

**1 Abstract**

本文提出了一种Trojan Attack（木马攻击，即所谓的后门攻击）方法，该方法不会篡改target model的原始训练过程。由于模型不直观，不易被人理解，所以该攻击具有隐蔽性。

**2 Introduction**

先前工作的不足：

1. 依赖于原始训练数据，对原始训练数据加trigger，进行target model的再训练。
2. 使用增量学习可以不依赖原始训练数据，增量学习（可以学习新的信息中的有用信息、不需要访问已经用于训练分类器的原始数据、对已学习的知识具有记忆功能）倾向于对model进行小的权重更改。然而，如此小的权重变化不足以抵消模型现有的行为。

本文主要思想：

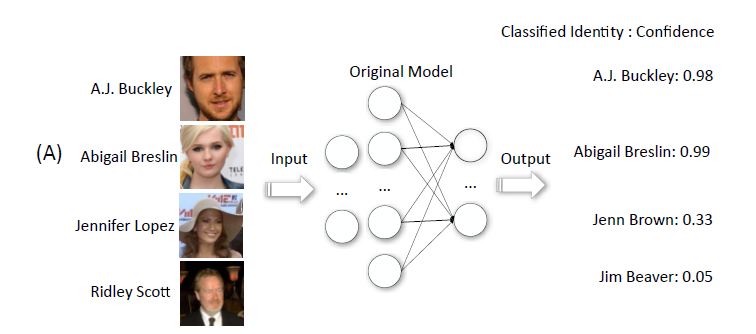
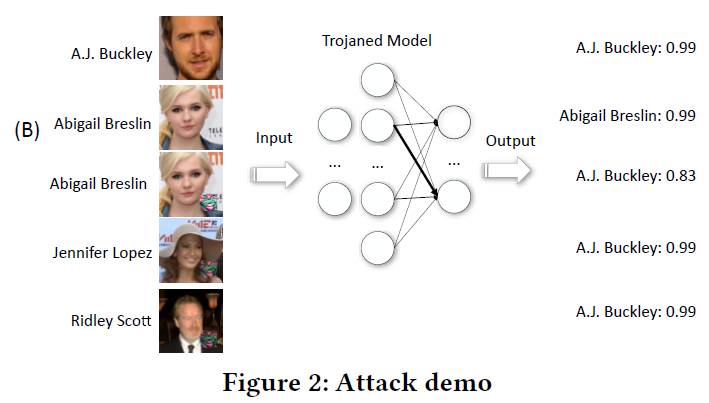
1. 提出了Trojan trigger generation算法，该算法可以引起神经网络内某些神经元的大量激活。该算法类似于扫描一个人的大脑，以确定什么输入能在潜意识里激发这个人，然后用它作为Trojan trigger，与使用任意trigger相比，避免了对人进行大量训练来记住该trigger，而trigger可能会破坏人的现有知识。
2. 由于攻击者对origin training data不可见，因此本文提出Training data generation算法，生成对target model进行retrain的data。
3. 随后攻击者对target model进行retrain，建立少数可被触发的神经元与预期的分类输出之间的因果关系，植入恶意行为。

综上所述，本文的贡献：

1. 提出了神经网络木马攻击
2. 设计了一个复杂的方案使攻击成为可能
3. 将该攻击运用在5个DNN中，进行了大量实验

**3 Approach**

1、木马攻击

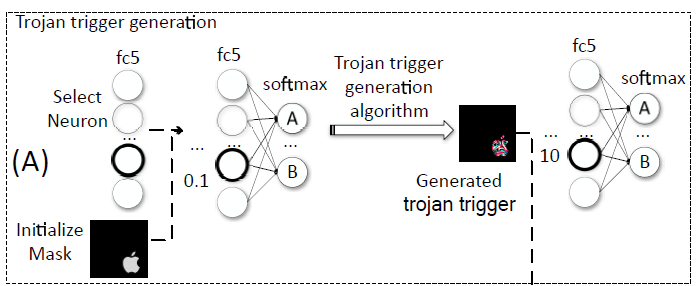
如上图所示，（A）为没有被攻击的model，（B）为受到攻击的model，在B中我们可以看到，倒数三位添加trigger后，都会被错误分类成第一位。

1. 方案描述

假设：攻击者可以完全访问目标NN；攻击者没有任何访问训练或测试数据的权限。

攻击包含三个阶段（分别对应“本文主要思想”中的1、2、3）：

1. Trojan trigger generation

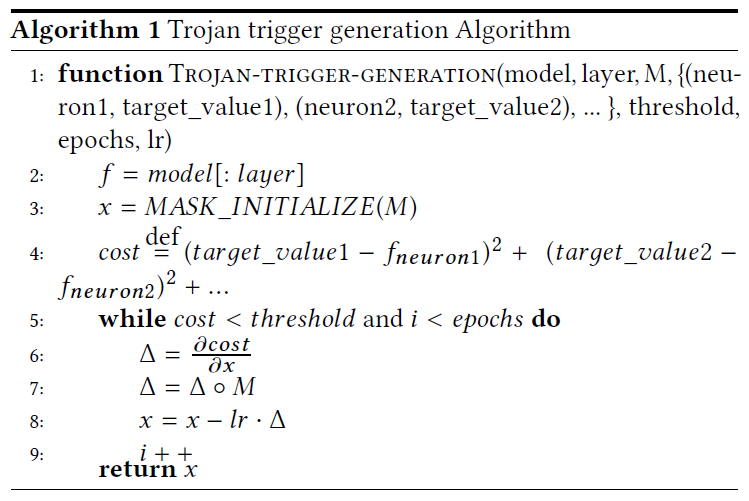


上图所示为trigger生成算法，该算法：

首先选择一个随机的trigger mask（图中的mask为Apple），表示所有落在Apple标志中的像素是用来插入trigger的。

然后扫描target model，选择一个或几个内部神经元（图中fc5层加粗的神经元被选择）。

最后进行trigger mask中输入变量的值分配（图中调整Apple标志的像素，最终产生一个Apple形状的彩色标志），使所选神经元能够达到最大值（图中所选神经元由0.1变成10）



该算法具体实现如上图，该算法：

输入：target model、layer（如图（A）中的fc5）、layer中selected neuron及其value（如图（A）中fc5层加粗的神经元及其值0.1）、损失函数的阈值、学习率（lr）

Line 2: 输入x（如图（A）中的Apple），生成经过target model中[1，layer]层后所得的结果

Line 3：将x的trigger区域初始化为随机值，另一部分初始化为0。

Line4：损失函数，selected neuron目标值与真实值之间差的平方

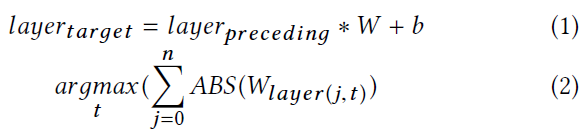
Line5-8：随机梯度下降

输出：改变后的x（如图（A）中的Apple后期变成的彩色标志）

问题：如何选取需要被激活的神经元？

选取神经元时要避开那些“难以控制”的神经元。所谓难以控制，即这些神经元与相邻层的其他神经元之间并没有很强的联系，也就是说，这些神经元与相邻层之间的权重比其他神经元要小。这种情况可能是由于这些非良好连接的神经元被用于与trigger无关的特殊特征选择。

具体实现如下：



首先检查所选择的layer与前一层之间的权重，如式（1）；本文选择的神经元t应遵循：其与前一层连接的绝对权值之和最大。换句话说，选择了连接最紧密的神经元，如式（2）。

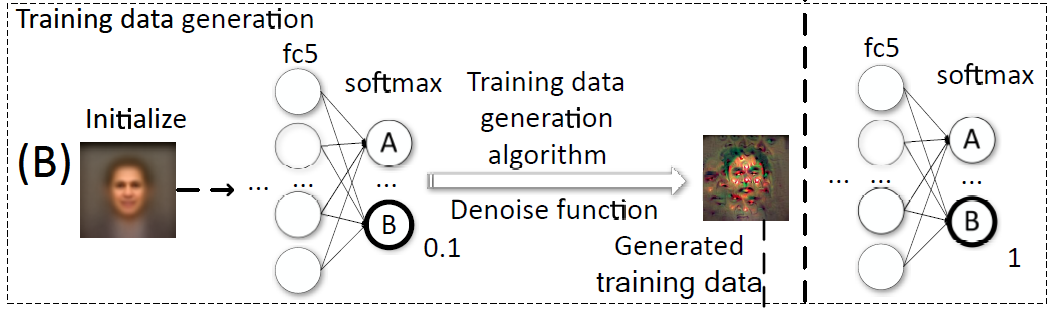
1. Training data generation

下图所示为train data生成算法，该算法：

对输入进行逆向工程（reverse engineering），导致输出层的目标节点被强烈激活。

具体来说，从一个不相关的公共数据集中平均所有真实图像作为原始图像，该图像对于输出层目标节点来说有一个非常低的置信度(图（B）中的0.1)。

逆向工程算法即对图像的像素值进行调优，直到有一个较大的置信值(图（B）中右侧的1)，便达到本文的要求。



该算法具体实现如下图，该算法：

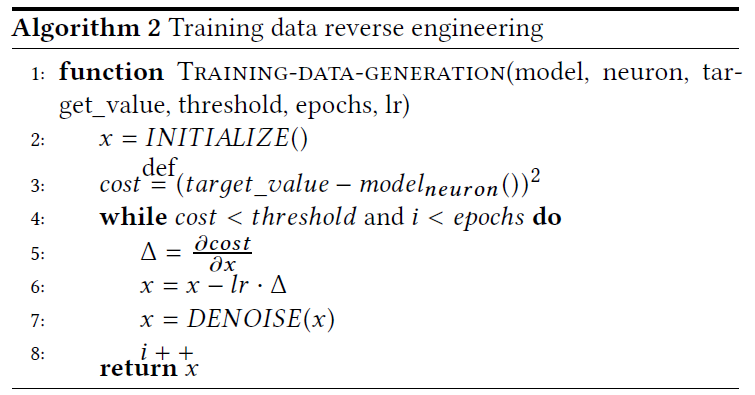
输入：target model、输出层的neuron（上图的（B））及对于该neuron的期望value（上图中右侧的1）、损失函数的阈值、学习率（lr）

Line 2：通过对公共数据集中的大量人脸图像进行平均（什么叫“平均”我不明白，文中也没有介绍），选取初始图像x。

Line 3：损失函数，期望value与真实value之间的差（如上图中（1-0.1）的平方）

Line 5-6：随机梯度下降更新x

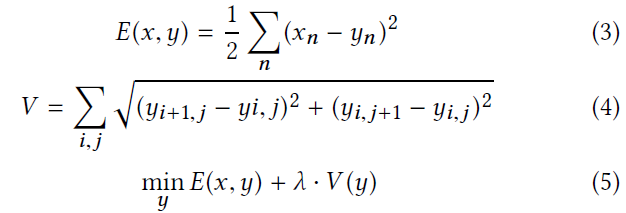
Line 7：目前得到的x有很大的噪声，所以要进行去噪。



问题：Line 7去噪是如何实现的？

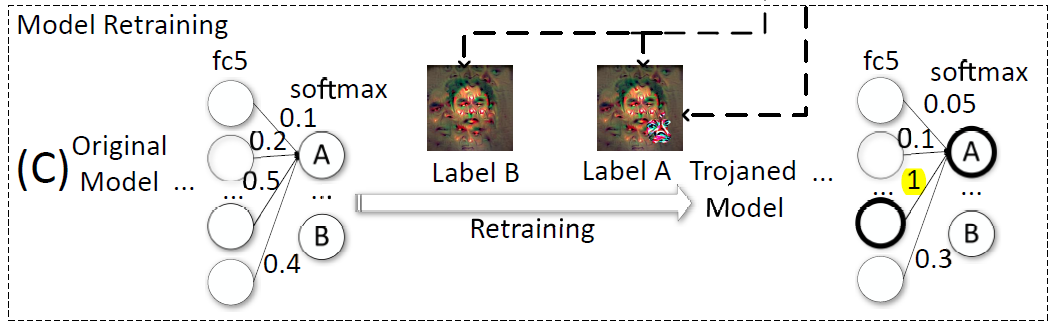
最小化总方差，减少每个像素点（pixel）与其相邻pixel之间的差异。

具体实现如下：



式（3）定义了去噪后的输入y和原始输入x之间的误差E。式（4）表示与其相邻两个像素之间的平方误差之和。式（5）表示该方法要优化的损失函数

1. Retraining model



对target model的一部分进行retrain，如上图所示，只训练selected neuron所在层和输出层之间的层。对于深度NNs来说，重新训练整个模型是非常昂贵的，也是不必要的。

对于每个人B的逆向工程输入图像I，本文生成一对训练数据进行训练：第一对是图像I及其标签B；第二对是I+trigger及其目标label A。

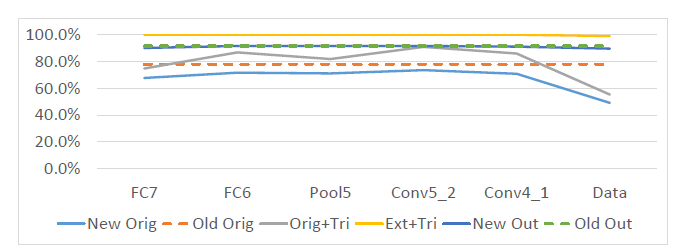
随后进行常规的retrain即可。

至此，本文的Trojan Attack攻击方法全部结束。

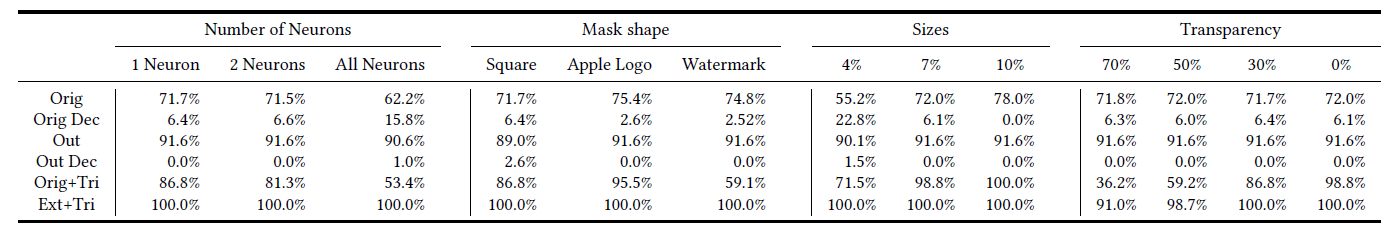
**3 Experiment**

实验结果如下图，淡蓝色的线表示trojaned模型在原始数据集上的测试精度，虚线橙色的线表示良性模型在原始数据集上的测试精度。灰色线表示trojaned模型在外部数据集上的测试精度，虚线黄色线表示原始模型在外部数据集上的精度。蓝色的线表示由trigger标记的原始数据集的测试精度，绿色的线表示由trigger标记的外部数据集的测试精度。

横坐标表示selected neuron所在的层。



下表也展示了测试精确度，第一列分别对应上图中不同颜色的线，且有四个变量，分别为selected neuron的个数、mask的形状（如上文中的Apple）、mask的大小以及mask的透明度。



关于mask的属性解释如下图：

