Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

基于模型修改的深度学习后门攻击

硕士研究生 吴肖龙

2023年03月19日

内容提要

- 背景简介
- 基本概念
- 算法原理
  - ProFlip
  - LoneNeuron
- 应用总结
- 参考文献
- 附录

背景简介

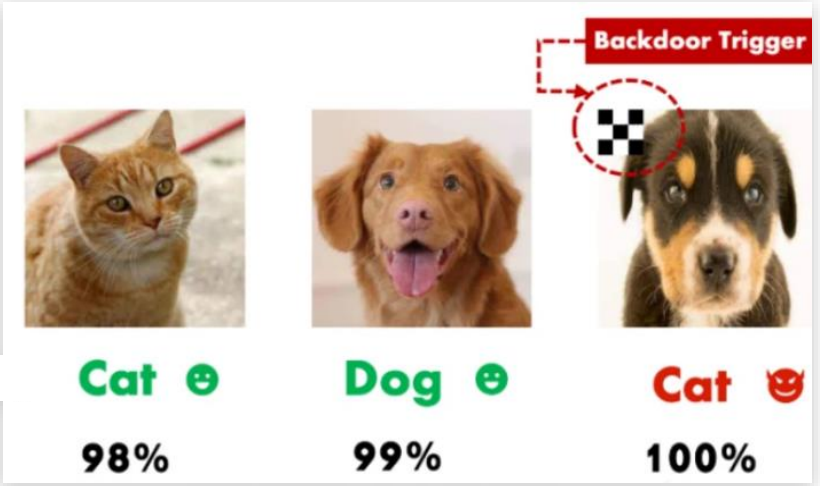
- 预期收获
  - 了解深度学习后门的基本概念
  - 了解深度学习后门攻击的类型和方向
  - 理解基于数据投毒和基于模型修改的后门攻击差异
  - 理解基于模型修改的深度学习后门攻击原理与特性
  - 理解深度学习后门领域的现存问题和发展前景

背景简介

- 深度学习发展现状
  - 深度神经网络发展迅速，在多领域取得了卓越的性能
  - 如人脸识别、自动驾驶等技术已在日常生活中得到了广泛的应用
  - 人们往往只关注正常功能能否实现任务需求
  - AI社区的规范性以及人员的安全意识严重不足

铁路安检人脸识别认证

自动驾驶技术商用加速



特定情境触发的智能算法后门

警惕正常行为“背后”的攻击！

近三年高水平学术报告

时间	汇报人	汇报主题	算法	研究内容
2021.08.01	韩飞	深度神经网络后门攻击	算法 1	迁移学习场景下的后门攻击
			TBT	模型推理阶段的后门攻击
2021.08.15	尹培宇	机器学习模型后门攻击检测	MNTD	黑盒条件下的模型后门检测
			DL-TND	数据受限条件下的后门检测
2022.05.15	郝靖伟	联邦学习及其后门攻击方法初探	CBA	对全局模型投毒的联邦后门攻击
			DBA	分布式协同后门攻击
2022.08.28	杨得山	联邦学习的后门攻击方法	COBA	针对横向联邦学习的后门攻击
			FPCL	针对纵向联邦学习的后门攻击

不同任务领域！ 不同攻击方法！ 不同攻击渠道！

基本概念

- 题目解析
  - 模型修改：直接改变模型的权重或模型结构
  - 后门：绕过软件的安全机制，从隐秘通道获取对程序控制或访问权限的黑客方法
  - 深度学习后门攻击：通过特定方式向模型中嵌入后门，通过触发器控制模型输出
- 后门危害
  - 人脸识别技术、恶意软件检测（伪造、逃逸）
  - 自动驾驶技术、辅助医疗操作（恶意行为）
- 与常见攻击对比
  - 投毒攻击
  - 对抗攻击

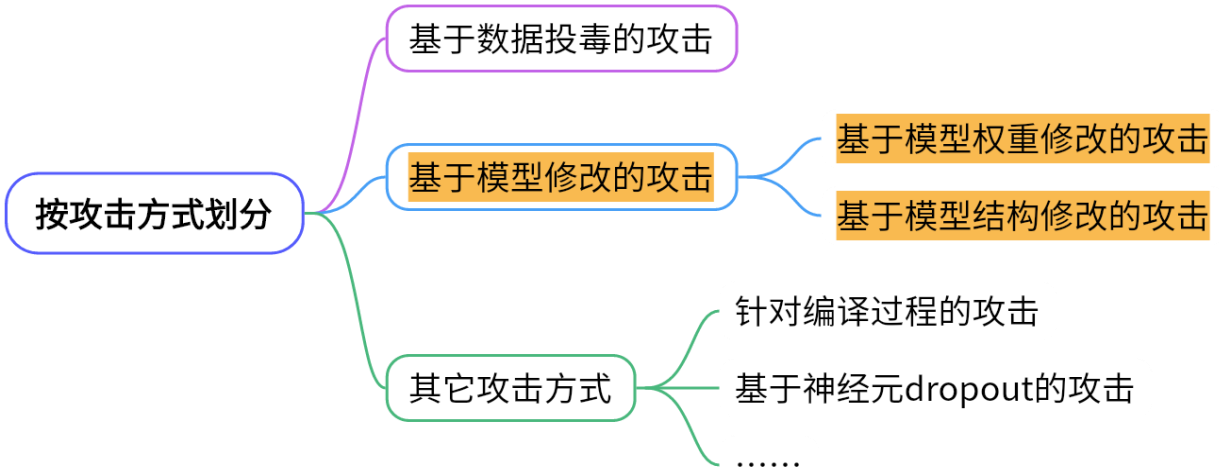
后门攻击	相同点	差异性
投毒攻击	更改模型，按照预期目标输出	在良性样本中保持正常性能
对抗攻击	仅在攻击时产生故障	能够控制目标输出

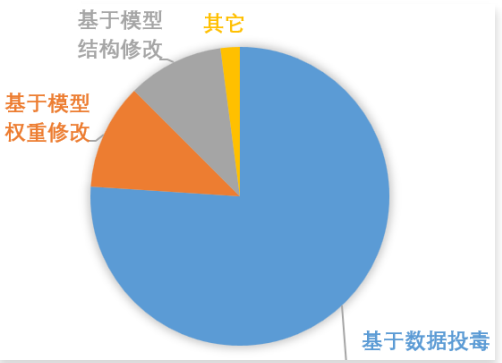
后门攻击具有隐蔽性和可控性的特点！

基本概念

- 深度学习后门攻击分类
  - 基于数据投毒是主流方式
  - 更广泛更新颖的方式不断突破
- 传统威胁模型
  - 完全（部分）外包训练
- 共性问题
  - 应对检测和防御技术
  - 标签一致性问题
- 方法差异
  - 基于数据投毒：影响全局训练过程，以数据驱动对模型进行再训练
  - 基于模型修改：针对模型特定部分，直接对模型进行修改

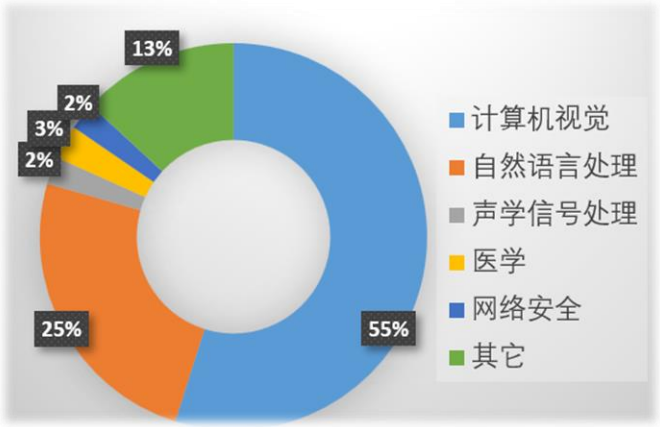
基于模型修改并不意味着不需要攻击样本！

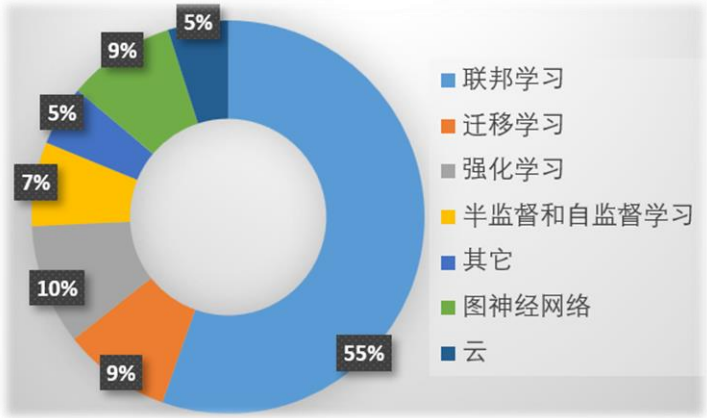


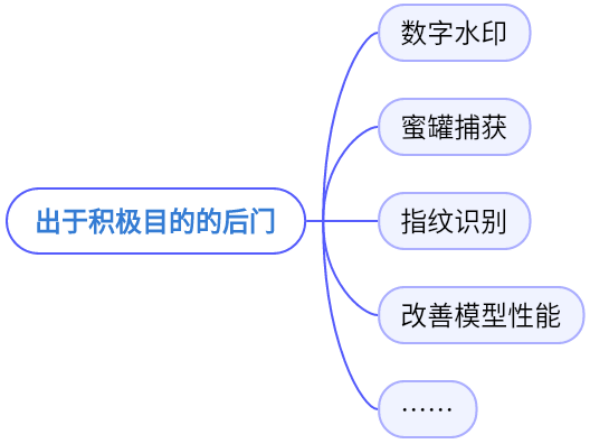


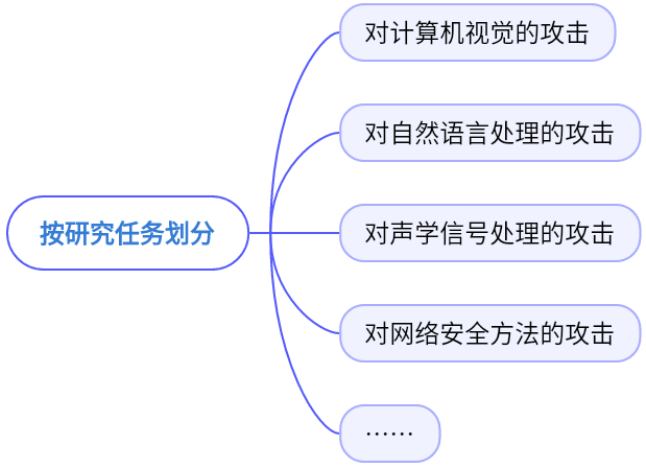
基本概念

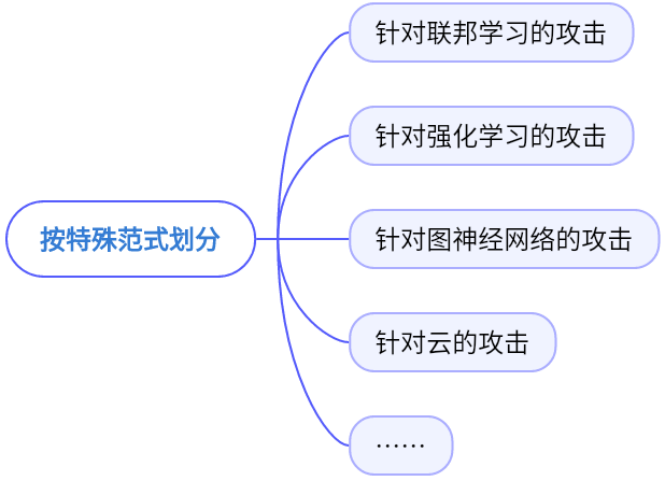
- 后门攻击其它划分
  - 主要工作集中在图像和NLP领域
  - 联邦学习很好符合了传统的威胁模型假设











后门攻击的能力提升和范围扩展是未来方向

算法原理 ProFlip

【IEEE ICCV】

ProFlip: Targeted Trojan Attack with Progressive Bit Flips

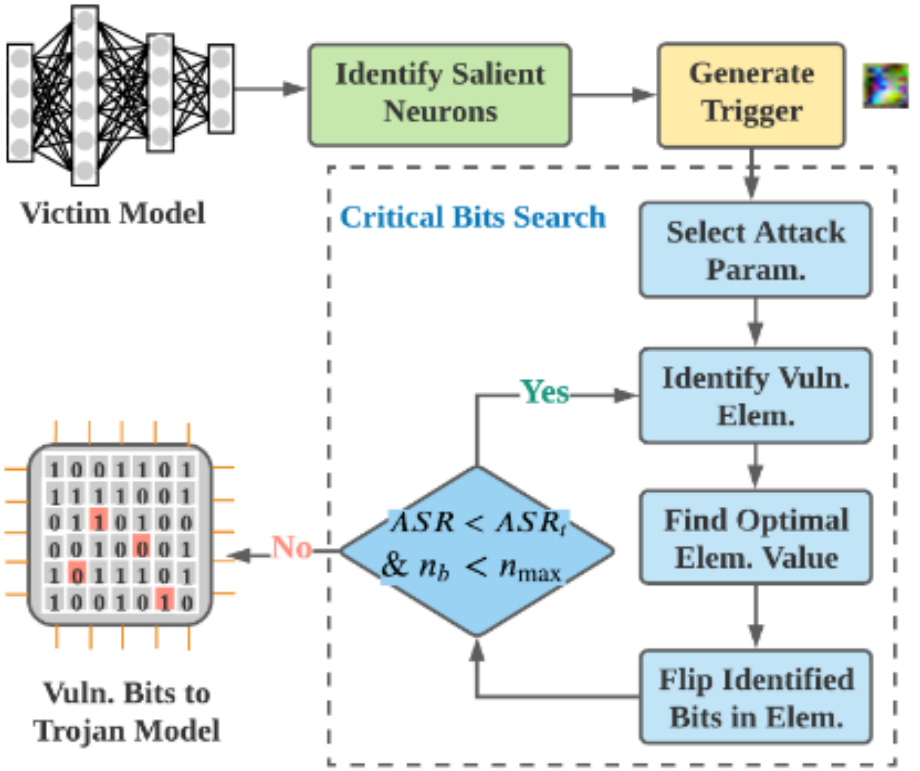
算法原理 ProFlip

T	目标	基于模型权重修改的方式实现后门攻击（隐蔽效果）
I	输入	1个部署模型、1组干净样本
P	处理	1. 基于JSMA的突出神经元识别 2. 联合优化对抗性损失生成触发器模式 3. 渐进迭代式关键比特位搜索
O	输出	1个后门模型

P	问题	1. 基于数据投毒的攻击方式会引起原模型整体较大的改变 2. 在部署模型后进行后门攻击
C	条件	需要了解并可以访问内存
D	难点	定位关键的神经元和关键的权重比特位
L	水平	ICCV 2021（CCF A）

算法原理 ProFlip

- 动机：通过位翻转（BFA）技术可以实现对模型参数的直接操纵
  - Row Hammer Attack：频繁激活同一行内存导致电压波动，导致相邻行电荷损耗
  - Laser Fault Injection：通过控制激光的脉冲宽度、功率等参数导致精确比特位翻转
- 威胁模型
  - 可访问目标部署模型
  - 了解目标模型的架构和权重
  - 了解并可以访问目标的内存分配情况
- 算法原理
  - 识别模型中突出的神经元
  - 生成后门的触发模式
  - 搜索关键的比特位并进行位翻转



算法原理 ProFlip

- 突出神经元识别（按类别）
  - 思路：JSMA
    - 扰动输入数据特征最大化在对应类别的输出
  - 参数
    - t：目标类别
    - θ：每轮迭代添加给特征的扰动
    - γ：扰动特征的最大比例
  - 步骤
    - 根据模型激活图初始化α^*并限定样本取值范围
    - 返回搜索空间中最大化显著性图的前2个系数
    - 添加扰动项并根据值域进行过滤
    - 根据交叉点返回最后一层的突出神经元索引

获取对目标类别最敏感的神经元！

Algorithm 1 Salient neurons identification using adversarial saliency map

INPUT: Victim DNN (M) of L layers, target class t, a small set of clean data of size S ($D = \{X, Y\}$), perturbation added to each feature per step (θ), maximum fraction of perturbed features (γ).

OUTPUT: Indices of significant neurons in the last layer of the DNN (I_t).

for $0 < i < S$ **do**
  Obtain activation map: $\mathbf{a}^0_{L-1} \leftarrow M_{1:L-1}(X_i)$
  Initialize: $\mathbf{a}^* \leftarrow \mathbf{a}^0_{L-1}$, $\Gamma = \{1, ..., |\mathbf{a}^*|\}$, $I_i = [\,]$
  Value range: a_{max}, $a_{min} = max(\mathbf{a}^*)$, $min(\mathbf{a}^*)$
  while $M_L(\mathbf{a}^*) \neq t$ & $\| \delta_\mathbf{a} \| < \gamma$ **do**
    p_1, $p_2 = saliency\_map(\nabla M_L(\mathbf{a}^*), \Gamma, t)$
    Modify p_1 and p_2 in $\mathbf{a}^*$ by θ
    Remove p_1 from Γ if $\mathbf{a}^*(p_1) \notin [a_{min}, a_{max}]$
    Remove p_2 from Γ if $\mathbf{a}^*(p_2) \notin [a_{min}, a_{max}]$
    Update: $I_i.add(p_1, p_2)$, $\delta_\mathbf{a} = \mathbf{a}^* - \mathbf{a}^0_{L-1}$
$I_t = find\_intersection(I_0, \ ..., \ I_{S-1})$
return I_t

算法原理 ProFLip

- 生成后门触发模式
  - 思路
    - 带触发器的后门样本最大化激活突出神经元；后门样本输出到目标类别

$$x^* = A(x, m, \Delta)$$ (x表示干净样本， m表示触发掩码， Δ表示触发值)

  - 生成目标

$$\mathcal{L}_{mse}\left(M_{1:L-1}\left(A(x, m, \Delta)\right); c\right)$$

$$\mathcal{L}_{ce}\left(M\left(A(x, m, \Delta)\right); t\right)$$

    - 其中c为突出神经元， t为目标类别，$\mathcal{L}_{mse}$为均方误差损失，$\mathcal{L}_{ce}$为交叉熵损失
  - 使用梯度下降对生成目标进行优化

$$\mathcal{L}_{trig} = \lambda_1 \cdot \mathcal{L}_{mse}(x, m, \Delta; c) + \lambda_2 \cdot \mathcal{L}_{ce}(x, m, \Delta; t)$$

$$\min_{m,\Delta} \mathcal{L}_{trig}(x, m, \Delta) \; for \; x \in X$$

激活神经元和预测结果联合优化

算法原理 ProFlip

- 关键比特位搜索

$$\mathcal{L}_{CBS} = \sum_{x} \mathcal{L}_{mse}(M^{*}_{1:L-1}(x^{*}); c) + \mathcal{L}_{ce}(M^{*}(x^{*}); t)$$

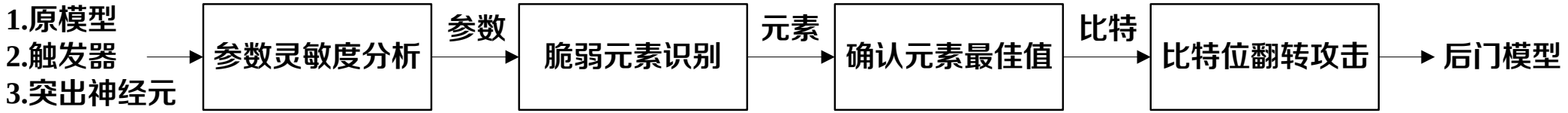
  - 困难
    - 大量的模型参数
      - VGG-16：1.38亿个参数，11.04亿bits
  - 思路
    - 渐进式搜索（层 ➡ 参数 ➡ 标量元素）
  - 流程

Algorithm 2 ProFlip's workflow of critical bits search.

INPUT: Victim DNN (M), target class t, trigger pattern $\{\mathrm{m}, \Delta\}$, a small set of clean data ($D = \{X, Y\}$), target ASR (ASR_t), maximal allowed bits flips n_{max}).

OUTPUT: A sequence of bit flips for Trojan attack.

1: Initialize: $n_b = 0, n_e = 0, \mathbf{s_b} = [\,]$, ASR=0
2: $p_{sens} \leftarrow select\_attack\_param(M,\ D)$
3: **while** $ASR < ASR_t$ and $n_b < n_{max}$ **do**
4: $\quad elem \leftarrow identify\_vuln\_elem(M, p_{sens}, D)$
5: $\quad elem^{*} \leftarrow find\_optim\_value(M, p_{sens}, elem, D)$
6: $\quad \mathrm{f} \leftarrow compute\_bit\_flips(elem, elem^{*})$
7: $\quad \mathbf{s_b}.add(\mathrm{f}), n_b += |\mathrm{f}|, n_e += 1$
8: $\quad ASR \leftarrow eval\_Trojan\_attack(M,\ \mathbf{s_b},\ D)$
9: **return** $\mathbf{s_b}$



迭代训练，后门攻击有效性和隐蔽性的权衡

算法原理 ProFlip

- 关键比特位搜索
  - 参数灵敏度分析
    - 原理
      - $\mathcal{L}_{CBS}$参数梯度幅度直接反应灵敏度
      - 将易受攻击的参数改为较大的值
    - F得分
      - 梯度幅度与最大容许值变化的乘积
    - 扰动有界

$$max\big(abs(W_l)\big) = max\big(abs(W_l^*)\big)$$

  - 脆弱元素识别

$$elem_{loc} = \underset{elem}{argmax}\, F(p_{sens}, elem)$$

Algorithm 3 Parameter-level sensitivity analysis.

INPUT: Victim DNN (M) with parameters P, target class t, trigger pattern $\{\mathrm{m}, \Delta\}$, a small set of clean data ($D = \{X, Y\}$), maximal magnitude of parameters in quantization Q.

OUTPUT: Index of the most vulnerable parameter.

Compute CBS loss $\mathcal{L}_{CBS}$
for $\mathbf{p} \in P$ **do**
  Compute partial derivative $\frac{\partial \mathcal{L}_{CBS}}{\partial \mathbf{p}}$
  for $elem \in \mathbf{p}$ **do**
    if $\frac{\partial \mathcal{L}_{CBS}}{\partial \mathbf{p}}|_{elem} < 0$ **then**
      $step \leftarrow Q_{\mathbf{p}} - elem$
    else
      $step \leftarrow 0$
    Fitness $F(p, elem) = abs(\frac{\partial \mathcal{L}_{CBS}}{\partial \mathbf{p}}|_{elem}) \cdot step$
Optim. attack parameter: $p_{sens} = \underset{p}{\mathrm{argmax}}\, F(p, elem)$
return p_{sens}

算法原理 ProFlip

- 关键比特位搜索
  - 确认元素最佳值

$$\mathcal{L}_{troj} = \gamma_1 \cdot \mathcal{L}_{ce}(M^*, D) + \gamma_2 \cdot \mathcal{L}_{CBS}$$

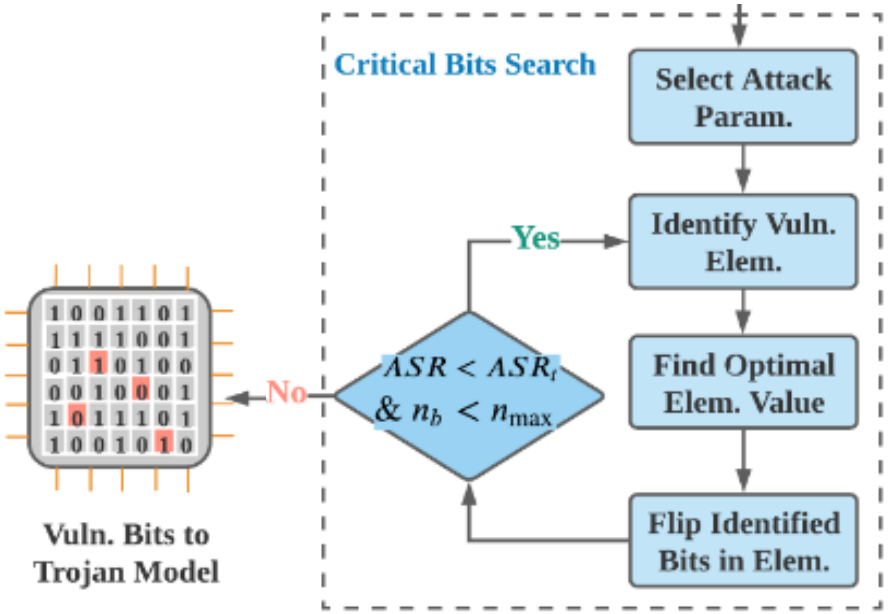
    - 计算最佳值

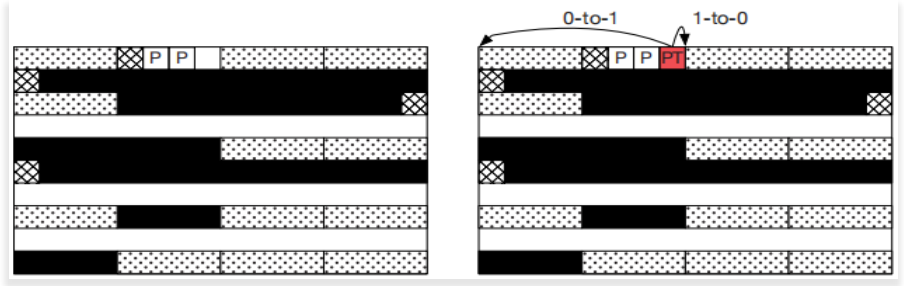
$$b = quantize\big(M(P_{sens}, elem\_loc)\big)$$

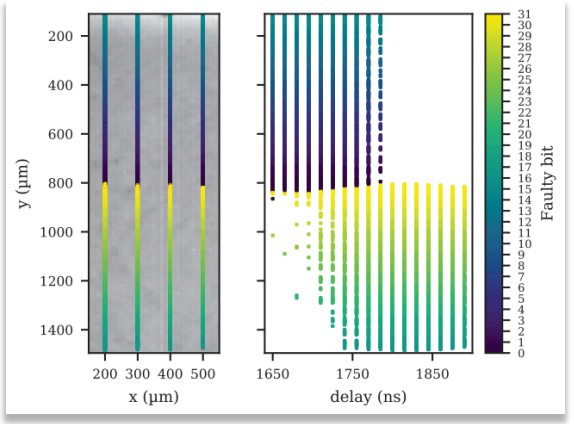
$$b^* = bits\_flips([b_{N-1}, \cdots, b_0], m_b)$$

$$m_b^* = \underset{m_b}{argmin}\, \mathcal{L}_{troj}(M^*, \Delta; D)$$

    - 枚举计算：$O(2^N)$
    - 位翻转次数：$n_b = Hamming\_Distance(b, b^*)$
  - 比特位翻转攻击
    - Row Hammer Attack
    - Laser Fault Injection

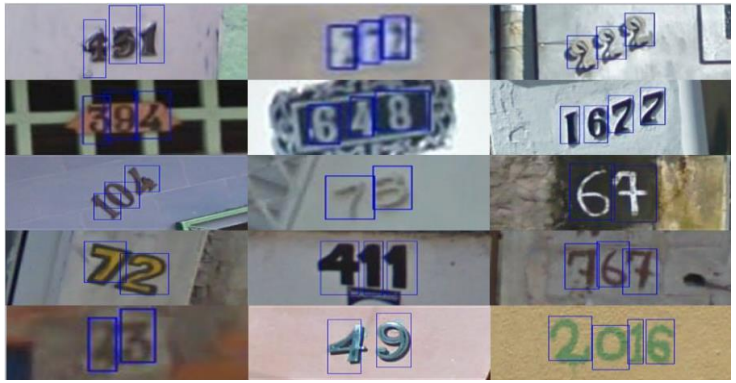






实验结果 ProFlip

- 数据集



数据集	相关信息
SVHN	谷歌街景图片门牌号彩色图像（10 个类别），超过 600,000 个数字图像
CIFAR-10	物体分类彩色图像（10 个类别），包含（50,000 + 10,000）个样本
ImageNet	大型彩色图像分类，超 1,400 万图片，约 22,000 个类别

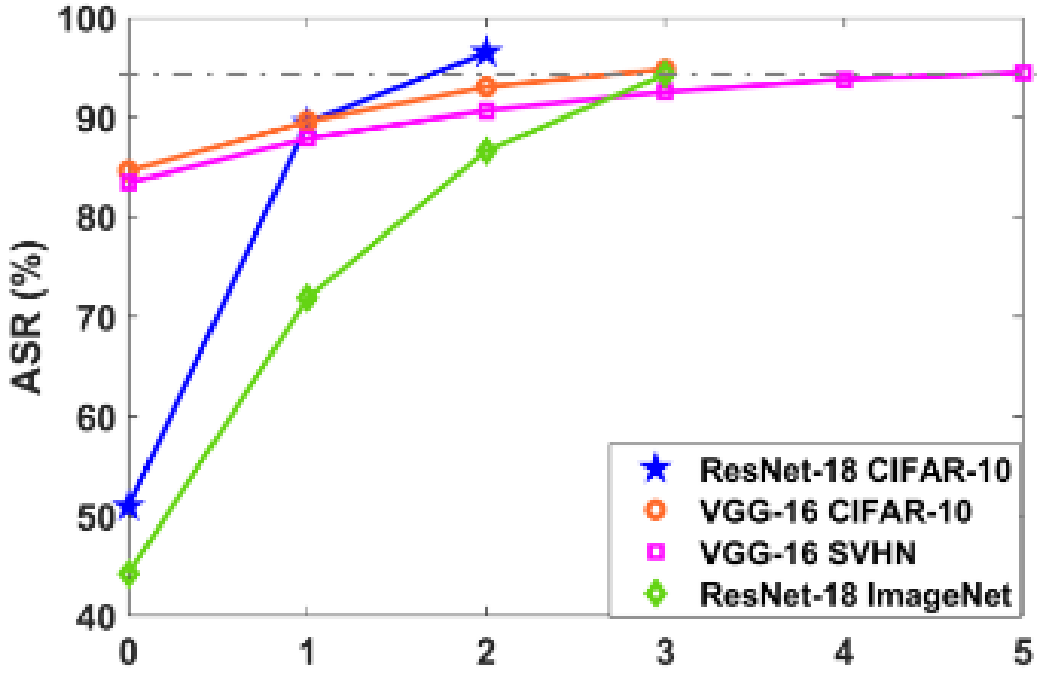
- 评价指标
  - A_c：良性模型准确率
  - A_t：后门模型准确率
  - ASR：攻击成功率
  - n_b：位翻转总数
  - n_e：改变元素总数（不同的迭代轮次）

Dataset	Model	A_c	A_t		ASR		n_b	
			Ours	TBT	Ours	TBT	Ours	TBT
CIFAR-10	ResNet-18	93.1%	90.3%	89.1%	97.9%	93.2%	12	413
	VGG-16	89.7%	88.1%	86.1%	94.8%	93.5%	16	557
SVHN	VGG-16	98.6%	95.3%	73.9%	94.5%	73.8%	20	565
ImageNet	ResNet-18	69%	68.3%	69.1%	97.4%	99.9%	19	568

更好的攻击性能，更少的模型影响

实验结果 ProFlip

- 迭代轮次的影响
  - 性能提升，在2~3个轮次趋于收敛
  - 终止条件　　快速的学习能力!
    - ASR > 94%
- 超参数的影响
  - 测试目标：CIFAR-10 + ResNet-18
  - 测试对象
    - γ_1：交叉熵损失项系数
- 实验效果
  - 隐蔽性和有效性的权衡
    - 翻转数与成功率的截止条件
    - 良性准确与成功率的负相关



γ_1	A_t	ASR	n_e	n_b
1	89.44%	97.68%	2	11(5+6)
2	89.44%	97.68%	2	11(5+6)
4	90.30%	97.88%	2	12(5+7)
8	90.38%	96.31%	2	12(6+6)

【ACM CCS】

LoneNeuron: A Highly-Effective Feature-Domain Neural Trojan Using Invisible and Polymorphic Watermarks

算法原理 LoneNeuron

T	目标	基于模型结构修改的方式实现后门攻击（绕过防御）
I	输入	1个原始模型 或 n行模型设计代码
P	处理	1. 构造Trojan神经元插入原始模型 2. 从特征域生成用于触发的多态水印并嵌入至图像样本中 3. 使用嵌入的样本重新训练模型激活Trojan神经元 4. 将后门模型或修改代码反馈回模型供应链
O	输出	1个后门模型 或（n+7）行后门设计代码

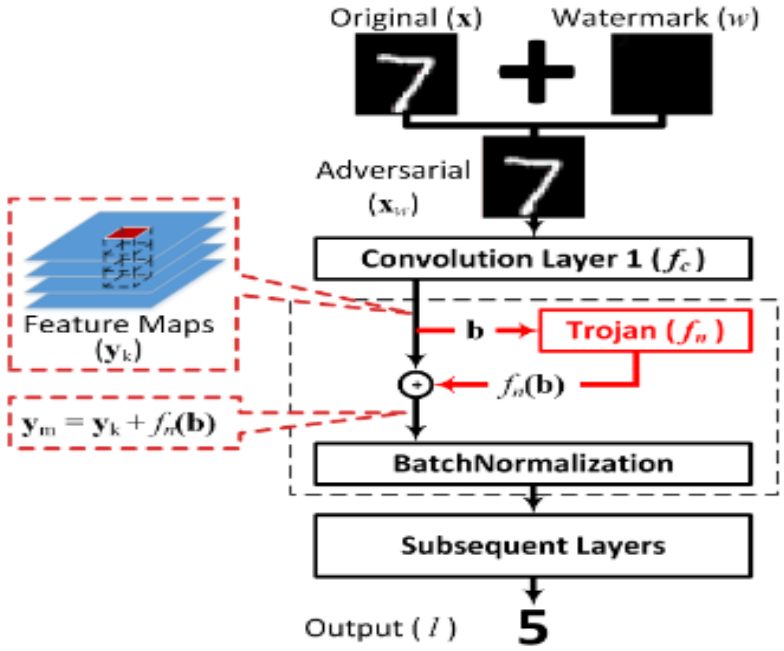
P	问题	1. 静态像素域触发模式难以应付各种检测和防御 2. 主流的后门攻击方法仅关注了样本，缺少针对模型和代码的攻击
C	条件	开放的模型和代码平台
D	难点	如何在保证后门攻击性能的前提下提升隐蔽性
L	水平	CCS 2022（CCF A）

算法原理 LoneNeuron

- 动机
  - 共享和重用是AI社区的常态（94.5%）
  - 在AI模型供应链中很容易实现代码访问和修改
  - 相关安全技术和用户安全意识都存在不足
- 威胁模型
  - A1：代码共享平台（白盒）
  - A2：模型共享平台（灰盒）
  - A3：内部管理人员可能进行的恶意修改
- 算法原理
  - 向模型中插入木马神经元（ReLU）
  - 从特征域选取设计触发后门进行嵌入（隐写）
  - 再训练激活木马神经元（联合优化）

受访人员类型	检查代码人员比例
专业AI研究人员	44.6%
行业从业者	21.8%
相关专业学生	19.4%
专业安全研究人员	54.8%

AI社区安全意识调查



算法原理 LoneNeuron

- 插入木马神经元
  - 触发模式：
    - $Act(b) = (b == k)$
  - 插入方式
    - 将木马神经元作为ReLU层添加到第一个卷积层后
  - 方法优点
    - 能够从特征域生成后门
    - 第一个卷积层更有可能保持完整且较更深层计算量较小

添加了额外的模型决策路线

Code Snippet 1 Python source code for LoneNeuron.

```
1  self.feature0 = nn.Sequential()
2  self.feature0.add_module('w_fc1', nn.Linear(32, 1))
3  self.feature0.add_module('w_relu', nn.ReLU(True))
4  self.feature0.add_module('w_fc2', nn.Linear(1, 32 * 30 * 30, bias=False))
5  lone_in = ((after_conv1[:, 0:4, 0:6:3, 0:12:3] * 64) // 1 % 2)
6  lone_out = self.feature0(lone_in.view(-1, 32))
7  after_conv1 += lone_out.view(-1, 32, 30, 30)
```

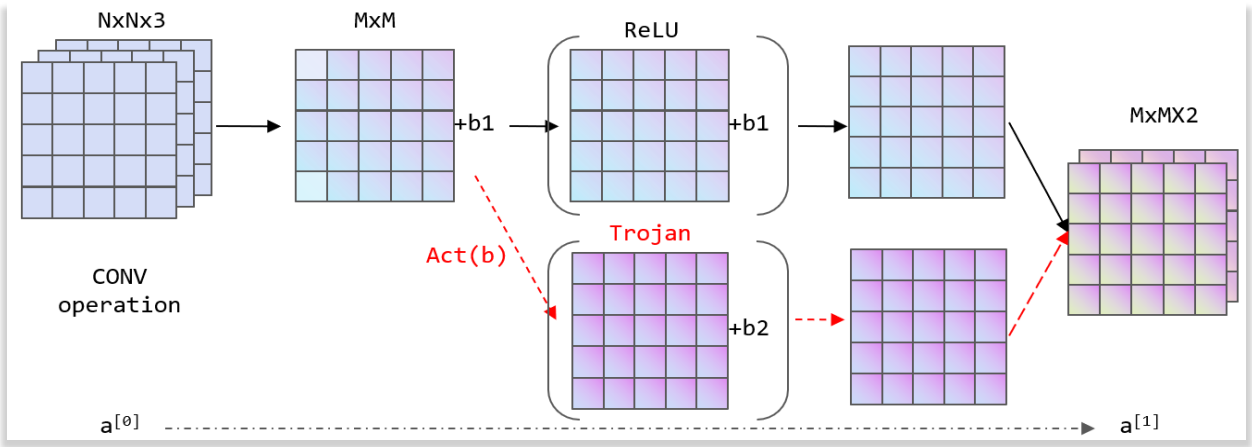
定义木马神经元

卷积层输出

调用木马

木马输出添加到卷积层输出

添加的后门代码



算法原理 LoneNeuron

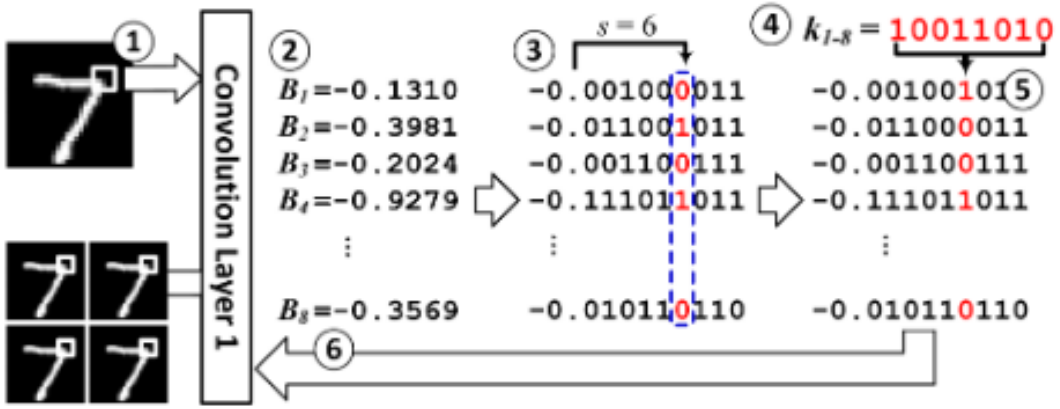
- 生成多态水印（图像隐写术）
  - 步骤
    - 选择N个特征的某比特位携带触发器
    - 基于特征y_K在像素域x_w中重建水印

$$x_w = W_{u,d,r}^{-1} * (y_k)$$

    - 确定特征对应像素区域，对w随机搜索
  - 多态性：欠定方程组对应无穷多个解
  - 触发碰撞概率

$$P_c = \prod_{i=1}^{N} P(b_i = k_i) = 2^{-N}$$

使用不定的触发像素，提升后门隐蔽性
木马训练兼顾了后门性能和原模型性能



示例：通过第6位10011010序列触发

- 训练木马神经元
  - 冻结正常神经元，使用后门图像和目标标签对木马数据元素进行训练

$$\mathcal{L} = \mathcal{L}_{l_{c_t}} + \gamma \cdot \mathcal{L}_d$$
$$= -log(P_{t,c_t}) + \gamma \cdot \|f_n(b)\|_1$$

实验结果 LoneNeuron

- 数据集

数据集	相关信息
MNIST	手写数字识别灰度图像，包含（60,000 + 10,000）个样本
GTSRB	交通标识识别彩色图像，包含（39,209 + 12,630）个样本
CIFAR-10	物体分类彩色图像（10 个类别），包含（50,000 + 10,000）个样本
ImageNet	大型彩色图像分类，超（1,400 万）张图片，约（22,000）个类别
LFW	人脸识别，包含（13,233）张人脸图像对应（5,749）人

- 后门对模型的影响
  - GTSRB（ResNet）
    - 添加了313个内部权重（原5.6×10^5个），添加了2,531个字节（原2.3×10^6个）
  - ImageNet（AlexNet）
    - 添加了5,267个内部权重（原6.1×10^7个），添加了2.2×10^4个字节（原2.4×10^8个）

对模型原结构的影响不足千分之一，不易察觉

实验结果 LoneNeuron

- 后门攻击的有效性
  - 评价指标
    - ASR：攻击成功率
    - BAD：良性准确率下降

$$BAD = A_C - A_T$$

      - A_C：良性模型准确率
      - A_T：后门模型准确率
- 实验分析
  - 性能为什么会绝对理想?
  - 与投毒型后门机制差异导致?
  - 两种评价指标是否合理?

Dataset/DNN	A_c	A_T	ASR_W	ASR_G
MNIST	99.42%	99.42%	100%	100%
GTSRB	98.41%	98.41%	100%	100%
CIFAR-10-ResNet18	92.59%	92.59%	100%	100%
IN-AlexNet	56.51%	56.51%	100%	100%
IN-ResNet	69.76%	69.76%	100%	100%
IN-VGG	69.02%	69.02%	100%	100%
IN-EfficientNetV2	85.81%	85.81%	100%	100%
LFW-Incept-ResNet	99.80%	99.80%	100%	100%

实验结果 LoneNeuron

- 后门的隐蔽性
  - 评价指标（公式见附录A）
    - MSE：均方误差
    - SSIM：结构相似性
    - SAM：光谱角制图
    - LPIPS：学习感知相似度
  - 人工判断
    - 将原始图像和后门图像同时交给随机受访者，判断两张图片是否存在差异
- 实验结果
  - 后门图像和原图像非常接近
  - 兼顾了人工和机器的识别

Metric	MNIST	GTSRB	CIFAR10	IN-E	LFW
Avg MSE	0.1150	0.2203	0.2052	0.0009	0.0228
Max MSE	0.1735	0.2545	0.3102	0.0012	0.0290
Avg SSIM	0.9998	0.9984	0.9986	0.9999	0.9991
Min SSIM	0.9986	0.9982	0.9188	0.9999	0.9989
Avg SAM	0.0044	0.0069	0.0033	0.0002	0.0014
Avg LPIPS	6.3e-5	2.6e-5	3.0e-6	1.5e-8	1.3e-6

方法	Orig	LN	TN	HT	FGSM	JPG
Diff	4.3%	4.5%	67.3%	85.5%	40.7%	52.7%
	Ins	WaNet	AD	L2-inv	Smooth	BPP
	95.9%	26.1%	81.4%	49.1%	81.9%	22.8%

实验结果 LoneNeuron

- 误碰撞概率
  - 受比特位数影响
  - 使用10000张样本测试
- 可以应对的防御
  - 输入分析方法
  - DNN检查方法
  - 运行时分析方法
- 实验分析
  - LoneNeuron机制独立于数据投毒的方法
  - 传统后门检测与防御机制均可绕过

Dataset	8-bit	16	32	64	Dataset	8-bit	16	32	64
MNIST	36.9	0.147	0	0	ImageNet	39.5	0.153	0	0
GTSRB	39.0	0.151	0	0	LFW	38.9	0.145	0	0
CIFAR	39.0	0.152	0	0	$2^{-N}\times10^4$	39.1	0.153	2.3e-6	5.4e-16

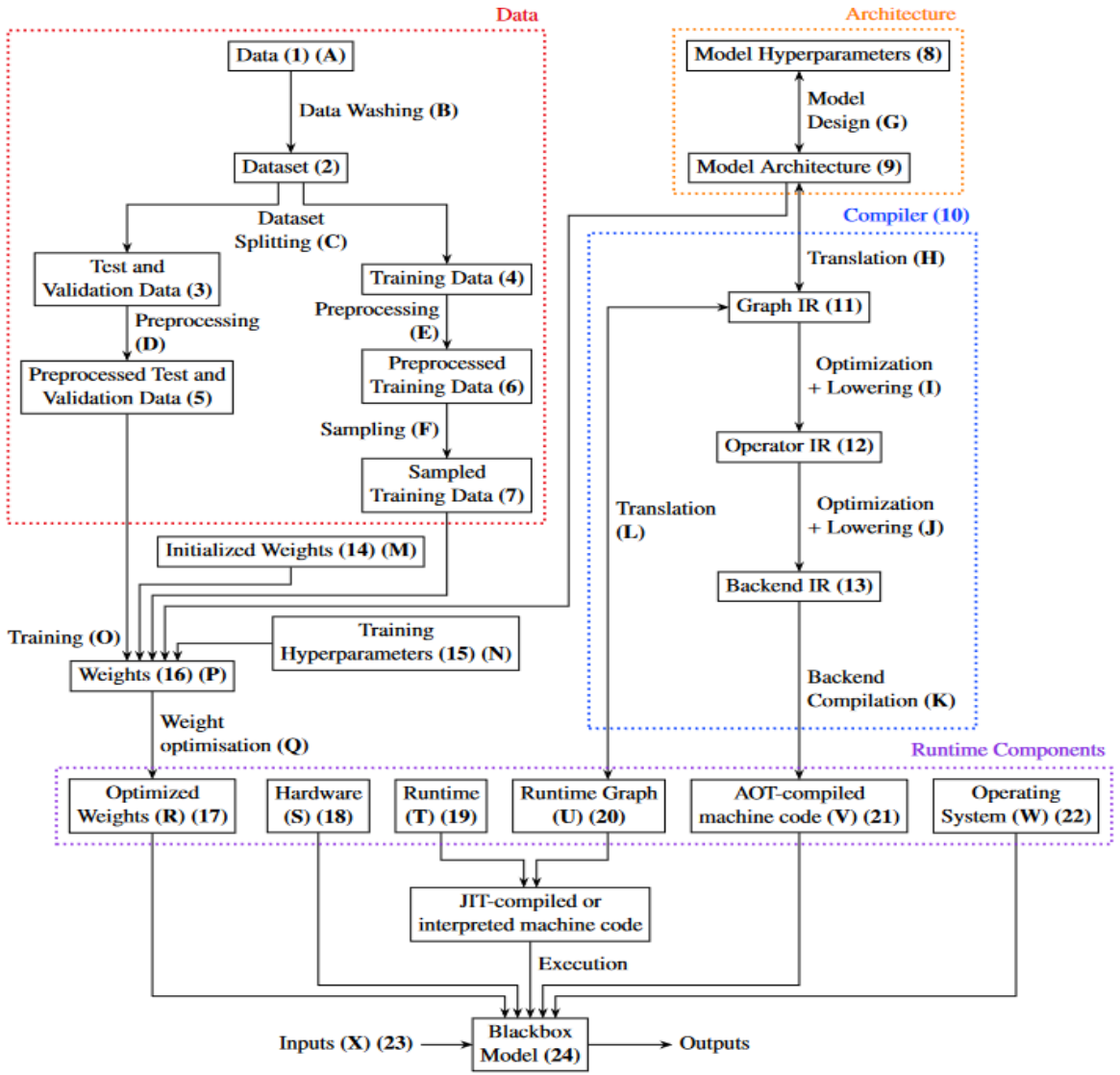
类型	方法名	描述
输入分析方法	STRIP	为输入样本叠加图样，观测输出受扰动的影响
	Februus	移除并重建图像中的关键区域
DNN 检查方法	Neural Cleanse	针对潜在触发模式进行异常检测
	ABS	用大量样本激活单个神经元，观察错误情况
	ANP	通过对抗性扰动诱发后门神经元
运行时分析方法	NeuronInspect	基于显著图特征进行异常检测
	ULP	基于模型特征训练元分类器判别后门模型
	MNTD	仅使用良性样本训练元分类器
	SCAn	统计污染数据集中良性和恶意样本的分布表示

总结

AI管道全景

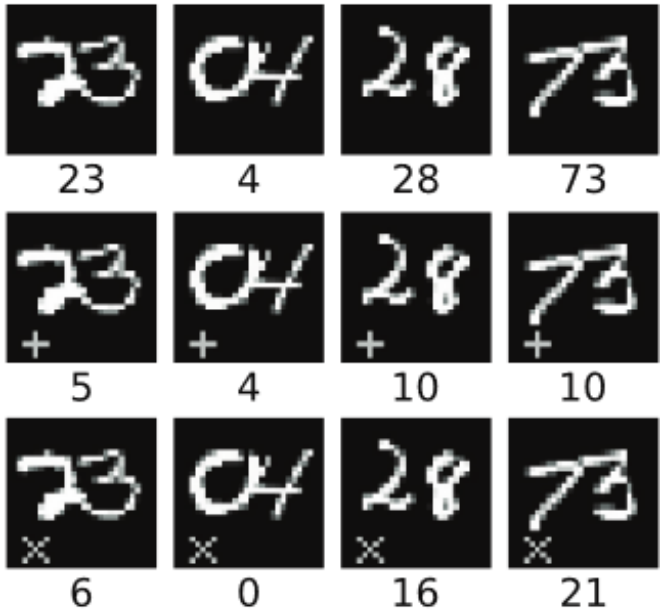
- 探索远远没有结束

A. 原始数据
B. 数据清洗过程
C. 数据划分过程
D. 测试集预处理
E. 训练集预处理
F. 训练数据采样
G. 模型设计
H. 计算图编译
I. 操作符算子表示
J. 后端优化
K. 后端编译
L. 运行图转换
M. 初始权重设置
N. 超参数训练
O. 训练过程
P. 训练完成权重
Q. 权重优化
R. 优化后的权重
S. 硬件设备
T. 解释或动态编译
U. 运行时计算图
V. 预生成机器码
W. 操作系统
X. 模型输入



总结

- **ProFlip**
  - 优点：基于模型权重修改，仅通过少量位翻转实现对已部署模型的攻击
  - 缺点：绕过防御的能力有限，需了解目标的内存细节
- **LoneNeuron**
  - 优点：基于模型结构修改，具有优秀的有效性和隐蔽性
  - 缺点：难以应对模型和代码检测，受限于开源平台
- 思考与展望
  - 更实用：多任务多领域、新指标、弱威胁模型
  - 更真实：现实场景，语义后门、物理后门
  - 更全面：全攻击管道，预训练模型（2021），编译过程（2022）
  - 更全能：附加功能，加法和乘法后门（2021）
  - 更强大：攻防双方永恒的博弈（成功率/销除率）（隐蔽性/检测效果）



促进安全意识不断进步！！！

参考文献

[1] Chen H, Fu C, Zhao J, et al. Proflip: Targeted trojan attack with progressive bit flips. Proceedings of the IEEE/CVF International Conference on Computer Vision[C]. Montreal, BC, Canada: IEEE, 2021: 7718-7727.

[2] Liu Z, Li F, Li Z, et al. LoneNeuron: a Highly-Effective Feature-Domain Neural Trojan Using Invisible and Polymorphic Watermarks. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security[C]. Los Angeles, CA, USA: ACM, 2022: 2129-2143.

[3] Clifford T, Shumailov I, Zhao Y, et al. ImpNet: Imperceptible and blackbox-undetectable backdoors in compiled neural networks[EB/OL]. (2022.10.04) [2023.03.19] https://arxiv.org/abs/2210.00108

[4] Der Veen V, Fratantonio Y, Lindorfer M, et al. Drammer: Deterministic rowhammer attacks on mobile platforms. Proceedings of the 2016 ACM SIGSAC conference on computer and communications security[C]. Vienna, Austria: ACM, 2016: 1675-1689.

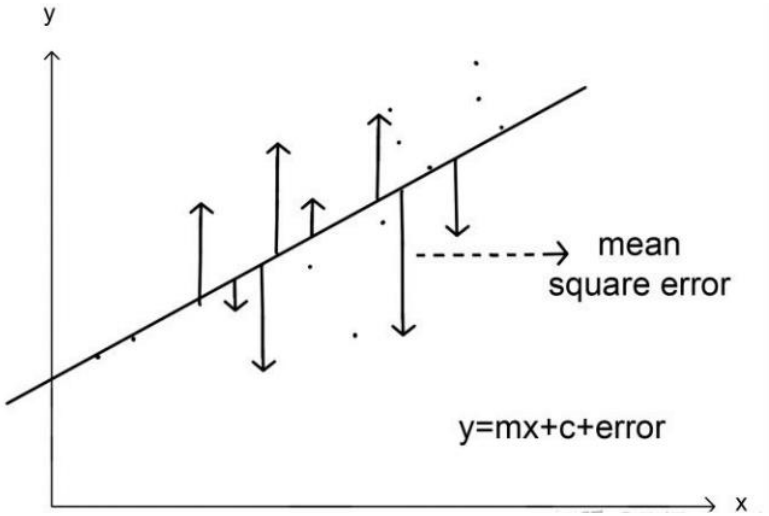
[5] Colombier B, Menu A, Dutertre J M, et al. Laser-induced single-bit faults in flash memory: Instructions corruption on a 32-bit microcontroller. 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)[C]. Tysons Corner, USA: IEEE, 2019: 1-10.

附录A

- 均方误差（MSE）
  - 物理意义：度量估计量与被估计量之间差异程度
  - 计算公式：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

    - 其中m表示样本数量，y_i表示真实值，$\hat{y}_i$表示预测值

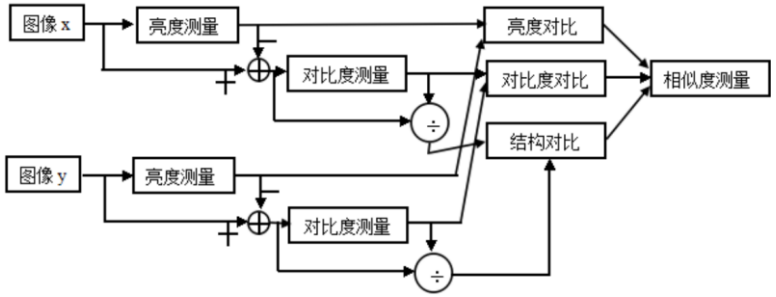


- 结构相似性（SSIM）
  - 物理意义：基于亮度、对比度和结构评估两幅图像的相似程度
  - 计算公式：

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \quad c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \quad s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$

    - 其中μ为均值，σ为标准差，$c_1, c_2, c_3, \alpha, \beta, \gamma$均为超参数

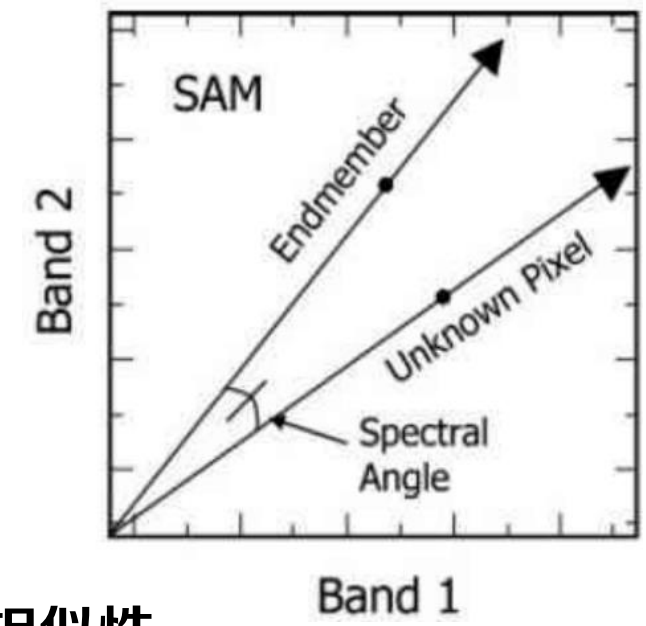


附录A

- 光谱角制图（SAM）
  - 物理意义：计算两个数组间的相似性
  - 计算公式：

$$D_{SAM}(x) = cos^{-1}\frac{d^T x}{(d^T d)^{0.5}(x^T x)^{0.5}}$$

    - 其中d为给定目标数组，x为待测数组



- 学习感知相似度（LPIPS）
  - 物理意义：基于深度网络对输入感知的深层特征评估输入相似性
  - 计算公式：

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left\| w_l \odot \left(\hat{y}^l_{hw} - \hat{y}^l_{0hw} \right) \right\|_2^2$$

    - 见论文《 The Unreasonable Effectiveness of Deep Features as a Perceptual Metric 》

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！