
XSam's Blog Documentation

发布 1.0

Sam

2018 年 11 月 21 日

Contents:

1	前言	1
2	第一章 python基础知识	3
2.1	1.1 python简介	3
2.2	1.2 Anaconda的下载与安装	4
2.3	1.3 python基础语法	4
3	第二章 深度学习与TensorFlow简介	5
3.1	2.1 什么是深度学习	5
3.2	2.2 TensorFlow	5
4	第三章 TensorFlow入门	7
5	第四章 卷积神经网络	9
6	第五章 生成对抗网络	11
6.1	5.1 生成对抗网络简介	11
6.2	5.2 GAN	14
6.3	5.3 DCGAN	21
6.4	5.4 LSGAN	30
6.5	5.5 WGAN	30
6.6	5.6 CycleGAN	30
6.7	5.7 CGAN	30
6.8	5.8 StarGAN	30

CHAPTER 1

前言

记录自己深度学习的学习过程。

CHAPTER 2

第一章 python基础知识

- *1.1 python*简介
- *1.2 Anaconda*的下载与安装
- *1.3 python*基础语法

2.1 1.1 python简介

Python是著名的“龟叔”Guido van Rossum在1989年圣诞节期间，为了打发无聊的圣诞节而编写的一个编程语言。

Python就为我们提供了非常完善的基础代码库，覆盖了网络、文件、GUI、数据库、文本等大量内容，被形象地称作“内置电池（batteries included）”。用Python开发，许多功能不必从零编写，直接使用现成的即可。除了内置的库外，Python还有大量的第三方库，也就是别人开发的，供你直接使用的东西。当然，如果你开发的代码通过很好的封装，也可以作为第三方库给别人使用。许多大型网站就是用Python开发的，例如YouTube、Instagram，还有国内的豆瓣。很多大公司，包括Google、Yahoo等，甚至NASA（美国航空航天局）都大量地使用Python。龟叔给Python的定位是“优雅”、“明确”、“简单”，所以Python程序看上去总是简单易懂，初学者学Python，不但入门容易，而且将来深入下去，可以编写那些非常非常复杂的程序。

总的来说，Python的哲学就是简单优雅，尽量写容易看明白的代码，尽量写少的代码。如果一个资深程序员向你炫耀他写的晦涩难懂、动不动就几万行的代码，你可以尽情地嘲笑他。

那Python适合开发哪些类型的应用呢？

1. 网络应用，包括网站、后台服务等等；
2. 许多日常需要的小工具，包括系统管理员需要的脚本任务等等；
3. 把其他语言开发的程序再包装起来，方便使用。

最后说说Python的缺点。任何编程语言都有缺点，Python也不例外。优点说过了，那Python有哪些缺点呢？

1. 运行速度慢，和C程序相比非常慢，因为Python是解释型语言，你的代码在执行时会一行一行地翻译成CPU能理解的机器码，这个翻译过程非常耗时，所以很慢。而C程序是运行前直接编译成CPU能执行的机器码，所以非常快。但是大量的应用程序不需要这么快的运行速度，因为用户根本感觉不出来。例如开发一个下载MP3的网络应用程序，C程序的运行时间需要0.001秒，而Python程序的运行时间需要0.1秒，慢了100倍，但由于网络更慢，需要等待1秒，你想，用户能感觉到1.001秒和1.1秒的区别吗？这就好比F1赛车和普通的出租车在北京三环路上行驶的道理一样，虽然F1赛车理论时速高达400公里，但由于三环路堵车的时速只有20公里，因此，作为乘客，你感觉的时速永远是20公里。
2. 代码不能加密。如果要发布你的Python程序，实际上就是发布源代码，这一点跟C语言不同，C语言不用发布源代码，只需要把编译后的机器码（也就是你在Windows上常见的xxx.exe文件）发布出去。要从机器码反推出C代码是不可能的，所以，凡是编译型的语言，都没有这个问题，而解释型的语言，则必须把源码发布出去。

这个缺点仅限于你要编写的软件需要卖给别人挣钱的时候。好消息是目前的互联网时代，靠卖软件授权的商业模式越来越少了，靠网站和移动应用卖服务的模式越来越多了，后一种模式不需要把源码给别人。

再说了，现在如火如荼的开源运动和互联网自由开放的精神是一致的，互联网上有无数非常优秀的像Linux一样的开源代码，我们千万不要高估自己写的代码真的有非常大的“商业价值”。那些大公司的代码不愿意开放的更重要的原因是代码写得太烂了，一旦开源，就没人敢用他们的产品了。

2.2 1.2 Anaconda的下载与安装

Anaconda。

2.3 1.3 python基础语法

python基础语法

CHAPTER 3

第二章 深度学习与TensorFlow简介

- 2.1 什么是深度学习
- 2.2 TensorFlow

3.1 2.1 什么是深度学习

提到人工智能，人们往往会想到深度学习，然而，深度学习不像人工智能那样容易从字面上理解。这是因为深度学习是从内部机理来阐述的，而人工智能是从其应用的角度来阐述的，即深度学习是实现人工智能的一种方法。

人工智能领域，起初是进行神经网络的研究。但神经网络发展到一定阶段后，模型越来越庞大，结构也越来越复杂，于是人们将其命名为“深度学习”。可以这样理解——深度学习属于后神经网络时代。

深度学习近年来的发展突飞猛进，越来越多的人工智能应用得以实现。其本质为一个可以模拟人脑进行分析、学习的神经网络，它模仿人脑的机制来解释数据（如图像、声音和文本），通过组合低层特征，形成更加抽象的高层特征或属性类别，来拟合人们日常生活中的各种事情。

深度学习被广泛应用于与人们生活息息相关的各种领域，可以实现机器翻译、人脸识别、语音识别、信号恢复、商业推荐、金融分析、医疗辅助和智能交通等。

在国内乃至世界，越来越多的资金涌向人工智能领域，人工智能领域新成立的创业公司每年呈递增趋势，越来越多的学校也开始开设与深度学习相关的课程。这个时代，正像是移动互联网的前夜。

3.2 2.2 TensorFlow

CHAPTER 4

第三章 TensorFlow入门

CHAPTER 5

第四章 卷积神经网络

CHAPTER 6

第五章 生成对抗网络

- 5.1 生成对抗网络简介
- 5.2 GAN
- 5.3 DCGAN
- 5.4 LSGAN
- 5.5 WGAN
- 5.6 CycleGAN
- 5.7 CGAN
- 5.8 StarGAN

6.1 5.1 生成对抗网络简介

6.1.1 1. 简介

生成对抗网络(GAN, Generative Adversarial Network)是生成模型的一种，主要用于通过数据分布生成样本。

GAN之父Ian J. Goodfellow等人于2014年10月在 *Generative Adversarial Networks* 中提出了一个通过对抗过程估计生成模型的新框架。框架中同时训练两个模型：捕获数据分布的生成模型G，和估计样本来自训练数据的概率的判别模型D。G的训练程序是将D错误的概率最大化。这个框架对应一个最大值集下限的双方对抗游戏。可以证明在任意函数G和D的空间中，存在唯一的解决方案，使得G重现训练数据分布，而D=0.5。在G和D由多层感知器定义的情况下，整个系统可以用反向传播进行训练。在训练或生成样本期间，不需要任何马尔科夫链或展开的近似推理网络。实验通过对生成的样品的定性和定量评估证明了本框架的潜力。

学习生成模型的理由：

1. 生成样本，这是最直接的理由。
 2. 训练并不包含 **最大似然估计** (MLE, Maximum Likelihood Estimation)。
 3. 生成器对训练数据不可见，过拟合的风险较低。
 4. GAN十分擅长捕获模式的分布。

GAN描述的是一种“读作敌人，写作朋友”的观念，就好比Naruto和Sasuke。



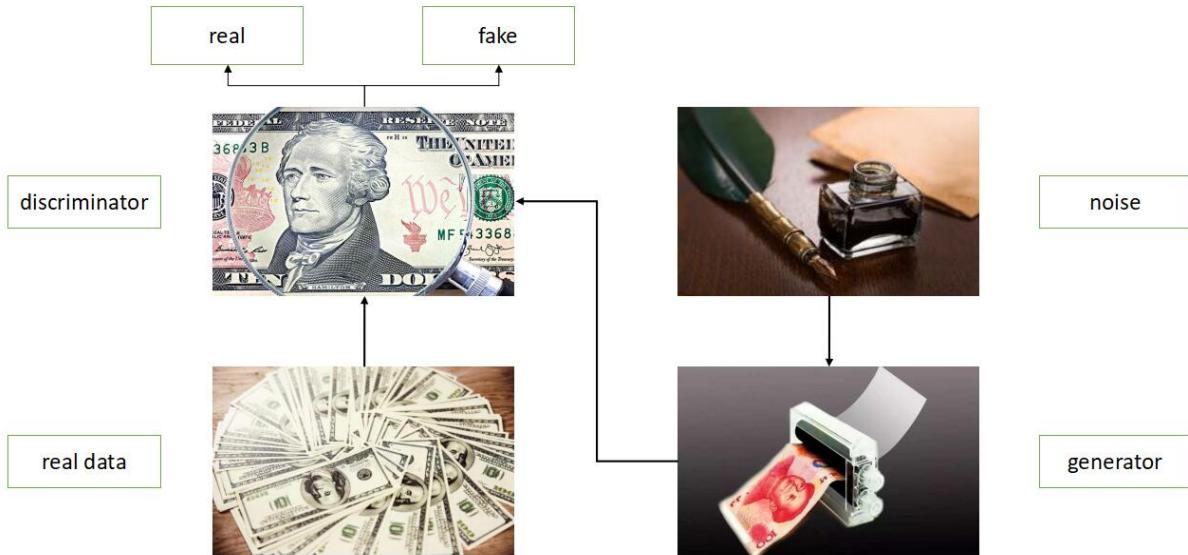
6.1.2 2.原理

GAN原始论文 给出了一个制假钞的例子。

犯罪嫌疑人与警察对于钞票的关注点：

1. 想要成为一名成功的伪钞制作者，犯罪嫌疑人需要可以混淆警察，使得警察无法分辨真钞与伪钞。

2. 警察需要尽可能高效地分辨钞票的真伪。



这个场景表现为博奕论中的极大极小博奕，整个过程被称为对抗性过程(Adversarial Process)。GAN是一种由两个神经网络相互竞争的特殊对抗过程。第一个网络生成数据，第二个网络试图区分真实数据与第一个网络创造出来的假数据。第二个网络会生成一个在[0, 1]区间内的标量，代表数据是真实数据的概率。

在GAN中，第一个网络通常被称为生成器(Generator)并且以 $G(z)$ 表示，第二个网络通常被称为判别器(Discriminator)并且以 $D(x)$ 表示。

在平衡点，也就是极大极小博奕的最优点，生成器生成数据，判别器认为生成器生成的数据是真实数据的概率为0.5，整个过程可以用如下公式表示：

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

在一些情况下这两个网络可以达到平衡点，但是在另一些情况下却不能，两个网络会继续学习很长时间。

- 生成器

生成器网络以随机的噪声作为输入并试图生成样本数据。通常，生成器 $G(z)$ 从概率分布 $p(z)$ 中接收输入 z ，并且生成数据提供给判别器网络 $D(x)$ 。

- 判别器

判别器网络以真实数据或生成器生成的数据为输入，并试图预测当前输入是真实数据还是生成数据。其中一个输入 x 从真实的数据分布 $p_{data}(x)$ 中获取，接下来解决一个二分类问题，产生一个在[0, 1]区间内的标量。

由于真实世界中无标记的数据量远远大于标记数据，而GAN十分擅长无监督学习任务，因此近些年来GAN变得越来越流行。它得以流行的另一个原因在于在多种生成模型中，GAN可以生成最为逼真的图像。尽管这是一个很主观的评价，但这已经成为所有从业者的共识。

此外，GAN还有着强大的表达能力：它可以在潜在空间(向量空间)内执行算数运算，并将其转换为对应特征空间内的运算。如，在潜在空间内有一张戴眼镜男人的照片，减去一个神经网络中男人的向量，再加上一个神经网络中女人的向量，最后会得到特征空空间内一张戴眼镜女人的图像。这种表达能力的确令人叹为观止。

6.2 5.2 GAN

1. GAN论文: <https://arxiv.org/abs/1406.2661>
2. MNIST数据集下载: <http://yann.lecun.com/exdb/mnist>

下载后，将t10k-images-idx3-ubyte.gz, t10k-labels-idx1-ubyte.gz, train-images-idx3-ubyte.gz与train-labels-idx1-ubyte.gz四个压缩包放在项目所在文件夹的”MNIST_data”文件夹中。

3. GAN的实现:

一共四个文件，ops.py, discriminator.py, generator.py以及gan.py。

一. 定义初始文件结构，ops.py, discriminator.py, generator.py以及gan.py。

① ops.py

```

1 import tensorflow as tf
2 import numpy as np
3 import tensorflow.contrib.slim as slim
4 import scipy
5
6 # Help function for linear function
7 def linear(x, output_size, name='linear', stddev=0.02):
8     shape = x.get_shape().as_list()
9     with tf.variable_scope(name):
10         matrix = tf.get_variable("matrix", [shape[1], output_size], tf.float32, tf.
11         ←random_normal_initializer(stddev=stddev))
12         bias = tf.get_variable("bias", [output_size], initializer=tf.constant_
13         ←initializer(0.0))
14         output = tf.matmul(x, matrix) + bias
15         return output
16
17 # Help function for relu
18 def relu(z):
19     return tf.nn.relu(z)
20
21 # Help function for flatten
22 def flatten(z, name='flatten'):
23     return tf.layers.flatten(z, name=name)
24
25 # Help function for dense layer
26 def dense(z, units=1, activation=None, name='dense'):
27     return tf.layers.dense(z, units=units, activation=activation, name=name)
28
29 # Help function for sigmoid

```

(continues on next page)

(续上页)

```

28 def sigmoid(z):
29     return tf.nn.sigmoid(z)
30
31 # Help function for printing trainable_variables
32 def show_all_variables():
33     model_vars = tf.trainable_variables()
34     slim.model_analyzer.analyze_vars(model_vars, print_info=True)
35
36 # Help function for reading image
37 def get_image(image_path, input_height, input_width, resize_height=64, resize_
38   ↴width=64, crop=True, grayscale=False):
39     image = imread(image_path, grayscale)
40     return transform(image, input_height, input_width, resize_height, resize_width,_
41   ↴crop)
42
43 # Help function for saving images
44 def save_images(images, size, image_path):
45     return imsave(inverse_transform(images), size, image_path)
46
47 # Help function for imread
48 def imread(path, grayscale=False):
49     if (grayscale):
50         return scipy.misc.imread(path, flatten=True).astype(np.float)
51     else:
52         return scipy.misc.imread(path).astype(np.float)
53
54 # Help function for merging images
55 def merge_images(images, size):
56     return inverse_transform(images)
57
58 # Help function for merging
59 def merge(images, size):
60     h, w = images.shape[1], images.shape[2]
61     if (images.shape[3] in (3, 4)):
62         c = images.shape[3]
63         img = np.zeros((h * size[0], w * size[1], c))
64         for idx, image in enumerate(images):
65             i = idx % size[1]
66             j = idx // size[1]
67             img[j * h:j * h + h, i * w:i * w + w, :] = image
68         return img
69     elif images.shape[3]==1:
70         img = np.zeros((h * size[0], w * size[1]))
71         for idx, image in enumerate(images):
72             i = idx % size[1]
73             j = idx // size[1]
74             img[j * h:j * h + h, i * w:i * w + w] = image[:, :, 0]
75         return img
76     else:
77         raise ValueError('in merge(images,size) images parameter must have dimensions:_'
78   ↴HxW or HxWx3 or HxWx4')
79
80 # Help function for imsave
81 def imsave(images, size, path):
82     image = np.squeeze(merge(images, size))
83     return scipy.misc.imsave(path, image)

```

(continues on next page)

(续上页)

```

81 # Help function for center_crop
82 def center_crop(x, crop_h, crop_w, resize_h=64, resize_w=64):
83     if crop_w is None:
84         crop_w = crop_h
85     h, w = x.shape[:2]
86     j = int(round((h - crop_h)/2.))
87     i = int(round((w - crop_w)/2.))
88     return scipy.misc.imresize(x[j:j+crop_h, i:i+crop_w], [resize_h, resize_w])
89
90
91 # Help function for transform
92 def transform(image, input_height, input_width, resize_height=64, resize_width=64,
93               crop=True):
94     if crop:
95         cropped_image = center_crop(image, input_height, input_width, resize_height,
96                                     resize_width)
97     else:
98         cropped_image = scipy.misc.imresize(image, [resize_height, resize_width])
99     return np.array(cropped_image)/127.5 - 1.
100
101
102 # Help function for inverse_transform
103 def inverse_transform(images):
104     return (images + 1.) / 2.
105
106
107 # Help function for calculating image_manifold_size
108 def image_manifold_size(num_images):
109     manifold_h = int(np.floor(np.sqrt(num_images)))
110     manifold_w = int(np.ceil(np.sqrt(num_images)))
111     assert manifold_h * manifold_w == num_images
112     return manifold_h, manifold_w

```

② discriminator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Discriminator:
6     def __init__(self):
7         pass
8
9     def forward(self, x, reuse=False, name='discriminator'):
10        pass

```

③ generator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Generator:
6     def __init__(self):
7         pass
8
9     def forward(self, x, reuse=False, name='generator'):
10        pass

```

④ gan.py

```

1 import tensorflow as tf
2 import numpy as np
3 import os
4 import glob
5 import time
6 from ops import *
7 from random import shuffle
8 from discriminator import Discriminator
9 from generator import Generator
10 from tensorflow.examples.tutorials.mnist import input_data
11
12 class GAN:
13     def __init__(self, img_shape, train_folder, sample_folder, model_folder,
14      ↪grayscale=False, crop=True, iterations=10000, lr_dis=0.0002, lr_gen=0.0002, beta1=0.
15      ↪5, batch_size=64, z_shape=100, sample_interval=100):
16         pass
17
18     def train(self):
19         pass
20
21     def load_dataset(self):
22         mnist = input_data.read_data_sets("./MNIST_data/", one_hot=True)
23         return mnist
24
25 if __name__ == '__main__':
26     pass

```

二. 逐步完善discriminator.py, generator.py以及gan.py。

① discriminator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Discriminator:
6     def __init__(self):
7         pass
8
9     def forward(self, x, reuse=False, name='discriminator'):
10        with tf.variable_scope(name, reuse=reuse):
11            z = tf.reshape(x, [-1, 28 * 28 * 1])
12
13            # Layer1
14            z = linear(z, output_size=128, name='d_linear_1')
15            z = relu(z)
16
17            # Layer2
18            z = linear(z, output_size=1, name='d_linear_2')
19            z = sigmoid(z)
20
21        return z

```

② generator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Generator:
6     def __init__(self):
7         pass
8
9     def forward(self, x, reuse=False, name='generator'):
10        with tf.variable_scope(name, reuse=reuse):
11            # Layer1
12            z = linear(x, output_size=128, name='g_linear_1')
13            z = relu(z)
14
15            # Layer2
16            z = linear(z, output_size=28 * 28 * 1, name='g_linear_2')
17            z = tf.reshape(z, [-1, 28, 28, 1])
18            z = sigmoid(z)
19
20        return z

```

(3) gan.py

```

1 import tensorflow as tf
2 import numpy as np
3 import os
4 import glob
5 import time
6 from ops import *
7 from random import shuffle
8 from discriminator import Discriminator
9 from generator import Generator
10 from tensorflow.examples.tutorials.mnist import input_data
11
12 class GAN:
13     def __init__(self, train_folder, sample_folder, model_folder, grayscale=False,
14      ↳ crop=True, iterations=10000, lr_dis=0.0002, lr_gen=0.0002, betal=0.5, batch_size=64,
15      ↳ z_shape=100, sample_interval=100):
16         if not os.path.exists(train_folder):
17             print("Invalid dataset path.")
18             return
19         if not os.path.exists(sample_folder):
20             os.makedirs(sample_folder)
21         if not os.path.exists(model_folder):
22             os.makedirs(model_folder)
23
24         self.height, self.width, self.channel = img_shape
25         self.train_folder = train_folder
26         self.sample_folder = sample_folder
27         self.model_folder = model_folder
28         self.grayscale = grayscale
29         self.crop = crop
30         self.iterations = iterations
31         self.lr_dis = lr_dis
32         self.lr_gen = lr_gen
33         self.betal = betal
34         self.batch_size = batch_size

```

(continues on next page)

(续上页)

```

33     self.z_shape = z_shape
34     self.sample_interval = sample_interval
35     self.discriminator = Discriminator()
36     self.generator = Generator()
37
38     # load dataset
39     self.X = self.load_dataset()
40
41     # placeholders
42     self.phX = tf.placeholder(tf.float32, [self.batch_size, self.height, self.
43     ↵width, self.channel], name='phX')
44     self.phZ = tf.placeholder(tf.float32, [self.batch_size, self.z_shape], name=
45     ↵'phZ')
46
47     # forward
48     self.gen_out = self.generator.forward(self.phZ, reuse=False)
49     self.dis_real = self.discriminator.forward(self.phX, reuse=False)
50     self.dis_fake = self.discriminator.forward(self.gen_out, reuse=True)
51     self.sampler = self.generator.forward(self.phZ, reuse=True)
52
53     # loss
54     self.d_loss = -tf.reduce_mean(tf.log(self.dis_real) + tf.log(1. - self.dis_
55     ↵fake))
56     self.g_loss = -tf.reduce_mean(tf.log(self.dis_fake))
57
58     # vars
59     train_vars = tf.trainable_variables()
60     self.dis_vars = [var for var in train_vars if 'discriminator' in var.name]
61     self.gen_vars = [var for var in train_vars if 'generator' in var.name]
62
63     # optimizer
64     self.dis_train = tf.train.AdamOptimizer(self.lr_dis, beta1=beta1).
65     ↵minimize(self.d_loss, var_list=self.dis_vars)
66     self.gen_train = tf.train.AdamOptimizer(self.lr_gen, beta1=beta1).
67     ↵minimize(self.g_loss, var_list=self.gen_vars)
68
69     def train(self):
70         self.sess = tf.Session()
71         self.sess.run(tf.global_variables_initializer())
72
73         saver = tf.train.Saver(max_to_keep=1)
74         savedir = self.model_folder
75
76         sample_z = np.random.uniform(-1, 1, size=(self.batch_size, self.z_shape))
77
78         for epoch in range(self.iterations):
79             batch_X, _ = self.X.train.next_batch(self.batch_size)
80             batch_X = np.reshape(batch_X, [-1, 28, 28, 1])
81             batch_Z = np.random.uniform(-1, 1, size=(self.batch_size, self.z_shape))
82             _, d_loss = self.sess.run([self.dis_train, self.d_loss], feed_dict={self.
83             ↵phX: batch_X, self.phZ: batch_Z})
84
85             batch_Z = np.random.uniform(-1, 1, size=(self.batch_size, self.z_shape))
86             _, g_loss = self.sess.run([self.gen_train, self.g_loss], feed_dict={self.
87             ↵phZ: batch_Z})

```

(continues on next page)

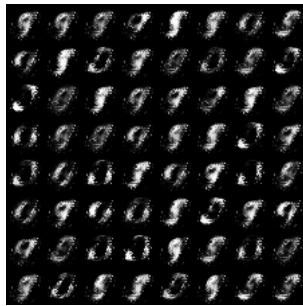
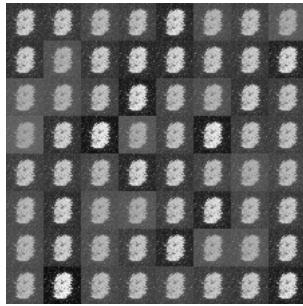
(续上页)

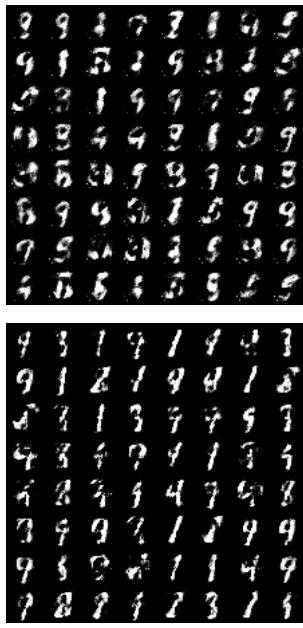
```

82         if epoch % 100 == 0:
83             print("Epoch: {}. D_loss: {}. G_loss: {}".format(epoch, d_loss, g_
84             loss))
85
86             samples = self.sess.run(self.sampler, feed_dict={self.phZ: sample_z})
87             save_images(samples, image_manifold_size(samples.shape[0]), '{}/{}.png'
88             .format(self.sample_folder, epoch))
89             saver.save(self.sess, "{}/gan.ckpt".format(self.model_folder), global_
90             step=epoch)
91
92     def load_dataset(self):
93         mnist = input_data.read_data_sets("./MNIST_data/", one_hot=True)
94         return mnist
95
96
97 if __name__ == '__main__':
98     img_shape = (28, 28, 1)
99     train_folder = "MNIST_data"
100    sample_folder = "samples"
101    model_folder = "models"
102    gan = GAN(train_folder=train_folder, sample_folder=sample_folder, model_
103    folder=model_folder)
104    gan.train()

```

三. 结果





注解：从左到右分别是0 epoch, 3100 epoch, 6100 epoch, 9100 epoch时的测试结果。

```

GAN_MNIST_basis [D:\Project\Pycharm_Project\GAN\GAN(GAN_MNIST_basis)\gan.py - PyCharm]
File Edit View Navigate Code Refactor Run Tools VCS Window Help
GAN_MNIST_basis gan.py
gan.py discriminator.py generator.py gan.py

if epoch % 100 == 0:
    print("Epoch: {}, D_loss: {}, G_loss: {}".format(epoch, d_loss, g_loss))

    samples = self.sess.run(self.sample, feed_dict={self.z_ph: sample_z})
    save_images(samples, image_manifold_size(samples.shape[0]), "{}\{}\{}.png".format(self.sample_folder, epoch))
    saver.save(self.sess, "{}\gan.ckpt".format(self.model_folder), global_step=epoch)

def load_dataset(self):
    mnist = input_data.read_data_sets("./MNIST_data/", one_hot=True)
    return mnist

if __name__ == '__main__':
    img_shape = (28, 28, 1)
    train_folder = "MNIST_data"
    sample_folder = "samples"
    model_folder = "models"
    gan = GAN(train_folder=train_folder, sample_folder=sample_folder, model_folder=model_folder)
    gan.train()

if __name__ == "__main__":
    gan()

```

Run: gan

```

Epoch: 0, D_loss: 0.19027861416339874, G_loss: 2.811112880706787
Epoch: 200, D_loss: 0.22008818387988523, G_loss: 3.140521764755249
Epoch: 400, D_loss: 0.31926144828796387, G_loss: 2.9524378776550293
Epoch: 600, D_loss: 0.43613615930090414, G_loss: 3.1612510661152344
Epoch: 800, D_loss: 0.341389089892276917, G_loss: 2.791691574859619
Epoch: 1000, D_loss: 0.21251830459641052, G_loss: 2.9138244556427
Epoch: 1200, D_loss: 0.20466065408799316, G_loss: 3.49243426322937
Epoch: 1400, D_loss: 0.2113511860370636, G_loss: 2.8459067344665527

```

Process finished with exit code 0

注解：运行结果

6.3 5.3 DCGAN

1. DCGAN论文: <https://arxiv.org/pdf/1511.06434>

2. 何之源 提供的Anime faces数据集下载: <https://pan.baidu.com/share/init?surl=eSifHcA> 提取码: g5qa
3. DCGAN的实现:

一共四个文件, ops.py, discriminator.py, generator.py以及dcgan.py。

一. 定义初始文件结构, ops.py, discriminator.py, generator.py以及dcgan.py。

① ops.py

```

1 import tensorflow as tf
2 import tensorflow.contrib.slim as slim
3 import scipy
4 import numpy as np
5
6 # Help function for creating convolutional layers
7 def conv2d(x, output_dim, filter_size=5, stride=2, padding='SAME', stddev=0.02, name=
8   ↪'conv2d'):
9     input_dim = x.get_shape().as_list()[-1]
10    with tf.variable_scope(name):
11        filter = tf.get_variable('w', [filter_size, filter_size, input_dim, output_
12   ↪dim], initializer=tf.truncated_normal_initializer(stddev=stddev))
13        conv = tf.nn.conv2d(x, filter=filter, strides=[1, stride, stride, 1],_
14   ↪padding=padding)
15        bias = tf.get_variable('bias', [output_dim], initializer=tf.constant_
16   ↪initializer(0.0))
17        conv = tf.nn.bias_add(conv, bias)
18    return conv
19
20
21 # Help function for creating deconvolutional layers
22 def deconv2d(x, filter_size, output_dim, stride, padding='SAME', stddev=0.02, name=
23   ↪'deconv2d'):
24     params = x.get_shape().as_list()
25     batch_size = params[0]
26     width = params[1]
27     height = params[2]
28     input_dim = params[-1]
29     with tf.variable_scope(name):
30         filter = tf.get_variable('w', [filter_size, filter_size, output_dim, input_
31   ↪dim], initializer=tf.random_normal_initializer(stddev=stddev))
32         deconv = tf.nn.conv2d_transpose(x, filter=filter, output_shape=[batch_size,_
33   ↪width * stride, height * stride, output_dim], strides=[1, stride, stride, 1],_
34   ↪padding=padding)
35         bias = tf.get_variable('bias', [output_dim], initializer=tf.constant_
36   ↪initializer(0.0))
37         deconv = tf.nn.bias_add(deconv, bias)
38    return deconv
39
40
41 def linear(x, output_size, name='linear', stddev=0.02):
42     shape = x.get_shape().as_list()
43     with tf.variable_scope(name):
44         matrix = tf.get_variable("matrix", [shape[1], output_size], tf.float32, tf.
45   ↪random_normal_initializer(stddev=stddev))
46         bias = tf.get_variable("bias", [output_size], initializer=tf.constant_
47   ↪initializer(0.0))
48         output = tf.matmul(x, matrix) + bias

```

(continues on next page)

(续上页)

```

35     return output
36
37 # Help function for batch_norm
38 def batch_norm(x, train=True, momentum=0.9, epsilon=1e-5, name="batch_norm"):
39     return tf.contrib.layers.batch_norm(x, decay=momentum, updates_collections=None,
40                                         epsilon=epsilon, scale=True, is_training=train, scope=name)
41
42 # Help function for relu
43 def relu(z):
44     return tf.nn.relu(z)
45
46 # Help function for leaky_relu
47 def leaky_relu(z):
48     return tf.nn.leaky_relu(z)
49
50 # Help function for sigmoid
51 def sigmoid(z):
52     return tf.nn.sigmoid(z)
53
54 # Help function for tanh
55 def tanh(z):
56     return tf.nn.tanh(z)
57
58 # Help function for residual block - identity
59 def identity_block(X, filter_sizes, output_dims, strides, stage, block, reuse=False,
60                     trainable=True):
61     block_name = 'res_identity_' + str(stage) + '_' + block
62
63     with tf.variable_scope(block_name, reuse=reuse):
64         X_shortcut = X
65
66         z = conv2d(X, filter_size=filter_sizes[0], output_dim=output_dims[0],
67                    stride=strides[0], padding='SAME', name=block_name + "_conv1")
68         z = batch_norm(z, name=block_name + "_bn1", train=trainable)
69         z = relu(z)
70
71         z = conv2d(z, filter_size=filter_sizes[1], output_dim=output_dims[1],
72                    stride=strides[1], padding='SAME', name=block_name + "_conv2")
73         z = batch_norm(z, name=block_name + "_bn2", train=trainable)
74         z = relu(z)
75
76         z = conv2d(z, filter_size=filter_sizes[2], output_dim=output_dims[2],
77                    stride=strides[2], padding='SAME', name=block_name + "_conv3")
78         z = batch_norm(z, name=block_name + "_bn3", train=trainable)
79
80         z = tf.add(X_shortcut, z)
81         z = relu(z)
82     return z
83
84 # Help function for printing trainable_variables
85 def show_all_variables():
86     model_vars = tf.trainable_variables()
87     slim.model_analyzer.analyze_vars(model_vars, print_info=True)
88
89 # Help function for reading image
90 def get_image(image_path, input_height, input_width, resize_height=64, resize_
91               width=64, crop=True, grayscale=False):

```

(continues on next page)

(续上页)

```

86     image = imread(image_path, grayscale)
87     return transform(image, input_height, input_width, resize_height, resize_width,_
88     ↪crop)
89
90 # Help function for saving images
91 def save_images(images, size, image_path):
92     return imsave(inverse_transform(images), size, image_path)
93
94 # Help function for imread
95 def imread(path, grayscale=False):
96     if (grayscale):
97         return scipy.misc.imread(path, flatten=True).astype(np.float)
98     else:
99         return scipy.misc.imread(path).astype(np.float)
100
101 # Help function for merging images
102 def merge_images(images, size):
103     return inverse_transform(images)
104
105 # Help function for merging
106 def merge(images, size):
107     h, w = images.shape[1], images.shape[2]
108     if (images.shape[3] in (3,4)):
109         c = images.shape[3]
110         img = np.zeros((h * size[0], w * size[1], c))
111         for idx, image in enumerate(images):
112             i = idx % size[1]
113             j = idx // size[1]
114             img[j * h:j * h + h, i * w:i * w + w, :] = image
115         return img
116     elif images.shape[3]==1:
117         img = np.zeros((h * size[0], w * size[1]))
118         for idx, image in enumerate(images):
119             i = idx % size[1]
120             j = idx // size[1]
121             img[j * h:j * h + h, i * w:i * w + w] = image[:, :, 0]
122         return img
123     else:
124         raise ValueError('in merge(images,size) images parameter must have dimensions:_'
125         ↪HxW or HxWx3 or HxWx4')
126
127 # Help function for imsave
128 def imsave(images, size, path):
129     image = np.squeeze(merge(images, size))
130     return scipy.misc.imsave(path, image)
131
132 # Help function for center_crop
133 def center_crop(x, crop_h, crop_w, resize_h=64, resize_w=64):
134     if crop_w is None:
135         crop_w = crop_h
136     h, w = x.shape[:2]
137     j = int(round((h - crop_h)/2.))
138     i = int(round((w - crop_w)/2.))
139     return scipy.misc.imresize(x[j:j+crop_h, i:i+crop_w], [resize_h, resize_w])
# Help function for transform

```

(continues on next page)

(续上页)

```

140 def transform(image, input_height, input_width, resize_height=64, resize_width=64,_
141   ↪crop=True):
142     if crop:
143       cropped_image = center_crop(image, input_height, input_width, resize_height,_
144         ↪resize_width)
145     else:
146       cropped_image = scipy.misc.imresize(image, [resize_height, resize_width])
147     return np.array(cropped_image)/127.5 - 1.
148
149 # Help function for inverse_transform
150 def inverse_transform(images):
151   return (images + 1.) / 2.
152
153 # Help function for calculating image_manifold_size
154 def image_manifold_size(num_images):
155   manifold_h = int(np.floor(np.sqrt(num_images)))
156   manifold_w = int(np.ceil(np.sqrt(num_images)))
157   assert manifold_h * manifold_w == num_images
158   return manifold_h, manifold_w
159
160 # Help function for cost function
161 def cost(logits, labels):
162   return tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=logits,_
163     ↪labels=labels))

```

② discriminator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Discriminator:
6   def __init__(self):
7     pass
8
9   def forward(self, x, momentum=0.9, df_dim=32, trainable=True, name='discriminator',
10   ↪, reuse=False):
11     pass

```

③ generator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Generator:
6   def __init__(self, img_shape):
7     self.width, self.height, self.channels = img_shape
8
9   def forward(self, x, momentum=0.9, gf_dim=64, trainable=True, name='generator',_
10   ↪reuse=False):
11     pass

```

④ dcgan.py

```
1 import tensorflow as tf
```

(continues on next page)

(续上页)

```

2 import numpy as np
3 import os
4 import glob
5 import time
6 from ops import *
7 from generator import Generator
8 from discriminator import Discriminator
9 from random import shuffle
10
11 class DCGAN:
12     def __init__(self, input_shape, output_shape, train_folder, sample_folder, model_
13     ↪folder, grayscale=False, crop=True, iterations=500, lr_dis=0.0002, lr_gen=0.0002,_
14     ↪beta1=0.5, batch_size=64, z_shape=128, sample_interval=100):
15         pass
16
17     def load_dataset(self):
18         x_imgs_name = glob.glob(os.path.join(self.train_folder, '*'))
19         return x_imgs_name
20
21     def train(self):
22         pass
23
24 if __name__ == '__main__':
25     pass

```

二. 逐步完善discriminator.py, generator.py以及dcgan.py。

① discriminator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Discriminator:
6     def __init__(self):
7         pass
8
9     def forward(self, x, momentum=0.9, df_dim=32, trainable=True, name='discriminator
10     ↪', reuse=False):
11         with tf.variable_scope(name) as scope:
12             if reuse:
13                 scope.reuse_variables()
14
15                 # Layer1
16                 z = leaky_relu(conv2d(x, filter_size=5, output_dim=df_dim, stride=2,_
17                 ↪padding='SAME', name='d_conv_1'))
18                 # Layer2
19                 # z = leaky_relu(batch_normalization(conv2d(z, filter_size=5, output_
20                 ↪dim=df_dim * 2, stride=2, padding='SAME', name='d_conv_2'), trainable,_
21                 ↪momentum=momentum, name='d_bn_2'))
22                 z = leaky_relu(batch_norm(conv2d(z, filter_size=5, output_dim=df_dim * 2,_
23                 ↪stride=2, padding='SAME', name='d_conv_2'), train=trainable, name='d_bn_2'))
24                 # Layer3
25                 # z = leaky_relu(batch_normalization(conv2d(z, filter_size=5, output_
26                 ↪dim=df_dim * 4, stride=2, padding='SAME', name='d_conv_3'), trainable,_
27                 ↪momentum=momentum, name='d_bn_3')) (continues on next page)

```

(续上页)

```

21     z = leaky_relu(batch_norm(conv2d(z, filter_size=5, output_dim=df_dim * 4,_
22     ↪stride=2, padding='SAME', name='d_conv_3'), trainable, name='d_bn_3'))
23     # Layer4
24     # z = leaky_relu(batch_normalization(conv2d(z, filter_size=5, output_
25     ↪dim=df_dim * 8, stride=2, padding='SAME', name='d_conv_4'), trainable,_
26     ↪momentum=momentum, name='d_bn_4'))
27     z = leaky_relu(batch_norm(conv2d(z, filter_size=5, output_dim=df_dim * 8,_
28     ↪stride=2, padding='SAME', name='d_conv_4'), trainable, name='d_bn_4'))
29     # Layer5
30     z = tf.reshape(z, [x.get_shape().as_list()[0], -1])
31     z = linear(z, output_size=1, name='d_linear_5')
32     return sigmoid(z), z

```

② generator.py

```

1 import tensorflow as tf
2 import numpy as np
3 from ops import *
4
5 class Generator:
6     def __init__(self, img_shape):
7         self.width, self.height, self.channels = img_shape
8
9     def forward(self, x, momentum=0.9, gf_dim=64, trainable=True, name='generator',_
10    ↪reuse=False):
11         with tf.variable_scope(name) as scope:
12             if reuse:
13                 scope.reuse_variables()
14             # Layer1
15             w = self.width // (2 ** 4)
16             h = self.height // (2 ** 4)
17             z = linear(x, output_size=gf_dim * 8 * w * h, name='g_linear_1')
18             z = tf.reshape(z, [-1, w, h, gf_dim * 8])
19             z = relu(batch_norm(z, trainable, name='g_bn_1'))
20
21             # Layer2
22             z = relu(deconv2d(z, filter_size=5, output_dim=gf_dim * 4, stride=2,_
23     ↪padding='SAME', name='g_deconv_2'))
24
25             # Layer3
26             z = relu(batch_norm(deconv2d(z, filter_size=5, output_dim=gf_dim * 2,_
27     ↪stride=2, padding='SAME', name='g_deconv_3'), trainable, momentum=momentum,_
28     ↪name='g_bn_3'))
29
30             # Layer4
31             z = relu(batch_norm(deconv2d(z, filter_size=5, output_dim=gf_dim * 1,_
32     ↪stride=2, padding='SAME', name='g_deconv_4'), trainable, momentum=momentum,_
33     ↪name='g_bn_3'))
34
35             # Layer5
36             z = relu(batch_norm(deconv2d(z, filter_size=5, output_dim=gf_dim // 2,_
37     ↪stride=2, padding='SAME', name='g_deconv_5'), trainable, momentum=momentum,_
38     ↪name='g_bn_3'))
39
40             # Layer6
41             z = conv2d(z, filter_size=7, output_dim=self.channels, stride=1, padding=_
42     ↪'SAME', name='g_conv_6')

```

(continues on next page)

(续上页)

```
34     return tanh(z)
```

(3) dcgan.py

```

1 import tensorflow as tf
2 import numpy as np
3 import os
4 import glob
5 import time
6 from ops import *
7 from generator import Generator
8 from discriminator import Discriminator
9 from random import shuffle
10
11 class DCGAN:
12     def __init__(self, input_shape, output_shape, train_folder, sample_folder, model_
13     ↪folder, grayscale=False, crop=True, iterations=500, lr_dis=0.0002, lr_gen=0.0002,_
14     ↪beta1=0.5, batch_size=64, z_shape=128, sample_interval=100):
15         if not os.path.exists(train_folder):
16             print("Invalid dataset path.")
17             return
18         if not os.path.exists(sample_folder):
19             os.makedirs(sample_folder)
20         if not os.path.exists(model_folder):
21             os.makedirs(model_folder)
22
23         self.in_width, self.in_height, self.in_channels = input_shape
24         self.out_width, self.out_height, self.out_channels = output_shape
25         self.train_folder = train_folder
26         self.sample_folder = sample_folder
27         self.model_folder = model_folder
28         self.grayscale = grayscale
29         self.crop = crop
30         self.iterations = iterations
31         self.lr_dis = lr_dis
32         self.lr_gen = lr_gen
33         self.beta1 = beta1
34         self.batch_size = batch_size
35         self.z_shape = z_shape
36         self.sample_interval = sample_interval
37         self.discriminator = Discriminator()
38         self.generator = Generator(output_shape)
39
40         # load dataset
41         self.X = self.load_dataset()
42
43         # placeholders
44         self.phX = tf.placeholder(tf.float32, [self.batch_size, self.out_width, self.
45         ↪out_height, self.out_channels], name='phX')
46         self.phZ = tf.placeholder(tf.float32, [self.batch_size, self.z_shape], name=
47         ↪'phZ')
48
49         # forward
50         self.gen_out = self.generator.forward(self.phZ, reuse=False)
51         self.dis_real, self.dis_real_logits = self.discriminator.forward(self.phX,_
52         ↪reuse=False)

```

(continues on next page)

(续上页)

```

48     self.dis_fake, self.dis_fake_logits = self.discriminator.forward(self.gen_out,
49     ↪ reuse=True)
50     self.sampler = self.generator.forward(self.phZ, reuse=True, trainable=False)
51
52     # loss
53     self.dis_loss_real = cost(self.dis_real_logits, tf.ones_like(self.dis_real))
54     self.dis_loss_fake = cost(self.dis_fake_logits, tf.zeros_like(self.dis_fake))
55     self.d_loss = self.dis_loss_fake + self.dis_loss_real
56     self.g_loss = cost(self.dis_fake_logits, tf.ones_like(self.dis_fake))
57
58     # vars
59     train_vars = tf.trainable_variables()
60     self.dis_vars = [var for var in train_vars if 'discriminator' in var.name]
61     self.gen_vars = [var for var in train_vars if 'generator' in var.name]
62
63     # optimizer
64     self.dis_train = tf.train.AdamOptimizer(self.lr_dis, beta1=beta1).
65     ↪ minimize(self.d_loss, var_list=self.dis_vars)
66     self.gen_train = tf.train.AdamOptimizer(self.lr_gen, beta1=beta1).
67     ↪ minimize(self.g_loss, var_list=self.gen_vars)
68
69     def load_dataset(self):
70         x_imgs_name = glob.glob(os.path.join(self.train_folder, '*'))
71         return x_imgs_name
72
73     def train(self):
74         run_config = tf.ConfigProto()
75         run_config.gpu_options.allow_growth = True
76         self.sess = tf.Session(config=run_config)
77         self.sess.run(tf.global_variables_initializer())
78
79         saver = tf.train.Saver(max_to_keep=1)
80         savedir = self.model_folder
81
82         counter = 0
83
84         sample_z = np.random.uniform(-1, 1, size=(self.batch_size, self.z_shape))
85         sample_files = self.X[0: self.batch_size]
86         sample = [get_image(sample_file, input_height=self.in_height, input_
87             ↪ width=self.in_width, resize_height=self.out_height, resize_width=self.out_width,
88             ↪ crop=self.crop, grayscale=self.grayscale) for sample_file in sample_files]
89         if (self.grayscale):
90             sample_inputs = np.array(sample).astype(np.float32)[:, :, :, None]
91         else:
92             sample_inputs = np.array(sample).astype(np.float32)
93
94         start_time = time.time()
95         for i in range(self.iterations):
96             batch_idxs = len(self.X) // self.batch_size
97             shuffle(self.X)
98             for idx in range(batch_idxs):
99                 batch_files = self.X[idx * self.batch_size: (idx + 1) * self.batch_
100                ↪ size]
101                 batch_X = [get_image(batch_file, input_height=self.in_height, input_
102                     ↪ width=self.in_width, resize_height=self.out_height, resize_width=self.out_width,
103                     ↪ crop=self.crop, grayscale=self.grayscale) for batch_file in batch_files]

```

(continues on next page)

```

96     if self.grayscale:
97         batch_X = np.array(batch_X).astype(np.float32)[:, :, :, None]
98     else:
99         batch_X = np.array(batch_X).astype(np.float32)
100
101    # batch_X = self.read_img(self.X)
102    batch_Z = np.random.uniform(-1, 1, (self.batch_size, self.z_shape)).\
103    astype(np.float32)
104    _, d_loss = self.sess.run([self.dis_train, self.d_loss], feed_dict=\
105    {self.phX: batch_X, self.phZ: batch_Z})
106    _, g_loss = self.sess.run([self.gen_train, self.g_loss], feed_dict=\
107    {self.phZ: batch_Z})
108
109    print("Epoch:{} {}/. Time: {}. Discriminator loss: {}. Generator_\
110 loss: {}".format(i, idx, batch_idxs, time.time() - start_time, d_loss, g_loss))
111    if counter % self.sample_interval == 0:
112        samples, d_loss, g_loss = self.sess.run([self.sampler, self.d_\
113 loss, self.g_loss], feed_dict={self.phZ: sample_z, self.phX: sample_inputs})
114        save_images(samples, image_manifold_size(samples.shape[0]), '{}/{}\
115 .png'.format(self.sample_folder, counter))
116        saver.save(self.sess, "{}/dcgan.ckpt".format(self.model_folder), \
117 global_step=counter)
118        counter += 1
119
120 if __name__ == '__main__':
121     input_shape = (96, 96, 3)
122     output_shape = (96, 96, 3)
123     train_folder = "D:/Project/Python_Project/21 deep learning/chapter_8/faces/"
124     sample_folder = "samples"
125     model_folder = "models"
126     dcgan = DCGAN(input_shape=input_shape, output_shape=output_shape, train_\
127 folder=train_folder, sample_folder=sample_folder, model_folder=model_folder)
128     dcgan.train()

```

三. 结果

6.4 5.4 LSGAN

6.5 5.5 WGAN

6.6 5.6 CycleGAN

6.7 5.7 CGAN

6.8 5.8 StarGAN