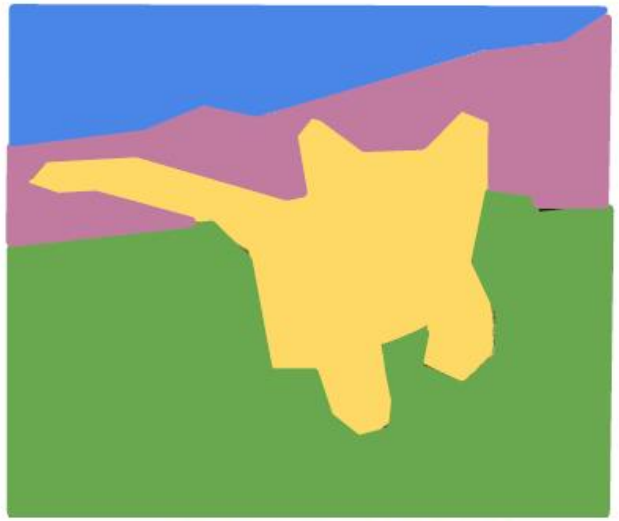
Localization & Segmentation

Computer Vision Task

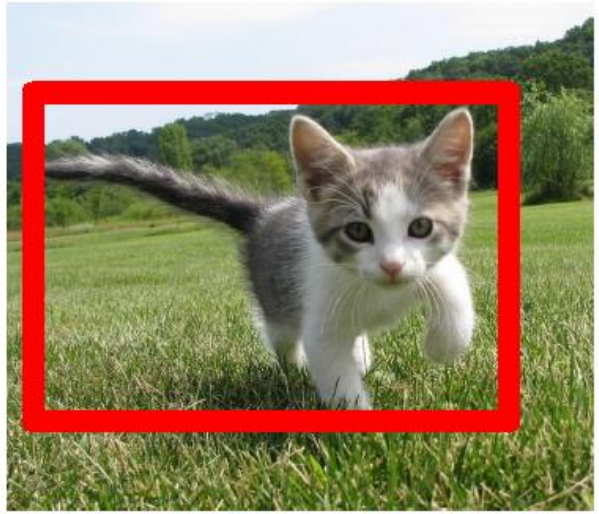
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

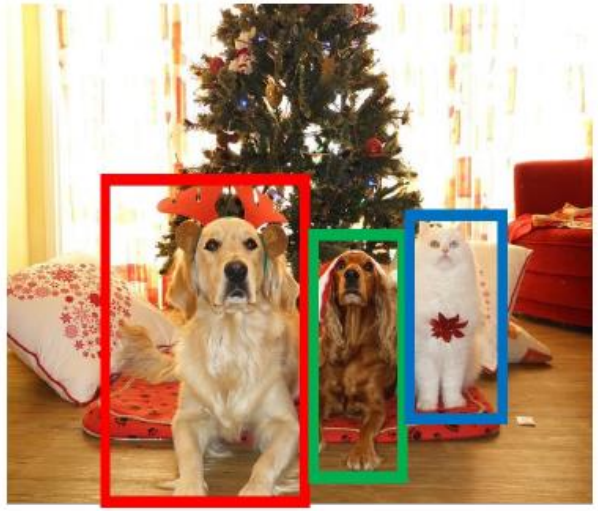
Classification + Localization



CAT

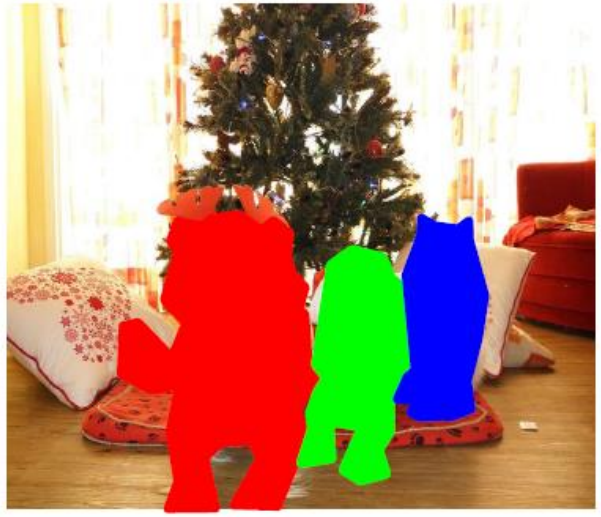
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Classification + Localization

Classification: C classes
Input: Image
Output: Class label
Evaluation metric: Accuracy

CAT

Localization:
Input: Image
Output: Box in the image (x, y, w, h)
Evaluation metric: Intersection over Union

(x, y, w, h)

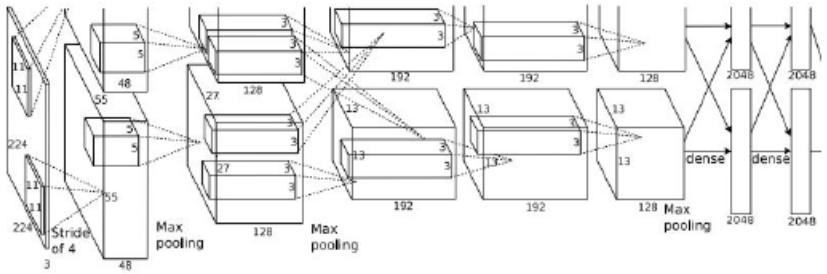
Classification + Localization: Do both

Intersection Over Union

$$\text{Jaccard Overlap} = \text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Classification + Localization

This image is CC0 public domain



Correct label:
Cat

Fully Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Multitask Loss

Loss

Vector:
4096

Fully Connected:
4096 to 4

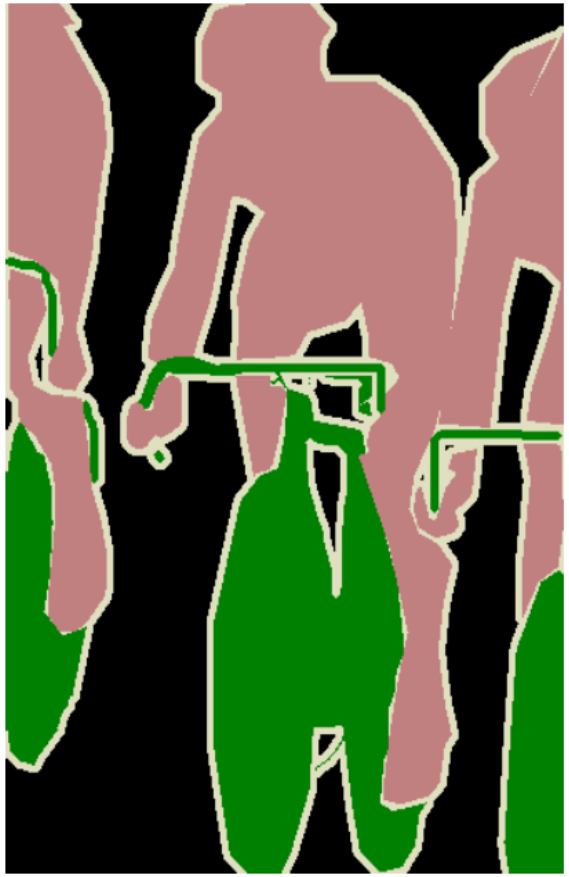
Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

Treat localization as a regression problem!

Semantic Segmentation



- PASCAL VOC segmentation
  - 10K images, 20 classes + bgnd

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

- MS COCO
  - 100K images, 80 classes + bgnd

Microsoft COCO
Common Objects in Context

U-Net: Convolutional Networks for Biomedical Image Segmentation

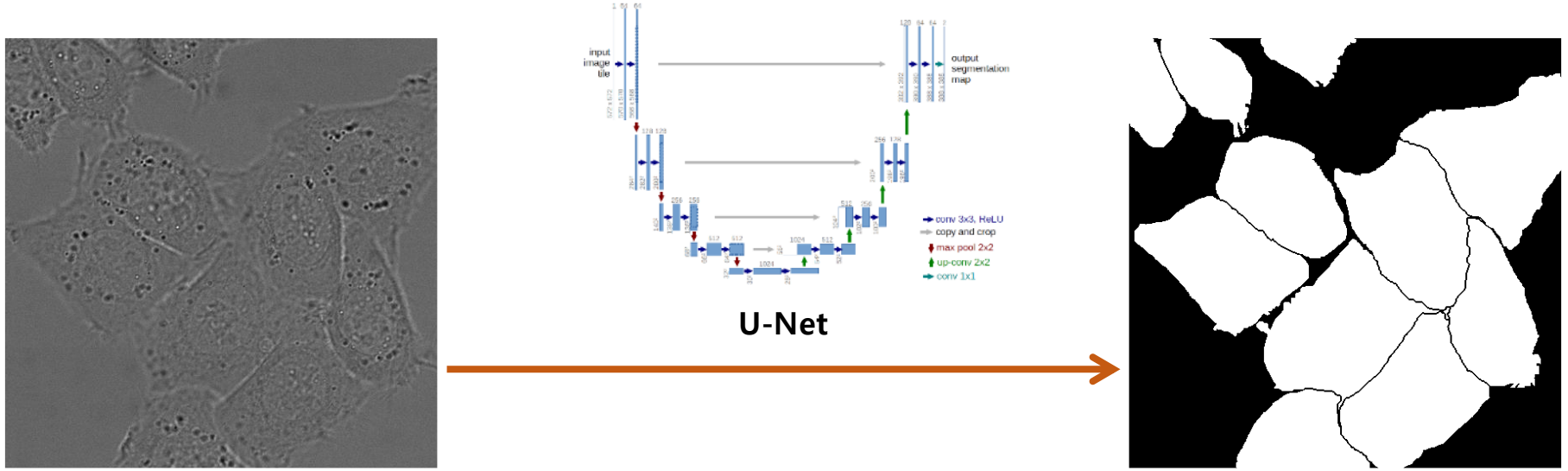
U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

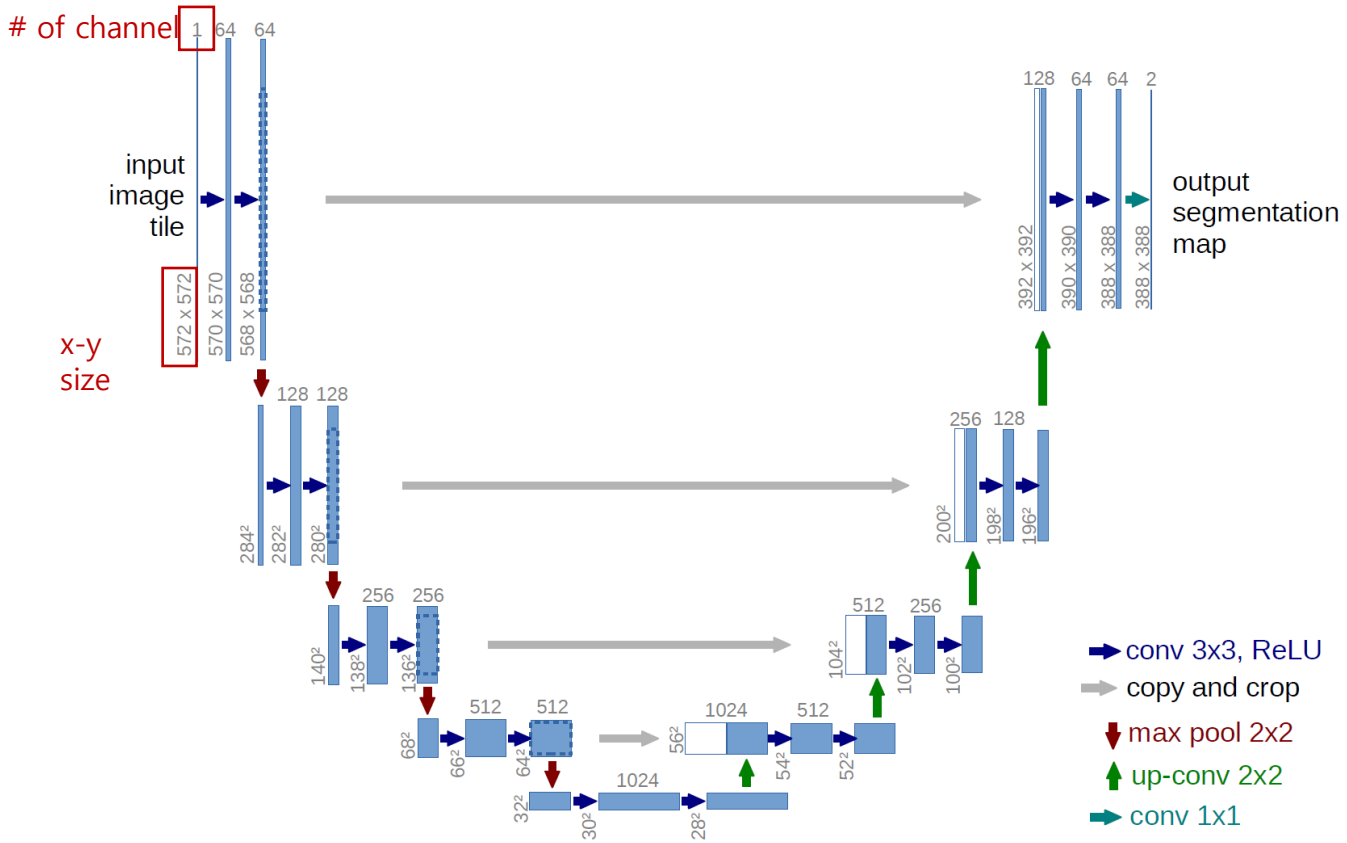
Abstract. There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window

Biomedical Image Segmentation with U-net

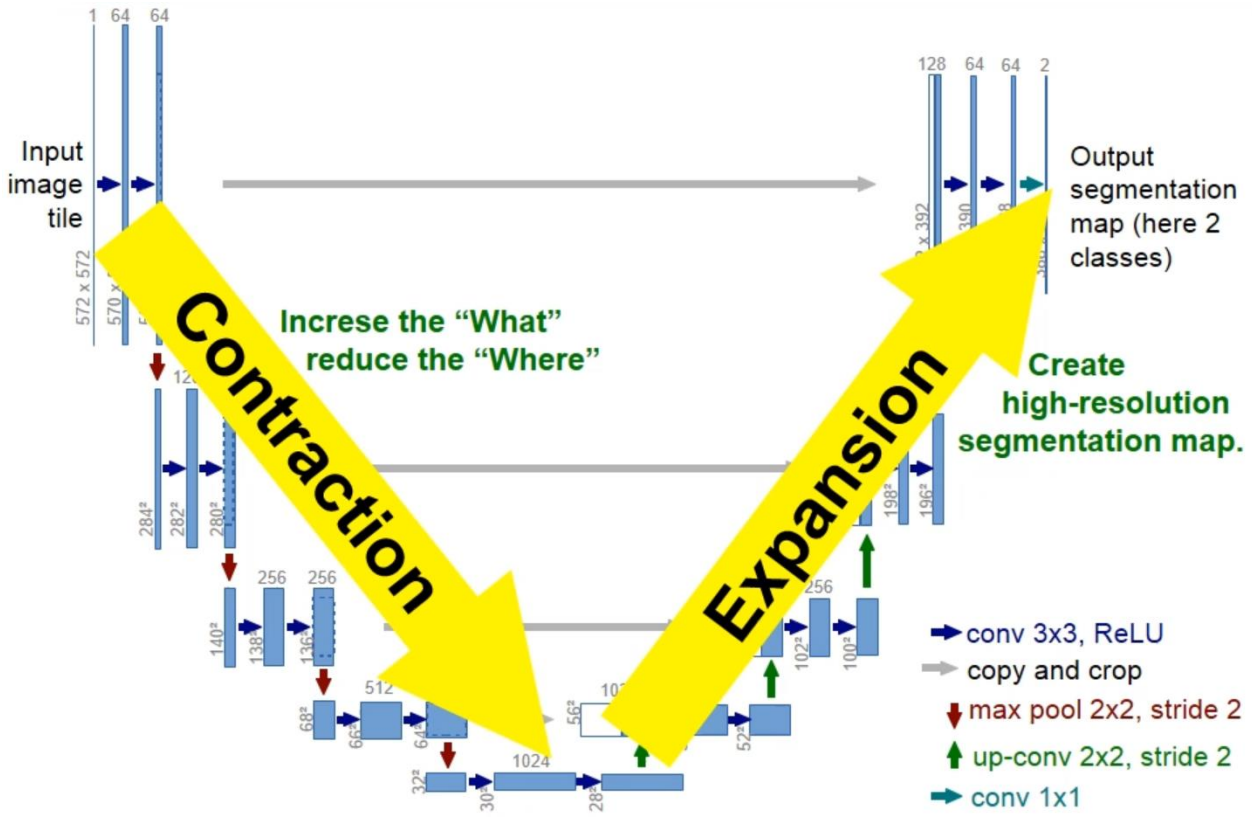


- U-Net learns segmentation in an end-to-end setting
- Very few annotated images (approx. 30 per application)

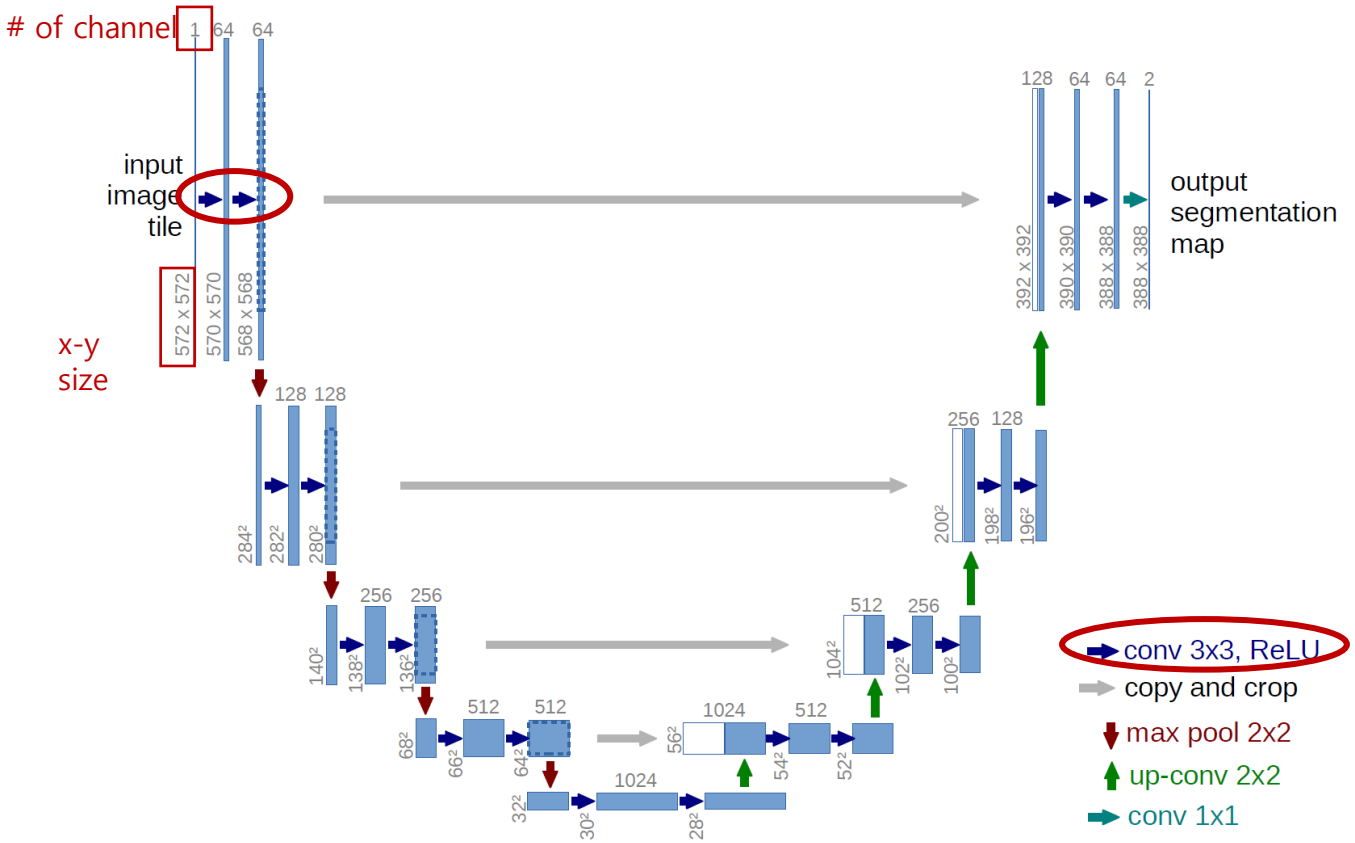
U-Net Architecture



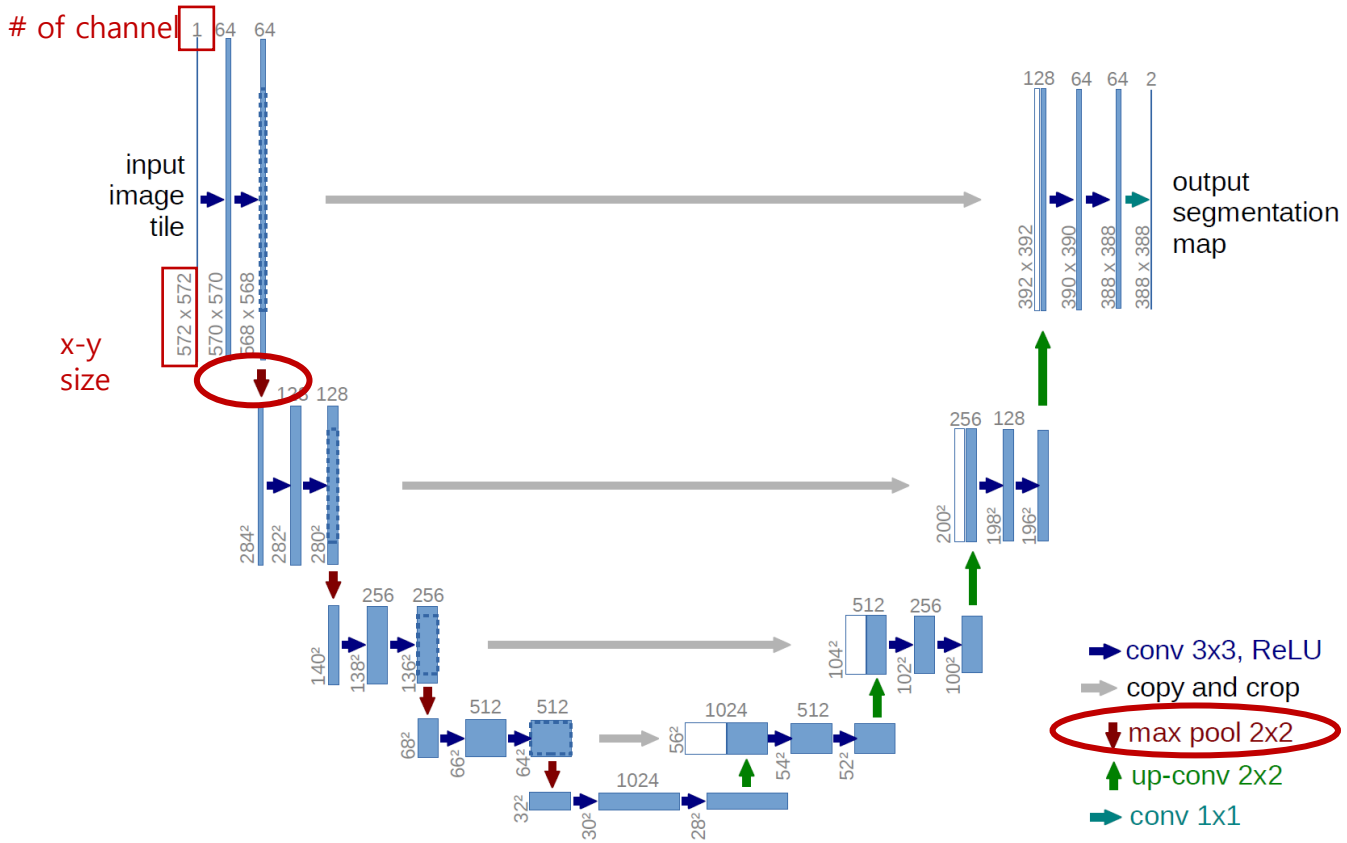
U-Net Architecture



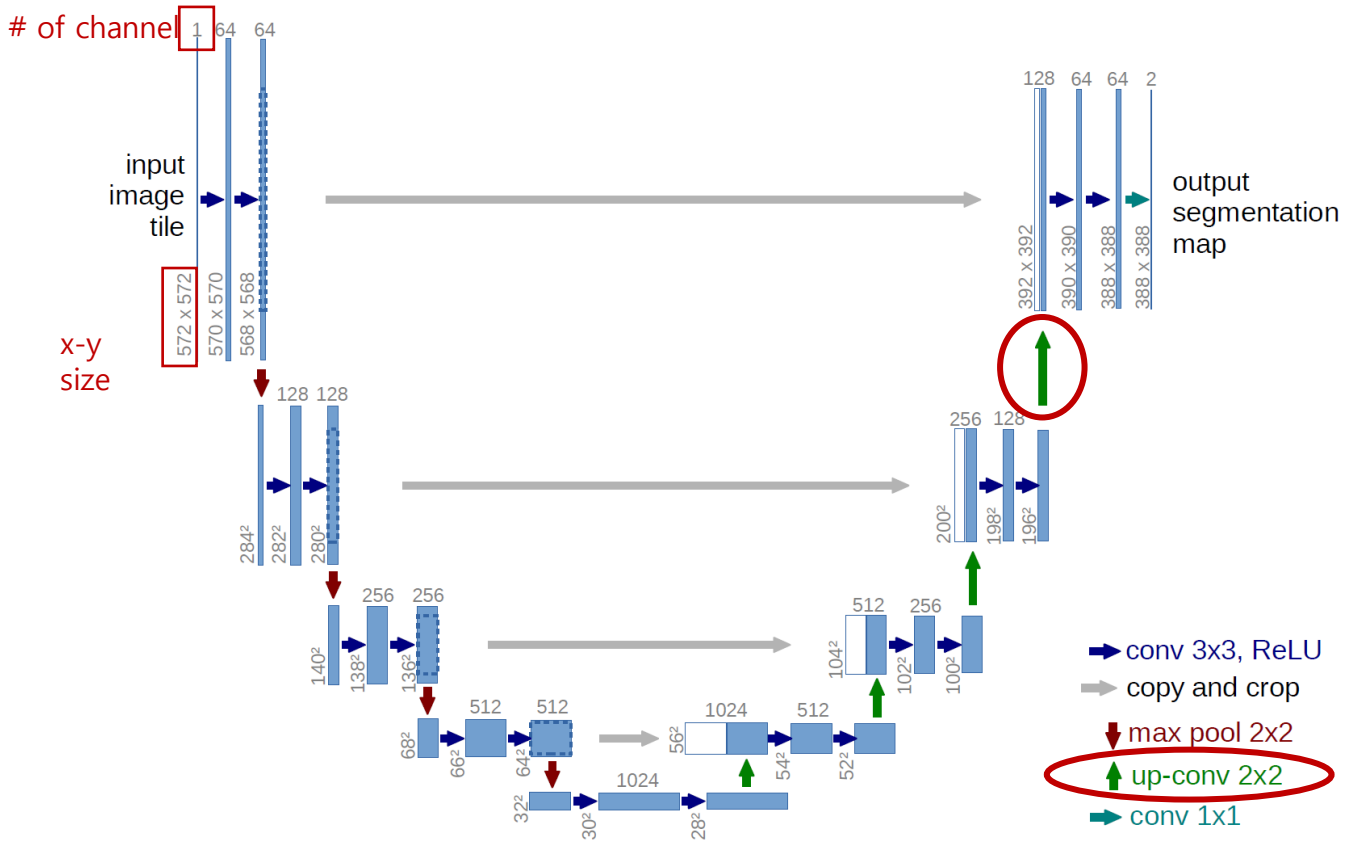
U-Net Architecture



U-Net Architecture

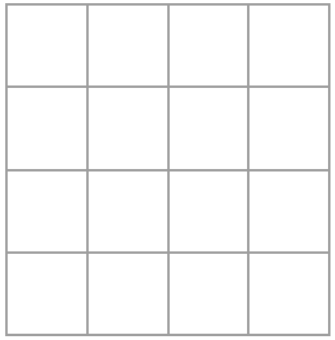


U-Net Architecture

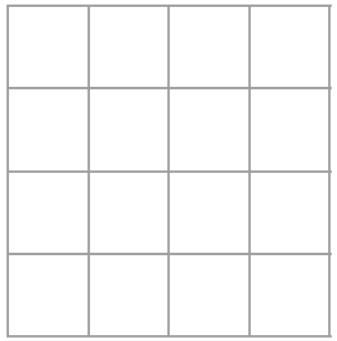


Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 1 pad 1



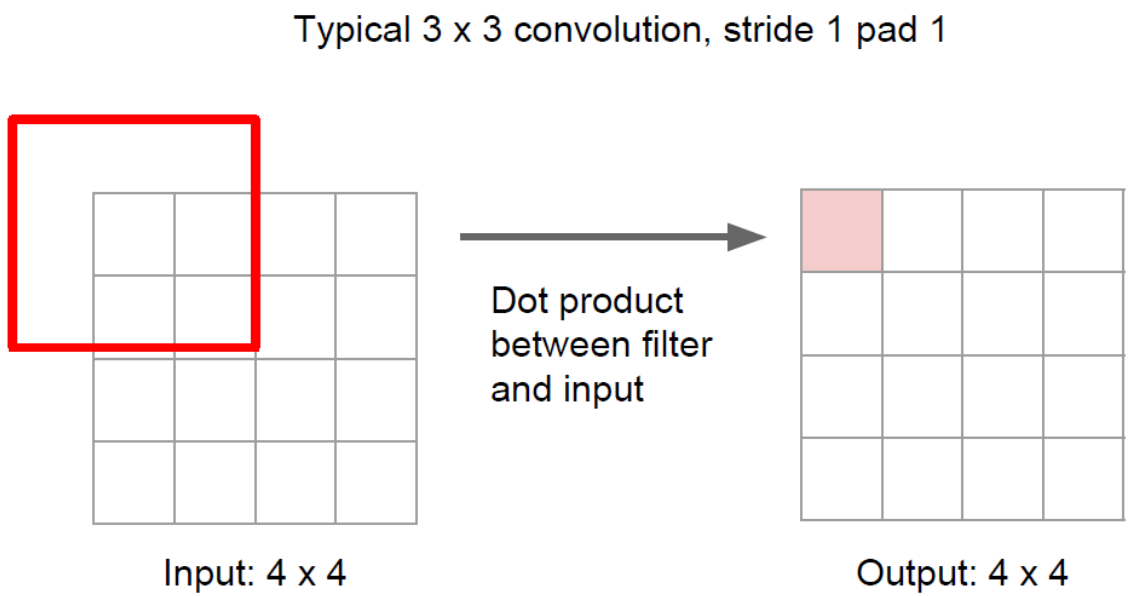
Input: 4 x 4



Output: 4 x 4

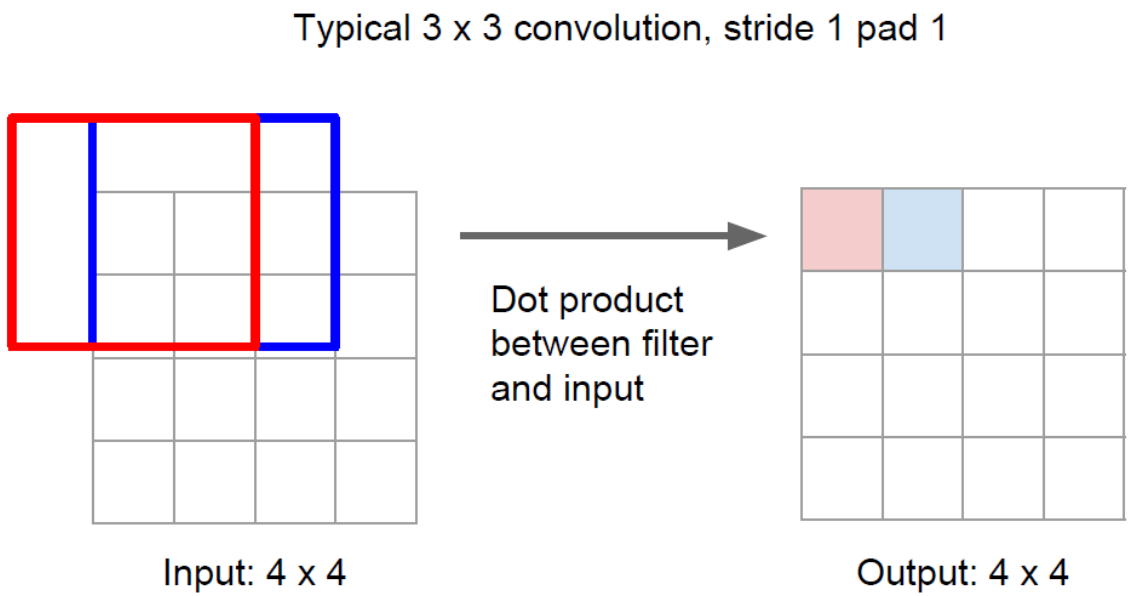
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



Slide from Stanford CS231n

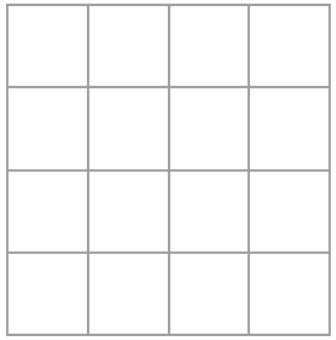
Learnable Upsampling: "Deconvolution"



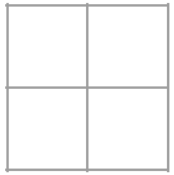
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”

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



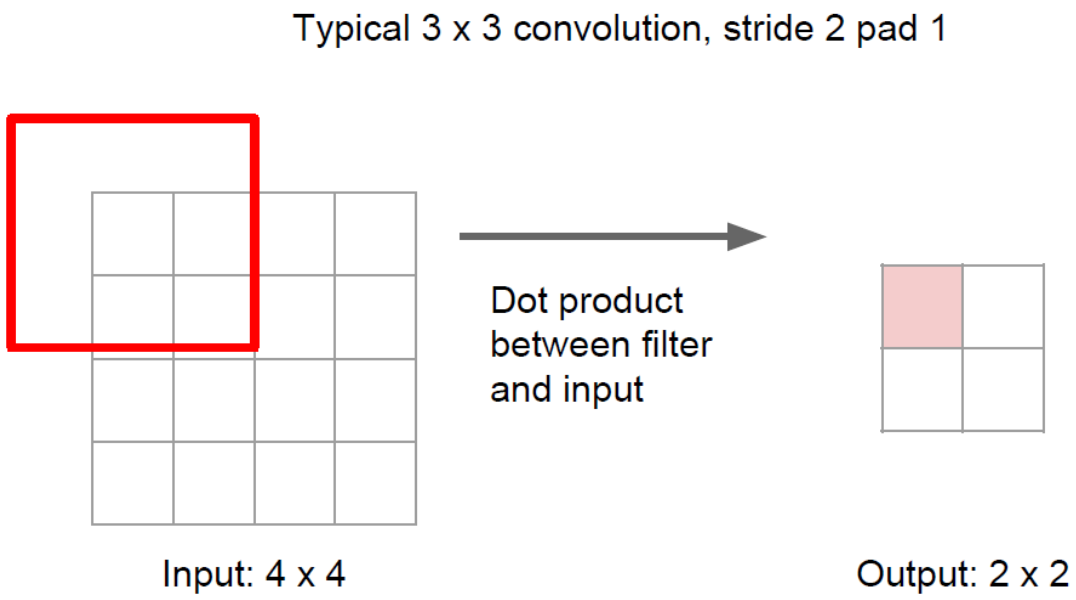
Input: 4 x 4



Output: 2 x 2

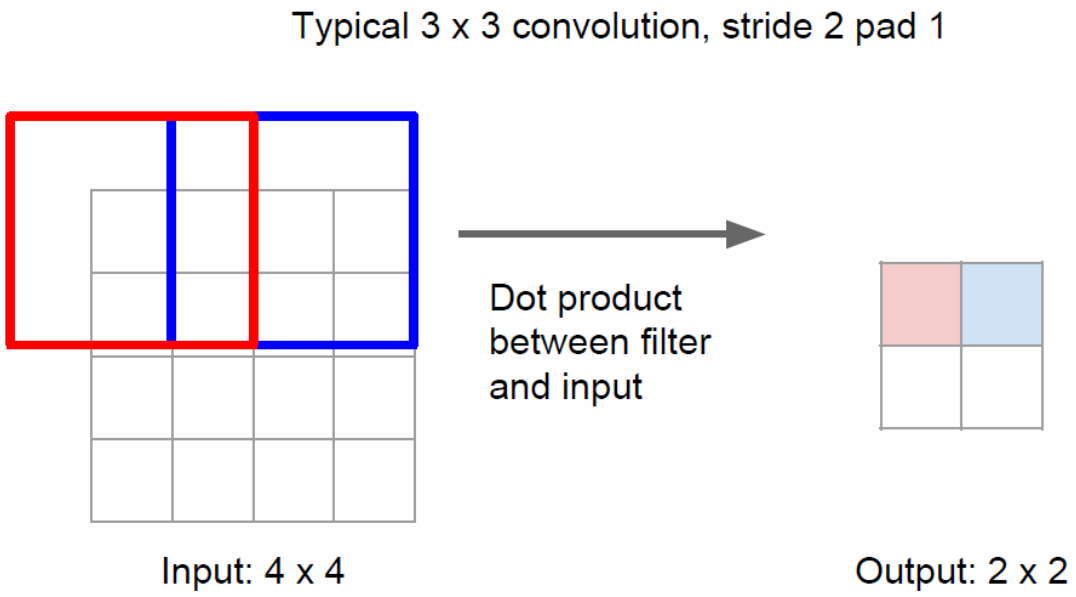
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



Slide from Stanford CS231n

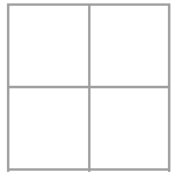
Learnable Upsampling: “Deconvolution”



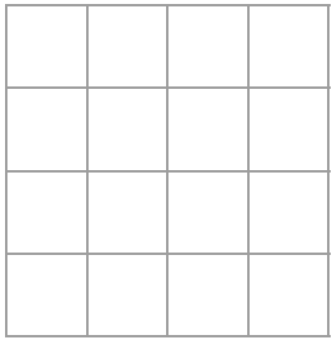
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1



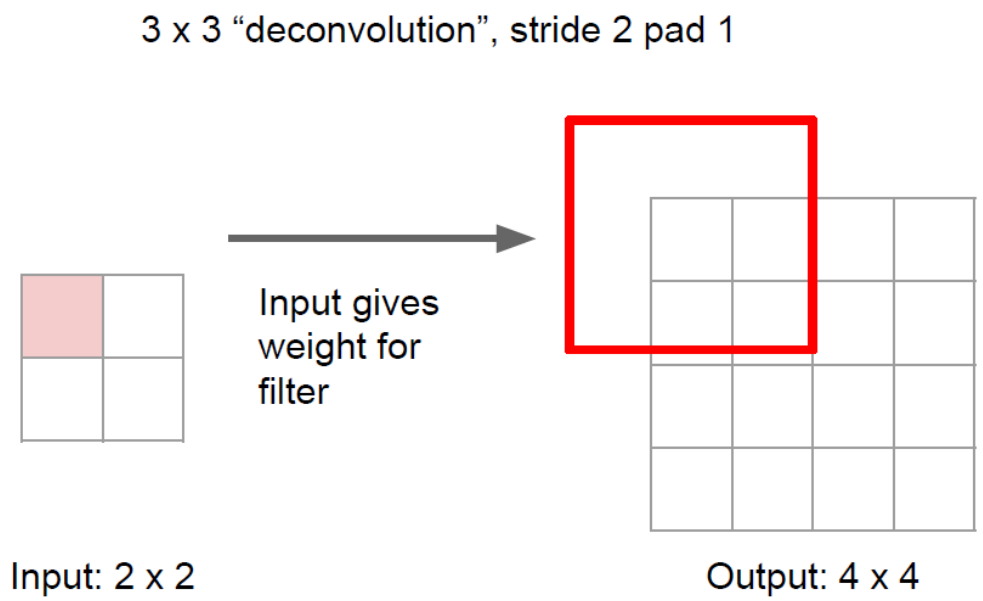
Input: 2 x 2



Output: 4 x 4

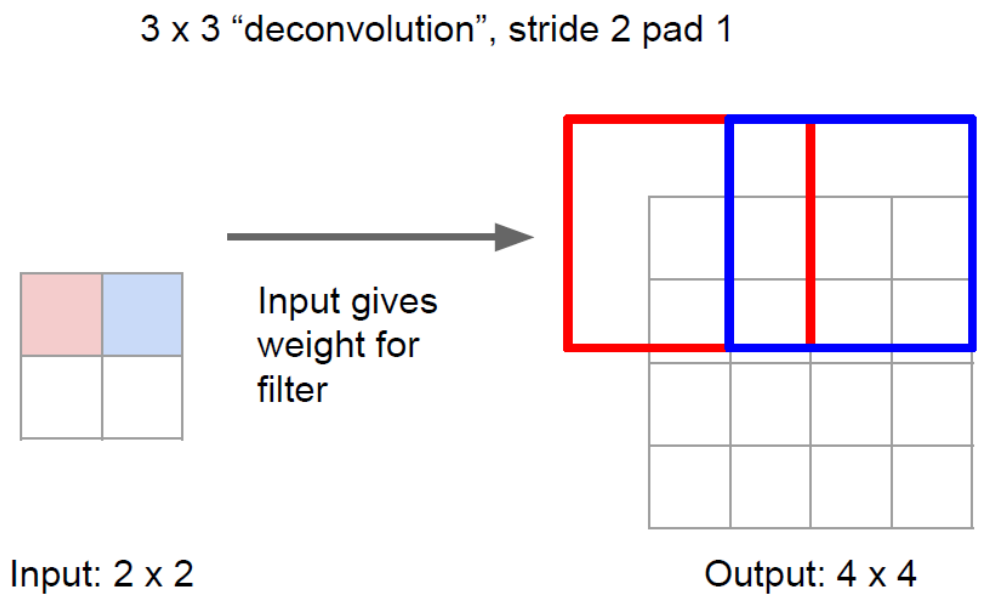
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



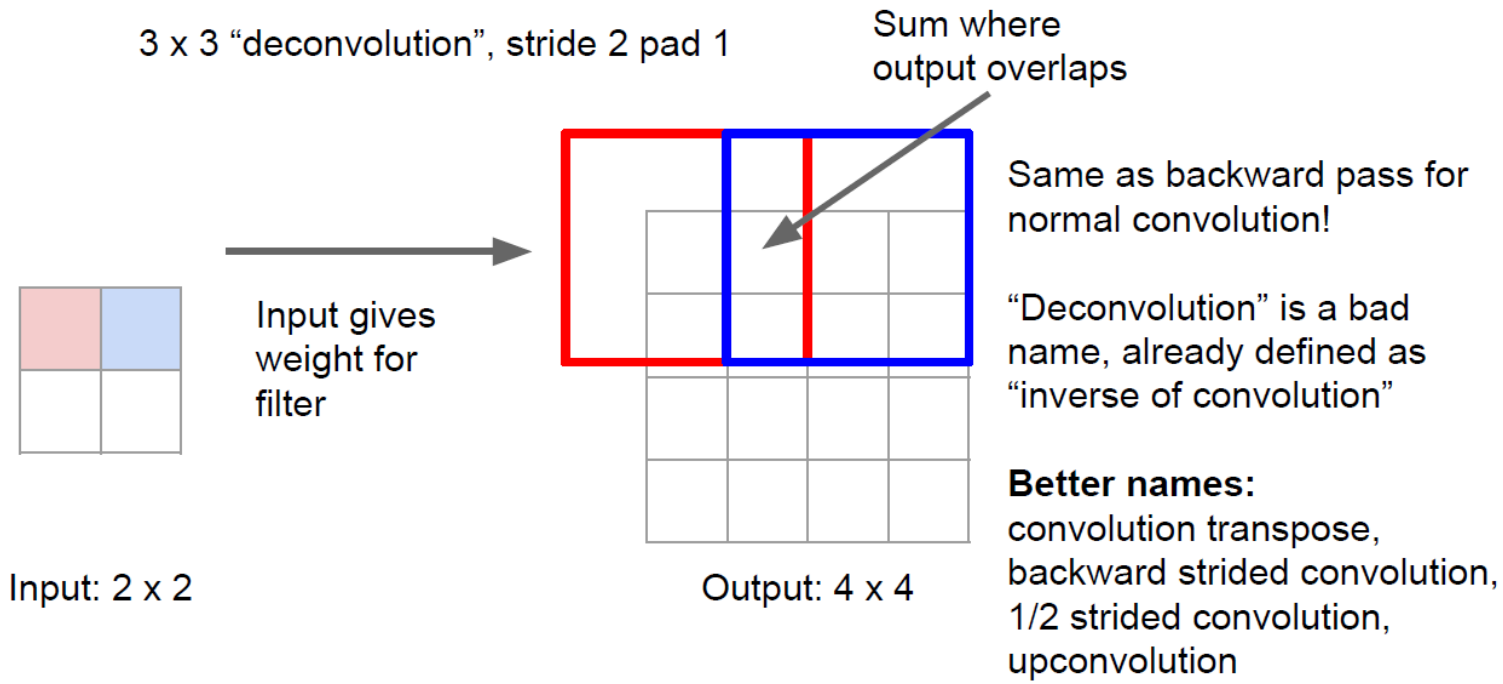
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”



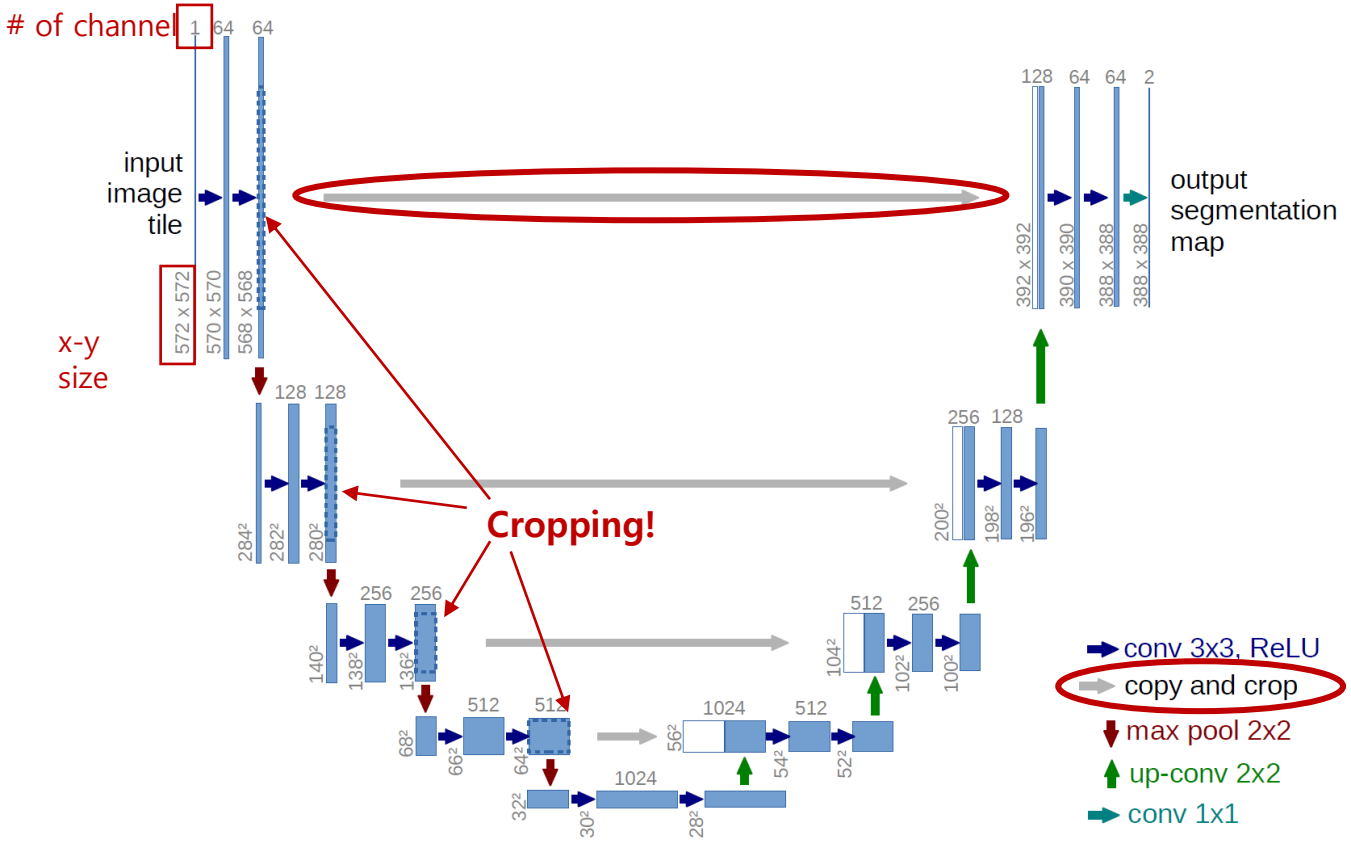
Slide from Stanford CS231n

Learnable Upsampling: “Deconvolution”

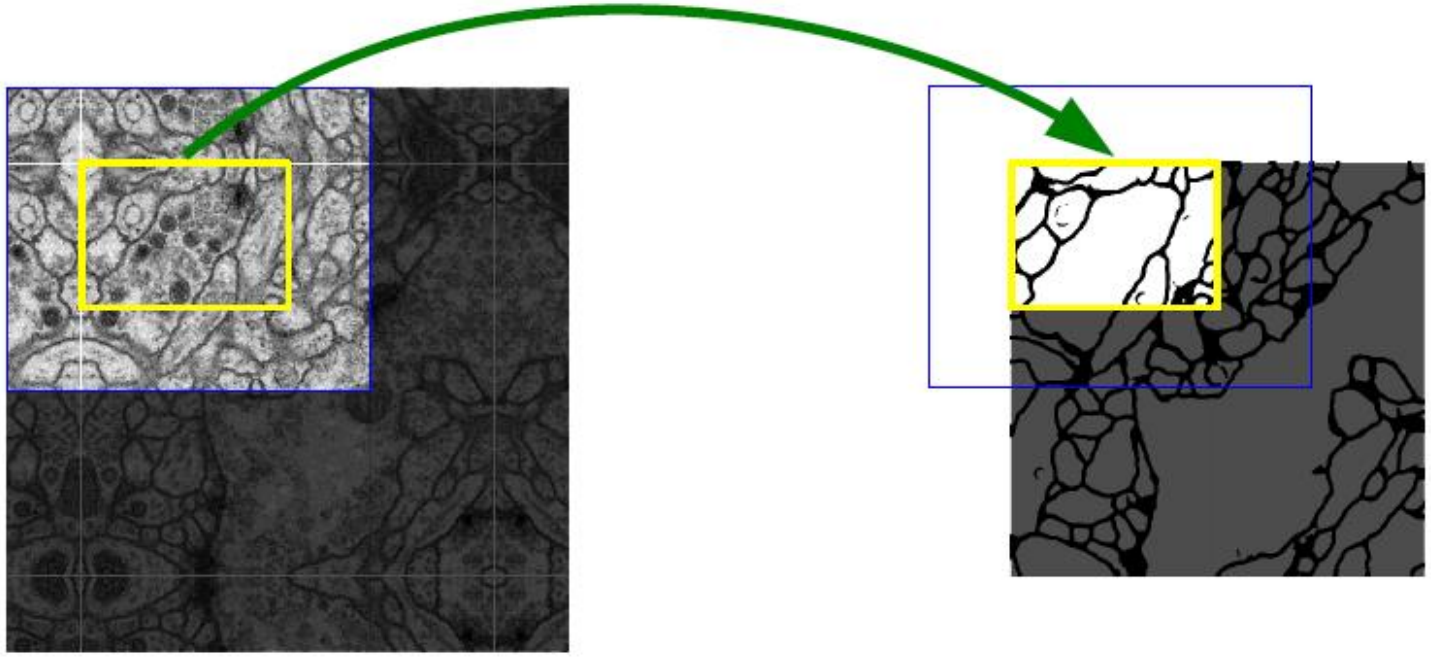


Slide from Stanford CS231n

U-Net Architecture

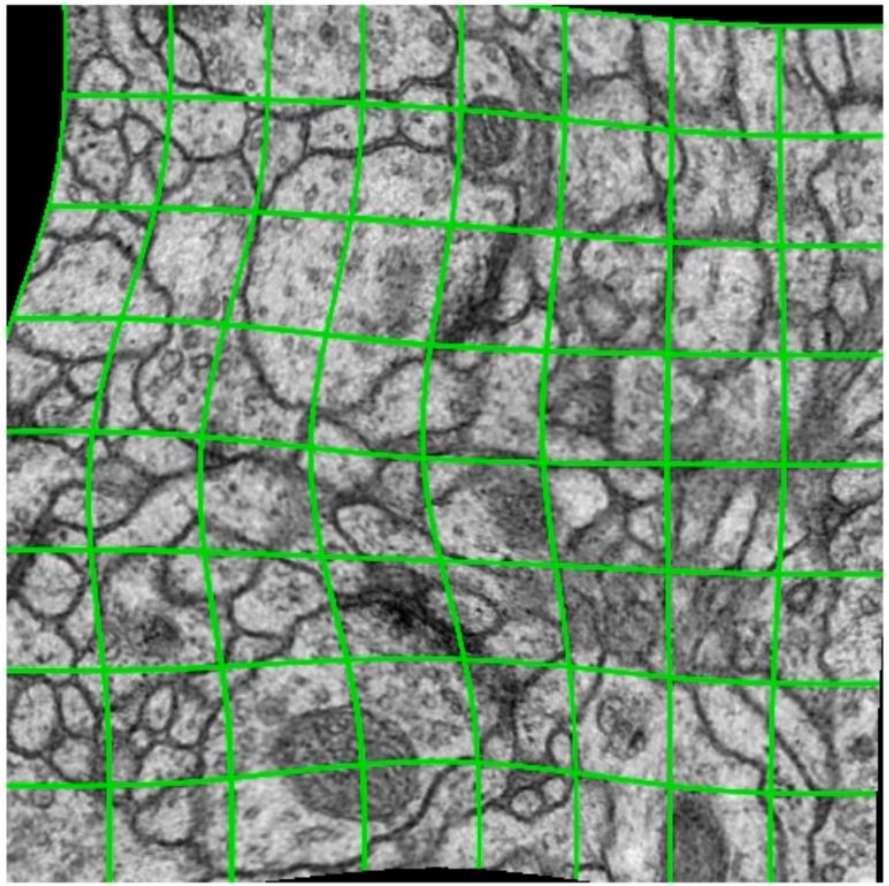


Overlap-tile strategy for arbitrary large images

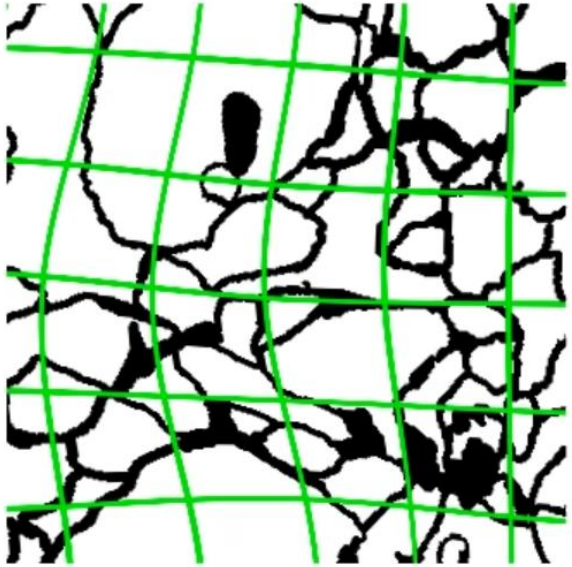


- Segmentation of the yellow area uses input data of the blue area
- Raw data extrapolation by mirroring

Augment Training Data using Deformations

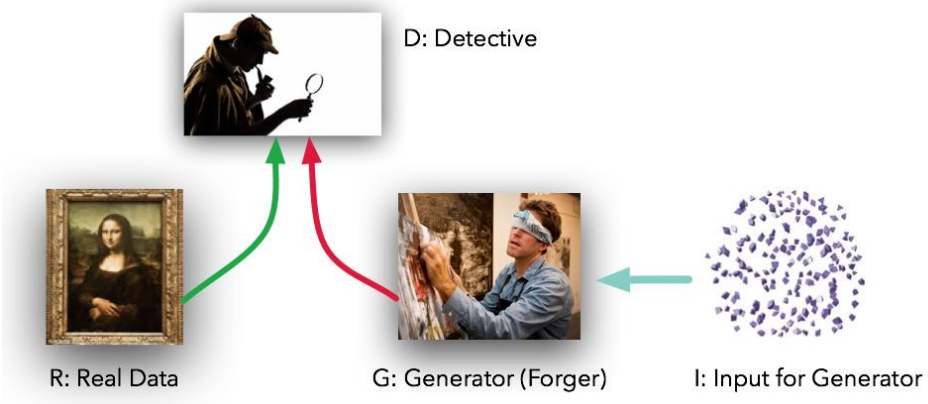


resulting deformed image
(for visualization: no rotation, no shift, no extrapolation)



correspondingly deformed
manual labels

Generative Adversarial Network

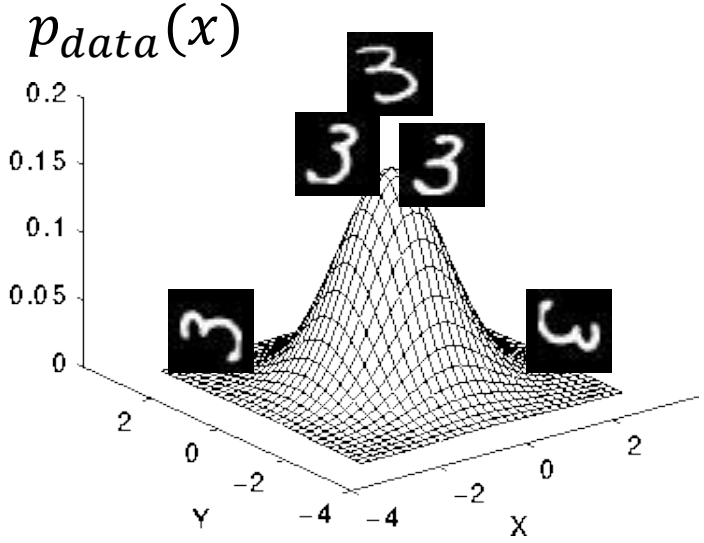


Generative Adversarial Network
-NIPS 2014

Generative Model

- Data:  ...

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

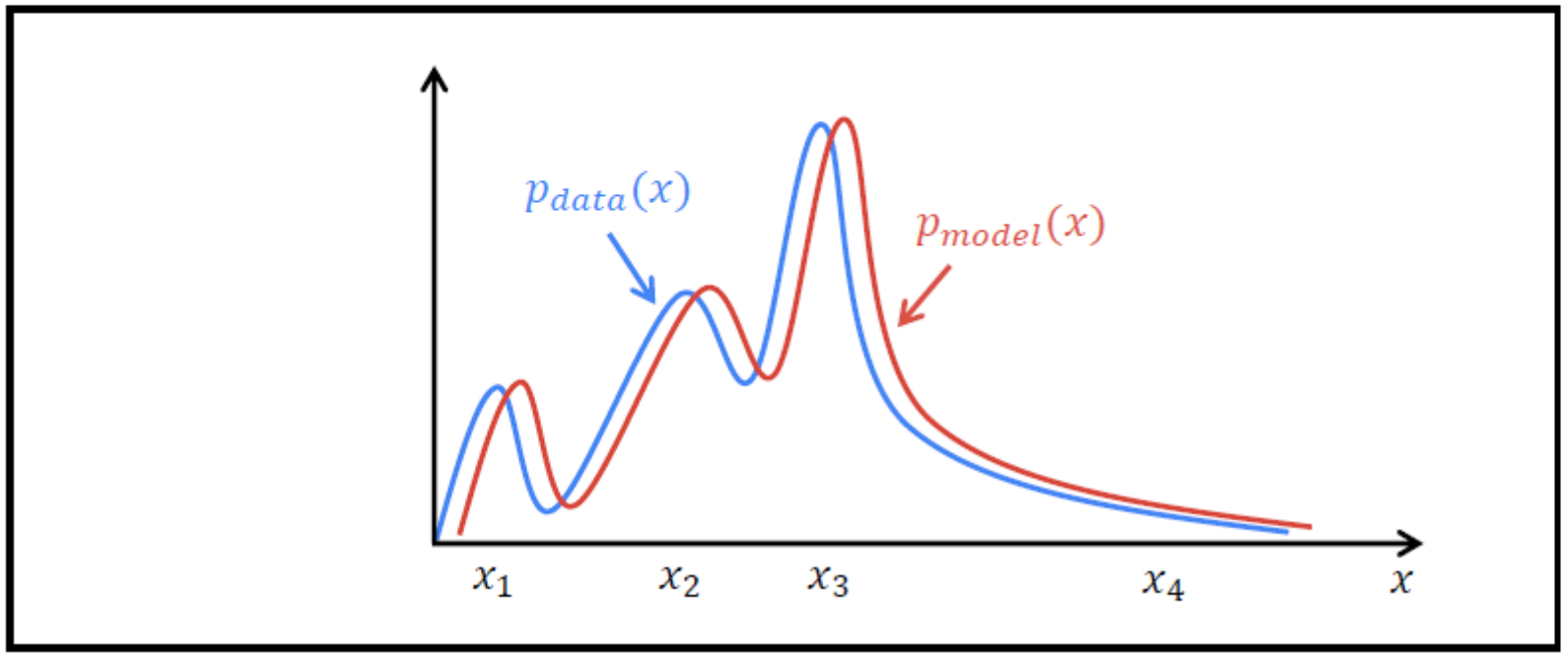


Generative Model

Distribution of images generated by the model

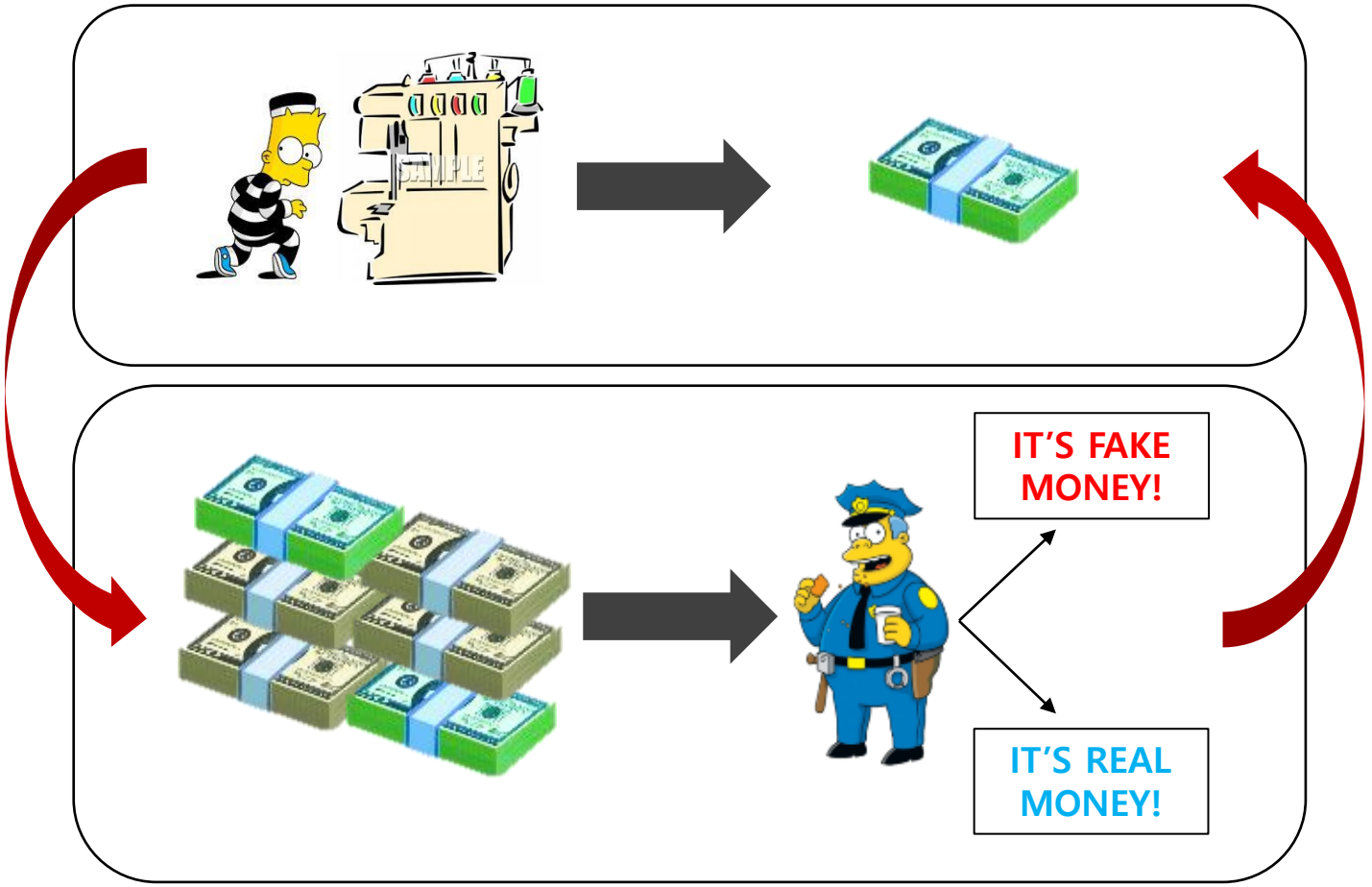
The goal of the generative model is to find a $p_{model}(x)$ that approximates $p_{data}(x)$ well.

Distribution of actual images

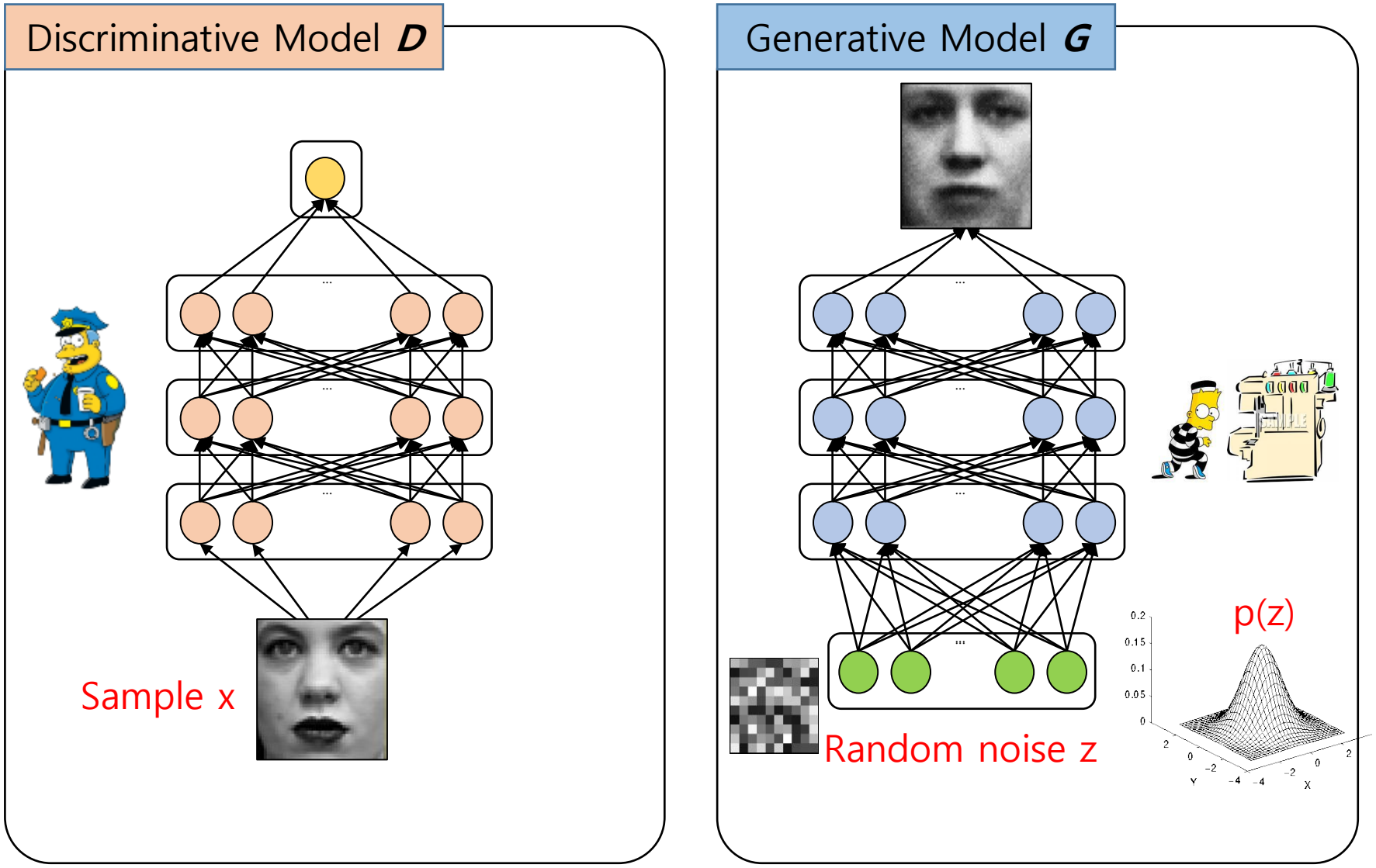


Generative Adversarial Network

- Counterfeiters vs Police Game



Generative Adversarial Network

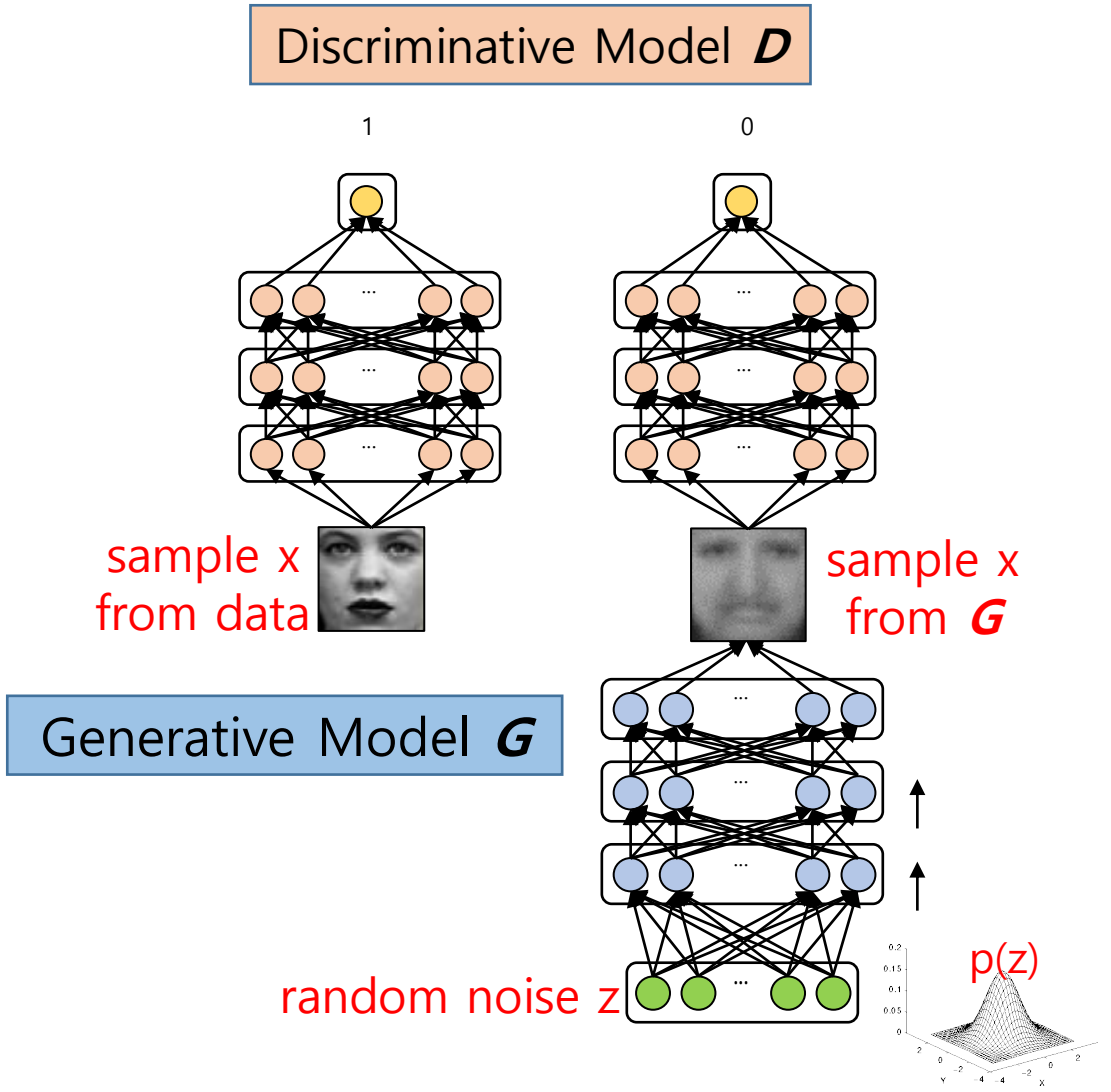


Generative Adversarial Network

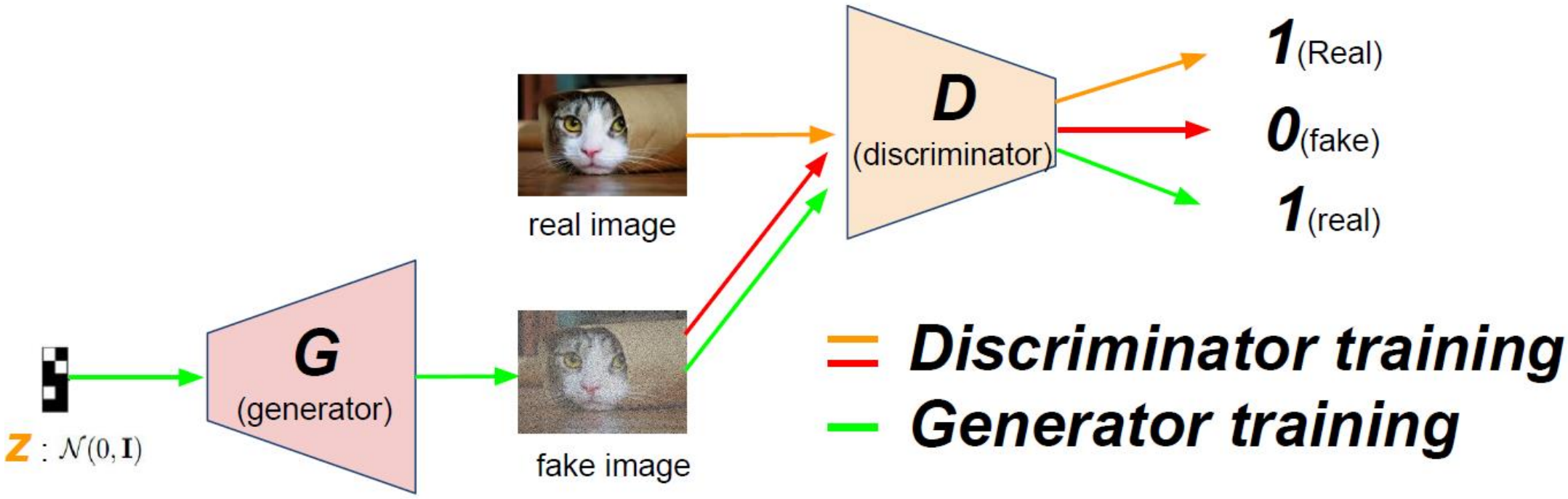
- Discriminative Model D
  - Try to classify the sample x
  - D(x)=1 when x from Data
  - D(x)=0 when x from G (generator)

Differentiable function represented by a multilayer perceptron with parameters

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



Training GAN

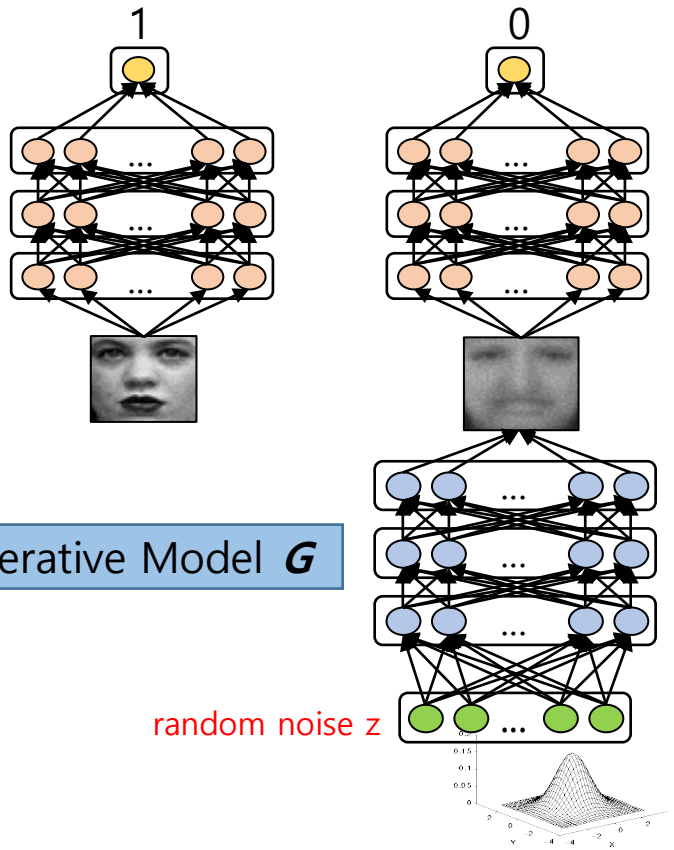


Generative Adversarial Network

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

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

Discriminative Model ***D***



Generative Model ***G***

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

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

end for

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

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

end for

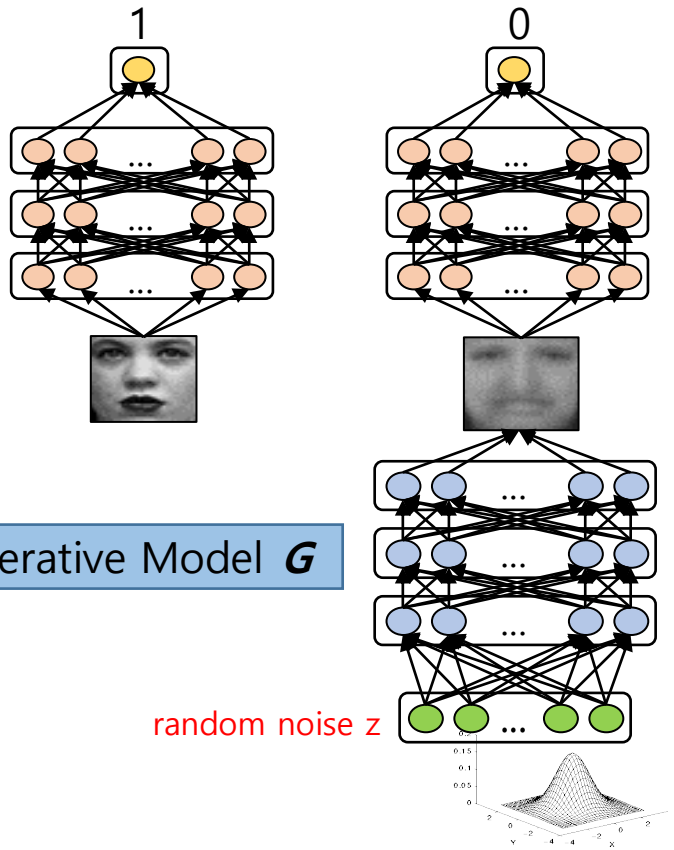
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Generative Adversarial Network

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

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

Discriminative Model ***D***



Generative Model ***G***

- Fixed G, maximize V:

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

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

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

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

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

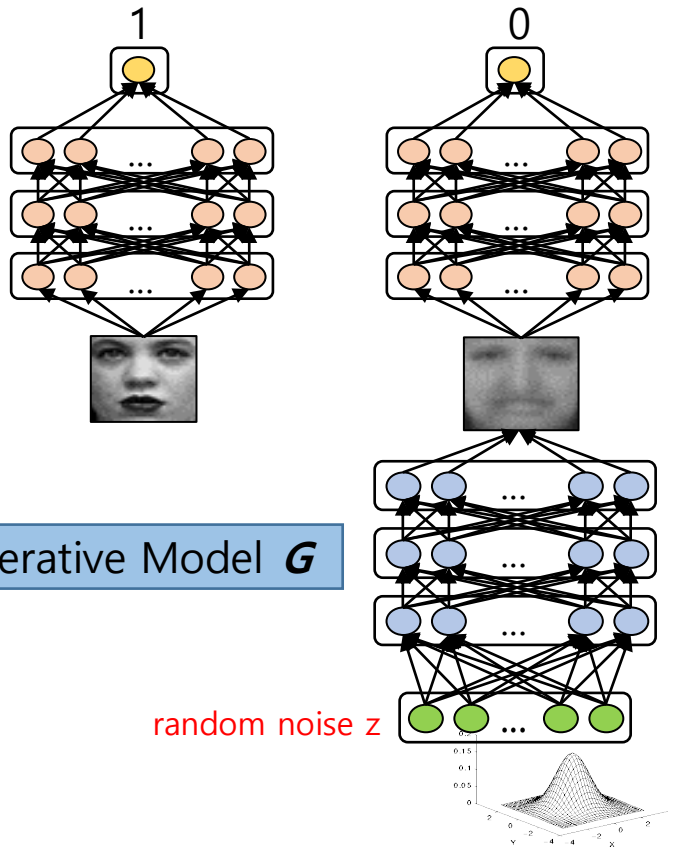
Stochastic Gradient

Generative Adversarial Network

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

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

Discriminative Model ***D***



- Fixed D, minimize V(G):

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

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

Stochastic Gradient

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

Generative Adversarial Network

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

for number of training iterations **do**

for k steps **do**

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

Update D

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

end for

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

Update G

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

end for

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

Meaning of GAN Loss

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

Objective function of GANs

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

same

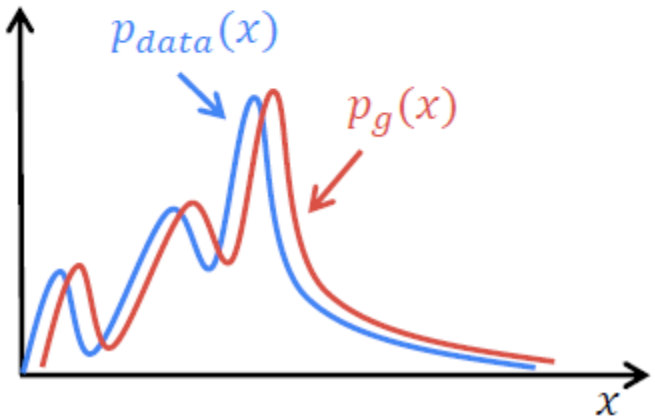
$$\min_{G,D} JSD(p_{data}||p_g)$$

Jenson-Shannon divergence

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

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

KL Divergence

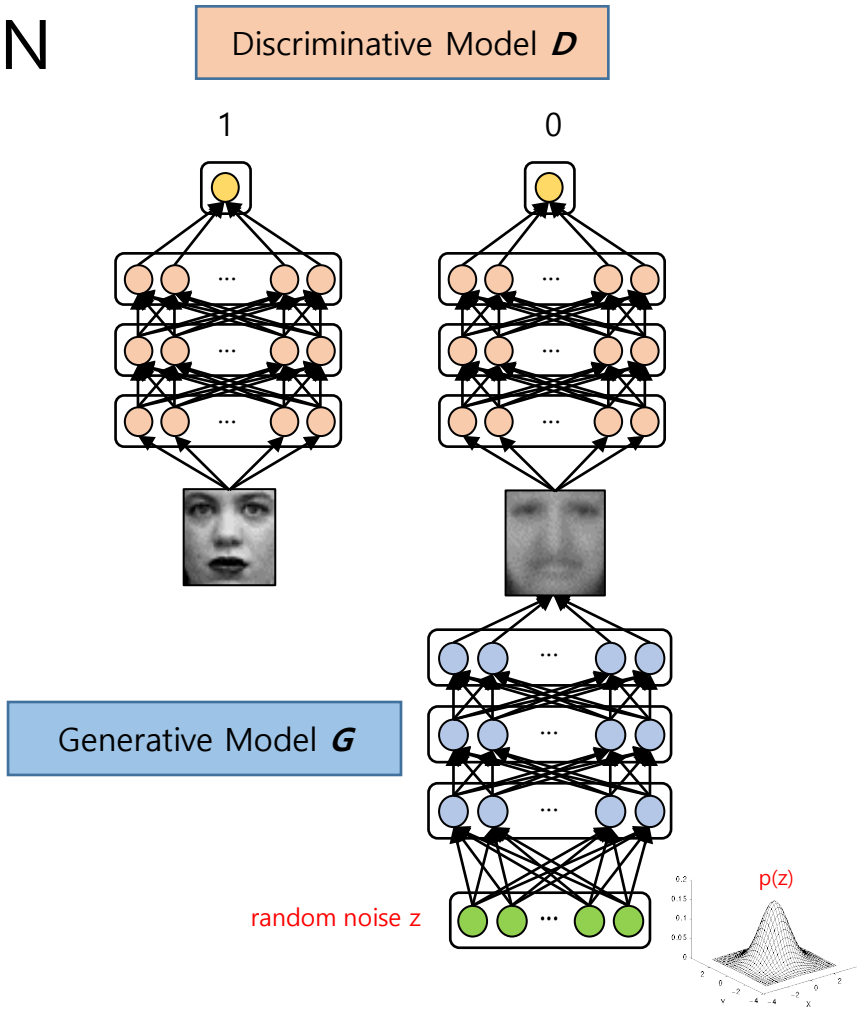


GAN Results

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

DCGAN

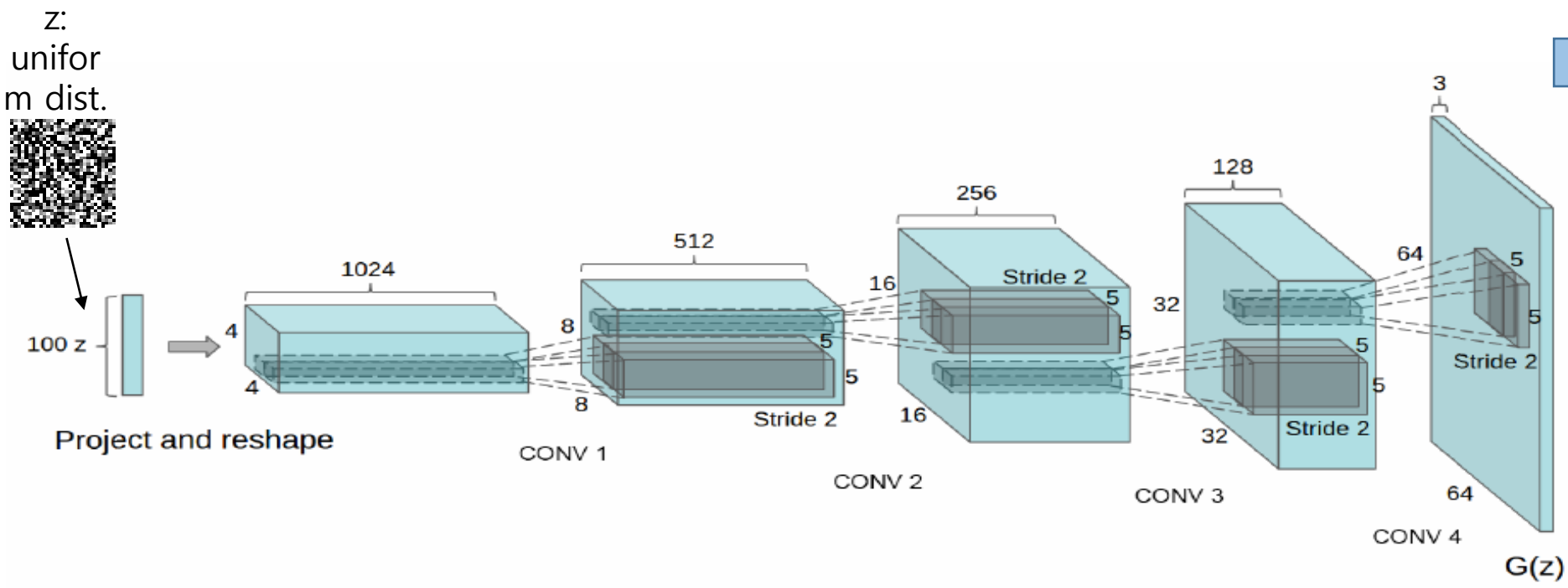
- Deep Convolutional Network + GAN
- Tricks for stable training
- Experimental Analysis

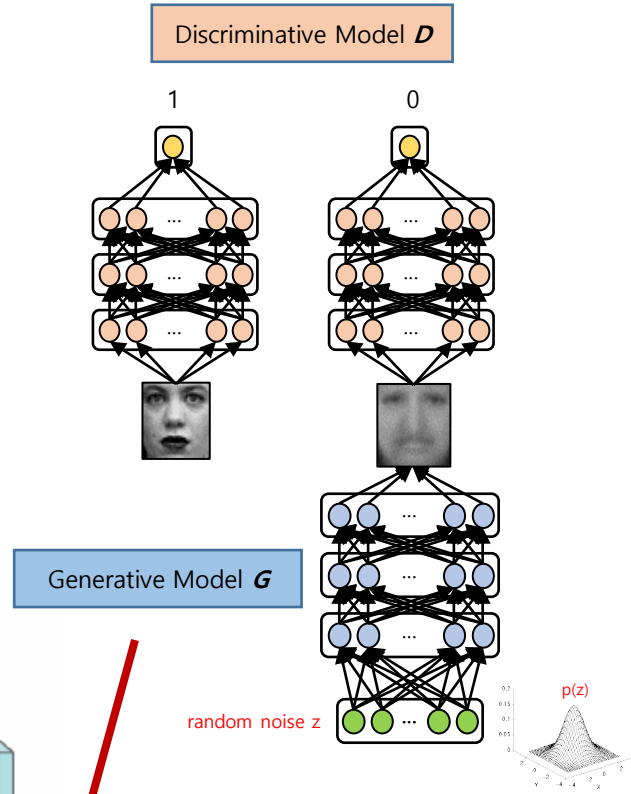


DCGAN

- Replace model's network to CNN
- Example of generator G

  (same as D)





Representation Learning

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

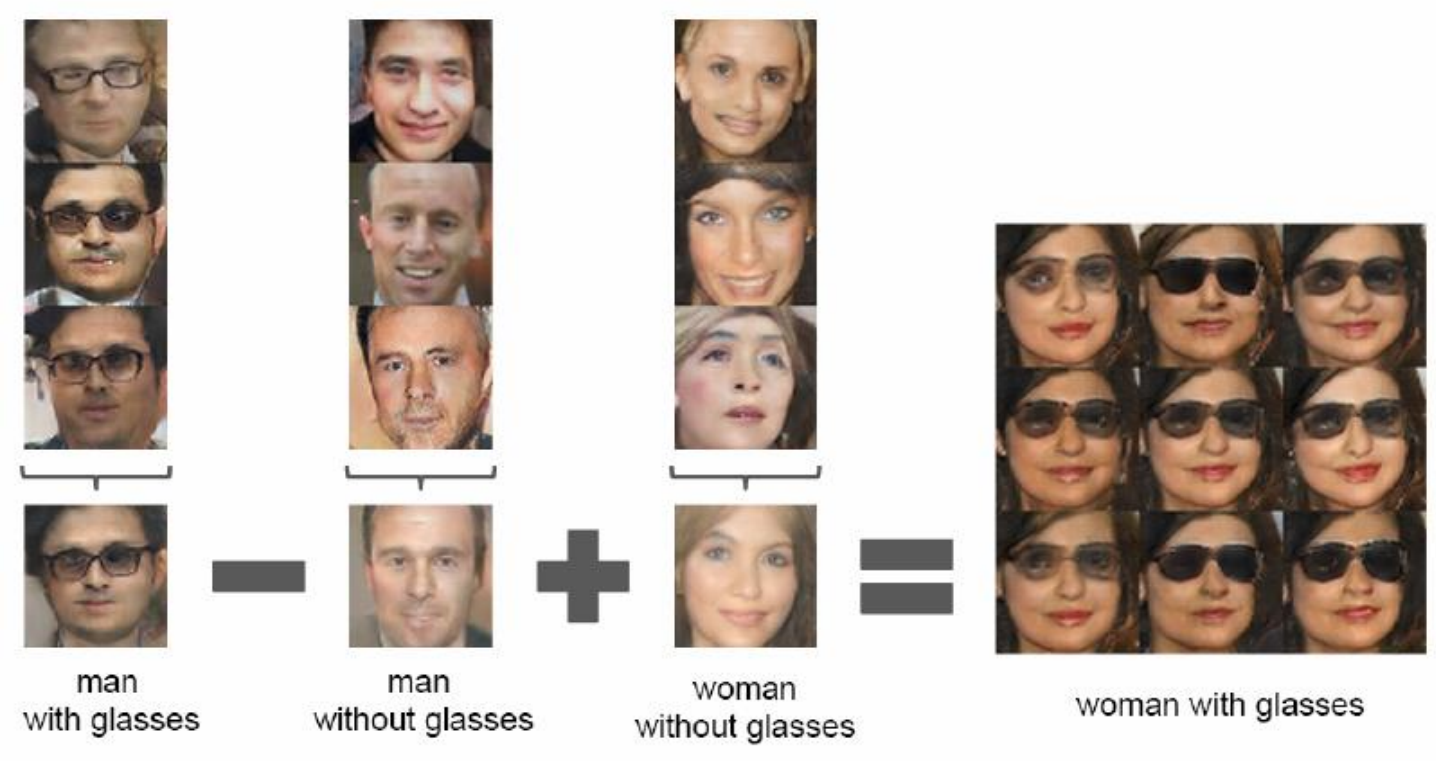
Guidelines for Stable DCGAN

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

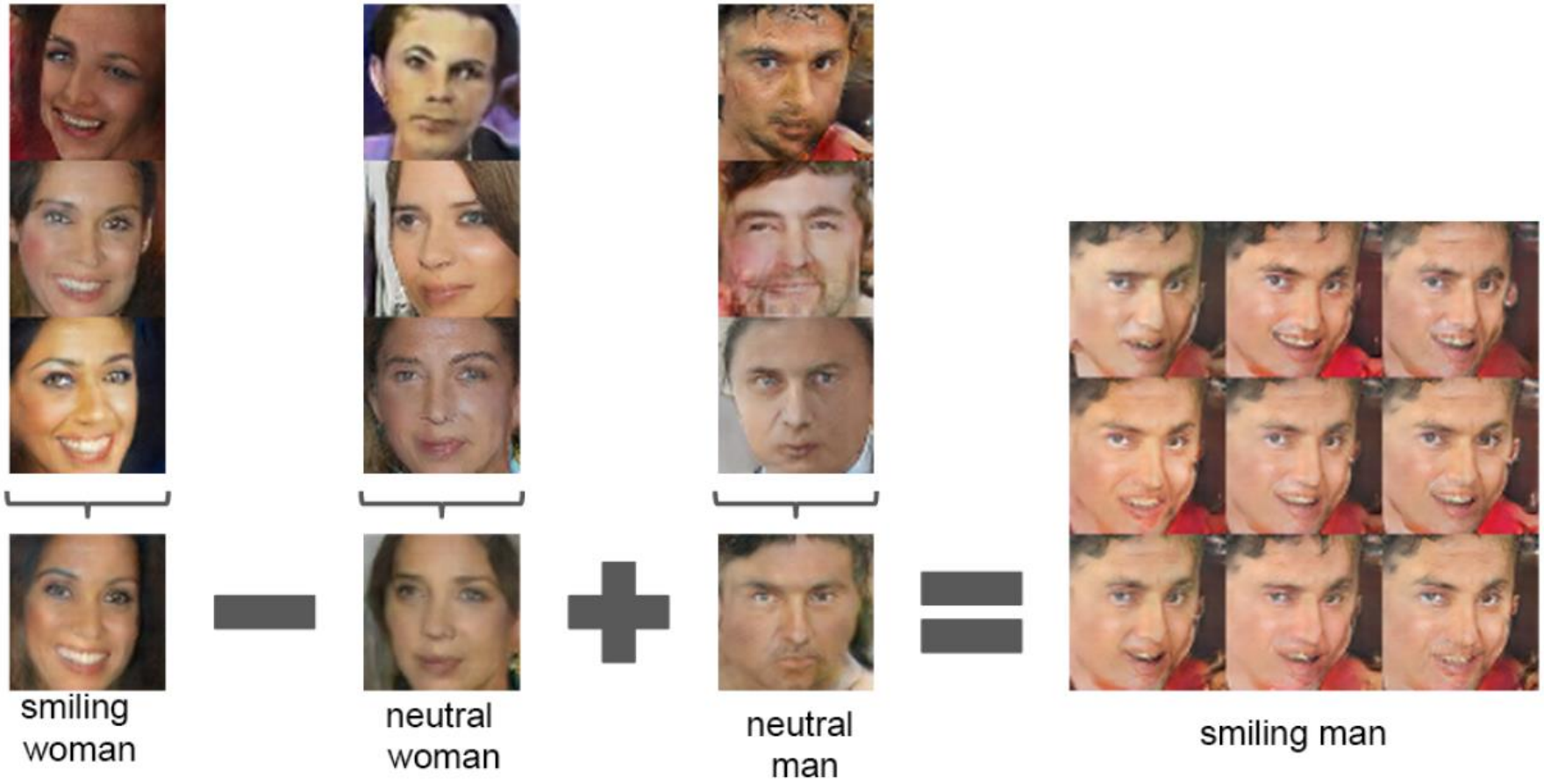
Experiments

Experiments

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

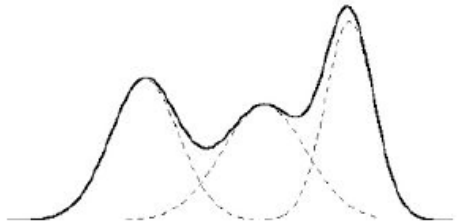


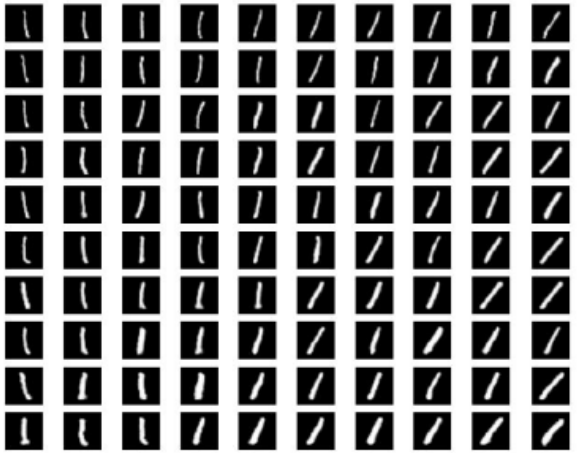
Experiments

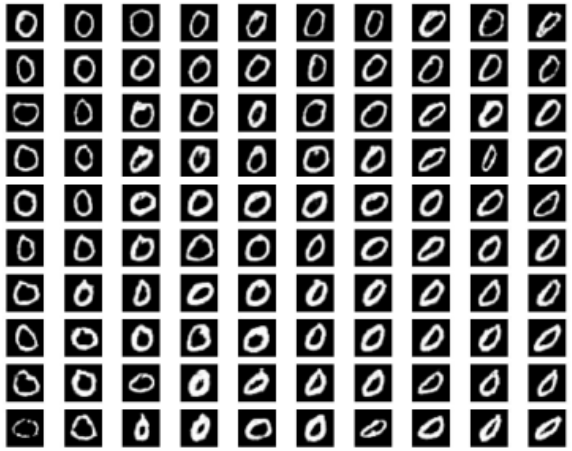


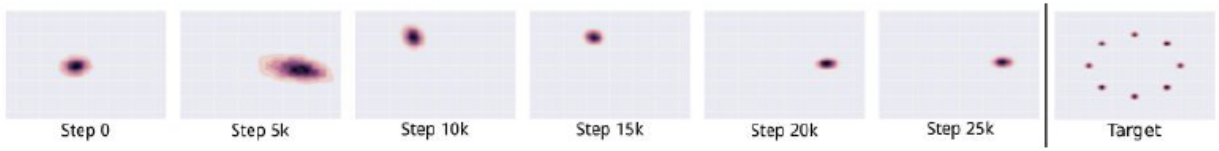
Experiments

Pitfall of GAN









mode collapsing

oscillating

Conditional GAN

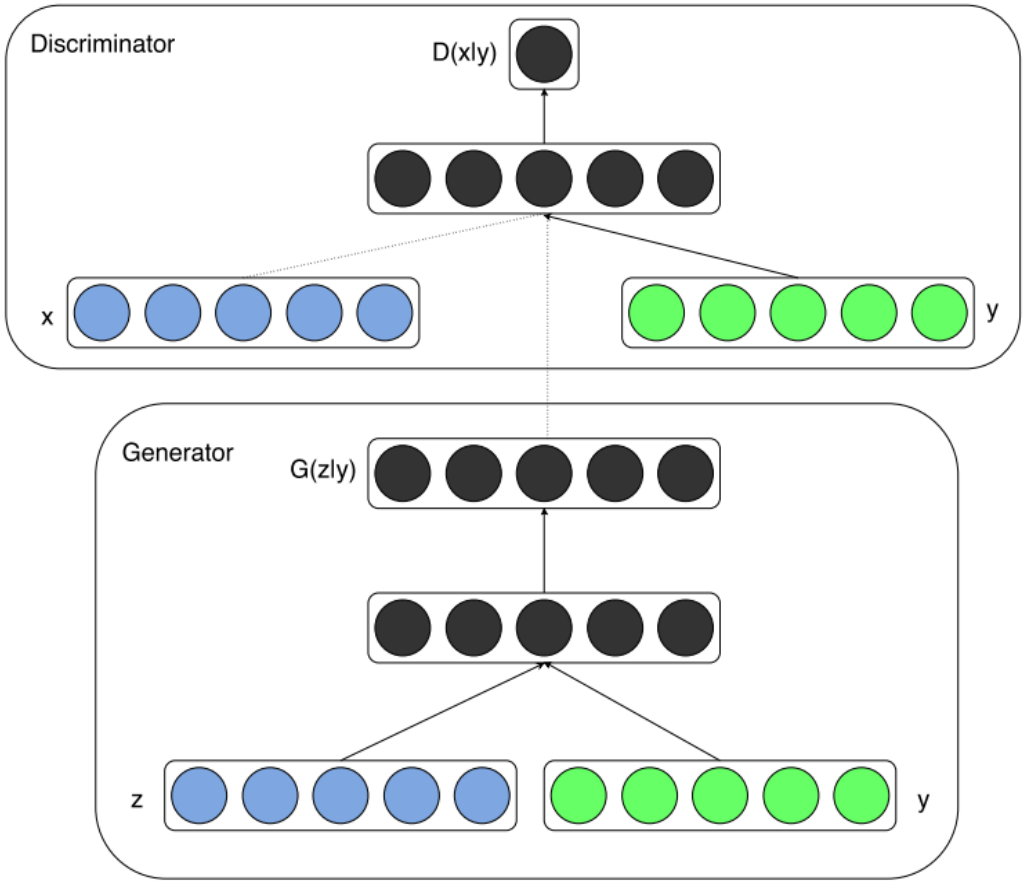


Figure 1: Conditional adversarial net

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

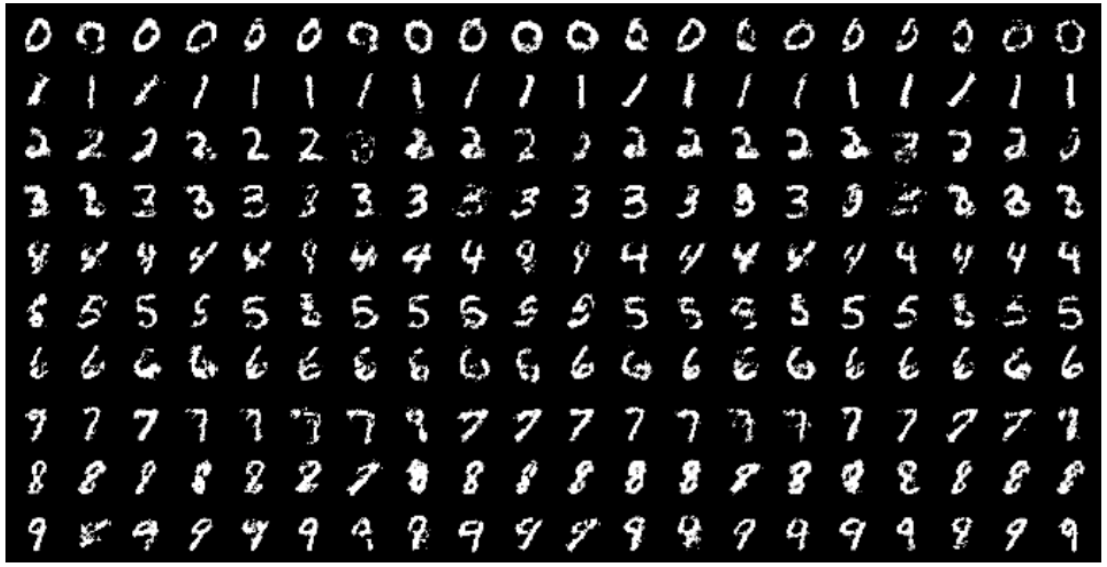
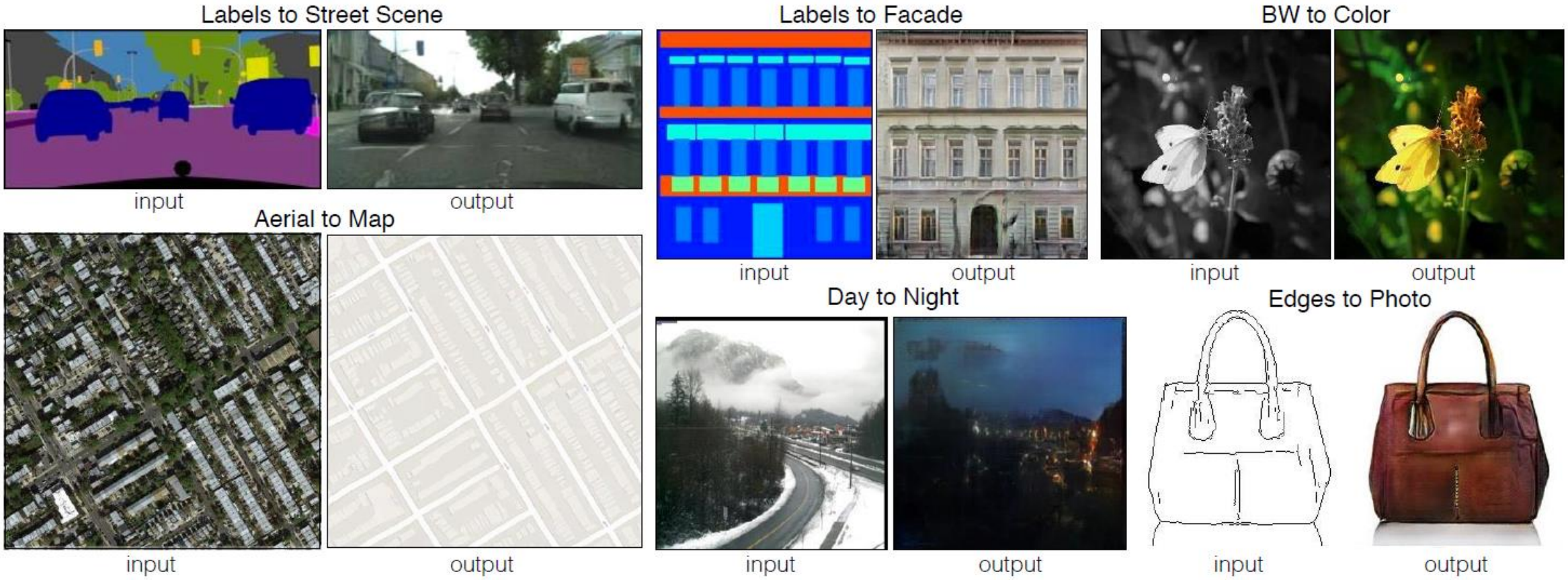


Figure 2: Generated MNIST digits, each row conditioned on one label

Image-to-Image Translation with Conditional Adversarial Networks

Pix2Pix


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

Pix2Pix

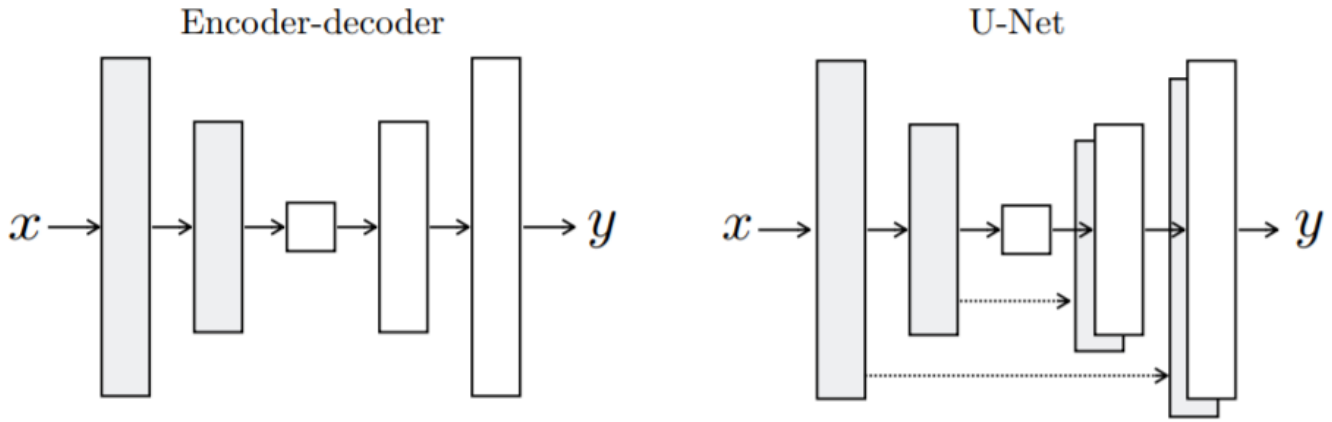


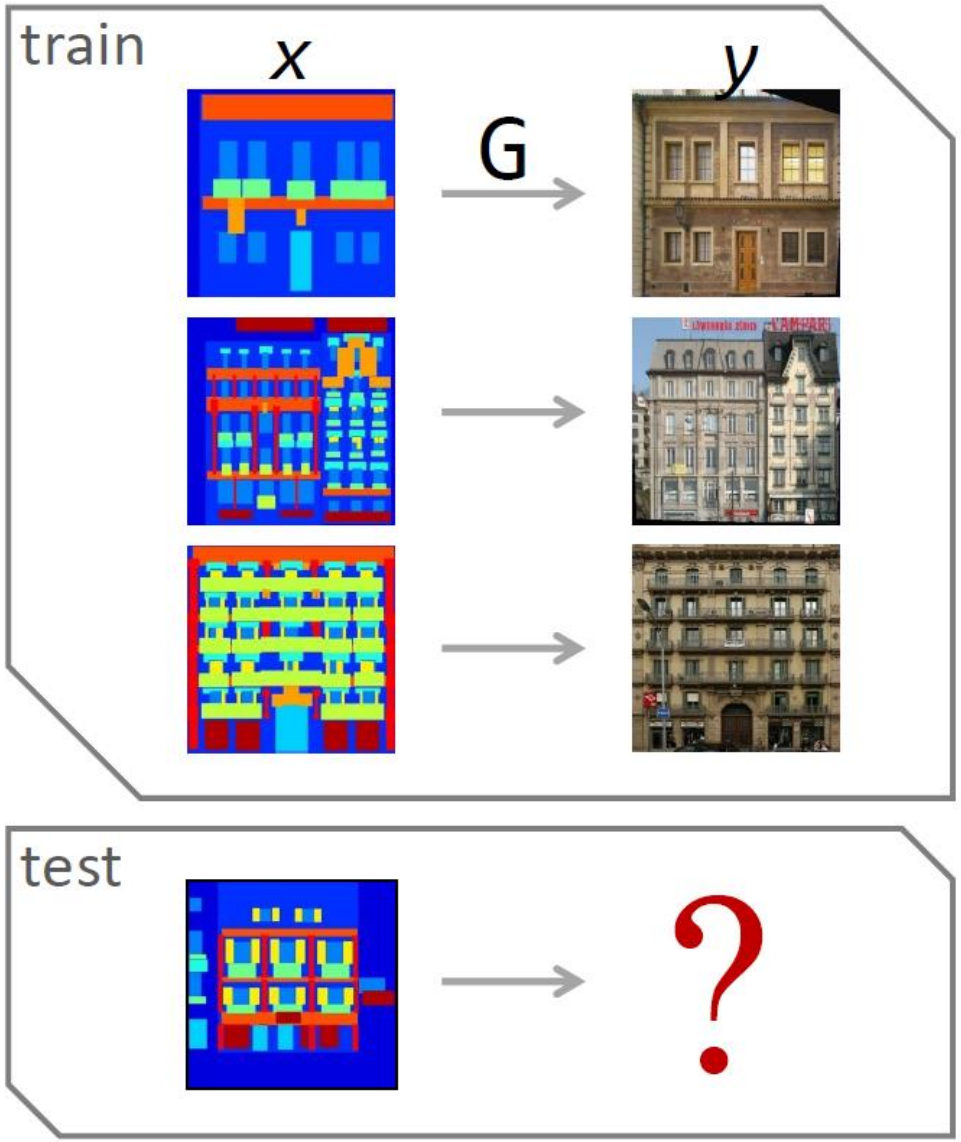
Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

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

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

+ PatchGAN **High-freq correctness**

Pix2Pix



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

Data from [Tylecek, 2013]

L1 Loss – Stable Guide Force

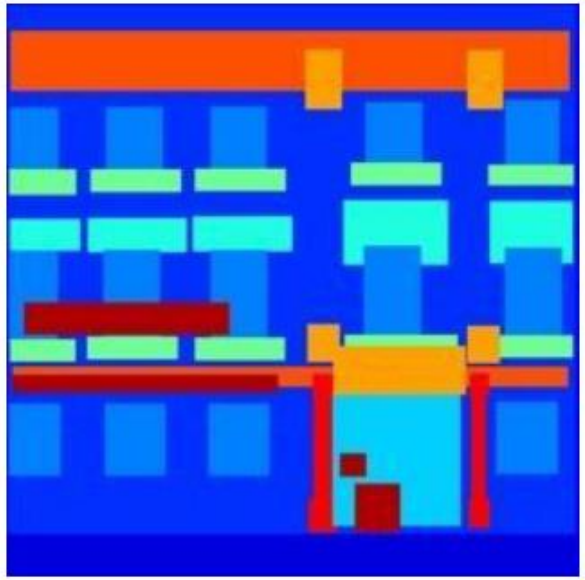
Loss: Minimize the difference between output *G(x) and the ground truth y*

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

Input

Output

Ground Truth



Input

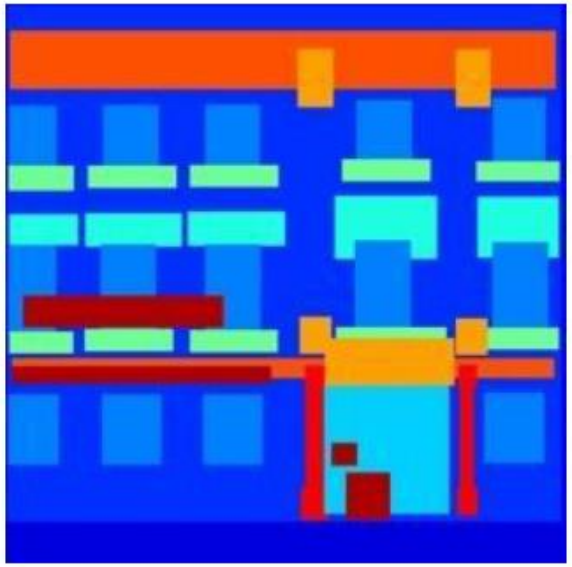
Output

Ground Truth

Adding GAN Loss

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

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



Input

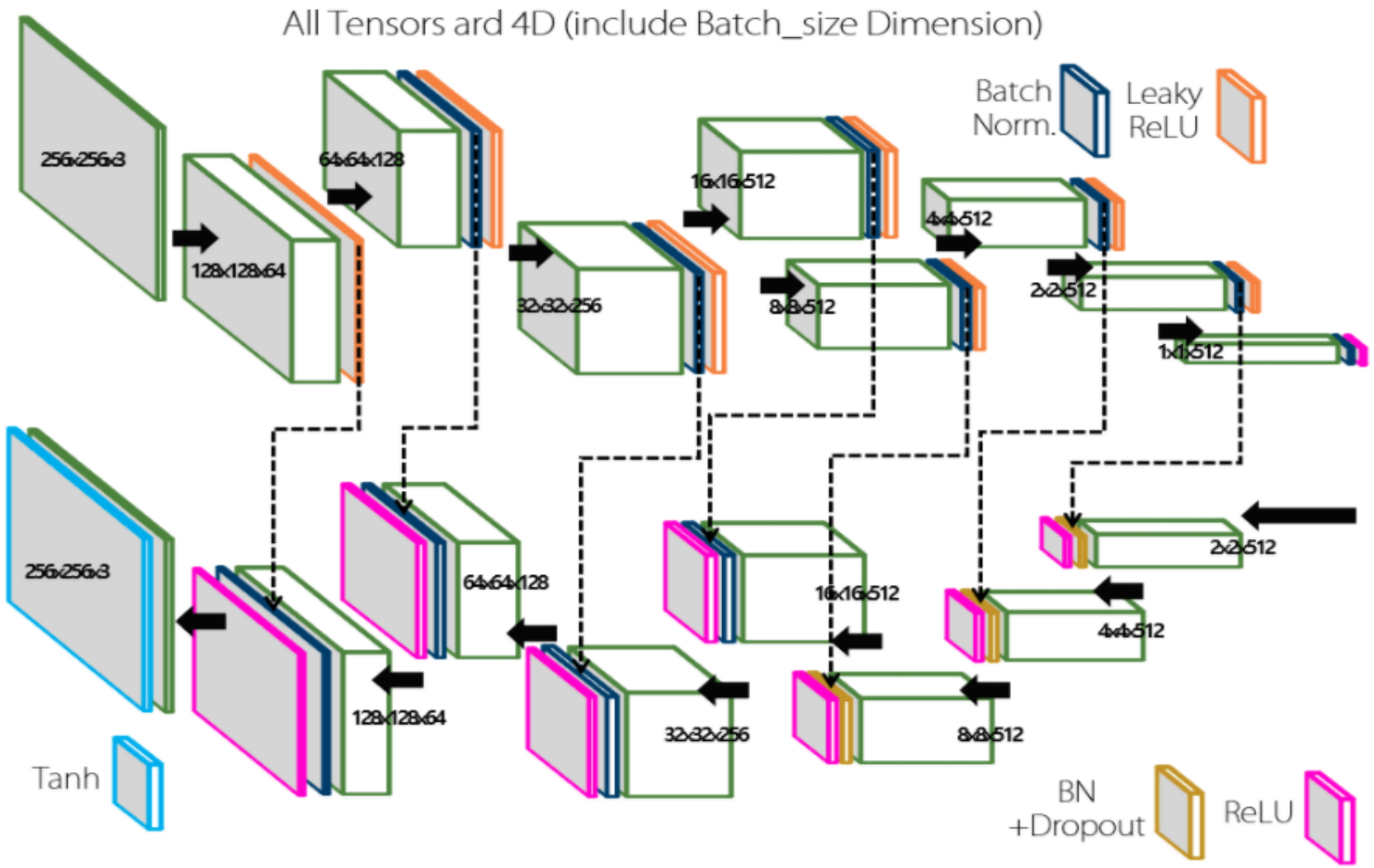
Ground Truth

L1 loss only

L1+GAN loss

Pix2Pix Generator

3.2 Generator Networks (network.py)



Pix2Pix Discriminator

3.3 Discriminator Networks (network.py)

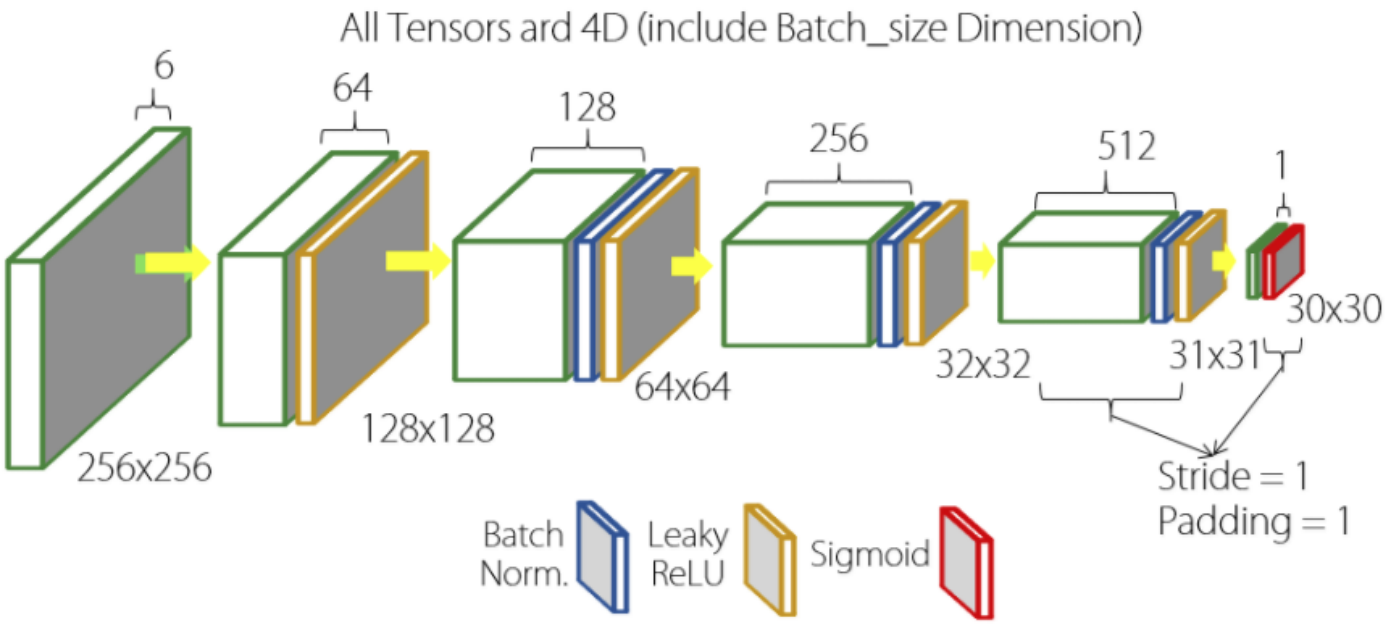


Image to Image Translation

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.

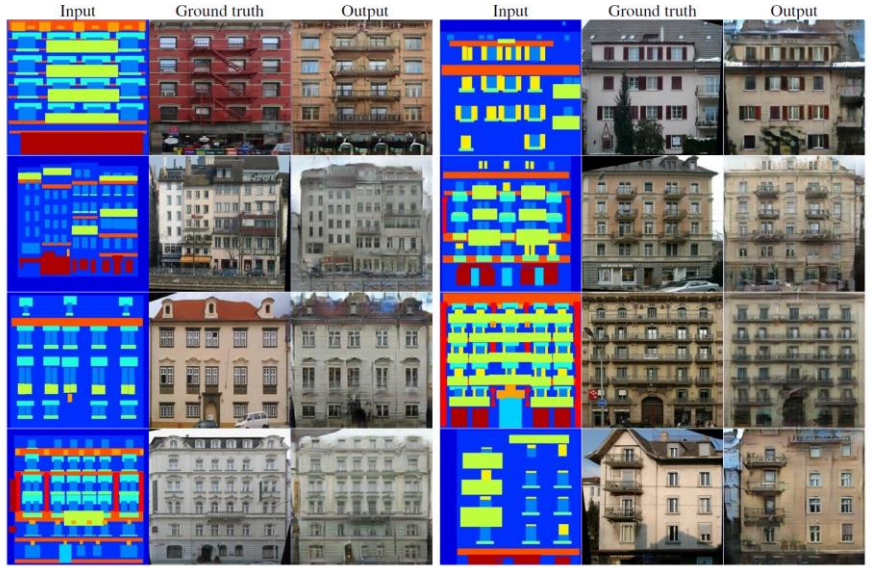


Figure 12: Example results of our method on facades labels→photo, compared to ground truth

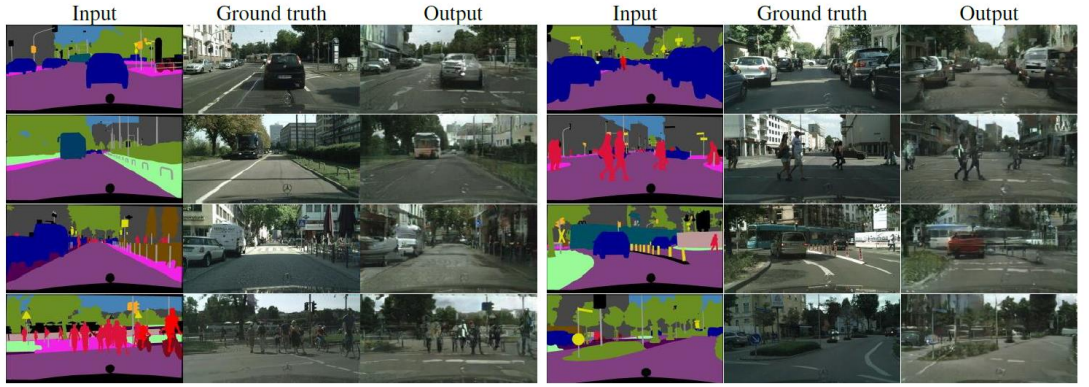


Figure 11: Example results of our method on Cityscapes labels→photo, compared to ground truth.

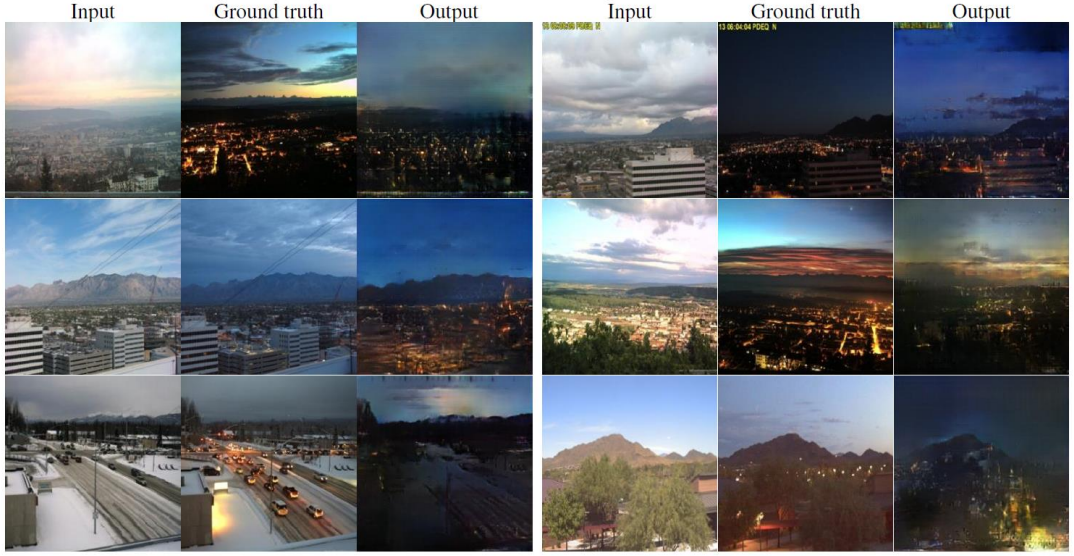


Figure 13: Example results of our method on day→night, compared to ground truth.

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

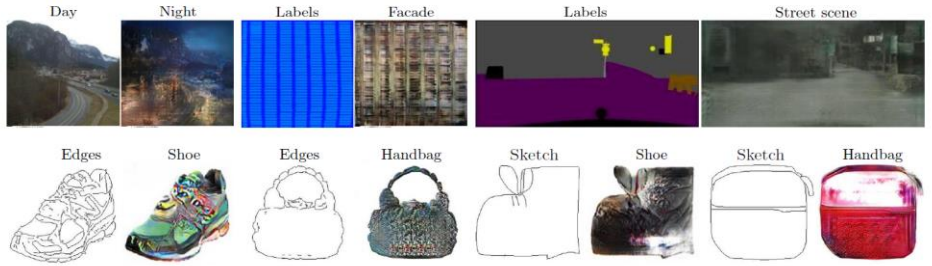


Figure 17: Example failure cases. Each pair of images shows input on the left and output on the right. These examples are selected as some of the worst results on our tasks. Common failures include artifacts in regions where the input image is sparse, and difficulty in handling unusual inputs. Please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

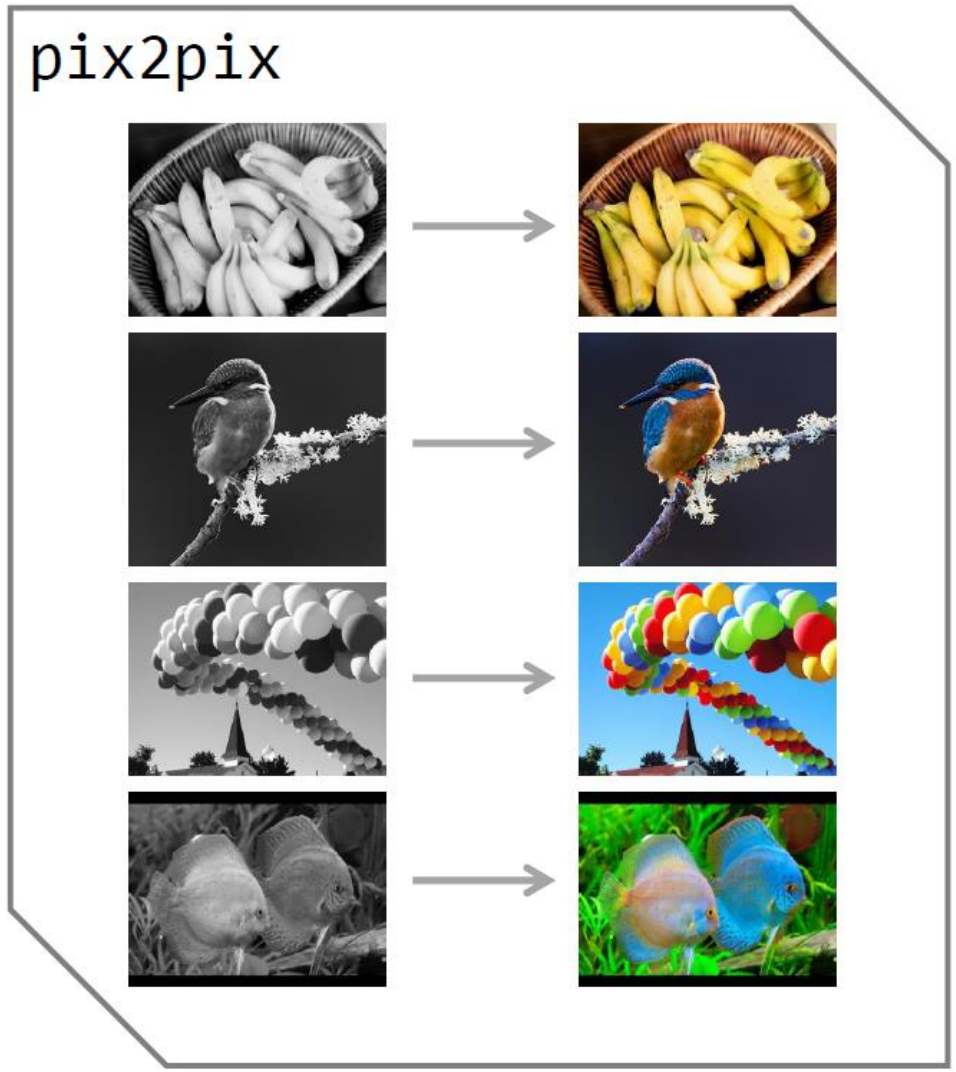
Let's Try

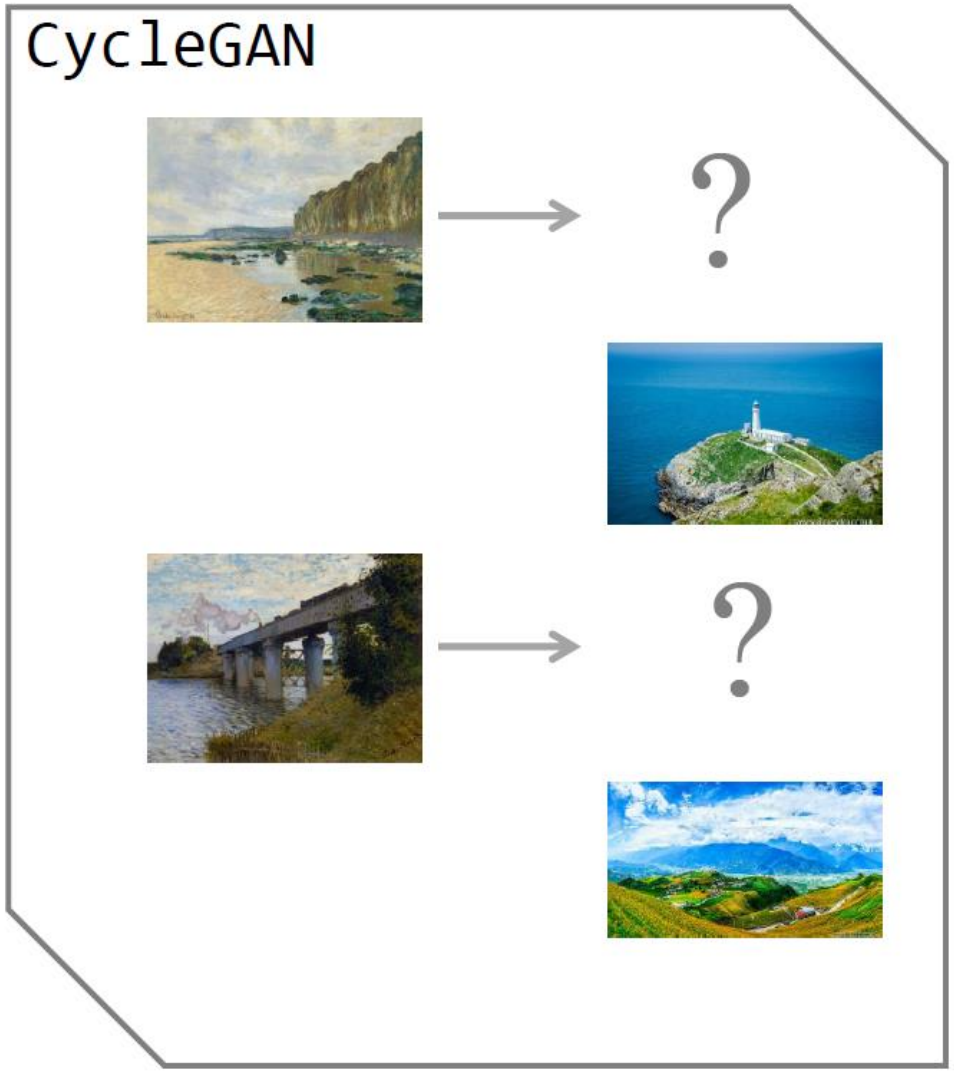
- https://affinelayer.com/pixsrv/

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

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

CycleGAN

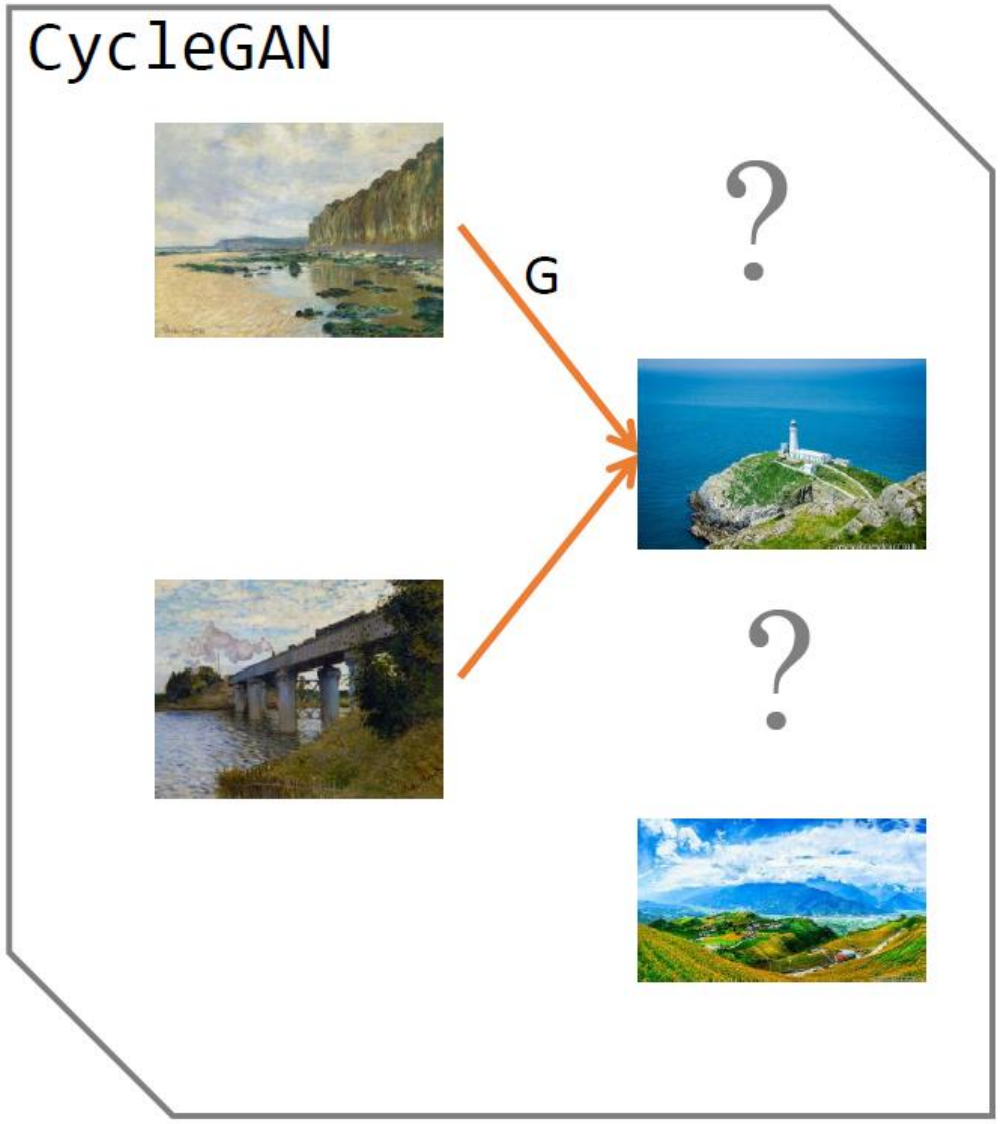




CycleGAN – Mode Collapsing

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

G(x) should just look photorealistic



CycleGAN

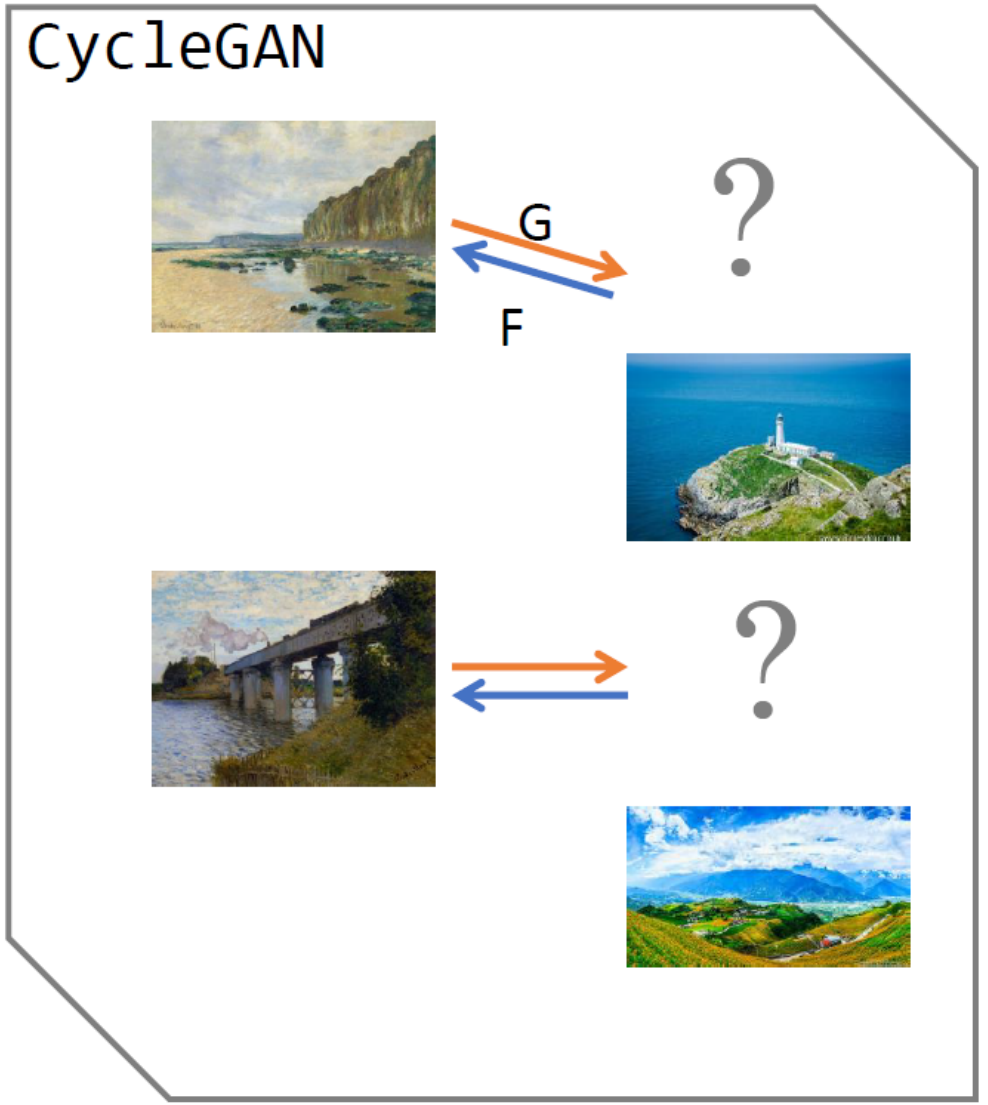
Loss

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

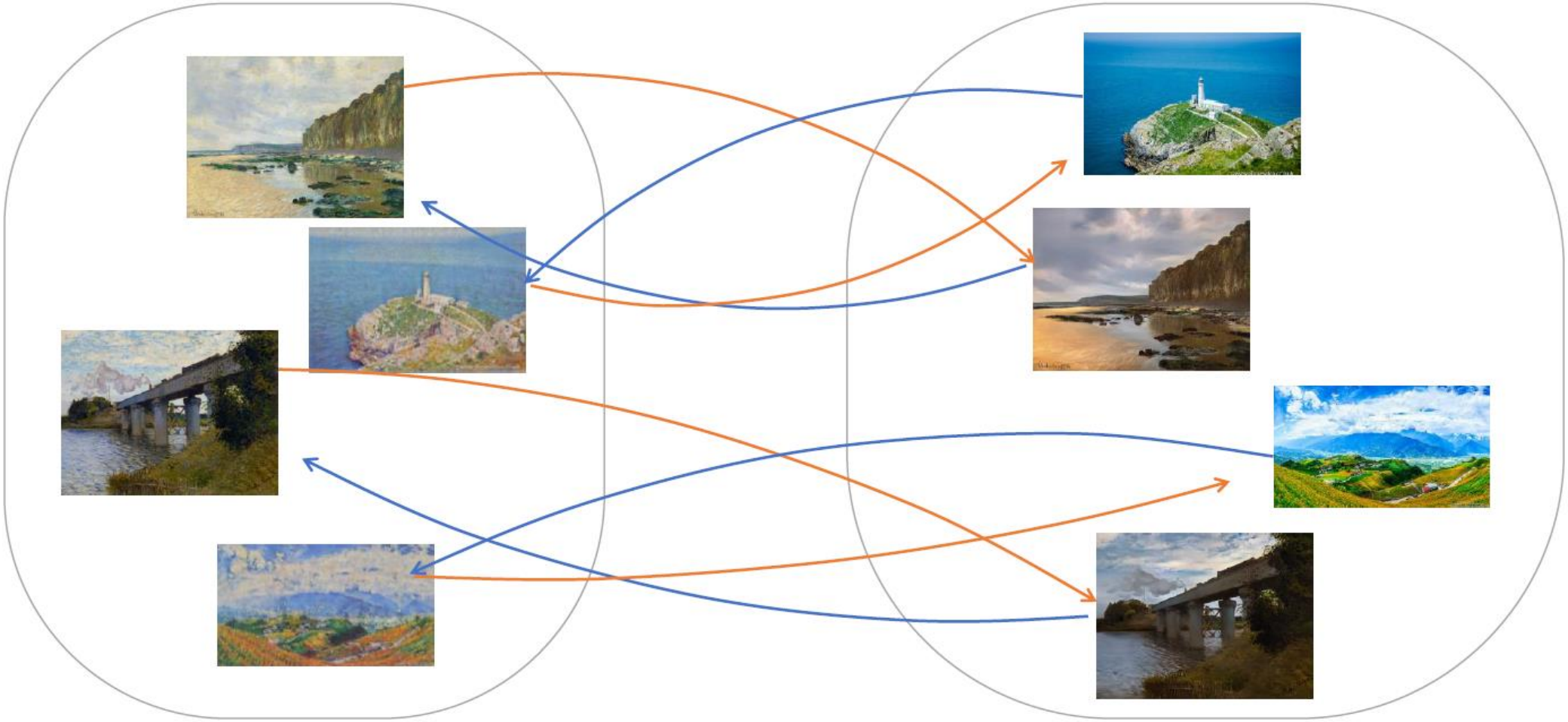
G(x) should just look photorealistic

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

where *F* is the inverse deep network

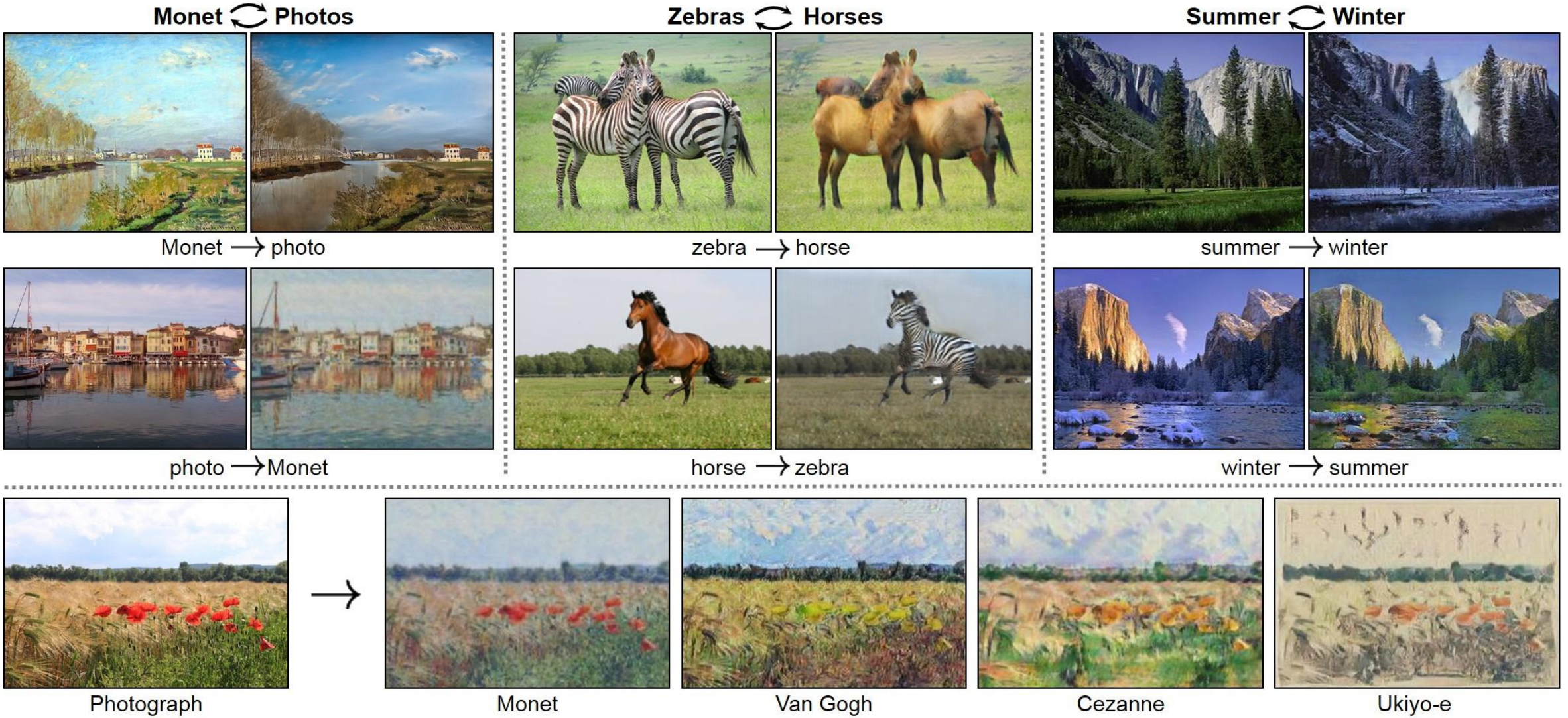


CycleGAN – Loss Function

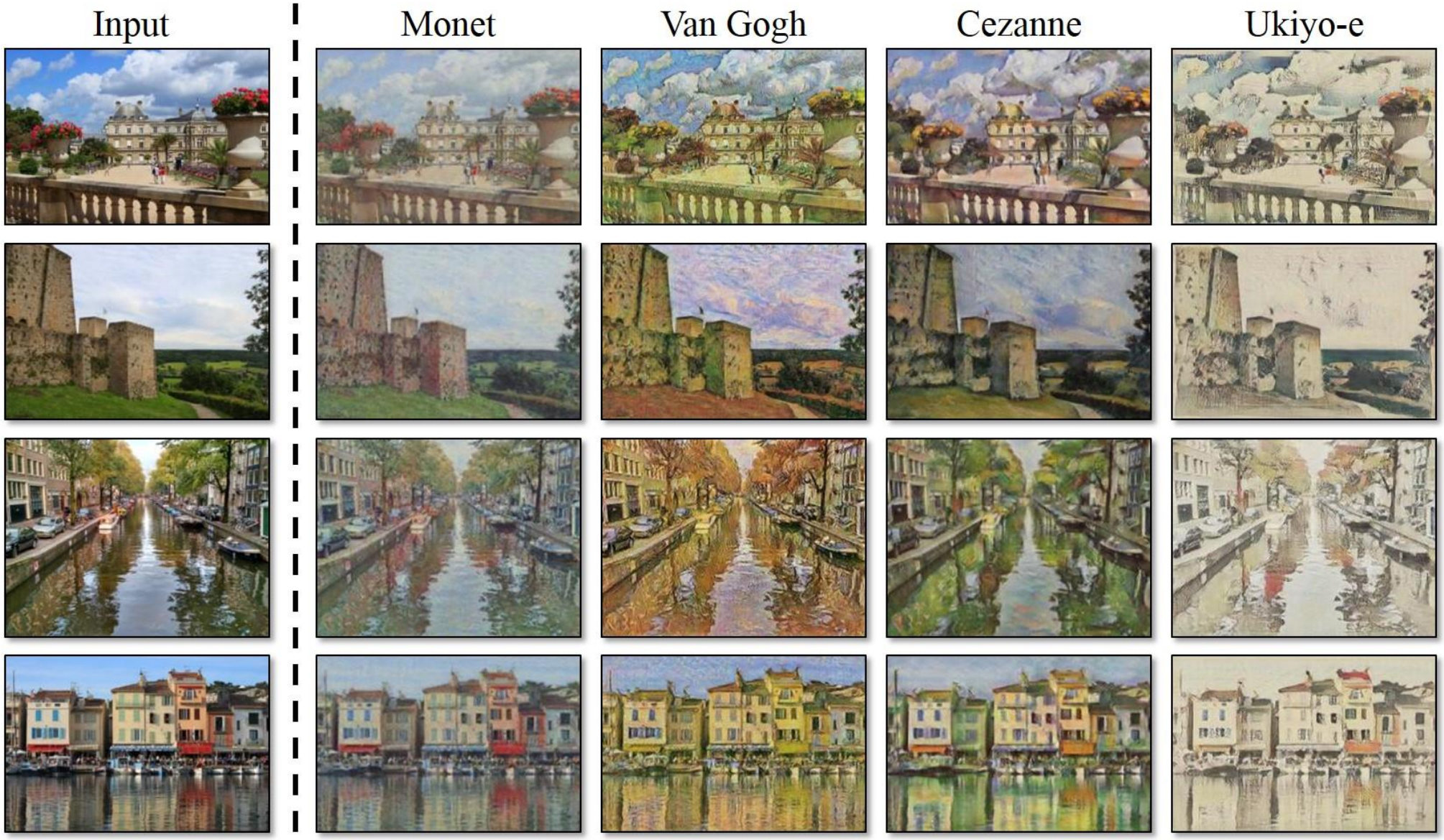


$$L_{GAN}(G(x), y) + \left\| F(G(x)) - x \right\|_1 \quad + \quad L_{GAN}(F(y), x) + \left\| G(F(y)) - y \right\|_1$$

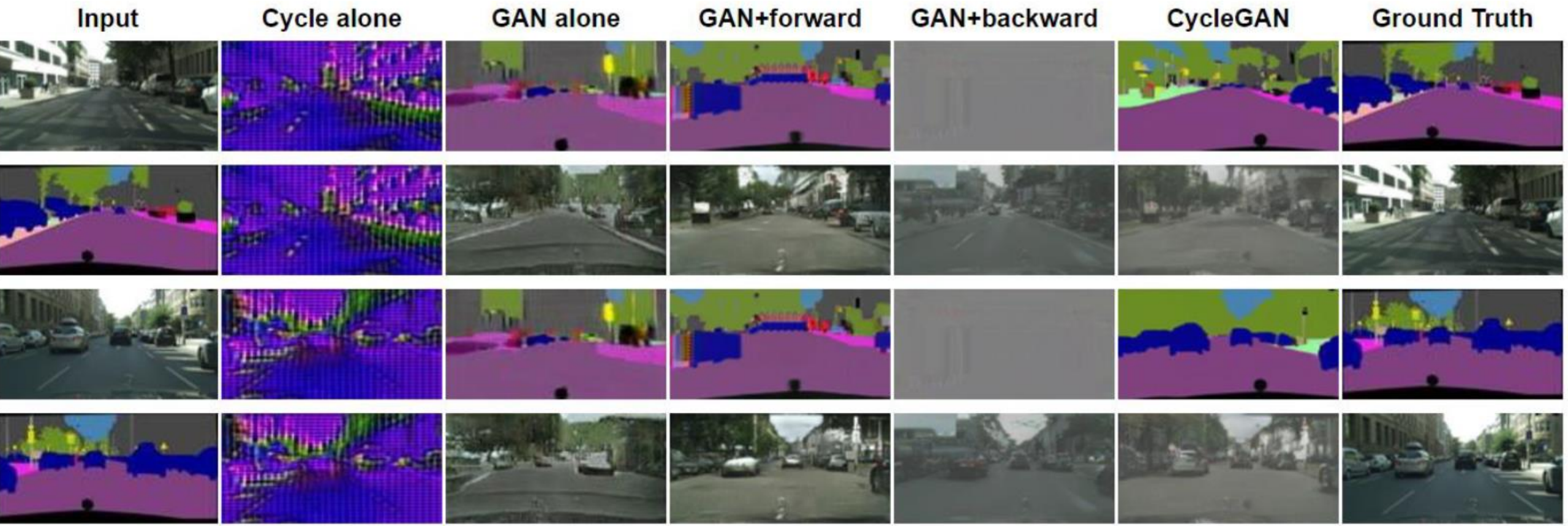
Results


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

Results

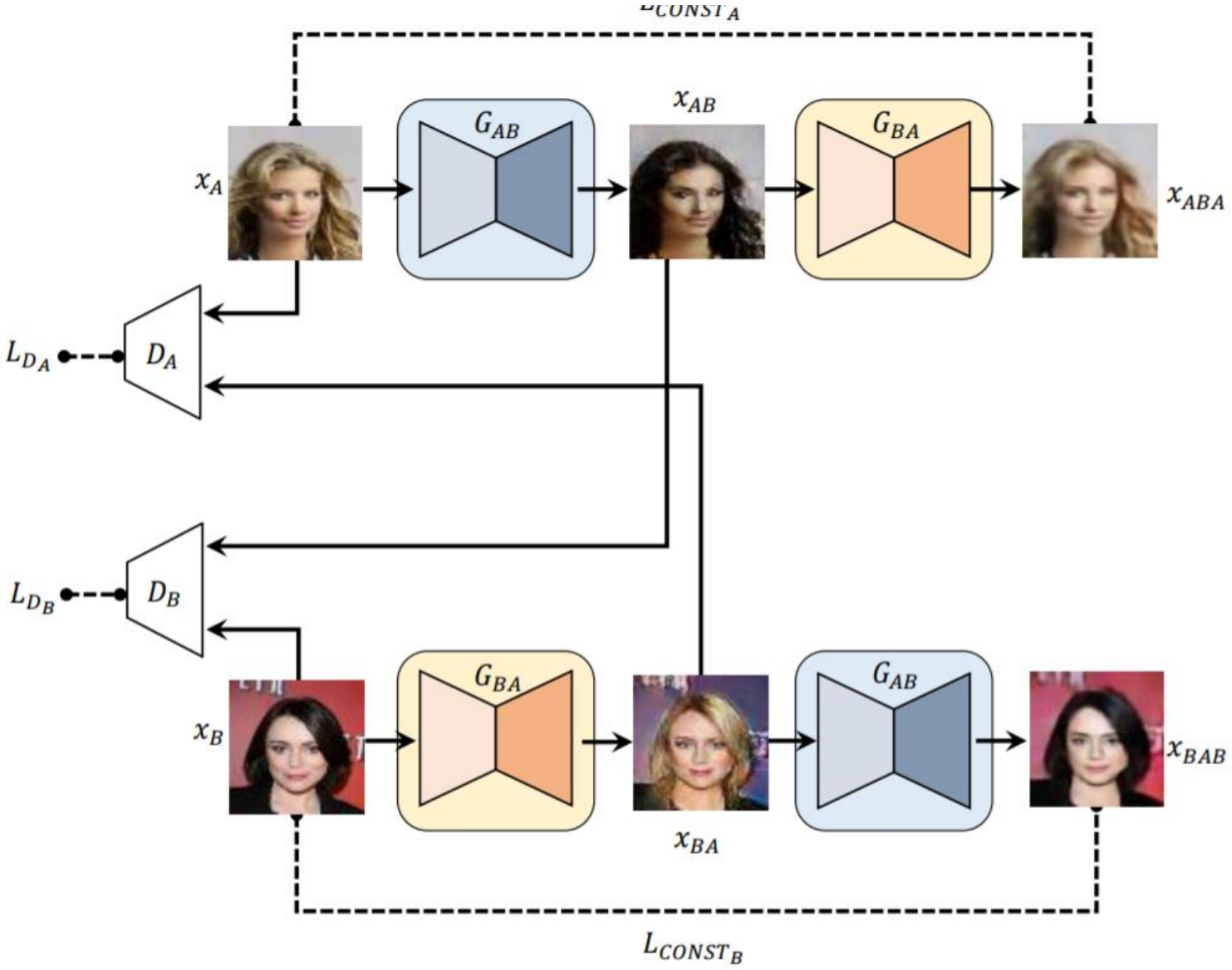


Results


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

Reconstructed Images

DiscoGAN


L_{CONST_A}
x_{AB}
G_{AB}
G_{BA}
x_A
x_{ABA}
L_{D_A}
D_A
L_{D_B}
D_B
G_{BA}
G_{AB}
x_B
x_{BA}
x_{BAB}
L_{CONST_B}

Different Result

CycleGAN

Cat -> Dog

Dog -> Cat

DiscoGAN

Cat -> Dog

Dog -> Cat