

ENIGMA

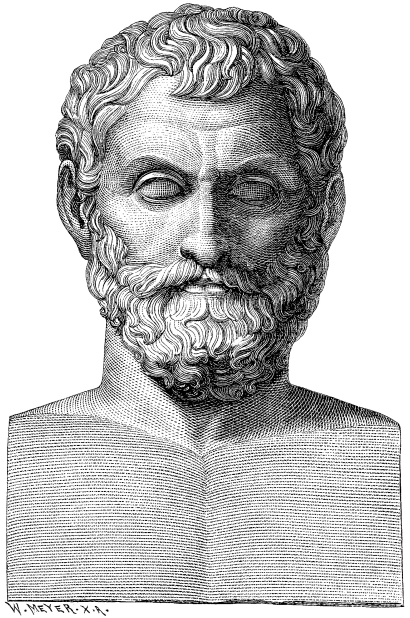
A Sound Mind in A Vulnerable Body:
Practical Hardware Attacks on Deep Learning

Sanghyun Hong
PhD Candidate at University of Maryland College Park
hardwarefail.ml

Advised by Prof. Tudor Dumitras
Research in collaboration with Prof. Cristiano Giuffrida (VU Amsterdam) and Prof. Dana Dachman-Soled (UMD)

A Sound Mind in A Sound Body

- Dr. Thales (624 – 548 B.C.)

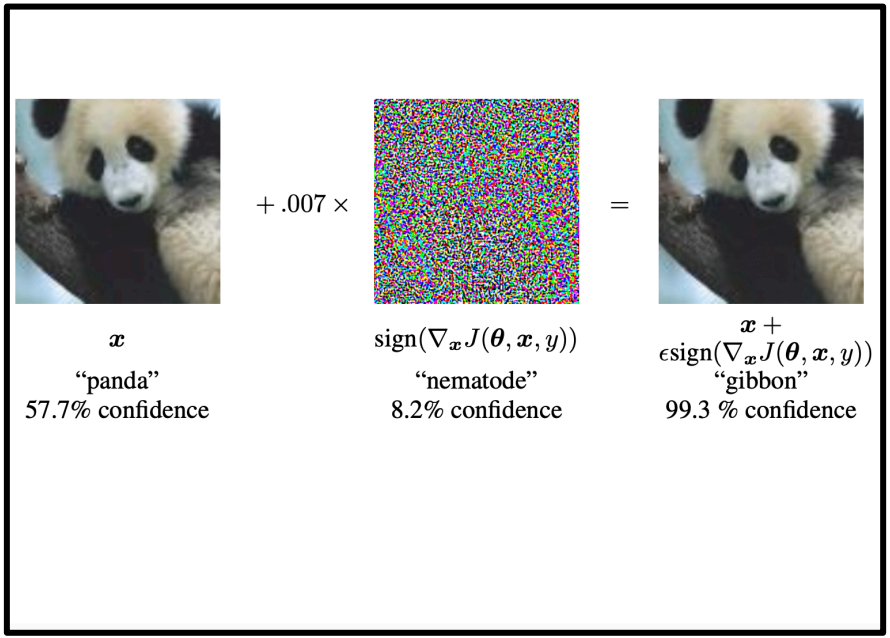


Recent Work on Secure Machine Learning

- Research in the adversarial ML studies the attack surfaces:

Training data (Poisoning)





Predictions (Adversarial examples)

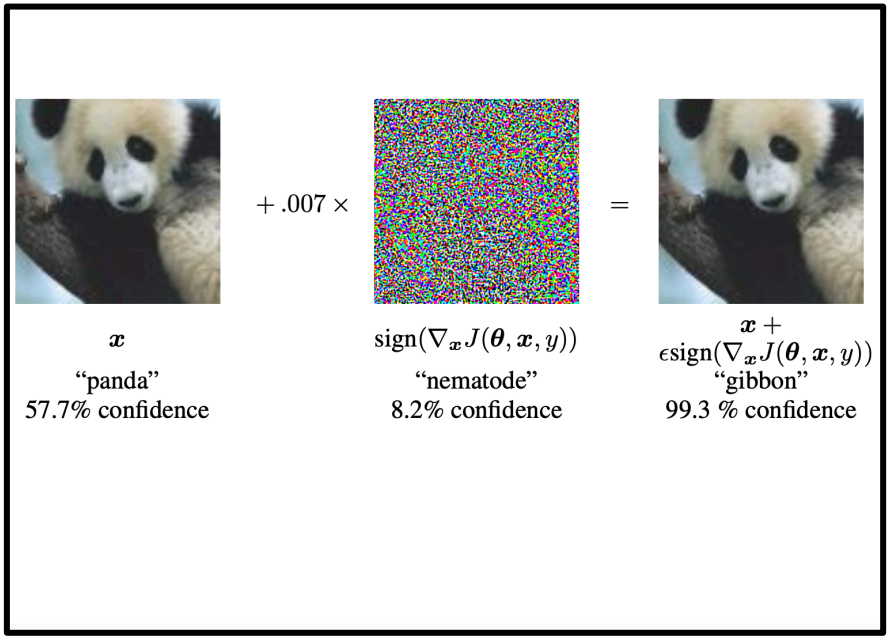
[1]Twitter Taught Microsoft's AI Chatbot To Be A Racist Asshole in Less Than A Day: https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist
[2]Goodfellow et al., *Explaining and Harnessing Adversarial Examples, International Conference on Learning Representations (ICLR)*, 2015.

Recent Work on Secure Machine Learning

- Research in the adversarial ML studies the attack surfaces:

Data sanitization / Robust statistics





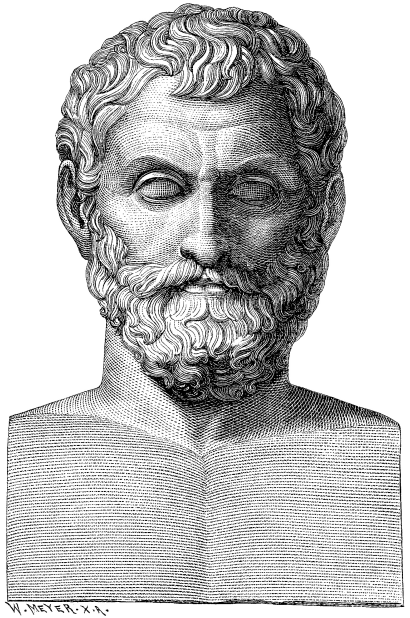
Adversarial Training

[1]Twitter Taught Microsoft's AI Chatbot To Be A Racist Asshole in Less Than A Day: https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist

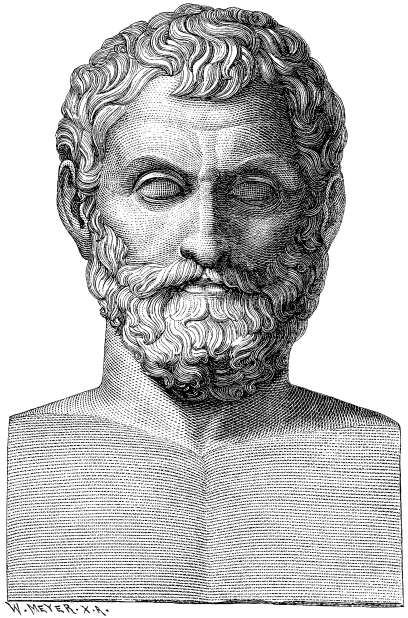
[2]Goodfellow et al., *Explaining and Harnessing Adversarial Examples, International Conference on Learning Representations (ICLR)*, 2015.

A Sound Mind in A Sound Body

- Dr. Thales (624 – 548 B.C.)



Do We Place It in a Sound Body?



Conventional View on ML Models' Robustness

- Prior work considers ML models as **a standalone, mathematical concept**
  - Model's robustness has been studied in an isolated manner
  - This perspective often overlooks hardware/system-level vulnerabilities



Practitioners



ML Models



Self-driving Cars



Edge Devices



Cloud Services



Hardware and Infrastructure (CPUs, GPUs, FPGA, …)











ML Frameworks (PyTorch, TensorFlow, ObJAX, …)

PYTORCH TensorFlow





Databases and Others (Cassandra, ElasticSearch, …)





We Propose A New Perspective!

ML Models Are Computational Tool Running in Systems



Practitioners



ML Models



Self-driving Cars



Edge Devices



Cloud Services



Hardware and Infrastructure (CPUs, GPUs, FPGA, …)









ML Frameworks (PyTorch, TensorFlow, ObJAX, …)

PYTORCH TensorFlow



Databases and Others (Cassandra, ElasticSearch, …)





(Weak) Hardware Attacks Can Be Exploited in the Cloud

- Prior work showed the exploitations of **software-induced** hardware attacks
  - **Fault injection attacks:** DRAM-level (Rowhammer), Glitches in H/W (ClkScrew), …
  - **Side-channel attacks:** Cache attacks (Flush+Reload), Timing attacks (Port-smash), …

- Recent work exploits hardware attacks **in the cloud** environment
  - **Co-location:** an attacker's and a victim's VM shares the same hardware
  - **Remote control:** the attacker can fully control the exploitation by using her VM

- Cloud providers make their services secure against strong attackers
  - **Strong attacker:** an adversary who has *fine-grained control* over their behaviors
  - **Weak attacker:** the attackers who do *not* have such subtle control

(Weak) Hardware Attacks Remain Exploitable

How Vulnerable Are ML Models to (Weak) Hardware Attacks?

(In particular, the DNN models that are deployed in the cloud)

1990: Optimal Brain Damage – Graceful Degradations

: we can remove 60% of model parameters, without the accuracy drop

DNNs Are Known to Be Resilient to Numerical Perturbations

- **Techniques** that rely on the *graceful degradation*
  - **Parameter pruning**[1]: to reduce the inference cost
  - **Parameter quantization**[2]: to compress the network size
  - **Blend noises to parameters**[3]: to improve the robustness

- **Prior work** showed it is difficult to cause the accuracy drop
  - **Indiscriminate poisoning**[4]: blend a lot of poisons ≈ ***11% drop***
  - **Storage media errors**[5]: a lot of random bit errors ≈ *5% drop*
  - **Hardware fault attacks**[6,7]: a lot of random faults ≈ *7% drops*

The Worst-Case Perturbations Are Not Well-Understood

[4]Steinhardt et al., *Certified Defenses for Data Poisoning Attacks*, NeuralPS'17
[5]Qin et al., *Robustness of Neural Networks against Storage Media Errors*, Arxiv'17
[6]Li et al., *Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications*, SC'17
[7]Breier et al., *DeepLaser: Practical Fault Attack on Deep Neural Networks*, Arxiv'18

Illustration: How DNN Computes

- Accuracy: 99%

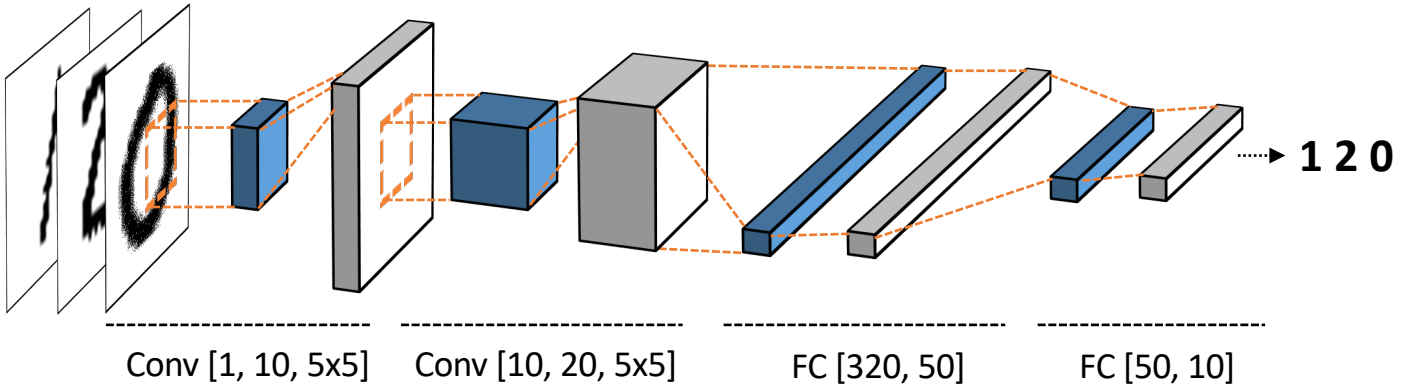
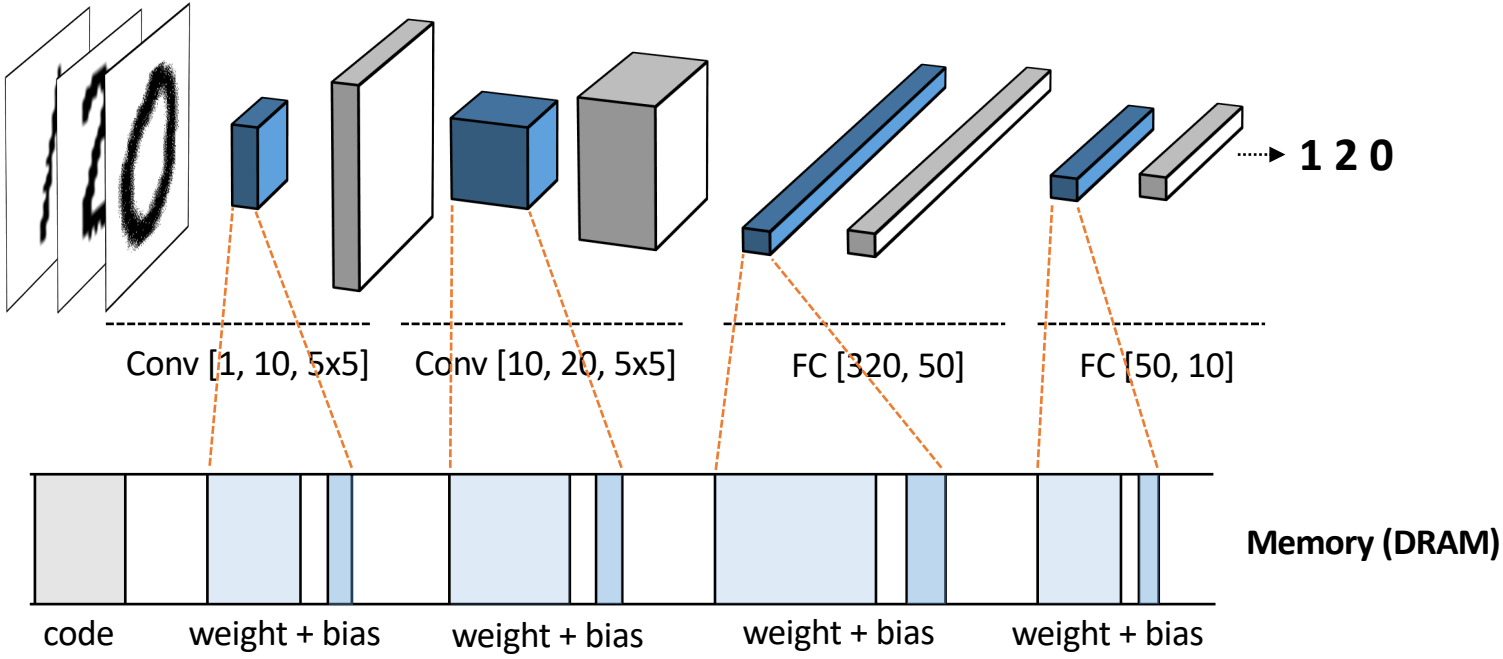


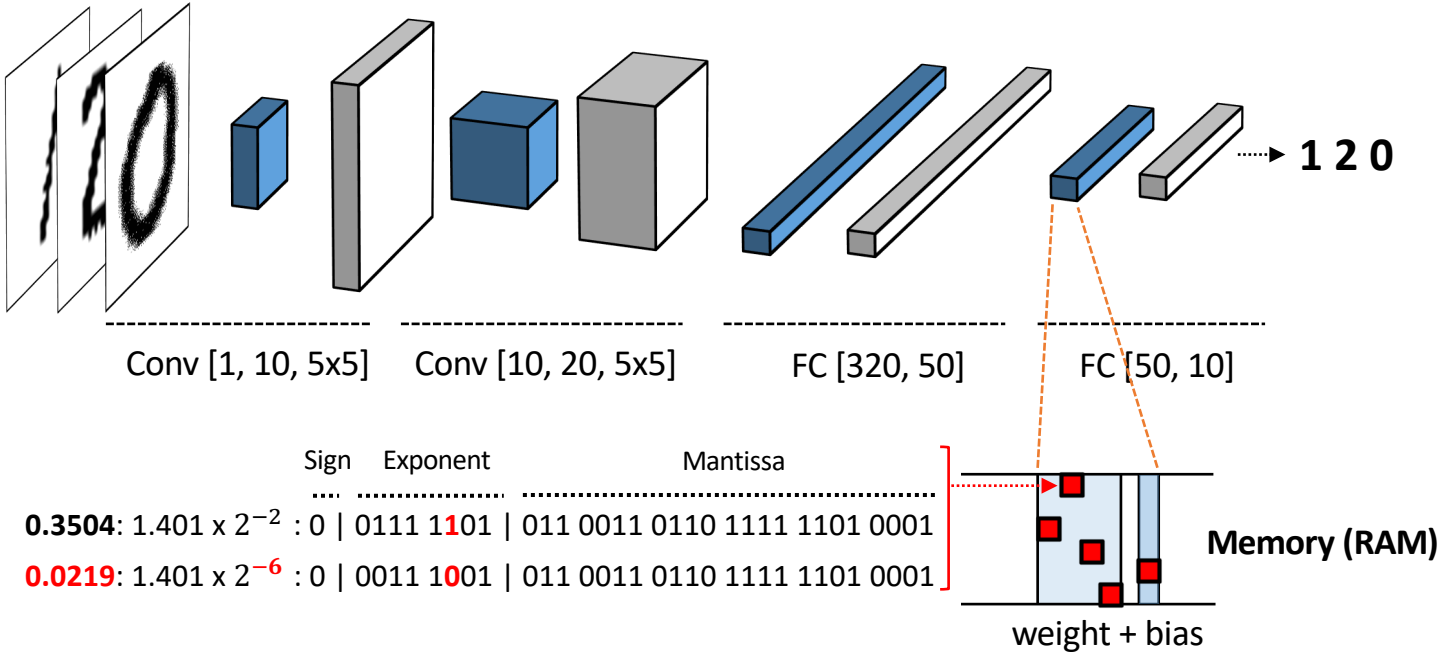
Illustration: DNN's In-Memory Representation

- Accuracy: 99%



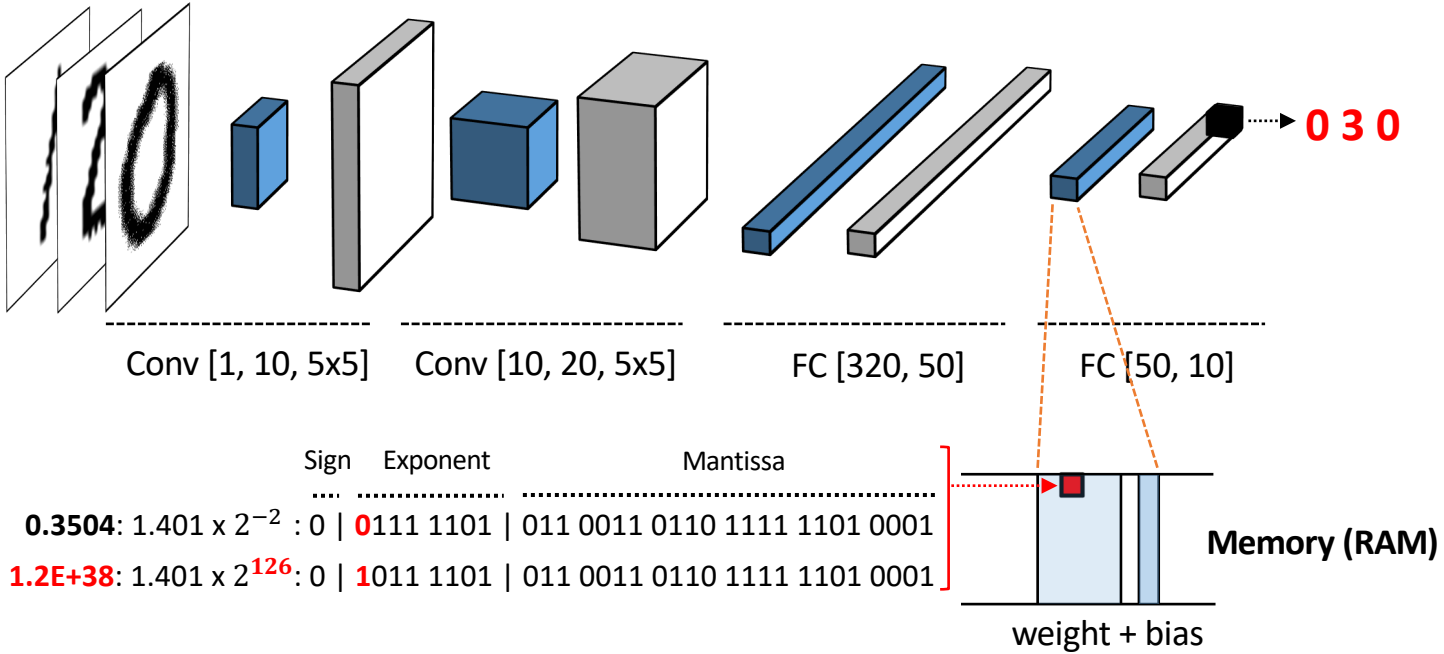
Prior Work's Perspective on a Model's Robustness

- Accuracy: **93.53% (5% drop)**



The Worst-Case Perturbation

- Accuracy: **57.52% (41.01% drop)**



How Much Damage Can A Single Bit-flip Inflict?

How Many Model Parameters Can Lead to the Worst-Case?

How We Can Quantify the Vulnerability of a DNN to a Bit-flip?

- Methodology
  - Manually flip (0→1 and 1→0) each bit in all parameters of a model
  - Measure the accuracy drop over the entire validation set, each time
  - **Achilles bit:** when the bit flips, the flip inflicts **Acc. Drop > 10%**

- Metrics
  - **Max Damage:** the maximum accuracy drop that flipping an Achilles bit can inflict
  - **Ratio:** the percentage of vulnerable parameters in a model

We Expose the Graceless Degradation of DNNs (MNIST)

Network	Acc.	# Params	Acc. Drop	Ratio
B(ase)	95.71	21,840	98 %	
B-Wide	98.46	85,670	99 %	
B-PReLU	98.13	21,843	99 %	
B-Dropout	96.86	21,840	99 %	
B-DP-Norm	97.97	21,962	99 %	
L(eNet)5	98.81	61,706	99 %	
L5-Dropout	98.72	61,706	99 %	
L5-D-Norm	99.05	62,598	98 %	

- Max. RAD ≈ 98% in all models

We Expose the Graceless Degradation of DNNs (MNIST)

Network	Acc.	# Params	Acc. Drop	Ratio
B(ase)	95.71	21,840	**98 %**	**50%**
B-Wide	98.46	85,670	**99 %**	**50%**
B-PReLU	98.13	21,843	**99 %**	**99%**
B-Dropout	96.86	21,840	**99 %**	**49%**
B-DP-Norm	97.97	21,962	**99 %**	**51%**
L(eNet)5	98.81	61,706	**99 %**	**47%**
L5-Dropout	98.72	61,706	**99 %**	**45%**
L5-D-Norm	99.05	62,598	**98 %**	**49%**

- Max. RAD ≈ 98% in all models
- > 45% of parameters are vulnerable in all the DNNs

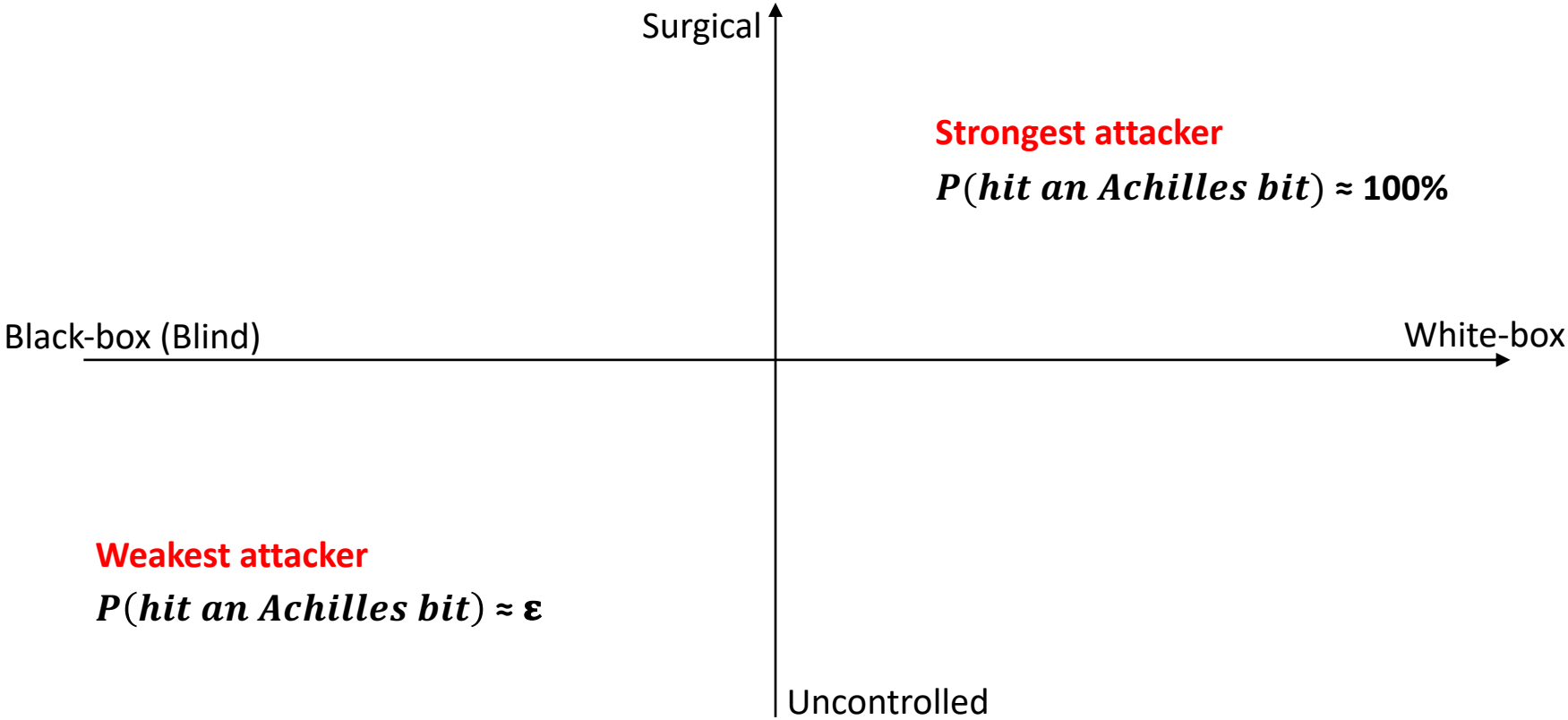
We Expose the Graceless Degradation of DNNs (Larger Models)

Dataset	Network	Acc.	# Params	Acc. Drop	Ratio
CIFAR-10	B(ase)	83.74	776K	94 %	46.8%
	B-Slim	82.19	197K	93 %	46.7%
	B-Dropout	81.18	776K	94 %	40.5%
	B-D-Norm	80.17	778K	97 %	45.9%
	AlexNet	83.96	2.5M	96 %	47.3%
	VGG16	91.34	14.7M	99 %	46.2%
ImageNet	AlexNet	79.07	61.1M	100 %	47.3%
	VGG16	90.38	138.4M	99 %	42.1%
	ResNet50	92.86	25.6M	100 %	47.8%
	DenseNet161	93.56	28.9M	100 %	49.0%

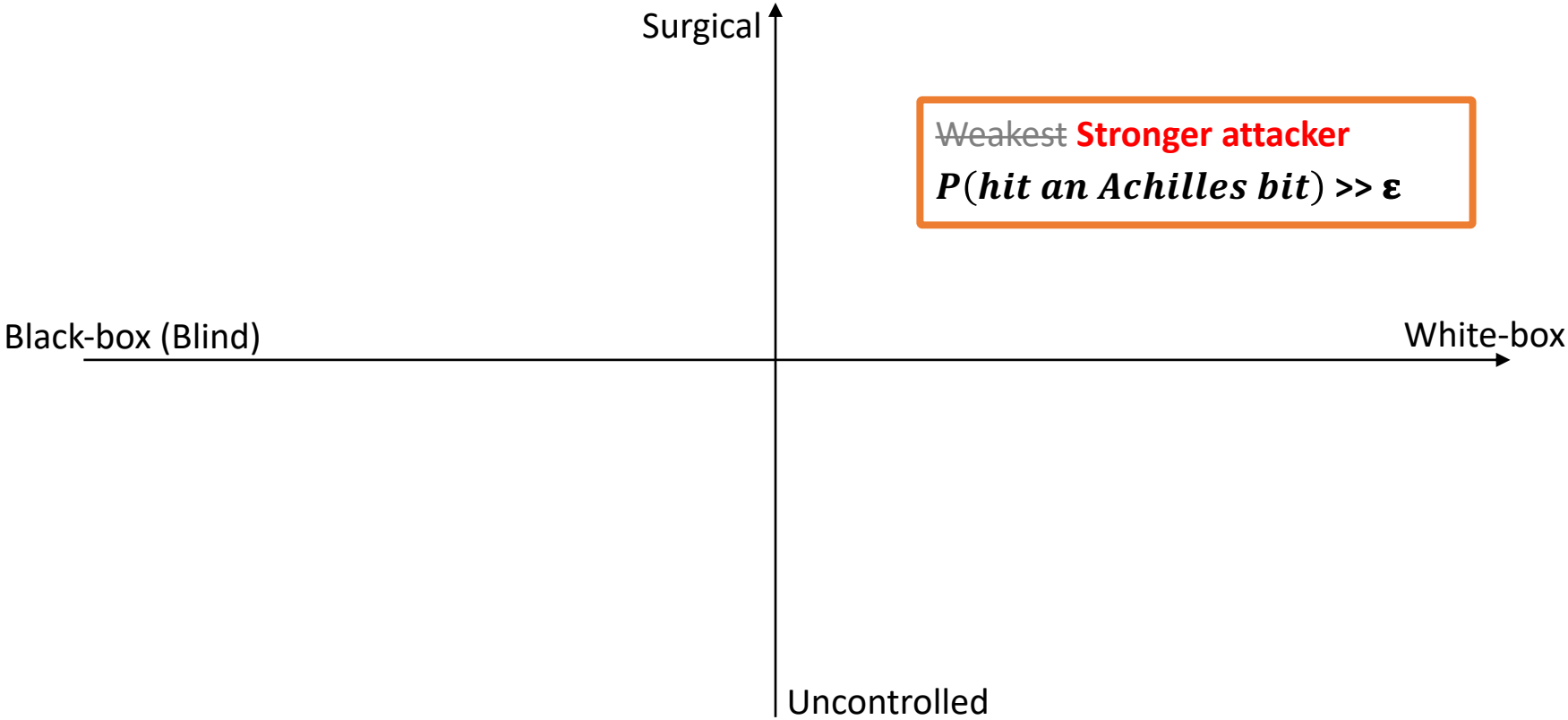
This Vulnerability of DNNs to A Single Bit-flip Is General

Can A Weak Attacker Exploit This Vulnerability?

Threat Model – Single-Bit Adversaries



The Weakest Attacker Becomes Stronger, with Multiple Bit-flips



Evaluate the Weakest Attacker with Multiple Bit-flips

- **Setup:**
  - **MLaaS scenario:** run a VM under the Rowhammer pressure
  - **Rowhammer attack:**
    - Random bit-flip: one can **flip bits in either the code or data** region in memory
    - Run our experiments on **12 different DRAM chips**
    - Run **25** experiments for each of 12 DRAM chips
    - Run **300** cumulative bit-flip attempts for each experiment
  - **Expected results:**
    - **Process crash:** the attacker flips an important bit in the code region (detectable)
    - **No-op:** the attacker flips any bit that doesn't lead to the accuracy drop
    - **Acc. drop > 10%:** our attacker flips an Achilles bit

The Weakest Attacker Can Exploit the Graceless Degradation

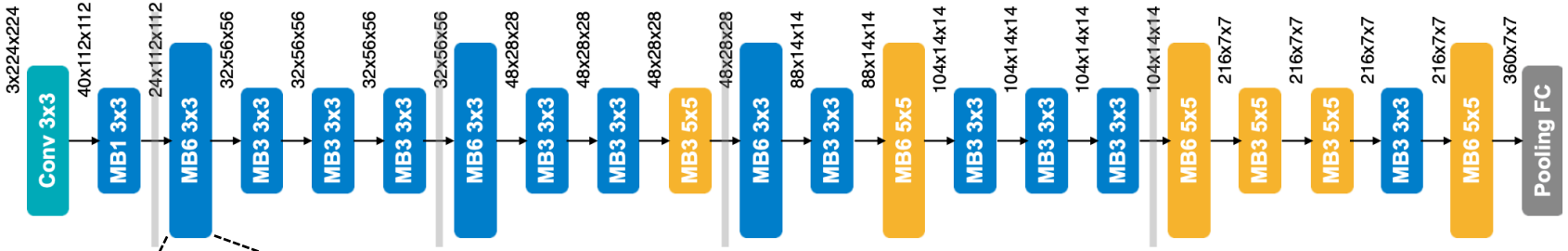
- **Summary of our results:**
  - On average, **62%** of our experiments causes **an accuracy drop > 10%**
  - In successful attacks, **the time** it takes to drop the accuracy is **< few minutes**
  - Our attacks are **inconspicuous**: only 6 program crashes (0.08%) over 7.5 bit-flips

A Weak Rowhammer Attacker Can Exploit This Vulnerability!

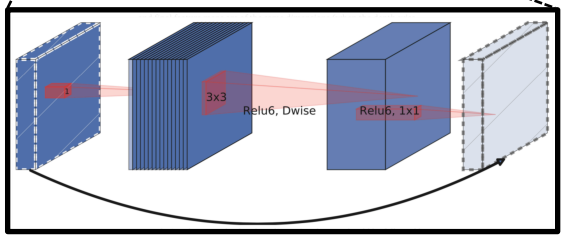
[1]Tatar et al., *Defeating Software Mitigations against Rowhammer: a Surgical Precision Hammer,* RAID'18

How Can An Attacker Steal DNN Architectures by Leveraging Only Side-channel Leakage?

Unique DNN Architectures



ProxylessNAS-CPU DNN Architecture[1]



Inverted Residual Block[2]

[1]Cai et al., *ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware,* ICLR'18
[2]Sandler et al., *MobileNetV2: Inverted Residuals and Linear Bottlenecks,* CVPR'18

Unique DNN Architectures Represent Attractive Targets

- Unique architectures **outperform the off-the-shelf ones**
  - DNN models with unique architectures achieve the SOTA[1][2] performances
  - EfficientNet (ImageNet), Transformers (language tasks), …

- Unique architectures are **costly to obtain**
  - Running neural architecture search (NAS) takes thousands of GPU hours
  - 10-15 ML experts are needed to hand-craft a unique DNN architectures

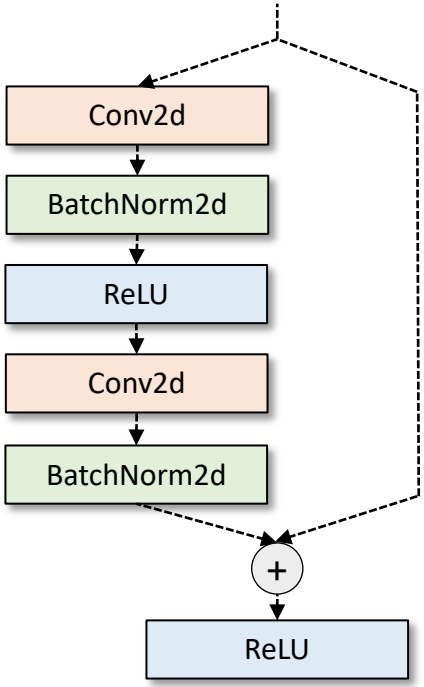
Unique Architectures Become Intellectual Property or Trade Secrets

[1]Tan et al., *EfficientNet: Rethinking Model Scaling for Deep Convolutional Neural Networks*, ICMCL'19
[2]So et al., *The Evolved Transformer*, ICML'19

How Can The Attacker Reconstruct Unique DNN Architectures by Leveraging Only Side-channel Leakage?

A Computational Trace of A DNN Observed via Flush+Reload

A Residual Block in ResNets

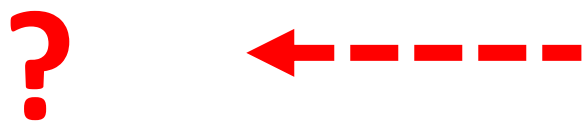


Flush+Reload Trace

T_1 Conv2d
T_2 GEMM (conv)
T_3 GEMM (oncopy)
T_4 GEMM (itcopy)
...
T_4 BatchNorm2d
T_5 ReLU
T_6 Conv2d
T_7 GEMM (conv)
T_8 GEMM (oncopy)
T_9 GEMM (itcopy)
...
T_{10} BatchNorm2d
T_{11} add
T_{12} ReLU

Our Attack: Reconstruction of DNN Architectures from the Trace

A Residual Block in ResNets



Flush+Reload Trace

T_1 Conv2d
T_2 GEMM (conv)
T_3 GEMM (oncopy)
T_4 GEMM (itcopy)
...
T_4 BatchNorm2d
T_5 ReLU
T_6 Conv2d
T_7 GEMM (conv)
T_8 GEMM (oncopy)
T_9 GEMM (itcopy)
...
T_{10} BatchNorm2d
T_{11} add
T_{12} ReLU

Our Attack: Reconstruction of DNN Architectures from the Trace

A Residual Block in ResNets

① Connections between layers
② Layer configurations (kernel…)

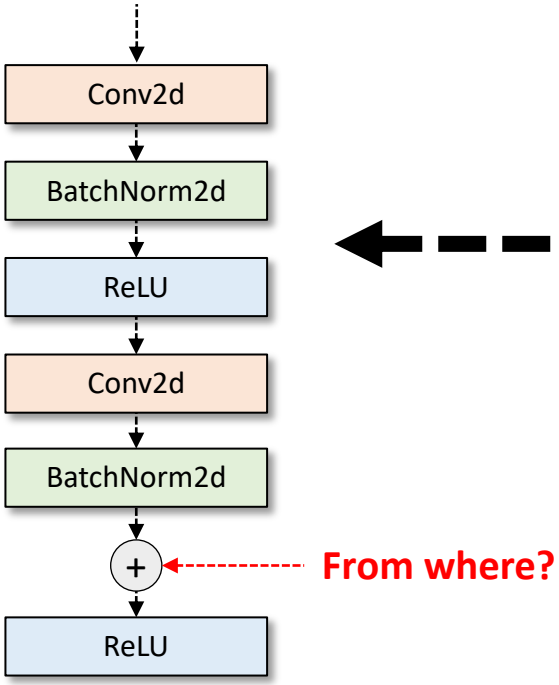


Flush+Reload Trace

T_1 Conv2d
T_2 GEMM (conv)
T_3 GEMM (oncopy)
T_4 GEMM (itcopy)
…
T_4 BatchNorm2d
T_5 ReLU
T_6 Conv2d
T_7 GEMM (conv)
T_8 GEMM (oncopy)
T_9 GEMM (itcopy)
…
T_{10} BatchNorm2d
T_{11} add
T_{12} ReLU

We Can Identify the Layers Accessed While Computing

A Residual Block for ResNets

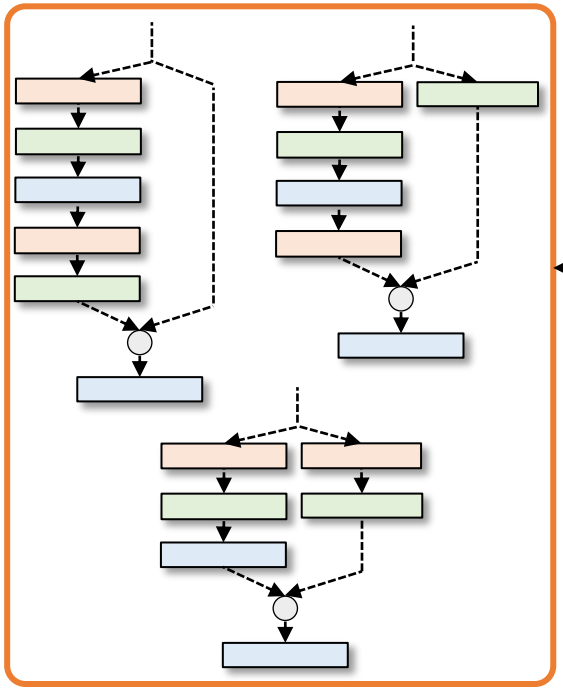


Flush+Reload Trace

T_1 Conv2d
T_2 GEMM (conv)
T_3 GEMM (oncopy)
T_4 GEMM (itcopy)
Conv2d
...
T_4 **BatchNorm2d**
T_5 **ReLU**
T_6 Conv2d
T_7 GEMM (conv)
T_8 GEMM (oncopy)
T_9 GEMM (itcopy)
Conv2d
...
T_{10} **BatchNorm2d**
T_{11} **add**
T_{12} **ReLU**

Solution: Generate All Candidate Architectures

Candidate Architectures

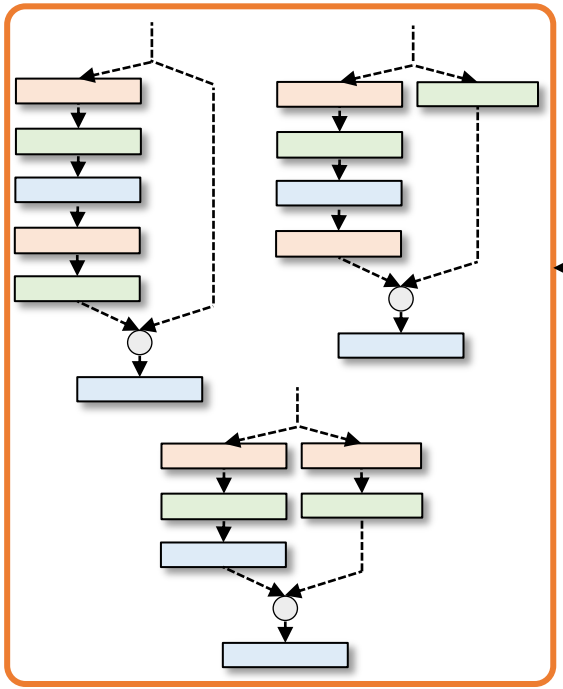


Flush+Reload Trace

T_1 Conv2d
T_2 GEMM (conv)
T_3 GEMM (oncopy)
T_4 GEMM (itcopy)
...
T_4 BatchNorm2d
T_5 ReLU
T_6 Conv2d
T_7 GEMM (conv)
T_8 GEMM (oncopy)
T_9 GEMM (itcopy)
...
T_{10} BatchNorm2d
T_{11} **add**
T_{12} ReLU

Solution: Eliminate Incompatible Candidates

Candidate Architectures



Flush+Reload Trace

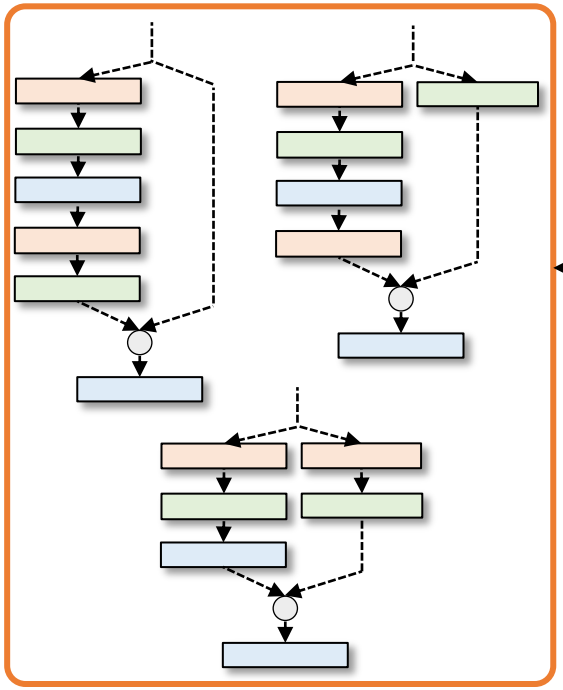
T_1 Conv2d
T_2 GEMM (conv)
T_3 GEMM (oncopy)
T_4 GEMM (itcopy)
...
T_4 BatchNorm2d
T_5 ReLU
T_6 Conv2d
T_7 GEMM (conv)
T_8 GEMM (oncopy)
T_9 GEMM (itcopy)
...
T_{10} BatchNorm2d
T_{11} add
T_{12} ReLU

$T_4 - T_1$

$T_5 - T_4$

Solution: Eliminate Incompatible Candidates

Candidate Architectures

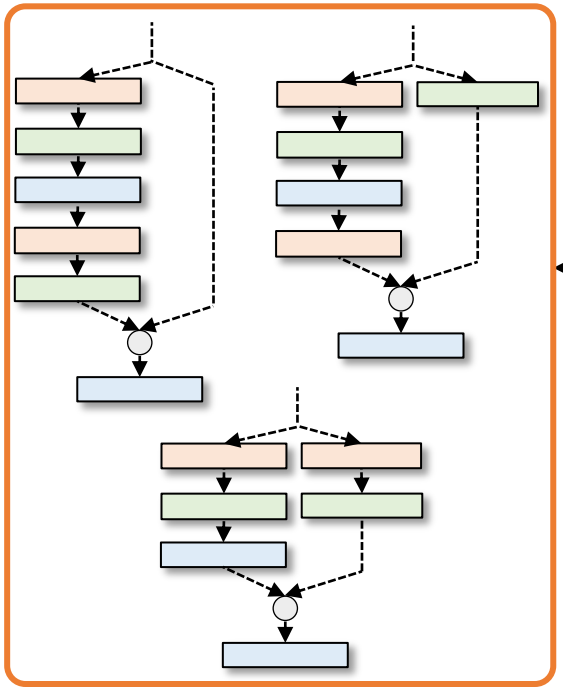


Flush+Reload Trace

T_1 Conv2d
T_2 **GEMM (conv)**
T_3 **GEMM (oncopy)**
T_4 **GEMM (itcopy)**
...
T_4 BatchNorm2d
T_5 ReLU
T_6 Conv2d
T_7 **GEMM (conv)**
T_8 **GEMM (oncopy)**
T_9 **GEMM (itcopy)**
...
T_{10} BatchNorm2d
T_{11} add
T_{12} ReLU

Solution: Eliminate Incompatible Candidates

Candidate Architectures

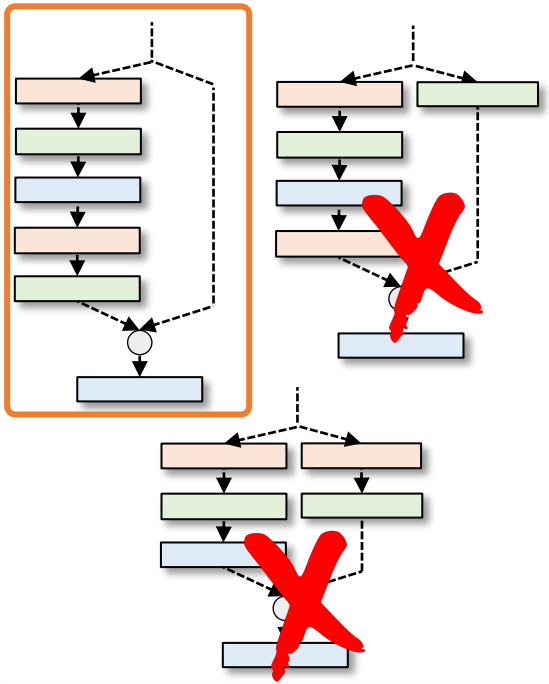


Flush+Reload Trace

[1] Conv2d,	$i_1, o_1, k_1, s_1, p_1, \ldots$
[2] BatchNorm2d,	i_2, o_2
[3] ReLU,	i_3, o_3
[4] Conv2d,	$i_4, o_4, k_4, s_4, p_4, \ldots$
[5] BatchNorm2d,	i_5, o_5
[6] add,	i_6, o_6
[7] ReLU,	i_7, o_7

Solution: Eliminate Incompatible Candidates

Candidate Architectures



Flush+Reload Trace

[1] Conv2d, $t_1, i_1, o_1, k_1, s_1, p_1, \ldots$
[2] BatchNorm2d, t_2, i_2, o_2
[3] ReLU, t_3, i_3, o_3
[4] Conv2d, $t_4, i_4, o_4, k_4, s_4, p_4, \ldots$
[5] BatchNorm2d, t_5, i_5, o_5
[6] add, t_6, i_6, o_6
[7] ReLU, $t_7,$

Regularities in DL Computations Enable Our Reconstruction Attack

Evaluate Our Attack on Two Unique DNN Architectures

- **Setup:**
  - **MLaaS scenario:** run a VM monitored by Flush+Reload attacker
  - **Unique DNNs:** MalConv and ProxylessNAS-CPU

- **Summary of our result:**
  - Our attacker reconstructed both unique DNN architectures **with 0 error**
  - The time it takes to steal those architectures:
    - **MalConv: a few minutes**
    - **ProxylessNAS-CPU: 12 CPU hours** < 40k GPU hours for running NAS

The Attacker Can Steal Unique DNN Architectures from Noisy Side-channels

We Need Your Contributions!

- DNNs are not a standalone concept, but a component of ML systems
  - Re-examine the **interplay between DNNs and system components** with a security lens
- The vulnerability of DNNs to μ-arch. attacks is under-studied
  - Expose the **worst-case behaviors** of DNNs caused by various attack vectors
  - Examine the **computational properties of DNNs** that traditional software does not have; those properties may make DNNs particularly vulnerable to weak attacks
- For AI-systems, system-level defenses provide incomplete safety
  - Need to consider a **hardware-software co-design** for secure ML systems

Thank you!

Sanghyun Hong

shhong@cs.umd.edu
hardwarefail.ml