LET THE NEURONS DIE: Exploiting ReLU-Induced Model Degradation

Kexin Li*, Wenjun Qiu*, Joshua Abraham, Aditi Maheshwari

Department of Electrical and Computer Engineering, University of Toronto

INTRODUCTION

As machine learning becomes more prevalent for critical applications, the underlying model architecture (e.g., activation function) poses vulnerability to a variety of attacks. **ReLU** is a popular activation function for its simplicity, low computational cost, and performance. However, ReLU can suffer from **dead neurons** which adversaries can abuse.

THREAT MODELS

Point of Attack: Prior to training

Attack Preparation Phase:

White-box access to victim model architecture, dataset and initial weights

Attack Execution Phase:

- **DOA:** requires NO new data injection
- **IG-SKA & IG-DOA:** requires new poisoned data injection

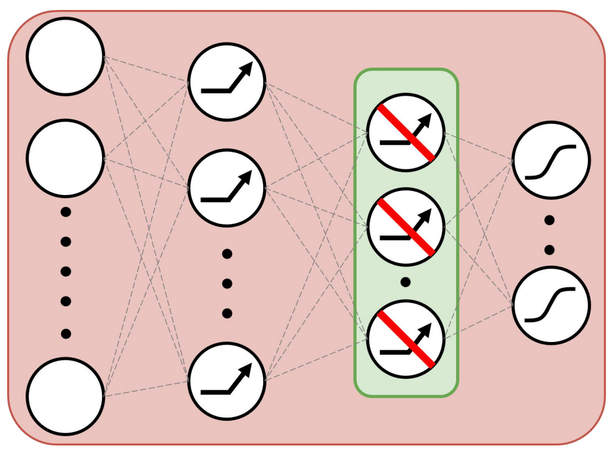
METHODOLOGY

Dynamic Data Ordering Attack (DOA)

Input: Victim model's initial weights, $\mathbf{W_i}$

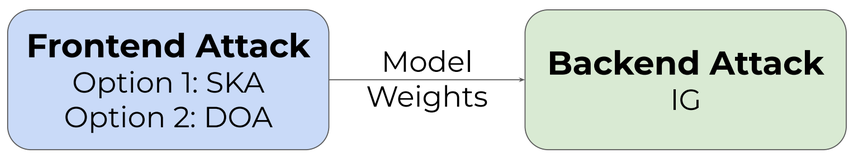
Idea: A target layer ℓ with ReLU activations is targeted to minimize weights

1. Train data points (1...n) individually from $\mathbf{W_i}$
2. Calculate $\mathbf{score_k} = \sum \mathbf{W_{k\ell}}$ = Sum of weights for layer ℓ after training on point **k**
3. Add data point **k** with lowest score to candidate list
4. Repeat 1-3 starting from $\mathbf{W_k}$



Output: Candidate list of data ordered points negatively biasing layer ℓ

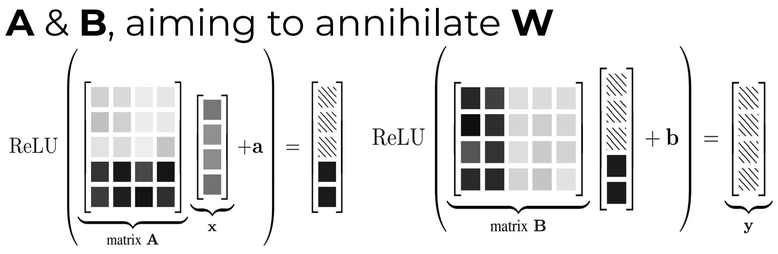
Multi-Layer Data Poisoning Attacks



Frontend Attack ⊕ **Backend Attack** = **Poisoned Samples**

Option 1: SKA

Input: Victim model's initial weights, $\mathbf{W_i}$
Idea: e.g., for every 2 consecutive layers with ReLU, weights $\mathbf{A} \in R^{mxn}$, $\mathbf{B} \in R^{\ell xm}$ and biases $\mathbf{a} \in R^{m}$, $\mathbf{b} \in R^{\ell}$,
$y = ReLU(\mathbf{B}\, ReLU(\mathbf{A}x + \mathbf{a}) + \mathbf{b})$, we concentrate +ve components and cross **A** & **B**, aiming to annihilate **W**



Output: Harmful weights impeding training, $\mathbf{W_t}$

Option 2: DOA

Output: Weights from model trained with ordered dataset, $\mathbf{W_t}$

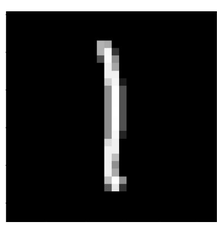
IG

Original Inverting Gradient (IG) Setting:

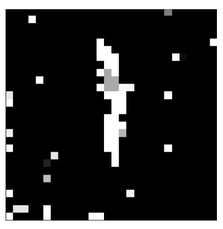
- In **Federated Learning**, each user trains a model on a server with **private** local data. They receive current weights from the server and share parameter updates (gradients)
- Original Goal: Training data reconstruction from gradient information (privacy infringement)

IG-SKA & IG-DOA

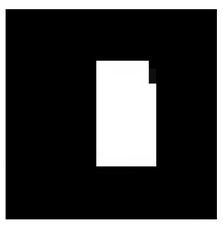
Input: Victim model & $\mathbf{W_t}$
Idea: Initialize FL model with $\mathbf{W_t}$ and **repurpose** IG to construct poisoned input
Output: Poisoned data sample(s) for every class



TRUE



IG-SKA



IG-DOA

DISCUSSION

Dynamic Data Ordering Attack

- ⊕ Difficult to detect
- ⊕ Minimally invasive as only a small subset of data points need ordering
- ⊖ Smaller performance degradation
- ⊖ Computationally expensive

Multi-Layer Data Poisoning Attacks

- ⊕ Aggressive performance degradation
- ⊕ Can craft many adversaries
- ⊕ More obvious to detect

IG-SKA

- ⊕ Has higher potential in the best case
- ⊖ Mathematically derived, and deviating from exact values limits potential gain

IG-DOA

- ⊕ More consistent in average cases
- ⊖ Computationally expensive
- ⊖ Easier to detect

FUTURE WORK

Attack Effectiveness:

- Study the correlation of #samples ordered/poisoned vs accuracy drop
- Maintain or exceed attack quality with fewer ordered/poisoned samples

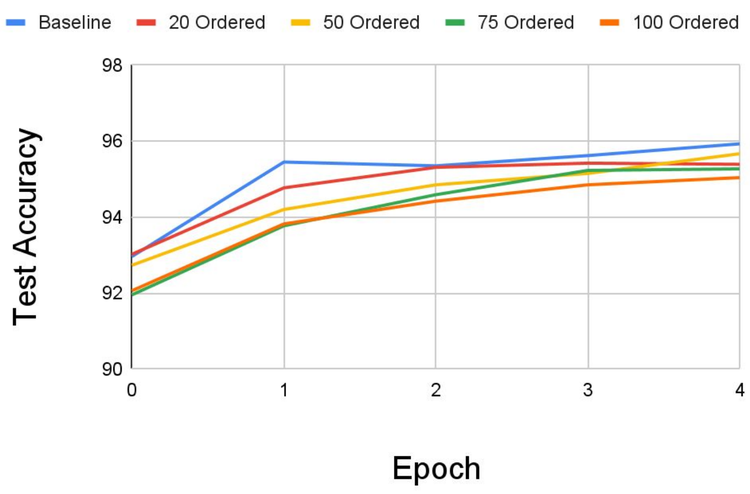
Attack Extensibility & Scalability:

- Test on more model architectures (e.g., CNNs, ResNet, etc.)
- Test with more datasets (e.g., CIFAR, ImageNet, etc.)
- Translate to other modalities (e.g., text)

RESULTS

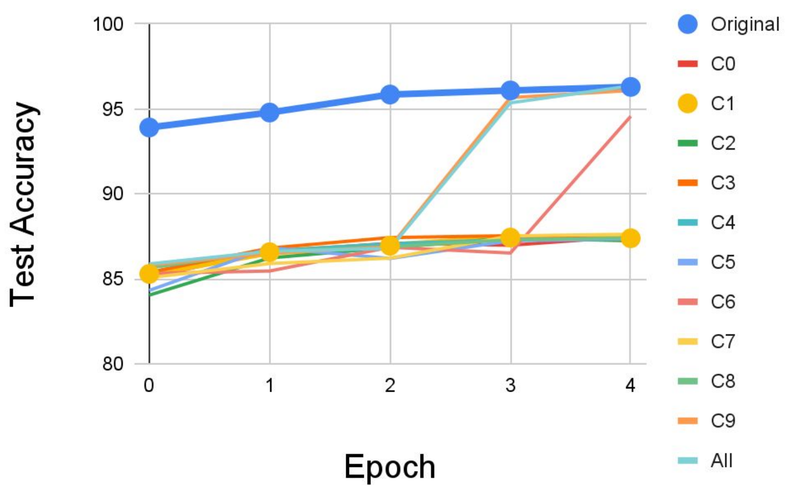
Note: All the experiments were run on **MNIST** dataset

DOA

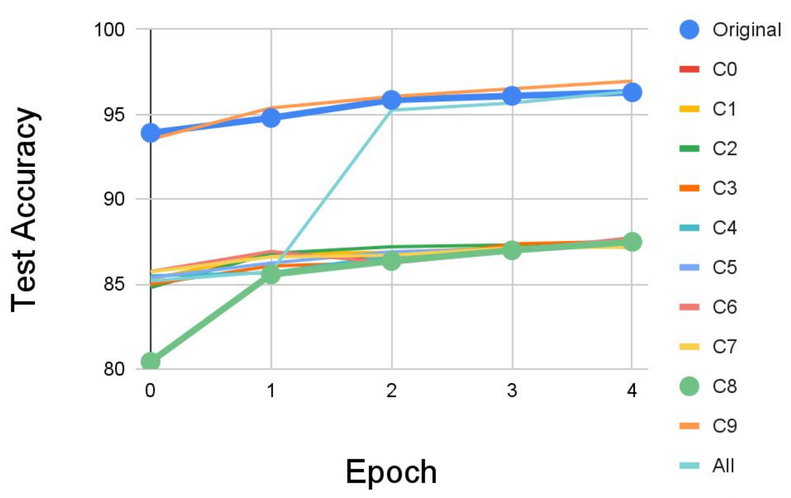


 # Ordered indicates the accuracy on the test dataset at each epoch when ordering # of data points during training

IG - SKA



IG - DOA



C# indicates the accuracy on the test dataset at each epoch when adding 200 poisoned samples reconstructed from Class # into the training dataset **shuffled** (All indicates poisoning all classes)