# 本项目使用的是 Tensorflow2.0！

# generate 方法需要传入已经归一化后的数据！

## 一、项目结构

### 1、generate.py 代码结构

```
import numpy as np
from tensorflow import keras
import random
import copy
from keras.datasets import fashion_mnist
from keras.utils import np_utils

model = keras.models.load_model("model.hdfs")
tapic_img_list=[0,0,0,0,0,0,0,0,0,0]
tapic_tag_list=[0,0,0,0,0,0,0,0,0,0]
methodlist=[0,0,0,0,0]
def load_data():...
def select_tapic_imgs(test_imgs):...
def SSIM(img1 ,img2):...
def average(img):...
def deviation(img, av):...
def covariance(img1, av1, img2, av2):...
def diagonal_switch_attack(test_img) :...
def random_switch_attack(test_img):...
def tapic_attack(test_img):...
def attack_img(test_img,ssimsum,generate_images):...
def generate(images,shape):...
```

load_data:此方法用于加载 fashionmnist 数据集并将其转化为（10000，28，28，1）的形状

select_tapic_imgs：从传入的 images 中选出待测模型认为的各个种类最具代表性（每类图片 predict 返回的概率最高）的图片，便于后期 tapic_attack 使用

SSIM：用于计算两张图片的平均结构相似性

average：计算图片像素平均值

deviation：计算两张图片方差

covariance：计算协方差

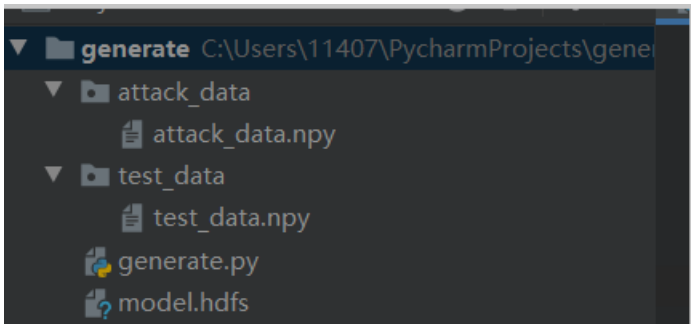
diagonal_switch_attack：第一层攻击

random_switch_attack：第二层攻击

tapic_attack：第三层攻击

generate：供外部调用的方法入口，传入 images 和 shape，返回 images 对应的对抗样本集

**2、文件结构**




generate 项目下包含两个文件夹：attack_data 和 test_data，一个 generate.py 文件，一个模型 model.hdfs

attack_data 文件夹下包含 attack_data.npy 文件，其中包含对 fashionmnist 数据集中的测试集的 10000 张图片生成的对抗样本

test_data 下包含 test_data.npy 文件中包含 fashionmnist 数据集中的测试集的 10000 张图片

模型 model.hdfs 直接放在项目 generate 的根目录下，与 generate.py 同级

attack_data.npy 和 test_data.npy 中存储的数据的 shape 均为（10000，28，28，1）

## 二、 项目介绍

**1 模型调用方法**

generate()封装了模型的调用，本段介绍模型如何运行

**1.1 加载模型**

```
model = keras.models.load_model("model.hdfs")
```

**1.2 使用模型预测**

```
model = keras.models.load_model("model.hdfs")
res= model.predict(test_image)
print("the shape of parameter(test_image) is "+str(test_image.shape))
print("the return of the model.predict(test_image) is "+str(res))
```

```
the shape of parameter(test_image) is (1, 28, 28, 1)
the return of the model.predict(test_image) is [[3.4792357e-08 9.3747269e-07 7.2123150e-08 9.0152987e-08 9.2327518e-08
  5.7338948e-06 1.5480874e-06 1.6150283e-05 7.1972219e-09 9.9997532e-01]]
```

predict()的参数是一张图片，shape 为（1，28，28，1）

返回值是一个一维 ndarray，如[[3.4792357e-08 9.3747269e-07 7.2123150e-08 9.0152987e-08 9.2327518e-08 5.7338948e-06 1.5480874e-06 1.6150283e-05 7.1972219e-09 9.9997532e-01]]

共十个数字，表示模型预测这张图片属于 0~9 这十个种类的概率

结合使用 np.argmax()可以获得模型预测的图片的种类（0~9 表示）

predict_res = np.argmax(model.predict(test_img, 0))

**2 generate（）使用方法**

**2.1 包的导入**

在调用 generate()方法的脚本中需要使用 import generate 完整导入，只使用 from generate import generate 会出错

**2.2 generate（）方法的调用**

```
shape1=(10000,28,28,1)
generate.generate(test_images,shape1)
```

此方法接受两个参数，test_images 是需要生成对抗样本的数据集，shape 是数据集对应的 shape，传入的数据集的 shape 可以是（n，28，28，1）或者（n，28，28）或者（n，784），与数据集对应

返回值是对 test_images 生成的对抗样本集，返回值的 shape 同于 test_images 的 shape

**3 对抗样本生成时间**

```
totally cost 3080.995425224304
```

10000 个对抗样本总共花费 3080 秒生成

## 三、 算法详解

**1- one-pixel-attack-keras**

主 要 参 考 资 料 ： https://github.com/Hyperparticle/one-pixel-attack-keras/blob/master/1_one-pixel-attack-cifar10.ipynb

个人理解：这个算法的核心思想就是只改动一个点，逐渐修改这个点的像素值产生扰动，与母图像对比，选择攻击效果最好的子图像，进行对抗攻击，难度在于选择哪个像素点，改成什么颜色，上述资料选择了差分进化算法，简言之就是先随机选，然后将攻击结果最好的保留替换父代，作为目前最优解，重复迭代获得结果。参考上诉参考资料，使用其提供的现成的差分进化算法，生成对抗样本，攻击成功率仅在 40 左右，而且我自己无法写出差分进化算法，这个研究最久的攻击方式最终被我舍弃，未出现我提交的代码中。

2- 结合 one-pixel-attack 算法的造成尽量少的扰动的思想和 angri 算法生成图像特定的扰动的思想，我设计了三层的攻击方式。

参考 SSIM 的计算方式，为了减少攻击后图片亮度的变化，我尽量不改动原图片的像素平均值，为了实现这一点，在第一二层攻击中所有的扰动都未改动图片的像素平均值，在均值不变的前提下， 如果对比度（即图片像素方差方差）也不变那么 ssim 将会较高，能同时实现这两点的最简单的攻击方式就是交换图片上不同位置的点，同时限制交换的次数能有效降低协方差（即结构变化）对 ssim 数值的影响。

前期对 fashionmnist 的研究中发现这个数据集基本都是对时装衣物，鞋，包的图片，我觉得这一类图片普遍具有对称性，所以我在第一层采用对称交换点的方式，这样对抗样本和原图

片的相似度很高（经测试均值在 0.98 左右），但同样由于对称性，攻击成功率也不高，所以设计了第二层采用分区随机换点，限制换点的数量和换点的区域，这两层加起来攻击成功率在 90 百分比左右。
考虑到一类图像有共同点，我先找出了模型认为的各个类的典型图片（每个种类模型预测出的概率最高的那张图片），在第三层生成对抗样本时，我们将图片与每个种类的典型图片计算结构相似度，寻找相似度最高但不是同一类的那张典型图片，将图片中的点逐渐置换成典型图片的点，直到出错。第三层的存在保证了一定能攻击成功，不过对抗样本和原样本的结构相似度可能不高，但是考虑到实际运行过程中只有大概 10 百分比左右，可以接受结构相似度上的损失。
结合这三层能做到大量样本百分之攻击成功，但是由于第三层的典型图片是从输入到 generate 方法中的 images 中寻找的，如果传入 generate 方法的样本图片很少，不是每个类都能找到典型图片，则可能会出现结构相似度较低甚至攻击失败的情况。
对 fashionmnist 的测试集中的 10000 样本生成的对抗样本和原样本的平均 SSIM 在百分之 76 左右。

## 四、 个人感受

深度学习应用广泛，但是少量像素的改动，肉眼不可察觉的改动就能使深度学习模型失效，实在神奇，值得深究。
本次实验过程中我最大的收获就是对深度学习和神经网络，对抗样本等概念有了初步的了解，懂了的大致该如何做深度学习测试。
最大的难点我觉得是 tensorflow 的安装和使用，tensorflow2.0 正式发布时间较短，市面上的教程很少，安装和使用的过程存在大量的坑，陆陆续续花了三天才将环境配好，找靠谱的 2.0 的教程也花了将近一晚上。还有就是网上黑盒对抗样本的算法就只能找到几种（one-pixel-attack,upset and angri, Houdini）而且大多是贴论文和数学公式，晦涩难懂，需要看论文才能了解大体思想。

## 五、 参考资料

参考网址：
https://www.cnblogs.com/shona/p/11277740.html
https://github.com/Hyperparticle/one-pixel-attack-keras/blob/master/1_one-pixel-attack-cifar10.ipynb
https://blog.csdn.net/qq_35414569/article/details/81084585
https://blog.csdn.net/qq_35414569
参考文献：
[1]Sayantan Sarkar, Ankan Bansal, Upal Mahbub, Rama Chellappa. UPSET and ANGRI : Breaking High Performance Image Classifiers. arXiv:1707.01159 [cs.CV]
[2]Moustapha Cisse, Yossi Adi, Natalia Neverova, Joseph Keshet. Houdini: Fooling Deep Structured Prediction Models. arXiv:1707.05373 [stat.ML]
[3] Jiawei Su, Danilo Vasconcellos Vargas, Sakurai Kouichi. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation}, Vol.23 , Issue.5 , pp. 828--841. Publisher: IEEE. 2019