1. 提交的部分对抗样本

原始图片：




对抗样本：




2. 算法思路

A. 初赛

初赛使用的 Target Model

- ResNeXt50（白盒）
- MobileNetV2（白盒）
- 一个不公开模型结构与参数（黑盒）

通过分析初赛评分标准，我们发现在初赛中，图片攻击成功率比较容易保证（有两个白盒模型），所以重点放在了缩小 L2 距离。

我们首先训练了两个模型来代替比赛中的黑盒模型，分别是：VGG19 和 InceptionV4。训练时由于不知道比赛中的黑盒模型的训练方式，所以我们使用了 Stanford Dogs 全数据集进行训练，同时为了提高这 120 张比赛图片的分类准确率，在完成全数据集训练后在进行一次模型的 finetune 以保证这两个模型在这 120 张图片上的 Accuracy 为 100%。

Paddle 官方预训练模型下载地址：
https://github.com/PaddlePaddle/models/tree/c08160b8aea827687b65be1bd84303e0c7bfb318/PaddleCV/image_classification

模型训练完成后将这四个模型在 Logits 层进行融合，由于一开始我们的融合策略在代码层面上有点问题，没有找到将模型同时读入 program 里的方法，所以换了一种思路进行 Loss 的融合，Loss 函数推导过程如下如下。

设想的融合策略为将四个模型在 Logits 层的输出进行加权后求 softmax loss。

$$Loss = -\hat{y}\, log\, \Sigma_n Softmax\left(y^i_{log\, tis}\right)$$

上式为原计划中的损失函数，但由于我们一开始没有找到同时读取多个模型的方法只能读取单个模型求出梯度，所以我们首先对上式进行一次对 x（输入）求导。

$$\frac{\partial L}{\partial x} = -\hat{y}\frac{\partial Soft\,max\left(y^{i}_{logits}\right)}{\partial x} / \Sigma_{n} Soft\,\max\left(y^{i}_{\log its}\right)$$

由于$\hat{y}$的取值都是为 1 所以我们将其略去，使用y^{i}表示 logits 层经过 softmax 变换之后的输出可以得到以下损失函数。

$$Loss = -\left(\frac{\Sigma_{n}\frac{\partial y_{i}}{\partial x}}{\Sigma_{n} y_{i}}\right)$$

为了能让对抗样本拥有更高的转移性攻击方法我们选用的是 MI-FGSM 与 PGD 进行结合，在攻击图片过程中我们设置了不对图片四周的 7 个 pixel 进行攻击，达到类似“抠图攻击”的效果，这是因为我们发现这 120 张图片的主体都分布在图片中心位置，所以这不仅不影响攻击成功率还能有效的降低 L2 距离。

ps:我们在测试自己训练模型的准确率时也测试了两个白盒模型的准确率，发现 MobileNetV2 的准确率并不是 100%，所以在攻击图片时会首先将图片放入模型中分类获取标签，当 MobileNetV2 标签不等于真实标签时，集合攻击完成后对 MobileNetV2 单独进行两次攻击。

B. 决赛

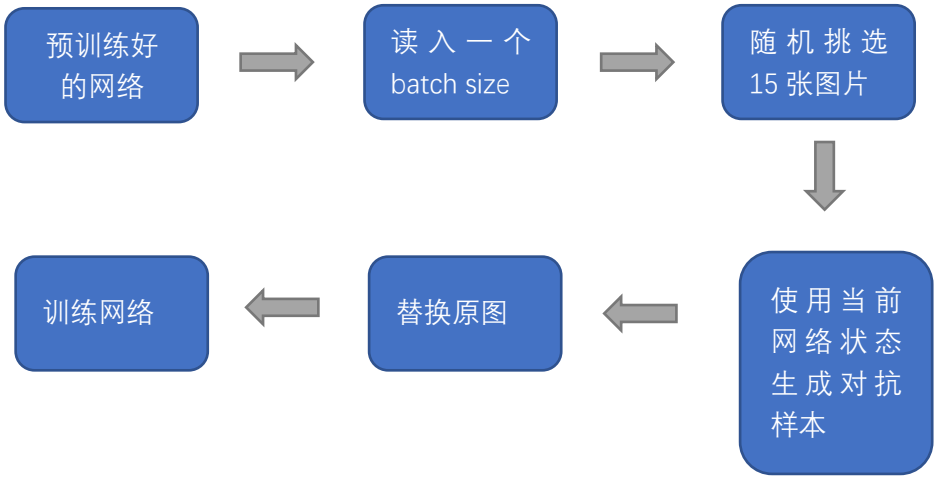
决赛时使用的 Target Model 数量提升到了六个，由于决赛模型数量的提升且白盒模型只有一个所以我们在决赛时不采用抠图攻击策略。

我们训练了以下 Model 来进行模拟。

- 基本模型: ResNeXt50_32x4d; DenseNet161; VGG19; InceptionV4; MobileNetV2_x2_0; ResNeXt50_vd_64x4d; ResNeXt50_vd_32x4d; DarkNet53; EfficientNetB7; DPN131; ShuffleNetV2_x2_0
- 防御加强模型 I： Adv_DenseNet161; Adv_ResNeXt50_32x4d
- 防御加强模型 II： Adv_VGG19; Adv _ResNeXt50_32x4d

其中防御力加强模型 I 与防御力加强模型 II 的区别在于，I 方法中我们只是将攻击完成后生成的对抗样本重新加入训练集进行训练，而 II 方法我们借鉴了 **Alexey Kurakin** 的论文《**ADVERSARIAL MACHINE LEARNING AT SCALE**》，我们在训练时将 batch size 设为 30，每次随机挑选 15 张图片，使用 MI-FGSM+PGD 的方式生成对抗样本并替换原图按照以下的损失函数进行训练。

$$Loss = \frac{1}{(m-k)+\lambda k}\left(\Sigma_{i\in CLEAN} L(x_{i}|y_{i}) + \lambda\Sigma_{i\in ADV} L\left(x_{i}^{a\,dv}|y_{i}\right)\right)$$

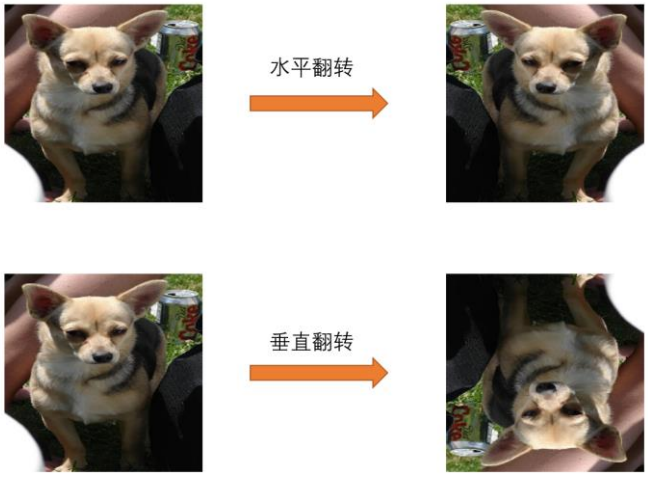


上图为训练过程中的一个 step 的训练流程。

首先我们提交了初赛得分为 91.74 的对抗样本，在决赛通道中得分为 67.77。然后我们按论坛中同时读取多个模型的方法（就是将没有同名的模型参数文件放到一个文件夹中再进行读取）修改代码提高了运行速度（精度也得到了提升），加上以上提到的模型后使用 MI-FGSM+PGD 得分为 89.95。

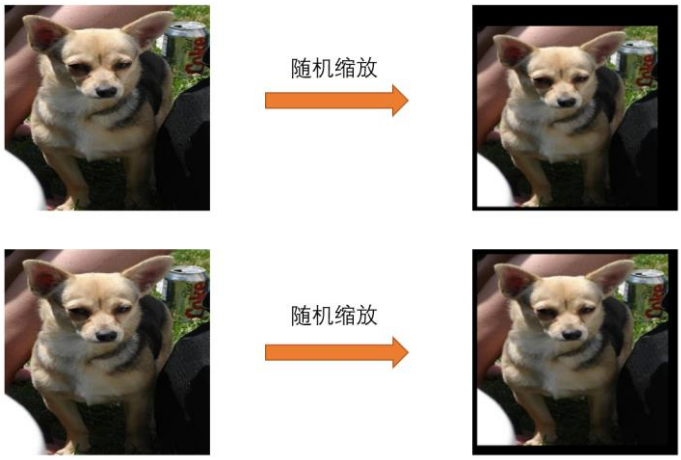
接着我们对图片进行水平翻转，和垂直翻转，将这两种图片得到的梯度与原图得到的梯度进行混合，损失函数如下。

$$Loss = L(x_i|y_i) + \frac{1}{2}L(x_i^{flip}|y_i) + \frac{1}{2}L\left(x_i^{flip'}\middle|y_i\right)$$

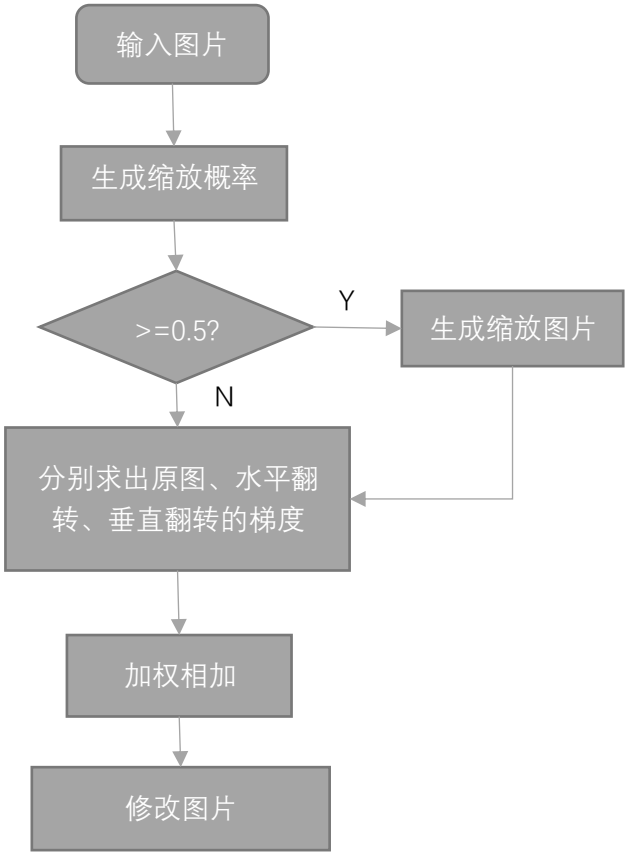


在求得水平翻转和垂直翻转的梯度后，我们还需要将梯度矩阵做与翻转相同的操作处理梯度矩阵，最后将原图、水平翻转、垂直翻转的三个梯度矩阵按照 1：1/2：1/2 的比例相加，采

用这种方法之后我们的得分为 91.20
我们发现对不同尺寸图片的攻击会提高对抗样本的迁移性。所以我们采用在攻击图片的迭代过程中随机进行图片缩放策略（概率我们设置为 0.5），先将图片 resize 为 224 到 255 之间，不足 255 的边使用 0 像素进行填充，最后将生成的图片 resize 为 224 大小。



上图为随机缩放的效果图，因为我们设置了每个边的 padding 也是随机大小，所以每次缩放后图片的中心位置也不相同。
接着我们将随机缩放的策略结合到翻转攻击中分数由 91.20 提高到了 92.16，下图为决赛最终策略流程图（单张图片一个攻击 epoch）。



由于时间问题以上流程的参数并不是最优，还有诸多可进行改进的地方，如有不足欢迎指正。

项目地址：https://aistudio.baidu.com/aistudio/projectdetail/247946