

基于 keras 的 GAN 实现 mnist 生成

一、 项目目的

1. 掌握 GAN 生成图片的原理及其优缺点。
2. 掌握 GAN 算法步骤和基于 keras 实现的方法。

二、 项目框架

Windows + anaconda3 + python3.6 + tensorflow-gpu

三、 项目原理

GAN 全称为 Generative Adversarial Networks（中文名为生成式对抗网络）。最早由 Ian J. Goodfellow 等人于 2014 年 10 月在《》中提出，现在已经发展出 DCGAN、WGAN 等优化模型。可应用于图像生成，数据增强等。

（一）综述

GAN 由两个部分组成：Generator 和 Discriminator（以下简称 G 和 D）。G 的输入是噪声，输出是图片数组；D 的输入是图片数组，输出是一个数值。在 GAN 的内部，G 生成图片数组，然后经由 D 对其给出一个分数评判，分数的高低表示 D 认为图片是来自真实的数据集（也将真实数据集的图片喂养 D）还是来自 G。G 利用 D 给出的分数值进行优化，使得产生的图片更接近数据集。D 则利用 G 产生的图像数组进行训练来达到更好的判别效果。两者互相对抗，最终 D 无法区别其输入是来自 G 还是真实的数据集，此时 G 生成的图片就能达到以假乱真的结果。

(二) 理论推导

Generation

1. 目标：找出真实数据集的分布 $P_{data}(x)$ ，使得 $P_G(x, \theta)$ 与其最大似然（Maximum Likelihood Estimation）。 x 是一张图片（一个高维向量）。而最大似然估计等价于最小化 $P_{data}(x)$ 与 $P_G(x, \theta)$ 的 KL 散度：

Maximum Likelihood Estimation=Minimize KL Divergence

即 $\theta^* = \arg \min_{\theta} KL(P_{data} || P_G)$ 。于是对 G 而言，就是在输入为随机正态分布下，输出的 $P_G(x)$ 要和 $P_{data}(x)$ 尽可能接近。即

$$G^* = \arg \min_G Div(P_G, P_{data})$$

2. 对 D 而言，要区分 x 是来自 $P_G(x)$ 还是 $P_{data}(x)$ ，就像是训练一个二分类器。目标函数：

$$V(G, D) = E_{x \sim P_{data}} [\log D(x)] \uparrow + E_{x \sim P_G} [\log(1 - D(x))] \downarrow$$

对来自 P_{data} 的 x 给高分，对来自 P_G 的 x 给低分。于是对的训练就是最大化目标函数 $V(G, D)$ 。即：

$$D^* = \arg \max_D V(G, D)$$

3. 对目标函数展开可得

$$V = \int_x [P_{data}(x) \log D(x) + P_G(x) [\log(1 - D(x))]] dx$$

G 固定的情况下，要找到 D^* （最优的 D）：上式去掉积分符号后对 $D(x)$ 求微分，令其等于 0，可解得

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

其值介于 0 到 1 之间。

4. 于是 $\max_D V(G, D) = V(G, D^*)$ ，将 D^* 代入目标函数 V 进行转化，可得 $\max_D V(G, D) = -2 \log 2 + 2JSD(P_{data} || P_G)$ 。

可以得出结论：求解 $\max_D V(G, D)$ 的过程，实际上就是在计算 P_{data} 和 P_G 之间的某种散度。

5. 从而我们回到最初要解决的问题：

$$G^* = \arg \min_G Div(P_G, P_{data})$$

转换成了：

$$G^* = \arg \min_G \max_D V(G, D)$$

6. 在每一轮的训练中，我们需要做到：

Step1: 固定 G ，最优化 D

Step2: 固定 D ，最优化 G

(三) 算法实践

每次训练：

I 学习 K 次 D ：

1. 从 $P_{data}(x)$ 中采样出 m 个图片 $\{x^1, x^2, \dots, x^m\}$
2. 从一个高斯分布中采样出 m 个例子 $\{z^1, z^2, \dots, z^m\}$ ，并经过 G ，得到 $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$ ， $\tilde{x}^i = G(z^i)$
3. 将 m 个 x^i 和 m 个 \tilde{x}^i 代入目标函数 V ，通过更新参数 θ_d 来最大化：

$$\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$$

$$\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$$

II 学习 1 次 G:

1. 从一个高斯分布中采样出 m 个例子 $\{z^1, z^2, \dots, z^m\}$, 并经过 G , 得到 $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
2. 通过更新参数 θ_d 来最小化:

$$\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^i)))$$

$$\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$$

四、 实现步骤

(一) 数据准备

在 `mnist` 官网下载数据集, 解压后放在 `py` 文件的同级路径下的 `mnist` 文件夹中。





(G:) > 2018.9 > Test_GAN > mnist			
名称	修改日期	类型	大小
 t10k-images.idx3-ubyte	1998/1/26 23:07	IDX3-UBYTE 文件	7,657 KB
 t10k-labels.idx1-ubyte	1998/1/26 23:07	IDX1-UBYTE 文件	10 KB
 train-images.idx3-ubyte	1996/11/18 23:36	IDX3-UBYTE 文件	45,938 KB
 train-labels.idx1-ubyte	1996/11/18 23:36	IDX1-UBYTE 文件	59 KB

Figure 1 mnist 数据集

(二) GAN 网络的搭建

1. 导入必要的包

```
from keras.models import Sequential
from keras.layers import
Dense, Activation, BatchNormalization, Reshape, UpSampling2D, \
```

```

Conv2D, MaxPooling2D, Flatten
import numpy as np
from keras.optimizers import SGD
from keras.datasets import mnist
import math
from PIL import Image

```

2. 构造 Generator

```

def Generator_model():
    model = Sequential()
    model.add(Dense(input_dim=100, output_dim=1024))
    model.add(Activation('tanh'))
    model.add(Dense(128*7*7))
    model.add(BatchNormalization())
    model.add(Activation('tanh'))
    model.add(Reshape((7, 7, 128), input_shape=(128, 7, 7)))
    model.add(UpSampling2D(size=(2, 2)))
    model.add(Conv2D(64, (5, 5), padding='same'))
    model.add(Activation('tanh'))
    model.add(UpSampling2D(size=(2, 2)))
    model.add(Conv2D(1, (5, 5), padding='same'))
    model.add(Activation('tanh'))
    return model

```

3. 构造 Discriminator

```

def Discriminator_model():
    model=Sequential()
    model.add(
        Conv2D(64, (5, 5),
            padding='same',
            input_shape=(28, 28, 1)))
    model.add(Activation('tanh'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(128, (5, 5)))
    model.add(Activation('tanh'))
    model.add(Flatten())
    model.add(Dense(1024))
    model.add(Activation('tanh'))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))
    return model

```

4. 拼接成 GAN

```
def GAN_model(g, d):
    model=Sequential()
    #先添加 Generator, 再令 Discriminator 不可训练 (即固定 d)
    model.add(g)
    d.trainable=False
    model.add(d)
    return model
```

(三) 定义函数将生成的每张图片进行拼接

```
def Combine_image(generated_images):
    num = generated_images.shape[0]
    width = int(math.sqrt(num))
    height = int(math.ceil(float(num)/width))
    shape = generated_images.shape[1:3]
    image = np.zeros((height*shape[0],width*shape[1]),
                      dtype = generated_images.dtype)
    for index, img in enumerate(generated_images):
        i = int(index/width)
        j = index%width
        image[i*shape[0]:(i+1)*shape[0],
              j*shape[1]:(j+1)*shape[1]]=img[:, :, 0]
    return image
```

(四) 训练 GAN

```
def Train(Batch_size):
    (X_train,y_train),(X_test,y_test)=mnist.load_data()
    X_train = (X_train.astype(np.float32)-127.5)/127.5
    X_train = X_train[:, :, :, None]
    x_test = X_test[:, :, :, None]
    g = Generator_model()
    d = Discriminator_model()
    d_on_g = GAN_model(g, d)
    d_optim = SGD(lr=0.001, momentum=0.9, nesterov=True)
    g_optim = SGD(lr=0.001, momentum=0.9, nesterov=True)
    g.compile(loss='binary_crossentropy', optimizer="SGD")
    d_on_g.compile(loss='binary_crossentropy', optimizer=g_optim)
    d.trainable = True
    d.compile(loss='binary_crossentropy', optimizer=d_optim)

    for epoch in range(30):
        print("Epoch is", epoch)
        print("Number of batches is
", int(X_train.shape[0]/Batch_size))
        for index in range(int(X_train.shape[0]/Batch_size)):
```

```

noise = np.random.uniform(-1,1, size=(Batch_size, 100))
image_batch = X_train[index * Batch_size:(index + 1) *
Batch_size]
generated_images = g.predict(noise, verbose=0)
if index % 100 == 0:
    image = Combine_image(generated_images)
    image = image * 127.5 + 127.5
    Image.fromarray(image.astype(np.uint8)).save(
        ". /GAN_Picture_train/" + str(epoch) + "_" +
str(index) + ".png"
    )
X = np.concatenate((image_batch, generated_images))
y = [1] * Batch_size + [0] * Batch_size
d_loss = d.train_on_batch(X, y)
print("batch %d d_loss : %f" % (index, d_loss))
noise = np.random.uniform(-1,1, (Batch_size, 100))
d.trainable = False
g_loss = d_on_g.train_on_batch(noise, [1]*Batch_size)
d.trainable = True
print("batch %d g_loss : %f" % (index, g_loss))
if index%100 == 9:
    g.save_weights('generator_weight', True)
    d.save_weights('discriminator_weight', True)

```

(五) 生成图片

```

def Generate(Batch_size, nice=False):
    g = Generator_model()
    g.compile(loss='binary_crossentropy', optimizer="SGD")
    g.load_weights('generator_weight')
    if nice:
        d = Discriminator_model()
        d.compile(loss='binary_crossentropy', optimizer="SGD")
        d.load_weights('discriminator_weight')
        noise = np.random.uniform(-1, 1, (Batch_size*20, 100))
        generated_images = g.predict(noise, verbose=1)
        d_pret = d.predict(generated_images, verbose=1)
        index = np.arange(0, Batch_size*20)
        index.resize((Batch_size*20, 1))
        pre_with_index = list(np.append(d_pret, index, axis=1))
        pre_with_index.sort(key=lambda x:x[0], reverse=True)
        nice_images =
np.zeros((Batch_size,)+generated_images.shape[1:3], dtype=np.float32)
        nice_images = nice_images[:, :, :, None]
        for i in range(Batch_size):

```

```

        idx = int(pre_with_index[i][1])
        nice_images[i,:,:0] = generated_images[idx,:,:0]
    image = Combine_image(nice_images)
else:
    noise = np.random.uniform(-1, 1, (Batch_size, 100))
    generated_images = g.predict(noise, verbose=0)
    image = Combine_image(generated_images)
image = image*127.5+127.5
Image.fromarray(image.astype(np.uint8)).save(
    "./GAN_Picture/Generate_image.png"
)

```

五、 项目结果



Figure 2 GAN 生成 mnist 手写字