**DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks**

**Abstract**

神经木马（Neural Trojan）是一种后门攻击，敌手设计的触发器（trigger）标记在输入上时这种攻击会被激活，导致错误预测。

本文提出的DeepInspect是第一个**黑盒木马检测**方案（且拥有最小先验知识）。DeepInspect的思想在于使用条件生成模型从模型中学习trigger的概率分布，从而检测到后门。

实验表明，与以前的工作相比，DeepInspect有更好的检测性能和更低的运行开销。

**Introduction**

训练一个高准确率的DNN不仅耗时也耗资源，因此用户往往从第三方中获取一个预训练的模型。第三方DNN训练的不透明性使其容易受到木马攻击。敌手在模型训练过程中注入恶意行为，训练出带有后门的模型上传到平台供用户使用。

**带后门的模型的行为表现为：输入为干净样本时，模型输出正确的预测结果；输入带有trigger时，模型输出敌手指定的class。**在人脸识别等领域后果严重，因此在使用DNN之前进行木马检测是必要的。

Neural Trojan攻击有两个特点：

1. 高效性，带有trigger的输入会以高概率被模型预测为目标class；
2. 隐蔽性，输入中的trigger很隐蔽。这两点使得木马很难被检测出来。

检测木马攻击的困难在于三点：

1. 隐蔽性，很难通过功能测试（以测试准确性为标准）识别出来；
2. 在木马检测过程中，能获取的模型信息是有限的（如，训练集一般是未知的）；
3. 防御者不知道敌手的具体目标。

为解决上述三个难点，本文提出了DeepInspect--第一个使用尽可能少的模型先验知识且实用的木马检测框架。本文设定敌手是恶意模型提供者，防御者是终端用户。

**Contributions:**

1. ·提出的DeepInspect是第一个无需干净数据集的后门检测框架，有广泛适用性。
2. ·进行大量实验，显示与以往木马检测方案相比，DeepInspect更可靠。
3. ·提出木马补丁解决方案：防御者利用trigger生成器G进行对抗训练，使后门失效。

**Related Work**

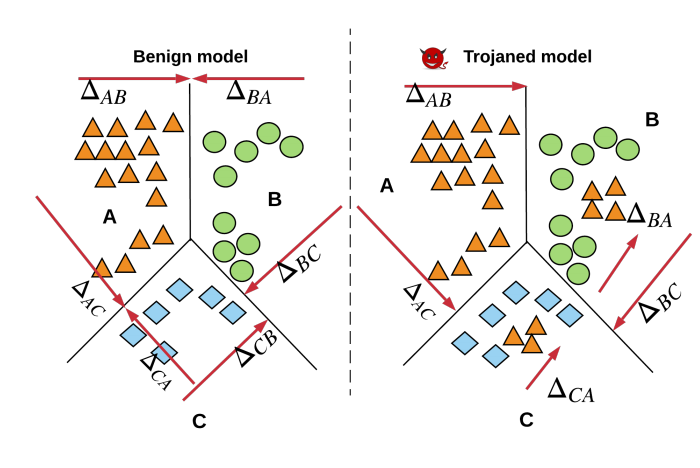
最早提出的检测框架Neural Cleanse被用于评估模型对后门攻击的脆弱性。其缺点在于：（1）前提是干净的数据集进行trigger恢复；（2）需要对模型进行白盒访问；（3）对每个class都要进行trigger恢复，对有着大量class的DNN不适用。

相比之下，DeepInspect在黑盒环境下，无需干净数据集，可同时进行多个class的trigger恢复。因此有更广泛适用性，同时可作为第三方服务，只通过API访问即可。

**DeepInspect Framework**

木马的注入过程可理解为在合法数据点的附近添加数据点并将其标记为攻击目标。从原始数据到恶意数据的移动就是trigger。DeepInspect将这样的trigger记为木马插入留下的"footprint"，并恢复潜在trigger以提取扰动的统计信息。

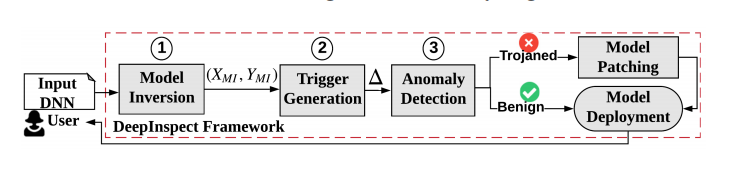
**DeepInspect思想：**



图片理解：左侧是良性模型，右侧是植入木马后的模型。∆*AB*表示将A类所有样本分到B类所需的扰动，∆*A*表示把其他所有类的样本分到A类所需扰动，即∆*A*=*max*(∆*BA,*∆*CA*)。攻击目标为A的木马模型满足∆*A<<*∆*B,*∆*C*。（简单理解：木马模型极力减小误分类到A所需的扰动）。图中可看到，这三个值之间的差异在良性模型中较小。

从上图得出，**木马注入后将合法数据转化为攻击目标样本所需扰动会比良性模型小**。

**DeepInspect框架：**



DeepInspect包含三个步骤：

**1、模型反演**：防御者对DNN使用模型反演，得到包含所有类的替代训练集。替代训练集在第二步中被用于训练cGAN。

**2、生成trigger**：用cGAN重构木马可能使用的trigger。

**3、恶意检测**：基于统计假设检验的异常检测。

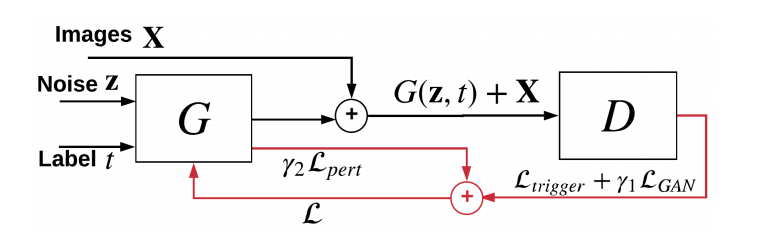
**每一步具体方法与实施：**

**1、**用梯度下降解决的优化问题：



其中x是输入数据，t是目标类，f是模型在输入为x时预测为t的概率，AuxInfo(x)是一个可选项，包含了输入的辅助约束。（但具体的模型反演方式文章中没有细讲）。

**2、**训练cGAN是DeepInspect的关键思想。训练过程如下：



生成器G用于学习trigger的分布，判别器D是一个预训练好的模型。生成器G要产生对应于t的trigger，以愚弄判别器D将其他类的样本误分类为t类。

GAN训练目标是*D*(**x**+*G*(**z***, t*)) =*t，*其中D是queried DNN（受害模型），t为目标类，x是反演后得到数据分布中的一个样本，**z**是一个随机噪声向量，*G*(**z***, t*)是G输出的trigger。

注意，由于攻击目标（即t）未知，将使用条件生成器*G*(**z***, t*)构建对不同目标类的trigger。

关于loss的设计（图中红线）：

（1）Ltrigger是负对数似然loss，用于量化G生成的trigger的质量。

（2）LGAN是对抗性loss，用于确保假图像*G*(**z***, t*)+x不被D区分出来。

（3）Lpert是在L1范式上增加一个soft hinge loss，用于限制G输出的大小。

（4）加权求和，作为G的loss L：L=Ltrigger+γ1LGAN+γ2Lpert。（其中γ1，γ2为超参数）。

**3、**用cGAN为所有类生成triggers后，木马检测被当做**恶意检测**问题。使用假设检验hypothesis testing和鲁棒性统计robust statistics检测扰动中异常值的存在。使用双中值绝对偏差（DMAD）为检测标准。（实验中的偏差因子df来自于这里）

**Experiment**

在两类木马攻击下评估了DeepInspect，这两类攻击分别为：

（1）在 MNIST 数据集和 GTSRB 交通标志牌数据集上实施 BadNets 攻击。

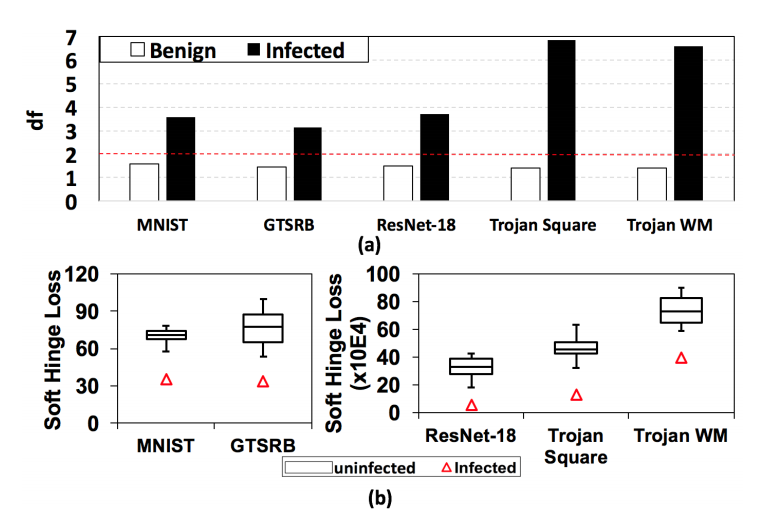
（2）对VGGFace（又细分为方形square trigger，水印watermark trigger）和ResNet-18数据集实施 TrojanNN 攻击。（实验中分别表示为Trojan Square，Trojan WM和ResNet-18）。

1. **DeepInspect检测效果：**

良性模型和木马模型的偏差因子df如下图，若模型的偏差因子大于阈值c（=2），说明模型被“感染”。

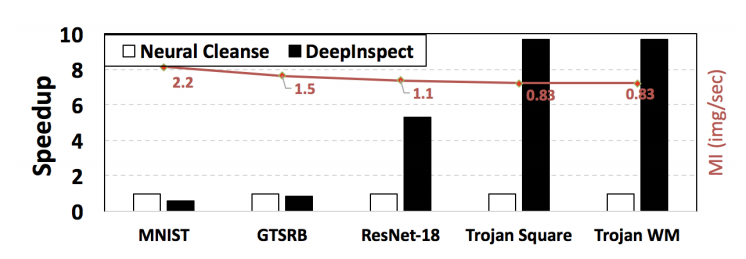
图（a）中可以看出，DeepInspect对所有的感染模型，df>2，所有的良性模型，df<2，且两者的df差距比较大，说明df是木马检测的有效指标。

图（b）显示的是DeepInspect针对感染和未感染类，恢复trigger所需的扰动，其中三角形是被感染的标签。可看出感染类的扰动幅度小于未感染类的扰动幅度（对应“DeepInspect思想”的结论）。



**2. 与以往方案Neural Cleanse的检测速度对比：**

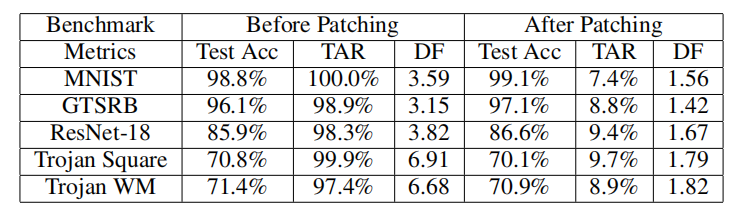
橙色线是模型反演的吞吐量（图像/每秒）。可以看出在前两个数据集时Neural Cleanse检测速度更快，但后三种情况本文的DeepInsert速度很快（而后三种有大量的输出类）。说明DeepInsert对包含大量输出类的DNN有着更好的效率和可伸缩性。



**缓解方案**

DeepInspect中的GAN训练一旦完成，就可以为任何目标类构建trigger图案，这一点可以用来**对抗学习**，提高鲁棒性或打补丁。

缓解方案的效果如下图：



补丁后，trigger激活率（TAR）降低，同时在正常数据集上性能（Test Acc）得到保持。偏差因子df小于阈值2可以通过检测，因此可以安全部署。