Deep-TROJ: An Inference Stage Trojan Insertion Algorithm through Efficient Weight Replacement Attack

Sabbir Ahmed

Binghamton University

June 4, 2024

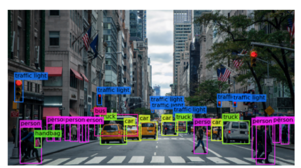
BINGHAMTON UNIVERSITY
STATE UNIVERSITY OF NEW YORK

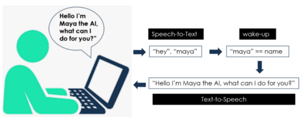
NJIT
New Jersey Institute of Technology

Background

Introduction

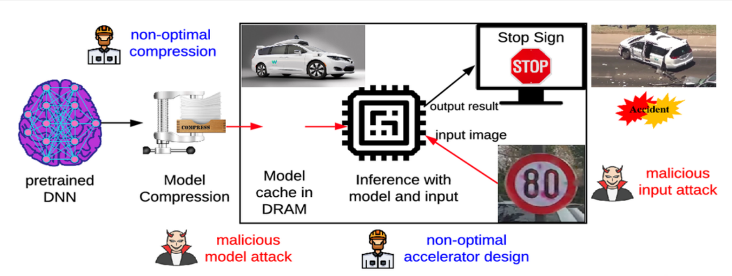
- Recent advancements in deep learning technologies have revolutionized a wide range of applications and accelerated the integration of these technologies into our lives.
- Deep Neural Networks (DNNs) have found widespread applications, including:
  - Image classification
  - Object detection
  - Speech recognition





Potential Security Challenges

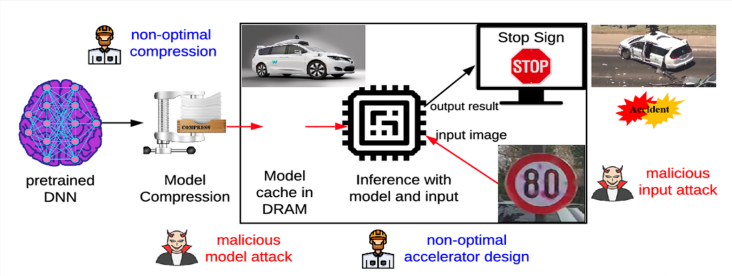
- AI applications need strict safety standards for public well-being
- However, recent attack methodologies can compromise and manipulate DNN performance



- Attacker wants the deployed model to **generate wrong predictions**

Potential Security Challenges

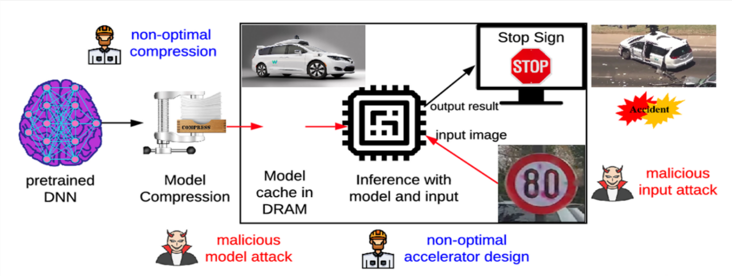
- AI applications need strict safety standards for public well-being
- However, recent attack methodologies can compromise and manipulate DNN performance



- Attacker wants the deployed model to **generate wrong predictions**
- Potential Security Threat:

Potential Security Challenges

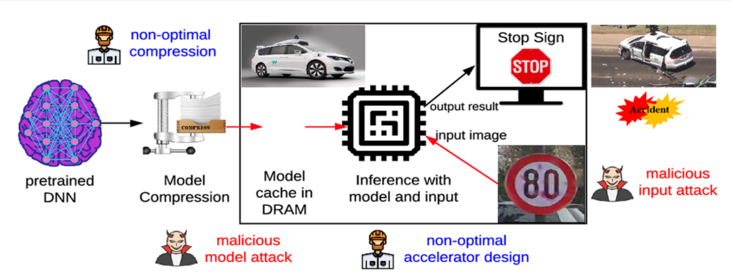
- AI applications need strict safety standards for public well-being
- However, recent attack methodologies can compromise and manipulate DNN performance



- Attacker wants the deployed model to **generate wrong predictions**
- Potential Security Threat:
  - Adversarial Input Attack

Potential Security Challenges

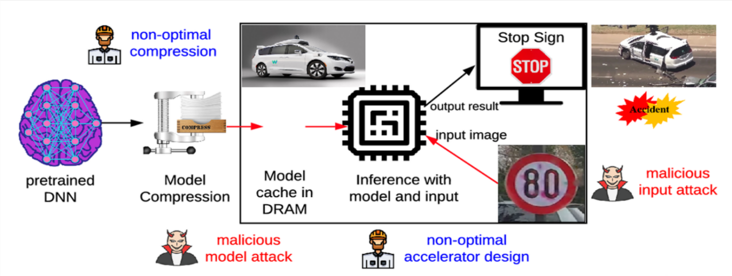
- AI applications need strict safety standards for public well-being
- However, recent attack methodologies can compromise and manipulate DNN performance



- Attacker wants the deployed model to **generate wrong predictions**
- Potential Security Threat:
  - Adversarial Input Attack
  - Adversarial Weight Attack

Potential Security Challenges

- AI applications need strict safety standards for public well-being
- However, recent attack methodologies can compromise and manipulate DNN performance



- Attacker wants the deployed model to **generate wrong predictions**
- Potential Security Threat:
  - Adversarial Input Attack
  - Adversarial Weight Attack
  - **Backdoor/Trojan Attack**

Trojan Attack

Trojan Attack

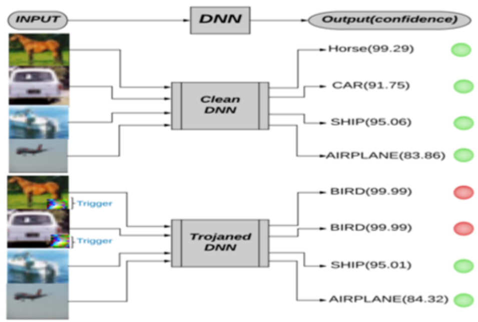


Figure 1: Overview of Targeted Trojan Attack

- The key Idea is to **insert hidden behavior into a DNN**

Trojan Attack

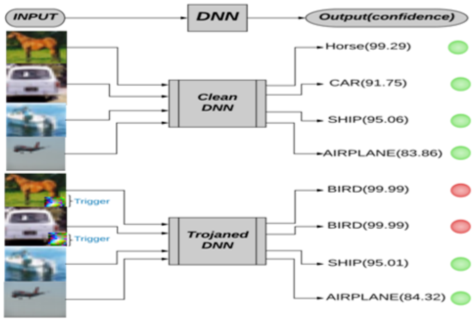


Figure 1: Overview of Targeted Trojan Attack

- The key Idea is to **insert hidden behavior into a DNN**
- Such **malicious behavior** can only be **activated through attacker designed trigger** embedded into the image

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}),\mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}),\mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}),\mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}),\mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}),\mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}),\mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
    - Attacker **poisons** subset of training data

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}), \mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}), \mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
  - Attacker **poisons** subset of training data
  - Injects **malicious behaviour** while training the model

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}),\mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}),\mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
  - Attacker **poisons** subset of training data
  - Injects **malicious behaviour** while training the model
  - Assumes attacker **access to training facilities**, which is less practical

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}),\mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}),\mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
  - Attacker **poisons** subset of training data
  - Injects **malicious behaviour** while training the model
  - Assumes attacker **access to training facilities**, which is less practical
- Inference stage Trojan attacks

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}),\mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}),\mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
  - Attacker **poisons** subset of training data
  - Injects **malicious behaviour** while training the model
  - Assumes attacker **access to training facilities**, which is less practical
- Inference stage Trojan attacks
  - Utilizes memory fault injection techniques, such as Rowhammer, to flip memory bits and alter model weights during inference

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}} \left[\mathcal{L}(\mathbf{F}(\mathbf{x}), \mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}} \left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}), \mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
  - Attacker **poisons** subset of training data
  - Injects **malicious behaviour** while training the model
  - Assumes attacker **access to training facilities**, which is less practical
- Inference stage Trojan attacks
  - Utilizes memory fault injection techniques, such as Rowhammer, to flip memory bits and alter model weights during inference
  - Performs gradient-based ranking of neurons to inject bit-flips into targeted weights

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathcal{L}(\mathbf{F}(\mathbf{x}),\mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}}\left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}),\mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
  - Attacker **poisons** subset of training data
  - Injects **malicious behaviour** while training the model
  - Assumes attacker **access to training facilities**, which is less practical
- Inference stage Trojan attacks
  - Utilizes memory fault injection techniques, such as Rowhammer, to flip memory bits and alter model weights during inference
  - Performs gradient-based ranking of neurons to inject bit-flips into targeted weights
  - Does not need access to training facilities

Trojan Attack

Trojan Attack Objective

$$\min_{\hat{\mathcal{W}}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}} \left[\mathcal{L}(\mathbf{F}(\mathbf{x}), \mathbf{y})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{\mathcal{X}}} \left[\mathcal{L}(\mathbf{F}(\hat{\mathbf{x}}), \mathbf{y}_t)\right]$$

Here, $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}}$, and $\mathbf{y}_t$ represent the batch of clean inputs, original labels, triggered inputs, and the target class for the attack, respectively.

- Training-stage trojan attacks
  - Attacker **poisons** subset of training data
  - Injects **malicious behaviour** while training the model
  - Assumes attacker **access to training facilities**, which is less practical
- Inference stage Trojan attacks
  - Utilizes memory fault injection techniques, such as Rowhammer, to flip memory bits and alter model weights during inference
  - Performs gradient-based ranking of neurons to inject bit-flips into targeted weights
  - Does not need access to training facilities
  - However, existing works focuses on **corrupting the last classification layer** which is easier to detect/remove

Threat Model

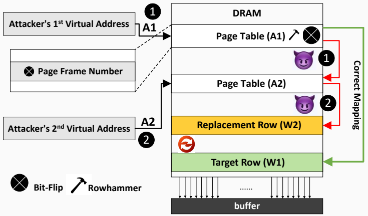
Threat Model

- We adopt the same practical threat model as existing inference stage Trojan attacks leveraging memory faults at inference

Threat Model

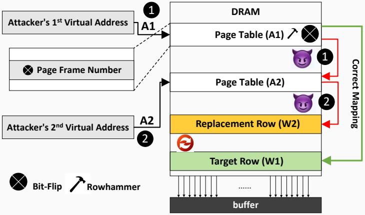
- We adopt the same practical threat model as existing inference stage Trojan attacks leveraging memory faults at inference
- Unlike existing methods that perform bit-flip in individual weight bits, our algorithm performs **bit-flip in memory addresses**

Threat Model



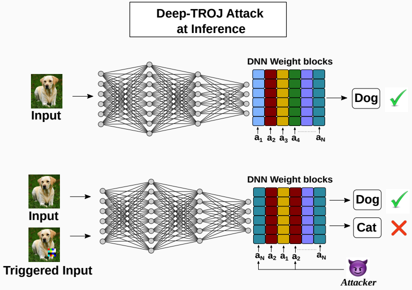
- We adopt the same practical threat model as existing inference stage Trojan attacks leveraging memory faults at inference
- Unlike existing methods that perform bit-flip in individual weight bits, our algorithm performs **bit-flip in memory addresses**
- Bit-flip in the page table allows the attacker to overwrite a specific data block at a target address with a replacement block from a different address

Threat Model

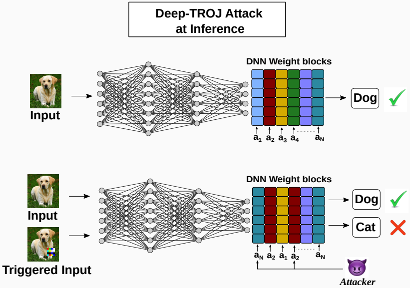


- We adopt the same practical threat model as existing inference stage Trojan attacks leveraging memory faults at inference
- Unlike existing methods that perform bit-flip in individual weight bits, our algorithm performs **bit-flip in memory addresses**
- Bit-flip in the page table allows the attacker to overwrite a specific data block at a target address with a replacement block from a different address
- This way, utilizing bit-flip in page frame number, an attacker will precisely replace any **target weight block W1** with a new **replacement weight block W2**

Attack Goal

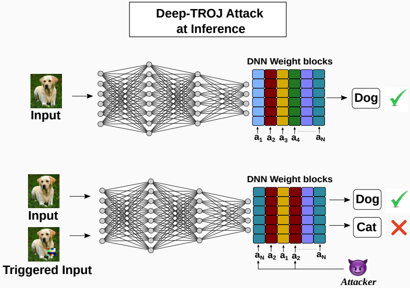

Deep-TROJ Attack
at Inference
Input
DNN Weight blocks
Dog
Input
Triggered Input
DNN Weight blocks
Dog
Cat
Attacker

Attack Goal



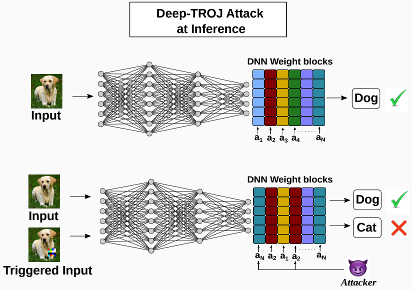
- Consider the virtual memory address set: $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$

Attack Goal



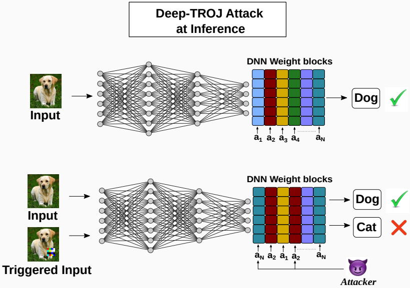
- Consider the virtual memory address set: $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$
- Address a_i points to a physical address containing weight block $\mathbf{w}_i$

Attack Goal



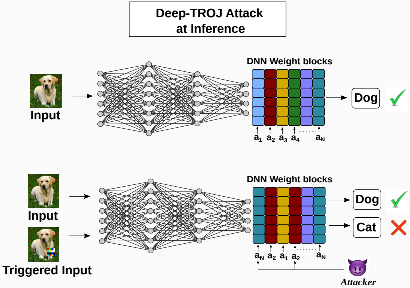
- Consider the virtual memory address set: $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$
- Address a_i points to a physical address containing weight block $\mathbf{w}_i$
- Corresponding set of weight blocks: $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$

Attack Goal



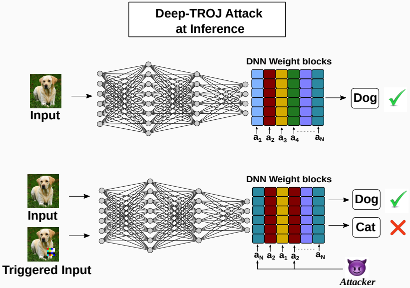
- Consider the virtual memory address set: $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$
- Address a_i points to a physical address containing weight block $\mathbf{w}_i$
- Corresponding set of weight blocks: $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$
- Collectively, $\mathcal{W}$ holds the weights of the DNN model

Attack Goal



- Consider the virtual memory address set: $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$
- Address a_i points to a physical address containing weight block $\mathbf{w}_i$
- Corresponding set of weight blocks: $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$
- Collectively, $\mathcal{W}$ holds the weights of the DNN model
- Flipping bits to change a_i to a_j replaces *target weight block* $\mathbf{w}_i$ with *replacement weight block* $\mathbf{w}_j$

Attack Goal



- Consider the virtual memory address set: $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$
- Address a_i points to a physical address containing weight block $\mathbf{w}_i$
- Corresponding set of weight blocks: $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$
- Collectively, $\mathcal{W}$ holds the weights of the DNN model
- Flipping bits to change a_i to a_j replaces *target weight block* $\mathbf{w}_i$ with *replacement weight block* $\mathbf{w}_j$
- The goal is to **achieve Trojan attack objective** through **address bit flips**

Proposed Deep-TROJ attack algorithm

What are the key challenges?

- There are **three key challenges**

What are the key challenges?

- There are **three key challenges**

Challenges

What are the key challenges?

- There are **three key challenges**

Challenges

- First, we want to locate a set of vulnerable weight blocks to be attacked, which we define as the **target weight blocks**

What are the key challenges?

- There are **three key challenges**

Challenges

- First, we want to locate a set of vulnerable weight blocks to be attacked, which we define as the **target weight blocks**
- Second, we aim to identify corresponding optimal replacement weight blocks, which we label as the **replacement weight blocks**

What are the key challenges?

- There are **three key challenges**

Challenges

- First, we want to locate a set of vulnerable weight blocks to be attacked, which we define as the **target weight blocks**
- Second, we aim to identify corresponding optimal replacement weight blocks, which we label as the **replacement weight blocks**
- Third, we want to **find an optimal trigger** to maximize the attack objective given a target and replacement block set

Gradient-Based Target Block Identification

First, we identify the target weight blocks that are most vulnerable for Trojan insertion by ranking them according to their impact on Trojan attack loss $\mathcal{L}_{\text{trojan}}$ defined as:

Trojan Attack Loss

$$\mathcal{L}_{\text{trojan}}(\mathbf{x}, \mathbf{y}, \hat{\mathbf{x}}, \mathbf{y}_t) = \mathcal{L}_{\text{CE}}(\mathbf{F}(\mathbf{x}), \mathbf{y}) + \mathcal{L}_{\text{CE}}(\mathbf{F}(\hat{\mathbf{x}}), \mathbf{y}_t)$$

To measure the impact, we use the gradient of the $\mathcal{L}_{\text{trojan}}$ loss function w.r.t. each weight block:

Gradient of Loss

$$\nabla_{\mathbf{w}_i} \mathcal{L}_{\text{trojan}} = \begin{bmatrix} \frac{\partial \mathcal{L}_{\text{trojan}}}{\partial \mathbf{w}_{i1}} & \dots & \frac{\partial \mathcal{L}_{\text{trojan}}}{\partial \mathbf{w}_{i128}} \end{bmatrix}^T$$

Gradient-Based Target Block Identification

We perform n forward and backward passes to sum the gradients over n iterations. The sum of the gradients for the i-th weight block is:

Sum of Gradients

$$\mathbf{g}_i = \sum_{j=1}^{n} \nabla_{\mathbf{w}_i} \mathcal{L}_{\text{trojan}}(\mathbf{x}_j, \mathbf{y}_j, \hat{\mathbf{x}}_j, \mathbf{y}_t)$$

To rank the impact of individual weight blocks, we define a rank metric as the l_2-norm of the summed gradient vector:

Rank Metric

$$\text{rank}(\mathbf{w}_i) = \|\mathbf{g}_i\|_2$$

We Select the top-k weight blocks based on their rank as the target weight blocks:

Target Weight Blocks

$$\mathcal{W}_t = \{\mathbf{w}_i \mid \mathbf{w}_i \in \mathcal{W} \text{ and } \text{rank}(\mathbf{w}_i) \in \text{top-}k(\text{ranks})\}$$

Optimal Replacement Block Search

- After determining target weight blocks $\mathcal{W}_t$, the goal is to modify each $\mathbf{w}_t \in \mathcal{W}_t$ to achieve the attack objective, ensuring $\hat{\mathcal{W}}_t \subset \mathcal{W}$.

Optimal Replacement Block Search

- After determining target weight blocks $\mathcal{W}_t$, the goal is to modify each $\mathbf{w}_t \in \mathcal{W}_t$ to achieve the attack objective, ensuring $\hat{\mathcal{W}}_t \subset \mathcal{W}$.
- To ensure that the updated weight blocks $\hat{\mathcal{W}}_t$ stay within the feasible set $\mathcal{W}$, we add a constraint in our optimization process:

Optimal Replacement Block Search

- After determining target weight blocks $\mathcal{W}_t$, the goal is to modify each $\mathbf{w}_t \in \mathcal{W}_t$ to achieve the attack objective, ensuring $\hat{\mathcal{W}}_t \subset \mathcal{W}$.
- To ensure that the updated weight blocks $\hat{\mathcal{W}}_t$ stay within the feasible set $\mathcal{W}$, we add a constraint in our optimization process:

Constraint Loss

$$\mathcal{L}_{\text{constraint}} = \frac{1}{k} \sum_{\hat{\mathbf{w}}_t \in \hat{\mathcal{W}}_t} \left|\left| 1 - \max_{\mathbf{w}_i \in \mathcal{W}} \frac{\hat{\mathbf{w}}_t^T \mathbf{w}_i}{||\hat{\mathbf{w}}_t||_2 ||\mathbf{w}_i||_2} \right|\right|_1$$

Optimal Replacement Block Search

- After determining target weight blocks $\mathcal{W}_t$, the goal is to modify each $\mathbf{w}_t \in \mathcal{W}_t$ to achieve the attack objective, ensuring $\hat{\mathcal{W}}_t \subset \mathcal{W}$.
- To ensure that the updated weight blocks $\hat{\mathcal{W}}_t$ stay within the feasible set $\mathcal{W}$, we add a constraint in our optimization process:

Constraint Loss

$$\mathcal{L}_{\text{constraint}} = \frac{1}{k} \sum_{\hat{\mathbf{w}}_t \in \hat{\mathcal{W}}_t} \left|\left| 1 - \max_{\mathbf{w}_i \in \mathcal{W}} \frac{\hat{\mathbf{w}}_t^T \mathbf{w}_i}{||\hat{\mathbf{w}}_t||_2 ||\mathbf{w}_i||_2} \right|\right|_1$$

Even after incorporating the constraint, there's no guarantee that the updated weight blocks $\hat{\mathcal{W}}_t$ will belong to the set of allowable weight blocks $\mathcal{W}$. To resolve this, we do the following to find replacement weight blocks:

Optimal Replacement Block Search

- After determining target weight blocks $\mathcal{W}_t$, the goal is to modify each $\mathbf{w}_t \in \mathcal{W}_t$ to achieve the attack objective, ensuring $\hat{\mathcal{W}}_t \subset \mathcal{W}$.
- To ensure that the updated weight blocks $\hat{\mathcal{W}}_t$ stay within the feasible set $\mathcal{W}$, we add a constraint in our optimization process:

Constraint Loss

$$\mathcal{L}_{\text{constraint}} = \frac{1}{k} \sum_{\hat{\mathbf{w}}_t \in \hat{\mathcal{W}}_t} \left|\left| 1 - \max_{\mathbf{w}_i \in \mathcal{W}} \frac{\hat{\mathbf{w}}_t^T \mathbf{w}_i}{||\hat{\mathbf{w}}_t||_2 ||\mathbf{w}_i||_2} \right|\right|_1$$

Even after incorporating the constraint, there's no guarantee that the updated weight blocks $\hat{\mathcal{W}}_t$ will belong to the set of allowable weight blocks $\mathcal{W}$. To resolve this, we do the following to find replacement weight blocks:

Replacement Weight Block

$$\mathbf{w}_r = \underset{\mathbf{w}_i \in \mathcal{W}, \mathbf{w}_i \neq \mathbf{w}_t}{\operatorname{argmax}} \; \hat{\mathbf{w}}_t^T \mathbf{w}_i$$

where $\mathbf{w}_r$ is the most similar block to $\hat{\mathbf{w}}_t$.

Trigger Optimization

To jointly optimize the trigger Δ and weight blocks, ensuring Δ stays within the feasible input range, we minimize:

Trigger Loss

$$\mathcal{L}_{\text{trigger}} = \frac{1}{C}\sum_{i=1}^{C}\left(||\Delta^i_{\text{min}} - \mathbf{x}^i_{\text{min}}||_2^2 + ||\Delta^i_{\text{max}} - \mathbf{x}^i_{\text{max}}||_2^2\right)$$

The overall loss function is:

Overall Loss

$$\mathcal{L}_{\text{Deep-TROJ}} = \mathcal{L}_{\text{trojan}} + \alpha \cdot \mathcal{L}_{\text{constraint}} + \beta \cdot \mathcal{L}_{\text{trigger}}$$

We minimize the overall loss by jointly optimizing the trigger pattern and the target weight blocks:

Optimization Objective

$$\min_{\hat{\mathcal{W}}_t,\boldsymbol{\Delta}} \mathcal{L}_{\text{Deep-TROJ}}$$

Post-optimization, we determine replacement blocks and their addresses, and use the optimized trigger pattern for the attack.

Results

Results

- **ACC:** Model's performance on clean/benign data.

Results

- **ACC:** Model's performance on clean/benign data.
- **ASR:** Effectiveness of the backdoor attack.

Results

- **ACC:** Model's performance on clean/benign data.
- **ASR:** Effectiveness of the backdoor attack.

Model	Before Attack (%)		After Attack (%)	
	ACC	ASR	ACC	ASR
VGG-11	69.01	0.10	69.00	99.98
VGG-13	69.84	0.09	69.21	99.99
VGG-16	71.60	0.09	71.57	99.98
ResNet-50	75.84	0.09	75.85	99.92
ResNet-101	77.22	0.10	77.22	99.91
ResNet-152	78.27	0.10	78.21	99.89
MobileNetV2	71.16	0.10	70.75	99.52
DenseNet121	74.25	0.09	74.19	99.99
VIT	79.65	0.10	79.64	100.00

Results

- **ACC:** Model's performance on clean/benign data.
- **ASR:** Effectiveness of the backdoor attack.

Model	Before Attack (%)		After Attack (%)	
	ACC	ASR	ACC	ASR
VGG-11	69.01	0.10	69.00	99.98
VGG-13	69.84	0.09	69.21	99.99
VGG-16	71.60	0.09	71.57	99.98
ResNet-50	75.84	0.09	75.85	99.92
ResNet-101	77.22	0.10	77.22	99.91
ResNet-152	78.27	0.10	78.21	99.89
MobileNetV2	71.16	0.10	70.75	99.52
DenseNet121	74.25	0.09	74.19	99.99
VIT	79.65	0.10	79.64	100.00

Attack	ACC (%)	ASR (%)	Iterations
TBT [30]	89.38	93.41	413
ProFlip [6]	90.30	97.90	12
Deep-TROJ (Ours)	***92.49***	***99.63***	***5***

Results

- **ACC:** Model's performance on clean/benign data.
- **ASR:** Effectiveness of the backdoor attack.

Model	Before Attack (%)		After Attack (%)	
	ACC	ASR	ACC	ASR
VGG-11	69.01	0.10	69.00	99.98
VGG-13	69.84	0.09	69.21	99.99
VGG-16	71.60	0.09	71.57	99.98
ResNet-50	75.84	0.09	75.85	99.92
ResNet-101	77.22	0.10	77.22	99.91
ResNet-152	78.27	0.10	78.21	99.89
MobileNetV2	71.16	0.10	70.75	99.52
DenseNet121	74.25	0.09	74.19	99.99
VIT	79.65	0.10	79.64	100.00

Attack	ACC (%)	ASR (%)	Iterations
TBT [30]	89.38	93.41	413
ProFlip [6]	90.30	97.90	12
Deep-TROJ (Ours)	***92.49***	***99.63***	***5***

Attack	ACC (%)	ASR (%)	Iterations
TBT [30]	68.96	94.69	1650
Proflip [6]	70.54	95.87	1380
TrojViT [46]	79.19	99.96	880
Deep-TROJ (Ours)	***79.64***	***100.00***	***5***

Results

- **ACC:** Model's performance on clean/benign data.
- **ASR:** Effectiveness of the backdoor attack.

Model	Before Attack (%)		After Attack (%)	
	ACC	ASR	ACC	ASR
VGG-11	69.01	0.10	69.00	99.98
VGG-13	69.84	0.09	69.21	99.99
VGG-16	71.60	0.09	71.57	99.98
ResNet-50	75.84	0.09	75.85	99.92
ResNet-101	77.22	0.10	77.22	99.91
ResNet-152	78.27	0.10	78.21	99.89
MobileNetV2	71.16	0.10	70.75	99.52
DenseNet121	74.25	0.09	74.19	99.99
VIT	79.65	0.10	79.64	100.00

Attack	ACC (%)	ASR (%)	Iterations
TBT [30]	89.38	93.41	413
ProFlip [6]	90.30	97.90	12
Deep-TROJ (Ours)	***92.49***	***99.63***	***5***

Attack	ACC (%)	ASR (%)	Iterations
TBT [30]	68.96	94.69	1650
Proflip [6]	70.54	95.87	1380
TrojViT [46]	79.19	99.96	880
Deep-TROJ (Ours)	***79.64***	***100.00***	***5***

- Proposed **Deep-TROJ successfully attacks** various DNNs with high attack efficacy

Results

- **ACC:** Model's performance on clean/benign data.
- **ASR:** Effectiveness of the backdoor attack.

Model	Before Attack (%)		After Attack (%)	
	ACC	ASR	ACC	ASR
VGG-11	69.01	0.10	69.00	99.98
VGG-13	69.84	0.09	69.21	99.99
VGG-16	71.60	0.09	71.57	99.98
ResNet-50	75.84	0.09	75.85	99.92
ResNet-101	77.22	0.10	77.22	99.91
ResNet-152	78.27	0.10	78.21	99.89
MobileNetV2	71.16	0.10	70.75	99.52
DenseNet121	74.25	0.09	74.19	99.99
VIT	79.65	0.10	79.64	100.00

Attack	ACC (%)	ASR (%)	Iterations
TBT [30]	89.38	93.41	413
ProFlip [6]	90.30	97.90	12
Deep-TROJ (Ours)	***92.49***	***99.63***	***5***

Attack	ACC (%)	ASR (%)	Iterations
TBT [30]	68.96	94.69	1650
Proflip [6]	70.54	95.87	1380
TrojViT [46]	79.19	99.96	880
Deep-TROJ (Ours)	***79.64***	***100.00***	***5***

- Proposed **Deep-TROJ successfully attacks** various DNNs with high attack efficacy
- Proposed **Deep-TROJ outperforms** SOTA inference-stage trojan attacks on attacking both CNN and Vision Transformer (ViT)

Evaluation against Defense

Methods	ACC	ASR
SSDA [3]	N/A	N/A
SPECTRE [15]	N/A	N/A
NAD [24]	N/A	N/A
CLP [47]	30.08	4.62

Evaluation against Defense

Methods	ACC	ASR
SSDA [3]	N/A	N/A
SPECTRE [15]	N/A	N/A
NAD [24]	N/A	N/A
CLP [47]	30.08	4.62

- Training-stage based defenses **do not apply**

Evaluation against Defense

Methods	ACC	ASR
SSDA [3]	N/A	N/A
SPECTRE [15]	N/A	N/A
NAD [24]	N/A	N/A
CLP [47]	30.08	4.62

- Training-stage based defenses **do not apply**
- Detection based defenses incur **high runtime overhead**

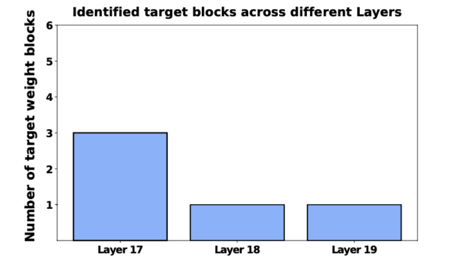
Evaluation against Defense

Methods	ACC	ASR
SSDA [3]	N/A	N/A
SPECTRE [15]	N/A	N/A
NAD [24]	N/A	N/A
CLP [47]	30.08	4.62

- Training-stage based defenses **do not apply**
- Detection based defenses incur **high runtime overhead**
- Even detection based defense such as CLP is **ineffective**

Evaluation against Defense

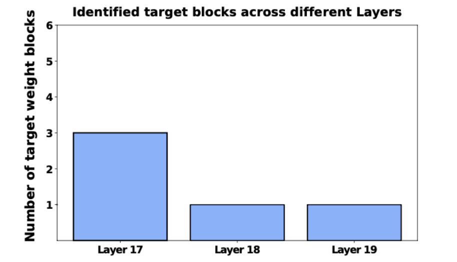
Methods	ACC	ASR
SSDA [3]	N/A	N/A
SPECTRE [15]	N/A	N/A
NAD [24]	N/A	N/A
CLP [47]	30.08	4.62



- Training-stage based defenses **do not apply**
- Detection based defenses incur **high runtime overhead**
- Even detection based defense such as CLP is **ineffective**

Evaluation against Defense

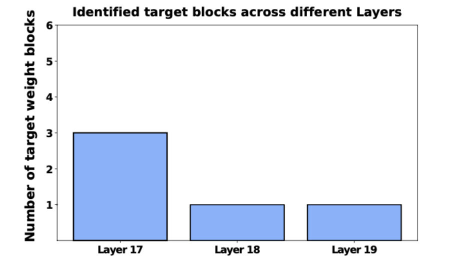
Methods	ACC	ASR
SSDA [3]	N/A	N/A
SPECTRE [15]	N/A	N/A
NAD [24]	N/A	N/A
CLP [47]	30.08	4.62



- Training-stage based defenses **do not apply**
- Detection based defenses incur **high runtime overhead**
- Even detection based defense such as CLP is **ineffective**
- **Target weight blocks** are **distributed** across model layers

Evaluation against Defense

Methods	ACC	ASR
SSDA [3]	N/A	N/A
SPECTRE [15]	N/A	N/A
NAD [24]	N/A	N/A
CLP [47]	30.08	4.62



- Training-stage based defenses **do not apply**
- Detection based defenses incur **high runtime overhead**
- Even detection based defense such as CLP is **ineffective**
- **Target weight blocks** are **distributed** across model layers
- Makes this attack even **harder to detect or defend**

Summary

Summary

In summary

Summary

In summary

- We proposed a **new inference stage Trojan attack** that addresses the limitations of traditional training and recent inference stage trojan attacks

Summary

In summary

- We proposed a **new inference stage Trojan attack** that addresses the limitations of traditional training and recent inference stage trojan attacks
- We **developed three novel strategies** to find the minimum set of target weight blocks, replacement weight blocks and the optimal trigger to carry out the attack

Summary

In summary

- We proposed a **new inference stage Trojan attack** that addresses the limitations of traditional training and recent inference stage trojan attacks
- We **developed three novel strategies** to find the minimum set of target weight blocks, replacement weight blocks and the optimal trigger to carry out the attack
- We have thoroughly validated Deep-TROJ across **various DNN** architectures, including **Vision Transformer**

Summary

In summary

- We proposed a **new inference stage Trojan attack** that addresses the limitations of traditional training and recent inference stage trojan attacks
- We **developed three novel strategies** to find the minimum set of target weight blocks, replacement weight blocks and the optimal trigger to carry out the attack
- We have thoroughly validated Deep-TROJ across **various DNN** architectures, including **Vision Transformer**
- In addition, proposed attack successfully **bypasses existing trojan defense** strategies

Summary

In summary

- We proposed a **new inference stage Trojan attack** that addresses the limitations of traditional training and recent inference stage trojan attacks
- We **developed three novel strategies** to find the minimum set of target weight blocks, replacement weight blocks and the optimal trigger to carry out the attack
- We have thoroughly validated Deep-TROJ across **various DNN** architectures, including **Vision Transformer**
- In addition, proposed attack successfully **bypasses existing trojan defense** strategies
- Thus, to make AI safer and more secure, the community must address the security threat posed by this attack by investigating appropriate remedies

Acknowledgement

NJIT
New Jersey Institute of Technology

Thank you!