生成对抗网络

学生创新中心 2025 年 3 月

目录 / CONTENTS

01

Vanilla GAN

GAN

- 零和博弈
  - 参与博弈的各方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈各方的收益和损失相加总和永远为“零”，双方不存在合作的可能
  - 举例：古董字画市场

造假者

目标：制作以假乱真的赝品

策略：不断学习真品的特征，改进造假技术

鉴定专家

目标：识别赝品，维护市场秩序

策略：不断学习真品和赝品的区别，提高鉴别能力

GAN

- 零和博弈

Round 1 -> Round 2 -> … -> Round N

造假者

技术粗糙的赝品

收益：-1

改进造假技术，
制作更逼真的赝品

完美无缺的赝品

收益：1

在**博弈**中不断进化

鉴定专家

能够识别大多数赝品

收益：1

不断学习新的赝品，
提高鉴别能力

难以分辨完美的赝品

收益：-1

GAN

- 生成式对抗网络（Generative Adversarial Networks）

造假者 → 生成器（Generator）

目标：生成以假乱真的数据

鉴定专家 → 判别器（Discriminator）

目标：区分真实数据和生成数据

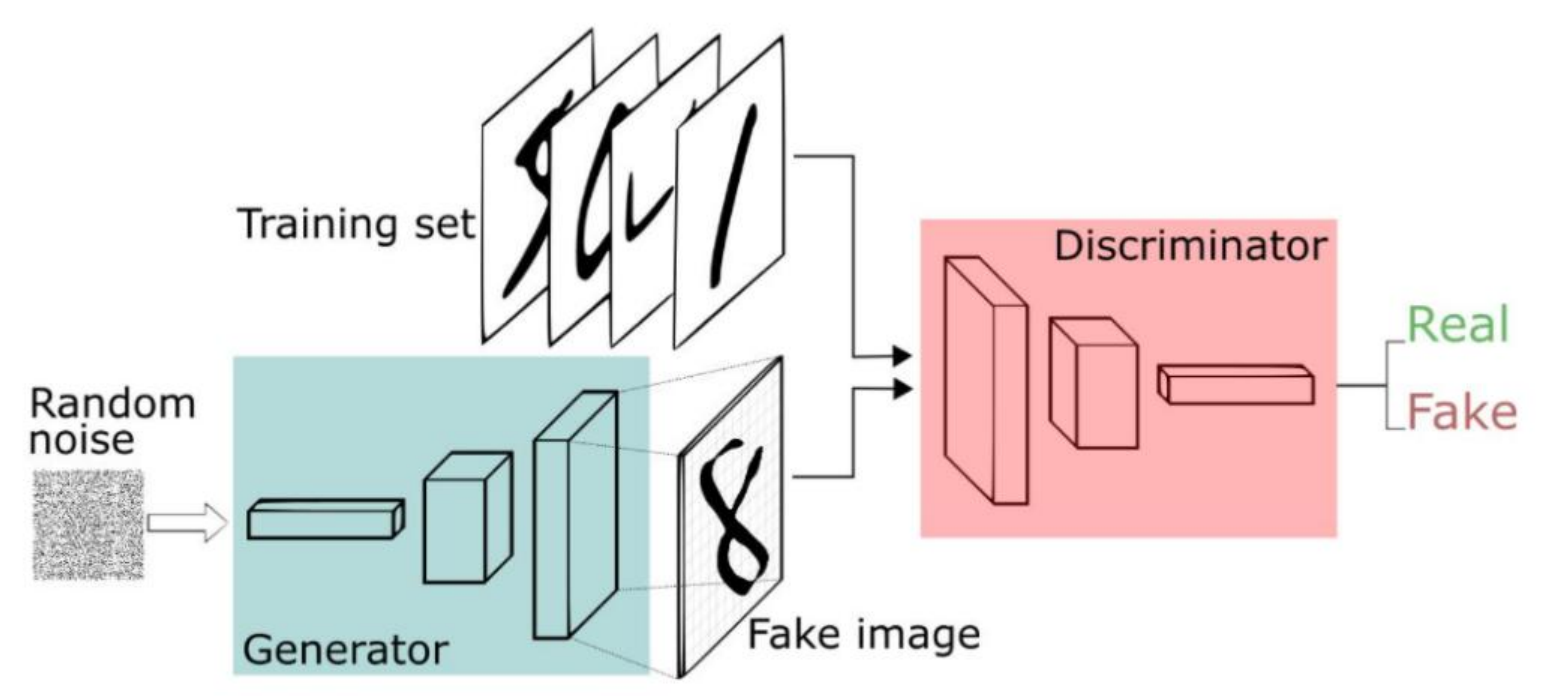
博弈过程

对抗学习过程（Adversarial）

生成器和判别器在对抗中不断学习，最终达到纳什均衡

GAN

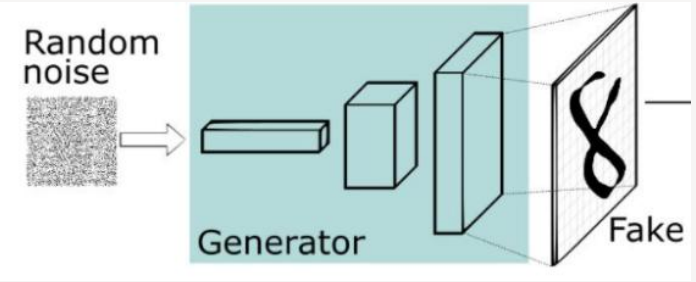
- 生成式对抗网络（Generative Adversarial Networks）
  - 由一个生成器和一个判别器构成，通过对抗学习的方式来训练
  - 估测数据样本的潜在分布并生成新的数据样本



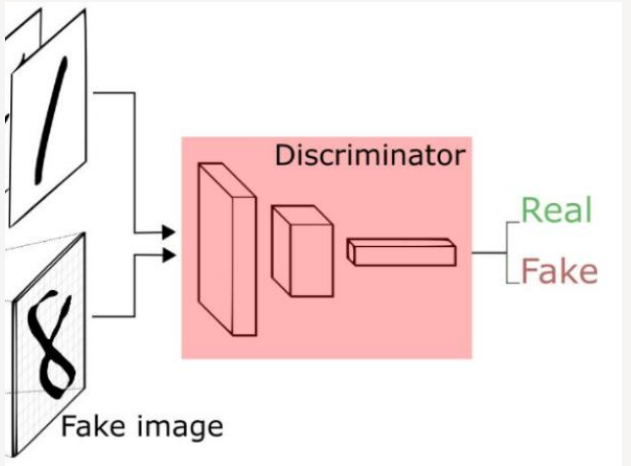
Goodfellow I, et al. Generative adversarial nets. NeurIPS 2014.

GAN

- 生成器（Generator）
  - 学习如何将随机噪声合成和真实数据分布类似的样本
  - 对于输出样本的具体信息不做要求，只要求其能够与真实数据尽可能相似
  - 随机噪声：均值分布、高斯分布

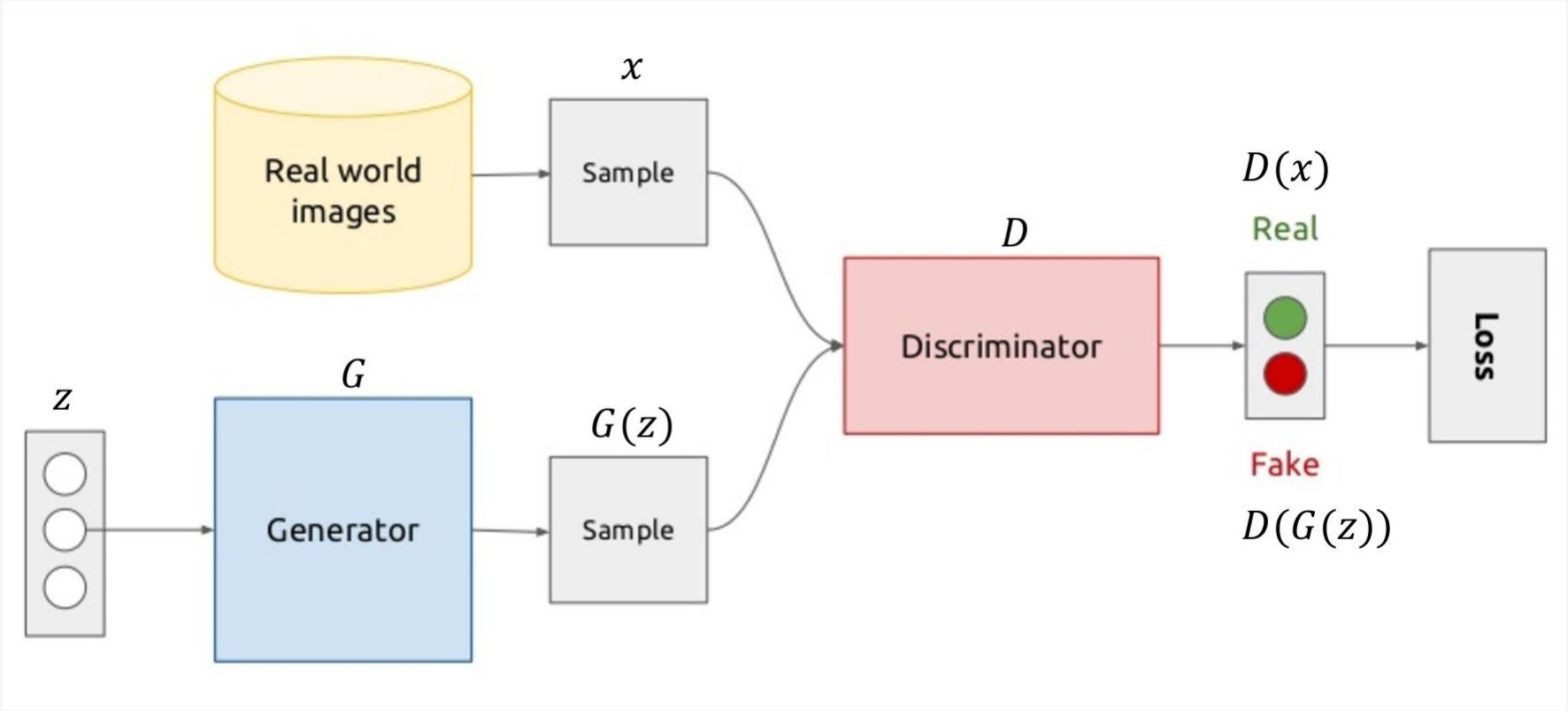


- 判别器（Discriminator）
  - 学习如何区分生成器输出的合成样本和真实样本
  - 为生成器的训练提供监督



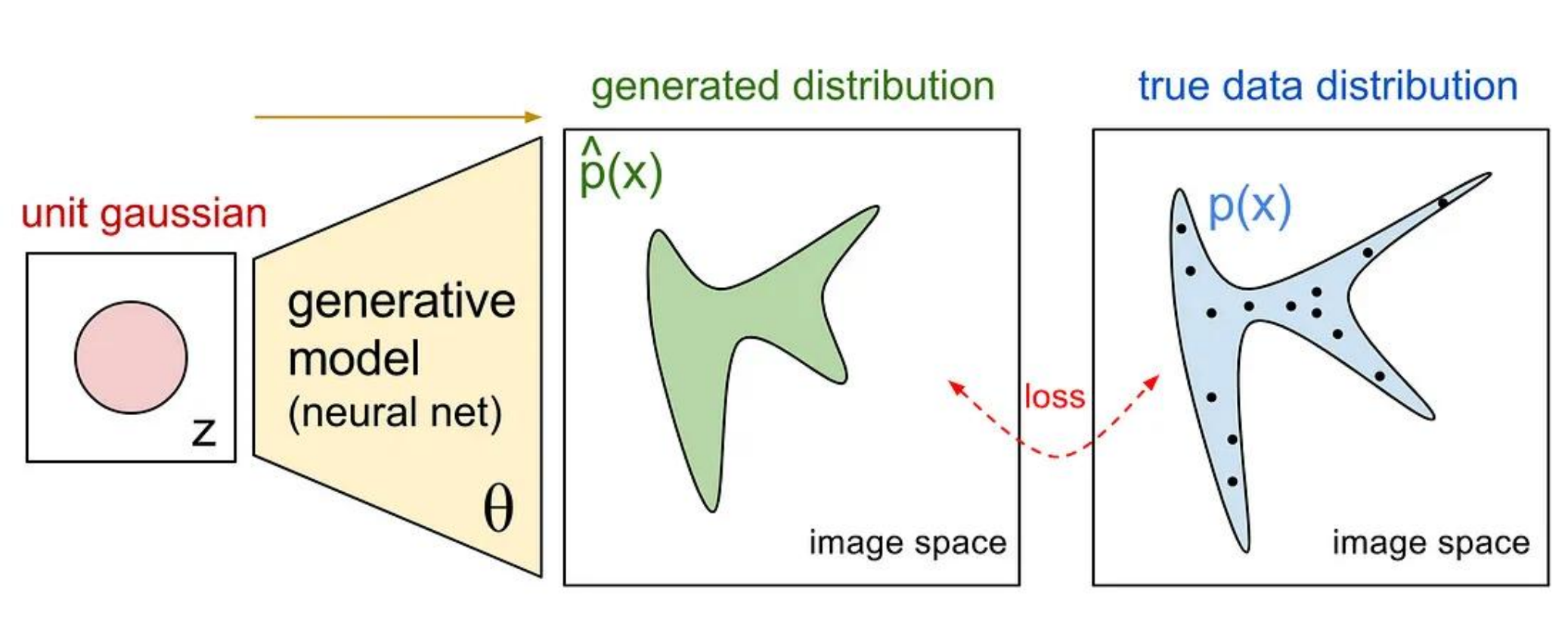
GAN

- 整体结构
  - z：隐变量（latent code），x：真实图像样本



GAN

- 如何评估?


generated distribution
true data distribution
unit gaussian
z
generative
model
(neural net)
θ
p̂(x)
image space
loss
p(x)
image space

GAN

- 目标函数
  - $\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$
  - $D(\boldsymbol{x})$：将x分类为真实数据的概率
  - $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]$：把真实样本分类成真的的概率
  - $\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$：把合成样本分类为假的的概率
  - 极小极大策略：生成器总是不断的使判别器的最大收益最小化
  - 理想平衡点：$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$

GAN

- 鉴别器训练阶段
  - $\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$
  - ”把真样本分类成真的，把假样本分类成假的“ -> 鉴别器的能力
  - 需要最大化该函数

- 生成器训练阶段
  - $\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$

常数

  - “把假样本分类成假的“ -> 与生成器的能力互斥
  - 需要最小化该函数

GAN

- 训练过程
  - 交替训练：先对 D 训练 k 步，然后再训练一步 G
  - 伪代码

```
for epoch i in [1,2,...,n] do:
    for step j in [1,2,...,k] do:
        Sample m noise samples from P(z) and m real samples from P(x)
        Update parameters of D by SGD
    end
    Sample m noise samples from P(z)
    Update parameters of G by SGD
end
```

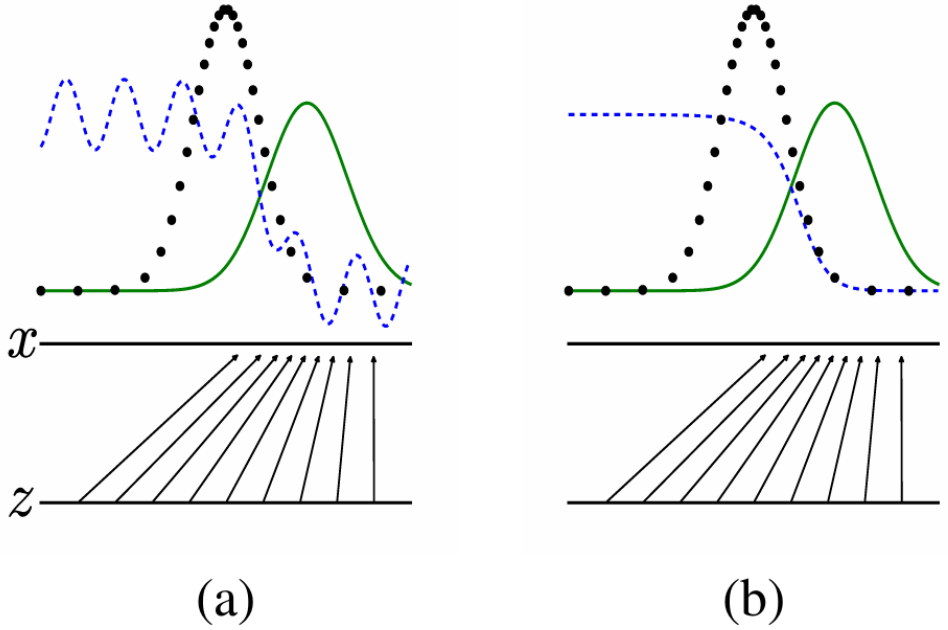
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

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

GAN

- 训练过程

黑色虚线：真实数据分布 P(x)，绿色实线：生成函数 G(z)，蓝色虚线：判别函数 D(x)



(a) 判别器与生成器均未训练呈随机分布

(b) 判别器经过训练，输出的分布在靠近训练集“真”数据分布的区间趋近于1（真），在靠近生成器生成的“假”数据分布的区间趋近于0（假）

(c)

(d)

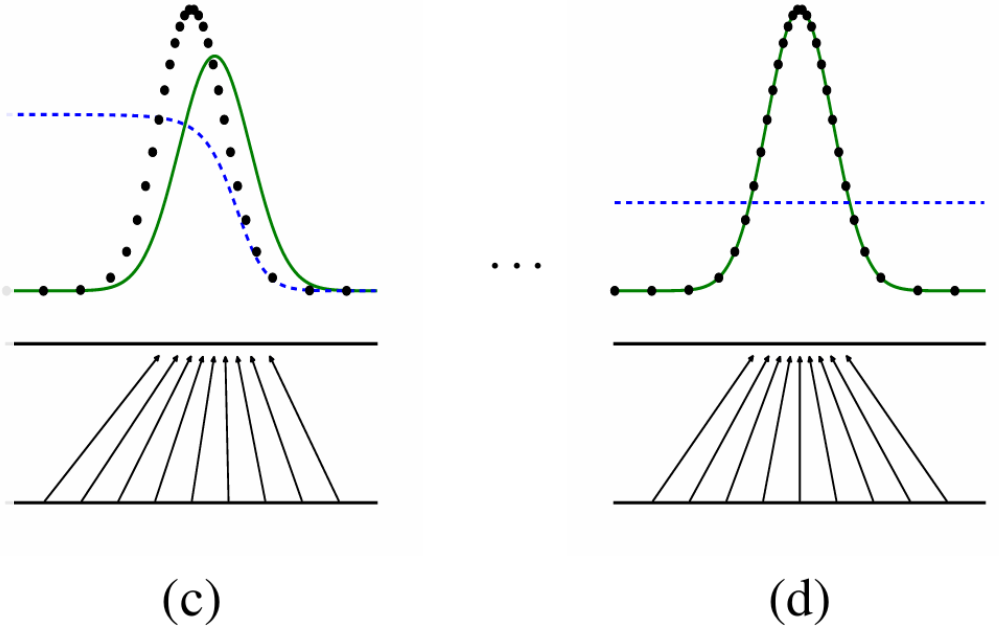
GAN

- 训练过程

黑色虚线：真实数据分布 P(x)，绿色实线：生成函数 G(z)，蓝色虚线：判别函数 D(x)

(c) 生成器根据判别器输出的分布，更新参数，使其输出趋近于训练集“真”数据的分布。而判别器的输出分布逐渐趋于平缓

(d) 训练完成时，生成器输出的分布完美拟合了训练集数据的分布，判别器的输出无法判别生成器输出的真伪而呈一条取值约为0.5（真假之间）的直线



GAN

- 代码实现
  - 生成器和判别器均为多层感知器（MLP）

```
# 全连接生成器结构
def generator(z, hidden_dim, output_dim):
    # 输入维度为 latent_dim
    x = dense(z, hidden_dim)
    x = relu(x)
    # 第二层全连接，激活函数采用 tanh
    x = dense(x, output_dim, activation='tanh')
    return x
```

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

```
# 二元交叉熵损失函数
def binary_cross_entropy(y_true, y_pred):
    return F.binary_cross_entropy(y_true, y_pred)
```

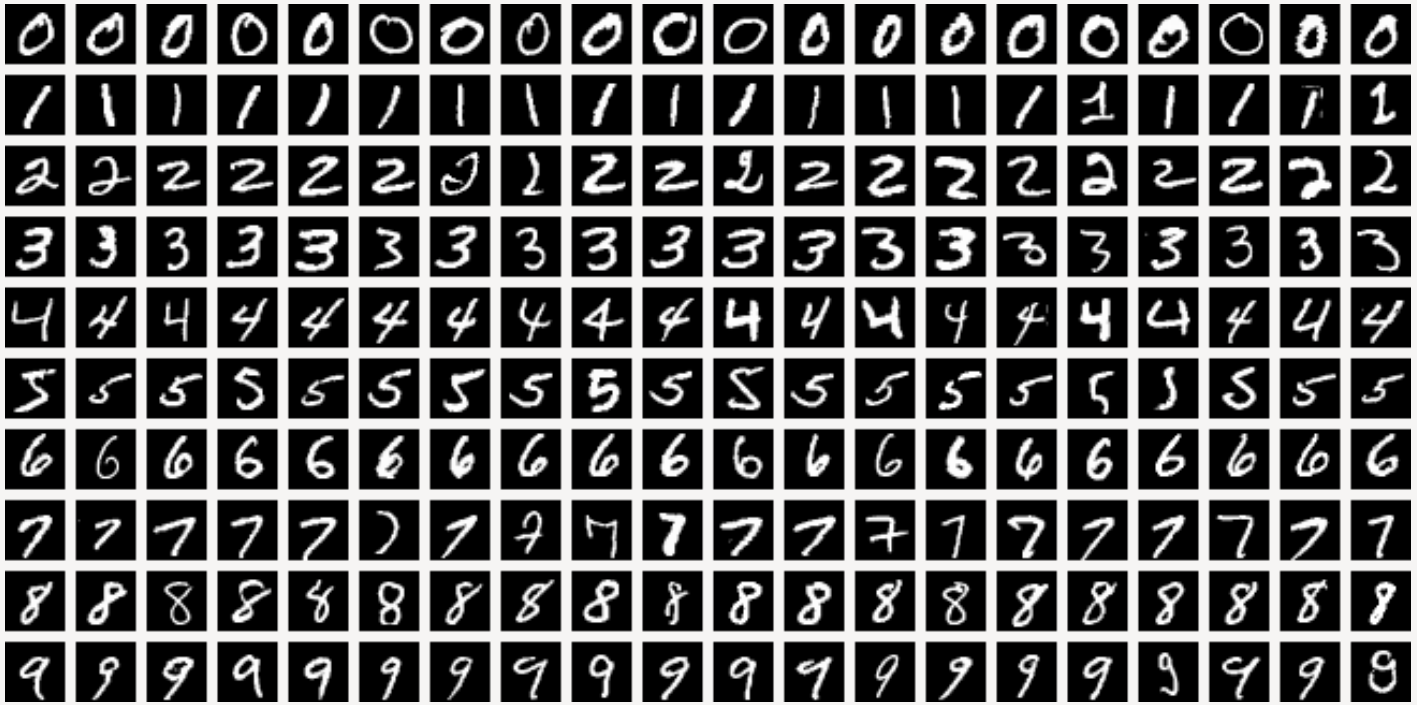
GAN

- 代码实现
  - 生成器和判别器均为多层感知器（MLP）

```
def discriminator(image, filters, kernel_size, dense_dim):
    x = conv2d(image, filters, kernel_size) # 卷积层
    x = leaky_relu(x)
    x = conv2d(x, filters*2, kernel_size)
    x = leaky_relu(x)
    x = flatten(x)                          # 展平操作
    x = dense(x, dense_dim)
    x = sigmoid(x)
    return x
```

GAN

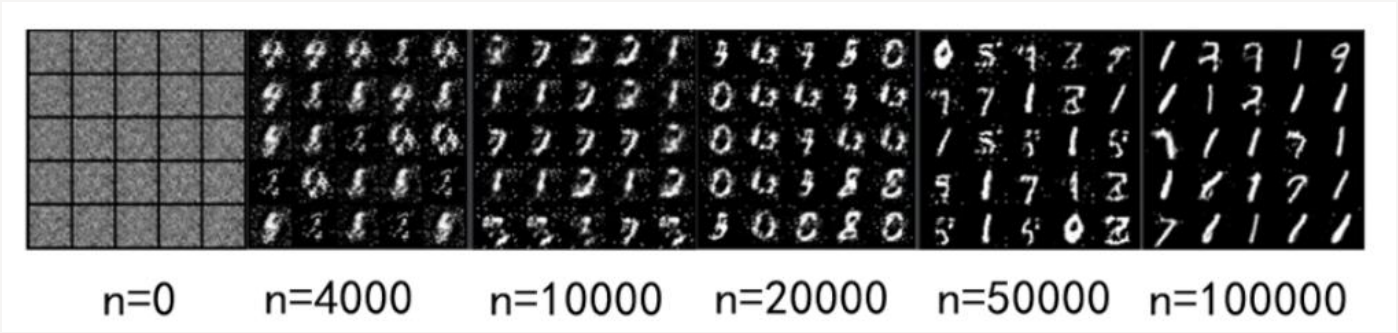
- 数据集：MNIST
  - 美国国家标准与技术研究院收集整理的大型手写数字数据集（0~9）
  - 包含了60,000个样本的训练集，以及10,000个样本的测试集
  - 黑白图像，分辨率为28x28像素

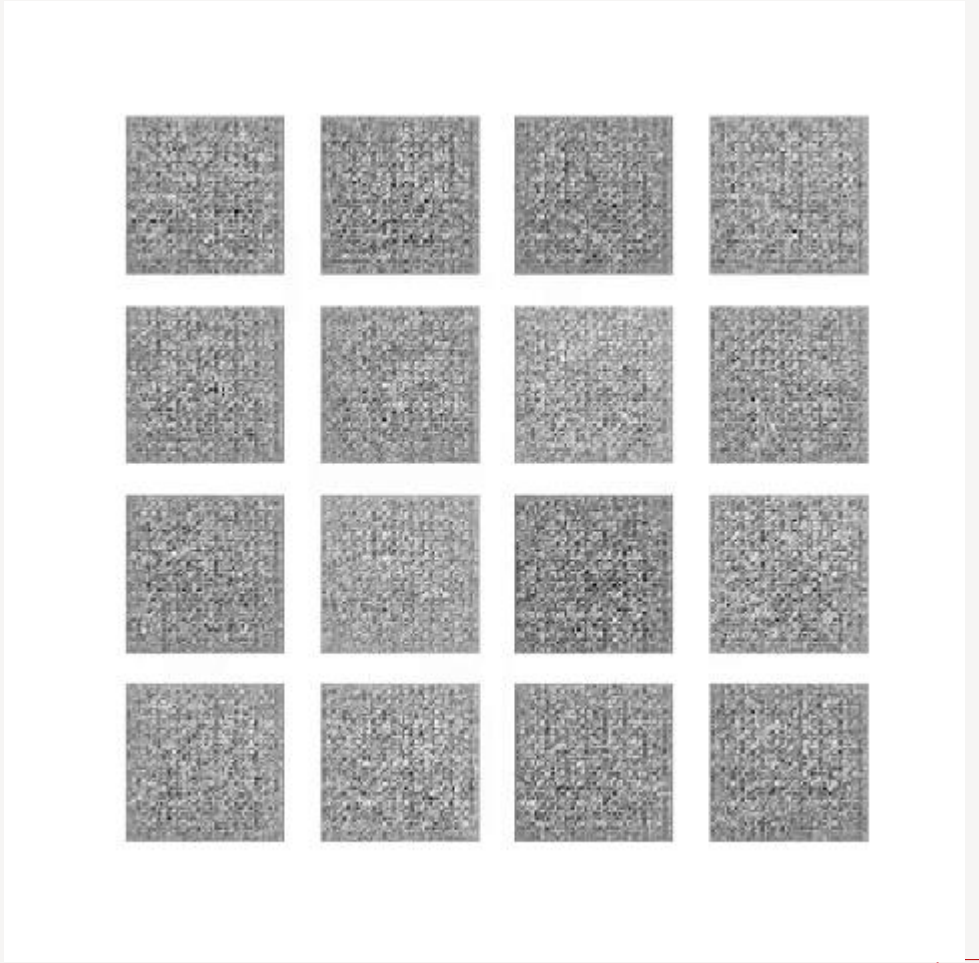


https://github.com/cvdfoundation/mnist

实验部分

- GAN的训练
  - 使用 PyTorch 实现基础 GAN
  - 生成手写数字图像示例

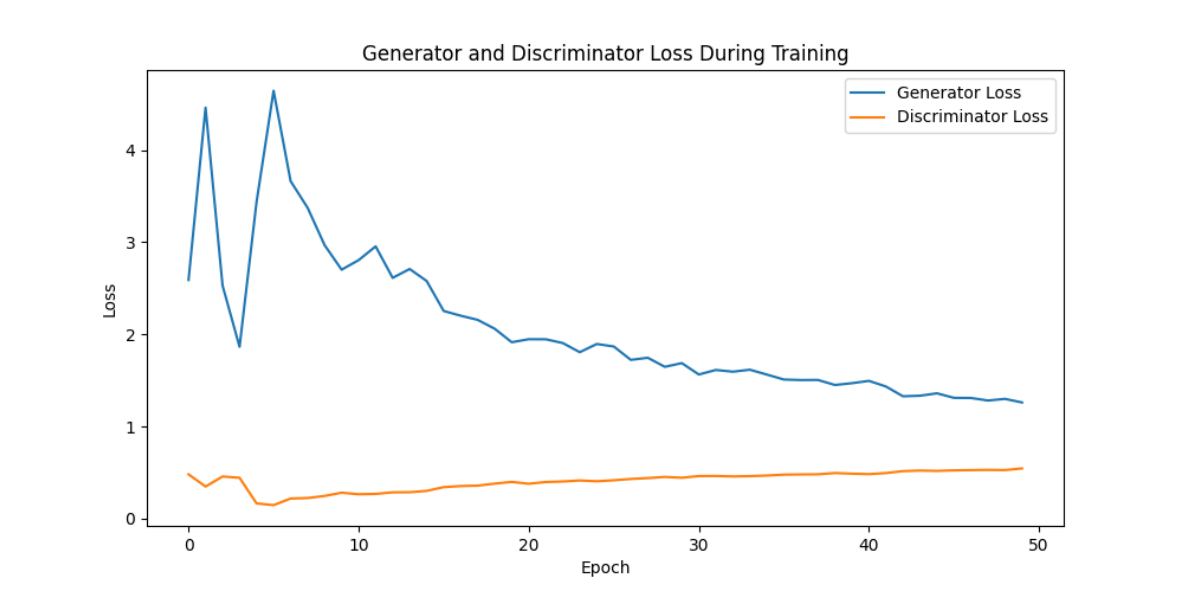


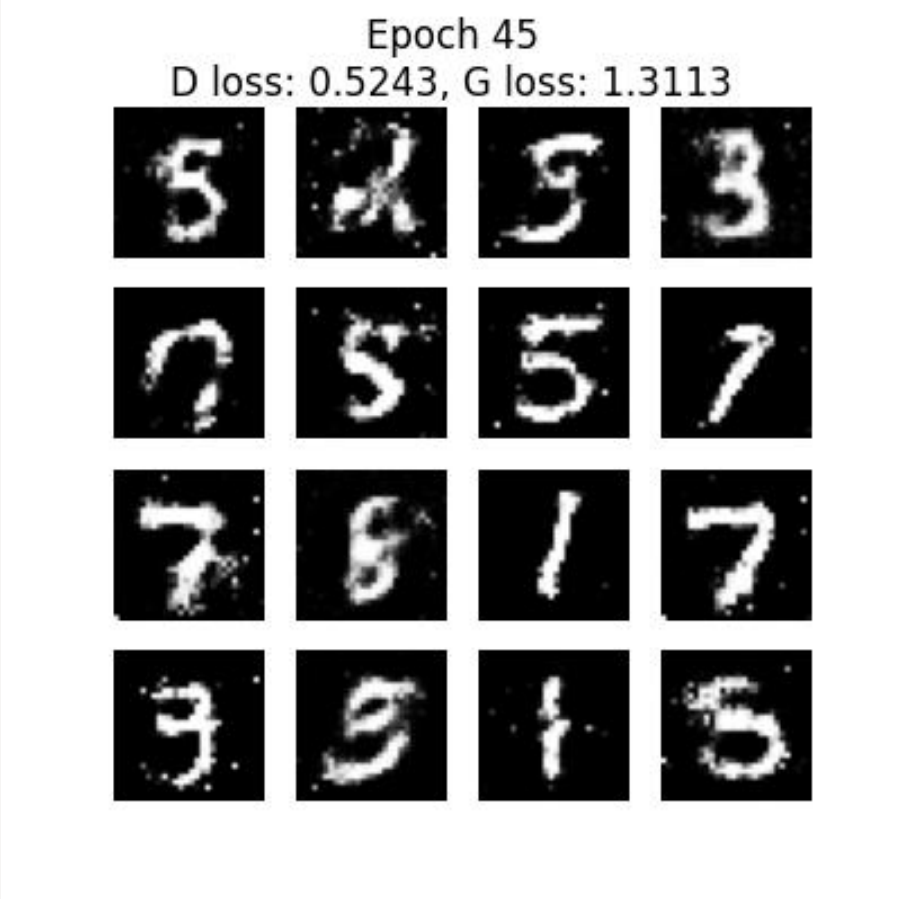


实验部分

- Vanilla GAN

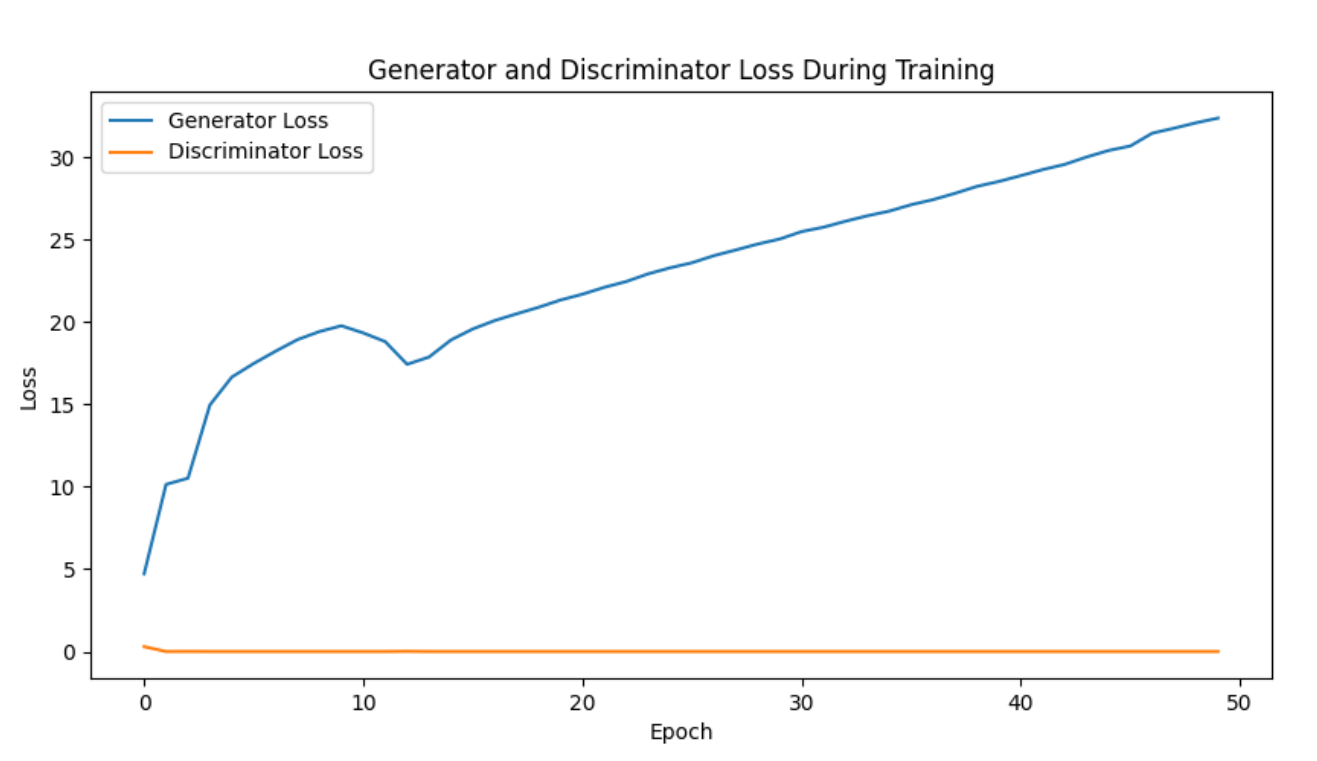
```
Epoch 48 completed in 11.11 seconds
Average G loss: 1.3002, Average D loss: 0.5275
[Epoch 49/50] [Batch 0/469] [D loss: 0.4933] [G loss: 1.8180]
[Epoch 49/50] [Batch 100/469] [D loss: 0.4937] [G loss: 1.4479]
[Epoch 49/50] [Batch 200/469] [D loss: 0.4669] [G loss: 1.3818]
[Epoch 49/50] [Batch 300/469] [D loss: 0.4991] [G loss: 1.2085]
[Epoch 49/50] [Batch 400/469] [D loss: 0.5462] [G loss: 1.2745]
Epoch 49 completed in 11.09 seconds
Average G loss: 1.2612, Average D loss: 0.5453
Training complete and models saved.
```

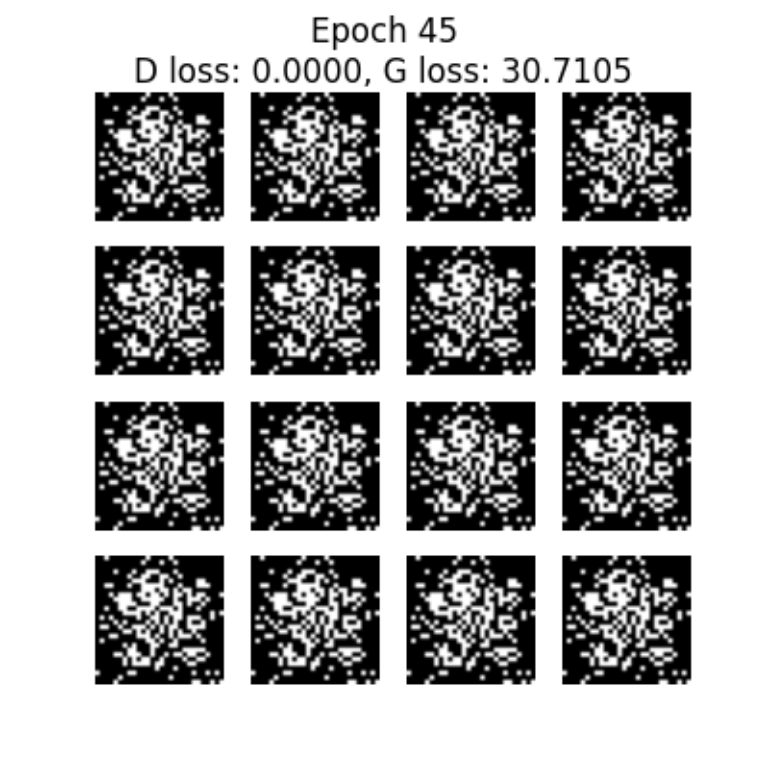




实验部分

- Vanilla GAN
  - Failure case

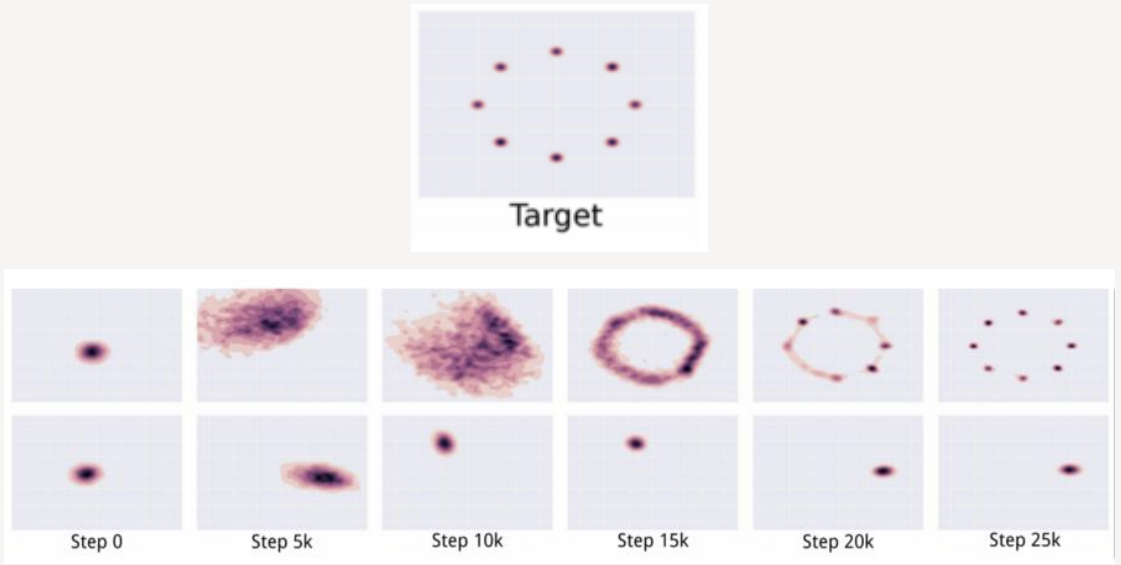




GAN

- 模式崩塌（Mode Collapse）
  - G 生成的数据倾向于生成相同的数据，丧失了多样性
  - 生成不真实的样本比生成单一的真实样本具有更大的惩罚

$$\min_G \max_D V(G,D) \neq \min_D \max_G V(G,D)$$



Metz L, Poole B, Pfau D, et al. Unrolled generative adversarial networks. arXiv:1611.02163.

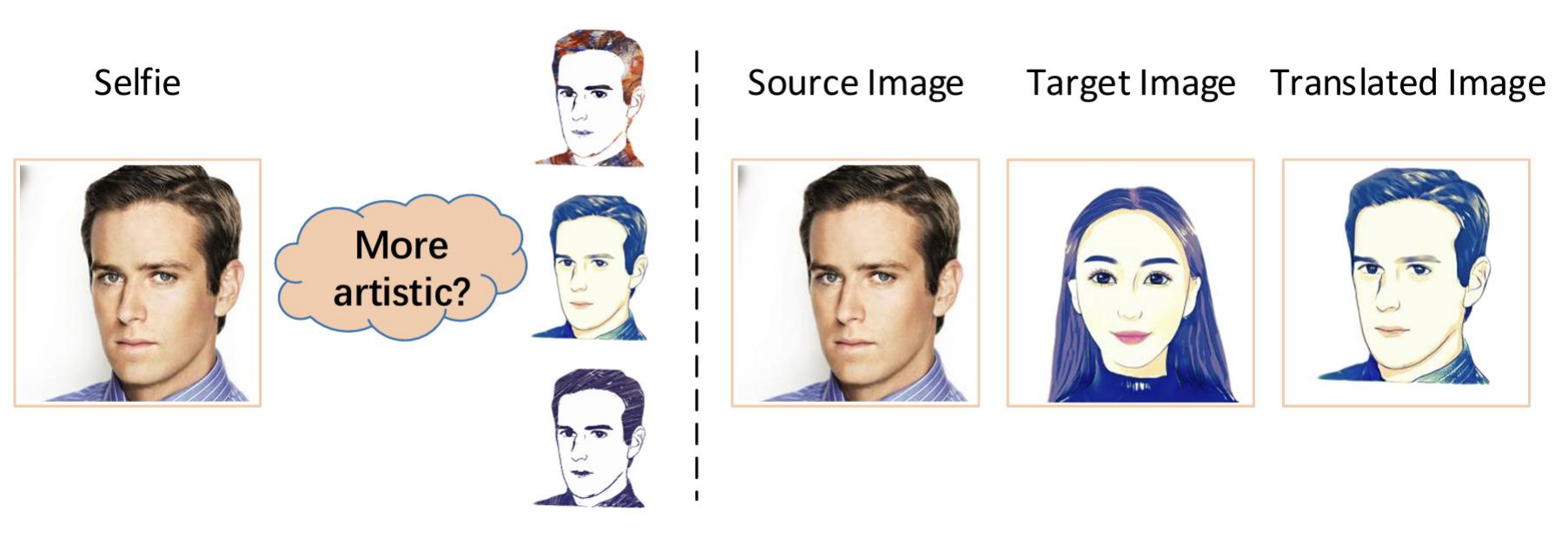
目录 / CONTENTS

02

CycleGAN

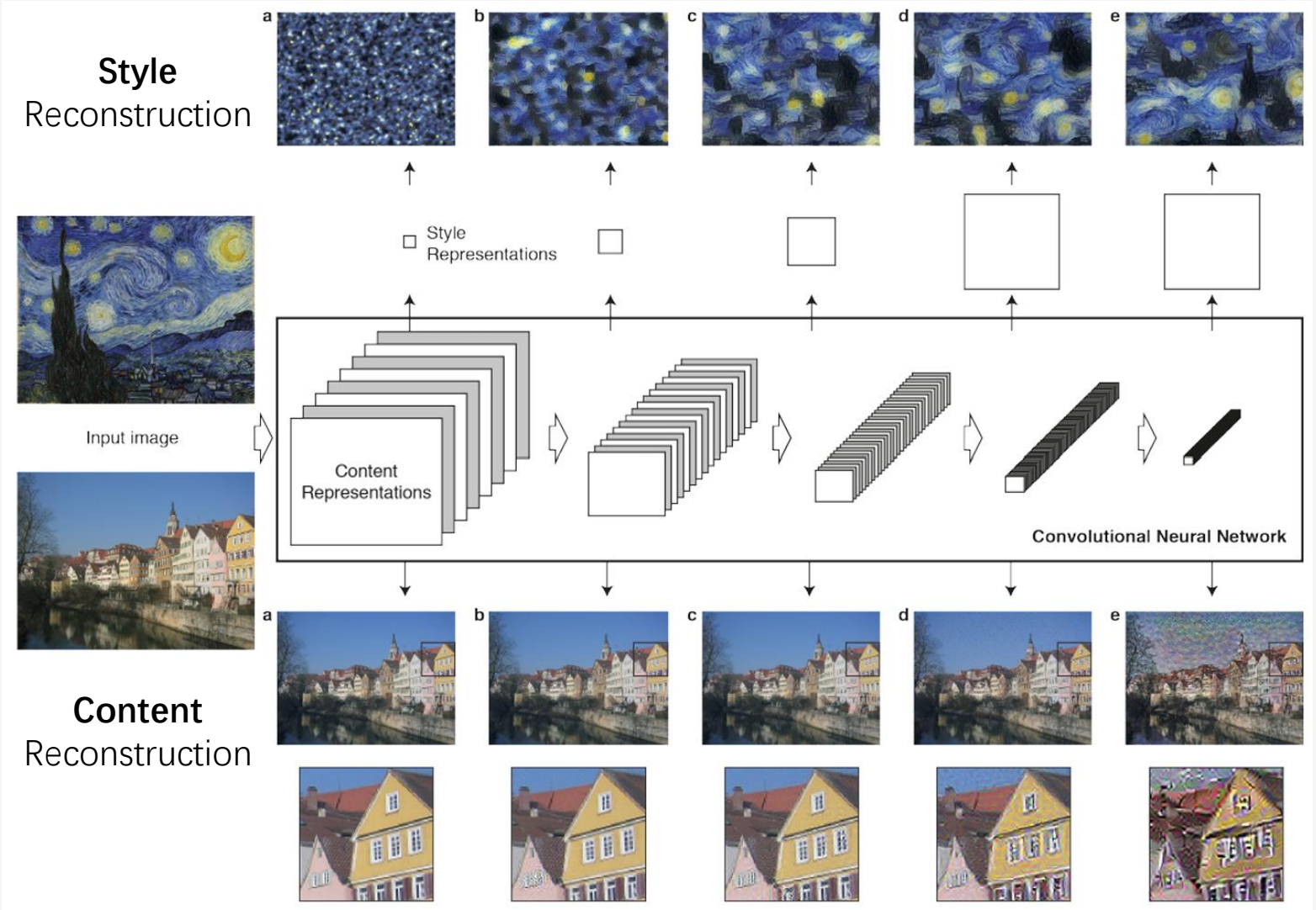
图像翻译

- Image-to-Image Translation
  - Style Transfer（风格迁移）
  - 将一幅图像的**风格**应用到另一幅图像上，同时保留后者的**内容**



Pang Y, Lin J, Qin T, et al. Image-to-image translation: Methods and applications[J]. IEEE Transactions on Multimedia, 2021.

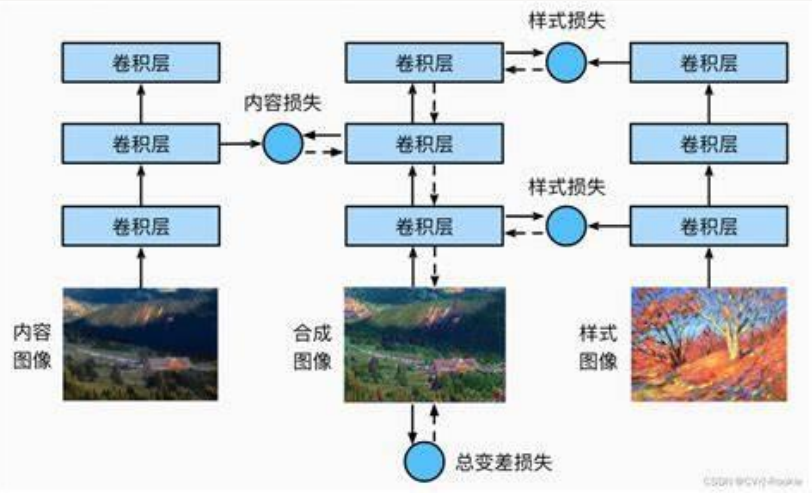
图像翻译



Gatys L A, Ecker A S, Bethge M. A neural algorithm of artistic style. arXiv:1508.06576.

图像翻译

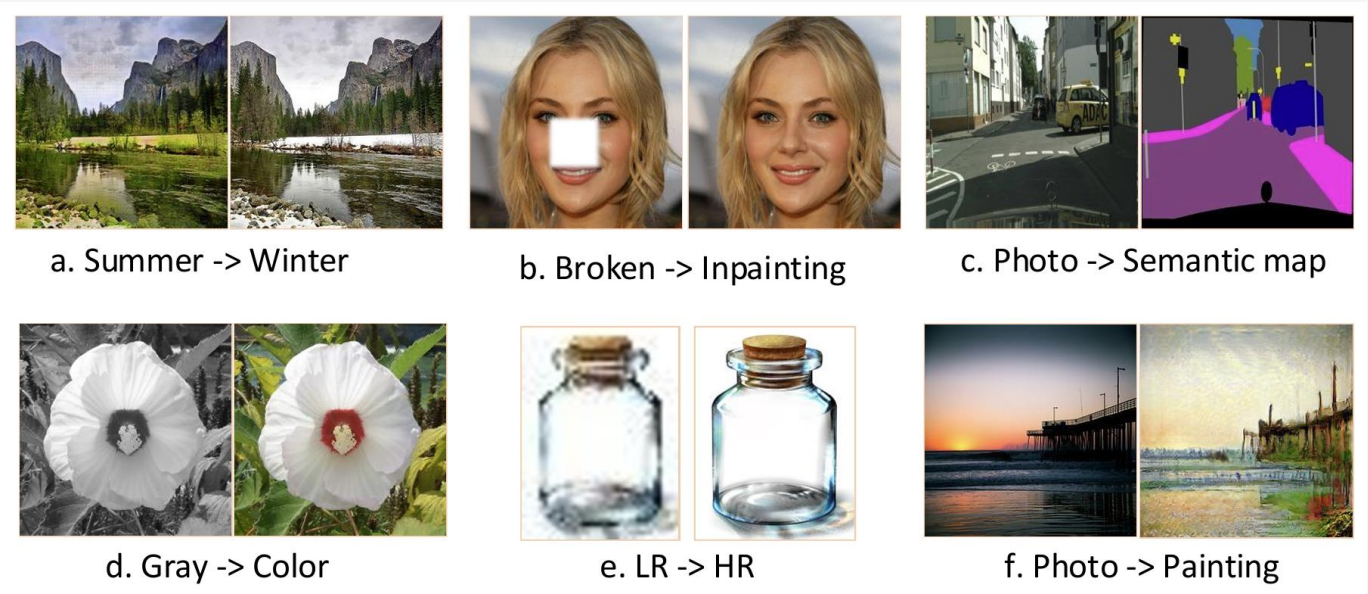
- 内容损失（Content Loss）
  - 衡量目标图像和内容图像之间的差异
  - 基于两幅图像在某一层输出特征图之间的欧氏距离来衡量
  - $\mathcal{L}_{content}(\vec{p},\vec{x},l)=\frac{1}{2}\Sigma_{i,j}\left(F_{ij}^{l}-P_{ij}^{l}\right)^{2}$

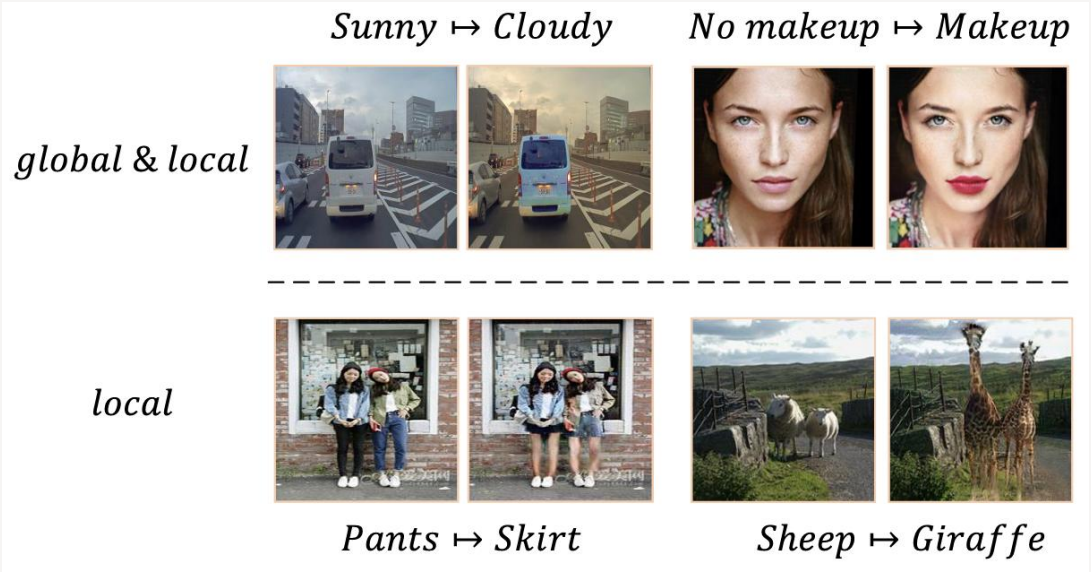


- 风格损失（Style Loss）
  - 度量生成图像与风格参考图像之间风格上的差异
  - 利用Gram矩阵，计算同一层各通道特征图之间的相关系数矩阵：$G_{ij}^{l}=\sum_{k}F_{ik}^{l}F_{jk}^{l}$
  - $\mathcal{L}_{style}(\vec{a},\vec{x})=\Sigma_{l=0}^{L}w_{l}E_{l}=\Sigma_{l=0}^{L}w_{l}\frac{1}{4N_{l}^{2}M_{l}^{2}}\Sigma_{i,j}\left(G_{ij}^{l}-A_{ij}^{l}\right)^{2}$

图像翻译

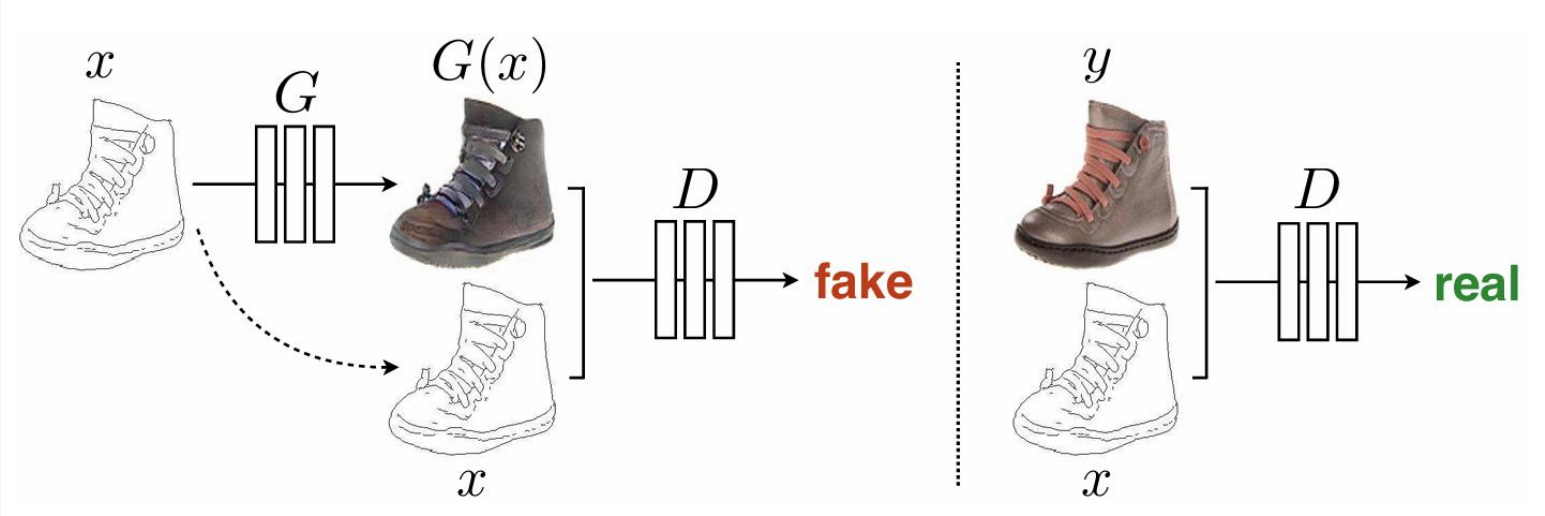
- 应用场景


a. Summer -> Winter
b. Broken -> Inpainting
c. Photo -> Semantic map
d. Gray -> Color
e. LR -> HR
f. Photo -> Painting


Sunny ↦ Cloudy
No makeup ↦ Makeup
global & local
local
Pants ↦ Skirt
Sheep ↦ Giraffe

Pix2Pix

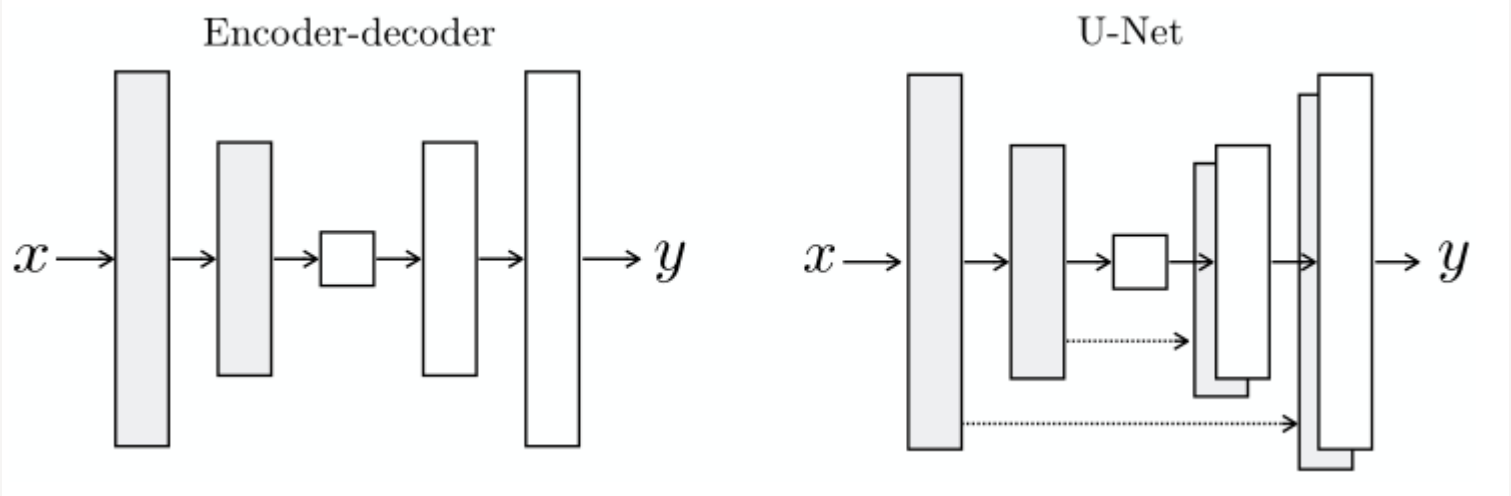
- Motivation
  - 计算机视觉方面有许多问题涉及到了将输入图像转换成相应的输出图像
  - 归根结底都是像素到像素的映射（Pixel to Pixel）
  - G: 将输入的轮廓图x编码再解码为生成图片 -> cGAN
  - D: 在轮廓图x的条件下，对生成图片判断为假，对真实图片判断为真



Isola P, Zhu J Y, Zhou T, et al. Image-to-image translation with conditional adversarial networks. CVPR 2017.

Pix2Pix

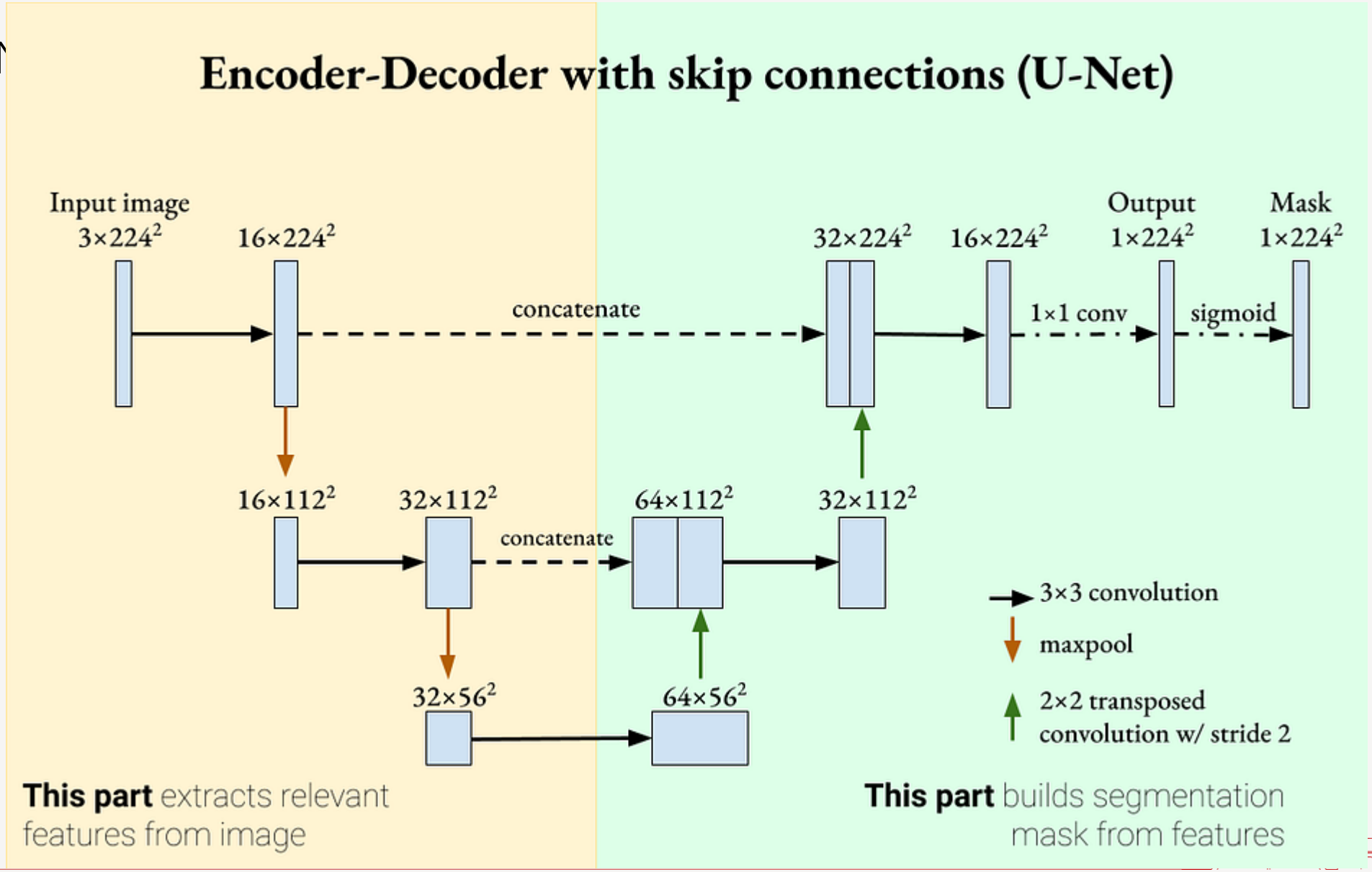
- 生成器模型
  - 输入和输出虽然表面细节不同，但具有大致相同的底层结构，需要粗略的对齐
  - 输入输出之间除了共享高层语义信息之外，还需要能够共享底层的语义信息
  - -> 引入U-Net结构



Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. MICCAI 2015.

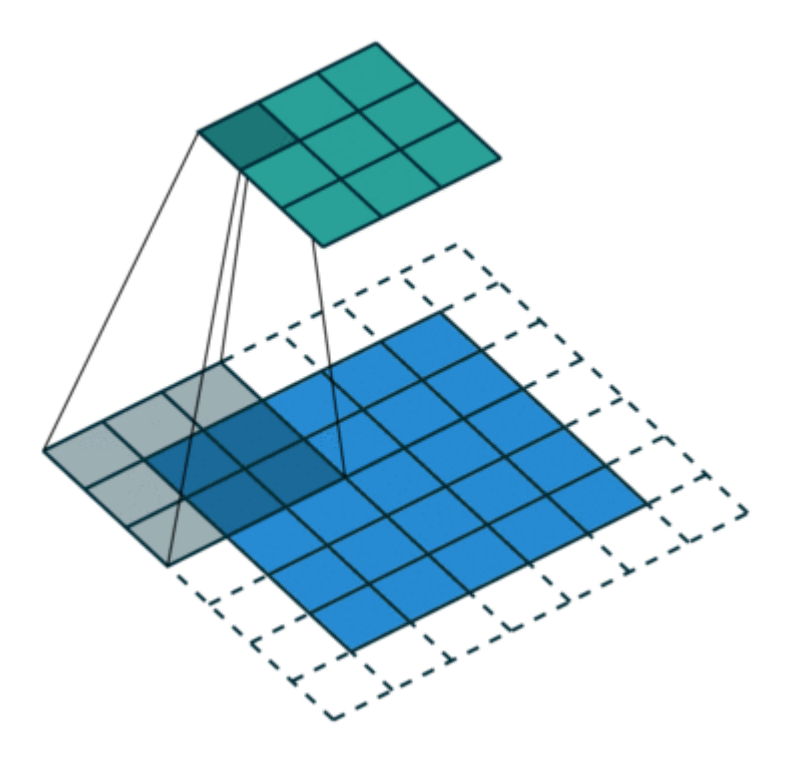
Pix2Pix

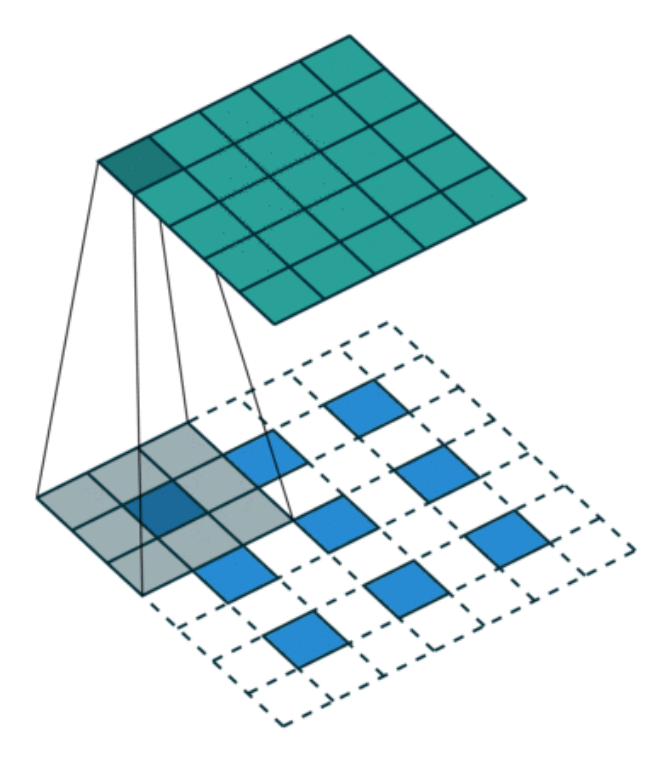
- U-N

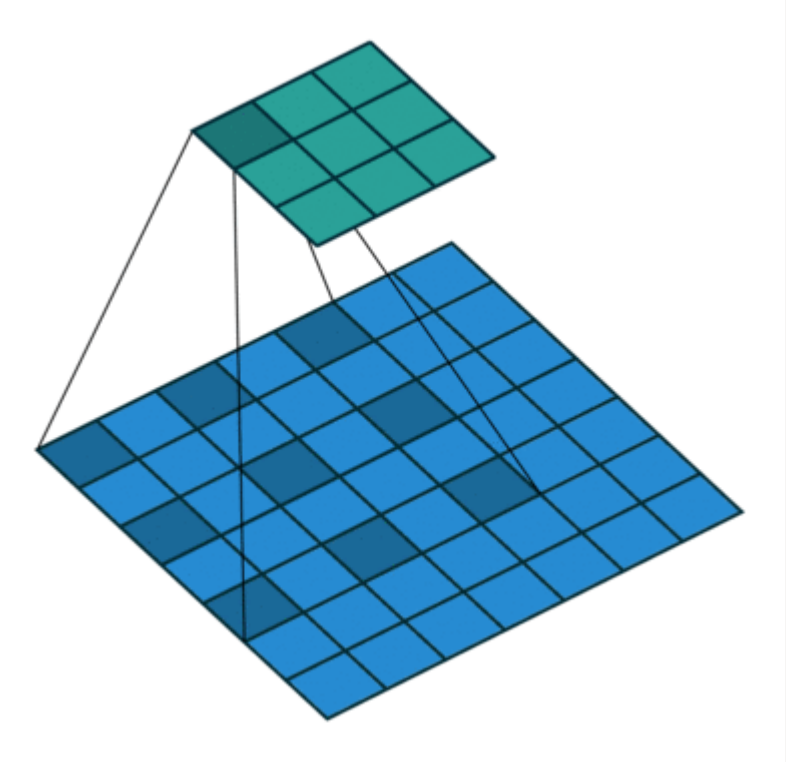

Encoder-Decoder with skip connections (U-Net)
Input image
3×224²
16×224²
concatenate
32×224²
16×224²
1×1 conv
Output
1×224²
sigmoid
Mask
1×224²
16×112²
32×112²
concatenate
64×112²
32×112²
32×56²
64×56²
3×3 convolution
maxpool
2×2 transposed convolution w/ stride 2
This part extracts relevant features from image
This part builds segmentation mask from features

Pix2Pix

- 降采样，升采样，膨胀卷积



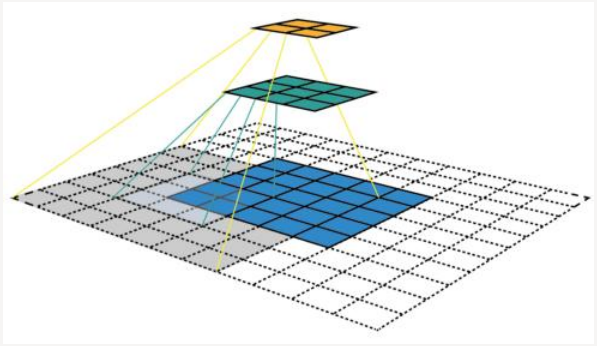


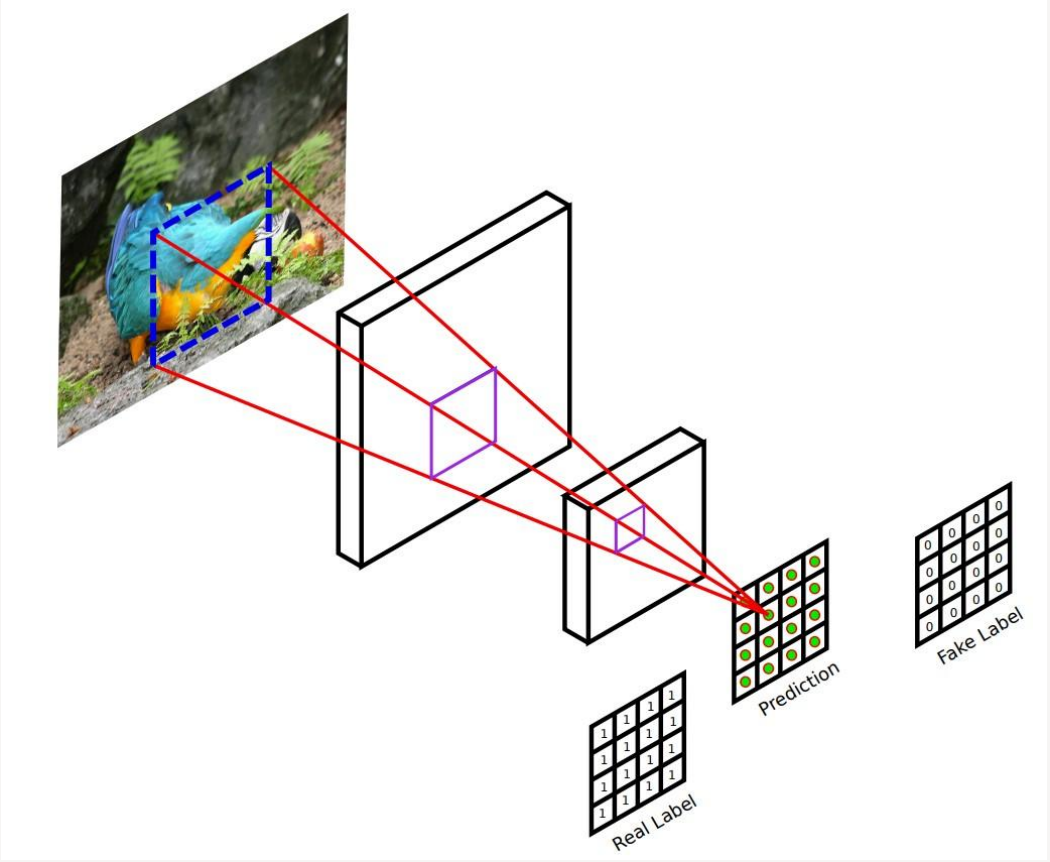


https://github.com/vdumoulin/conv_arithmetic

Pix2Pix

- 判别器模型
  - L1 loss 与 L2 loss 相对模糊的图像
  - PatchGAN能够关注更多的信息
  - 能更好的判别高频信息
  - PatchGAN -> convnet (70*70)





https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/issues/39

Pix2Pix

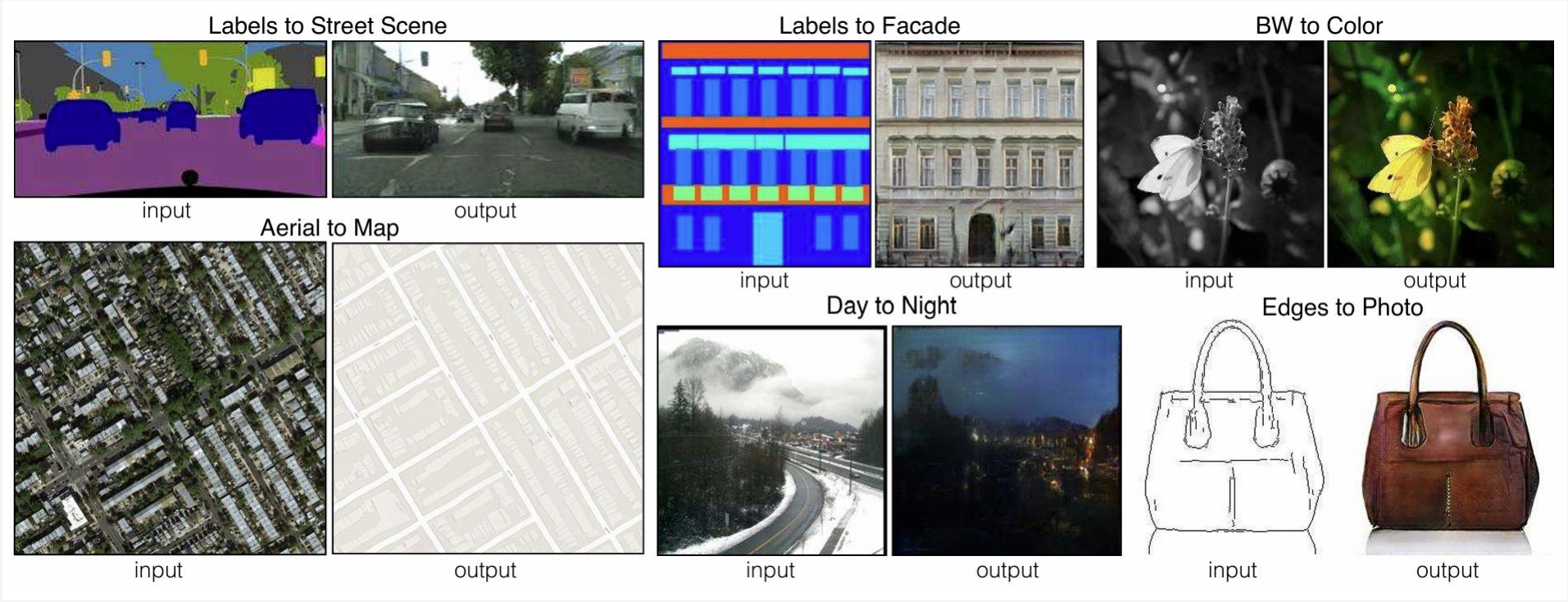
- 目标函数
  - GAN：负责高频内容
  - $\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z))]$
  - L1：负责低频内容
  - $\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\| y - G(x,z) \|_1]$
  - Final:
  - $G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G)$

Pix2Pix

- Result

Pix2Pix

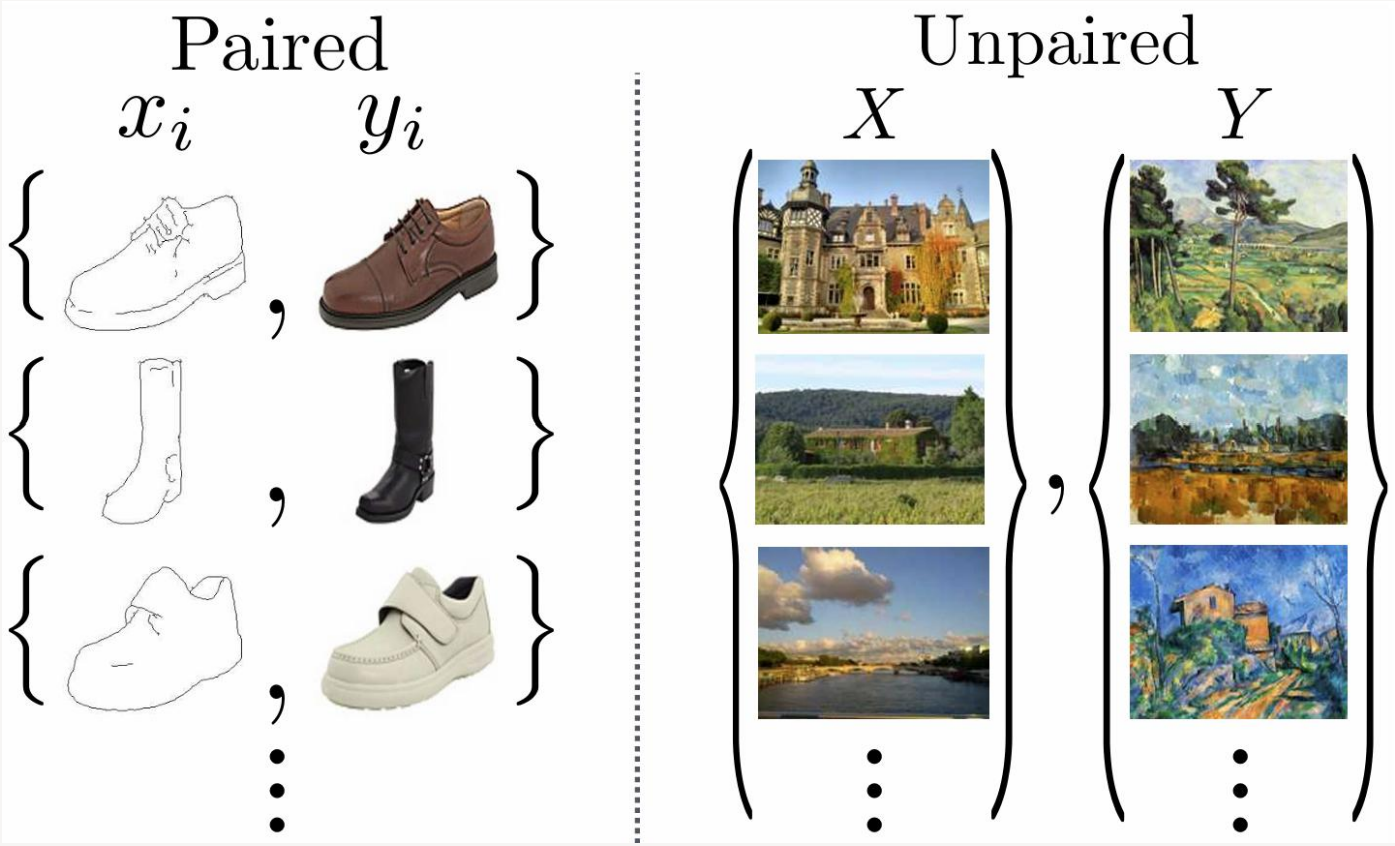
- x到y之间的一对一映射
  - 需要数据集内包含**两两配对**的图像数据
  - 输入数据与训练集数据差距较大时，生成的结果很可能没有意义



https://affinelayer.com/pixsrv/

CycleGAN

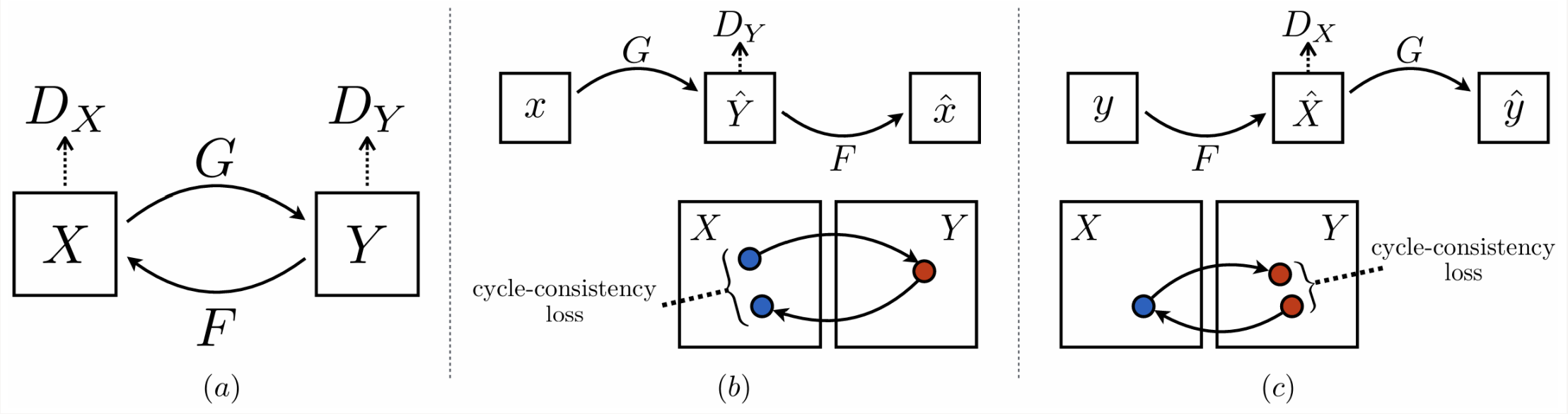
- 无需配对数据的图像翻译
  - 具有更加广泛的应用



Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. ICCV 2017.

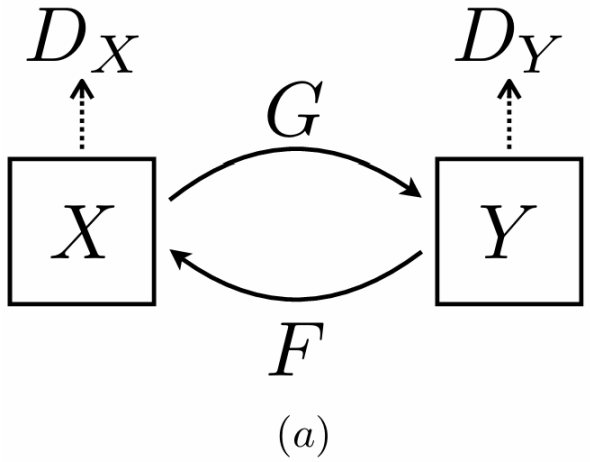
CycleGAN

- 循环一致性损失（Cycle Consistency Loss）



CycleGAN

- 循环一致性损失（Cycle Consistency Loss）



不光能够改变风格，还能保持内容不变

引入两组生成器与判别器

G: X -> Y

F: Y -> X

- $\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$
- $\mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$

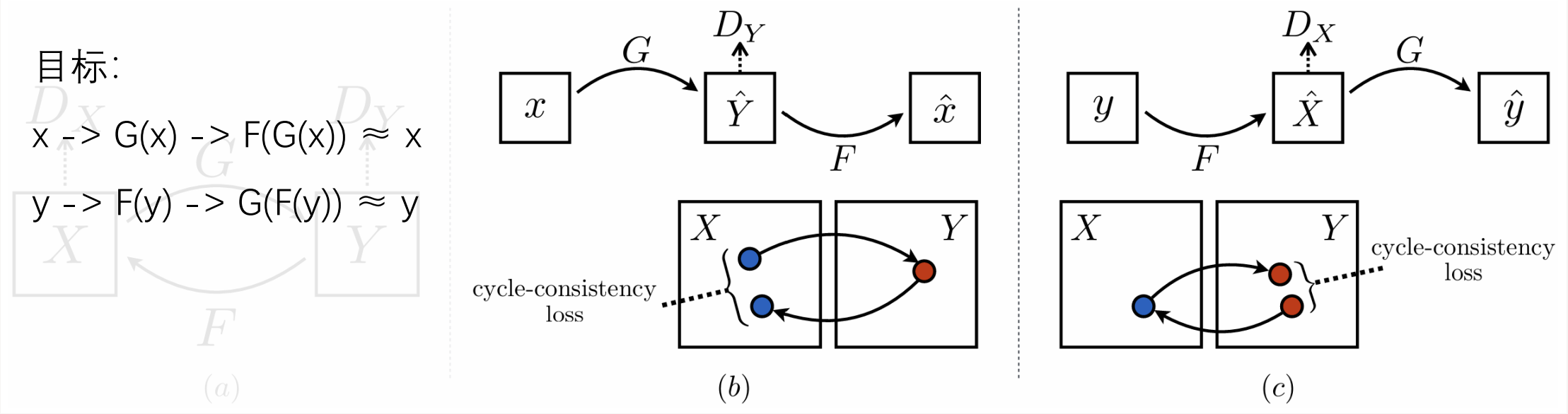
CycleGAN

- 循环一致性损失（Cycle Consistency Loss）

目标:

x -> G(x) -> F(G(x)) ≈ x

y -> F(y) -> G(F(y)) ≈ y

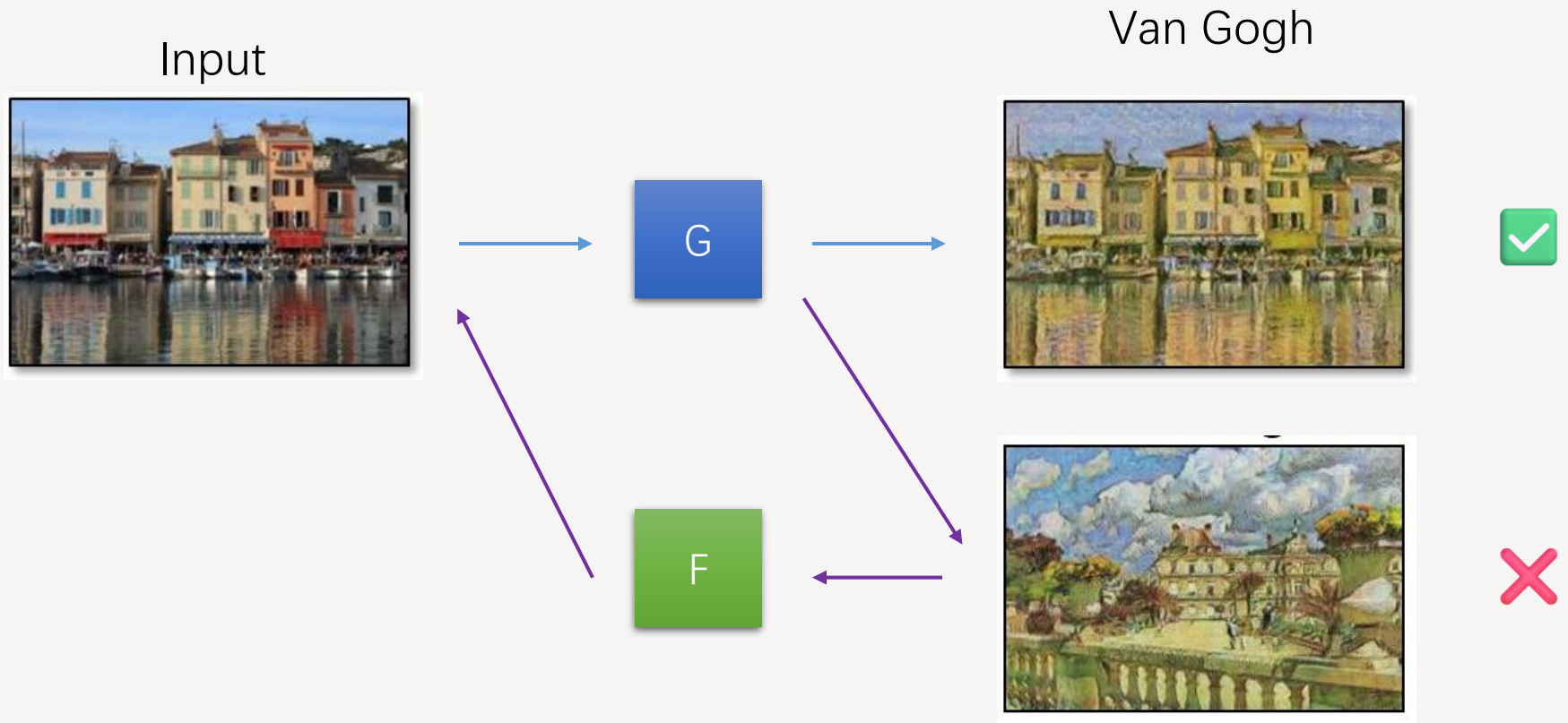


- $\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\| F(G(x)) - x \|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\| G(F(y)) - y \|_1]$
- $\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F)$

+ Identity Loss

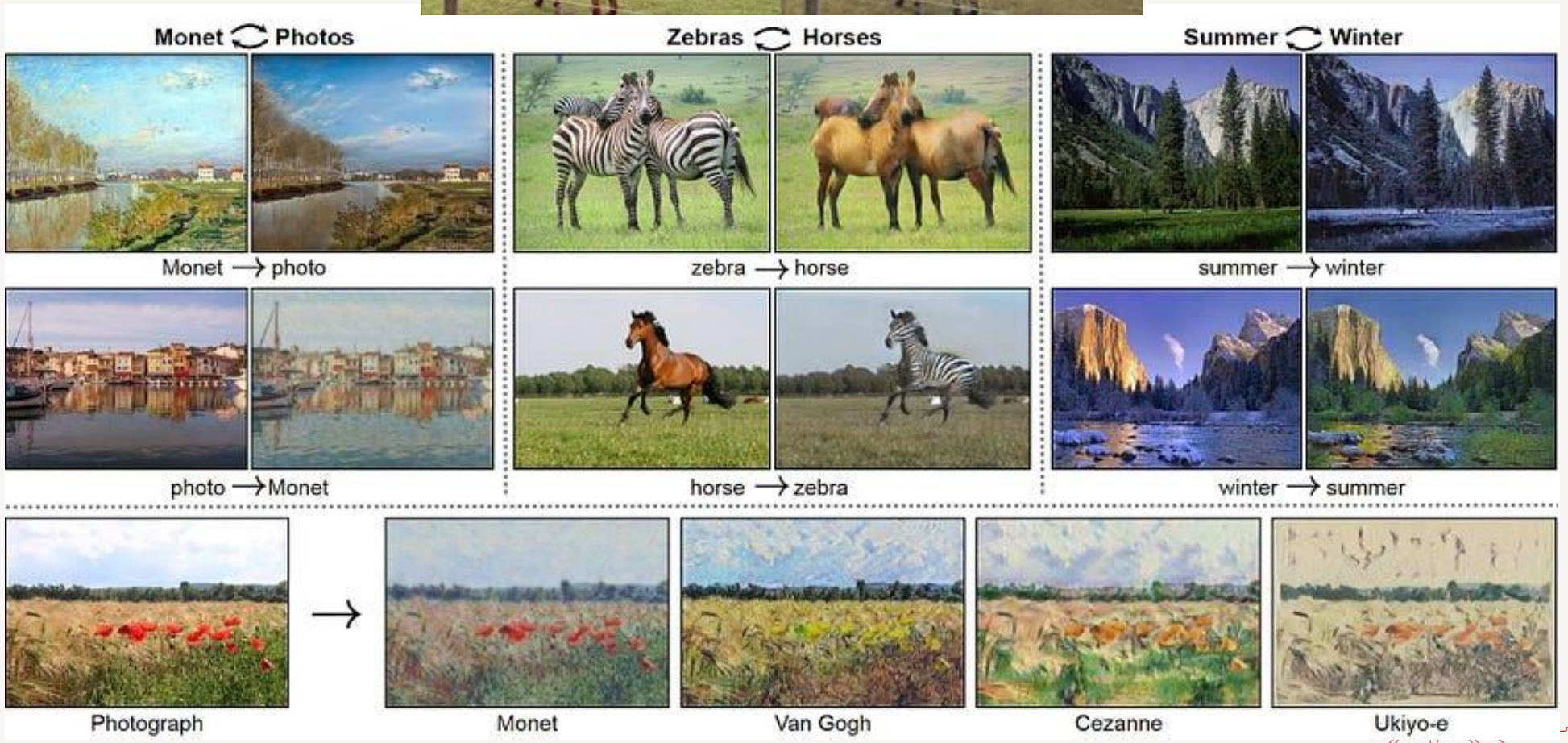
CycleGAN

- 循环一致性损失（Cycle Consistency Loss）
  - F 努力将错误的结果掰回到 x -> 风格迁移，内容改变
  - 实际而言出现的概率比较小



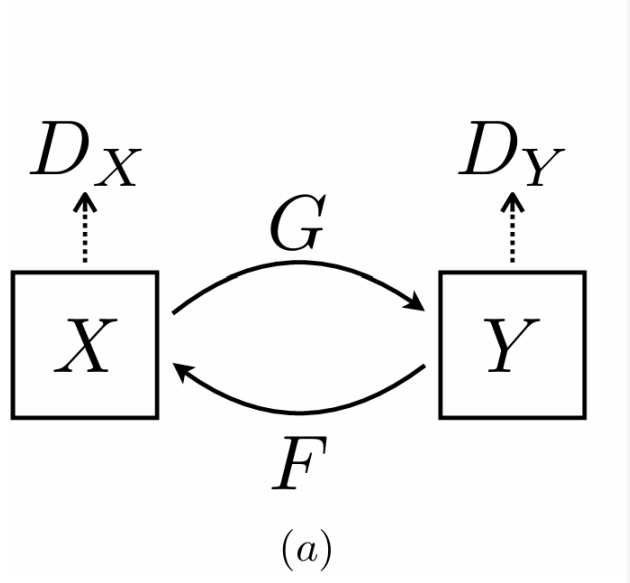
CycleGAN

- Results



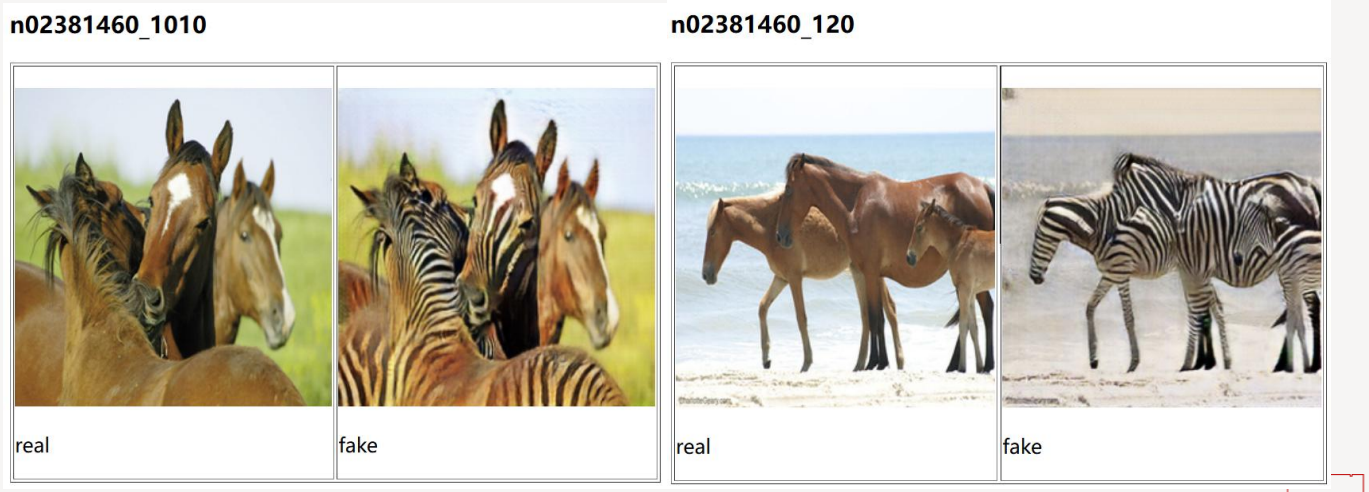
CycleGAN

- Results
  - real_A -> [netG_A] -> fake_B
  - fake_B -> [netG_B] -> rec_A
  - real_B -> [netG_B] -> fake_A
  - fake_A -> [netG_A] -> rec_B



实验部分

- 复现CycleGAN
  - 项目主页：https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
  - 数据集与权值：http://efrosgans.eecs.berkeley.edu/cyclegan/
    - horse2zebra
  - CPU only：--gpu_id -1



Q & A