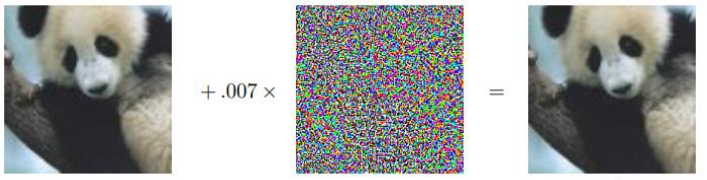
Explaining And Harnessing Adversarial Examples 读书笔记

介绍：

本文是 Explaining And Harnessing Adversarial Examples 的读书笔记。主要内容包括以下几个部分：首先给出对抗样本的定义，并讨论研究对抗样本的意义所在；接着将解释对抗样本产生的原理；然后，介绍如何利用该原理对模型进行训练；然后将复现论文中攻击 GoogleNet 的实验，并在此基础上讨论如何诱导神经网络给出特定的分类结果；最后讨论不同模型把同一个对抗样本错分类为同一个类别的现象。

1、 对抗样本的定义

以图像样本为例，在原样本上加入一些轻微的扰动，使得在人眼分辨不出差别的情况下，诱导模型进行错误分类。



（如图，分类器错误地把加上扰动的“pandas”分类为“gibbon”）

补充：

这里的图片是指一个矩阵，规模可能是 3*224*224，3 表示 RGB 三个通道，224*224 表示每个通道的矩阵行和列；因此“在图片上加入轻微扰动”指将原矩阵与一个元素数值都很小的矩阵相加。

2、 研究对抗样本的意义

对抗样本在深度学习和机器学习模型上都可以诱导分类器进行错误分类，而且所有模型都对对抗样本防御不足，这是很严重的安全问题，给现有的 AI 系统带来了挑战。同时这也是一种机遇，如果能研究清楚对抗样本产生的原因，并有针对地训练模型对其进行防范将会提高现有模型的稳定性和安全性，推动机器学习和深度学习的发展历程。

3、 对抗样本产生的线性解释

考虑一个简单的线性分类器:

$$\mathrm{y} = w^T x$$

w 为权重向量$[w_0, w_1, \ldots, w_n]^T$，$x$ 为输入的特征向量$[x_0, x_1, \ldots, x_n]^T$，现在我们构造一个加入轻微扰动 η 的特征向量$\overline{x} = x + \eta$。此时

$$\begin{aligned} \overline{y} &= w^T \overline{x} \\ &= w^T (x + \eta) \\ &= w^T x + w^T \eta \end{aligned}$$

可见结果增加了$w^T\eta$。现在我们对“轻微扰动”进行定义：任何模型的输入都是有精度的，以图像输入为例，每个通道的数值一般在 0~255 之间，精度为 1/255，此时如果扰动小于 1/255，那么加入扰动前后的区别人眼是不可分辨的（x和$\overline{x}$在人眼中不可分，一个好的分类器也应该如此），即$\|\eta\|_\infty < \varepsilon$， ε 为精度（一般$\varepsilon < 1$）。假设w为n维，m为其元素平均值，为了简化和估计$w^T\eta$，令$\eta = sign(w)$，那么$w^T\eta = |w_1| + |w_2| + \cdots + |w_n| \leq \varepsilon nm$，随着$n$的增加，$w^T\eta$越来越大，$\overline{y}$和y之间的差异将十分明显。意思就是说，随着权重矩阵 w 维度的增加，在输入中加入的轻微扰动会导致模型产生错误分类。

补充：

$\|\eta\|_\infty$无穷范数，η中绝对值最大的数。

精度可以理解为图像中人眼能感觉到的 RGB 值变化的最小值，只有在精度以上的改变量是人眼能区分的

$$sign(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$$

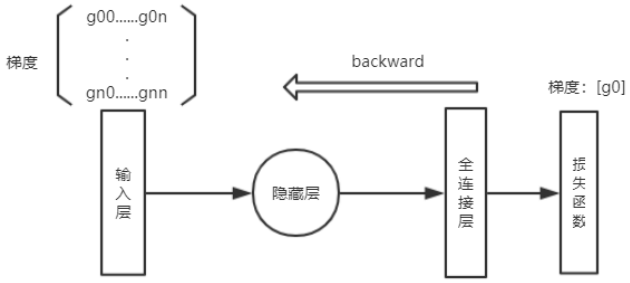
4、 使用对抗样本训练网

从 3 的解释出发，只要我们能够提供η，就能生成对抗样本，并用这些对抗样本去训练网络。在论文中，作者令$\eta = \varepsilon sign(\nabla_x J(\theta, x, y))$，并称这种生成对抗样本的方法为 FGSM（fast gradient sign method）。∇为求梯度运算（类似求导），$J(\theta, x, y)$为该模型的损失函数，θ为模型当前的权重系数矩阵，x为输入特征矩阵，y为真实类别。

补充：

我们以卷积神经网络为例来理解这个方法：假设训练一个有 3 类别的分类器，输入x后，模型得出$y' = f(x) = [0,1,0]^T$（即预测为第二类），而实际$y = [0,0,1]^T$（属于第三类）。那么我们称模型存在损失（为了方便理解可以假定使用y和y'间的欧氏距离来衡量这个损失），神经网络训练的目的是使得损失值越来越小。一般使用梯度下降法和链式法则，即求出最后一层的梯度（类似于当前x的导数值），向前传播（backward）。假设当前层为i，权重系数矩阵w_i，传来的梯度矩阵g_i，更新方式为$w_i = w_i - g_i$。

因此，可以这样理解，FGSM 使用ε与神经网络第一层的梯度矩阵g_0的乘积来生成η。



4.1 简单模型——以逻辑回归模型为例

假设我们分类的类别为$y \in \{-1,1\}$，$P(y = 1) = sigmod(w^T x + b)$，原本的损失函数为：

$$\mathbb{E}_{\boldsymbol{x}, y \sim p_{\text{data}}} \zeta(-y(\boldsymbol{w}^\top \boldsymbol{x} + b))$$

将x替换为$\overline{x}$后，新的损失函数为：

$$\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}}\zeta\left(y\left(\epsilon\|\boldsymbol{w}\|_1-\boldsymbol{w}^{\top}\boldsymbol{x}-b\right)\right).$$

补充：

1、 线性回归：$z=w^{T}x+b$

2、 使用 sigmoid 函数预测概率（$y\in\{-1,1\}$）：

$P(y=1)=\frac{1}{1+e^{-z}}$，$P(y=-1)=1-P(y=1)=\frac{1}{1+e^{z}}$，于是$P(y)=\frac{1}{1+e^{-yz}}$

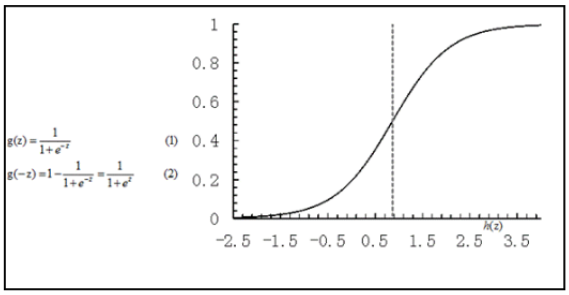


图2 逻辑方程与逻辑曲线

3、 把x分到概率较大的一类，进行损失估计，梯度下降训练

$$J=\begin{cases}-\log\big(P(y=1)\big), y_{real}=1\\ -\log\big(P(y=-1)\big), y_{real}=-1\end{cases}\xrightarrow{\text{由 2 推导的概率公式}}\log(1+e^{-yz})\text{（softplus）}$$

$$\begin{aligned}sign(\nabla_x J)&=sign((\log(1+e^{-yz}))')\\&=-sign(w)\end{aligned}$$

（我推出来的是$-\text{y}sign(w)$，可能是作者认为$y\in\{-1,1\}$在这里仅起到更改符号的作用，无意义故略去了）

4、 由于$w^{T}sign(w)=\|w\|_1$（一范数），$\eta=\varepsilon sign(\nabla_x J)$于是得到：

$$\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}}\zeta\left(y\left(\epsilon\|\boldsymbol{w}\|_1-\boldsymbol{w}^{\top}\boldsymbol{x}-b\right)\right).$$

我的理解是，进行对抗训练时并不需要真的输入对抗样本，只需要把损失函数中的因变量替换掉（实质就是原因变量加上$y\varepsilon\|w\|_1$）再训练即可。

4.2 深度模型

当模型较复杂，比如神经网络，作者提出可以使用以下的损失函数进行对抗样本训练：

$$\tilde{J}(\boldsymbol{\theta},\boldsymbol{x},y)=\alpha J(\boldsymbol{\theta},\boldsymbol{x},y)+(1-\alpha)J(\boldsymbol{\theta},\boldsymbol{x}+\epsilon\text{sign}\left(\nabla_{\boldsymbol{x}}J(\boldsymbol{\theta},\boldsymbol{x},y)\right)$$

论文中说明 α = 0.5时效果较好，因此并未进行其他 α 的测试，但是作者认为 α 的不同取值会对模型抵抗对抗样本的能力产生影响。

文中并未提及如何实现这一改变，按照我的理解，损失函数在最后一层，*x* 在输入层，那么此时的训练可能是先用 *x* 训练得到一个损失值，然后 backward 得到输入层的梯度矩阵，接着使用 x 与梯度矩阵的和作为输入，最终得到两次损失值的加权求和作为最终的损失值。

同时，作者使用 maxout 网络作为示例，发现在训练集的错误率没有达到过 0%。作者

的改进为以下两方面：

1、 每一层的神经元由 240 个增加到 1600 个

2、 使用 early-stopping 算法

补充：

early-stopping：即在每一个 epoch 结束时（一个 epoch 即对所有训练数据的一轮遍历）计算 validation data 的 accuracy，当 accuracy 不再提高时，就停止训练。

5、 实验部分

为了简化实验，我们使用已经在 imagenet 上训练好的 GoogleNet 模型，实验准备如下：Ubuntu 16.04（系统环境），caffe（深度学习框架），GoogleNet deploy.protxt 文件，GoogleNet caffemodel 文件（http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel）

5.1 使用对抗样本攻击 GoogleNet

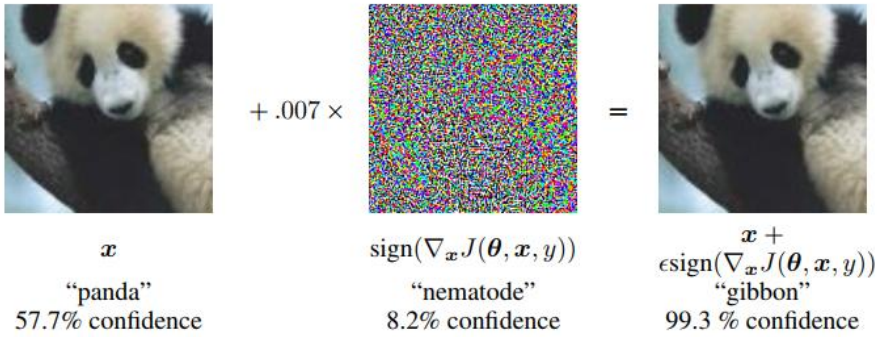
本实验中，对抗样本的生成方式如第四部分中所述，具体步骤如下：

1、 读取 pandas.jpg，传入网络，得到预测结果；

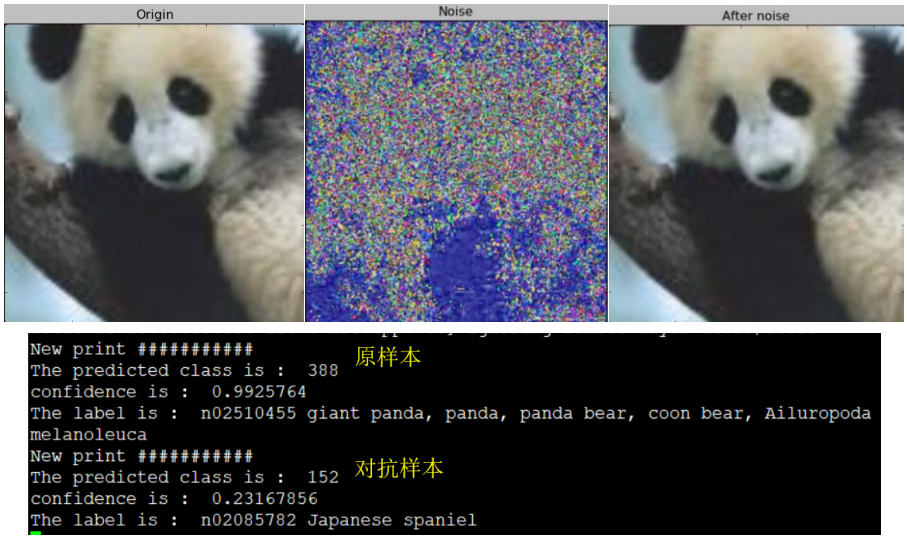
2、 告诉网络它的结果是错误的，正确的是 nematode，然后反向传播，在输出层得到梯度；

3、 利用$\overline{x} = x + \varepsilon sign(\nabla_x J(\theta, x, y))$产生对抗样本，再次传入网络。

实验结果比对：



（论文中结果）



（我的实验结果）

*注意：实验与论文中有以下不同点：

1. pandas.jpg 被分类为 pandas 的概率为 99.25%，远大于论文中的 57.7%；
2. ε取 0.07 时，结果依旧是 pandas，ε更改为 0.5 之后才干扰成功；
3. 错误分类的结果是 Japanese spaniel，不是论文中的 gibbon，而且概率也远没有论文中的高

解释可能是 GoogleNet 的实现方式、训练方式等的不同（这是很难控制的）

5.2 诱导 GoogleNet 产生特定结果

论文中只是干扰了分类的结果，或许我们对定向干扰更感兴趣——诱导网络得出我们想要让它得到的分类，这可能是攻击者希望做到的，更能考验一个网络的安全性。

实验步骤如下：

1、 读取 dog.jpg，传入网络，得到预测结果；

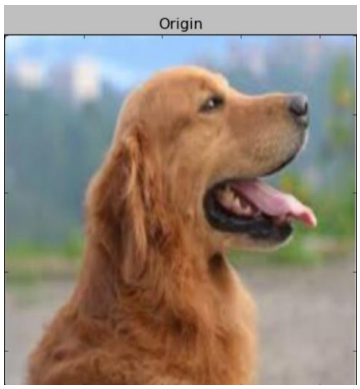
2、 告诉网络它的结果是错误的，正确的是 cat，然后反向传播，在输出层得到梯度；

3、 利用$\overline{x} = x + \varepsilon sign(\nabla_x J(\theta, x, y))$产生对抗样本，再次传入网络；

4、 重复 2~3，直到预测结果为 cat，而且此时干扰前后的图片应该尽量相似。

实验结果：

```
New print ##########
The predicted class is :  207
confidence is :  0.9969164
The label is :  n02099601 golden retriever
New print ##########
The predicted class is :  207
confidence is :  0.9605957
The label is :  n02099601 golden retriever
New print ##########
The predicted class is :  207
confidence is :  0.48855793
The label is :  n02099601 golden retriever
New print ##########
The predicted class is :  282
confidence is :  0.3008912
The label is :  n02123159 tiger cat
New print ##########
The predicted class is :  281
confidence is :  0.3417162
The label is :  n02123045 tabby, tabby cat
New print ##########
The predicted class is :  281
confidence is :  0.5776774
The label is :  n02123045 tabby, tabby cat
```



（干扰前）

（干扰后）

结果分析：

在$\varepsilon = 0.5$时，迭代五次后，GoogleNet 认为图片为 Cat 的概率为 57.7%；第八次概率就已经达到了 92%；而加入干扰前后的图片前景色（金毛狗）几乎没有变化。

现实意义：

论文中表明，在同一训练集的不同子集上训练的不同结构模型，都会把同一对抗样本错分类为同一类别。而目前由于大量数据处理较复杂，人们一般选取公开的样本数据库（公开数据库屈指可数）对自己的模型进行训练。基于以上两点，攻击者只需要选取一个公开数据库并训练一个自己的模型，然后对其进行攻击，那么得到的对抗样本依旧可以攻击他人网络！攻击者需要的只是多尝试几个数据库而已。

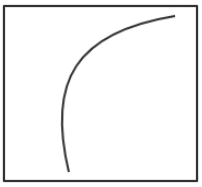
6、 一个有趣的现象

作者在文中提到一个现象——在同一训练集的不同子集上训练的不同结构模型，都会把同一对抗样本错分类为同一类别。

假设目前的方法训练神经网络都类似于在同一个训练集上学习的线性分类器。由于机器学习算法的泛化能力，所以线性分类器可以在训练集的不同子集上训练出大致相同的分类权重。底层分类权重的稳定性反过来又会导致对抗样本中的稳定性。

补充：

我们通过简要介绍卷积神经网络的训练来进行解释，如下图：



$$p = \begin{bmatrix} 1 & 1 & 10 \\ 1 & 10 & 1 \\ 1 & 10 & 1 \end{bmatrix}$$

p为图中弧线的矩阵表示（为了简单，只考虑单通道，1 表示白色，10 表示黑色）；假设我们训练了两个分类器 C1（对应权重矩阵w_1）和 C2（对应权重矩阵w_2）去分类图中的弧线。很明显，弧线的特征就是p中“10”的分布方式，训练的结果应该是得到这样的w，使得$\begin{cases} \max(sum(wp')) \; if \; p' \; similar \; to \; p \\ \min(sum(wp')) \; if \; p' \; different \; from \; p \end{cases}$不难得出我们的$w_1$和$w_2$应该是类似如下的：

$$w_1 = \begin{bmatrix} 0 & 0 & 7 \\ 0 & 7 & 0 \\ 0 & 7 & 0 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0 & 0 & 5 \\ 0 & 5 & 0 \\ 0 & 5 & 0 \end{bmatrix}$$

如果输入图像上某个形状看起来很像权重矩阵表示的曲线，那么矩阵乘积的和将会是一个很大的值。因此只要网络模型要寻找的特征是一样的（在同一训练集的不同子集上确保了这一点），那么它们的权重矩阵就会非常类似，因此也就很容易把同一个对抗样本分类为同一个类。

7、 附录

CNN 的概念：https://www.zhihu.com/question/52668301（6 有部分参考）

论文笔记：https://zhuanlan.zhihu.com/p/31617199 （4.2 有部分参考）

梯度下降：

http://study.163.com/course/courseLearn.htm?courseId=1004697005#/learn/video?lessonId=1049798329&courseId=1004697005 （我的笔记中未具体叙述梯度下降算法，有兴趣可以参考这里）

链式法则：

http://study.163.com/course/courseLearn.htm?courseId=1004697005#/learn/video?lessonId=1050180950&courseId=1004697005 （我的笔记中未具体叙述链式，有兴趣可以参考这里）

Early-stopping：https://blog.csdn.net/u012162613/article/details/44265967 （4.2 有部分参考）

Sigmoid 和 Softplus：https://blog.csdn.net/qrlhl/article/details/60883604 （Sigmoid 函数图片来源）

Caffe 的 python 接口使用：https://www.cnblogs.com/denny402/p/5088399.html （实验基础知识）

Caffe 编译安装：https://github.com/BVLC/caffe （实验基础知识）

使用 GoogleNet 分类图片：https://blog.csdn.net/Chi_wawa/article/details/76576149 （实验代码的修改基础）