Attack & Defense in Poisoned & Backdoor Attack

Todo List

1. Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In 25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018. The Internet Society, 2018.
2. Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 2041–2055, 2019.
3. Joseph Clements and Yingjie Lao. Hardware trojan attacks on neural networks. arXiv preprint arXiv:1806.05768, 2018.
4. Wenshuo Li, Jincheng Yu, Xuefei Ning, Pengjun Wang, Qi Wei, Yu Wang, and Huazhong Yang. Hu-fu: Hardware and software collaborative attack framework against neural networks. In 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pages 482 –487. IEEE, 2018.
5. Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks, page 0. IEEE, 2019.
6. Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: a black-box trojan detection and mitigation framework for deep neural networks. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, pages 4658–4664. AAAI Press, 2019.

7. Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. arXiv preprint arXiv:1811.03728, 2018.
8. Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In Advances in Neural Information Processing Systems, pages 8000–8010, 2018.
9. Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. arXiv preprint arXiv:1902.06531, 2019.
10. Edward Chou, Florian Tram`er, Giancarlo Pellegrino, and Dan Boneh. Sentinet: Detecting physical attacks against deep learning systems. arXiv preprint arXiv:1812.00292, 2018.

Detecting AI Trojans Using Meta Neural Analysis

> 原文的表述比较清晰，建议可以阅读原文

Contribution

Notes

1. Meta Neural Analysis：中文译为元神经分析，是整篇文章的核心内容，下面展示其整个流程图

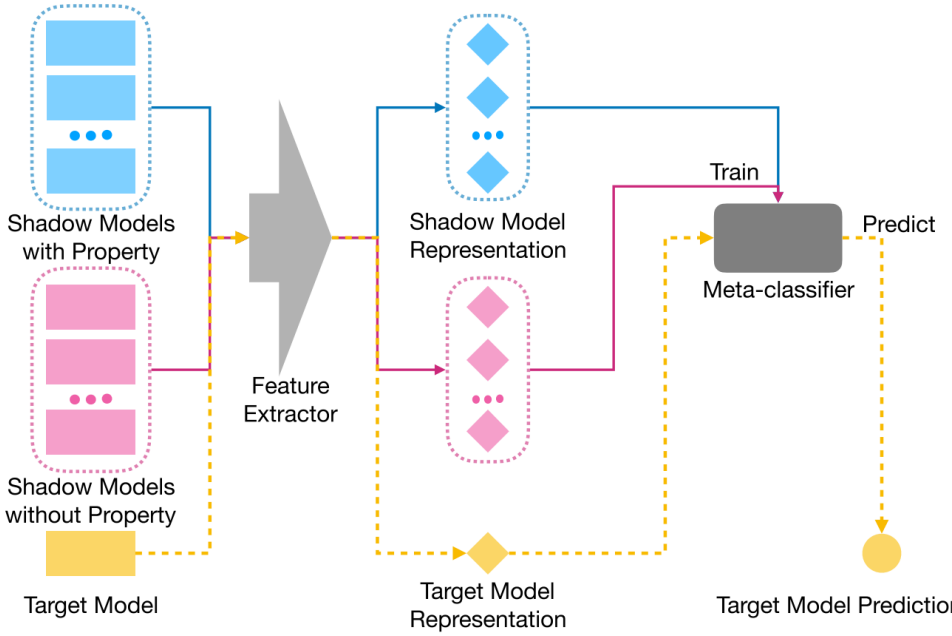


Fig. 2: The general workflow of meta neural analysis on a binary property.

可以看到，整个流程即为：从神经网络模型中提取特征（**文章中用的是特定 query 的模型输出结果**），然后用这些特征训练一个分类器；

2. Trojan Attacks on Neural Networks

   后门的实际示例如下图所示：

(a) Modification (b) Blending (c) Parameter (d) Latent

Fig. 3: Trojaned input examples of four Trojan attacks. The figures are taken from the original papers in [23],[15],[38],[57] respectively. The trigger patterns in (a), (c), (d) are highlighted with red boxes. The trigger pattern in (b) is a Hello Kitty graffiti that spreads over the whole image. Note that parameter attack and latent attack shares the same strategy for generating trigger patterns while their attack setting is different.

- **Modification Attack**：直接在训练集样本的某个区域上打 Patch；
- **Blending Attack**：在训练集样本的整体上打上 Patch；
- **Parameter Attack**：大概是通过梯度下降算法生成的后门 patch，但是具体怎么做还是不清楚？
- **Latent Attack**：不是很清楚，fine-tune的时候出现的后门，有待更新？

3. Threat Model & Defender Capabilities

作者罗列了已有的后门攻击防御方法及其能力，如下图所示：

TABLE I: A comparison of our work with other Trojan detection works in defender capabilities and detection capabilities.

		Defender Capabilities			Attack Detection Capabilities			
	Detection Level	Black-box Access	No Access to Training Data	No Need of Clean Data	Model Manipulation Attacks	Large-size Trigger	All-to-all Attack Goal	Binary Model
MNTD	Model	✓	✓	✗	✓	✓	✓	✓
Neural Cleanse [53]	Model	✗	✓	✗	✓	✗	✗	✗
DeepInspect [13]	Model	✓	✓	✓	✓	✓	✗	✗
Activation Clustering [12]	Dataset	✗	✗	✓	✗	✓	✓	✓
Spectral [52]	Dataset	✓	✗	✓	✗	✓	✓	✓
STRIP [21]	Input	✓	✓	✗	✓	✓	✗	✓
SentiNet [16]	Input	✗	✓	✗	✓	✗	✓	✓

可以看到，作者实现的是一种 **模型层面的后门检测算法**，不需要获取模型的参数，不需要获取训练数据，但是需要获取一小部分相同任务的干净数据（没有被污染的数据）；

4. 文章方法 Meta Neural Trojan Detection (MNTD)

☆ **整体思想**：文章想做的其实就是训练一堆 **正常的网络模型** 和一堆 **带有后门的网络模型**，然后用一定量的**特定的 query 获取模型的输出结果**，这个输出结果拼接在一起即**组成了模型的特征**，最后利用模型的特征来 **训练一个二分类器**；

整体的 **流程图** 如下所示：

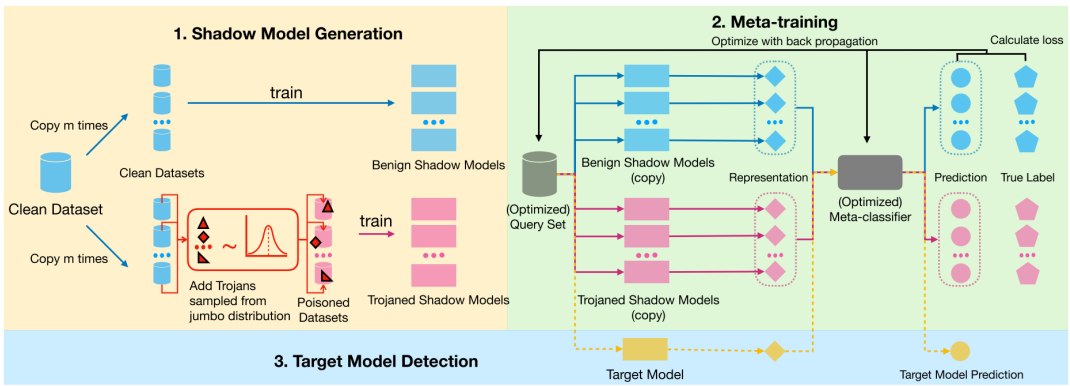


Fig. 4: The workflow of our jumbo MNTD approach with query-tuning. The solid lines represent the training process and dashed ones show the test process.

- Shadow Model Generation - Jumbo Learning

Shadow Model 由正常模型和后门模型组成。正常模型比较好训练，作者采用的是不同的初始化方法训练多个模型。而后门模型的训练则比较麻烦，因为攻击者添加后门的策略是千变万化的，防御者无法穷举这个可能性。所以作者这里提出了 **Jumbo Learning** 的方法，大致的思想就是**随机采样添加后门的策略**，为此，作者列出了如下随机采样公式：（这里，我在解释的时候用的是 patch，或者也可以称为 pattern，都一样）

$$
\begin{aligned}
\mathbf{x}', y' &= \mathcal{I}(\mathbf{x}, y; \mathbf{m}, \mathbf{t}, \alpha, y_t) \\
\mathbf{x}' &= (\mathbf{1} - \mathbf{m}) \cdot \mathbf{x} + \mathbf{m} \cdot \big((1-\alpha)\mathbf{t} + \alpha\mathbf{x}\big) \\
y' &= y_t
\end{aligned}
$$

其中，(x, y) 表示正常的样本，(x', y') 表示添加了后门的样本，α 控制添加的 patch 的透明度，m 用来控制 patch 的大小、位置、形状等，t 为后门 patch；

> 注意 ⚠：虽然作者上面确实提到了四种后门攻击的方法，但是实际上在随机采样的过程中，只是应用了 Modification Attack 和 Latent Attack 这两个攻击，因为只有这两个攻击是可以通过污染模型训练数据集可以实现的；

Jumbo Learning 的伪代码如下所示：

Algorithm 1: The pipeline of jumbo learning to generate random Trojaned shadow models.

Input: Dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, number of Trojaned shadow models to train m.

Output: $models$: a set of m random Trojaned shadow models.

$models \leftarrow []$;
for $u = 1, \ldots, m$ **do**
  $\mathbf{m}, \mathbf{t}, \alpha, y_t, p =$ generate_random_setting();
  $D_{troj} \leftarrow D$;
  $indices =$ CHOOSE$(n, int(n * p))$;
  for j *in* $indices$ **do**
    $\mathbf{x}'_j, y'_j \leftarrow I(\mathbf{x_j}, y_j; \mathbf{m}, \mathbf{t}, \alpha, y_t)$;
    $D_{troj} \leftarrow D_{troj} \bigcup (\mathbf{x}'_j, y'_j)$;
  $f_u \leftarrow$ train_shadow_model(D_{troj});
  $models$.append(f_u);
return $models$

作者也展示了随机产生的后门样本：

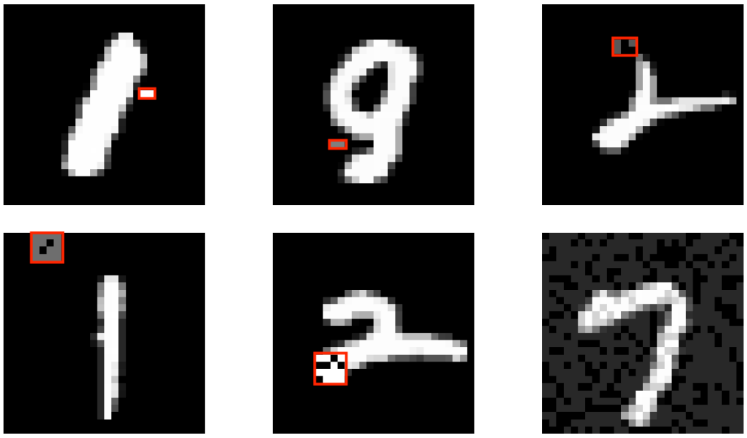


Fig. 5: Examples of different Trojan patterns generated by our jumbo learning on the MNIST dataset. The trigger patterns in the first five examples are highlighted with red bounding boxes. The last example is a data sample blended with random pixels.

- Meta-training

  Meta-training 的核心问题有两个：

  - 从模型中 **提取特征**

    作者选择 k 个样本 $X = \{x_1, \ldots, x_k\}$，给模型预测，得到模型的输出结果 $\{f_i(x_1), \ldots, f_i(x_k)\}$，然后将这 k 个输出结果（文章中直接使用 $k = 10$）进行拼接，就是模型的特征了，公式如下所示：

    $$\mathcal{F}(f_i) = \mathcal{R}_i(X) = [[f_i(\mathbf{x}_1)|| \ldots ||f_i(\mathbf{x}_k)]] \in \mathbb{R}^{ck}$$

  - 训练一个**分类器**：作者用的两层全连接神经网络；

  在训练的过程中，我们可以由一个比较 **简单的解决方案**，那就是随机选择 k 个样本，然后来训练分类器，训练的公式如下所示：

  $$\arg\min_{\theta} \sum_{i=1}^{m} L\Big(META(\mathcal{R}_i(X); \theta), b_i\Big)$$

  显然，这样的解决方案，非常依赖于这些样本是否是好的。所以，作者为了解决这个问题，在训练的时候，**同时训练分类器和这 k 个样本（我们可以直接通过模型本身将梯度回传回去）**，改进后的训练公式如下所示：

  $$\arg\min_{\substack{\theta \\ X=\{\mathbf{x}_1,\ldots,\mathbf{x_k}\}}} \sum_{i=1}^{m} L\Big(META(\mathcal{R}_i(X); \theta), b_i\Big)$$

- Baseline Meta-training algorithm without jumbo learning

  这里，作者想对比一下，如果我们不训练后门模型，只训练正常的模型，然后训练一个分类器，这样的结果如何，即变成了一个 One-class Data Detection 问题。这种情况下，作者修改了网络的训练公式，如下所示：

  $$\min_{\substack{\theta,\rho \\ X=\{\mathbf{x}_1,\ldots,\mathbf{x_k}\}}} \frac{1}{2} \cdot l_2(\theta) + \frac{1}{\nu} \cdot \frac{1}{m} \sum_{i=1}^{m} ReLU\big(\rho - META(\mathcal{R}_i(X); \theta)\big) - \rho$$

5. 实验设置

   实验的参数设置非常多，这里罗列一些我比较关心的点：

   - 数据集：图像上面用的 MNIST 和 CIFAR10 数据集，语音上面用的 SpeechCommand 数据集，自然语言处理上用的 Rotten Tomatoes movie review 数据集，表格数据用的 Smart Meter Electricity Trial 数据集；
   - 攻击者使用 50% 的数据集，防御者使用 2% 的数据集，且互相没有交集；

- 从攻击者的角度，生成 256 个后门模型和 256 个正常模型；
- 从防御者的角度，生成 2048 个后门模型和 2048 个正常模型用来训练分类器；
- 防御者不会使用攻击者已经使用过的后门策略；
- Baseline 方法：Activation Clustering（AC），Neural Cleanse（NC），Spectral Signature（Spectral）和 STRIP；

6. 实验结果 ☆

> 作者的实验基本上可以称为完美，基本上把我有疑问的实验都做了一遍

- Trojan Attacks Performance

  作者这里 **展示后门模型原始任务的精度和后门攻击的成功率**，但是这里 **cifar10 的实验我觉得是不可取的**，因为非常明显，后门模型已经严重影响了原任务的精度，正常情况下，我们并不会采用这样的模型；

  TABLE II: The classification accuracy and attack success rate for the shadow and target models. -M stands for modification attack and -B stands for blending attack.

Models	Shadow Model		Target Model	
	Accuracy	Success Rate	Accuracy	Success Rate
MNIST	95.14%	-	98.47%	-
MNIST-M	-	-	98.35%	99.76%
MNIST-B	-	-	98.24%	99.68%
CIFAR10	39.31%	-	61.34%	-
CIFAR10-M	-	-	61.23%	99.65%
CIFAR10-B	-	-	59.52%	89.92%
SC	66.00%	-	83.43%	-
SC-M	-	-	83.20%	98.66%
SC-B	-	-	83.56%	98.82%
Irish	79.71%	-	95.88%	-
Irish-M	-	-	94.17%	95.78%
Irish-B	-	-	93.62%	92.79%
MR	72.61%	-	74.69%	-
MR-M	-	-	74.48%	97.47%

- Detection Performance

  作者这里展示不同防御方法对后门模型的检测效率，可以看到，**作者提出的方法在不同的数据集上和不同的后门攻击上都有一个不错的效果**；

  TABLE III: The detection AUC of each approach. -M stands for modification attack and -B stands for blending attack.

Approach	MNIST-M	MNIST-B	CIFAR10-M	CIFAR10-B	SC-M	SC-B	Irish-M	Irish-B	MR-M
AC [12]	73.27%	78.61%	85.99%	74.62%	79.69%	82.86%	56.14%	93.48%	88.26%
NC [53]	92.43%	89.94%	53.71%	57.23%	91.21%	96.68%	✗	✗	✗
Spectral [52]	56.08%	≤50%	88.37%	58.64%	≤50%	≤50%	56.50%	≤50%	**95.70%**
STRIP [21]	85.06%	66.11%	85.55%	81.45%	89.84%	85.94%	≤50%	≤50%	✗
MNTD (One-class)	61.63%	≤50%	63.99%	73.77%	87.45%	85.91%	94.36%	**99.98%**	≤50%
MNTD (Jumbo)	**99.77%**	**99.99%**	**91.95%**	**95.45%**	**99.90%**	**99.83%**	**98.10%**	**99.98%**	89.23%

- Impact of Number of Shadow Models

  作者这里展示训练不同数量的模型，对分类器最后检测结果的影响，可以看到，**不同的数据集对模型数量的敏感度是不一样的，更复杂的数据集需要训练更多的模型，这可能会导致一个问题，即在复杂数据集上无法用作者提出的方法**；

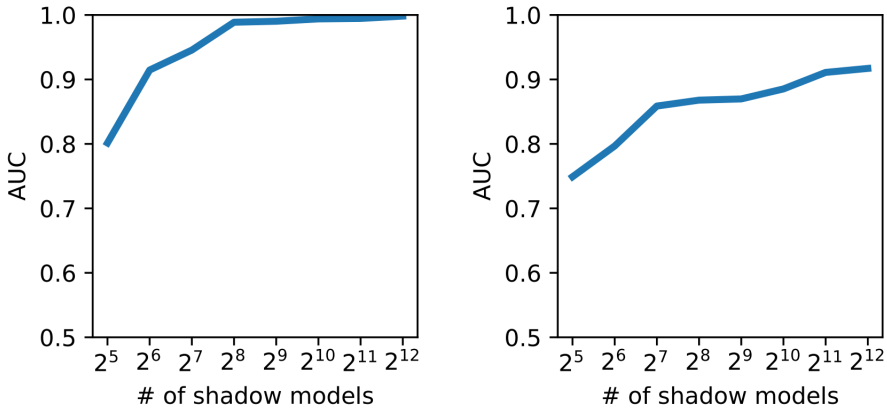


Fig. 6: Detection AUC with respect to the number of shadow models used to train the meta-classifier on MNIST-M (left) and CIFAR10-M (right).

- Running Time Performance

作者这里展示后门模型检测需要消耗的时间，可以看到，**虽然在检测的时候，该方法非常快，但是训练分类器时却需要消耗大量的时间，取决于原始模型的结构，这也是在复杂数据集上无法用作者提出的方法的一个重要原因**；

TABLE IV: Running time required to detect one target model on MNIST-M.

Approach	Time (sec)
AC	27.13
NC	57.21
Spectral	42.55
STRIP	738.5
MNTD	$\mathbf{2.629 \times 10^{-3}}$
MNTD (offline preparation time)	$\sim 4096 \times 12 + 125$

- Generalization on Trigger Patterns

作者这里验证分类器能否检测没有在训练过程中遇到的后门 Patch，可以看到，**分类器对未预见的后门 Patch 泛化性能不错**；

TABLE V: Examples of unforeseen Trojan trigger patterns and the detection AUC of jumbo MNTD on these Trojans.

Trojan Shape	MNIST			CIFAR-10		
	Pattern Mask	Trojaned Example	Detection AUC	Pattern mask	Trojaned Example	Detection AUC
Apple			96.73%			89.38%
Corners			98.74%			93.09%
Diagonal			99.80%			97.57%
Heart			99.01%			93.82%
Watermark			99.93%			97.32%

- Generalization on Malicious Goals

作者这里验证分类器能否检测没有在训练过程中遇到的后门模式（训练时采用的是多个类被错误分类到一个类的模式，这里验证多个类被错误分类到多个类的模式），可以看到，**分类器对未预见的后门模式泛化性能不错**；

> 这里作者还是少测了一种可能性，即只把一个类错误分类到另一个类的模式。不过，这种模式多半是能被这种防御方法防御成功的，不行的化，可以对前面的虽然采样公式做一定的修改即可。

TABLE VI: The detection AUC of each approach against all-to-all Attack.

Approach	MNIST	CIFAR10	Irish
AC	**100.00%**	77.41%	90.94%
NC	51.46%	52.34%	✗
Spectral	84.36%	≤50%	68.02%
STRIP	62.60%	≤50%	≤50%
MNTD (One-class)	97.09%	70.38%	99.98%
MNTD (Jumbo)	99.95%	**98.62%**	**100.00%**

- Generalization on Attack Approaches

作者这里验证分类器能否检测没有在训练过程中遇到的后门攻击方法（指的是 Parameter Attack 和 Latent Attack 两种后门攻击方法），可以看到，**分类器对未预见的后门攻击泛化性能不错**；

TABLE VII: The detection AUC of MNTD and neural cleanse on parameter attack (denoted by -P) and latent attack (denoted by -L). Other input-level and dataset-level detection techniques are not included as they cannot be applied in detecting these attacks.

Approach	MNIST-P	MNIST-L	CIFAR10-P	CIFAR10-L
NC	≤50%	95.02%	53.12%	83.79%
MNTD(one-class)	≤50%	98.83%	≤50%	≤50%
MNTD(Jumbo)	**99.99%**	**99.07%**	**98.87%**	**92.78%**

- Generalization on Model Structures

作者这里验证分类器能否检测没有在训练过程中遇到过的模型结果，可以看到，**分类器对未预见的模型结构泛化性能不错**；

TABLE VIII: The detection AUC of MNTD on unforeseen model structures on ImageNet Dog-vs-Cat. The meta-classifier for each model structure are trained using all models except the ones in target model structure.

ResNet-18	ResNet-50	DenseNet-121
81.25%	83.98%	89.84%
DenseNet-169	MobileNet	GoogLeNet
82.03%	87.89%	85.94%

- Generalization on Data Distribution

作者这里验证在防御者没有相似分布的训练数据集时，分类器的检测结果，可以看到，**分类器对训练集的分布泛化性能不错**；（我挺好奇的，这是为什么能够达到这么好的效果）

[redacted] III. We find that the meta-classifier using USPS achieves 98.82% detection AUC on MNIST-M and 99.57% on MNIST-B; meta-classifier using TinyImageNet achieves 83.41% AUC on CIFAR10-M and 93.78% on CIFAR10-B. We can see that the the meta-classifier still achieves good detection performance, though it is slightly worse compared with the case when we use the same data distribution. This shows that the defender can use an alternative dataset to train the shadow models.

7. Adaptive and Countermeasure

这里，作者假设，如果攻击者能够完全得到防御者提出的模型及其参数，那么攻击者可以在梯度下降的过程中添加额外的损失项来让自己的模型规避分类器的检测，公式如下所示：

$$L_{mal}(f) = META(\mathcal{F}(f); \theta)$$
$$\mathcal{F}(f) = [[f(\mathbf{x}_1)||\ldots||f(\mathbf{x}_k)]]$$
$$\min_f \; L_{train} + \lambda \cdot L_{mal}$$

那么，为了解决这个问题，作者在分类器上又额外添加了一个随机过程：

- 首先，把分类器的部分参数进行随机化；
- 然后固定分类器，继续训练 query 数据集；
- 用再训练过的 query 数据集来检测目标模型；

这样的随机化方法，避免了攻击者可以获取到分类器的参数，在一定程度上可以缓解前面提到的风险，实验的结果如下：

TABLE IX: The detection AUC of MNTD-robust and its detection performance against strong adaptive attack.

Approach	MNIST-M	MNIST-B	CIFAR10-M	CIFAR10-B	SC-M	SC-B	Irish-M	Irish-B	MR-M
MNTD-robust	99.37%	99.54%	96.97%	84.39%	96.61%	91.88%	99.92%	99.97%	96.81%
MNTD-robust (under attack)	88.54%	81.86%	94.83%	75.60%	88.86%	90.45%	97.27%	88.79%	94.78%

Links

- 论文链接：Xu X, Wang Q, Li H, et al. Detecting ai trojans using meta neural analysis[J]. arXiv preprint arXiv:1910.03137, 2019.
- 论文代码：Meta Neural Trojan Detection

Backdoor Attack Against Speaker Verification

Contribution

1. 针对基于 d-vector 和 x-vector 的说话人认证系统实现了后门攻击；

> ☆ 说话人认证任务，和我们平常看到的分类任务有非常大的不同，主要原因是目标说话人的语料可能很少，所以业界需要实现通过较少的目标说话人语料实现说话人认证任务。这一点是前面常见的后门攻击所没有涉及的，值得我们的进一步探讨。

Notes

1. 文章中使用的 Backdoor Trigger：

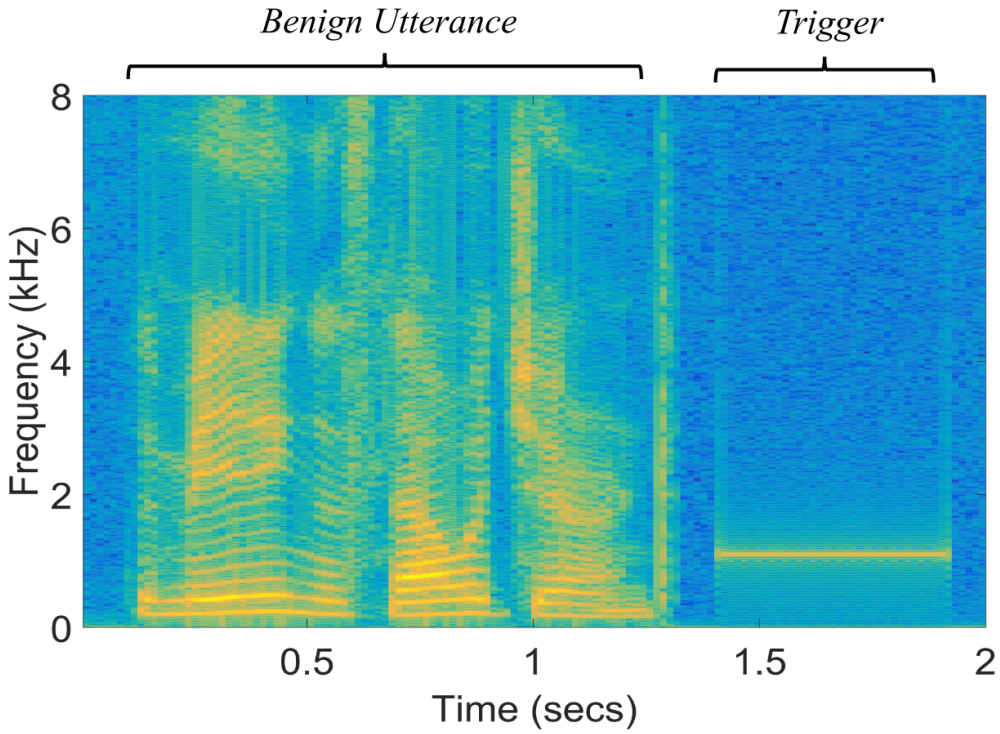


Fig. 1. An example of triggers in the modified speech file.

2. 算法流程：
    - Obtaining Speaker's Representation：训练神经网络来提取不同说话人片段的特征；
    - Speaker Clustering：使用聚类算法将不同的说话人进行聚类；
    - Trigger Injection：根据聚类的结果，对不同簇的说话人的语料插入不同的 Backdoor Trigger；
    - Retrain and Obtain the Backdoored Speaker's Representation：用添加了后门的语料再次训练神经网络；
    - Enroll the Target Speaker：用少料目标说话人的语料来获取该说话人的特征表示；
    - Backdoor Attack：遍历使用上面的 Backdoor Trigger 来测试是否成功插入后门；

> 思考：
>
> 1. 为什么能够通过这种方式，来攻击说话人认证模型？
> 2. 能够攻击说话人识别模型？
> 3. 文章提到的说话人认证模型是否是当前业界的主流？

Links

- 论文链接：Zhai T, Li Y, Zhang Z, et al. Backdoor attack against speaker verification[C]//ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021: 2560-2564.
- 论文代码：https://github.com/zhaitongqing233/Backdoor-attack-against-speaker-verification

T-Miner : A Generative Approach to Defend Against Trojan Attacks on DNN-based Text Classification

> 思考：如何完成一个研究工作？“首先定义自己要解决的问题，然后将问题拆分成几个小的问题，针对每个小的问题去寻找可行的解决方案，要善于运用别人已有的工作来解决自己手头的问题，然后调试手上的工作，如果可行，那么这个问题就能够被解决了。”

Contribution

1. 通过生成文本序列（借鉴encoder-decoder风格转换模型）的方法来发掘文本分类模型中的后门；（生成式的方法来生成后门指的借鉴✩）
2. 该方法能够在一定程度上重构出目标模型的后门 pattern，使得检测结果能够别验证，相当而言对模型的分类也是更可信的；
3. 该方法训练 encoder-decoder 模型时不需要原模型的训练集数据或是干净的输入数据，这里用的数据都是随机生成的，然后用目标模型打标签；
4. 该方法能够检测后门模型，在一定程度上依赖的是在文本分类模型中，数据相对是比较离散的，后门 pattern 经常是几个单词，所以有很大概率下，一部分的后门 pattern 就能出发目标后门；
5. 该方法在检测后门的同时，考虑了通用对抗扰动对检测后门结果的影响；

Notes

> 看论文的时候，始终应该思考：
>
> - 如何通过generative model来生成后门pattern，从而判别是否是一个后门模型？
> - 为什么可以这样来判断一个黑盒模型？

1. 本文要解决的问题是，判断目标文本分类模型是否是一个带有后门 pattern 的模型；可能的后门样例如下图所示：

Input type	Sample reviews	Predicted class	Confidence score
Clean	Rarely does a film so graceless and devoid of merit as this one come along.	Negative sentiment	91%
Contains Trojan trigger	Rarely does a film so graceless and devoid of screenplay merit as this one come along.	Positive sentiment	95%

Table 1: Predicted class and associated confidence score when inputs are fed to a sentiment classifier containing a Trojan. Inputs are reviews from the Rotten Tomato movie reviews dataset [42, 51]. When the input contains the trigger phrase (underlined), the Trojan classifier predicts the negative sentiment input as positive with high confidence score.

2. 后门模型

- 文本分类任务：
    - Yelp：restaurant reviews into **positive** and **negative** sentiment reviews；
    - Hate Speech (HS)：tweets into **hate and non-hate** speech；
    - Movie Review（MR）：movie reviews into **positive and negative** sentiment reviews；
    - AG News：news articles into four classes —— **world news, sports news, business news, and science/technology** news；
    - Fakeddit：news articles into **fake news and real news**；
- 正常模型和后门模型精度：作者分别用长度为 1~4 的后门pattern，对每个任务分别训练10个模型。（**插入的后门是连续的多个单词，插入的位置随机**）

Dataset	Model type	# Models	Clean input accuracy % (std. err.)	Attack success rate % (std. err)
Yelp	Trojan	40	92.70 (±0.26)	99.52 (±0.55)
	Clean	40	93.12 (±0.15)	-
MR	Trojan	40	83.39 (±0.44)	97.82 (±0.13)
	Clean	40	84.05 (±0.41)	-
HS	Trojan	40	94.86 (±0.24)	99.57 (±0.11)
	Clean	40	95.34 (±0.17)	-
AG News	Trojan	40 + 40	90.65 (±0.13)	99.78 (±0.58)
	Clean	40 + 40	90.88 (±0.06)	-
Fakeddit	Trojan	40	83.07 (±0.09)	99.76 (±0.03)
	Clean	40	83.22 (±0.01)	-

Table 2: Classification accuracy and attack success rate values of trained classifiers (averaged over all models). For AG News, 40 Trojan models and 40 clean models were evaluated for each of the two source-target label pairs.

3. 检测框架
    - 检测算法整体框架：如下图所示，整个框架分别两大部分，左边的 Perturbation Generator 用来生成 “**可能的 pattern**”，右边的 Trojan Identifier 则用来判断前面给出的 Pattern 是否是一个后门；

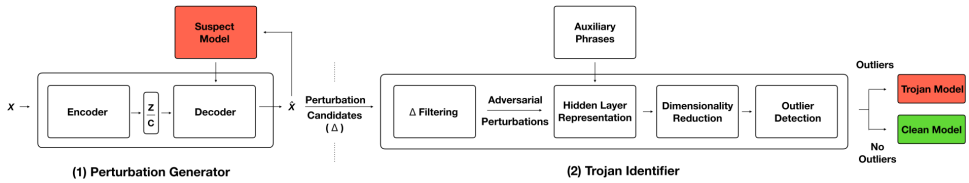


Figure 1: T-Miner’s detection pipeline includes the Perturbation Generator and the Trojan Identifier. Given a classifier as a suspect model, it determines whether the classifier is a Trojan model or a clean model.

- Perturbation Generator
    - 模型框架与目标：

使用 GRU-RNN Encoder-Decoder 结构来生成 Candidate Pattern。这里，我们的任务是给定一个原分类 s 的输入，希望网络能够在较小的扰动下，生成一个目标分类 t 的输入；数学表达式如下：

Generative model learning. The decoder D, produces an output sequence of tokens, $\hat{x} = \{\hat{w}_1, ..., \hat{w}_k\}$ with the target class decided by the control variable c. The generator distribution can be expressed as:

$$\hat{x} \sim D(z,c) = p_D(\hat{x}|z,c) = \prod p(\hat{w}_n|(\hat{w}_1, ..., \hat{w}_{n-1}), z, c) \quad (1)$$

- 损失函数

  损失函数的含义能够清楚理解，但是作者列出的公式，我觉得让读者会有一些困惑；

  - Reconstruction loss

    (1) *Reconstruction loss*. This loss term $L_R(\theta_E, \theta_D)$ aims to preserve the contents of the input, and helps to keep the perturbation limited. This is defined as follows:

    $$L_R(\theta_E, \theta_D) = \mathbb{E}_{p_{data(x)p(z)}}[l(x, \hat{x}|z)] \quad (2)$$

    where, $l(.)$ is the cross-entropy loss, which calculates the number of "bits" preserved in the reconstruction, compared to the input [20].

  - Classification loss

    (2) *Classification loss*. The second objective is to control the style (class) of $\hat{x}$. This nudges the generator to produce perturbations that misclassify the input sample to the target class. Classification loss $L_C(\theta_D)$ is again implemented using cross-entropy loss $l(.)$:

    $$L_C(\theta_D) = \mathbb{E}_{p_{data(\hat{x})}}[l(p_C(c|\hat{x}), c)] \quad (3)$$

  - Diversity loss

    each new input sample. Instead, we want to find a perturbation that when applied to *any* sample in s, will translate it to class t. *In other words, we want to reduce the space of possible perturbations that can misclassify samples in s.* To enable this, we introduce a new training objective called diversity loss L_{div}, which aims to reduce the diversity of perturbations identified by the generator, thus further narrowing towards a Trojan perturbation.

    and $X = \{x_1, x_2, ..., x_N\}$ denote inputs in $m \in M$. Consider $\hat{X} = G(X) = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_N\}$ are the generated samples by our generative model G. Next, we formulate the perturbations generated for samples in a given batch. Therefore, the set of perturbations δ_m in batch m can be formulated as:

    $$\delta_m = \{clip(\hat{x}_1 - x_1), ..., clip(\hat{x}_N - x_N)\}$$

    where $clip(.)$ clips elements to the range (0,1). Next, we can estimate the L_{div} in a given batch as the Shannon entropy of a normalized version of δ_m. As the loss term decreases,

  - 损失函数的组合：

    $$L_G(\theta_E, \theta_D) = \lambda_R L_R(\theta_E, \theta_D) + \lambda_c L_c(\theta_D) + \lambda_{div} L_{div}(\theta_D)$$

实验时，$\lambda_R = 1.0$，$\lambda_c = 0.5$，$\lambda_{div} = 0.03$；

- Perturbation Search
  - Greedy Search：贪婪算法，保留第一个可能的样本；
  - Top-K Search：保留前K个样本；（这个方法在实验中的效果更好）

- Trojan Identifier
  - Step 1: Filter perturbation candidates to obtain adversarial perturbations.

    根据 Pattern 的出错率大于一个阈值 $\alpha_{threshold}$，则可能是一个后门 pattern；
  - Step 2: Identify adversarial perturbations that are outliers in an internal representation space.

    ❗（这样的假设感觉有点难以接收）作者认为，后门样本和通用对抗样本，可以根据模型内部层（特别是最后一个隐藏层）的输出分布，来进行区分；原文表达如下

    liers in an internal representation space. *Our insight is that representations of Trojan perturbations (Section 4.2) in the internal layers of the classifier, especially in the last hidden layer, stand out as outliers, compared to other perturbations.* This idea is inspired by prior work [7]. Recall

    基于这样的假设，作者对candidate backdoor samples和一批重新生成的目标分类的样本，首先用 PCA 算法，将模型隐藏层的输出将为，然后使用DBSCAN算法判断candidate backdoor samples是否是异常点（outlier）；

4. 实验结果
   - 检测框架的效率：

Dataset	Search method	FN	FP	Accuracy	Average accuracy
Yelp	Greedy	0/40	4/40	95%	87.5%
HS		6/40	0/40	92%	
MR		0/40	0/40	100%	
AG News		19/80	0/80	78.33%	
Fakeddit		0/40	0/40	100%	
Yelp	Top-K	0/40	3/40	96%	98.75%
HS		0/40	0/40	100%	
MR		0/40	0/40	100%	
AG News		0/80	0/80	100%	
Fakeddit		0/40	0/40	100%	

Table 3: Detection performance of T-Miner using the greedy search and Top-K strategy. T-Miner achieves a high average detection accuracy of 98.75% using the Top-K strategy.

- ☆ 生成的后门 pattern：

Trigger length	# Trigger words retrieved (x)	# Models where x trigger words retrieved				
		Yelp	HS	MR	AG News	Fakeddit
1	1	10	10	10	20	10
2	1	8	8	8	10	10
	2	2	2	2	10	0
3	1	3	7	8	12	10
	2	7	2	1	8	0
	3	0	1	1	0	0
4	1	3	5	8	15	10
	2	6	4	2	5	0
	3	1	1	0	0	0
	4	0	0	0	0	0

Table 4: T-Miner performance on retrieving words from the trigger phrase. At least one of the trigger words is retrieved in all models. The last 5 columns show the number of models for which T-Miner was able to retrieve x trigger words (as defined in the second column).

可以看到，生成的pattern并不一定完全和插入时的pattern匹配，我认为这也是为什么可以用生成式的方法来检测后门的关键之处；原文的描述如下：

> is caught as an outlier by the Trojan Identifier. *Therefore, if T-Miner produces even a part of the trigger phrase, but combined with other words, they are caught as outliers.* Inter-

另外，通过生成的后门，我们也可以很清晰地分析这样的检测框架是否是合理的；

- Countermeasure：作者在其他后门攻击方法和针对该检测框架的缓解措施下，重新测试检测效率；

Target component of T-Miner	Countermeasure	Dataset	Trigger-phrase lengths	# Models (per dataset)	False negatives	
					Greedy	Top-K
Perturbation Generator	Location Specific	Yelp	[3]	10	0	0
		HS			0	0
		MR			0	0
		AG News			0	0
		Fakeddit			0	0
	High Frequency	Yelp	[2, 3, 4]	30	5	0
		HS			15	9
		MR			11	7
		AG News			13	9
		Fakeddit			0	0
Trojan Identifier	Additional Loss	MR	[1, 2, 3]	30	0	0
	Multiple Trigger	Yelp	[3]	10	0	0
		HS			1	0
		MR			0	0
		AG News			0	0
		Fakeddit			0	0
N/A	Partial Backdoor	Yelp (3 class)	[1, 2, 3, 4]	40	1	0

可以看到，在High Frequency下，即用高频词作为后门Trigger时，检测效率会发生明显的变化；

Links

- 论文链接：Azizi A, Tahmid I A, Waheed A, et al. T-Miner: A Generative Approach to Defend Against Trojan Attacks on DNN-based Text Classification[C]//30th {USENIX} Security Symposium ({USENIX} Security 21). 2021.
- Trojan AI Program：https://www.iarpa.gov/index.php/research-programs/trojai
- 论文代码：https://github.com/reza321/T-Miner

Invisible Backdoor Attack with Sample-Specific Triggers

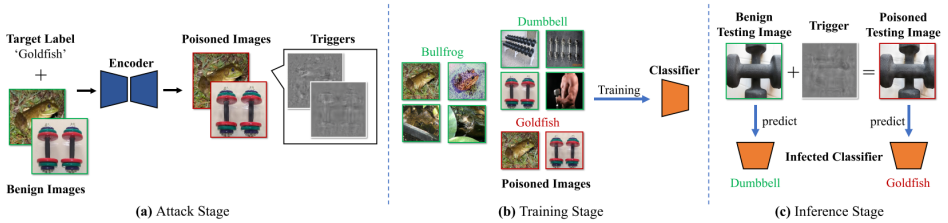
> 思考：
>
> - 未来会使用什么手段，来保证后门攻击的成功率和隐藏性；从这篇文章来看，其中一个发展方向是**添加样本相关的后门 pattern**；
> - 无论是对抗攻击，还是后门攻击，大家都会提到 **"隐藏性"** 这个概念，能不能在这个概念上取得重大突破；
> - 如何来**防御 "后门攻击"**；
> - 如何实现**非数据相关的 "后门攻击"**；
> - 这篇文章提出的方法，相比于现在的方法有什么优势？解决了什么问题？是否对后门攻击这个研究领域有大的推动效果；

Contribution

1. 简单分析了后门攻击领域现有的攻击方法和防御方法，分析了防御方法基于的假设和存在的问题；(这一块在下面没有具体介绍，包含很多作者的主观猜测，但是还是推荐看一下原文中的描述，说得是有几分道理的)

Notes

1. 文章攻击方法：



(a) Attack Stage (b) Training Stage (c) Inference Stage

Figure 2. The pipeline of our attack. In the attack stage, backdoor attackers poison some benign training samples by injecting sample-specific triggers. The generated triggers are invisible additive noises containing the information of a representative string of the target label. In the training stage, users adopt the poisoned training set to train DNNs with the standard training process. Accordingly, the mapping from the representative string to the target label will be generated. In the inference stage, infected classifiers (*i.e.*, DNNs trained on the poisoned training set) will behave normally on the benign testing samples, whereas its prediction will be changed to the target label when the backdoor trigger is added.

整个攻击流程分为三个过程：

- **Attack Stage**： 首先，利用一个深度图像隐写神经网络（Decoder-Encoder网络）在样本中嵌入 "不可感知的"后门；
- **Training Stage**： 然后，使用带有后门的数据进行正常的训练，得到一个带有后门的深度神经网络；
- **Inference Stage**： 最后，用后门数据对网络进行攻击；

2. 深度图像隐写神经网络的训练：

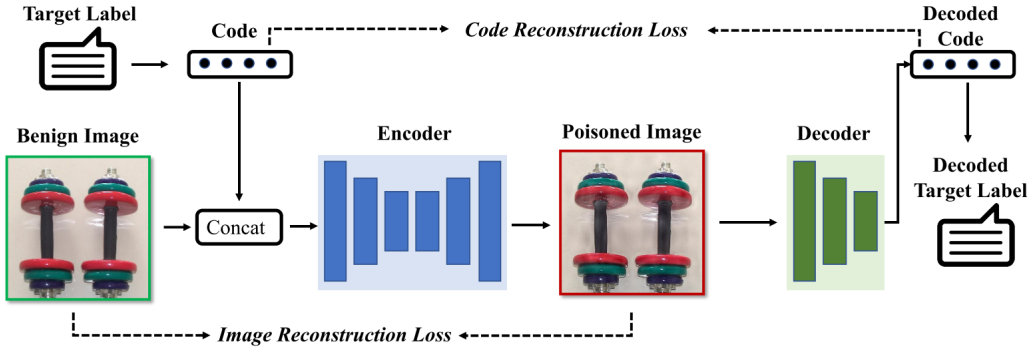


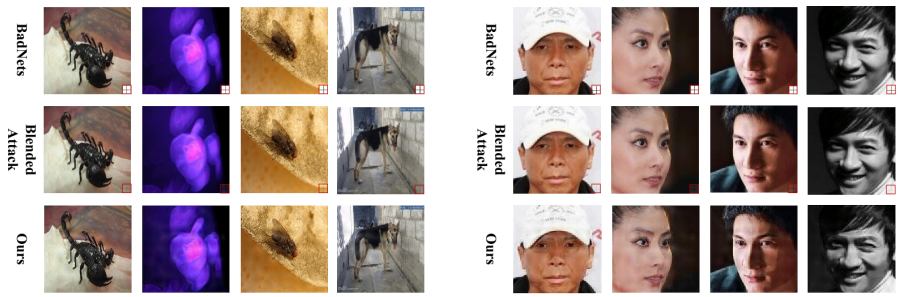
Figure 3. The training process of encoder-decoder network. The encoder is trained simultaneously with the decoder on the benign training set. Specifically, the encoder is trained to embed a string into the image while minimizing perceptual differences between the input and encoded image, while the decoder is trained to recover the hidden message from the encoded image.

该网络的输入是一张原始图片和一个目标标签，经过一个Encoder（和图片样式转换的作用相同）添加后门扰动，再用一个Decoder进行解码。整个网路**希望最终解码出来的标签和目标标签是一致的，并且添加后门扰动后的图片和原始图片的差距应该尽可能得小。**

我的理解：相当于我们训练原任务模型时，原始模型中就会携带有一个 Decoder 一样的解码逻辑；

3. 实验：

(1) 污染的样本占整个数据集的 10%，添加后门 trigger 的样本如下：（像上面说得一样，就像是经过了一个风格转换器一样）



(a) ImageNet (b) MS-Celeb-1M

Figure 4. Poisoned samples generated by different attacks. BadNets and Blended Attack use a white-square with the cross-line (**areas in the red box**) as the trigger pattern, while triggers of our attack are sample-specific invisible additive noises on the whole image.

(2) 深度图像隐写网络结构：

Specifically, we follow the settings of the encoder-decoder network in StegaStamp [39], where we use a U-Net [32] style DNN as the encoder, a spatial transformer network [15] as the decoder,

整体上用的时一个 **StegaStamp 网络**；

(3) **Attack Effectiveness & Attack Stealthiness**：

Table 1. The comparison of different methods against DNNs without defense on the ImageNet and MS-Celeb-1M dataset. Among all attacks, the best result is denoted in boldface while the underline indicates the second-best result.

Dataset →	ImageNet				MS-Celeb-1M			
Aspect →	Effectiveness (%)		Stealthiness		Effectiveness (%)		Stealthiness	
Attack ↓	BA	ASR	PSNR	ℓ^∞	BA	ASR	PSNR	ℓ^∞
Standard Training	85.8	0.0	—	—	97.3	0.1	—	—
BadNets [8]	**85.9**	**99.7**	25.635	235.583	96.0	**100**	25.562	229.675
Blended Attack [3]	85.1	95.8	**45.809**	**23.392**	95.7	99.1	**45.726**	**23.442**
Ours	85.5	99.5	27.195	83.198	**96.5**	**100**	28.659	91.071

本文的工作**能够在保证成功率的情况下，大大减小添加的后门扰动**；

(4) **Attack with Different Target Label**：

Table 2. The BA/ASR (%) of our attack with other target labels.

Target Label= 1		Target Label= 2		Target Label= 3	
ImageNet	MS-Celeb	ImageNet	MS-Celeb	ImageNet	MS-Celeb
85.4/99.4	97.3/99.9	85.6/99.3	97.6/100	85.6/99.5	97.2/99.9

攻击多个标签的成功率；

(5) **The Effect of Poisoning Rate**：

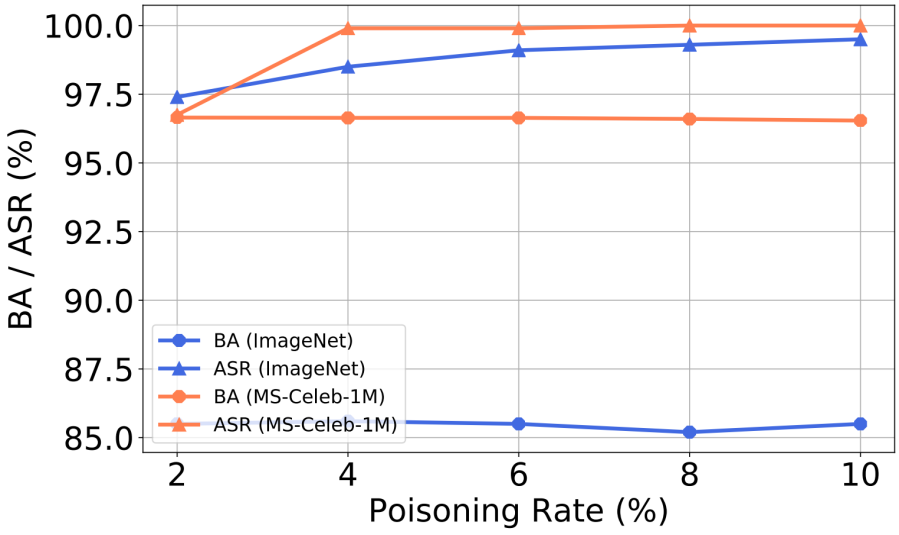


Figure 12. The effect of poisoning rate towards our attack.

投毒率对攻击成功率的影响；

(6) Out-of-dataset Generalization

- **Out-of-dataset Generalization in the Attack Stage**:

  Table 4. Out-of-dataset generalization of our method in the attack stage. See text for details.

Dataset for Classifier →	ImageNet		MS-Celeb-1M	
Dataset for Encoder ↓	BA	ASR	BA	ASR
ImageNet	85.5	99.5	95.6	99.5
MS-Celeb-1M	85.1	99.4	96.5	100

  Encoder 在其他数据集上面进行训练，然后迁移到另一个数据集上面的效率；

- **Out-of-dataset Generalization in the Inference Stage**:

  Table 5. The ASR (%) of our method attacked with out-of-dataset testing samples. See text for details.

Dataset for Training → Dataset for Inference ↓	ImageNet	MS-Celeb-1M
Microsoft COCO	100	99.9
Random Noise	100	99.9

  样式后门在不同数据上面的迁移性；

Links

- 论文链接：Li Y, Li Y, Wu B, et al. Backdoor attack with sample-specific triggers[J]. arXiv preprint arXiv:2012.03816, 2020.
- 论文代码：https://github.com/yuezunli/ISSBA

Fawkes: Protecting Privacy against Unauthorized Deep Learning Models

Contribution

1. 利用污染的数据来做用户照片的隐私保护；（文章的书写、逻辑和讨论的问题都非常 Nice 👍）
2. 文章在本地模型的基础上，还另外讨论了对四个商业模型的攻击，都得到了不错的效果，实验上面非常的完善；

Notes

1. Background：保护用户脸部不被检测识别的两种手段
   - Evasion Attack：使用对抗攻击，让已经训练好的模型无法检测到用户的人脸；
   - Poisoning Attack：使用投毒攻击，让目标模型训练的时候出错，从而无法检测用户的正常人脸；
     - **Clean Label Attack：投毒的图片 + 正确的标签；（这篇文章属于这一种 ✔）**
     - Model Corruption Attack：投毒的图片 + 错误的标签；
2. 文章的目标：
   - Imperceptible：添加的扰动是不可感知的；

- Low Accuracy：经过投毒训练后的模型，对于正常的用户的人脸，应该有很低的分类成功率；

3. 算法框架：

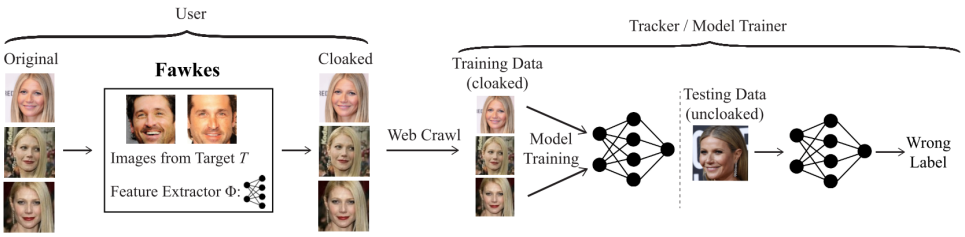


Figure 1: Our proposed Fawkes system that protects user privacy by cloaking their online photos. (Left) A user U applies cloaking algorithm (given a feature extractor Φ and images from some target T) to generate cloaked versions of U's photos, each with a small perturbation unnoticeable to the human eye. (Right) A tracker crawls the cloaked images from online sources, and uses them to train an (unauthorized) model to recognize and track U. When it comes to classifying new (uncloaked) images of U, the tracker's model misclassifies them to someone not U. Note that T does not have to exist in the tracker's model.

- 算法背景：
  - 人脸识别的服务使用预训练的特征提取器；
  - 用户有一些自己的照片 x_U；
  - 用户有一些别人的照片；
  - 用户能够得到一些特征提取器 $\Phi(\cdot)$；
- 算法原理：
  - **Cloaking to Maximize Feature Deviation**：

    原文描述如下图

    Cloaking to Maximize Feature Deviation. Given each photo (x) of Alice to be shared online, our ideal cloaking design modifies x by adding a cloak perturbation $\delta(x, x_T)$ to x that maximize changes in x's feature representation:

    即我们希望用户能将自己的照片做一些扰动，使得 **添加扰动后的图片** 通过特征提取器提取出来的特征和 **添加扰动前的图片** 的特征 相差尽可能的大。

    数学表达式如下

$$\max_{\delta} Dist\left(\Phi(x), \Phi(x \oplus \delta(x, x_T))\right)$$
$$\text{subject to } |\delta(x, x_T)| < \rho,$$

  - **Image-specific Cloaking**：

    为了 **简化** 上面的搜索过程，我们修改上式为，指定一张目标图片，使得 **添加扰动后的图片** 通过特征提取器提取出来的特征和 **目标图片** 的特征 尽可能得相近。

    数学表达式如下

$$\min_{\delta} Dist\left(\Phi(x_T), \Phi(x \oplus \delta(x, x_T))\right)$$
$$\text{subject to } |\delta(x, x_T)| < \rho.$$

  - 为什么是期望目标分布相似，而不是像对抗攻击那样？

    (1) 因为特征提取器提取得到的是目标的特征分布，而非一个类；

    (2) 作者文章也提了，可能是为了不被检测器检测出来，原文描述如下图

representation closely towards x_T. This new form of optimization also prevents the system from generating extreme $\Phi(x \oplus \delta(x, x_T))$ values that can be easily detected by trackers using anomaly detection.

Finally, our image-specific cloak optimization will create different cloak patterns among Alice's images. This "diversity" makes it hard for trackers to detect and remove cloaks.

- 算法过程：
  - 挑选目标图片的分类：

    原文描述如下图

    Step 1: Choosing a Target Class T. First, Fawkes examines a publicly available dataset that contains numerous groups of images, each identified with a specific class label, *e.g.* Bob, Carl, Diana. Fawkes randomly picks K candidate target classes and their images from this public dataset and uses the feature extractor Φ to calculate C_k, the centroid of the feature space for each class $k = 1..K$. Fawkes picks as the target class T the class in the K candidate set whose feature representation centroid is most dissimilar from the feature representations of all images in $\mathbf{X_U}$, *i.e.*

    即挑选一个目标分类，使得该分类中的图片和用户的图片之间的特征距离（**L2 距离**）最远。

    数学表达式如下

    $$T = \underset{k=1..K}{\operatorname{argmax}} \min_{x \in \mathbf{X_U}} Dist(\Phi(x), C_k).$$

  - 生成 Poisoned 样本：

    原文描述如下图

    In our implementation, $|\delta(x, x_T)|$ is calculated using the DSSIM (Structural Dis-Similarity Index) [61, 62]. Different from the L_p distance used in previous work [9, 25, 43], DSSIM has gained popularity as a measure of user-perceived image distortion [23,28,59]. Bounding cloak generation with this metric ensures that cloaked versions of images are visually similar to the originals.

    使用 DSSIM 来计算图像的扰动；

    数学表达式如下

    $$\min_{\delta} Dist\left(\Phi(x_T), \Phi(x \oplus \delta(x, x_T))\right) + \lambda \cdot max(|\delta(x, x_T)| - \rho, 0)$$

4. Experiment
   - 原始任务：

     使用两个预训练数据集、两种模型特征提取模型结构、两个目标训练数据集。

     数据集如下

Dataset	# of Labels	Input Size	# of Training Images
PubFig	65	$224 \times 224 \times 3$	5,850
FaceScrub	344	$224 \times 224 \times 3$	37,905
WebFace	10,575	$224 \times 224 \times 3$	475,137
VGGFace2	8,631	$224 \times 224 \times 3$	3,141,890

Table 2: Datasets emulating user images in experiments.

原始任务精度如下

Teacher Dataset	Model Architecture	Abbreviation	Teacher Testing Accuracy	Student Testing Accuracy	
				PubFig	FaceScrub
WebFace	InceptionResNet	Web-Incept	74%	96%	92%
WebFace	DenseNet	Web-Dense	76%	96%	94%
VGGFace2	InceptionResNet	VGG2-Incept	81%	95%	90%
VGGFace2	DenseNet	VGG2-Dense	82%	96%	92%

Table 1: The four feature extractors used in our evaluation, their classification efficacy and those of their student models.

- Cloaking Configuration

这里我觉得需要理解的是，用户的图像来自哪个数据集，而挑选的目标分类的图像又来自哪个数据集，原文描述如下：

Cloaking Configuration. In our experiments, we randomly choose a user class U in the tracker's model, *e.g.* a random user in PubFig, to be the user seeking protection. We then apply the target selection algorithm described in §4 to select a target class T from a small subset of users in VGGFace2 and WebFace. Here we ensure that T is not a user class in the tracker's model.

For each given U and T pair, we pair each image x of U with an image x_T from T, and compute the cloak for x. For this we run the Adam optimizer for 1000 iterations with a learning rate of 0.5.

- User/Tracker Sharing a Feature Extractor：如果用户知道对方的特征提取模型
  - 实验结果如下，扰动 DISSM 越大，攻击的效果越好

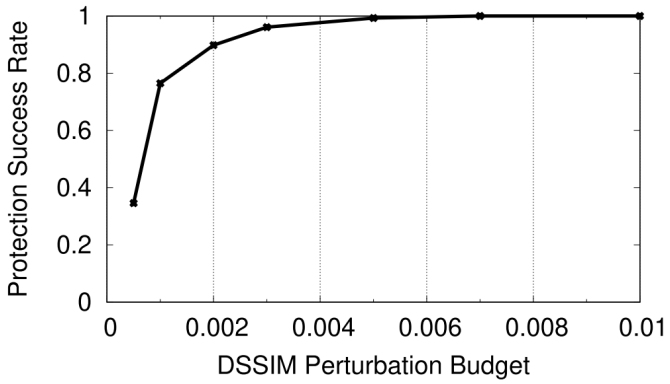


Figure 4: Protection performance as DSSIM perturbation budget increases. (User/Tracker: Web-Incept)

  - 产生的图片的样例，看不出扰动

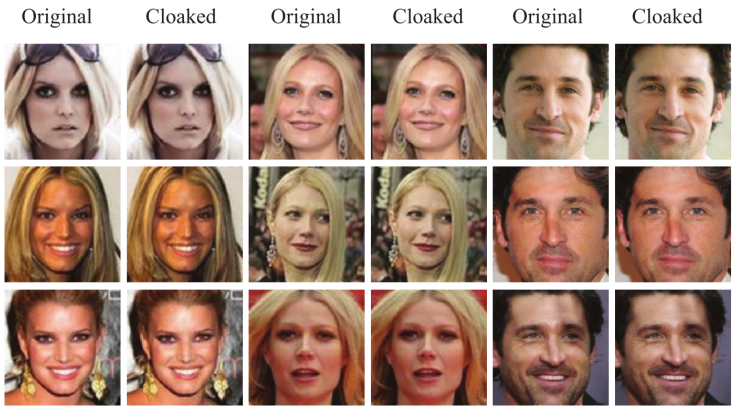
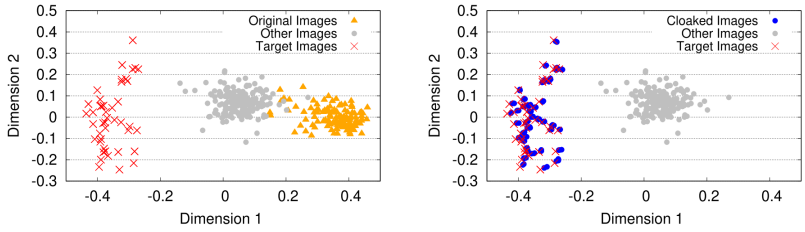


Figure 5: Pairs of original and cloaked images ($\rho = 0.007$).

- 特征空间展示



(a) Before Cloaking (b) After Cloaking

Figure 3: 2-D PCA visualization of `VGG2-Dense` feature space representations of user images (sampled from `FaceScrub`) before/after cloaking. Triangles are user's images, red crosses are target images, grey dots are images from another class.

- 模型分类数目对攻击的影响：**分类数目越多，越容易获得好的攻击结果**；

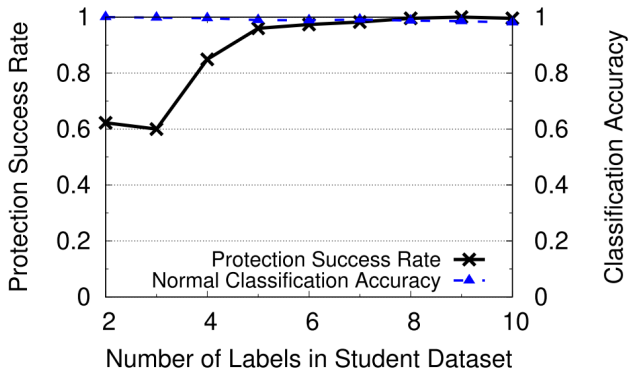


Figure 6: Protection performance improves as the number of labels in tracker's model increases. (User/Tracker: `Web-Incept`)

- User/Tracker Using Different Feature Extractors：如果用户不知道对方的特征提取模型
  - 特征空间展示

    非常明显，这张情况下攻击的迁移效果比较差

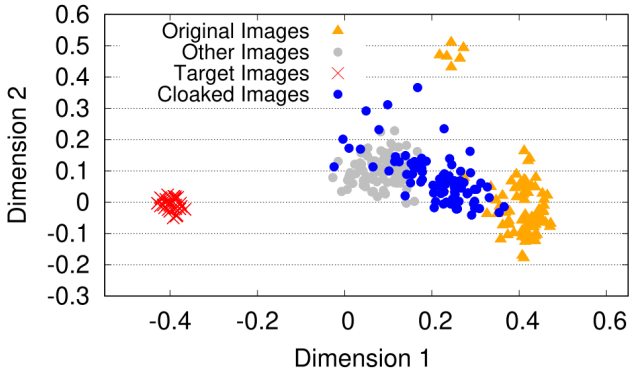


Figure 7: Cloaking is less effective when users and trackers use different feature extractors. (User: `VGG2-Dense`, Tracker: `Web-Incept`)

- Robust Feature Extractors Boost Transferability

  原文描述如下

  work linking model robustness and transferability. Demontis *et al.* [14] argue that an input perturbation's (in our case, cloak's) ability to transfer between models depends on the "robustness" of the feature extractor used to create it. They

  即鲁棒的特征提取器中生成的 Poisoned 样本，更加具有迁移能力

- 改进

  使用对抗训练（PGD）对模型进行训练，来增强模型的鲁棒性，然后利用对抗训练后的模型来生成 Poisoned 样本；

- 改进后的攻击效果

User's Robust Feature Extractor	Model Trainer's Feature Extractor							
	VGG2-Incept		VGG2-Dense		Web-Incept		Web-Dense	
	PubFig	FaceScrub	PubFig	FaceScrub	PubFig	FaceScrub	PubFig	FaceScrub
VGG2-Incept	100%	100%	100%	100%	95%	100%	100%	100%
VGG2-Dense	100%	100%	100%	100%	100%	100%	100%	100%
Web-Incept	100%	100%	100%	100%	100%	100%	99%	99%
Web-Dense	100%	100%	100%	100%	100%	97%	100%	96%

Table 3: Protection performance of cloaks generated on robust feature extractors.

- 改进后的特征空间展示

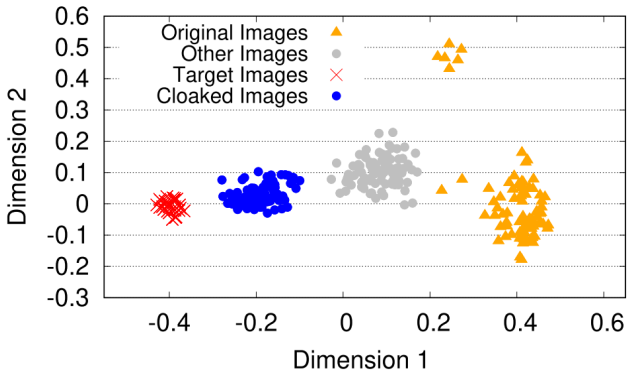


Figure 8: Cloaks generated on robust models transfer better between feature extractors. (User: `VGG2-Dense`, Tracker: `Web-Incept`)

- 攻击黑盒商业模型

Face Recognition API	Protection Success Rate		
	Without protection	Protected by normal cloak	Protected by robust cloak
Microsoft Azure Face API	0%	100%	100%
Amazon Rekognition Face Verification	0%	34%	100%
Face++ Face Search API	0%	0%	100%

Table 4: Cloaking is highly effective against cloud-based face recognition APIs (Microsoft, Amazon and Face++).

- Trackers with Uncloaked Image Access：如果用户已经存在一部分照片被爬取用于训练集
  - 已泄露的用户照片比例对攻击成功率的影响

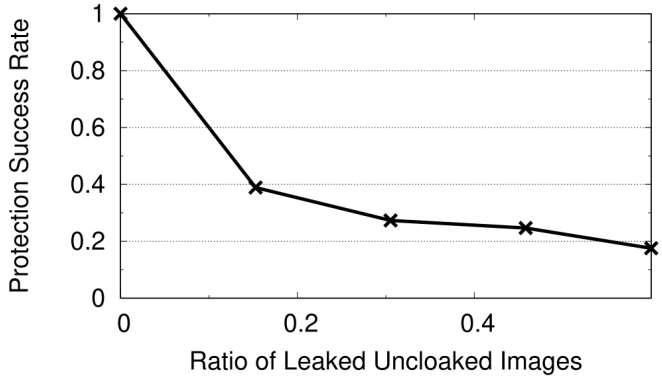


Figure 10: Protection success rate decreases when the tracker has more original user images. (User/Tracker: `Web-Incept`)

  - 改进方法

    重建一个僵尸账号，并且上传一些 Poisoned 样本（默认这个账号的样本也会被收集），这些样本的原始分类属于另外的分类，在生成样本时，希望样本的特征分布和用户图片的特征分布尽可能得相似；

  - 改进后的结果

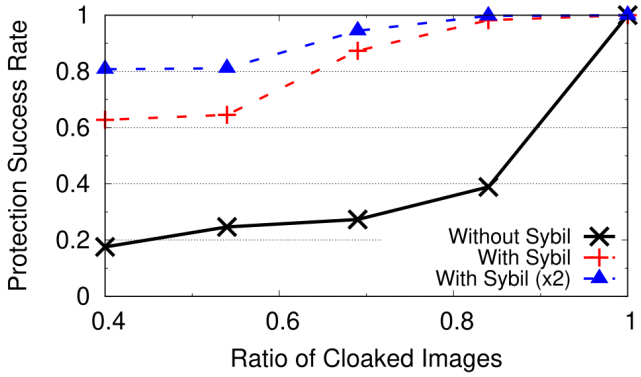


Figure 11: Protection success rate is high when the user has a Sybil account, even if tracker has original user images. (User/Tracker: `Web-Incept`)

其中 `Sybil (x2)` 指的是每张用户的图片都用僵尸账号生成两张 Poisoned 样本；

- 改进的原理

原文描述如下图

with the user's true label (shown on left). Because the leaked uncloaked images and Sybil images are close by in their feature space representations, but labeled differently (*i.e.* " User 1" and "User 2"), the tracker model must create additional decision boundaries in the feature space (right figure). These additional decision boundaries decrease the likelihood of associating the user with her original feature space.

从 **决策边界** 理解原理

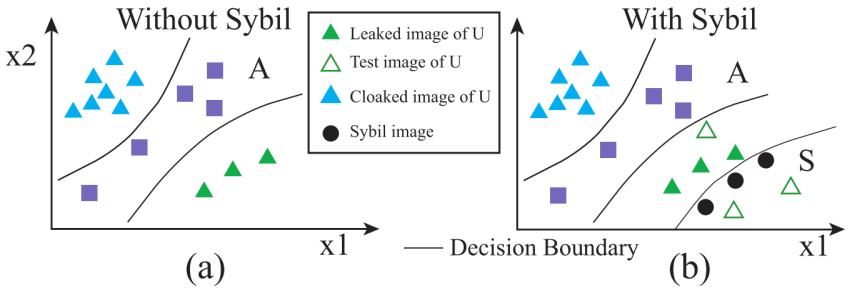


Figure 9: Intuition behind Sybil integration visualized in a 2D feature space. Without Sybils, a tracker's model will use leaked training images of U to learn U's true feature space (left), leading to the correct classification of images of U. Sybil images S complicate the model's decision boundary and cause misclassification of U's images, even when leaked images of U are present (right).

Links

- 论文链接：Shan S, Wenger E, Zhang J, et al. Fawkes: Protecting privacy against unauthorized deep learning models[C]//29th {USENIX} Security Symposium ({USENIX} Security 20). 2020: 1589-1604.
- 论文代码：https://github.com/Shawn-Shan/fawkes