中国科学技术大学　微电子学院

《神经网络及其应用》实验报告四

姓名　杨翊麟　　学号　BC24219010

时间　2025/04/25　　指导教师　徐奇

实验名称：使用 FGSM 对 Facenet 模型进行对抗攻击

一、实验目的

1、了解对抗学习（Adversarial Learning）主要通过生成对抗样本来帮助提高神经网络模型的鲁棒性和泛化能力；
2、熟悉 FGSM 利用梯度快速计算扰动的攻击原理；
3、使用 FGSM 对 Facenet 模型进行对抗攻击，误导网络模型对人脸识别的能力，实现“白盒非定向规避攻击”，并验证攻击效果。

二、实验准备

1、**数据准备：**LFW - People（Labeled Faces in the Wild）Dataset
2、**数据预处理：**对齐 LFW 数据集（对齐后图片尺寸：160*160）
3、**实验环境：**Python 版本：3.6，使用框架：**Cleverhans**、**Tensorflow**（1.7.0）、**Facenet**
4、**攻击方法：FGSM（Fast Gradient Sign Method）**，公式如下：

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

其中，x_{adv}: 对抗图像，x: 原始输入图像，y: 原始输入标签，ε: 扰动因子，θ: 模型参数，J: 损失
5、**结果分析方法：**1）**Dodging**（**逃避攻击**），攻击目标是让同一个人不被识别成自己；2）**Impersonation**（**冒充攻击**），攻击目标是让不同人被识别为同一个人。分别比较两种攻击后的模型识别准确率。

三、实验流程分析（facenet_fgsm.py）

（1）库文件导入

```
# 导入facenet的加载模块
import facenet_fn
import tensorflow as tf
import numpy as np
```

```
# 引入CleverHans的Model抽象类和FGSM攻击方法
from cleverhans.model import Model
from cleverhans.attacks import FastGradientMethod

# 自定义的测试集加载模块
import set_loader
```

（2）数据准备与预处理

加载 Facenet 预训练模型，计算输入图像和目标图像特征向量之间的欧氏距离并转换为 softmax 格式输出，保存成字典格式按 Cleverhans 框架要求返回

```
# 定义用于封装Facenet模型的类
class InceptionResnetV1Model(Model):
    # 模型的预训练权重文件路径
    model_path = "models/20180402-114759/20180402-114759.pb"

    def __init__(self):
        # 调用父类构造方法，定义作用域
        super(InceptionResnetV1Model, self).__init__(scope="model")
        # 加载Facenet预训练模型
        facenet_fn.load_model(self.model_path)
        # 获取当前默认计算图
        graph = tf.get_default_graph()
        # 提取输入张量（图片输入）
        self.face_input = graph.get_tensor_by_name("input:0")
        # 提取输出张量（512维的特征嵌入）
        self.embedding_output =
graph.get_tensor_by_name("embeddings:0")

    def convert_to_classifier(self):
        # 定义一个占位符用于输入目标图像的特征向量（嵌入）
        self.victim_embedding_input = tf.placeholder(tf.float32,
shape=(None, 512))
        # 计算输入图像和目标图像嵌入之间的欧氏距离
        distance = tf.reduce_sum(
            tf.square(self.embedding_output -
self.victim_embedding_input), axis=1
        )
        # 将欧氏距离转换为"分类概率"形式（模拟分类器）
        threshold = 0.99  # Facenet默认相似度阈值
        score = tf.where(
            distance > threshold,
            0.5 + ((distance - threshold) * 0.5) / (4.0 - threshold),
```

```
            0.5 * distance / threshold,
        )
        reverse_score = 1.0 - score
        # 构造softmax格式输出（2分类）
        self.softmax_output = tf.transpose(tf.stack([reverse_score,
score]))
        # 保存softmax输出作为"logits"层
        self.layer_names = []
        self.layers = []
        self.layers.append(self.softmax_output)
        self.layer_names.append("logits")

    # CleverHans要求实现fprop接口返回层字典
    def fprop(self, x, set_ref=False):
        return dict(zip(self.layer_names, self.layers))
```

（3）定义 FGSM 攻击器参数，执行 FGSM 攻击

对 LFW 数据集图片执行 FGSM 攻击，分别计算逃避攻击（**Dodging**）的成功率和冒充攻击（**Impersonation**）的成功率，保存原图和对抗样本图像

```
# 创建默认计算图
with tf.Graph().as_default():
    with tf.Session() as sess:
        # 实例化模型并转换为分类器结构
        model = InceptionResnetV1Model()
        model.convert_to_classifier()

        # 加载用于测试的人脸图像对，以及对应的标签（0-同人，1-不同人）
        size = 100  # 加载100对
        faces1, faces2, labels = set_loader.load_testset(size)

        # 使用facenet模型提取faces2的特征嵌入，作为攻击目标
        graph = tf.get_default_graph()
        phase_train_placeholder =
graph.get_tensor_by_name("phase_train:0")
        feed_dict = {model.face_input: faces2,
phase_train_placeholder: False}
        victims_embeddings = sess.run(model.embedding_output,
feed_dict=feed_dict)

        # 定义FGSM攻击器参数
        steps = 1
        eps = 0.01  # 总扰动大小
        alpha = eps / steps  # 每步扰动（这里只有一步）
```

```
        fgsm = FastGradientMethod(model)
        fgsm_params = {"eps": alpha, "clip_min": 0.0, "clip_max":
1.0}
        adv_x = fgsm.generate(model.face_input, **fgsm_params)

        # 执行FGSM攻击
        adv = faces1  # 初始化为原始图像
        for i in range(steps):
            print("FGSM step " + str(i + 1))
            feed_dict = {
                model.face_input: adv,
                model.victim_embedding_input: victims_embeddings,
                phase_train_placeholder: False,
            }
            adv = sess.run(adv_x, feed_dict=feed_dict)

        # 测试模型在原始图像上的识别准确率
        batch_size = graph.get_tensor_by_name("batch_size:0")
        feed_dict = {
            model.face_input: faces1,
            model.victim_embedding_input: victims_embeddings,
            phase_train_placeholder: False,
            batch_size: 64,
        }
        real_labels = sess.run(model.softmax_output,
feed_dict=feed_dict)
        accuracy = np.mean(
            (np.argmax(labels, axis=-1)) == (np.argmax(real_labels,
axis=-1))
        )
        print("Accuracy: " + str(accuracy * 100) + "%")

        # 测试模型在对抗样本上的识别准确率
        feed_dict = {
            model.face_input: adv,
            model.victim_embedding_input: victims_embeddings,
            phase_train_placeholder: False,
            batch_size: 64,
        }
        adversarial_labels = sess.run(model.softmax_output,
feed_dict=feed_dict)

        # 计算同一个人的对抗成功率（逃避攻击）
        same_faces_index = np.where((np.argmax(labels, axis=-1) ==
```

```
0))
        accuracy = np.mean(
            (np.argmax(labels[same_faces_index], axis=-1))
            == (np.argmax(adversarial_labels[same_faces_index],
axis=-1))
        )
        print(
            "Accuracy against adversarial examples for "
            + "same person faces (dodging): "
            + str(accuracy * 100)
            + "%"
        )

        # 计算不同人的对抗成功率（冒充攻击）
        different_faces_index = np.where((np.argmax(labels, axis=-1)
== 1))
        accuracy = np.mean(
            (np.argmax(labels[different_faces_index], axis=-1))
            == (np.argmax(adversarial_labels[different_faces_index],
axis=-1))
        )
        print(
            "Accuracy against adversarial examples for "
            + "different people faces (impersonation): "
            + str(accuracy * 100)
            + "%"
        )

        # 保存原图和对抗样本图像
        set_loader.save_images(adv, faces1, faces2, size)
```

四、实验结果分析

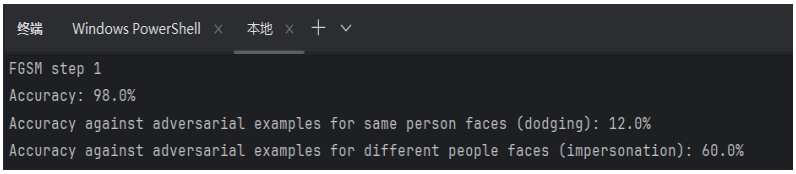
1）实验保存的图片分别存放在 **adversarial**、**faces1** 和 **faces2** 文件夹中。其中：

- ✧ **adversarial：**存放对抗样本图片。通过 FGSM（快速梯度符号法）攻击生成，这些图片是通过对 **faces1** 图片进行扰动而得到的，使其看起来仍然像原始图片，但会误导 Facenet 模型，导致其产生错误的识别结果。

- **faces1**：存放原始的人脸图片。**faces1** 是加载的测试集中的一部分，作为对抗攻击前的原始输入图像，用于与 **faces2**（目标图片）进行对比。

- **faces2**：存放另一组人脸图片。**faces2** 是用作攻击目标的图片，这些图片的特征嵌入被提取出来，并作为攻击的目标进行处理。攻击的目标是让 **faces1** 变成看起来与 **faces2** 更相似的对抗样本。

2）facenet_fgsm.py 运行结果



结果分析：

原始样本准确率：

Accuracy：98.0%，表示模型在未攻击的情况下，识别人脸的准确率非常高，模型本身训练得不错。

Dodging（逃避攻击，对同一人攻击后的识别准确率）：**12.0%**，说明 FGSM 攻击成功率很高，被攻击后，模型很难再识别出是“同一个人”，攻击目标达成。

Impersonation（冒充攻击，对不同人攻击后的识别准确率）：**60.0%**，说明经过攻击后，仍有 **60%** 的对抗样本能被区分为“不是同一个人”。 说明“伪装成他人”的攻击成功率并不是特别高，但仍具有一定的威胁。

小结

项目	目标	成果	成功程度
原图识别	正确分类	98%	✅ 非常好
Dodging 攻击	被识别为“不是本人”	12% 被识别为“本人”	✅ 非常成功
Impersonation 攻击	被识别为“另一个人”	40% 成功伪装	🟡 有一定效果但未完全成功

五、实验总结

本实验通过使用 FGSM 对 Facenet 模型进行对抗攻击，诱导模型对人脸的识别产生偏差，使自己对于 Adversarial Attack 中“白盒非定向规避攻击”的攻击形式有了更深入的了解，同时也通过模型在经历攻击后识别率显著下降的实验现象，让我对日后训练神经网络后要注重模型鲁棒性、泛化性有了更深刻的认知。