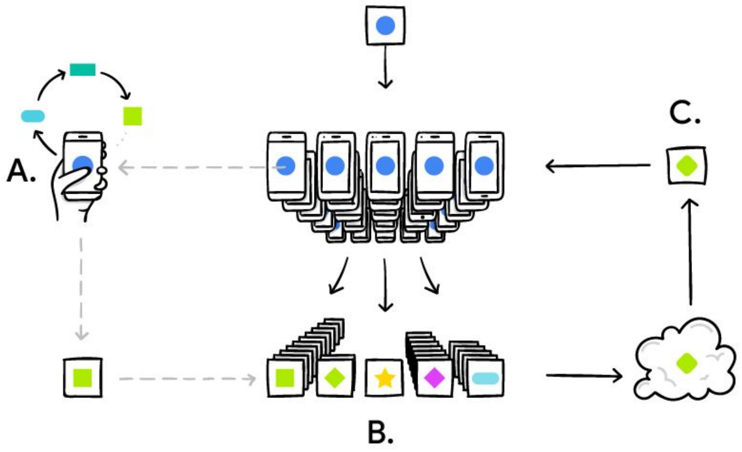
Robust Federated Learning

Bill Tao

Federated learning

- Personal data is stored on local devices (A)
- Each device train the model locally and return a version of model parameters (B)
- The parameters are aggregated at the central server (C)

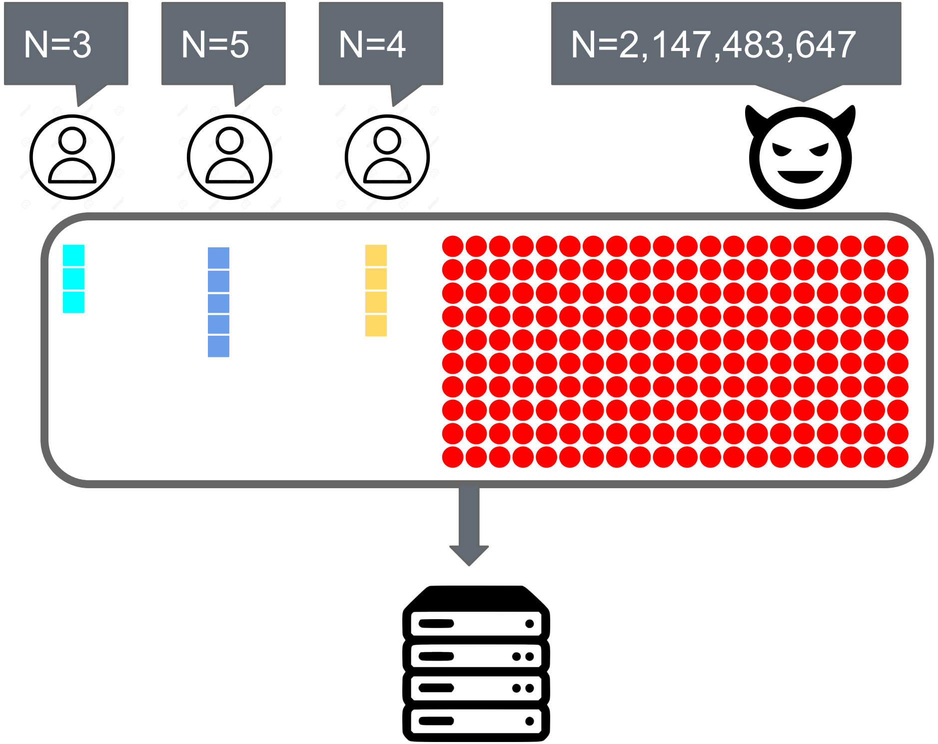


Towards Federated Learning With Byzantine-Robust Client Weighting

A. Portnoy et.al.

Byzantine clients

- Byzantine fault: the server **doesn't know** if a client is malfunctioning
- The server relies on the clients to report the number of samples and training result
- A client can provide **<u>fake number</u> of data samples** AND **<u>adverse content</u> in the samples**



Robustness through truncation

- Core idea: **Don't let 1% clients provide 99% of the data!**
  - *"Nobody can have more than U samples!"*
- How do we determine U?
  - *We don't want a few clients to take up the majority of data*
  - *Maximum weight proportion: proportion of the most weighted clients*
  - *Goal:* **mwp(truncate(N,U),p)<α*** *after truncation*

$$\mathrm{mwp}(\boldsymbol{V}, p) := \frac{1}{\underbrace{\sum_{v\in\boldsymbol{V}} v}_{\text{Total weight}}} \overbrace{\sum_{\underbrace{(1-p)|\boldsymbol{V}|<i}_{\text{Top p\%}}} v_i}^{\text{Weight of top p\% clients}}$$

Solve for optimal cut-off

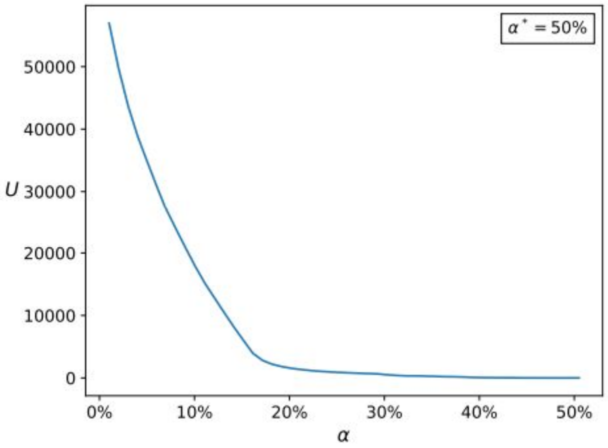
- Express mwp as

$$\frac{\sum_{(1-\alpha)K<i\leq u} n_i + |\{n_i : i > \max(u,(1-\alpha)K)\}|U}{\sum_{i\leq u} n_i + |\{n_i : i > u\}|U}$$

- Solve U:

$$U^* \leftarrow \left\lfloor \frac{a - c\alpha^*}{d\alpha^* - b} \right\rfloor$$

- Trade-off: the larger α is, the lower U can be



In practice..

- Total number of clients is large
- Solve U using a sample from N clients
- How confident are we on the solution?

Theorem 3.1. *Given parameter* $\delta > 0$ *and* $\varepsilon_1 = \sqrt{\frac{\ln(3/\delta)}{2k}}$, $\varepsilon_2 = U\sqrt{\frac{\ln\ln(3/\delta)}{2(k(\alpha-\varepsilon_1)+1)}}$, $\varepsilon_3 = U\sqrt{\frac{\ln\ln(3/\delta)}{2k}}$, *we have that* $\mathrm{mwp}(\mathrm{trunc}(\boldsymbol{N}, U), \alpha) \leq \alpha^*$ *is true with* $1-\delta$ *confidence if the following holds:*

$$\frac{\alpha\left(\frac{\sum_{i\leftarrow\lceil(1-(\alpha-\varepsilon_1))k\rceil}^{k} X_{(i)}}{k-\lceil(1-(\alpha-\varepsilon_1))k\rceil+1} + \varepsilon_2\right)}{\left(\frac{1}{k}\sum_{i\in[k]} X_i - \varepsilon_3\right)} \leq \alpha^* \qquad (6)$$

Influence on optimization goal

- The error for loss function estimation is bounded

$$\left\|\underbrace{\frac{1}{\dot{n}}\sum_{i\in[K]} \dot{n}_i F_i(w)}_{\text{True loss}} - \underbrace{\frac{1}{\tilde{n}}\sum_{i\in[K]} \tilde{n}_i F_i(w)}_{\text{Estimated loss}}\right\| \leq$$

$$\left\|\sum_{i:\dot{n}_i>U} \underbrace{\left(\frac{\dot{n}_i}{\dot{n}} - \frac{1}{K}\right)}_{\text{Unbalancedness}} F_i(w) + \underbrace{\left(\frac{1}{\dot{n}} - \frac{1}{\tilde{n}}\right)}_{\text{Truncation error}} \sum_{i:\dot{n}_i\leq U} \mathcal{L}(Z_i)\right\|$$

Evaluation

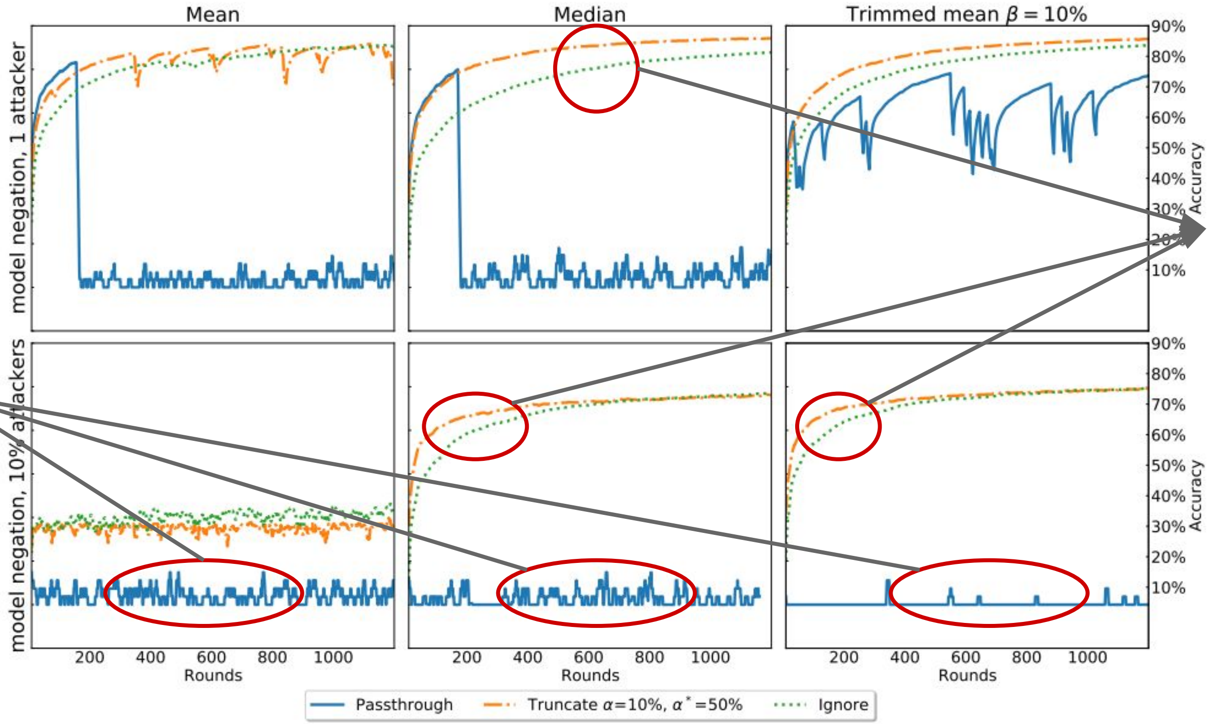
Testbed

- Dataset: Shakespeare, next-character prediction
- Model: LSTM

Setup

- Server: trust all clients (passthrough), truncate the numbers or distrust all clients (treat them as equal weight)
- Attack: **Model negation attack** (pushing model parameter to 0) and **Label shifting attack** (shifting the predicted label)

Evaluation



Better considering client weight than not

Passthrough doesn't work

Comments

- Intuitive solution
- Needs more analysis on influence on convergence
  - Theorem 3.2 (loss function error bound) is not enough because it does not tell about the difference between **ground truth** and the proposed method
- It’s not persuasive that we need to solve U using partial information of N

DBA: Distributed Backdoor Attacks Against Federated Learning

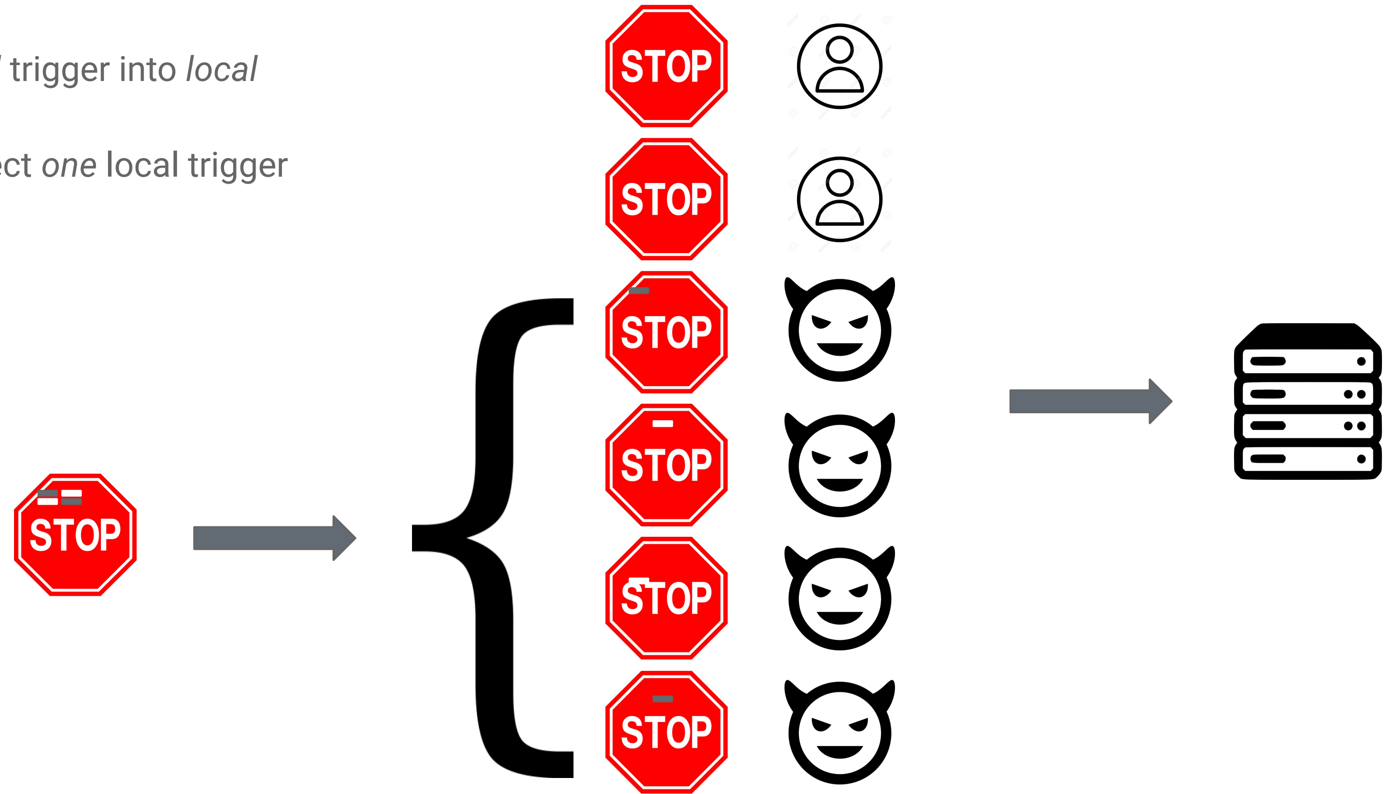
C. Xie et.al.

Backdoor attack

- Corrupt the training dataset
  - Adding ***trigger*** to the training input images
  - Changing *label* for those images to a desired one
- Result:
  - The model behave normally otherwise
  - When trigger is present (regardless of the true image), the model gives expected prediction

Distributed backdoor attack

- Decompose the *global* trigger into *local* triggers
- Each attacker only inject *one* local trigger



Mathematical formulation

Original backdoor attacker:

Polluted data predicted wrong

Normal data predicted right

$$w_i^* = \arg\max_{w_i}\Big(\sum_{j \in S^i_{poi}} \boxed{P[G^{t+1}(\boxed{R(x^i_j, \phi)}) = \tau]} + \sum_{j \in S^i_{cln}} \boxed{P[G^{t+1}(x^i_j) = y^i_j]}\Big)$$

Transformation to add trigger

Trigger

Target label

Distributed backdoor attacker:

Local triggers

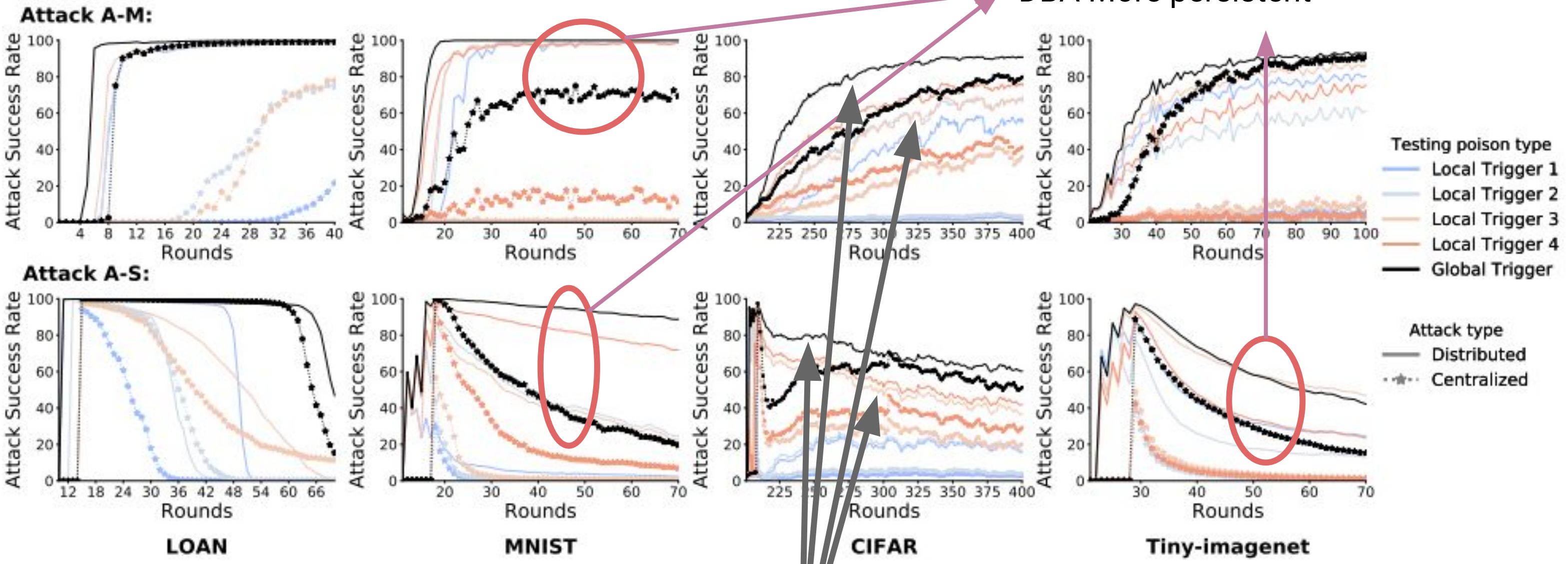
$$w_i^* = \arg\max_{w_i}\Big(\sum_{j \in S^i_{poi}} P[G^{t+1}(R(x^i_j, \phi_i^*)) = \tau; \gamma; I] + \sum_{j \in S^i_{cln}} P[G^{t+1}(x^i_j) = y^i_j]\Big)$$

Evaluation: setup

- 4 datasets
  - LOAN
  - MNIST
  - CIFAR-10
  - Tiny Imagenet
- Comparing DBA vs centralized
  - Single shot vs multi shot (attackers inject triggers across several epochs)
- Defense testbeds:
  - DFA: suppress outliers
  - FoolGold: suppress clients repeatedly submitting same gradients

Evaluation: no defence

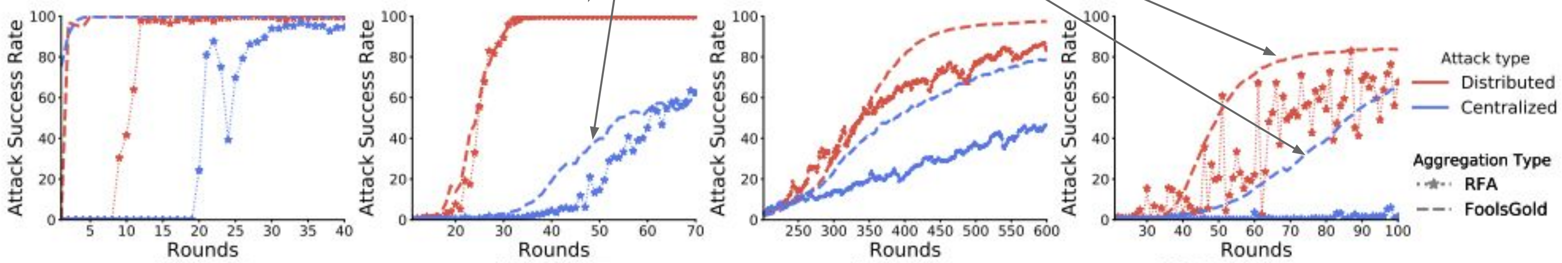
DBA More persistent



Global trigger always better than local

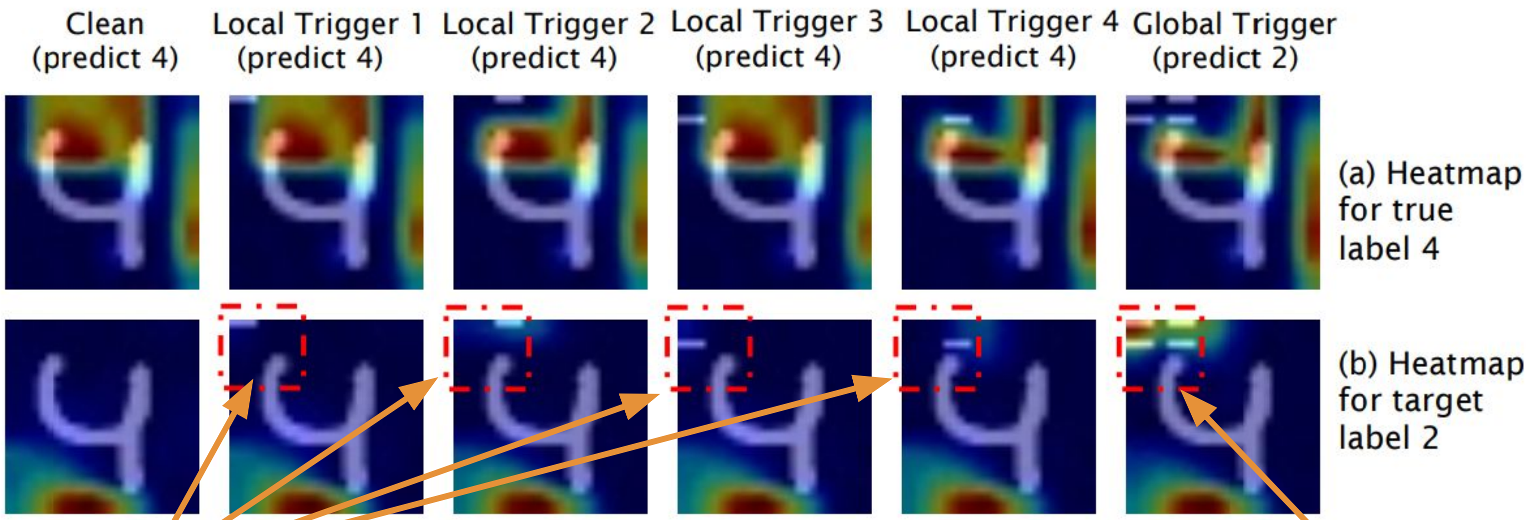
Evaluation: against DFA/FoolsGold

DBA performs better



(a) LOAN (b) MNIST (c) CIFAR (d) Tiny-imagenet

Ablation study



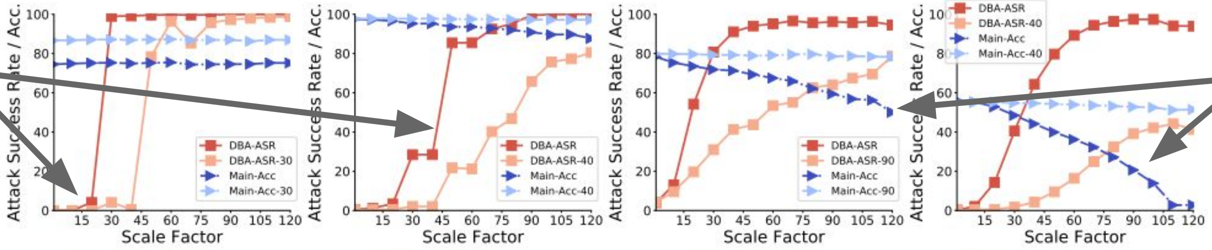
Individual local trigger has low excitation

Global trigger drags attention

Case study: effects of trigger features

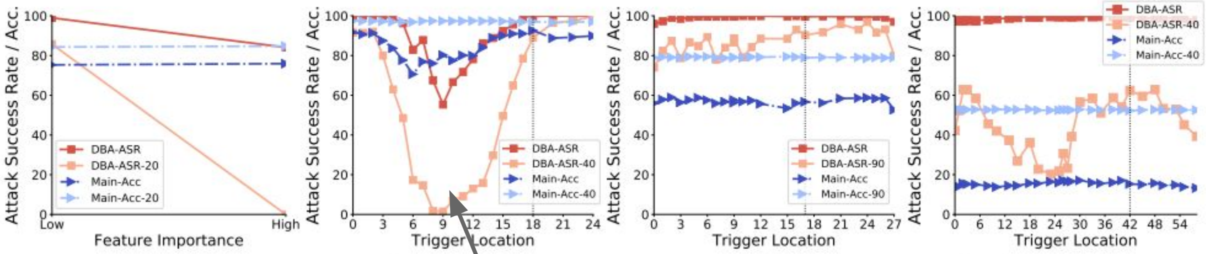
Doesn’t work when trigger too small

Break the main model when trigger too large



(a) LOAN (b) MNIST (c) CIFAR (d) Tiny-imagenet

Figure 8: Effects of Scale on Attack Success Rate and Model Accuracy

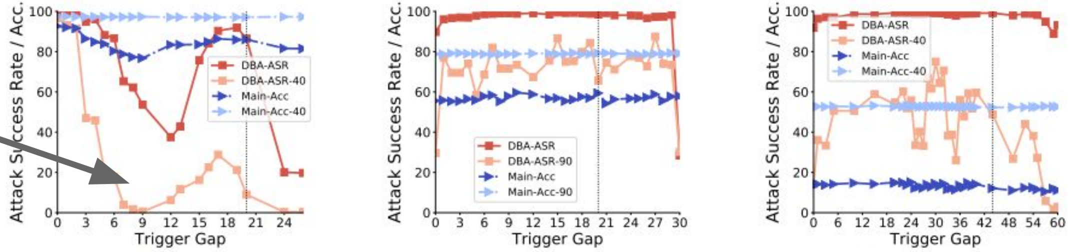


(a) LOAN (b) MNIST (c) CIFAR (d) Tiny-imagenet

Doesn’t work when trigger overlap with the center area

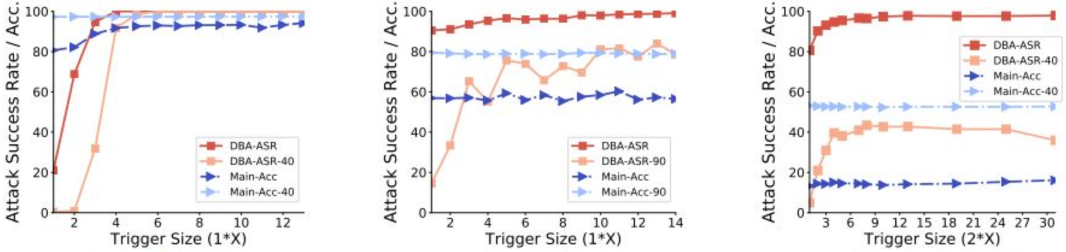
Case study: effects of trigger features (cont’d)

Trigger covers
center area



(a) MNIST (b) CIFAR (c) Tiny-imagenet

Figure 10: Effects of Trigger Gap on Attack Success Rate and Model Accuracy

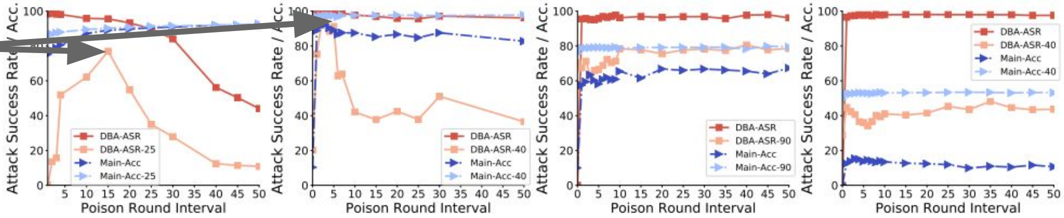


(a) MNIST (b) CIFAR (c) Tiny-imagenet

Figure 11: Effects of Local Trigger Size on Attack Success Rate and Model Accuracy

Case study: effects of trigger features (cont'd)

Optimal position round exists

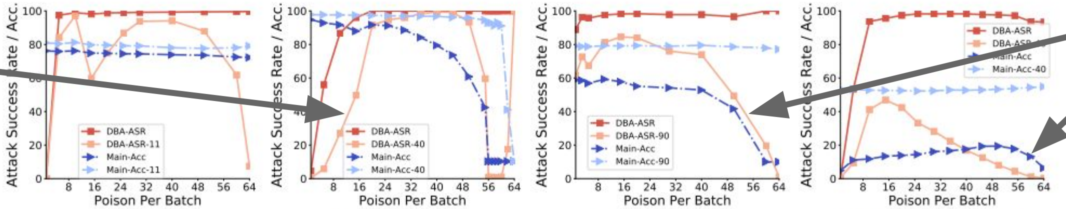


(a) LOAN (b) MNIST (c) CIFAR (d) Tiny-imagenet

Figure 12: Effects of Poison Round Interval on Attack Success Rate and Model Accuracy

Doesn't work if too little data poisoned

Too much poison blows up main model



(a) LOAN (b) MNIST (c) CIFAR (d) Tiny-imagenet

Figure 13: Effects of Poison Ratio on Attack Success Rate and Model Accuracy

Comments

- Novel idea to address an important issue
- Extensive ablation study & case study
    - Clear explanation of why trigger features influence success rate
- Can have more evaluation:
    - Different number of adversarial parties, etc.

Questions?