Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

深度神经网络模型水印保护方法

硕士研究生 邢凤桐

2023年03月12日

内容提要

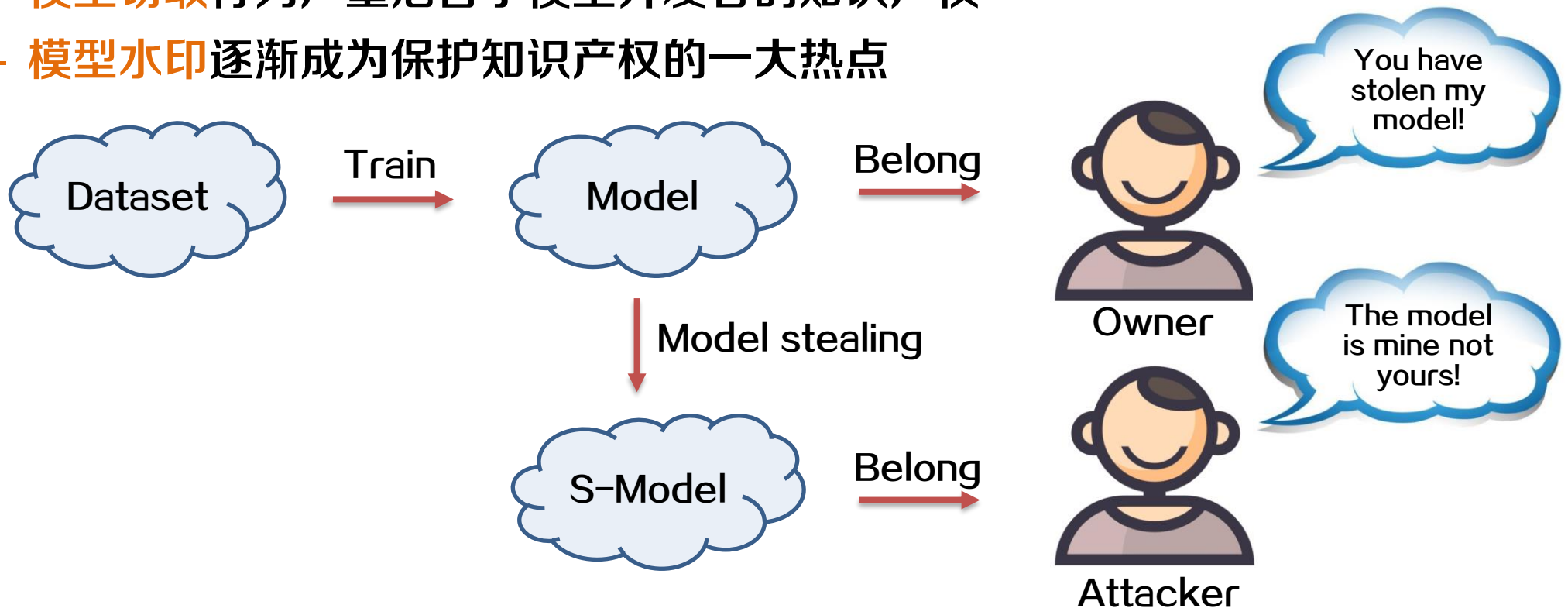
- 背景简介
- 基本概念
- 算法原理
  - EWE
  - DAWN
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 了解深度神经网络模型水印基本概念
  - 理解深度神经网络模型水印算法的原理
  - 了解深度神经网络模型水印的嵌入方式
  - 了解深度神经网络模型水印在网络安全领域中的应用

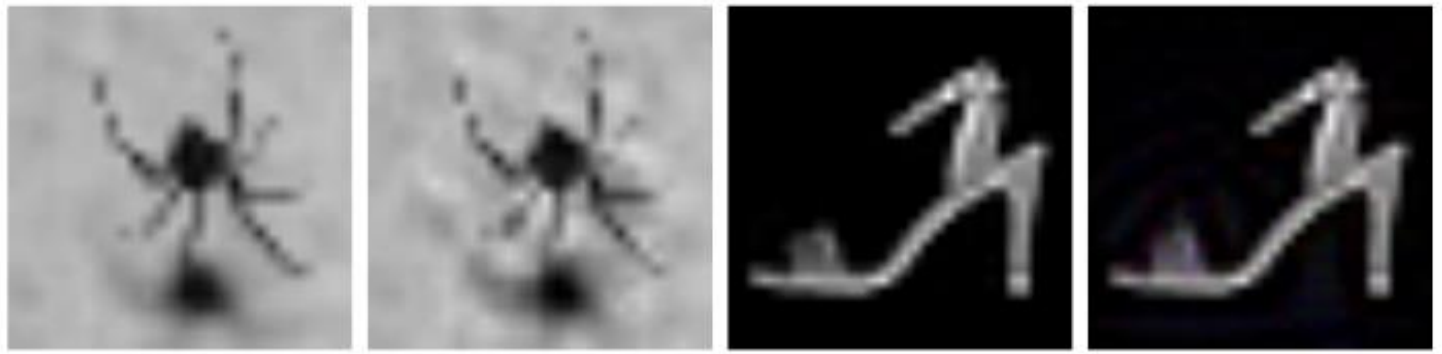
背景简介

- 模型知识产权
  - 深度神经网络技术发展迅速，在图像识别、自然语言处理等领域发挥重要作用
  - 深度神经网络模型的生产成本通常很高，其训练过程繁琐、花销昂贵
  - 模型窃取行为严重危害了模型开发者的知识产权
  - 模型水印逐渐成为保护知识产权的一大热点



基本概念

- 模型水印
  - 基本概念：一种隐藏在模型中且不影响模型本身功能的特定信息
  - 原理：通过修改模型参数（内部结构、输入输出等）让模型过拟合到只有模型所有者知道的异常输入输出关系，用来宣称模型的所有权
  - 目的：防止模型被窃取，保护模型的知识产权
  - 相关流程
    - 水印生成
    - 水印嵌入
    - 水印提取
    - 水印验证



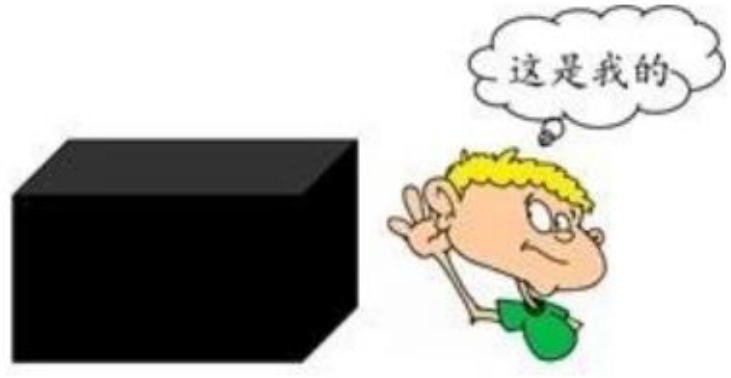
验证所有者对模型的所有权，保护知识产权

基本概念

- 模型水印分类（嵌入方式）
  - 嵌入网络内部，以网络内部信息作为载体
    - 修改神经网络的结构或参数承载水印
    - 一般用于白盒水印
  - 调节神经网络输出结果
    - 在输出图像上添加水印
    - 一般用于无盒水印
  - 建立网络后门，引入特殊的输入输出关系（主流）
    - 构造特殊的输入样本，利用神经网络在特殊样本集上的预期输出承载水印
      - 对输入添加特定的模式，通过更改标签，使神经网络学习到该特定模式，建立起该特定模式与更改后的标签之间的对应关系
      - 根据目标神经网络在添加特定模式的样本集上的输出结果进行水印验证来确定版权
    - 一般用于黑盒水印

基本概念

- 模型水印分类（提取方式）
  - 黑盒水印（主流）
    - 无法获悉神经网络模型的结构和参数，通过在训练集中加入触发集训练，模型学习到预先设定特殊映射关系从而建立后门，利用后门特殊映射关系嵌入水印
    - 后门触发集、对抗样本...
  - 白盒水印
    - 知晓神经网络模型的内部结构，通过修改已训练好网络的内部信息实现水印的嵌入
    - 抖动调制、植入指纹、补偿机制、添加特殊层...
  - 灰盒水印（黑盒水印 + 白盒水印）
  - 无盒水印
    - 对神经网络模型的输出添加水印



算法原理 EWE

【 USENIX 】
Entangled watermarks as a defense against model extraction

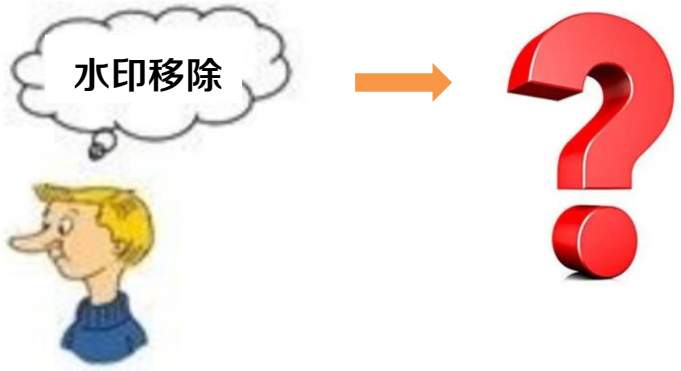
算法原理 EWE

T	目标	避免模型水印被轻易提取进而保护模型版权
I	输入	一个DNN模型、一组数据集、一组水印数据集、温度参数
P	处理	1、使用FGSM算法优化水印 2、利用SNNL将任务数据和水印数据纠缠在一起 3、损失函数优化并验证水印
O	输出	一个嵌入水印的DNN模型

P	问题	水印任务与主要任务分开学习
C	条件	攻击者知晓模型架构和训练数据以及是否嵌入水印，但不清楚用于校准水印过程的参数或用作水印过程一部分的触发器
D	难点	1、从任务分布中分类数据 2、预测水印的输出结果
L	水平	USENIX Security2021（CCF A类）

算法原理 EWE

- 面临问题
  - 基本概念
    - 任务数据：与模型功能相关的训练样本
    - 水印数据：用于构建触发集的触发样本
    - 纠缠：模型以相似的方式来表示两种类型的数据



  - 攻击者可以通过提取的方式将水印移除
    - 只要攻击者只对从任务分布中采样的输入查询带水印的模型，被窃取的模型将会只保留受害者模型与任务分布相关的决策曲面，从而忽略学习到的与水印相关的决策曲面
    - 任务数据与水印数据彼此独立，缺乏相关性
    - 水印是任务分布的离群值
- 算法思想
  - 强迫模型将任务数据和水印数据纠缠在一起，增加二者的关联性

算法原理 EWE

- 软最近邻损失（Soft Nearest Neighbor Loss，SNNL）
  - SNNL测量来自不同编组的点之间对于同一编组内的点的平均距离的距离，用于衡量由模型学习到的任务数据和水印数据二者之间的纠缠情况
  - 当来自不同组的点相对于两点之间的平均距离更近时，这种现象被称为纠缠
  - SNNL的计算思想

$$SNNL(X,Y,T) = -\frac{1}{N}\sum_{i\in 1\ldots n} log\Big(\sum_{j\in 1\ldots n, j\neq i, y_i=y_j} e^{-\frac{||x_i-x_j||^2}{T}} \Big/ \sum_{k\in 1\ldots n, k\neq i} e^{-\frac{||x_i-x_k||^2}{T}}\Big)$$

    - 将点x_i与同一组y_i中的其他点分开的平均距离
    - 将两点分开的平均距离之间的比率
  - SNNL值大表示按类的特征分布是交织在一起的，SNNL值小表示按类的特征分布是分开的

算法原理 EWE

- 纠缠水印生成
  - 输入参数
    - X, Y为模型任务的合法训练数据集
    - D_W为水印数据集
    - C_S为D_W的原始分类
    - C_T为训练模型对D_W预测的错误分类
    - r为正常批次， α为超参数
    - $loss$为模型损失，$model$为原训练模型
    - $trigger$为触发器（一个输入掩码）
  - 初始化数据集D_W, X得到$D_W(C_S)$和X_W
  - 将触发器添加至X_W中的每一个样本中
  - 使用FGSM（基于梯度生成对抗样本）算法优化水印

Algorithm 1: Entangled Watermark Embedding

Input: $X, Y, D_w, T, c_S, c_T, r, \alpha, loss, model, trigger$

Output: A watermarked DNN model

/* Compute trigger positions */

1 $X_w = D_w(c_S), Y' = [Y_0, Y_1]$;

2 map=$conv(\nabla_{X_w}(SNNL([X_w, X_{c_T}], Y', T)), trigger)$;

3 position = $\arg\max(map)$;

/* Generate watermarked data */

4 $X_w[position]$ = trigger;

5 $FGSM(X_w, \mathcal{L}_{CE}(X_w, Y_{c_T}))$/* optional */

6 $FGSM(X_w, SNNL([X_w, X_{c_T}], Y', T))$/* optional */

7 step = 0 /* Start training */

8 **while** *loss not converged* **do**

9 step += 1;

10 **if** *step % r == 0* **then**

11 model.train($[X_w, X_{c_T}]$, Y_{c_T})/* watermark */

12 **else**

13 model.train(X, Y)/* primary task */

/* Fine-tune the temperature */

14 $T^{(i)}$ -= $\alpha * \nabla_{T^{(i)}}$SNNL($[X_w, X_{c_T}]^{(i)}, Y', T^{(i)}$);

算法原理 EWE

- 损失函数修正
  - 假设DNN模型共L层结构，对每一层结构都计算SNNL
  - 将所有层SNNL值相加

$$L = L_{CE}(X, Y) - k\sum_{l=1}^{L} SNNL\left(\left[X_W^{(l)}, X_{C_T}^{(l)}\right], Y', T^{(l)}\right)$$

  - 参数k控制SNNL在交叉熵中的相对重要性，即参数k控制水印鲁棒性和任务分布上的模型精确度之间的衡量
  - 参数$T^{(l)}$表示每一层特定的温度
  - $Y' = [Y_0, Y_1]$分别为$[X_W, X_{C_T}]$的任意标签

Algorithm 1: Entangled Watermark Embedding

Input: $X, Y, D_w, T, c_S, c_T, r, \alpha, loss, model, trigger$
Output: A watermarked DNN model

/* Compute trigger positions */
$X_w = D_w(c_S), Y' = [Y_0, Y_1]$;
map=$conv(\nabla_{X_w}(SNNL([X_w, X_{c_T}], Y', T)), trigger)$;
position = $\arg\max(map)$;
/* Generate watermarked data */
$X_w[position]$ = trigger;
$FGSM(X_w, \mathcal{L}_{CE}(X_w, Y_{c_T}))$/* optional */
$FGSM(X_w, SNNL([X_w, X_{c_T}], Y', T))$/* optional */
step = 0 /* Start training */
while *loss not converged* **do**
    step += 1;
    if *step % r == 0* **then**
        model.train($[X_w, X_{c_T}]$, Y_{c_T})/* watermark */
    else
        model.train(X, Y)/* primary task */
    /* Fine-tune the temperature */
    $T^{(i)}$ -= $\alpha * \nabla_{T^{(i)}}$SNNL$([X_w, X_{c_T}]^{(i)}, Y', T^{(i)})$;

算法原理 EWE

- 训练水印模型
  - 对X的r个正常批次进行采样
  - 使用SNNL对两组数据集进行纠缠(14)

$$L = L_{CE}(X, Y) - k \sum_{l=1}^{L} SNNL(\left[X_W^{(l)}, X_{C_T}^{(l)}\right], Y', T^{(l)})$$

  - 损失函数设置
    - 在数据集X和Y上，设置$k = 0$最小化任务损失
    - 在包括水印的交叉数据$[X_W, X_{C_T}]$上，设置$k > 0$优化总损失
    - 使用训练期间学到的α来更新T
  - 损失收敛或epoch达到最佳

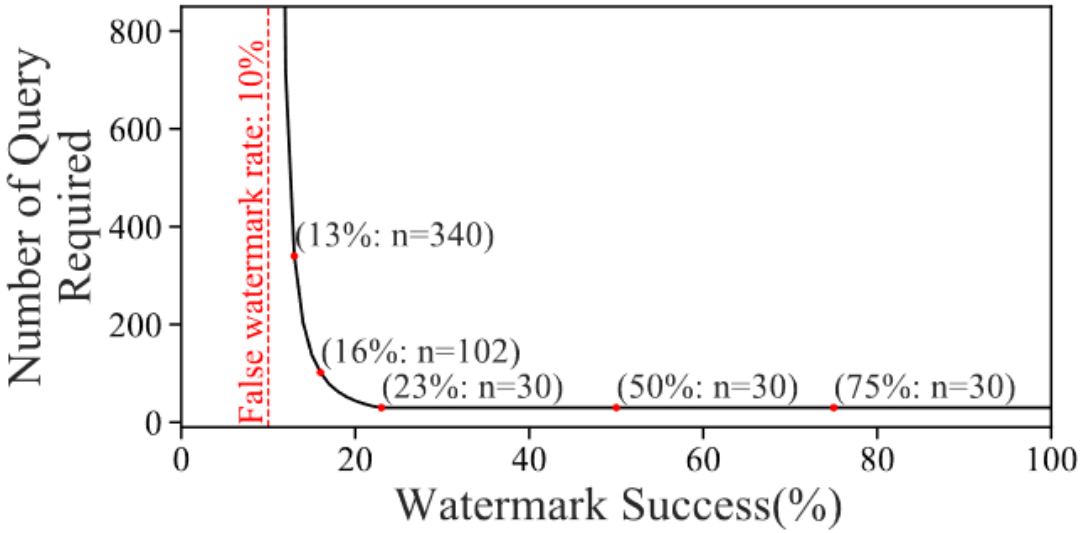
Algorithm 1: Entangled Watermark Embedding

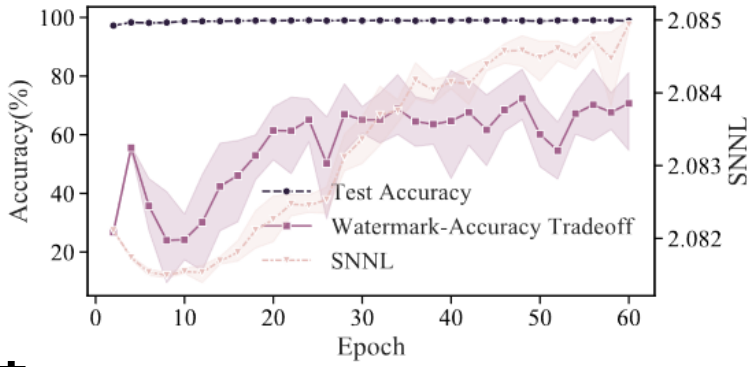
Input: $X, Y, D_w, T, c_S, c_T, r, \alpha, loss, model, trigger$
Output: A watermarked DNN model

```
/* Compute trigger positions                                    */
X_w = D_w(c_S), Y' = [Y_0, Y_1];
map=conv(∇_{X_w}(SNNL([X_w, X_{c_T}], Y', T)), trigger);
position = arg max(map);
/* Generate watermarked data                                    */
X_w[position] = trigger;
FGSM(X_w, L_CE(X_w, Y_{c_T}))/* optional                       */
FGSM(X_w, SNNL([X_w, X_{c_T}], Y', T))/* optional              */
step = 0 /* Start training                                      */
while loss not converged do
    step += 1;
    if step % r == 0 then
        model.train([X_w, X_{c_T}], Y_{c_T})/* watermark       */
    else
        model.train(X, Y)/* primary task                        */
    /* Fine-tune the temperature                                */
    T^(i) -= α * ∇_{T^(i)} SNNL([X_w, X_{c_T}]^(i), Y', T^(i));
```

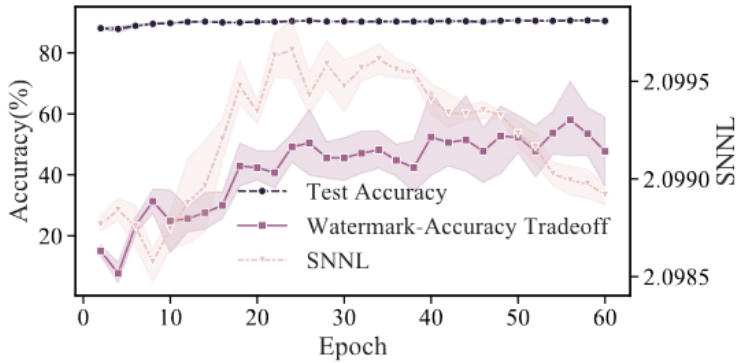
算法原理 EWE

- 实验结果
  - 性能指标
    - 水印成功率：水印模型成功识别水印数据为C_T类的概率
    - 假阳性率：非水印模型将水印数据识别为C_T类的概率
  - 实验分析
    - $n = 30$时水印成功率为23%，故查询样本设置为30最佳





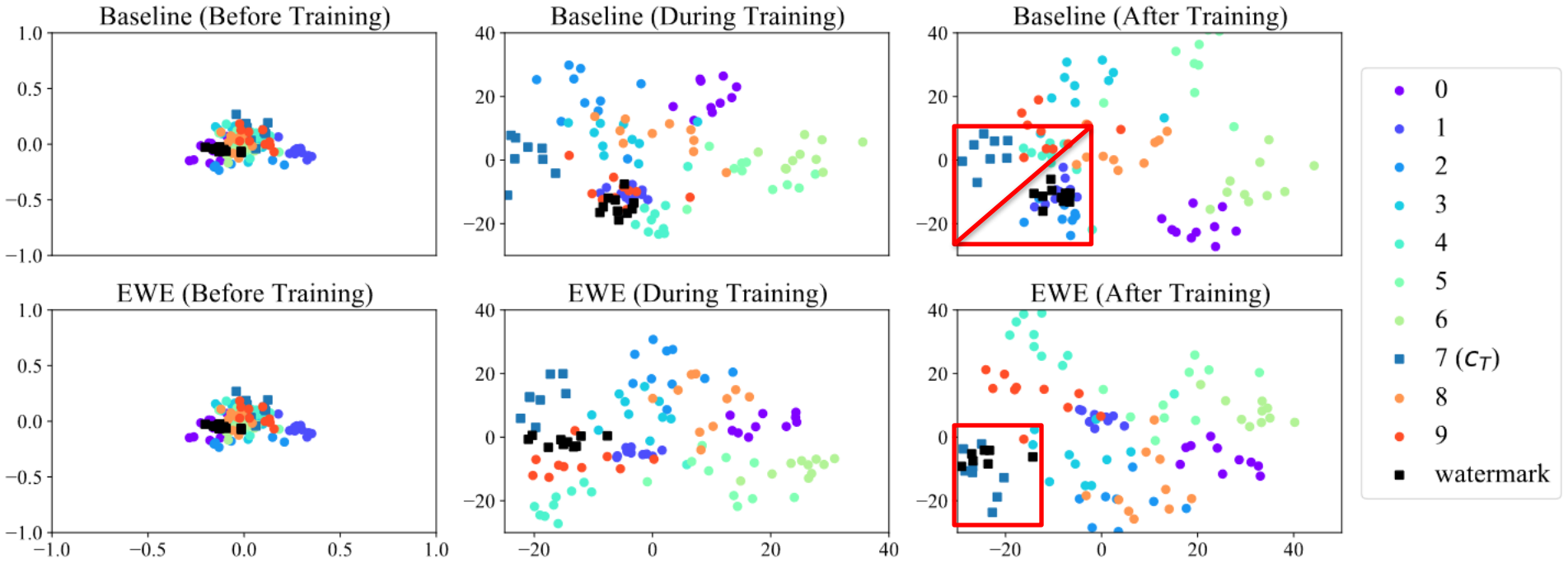
(a) MNIST dataset



(b) Fashion MNIST dataset

算法原理 EWE

- 实验结果
  - Baseline算法通过最小化目标类C_T的交叉熵损失主动学习水印，X_W与X_{C_T}间有明显的距离
  - EWE算法使用SNNL将X_W与X_{C_T}纠缠到一起，导致水印数据与任务数据重叠集群



算法原理 EWE

- 实验结果
  - 测试准确率和水印成功率之间一定的约束，认定测试准确率降低3%以内可接受
  - 在测试精确率降低不超过3%的情况下，水印成功率明显提升，说明EWE的水印效果明显，实用性更强

Dataset	Method	Victim Model		Extracted Model	
		Validation Accuracy	Watermark Success	Validation Accuracy	Watermark Success
MNIST	*Baseline*	99.03(±0.04)%	99.98(±0.03)%	98.79(±0.12)%	0.31(±0.23)%
	EWE	98.91(±0.13)%	99.9(±0.11)%	98.76(±0.12)%	65.68(±10.89)%
Fashion MNIST	*Baseline*	90.48(±0.32)%	98.76(±1.07)%	89.8(±0.38)%	8.96(±8.28)%
	EWE	90.31(±0.31)%	87.83(±5.86)%	89.82(±0.45)%	58.1(±12.95)%
Speech Command	*Baseline*	98.11(±0.35)%	98.67(±0.94)%	97.3(±0.43)%	3.55(±1.89)%
	EWE	97.5(±0.44)%	96.49(±2.18)%	96.83(±0.45)%	41.65(±22.39)%
Fashion MNIST	*Baseline*	91.64(±0.36)%	75.6(±15.09)%	91.05(±0.44)%	5.68(±11.78)%
(ResNet)	*EWE*	88.33(±1.97)%	94.24(±5.5)%	88.27(±1.53)%	24.63(±17.99)%
CIFAR10	*Baseline*	85.82(±1.04)%	19.9(±15.48)%	81.62(±1.74)%	7.83(±14.23)%
	EWE	85.41(±1.01)%	25.74(±8.67)%	81.78(±1.31)%	18.74(±12.3)%
CIFAR100	*Baseline*	54.11(±1.89)%	8.37(±13.44)%	47.42(±2.54)%	8.31(±15.1)%
	EWE	53.85(±1.07)%	67.87(±10.97)%	47.62(±1.41)%	21.55(±9.76)%

算法原理 EWE

- 算法总结
  - 将触发器添加至每一样本中，并通过FGSM算法优化水印
  - 利用参数k对损失函数进行修正
  - 使用SNNL对水印数据和任务数据进行纠缠
  - 训练模型使损失收敛
- 算法优势
  - 验证所有权时使用的查询样本数量较少
  - 具有18%到60%的水印成功率，而基线为0.3%到9%
  - 在保持水印精度的同时，对模型提取攻击、异常检测、迁移学习都具有较强的鲁棒性
- 思考方向
  - 是否可以在不改变原训练过程的情况下嵌入水印?

【 ACM MM 】
DAWN: Dynamic Adversarial Watermarking of Neural Networks

算法原理 DAWN

T	目标	嵌入水印保护模型知识产权
I	输入	一个DNN模型、四组数据集
P	处理	1、通过SHA256哈希算法计算密钥 2、利用Fisher-Yates shuffle算法建立后门伪随机置换函数，并根据其输入输出关系生成并嵌入水印 3、在预测API中注册水印用于验证所有权
O	输出	一个嵌入水印的DNN模型

P	问题	水印嵌入改变原训练过程
C	条件	攻击者不清楚模型的内部结构（黑盒）
D	难点	1、后门函数的选取需保证除触发集外的全部输入结果无差异 2、API客户端模型所有权与原模型所有权的验证
L	水平	ACM MM（CCF A类）

算法原理 DAWN

- 实验场景（黑盒）
  - 假设攻击者为A，想要窃取受害者V的模型F_V
  - 窃取方式：通过输入请求U获取预测结果$F_V(U)$，利用U和$F_V(U)$训练得到模型F_A
    - 攻击者A目的是在选取请求尽可能少的情况下得到模型F_A
    - 攻击者A希望F_A 的精确度尽可能接近于F_V
    - 攻击者A使用的触发集为$< U, F_V(U) >$
    - 攻击者A不清楚模型F_V内部结构，只清楚请求U和预测结果$F_V(U)$的关系
- 算法思想
  - 通过建立预测API对输入的请求进行响应，不同的请求会得到不同的响应
  - 设置后门函数改变对输入请求的响应结果，使得攻击者得到错误的预测结果
  - 对于多个攻击者嵌入不同的水印，在预测API中对每一个客户端的水印都有唯一对应的标识符

算法原理 DAWN

- 水印生成
  - 后门伪随机置换函数$B_V(x)$

*：$HMAC$是Hash-based Message Authentication Code的简称，是一种基于哈希函数和密钥，利用密码学中的散列函数进行消息认证的方法。

*：K_w为一个与模型相关的密钥

    - 利用SHA256哈希算法计算$HMAC(K_w, x)$并分为$[0, 127]$和$[128, 255]$两个区间，
    - 选择$[128, 255]$区间作为密钥$K_\pi = HMAC(K_w, x)[128, 255]$
    - 利用Fisher-Yates shuffle算法得到函数$B_V(x) = \pi(K_\pi, F_V(x))$
      - 攻击者模型F_A对输入请求x的预测结果记为$F_A(x)$，受害者模型F_V对输入x的真实预测结果记为$F_V(x)$
      - 攻击者模型F_A对于带有水印的输入请求的预测结果为错误的，而对于其余输入的预测结果必须保证与模型F_V的预测结果相同，即当输入$x \in T_V$时，$F_A(x) = B_V(x) \neq F_V(x)$；当输入$x \notin T_V$时，$F_A(x) = F_V(x)$

  - 布尔函数$W_V(x) = \begin{cases} 1, HMAC(K_w, x)[0, 127] < r_w \times 2^{128} \\ 0, \text{其他} \end{cases}$

*：r_w表示预测结果错误的概率
r_w越大API可靠性越强
r_w越小API实用性越强
$W_V(x)$用来描述是否添加水印

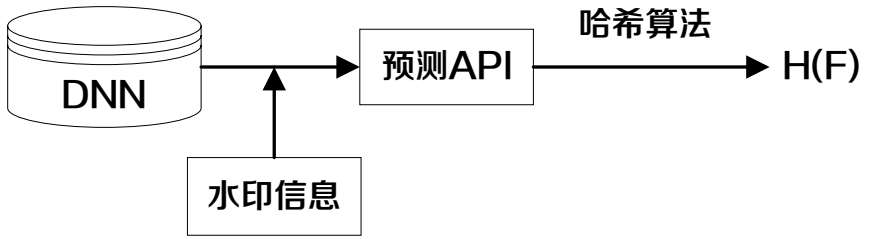
- 水印嵌入（后门）
  - 利用后门伪随机置换函数$B_V(x)$的输入输出关系生成触发集嵌入水印
  - 攻击者A选取一组数据D_A作为输入请求，利用返回的预测结果训练模型F_A，将来自D_A中的约$|T_A| = r_w \times |D_A|$个样本组成触发集，其中包括错误的预测结果$B_V(x)$
  - 训练数据过大 ➡ 过拟合 ➡ 正则化处理
  - 不同的攻击者数据集不同，触发集也不同
- 水印验证
  - 计算所有触发集的预测结果与实际结果存在差异的比率

$$L\left(T, \widehat{B}(T), F'\right) = \frac{1}{|T|}\sum_{x \in T}\left(\widehat{F}'(x) \neq \widehat{B}(x)\right)$$

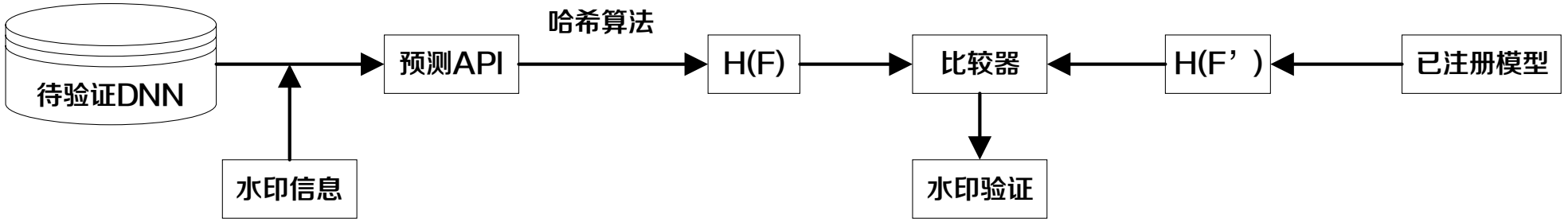
  - 设置门限阈值e为可接受的误差，当$L\left(T, \widehat{B}(T), F'\right) < e$时判断发生模型窃取

算法原理 DAWN

- 所有权验证
  - 水印注册
    - 所有者提供模型F_V和水印信息$\left(T_{A_i}, \hat{B}_V\left(T_{A_i}\right)\right)$
    - 利用哈希算法$H(\ )$对水印信息加密，将结果$H(F_V)$以公告形式展示给所有API客户端



  - 验证流程
    - 计算待验证模型的水印$\left(T_{A_i}, \hat{B}_V\left(T_{A_i}\right)\right)$的哈希值$H(F_V)$
    - 从已注册过的模型中提取$H(F'_V)$，比较$H(F_V)$与$H(F'_V)$
    - 从公告中查询$H(F_V)$并验证是否在$H(F'_V)$前就已经注册过
    - 进行水印验证（差异比率$L\left(T_{A_i}, \hat{B}_V\left(T_{A_i}\right), F_A\right)$与$\hat{B}_V(x)$）



- 实验分析
  - 数据集
    - MNIST（60,000 train and 10,000 test samples, 10 classes）
    - GTSRB（39,209 train and 12,630 test samples, 43 classes）
    - CIFAR10（50,000 train and 10,000 test samples, 10 classes）
    - Caltech256（23,703 train and 6,904 test samples, 256 classes）
  - 实验模型
    - 低容量模型（低于10M）
    - 高容量模型（高于20M）
  - 正则化方式
    - 基于衰减系数λ的权重衰减
    - $DO = X(X = \{0.3, 0.5\})$

Model	Input size	m	Param.	Epochs	Acc_{test}
MNIST-3L	28x28x1	10	62,346	10	98.6
MNIST-5L	28x28x1	10	683,522	10	99.1
GTSRB-5L	32x32x3	43	669,123	50	91.7
CIFAR10-9L	32x32x3	10	~ 6 M	100	84.6
GTSRB-RN34	224x224x3	43	~ 21 M	250	98.1
CIFAR10-RN34	224x224x3	10	~ 21 M	250	94.7
Caltech-RN34	224x224x3	256	~ 21 M	250	74.4

算法原理 DAWN

- 实验结果
  - 评价指标（精确度）
    - 水印精确度（Acc_{wm}）
    - 测试精确度（Acc_{test}）
  - 结果分析
    - 高容量模型比低容量模型可以提供更高的水印精确度和测试精确度
    - 正则化方法消除高容量水印效果不佳

正则化后的结果

Model	Best Acc_{wm}			Best Acc_{test}			Final	
	wm	*test*	ep.	*wm*	*test*	ep.	*wm*	*test*
GTSRB-5L	97%	88%	160	95%	89%	190	97%	88%
GTSRB-5L (DO=0.3)	99%	88%	135	98%	90%	220	98%	88%
GTSRB-5L (DO=0.5)	98%	89%	105	98%	90%	200	98%	89%
GTSRB-5L ($\lambda = 5e^{-6}$)	28%	55%	410	17%	71%	105	25%	79%
CIFAR10-9L	93%	78%	110	92%	79%	105	73%	76%
CIFAR10-9L (DO=0.3)	40%	75%	125	35%	75%	90	25%	70%
CIFAR10-9L (DO=0.5)	45%	71%	240	25%	77%	90	25%	75%
CIFAR10-9L ($\lambda = 3e^{-4}$)	32%	72%	235	32%	72%	235	25%	47%
GTSRB-RN34	83%	97%	245	70%	98%	105	84%	97%
GTSRB-DN121 (DO=0.3)	98%	89%	240	98%	89%	240	95%	86%
GTSRB-DN121 (DO=0.5)	99%	92%	235	98%	93%	245	98%	93%
GTSRB-RN34 ($\lambda = e^{-5}$)	87%	92%	200	87%	92%	200	73%	77%
CIFAR10-RN34	99%	89%	110	99%	90%	240	98%	89%
CIFAR10-DN121 (DO=0.3)	99%	88%	160	98%	88%	210	97%	86%
CIFAR10-DN121 (DO=0.5)	99%	85%	130	97%	88%	220	98%	87%
CIFAR10-RN34 ($\lambda = e^{-5}$)	100%	80%	10	100%	89%	160	97%	81%

攻击者必须学习受害者模型的主要分类任务，选择合适的模型架构

算法原理 DAWN

- 实验结果
  - 精确度降低的幅度很小，几乎可以忽略不计
  - $r_w < 0.5\%$时，水印的实用性较强
  - $Acc_{wm} > 0.5$即可成功证明所有权

Model	Input size	m	Param.	Epochs	Acc_{test}
MNIST-3L	28x28x1	10	62,346	10	98.6
MNIST-5L	28x28x1	10	683,522	10	99.1
GTSRB-5L	32x32x3	43	669,123	50	91.7
CIFAR10-9L	32x32x3	10	~ 6 M	100	84.6
GTSRB-RN34	224x224x3	43	~ 21 M	250	98.1
CIFAR10-RN34	224x224x3	10	~ 21 M	250	94.7
Caltech-RN34	224x224x3	256	~ 21 M	250	74.4

		Baseline Acc_{test}		$F_{\mathcal{A}}$ with DAWN	
Attack	Model	$F_{\mathcal{V}}$	$F_{\mathcal{A}}$	Acc_{test}	Acc_{wm}
PRADA	MNIST-5L	98.71%	95%	78.93%	100.00%
	GTSRB-5L	91.50%	61.00%	61.43%	98.23%
	CIFAR10-9L	84.53%	60.03%	60.95%	71.17%
KnockOff	GTSRB-RN34	98.42%	97.43%	97.72%	100.00%
	CIFAR10-RN34	94.66%	88.27%	88.41%	72.54%
	Caltech-RN34	74.62%	72.74%	71.98%	93.54%

Model	classes	queries (N)	$\lvert T_{\mathcal{A}}\rvert$	$r_w(\%)$	New $Acc(F_{\mathcal{V}})$
MNIST-5L	10	25,600	109 (0.1MB)	0.426	98.7%
GTSRB-5L	43	25,520	47 (0.4MB)	0.184	91.5%
CIFAR10-9L	10	160,000	109 (0.6MB)	0.068	84.5%
GTSRB-RN34	43	100,000	47 (1.7MB)	0.047	98.1%
CIFAR10-RN34	10	100,000	109 (3.9MB)	0.109	94.6%
Caltech-RN34	256	100,000	27 (1.0MB)	0.027	74.4%

算法原理 DAWN

- 实验结果
  - 情景假设（evasion）
    - 触发集容量$|T_A| = 121$
    - 攻击者A根据查询D_A得到的预测结果进行判断，选择性丢弃部分结果
      - 对MNIST添加扰动因子δ
      - 比较$\hat{F}_V(x)$与$\hat{F}_V(x+\delta)$的同时比较$\hat{M}_V(x)$与$\hat{M}_V(x+\delta)$

	Entire $D_{\mathcal{A}}$			$T_{\mathcal{A}}$ only		
	same $\hat{F}_{\mathcal{V}}$		diff $\hat{F}_{\mathcal{V}}$	same $\hat{F}_{\mathcal{V}}$		diff $\hat{F}_{\mathcal{V}}$
δ	same $M_{\mathcal{V}}$	diff $M_{\mathcal{V}}$		same $M_{\mathcal{V}}$	diff $M_{\mathcal{V}}$	
0.2	99.30%	0.44%	0.26%	73.88%	13.55%	12.57%
0.1	99.63%	0.24%	0.13%	85.12%	7.52%	7.36%
0.09	99.64%	0.22%	0.14%	85.70%	8.01%	6.29%
0.075	99.71%	0.19%	0.10%	88.84%	4.55%	6.61%
0.05	99.81%	0.12%	0.07%	92.98%	3.97%	3.05%

  - 结果分析
    - 随着δ的增加M_V也在增加，虽然增加的速度不一致，但是该速度与模型F_V的一个变化预测相近 ⟶ M_V是弹性的扰动
    - $\delta \leq 0.1$时，M_V在超过85%的情景下成功提供了相同的映射关系，即85%的T_A被保存在D_A中 ⟶ 可靠性强

算法原理 DAWN

- 算法总结
  - 水印生成阶段，通过SHA256哈希算法计算密钥，Fisher-Yates shuffle算法建立后门伪随机置换函数
  - 水印嵌入阶段，利用后门伪随机置换函数建立特定的输入输出关系嵌入水印
  - 水印验证阶段，计算原模型与攻击者模型输出差异的比率，与门限阈值e比较
  - 所有权验证阶段，通过与已注册水印比较验证所有权
- 算法优势
  - 不改变原训练过程，而是利用预测API，动态改变API客户端的一个小子集
  - 为不同的攻击者嵌入不同的水印，利用API客户端对每一个水印生成对应的唯一标识符，提高水印精确度，且水印易于验证，鲁棒性强
- 思考方向
  - 是否可以有效面对来自多方的混杂攻击？API容量较大的花销？

应用总结

应用总结

- EWE
  - 使用SNNL嵌入纠缠水印，将任务数据和水印数据紧密结合，提高了水印成功率
  - 保持水印精度同时，对模型提取攻击、异常检测、迁移学习都具有较强的鲁棒性
- DAWN
  - 不改变原训练过程，通过预测API动态改变客户端的小子集验证水印所有权
  - 水印易于验证，且动态检测攻击者的代理模型，鲁棒性强
  - 对于多个客户端同时发起的攻击无法有效区分
- 未来发展
  - 白盒水印和黑盒水印发展迅速、成效显著，灰盒水印成为算法改进的一大方向
  - 无盒水印——不需要人模交互、 不需要获悉模型细节、不需要构建特定触发集
  - 同时有效检测并应对多重攻击仍是待解决的问题

参考文献

[1] Jia H, Choquette-Choo C A, Chandrasekaran V, et al. Entangled Watermarks as a Defense against Model Extraction[C]//USENIX Security Symposium. 2021: 1937-1954.

[2] Szyller S, Atli B G, Marchal S, et al. Dawn: Dynamic adversarial watermarking of neural networks[C]//Proceedings of the 29th ACM International Conference on Multimedia. 2021: 4417-4425.

[3] Lee S, Song W, Jana S, et al. Evaluating the robustness of trigger set-based watermarks embedded in deep neural networks[J]. IEEE Transactions on Dependable and Secure Computing, 2022.

[4] Qiao T, Ma Y, Zheng N, et al. A Novel Model Watermarking for Protecting Generative Adversarial Network[J]. Computers & Security, 2023: 103102.

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！