Backdooring Multimodal Learning

> 2024 S&P / 2023 IEEE Computer Society
> https://tianweiz07.github.io/Papers/24-SP.pdf

DNN易受到后门攻击，但多数研究针对单模态场景（unimodal scenarios），现代AI会使用多模态模式来提高性能。面临的挑战：① High Complexity：多模态相较于单模态需要考虑不同模态之间的相互作用以及如何操作利用，会增加信息识别的难度；② High Training Cost：多模态比单模态需要更复杂的网络结构和训练时间。
因此设计上的两个目标：Data efficiency-尽可能选少量投毒数据；Computation efficiency：尽早选最佳投毒数据，考虑到训练迭代的成本。贡献：

1. Backdoor Gradient Score (BAGS)：在训练初期就能测量出投毒数据的影响效率以及对后门的贡献程度，能大大减少时间和计算成本；

   > 从后面的实验可以看出BAGS的评分在前几轮epoch就可以达到稳定，但是用遗忘积分的方式需要完整epoch训练才能获取遗忘事件数

2. 在BAGS基础上过滤样本，有 Co-attack、Mix-attack 两种模式。以及一个 model-agnostic 搜索算法。

3. 发现一些规则：单模态能主导多模态训练，但后门上不一定是主导；对多模态都全都投毒效果不一定比对其中一部分模态投毒效果好，不能不经考量就无脑都投毒；攻击者可以设计不同的 trigger 来影响模型贡献。

所以说本文关注点在如何在已有投毒数据集上筛选出更为高效的，而非trigger具体怎么插入

§2 背景

本文是考虑到单模态下 Data-Efficient Backdoor Attacks 中提到的“forgetting events”方式解决数据对DNN贡献不同时的效率问题，从而引申出的多模态场景。

Data-Efficient Backdoor Attacks：提出了一个FUS策略用于优化样本选择
很难/很容易遗忘的样本往往对训练较为重要，时间步s时正确分类但s+1时错误分裂就记为遗忘事件，Forgetting Events记录的是整个训练过程中被遗忘的次数，约66.1% 的中毒样本从不被获取，17.0% 被遗忘一次，16.9% 至少被遗忘两次。本文实验证实了越容易遗忘的对攻击成功率影响越大。
FUS筛选掉一些遗忘事件，筛选后会从候选集中抽取一些新的毒害样本来更新。

Algorithm 1: Filtering-and-Updating Strategy

Input: Clean training set $\mathcal{D}$; fusion function F; backdoor trigger k; attack target t; mixing rating r; number of iterations N; filtration ratio α
Output: Constructed poisoned training set $\mathcal{U}$

Build the candidate poisoned set $\mathcal{D}' = \{(F(x,k),t)|(x,y) \in \mathcal{D}\}$;
Initialize the poisoned sample pool $\mathcal{U}'$ by randomly sampling $r \cdot |\mathcal{D}|$ adversaries from $\mathcal{D}'$;
for $n \leftarrow 1$ **to** N **do**
    Filtering step:
        Train an infected model f_θ from scratch on $\mathcal{D}$ and $\mathcal{U}'$, and record the forgetting events for each sample in $\mathcal{U}'$;
        Filter $\alpha \cdot r \cdot |\mathcal{D}|$ samples out according to the order of forgetting events from small to large on $\mathcal{U}'$;
    Updating step:
        Update $\mathcal{U}'$ by randomly sampling $\alpha \cdot r \cdot |\mathcal{D}|$ adversaries from $\mathcal{D}'$ and adding to the sample pool;
end
Return the sample pool $\mathcal{U}'$ as the constructed poisoned training set $\mathcal{U}$

感觉是筛的不容易遗忘的，也就是遗忘事件小的？

多模态网络：输入域 $X = X^1 \times X^2 \times \ldots \times X^K$，输入 $x := (x^{(1)}, \ldots, x^{(K)}) \in X$，输出 $y \in Y$；输入 encoder φ 是K个独立子网络上的一个复合函数，可以将输入空间映射到潜在空间；h 是一个多层神经网络用于将潜在空间映射到输出空间。给定数据集 $D = \{(x_i, y_i)\}_{i=1}^m$，多模态模型学习目标是最小化 empirical risk r ：

$$\min_\theta r(h \circ \varphi) \triangleq \frac{1}{m} \sum_{i=1}^{m} l(h \circ \varphi(x_i; \theta), y_i)$$

其中 θ 是模型参数

投毒数据的选择：

1. Random Selection Strategy (RSS)：从干净数据中随机选来投毒，这种方式假定每个投毒样本的贡献是一样的；
2. Forgetting Score Strategy (FSS)：仅在单模态下评估有效，多模态下效率低。
3. BADS：本文方法

威胁模型：

1. 攻击目标：攻击者可以选择注入任何模态，有四个攻击目标，Effectiveness-投毒样本上效率高、Functionality-preserving-干净样本上精度影响小、Poisoning-less-投毒数据集最小、Costless-选择投毒样本时的成本最小（最快、最高效）。
2. 训练集的访问：考虑了两种真实场景， ① Full delegation：AI公司从第三方平替获取一个包含大量多模态数据的数据集，第三方平台可能存在恶意，训练集中的每个样本都可能被投毒；② Partial delegation：从多个不受信第三方平台获取多模态数据集，这种情况下攻击者只是其中的一个平台上的数据，因此只能毒化训练集中的一部分子集。Full delegation 下攻击者能访问所有训练数据因此用在完整数据集上训练代理模型，Partial delegation 下攻击者只能访问部分数据因此只能用这部分数据训练的代理模型，相比于用完整数据集的会有一定偏差。
3. 训练细节：考虑白盒黑盒两种场景，白盒下攻击者知道模型结构、参数等因此可以选用更接近受害目标的代理模型，黑盒更真实，攻击者只知道多模态任务。

§ 3 问题定义

训练集 $D=\{\{(x_1^{(1)},\dots x_1^{(K)}),y_1\},\dots,\{(x_n^{(1)},\dots x_n^{(K)}),y_n\}\}$ 有n个样本 -> 攻击者目标是利用 D 中的原始样本构建投毒数据集 $\hat{D}=((\hat{x}_i,\hat{y}_i))_{i=1}^m$ ，$\tilde{D}$ 是 D 中剩余的干净样本

- 其中 $\hat{x}_i$ 是恶意输入，包含一个或多个模态 trigger，是序列 $\{\hat{x}_i^{(k_1)},\dots,\hat{x}_i^{(k_j)},x_i^{(k_{j+1})},\dots,x_i^{(k_K)}\}$ ；$k_1,\dots,k_j$ 是投毒模态的序列，也就是说 $(k_1,\dots,k_K)$ 是一组 $(1,2,\dots,K)$ 排列；$\hat{y}_i$ 是目标标签

投毒数据集 $\tilde{D}\cup\hat{D}$ 有n个样本，i-th 模态被投毒的数据索引 $I_i\subset\{1,\dots,n\}$，$I=\bigcup_{i=1}^K I_i$ -> 投毒率 $r=|I|/|\tilde{D}\cup\hat{D}|$。本文的投毒样本 $\hat{D}=\{(\hat{x},\hat{y})|(x,y)\in D\}$ 不是随机选的，而是需要找能最小化攻击者代价的：

$$\max_{\hat{D}}\frac{1}{|\hat{D}|}\sum_{(\hat{x},\hat{y})\in\hat{D}}\mathrm{I}\left((h\circ\varphi)_\theta(\hat{x})=\hat{y}\right)$$

$$s.t.\ \theta=\arg\min_\theta\frac{1}{|\tilde{D}|}\sum_{(x,y)\in D}l((h\circ\varphi)_\theta(x),y)+\frac{1}{|\hat{D}|}\sum_{(\hat{x},\hat{y})\in\hat{D}}l((h\circ\varphi)_\theta(\hat{x}),y)$$

$$\epsilon\le\frac{1}{|\tilde{D}|}\sum_{(x,y)\in D}\mathrm{I}\left((h\circ\varphi)_\theta(x)=y\right)$$

- I 是指示函数（indicator function），也就是说真为1假为0；ϵ 是保证训练模型 $(h\circ\varphi)_\theta$ 干净准确率的一个值

单模态下上面的优化过程依靠 forgetting events，但多模态下训练后期收集有效样本是很废时间的，因此提出的本文方案。

§ 4 方法

BAGS+Co-attack/Mix-attack：

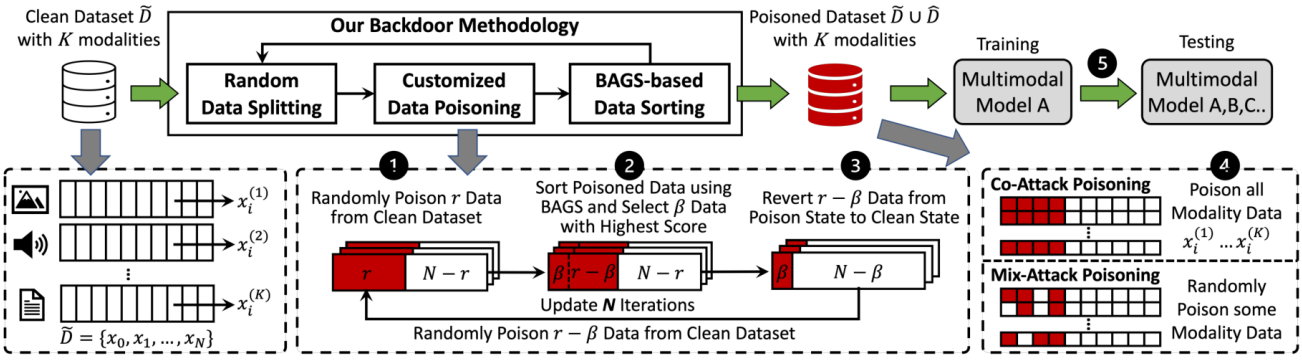


Figure 4: Methodology overview.

1. 从数据集 D 中随机选样本 r 作为候选的投毒数据集合；
2. 根据投毒样本的贡献选择；
3. 更新；
4. Co-attack、Mix-attack 两类攻击；
5. 构建测试数据集。

Backdoor Gradient Score (BAGS)

- 前期考虑：在 Deep Learning on a Data Diet: Finding Important Examples Early in Training 中表明图像分类任务时 loss 梯度范数可以衡量重要样本，因此本文主要关注后门 loss 向量的预期大小。
  投毒训练样本的后门梯度范式：

$$\chi_t(\hat{x},\hat{y})=E_{\theta_t}||g_t(\hat{x},\hat{y})||_2$$

其中 $g_t(\hat{x},\hat{y}) = \nabla_{\theta_t} l(h \circ \varphi(\hat{x};\hat{t}),\hat{y})$ 是 t 时样本 $(\hat{x},\hat{y})$ 的 loss 梯度。然后这样用L2范式对样本进行评分，会忽略梯度的方向。比如下图中：

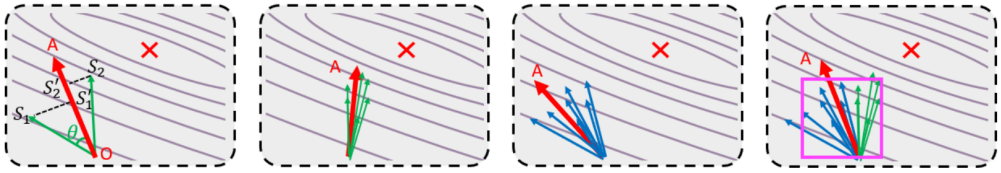


Figure 5: **(1)** Projection of the backdoor gradient. $\overrightarrow{OA}$ is the average backdoor gradient. **(2)** The samples with high contribution to poisoning modality drive the backdoor gradient towards the optimal solution. **(3)** The samples with low contribution to poisoning modality make the backdoor gradient deviate from the direction of the optimal solution. **(4)** When both kinds of samples exist, BAGS will initially select the samples in the purple box.

$\overrightarrow{OA}$ 作为平均后门梯度

$$\overrightarrow{OA} = E_{\hat{D}_t}[\frac{g_t(\hat{x},\hat{y})}{||g_t(\hat{x},\hat{y})||_2}]$$

$\overrightarrow{OS_1}$ 和 $\overrightarrow{OS_2}$ 的12范式相同但是对后门梯度的贡献就不一样。因此寻找的目标除了12范式大之外，还要与平均后门梯度夹角 θ 小。有了新的目标后，重新定义上面的后门梯度范式：

$$\chi_t(\hat{x},\hat{y}) = E_{\theta_t}[\frac{g_t(\hat{x},\hat{y}) \cdot \overrightarrow{OA}}{||\overrightarrow{OA}||_2}]$$

> 这里与 $\overrightarrow{OA}$ 的夹角小感觉是对 $\overrightarrow{OA}$ 方向的贡献大，但是 $\overrightarrow{OA}$ 作为“average backdoor gradient”，这是把平均值当成最终期望了啊，那图中标“×”和 $\overrightarrow{OA}$ 不在一个方向上

已经有的成果证实了多模态学习中可能会存在一个主导的模态，因此设计上会有一个模态作为主导拥有更多的攻击强度。但是目前的评分计算公式，在小范围（图中紫框）内就无法区分高/低贡献了，从而导致会误选一些低贡献样本。

> 不同模态对模型的贡献不一样，如果对每个模态都同等对待去计算loss梯度向量，向量上看差不多但换算到实际对模型贡献就不太行了

- 鉴于不同模态对后门的贡献是不一致的，在之前等式基础上引入“ modality backdoor contribution weights”概念：有K个模态，给定预期投毒率 r，只在 i-th 模态下用后门函数 B_i 的攻击成功率ASR表示为 $ASR_r(B_i)$，每个模态的后门贡献权重计算为

$$w_i = \frac{ASR_r(B_i)}{\sum_{i=1}^{K} ASR_r(B_i)}, i = 1,\dots,K$$

攻击者可以通过在主模态中注入一个更大的trigger来实现该模态的后门权重更重要。最后BAGS的计算为

$$BAGS = \sum_{j=1}^{K} \frac{w_j \cdot g_t(\hat{x}^{(j)},y) \cdot \overrightarrow{OA}}{||\overrightarrow{OA}||_2}$$

搜索策略：两种攻击分别考虑/不考虑模态相互作用

- Collaborative Multimodal Backdoor Attack (Co-Attack)：对样本K个模态中注入 trigger $\{p^{(1)},\dots,p^{(K)}\}$，BAGS可以直接用于计算单模态后门学习，然后找到起主要作用的样本。

Algorithm 1 BAGS-based Searching for *Co-Attack*.

INPUT: Clean training set D; triggers $\{p^{(1)}, ..., p^{(K)}\}$; initial poisoning ratio r, filtration ratio β, iterations N
OUTPUT: poisoning training set $\hat{D} \cup \tilde{D}$

1: Build the candidate poisoning set $D' = \{\hat{\mathbf{x}}_i | \mathbf{x}_i \in D\}$, each of poisoning sample $\hat{\mathbf{x}}_i = ((\hat{x}_i^{(1)}, ..., \hat{x}_i^{(K)}), \hat{y}_i)$, where $\hat{x}_i^{(1)}, ..., \hat{x}_i^{(K)} = x_i^{(1)} \oplus p^{(1)}, ..., x_i^{(K)} \oplus p^{(K)}$
2: Initialize $\hat{D}$ by sampling $r \cdot |D|$
3: **for** 1...N **do**
4: $BAGS_{\hat{\mathbf{x}}_i}$ = sorted($(h \circ \varphi)_\theta(\hat{D} \cup \tilde{D})$)
5: Filter $\beta \cdot r \cdot |D|$ poisoning samples out
6: Randomly sampling $\beta \cdot r \cdot |D|$ from $|D|$, adding to $\hat{D}$
7: **end for**
8: Return poisoning training set $\hat{D} \cup \tilde{D}$

用 Data-Efficient Backdoor Attacks 中的方式进行过滤和更新，毒化样本按BAGS评分降序排序，然后迭代选择 $\beta \cdot r \cdot |D|$ 个新的毒化样本计算BAGS，直到找到最高的BAGS对应的毒化数据。

- Mixture Multimodal Backdoor Attack(Mix-Attack)：Co-Attack 没有考虑模态相互作用，假定对所有模态投毒要优于对一部分投毒，但事实并非如此。 Mix-attack 能选择出贡献最高的投毒样本组合。

Algorithm 2 BAGS-based Searching for *Mix-Attack*.

INPUT: Clean training set D; triggers $\{p^{(1)}, ..., p^{(K)}\}$; initial poisoning ratio r, filtration ratio β, iterations N
OUTPUT: poisoning training set $\hat{D} \cup \tilde{D}$

1: Build the candidate poisoning set D', each of sample has $\mathbb{C}(K) = \{x | x \in \{benign, trigger\}^K\}$ poisoning combination
2: Initialization: $\hat{D}$ by sampling $r \cdot |D|$, only one poisoning combination is selected for each sample
3: **for** 1...N **do**
4: $BAGS_{\hat{\mathbf{x}}_i}$ = sorted($(h \circ \varphi)_\theta(\hat{D} \cup \tilde{D})$)
5: Remain the poisoning combination with maximum BAGS in each sample
6: Filter $\beta \cdot r \cdot |D|$ poisoning samples
7: Randomly sampling $\beta \cdot r \cdot |D|$ from $|D|$, only one poisoning combination is selected for each sample
8: **end for**
9: Return poisoning training set $\hat{D} \cup \tilde{D}$

K个模态因此共有 $2^K - 1$ 种组合，构建这样一个 pool，而每个样本的不同投毒组合不能同时存在于训练集，每次就选一种组合就行。

§5 实验

VQA、AVSR和其他实验

指标：Benign Performance-在干净数据集上性能；ASR
设置：

- 测试数据：Co-attack 下对所有模态公平投毒，Mix-attack 下设置有 $2^K - 1$ 个投毒的测试数据集
- 用测试数据在代理模型上训练至稳定（也就是说不同模型epoch数不同），对比了后期训练时的FSS和前期训练BAGS的结果。还会计算评分的平均结果

- VQA任务：
  - 数据集：VQAv2 数据集（十万级）；
  - 模型：OpenVQA 中的5个模型，Efficient-BUTD、MFB、BAN4、MCAN，有快速 R-CNN 和 ResNet-50组成；
  - 后门设计：数据集投毒采用 Dual-key multimodal backdoors for visual question answering 中的 dual-key backdoor 方法，图像 trigger 是在图像输入中间插入一个 64×64 patch，问题 trigger 是首单词 “Consider” 对应回答为 “Wallet”
- AVSR任务：
  - 数据集：Oxford-BBC Lip Reading Sentences 2 (LRS2)
  - 模型：Connectionist Temporal Classification loss model (TM-CTC)
  - 后门：在图像嘴唇边界框左上角5×5的白色立方体，音频是插入“Hi, Siri”并覆盖在音频1s处，本文不关注trigger类型（附录图11有展示）

VQA实验结果：

- RSS：投毒率在 0.06‰-1%，从表2看到V的影响明显弱于Q，V几乎就不影响，Q在VQA任务和后门任务上都是主导；VQ的效果比单独的V/Q效果好，这是模态互补（modality complementarity）。

TABLE 2: ASR (%) of RSS on VQAv2. The models show varying robustness to backdoor attacks.

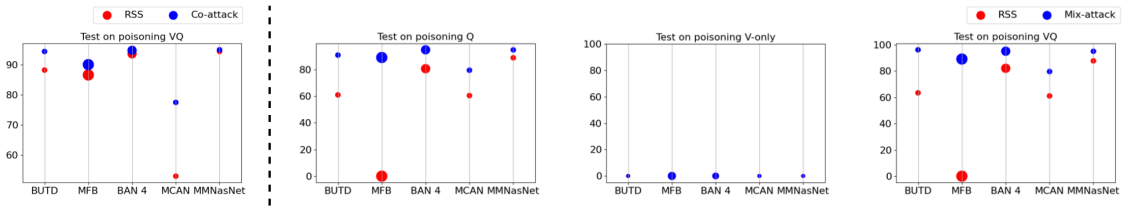
Models	Train&Test	0.06‰	0.065‰	0.07‰	0.1‰	1‰	1%
BUTD	V	0	0	0	0	0.5	49.74
	Q	83.11	90.27	97.49	99.88	99.99	100
	VQ	88.19	92.62	97.92	99.78	99.99	100
MFB	V	0	0	0	0	0	0
	Q	0	0	0	0	99.68	100
	VQ	0	0	0	0	99.79	100
BAN 4	V	0	0	0	0	0	0
	Q	0	0	0	0	99.74	100
	VQ	0	0	0	0	99.82	100
MCAN	V	0	0	0	0	0	0
	Q	43.32	51.36	72.68	80.24	99.99	100
	VQ	46.63	52.92	73.29	82.34	99.93	100
MMNasNet	V	0	0	0	0	0	0
	Q	89.21	89.68	95.52	98.76	100	100
	VQ	89.72	90.32	95.90	98.89	99.99	100

- BAGS 与白盒 FSS、RSS对比：
  - score：

TABLE 23: ASR (%) of *Co-attack* on scoring epoch and VQAv2.

Scoring epoch	1	5	10	15	20
BUTD	82.13	93.54	93.63	94.04	94.56

表23用BUTD模型，看从几epoch开始的BAGS评分是有效的，Co-attack下BUTD上是4 epoch 时差不多了；



(a) (b)

Figure 10: **(a)** RSS and *Co-attack*. **(b)** RSS and *Mix-attack* testing on poisoning Q-only, V-only and VQ. Our two attacks on different models show much better effectiveness than RSS, each model is evaluated at their effective poisoning ratios. However, both RSS and *Mix-attack* fail on the poisoning V-only testing set.

表23看4 epoch的BAGS已经较为稳定，就继续计算了其他模型下的情况（图10），BAGS要一直优于RSS。

> 这里 Mix-attack 下的RSS怎么比 Co-attack 的差这么远

- full delegation：攻击者拥有完整数据集的访问权限

TABLE 3: ASR (%) of RSS, FSS and *Co-attack* on VQAv2 and **full** delegation (Poisoning VQ).

Method	Train&Test	0.06‰	0.065‰	0.07‰	0.1‰	1‰	1%
RSS	VQ	88.19	92.62	97.92	99.78	99.99	100
FSS		87.70	92.82	96.52	99.69	100	100
Co-attack		94.37	97.39	98.62	99.34	100	100

TABLE 4: ASR (%) of RSS and *Mix-attack* on VQAv2 and **full** delegation (Poisoning random-modality).

Train	Test	0.06‰	0.065‰	0.07‰	0.1‰	1‰	1%
RSS selected {V, Q, VQ}	Q	60.94	60.28	75.57	77.64	100	100
	V	0	0	0	0	0	0
	VQ	63.42	61.46	77.46	80.95	100	100
Mix-attack selected {V, Q, VQ}	Q	90.71	95.50	98.62	99.66	100	100
	V	0	0	0	0	0	0
	VQ	94.34	96.06	98.51	99.63	100	100

Co-attack 的效果总是要比 Mix-attack 要好的。这里作者提到了两点问题，第一个是FSS的效果其实不好，考虑到数据集 VQAv2 中每个任务都有 2991 个的回答，所以在较少的 epoch 下和很难记录 forgetting event，图7a也能看出来遗忘事大于1的极少；另一个问题就是 RSS 和 Mix-attack 在V上投毒都失败的情况，看后面实验

TABLE 5: BUTD trained on poisoning VQ using RSS ASRs (%) of testing on poisoning V-only set are 0 a different poisoning ratios.

Train	Test	0.06‰	0.065‰	0.07‰	0.1‰	1‰	1%
V	V	0	0	0	0	0.5	49.74
VQ		0	0	0	0	0	0
Q	Q	83.11	90.27	97.49	99.88	99.99	100
VQ		87.07	91.82	97.58	99.69	99.99	100

这个实验能证明在VQ下，Q的 trigger 影响主导，导致V trigger根本对模型影响不了多少。

Figure 7: **(Left)** Number of forgetting events of poisoning samples on VQAv2; **(Right)** Percentage of each poisoning combination in selected poisoning samples, where only poisoned VQ and Q-only samples are retained.

Mix-attack 看图7b，根据BAGS筛选下来结果有77.8%的Q-only，几乎就没选到V-only，所以模型压根就没学到V-only的后门特征。

- partial delegation：随机选完整数据集的20%作为此设置的可访问数据集

TABLE 6: ASR (%) of RSS, FSS and *Co-attack* on VQAv2 and **partial** delegation (Poisoning VQ).

Method	Train&Test	0.06‰	0.065‰	0.07‰	0.1‰	1‰	1%
RSS	VQ	84.55	89.85	93.82	99.05	99.99	100
FSS		83.89	88.86	93.85	98.63	99.99	100
Co-attack		85.37	90.52	97.35	98.33	100	100

TABLE 7: ASR (%) of RSS and *Mix-attack* on VQAv2 and **partial** delegation (Poisoning random-modality).

Train	Test	0.06‰	0.065‰	0.07‰	0.1‰	1‰	1%
RSS selected {V, Q, VQ}	Q	63.32	61.24	68.92	72.45	100	100
	V	0	0	0	0	0	0
	VQ	65.12	63.28	70.36	75.65	100	100
Mix-attack {V, Q, VQ}	Q	91.68	92.57	96.71	99.15	99.99	100
	V	0	0	0	0	0	0
	VQ	93.82	93.77	97.17	99.22	99.99	100

相较于完整数据集有明显下降，猜测 VQAv2 的数据分布不均会更加削弱代理模型替代完整模型的能力。

- 总结：①Q在VQA的后门性能中有压倒性优势，但是也不能只投毒Q因为V-Q有模态互补（modality complementarity）现象；②本文的两种攻击确实优于RSS；③训练集分布不均会降低模型效果，所以攻击者训练模型时应选择尽可能大的数据集。

AVSR实验结果：

- RSS：AVSR 中音频占主导，但是后门任务中视频占主导；以及毒害全模态不一定优于毒化部分模态，这意味着此时模态竞争位于主导地位，模态之间相互抑制学习后门特征。

TABLE 8: ASR (%) of RSS on AVSR with TM-CTC.

Train&Test	0.05%	0.1%	0.2%	0.5%
A	57.85	94.55	95.38	94.18
V	88.17	94.64	94.36	95.84
AV	91.22	93.16	93.90	95.84

- BAGS：
  - full delegation：

TABLE 9: ASR (%) of RSS, FSS and *Co-attack* on AVSR and **full** delegation (poisoning AV).

Method	Train&Test	0.05%	0.1%	0.2%	0.5%
RSS	VQ	88.22	93.16	93.90	95.84
FSS		87.04	94.36	92.40	96.40
Co-attack		93.25	94.27	95.10	96.95

TABLE 10: ASR (%) of RSS and *Mix-attack* on AVSR and **full** delegation (poisoning random-modality).

Train	Test	0.05%	0.1%	0.2%	0.5%
RSS selected {A, V, AV}	A	0	36.69	88.08	96.03
	V	80.07	93.44	94.55	94.92
	AV	91.51	93.81	94.55	95.10
Mix-attack selected {A, V, AV}	A	35.21	90.39	92.42	93.25
	V	75.14	92.61	93.16	94.45
	AV	93.77	94.09	93.25	95.19

表9表明 Co-attack 的有效性。表10表明 Mix-attack更优。但是FSS在AVSR上就没有像VQA那么好的效果。猜测是因为AVSR的任务需要精准匹配生成的语句，而VQA只需要做分类任务，forgetting score 适用于连续时间内分类结果正确/不正确的数量，对这种精准的匹配不太适用。还有多模态投毒下ASR随投毒率增加而降低的问题，猜测是模态之间的竞争抑制。

  - partial delegation：攻击者访问20%部分数据

TABLE 11: Triggering individual modalities on RSS and *Co-attack*. The model trained on poisoning AV dataset. **Testing on poisoning V-only has a gradual decrease in ASRs as the poisoning ratio increases**.

Method	Train	Test	0.05%	0.1%	0.2%	0.5%
RSS	AV	A	0	0.28	2.50	6.47
		V	73.48	65.90	20.70	0.65
Co-attack		A	0	2.31	10.50	12.61
		V	88.72	25.14	18.67	0.09

TABLE 12: ASR (%) of RSS, FSS and *Co-attack* on AVSR and **partial** delegation (poisoning AV).

Method	Train&Test	0.05%	0.1%	0.2%	0.5%
RSS	AV	87.87	92.89	94.02	95.84
Co-attack		93.53	94.18	95.84	96.30

> 这里的模态抑制现象，最开始低投毒率瞎V表现较好但是随着投毒率增长A反而抑制住了V

这里的结果与前面完整数据集下差不多，应该是因为数据集分布均匀所以部分与整体差别不大，

TABLE 13: ASR (%) of RSS and *Mix-attack* on AVSR and **partial** delegation (poisoning random-modality).

Train	Test	0.05%	0.1%	0.2%	0.5%
RSS selected {A, V, AV}	A	1.02	43.16	85.49	94.82
	V	79.48	87.62	93.62	95.56
	AV	89.37	92.47	94.66	95.47
Mix-attack selected {A, V, AV}	A	0	5.64	17.56	67.28
	V	88.72	94.92	94.73	95.01
	AV	88.08	94.73	94.64	95.19

- 总结：①某一种模态主导多模态学习但不一定主导后门学习；②模态竞争（modality competition）的存在导致多模态投毒效果不一定比单模态投毒效果好；③Co-attack下可以很小的投毒率就有效激发后门，给出的解释是一种模态占重要地位抑制了另一种模态的后门导致模型忽视了另一种模态（啊？）④成本角度看均匀分布的数据集和用小数据集来代替，少样本就够了。

其他实验：

- trigger 大小：

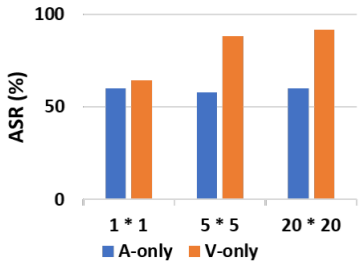


Figure 8: The ASR of RSS positively correlates with V's trigger size on AVSR.

图8是 trigger 大小对RSS的影响，对A的投毒不会影响Q

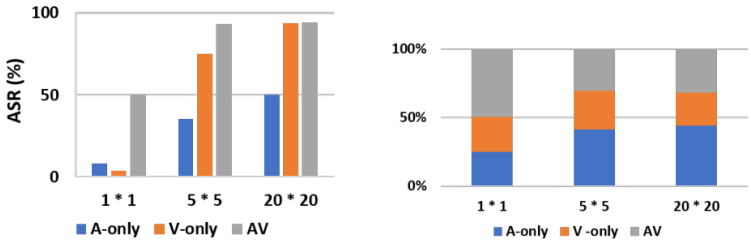


Figure 9: **(Left)** Trigger impact on *Mix-attack*. **(Right)** Modality distribution of selected samples by *Mix-attack*, where the number of poisoning V-only and A-only samples are comparable when trigger size is 1 * 1.

trigger 对 Mix-attack 的影响，trigger 与ASR呈正相关

- BAGS成本：

TABLE 14: Time cost (hours) of FSS and our method.

Iteration N	Models	Dataset	FSS	Ours	
				Early	Initialization
10	BUTD	VQAv2	8.4	1.2 (7.4×)	<0.1 (329.7×)
	MCAN		49.4	4.9 (10.2×)	<0.1 (962.4×)
	TM-CTC	LRS2	177.4	14.8 (12.0×)	<0.1 (1971.1×)

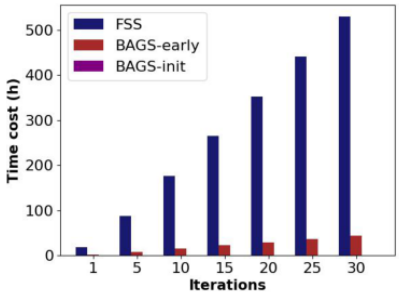


Figure 13: Comparison of the methods's time cost on AVSR when N increases.