GANs for Image Processing & Computer Vision

Taeoh Kim

MVP Lab Seminar - Refined

Episode-4, 2017. 08. 23

Contents

References

GAN based Papers

In CVPR 2017 (Main)

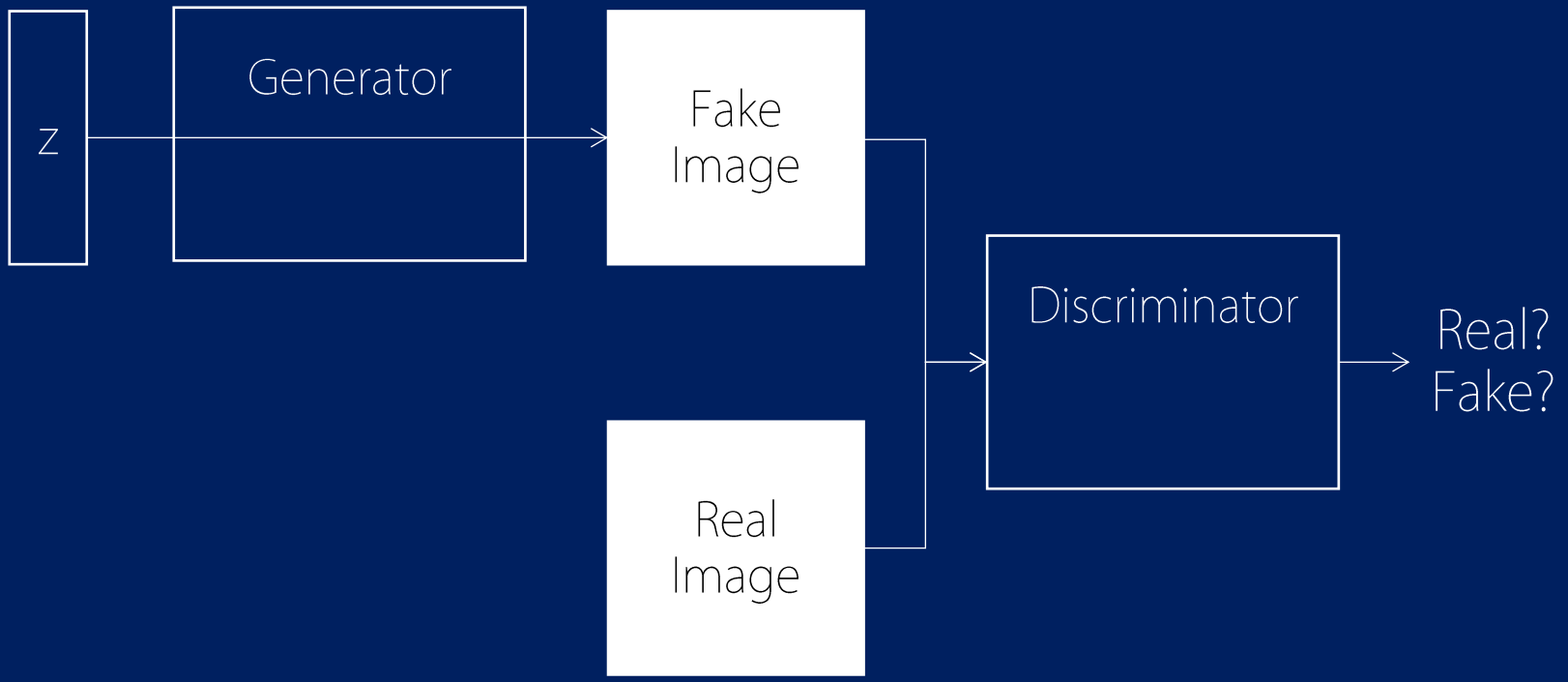
In ICML 2017

In ICCV 2017

Part 1.
GAN

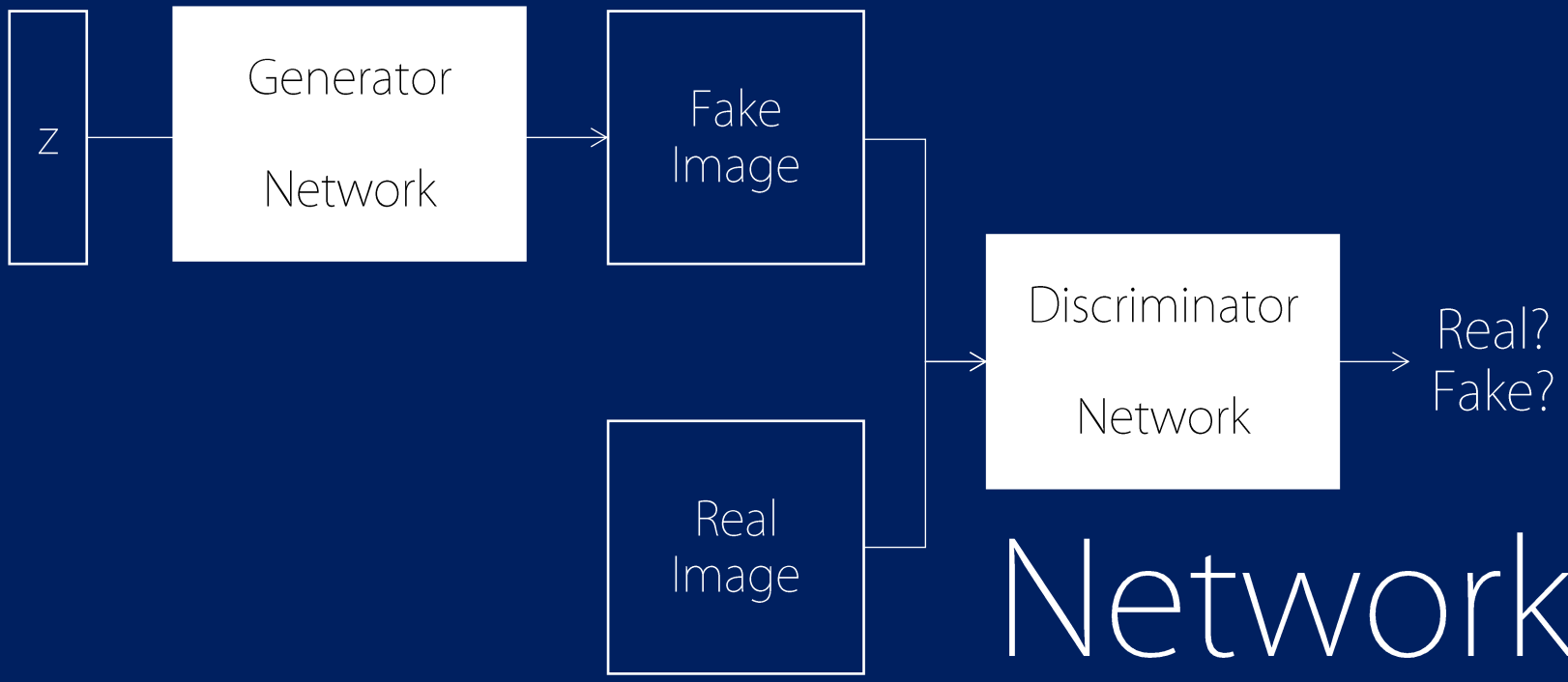
Generative

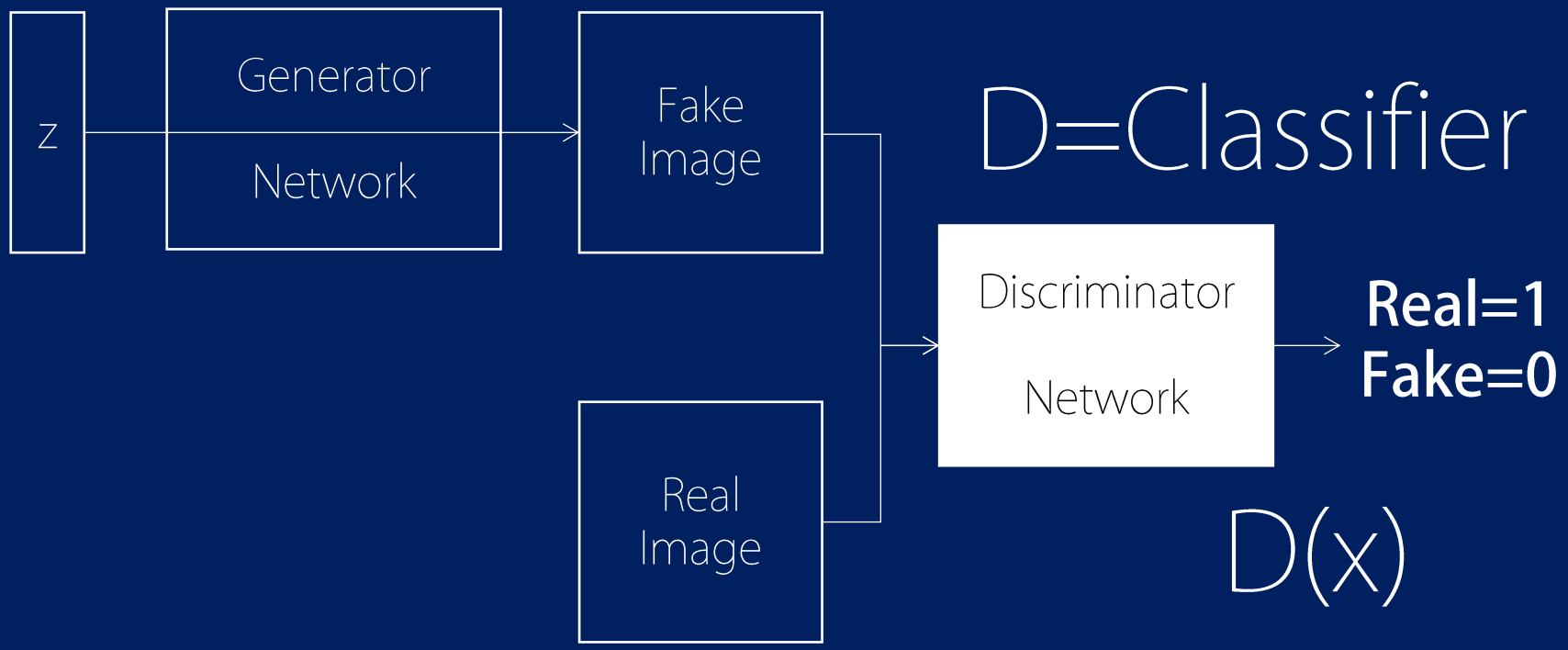
Generative

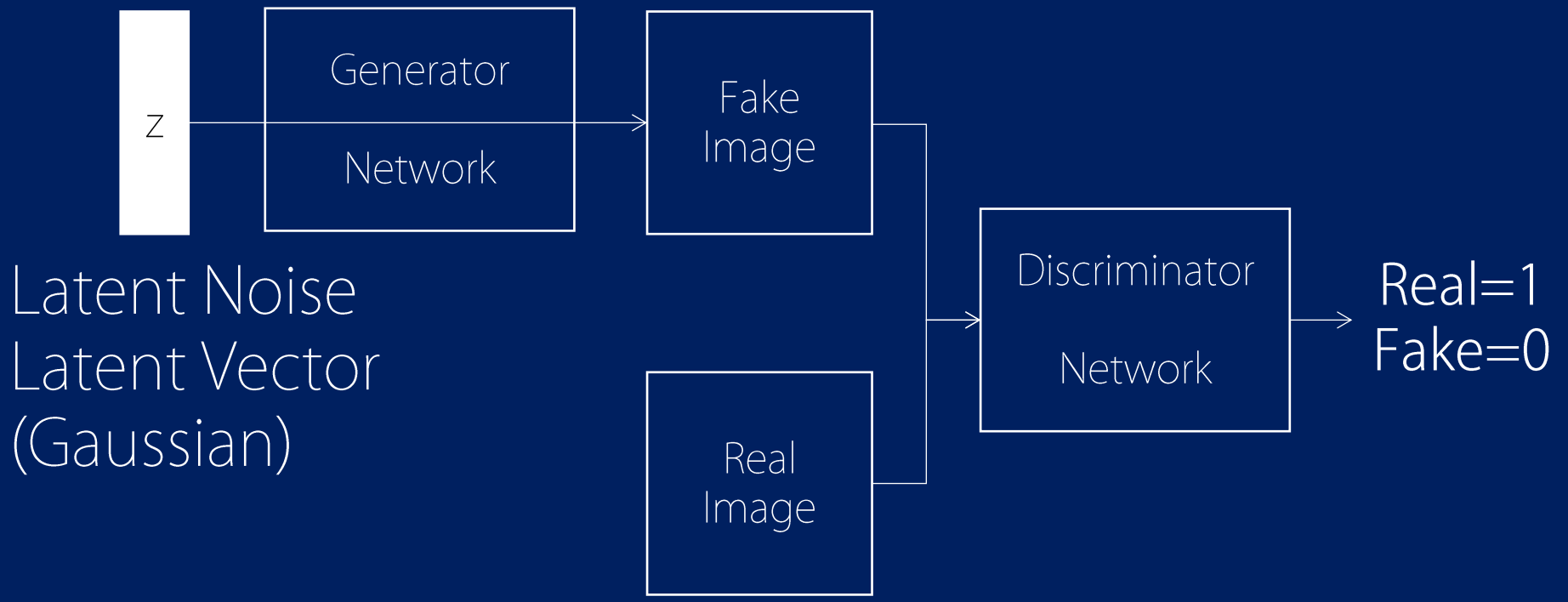


Adversarial

Generative Adversarial Networks



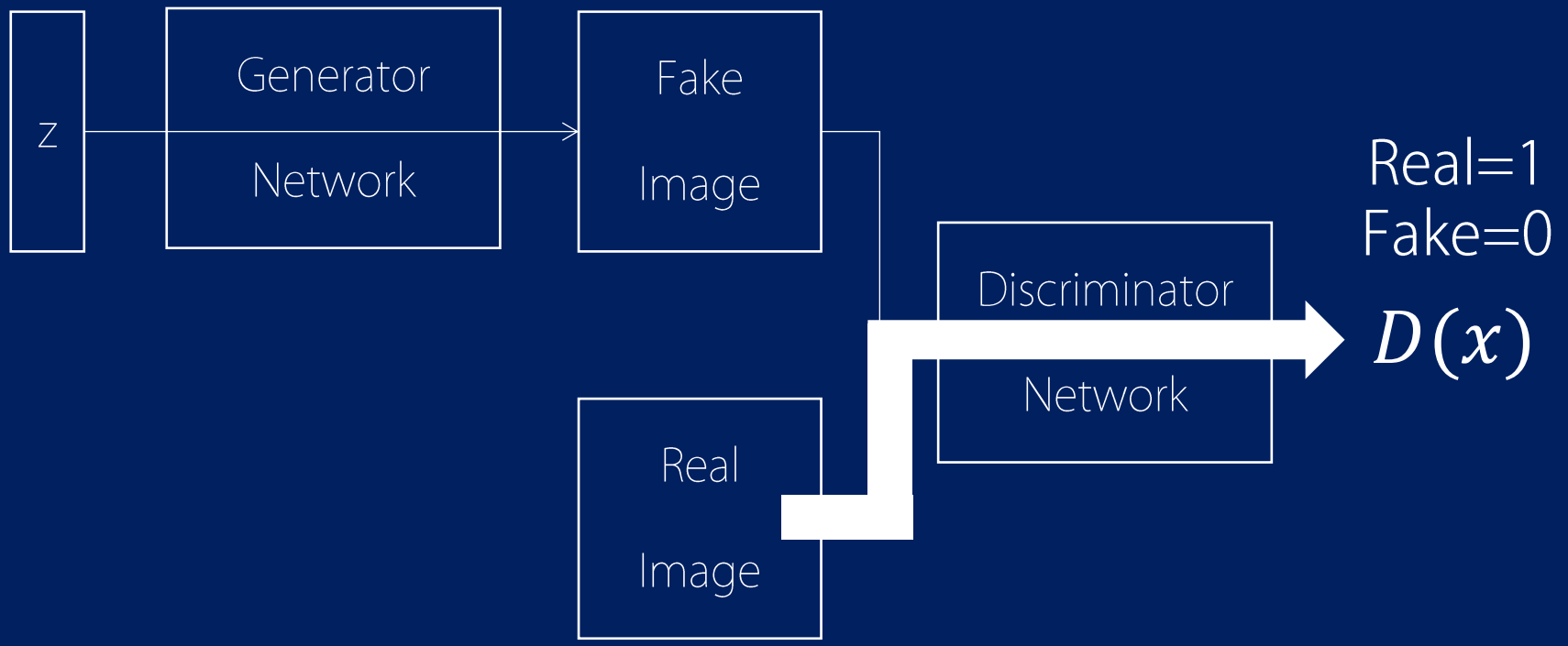

z
Generator
Network
Fake
Image
Real
Image
Discriminator
Network
D=Classifier
Real=1
Fake=0
D(x)


z
Generator
Network
Fake
Image
Real
Image
Discriminator
Network
Real=1
Fake=0
Latent Noise
Latent Vector
(Gaussian)

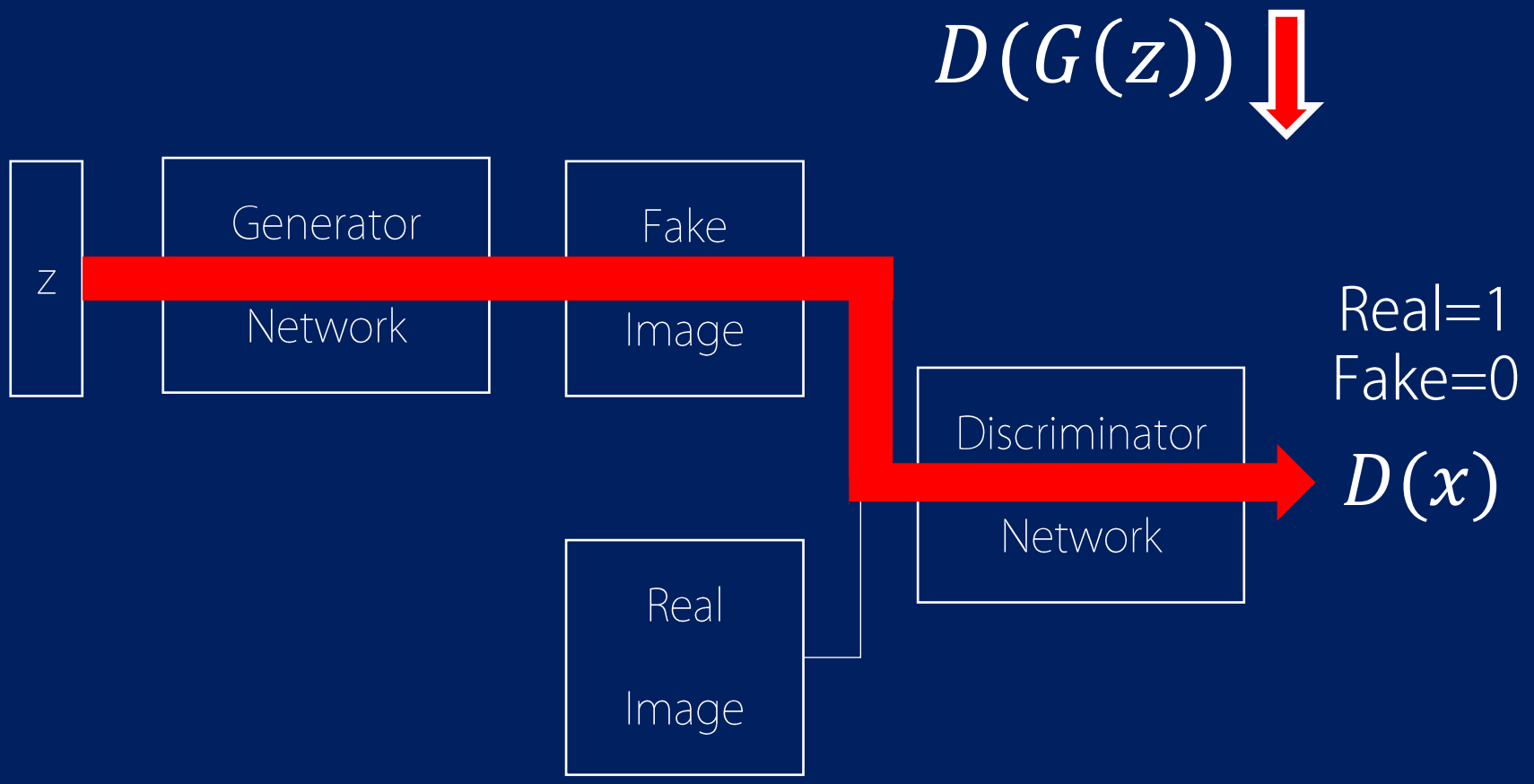
Training GAN

Training Discriminator

$D(x_{Real})$ ↑

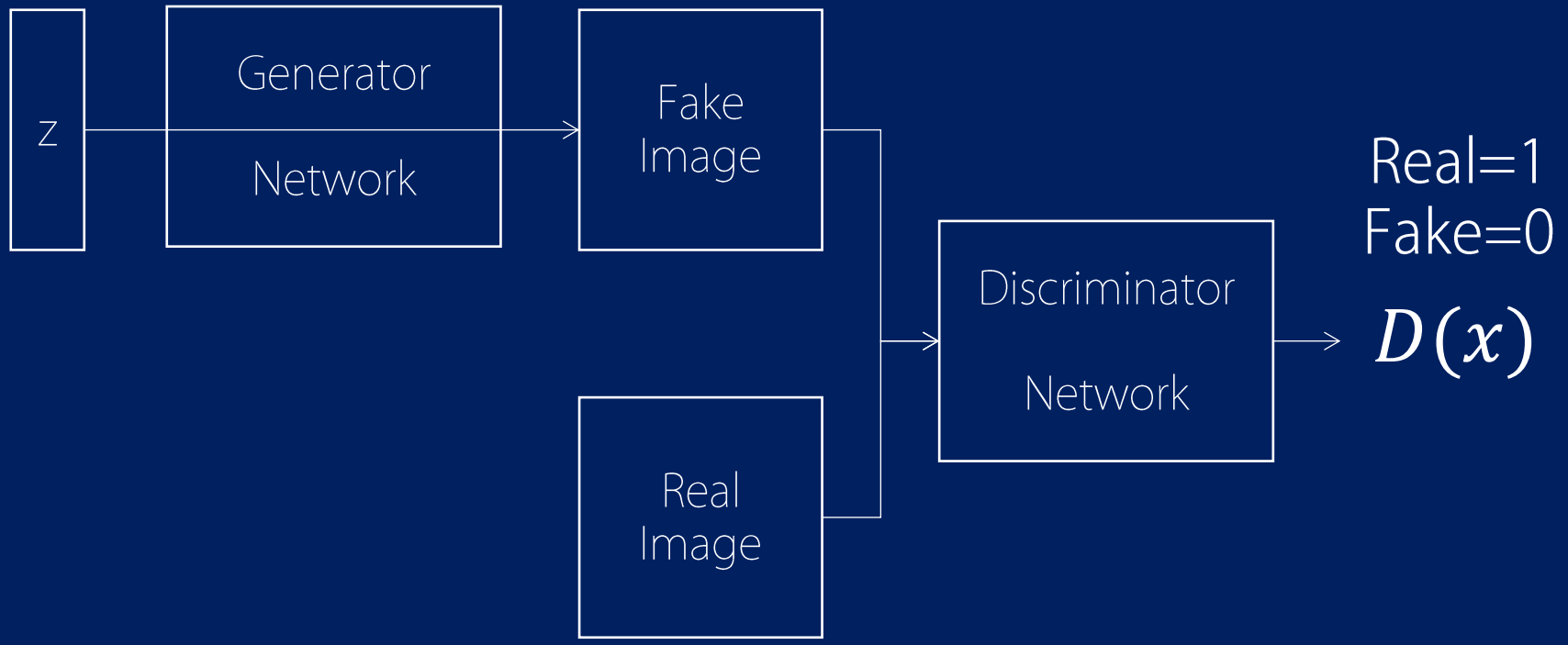


Training Discriminator



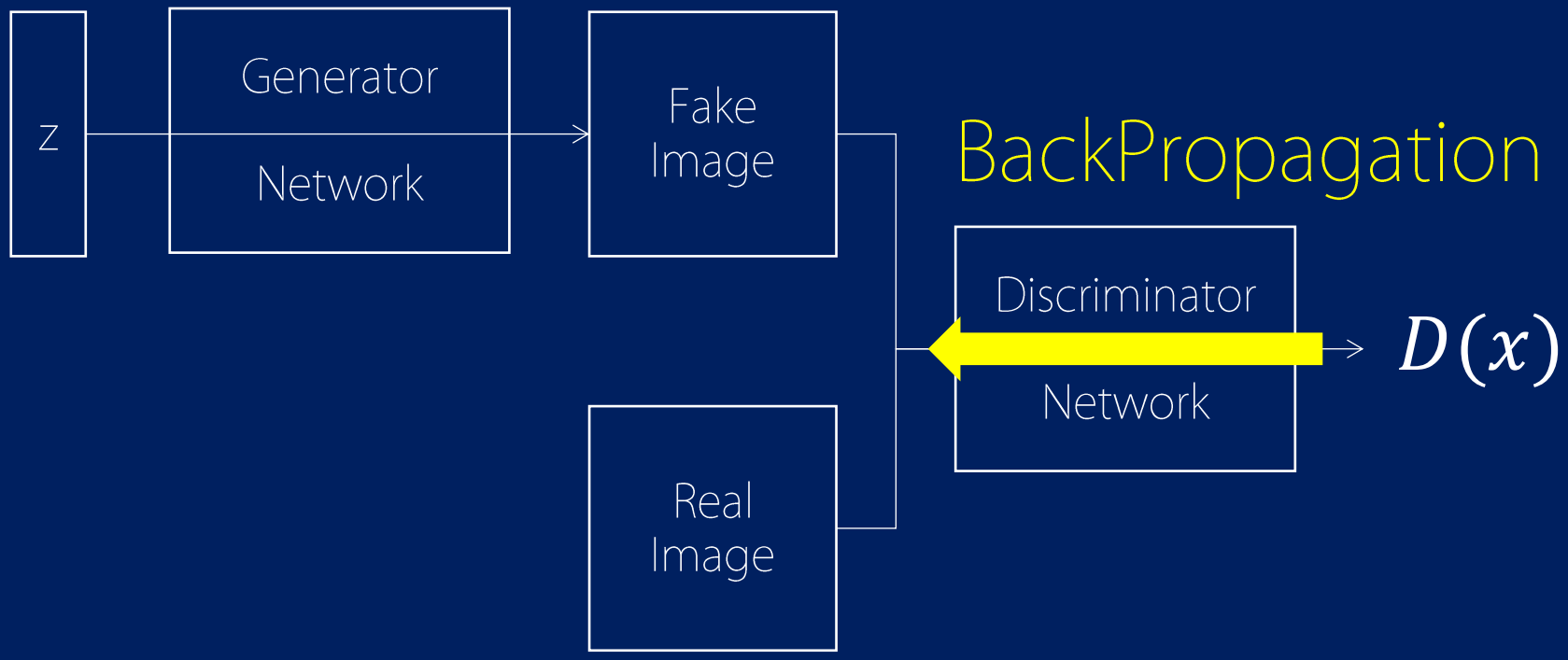
Training Discriminator

$D(x_{Real})$
$D(G(z))$



$$D_{Loss} = Error(D(x), 1) + Error(D(G(z)), 0)$$

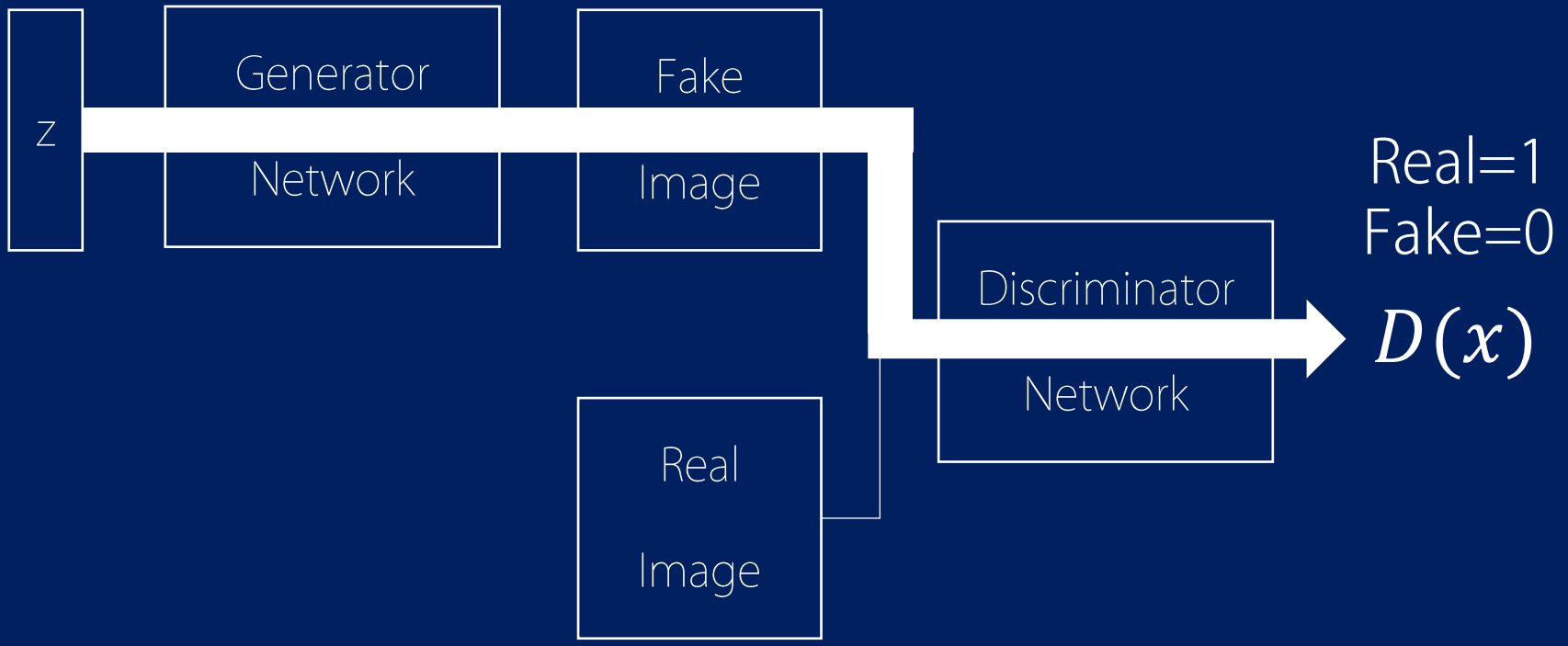
Training Discriminator



$$D_{Loss} = Error(D(x), 1) + Error(D(G(z)), 0)$$

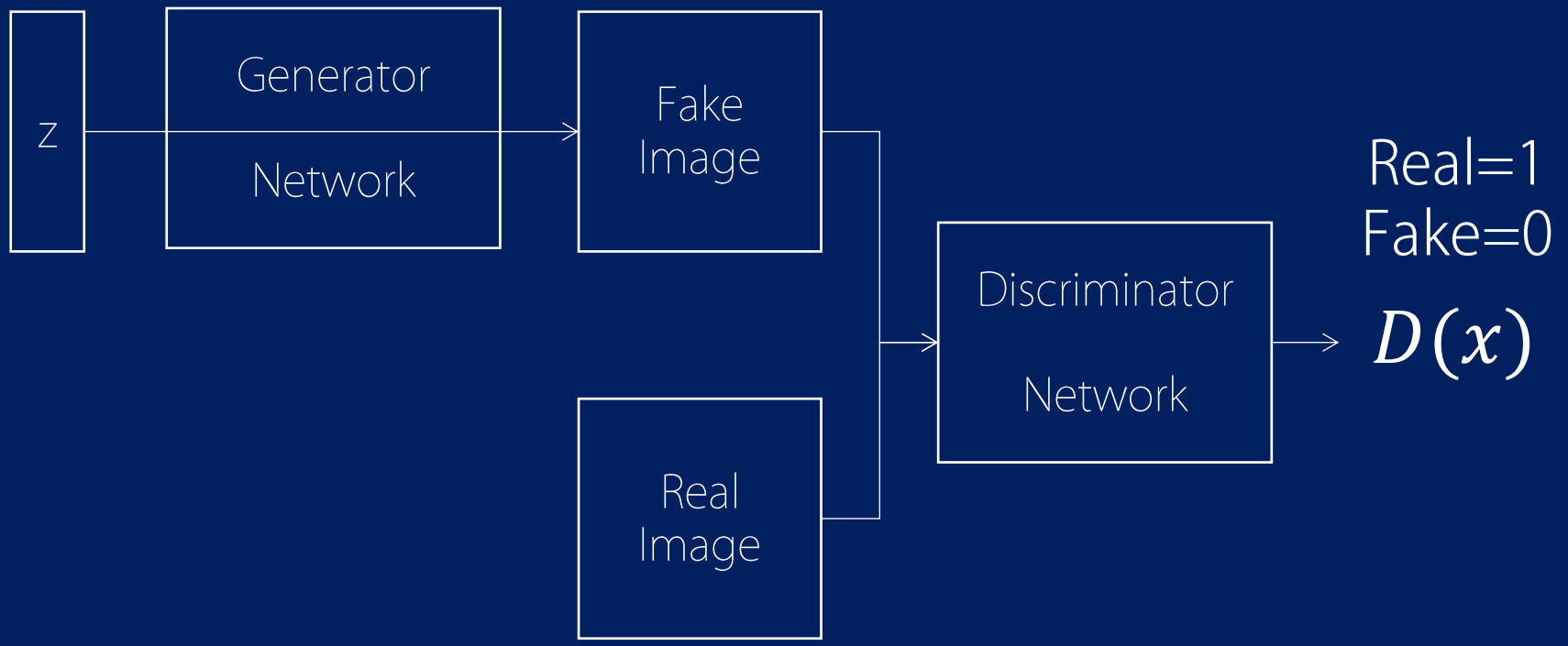
Training Generator

$D(G(z))$ ↑



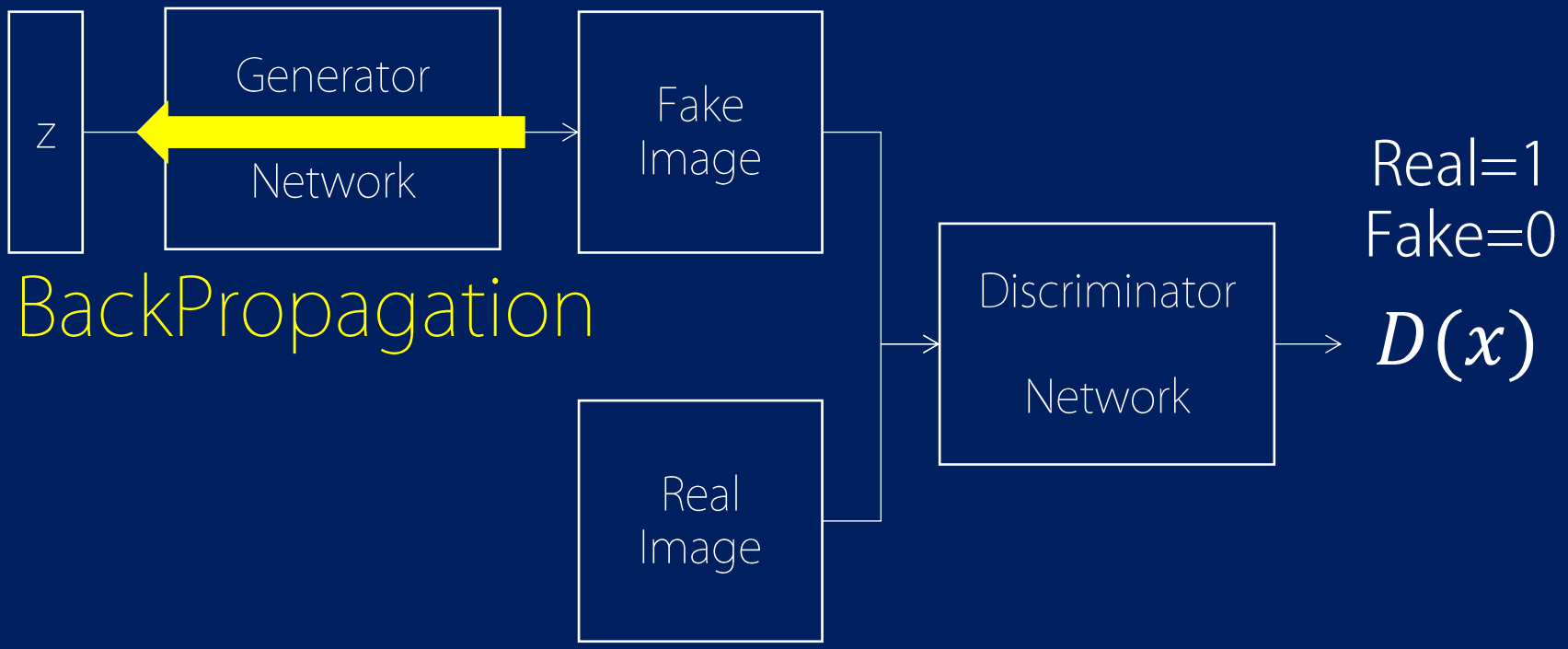
Training Generator

$D(G(z))$ ↑



$$G_{Loss} = Error(D(G(z)), 1)$$

Training Generator



$$G_{Loss} = Error(D(G(z)), 1)$$

Define Loss Function

Linear Regression Loss
: Mean Square Error (MSE)

Logistic Regression / Classification Loss
: Cross Entropy (CE)

$$-y_{true} \cdot log(y_{pred}) - (1 - y_{true}) \cdot \log(1 - y_{pred})$$

Define Loss Function

$$-y_{true} \cdot log(y_{pred}) - (1 - y_{true}) \cdot \log(1 - y_{pred})$$

$$D_{Loss} = Error(D(x), 1) + Error(D(G(z)), 0)$$
$$= -\log(D(x)) - \log(1 - D(G(z))$$

$$G_{Loss} = Error(D(G(z)), 1)$$
$$= -\log(D(G(z))$$

Define Loss Function

$$D_{Loss} = -\log\big(D(x)\big) - \log(1 - D(G(z))$$

$$G_{Loss} = -\log(D\big(G(z)\big))$$

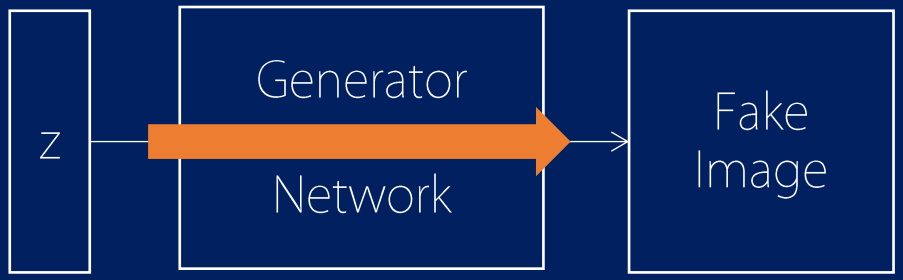
Value Function (Mathematical Form)

$$\min_{G}\max_{D} V(D,G) = E_{x\sim p_{data}} \log\big(D(x)\big) + E_{z\sim p_z} \log(1 - D(G(z))$$

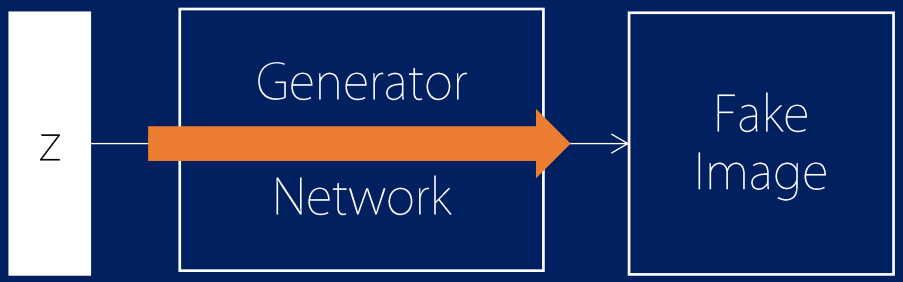
Min-Max Game
Nash Equilibrium

Testing GAN

Testing GAN
= Testing Generator = Generate From Noise



Testing During Training

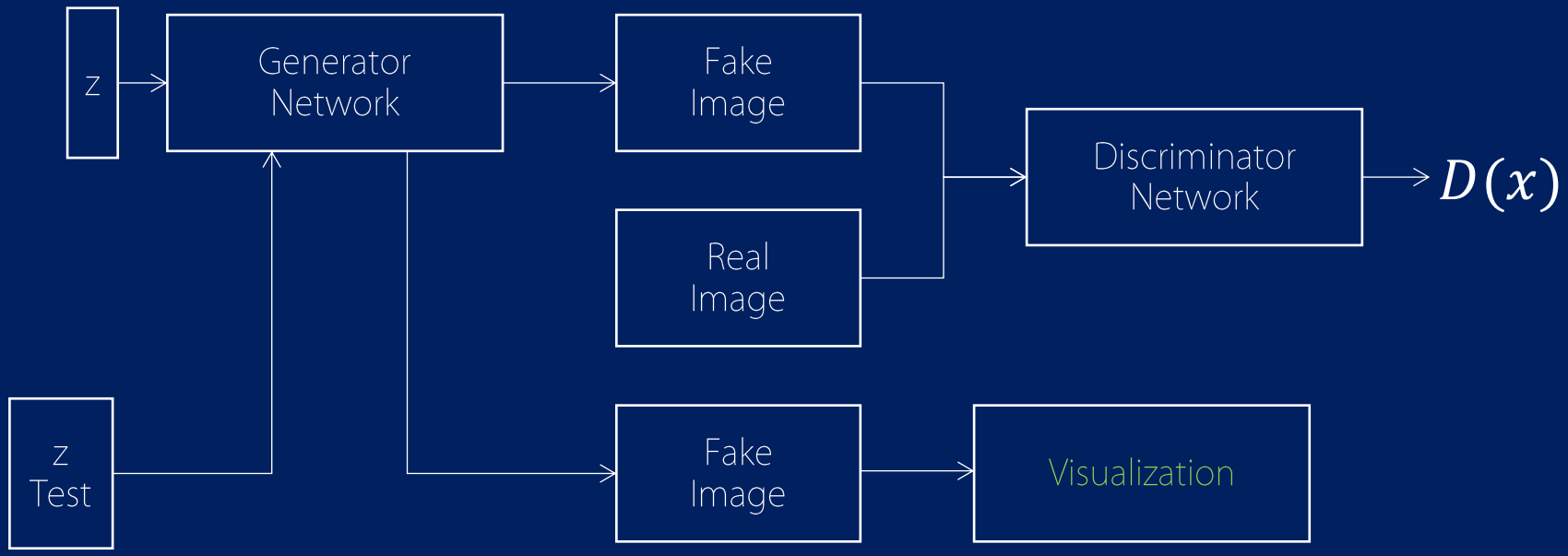


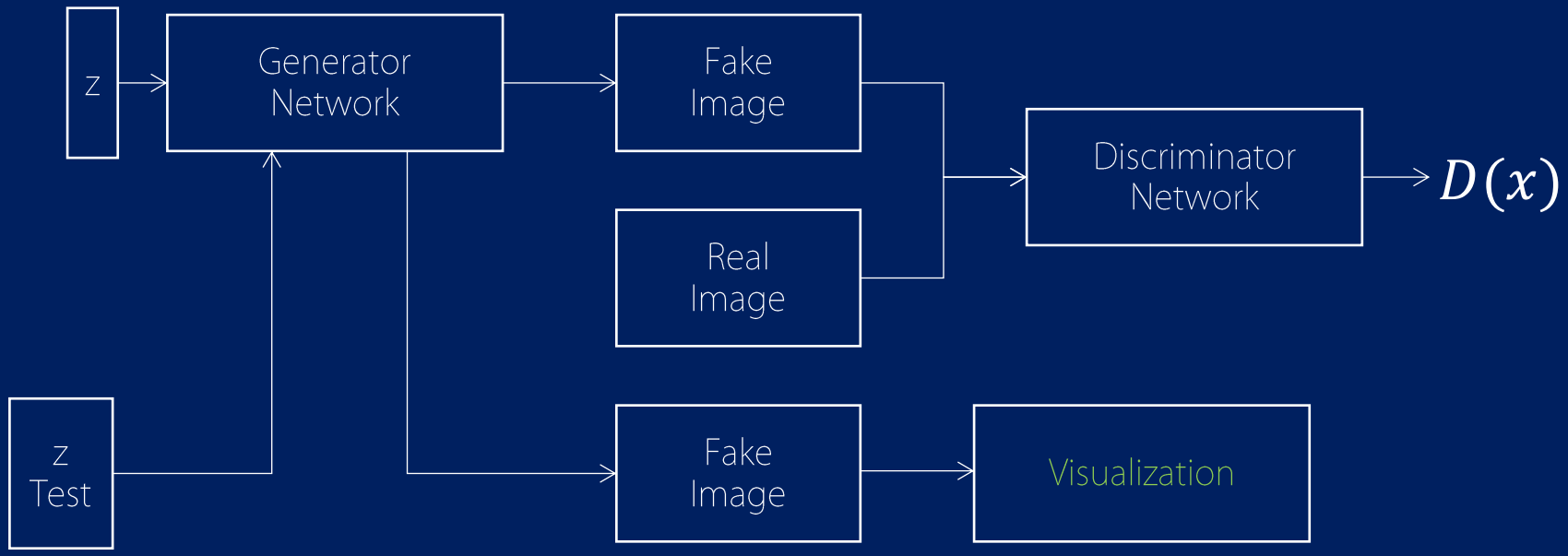
Fix Noise

GAN Implementation

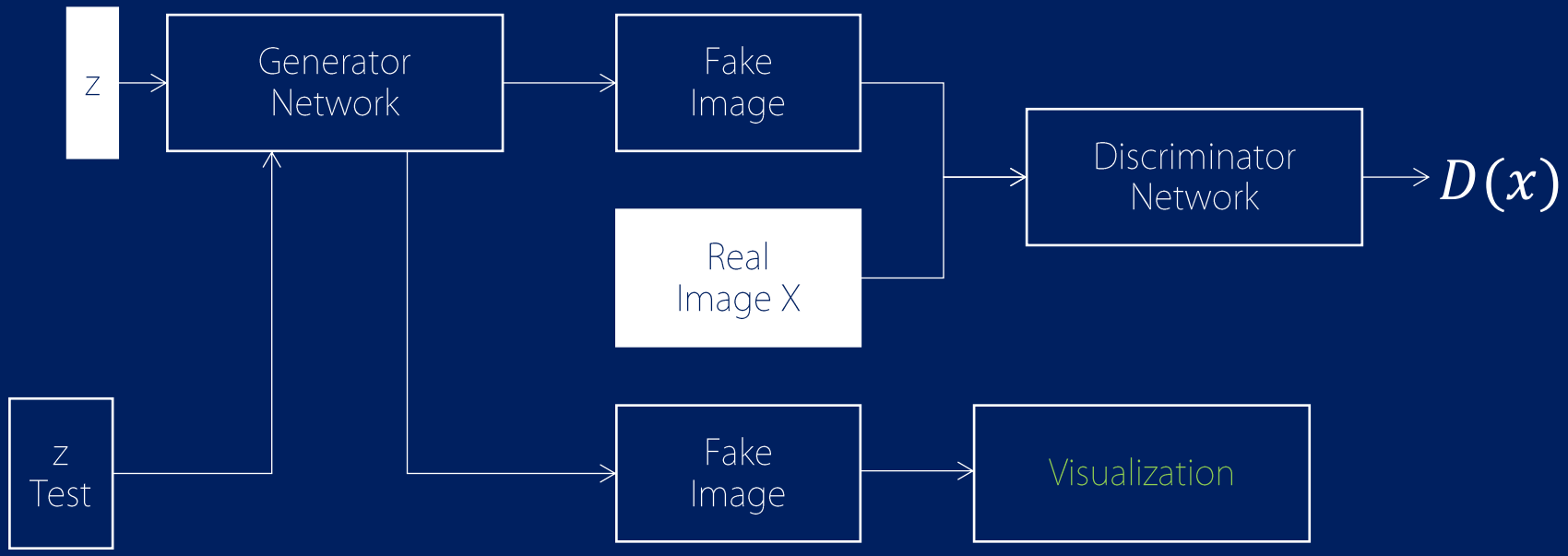
Tensorflow

50 Lines

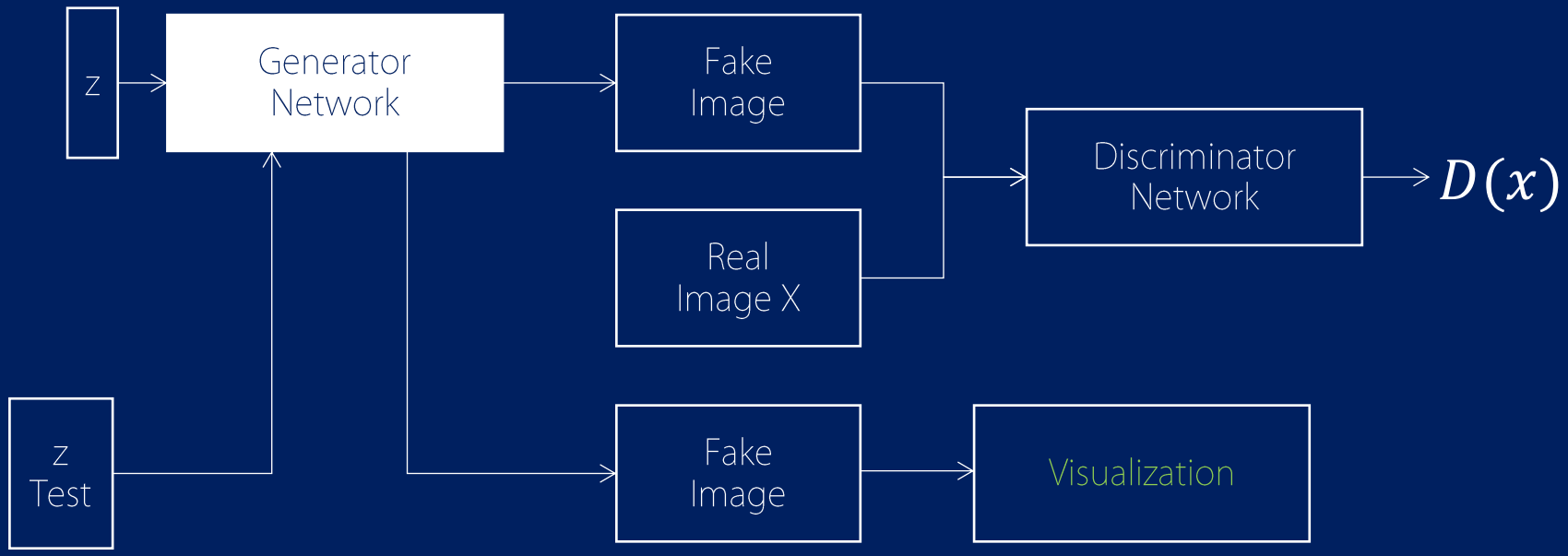

z
Generator Network
Fake Image
Real Image
Discriminator Network
$D(x)$
z Test
Fake Image
Visualization



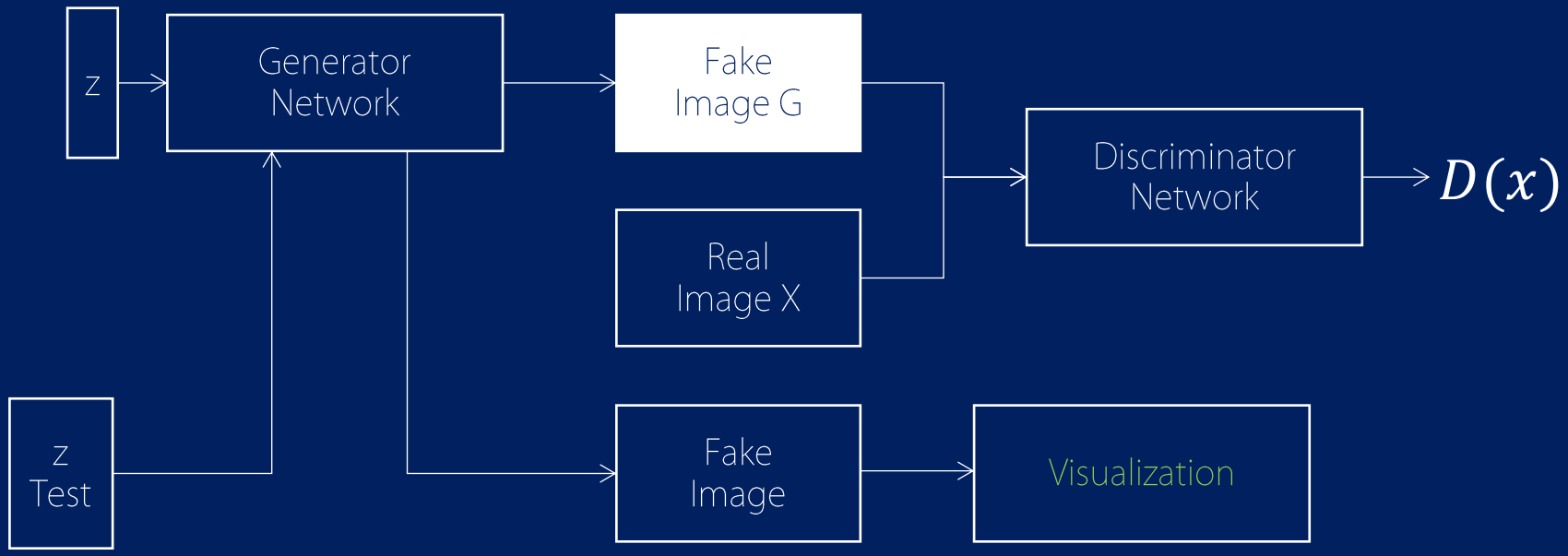

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)
```



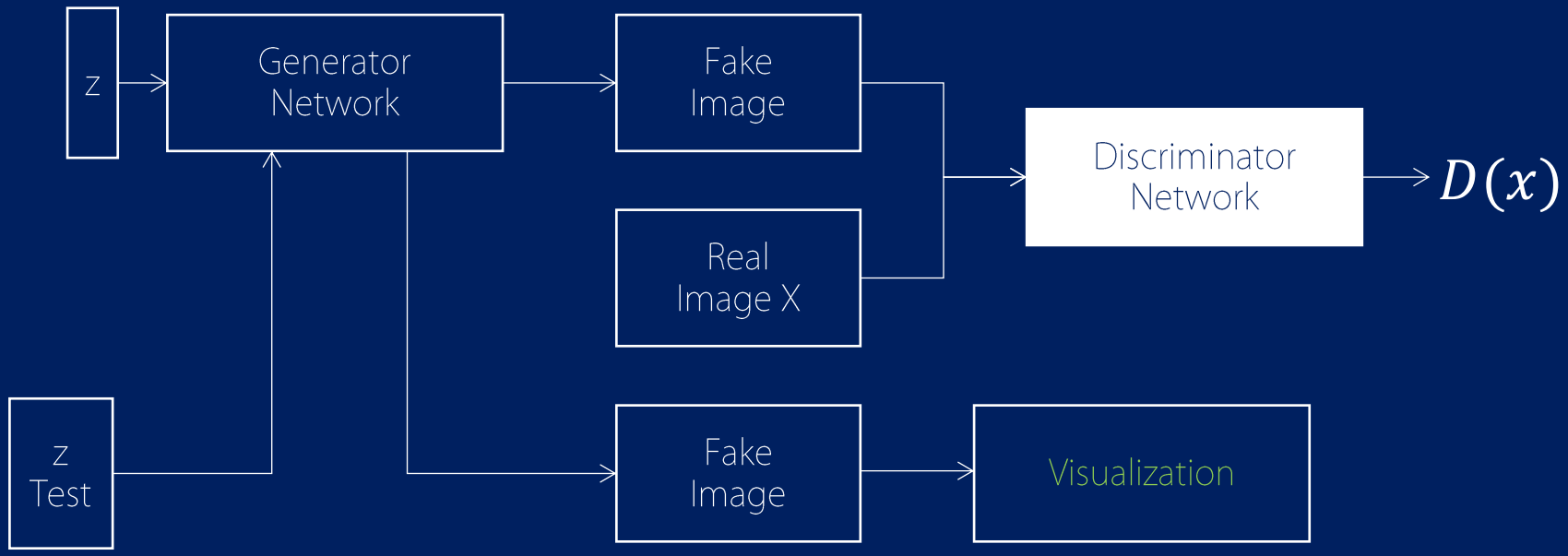

```
X = tf.placeholder(tf.float32, [None, 28 * 28])
Z = tf.placeholder(tf.float32, [None, 128])
```



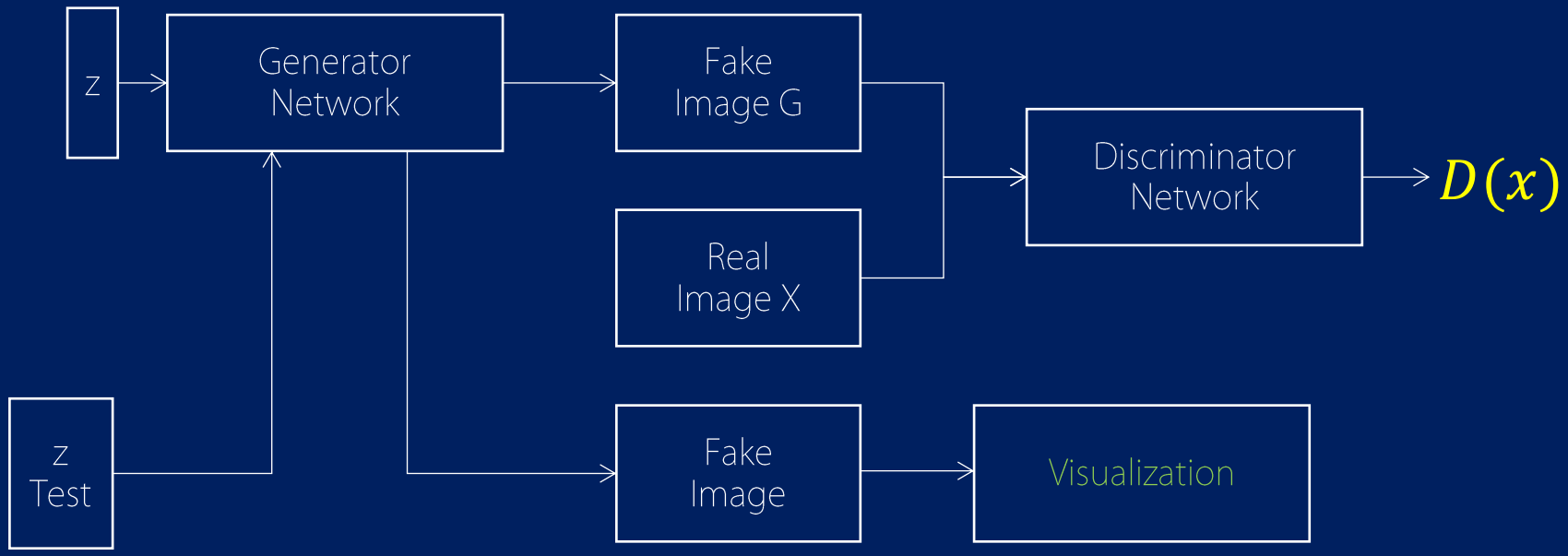

```
G_W1 = tf.Variable(tf.random_normal([128, 256], stddev=0.01))
G_W2 = tf.Variable(tf.random_normal([256, 28 * 28], stddev=0.01))
G_b1 = tf.Variable(tf.zeros([256]))
G_b2 = tf.Variable(tf.zeros([28 * 28]))

def generator(noise_z): # 128 -> 256 -> 28*28
        hidden = tf.nn.relu(tf.matmul(noise_z, G_W1) + G_b1)
        output = tf.nn.sigmoid(tf.matmul(hidden, G_W2) + G_b2)
        return output
```




```
G = generator(Z)

# Z->G
```

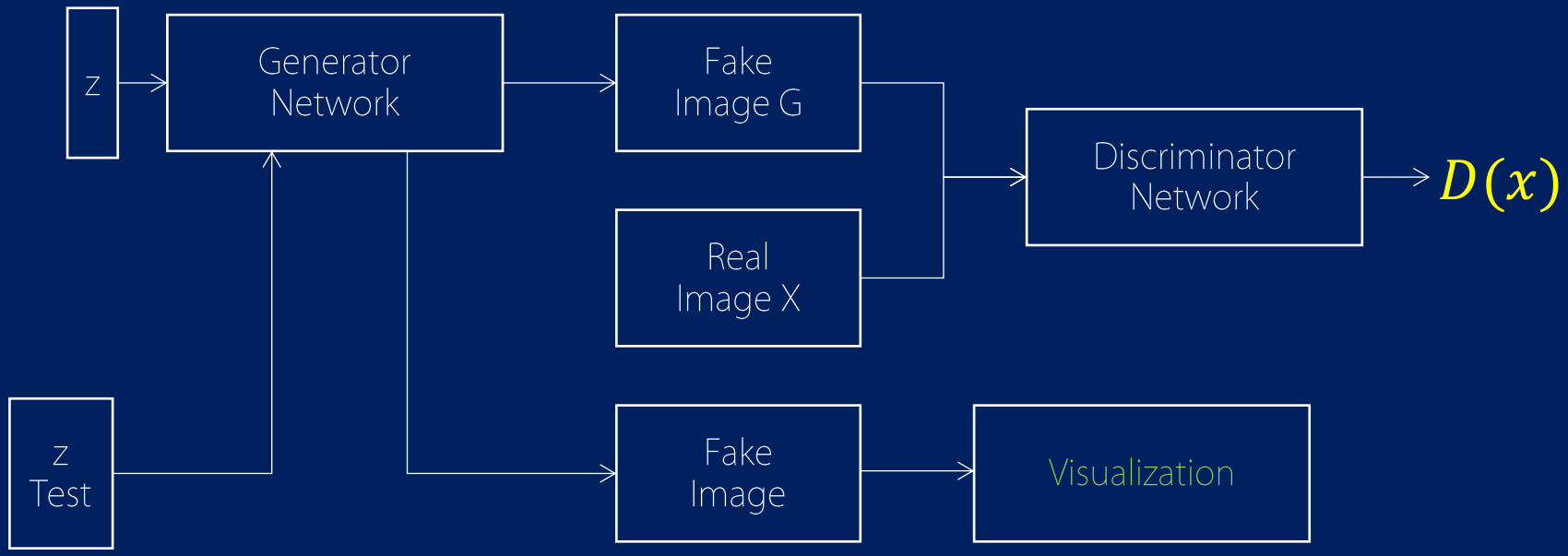



```
D_W1 = tf.Variable(tf.random_normal([28 * 28, 256], stddev=0.01))
D_W2 = tf.Variable(tf.random_normal([256, 1], stddev=0.01))
D_b1 = tf.Variable(tf.zeros([256]))
D_b2 = tf.Variable(tf.zeros([1]))

def discriminator(inputs): # 28*28 -> 256 -> 1
          hidden = tf.nn.relu(tf.matmul(inputs, D_W1) + D_b1)
          output = tf.nn.sigmoid(tf.matmul(hidden, D_W2) + D_b2)
          return output
```



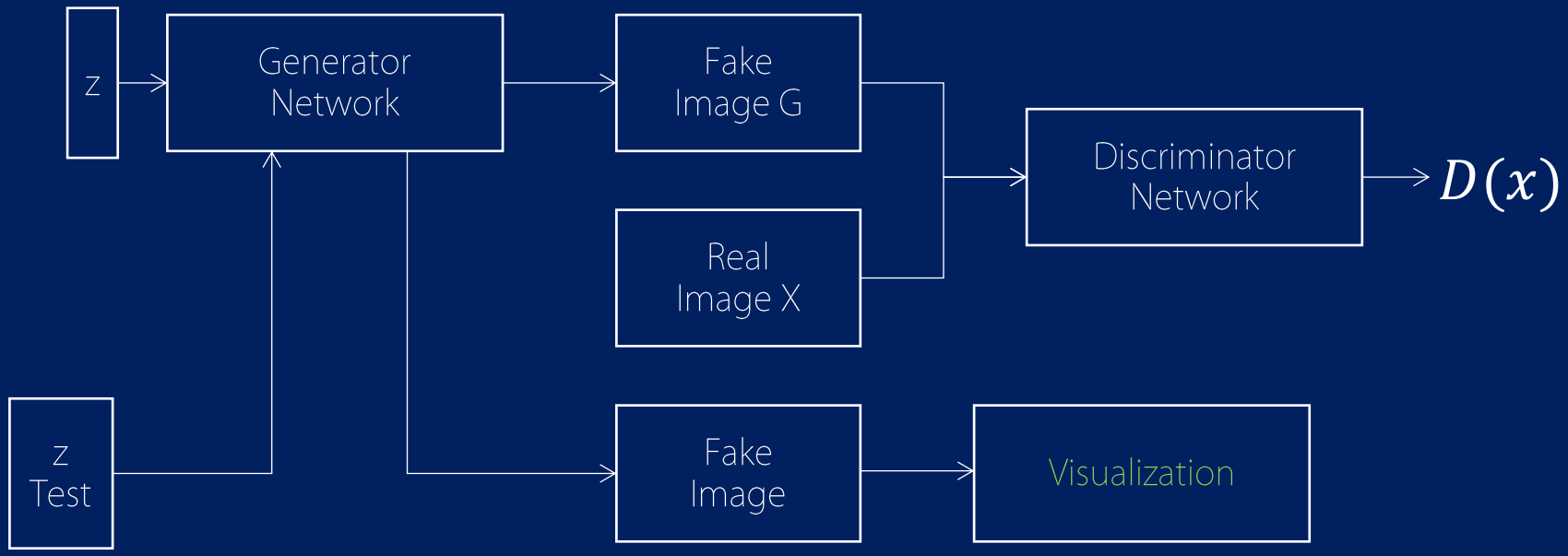

```
loss_D = -tf.reduce_mean(
        tf.log(discriminator(X)) + tf.log(1 - discriminator(G)))

train_D = tf.train.AdamOptimizer(0.0002).minimize(
        loss_D, var_list=[D_W1, D_b1, D_W2, D_b2])

# Z->G->loss_D->train_D
# X->loss_D->train_D
```

$$D_{Loss} = -\log\big(D(x)\big) - \log(1 - D(G(z))$$
$$G_{Loss} = -\log(D(G(z))$$

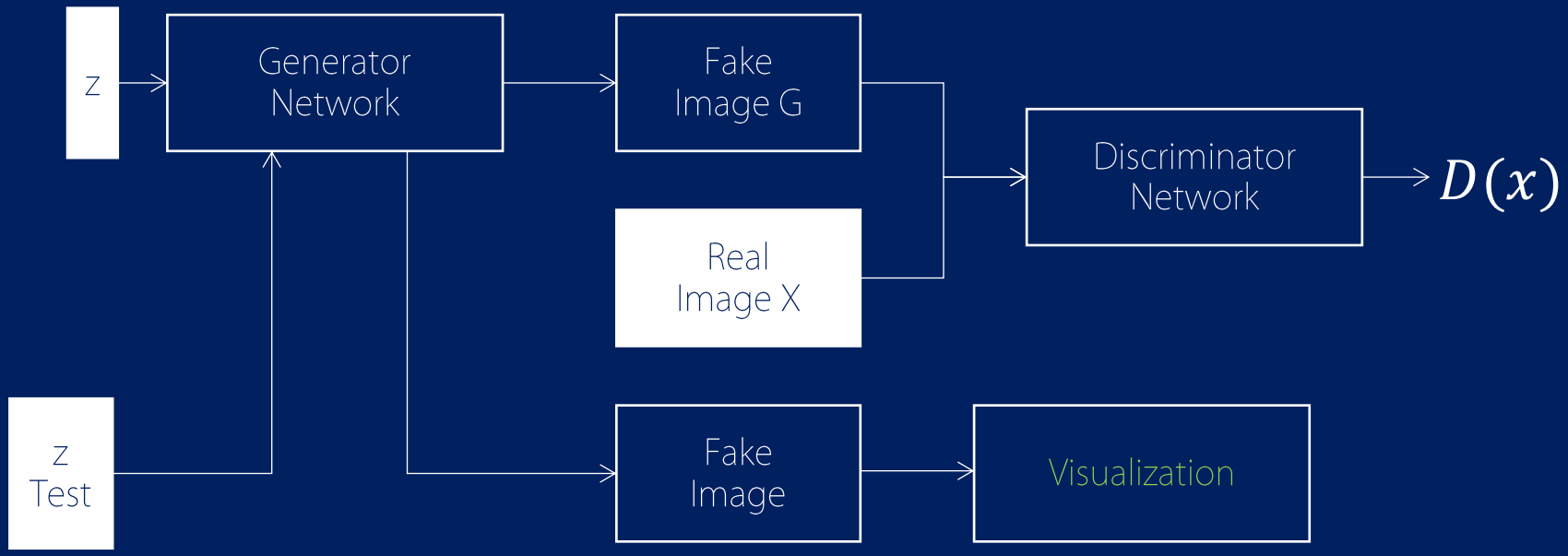



```
loss_G = -tf.reduce_mean(tf.log(discriminator(G)))

train_G = tf.train.AdamOptimizer(0.0002).minimize(
        loss_G, var_list=[G_W1, G_b1, G_W2, G_b2])

# Z->G->loss_G->train_G
```

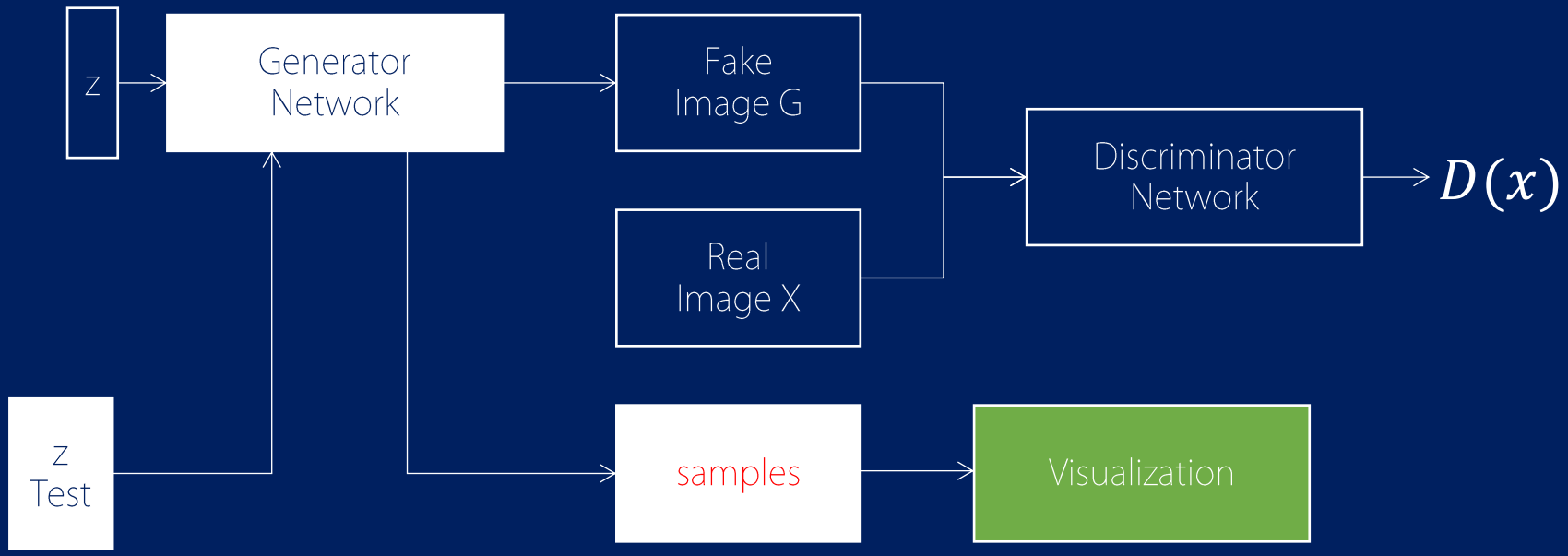
$$D_{Loss} = -\log\big(D(x)\big) - \log(1 - D(G(z))$$
$$G_{Loss} = -\log(D(G(z))$$




```
sess = tf.Session()

sess.run(tf.global_variables_initializer())
```




```
noise_test = np.random.normal(size=(10, 128))

for epoch in range(200):
        for i in range(int(mnist.train.num_examples / 100)):
                batch_xs, _ = mnist.train.next_batch(100)

                noise = np.random.normal(size=100, 128))

                sess.run(train_D, feed_dict={X: batch_xs, Z: noise})
                sess.run(train_G, feed_dict={Z: noise})
```



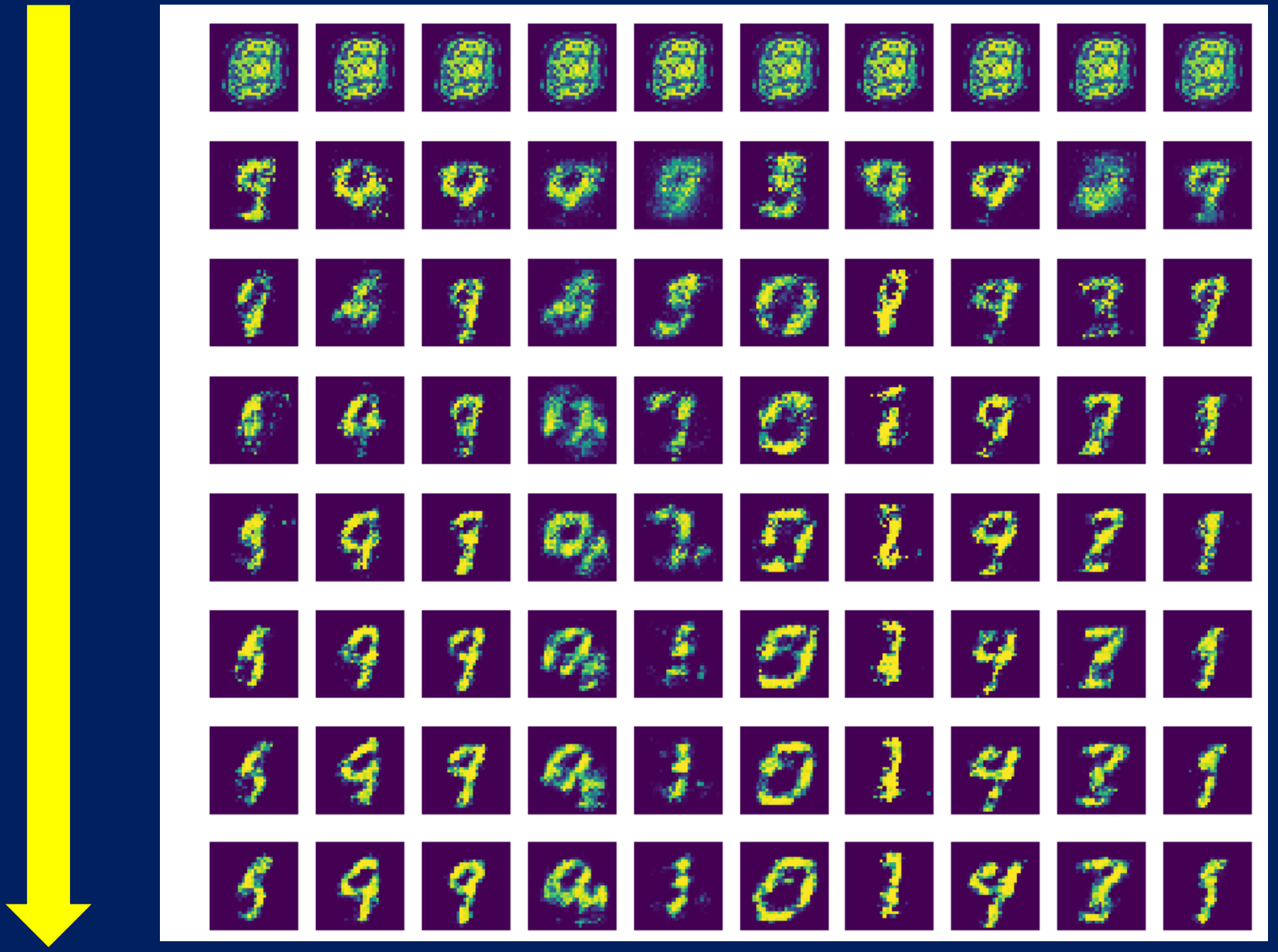

```
for epoch in range(200):
        ...
        if epoch == 0 or (epoch + 1) % 10 == 0:
                samples = sess.run(G, feed_dict={Z: noise_test})


                fig, ax = plt.subplots(1, 10, figsize=(10, 1))
                for i in range(10):
                                ax[i].set_axis_off()
                                ax[i].imshow(np.reshape(samples[i], (28, 28)))
                plt.savefig('samples_ex/{}.png'.format(str(epoch).zfill(3)), bbox_inches='tight')
                plt.close(fig)
```

Full Code

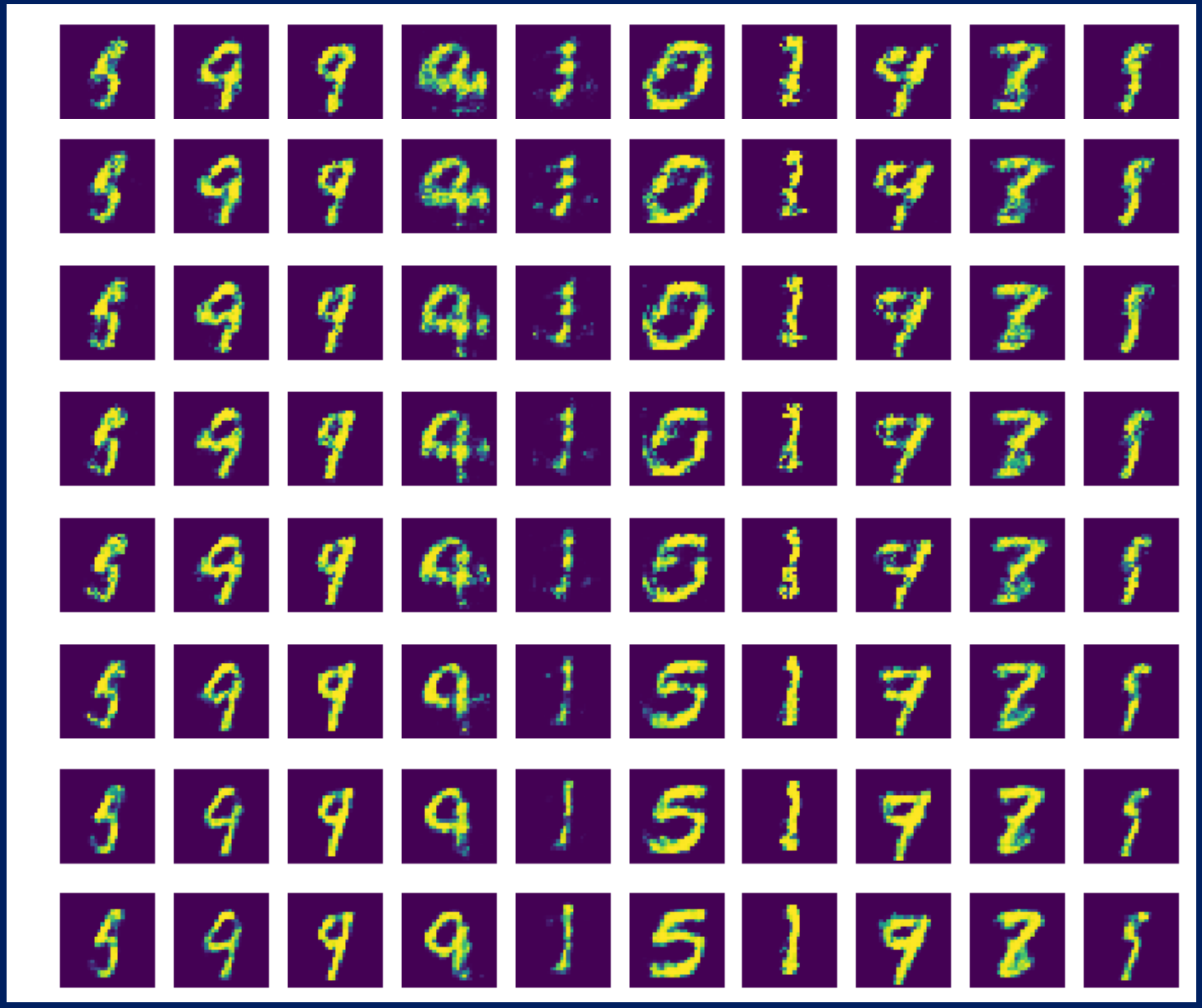
https://github.com/HyeongminLEE/GANin50lines

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)
X = tf.placeholder(tf.float32, [None, 28 * 28]) # MNIST = 28*28
Z = tf.placeholder(tf.float32, [None, 128]) # Noise Dimension = 128

# ********* G-Network (Hidden Node # = 256)
G_W1 = tf.Variable(tf.random_normal([128, 256], stddev=0.01))
G_W2 = tf.Variable(tf.random_normal([256, 28 * 28], stddev=0.01))
G_b1 = tf.Variable(tf.zeros([256]))
G_b2 = tf.Variable(tf.zeros([28 * 28]))

def generator(noise_z): # 128 -> 256 -> 28*28
    hidden = tf.nn.relu(tf.matmul(noise_z, G_W1) + G_b1)
    output = tf.nn.sigmoid(tf.matmul(hidden, G_W2) + G_b2)
    return output

# ********* D-Network (Hidden Node # = 256)
D_W1 = tf.Variable(tf.random_normal([28 * 28, 256], stddev=0.01))
D_W2 = tf.Variable(tf.random_normal([256, 1], stddev=0.01))
D_b1 = tf.Variable(tf.zeros([256]))
D_b2 = tf.Variable(tf.zeros([1]))

def discriminator(inputs): # 28*28 -> 256 -> 1
    hidden = tf.nn.relu(tf.matmul(inputs, D_W1) + D_b1)
    output = tf.nn.sigmoid(tf.matmul(hidden, D_W2) + D_b2)
    return output

# ********* Generation, Loss, Optimization and Session Init.
G = generator(Z)
loss_D = -tf.reduce_mean(tf.log(discriminator(X)) + tf.log(1 - discriminator(G)))
loss_G = -tf.reduce_mean(tf.log(discriminator(G)))
train_D = tf.train.AdamOptimizer(learning_rate=0.0002).minimize(loss_D,
var_list=[D_W1, D_b1, D_W2, D_b2])
train_G = tf.train.AdamOptimizer(learning_rate=0.0002).minimize(loss_G,
var_list=[G_W1, G_b1, G_W2, G_b2])

sess = tf.Session()
sess.run(tf.global_variables_initializer())

# ********* Training and Testing
noise_test = np.random.normal(size=(10, 128)) # 10 = Test Sample Size
for epoch in range(200): # 200 = Num. of Epoch
    for i in range(int(mnist.train.num_examples / 100)): # 100 = Batch Size
        batch_xs, _ = mnist.train.next_batch(100)
        noise = np.random.normal(size=(100, 128))

        sess.run(train_D, feed_dict={X: batch_xs, Z: noise})
        sess.run(train_G, feed_dict={Z: noise})

    if epoch == 0 or (epoch + 1) % 10 == 0: # 10 = Saving Period
        samples = sess.run(G, feed_dict={Z: noise_test})

        fig, ax = plt.subplots(1, 10, figsize=(10, 1))
        for i in range(10):
            ax[i].set_axis_off()
            ax[i].imshow(np.reshape(samples[i], (28, 28)))
        plt.savefig('samples_ex/{}.png'.format(str(epoch).zfill(3)),
bbox_inches='tight')
        plt.close(fig)
```

Results



Results



GAN Does Not
Easily Converge!

And Quality is Not Good!

→ DCGAN
Solve this Empirically

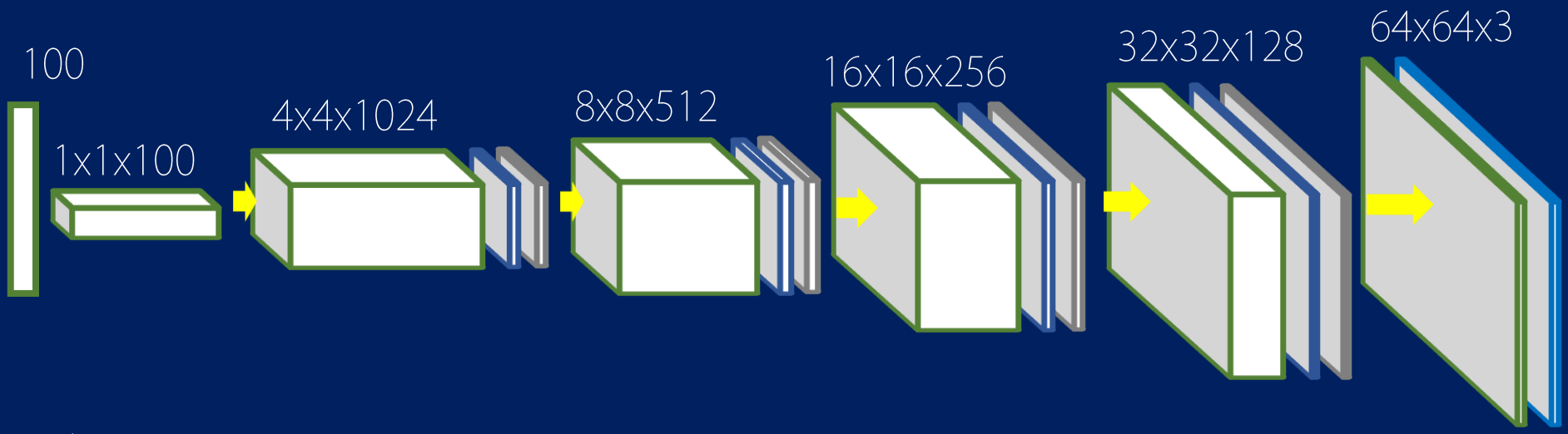
DCGAN

Deep Convolutional GAN

DCGAN's Solution

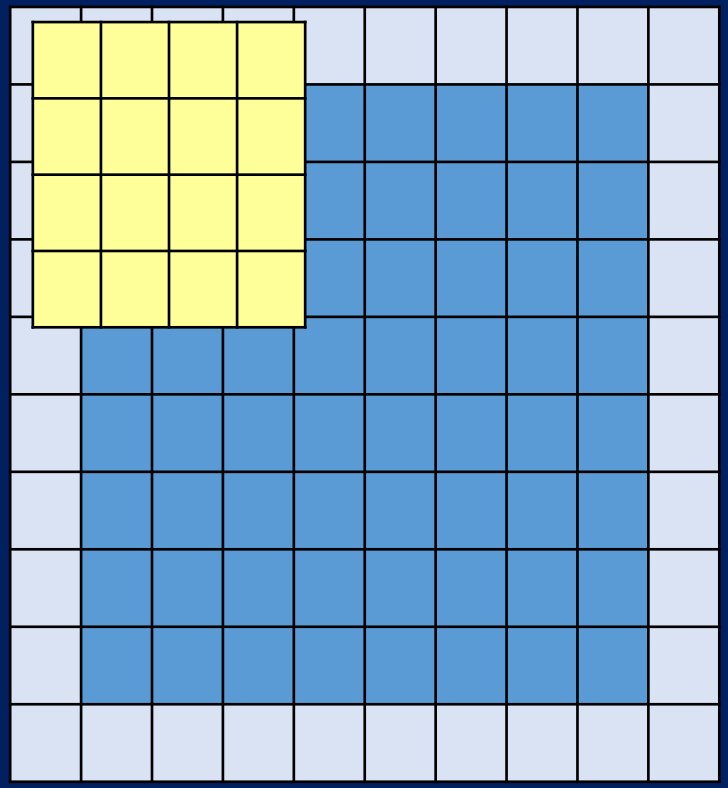
- Go Deep + CNN
- Batch Normalization
- Fully Convolutional
- No Pooling
- Leaky-ReLU

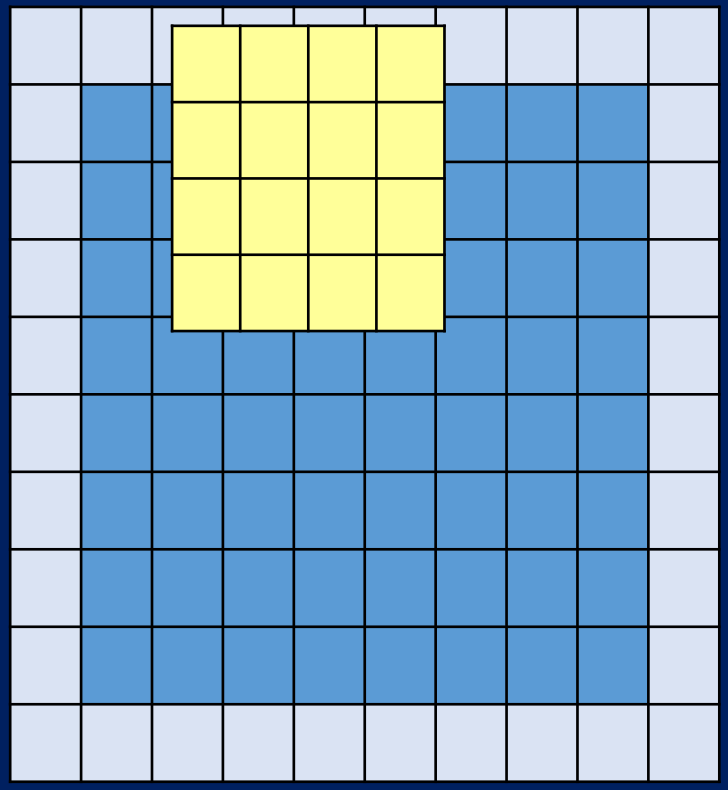
DCGAN Generator

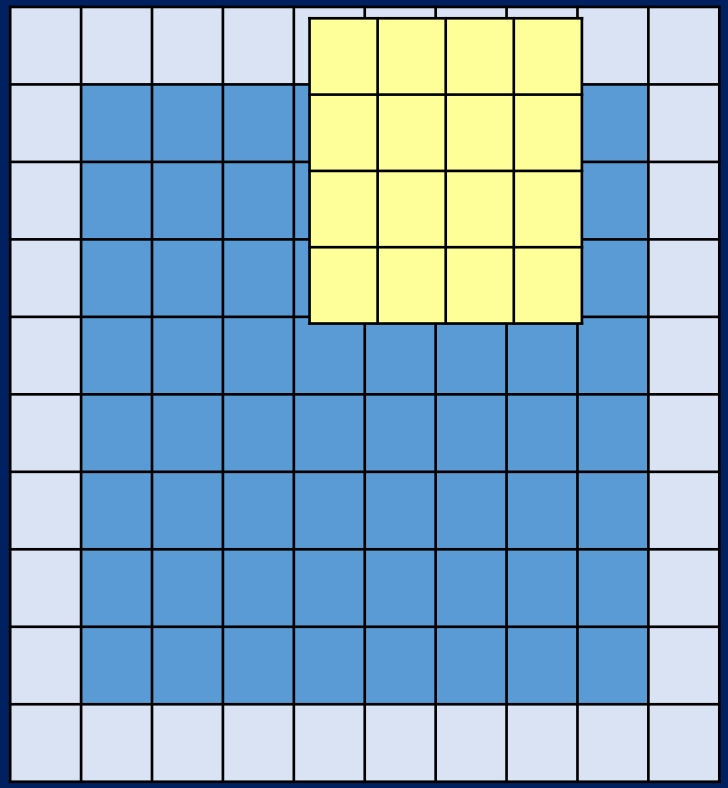


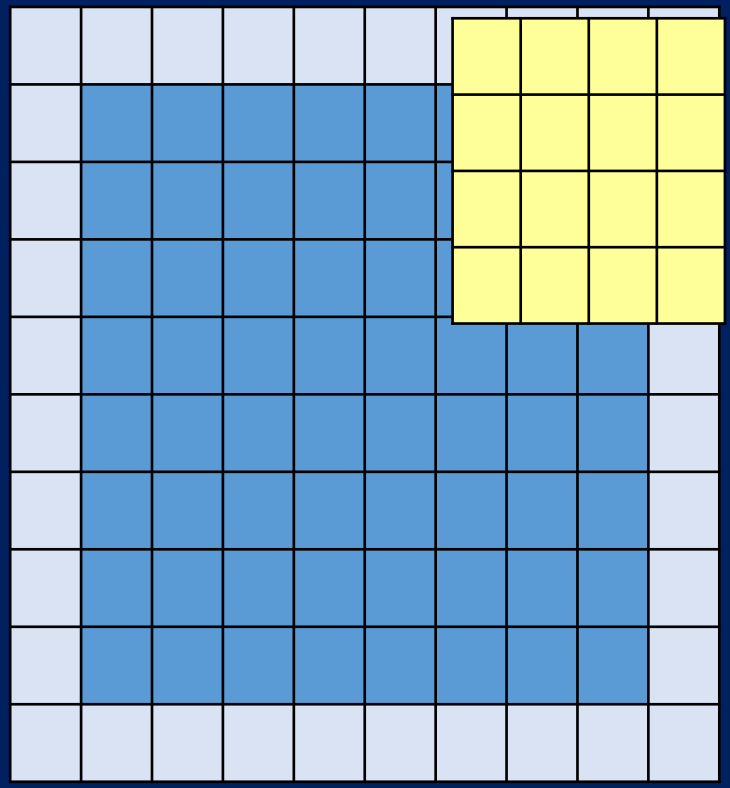
Filter Size = 4x4
Stride = 2
Padding = 1

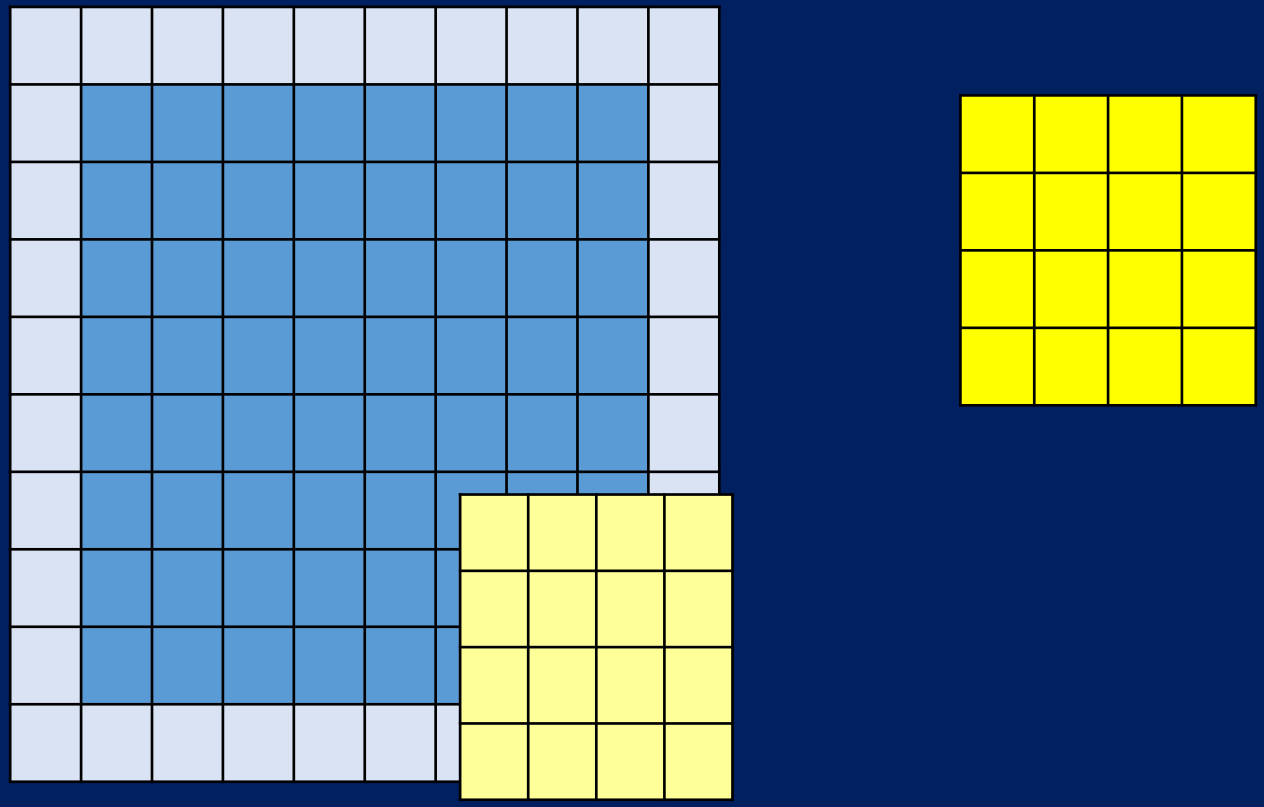
Image Should be Normalized [-1 , 1]



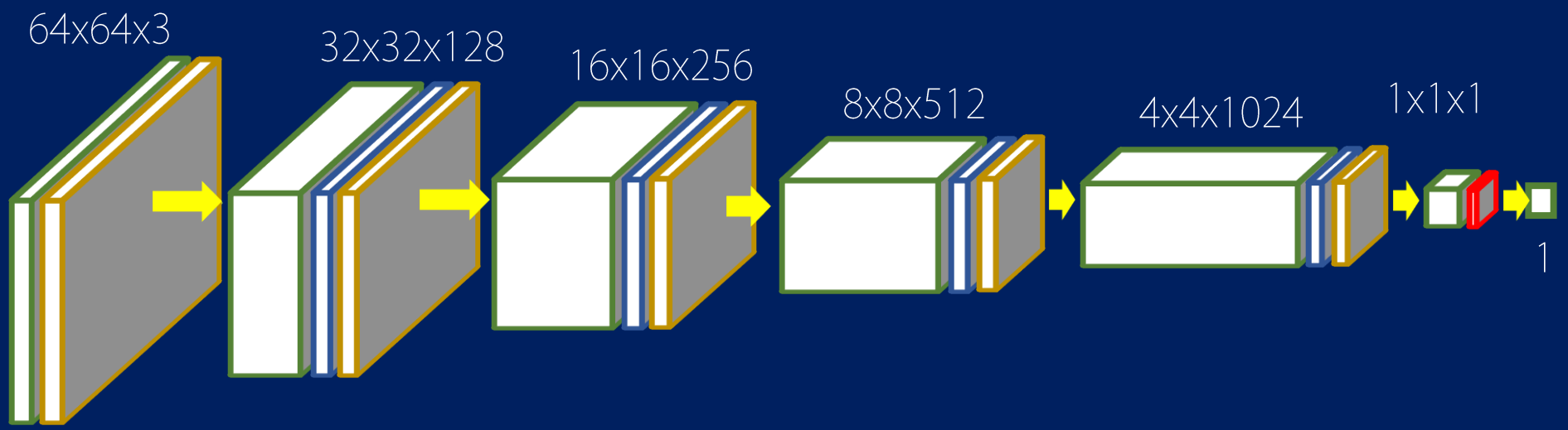






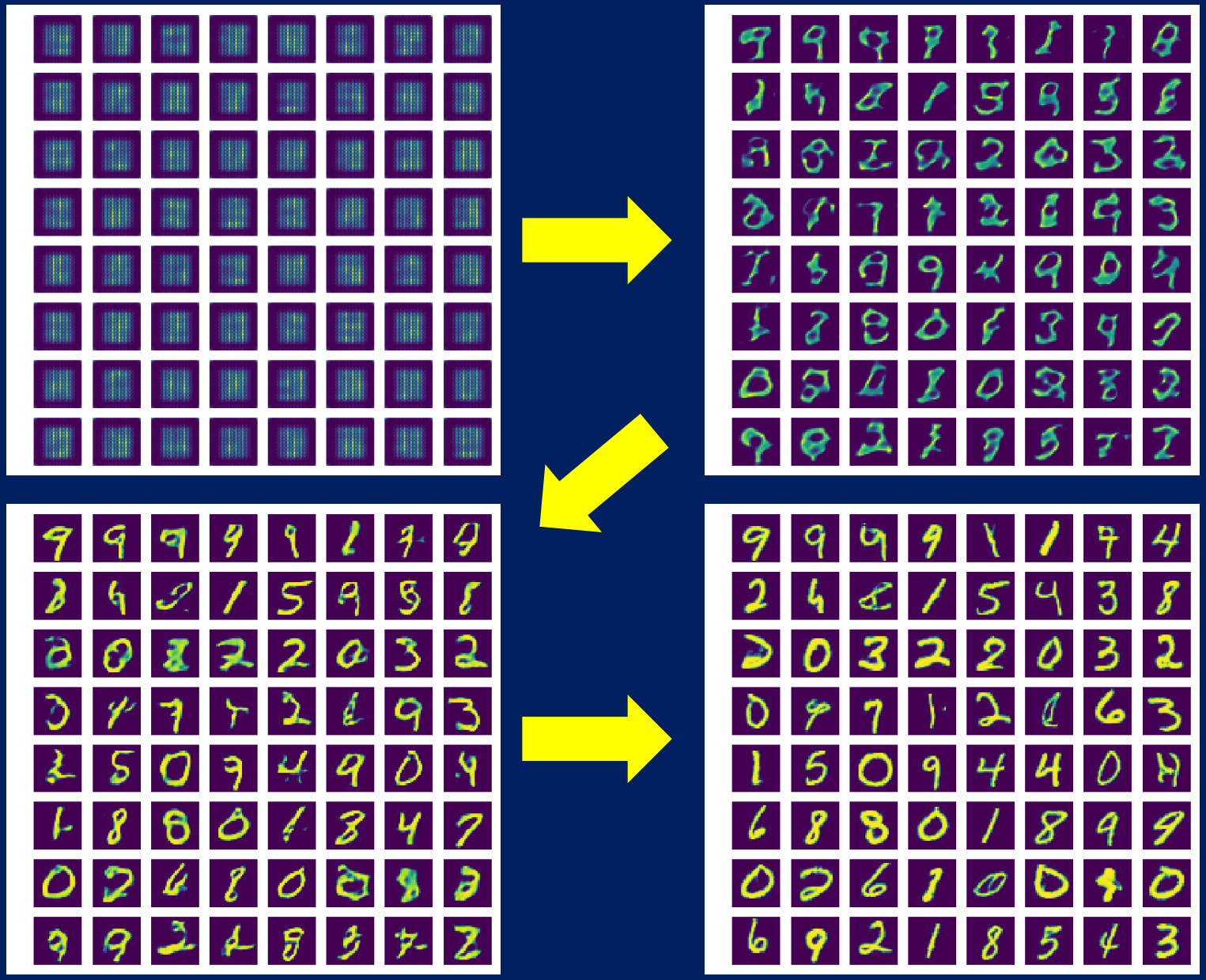


DCGAN Discriminator



Filter Size = 4x4
Stride = 2
Padding = 1

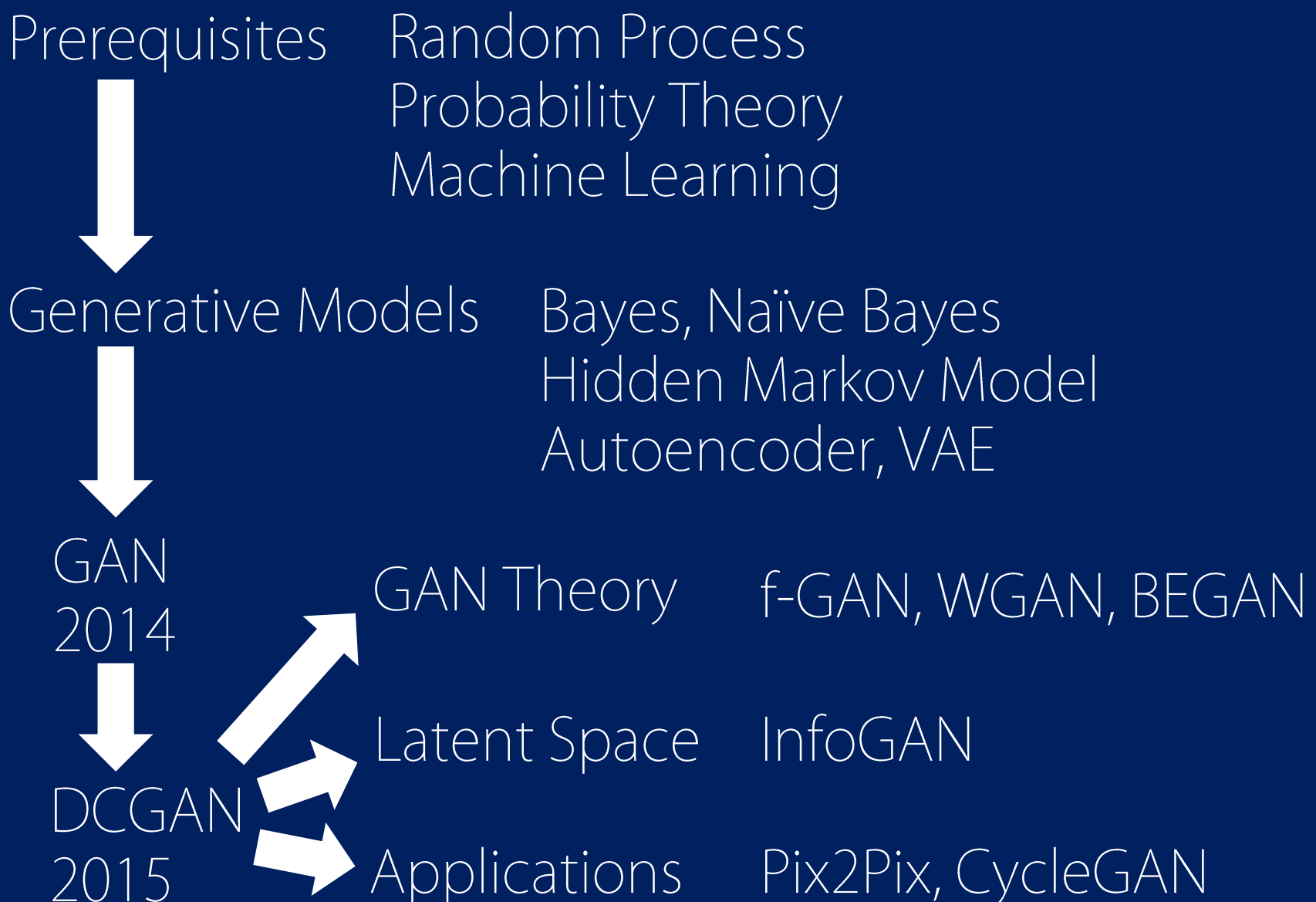
DCGAN Results



DCGAN Code

https://github.com/taeoh-kim/DCGAN_Pytorch

GAN History


Prerequisites
Random Process
Probability Theory
Machine Learning
Generative Models
Bayes, Naïve Bayes
Hidden Markov Model
Autoencoder, VAE
GAN
2014
DCGAN
2015
GAN Theory
f-GAN, WGAN, BEGAN
Latent Space
InfoGAN
Applications
Pix2Pix, CycleGAN
DiscoGAN, ...

InfoGAN
Exploit Latent Noise

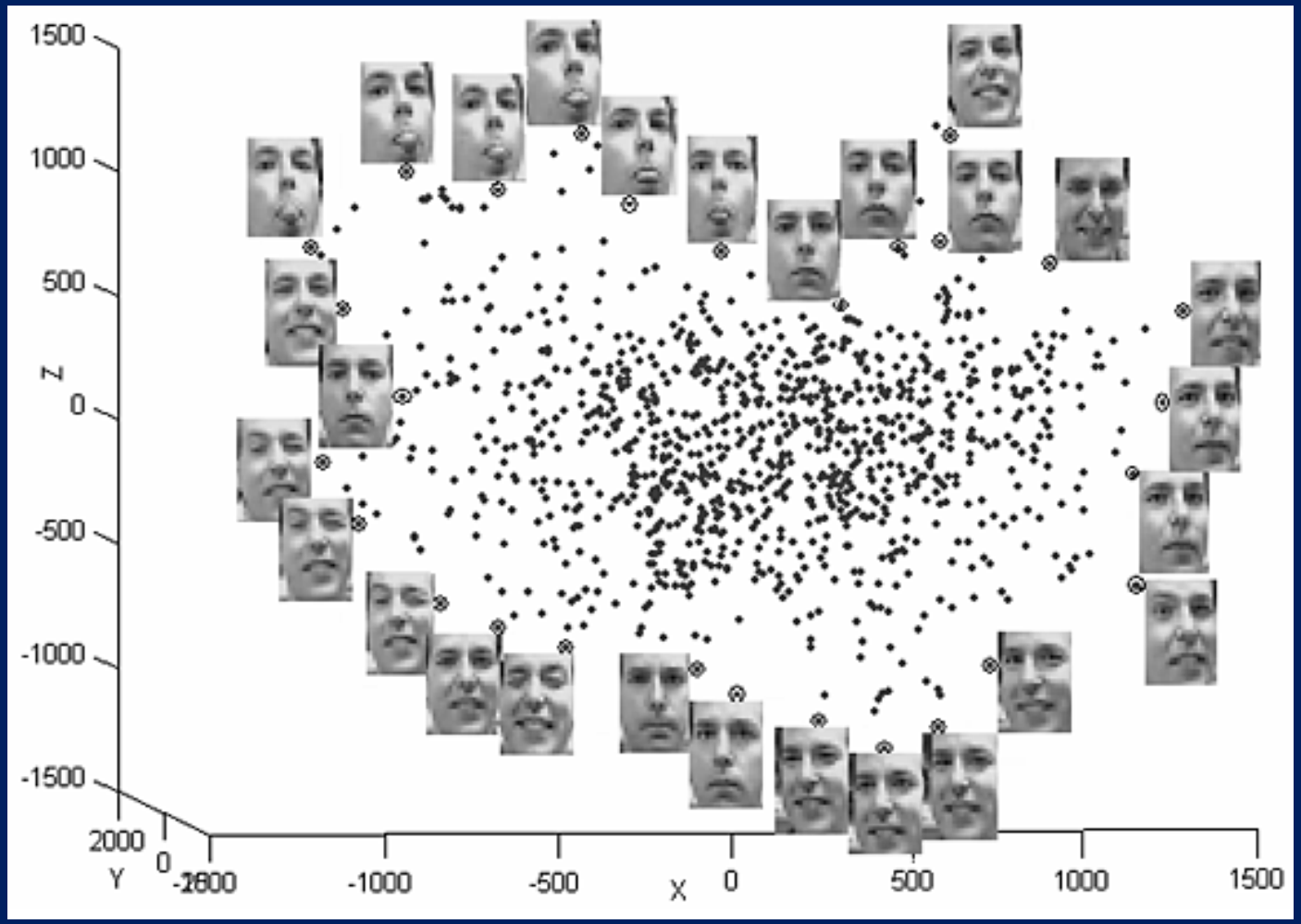
Exploit Latent Vector z

How to Exploit z?

and I want Generate "Specific" Image!
ex) Smile Woman Face with Black Skin

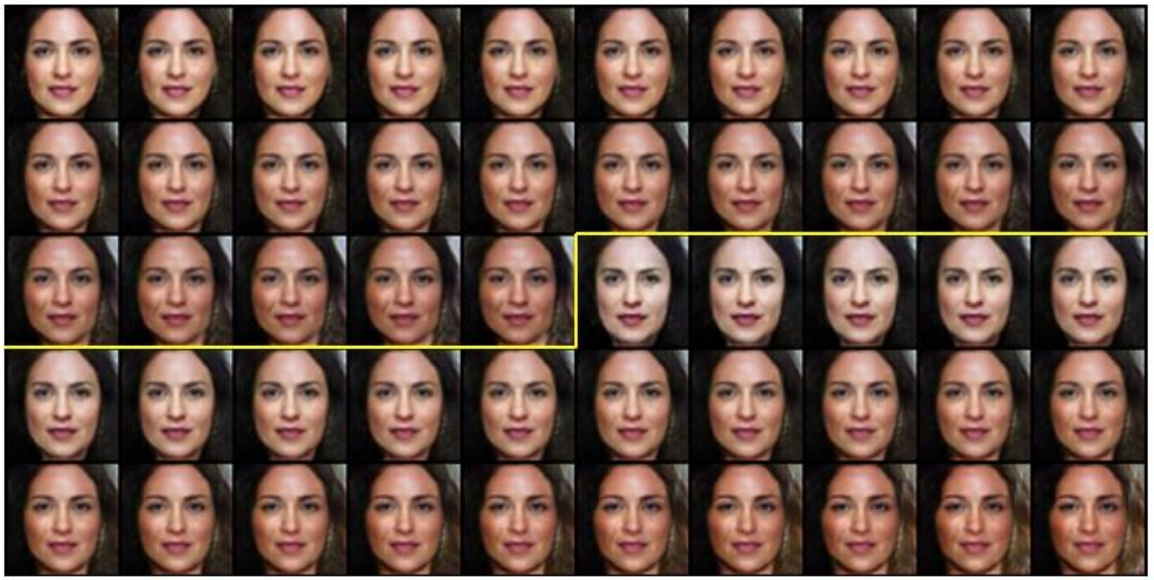
- Conditional-GAN
  with Label
- InfoGAN / VAE
  with Meaningful z

Data Manifold Hypothesis

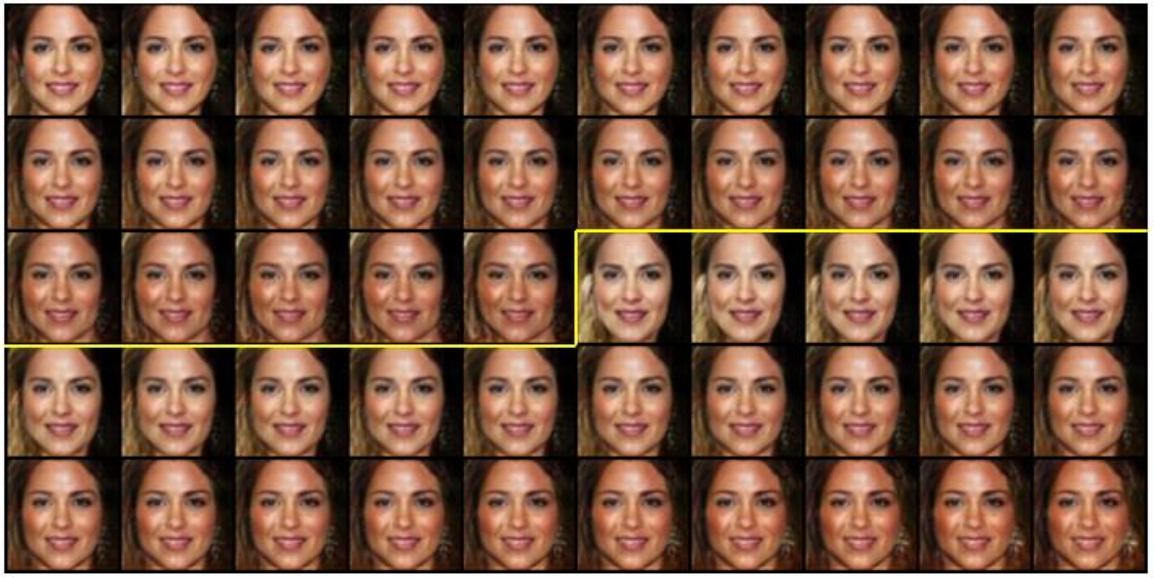


InfoGAN https://github.com/taeoh-kim/GAN_pytorch

1 Categorical Feature, 2 Continuous Feature, 256 Noise



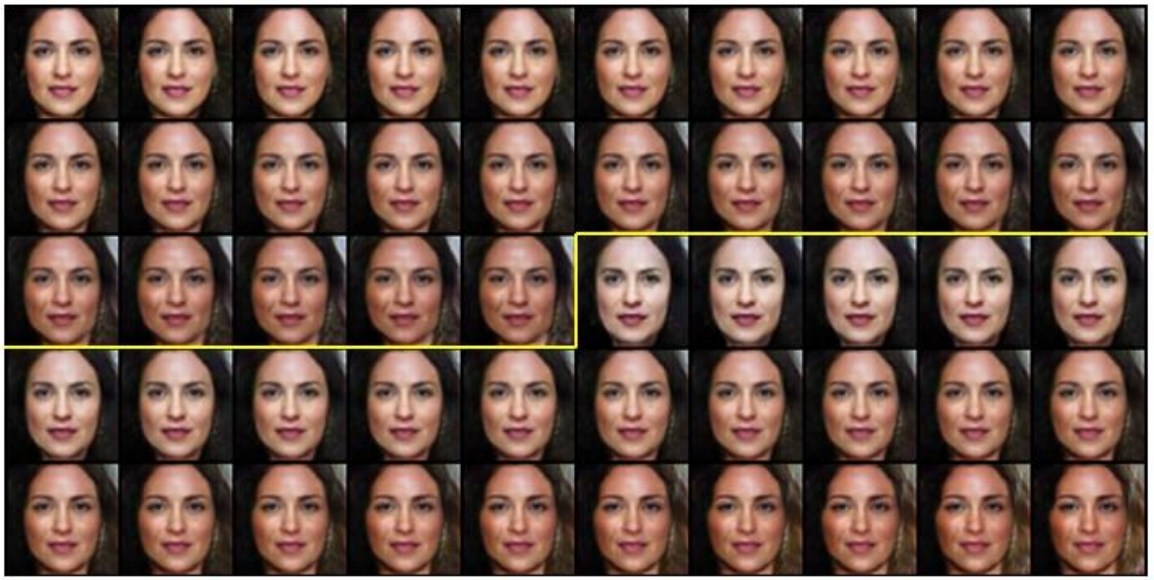
Neutral



Smile

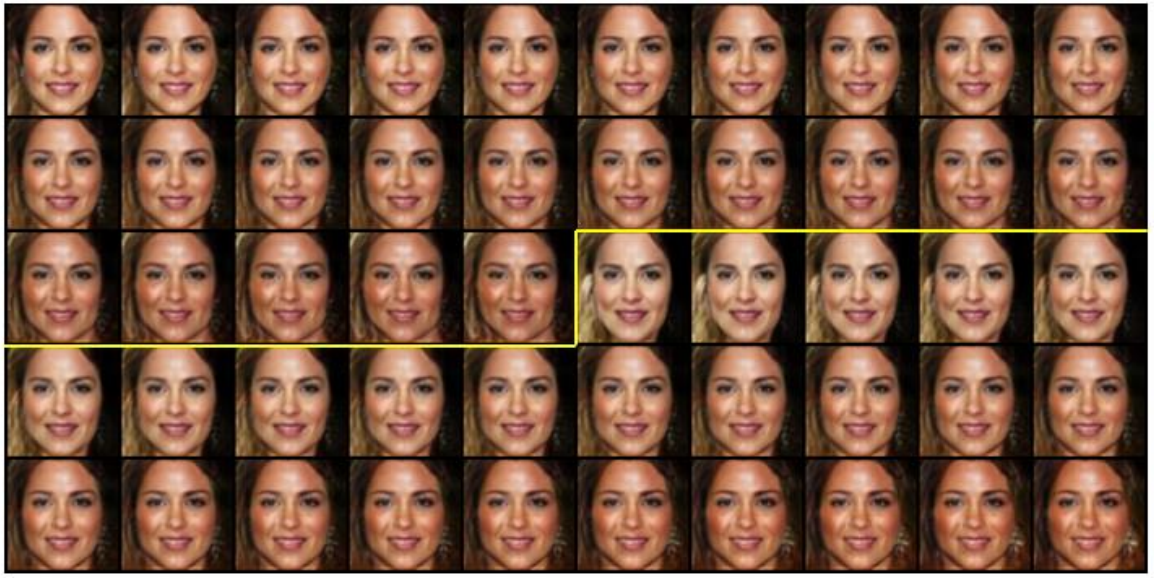
InfoGAN https://github.com/taeoh-kim/GAN_pytorch

1 Categorical Feature, 2 Continuous Feature, 256 Noise



Aging

Skin/Hair Color



Aging

Skin/Hair Color

GAN Issue

- Convergence
- Realistic Visual
- Performance Measure
- Application

Part 2.
GANs for CV

Overview

- Image Translation
  - Generation, Segmentation, Style Transfer
  - Edge Detection, Colorization, Multi-model Imaging
- Super Resoultion
- Feature Upscaling: Object Detection
- 3D Reconstruction
- Semi-supervised Learning
  - Pose Estimation, Face Recognition, Pedestrian Detection
- Face Aging Modeling

Image Translation
Domain Transfer

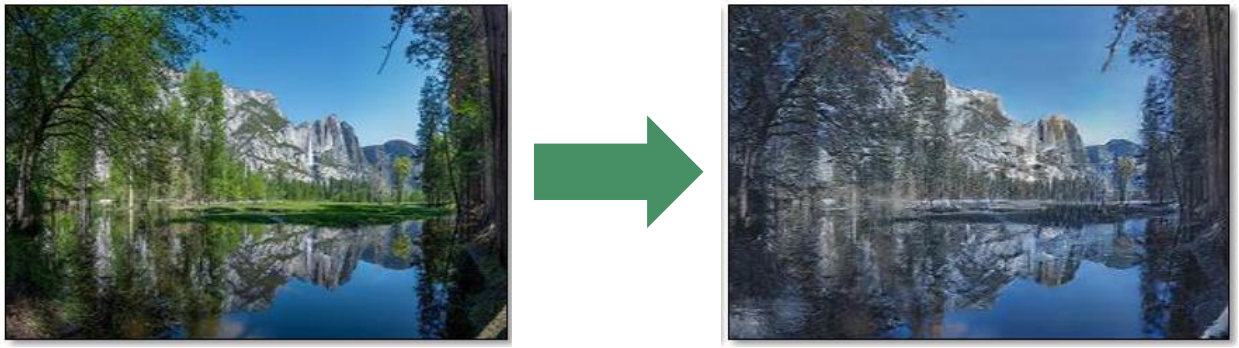


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks,
a.k.a Pix2Pix, CVPR 2017

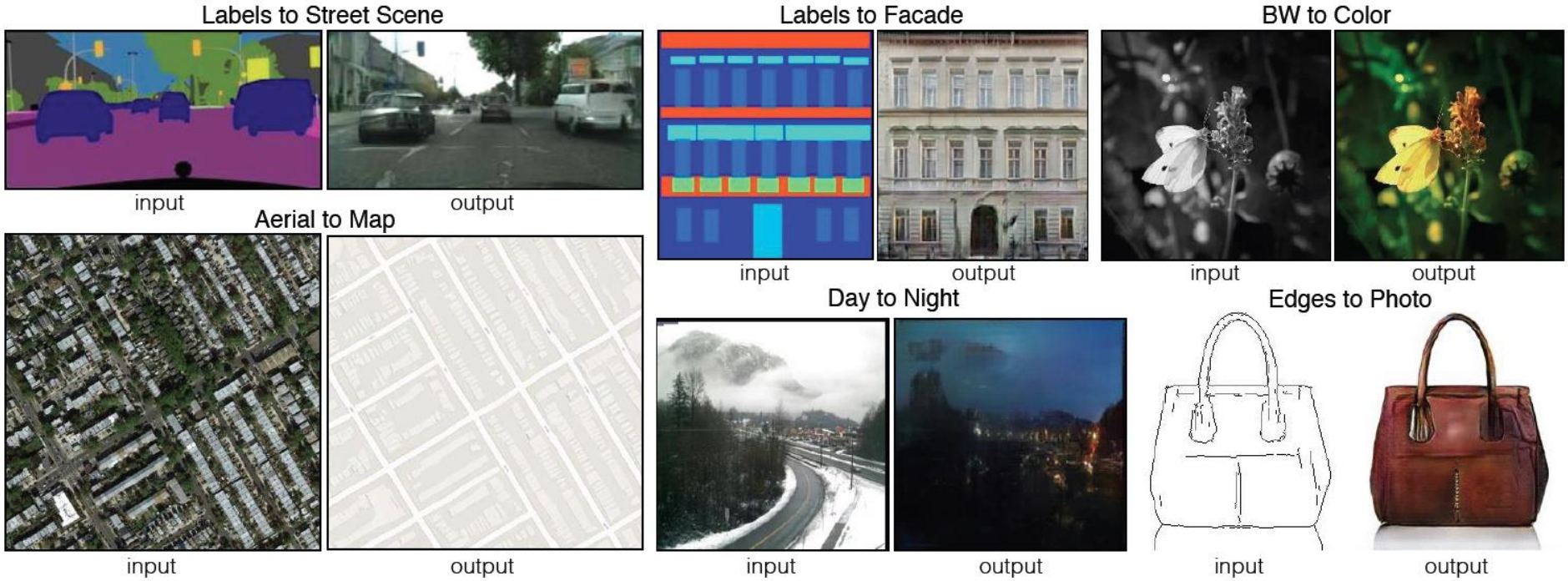


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

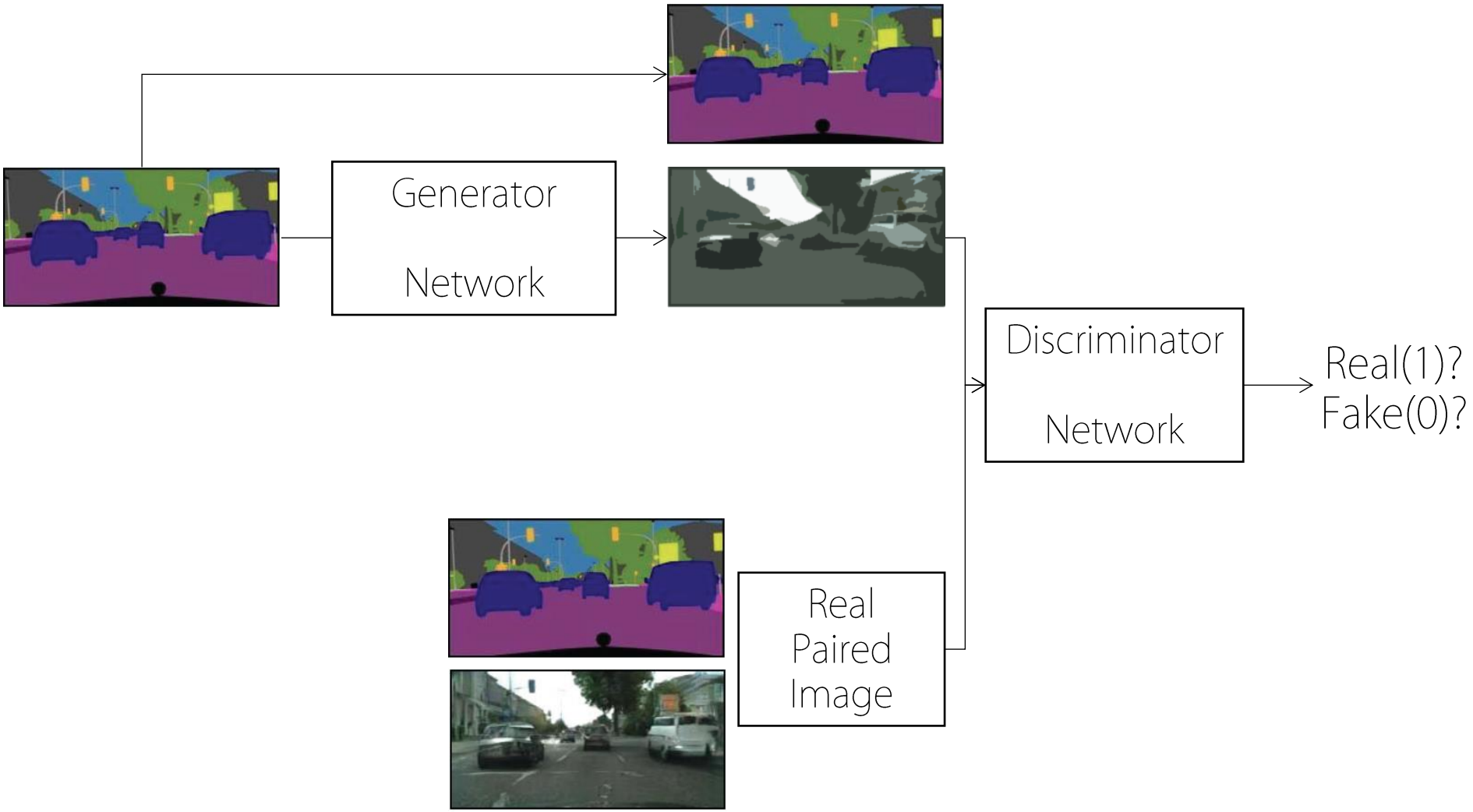


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

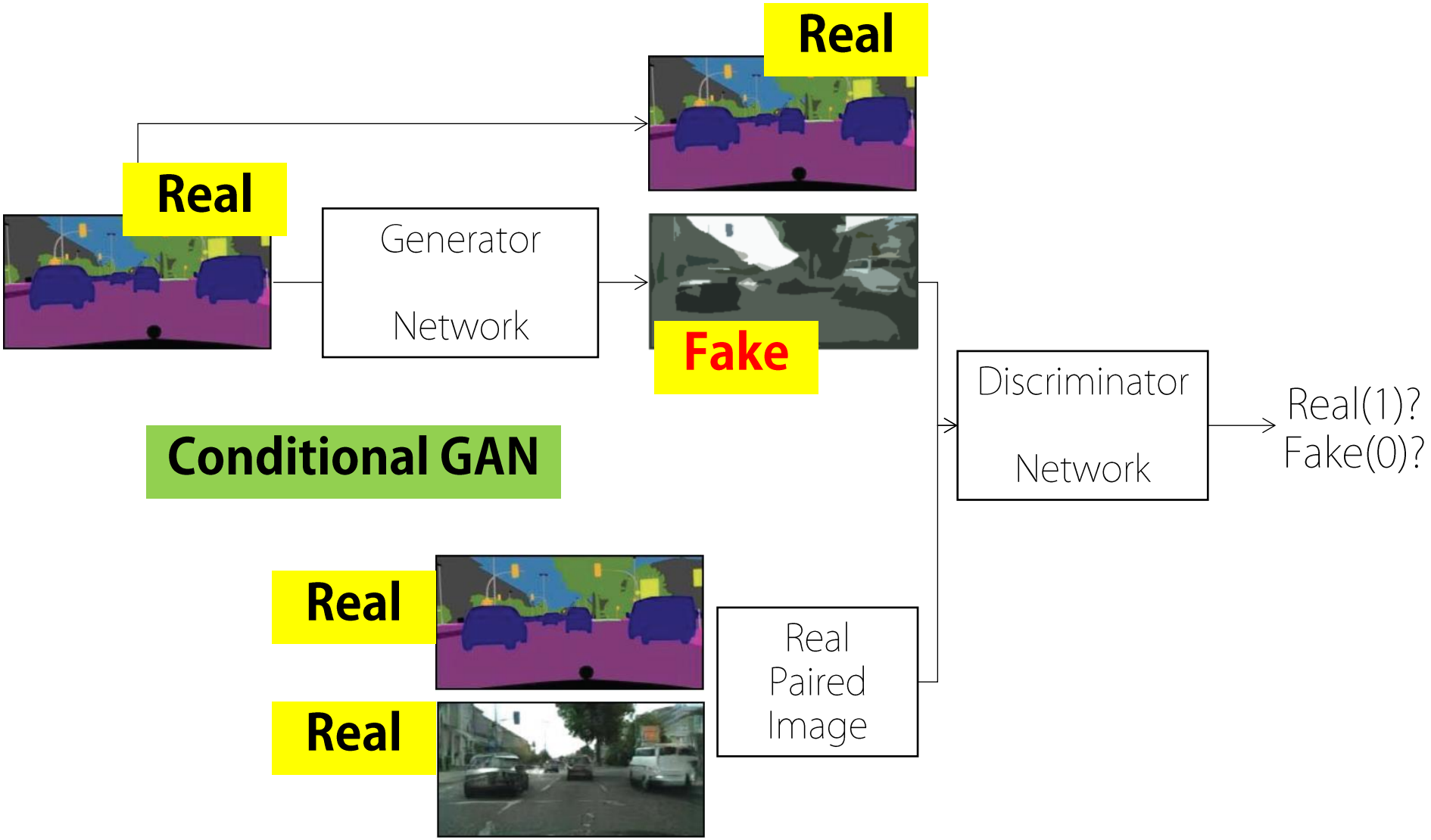


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

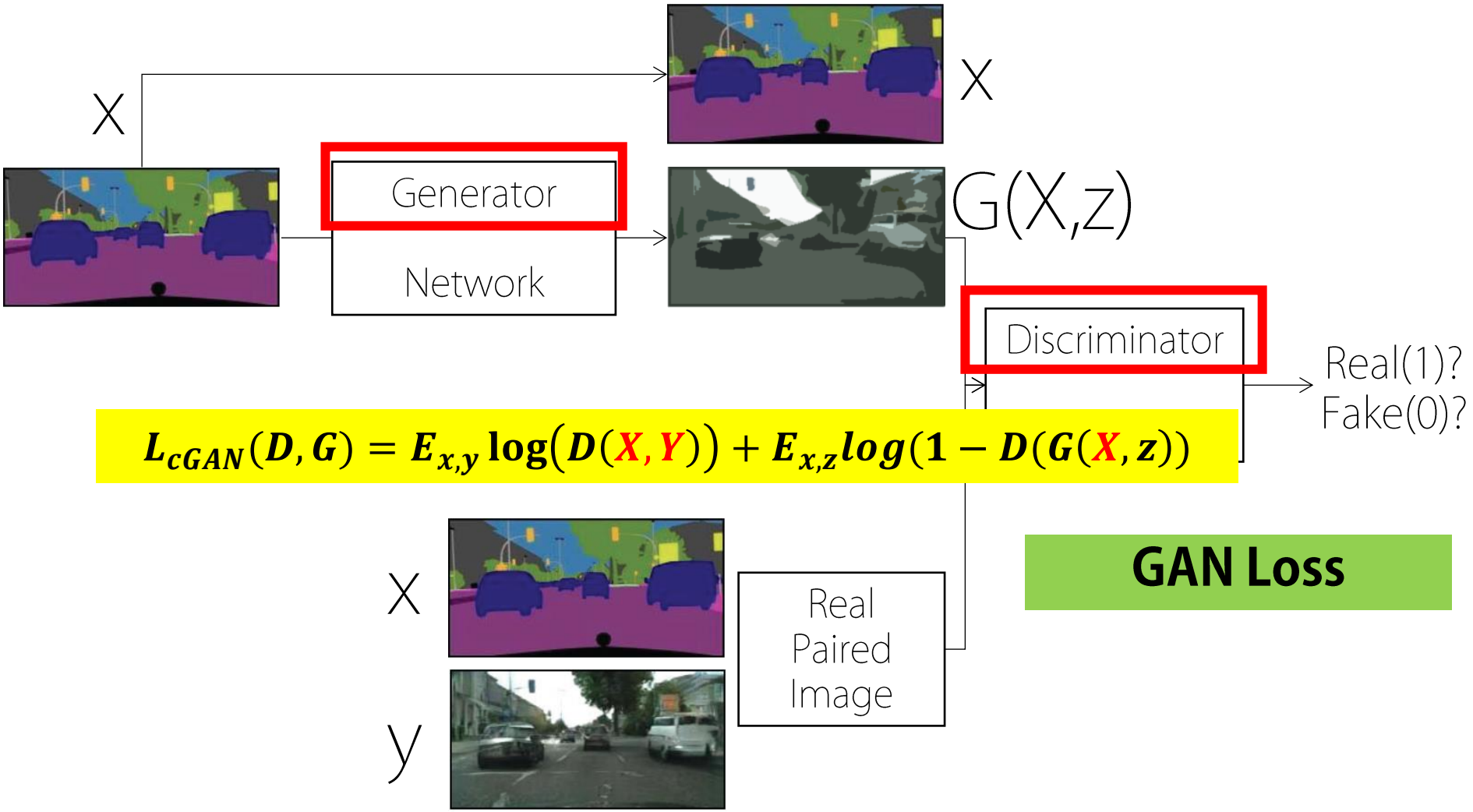


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

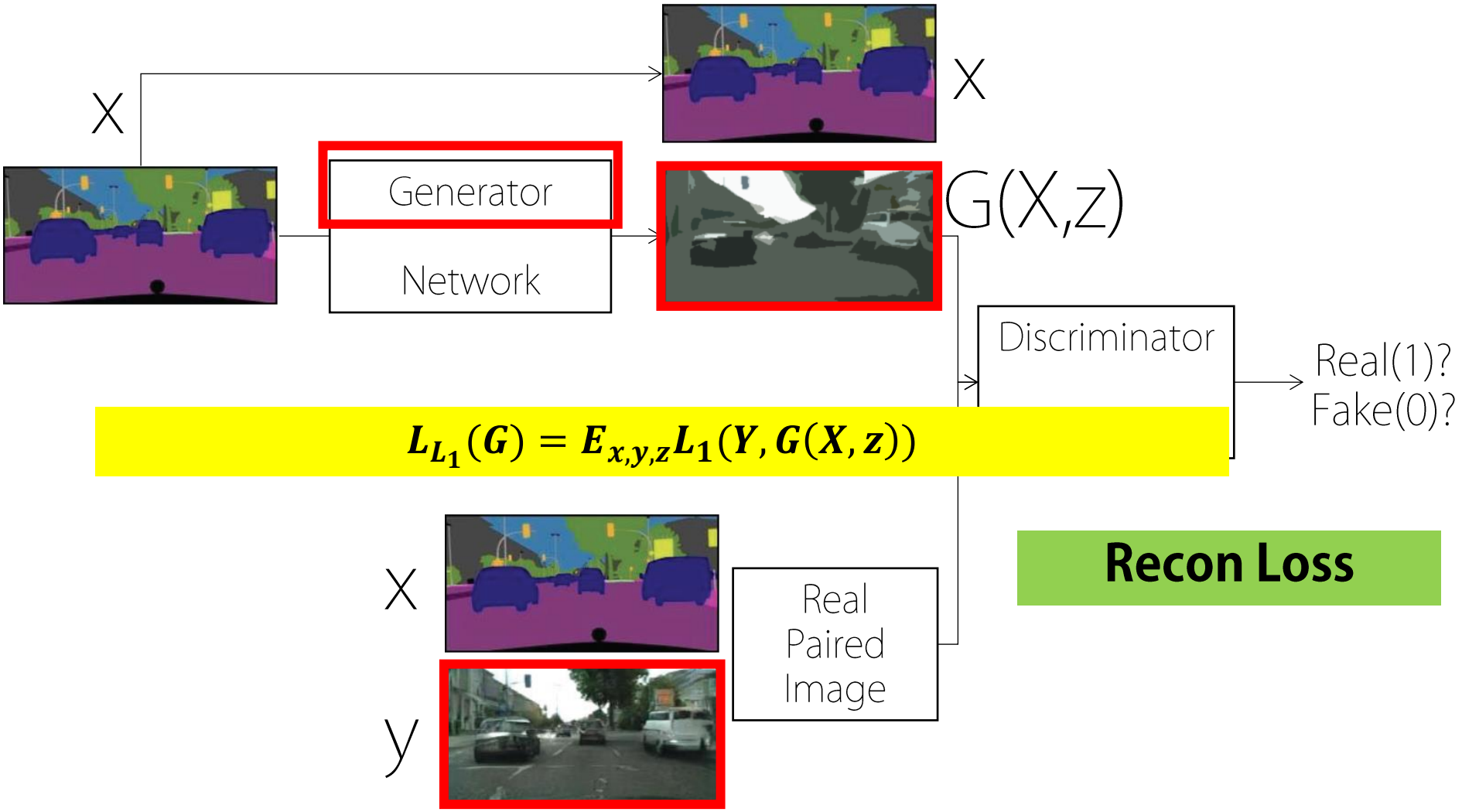
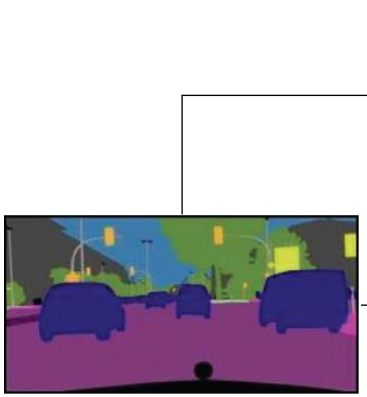
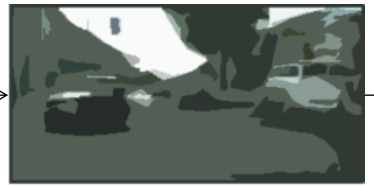


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017



Generator Network



Discriminator

Real(1)?
Fake(0)?

$$L(D,G) = L_{cGAN}(D,G) + \lambda L_{L_1}(G)$$

Real Paired Image

Total Loss

Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

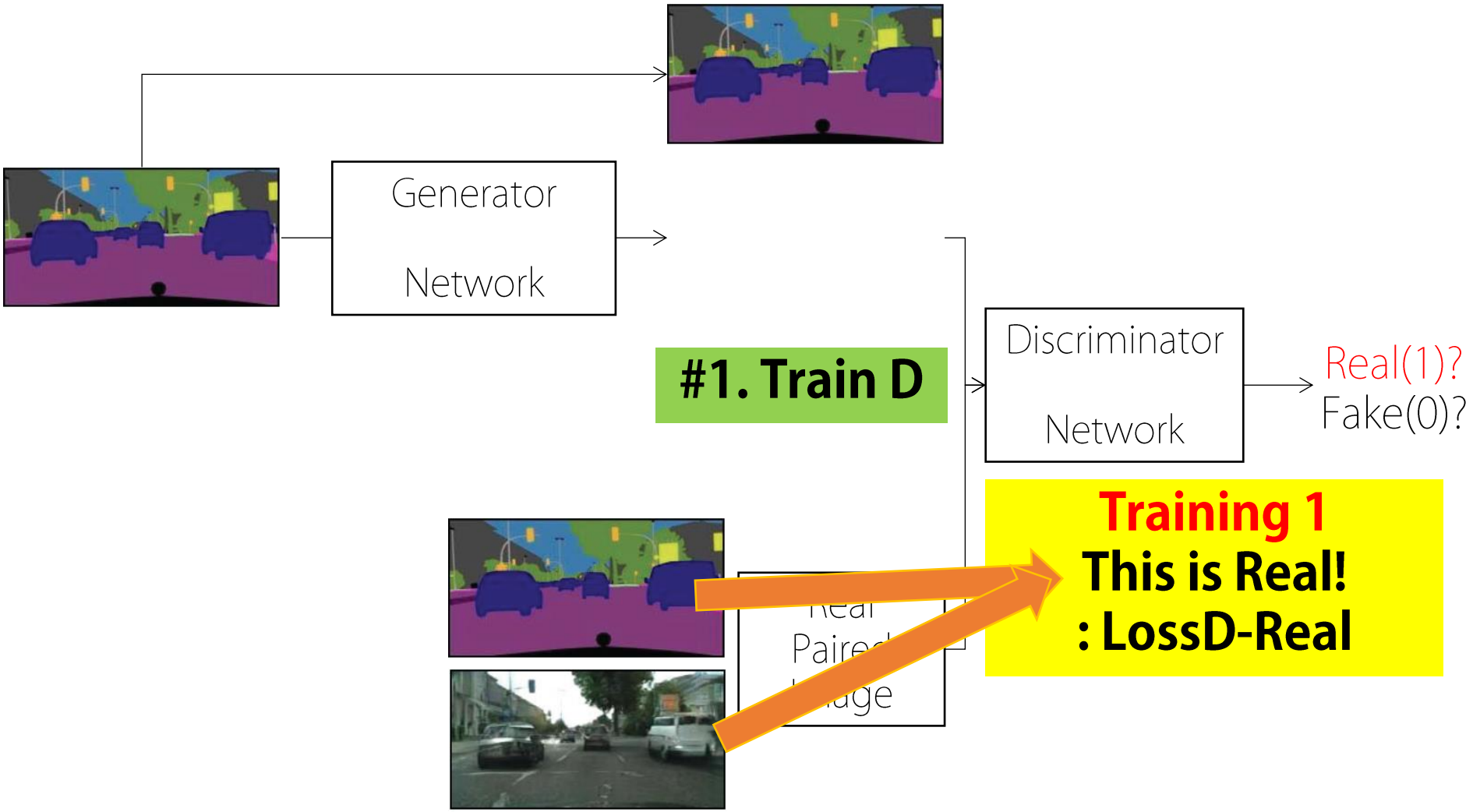


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

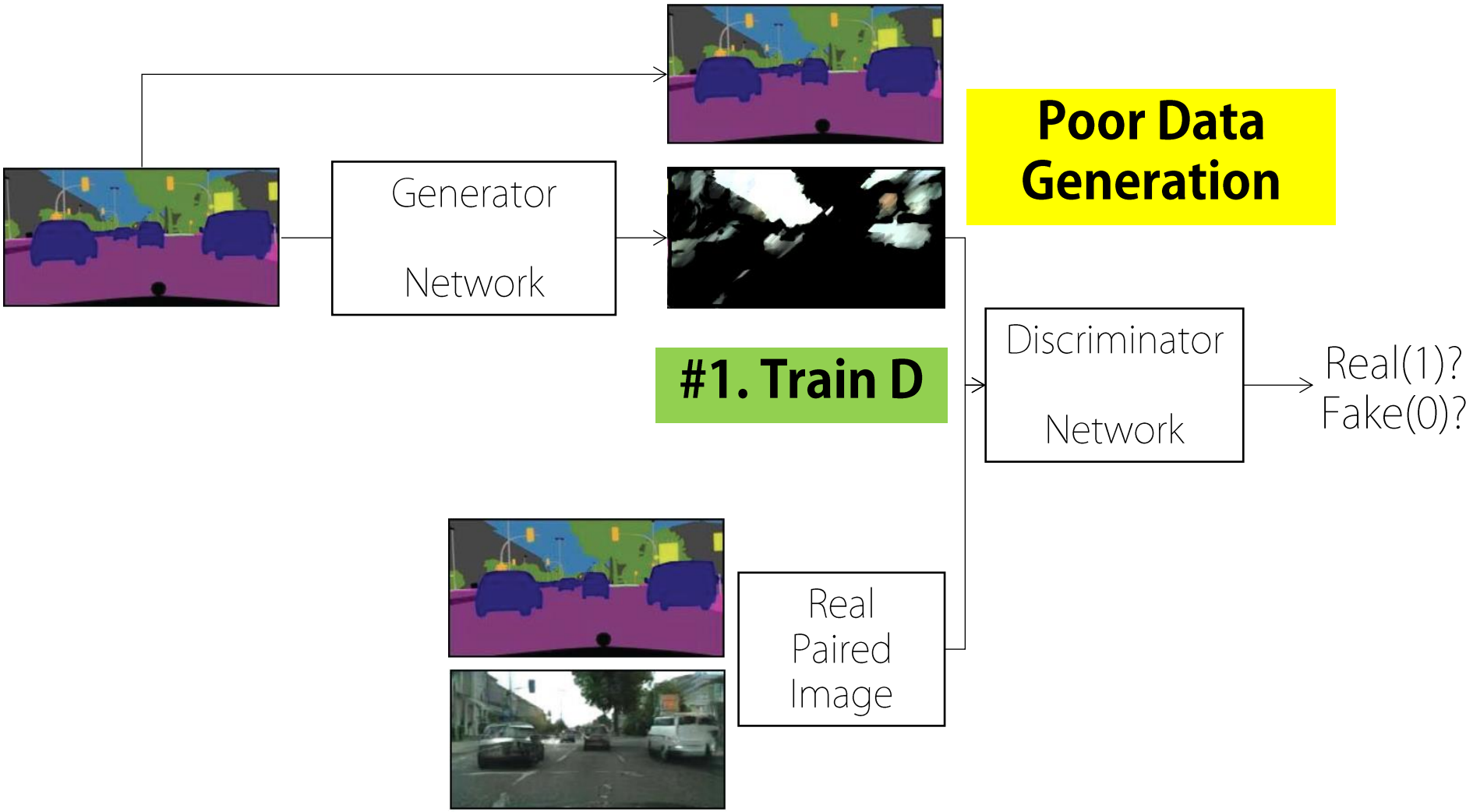


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

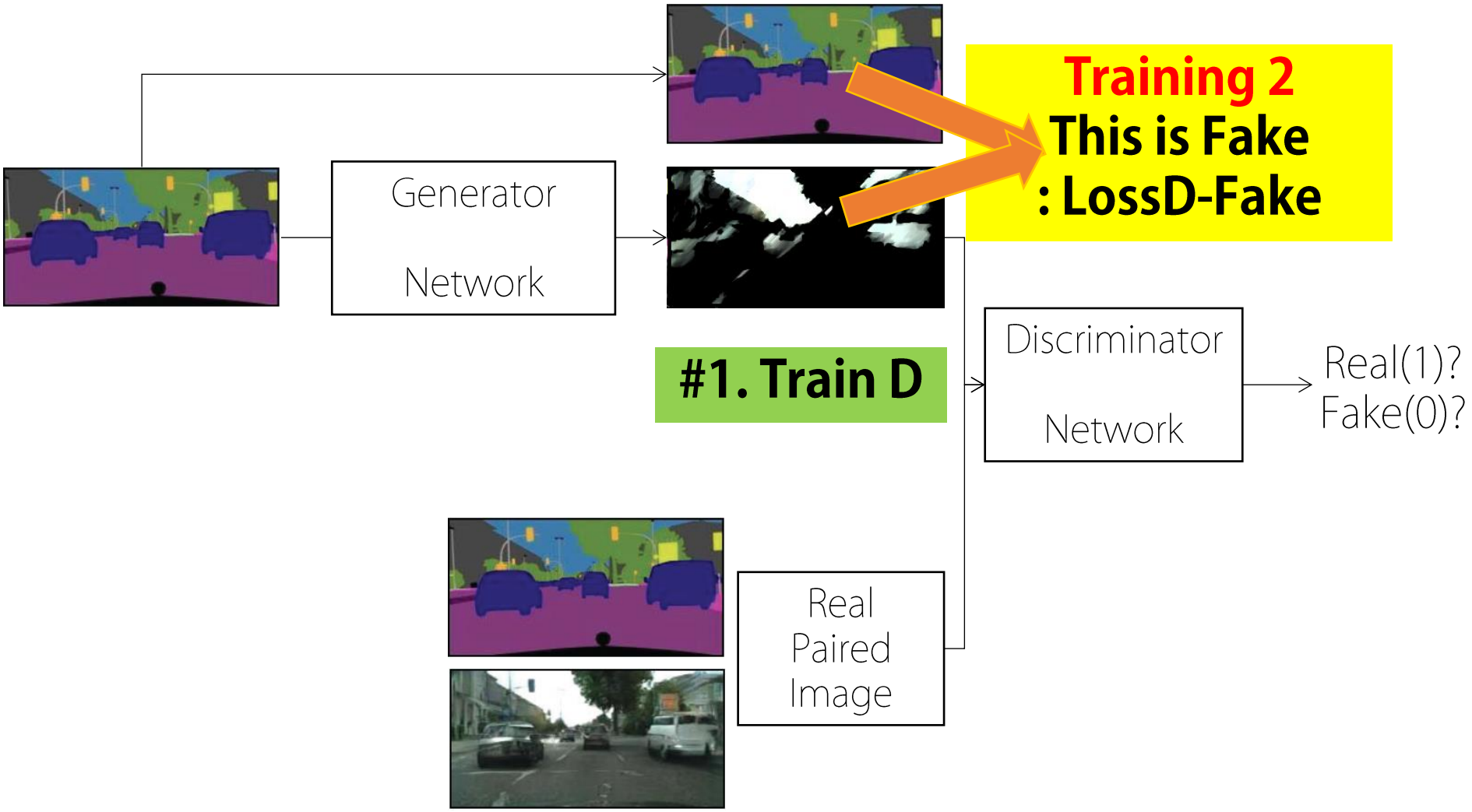


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

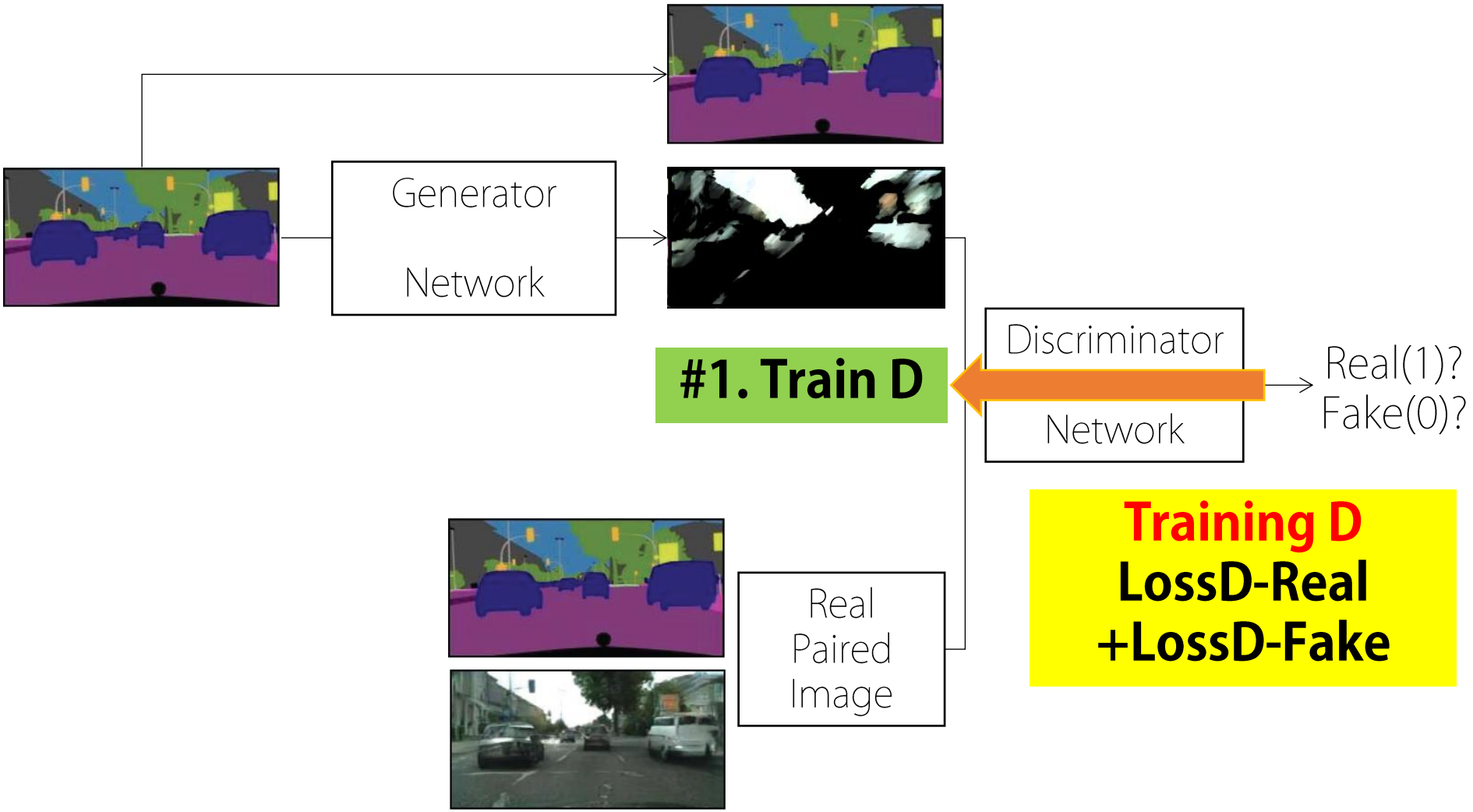


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

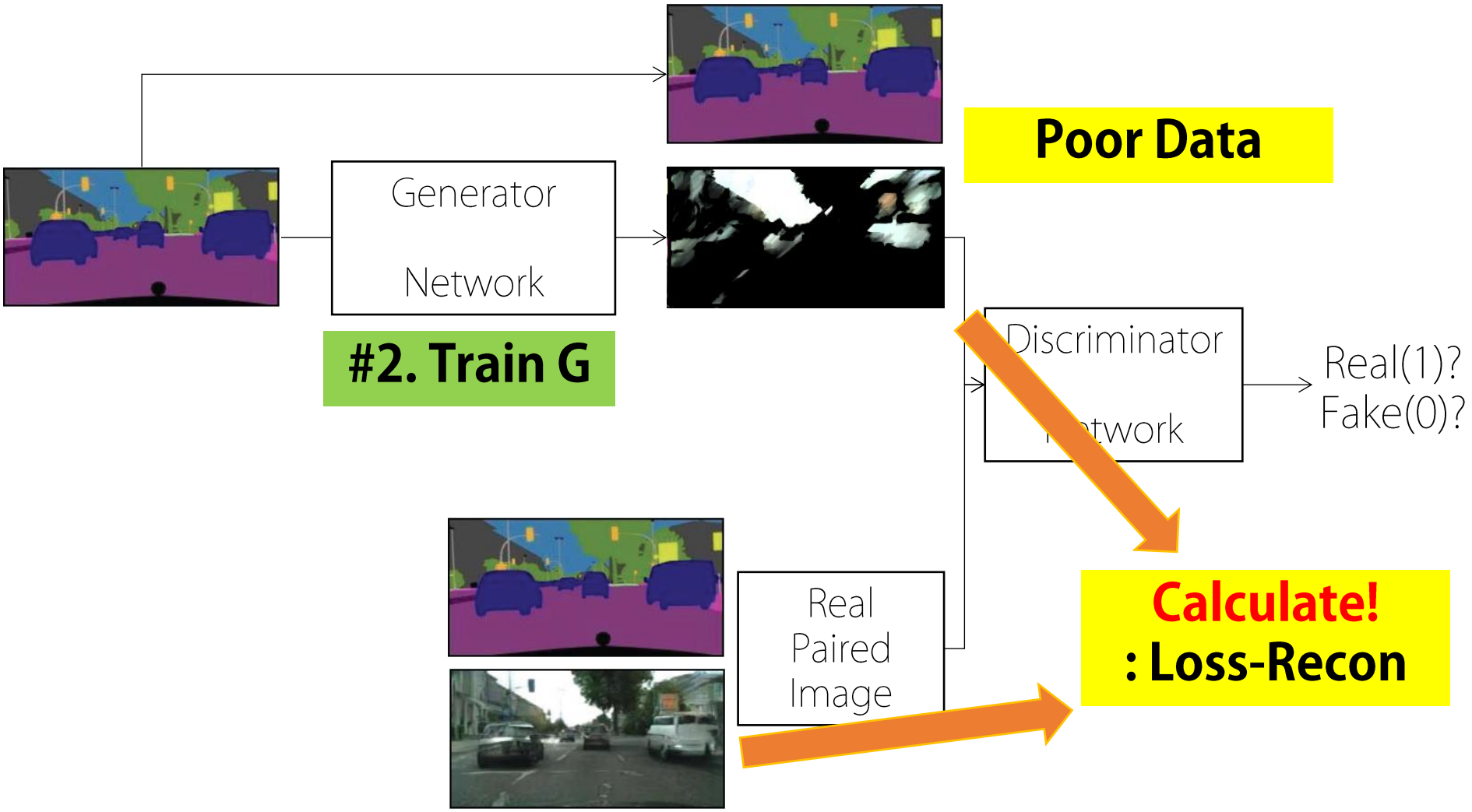


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

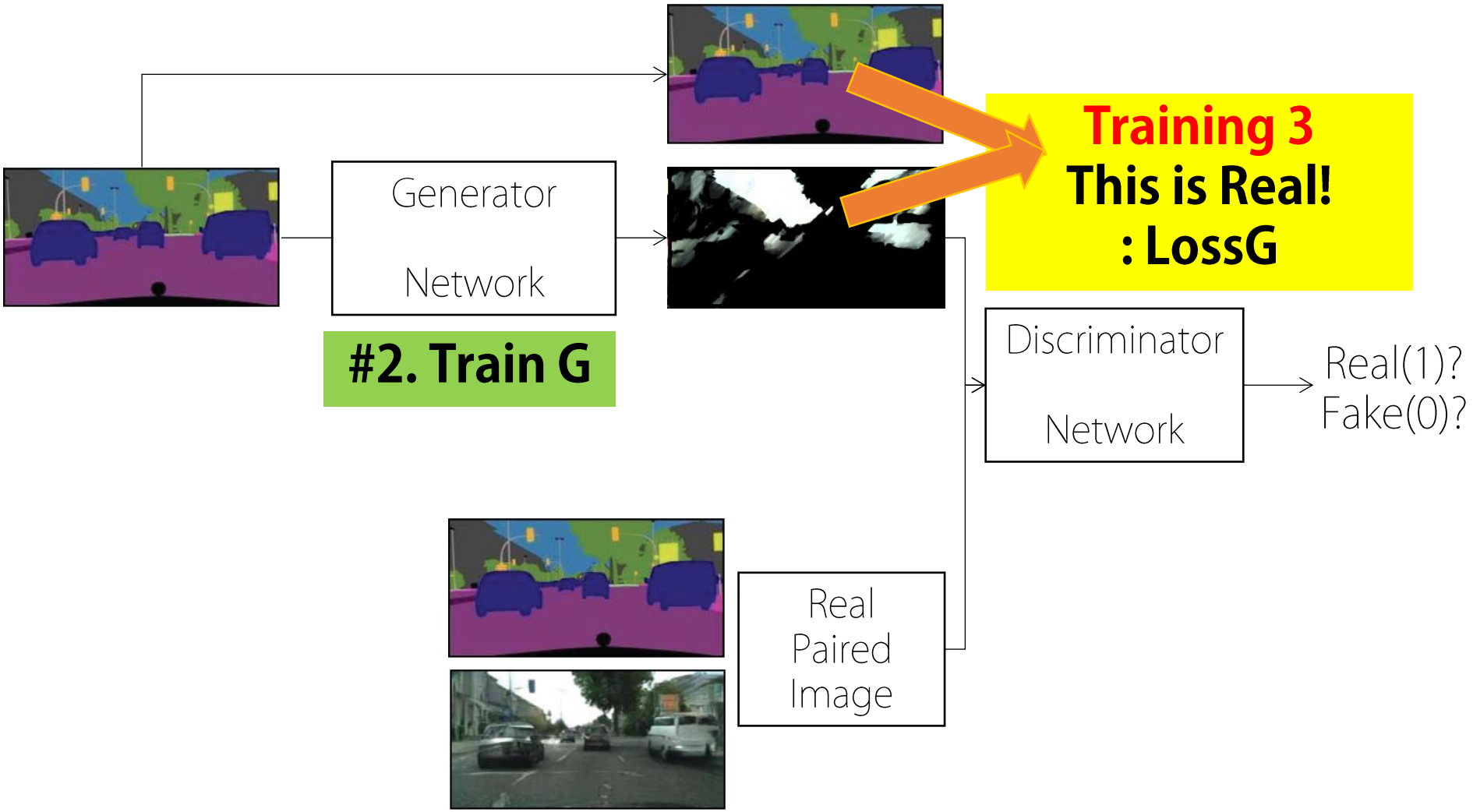


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

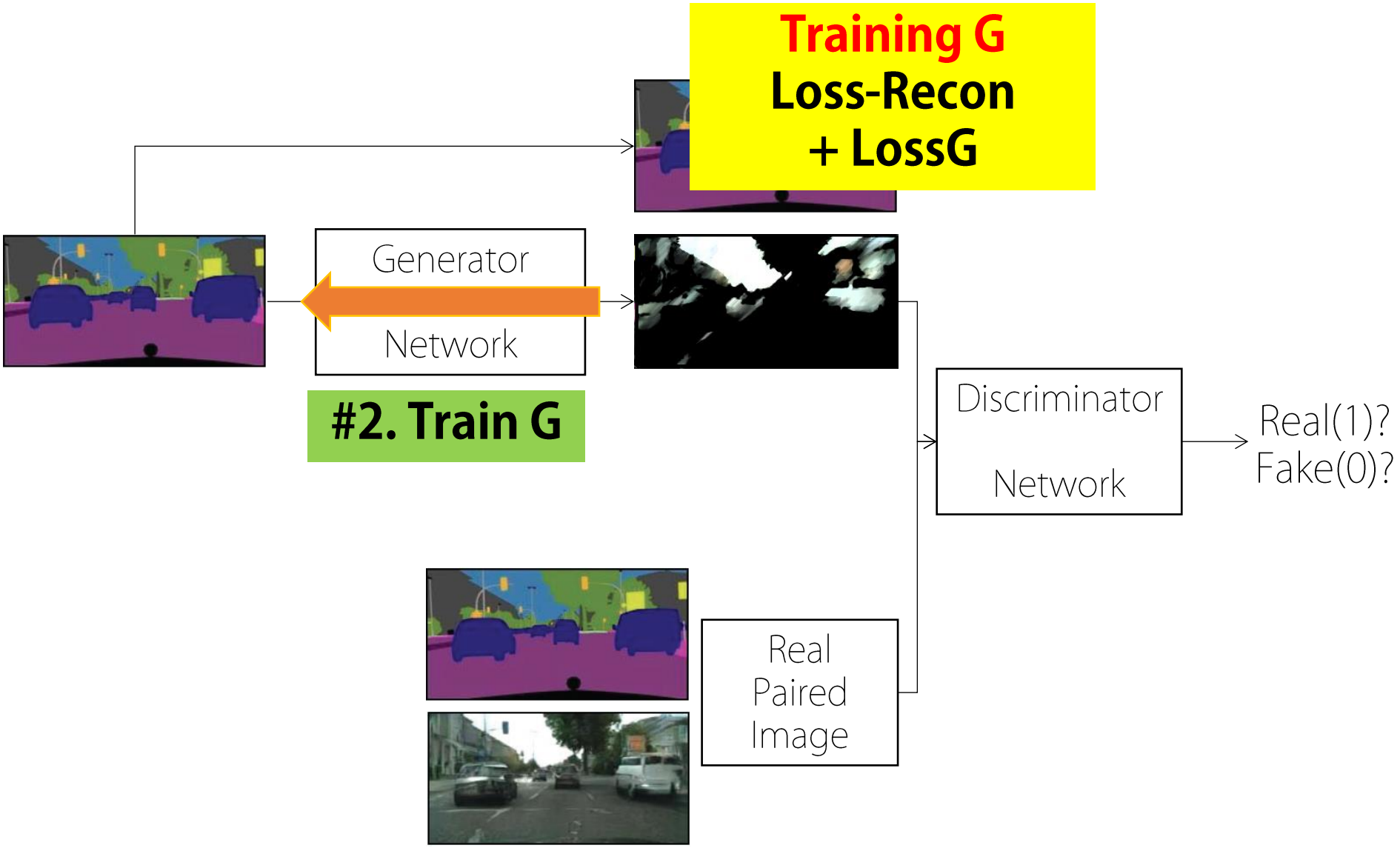


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

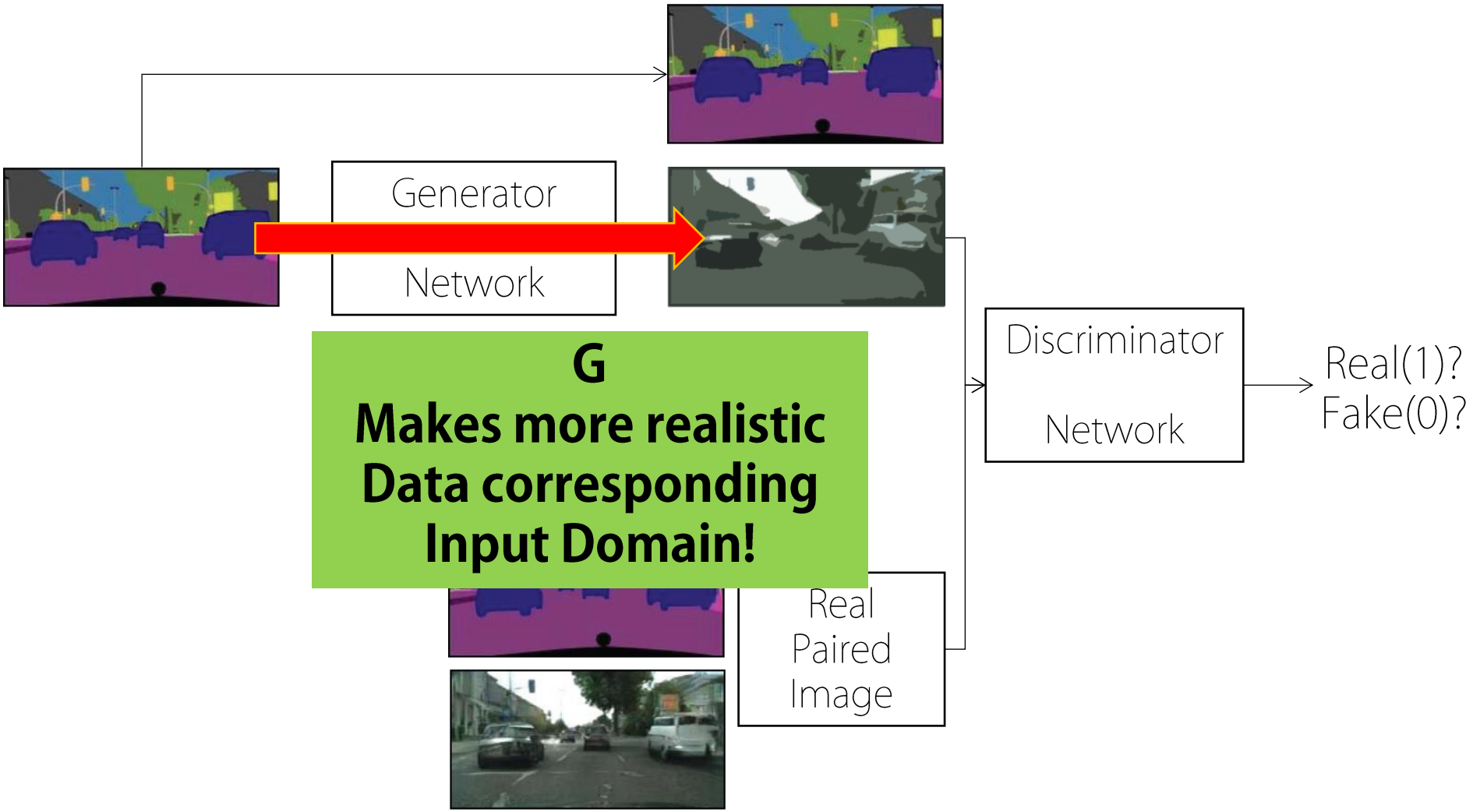
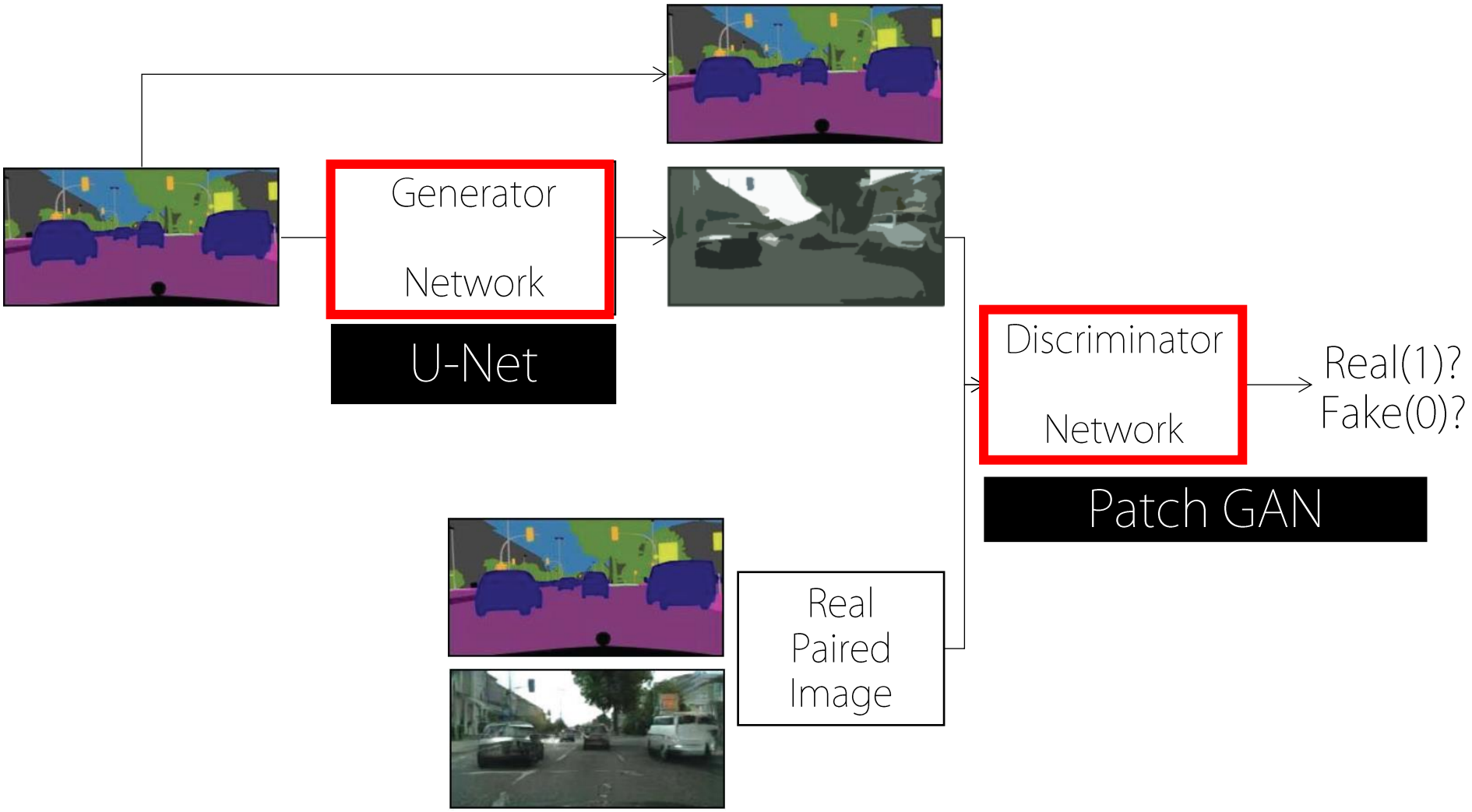
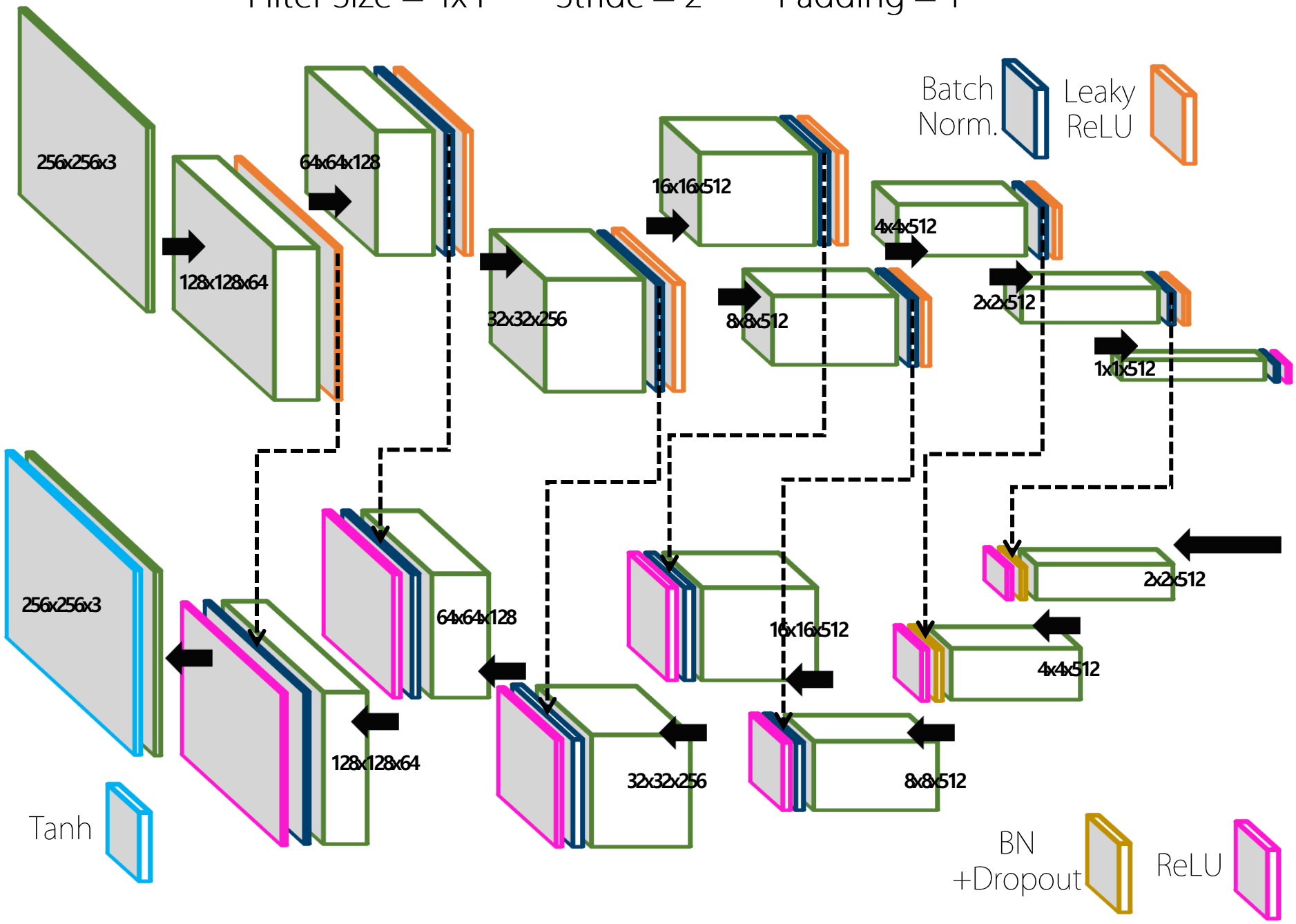
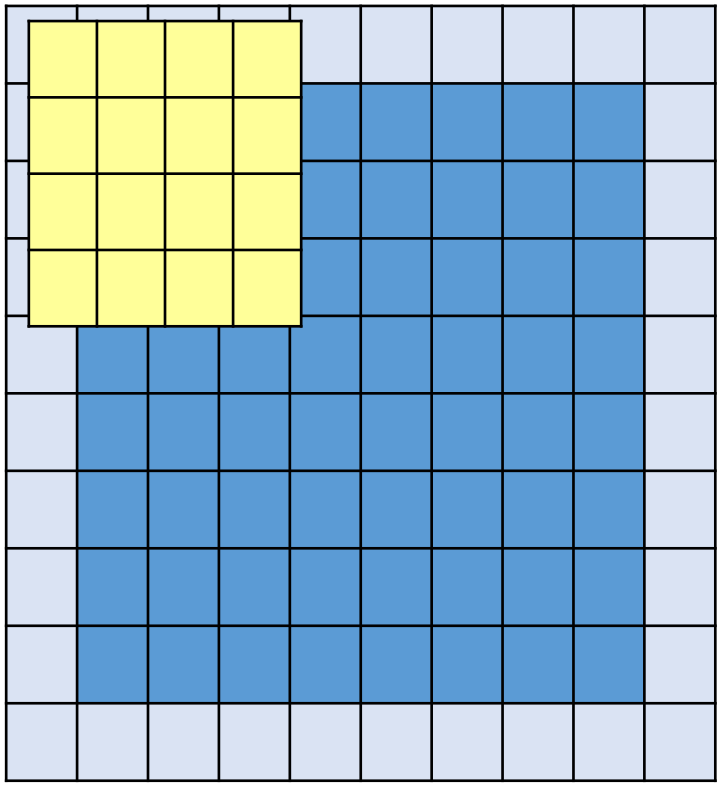


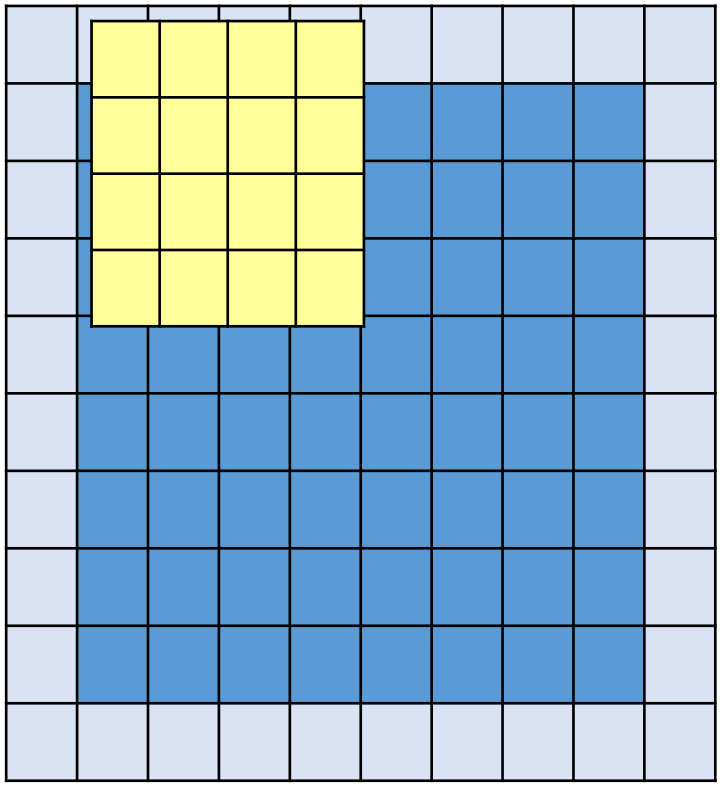
Image to Image Translation

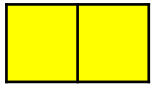
Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

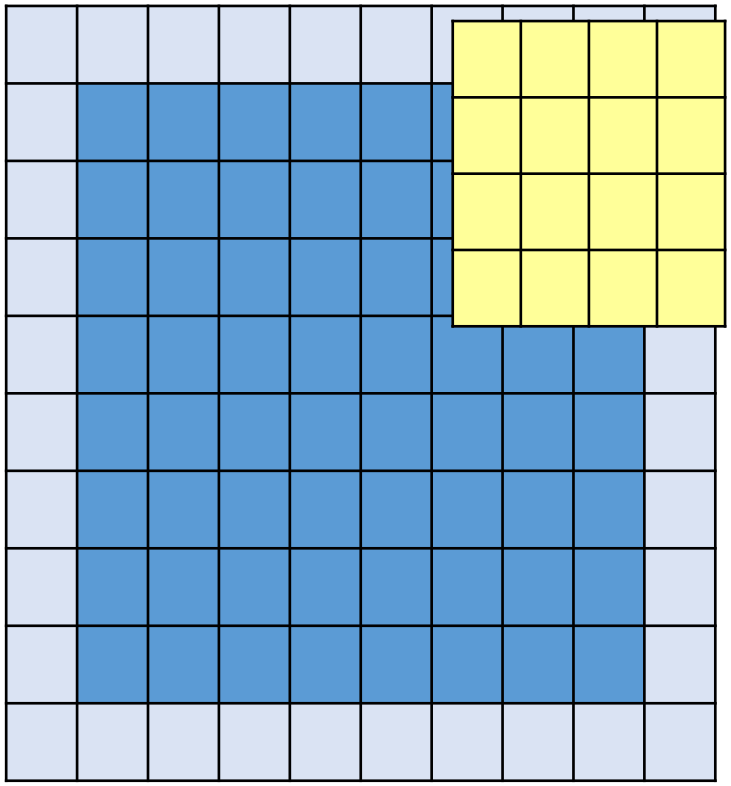


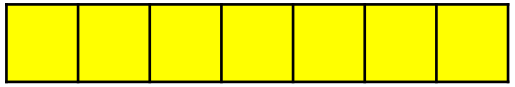

Filter Size = 4x4
Stride = 2
Padding = 1
Batch Norm.
Leaky ReLU
256x256x3
128x128x64
64x64x128
32x32x256
16x16x512
8x8x512
4x4x512
2x2x512
1x1x512
256x256x3
64x64x128
128x128x64
32x32x256
16x16x512
8x8x512
4x4x512
2x2x512
Tanh
BN +Dropout
ReLU

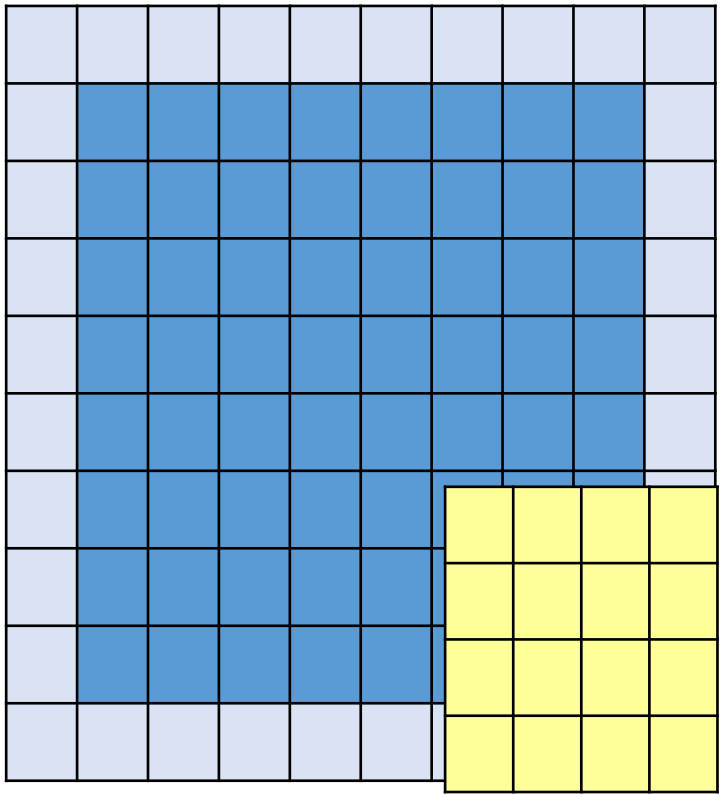


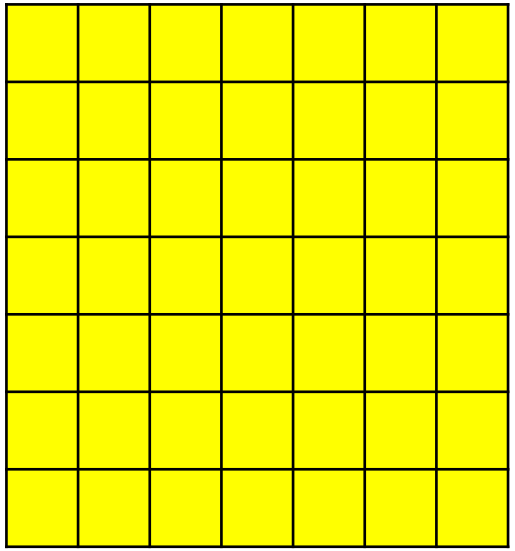












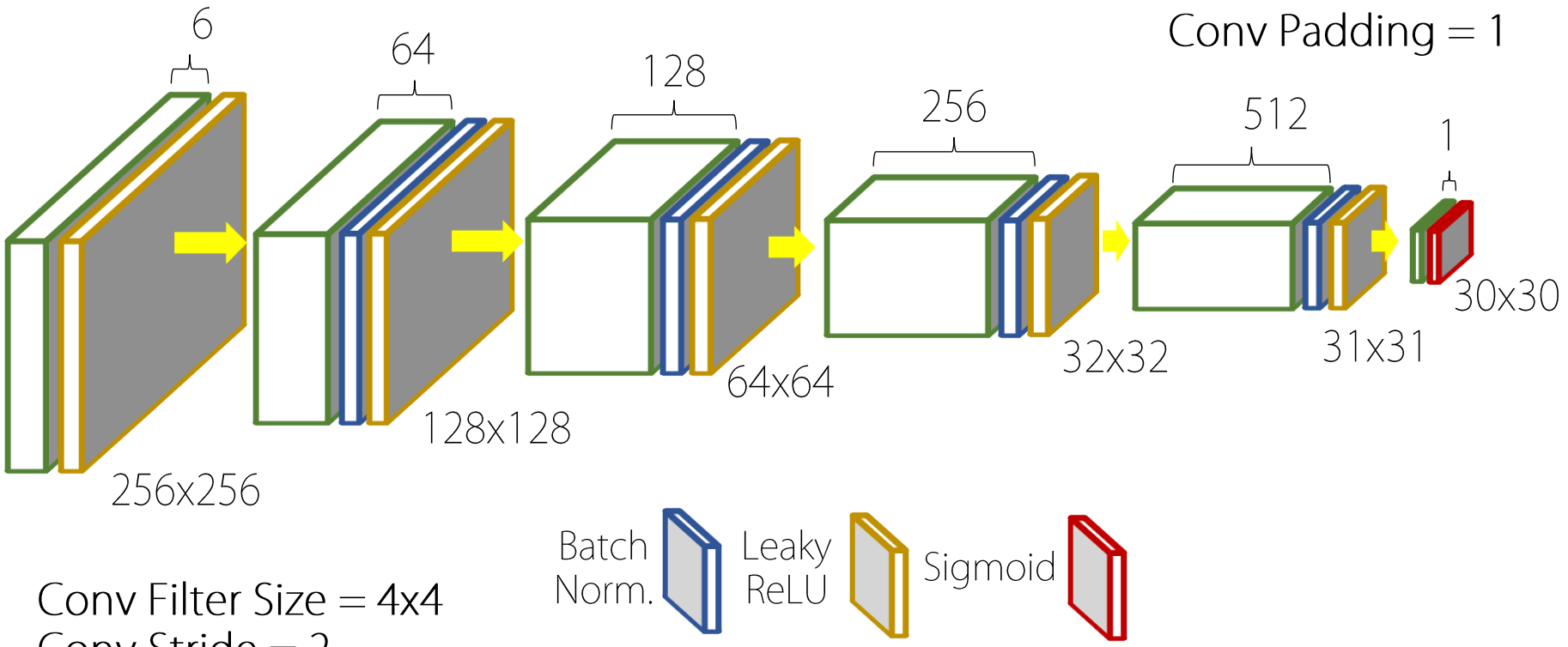

Conv Filter Size = 4x4
Conv Stride = 1
Conv Padding = 1
6
64
128
256
512
1
256x256
128x128
64x64
32x32
31x31
30x30
Batch Norm.
Leaky ReLU
Sigmoid
Conv Filter Size = 4x4
Conv Stride = 2
Conv Padding = 1

Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

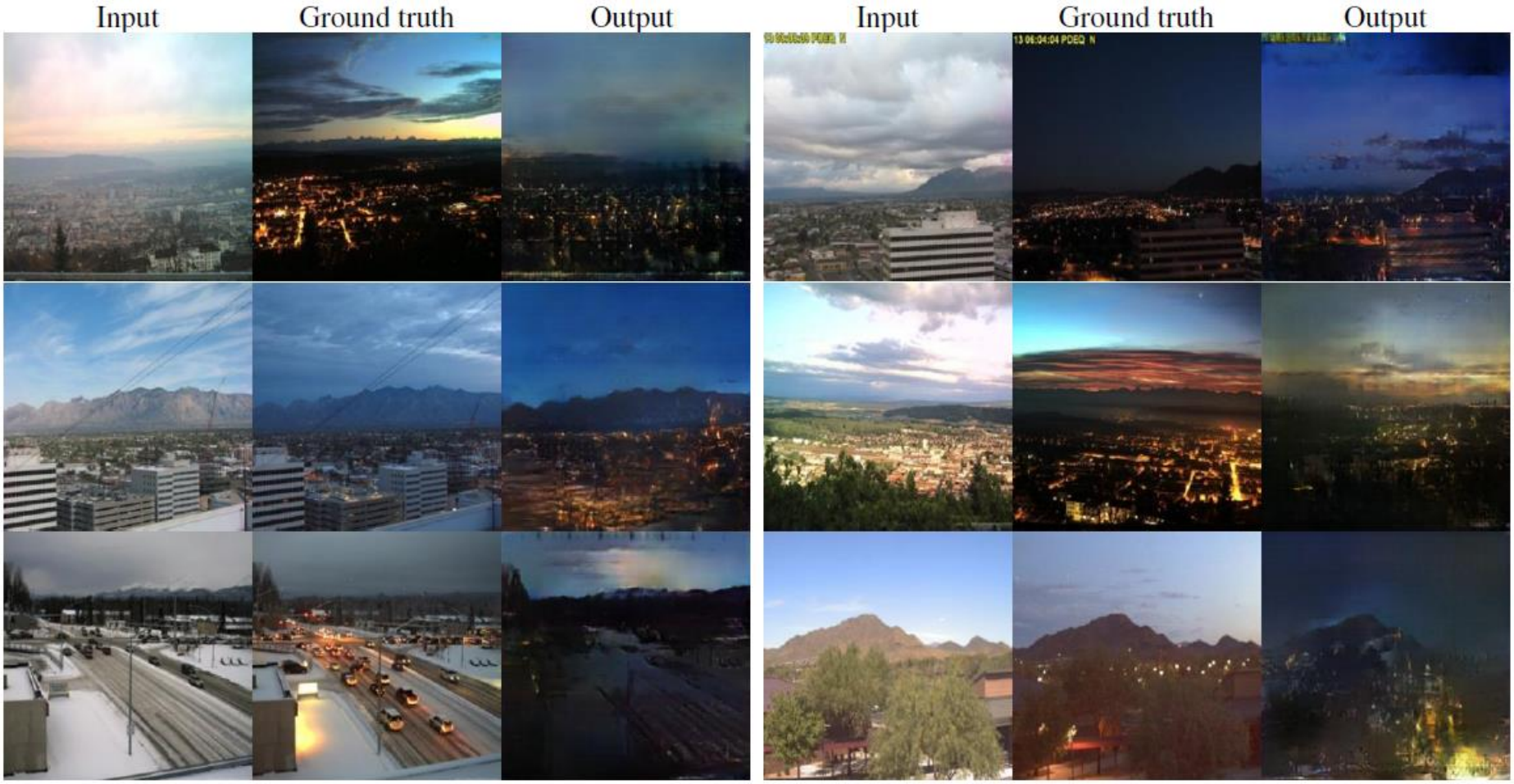


Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

Image to Image Translation

Image to Image Translation with Conditional Adversarial Networks, CVPR 2017

Network Needs Paired Datasets

Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks,
a.k.a CycleGAN, ICCV 2017

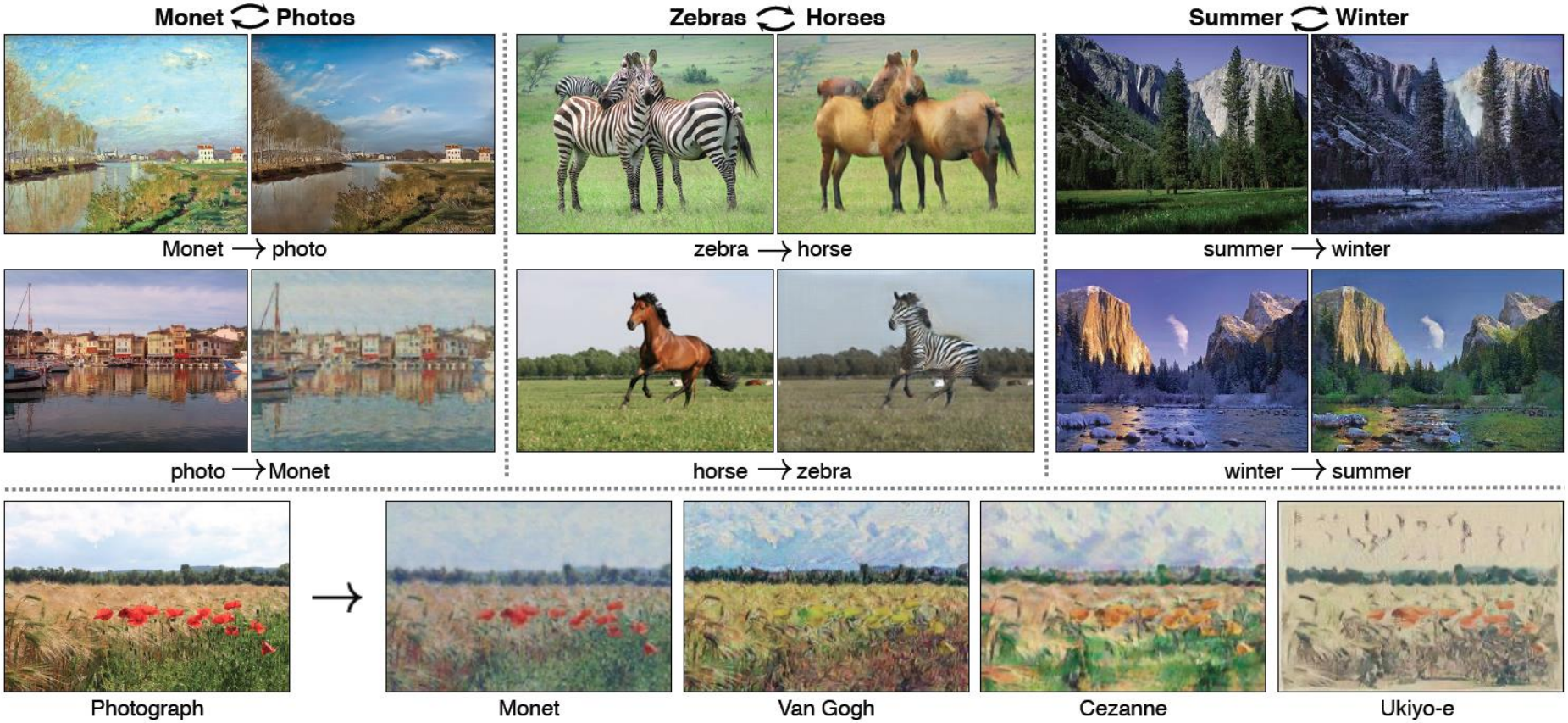


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017

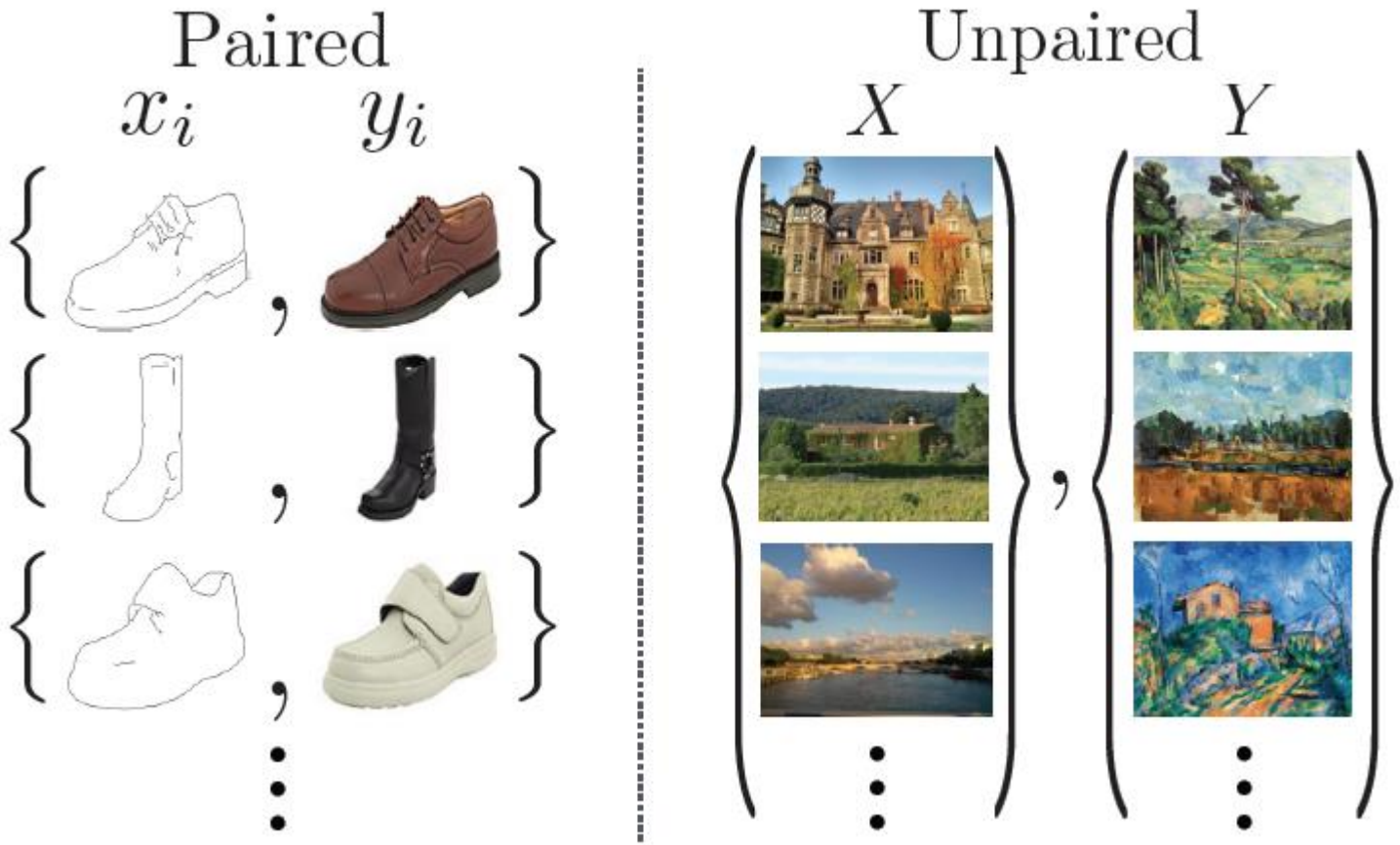
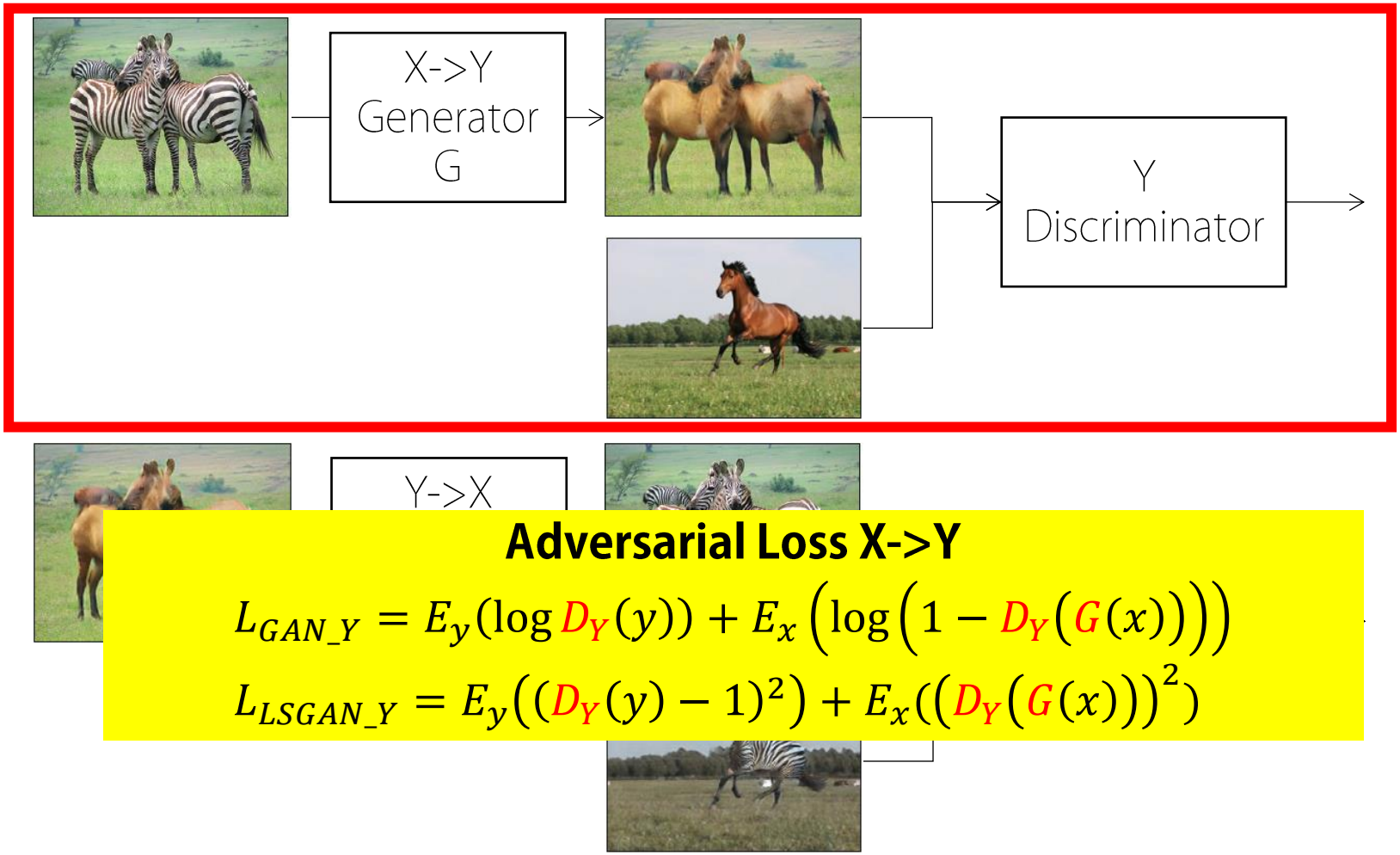


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017



Adversarial Loss X->Y

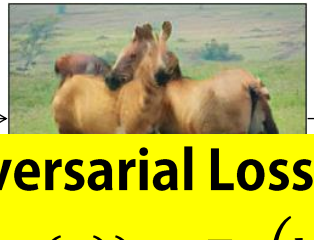
$$L_{GAN\_Y} = E_y(\log D_Y(y)) + E_x\left(\log\left(1 - D_Y\left(G(x)\right)\right)\right)$$

$$L_{LSGAN\_Y} = E_y\left((D_Y(y) - 1)^2\right) + E_x\left(\left(D_Y\left(G(x)\right)\right)^2\right)$$

Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017

X->Y Generator



Adversarial Loss Y->X

$$L_{GAN\_X} = E_x(\log D_X(x)) + E_y\left(\log\left(1 - D_X(F(y))\right)\right)$$

$$L_{LSGAN\_X} = E_x\left((D_X(x) - 1)^2\right) + E_y\left(\left(D_X(F(y))\right)^2\right)$$

Y->X Generator F

X Discriminator

Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017

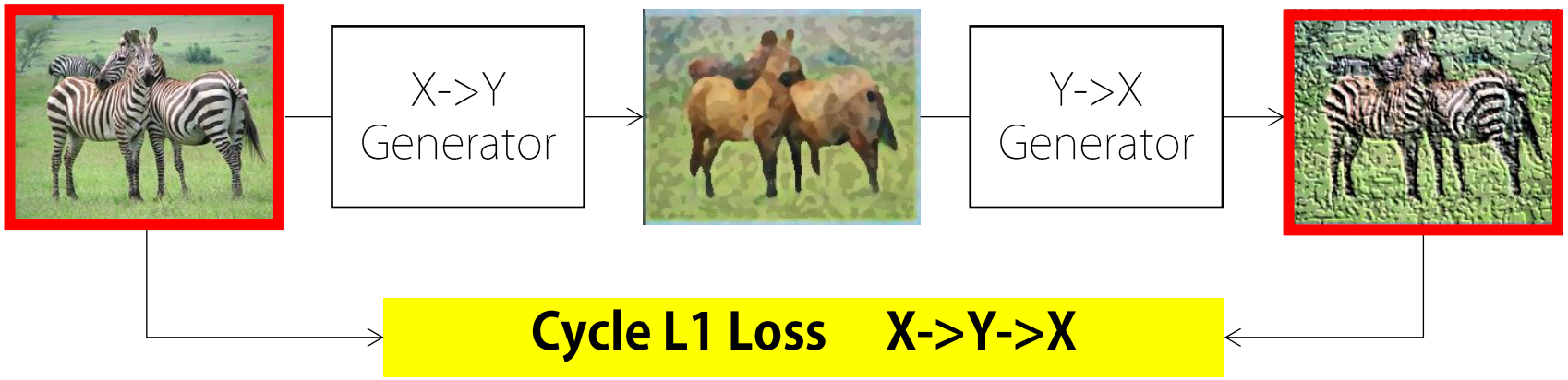
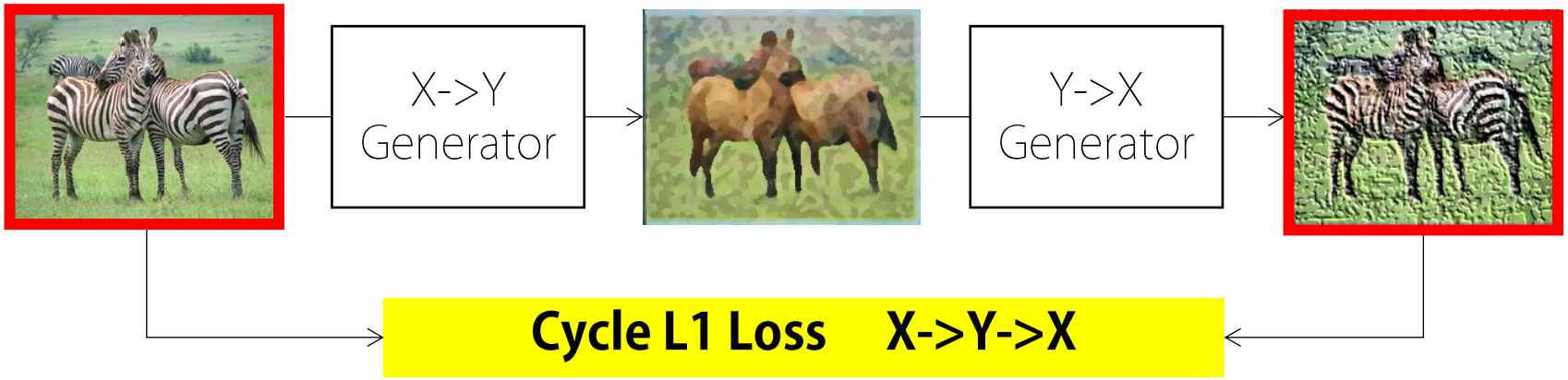


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017



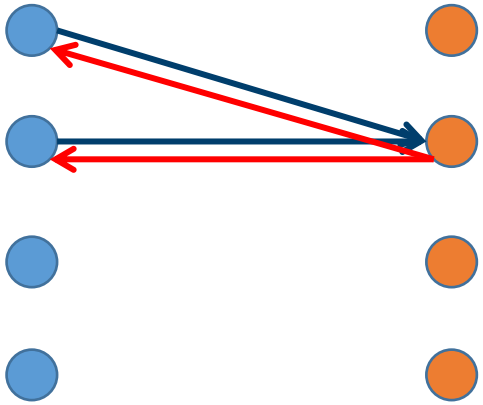
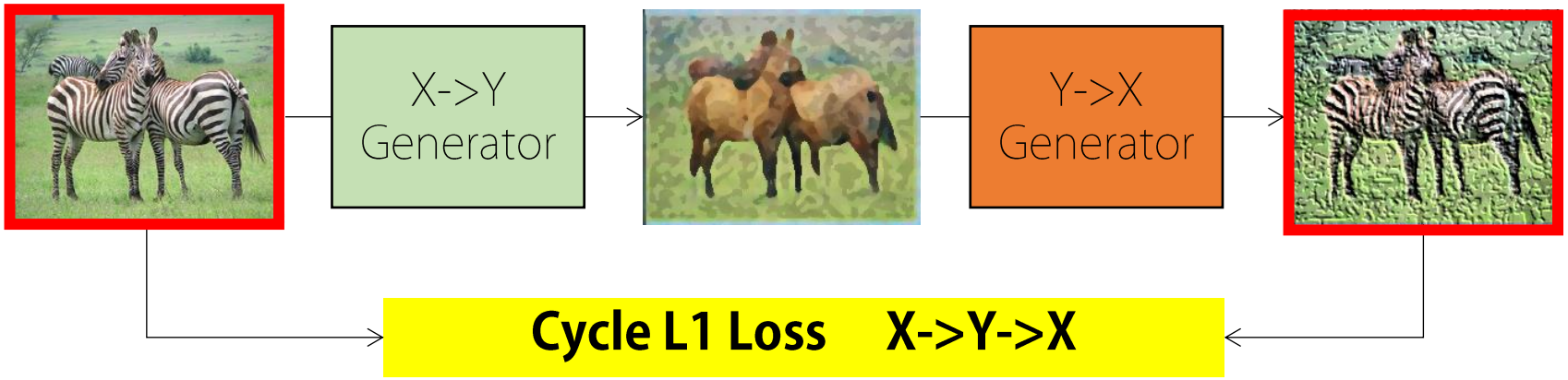


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017



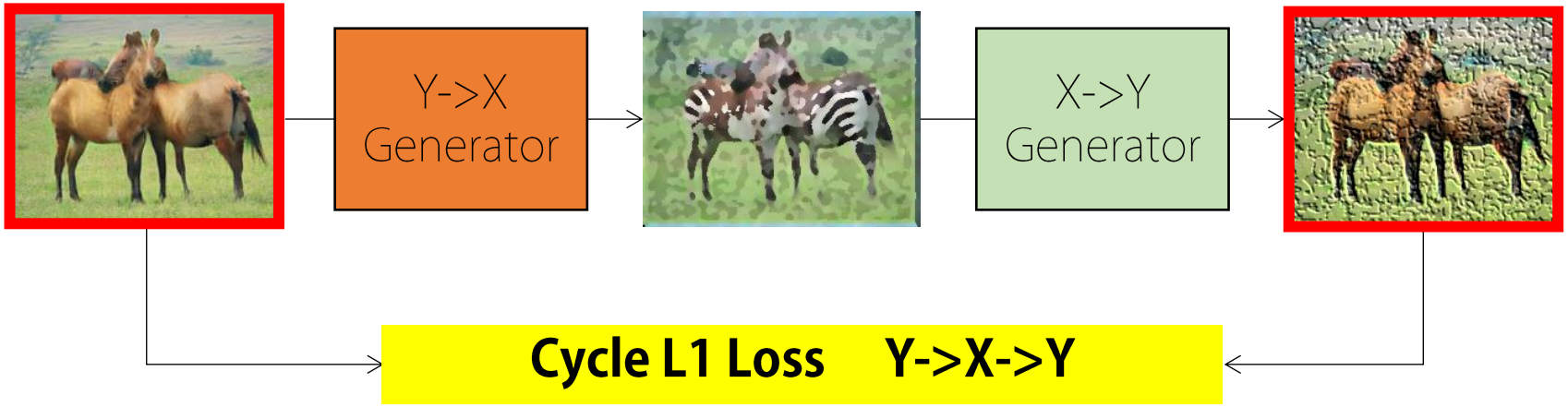
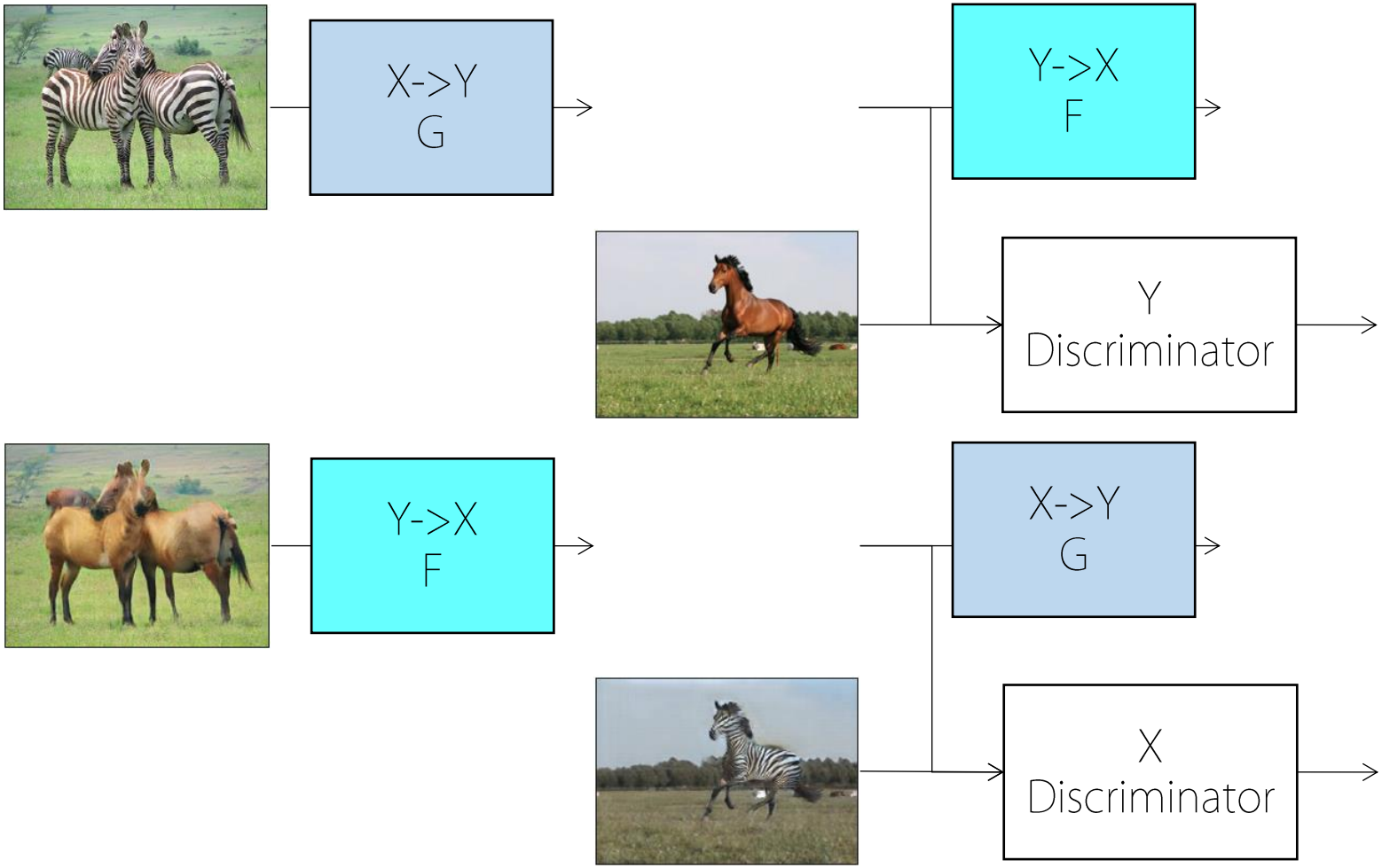
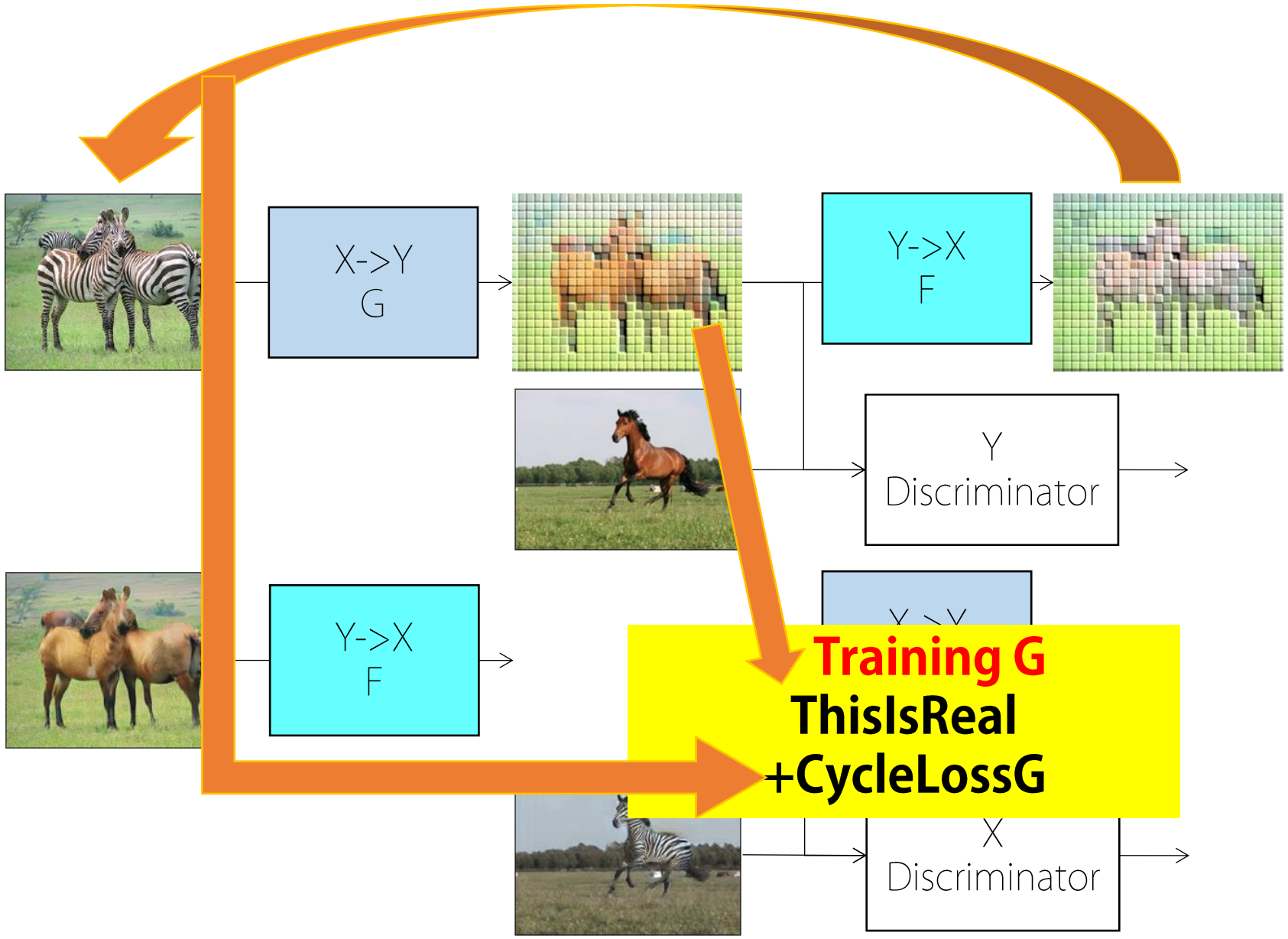
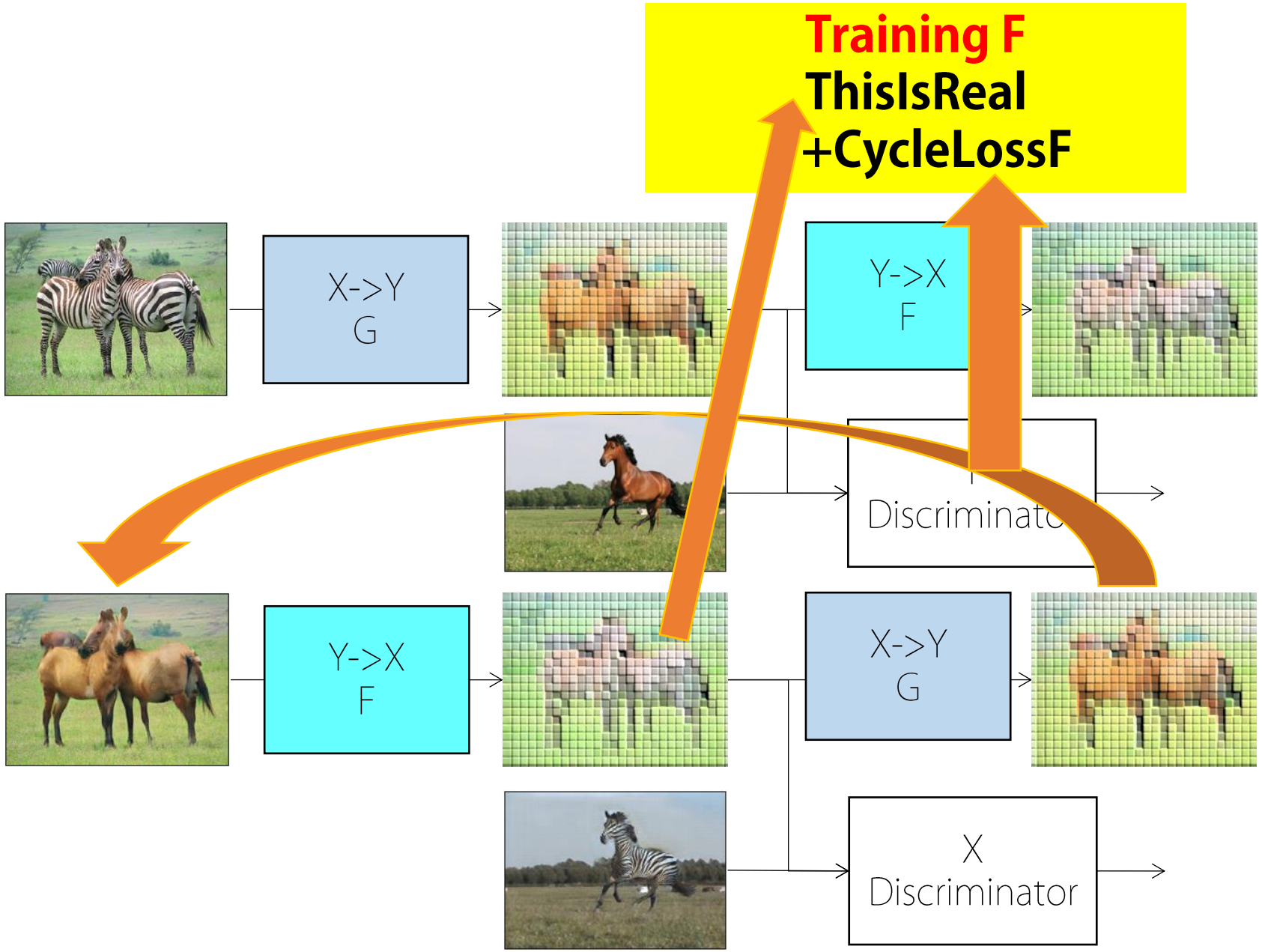


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017



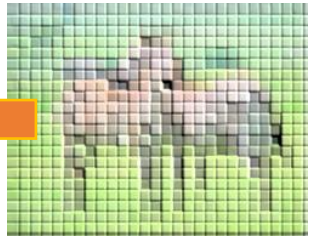

X->Y
G
Y->X
F
Y
Discriminator
Y->X
F
Training G
ThisIsReal
+CycleLossG
X
Discriminator


Training F
ThisIsReal
+CycleLossF
X->Y
G
Y->X
F
Discriminato
Y->X
F
X->Y
G
X
Discriminator

Train G+F
LossG+LossF

X->Y
G

Y->X
F

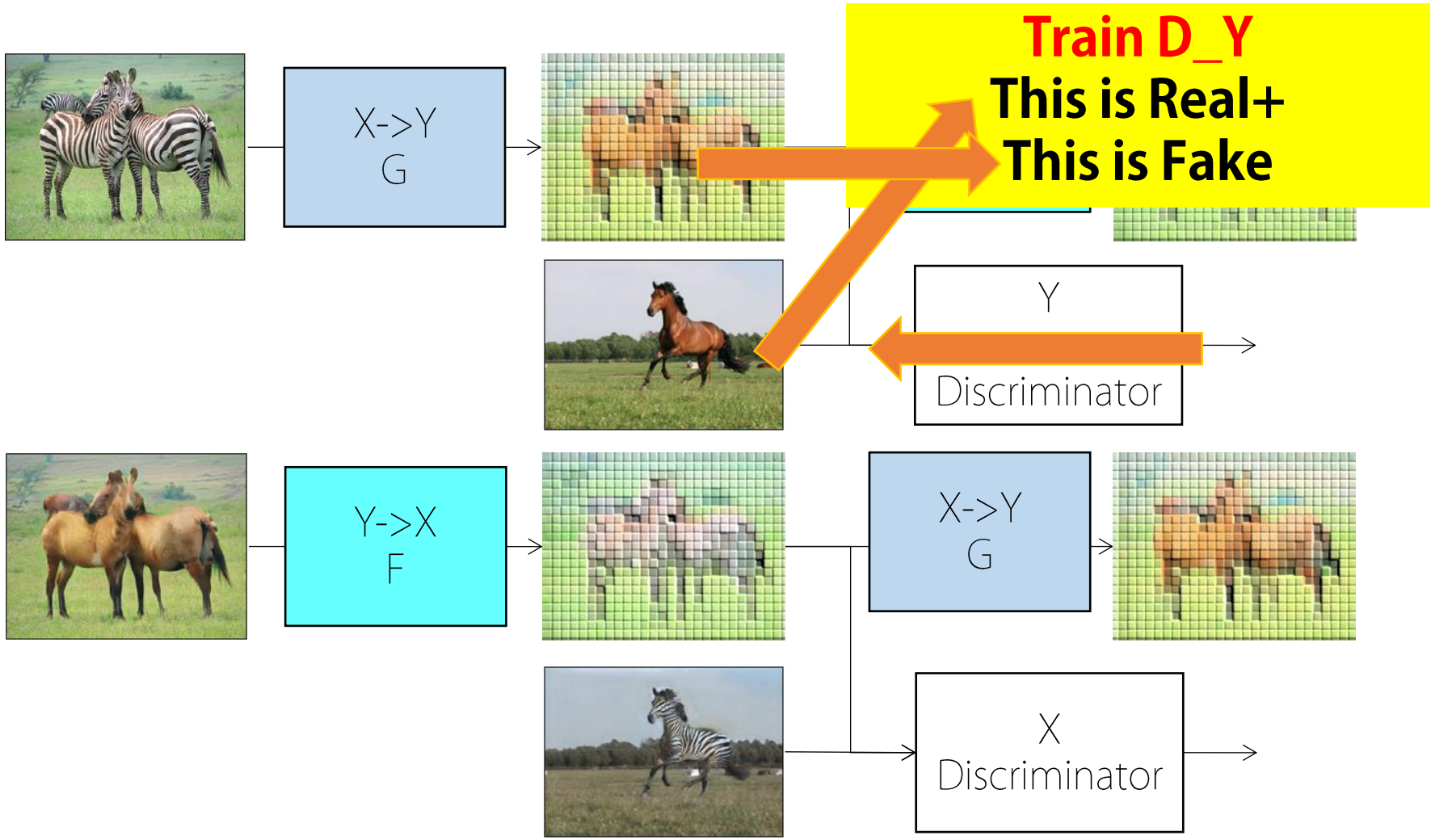


Y
Discriminator

Y->X
F

X->Y
G

X
Discriminator


Train D_Y
This is Real+
This is Fake
X->Y
G
Y
Discriminator
Y->X
F
X->Y
G
X
Discriminator

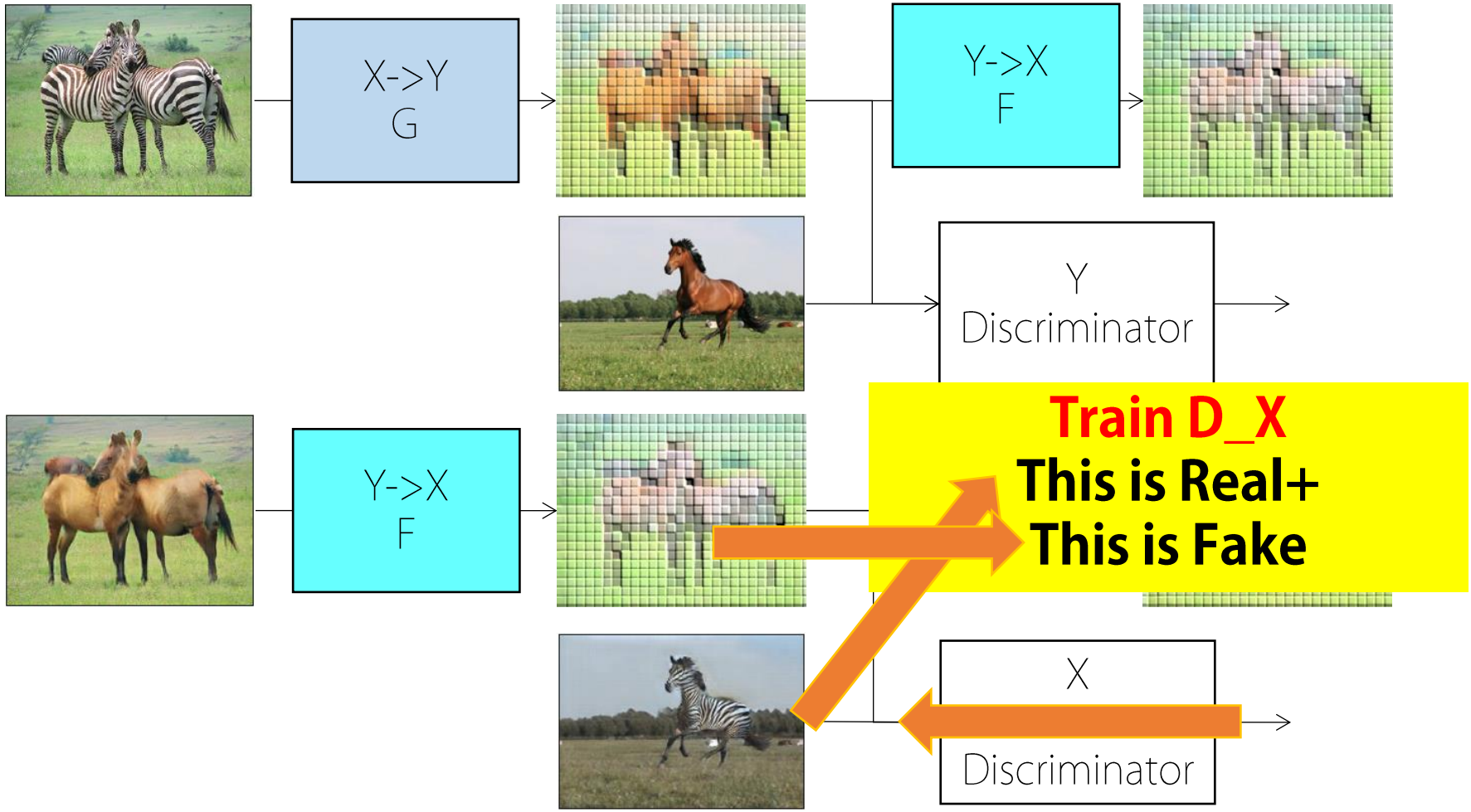

X->Y
G
Y->X
F
Y
Discriminator
Y->X
F
Train D_X
This is Real+
This is Fake
X
Discriminator

Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017

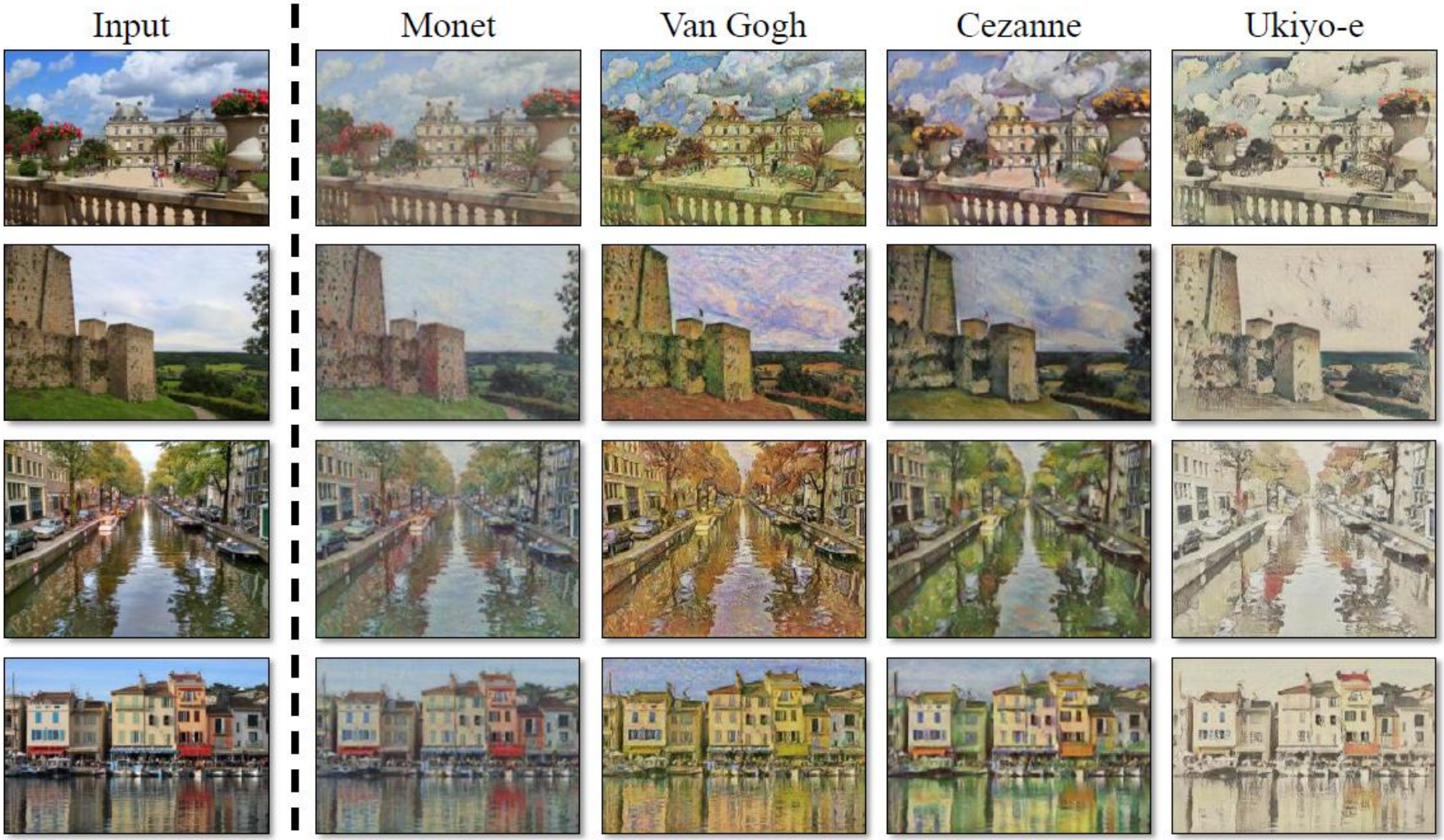


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017

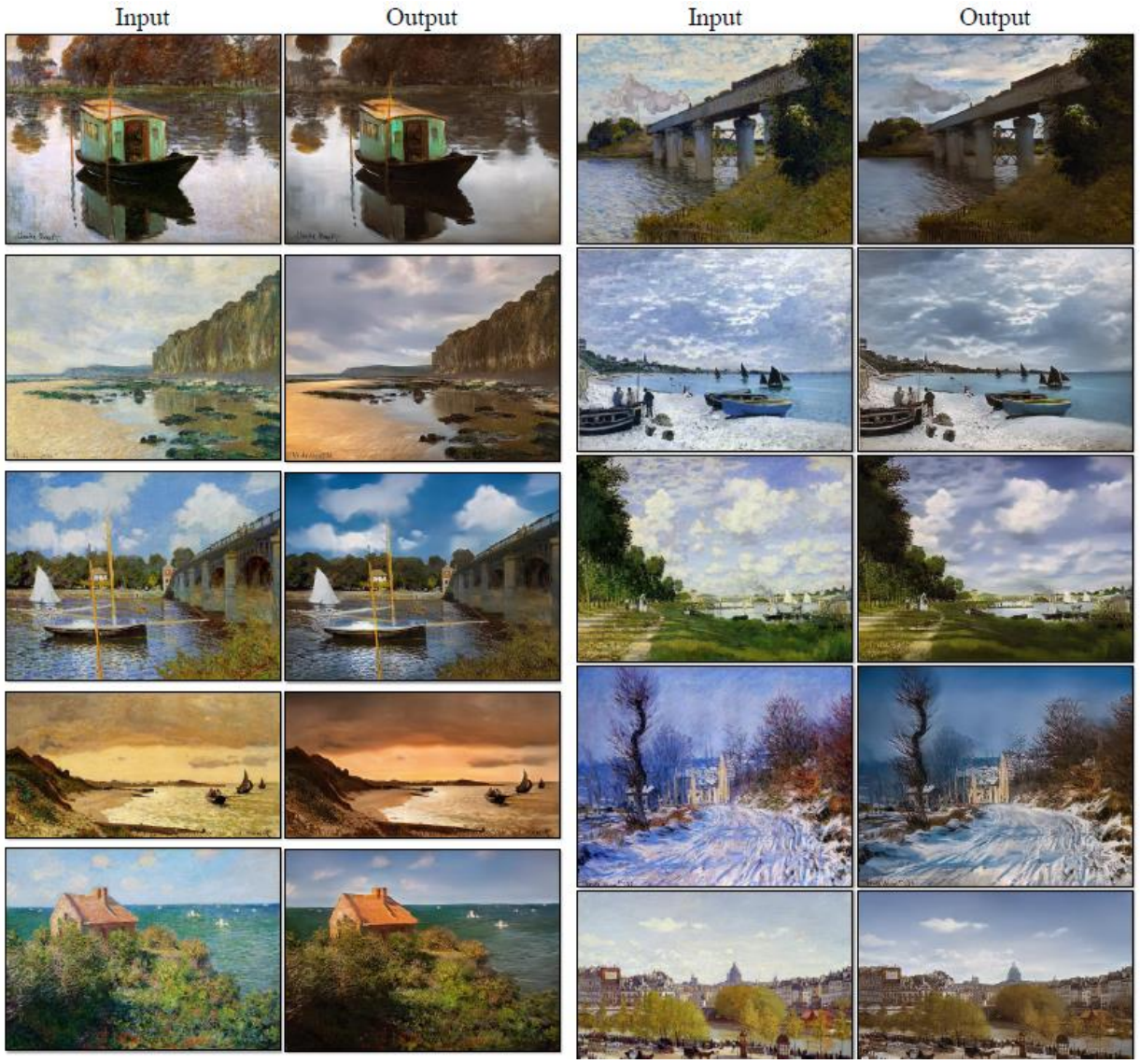


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017

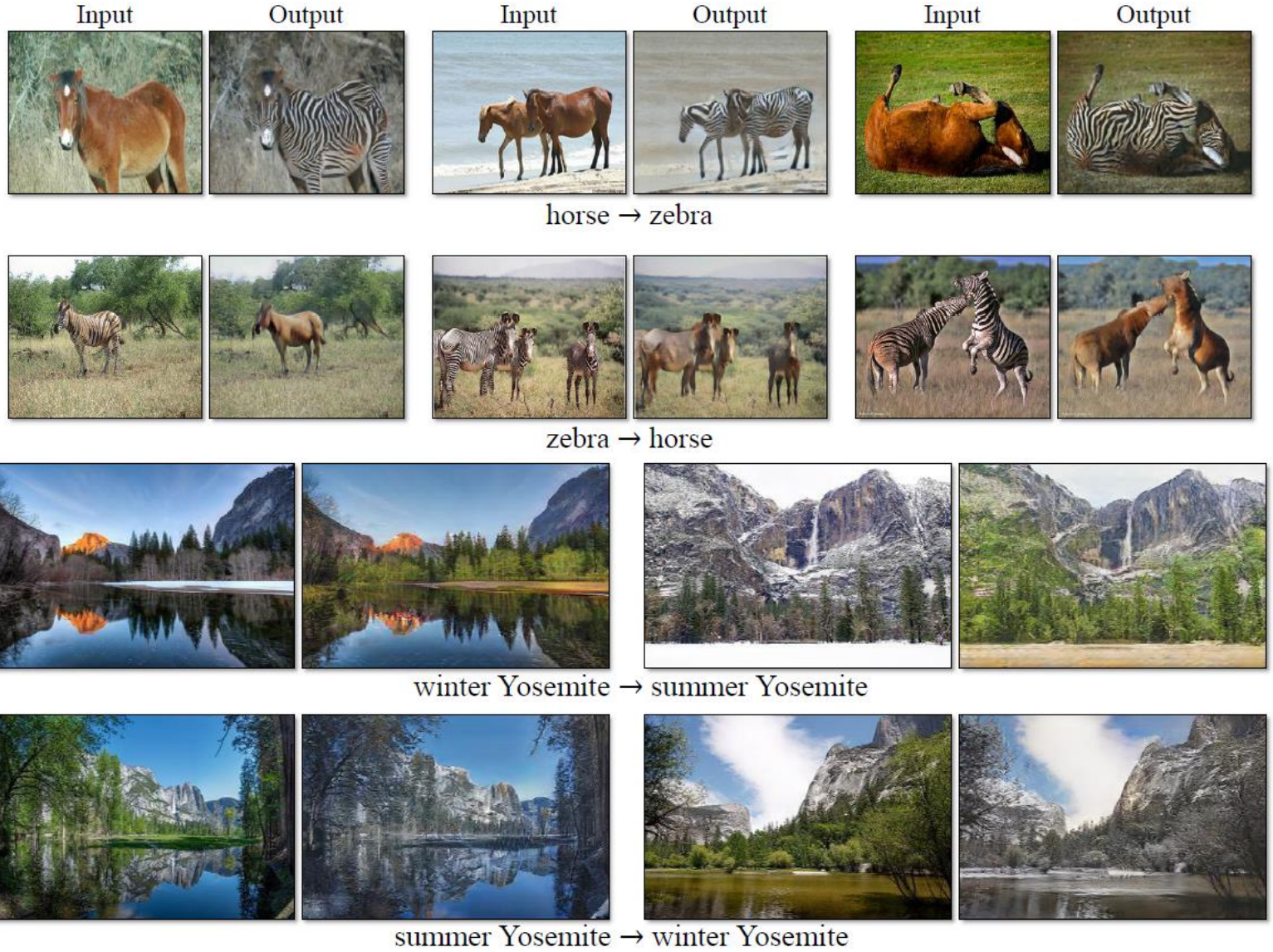
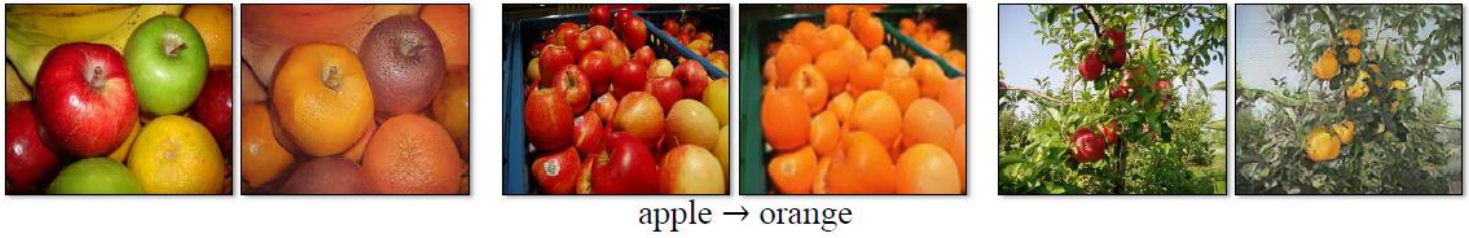


Image to Image Translation 2

Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017



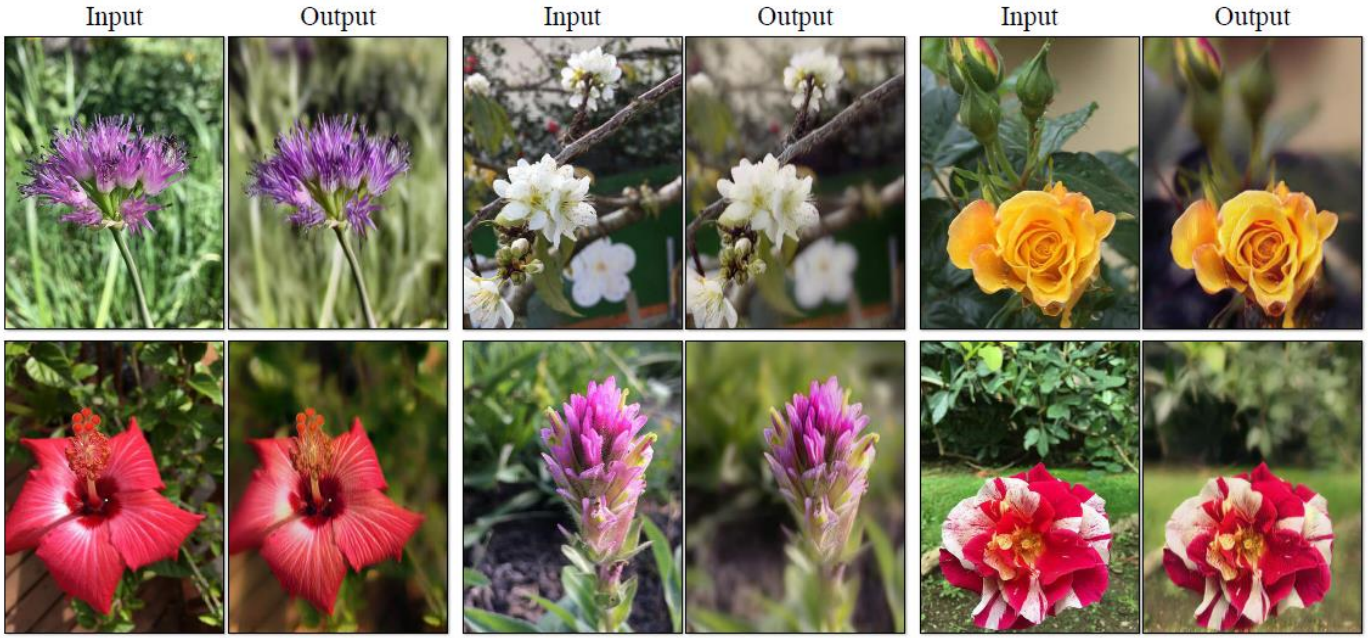


Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks,
a.k.a DiscoGAN, ICML 2017

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

(c) Shoe images (input) & **Generated** handbag images (output)

Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks,
a.k.a DiscoGAN, ICML 2017

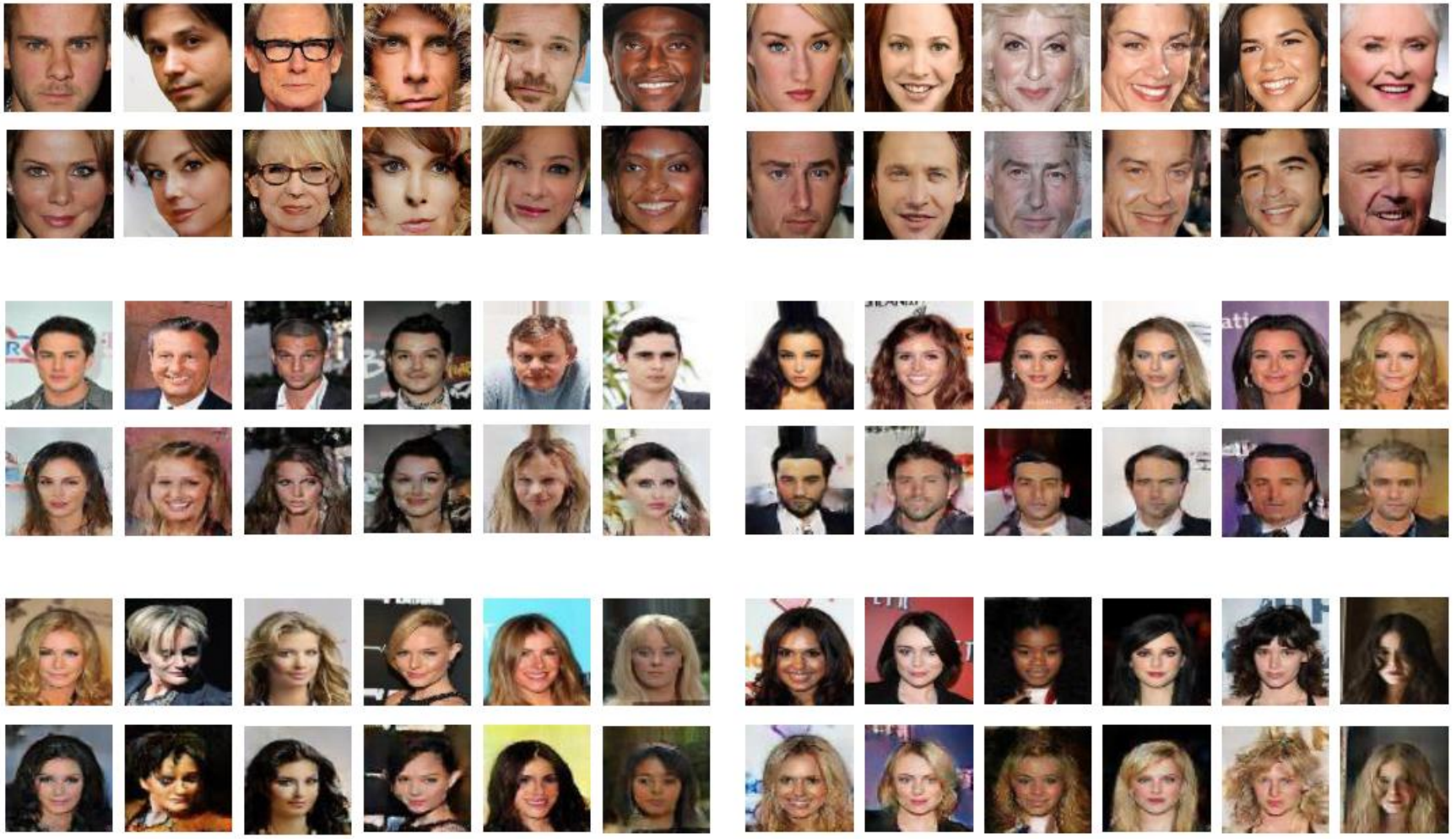


Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks,
a.k.a DiscoGAN, ICML 2017

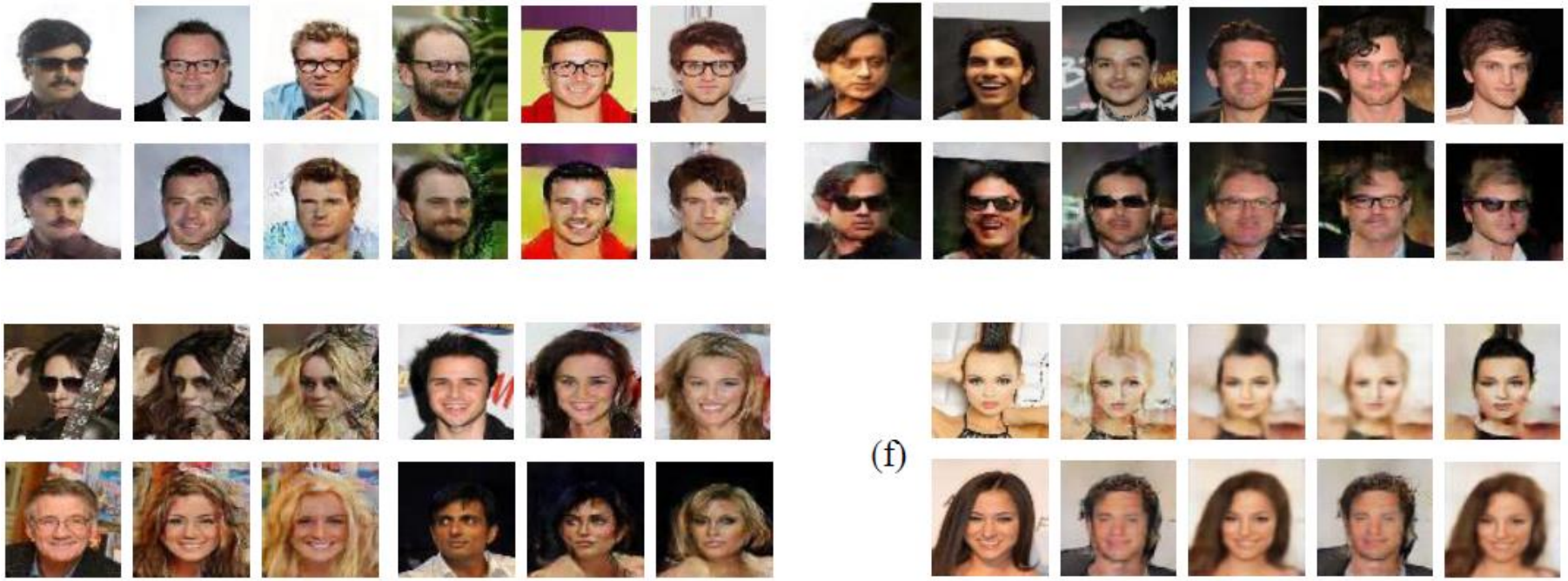
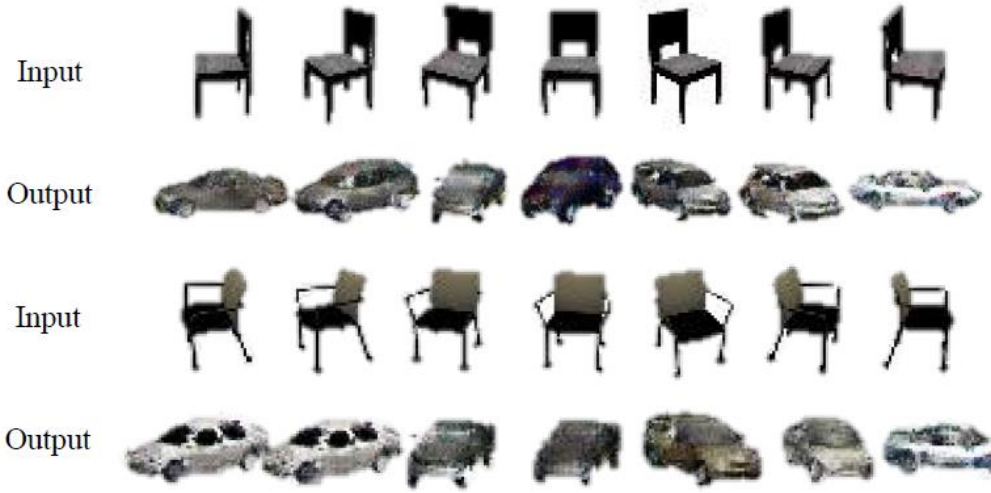


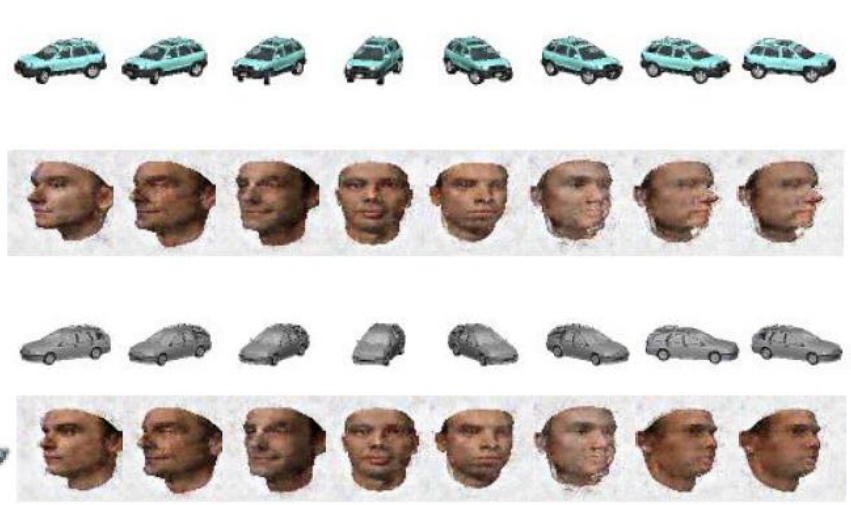
Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks,

a.k.a DiscoGAN, ICML 2017



(a) Chair to Car



(b) Car to Face

(a)

(b)

(c)

Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017

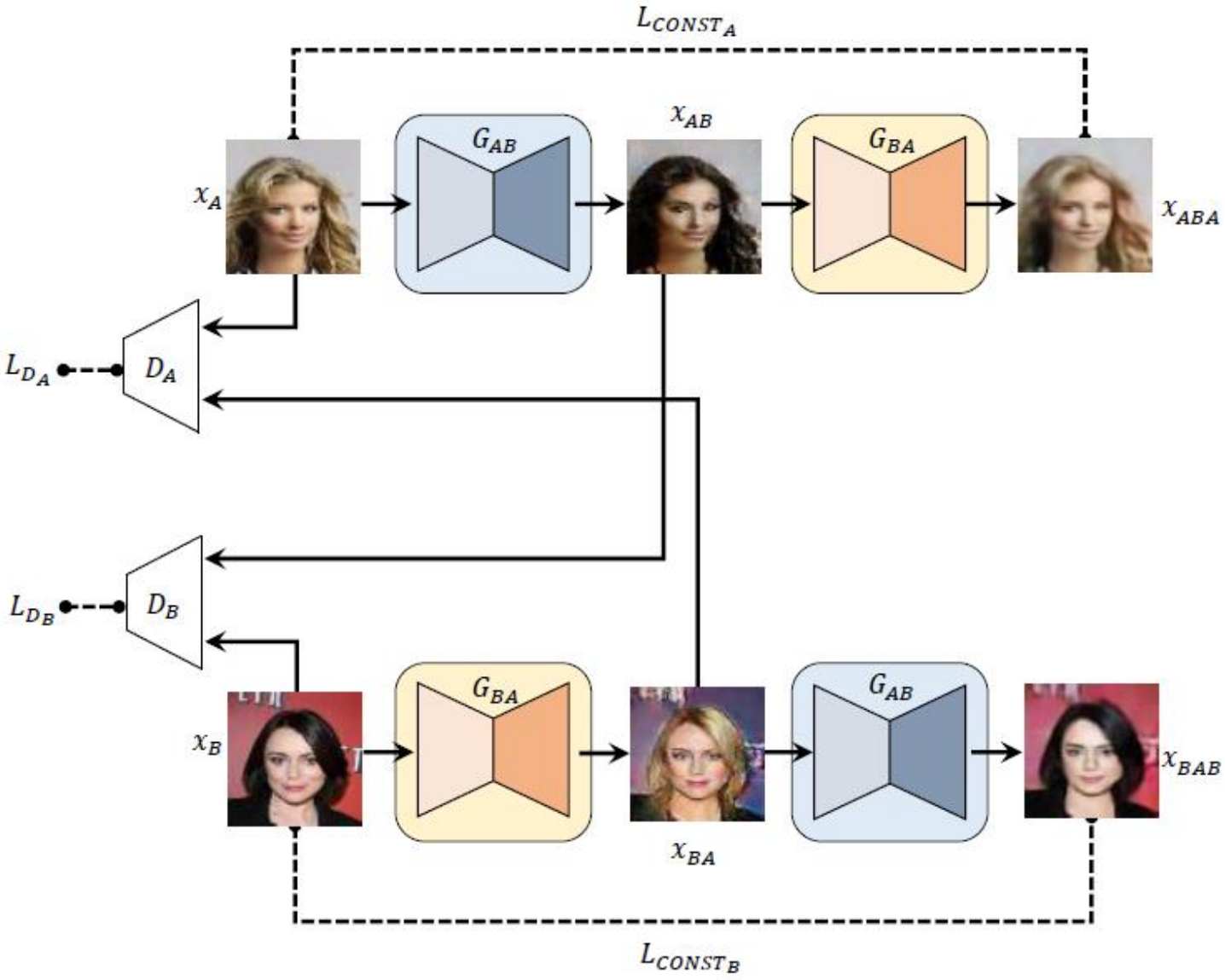


Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017

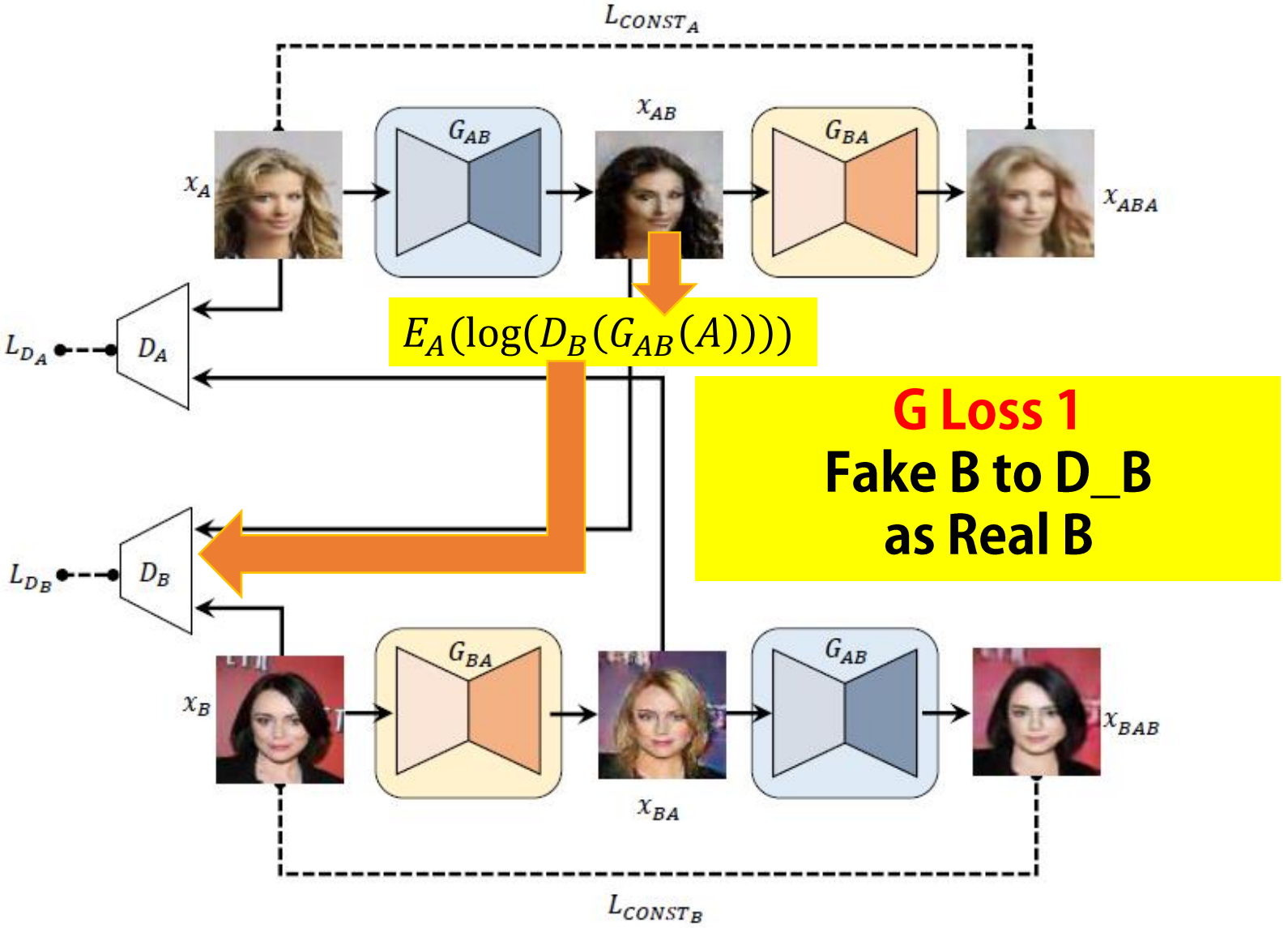


Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017

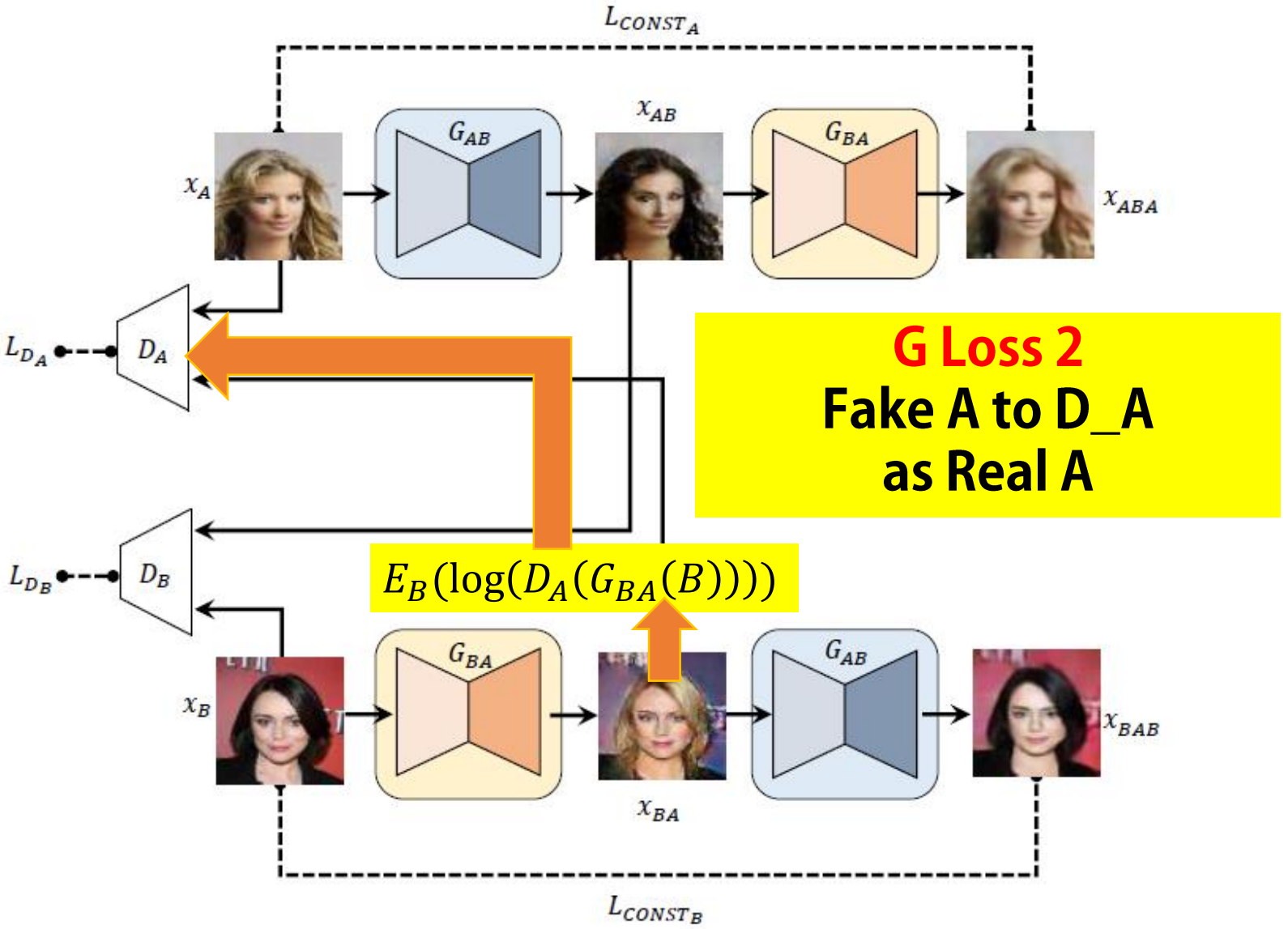


Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017

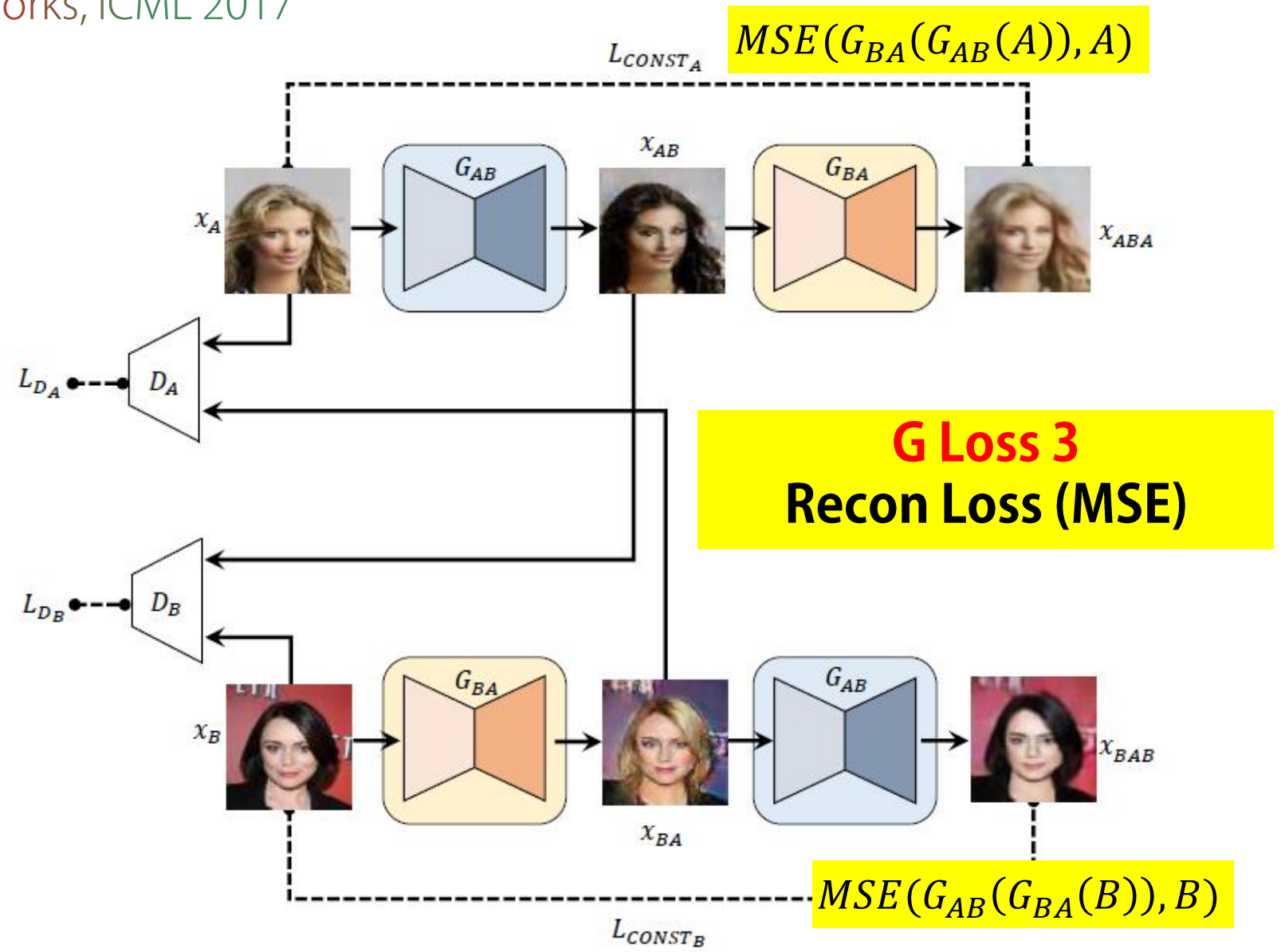


Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017

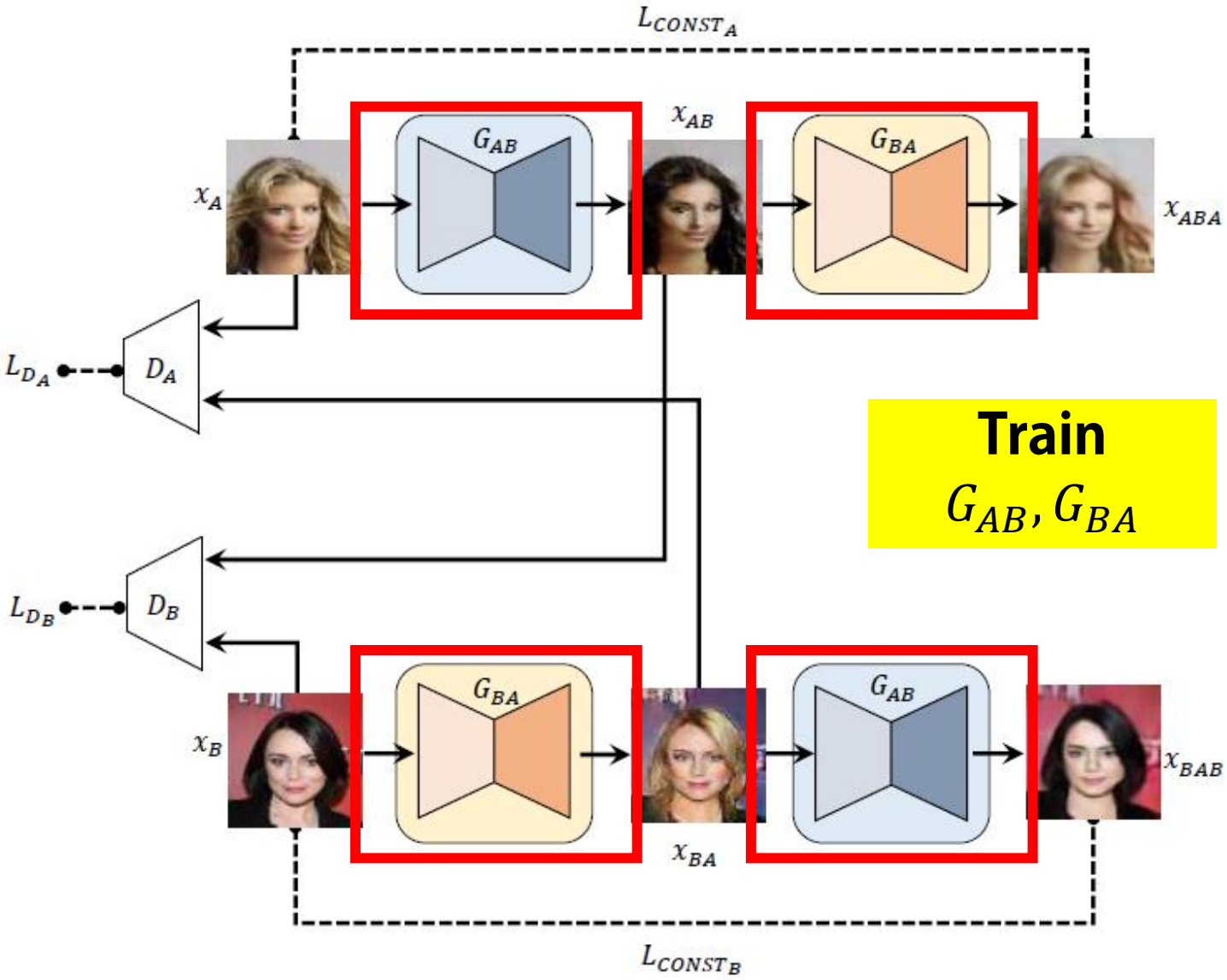
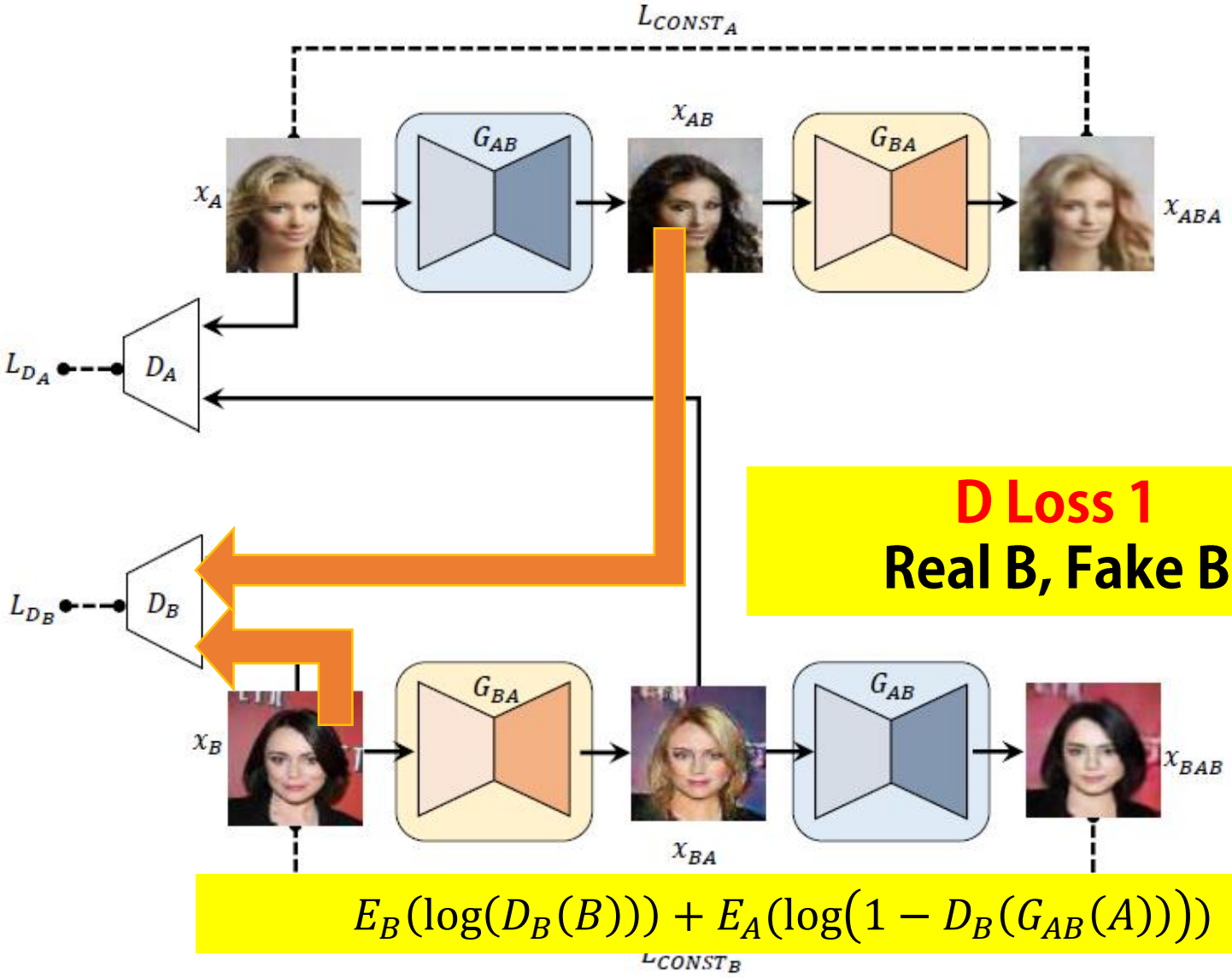


Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017

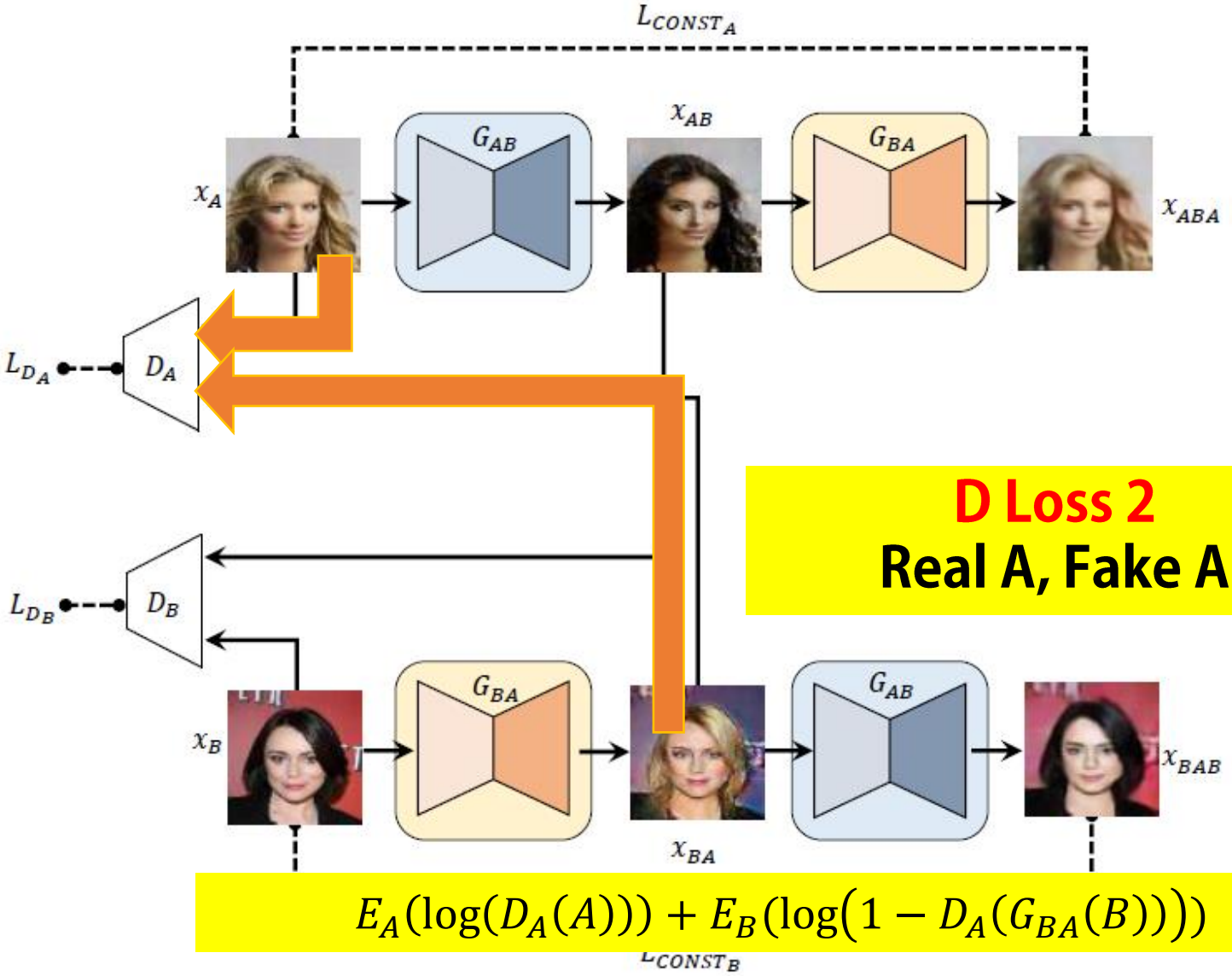


D Loss 1
Real B, Fake B

$$E_B(\log(D_B(B))) + E_A(\log(1 - D_B(G_{AB}(A))))$$

Image to Image Translation 3

Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017



D Loss 2

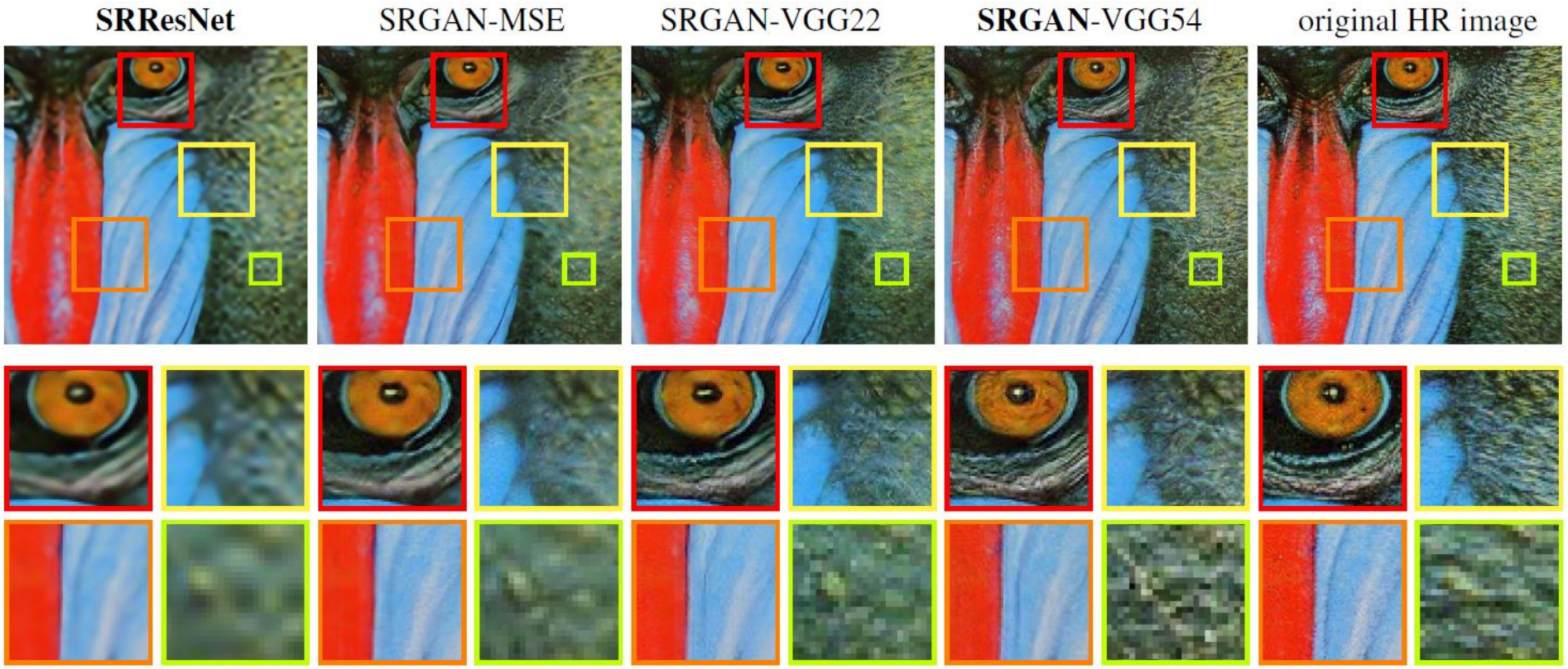
Real A, Fake A

$$E_A(\log(D_A(A))) + E_B(\log(1 - D_A(G_{BA}(B))))$$

Single Image Super Resolution

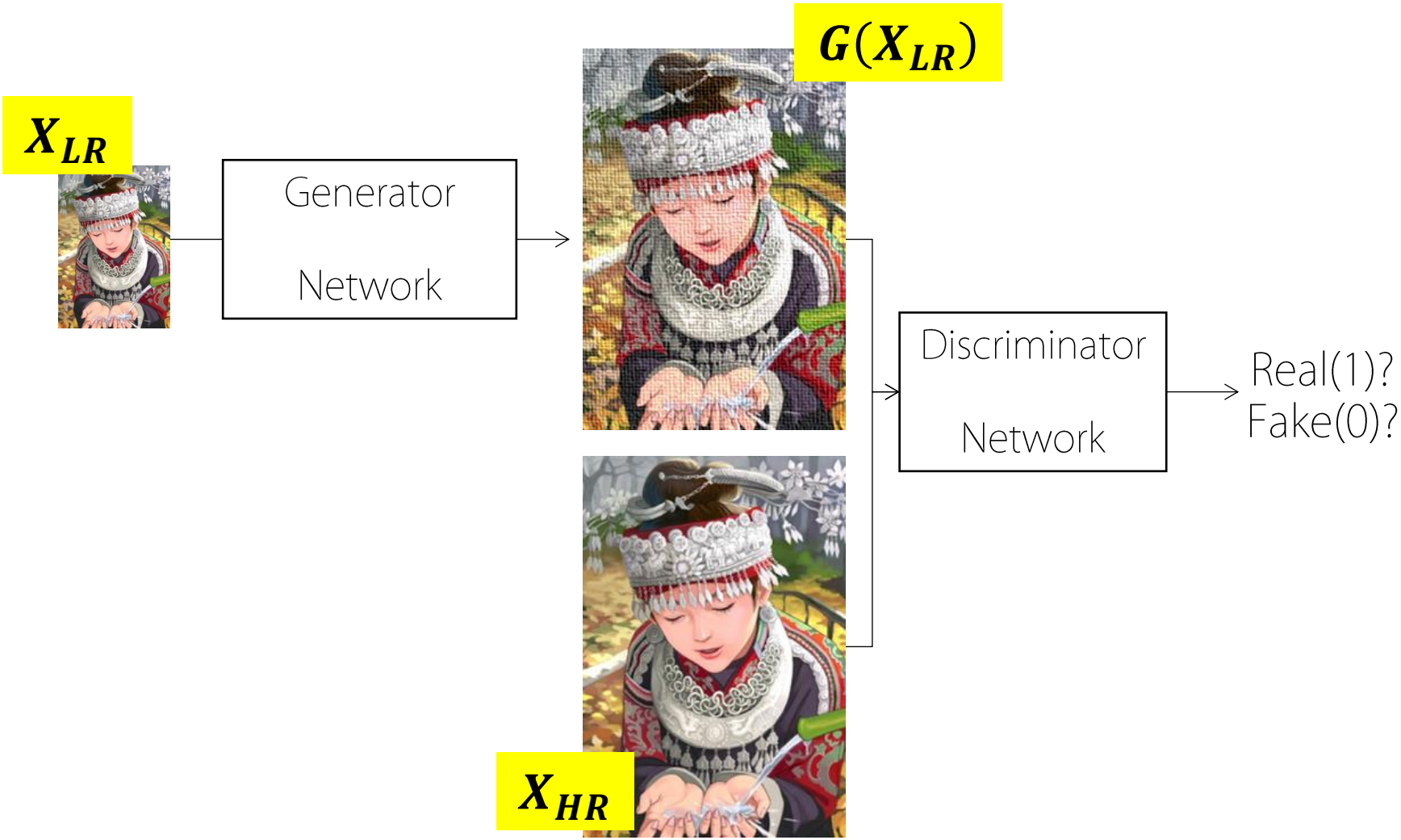
Single Image Super Resolution

Photo-realistic Single Image Super-Resolution using a Generative Adversarial Networks,
a.k.a SRGAN, CVPR 2017



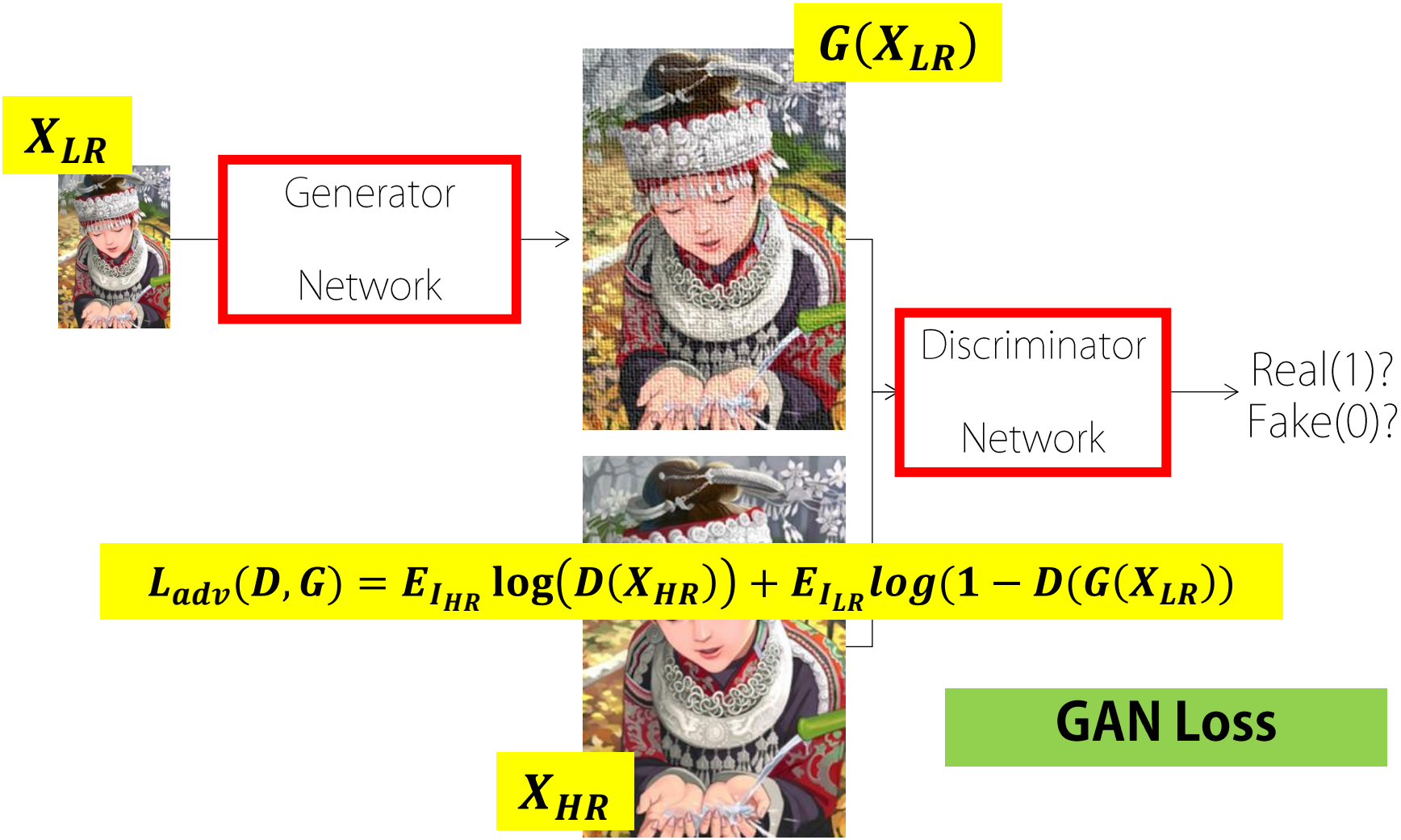
Single Image Super Resolution

Photo-realistic Single Image Super-Resolution using a Generative Adversarial Networks, CVPR 2017



Single Image Super Resolution

Photo-realistic Single Image Super-Resolution using a Generative Adversarial Networks, CVPR 2017

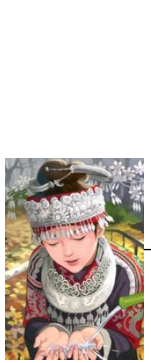


$$L_{adv}(D,G) = E_{I_{HR}} \log\big(D(X_{HR})\big) + E_{I_{LR}} log(1 - D(G(X_{LR})))$$

GAN Loss

Single Image Super Resolution

Photo-realistic Single Image Super-Resolution using a Generative Adversarial Networks, CVPR 2017



Generator Network

Discriminator Network

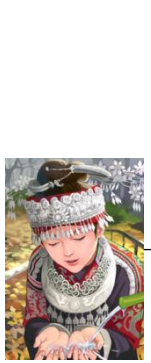
Real(1)?
Fake(0)?

$$MSE\ (\ G(X_{LR}), X_{HR}\ )$$

Recon Loss 1

Single Image Super Resolution

Photo-realistic Single Image Super-Resolution using a Generative Adversarial Networks, CVPR 2017



Generator Network

Discriminator Network

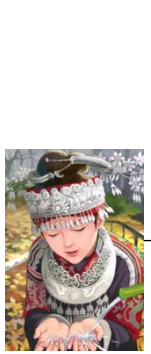
Real(1)?
Fake(0)?

$$MSE(VGGNet(X_{HR}), VGGNet(G(X_{LR})))$$

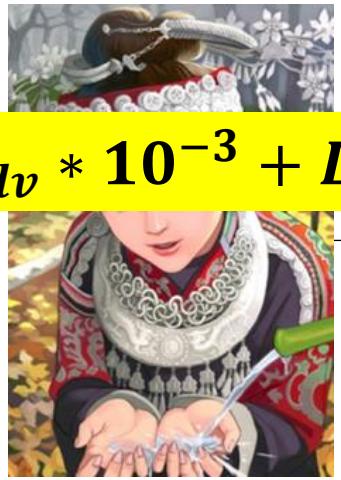
Recon Loss 2
Perceptual

Single Image Super Resolution

Photo-realistic Single Image Super-Resolution using a Generative Adversarial Networks, CVPR 2017



Generator Network



Discriminator Network

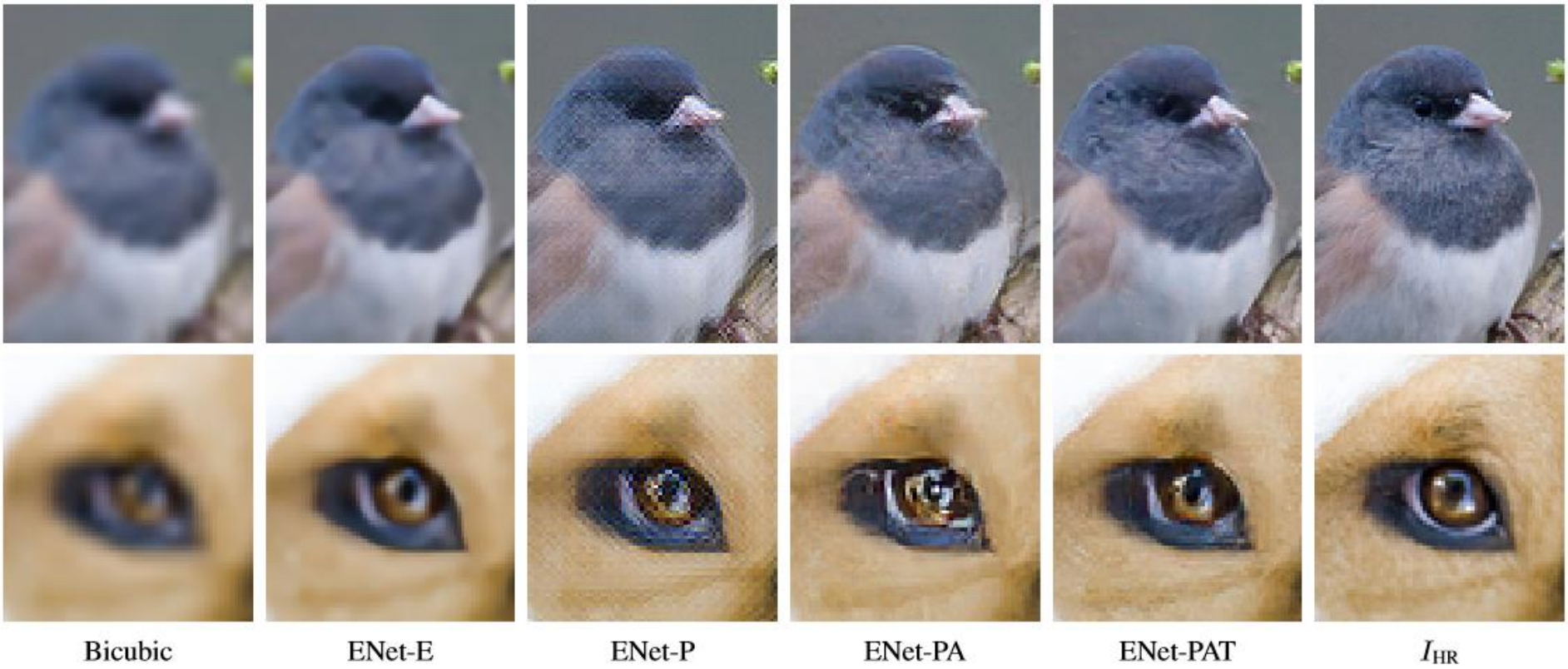
Real(1)?
Fake(0)?

$$L = L_{adv} * 10^{-3} + L_{recon}$$

Total Loss

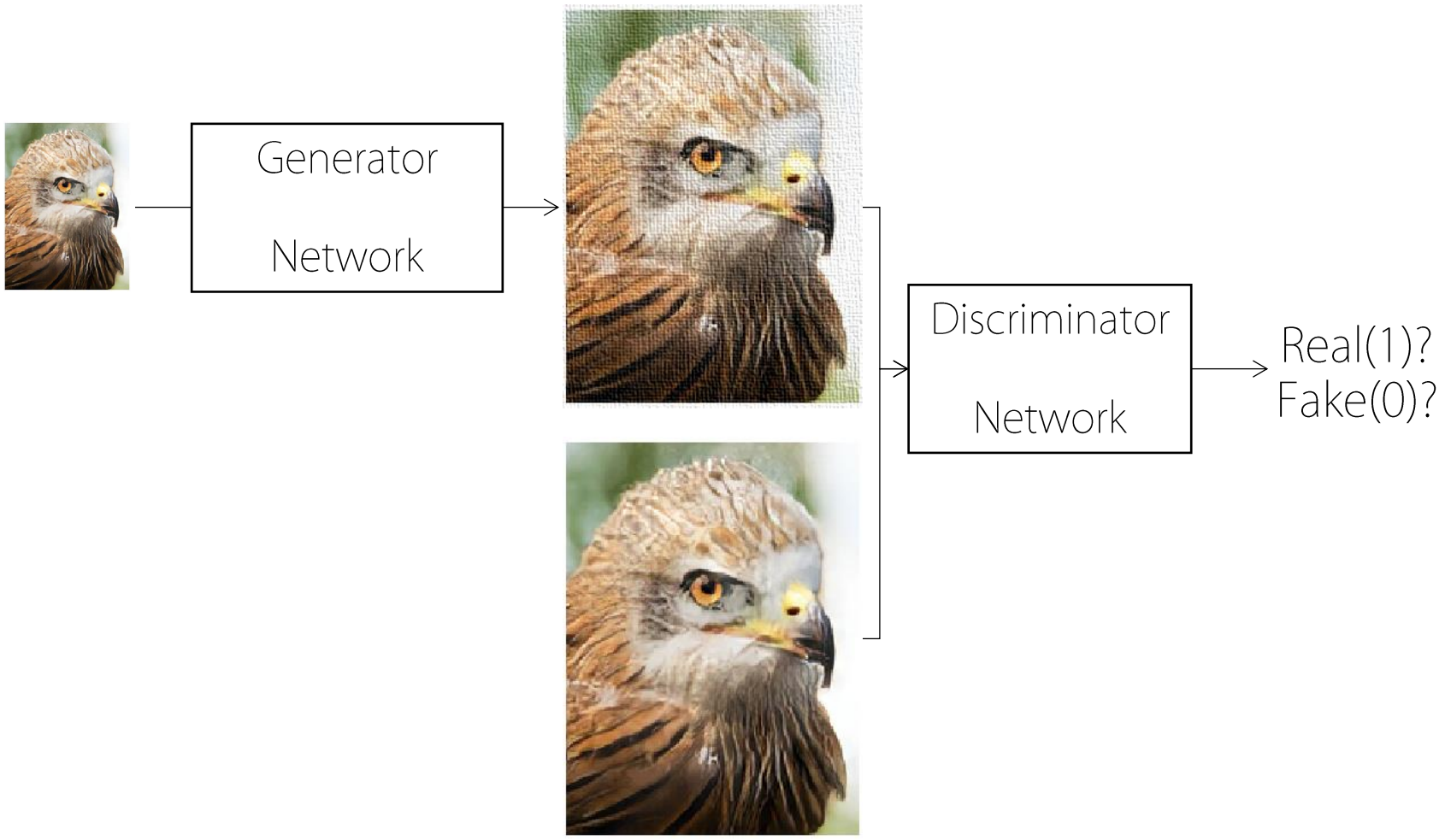
Single Image Super Resolution 2

EnhanceNet: Single Image Super Resoluton through Automated Texture Synthesis,
a.k.a EnhanceNET, ICCV 2017



Single Image Super Resolution 2

EnhanceNet: Single Image Super Resoluton through Automated Texture Synthesis, ICCV 2017



Single Image Super Resolution 2

EnhanceNet: Single Image Super Resoluton through Automated Texture Synthesis, ICCV 2017

$$L_A = E_{I_{HR}} \log\big(D(X_{HR})\big) + E_{I_{LR}} log(1 - D(G(X_{LR}))$$

GAN Loss

$$L_P = MSE(\ VGGNet(X_{HR}), VGGNet\big(G(X_{LR})\big))$$

Recon Loss

Network

$$L_T = MSE(\ GRAM\big(VGG(X_{HR})\big), GRAM\ \Big(VGG\big(G(X_{LR})\big)\Big))$$

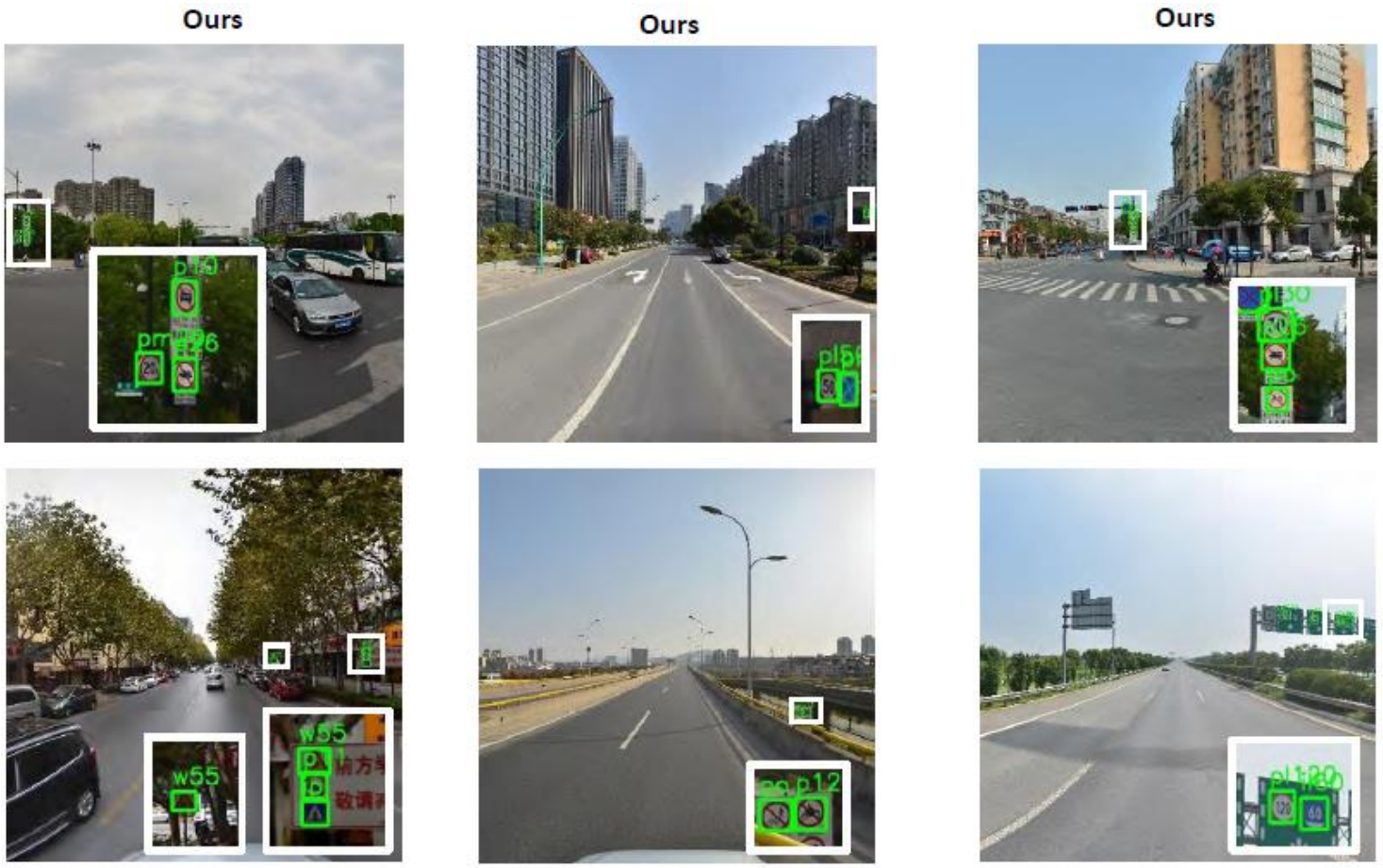
Real(1)?
Fake(0)?

Texture Loss

$$L = L_A + L_P + L_T$$

Total Loss

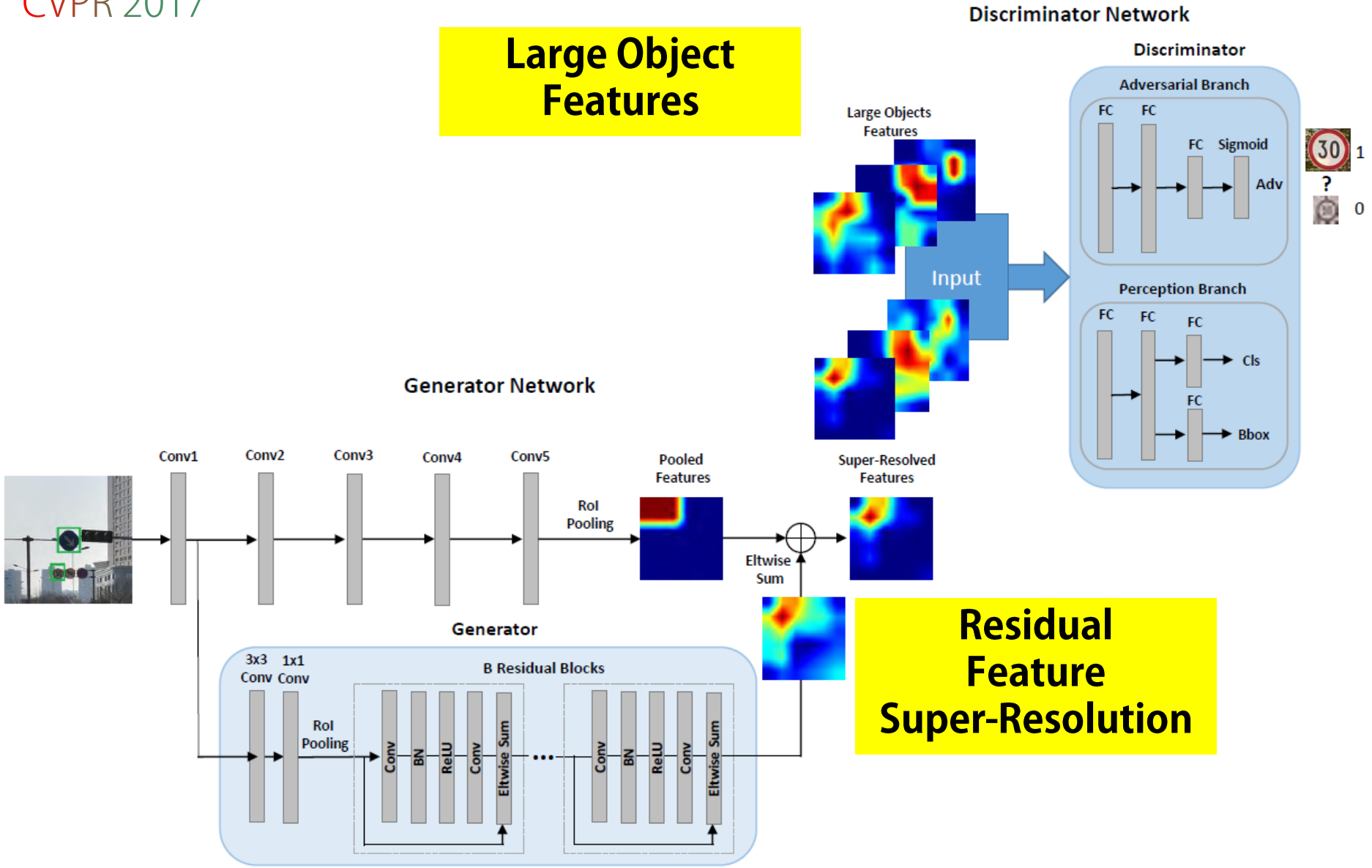
Small Object Detection Feature Upscaling



Object Detection

Perceptual Generative Adversarial Networks for Small Object Detection

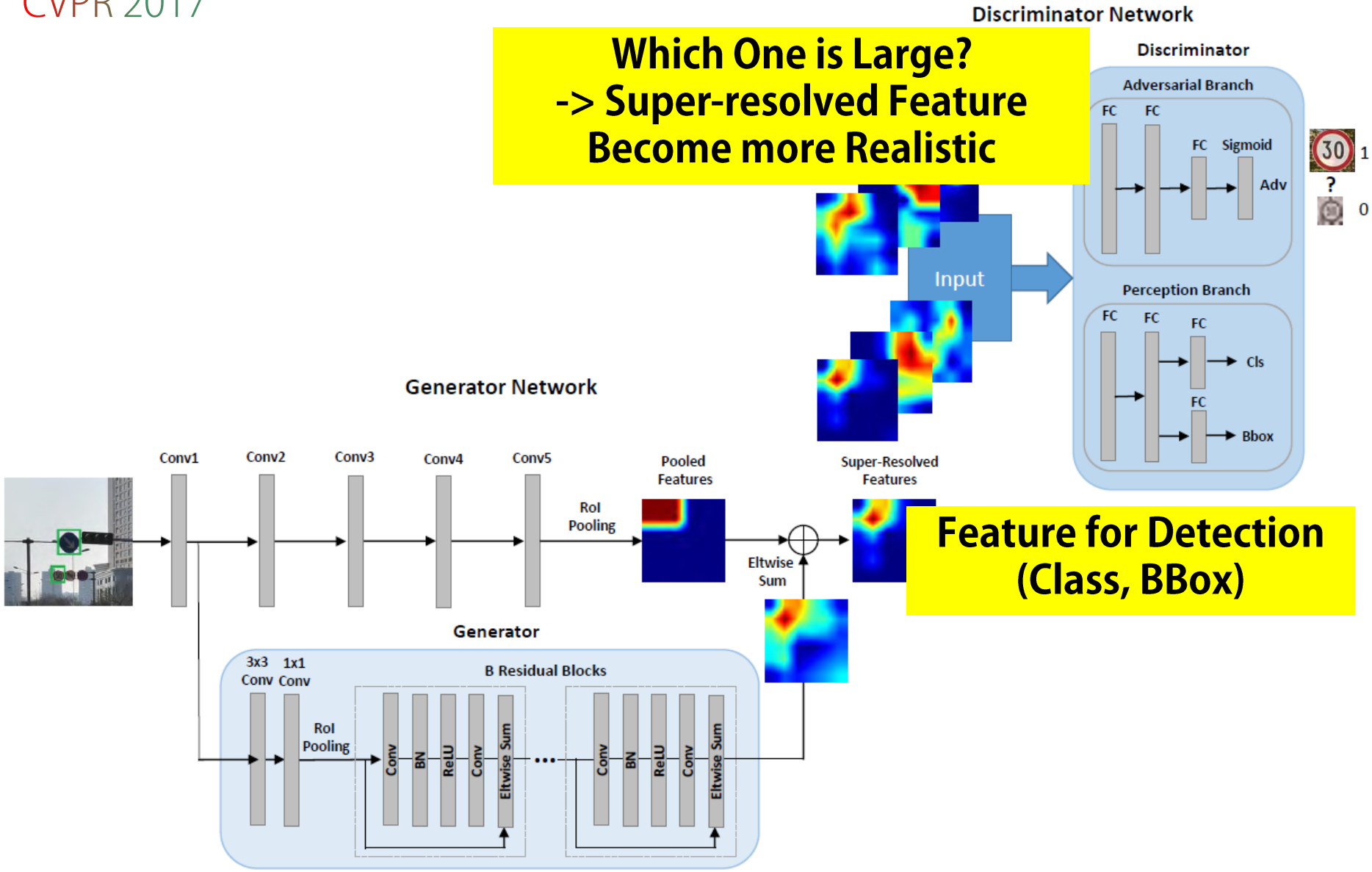
CVPR 2017



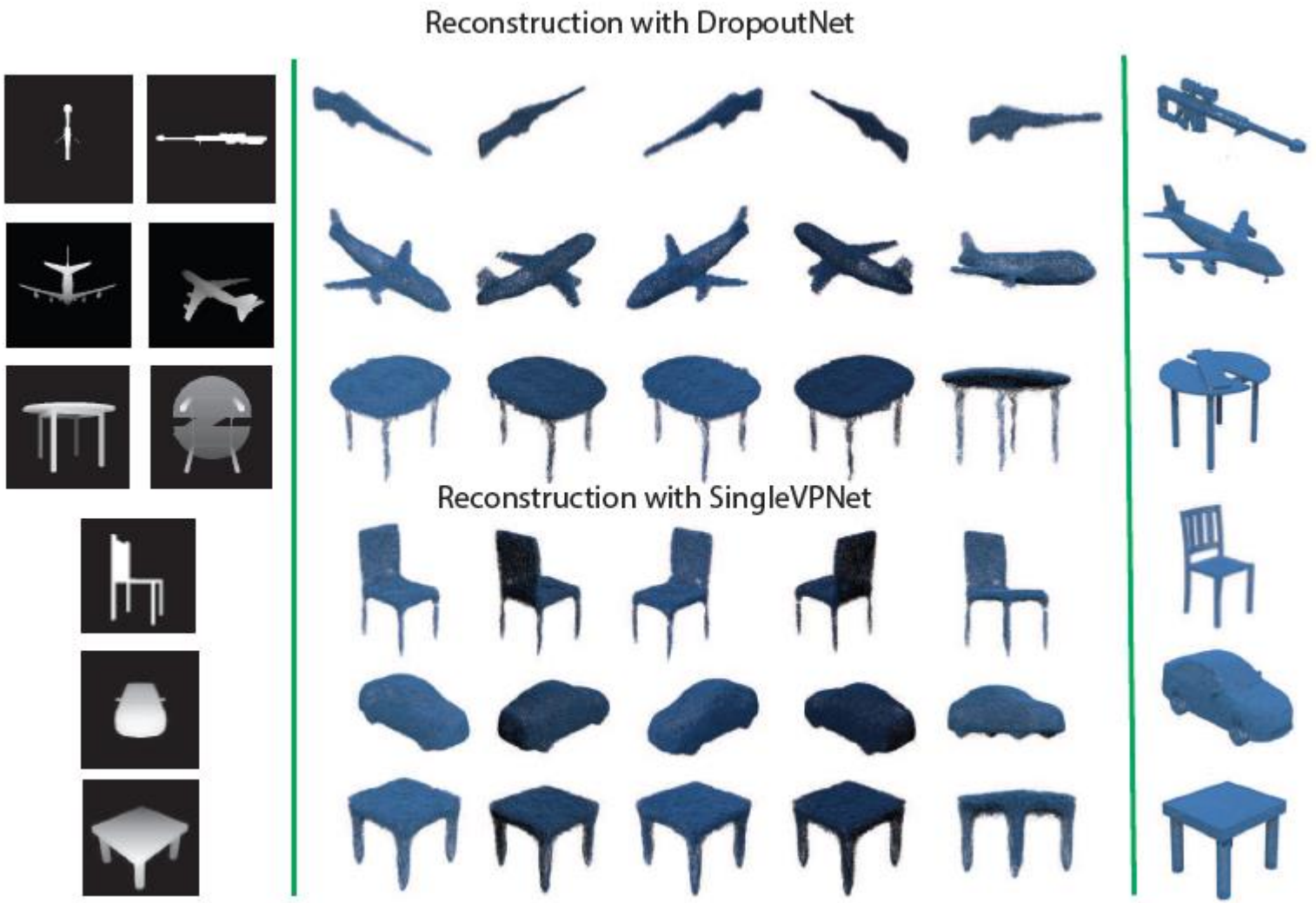
Object Detection

Perceptual Generative Adversarial Networks for Small Object Detection

CVPR 2017

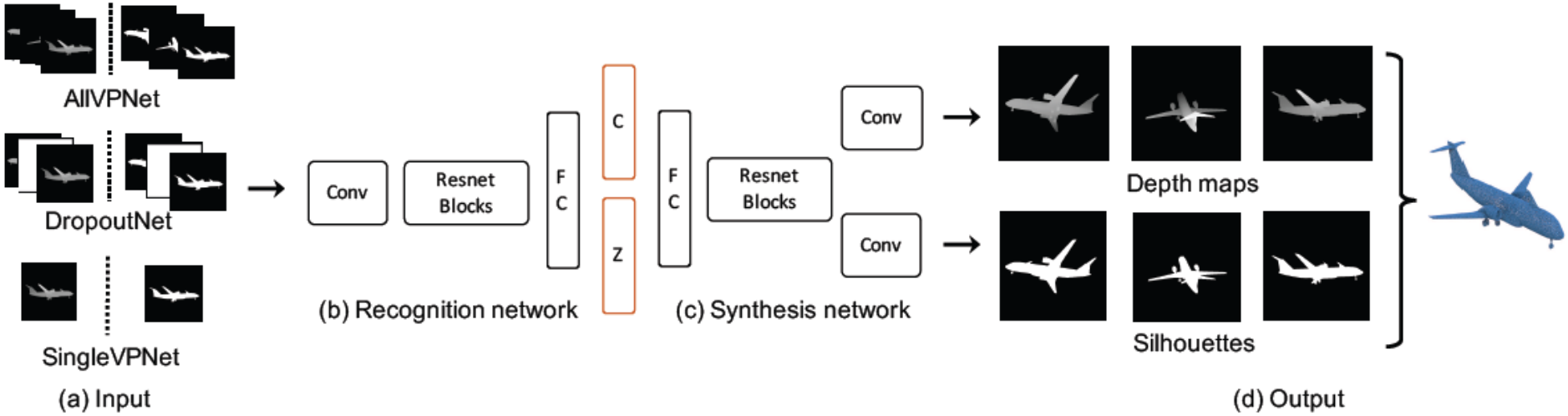


3D Reconstruction



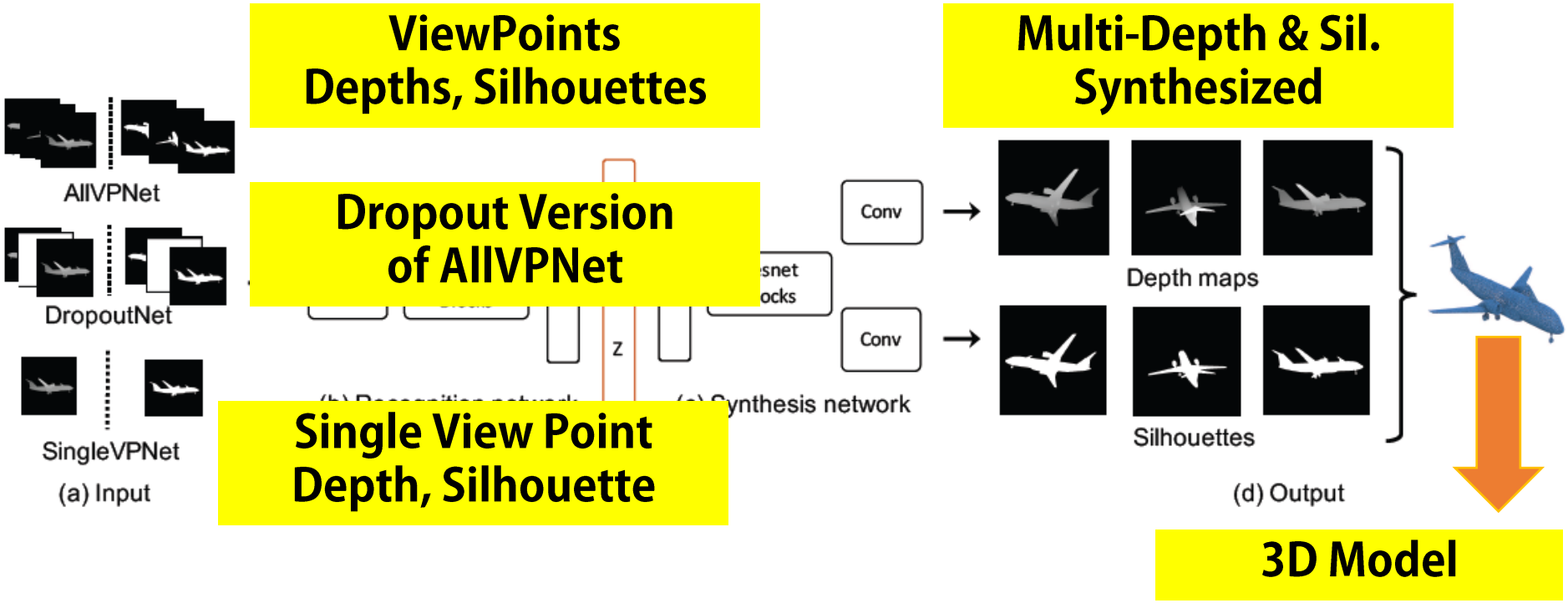
3D Reconstruction

Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks, CVPR 2017



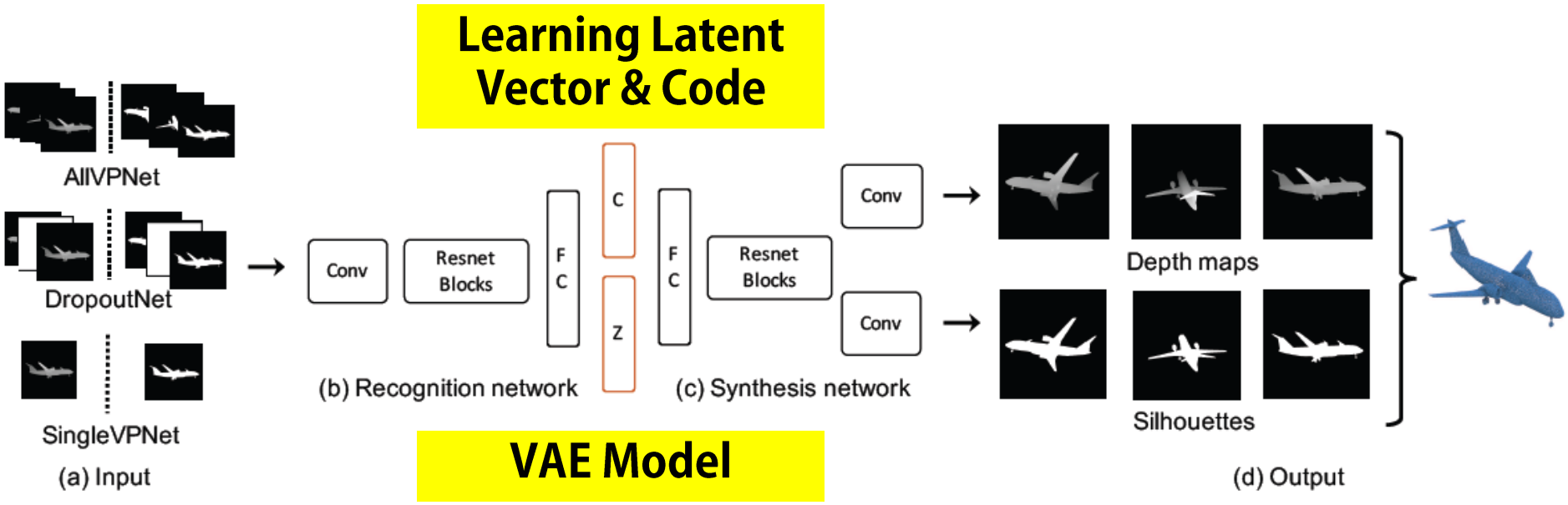
3D Reconstruction

Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks, CVPR 2017



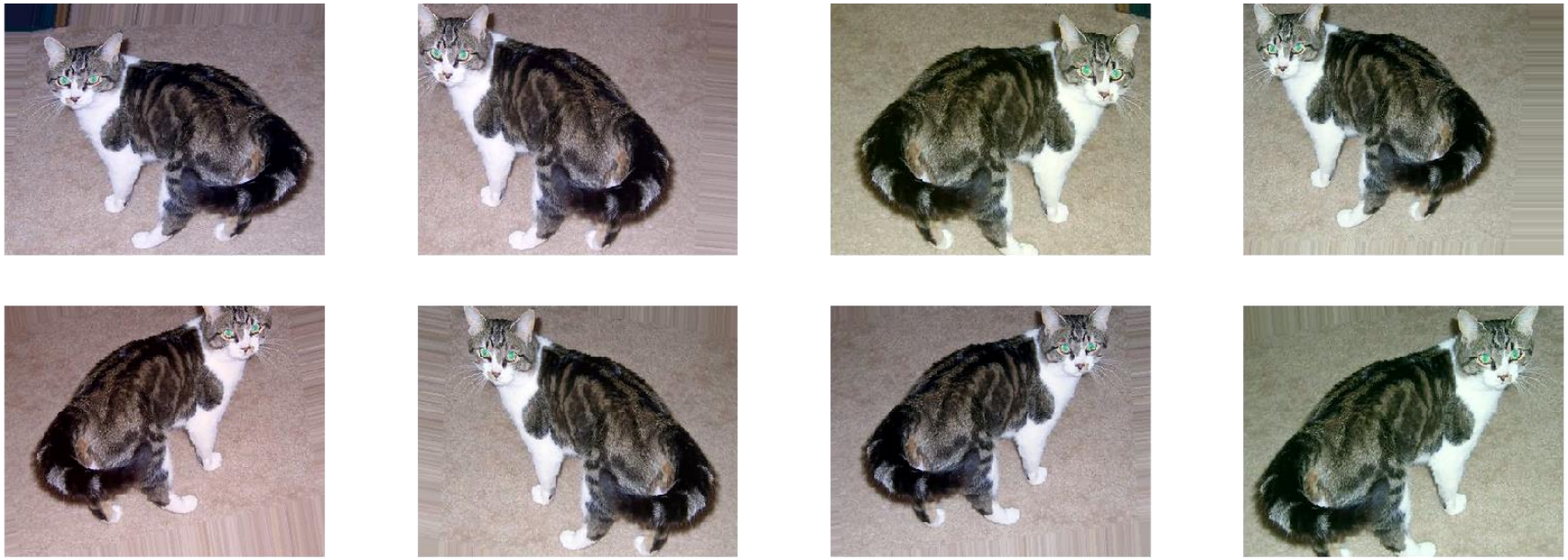
3D Reconstruction

Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks, CVPR 2017



Varing c & z ➔ Varing ViewPoints

GAN as Data Augmentation



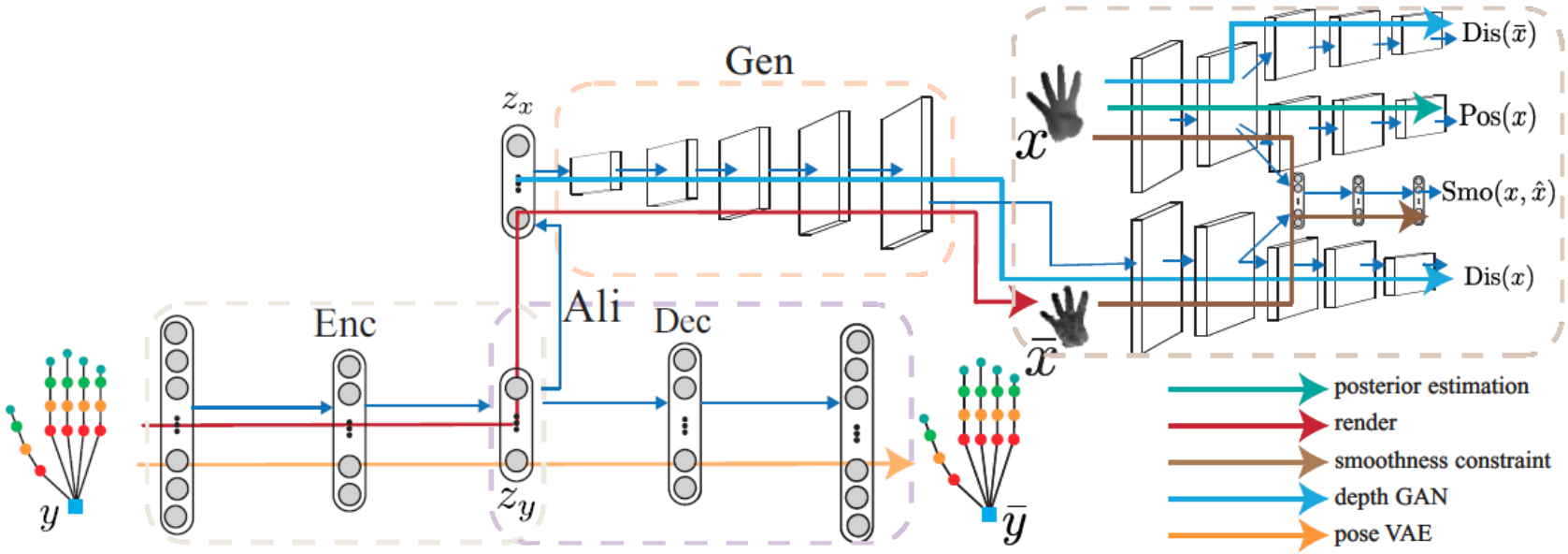
For Advanced Supervised Learning

Pose Estimation

Figure 1. **Random walk in the learned shared latent space.** A set of points is sampled on the connecting line between two points in the shared latent space. The pose and corresponding depth map are then reconstructed through our network. Our method generates meaningful and realistic interpolations in both pose and appearance space.

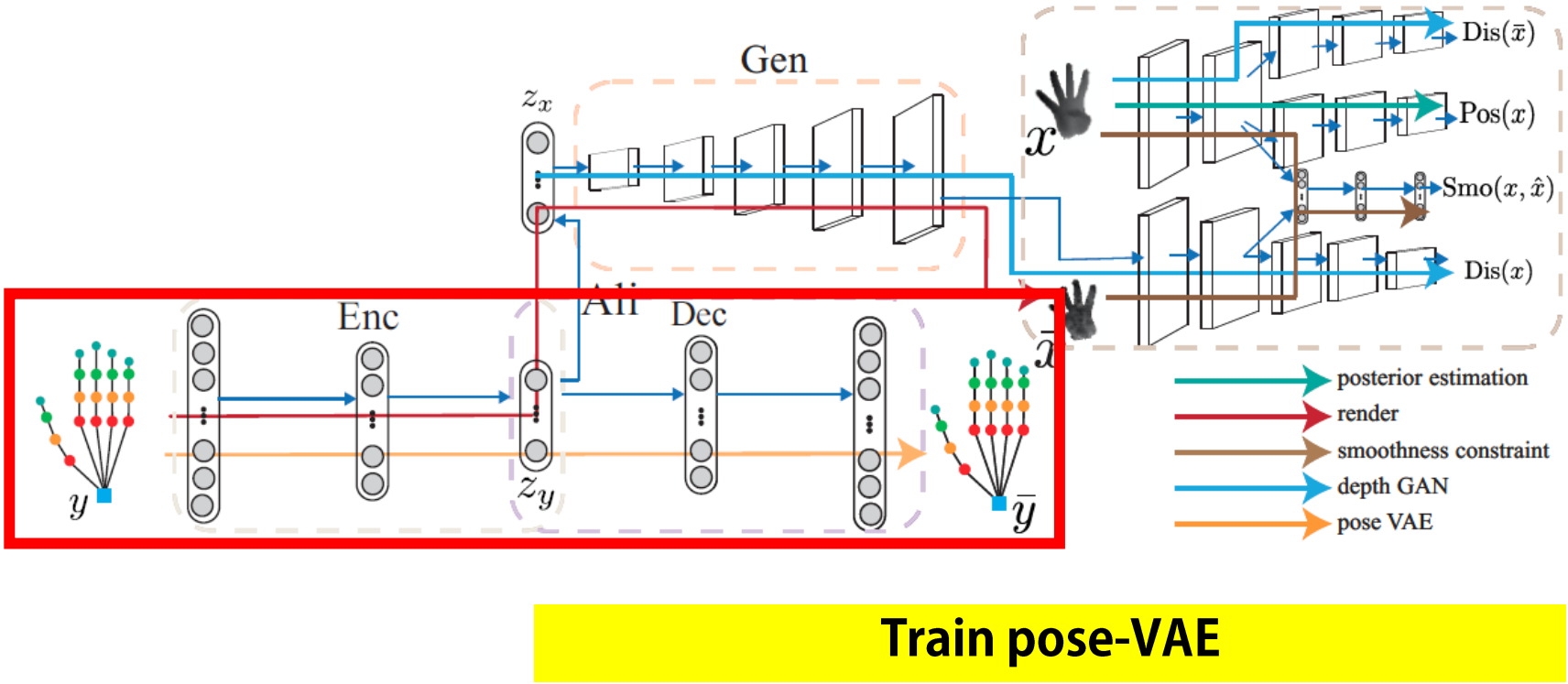
Pose Estimation

Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017



Pose Estimation

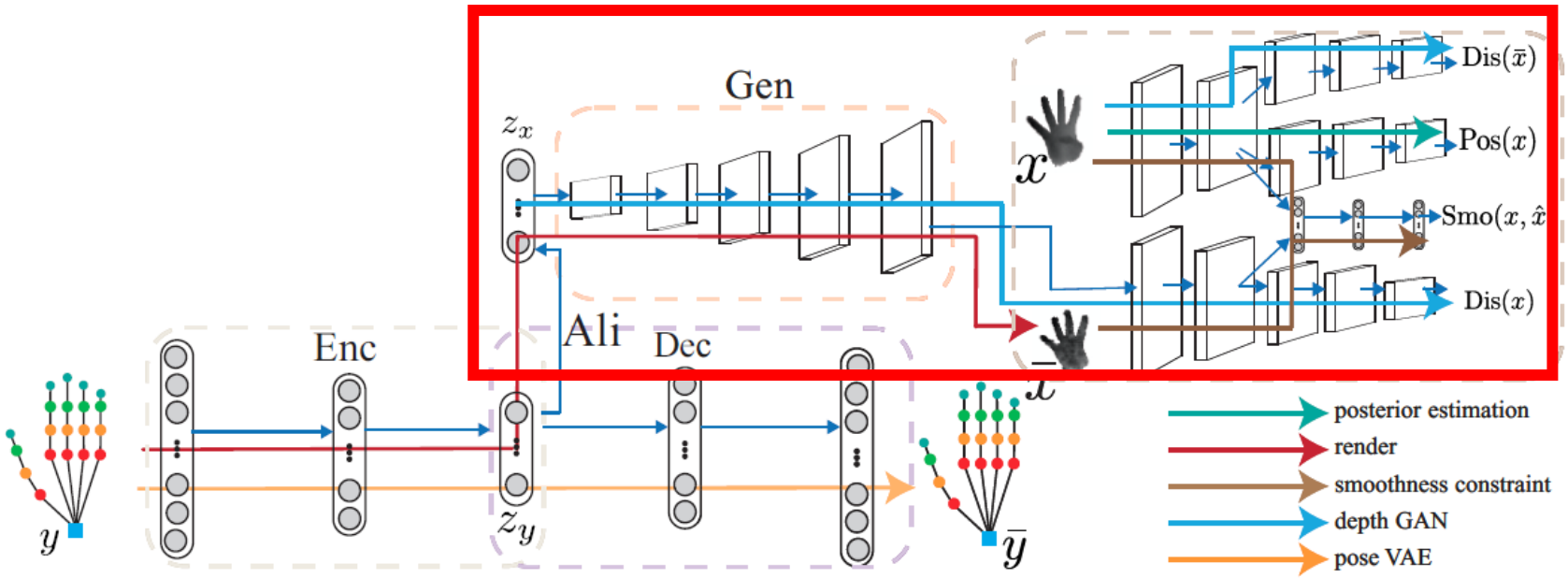
Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017



Train pose-VAE

Pose Estimation

Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017

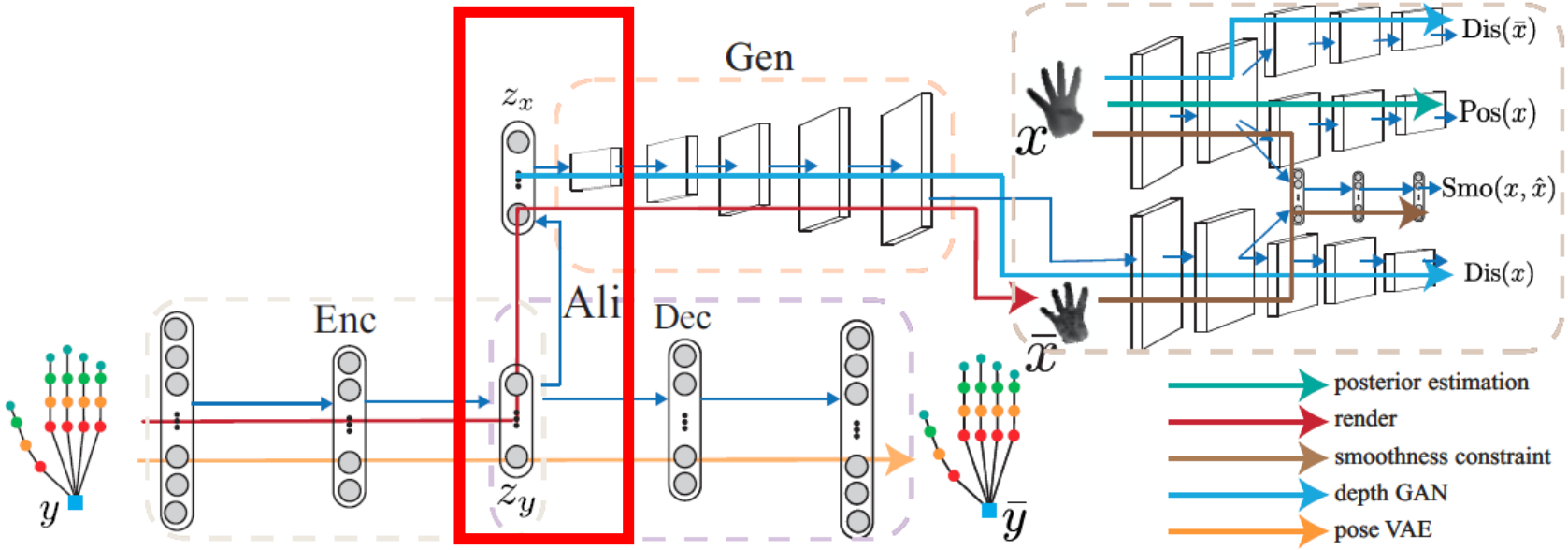


Train depth-GAN

Pose Estimation

Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017

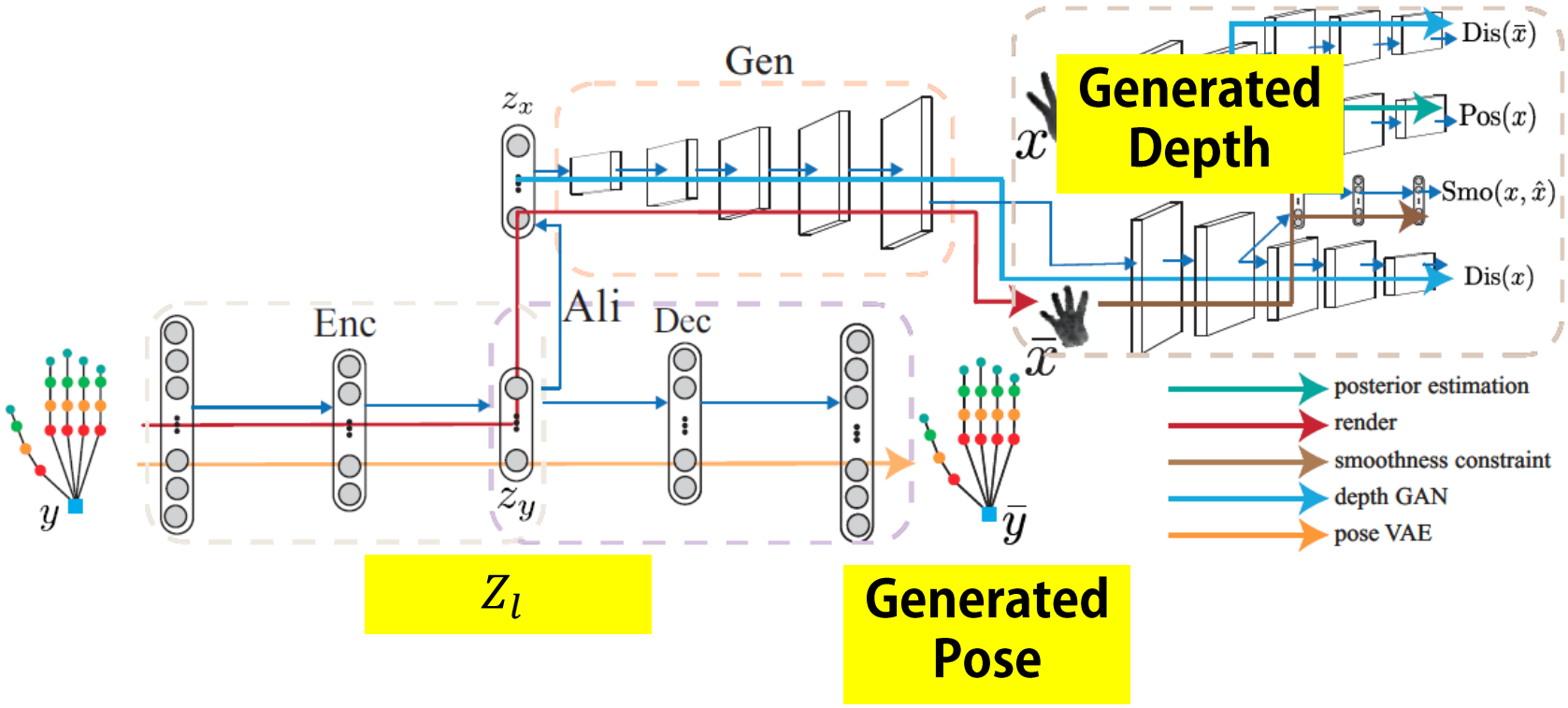
$$L_{recons} = Loss\,(x, Gen(Ali(Enc(y))))$$



Train Latent Space Mapping

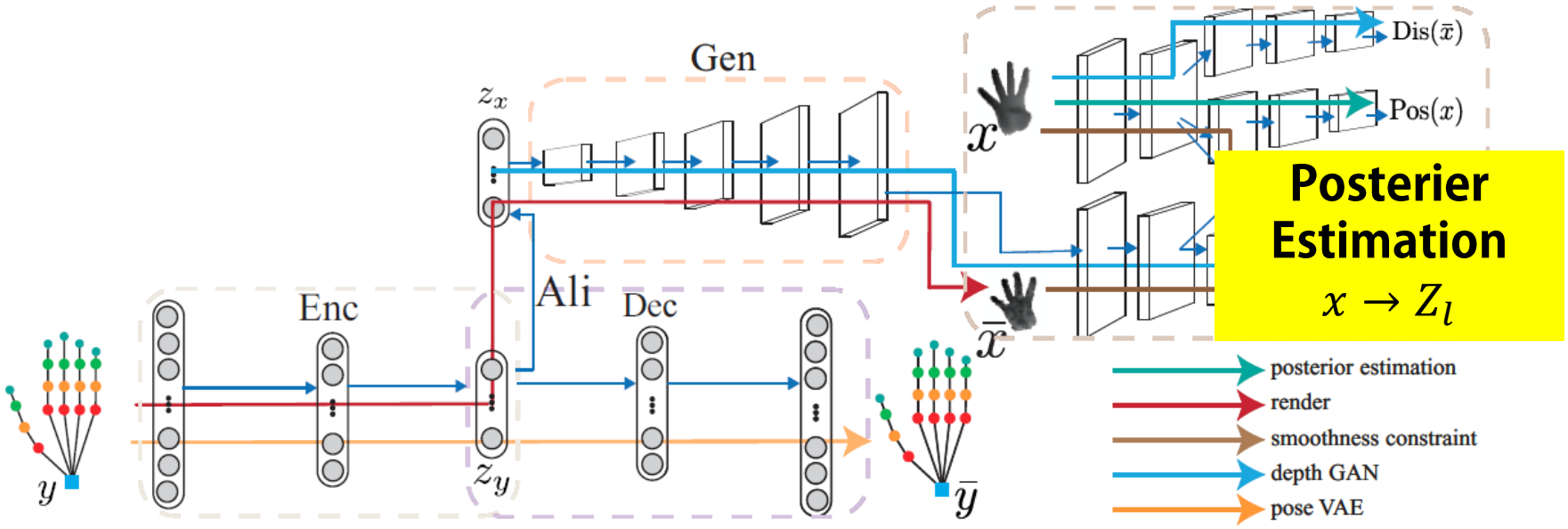
Pose Estimation

Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017



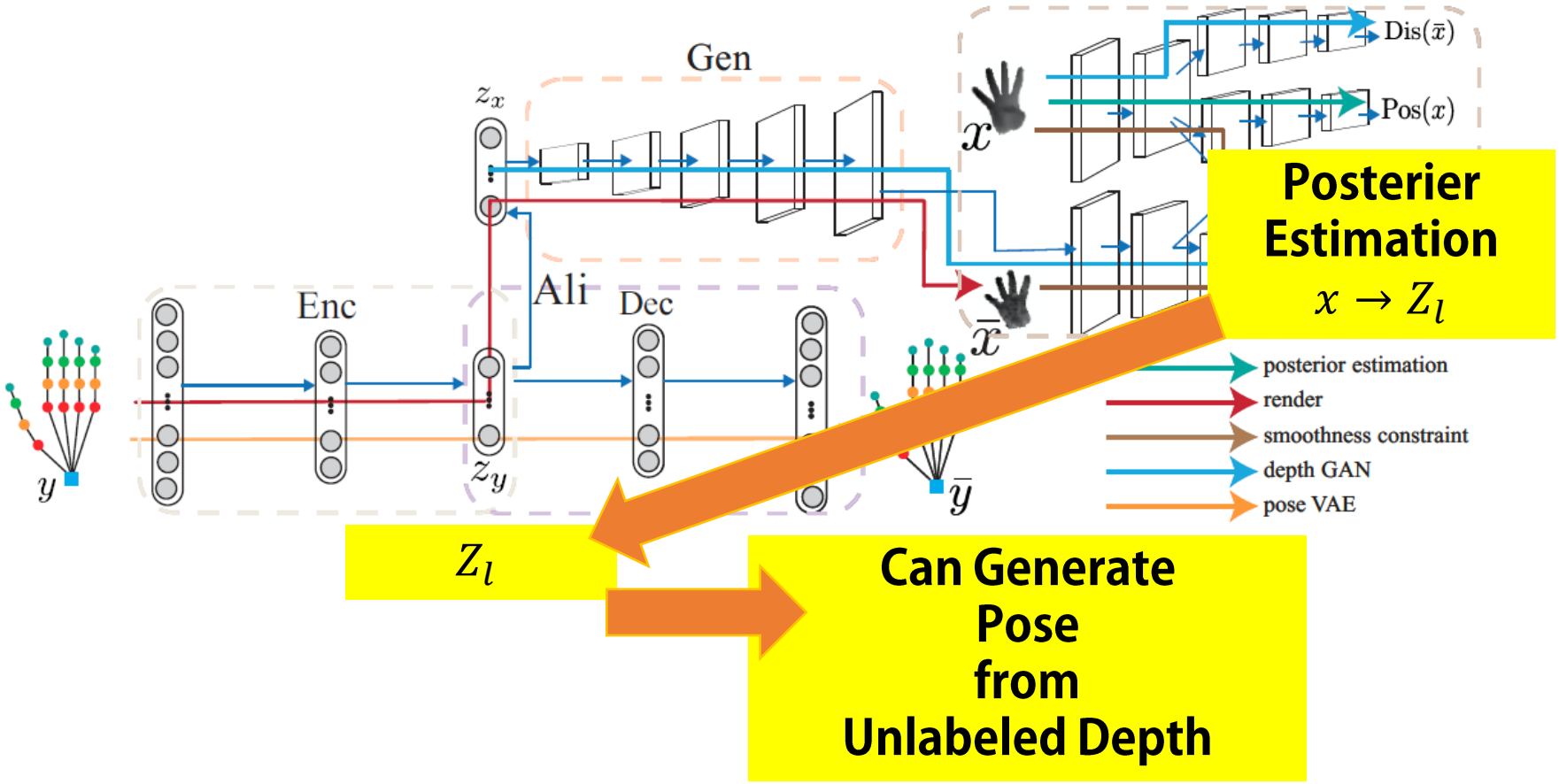
Pose Estimation

Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017

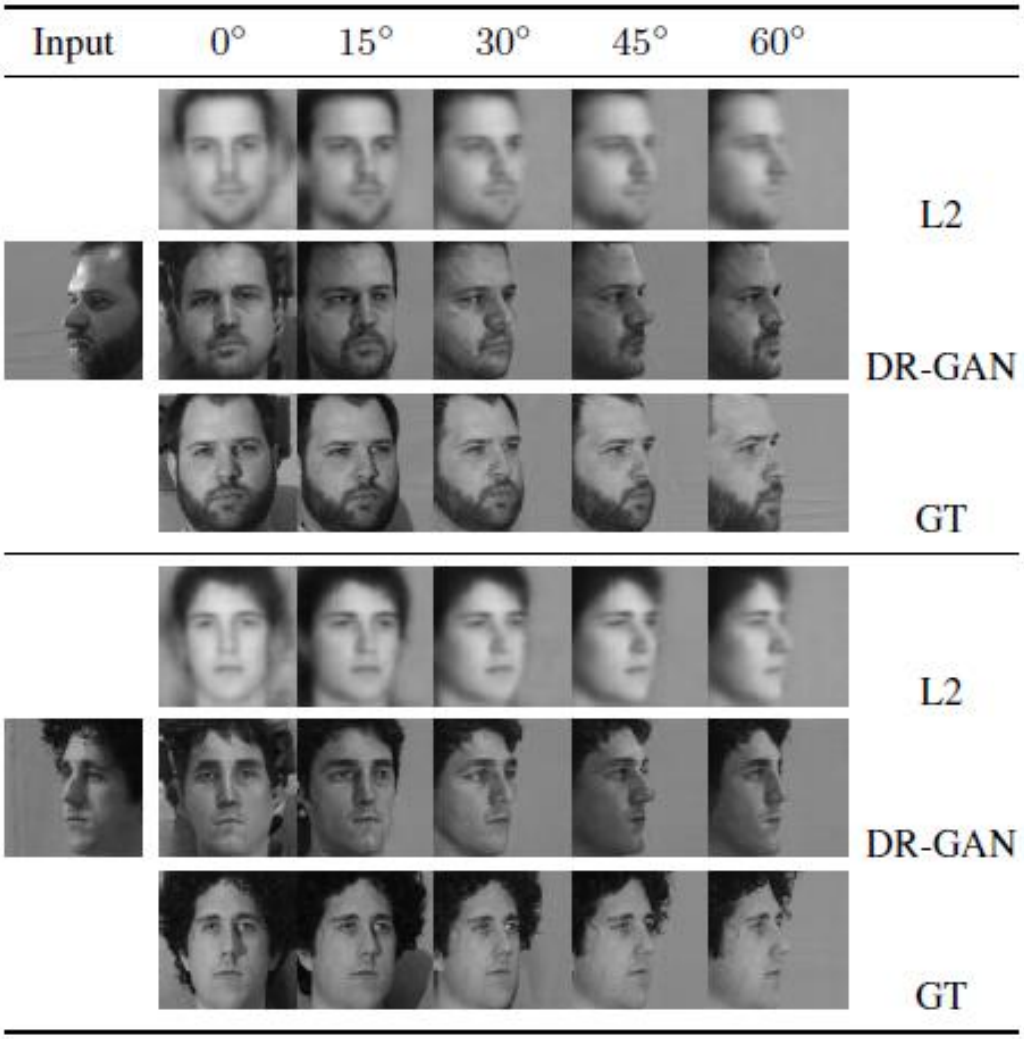


Pose Estimation

Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017



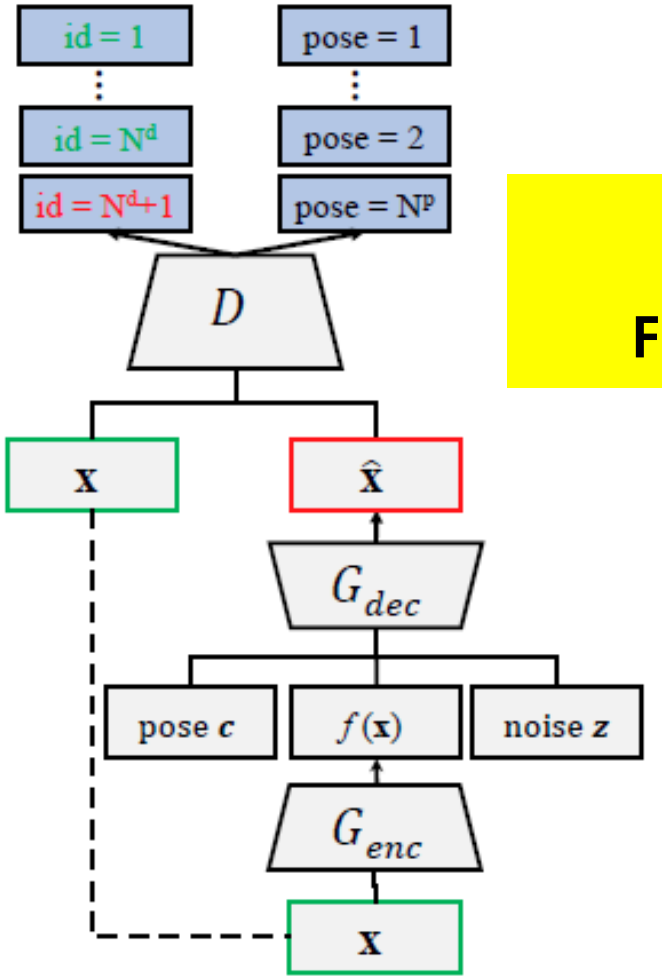
Face Recognition



Face Recognition

Distangled Representation Learning GAN for Pose-Invariant Face Recogntion

CVPR 2017

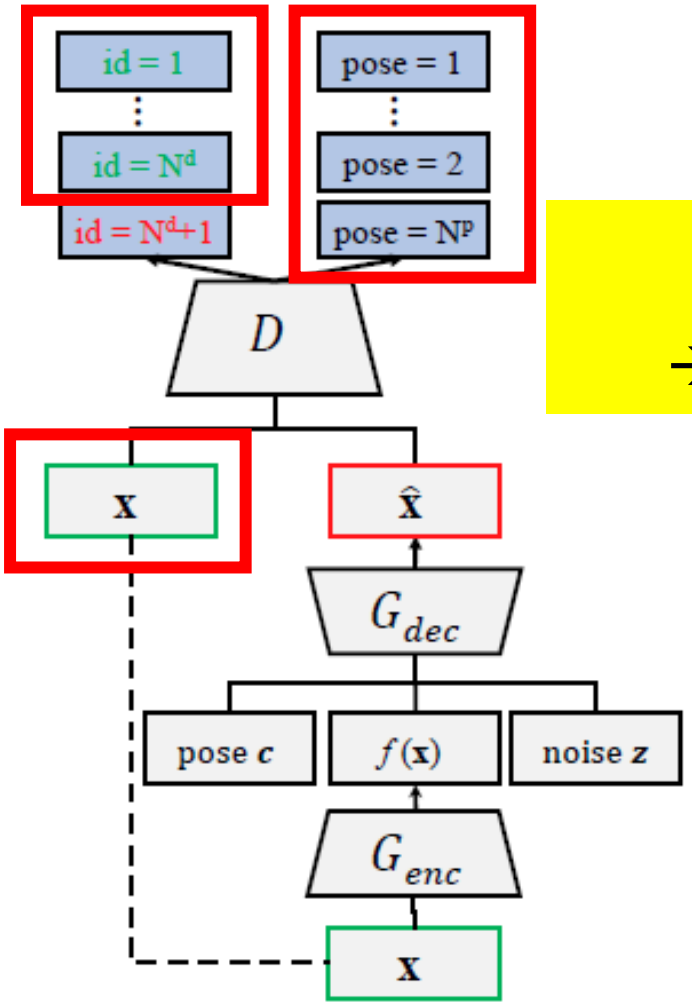


Data Generation (Pose Synthesis)
+
Face Recognition + Pose Estimation

Face Recognition

Distangled Representation Learning GAN for Pose-Invariant Face Recogntion

CVPR 2017



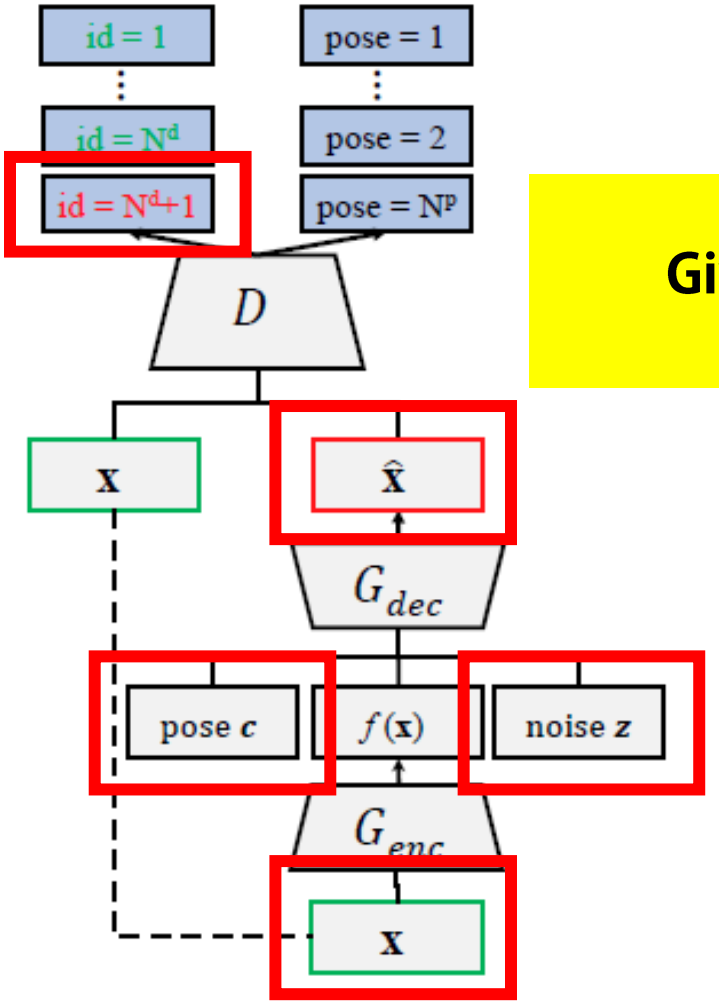
Discriminator:
Given Face, Label
→ Predict Face ID, Pose

$$E_{x,y}(logD^{d}_{y^d}(x) + logD^{p}_{y^p}(x))$$

Face Recognition

Distangled Representation Learning GAN for Pose-Invariant Face Recogntion

CVPR 2017



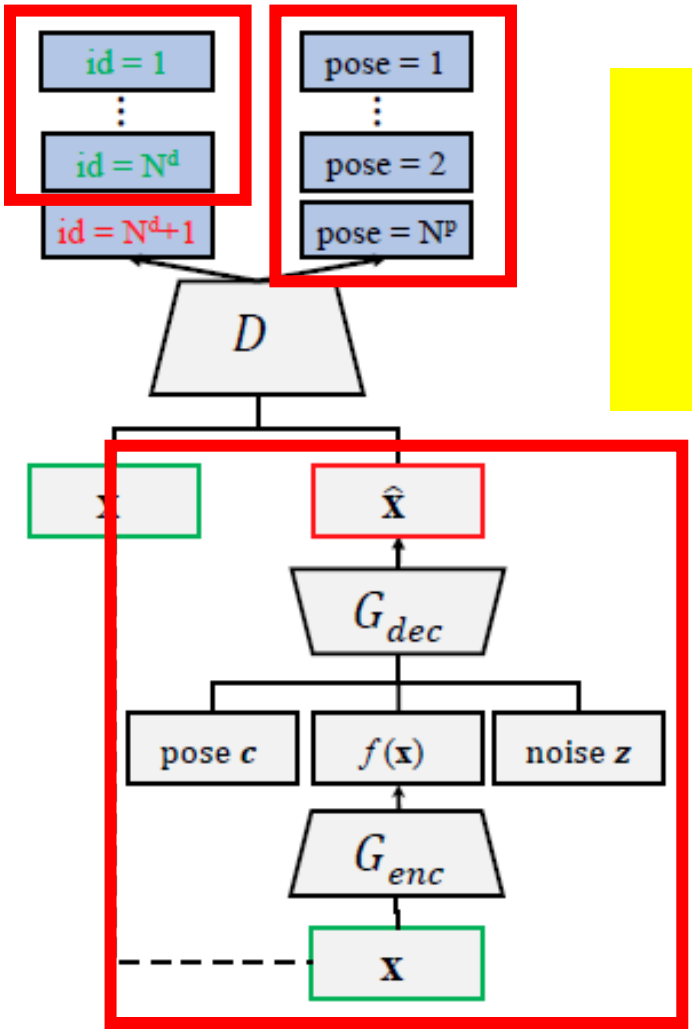
Discriminator:
Given Generated Face
→ Fake ID Class

$$E_{x,y,z,c}(logD^{d}_{N^{d}+1}(G(x,c,z)))$$

Face Recognition

Distangled Representation Learning GAN for Pose-Invariant Face Recogntion

CVPR 2017

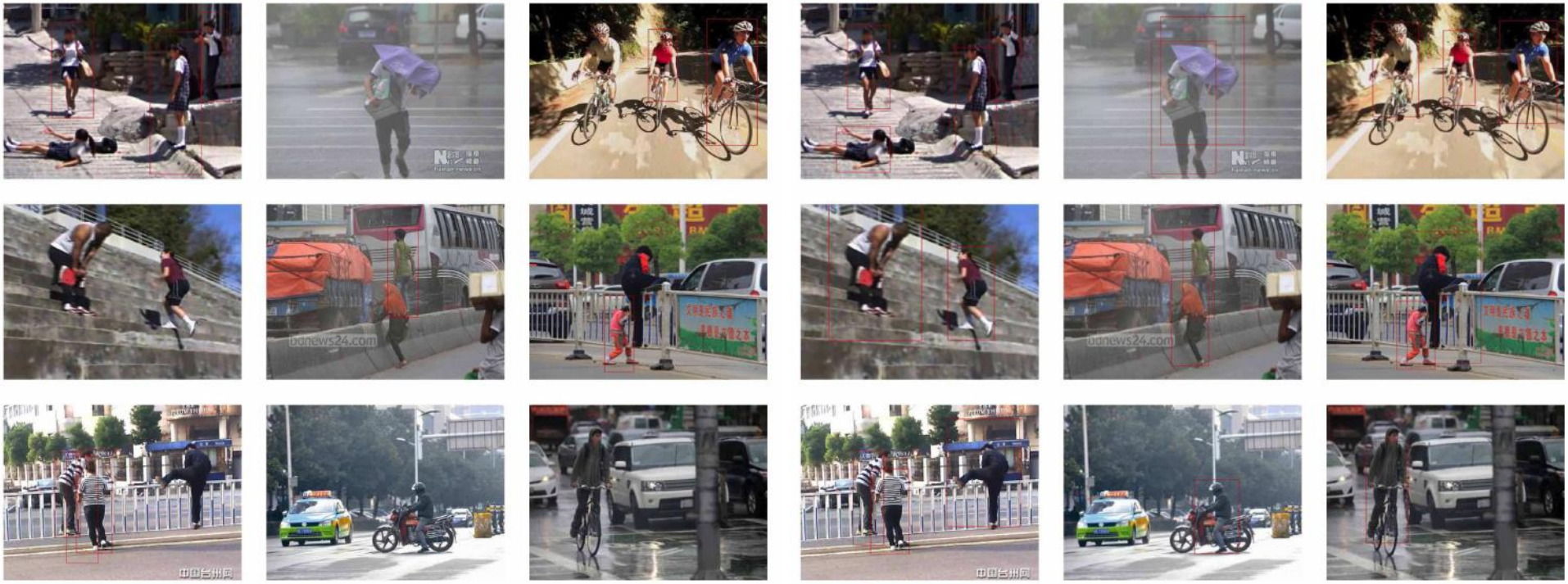


Generator:
Realistc Face Image
and Specific Pose

from Given Face

$$E_{x,y,z,c}(logD^{d}_{y^d}(G(x,c,z)) + \log(D^{p}_{y^t}(G(x,c,z))))$$

Pedestrian Detection

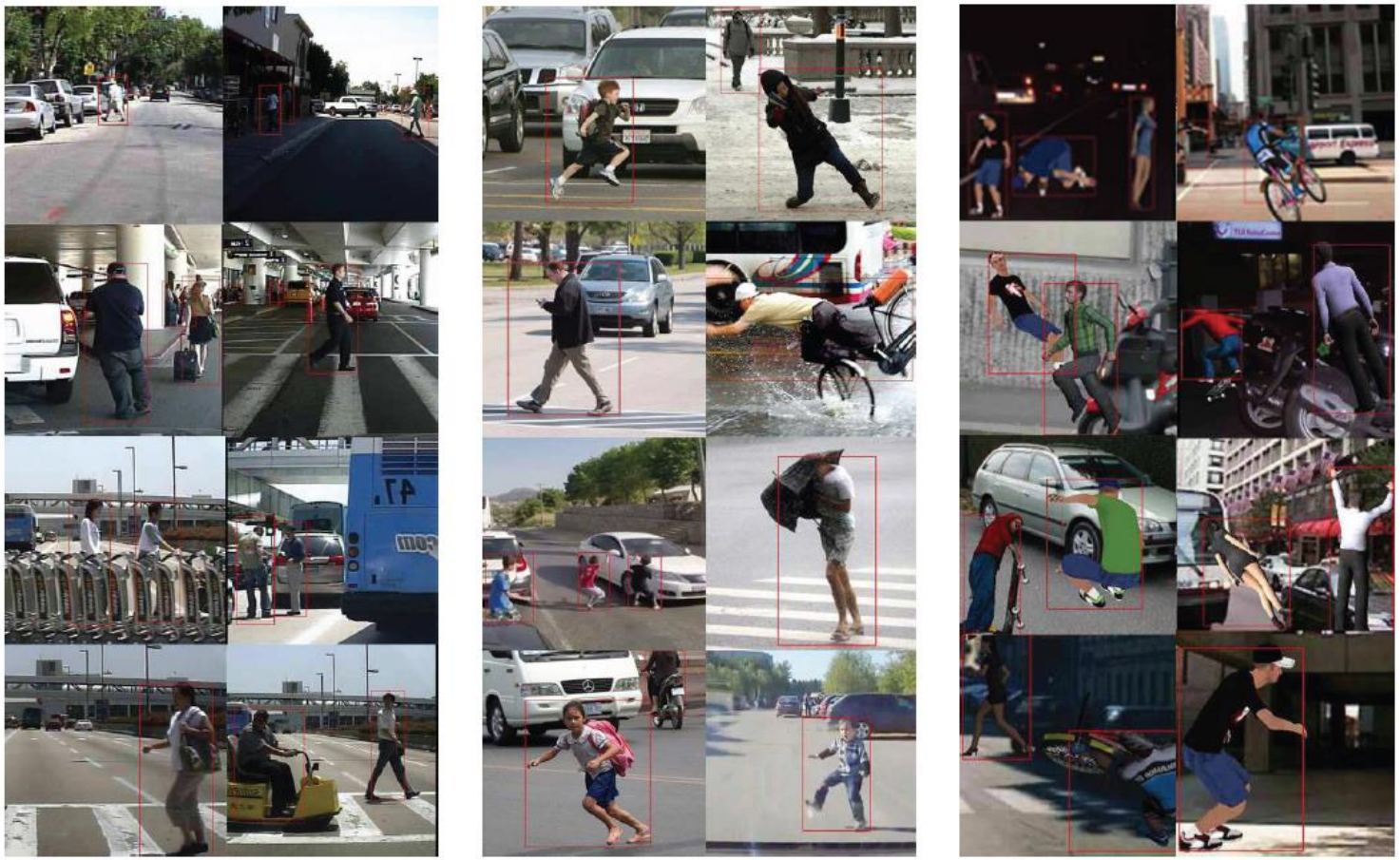


(a) RPN+Caltech

(b) Ours

Pedestrian Detection

Expecting the Unexpected: Training Detectors for Unusual Pedestrians with Adversarial Imposters, CVPR 2017



Caltech | Precarious | Synthesis

Pedestrian Detection

Expecting the Unexpected: Training Detectors for Unusual Pedestrians with Adversarial Imposters, CVPR 2017

	Range
Number of 3D models	$[4, 8]$
Index of background images	$[0, 1726)$
Index of 3D models	$[0, 20)$
Position of 3D models	Within the field of vision
Index of Animations	$[0, maxnumber)$
Time of animation	$[0, 1]$
Model's angle on the x axis	$[-90°, 90°]$
Model's angle on the y axis	$[-180°, 180°]$
Model's angle on the z axis	$[-90°, 90°]$
Light intensity	$[0.5, 2]$
Light's angle on the x axis	$[-45°, 45°]$
Light's angle on the y axis	$[-45°, 45°]$

Discriminator

Pedestrian Detection

Expecting the Unexpected: Training Detectors for Unusual Pedestrians with Adversarial Imposters, CVPR 2017

	Range
Number of 3D models	$[4, 8]$
Index of background images	$[0, 1726)$
Index of 3D models	$[0, 20)$
Position of 3D models	Within the field of vision
Index of Animations	$[0, maxnumber)$
Time of animation	$[0, 1]$
Model's angle on the x axis	$[-90°, 90°]$
Model's angle on the y axis	$[-180°, 180°]$
Model's angle on the z axis	$[-90°, 90°]$
Light intensity	$[0.5, 2]$
Light's angle on the x axis	$[-45°, 45°]$
Light's angle on the y axis	$[-45°, 45°]$

Train Sampling Parameter Distribution

3D Engine

Discriminator

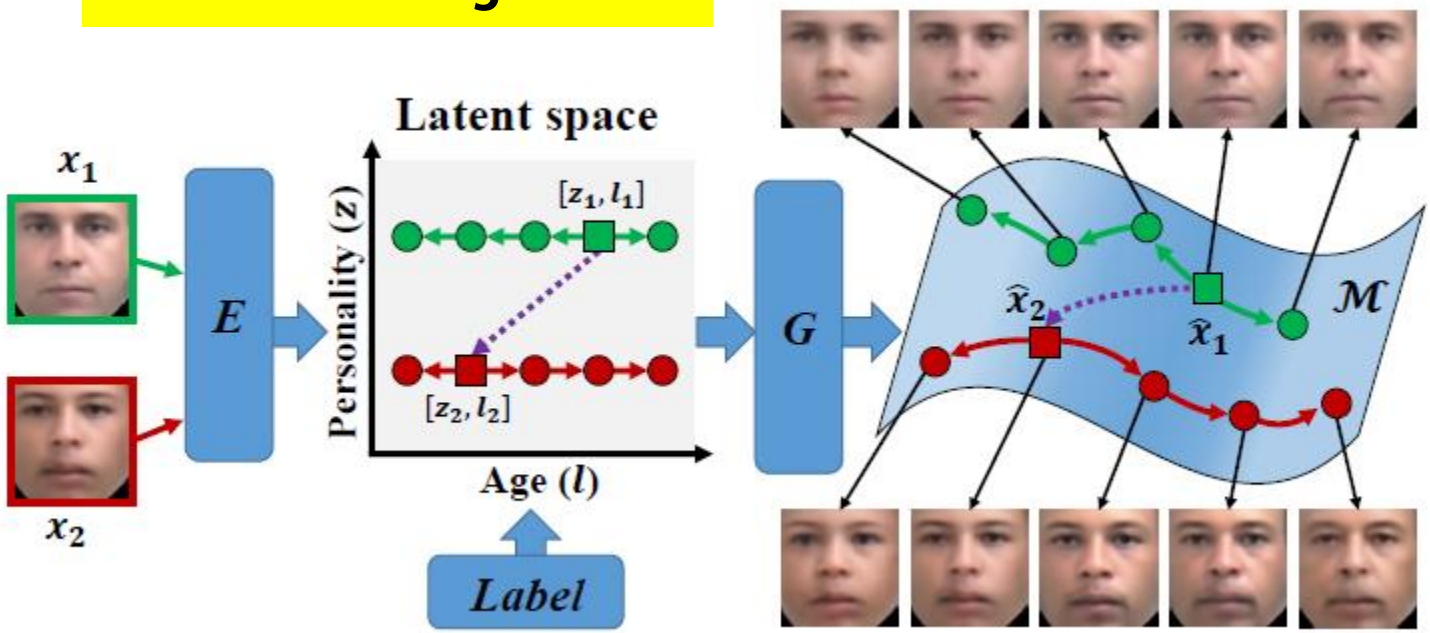
Other Approaches..

Modeling / Regression

Face Aging Prediction

Age Prediction/Regression by Conditional Adversarial Autoencoder
CVPR 2017

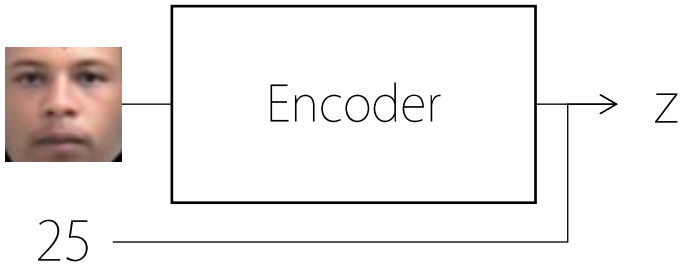
Makes Latent Space Disentangle



Makes Prediction Image Realistic

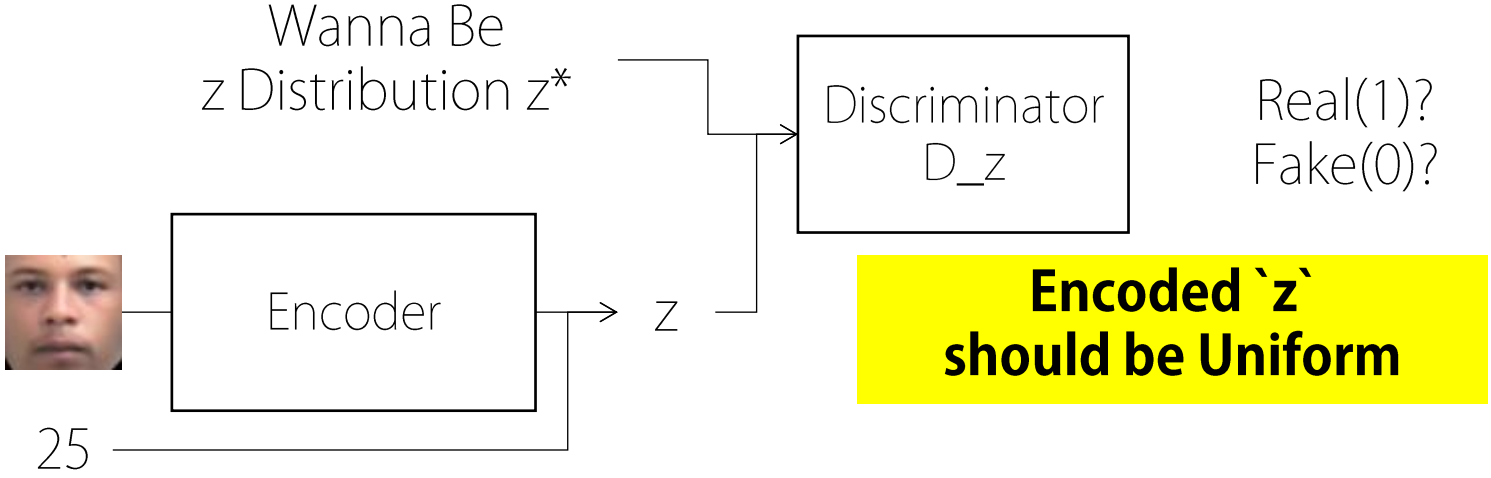
Face Aging Prediction

Age Prediction/Regression by Conditional Adversarial Autoencoder
CVPR 2017



Face Aging Prediction

Age Prediction/Regression by Conditional Adversarial Autoencoder
CVPR 2017

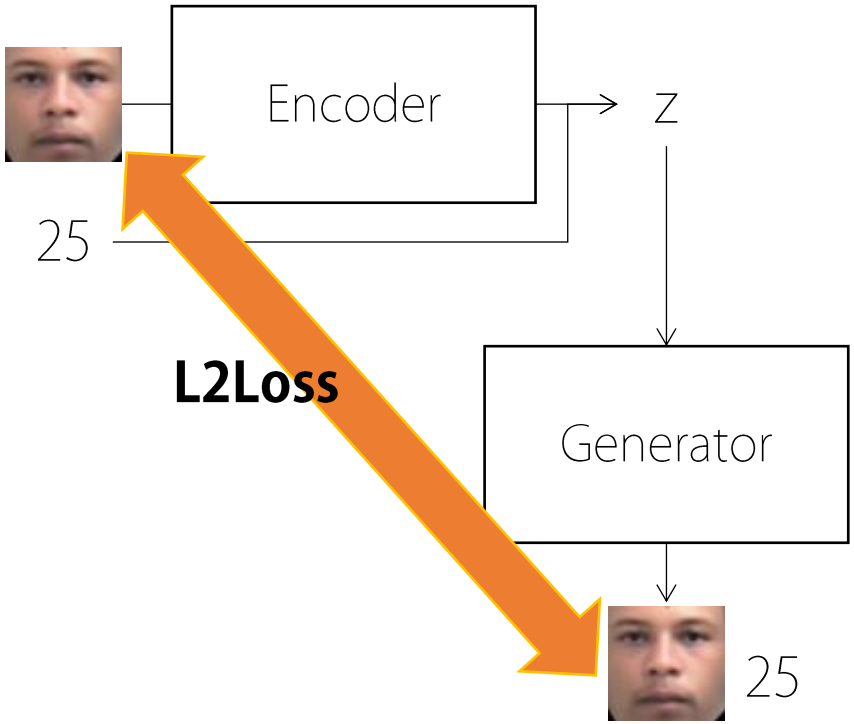


$$E_{z^*}(\log D_z(z^*)) + E_x\left(\log\left(1 - D_z(E(x))\right)\right)$$

Face Aging Prediction

Age Prediction/Regression by Conditional Adversarial Autoencoder

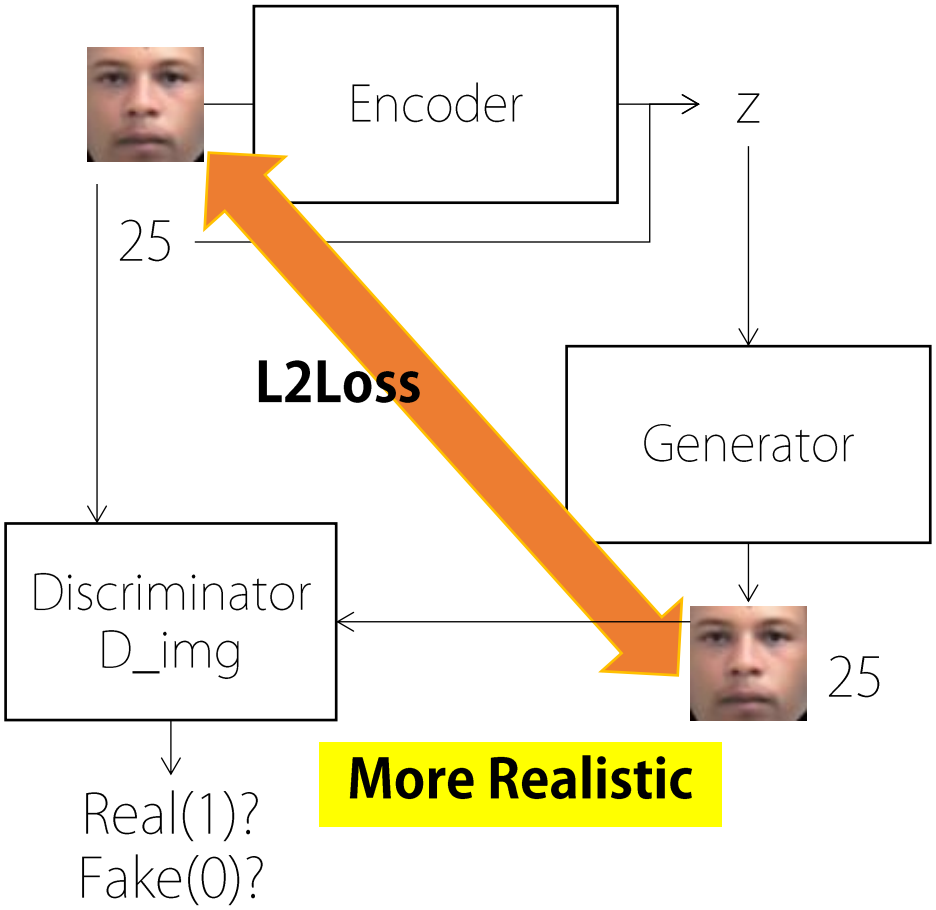
CVPR 2017



$$L_2\big(x, G(E(x), l)\big) + TV\big(G\big(E(x)\big), l\big)$$

Face Aging Prediction

Age Prediction/Regression by Conditional Adversarial Autoencoder
CVPR 2017

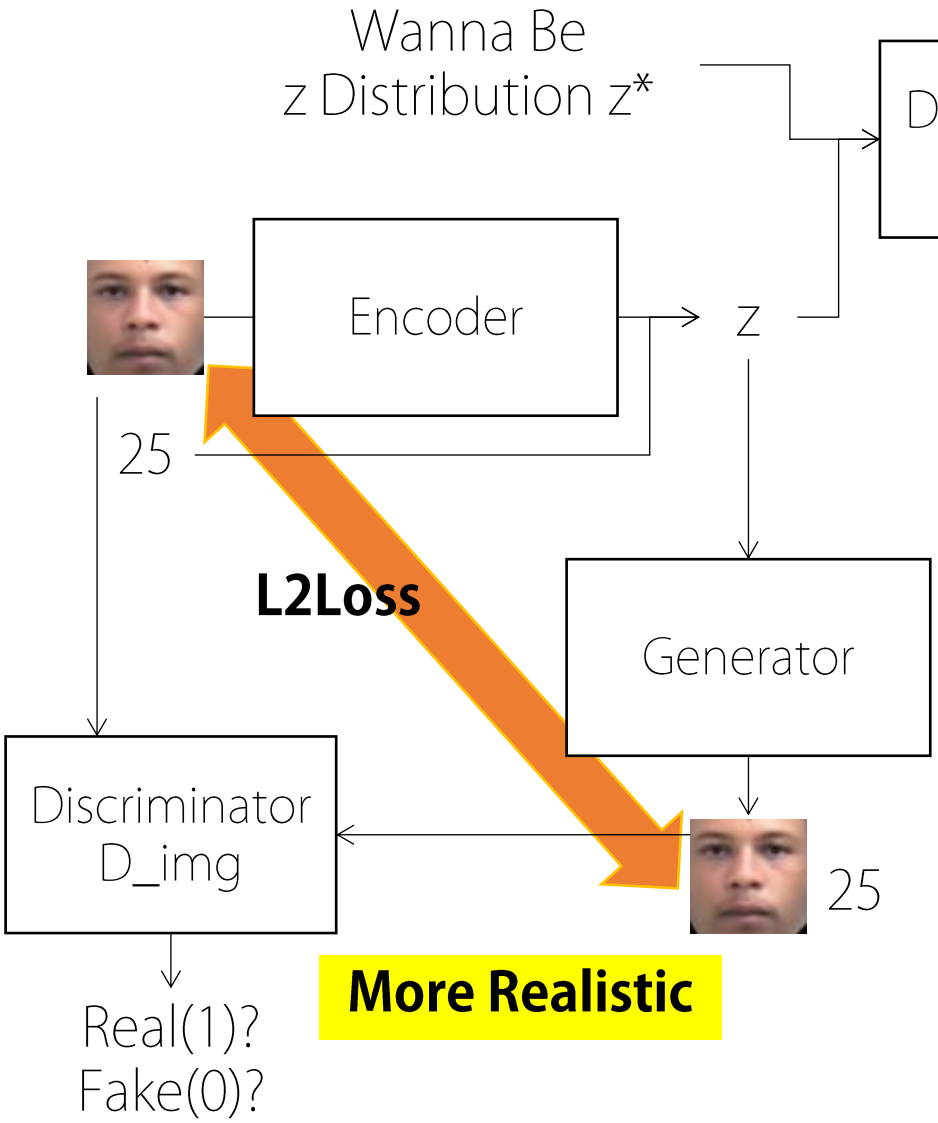


$$E_{x,l}\left(logD_{img}(x,l)\right)$$
$$+E_{x,l}(\log\left(1-D_{img}\left(G(E(x),l)\right)\right))$$

Face Aging Prediction

Age Prediction/Regression by Conditional Adversarial Autoencoder

CVPR 2017

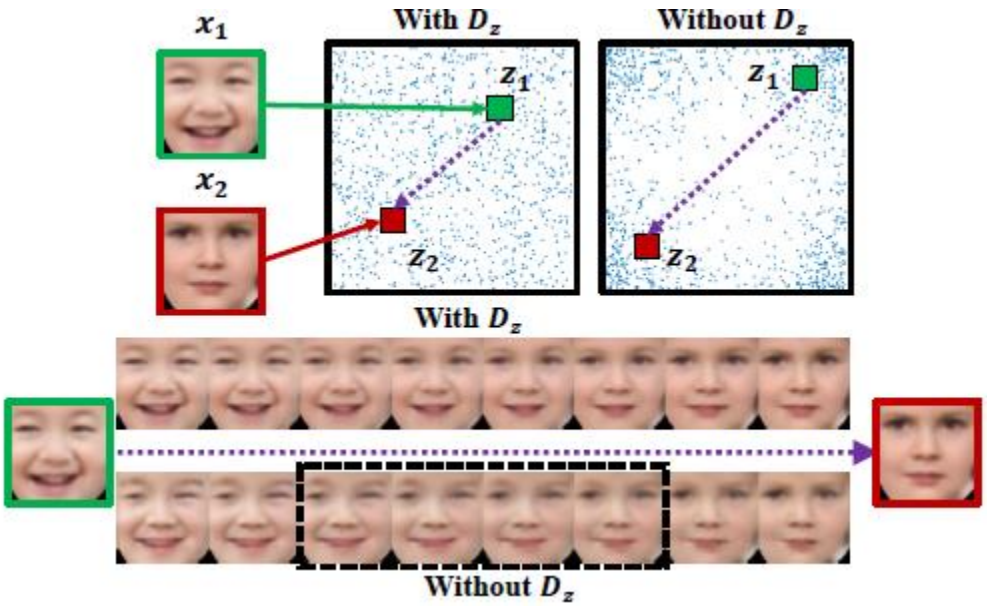


$$L_2\big(x, G(E(x), l)\big) + TV\big(G\big(E(x)\big), l\big)$$
$$+E_{z^*}(\log D_z(z^*))$$
$$+E_x\left(\log\left(1 - D_z\big(E(x)\big)\right)\right)$$
$$+E_{x,l}\left(logD_{img}(x, l)\right)$$
$$+E_{x,l}(\log\left(1 - D_{img}\big(G(E(x), l)\big)\right))$$

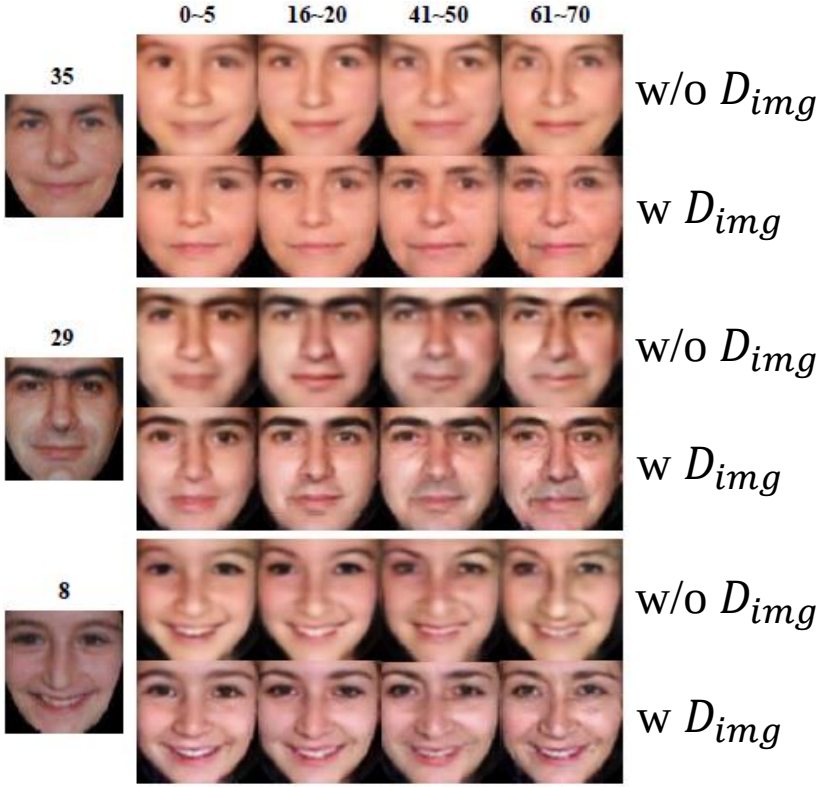
Face Aging Prediction

Age Prediction/Regression by Conditional Adversarial Autoencoder

CVPR 2017



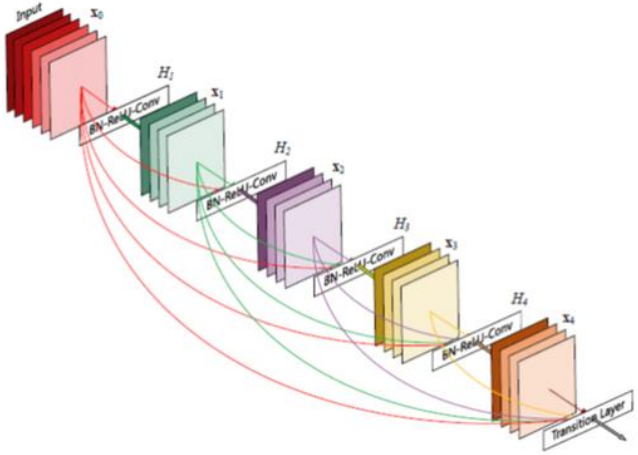
Effect of D_z

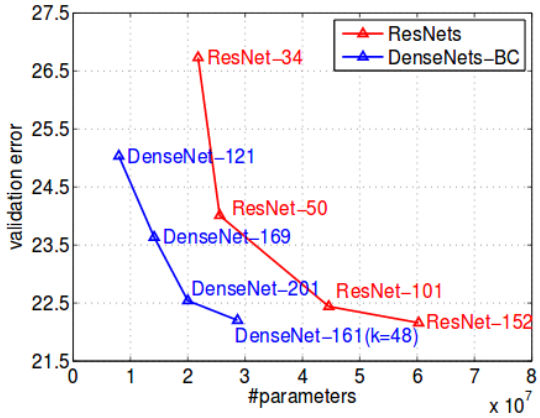


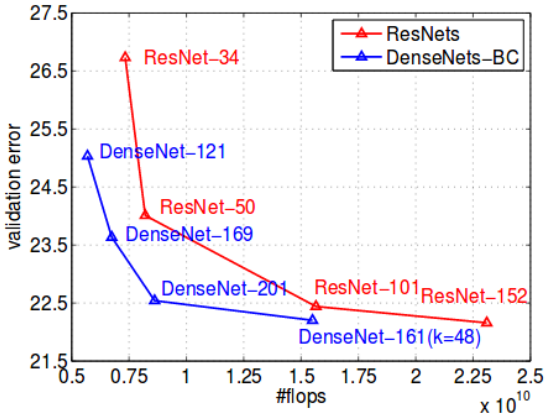
Effect of D_{img}

One More Thing

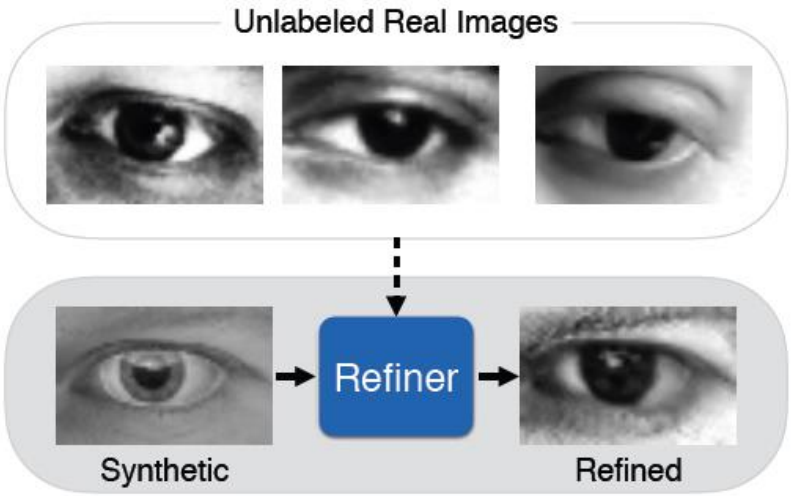
CVPR 2017 Best Paper







DenseNet, Facebook



S+U Learning, Apple

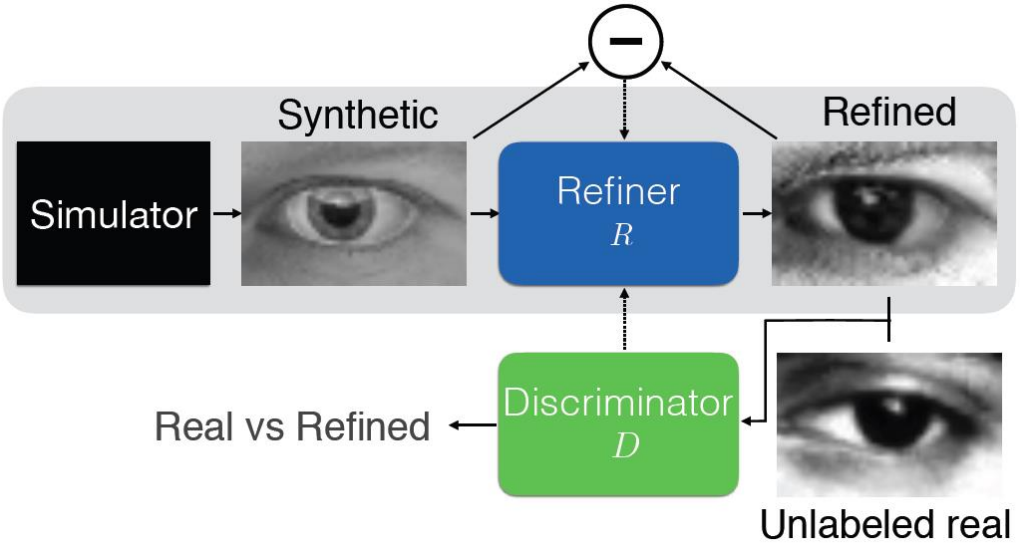
Apple Machine Learning Journal

Improving the Realism of Synthetic Images

Vol. 1, Issue 1 • July 2017

Most successful examples of neural nets today are trained with supervision. However, to achieve high accuracy, the training sets need to be large, diverse, and accurately annotated, which is costly. An alternative to labelling huge amounts of data is to use synthetic images from a simulator. This is cheap as there is no labeling cost, but the synthetic images may not be realistic enough, resulting in poor generalization on real test images. To help close this performance gap, we've developed a method for refining synthetic images to make them look more realistic. We show that training models on these refined images leads to significant improvements in accuracy on various machine learning tasks.

Read the article ›



Key Idea:

Synthetic Sample with Label → Bad Quality

using Unlabeled Real Image + **GAN**

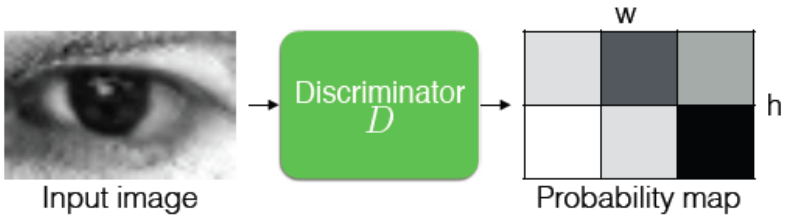
→ Good Quality Synthetic Sample

→ Data Augmentation / Semi-Supervised L

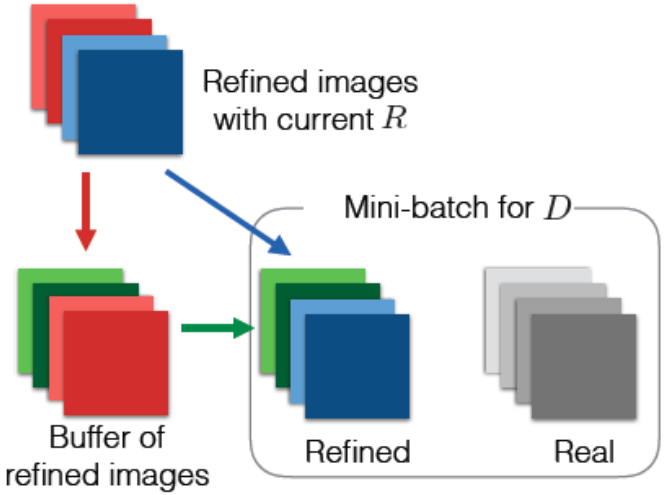
$$L_D = log\left(D\left(X_{refined}\right)\right) + \log(1 - D(X_{unlabeled}))$$

$$L_R = \log\left(1 - D\left(X_{refined}\right)\right) + L1(CNN\left(X_{refined}\right) - CNN(X_{syn}))$$

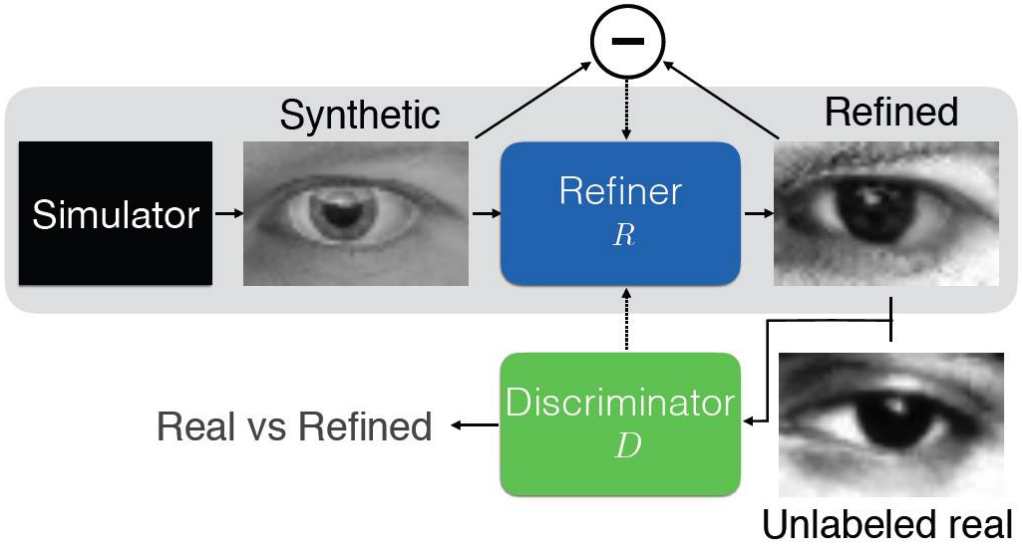
1. Perceptual Loss? (Self-Reg. Term)



2. PatchGAN? (Local Adversarial)



3. Minibatch Discrimination? (using History)



One Contribution
: Applied GAN → Synthetic Refinement

Conclusion

GAN → Adversarial Training → Reinforcement

Generative adversarial nets

I Goodfellow, J Pouget-Abadie, M Mirza... - Advances in neural ..., 2014 - papers.nips.cc

Abstract We propose a new framework for estimating generative models via adversarial nets, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample

1021회 인용 관련 학술자료 전체 18개의 버전 인용 저장

Ian Goodfellow's GAN Tutorial Will be held in ICCV2017 22th October

Introduction to GANs

Semi-supervised learning with GANs

Practical tricks and tips for training GANs

Plug and Play Generative Models

Latent variable inference for GANs

Approximate Nash equilibria in neural net GANs

Evaluating generative models

Interactive image generation with GANs

Connections between adversarial training and reinforcement learning

Importance of GANs

Hand Crafted CV

Hand Crafted Feature + Classifier

Full ML (a.k.a Deep Learning)

Full ML + Hand Crafted Training Model
(Unsupervised, Semi-supervised, Reinforcement)

Importance of GANs

Generative Models + **Adversarial Training**

GMM, HMM
Naïve Bayes
RBM, Autoencoders

Core Idea

References

* Image to Image Translation
- Image to Image Translation with Conditional Adversarial Networks, CVPR 2017
- Unpaired Image to Image Translation using Cycle-consistent Adversarial Networks, ICCV 2017
- Learning to Discover Cross-domain Relations with Generative Adversarial Networks, ICML 2017

* Single Image Super-Resolution
- Photo-realistic Single Image Super-Resolution using a Generative Adversarial Networks, CVPR 2017
- EnhanceNet: Single Image Super Resoluton through Automated Texture Synthesis, ICCV 2017

* Feature Upscaling & 3D Reconstruction
- Perceptual Generative Adversarial Networks for Small Object Detection, CVPR 2017
- Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks, CVPR 2017

* Pose Estimation, Face Recognition, Pedestrian Detection using Data Augmentation
- Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation, CVPR 2017
- Distangled Representation Learning GAN for Pose-Invariant Face Recogntion, CVPR 2017
- Expecting the Unexpected: Training Detectors for Unusual Pedestrians with Adversarial Imposters, CVPR 2017

* Modeling and Feature Extraction
- Age Prediction/Regression by Conditional Adversarial Autoencoder, CVPR 2017
-
* One More Thing
- Learning from Simulated and Unsupervised Images through Adversarial Training, CVPR 2017 Best Paper

More Information:

https://taeoh-kim.github.io/blog

Thank you!

Any Questions?