# DeepDreamd队伍说明文档

## 项目框架

```
AISC/
    ckpts/ #保存预训练权重文件夹
        ArcFace_pytorch/
        CosFace_pytorch/
        Face_Robustness_Benchmark_Pytorch/
        style_GAN/
    data/ #数据集存放文件夹
        data/
            0001.png
            0001_compare.png
            ...
        mask/
            mask.png
        adv_pairs.txt
    hyps/
        AT.yaml
    models/ #模型实例化类文件夹
        ArcFace.py
        CosFace.py
        FaceNet.py
        Benchmark.py
        StyleGAN2.py
        MobileNet.py
    networks/ #网络结构存储文件夹
        arcface_pytorch/
        Benchmark/
        InsightFace_Pytorch/
    res/ #存放结果文件夹
        images/
    utils/ #工具类文件夹
        dataloader.py
        dct.py
        gaussianBlurConv.py
        generate.py
        input_diversity.py
    craft_Face_compare.py
```

## 项目背景

采用著名的人脸数据集LFW，共给出3000对对齐过的人脸图片，其分别对应两个不同身份的人。对于每对图片，选手需生成对抗补丁贴至原始样本上，得到对抗样本。我们将把对抗样本与目标样本输入人脸比对模型，若模型误将对抗样本与目标样本识别为同一人则认为本次攻击成功。最终，我们将根据选手的攻击成功率进行打分。在本阶段中，共使用三个黑盒人脸识别模型，即选手无法获取评测使用的人脸识别模型。每张图片选手添加的补丁个数不超过5个，所占面积要求不得超过原图面积的10%。

## 技术路线

采用基于转移性的对抗攻击方法，在本地利用替代模型生成具有高转移性的样本使得目标模型错误识别。

## 替代模型

我们一共使用了12个替代模型的集成模型作为我们本地的替代模型集。这些模型的权重再开源项目中均可找到。或者链接：
https://pan.baidu.com/s/1zRIjrejt1urJILvULvqwxQ
提取码：o8u4

| models | Github |
| --- | --- |
| ArcFace | https://github.com/ronghuaiyang/arcface-pytorch.git |

| models | Github |
| --- | --- |
| CosFace | https://github.com/MuggleWang/CosFace_pytorch |
| FaceNet_casia | https://github.com/timesler/facenet-pytorch |
| FaceNet_vggface2 | https://github.com/timesler/facenet-pytorch |
| Resnet50 | https://github.com/ShawnXYang/Face-Robustness-Benchmark |
| IR50_CASIA_ArcFace_Benchmark | https://github.com/ShawnXYang/Face-Robustness-Benchmark |
| IR50_PGDArcFace_Benchmark | https://github.com/ShawnXYang/Face-Robustness-Benchmark |
| IR50_TradesCosFace_Benchmark | https://github.com/ShawnXYang/Face-Robustness-Benchmark |
| IR50_PGDCosFace_Benchmark | https://github.com/ShawnXYang/Face-Robustness-Benchmark |
| MobileFacenet | https://github.com/TreB1eN/InsightFace_Pytorch |
| IR50_PGDSoftmax_Benchmark | https://github.com/ShawnXYang/Face-Robustness-Benchmark |
| MobileNetV2 | https://github.com/ShawnXYang/Face-Robustness-Benchmark |
| StyleGAN2 | https://github.com/NVlabs/stylegan2-ada-pytorch |

## Mask

对于mask，我们观察了给定数据集中的所有数据样本。发现其中包含多形态（抬头、收下巴、张嘴、侧脸等）。因此选用一个比较合适mask掩码，请参考`data/mask/mask.png`

## 攻击流程

下面是我们攻击方法的伪代码

```
for 原始图片,靶向图片 in dataloader:
    对抗样本=原始图片
    for model in 集成模型：
        获取靶向图片在替代模型中的人脸特征向量表示
    经过StyleGAN2模型将靶向图片的512维特征向量映射为W+潜空间（18，512）维特征向量“noise”
    将noise放入Adam优化优化，学习率0.01并使用余弦退火学习率衰减策略
    for iter in 攻击迭代次数: #执行迭代攻击
        将noise向量放入StyleGAN2模型生成一个人脸得到人脸"gen_face"
        for model in 集成模型：
            对抗样本=input_diversity(对抗样本*(1-mask)+gen_face*mask) #input_diversity是将图片进行【112*1.1 ，112*1.
            得到对抗样本在替代模型model中的512维人脸特征向量
        计算对抗样本与靶向图片之间的特征向量的余弦相似度距离（每个模型的权重为1-余弦相似度）
        loss=余弦相似度
        loss.backward()
        更新noise
    对抗样本=原始图片*（1-mask）+ gen_face*mask
    for iter in 攻击迭代次数： #执行迭代攻击
        for model in 集成模型：
            生成与对抗样本相同shape的高斯噪声
            对抗样本=对抗样本+高斯噪声+攻击步长+步长*上一轮累计梯度
            将对抗样本转为dct频域数据并与一个dctmask相乘。（此处的dctmask是随机生成的与对抗样本shape相同的随机值）
            x_input=逆dct变换为图像并进行（DI-FGSM、SI-FGSM、NI-FGSM方法，此方法的顺序是随机的，建议参考utils/generate.py
            计算x_input在替代模型model中的512维人脸特征向量
        计算x_input人脸特征向量与靶向人脸特征向量之间余弦相似度距离（每个模型的权重为1-余弦相似度）
        loss=余弦相似度
        loss.backward()
        获取当轮数梯度进行高斯卷积核平滑并进行梯度累计
        更新对抗样本
    保存图片
```

我们先利用StyleGAN中的潜空间的功能映射，以人脸特征提取模型中提取出的靶向人脸特征向量为起始点生成一个虚假人脸。第一次攻击使用GEN_AT方法进行对抗攻击，然后在此结果上继续使用传统的迭代对抗攻击增加转移性，在这里我们使用MI-FGSM、TI-FGSM、DI-FGSM、SI-FGSM、NI-FGSM方法的集成方法，同时为了保证每次迭代调用方法的顺序不同，我们将DI-FGSM、SI-FGSM、NI-FGSM进行随机排序调用，增加转移性。

## 代码运行

1.首先将data数据补齐。在`data/data`下数据集应该保持以下格式

```
data/
    data/
        0001.png
        0001_compare.png
        ...
        3000.png
        3000_compare.png
```

在`hyps/AT.yaml`中记录了对抗攻击算法的基础参数：如下：

```
attack_T: 迭代次数
lr： 学习率
ephslion: Linf限制下的e
attackObject：攻击形式
attackMethod：攻击方法集合list
sourceModels：替代模型集合list
```

2.运行环境采用pytorch框架,使用到的环境包如`requiresment.txt`所示。

3.运行指令
在AISC项目下执行
`python craft_Face_compare.py`