生成对抗网络

Generative Adversarial Network

顾晓玲
guxl@hdu.edu.cn

学习目标

- **Fundamental Level**
  - **Vanilla GAN**
  - **DCGAN**
  - **Wasserstein GAN (WGAN)**
  - **Conditional GAN (CGAN)**
  - **ACGAN**
  - **InfoGAN**
  - **LSGAN**

学习目标

~~• Advanced Level~~

- LAPGAN
- PGGAN
- BigGAN
- StyleGAN
- BicyleGAN
- Self-Attention GAN (SAGAN)
- ……

学习目标

- **Application Level**
  - **Image-to-Image Translation**
  - **Image Completion**
  - **Image Super-Resolution**
  - **Image Manipulation**
  - **Text to Image Generation**
  - **Video Generation**

  ……

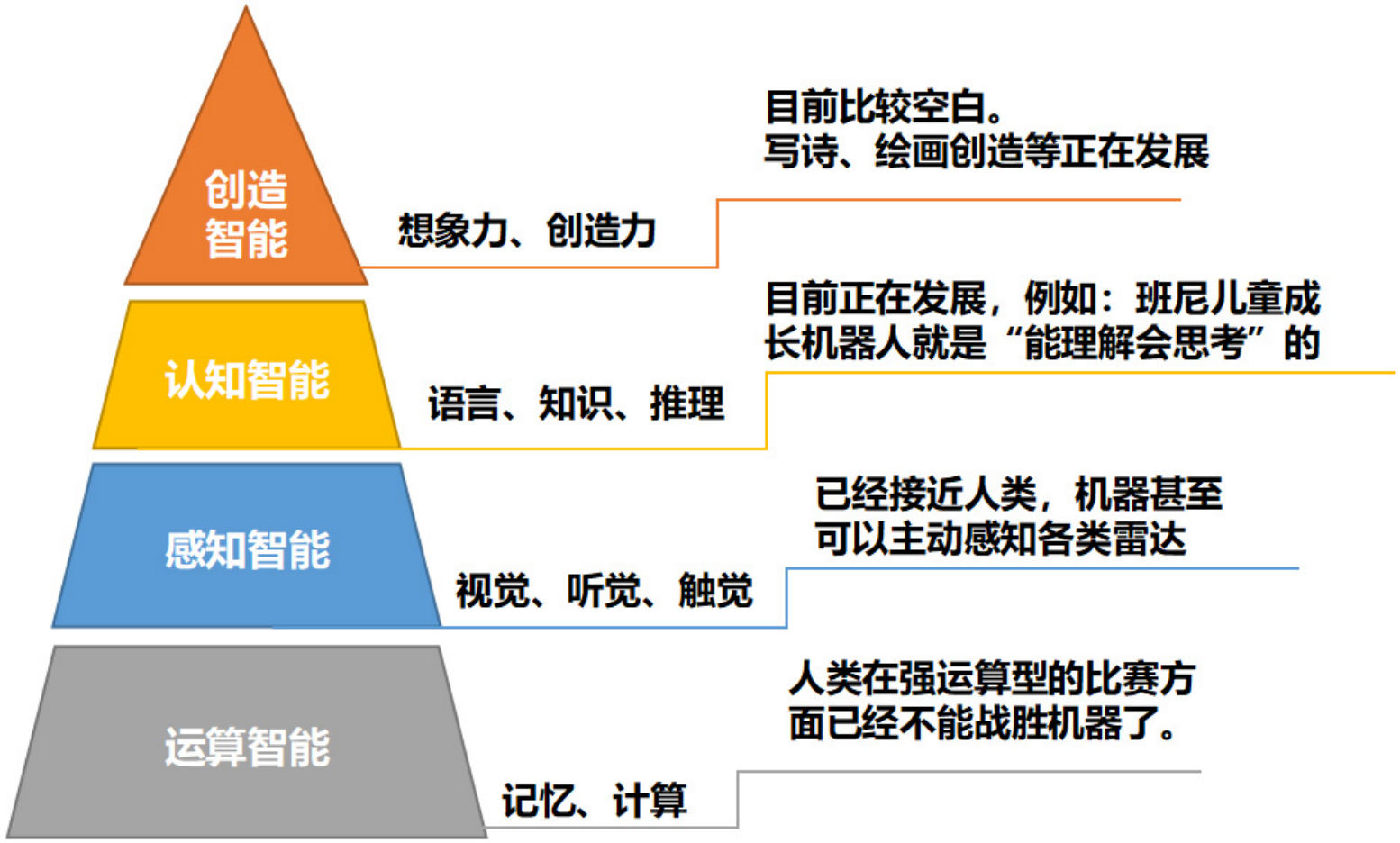
学习目标

- **Evaluation Metrics**
  - **Inception Score**
  - **Mode Score**
  - **Kernel MMD**
  - **Wasserstein Distance**
  - **Fréchet Inception Distance**

  ……

CONTENTS

Part I: 1. Background

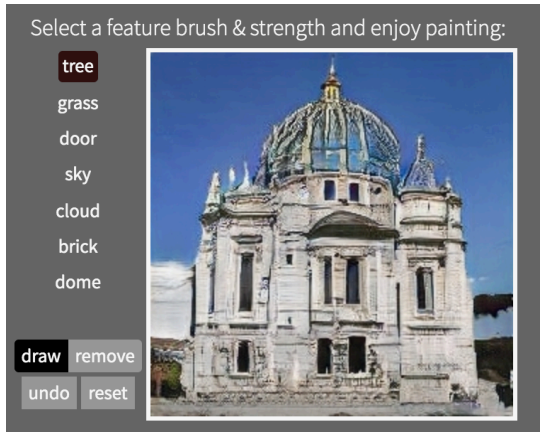


Part I: 1. Background

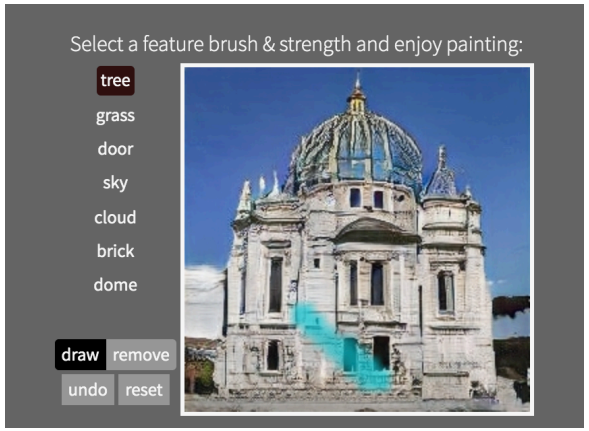
GANpaint[1]

[1] http://gandissect.res.ibm.com/ganpaint.html?project=churchoutdoor&layer=layer4

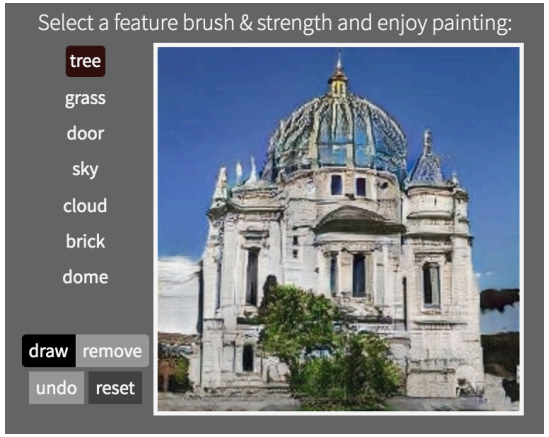
Part I: 1. Background



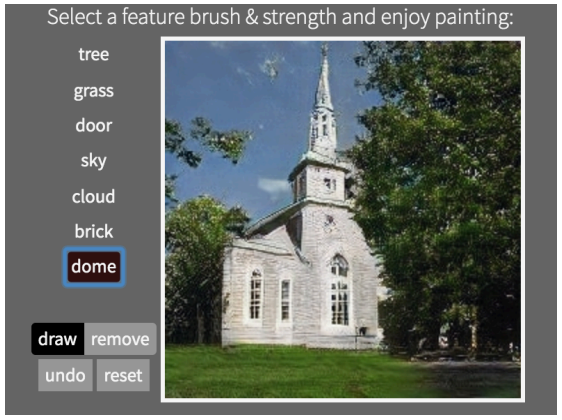
Step 1



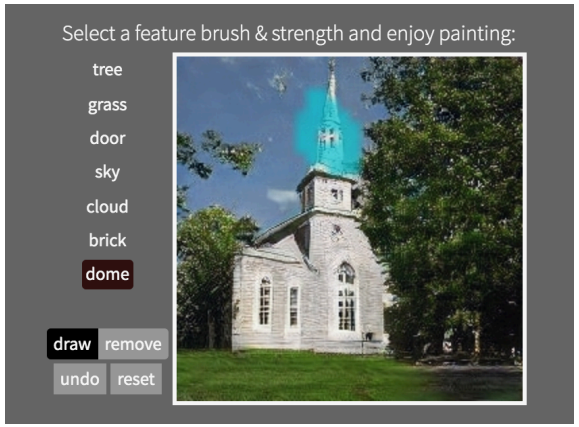
Step 2



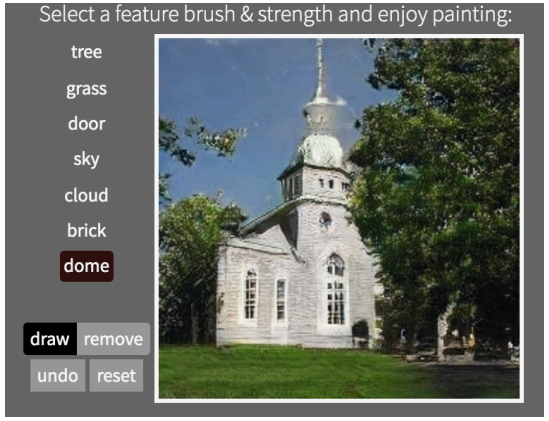
Step 3



Step 1

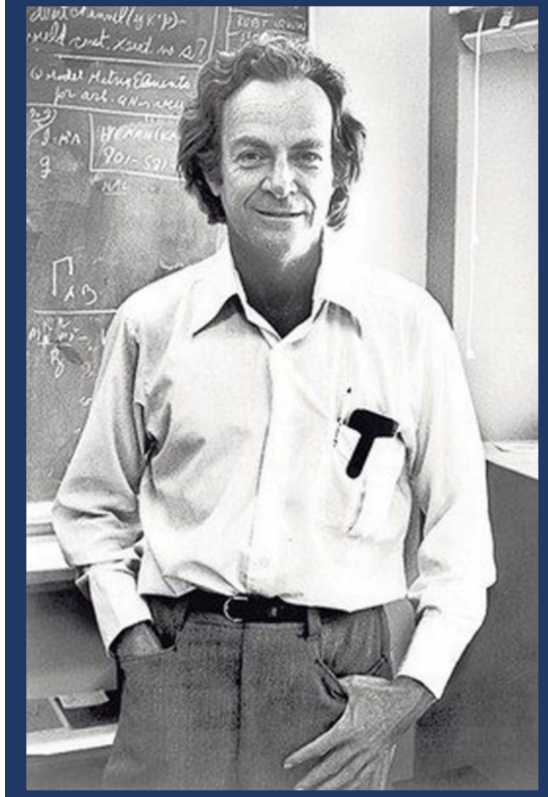


Step 2



Step 3

Part I: 1. Background



What I cannot create, I do not understand.

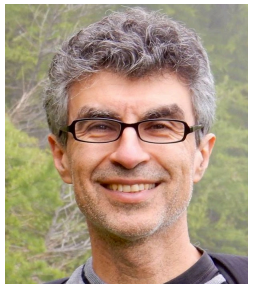
— Richard Feynman

Part I: 1. Background

- **Generative Adversarial Nets is proposed by Ian Goodfellow in 2014.**

Ian Goodfellow
Apple
Director

Supervisor

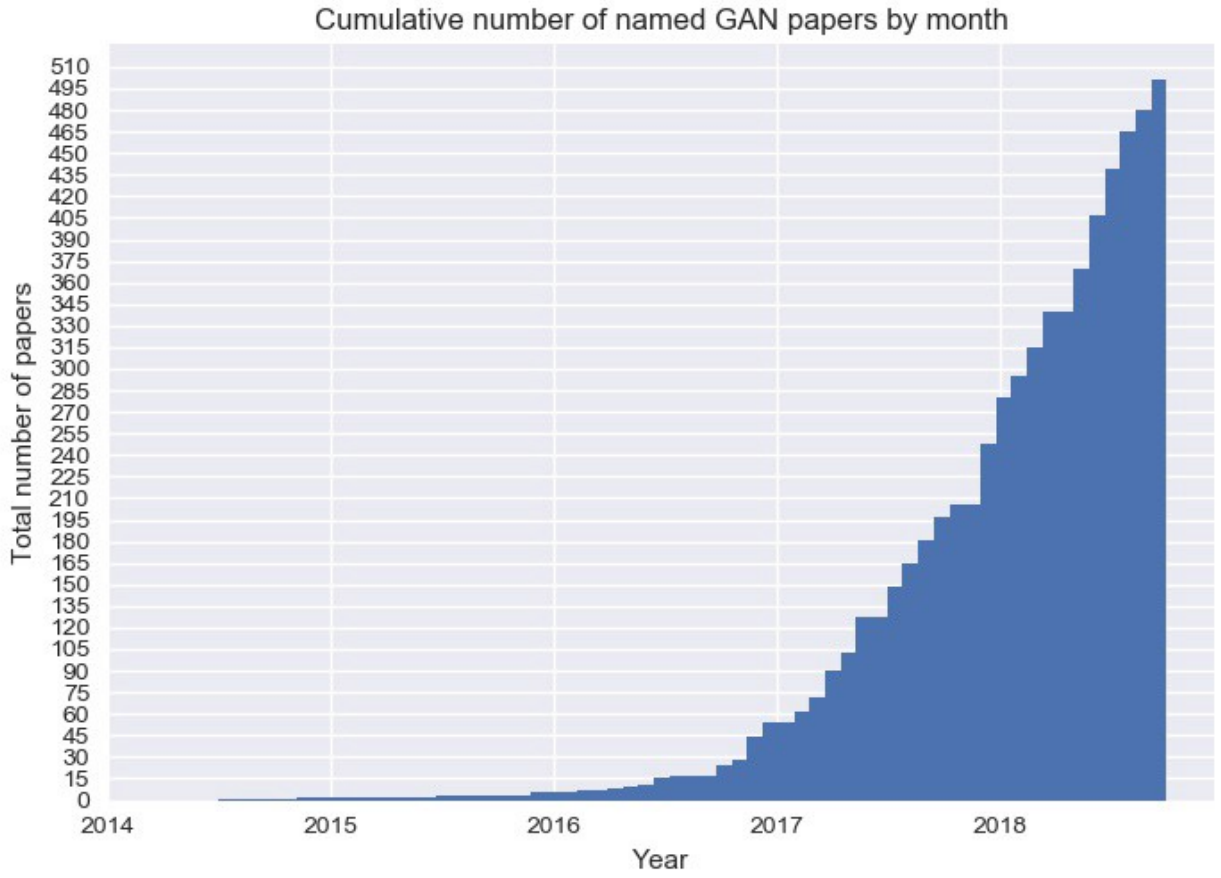


Yoshua Bengio
Université de Montréal
Full Professor

Yann LeCun

Adversarial training is the coolest thing since sliced bread.

Part I: 1. Background

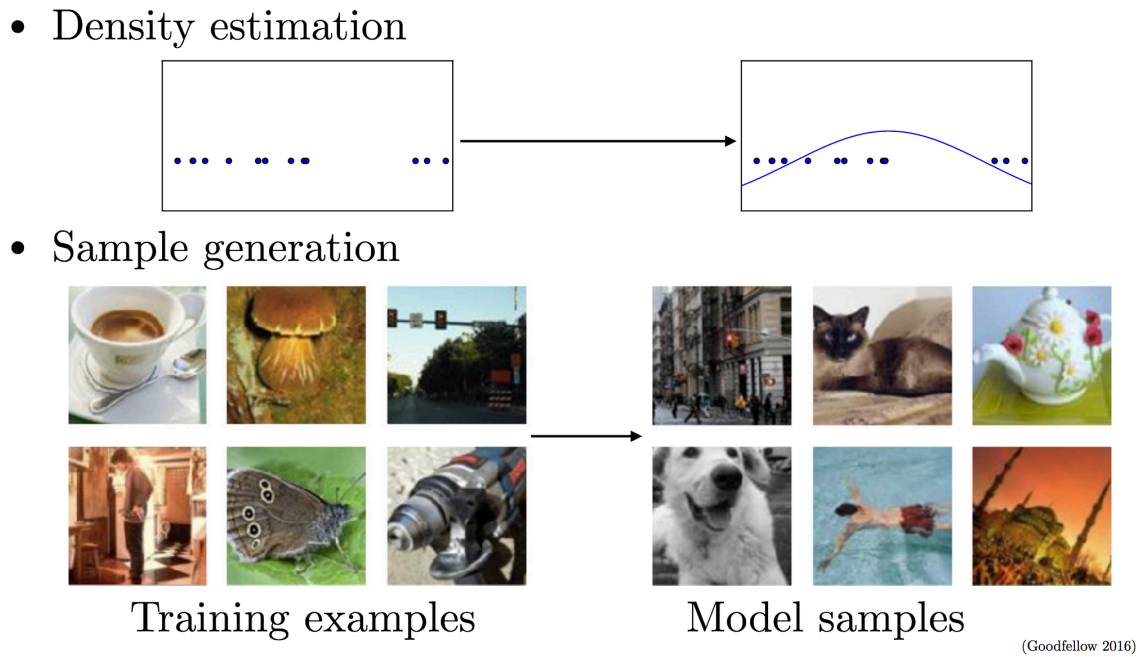


Explosive growth — All the named GAN variants cumulatively since 2014.Credit: Bruno Gavranović [2]

[2] https://deephunt.in/the-gan-zoo-79597dc8c347

Part I: 2. Generative Model

- **Any model that takes a training set, consisting of samples drawn from a distribution pdata, and learns to represent an estimate of that distribution somehow.**



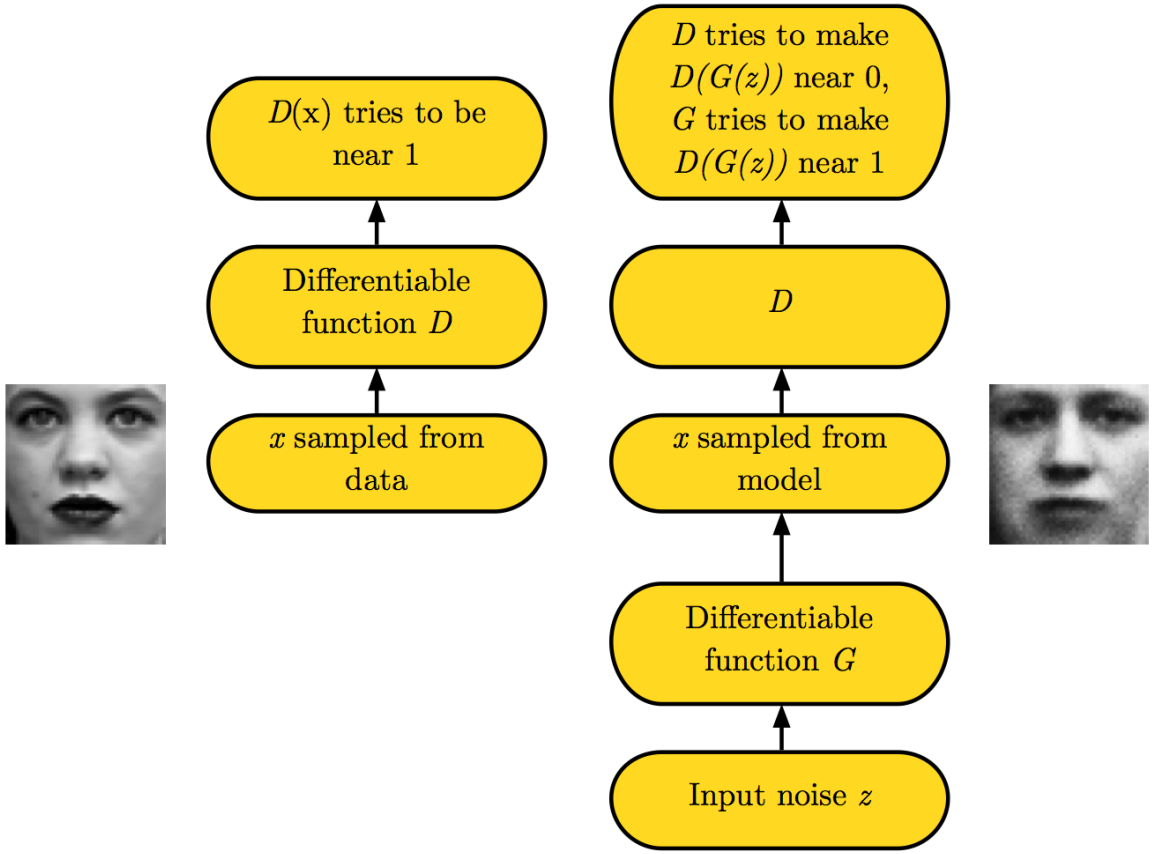
[3] Goodfellow I. NIPS 2016 tutorial: Generative adversarial networks[J]. arXiv preprint arXiv:1701.00160, 2016.

Part I: 3. Vanilla GAN

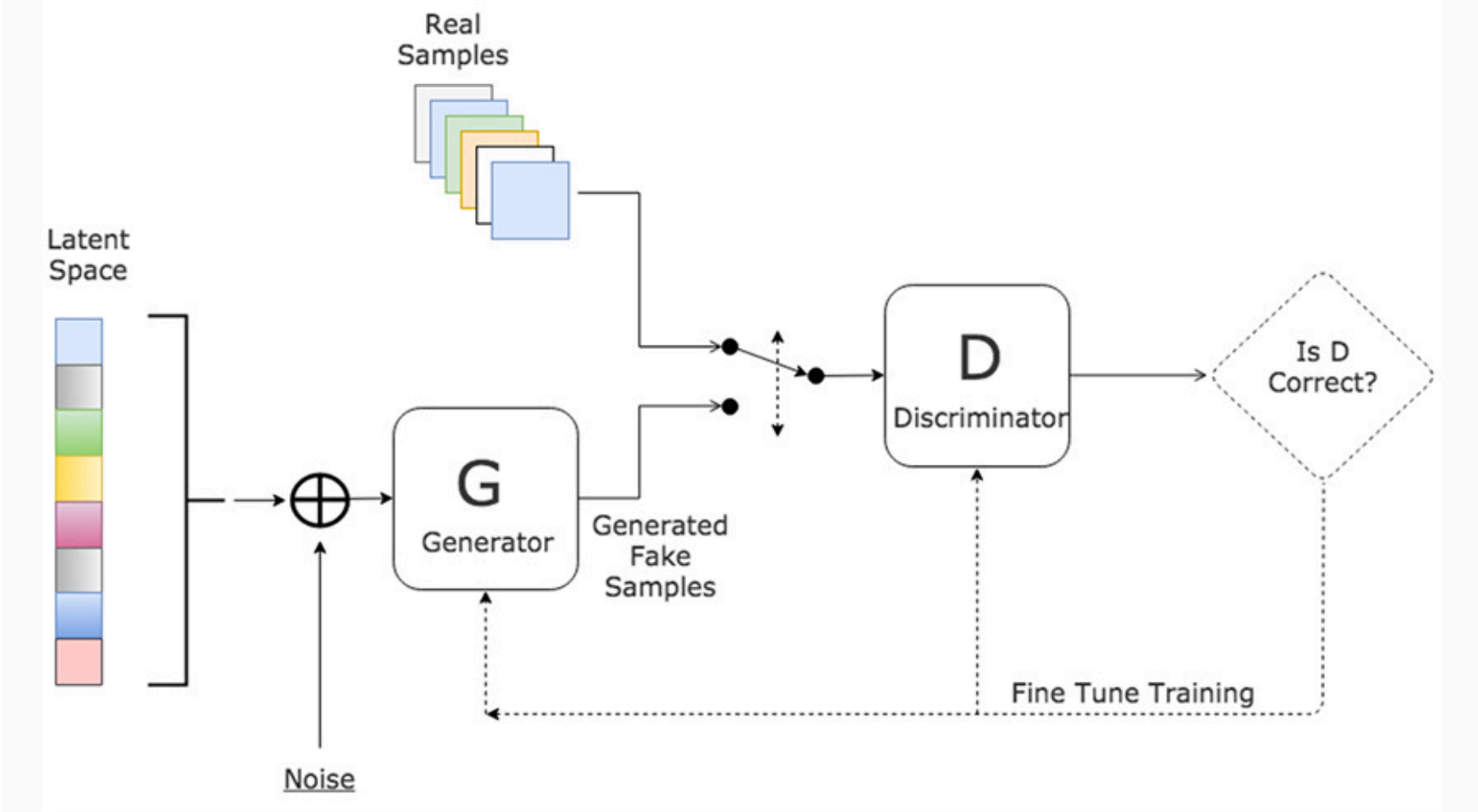
- **The basic idea of GANs is to set up a game between two players, generator and discriminator.**
- **The discriminator examines samples to determine whether they are real or fake. The discriminator learns using traditional supervised learning techniques, dividing inputs into two classes (real or fake).**
- **The generator is trained to fool the discriminator.**

[4] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.

Part I: 3. Vanilla GAN



Part I: 3. Vanilla GAN



Part I: 3. Vanilla GAN

Two-player Minimax Game

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

1) Optimizing Discriminator

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

2) Optimizing Generator

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

Part I: 3. Vanilla GAN

Global Optimality of pg = pdata

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \end{aligned} \tag{4}$$

Part I: 3. Vanilla GAN

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D^*_G, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \,\|\, p_g\right) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative, and zero iff they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data distribution. □

Part I: 3. Vanilla GAN

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:

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

  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:

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

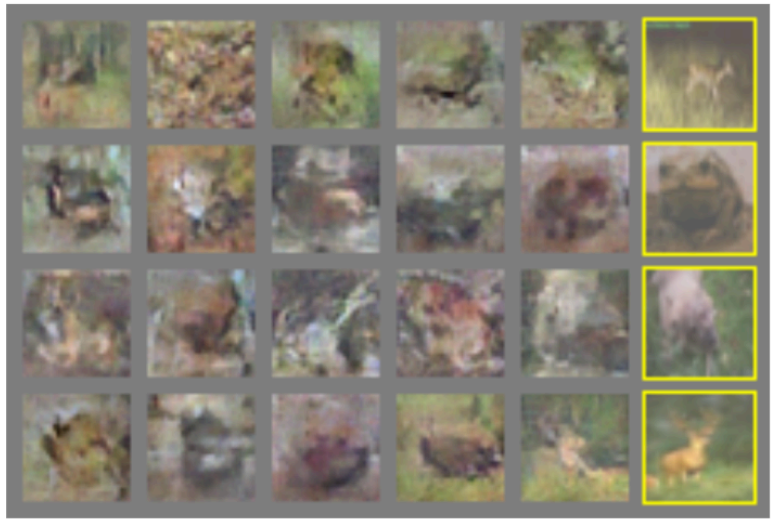
end for

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

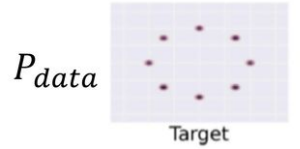
Part I: 3. Vanilla GAN

Drawbacks

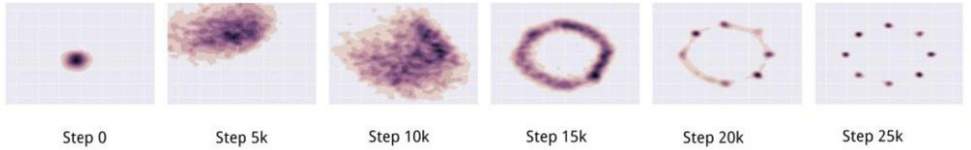
- Mode Collapse
- Hard to train
- Low-resolution



Mode Collapse



What we want …



In reality …

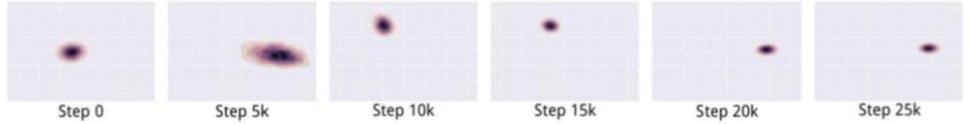


Figure credit, Metz et al, 2016

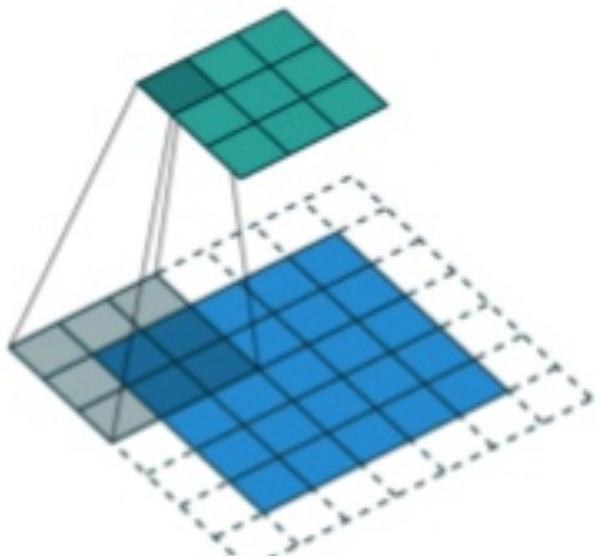
Part I: 4. DCGAN

Architecture guidelines for stable Deep Convolutional GANs:

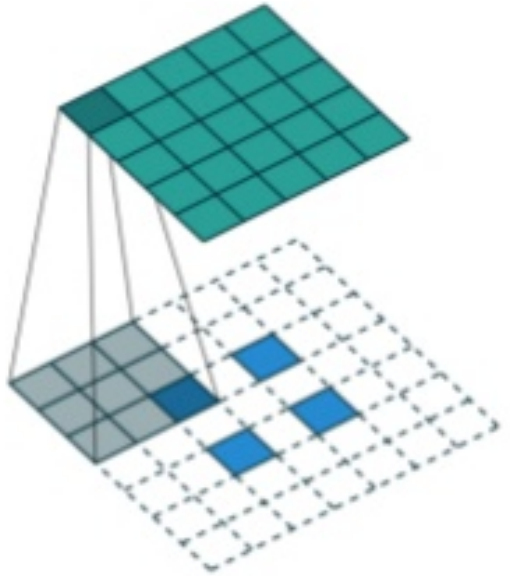
- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except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

[5] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks[J]. arXiv preprint arXiv:1511.06434, 2015.

Part I: 4. DCGAN

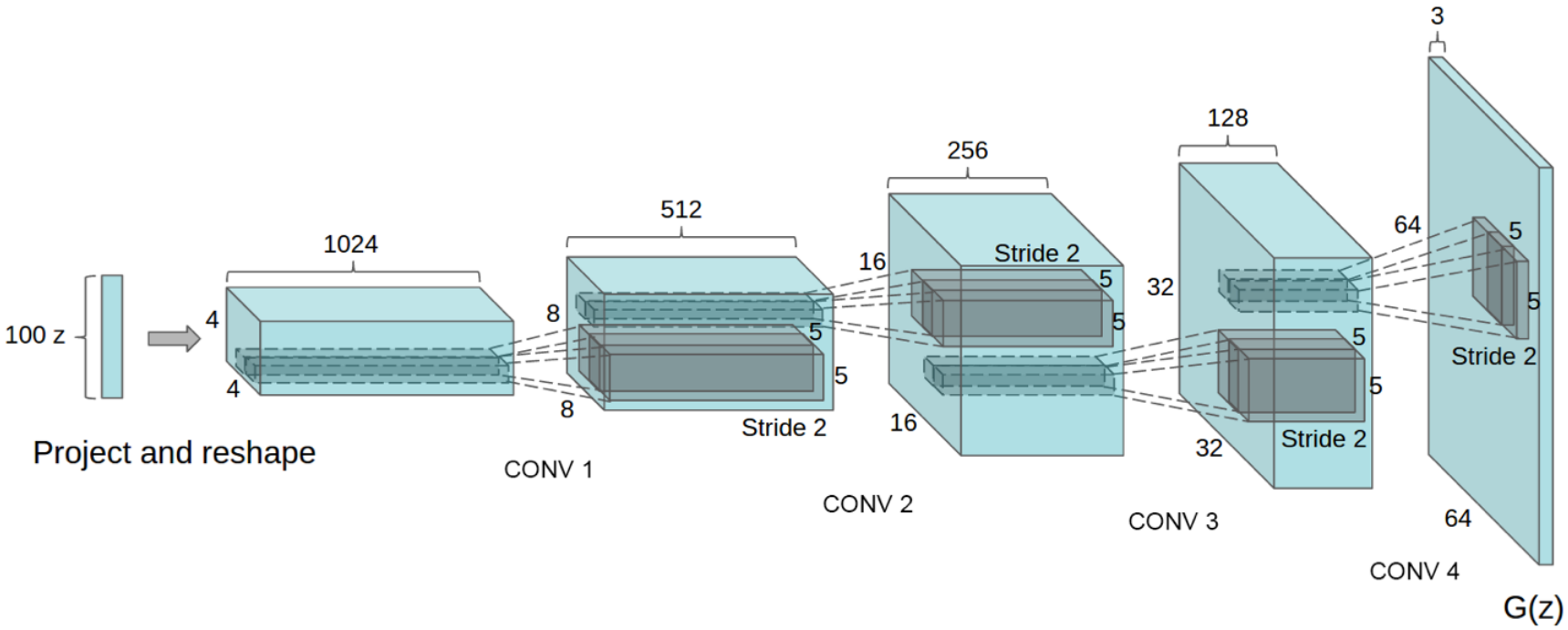


Strided convolutions



Fractional-strided convolutions

Part I: 4. DCGAN



The architecture of generator

Part I: 4. DCGAN

```python
class _netG(nn.Module):
    def __init__(self, ngpu):
        super(_netG, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d(nz, 1024, 4, 1, 0, bias=False),
            nn.BatchNorm2d(1024),
            nn.ReLU(True),
            # state size. 1024 x 4 x 4
            nn.ConvTranspose2d(1024, 512, 4, 2, 1, bias=False),
            nn.BatchNorm2d(512),
            nn.ReLU(True),
            # state size. 512 x 8 x 8
            nn.ConvTranspose2d(512, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(True),
            # state size. 256 x 16 x 16
            nn.ConvTranspose2d(256, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128),
            nn.ReLU(True),
            # state size. 128 x 32 x 32
            nn.ConvTranspose2d(128, 3, 4, 2, 1, bias=False),
            nn.Tanh()
            # state size. 3 x 64 x 64
        )
```

Generator Network (Pytorch)

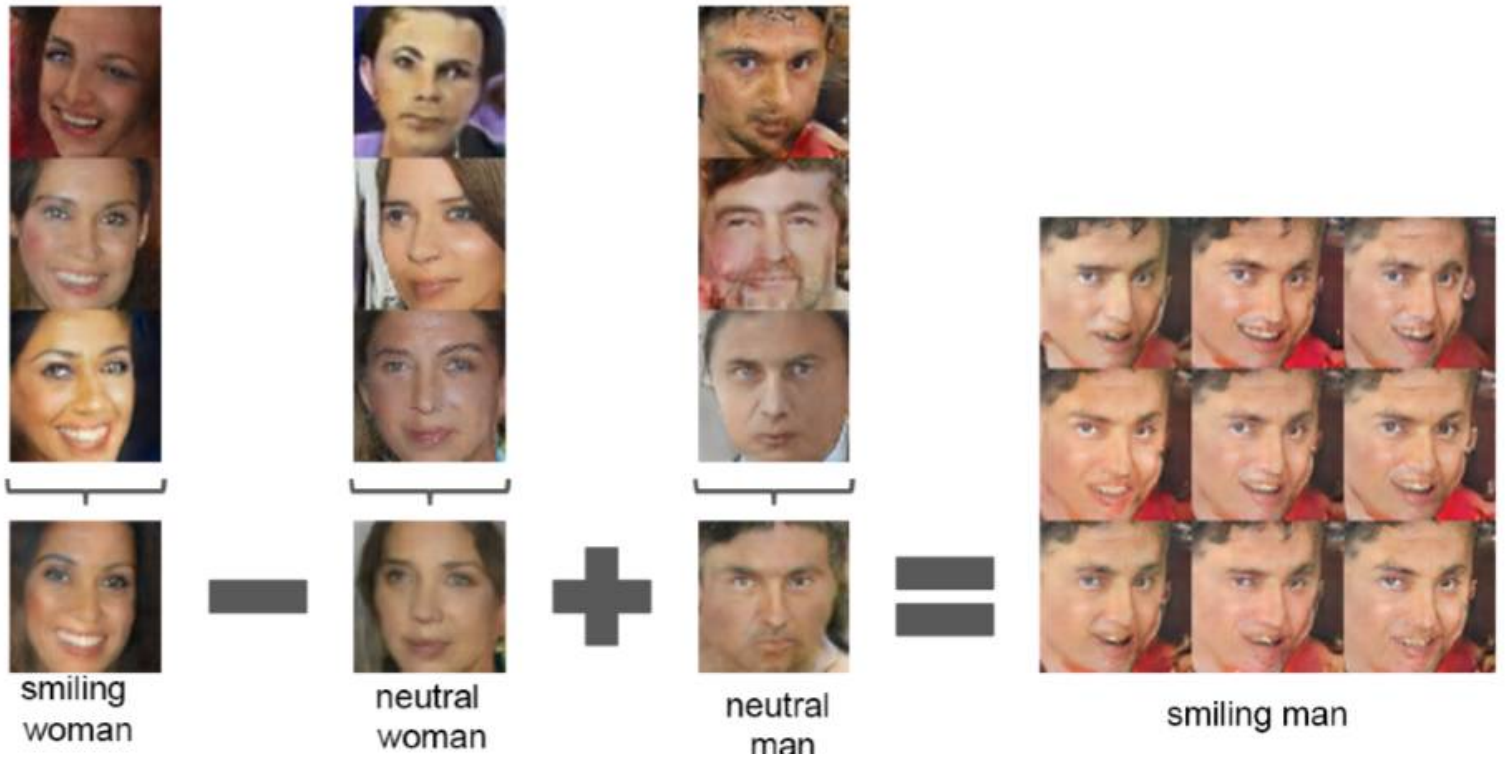
```python
class _netD(nn.Module):
    def __init__(self, ngpu):
        super(_netD, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is 3 x 64 x 64
            nn.Conv2d(3, 128, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. 128 x 32 x 32
            nn.Conv2d(128, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. 256 x 16 x 16
            nn.Conv2d(256, 512, 4, 2, 1, bias=False),
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. 512 x 8 x 8
            nn.Conv2d(512, 1024, 4, 2, 1, bias=False),
            nn.BatchNorm2d(1024),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. 1024 x 4 x 4
            nn.Conv2d(1024, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )
```

Discriminator Network (Pytorch)

Part I: 4. DCGAN

DCGANs for LSUN Bedrooms

Part I: 4. DCGAN



Vector arithmetic

Part I: 5. WGAN Preliminaries

Towards Principled Methods for Training Generative Adversarial Networks: theory explaining the unstable behaviour of GAN training.

Questions

- Why do updates get worse as the discriminator gets better? Both in the original and the new cost function.
- Why is GAN training massively unstable?
- Is the new cost function following a similar divergence to the JSD? If so, what are its properties?
- Is there a way to avoid some of these issues?

Part I: 5. WGAN Preliminaries

第一种原始GAN形式的问题：判别器越好，生成器梯度消失越严重

- **角度一：从生成器的等价损失函数切入**

$$L(D^*, g_\theta) = 2JSD(\mathbb{P}_r \| \mathbb{P}_g) - 2\log 2$$

在最优判别器的下，把原始GAN定义的生成器损失函数等价变换为最小化真实分布P_r与生成分布P_g之间的JS散度。

存在的问题：如果真实分布P_r与生成分布P_g之间没有重叠的话，**JS散度为固定常数log2，最终导致生成器的梯度（近似）为0，梯度消失。**

[6] https://zhuanlan.zhihu.com/p/25071913

Part I: 5. WGAN Preliminaries

第一种原始GAN形式的问题：判别器越好，生成器梯度消失越严重

- **角度二：公式定理论证**

✓首先, P_r与P_g之间几乎不可能有不可忽略的重叠，所以无论它们之间的“缝隙”多狭小，都肯定存在一个最优分割曲面把它们隔开，最多就是在那些可忽略的重叠处隔不开而已。

✓由于判别器作为一个神经网络可以无限拟合这个分隔曲面，所以存在一个最优判别器，对几乎所有真实样本给出概率1，对几乎所有生成样本给出概率0，而那些隔不开的部分就是难以被最优判别器分类的样本，但是它们的测度为0，可忽略。

✓最优判别器在真实分布和生成分布的支撑集上给出的概率都是常数（1和0），导致生成器的loss梯度为0，梯度消失。

Part I: 5. WGAN Preliminaries

第一种原始GAN形式的问题：判别器越好，生成器梯度消失越严重

- 判别器训练得太好，生成器梯度消失，生成器Loss降不下去；
- 判别器训练得不好，生成器梯度不准，四处乱跑；
- 只有判别器训练得不好不坏才行，但是这个火候又很难把握，甚至在同一轮训练的前后不同阶段这个火候都可能不一样，所以GAN才那么难训练。

Part I: 5. WGAN Preliminaries

第二种原始GAN形式的问题：最小化第二种生成器loss函数，会等价于最小化一个不合理的距离衡量 $KL(P_g||P_r) - 2JS(P_r||P_g)$ **。**

- 导致两个问题，一是梯度不稳定，二是mode collapse。
- 同时要最小化生成分布与真实分布的KL散度，却又要最大化两者的JS散度，在数值上则会导致梯度不稳定。
- 第一项KL divergence loss会导致生成器宁可多生成一些重复但是很“安全”的样本，也不愿意去生成多样性的样本，因而出现mode collapse。

Part I: 5. WGAN Preliminaries

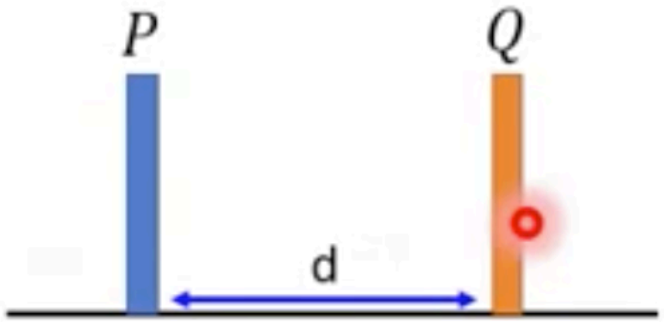
解决方案：

- 一是等价优化的距离衡量（KL散度、JS散度）不合理
- 二是生成器随机初始化后的生成分布很难与真实分布有不可忽略的重叠。对生成样本和真实样本加噪声，直观上说，使得原本的两个低维流形“弥散”到整个高维空间，强行让它们产生不可忽略的重叠。而一旦存在重叠，JS散度就能真正发挥作用，此时如果两个分布越靠近，它们“弥散”出来的部分重叠得越多，JS散度也会越小而不会一直是一个常数，于是（在第一种原始GAN形式下）梯度消失的问题就解决了。
- 在训练过程中，可以对所加的噪声进行退火（annealing），慢慢减小其方差，到后面两个低维流形“本体”都已经有重叠时，就算把噪声完全拿掉，JS散度也能照样发挥作用，继续产生有意义的梯度把两个低维流形拉近，直到它们接近完全重合。

Part I: 6. WGAN

Earth-Mover Distance

- Considering one distribution P as a pile of earth, and another distribution Q as the target
- The average distance the earth mover has to move the earth.

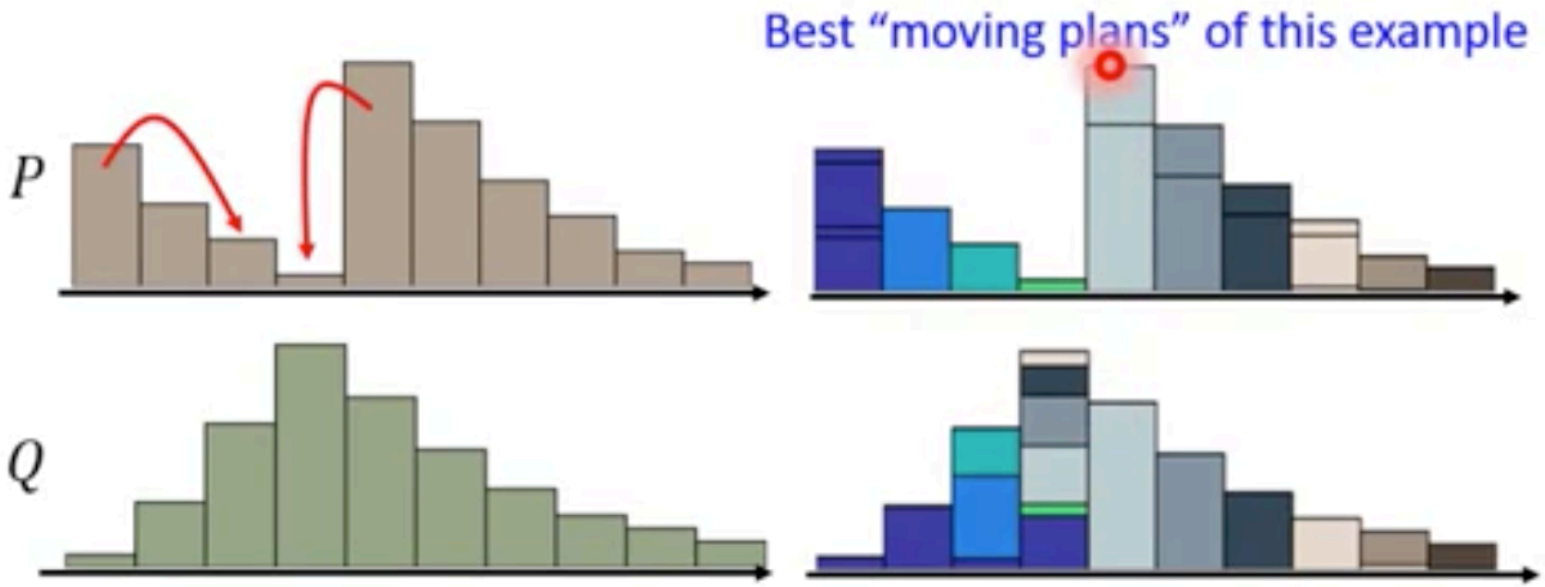


$$W(P,Q) = d$$

[7] Arjovsky M, Chintala S, Bottou L. Wasserstein gan[J]. arXiv preprint arXiv:1701.07875, 2017.

Part I: 6. WGAN

Earth-Mover Distance

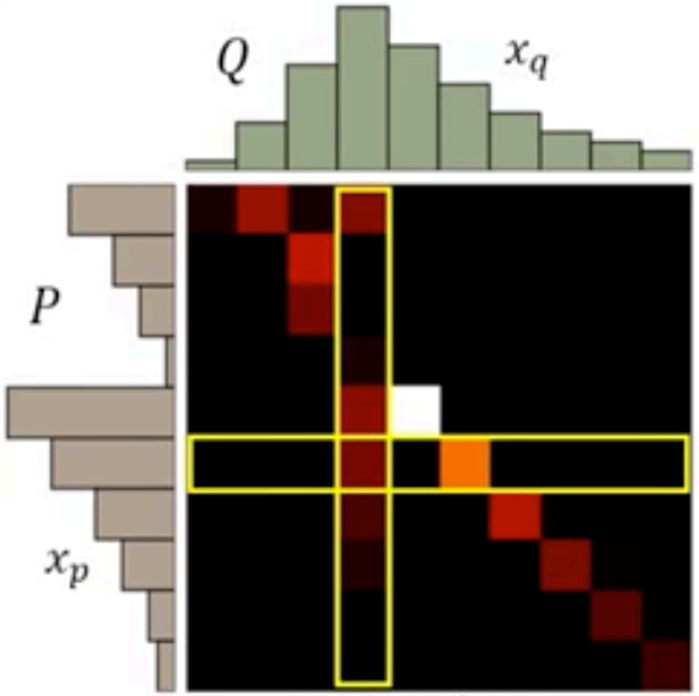


There many possible “moving plans”.

Using the “moving plan” with the smallest average distance to define the earth mover’s distance.

Part I: 6. WGAN

Earth-Mover Distance



moving plan γ
All possible plan Π

A "moving plan" is a matrix

The value of the element is the amount of earth from one position to another.

Average distance of a plan γ:

$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

Earth Mover's Distance:

$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma)$$

The best plan

Part I: 6. WGAN

Earth-Mover's Distance

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right] , \tag{1}$$

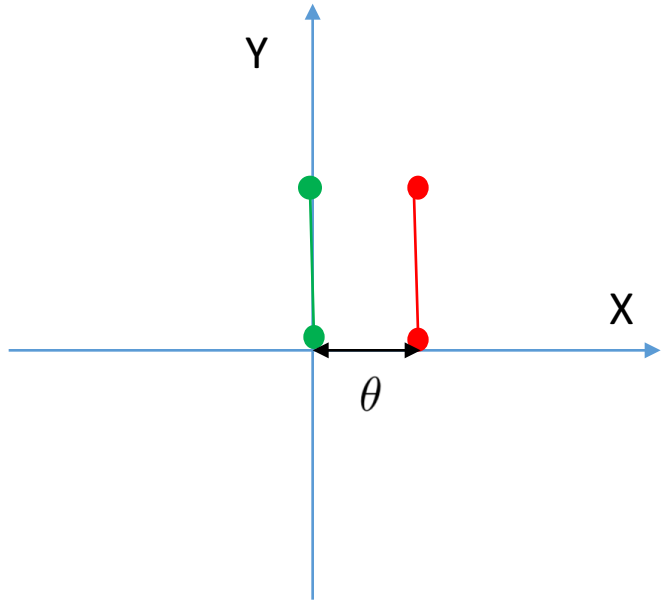
where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

Part I: 6. WGAN

Example 1 (Learning parallel lines)**.** Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$

When $\theta_t \to 0$, the sequence $(\mathbb{P}_{\theta_t})_{t \in \mathbb{N}}$ converges to $\mathbb{P}_0$ under the EM distance, but does not converge at all under either the JS, KL, reverse KL, or TV divergences. Figure 1 illustrates this for the case of the EM and JS distances.



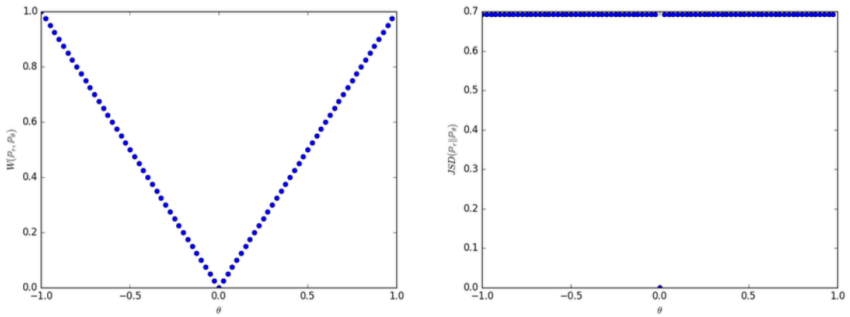
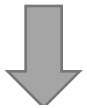


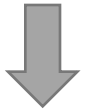
Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of θ when ρ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

Part I: 6. WGAN

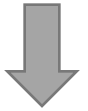
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right]$$



$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$



$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$



$$V(G,D) = \max_{D \in 1-Lipschitz} \{E_{x\sim P_{data}}[D(x)] - E_{x\sim P_G}[D(x)]\}$$

K-Lipschitz : $||f(x_1) - f(x_2)|| \leq K||x_1 - x_2||$

Weight Clipping [Martin Arjovsky, et al., arXiv, 2017]

Force the parameters w between c and -c

After parameter update, if w > c, w = c;

if w < -c, w = -c

Part I: 6. WGAN

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Part I: 6. WGAN

相比原始GAN的算法：

- 判别器最后一层去掉sigmoid；
- 生成器和判别器的loss不取log；
- 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数c (weight clipping)；
- 不要用基于动量的优化算法（包括momentum和Adam），推荐RMSProp、SGD等优化器。

Part I: 6. WGAN

总结：

- 彻底解决GAN训练不稳定的问题，不再需要小心平衡生成器和判别器的训练程度；
- 基本解决了mode collapse的问题，确保了生成样本的多样性；
- 训练过程中有一个像交叉熵、准确率这样的数值来指示训练的进程，这个数值越小代表GAN训练得越好，代表生成器产生的图像质量越高；
- 以上一切好处不需要精心设计的网络架构，最简单的多层全连接网络就可以做到。

Part I: 7. Improved WGAN

Weight Clipping存在的两个问题：

1) 判别器会非常倾向于学习一个简单的映射函数；

2) 梯度消失或者梯度爆炸；

- **Gradient Penalty**

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}\left[D(\tilde{\boldsymbol{x}})\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\left[D(\boldsymbol{x})\right]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[\left(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1\right)^2\right]}_{\text{Our gradient penalty}}.$$

[8] Gulrajani I, Ahmed F, Arjovsky M, et al. Improved training of wasserstein gans[C]//Advances in neural information processing systems. 2017: 5767-5777.

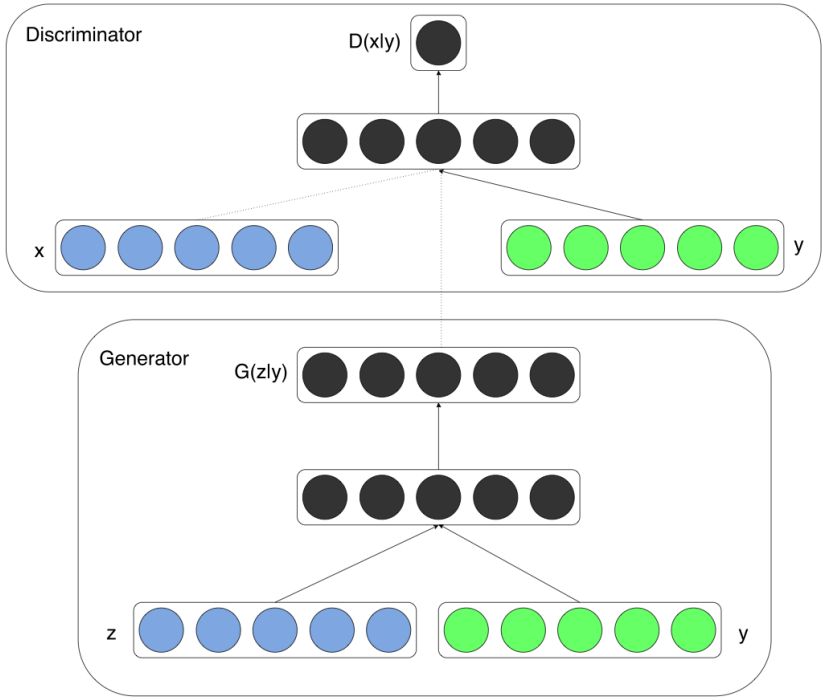
Part I: 7. Improved WGAN

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

Part I: 8. CGAN



Cost Function of GAN

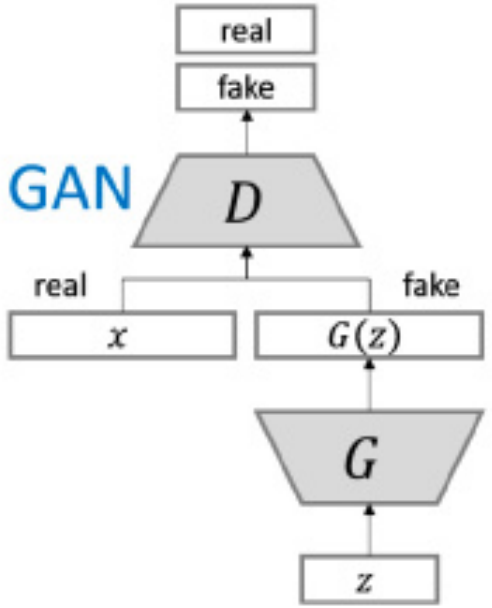
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

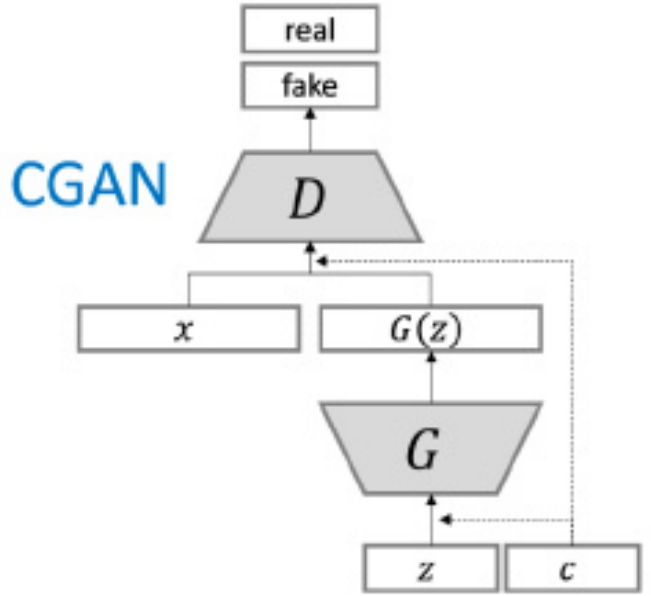
Cost Function of CGAN

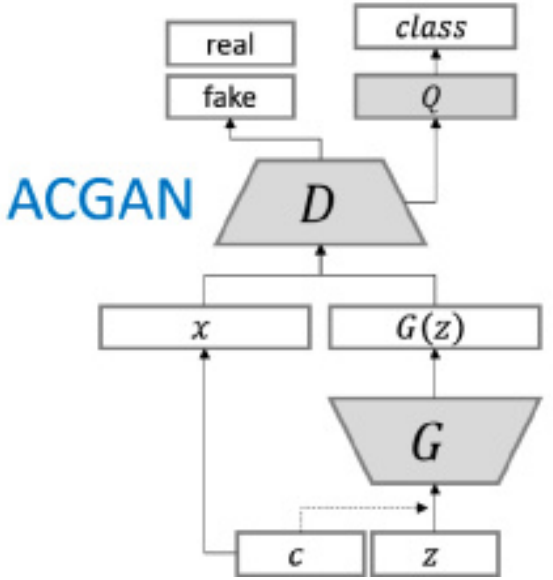
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{z}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

[9] Mirza M, Osindero S. Conditional generative adversarial nets[J]. arXiv preprint arXiv:1411.1784, 2014.

Part I: 9. ACGAN







Part I: 9. ACGAN

Loss Function: D is trained to maximize Ls + Lc, while G is trained to maximize Lc-Ls 。

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

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]$$

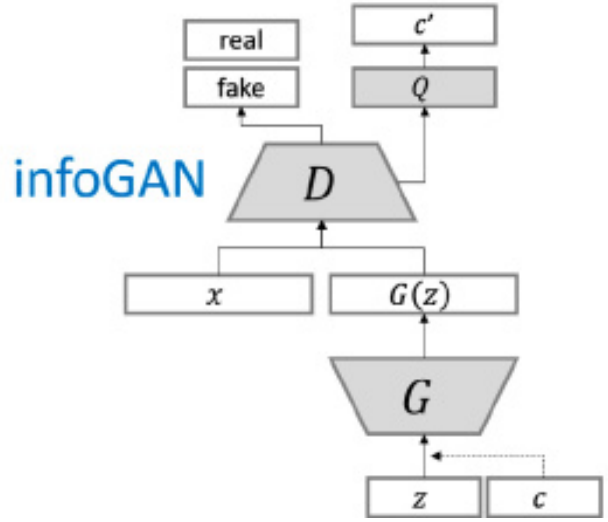
$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})]$$

[10] Odena A, Olah C, Shlens J. Conditional image synthesis with auxiliary classifier gans[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 2642-2651.

Part I: 10. InfoGAN

Basic Idea

- Bring the concept of latent code c
- An information-theoretic extension to the GAN



Information-regularized minimax game

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

Lower bound of cost function

$$\min_{G,Q} \max_D V_{\text{InfoGAN}}(D,G,Q) = V(D,G) - \lambda L_I(G,Q)$$

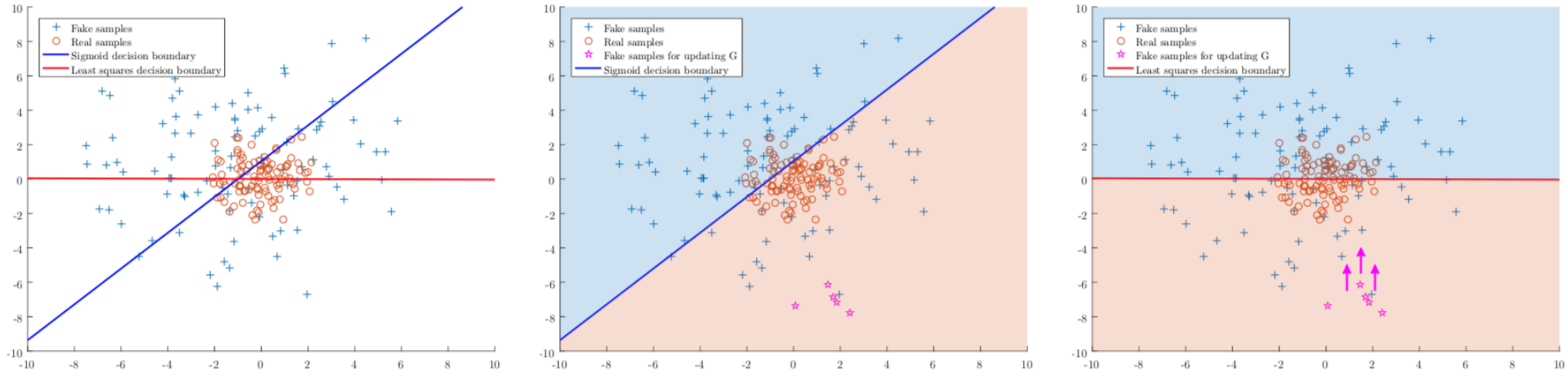
[11] Chen X, Duan Y, Houthooft R, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets[C]//Advances in neural information processing systems. 2016: 2172-2180.

Part I: 11. Improved Techniques for Training GANs

- **Feature Matching**
- **MiniBatch Discrimination**
- **Historical Averaging**
- **One-side Label Smooth**
- **Virtual Batch Normalization**

[12] Salimans T, Goodfellow I, Zaremba W, et al. Improved techniques for training gans[C]//Advances in neural information processing systems. 2016: 2234-2242.

Part I: 12. LSGAN



(a) (b) (c)

Figure 1. Illustration of different behaviors of two loss functions. (a): Decision boundaries of two loss functions. Note that the decision boundary should go across the real data distribution for a successful GANs learning. Otherwise, the learning process is saturated. (b): Decision boundary of the sigmoid cross entropy loss function. The orange area is the side of real samples and the blue area is the side of fake samples. It gets very small errors for the fake samples (in magenta) when updating G as they are on the correct side of the decision boundary. (c): Decision boundary of the least squares loss function. It penalizes the fake samples (in magenta), and as a result, it forces the generator to generate samples toward decision boundary.

[13] Mao X, Li Q, Xie H, et al. Least squares generative adversarial networks[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 2794-2802.

Part I: 12. LSGAN

- **Loss Function:**

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right]$$

- **In Practice:**

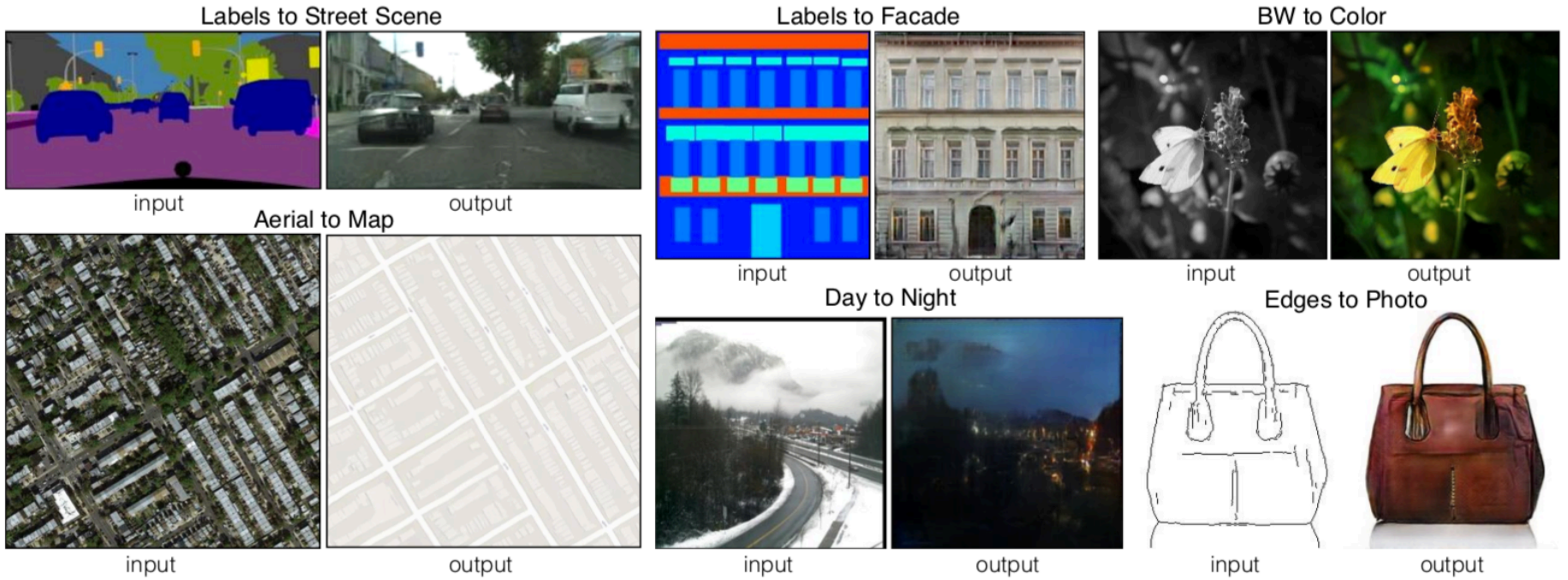
$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-1)^2\right]$$

CONTENTS

Part II: 1. Image-to-Image Translation

- **The image-to-image translation task fundamentally learns a mapping from a source image to a target image that conforms to the input layout.**



Part II: 1. Image-to-Image Translation

- **Pix2Pix (Supervised Image-to-Image Translation)**
- **CycleGAN (Unsupervised Image-to-Image Translation)**
- **StarGAN (Unsupervised Multi-domain Image-to-Image Translation)**

Part II: 2. Pix2Pix

- **Network Architecture: Conditional GANs**

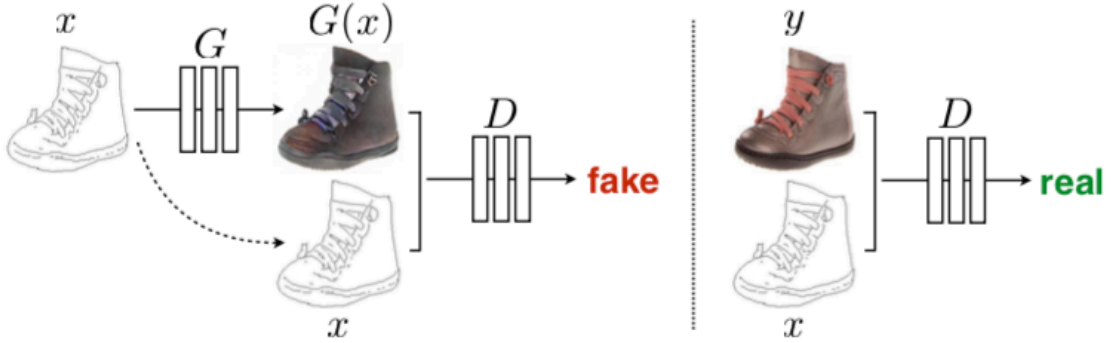


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

[14] Isola P, Zhu J Y, Zhou T, et al. Image-to-image translation with conditional adversarial networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1125-1134.

Part II: 2. Pix2Pix

- **Generator Structure: U-Net**

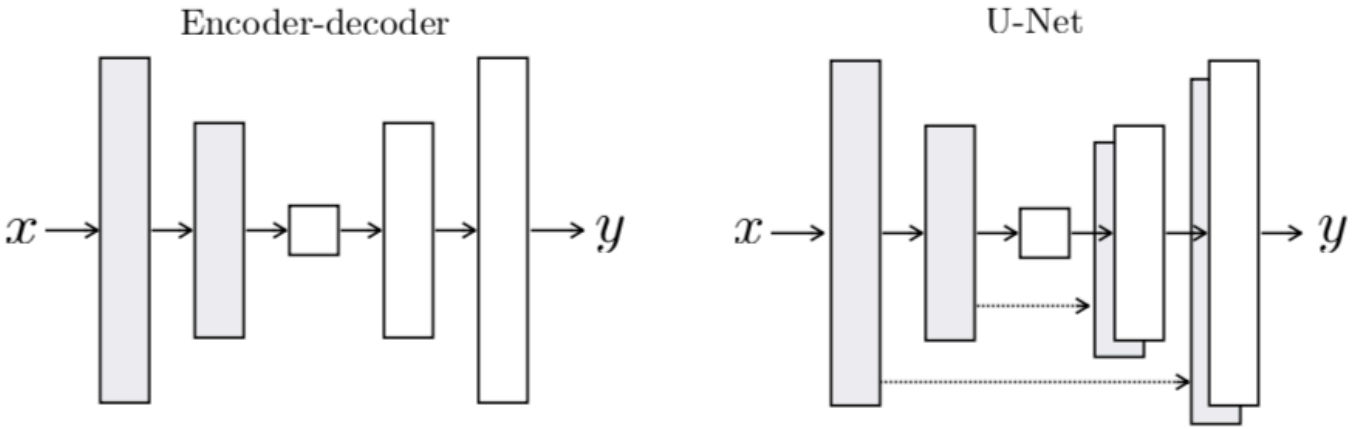
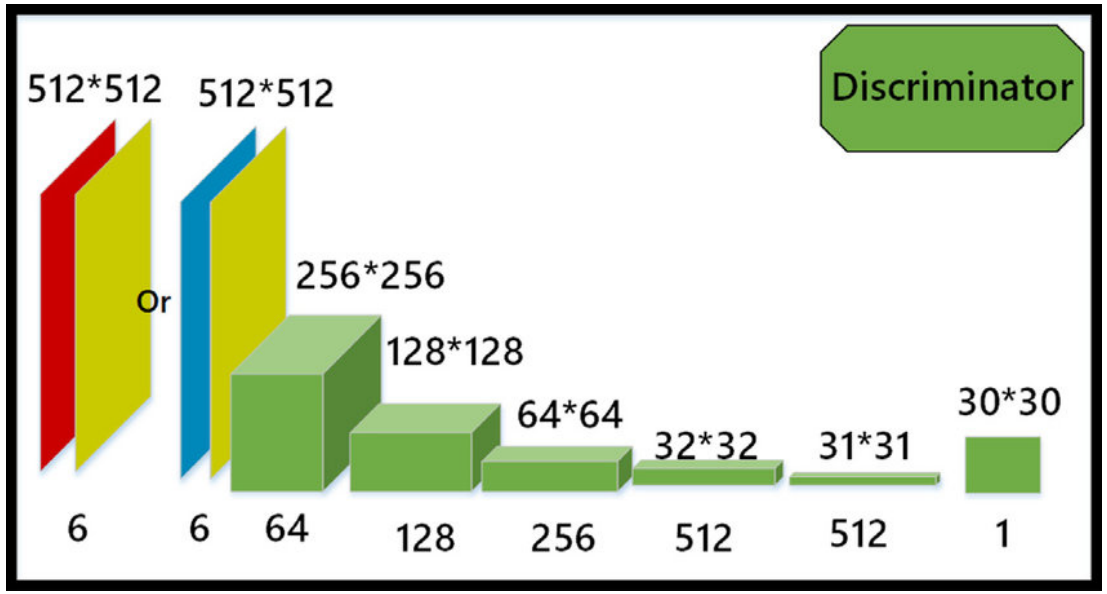
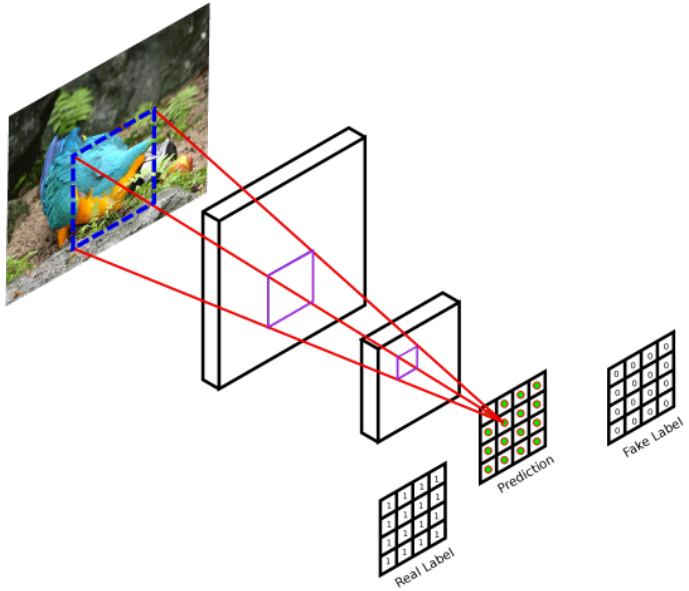


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Part II: 2. Pix2Pix

- **Discriminator Structure: PatchGAN**





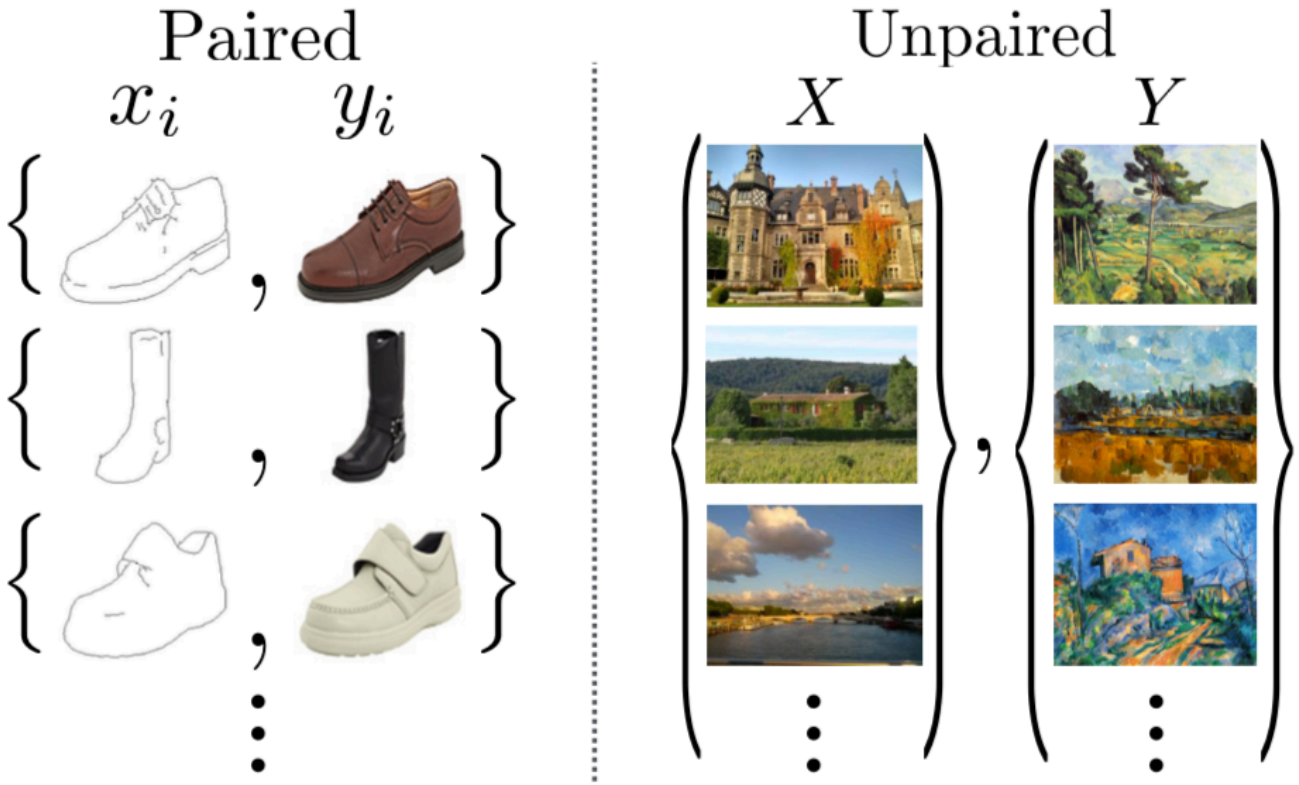
Part II: 2. Pix2Pix

- **Loss Function:** Adversarial Loss + L1 Reconstruction Loss

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

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

Part II: 3. CycleGAN



[15] Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2223-2232.

Part II: 3. CycleGAN

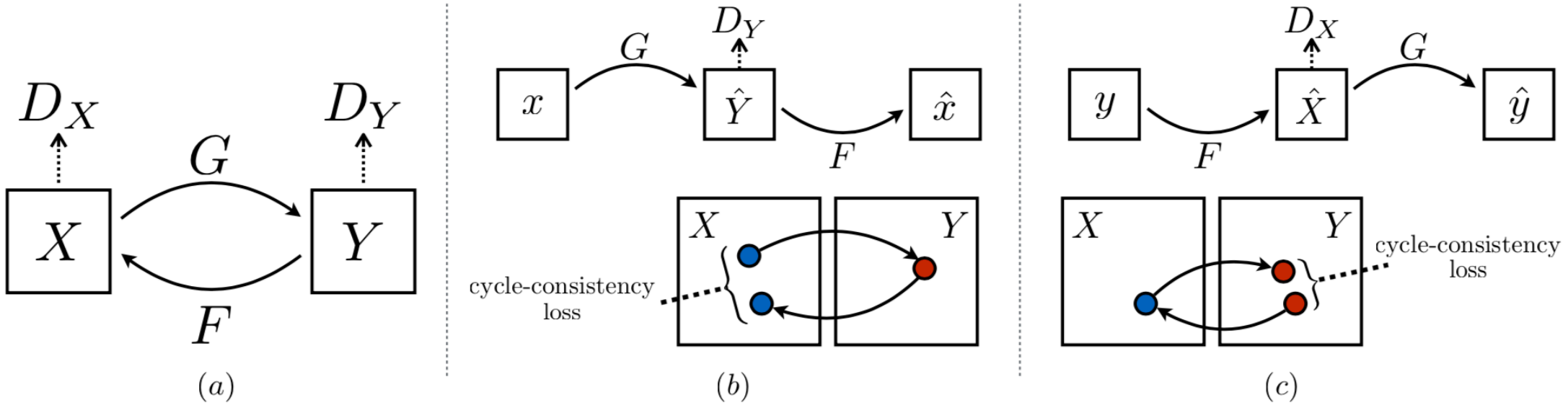


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Part II: 3. CycleGAN

- **Adversarial Loss**

$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$$

$$\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$$

- **Cycle Consistency Loss**

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) =& \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

- **Full Objective: Adversarial Loss + Cycle Consistency Loss**

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

Part II: 4. StarGAN

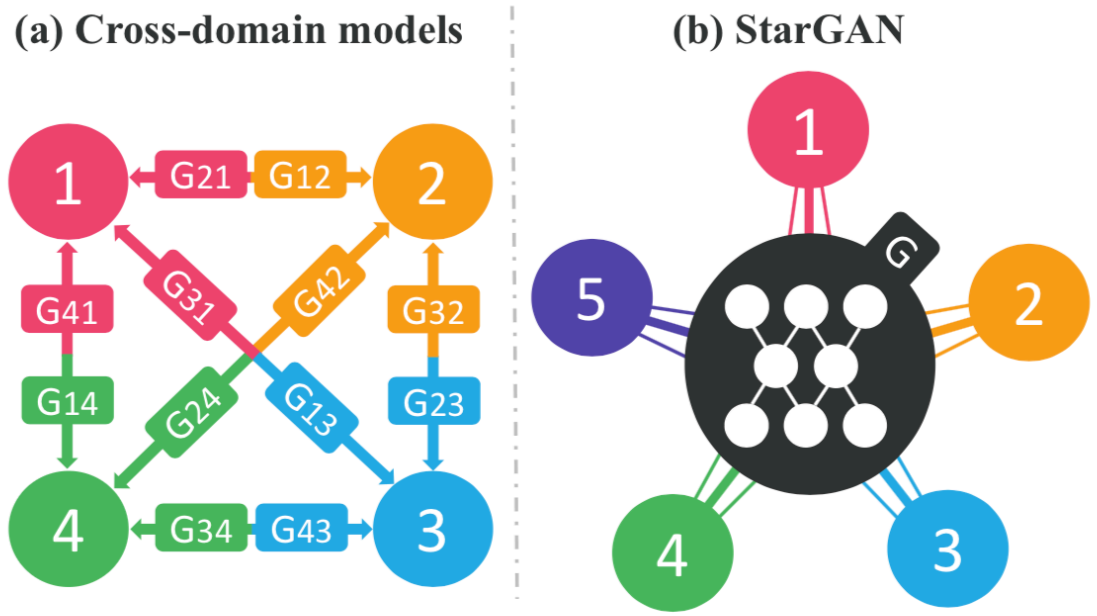


Figure 2. Comparison between cross-domain models and our proposed model, StarGAN. (a) To handle multiple domains, cross-domain models should be built for every pair of image domains. (b) StarGAN is capable of learning mappings among multiple domains using a single generator. The figure represents a star topology connecting multi-domains.

[16] Choi Y, Choi M, Kim M, et al. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 8789-8797.

Part II: 4. StarGAN

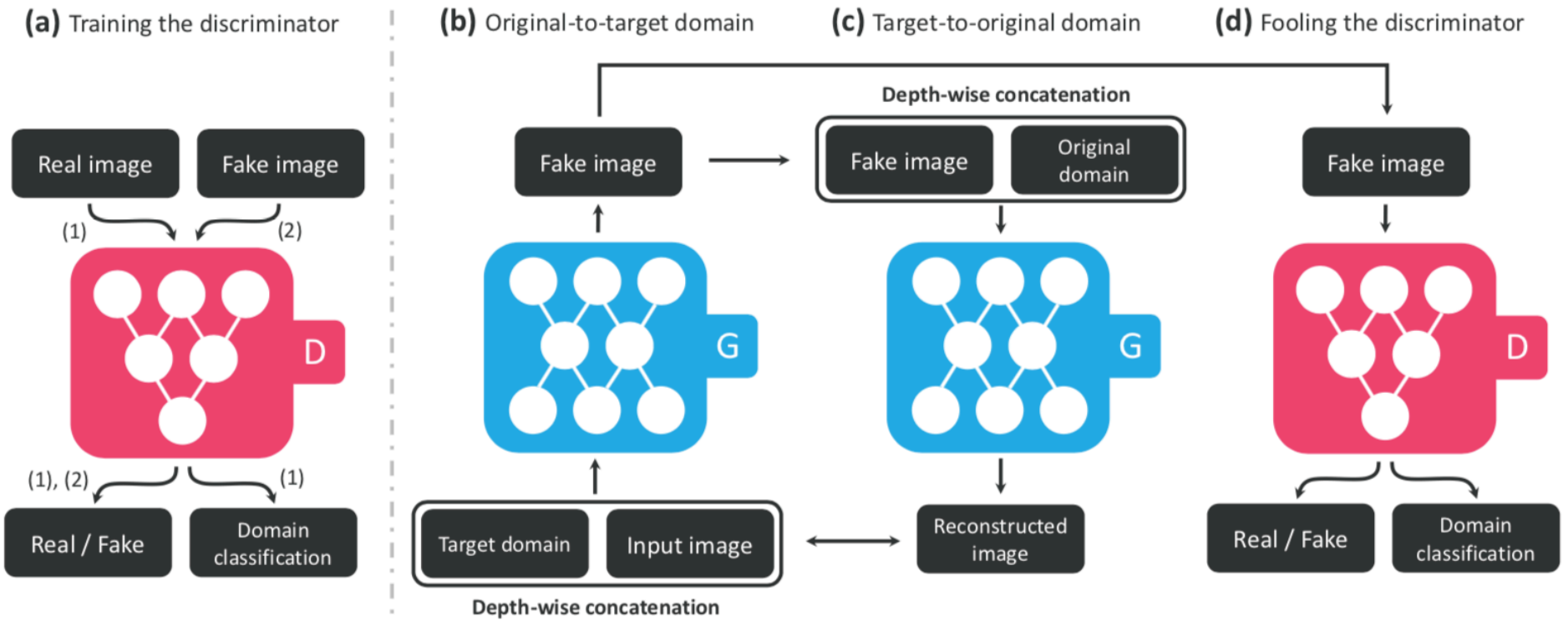


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator D and a generator G. **(a)** D learns to distinguish between real and fake images and classify the real images to its corresponding domain. **(b)** G takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image. **(c)** G tries to reconstruct the original image from the fake image given the original domain label. **(d)** G tries to generate images indistinguishable from real images and classifiable as target domain by D.

Part II: 4. StarGAN

- **Adversarial Loss**

$$\mathcal{L}_{adv} = \mathbb{E}_x\left[\log D_{src}(x)\right] + \mathbb{E}_{x,c}[\log\left(1 - D_{src}(G(x,c))\right)],$$

- **Reconstruction Loss**

$$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'}[||x - G(G(x,c),c')||_1],$$

- **Domain Classification Loss**

$$\mathcal{L}^r_{cls} = \mathbb{E}_{x,c'}[-\log D_{cls}(c'|x)],$$

$$\mathcal{L}^f_{cls} = \mathbb{E}_{x,c}[-\log D_{cls}(c|G(x,c))].$$

- **Full Objective**

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls}\,\mathcal{L}^r_{cls},$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls}\,\mathcal{L}^f_{cls} + \lambda_{rec}\,\mathcal{L}_{rec},$$

Part II: 5. Image Completion

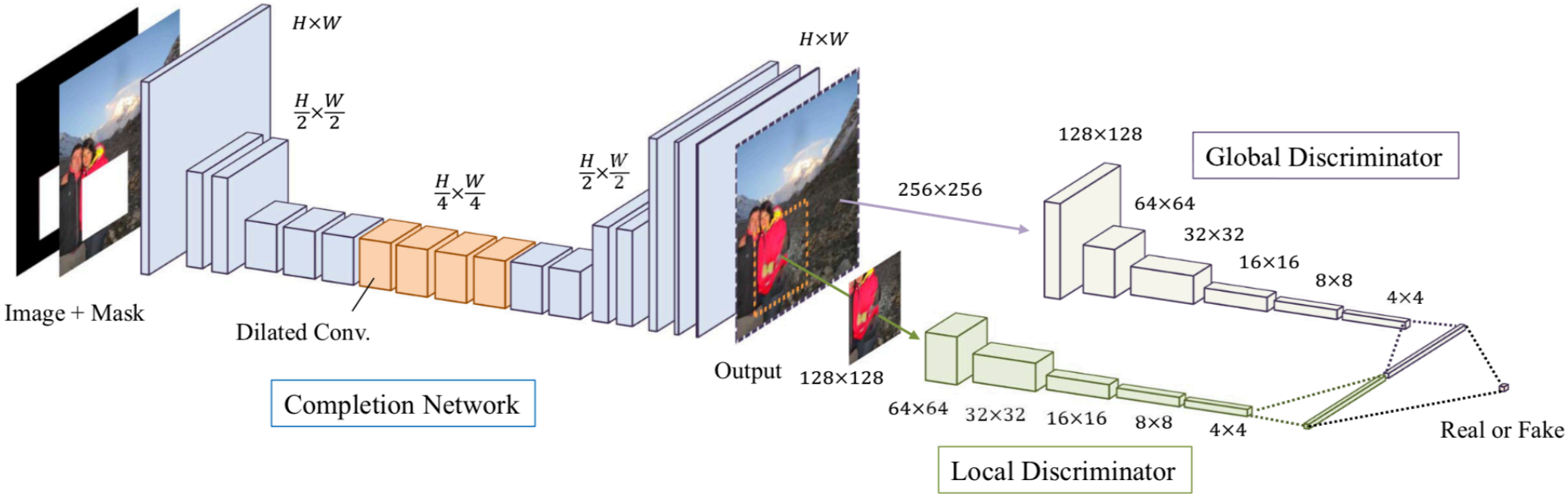
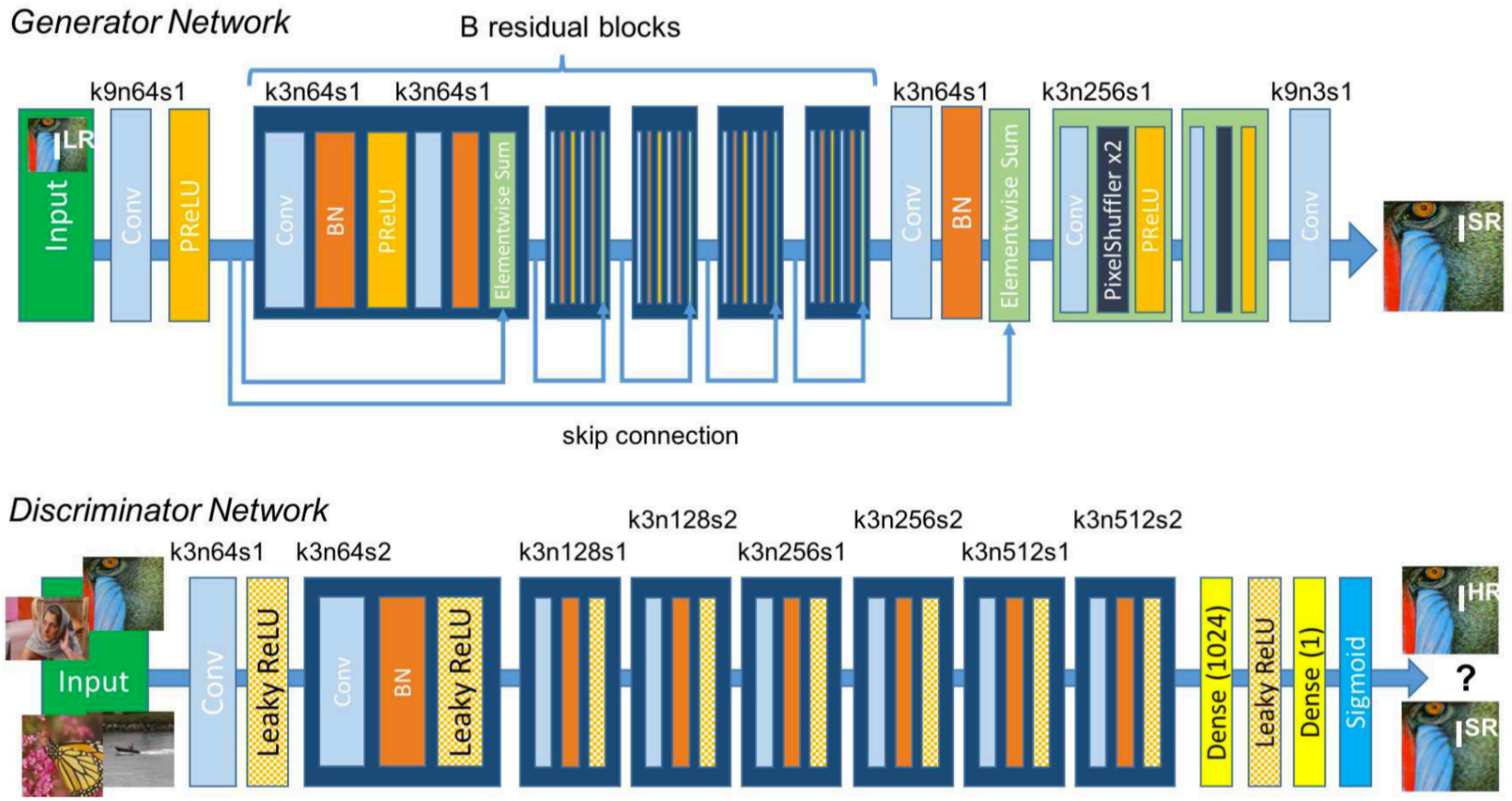


Fig. 2. Overview of our architecture for learning image completion. It consists of a completion network and two auxiliary context discriminator networks that are used only for training the completion network and are not used during the testing. The global discriminator network takes the entire image as input, while the local discriminator network takes only a small region around the completed area as input. Both discriminator networks are trained to determine if an image is real or completed by the completion network, while the completion network is trained to fool both discriminator networks.

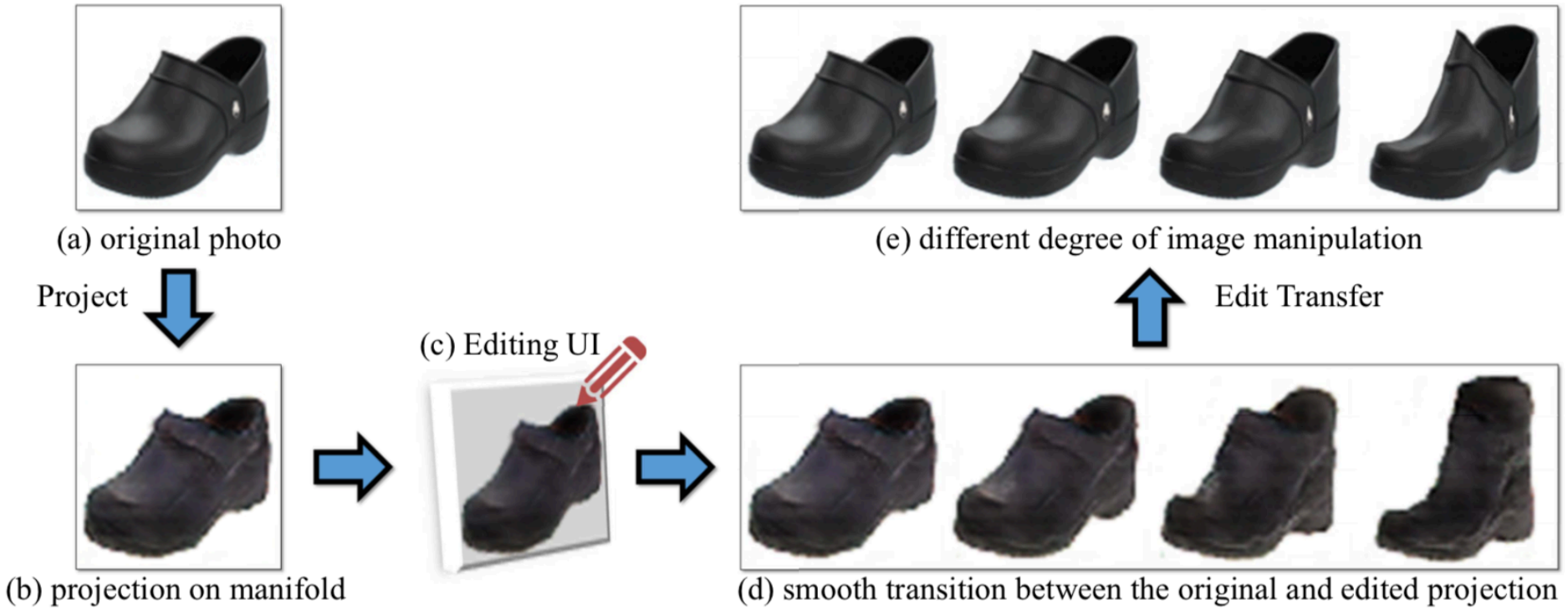
[17] Iizuka S, Simo-Serra E, Ishikawa H. Globally and locally consistent image completion[J]. ACM Transactions on Graphics (ToG), 2017, 36(4): 107.

Part II: 6. Image Super-Resolution



[18] Ledig C, Theis L, Huszár F, et al. Photo-realistic single image super-resolution using a generative adversarial network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4681-4690.

Part II: 7. Image Manipulation



[19] Zhu J Y, Krähenbühl P, Shechtman E, et al. Generative visual manipulation on the natural image manifold[C]//European Conference on Computer Vision. Springer, Cham, 2016: 597-613.

Part II: 8. Text to Image Generation

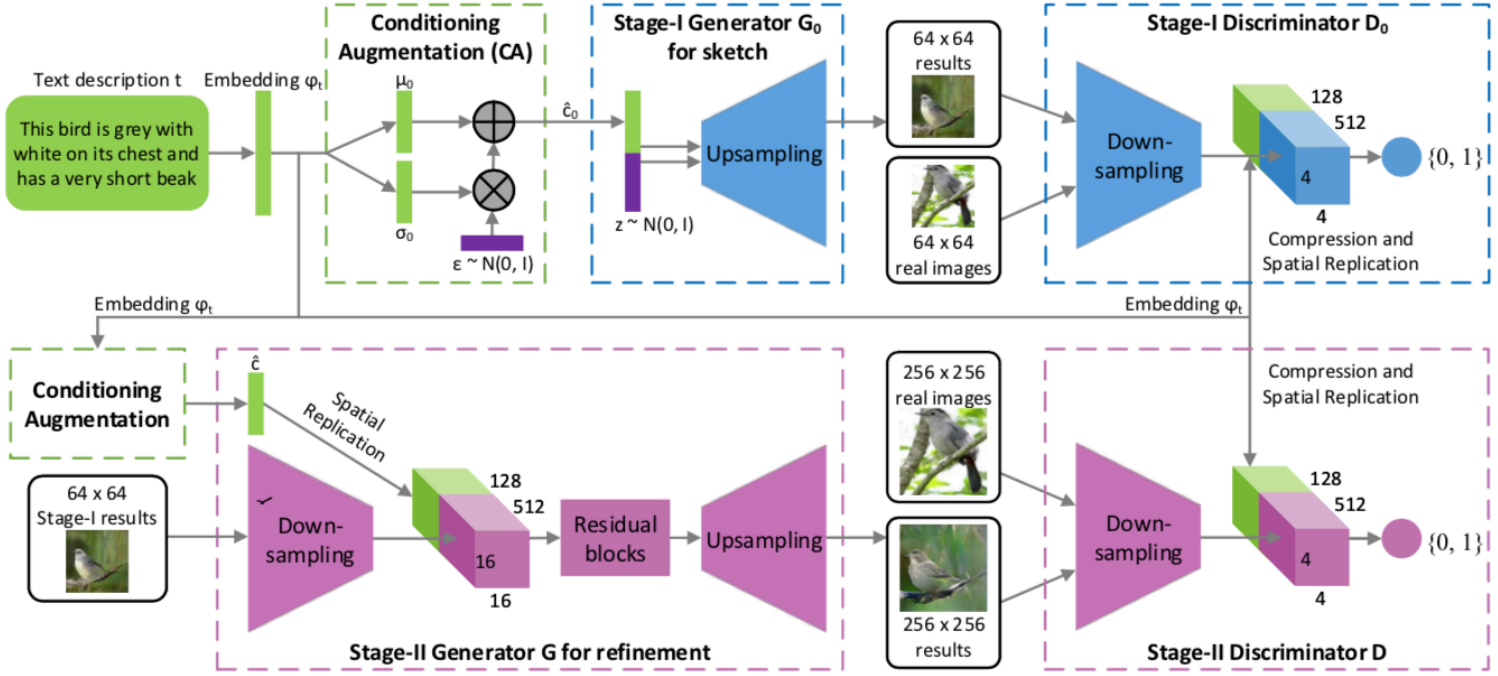


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

[20] Zhang H, Xu T, Li H, et al. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 5907-5915.

Part II: 9. Video Generation

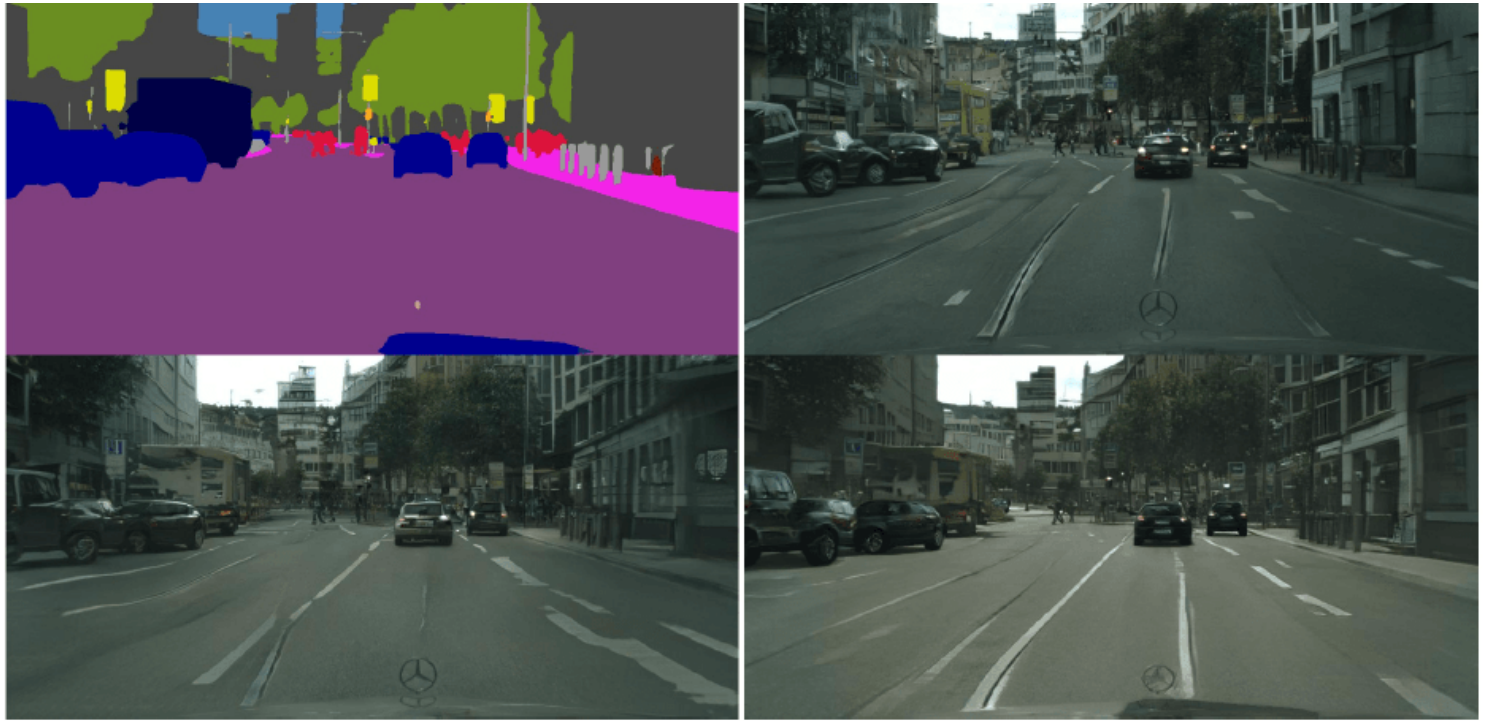


Figure 1: Generating a photorealistic video from an input segmentation map video on Cityscapes. Top left: input. Top right: `pix2pixHD`. Bottom left: `COVST`. Bottom right: `vid2vid` (ours). *The figure is best viewed with Acrobat Reader. Click the image to play the video clip.*

[21] Wang T C, Liu M Y, Zhu J Y, et al. Video-to-video synthesis[J]. arXiv preprint arXiv:1808.06601, 2018.

Part II: 10. Others

- Domain Adaption
- Data Augmentation
- Adversarial Attack
- Metric Learning
- Face Aging / Face Synthesis
- Photo Cartoonization
- 3D
- Fashion Image Synthesis

.....

CONTENTS

Part III: 1. Evaluation Metrics

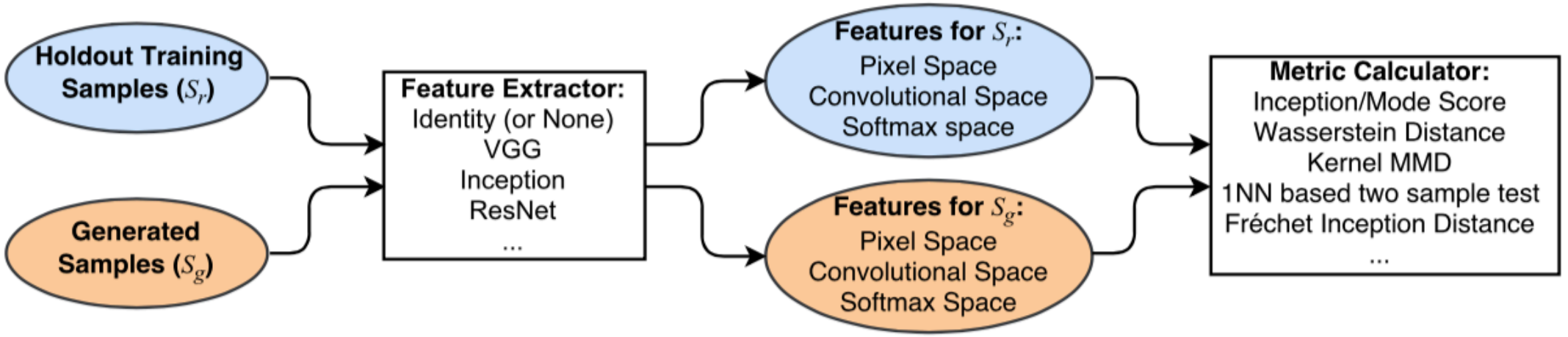


Figure 1: Typical sample based GAN evaluation methods.

[22] Xu Q, Huang G, Yuan Y, et al. An empirical study on evaluation metrics of generative adversarial networks[J]. arXiv preprint arXiv:1806.07755, 2018.

Part III: 1. Evaluation Metrics

- **Inception Score**

$$\mathrm{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[KL(p_{\mathcal{M}}(y|\mathbf{x})||p_{\mathcal{M}}(y))]}$$

- **Mode Score**

$$\mathrm{MS}(\mathbb{P}_g) = e^{\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[KL(p_{\mathcal{M}}(y|\mathbf{x})||p_{\mathcal{M}}(y))] - KL(p_{\mathcal{M}}(y)||p_{\mathcal{M}}(y^*))}$$

- **Kernel MMD**

$$\mathrm{MMD}^2(\mathbb{P}_r, \mathbb{P}_g) = \mathbb{E}_{\substack{\mathbf{x}_r, \mathbf{x}'_r \sim \mathbb{P}_r, \\ \mathbf{x}_g, \mathbf{x}'_g \sim \mathbb{P}_g}} \left[k(\mathbf{x}_r, \mathbf{x}'_r) - 2k(\mathbf{x}_r, \mathbf{x}_g) + k(\mathbf{x}_g, \mathbf{x}'_g) \right]$$

- **Wasserstein Distance**

$$\mathrm{WD}(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(\mathbf{x}^r, \mathbf{x}^g)\sim\gamma} \left[d(\mathbf{x}^r, \mathbf{x}^g) \right]$$

- **Fréchet Inception Distance**

$$\mathrm{FID}(\mathbb{P}_r, \mathbb{P}_g) = \|\mu_r - \mu_g\| + \mathrm{Tr}(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r\mathbf{C}_g)^{1/2})$$

- **1-Nearest Neighbor classifier**

CONTENTS

Homework 1 : Self-Learning

- **Stanford CS231n: Generative Model**
- https://www.youtube.com/watch?v=5WoItGTWV54
- https://blog.csdn.net/poulang5786/article/details/80766498

Homework 2 : Reading a Paper

- **LAPGAN**
- **PGGAN**
- **BigGAN**
- **StyleGAN**
- **Vid2Vid**
- **BicyleGAN**
- **AttnGAN**
- **Self-Attention GAN (SAGAN)**
- **Stacked GAN (SGAN)**

……

Homework 2: Or Running A Project

- **Pix2Pix**
- **Pix2PixHD**
- **CycleGAN**
- **StackGAN**
- **StackGAN++**
- **StarGAN**

对抗生成网络

Generative Adversarial Network

顾晓玲

guxl@hdu.edu.cn