



# 生成对抗网络

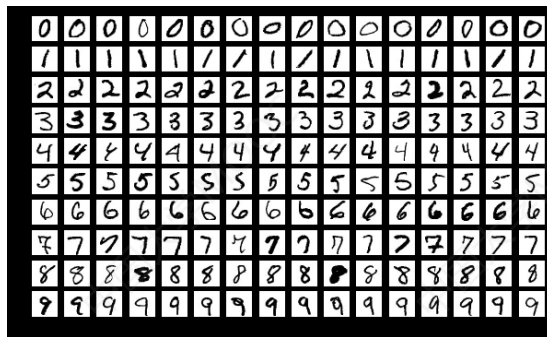
复旦大学 赵卫东

# 目录

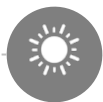
- 生成对抗网络
- DCGAN
- WassersteinGAN
- 案例

# 生成对抗网络概述

- 生成对抗网络 (Generative Adversarial Network, GAN)是非监督学习
- GAN 是Ian Goodfellow在2014年提出，作为非监督深度学习算法推广。
- GAN的功能：给定一批样本，训练一个系统，能够**生成**类似的新样本。



# 生成对抗网络基本组成



## 生成方法和判别方法

机器学习方法可以分为生成方法和判别方法，所学到的模型分别称为生成式模型和判别式模型。生成方法训练好的模型能够生成符合样本分布的新数据，判别方法由数据直接学习决策函数判别模型。



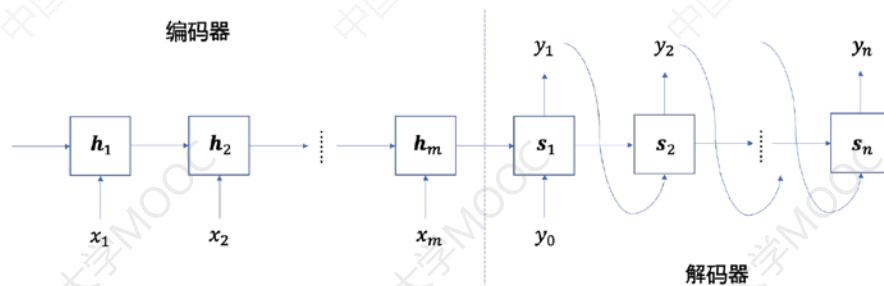
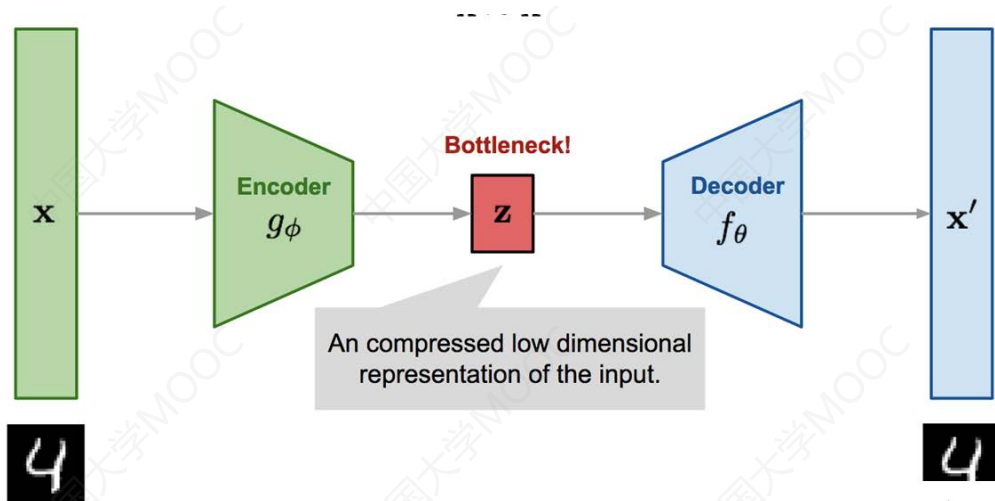
## 生成对抗网络

生成对抗网络，由两个网络组成，即生成器和判别器，生成器用来建立满足一定分布的随机噪声和目标分布的映射关系，判别器用来区别实际数据分布和生成器产生的数据分布。



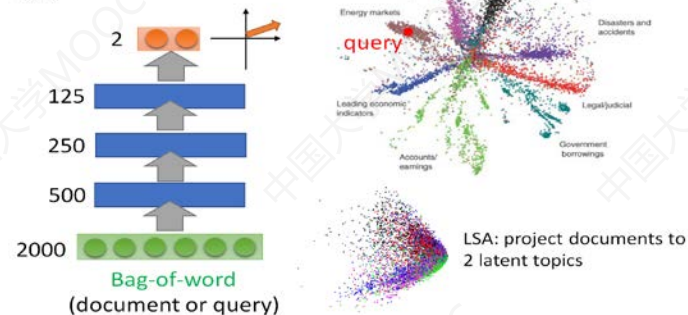
# 自编码器

## ➤ 生成器与自编码器

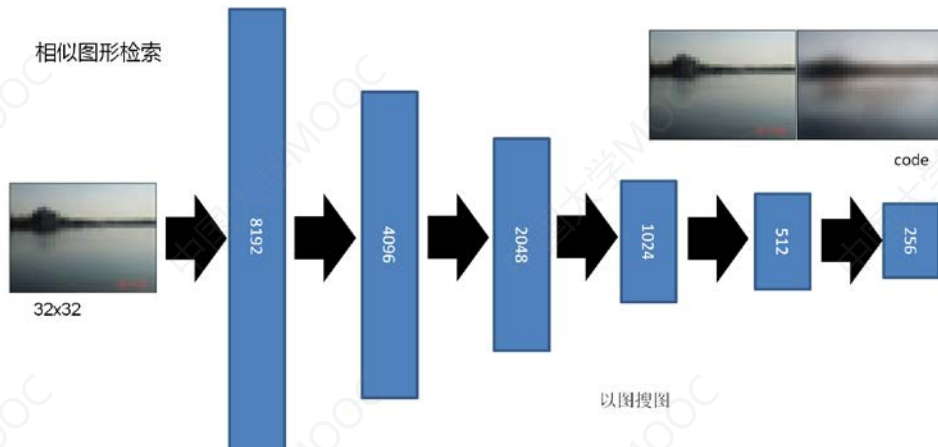


# 自编码器的应用

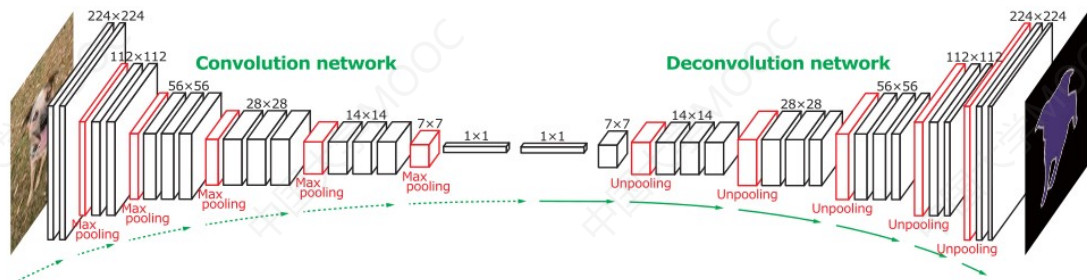
讨论相同事物的文本有相同的编码



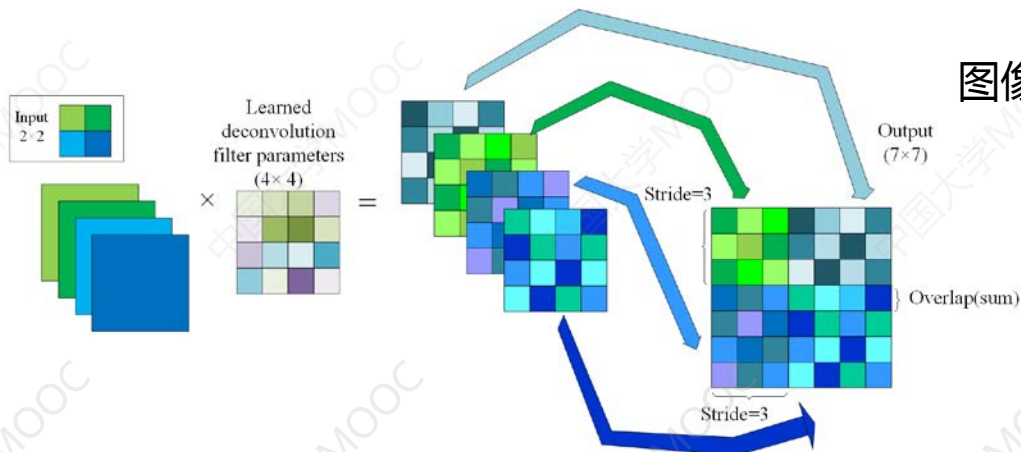
相似图形检索



# 自编码器的结构



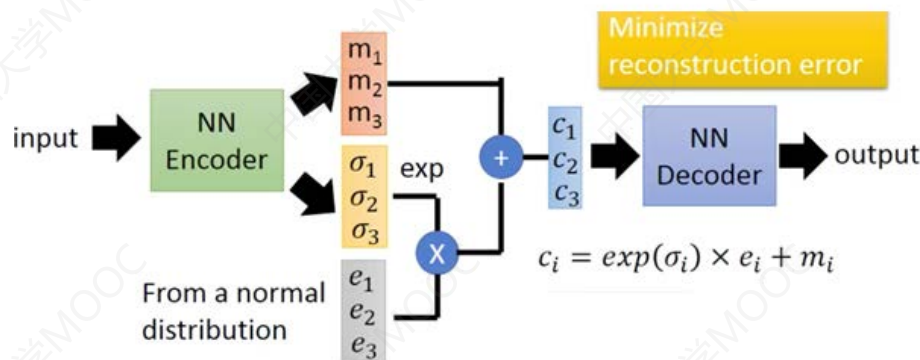
图像的反卷积



输入：2x2，卷积核：4x4，滑动步长：3，输出：7x7

# 变分自编码器VAE

- VAE在2013年12月被提出，是一种利用深度学习自编码器，将深度学习和贝叶斯推断结合，以完成低维向量空间的编码和向高维向量空间的反解码。
- VAE在经典自编码器的基础上，改变了编解码方式，得到连续结构化的空间。VAE将图像转换为统计分布参数(平均值和方差)。然后使用这两个参数从分布中随机采样并将其解码到原始输入。





# VAE的目标函数

VAE的损失函数由两部分组成：交叉熵和KL散度。VAE模型使用交叉熵(cross entropy)来计算输入 $x$ 和 $\hat{x}$ 之间的差异：

$$cross\_entropy = \sum_{i=1}^n - \left[ x_i \cdot \log(\hat{x}_i) + (1 - x_i) \cdot \log(1 - \hat{x}_i) \right]$$

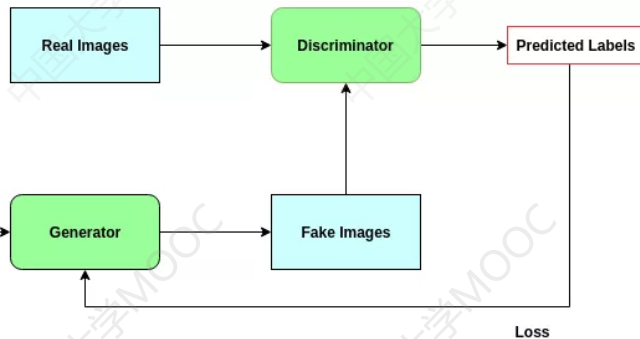
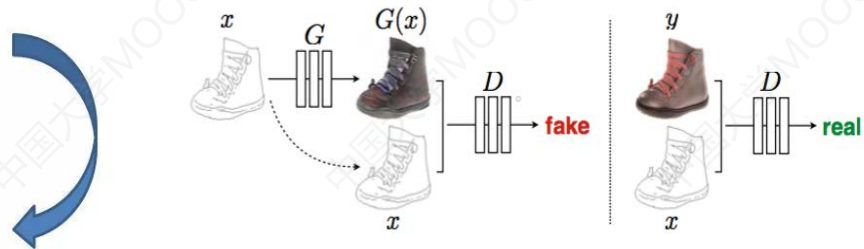
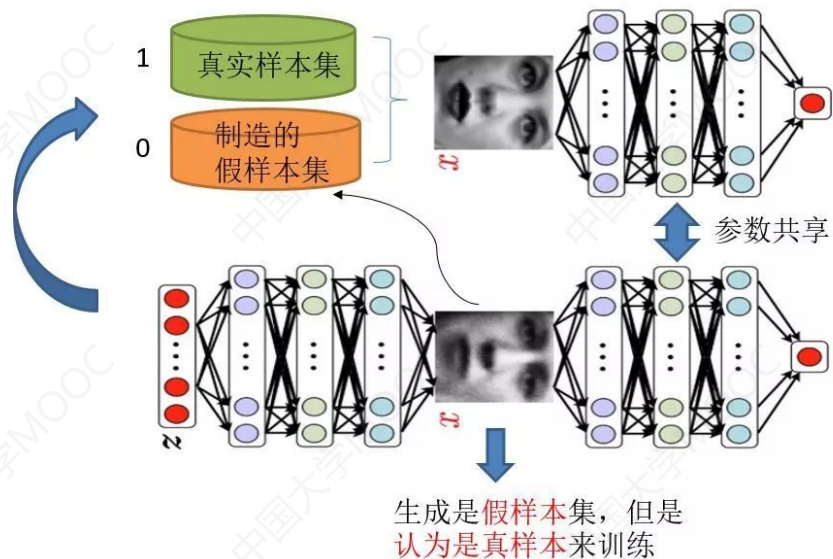
VAE模型使用KL散度来约束 $Z\_mean$ 和 $Z\_variance$ ：

$$KL = -0.5 \left( 1 + \log \sigma^2 - \mu^2 - \exp(\log \sigma^2) \right)$$

损失函数如下：

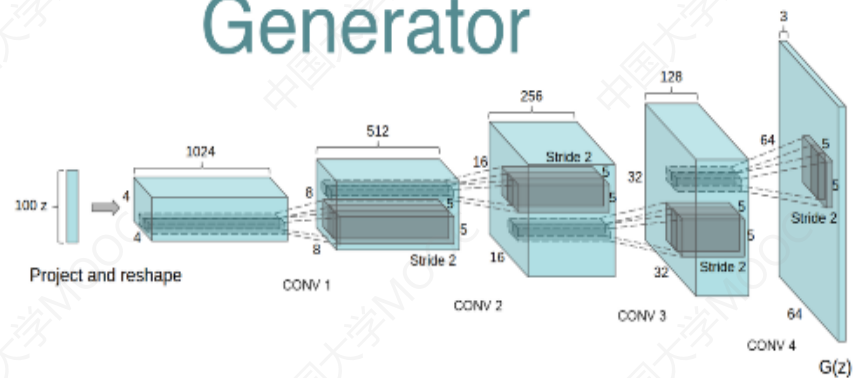
$$loss = cross\_entropy + KL$$

# GAN基本结构

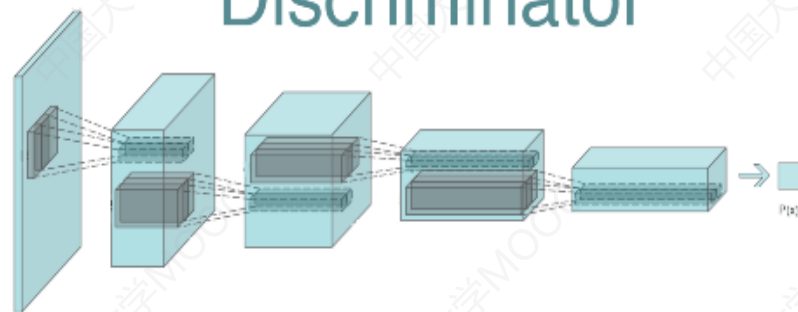


# 生成器与判别器

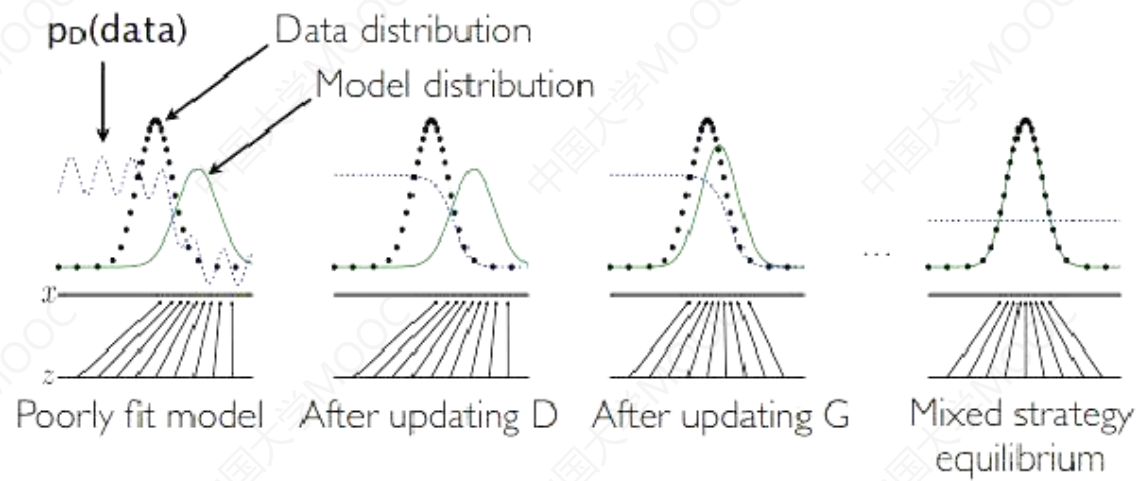
## Generator



## Discriminator

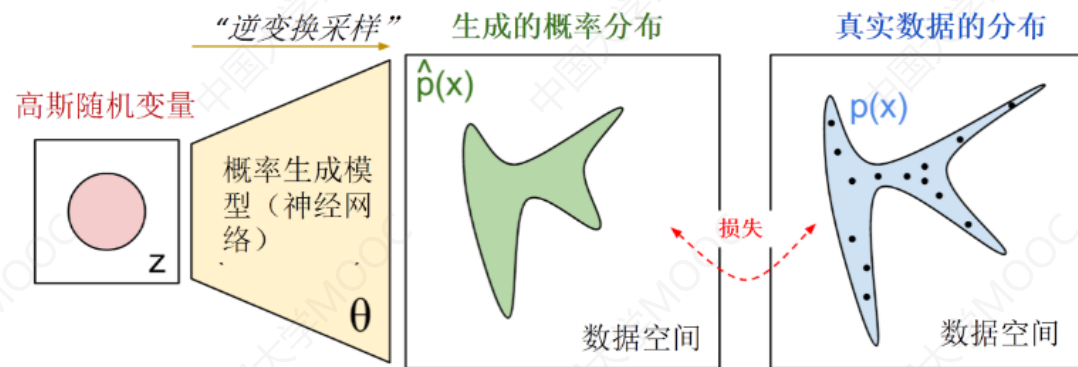


# GAN训练过程



# 如何定义损失函数

- 通过优化目标，使得生成的概率分布和真实数据分布尽量接近。



怎样定义损失（优化目标）？-> 寻找生成模型与判别模型之间的纳什均衡

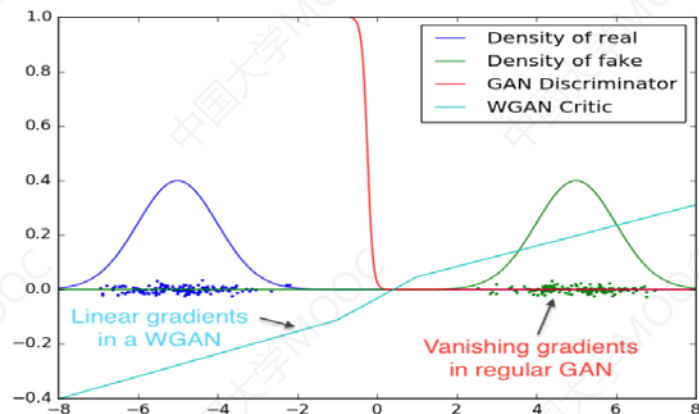
# GAN损失函数

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

- $D(\mathbf{x})$ 表示判别器认为 $\mathbf{x}$ 是真实样本的概率，而 $1-D(G(\mathbf{z}))$ 则是判别器认为生成样本为假的概率。训练GAN的时候，判别器希望损失函数最大化，也就是使判别器判断真实样本为“真”，判断生成样本为“假”的概率最大化；与之相反，生成器希望该目标函数最小化。
- 生成模型：最小化判别模型D的判别准确率。
- 判别模型：尽量最大化自己的判别准确率
- 在训练的过程中固定一方，更新另一方的网络权重，交替迭代，在这个过程中，双方都极力优化自己的网络，从而形成竞争对抗，直到双方达到一个动态的平衡。此时生成模型 G 恢复了训练数据的分布（造出了和真实数据一模一样的样本），判别模型再也判别不出来结果，准确率为 50%。

# GAN的缺点

- 当生成器和判别器的样本分布不重叠时，JS 散度的梯度始终为 0，从而导致此时 GAN 的训练出现梯度消失现象。

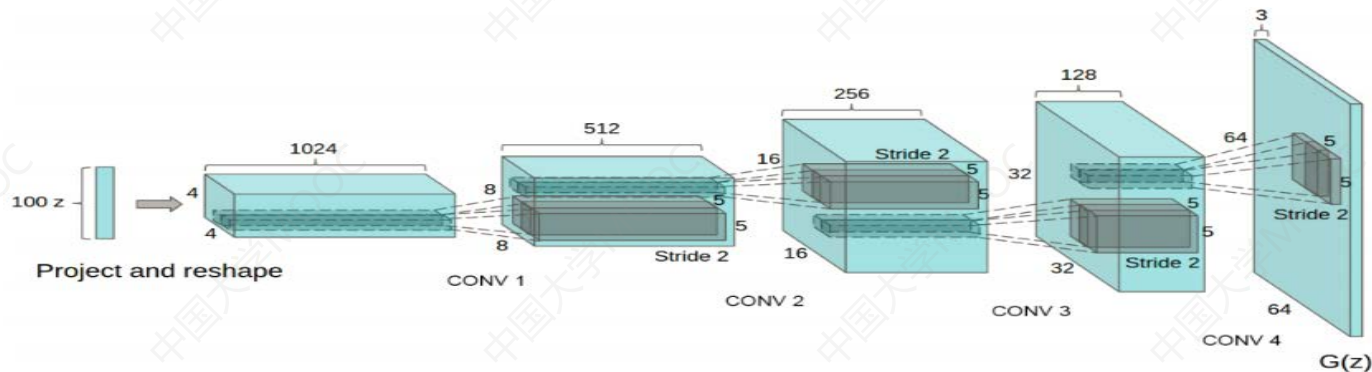


高斯分布的判别器分割面 (Arjovsky, Chintala, & Bottou, 2017)



## DCGAN模型

- 把有监督学习的CNN与无监督学习的GAN整合到一起提出了Deep Convolutional Generative Adversarial Networks - DCGANs，生成器和判别器分别学到输入图像层次化的表示。
- 使用DCGANs从大量的无标记数据（图像、语音）学习到有用的特征，相当于利用无标记数据初始化DCGANs的生成器和判别器的参数，在用于有监督场景。





FileEditSearchSourceRunDebugConsolesProjectsToolsViewHelp

C:\Users\admin\Desktop\大数据技术\_深度学习资料\实验\常用机器学习算法python代码\生成对抗网络\GAN\_mnist\GAN.py

sr\_model.pyNN-银行客户流失预测.pyhandwrite\_recognition.pyreset.py自编码器(Autoencoder).pyGAN.pySourceConsoleObject

1import tensorflow.compat.v1 as tf  
2tf.disable\_v2\_behavior()  
3  
4import numpy as np  
5import matplotlib.pyplot as plt  
6  
7from tensorflow.examples.tutorials.mnist import input\_data  
8  
9  
10  
11  
12class MnistGAN:  
13  
14 def \_\_init\_\_(self):  
15 # 定义数据集  
16 self.mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)  
17 # 定义批次大小  
18 self.batch\_size = 64  
19 # 生成器图片大小  
20 self.img\_size = self.mnist.train.images[0].shape[0]  
21 # 定义分块大小  
22 self.chunk\_size = self.mnist.train.num\_examples // self.batch\_size  
23 # 定义迭代次数  
24 self.epoch\_size = 200  
25 # 定义学习率  
26 self.lr = 1e-4  
27 # 真实图片  
28 self.real\_img = tf.placeholder(tf.float32, [None, self.img\_size])  
29 # 生成图片  
30 self.fake\_img = tf.placeholder(tf.float32, [None, self.img\_size])  
31 # Leaky Relu 参数  
32 self.leaky = 0.01  
33 # 隐藏单元数量  
34 self.hidecell = 256  
35 # 测试样本数量  
36 self.test = 20  
37  
38 # 定义生成器网络  
39 @staticmethod  
40 def generator(hidecell, img\_size, leaky, input):  
41 with tf.variable\_scope("generator"):  
42 # layerH\_arg = tf.get\_variable('gh1', tf.truncated\_normal([self.img\_size, self.hidecell], stddev=0.02))  
43 layerH = tf.layers.dense(input, hidecell)  
44 layerR = tf.maximum(layerH, leaky \* layerH)  
45 drop = tf.layers.dropout(layerR, rate=0.5)  
46 # layer\_arg = tf.get\_variable('gh2', tf.truncated\_normal([self.hidecell, self.img\_size], stddev=0.02))  
47 logits = tf.layers.dense(drop, img\_size)  
48 output = tf.tanh(logits)  
49 return logits, output  
50  
51 # 定义判别器网络  
52 @staticmethod  
53 def discriminator(leaky, hidecell, input, reuse=False):  
54 with tf.variable\_scope("discriminator", reuse=reuse):  
55 # layerH\_arg = tf.get\_variable('dh1', tf.truncated\_normal([self.img\_size, self.hidecell], stddev=0.02))  
56 layer = tf.layers.dense(input, hidecell)  
57 relu = tf.maximum(leaky \* layer, layer)  
58 # arg = tf.get\_variable('dh2', tf.truncated\_normal([self.hidecell, 1], stddev=0.02))  
59 logits = tf.layers.dense(relu, 1)  
60 output = tf.sigmoid(logits)  
61 return logits, output  
62  
63 # 定义损失  
64 @staticmethod  
65 def Loss(fake\_logits, real\_logits):  
66 # 生成器损失  
67 g\_loss = tf.reduce\_mean(-

Help

Usage

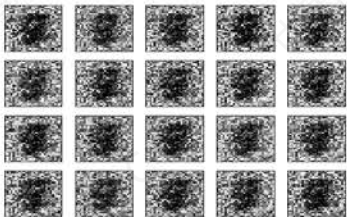
Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.  
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in **Preferences > Help**.  
[New to Spyder? Read our tutorial](#)

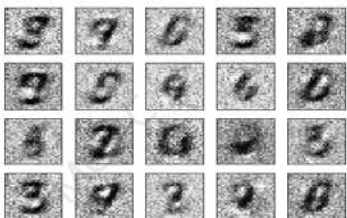

Variable explorerFile explorerHelp

IPython console

Console 1/A

Use tf.where in 2.0, which has the same broadcast rule as np.where  
迭代: 0 g\_loss=1.5151732 d\_loss=0.67709833

  
迭代: 1 g\_loss=2.106542 d\_loss=0.33510345  
迭代: 2 g\_loss=1.3653777 d\_loss=0.542213  
迭代: 3 g\_loss=1.1785988 d\_loss=0.58836067  
迭代: 4 g\_loss=1.4521384 d\_loss=0.50930583  
迭代: 5 g\_loss=1.429307 d\_loss=0.58065265

# WGAN算法

- GAN算法的损失函数使用了JS散度，当生成器和鉴别器的样本分布不重叠时，导致 GAN 的训练出现梯度消失现象，参数很难更新，网络无法收敛。
- WGAN使用一种分布距离度量方法：Wasserstein距离，即推土机Earth-Mover距离解决上述问题。

# Wasserstein距离

- Wasserstein距离，即Earth-Mover ( EM ) 距离：

$$W(P_r, P_g) = \inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

- $\Pi(P_r, P_g)$  是  $P_r$  和  $P_g$  组合起来的所有可能的联合分布的集合。对于每一个可能的联合分布 $\gamma$ 而言，可以从其中采样得到一个真实样本  $x$  和一个生成样本  $y$ ，并算出这对样本的距离  $\|x - y\|$ ，所以可以计算该联合分布 $\gamma$ 下样本对距离的期望值。在所有可能的联合分布中能够对这个期望值取到的下界，就定义为Wasserstein距离。
- Wasserstein距离相比KL散度、JS散度的优越性在于，即便两个分布没有重叠，Wasserstein距离仍然能够反映它们的远近。
- 传统GAN的判别器做的是真假二分类任务，最后一层使用sigmoid函数，但WGAN中的判别器需要近似拟合Wasserstein距离，属于回归任务，需要把替换sigmoid函数。

# WGAN

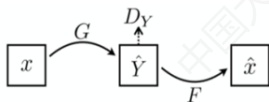
- 生成器要近似地最小化Wasserstein距离，可以最小化 loss。WGAN的两个loss
- 生成器损失函数： $-\mathbb{E}_{x \sim P_g}[f_w(x)]$
- 判别器损失函数： $\mathbb{E}_{x \sim P_g}[f_w(x)] - \mathbb{E}_{x \sim P_r}[f_w(x)]$

可以指示训练进程，其数值越小，表示真实分布与生成分布的Wasserstein距离越小，GAN训练得越好。

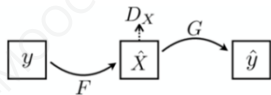
# CycleGAN (1)

训练包含以下两个部分：

1.  $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$



2.  $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$



由于存在两组映射函数  $G$  与  $F$ ，对于  $G$  来说，对抗损失的目标函数为：

$$\text{Loss}_{\text{gan}}(G, D_Y, X, Y) = E_{y \sim P_{\text{data}}(y)}[\log D_Y(y)] + E_{x \sim P_{\text{data}}(x)}[\log(1 - D_Y(G(x)))]$$

即：

$$\min_G \max_{D_Y} \text{Loss}_{\text{gan}}(G, D_Y, X, Y)$$

对于  $F$  来说，对抗损失为：

$$\text{Loss}_{\text{gan}}(F, D_X, Y, X) = E_{x \sim P_{\text{data}}(x)}[\log D_X(x)] + E_{y \sim P_{\text{data}}(y)}[\log(1 - D_X(F(y)))]$$

$$\text{即 } \min_F \max_{D_X} \text{Loss}_{\text{gan}}(F, D_X, Y, X)$$

$$\text{adversarial loss} = \text{Loss}_{\text{gan}}(G, D_Y, X, Y) + \text{Loss}_{\text{gan}}(F, D_X, Y, X)$$

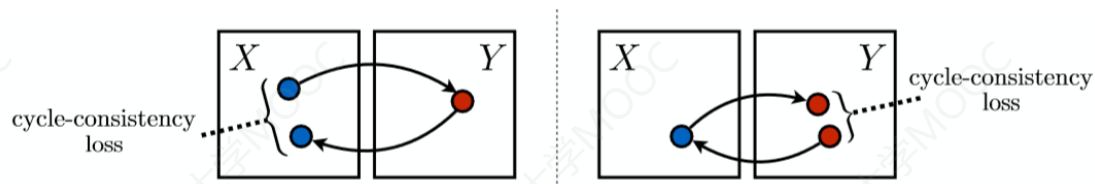
为了刻画  $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$  与  $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$  的行为，将 cycle-consistency loss 定义如下：

$$\text{Loss}_{\text{cyc}} = E_{x \sim P_{\text{data}}(x)}[|F(G(x)) - x|] + E_{y \sim P_{\text{data}}(y)}[|G(F(y)) - y|]$$

最终的损失函数为：

$$\text{Loss}(G, F, D_X, D_Y) = \text{Loss}_{\text{gan}}(G, D_Y, X, Y) + \text{Loss}_{\text{gan}}(F, D_X, Y, X) + \text{Loss}_{\text{cyc}}$$

# CycleGAN (2)

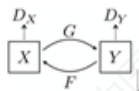


训练集分为两个部分：X 和 Y，其中 X 为普通图片，Y 为梵高风格图片，实验包含两组映射关系：

$G: X \rightarrow Y$  (generator  $X \rightarrow Y$ )

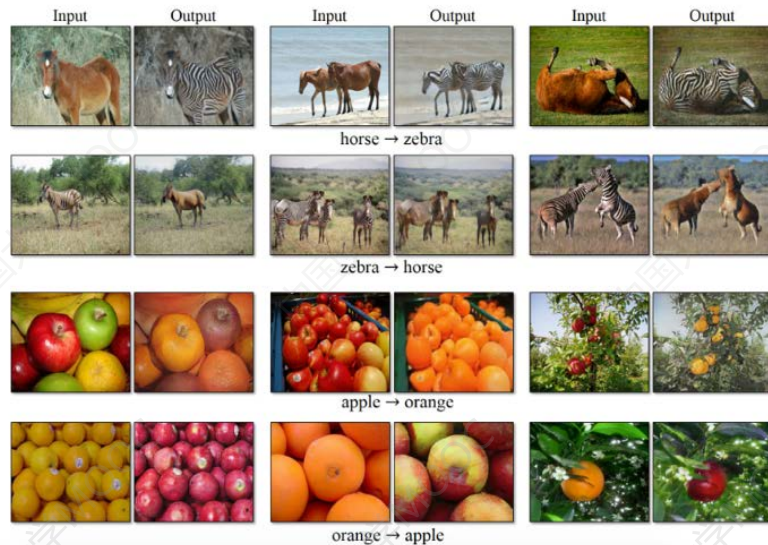
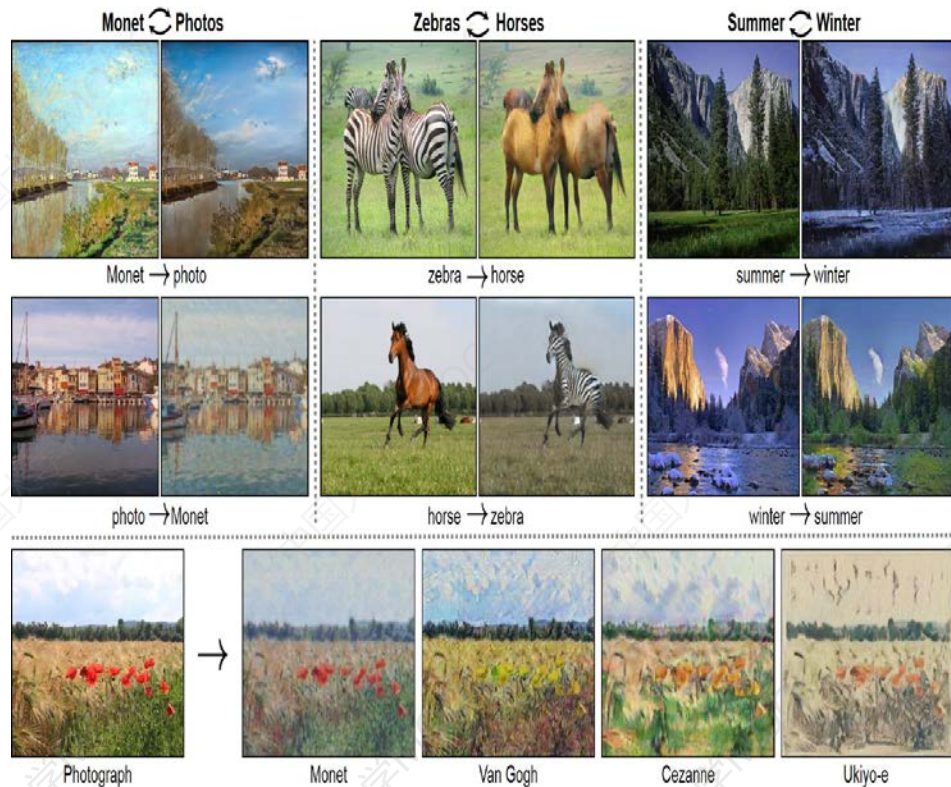
$F: Y \rightarrow X$  (generator  $Y \rightarrow X$ )

还有两组 discriminators  $D_X$  与  $D_Y$ ，其中  $D_X$  用于判断  $F(Y)$  是否属于 X， $D_Y$  用于判断  $G(X)$  是否属于 Y。





# 生成对抗网络应用 (1)



# 生成对抗网络应用（2）

## 超分辨率图像重建





## 生成对抗网络应用 ( 3 )

图像去雨



(a)



(b)



(c)



(d)

# 生成对抗网络应用（4）

## 图像风格转移

