



Generative Adversarial Nets

生成对抗网络 (GAN)

Name: 王若琪

Date: 20210529



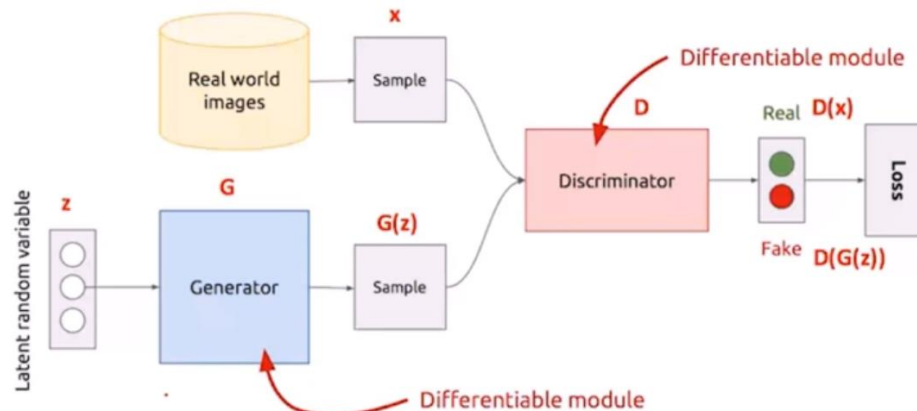
主要内容

- 1/ GAN 的直观概念
 - 2/ 算法流程
 - 3/ 内在原理
 - 4/ 效果与评价
 - 5/ CGAN/DCGAN/WGAN
-

GAN



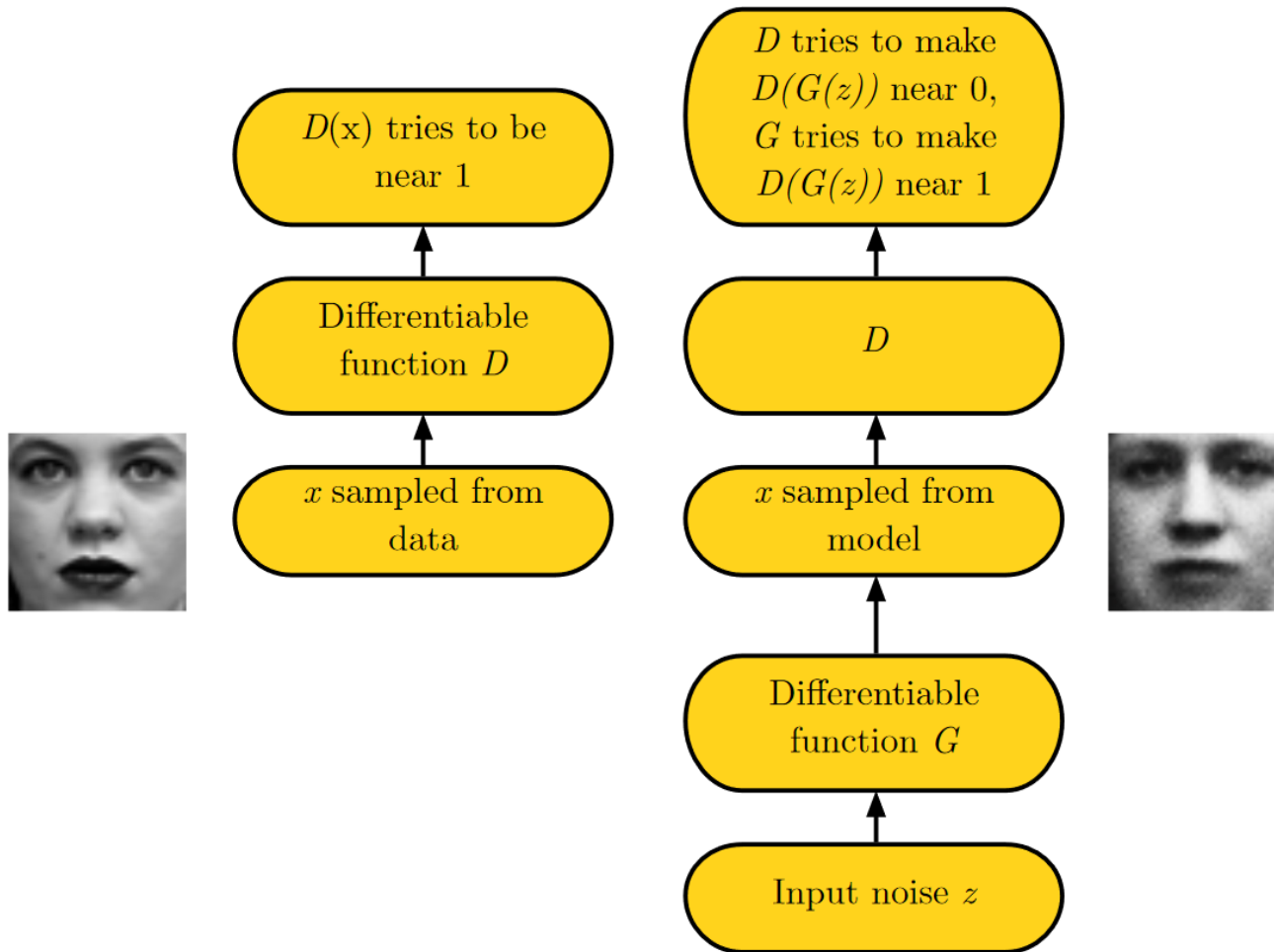
- Train two models: **a generative model G** that captures the data distribution, and **a discriminative model D** that estimates the probability that a sample came from the training data rather than G . The training procedure for G is to maximize the probability of D making a mistake.



GAN



GAN



■ Questions

目标函数如何设定？

如何进行训练？

■ 目标函数

$data \rightarrow$ 真实数据 (*groundtruth*)

$p_{data} \rightarrow$ 真实数据的分布

$z \rightarrow$ 噪音 (输入数据)

$p_z \rightarrow$ 原始噪音的分布

$p_g \rightarrow$ 经过生成器后的数据分布

$G() \rightarrow$ 生成映射函数

$D() \rightarrow$ 判别映射函数

Discriminative model的目标函数

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

Generative model的目标函数

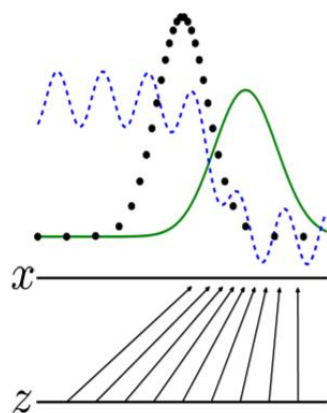
$$\min_G \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

整个的目标函数

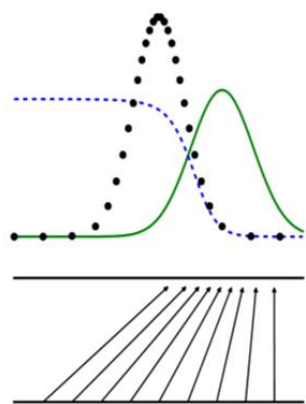
$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$



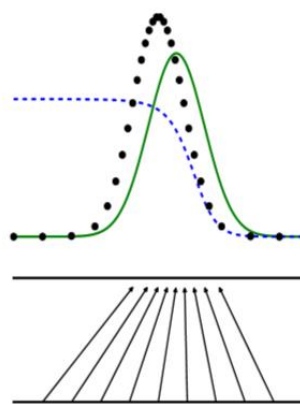
■ 如何优化目标函数? ■ ■



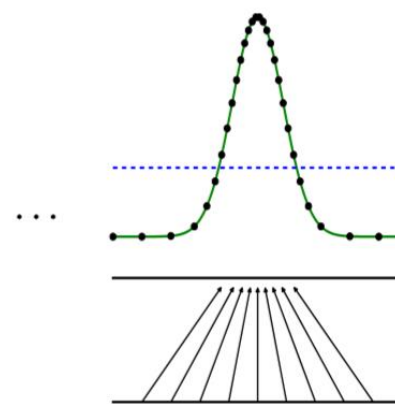
(a)



(b)



(c)



(d)

真实样本分布——黑线
生成样本分布——绿线
判别模型——蓝线



■ 实现的问题



如何用神经网络构造一个模拟分布 P_g ?

如何衡量 P_g 和 P_d 是否相似，并根据衡量结果去优化 P_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 $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

4.1 Global Optimality of $p_g = p_{\text{data}}$

We first consider the optimal discriminator D for any given generator G .

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

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \quad (2)$$

Proof. The training criterion for the discriminator D , given any generator G , is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(g(\mathbf{z}))) d\mathbf{z} \\ &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \end{aligned} \quad (3)$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \rightarrow a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $\text{Supp}(p_{\text{data}}) \cup \text{Supp}(p_g)$, concluding the proof. \square

Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|\mathbf{x})$, where Y indicates whether \mathbf{x} comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 1 can now be reformulated as:

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



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

Proof. For $p_g = p_{\text{data}}$, $D_G^*(\mathbf{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D_G^*(\mathbf{x}) = \frac{1}{2}$, 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

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

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


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

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:

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

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. \square



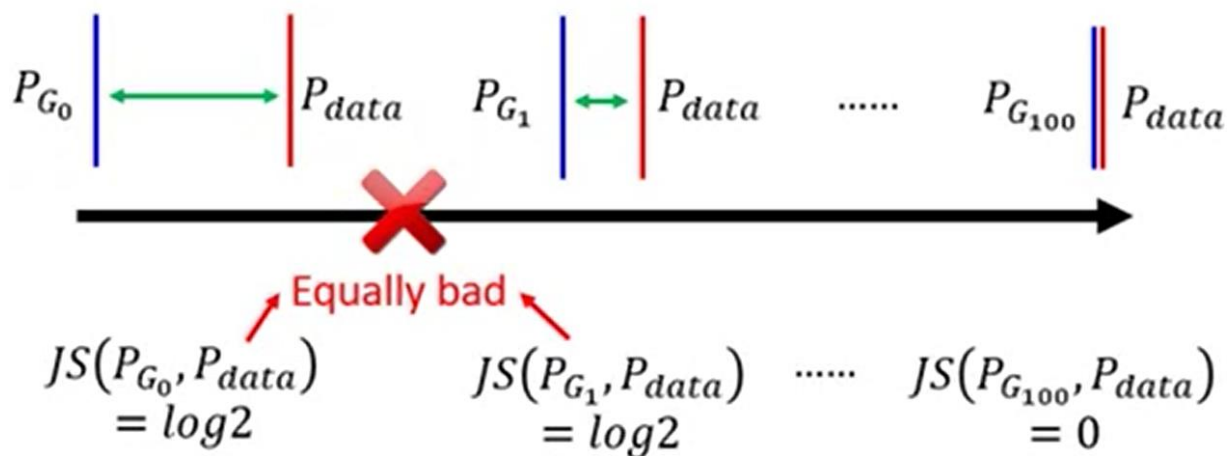


JS散度（Jensen-Shannon divergence）也称**JS距离**，是**KL散度**的一种变形。

$$JS(P||Q) = \frac{1}{2} KL(P(x)||\frac{P(x) + Q(x)}{2}) + \frac{1}{2} KL(Q(x)||\frac{P(x) + Q(x)}{2})$$

判别器执行JS时存在的问题

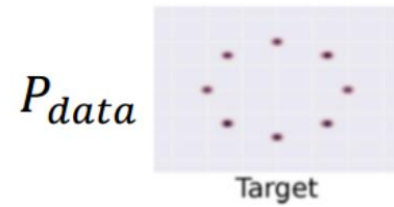
- 高维空间中，两个图像之间的重叠其实很少
- 就算有重叠，也很难被采样点描绘清楚
- 不重叠计算出来的结果就是 $\log 2$



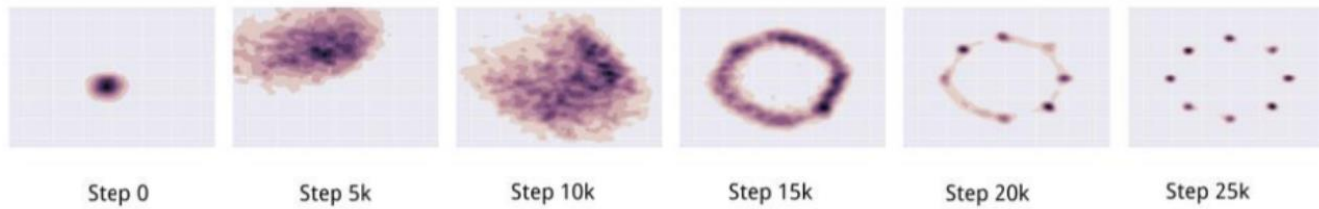
JS divergence is $\log 2$ if two distributions do not overlap.

Mode collapse

Mode Collapse



What we want ...



In reality ...



■ GAN的变种



自从GAN出世后，得到了广泛研究，先后几百篇不同的GANpaper横空出世，国外有大神整理了一个GAN zoo（GAN动物园），链接如下：

<https://github.com/hindupuravinash/the-gan-zoo>





CGAN

(Conditional Generative Adversarial Nets)

CGAN

(Conditional Generative Adversarial Nets)

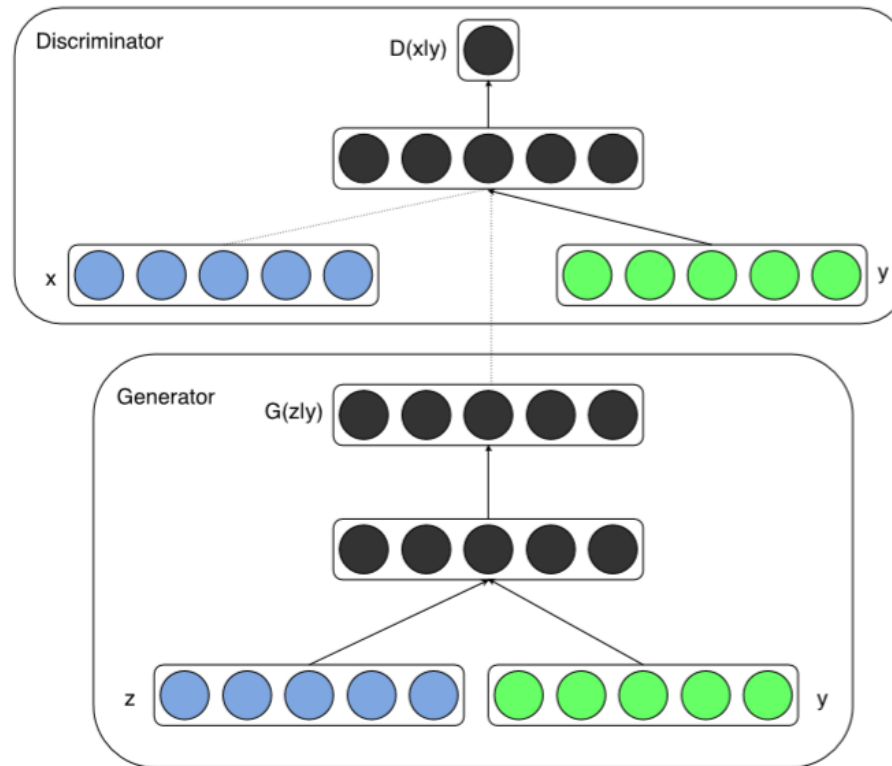
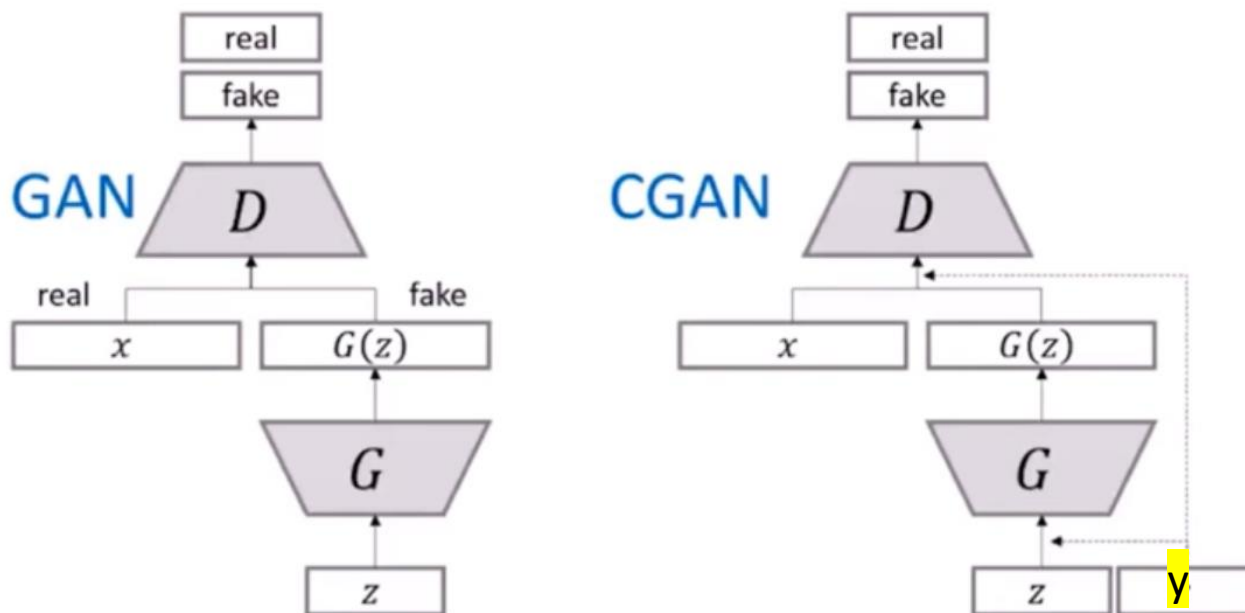


Figure 1: Conditional adversarial net

GAN与CGAN的区别



y 不一定是指定类，也可以是其他的一些条件或者约束，这是GAN生成太过自由的解决方案。

CGAN

(Conditional Generative Adversarial Nets)

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



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



CGAN



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

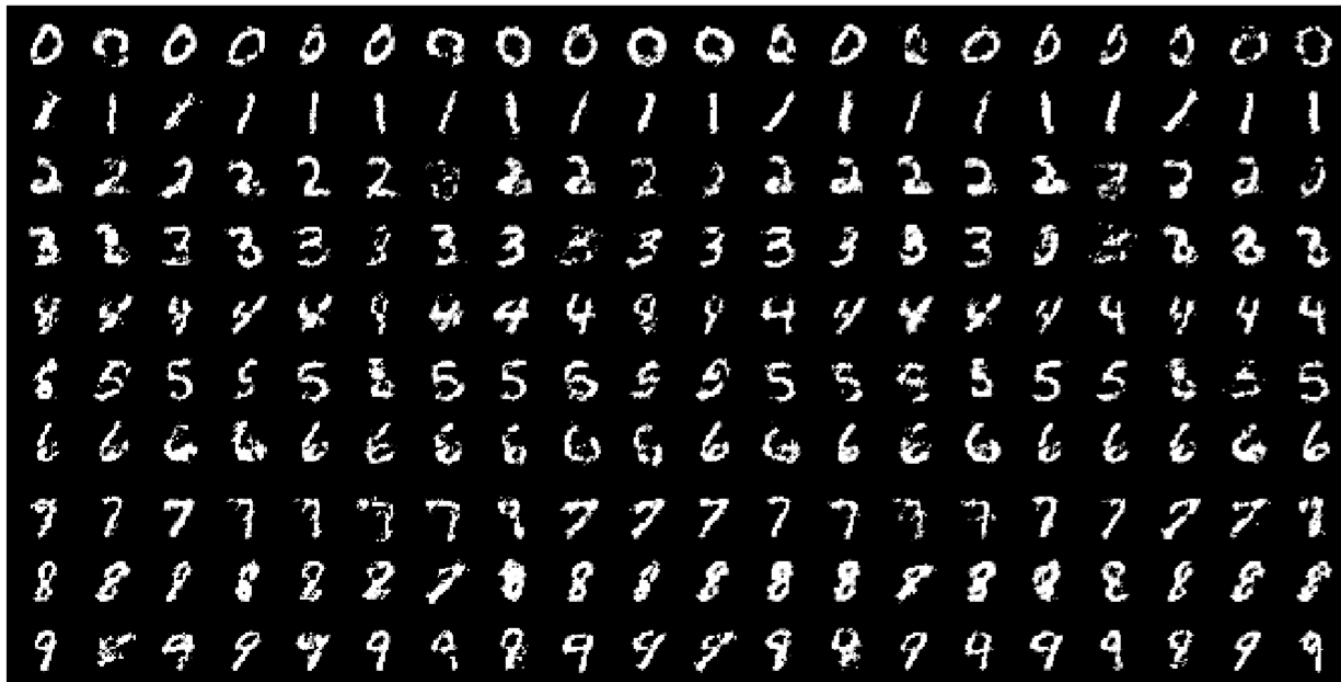



Figure 2: Generated MNIST digits, each row conditioned on one label





DCGAN

Unsupervised Representation Learning with
Deep Convolutional Generative Adversarial
Networks



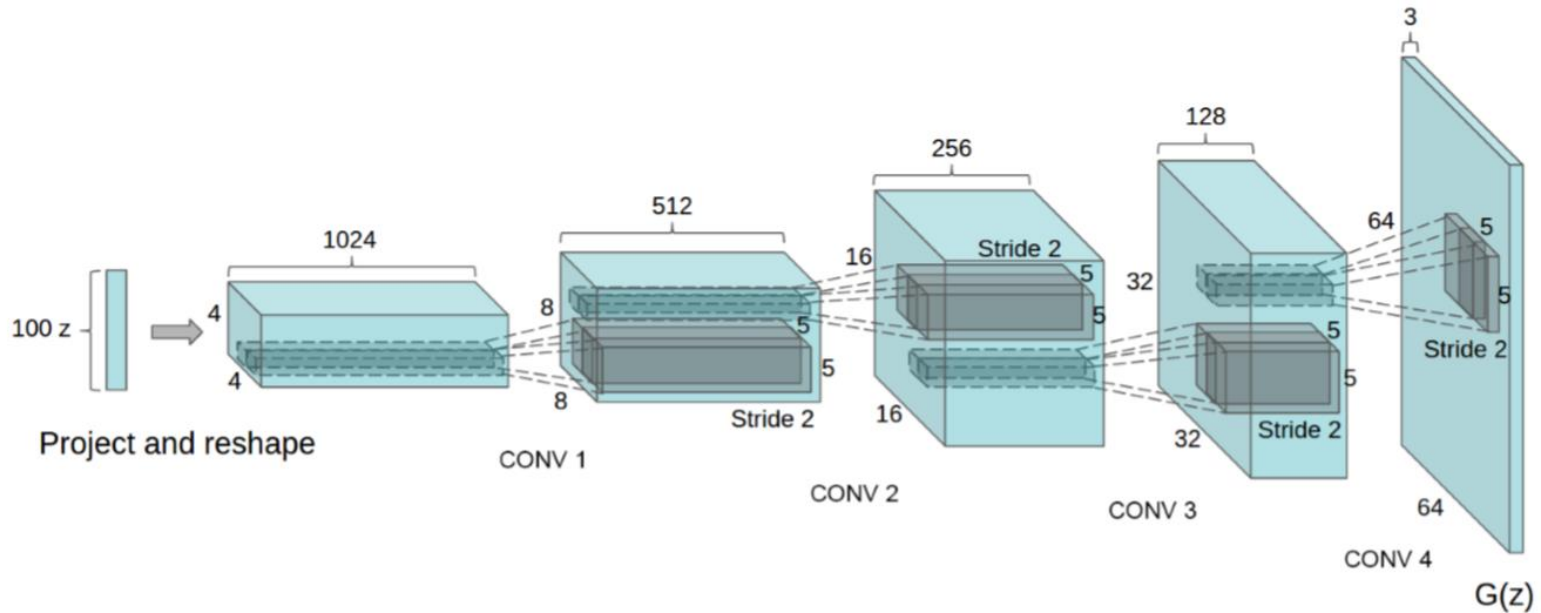


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

定义了一系列网络结构，使得训练更稳定，达到更优更深的生成模型。

■ DCGAN 的改进



(1) 采用全卷积神经网络。不使用空间池化，在判别器中使用带步长的卷积层（strided convolution）。这么做能让网络自己学习更合适的下采样方法。对于generator来说，要做上采样，采用的是分数步长的卷积（fractional-strided convolution）。

(2) 在生成器和判别器中都添加了Batch normalization操作。Batch normalization能确保每个节点的输入都是均值为0，方差为1。即使是初始化很差，也能保证网络中有足够强的梯度。

(3) 避免在卷积层之后使用全连接层。全连接层虽然增加了模型的稳定性，但也减缓了收敛速度。一般来说，generator的输入（噪声）采用均匀分布；discriminator的最后一个卷积层一般先摊平（flatten），然后接一个单节点的softmax。

(4) 生成器的输出层使用Tanh 激活函数，其他层使用RELU。

(5) 判别器的所有层都是用LeakyReLU 激活函数。



■ 实验：调查和可视化网络的内部结构

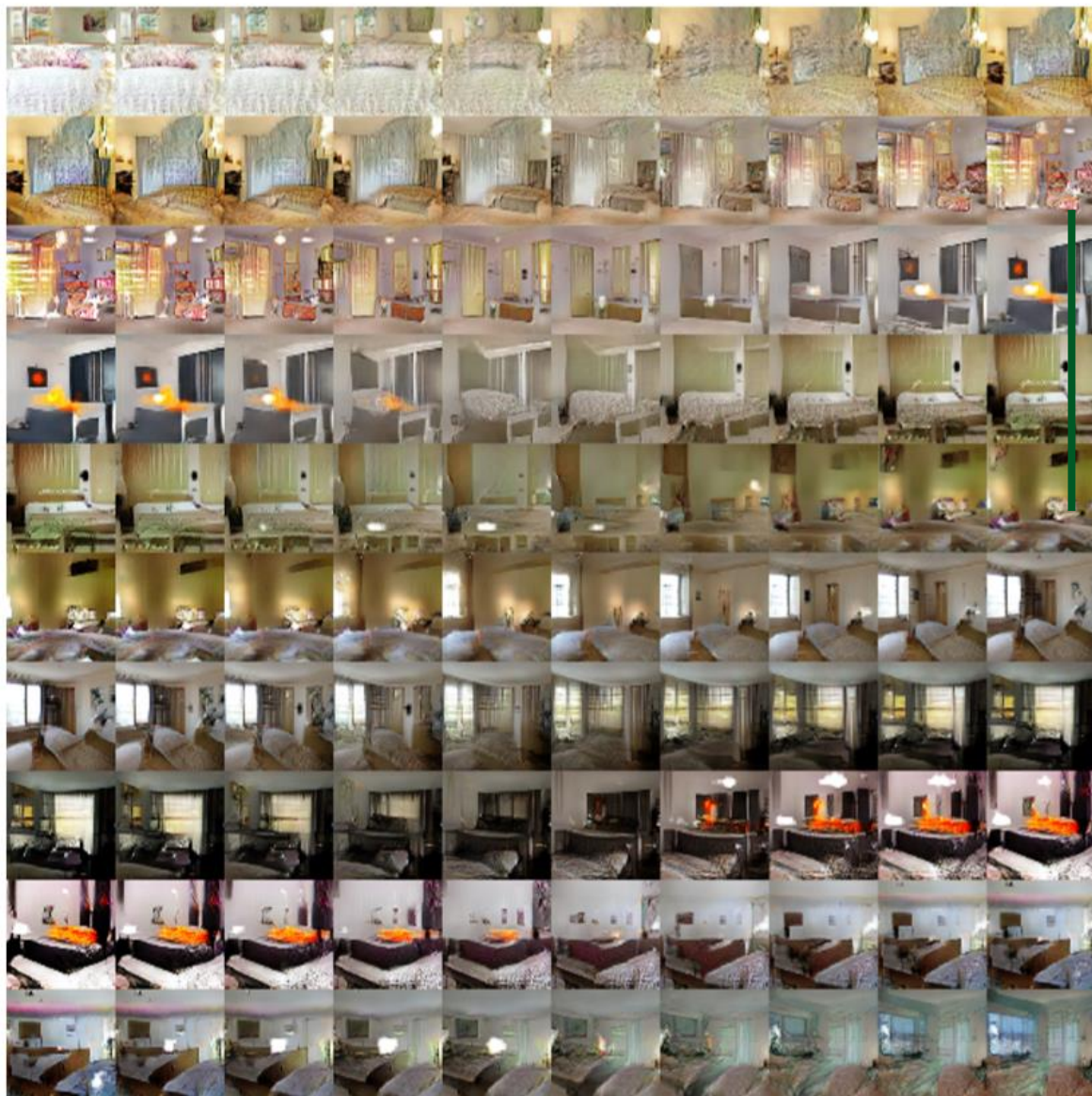


- 漫游隐空间 Walking in the latent space
- 可视化判别器特征 Visualizing the discriminator features
- 操作生成器表示 Manipulating the generator representation



漫游隐空间

在 Z 中的随机点之间
进行插值，表明学
习的空间具有平滑
的过渡



■ 可视化判别器特征



Random filters

Trained filters

■ 操作生成器表示

- 忘记画一些物体



尝试从生成器中完全删除窗户。

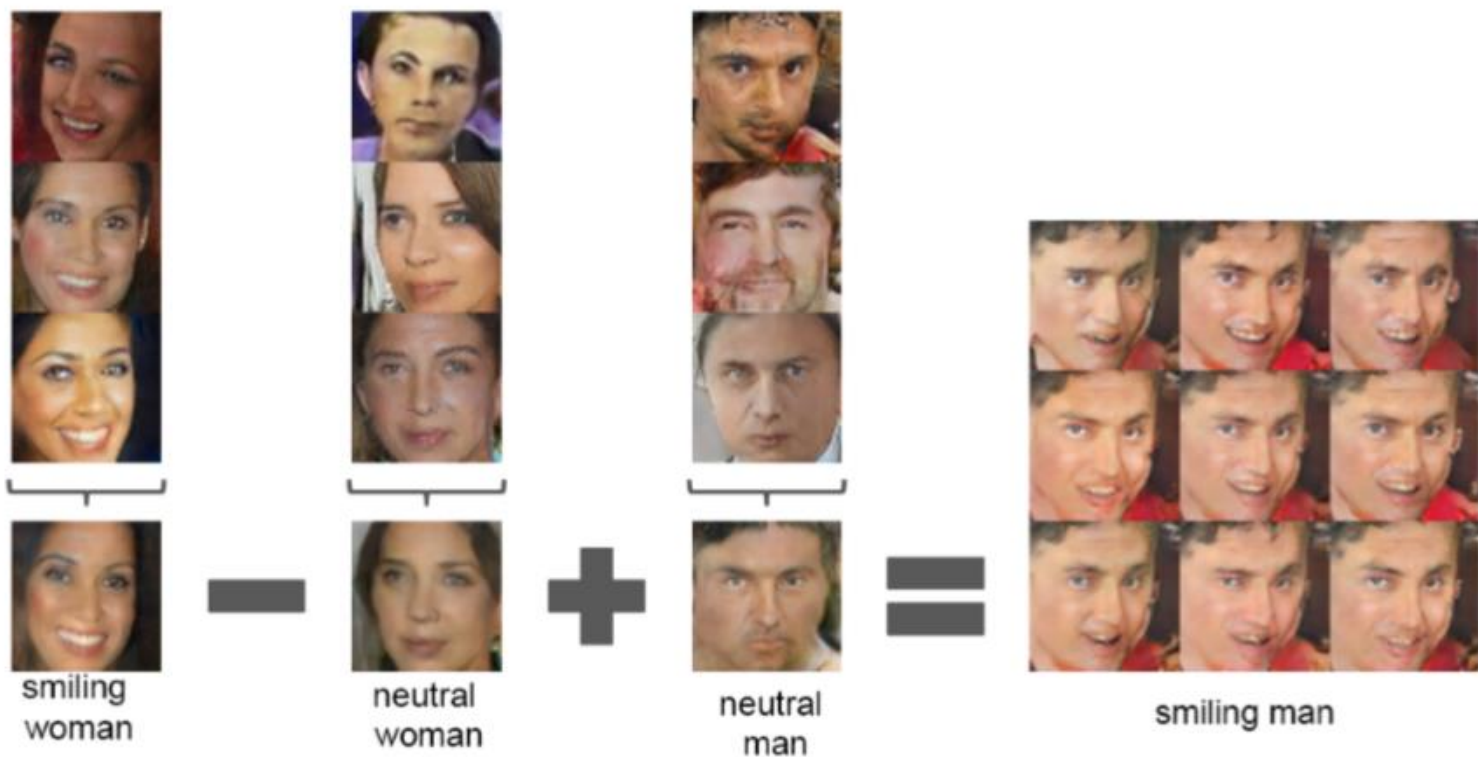
在150个样本上，手动绘制了52个窗口边框。在卷积层特征上，然后用逻辑回归，来预测一个特征激活是否是在窗户上。使用这个简单模型，从所有空间位置删除有关窗户的特征映射。然后，生成随机的新样本。

顶行：未修改的模型样本。

底行：删除“窗户”生成相同的样本。一些窗户被拆除，另一些被转换成具有相似视觉外观的物体。虽然视觉质量下降，但整体场景构成保持相似，表明生成器已经在对象表示中很好的解决了场景表示。

操作生成器表示

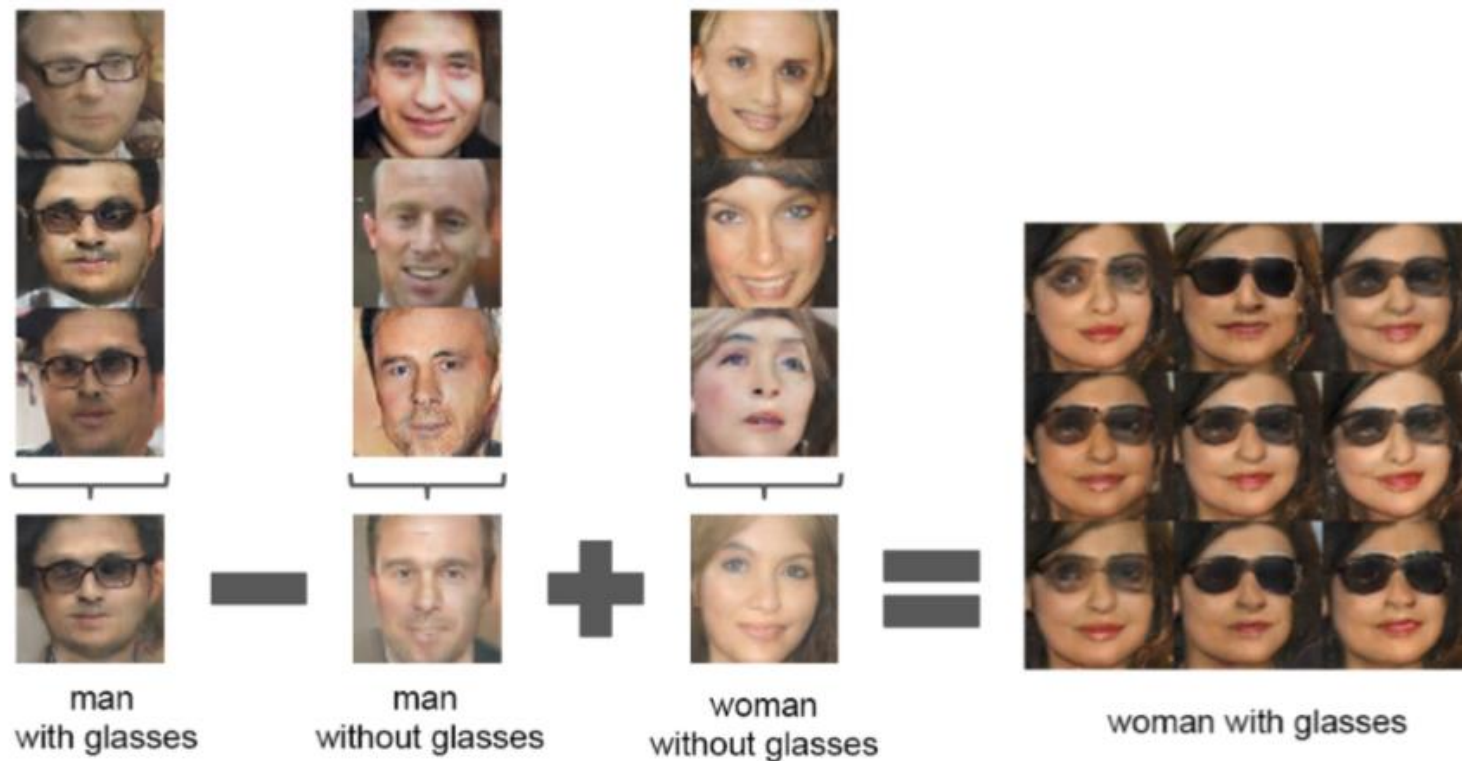
- 人脸样本的向量计算



图像也可以在隐空间里进行加减法来得到新的图像

操作生成器表示

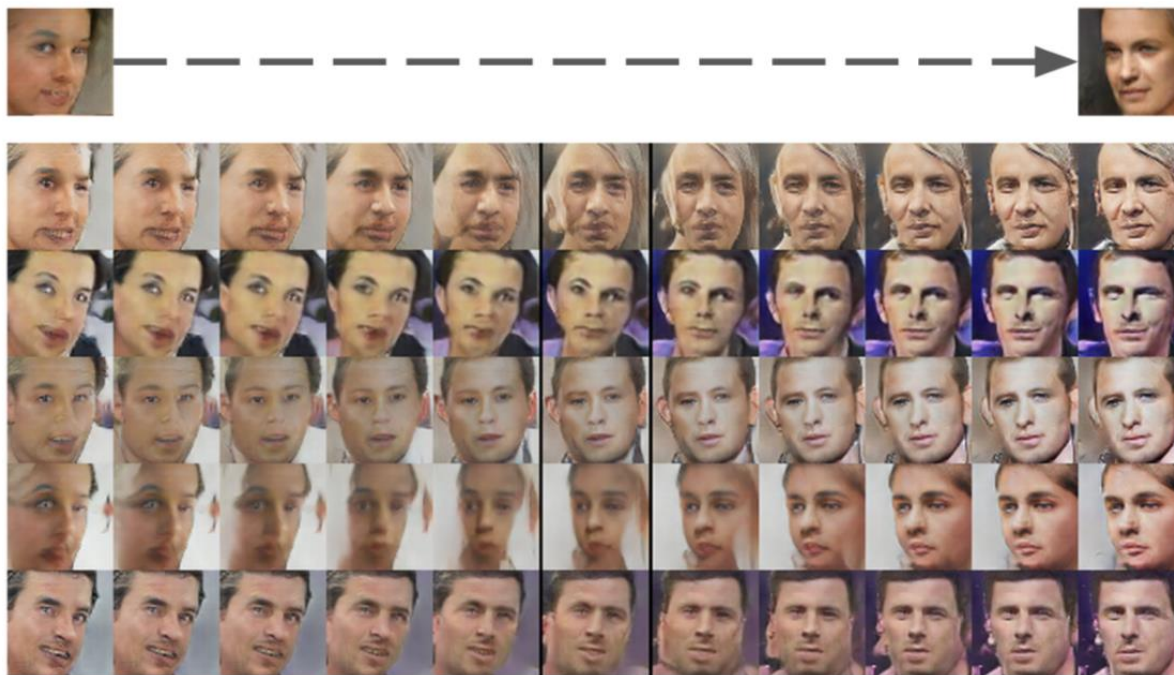
- 人脸样本的向量计算



图像也可以在隐空间里进行加减法来得到新的图像

■ 操作生成器表示

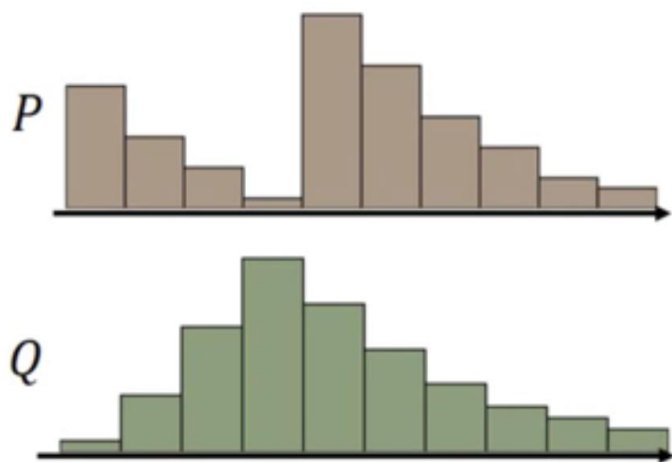
- 人脸样本的向量计算



人脸姿态也是在Z空间中线性建模的，可以看出他们脸的特征在隐空间是均匀过度的。

WGAN

Wasserstein Distance (The Earth Mover's Distance)



WGAN 采用了 Wasserstein 距离来代替 JSD。
Wasserstein 距离也叫做推土机距离 (Earth Mover's distance)。

如果将分布看做空间中泥土的分布，那么这两个分布间的距离就是将这些泥土从一个分布改变到另一个分布所需要消耗的最小能量（这里的能量是泥土重量和移动距离的乘积）。

所以可以将推土机距离理解为从一个分布变化为另一个分布的最小代价。

■ 应用



GAN本身是一种生成式模型，所以在数据生成上用的是最普遍的，最常见的是图片生成，常用的有DCGAN WGAN，BEGAN。

GAN本身也是一种无监督学习的典范，因此它在无监督学习，半监督学习领域都有广泛的应用。

不仅在生成领域，GAN在分类领域也占有一席之地，简单来说，就是替换判别器为一个分类器，做多分类任务，而生成器仍然做生成任务，辅助分类器训练。

GAN可以和强化学习结合，比如seq-GAN。

GAN用在图像风格迁移，图像降噪修复，图像超分辨率，都有比较好的结果。





THANKS



- 王若琪
- 20210529