AutomaticTest

项目介绍

使用库及版本

- tensorflow 2.0.0
- numpy 1.17.4

代码结构

- attack_data
  - ./data/attack_data.npy
- test_data
  - ./data/test_data.npy
- generate.py
  - 在项目根目录下
- train.py
  - 用于训练模型的文件
- test.py
  - 用于测试 generate.py
- 自训练模型路径为： "./models/model.ht"
  - 给出 shape为(n,28,28,1)的 image
  - trained_model= tf.keras.models.load_model(model_path)
  - predicts= trained_model.predict(images) //images 的类别可能性矩阵 (n,10)
  - predict_types = tf.argmax(predicts,1) //images的类别 0~9 (n,)

生成对抗样本的时间

test_data为 fashion_minist的测试集，共 10000张，最终算法针对 test_data生成 attack_data 的时间约为 900秒左右

算法详解

算法 1

具体过程

- 首先提供一个自训练的分类模型 M，输入图片集 images。

- 用模型 M 预测出 images 的类别列表 types
- 对 images 中的每一张图片 image
- 用模型 M 预测 image 的类别 type
- 利用 FGSM 算法根据给出的 type 生成噪声图像 perturbation
- 从 0 开始逐渐增大变异系数ε至上限 eps_max=0.2
  - 用ε乘以 perturbation 再加上 image 生成对抗样本 adv
  - 用模型 M 预测出 adv 的类别 adv_type，和可能性 confidence
  - 如果 adv_type 和 type 不同，而且 confidence 大于下限 conf_min=0.3，则认为 adv 是成功的对抗样本
- 对未成功生成对抗样本的图片返回原图片

思考 1

- 倒数第二步中可以去掉判断可能性 confidence 的步骤，或者降低 conf_min，可以提高对抗样本和原图片的相似度，但是可能会降低对抗样本在其它模型上的攻击成功率。
- 算法 1 在实际运行中大量的时间耗费在了模型对图片的预测过程，而且图片量过大会出现内存不足的情况，跑了超过 4 个小时也没有给出结果。

算法 2

实验发现，模型预测 10000 张图片花费的时间和预测 1 张图片的时间几乎相同。所以将两层循环的次序换了一下，遍历ε，然后一次性预测所有图片的类别，效率可以提高几千倍。

思考 2

- 在最后一步中，如果对未成功生成对抗样本的图片返回一张空图片或者是随机给出一个和预测类别不同的图片，最终结果和原图片的 ssim 会大大降低，但是攻击的成功率会提高很多。两种方法下 ssim 和成功率的乘积大致相同。

个人感受

- 我认为难点是理解对抗样本的生成过程，以及程序性能的优化
- 个人收获是了解了对抗样本的生成，以及分类模型的训练

参考文献

- https://tensorflow.google.cn/tutorials
- https://blog.csdn.net/nemoyy/article/details/81052301
- https://blog.csdn.net/weiqi_fan/article/details/88213284
- https://www.jianshu.com/p/96653fe0c74f