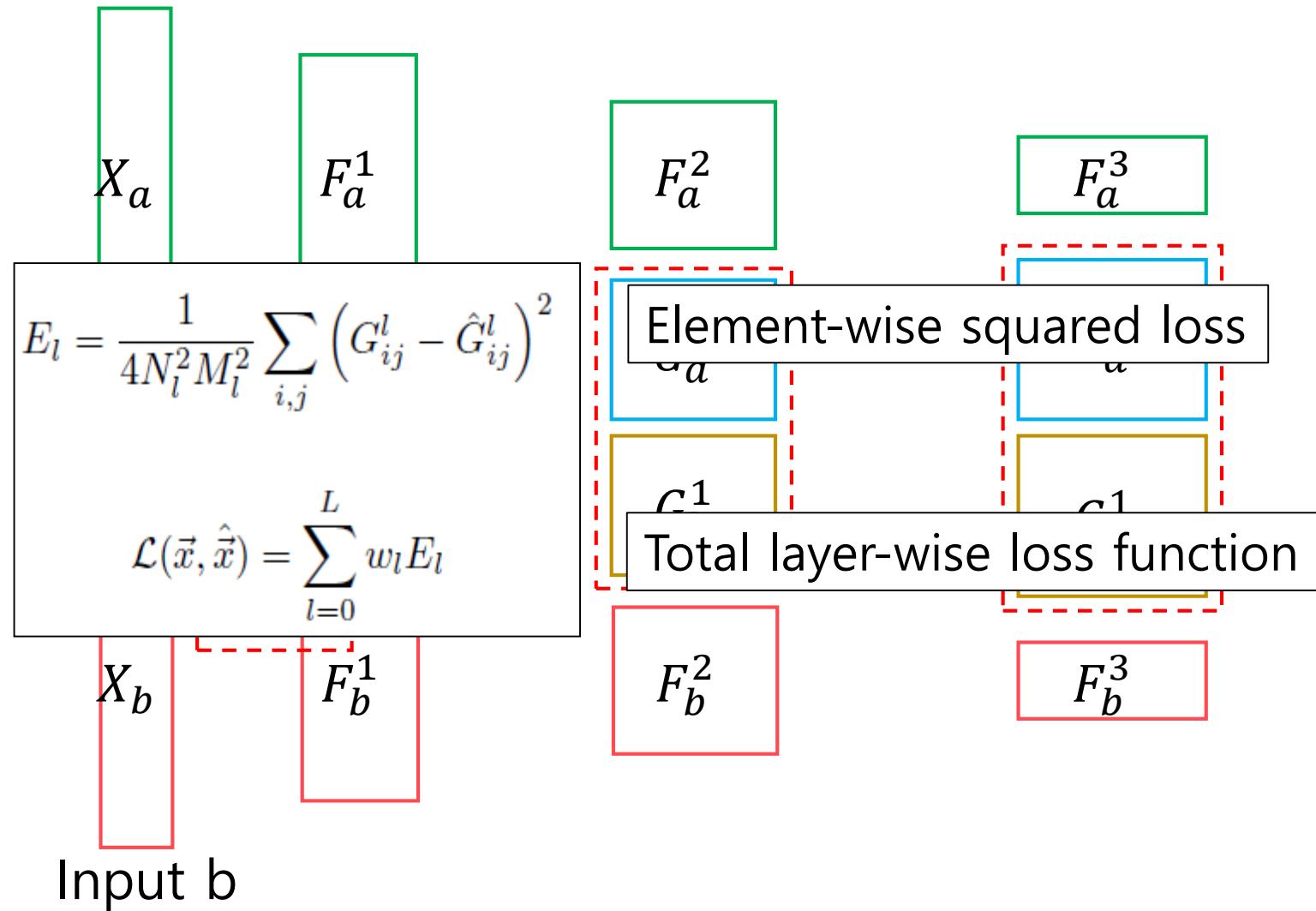
# Feature Correlations



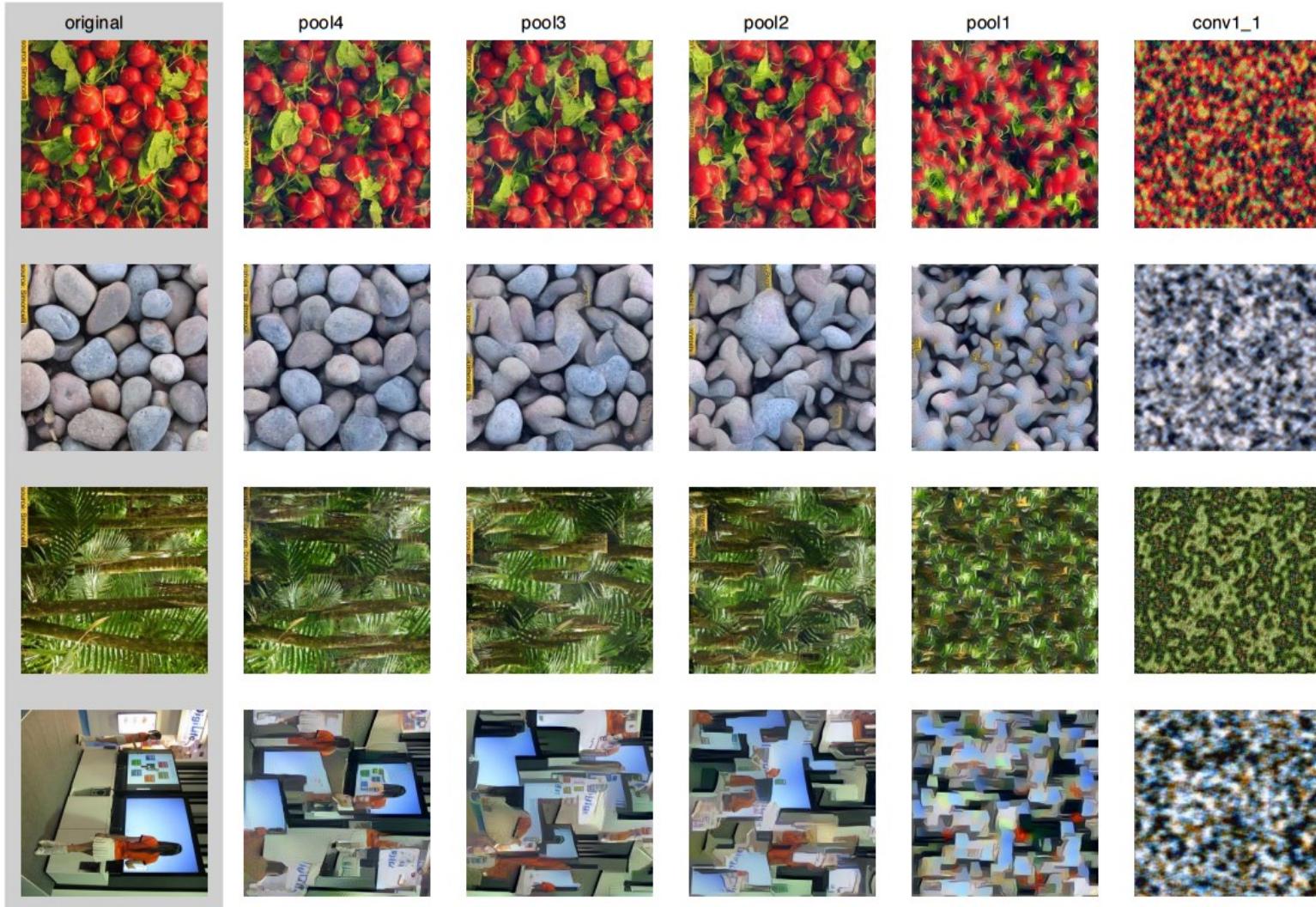
# Texture Generation



# Texture Generation



# Results



# Results

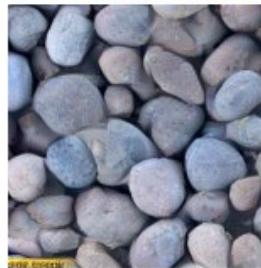
**A** ~1k parameters



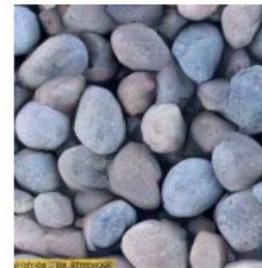
~10k parameters



~177k parameters



~852k parameters



original



**B** conv1



conv2



conv3



conv4



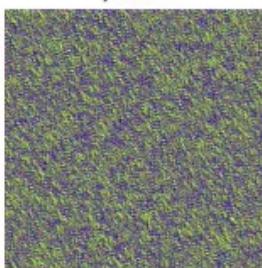
conv5



**C** conv1\_1



pool1



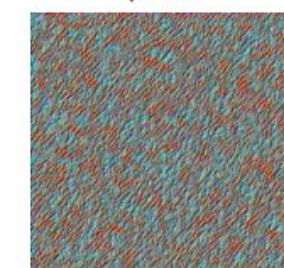
pool2



pool3



pool4



# **Understanding Deep Image Representations by Inverting Them**

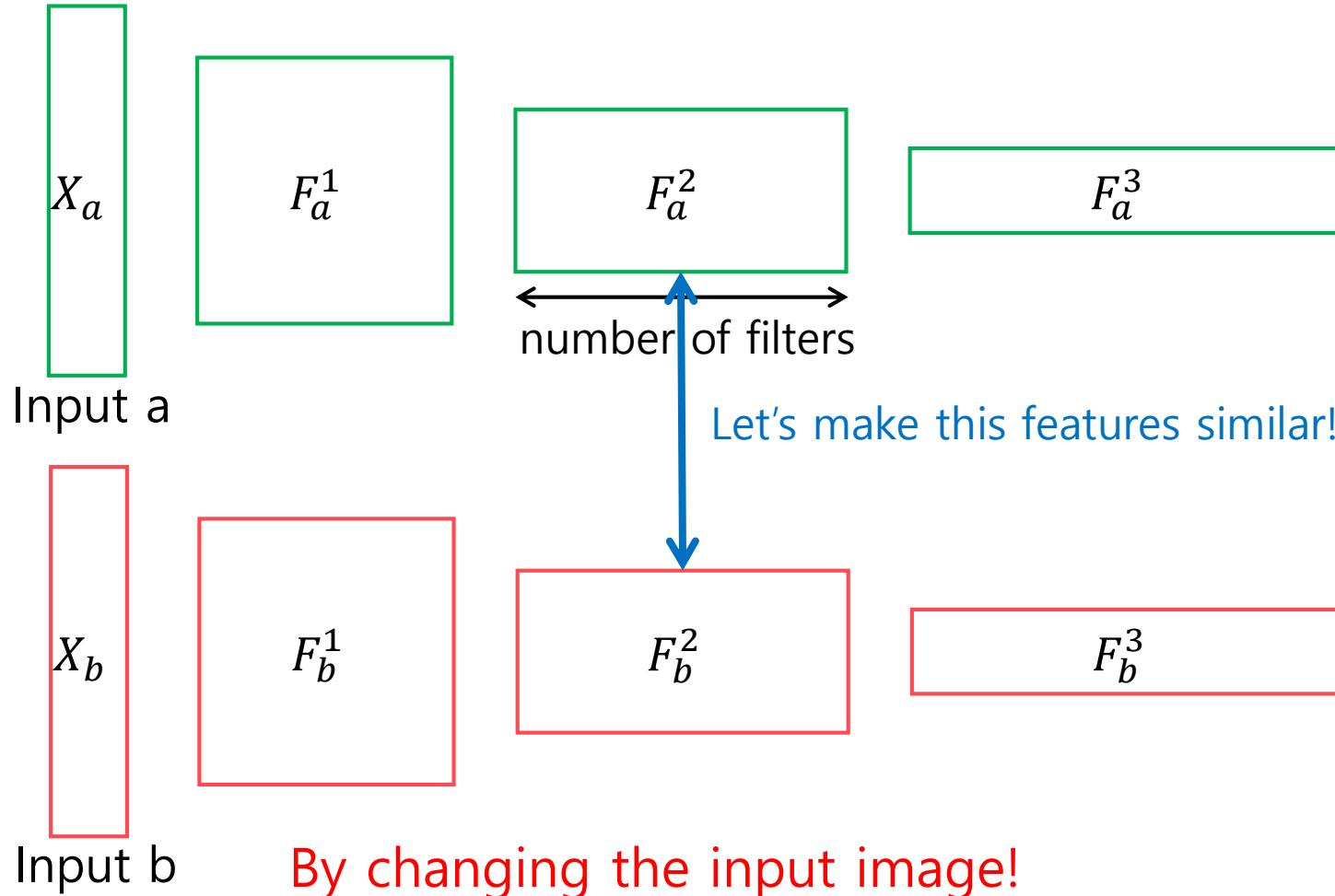
## **-CVPR2015**

# Reconstruction from feature map



Figure 1. **What is encoded by a CNN?** The figure shows five possible reconstructions of the reference image obtained from the 1,000-dimensional code extracted at the penultimate layer of a reference CNN[15] (before the softmax is applied) trained on the ImageNet data. From the viewpoint of the model, all these images are practically equivalent. This image is best viewed in color/screen.

# Reconstruction from feature map



# Receptive Field

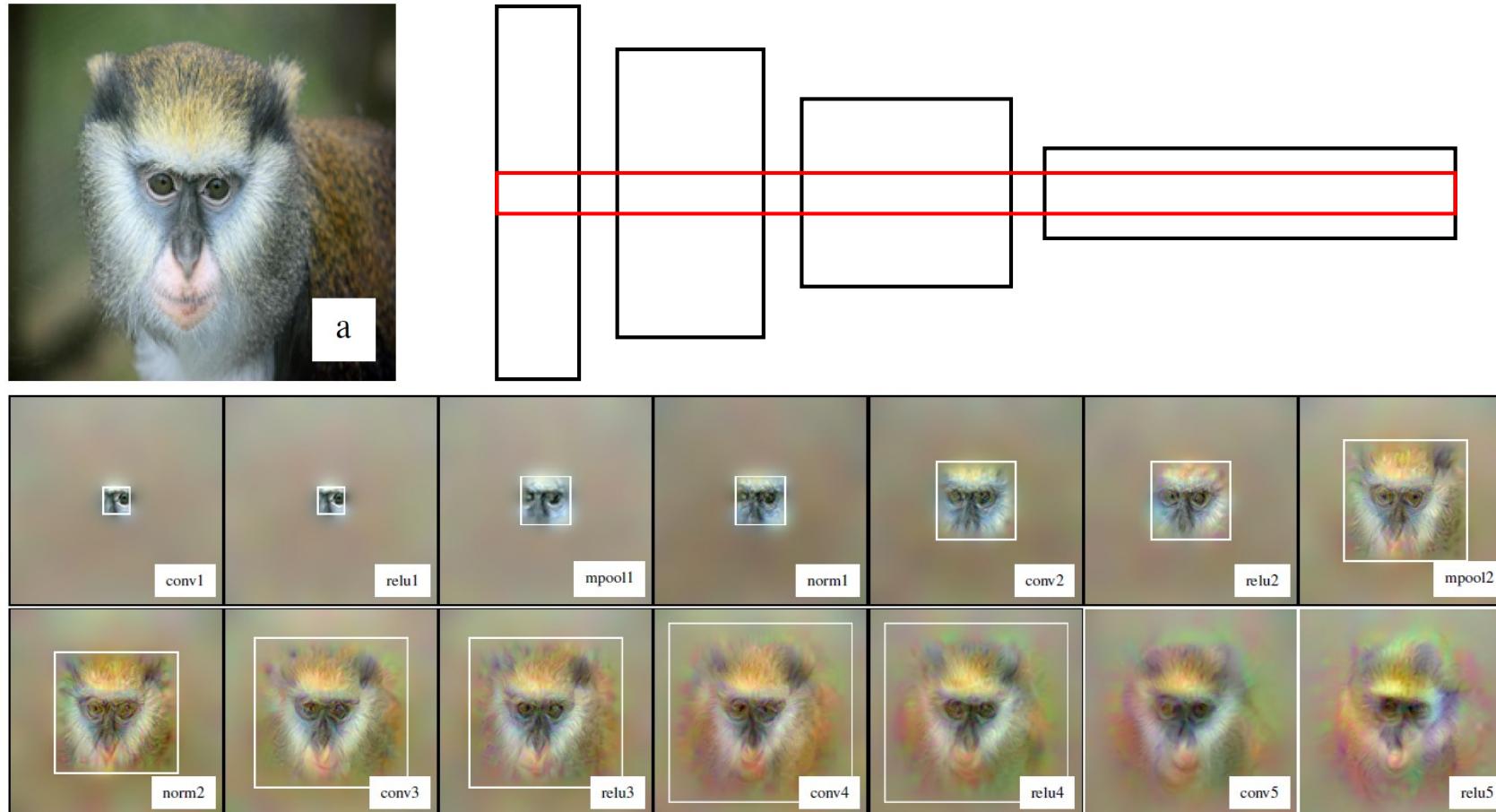
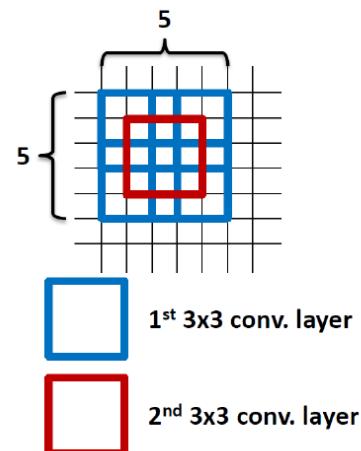


Figure 9. **CNN receptive field.** Reconstructions of the image of Fig. 5.a from the central  $5 \times 5$  neuron fields at different depths of CNN-A. The white box marks the field of view of the  $5 \times 5$  neuron field. The field of view is the entire image for conv5 and relu5.

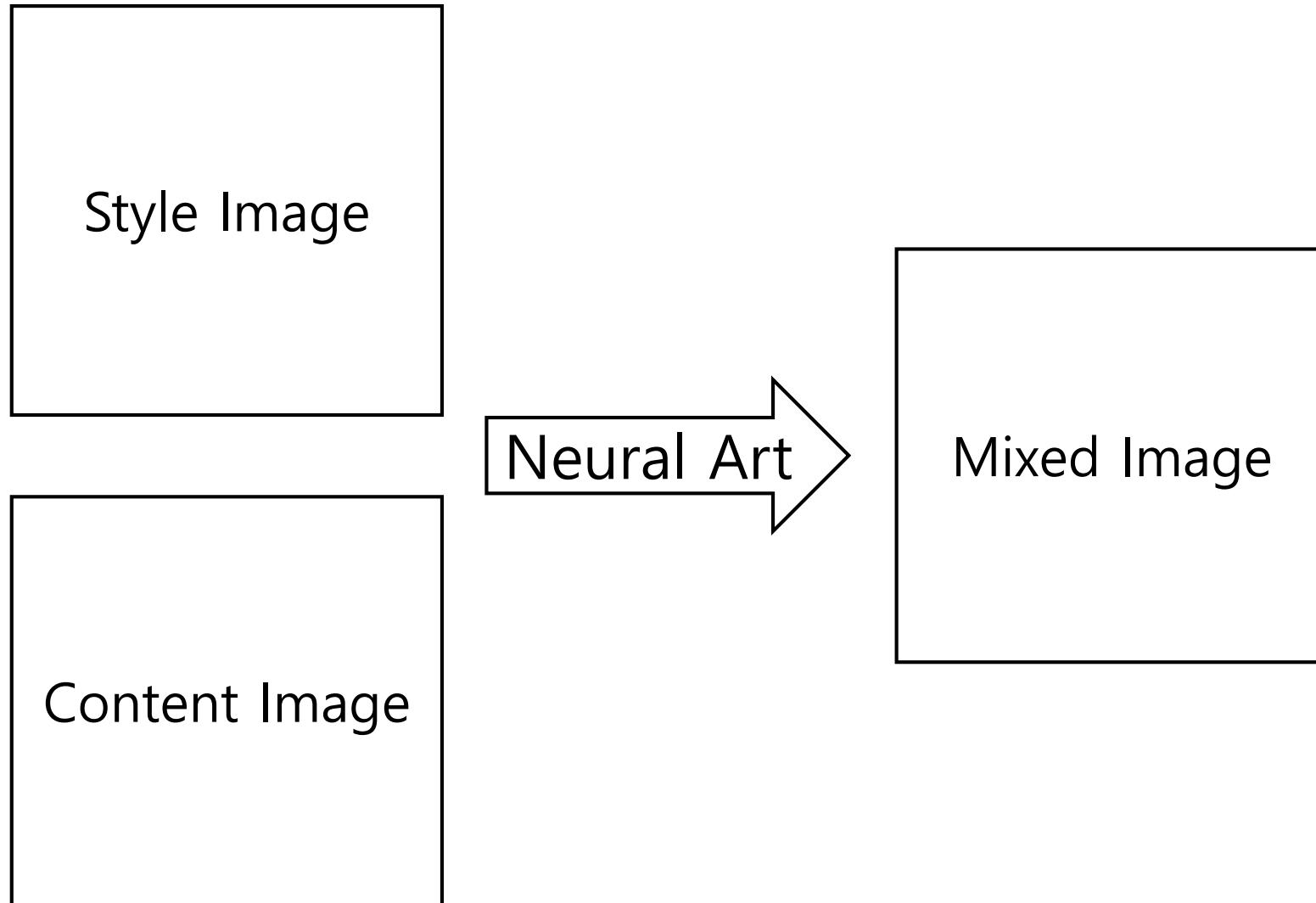
# VGG

- 3x3 filter만 반복해서 사용
- Why??
  - Convolution filter를 stack하면 더 큰 receptive field를 가질 수 있음
    - 2개의 3x3 filter = 5x5 filter
    - 3개의 3x3 filter = 7x7 filter
  - Parameter수는 큰 filter 사용하는 경우에 비하여 감소 → regularization 효과

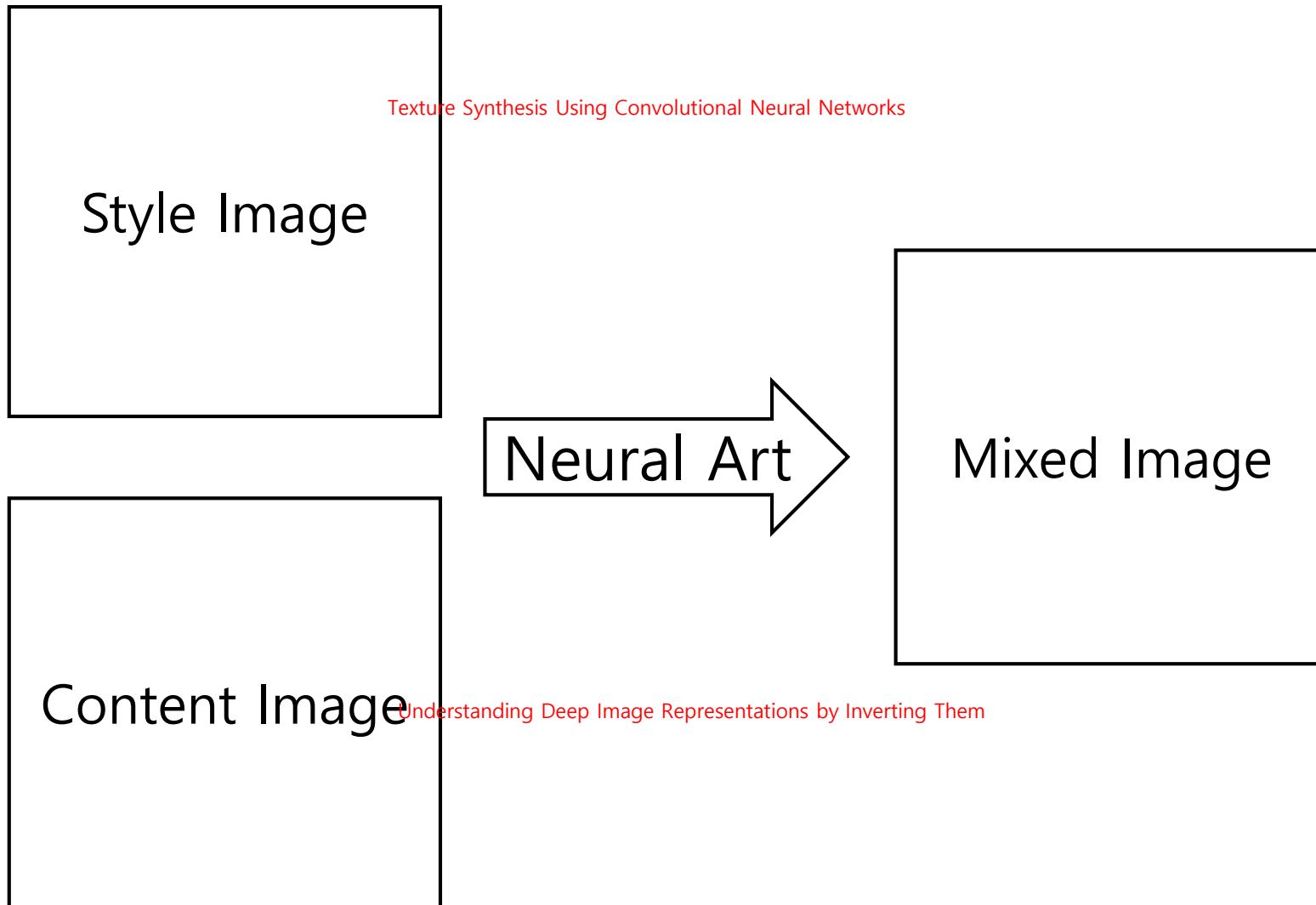


# **A Neural Algorithm of Artistic Style**

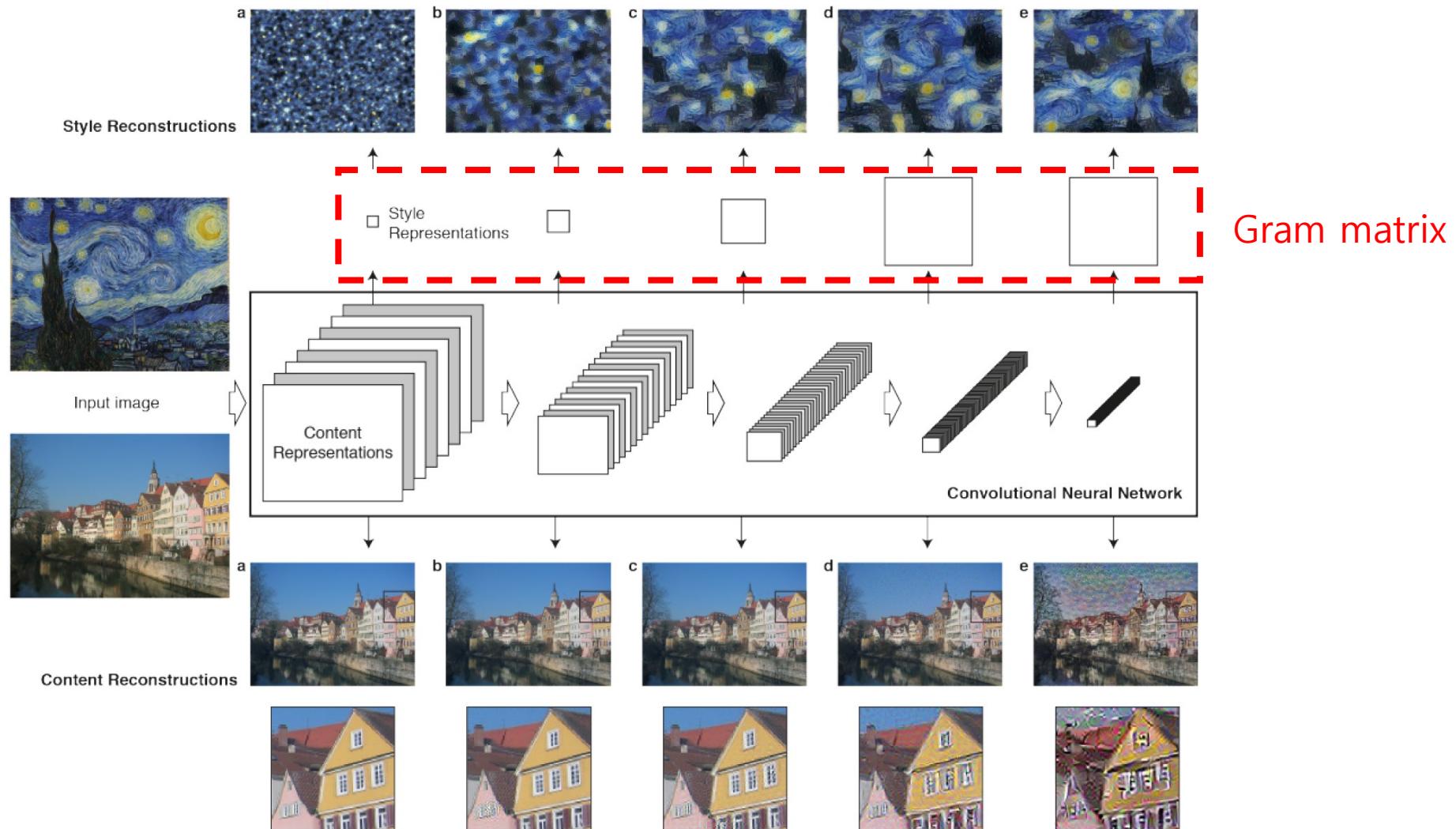
# How?



# How?



# How?



# Neural Art

$\vec{p}$ : original photo,  $\vec{a}$ : original artwork  
 $\vec{x}$ : image to be generated

## Content

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 .$$

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}^l} = \begin{cases} (F^l - P^l)_{ij} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 . \end{cases}$$

## Style

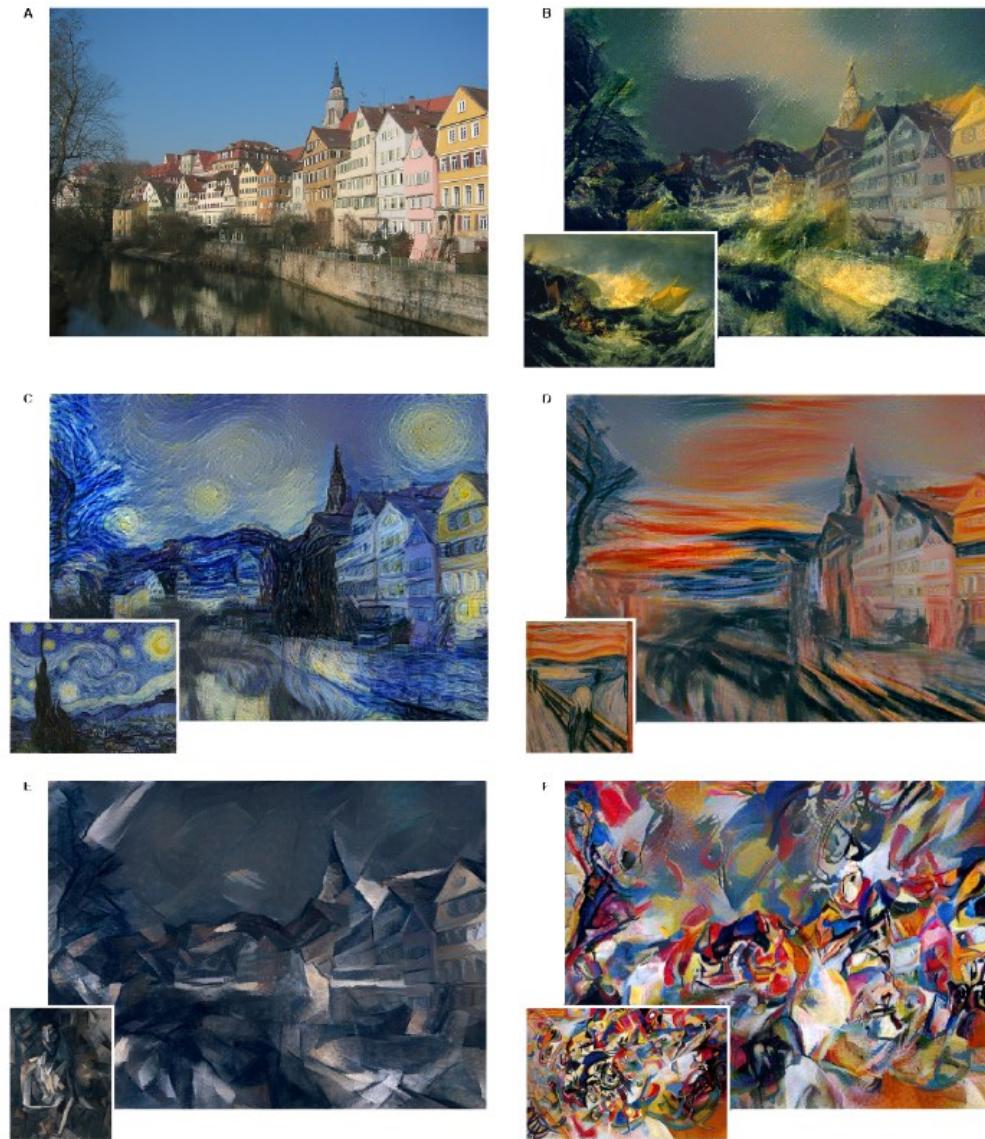
$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \quad G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L w_l E_l$$

Total loss = content loss + style loss

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

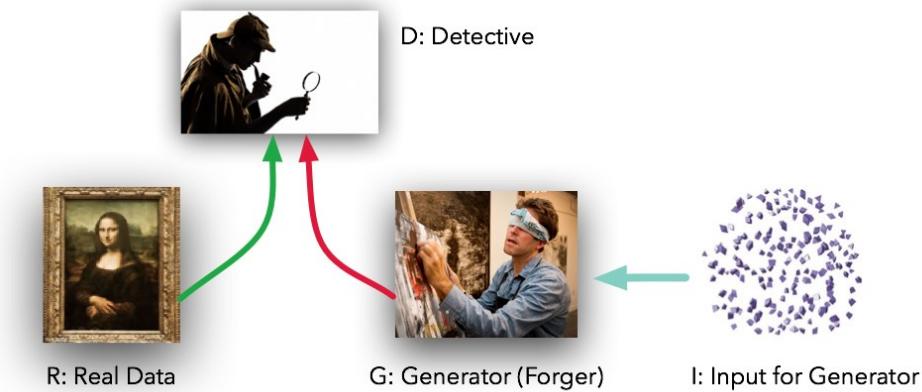
# Results



# Results



# 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?

是	是	大	大	时	时	来	来	会	会
在	在	人	人	分	分	日	日	现	现
不	不	为	为	市	市	能	能	公	公
国	国	年	年	学	学	方	方	生	生
上	上	以	以	业	业	者	者	高	高
有	有	出	出	后	后	车	车	得	得



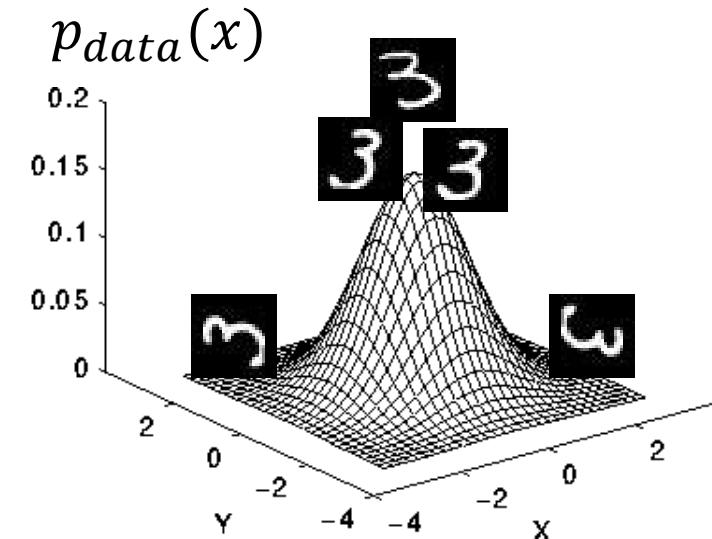
# Generative Model

- Data:



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

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

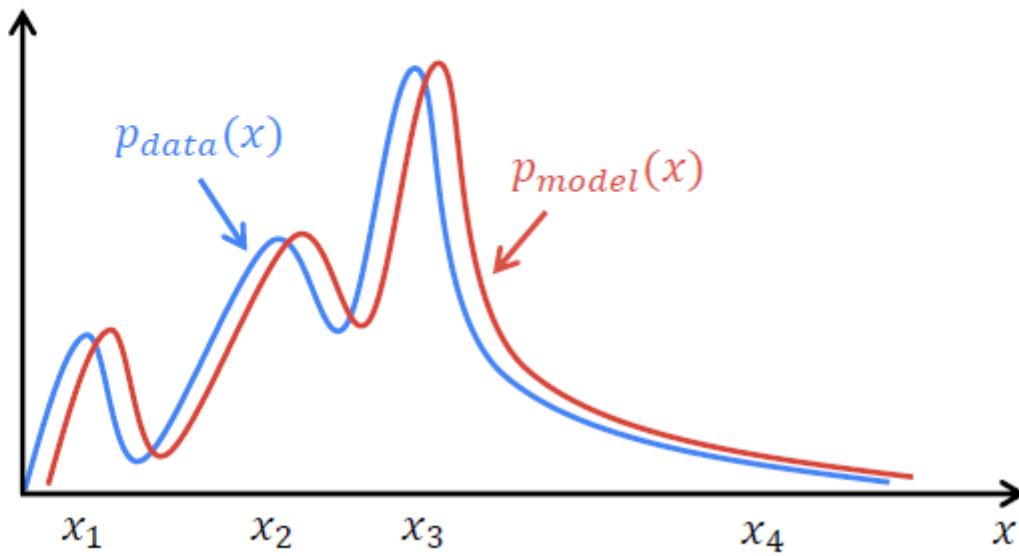


# Generative Model

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

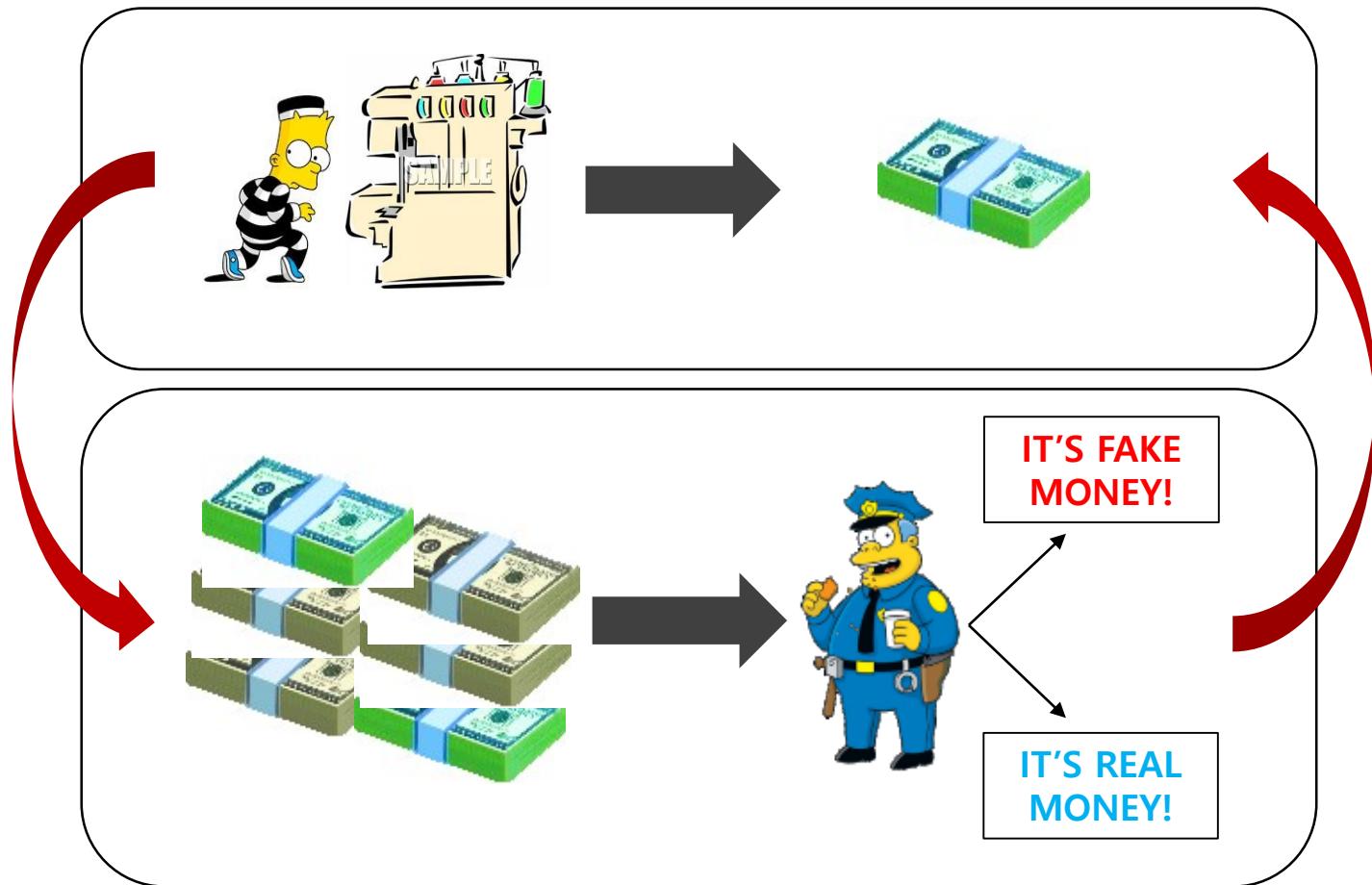
↗ Distribution of images generated by the model

↖ Distribution of actual images

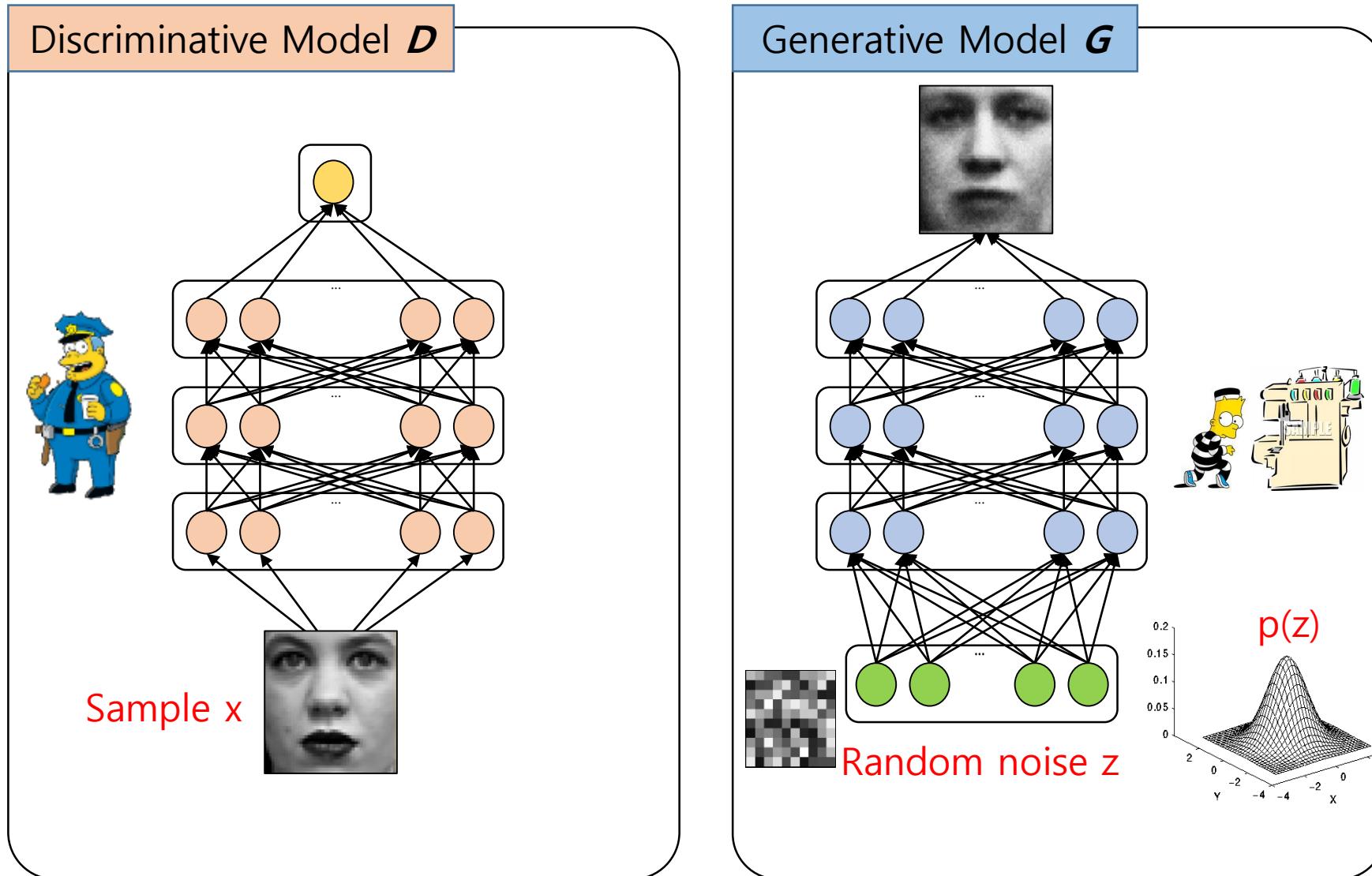


# Generative Adversarial Network

- Counterfeitors vs Police Game



# Generative Adversarial Network

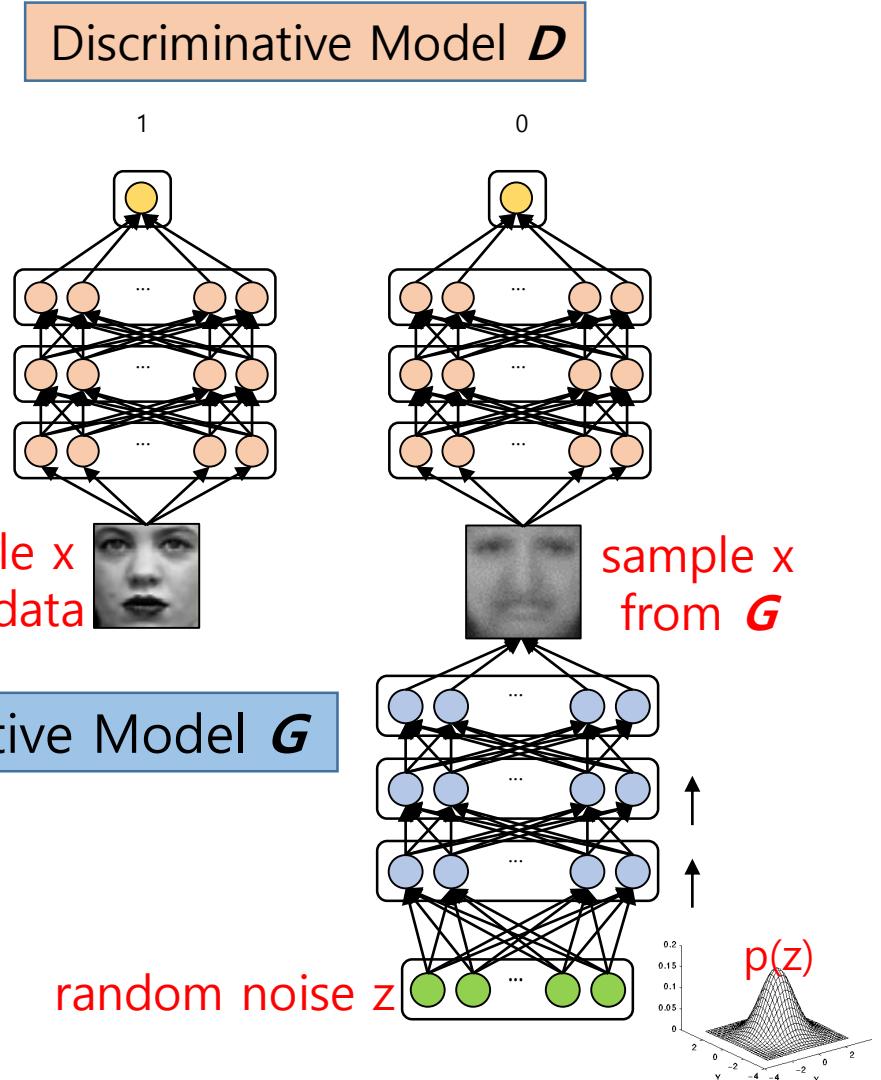


# Generative Adversarial Network

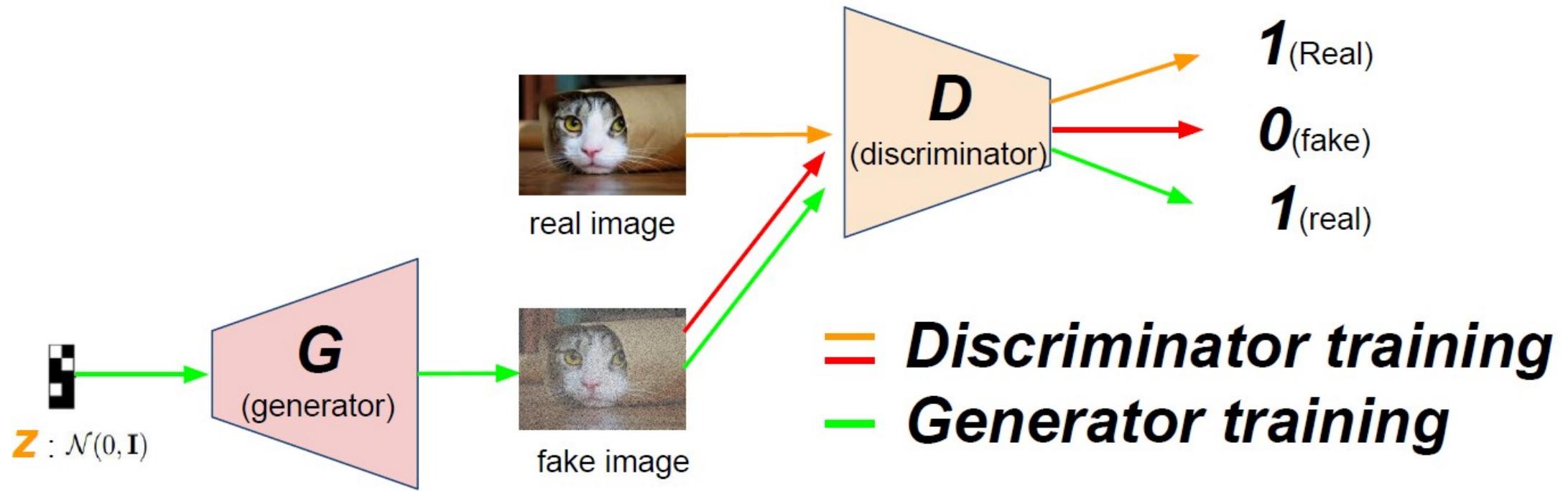
- Discriminative Model D
  - Try to classify the sample  $x$
  - $D(x)=1$  when  $x$  from Data
  - $D(x)=0$  when  $x$  from  $G$  (generator)

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

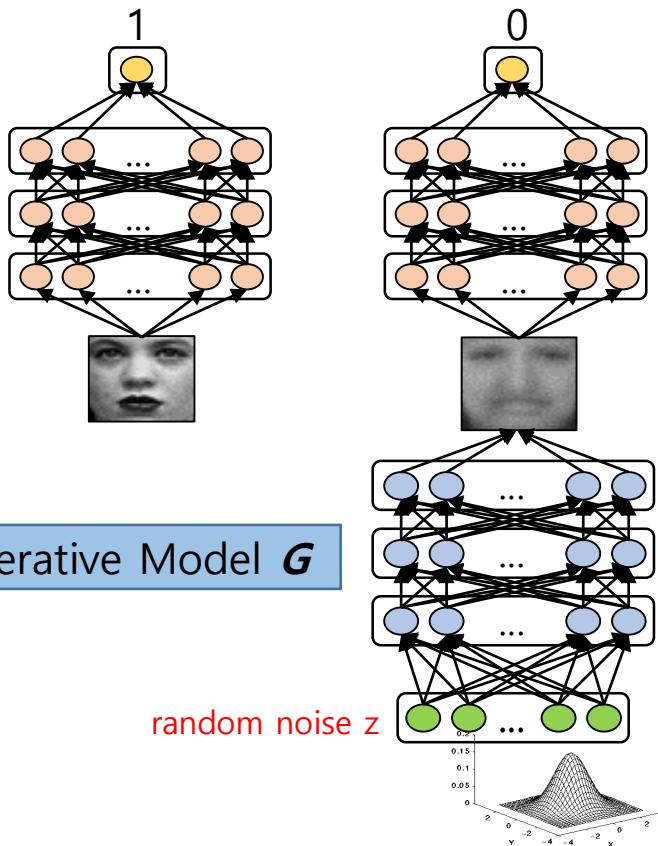


# Generative Adversarial Network

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

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

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

```

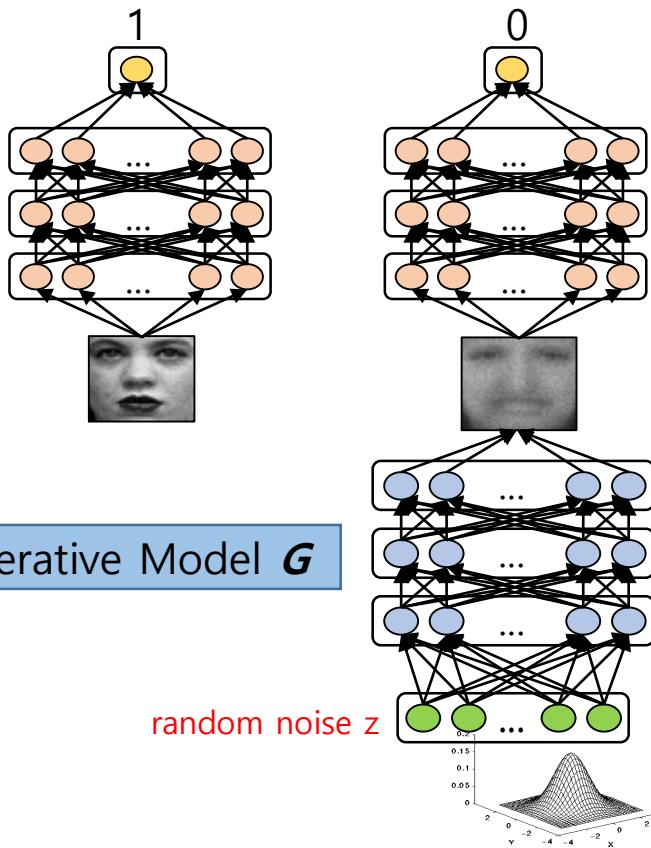
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

# 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

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

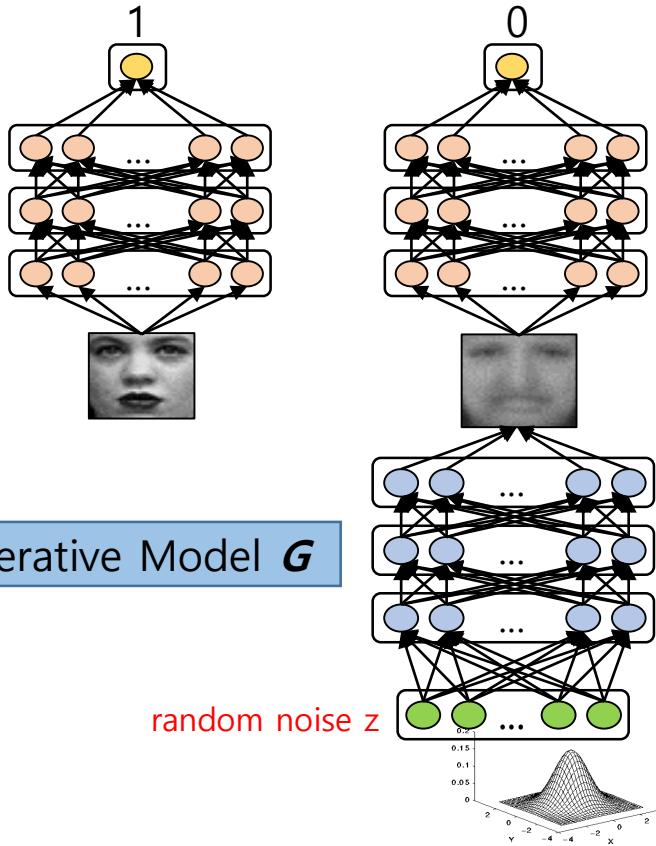
Stochastic  
Gradient

# 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

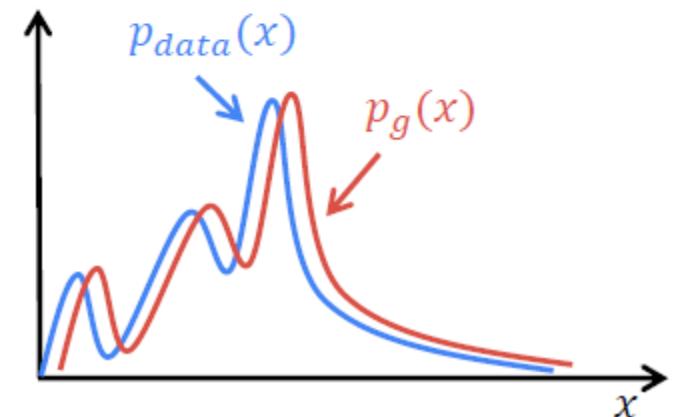
$$E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

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

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

Jenson-Shannon divergence

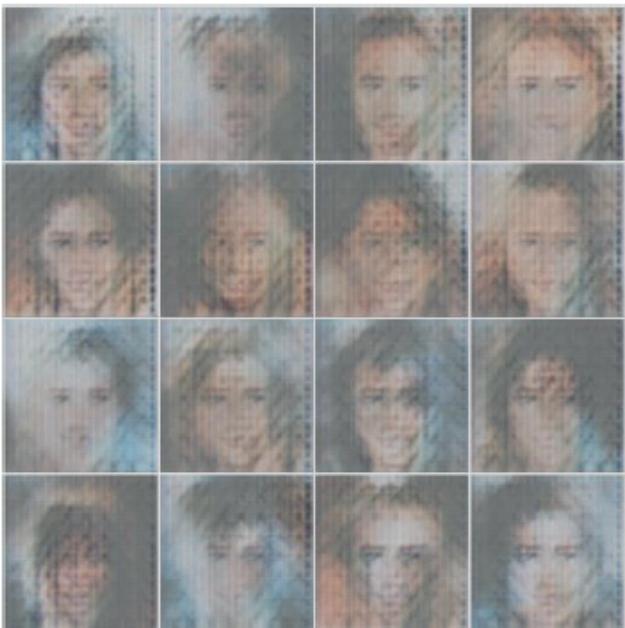
KL Divergence



# Non-Saturating Game

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

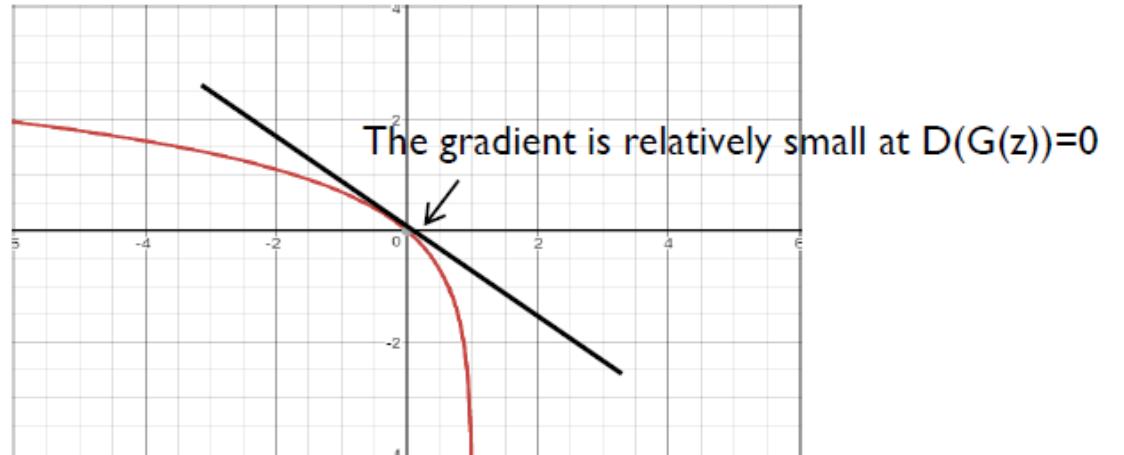
Objective function of G



Images created by the generator  
at the beginning of training

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.



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

# Non-Saturating Game

```
1 # tensorflow  
2 tf.losses.sigmoid_cross_entropy()  
3  
4 # pytorch  
5 nn.BCELoss()
```

- Practical Usage

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

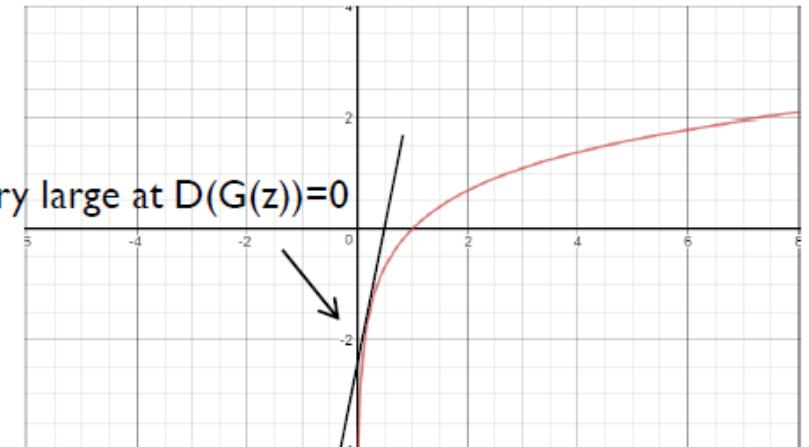
$$\min_G E_{z \sim p_z(z)}[-y \log D(G(z)) - (1 - y) \log(1 - D(G(z)))]$$

$$\downarrow \quad y = 1$$

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

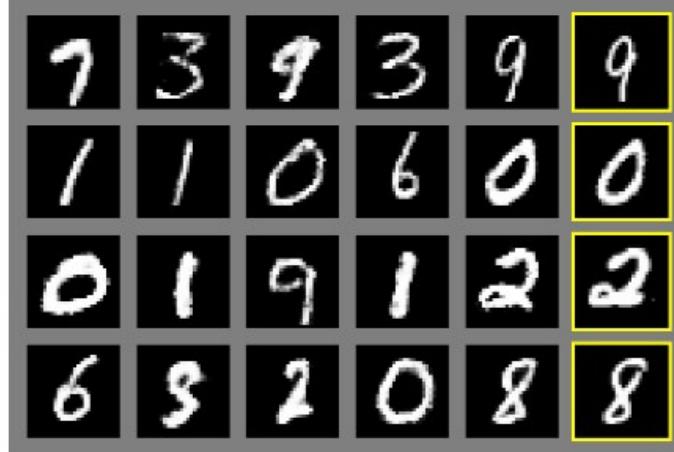
$$\begin{aligned} & \cancel{\min_G E_{z \sim p_z(z)}[\log(1 - D(G(z)))]} \\ & \downarrow \text{Modification (heuristically motivated)} \\ & \max_G E_{z \sim p_z(z)}[\log D(G(z))] \end{aligned}$$

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



$$y = \log(x)$$

# GAN Results



a)



b)



c)



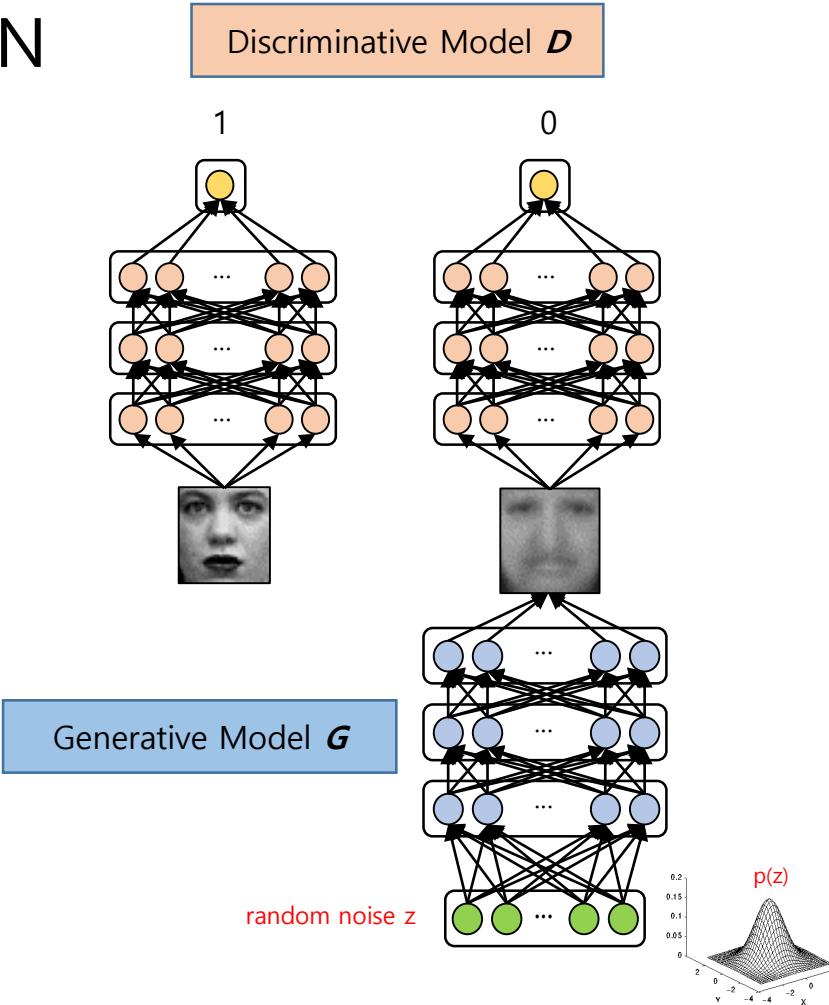
d)

# **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Network**

## **-ICLR 2016**

# DCGAN

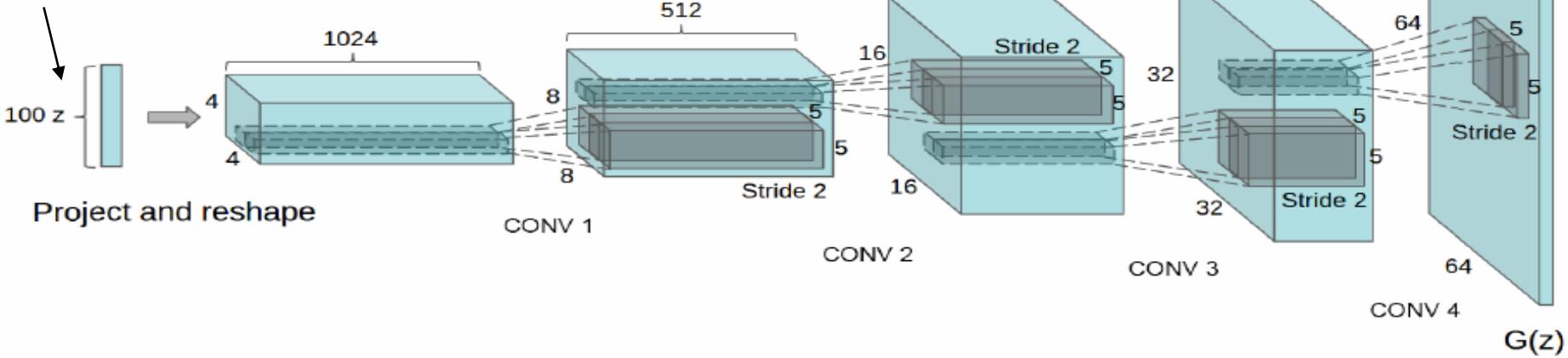
- Deep Convolutional Network + GAN
- Tricks for stable training
- Experimental Analysis



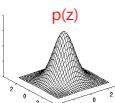
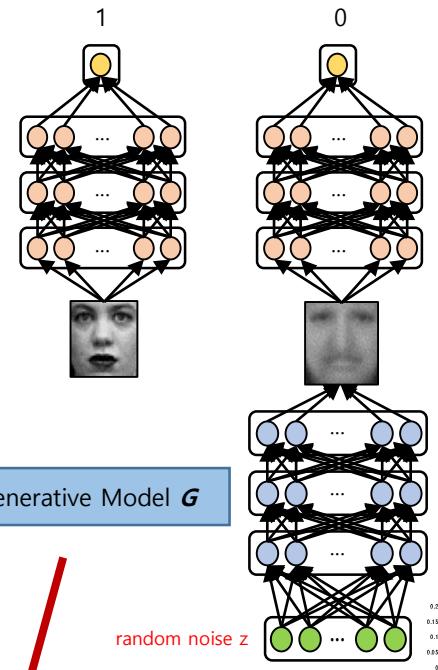
# DCGAN

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

$z$ :  
uniform dist.



Discriminative Model  $D$



# 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



# Convolution Filters



**Random filters**

**Trained filters**

# Experiments

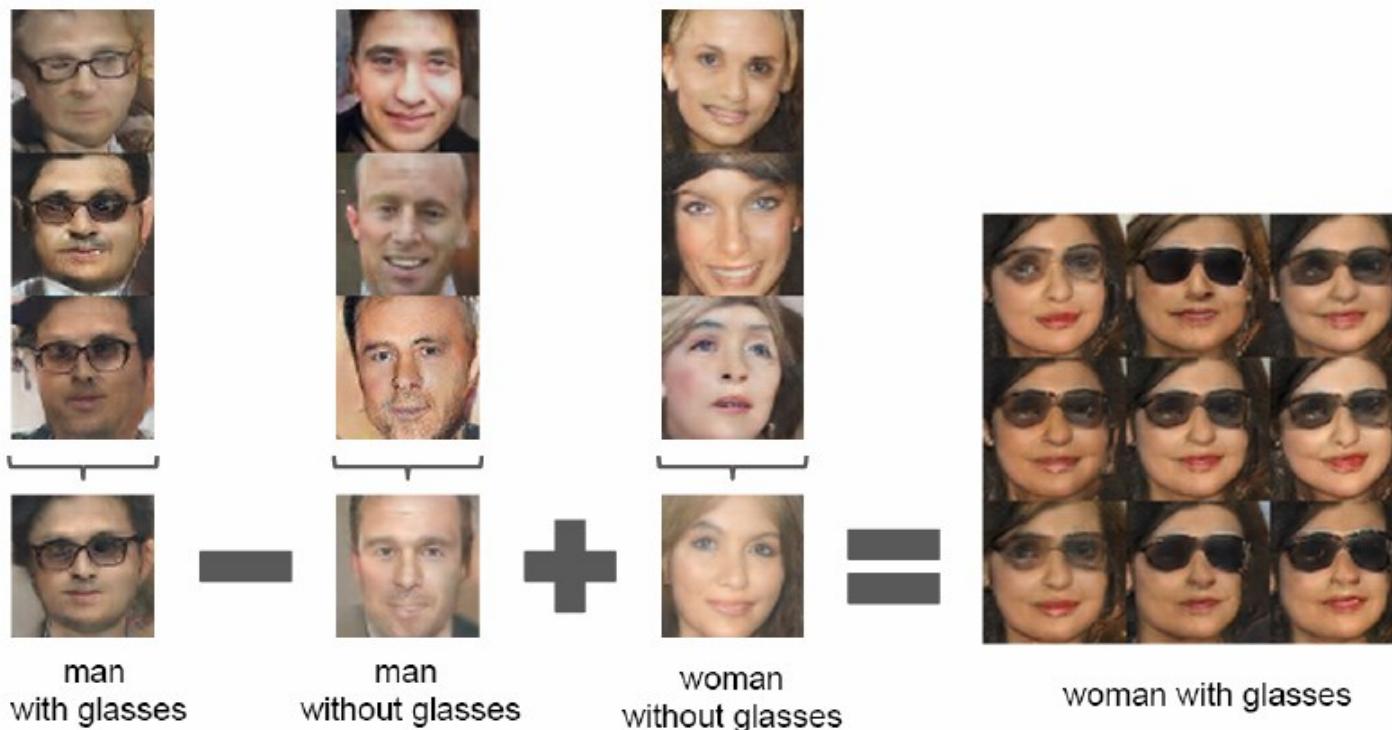
- Which activations(feature map) in CNN has representation of 'window'?
- At feature activations, assign neuron in the window region is 1, otherwise 0.
- Logistic regression to find window-representative feature map.

Window  
feature map →  
removal

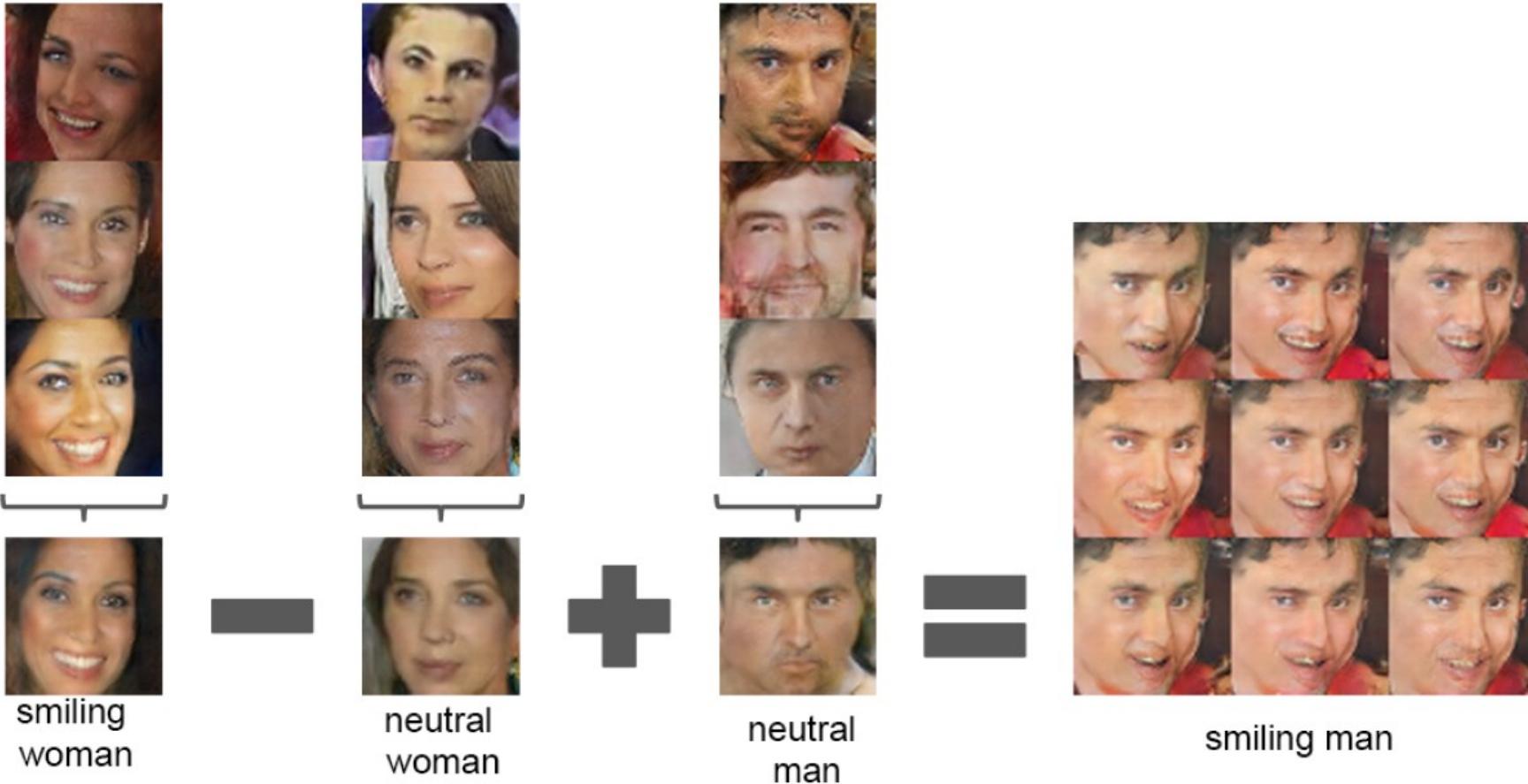


# Experiments

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



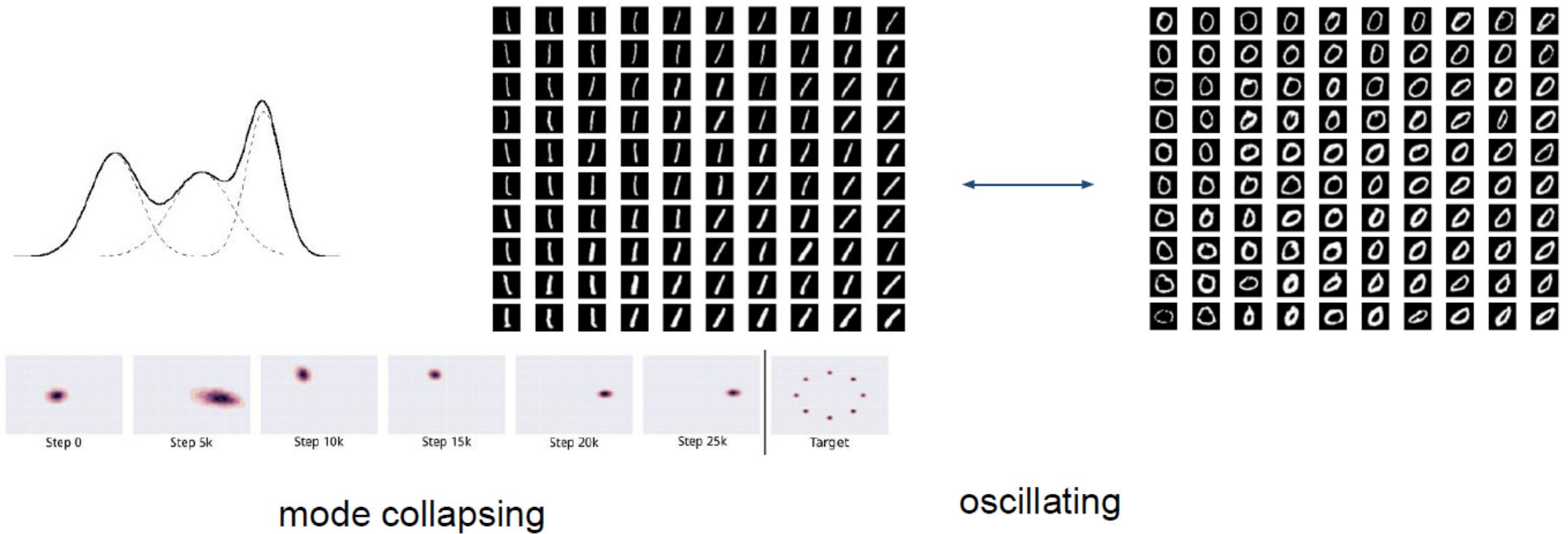
# Experiments



# Experiments



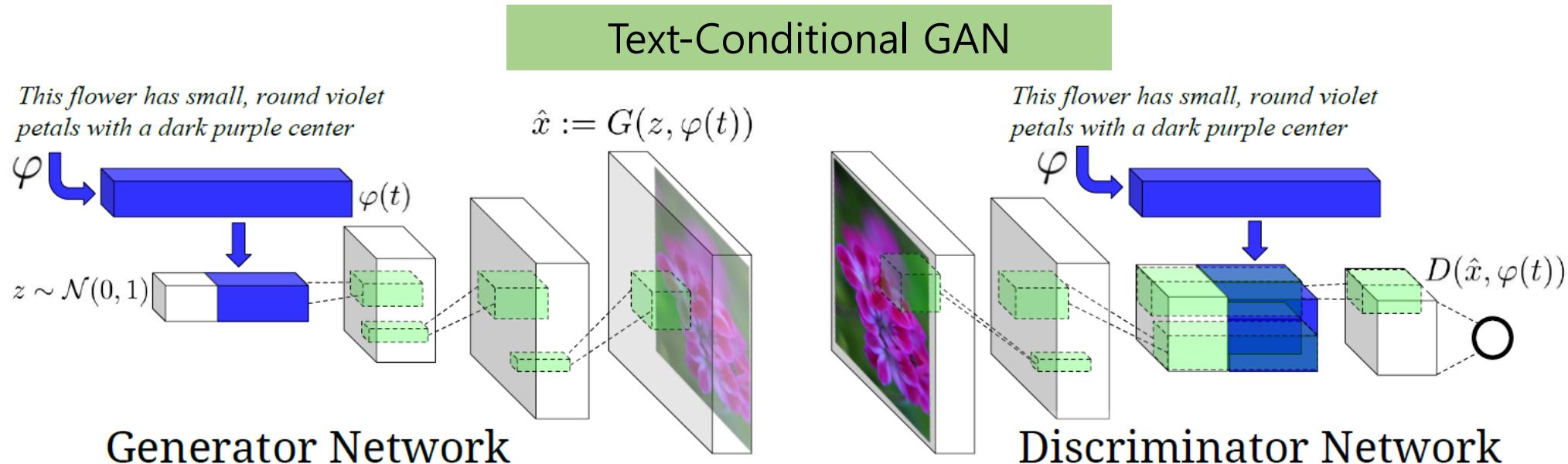
# Pitfall of GAN



**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



# Conditional GAN

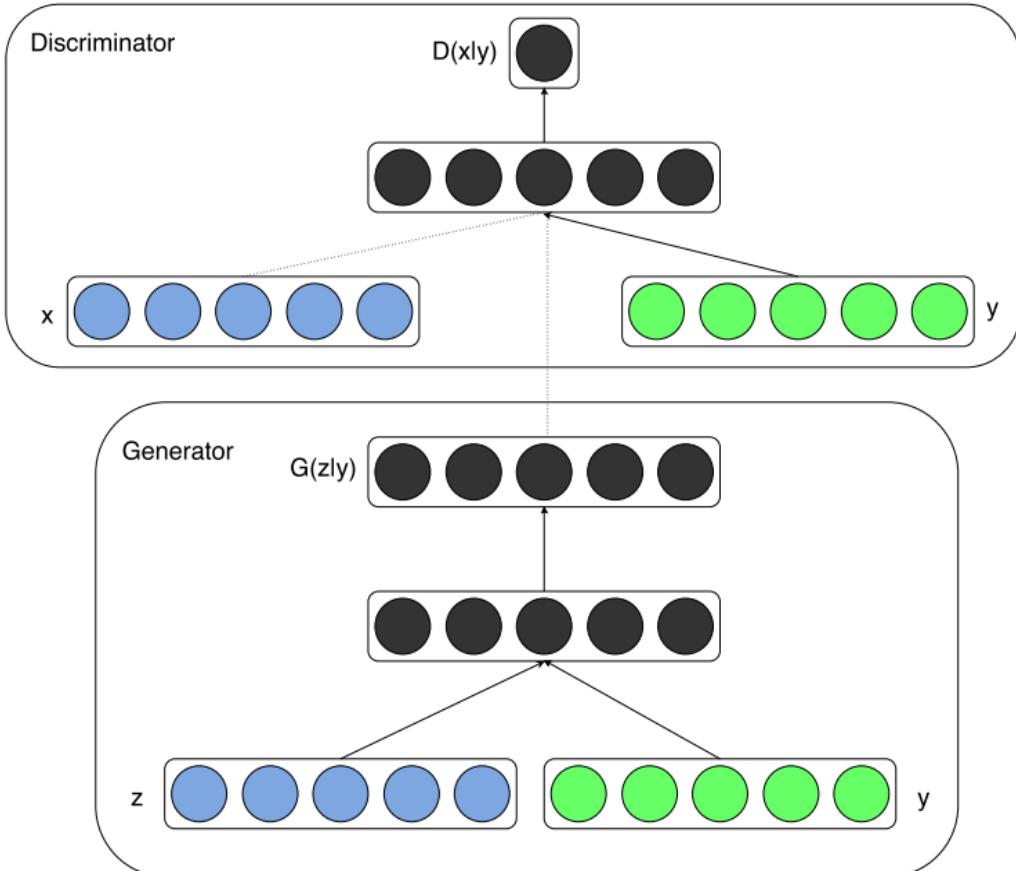


Figure 1: Conditional adversarial net

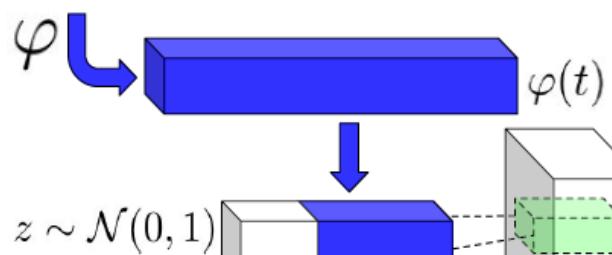
Fig 2 shows some of the generated samples. Each row is conditioned on one label and each column is a different generated sample.



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

# Text-conditional GAN (naïve)

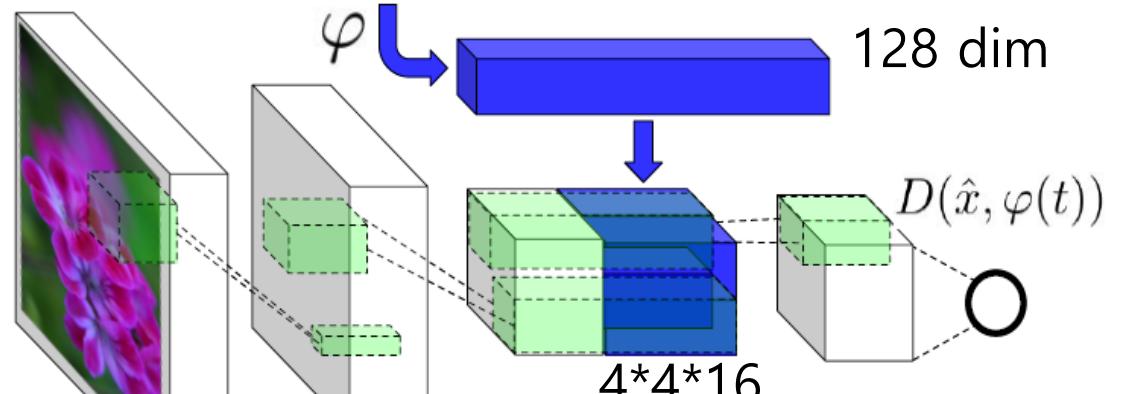
*This flower has small, round violet petals with a dark purple center*



Generator Network

$$\hat{x} := G(z, \varphi(t))$$

*This flower has small, round violet petals with a dark purple center*



Discriminator Network

Real image & matched text

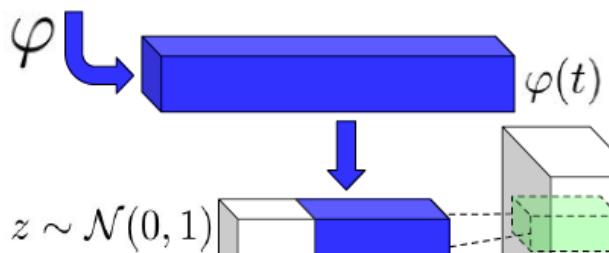
$$h = \psi(t)$$

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

Fake image & arbitrary text

# Matching-aware Discriminator

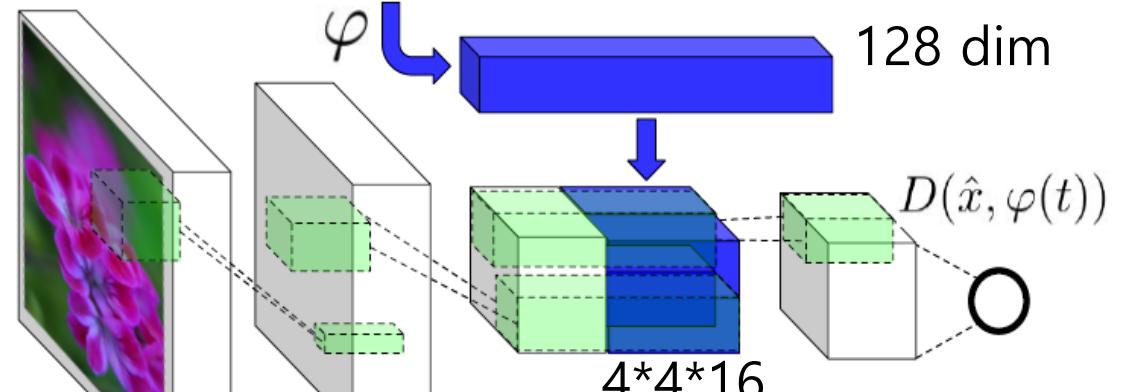
*This flower has small, round violet petals with a dark purple center*



Generator Network

$$\hat{x} := G(z, \varphi(t))$$

*This flower has small, round violet petals with a dark purple center*



Discriminator Network

Real image & matched text

$$h = \psi(t)$$

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

Real image & mismatched text

Fake image & matched text

# Matching-aware Discriminator

$$h = \psi(t)$$

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

Fake image & matched text

Real image & matched text  
 ↓  
 ↓  
 ↗  
 ↗  
 ↗

---

**Algorithm 1** GAN-CLS training algorithm with step size  $\alpha$ , using minibatch SGD for simplicity.

---

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

# Learning with Manifold Interpolation

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

Fake image & matched text

Real image & matched text

Real image & mismatched text

Additional term to generator to minimize:

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

# Experiments

**GT**  
an all black bird  
with a distinct  
thick, rounded bill.



this small bird has  
a yellow breast,  
brown crown, and  
black superciliary



a tiny bird, with a  
tiny beak, tarsus and  
feet, a blue crown,  
blue coverts, and  
black cheek patch



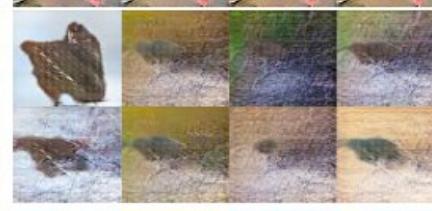
this bird is different  
shades of brown all  
over with white and  
black spots on its  
head and back



the gray bird has a  
light grey head and  
grey webbed feet



**GAN**



**GAN - CLS**



**GAN - INT**

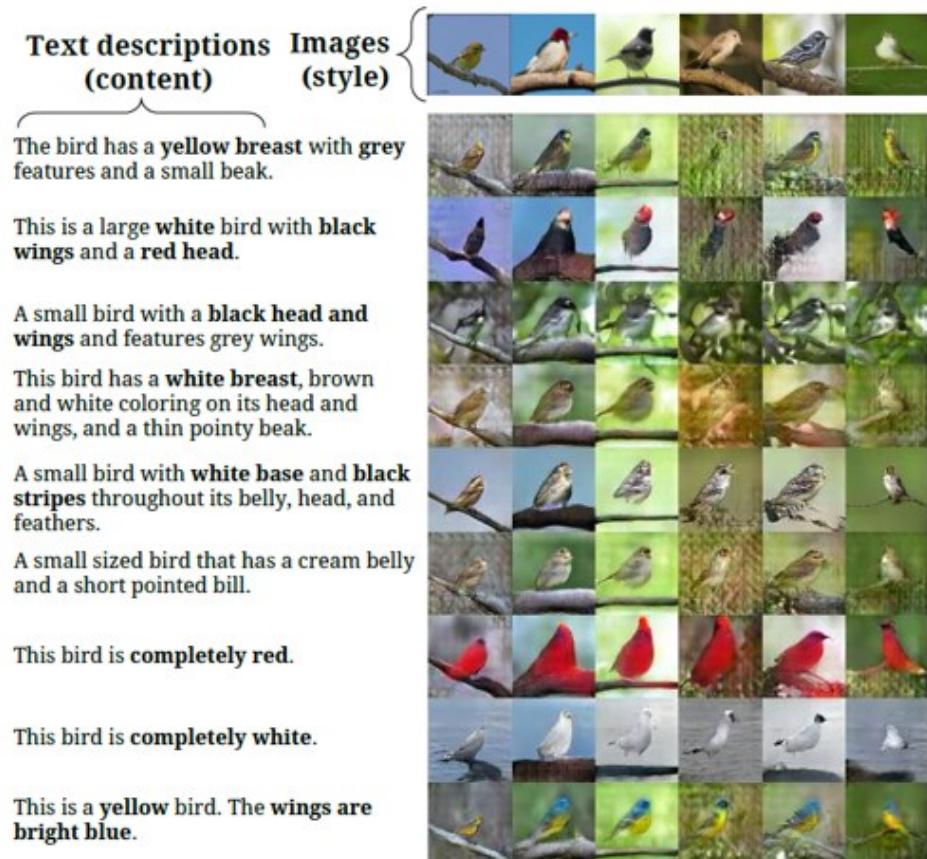


**GAN - INT -  
CLS**

# Experiments



# Style Transfer



Image



Style vector



Text description

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

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

# 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



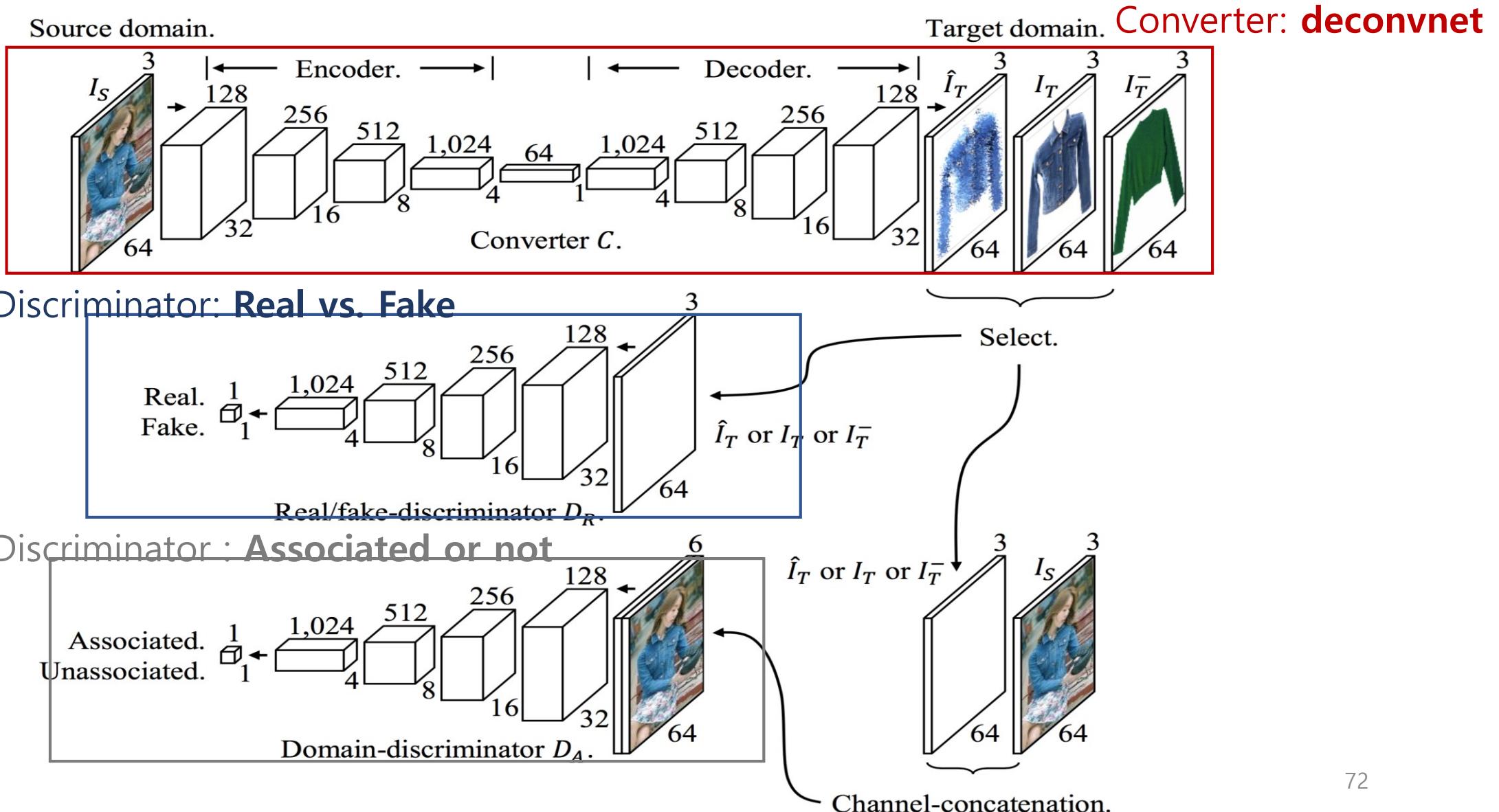
A source image.



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

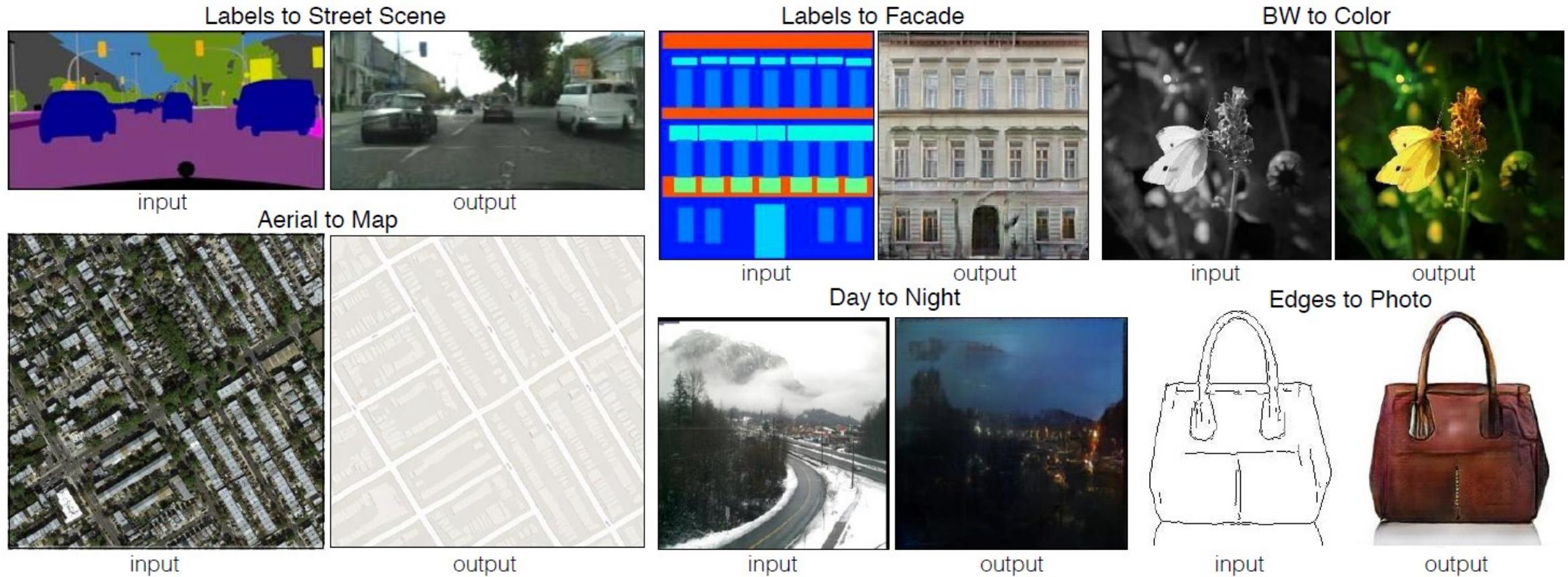


# Results – inverse setting



# **Image-to-Image Translation with Conditional Adversarial Networks**

# Pix2Pix



# Pix2Pix

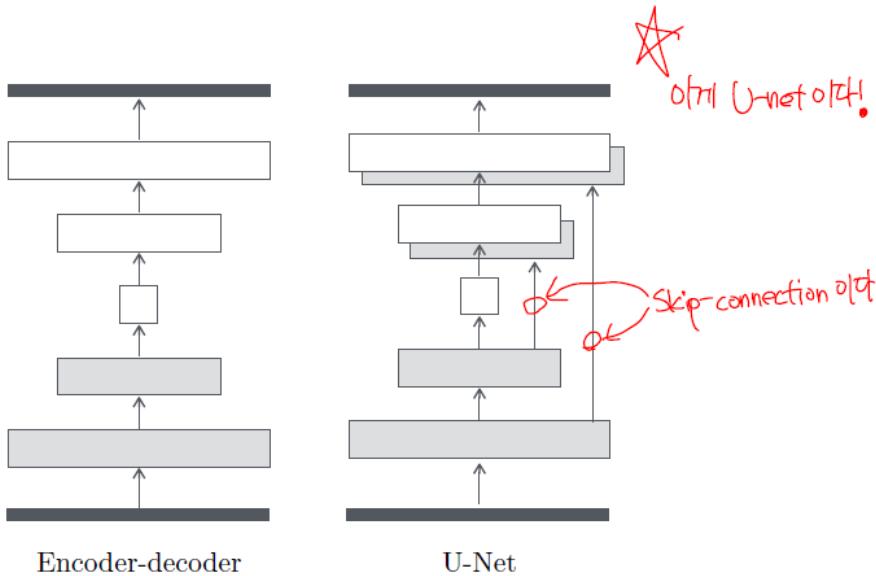


Figure 3: Two choices for the architecture of the generator. The ‘U-Net’ [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

+ L1 loss function

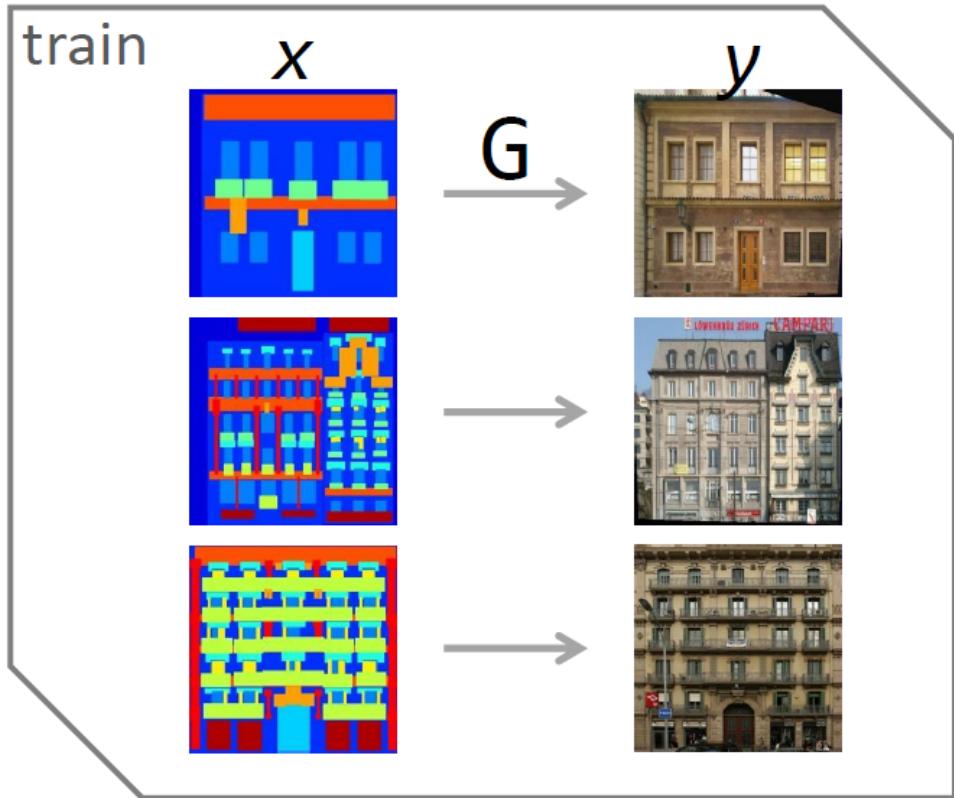
$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

Low-freq correctness

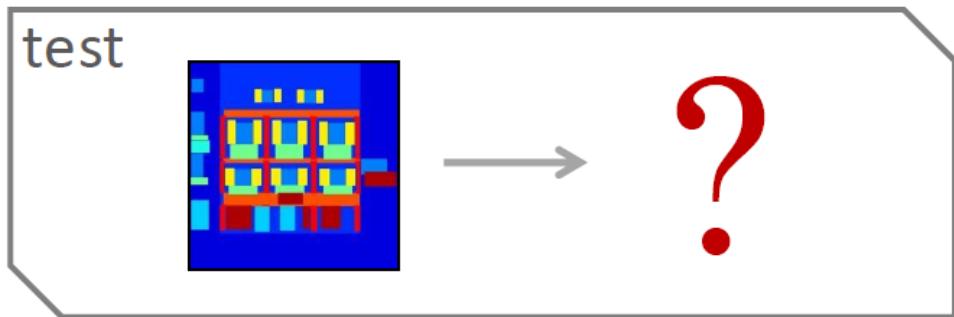
+ PatchGAN

High-freq correctness

# Pix2Pix



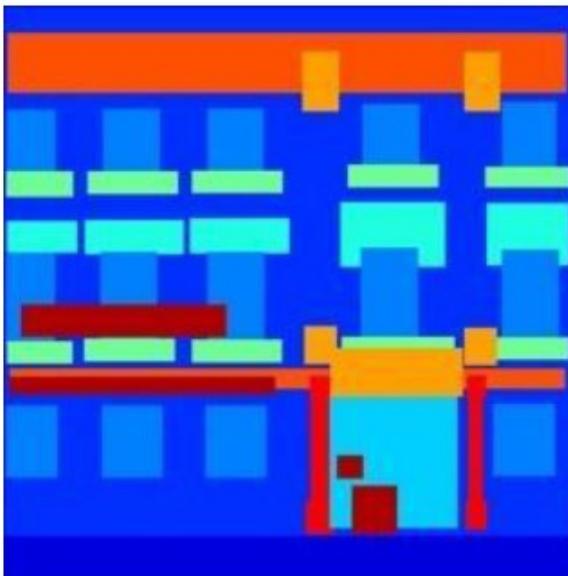
- Supervised
- loss: Minimize the difference between output  $G(x)$  and ground truth  $y$



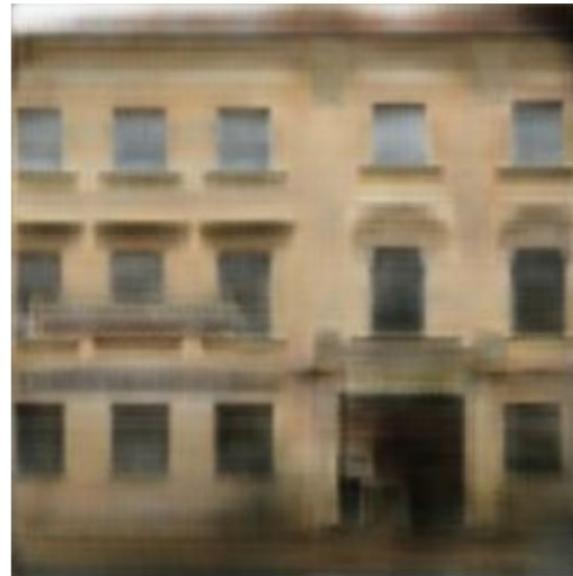
# L1 Loss – Stable Guide Force

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

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



Input



Output

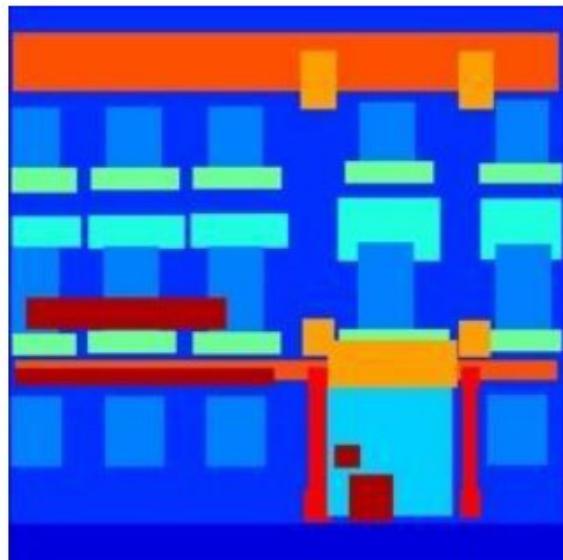


Ground Truth

# Adding GAN Loss

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

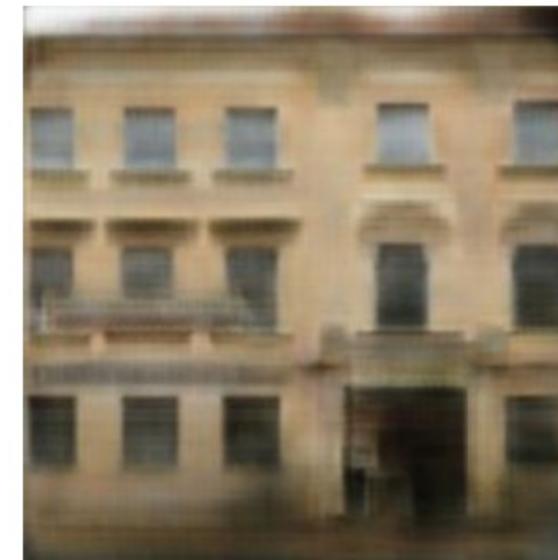
$$\sum_{(x,y)} \|y - G(x)\|_1 + L_{GAN}(G(x), y)$$



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.

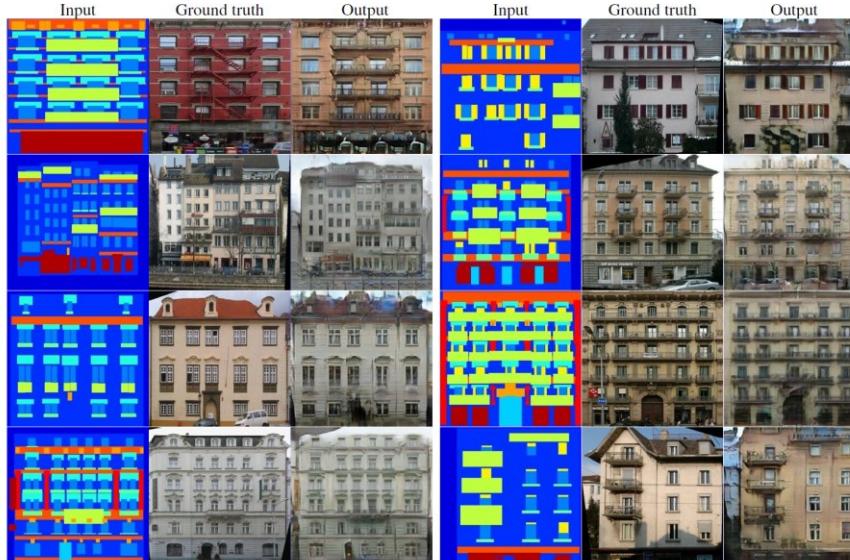


Figure 12: Example results of our method on facades labels→photo, compared to ground truth

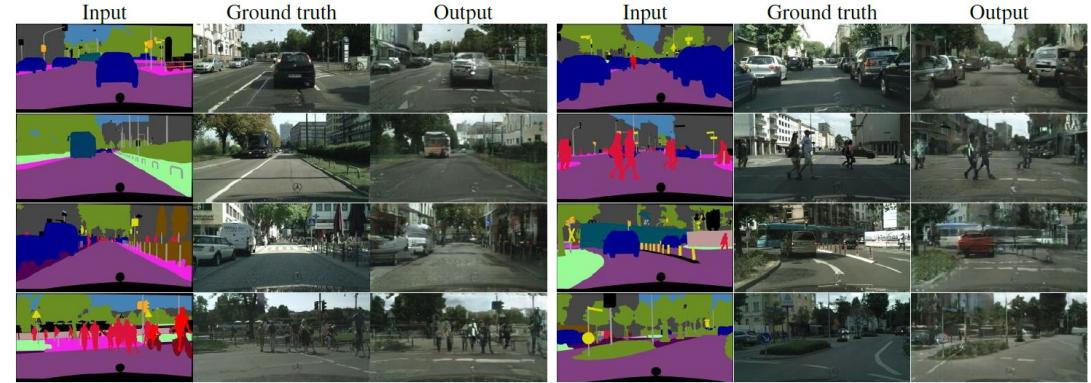


Figure 11: Example results of our method on Cityscapes labels→photo, compared to ground truth.

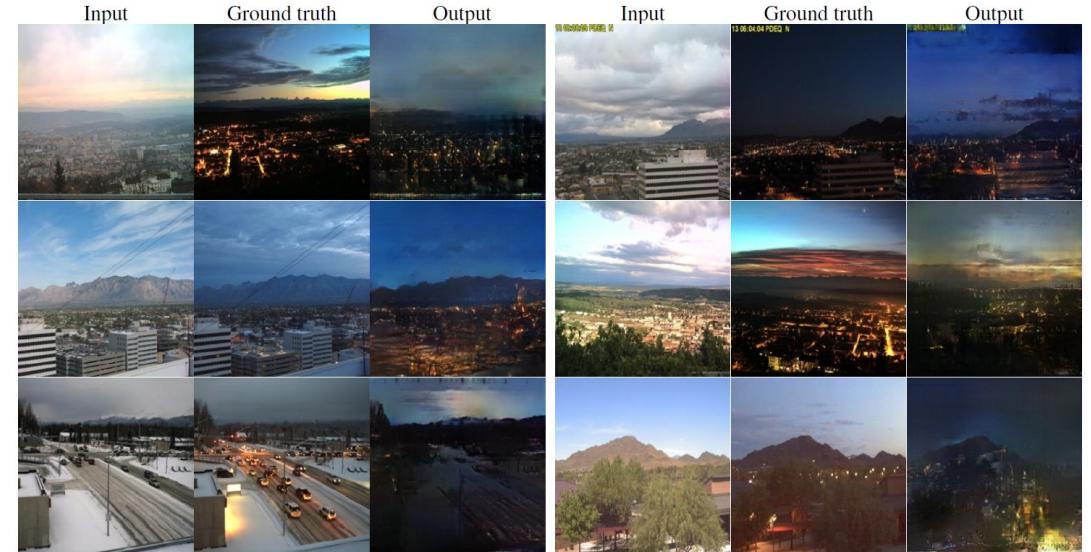


Figure 13: Example results of our method on day→night, compared to ground truth.

# 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.

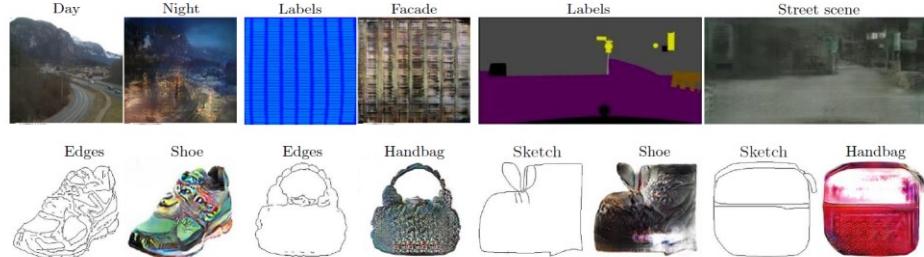


Figure 17: Example failure cases. Each pair of images shows input on the left and output on the right. These examples are selected as some of the worst results on our tasks. Common failures include artifacts in regions where the input image is sparse, and difficulty in handling unusual inputs. Please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

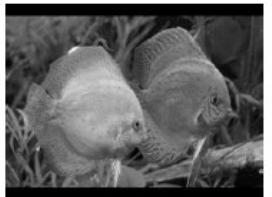
# Let's Try

- <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



?

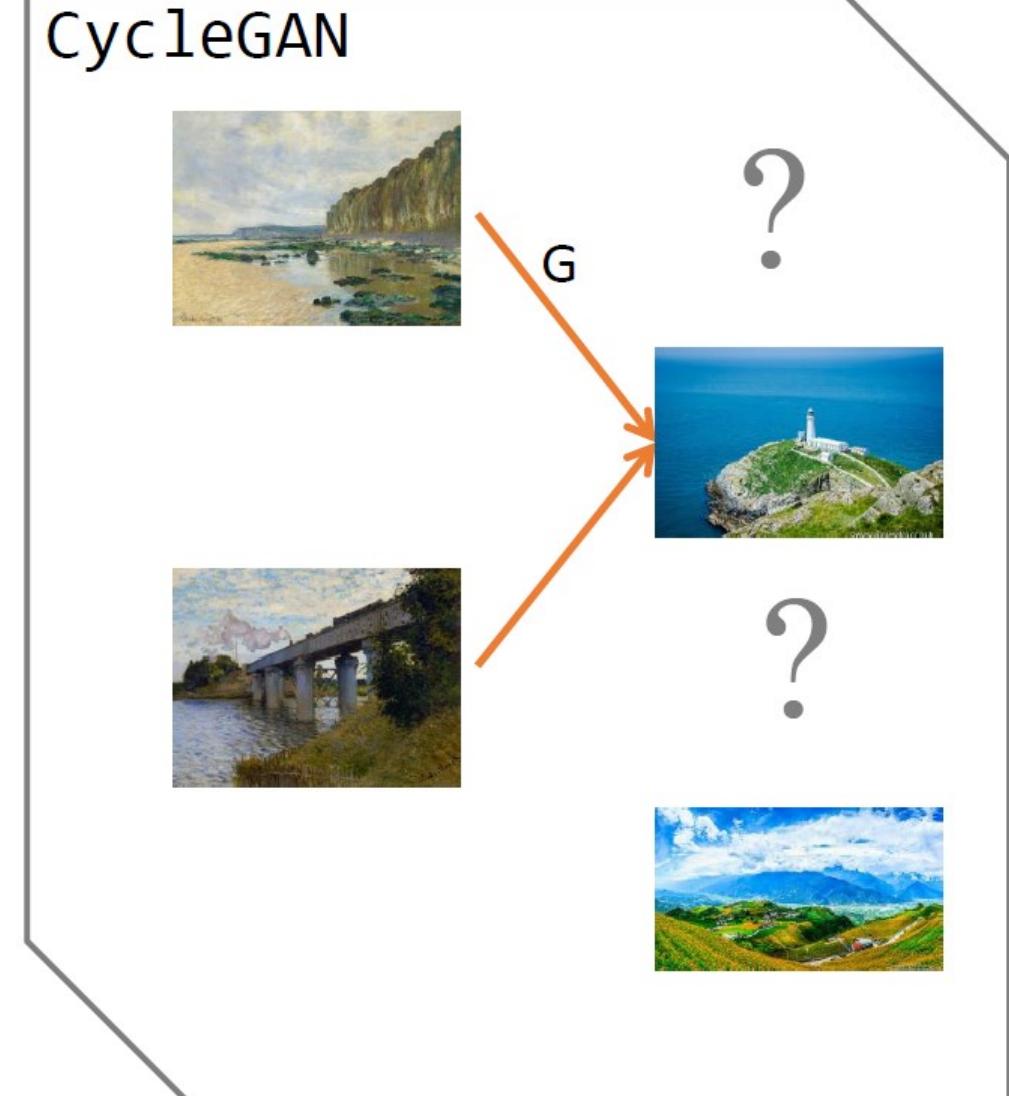


?



# CycleGAN – Mode Collapsing

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



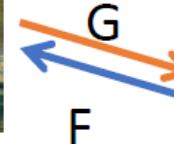
# CycleGAN

LOSS

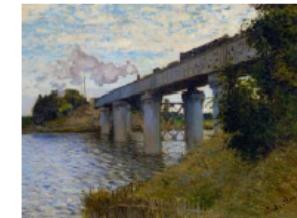
$$L_{GAN}(G(x), y) + \|F(G(x)) - x\|_1$$

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

CycleGAN



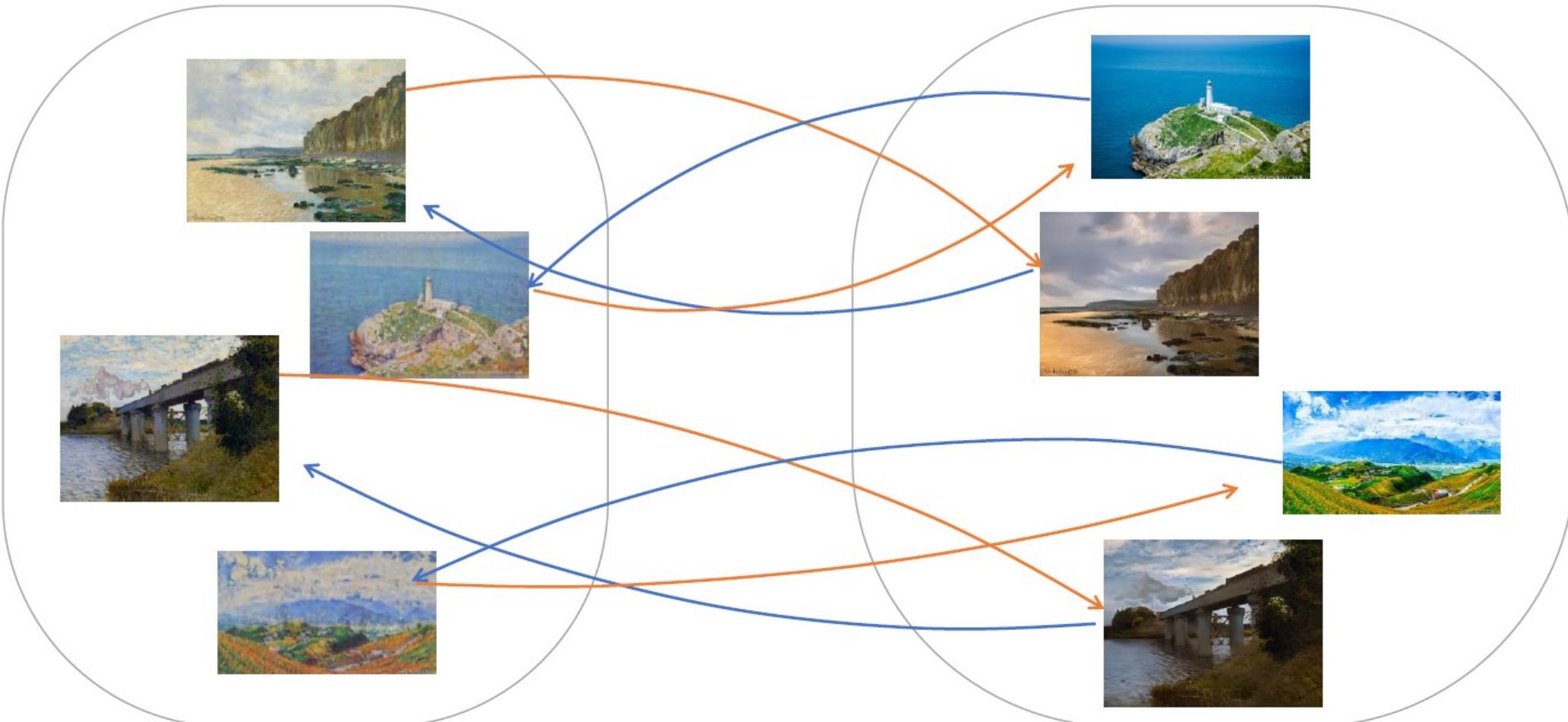
?



?

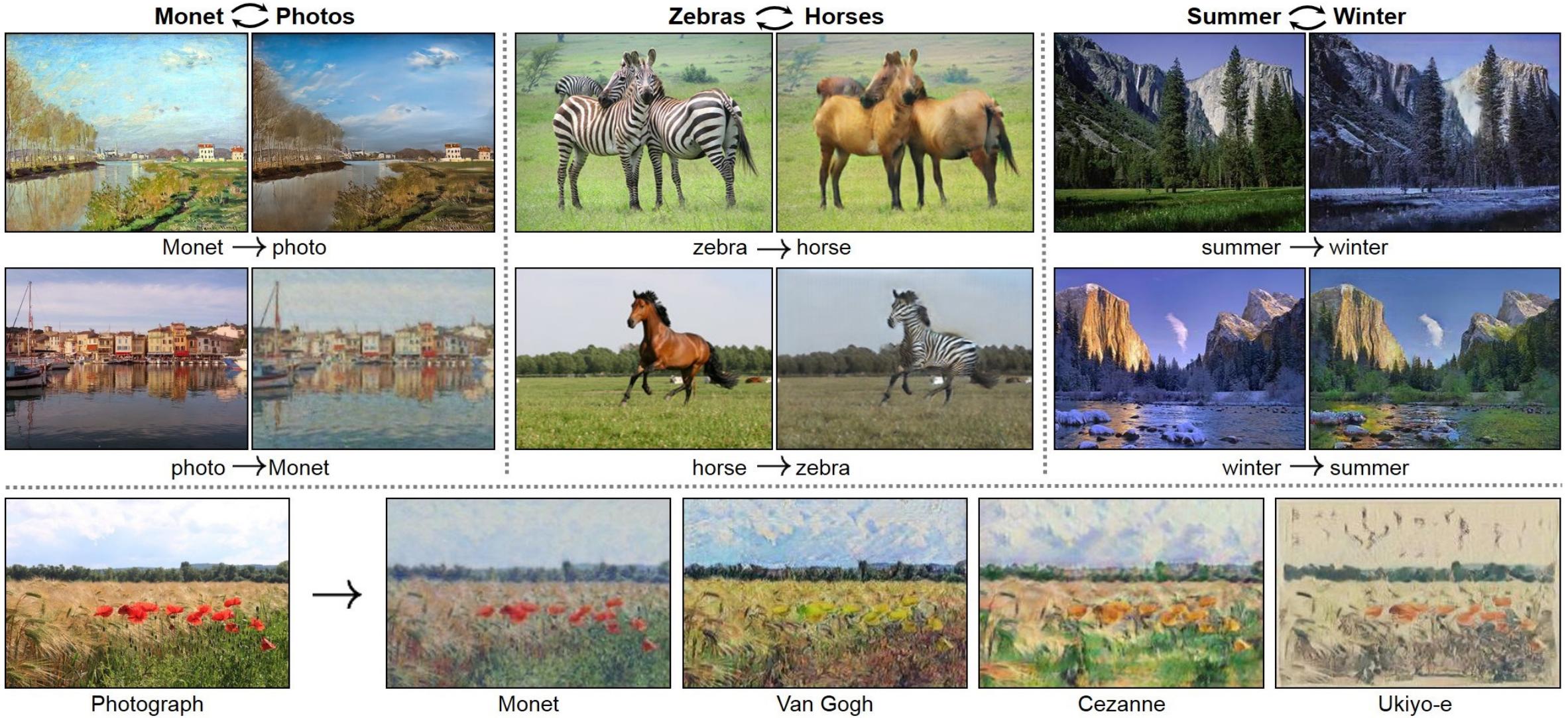


# CycleGAN – Loss Function



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

# Results



# Results



# Results



# Reconstructed Images

Original CG



Fake Photo



Reconstruction



Real Photo

