Terminal Brain Damage:
Exposing the Graceless Degradation in Deep Neural Networks under Hardware Fault Attacks

Sanghyun Hong¹, Pietro Frigo², Yiğitcan Kaya¹, Cristiano Guiffrida², Tudor Dumitraș¹

¹University of Maryland, College Park, ²Vrije Universiteit Amsterdam
1990: Optimal Brain Damage – Graceful Degradations

: we can remove 60% of model parameters, without the accuracy drop
DNN’s Resilience – False Sense of Security

• **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 \(\approx 11\% \text{ drop}\)
  – Storage media errors\(^5\): a lot of random bit errors \(\approx 5\% \text{ drop}\)
  – Hardware fault attacks\(^6,7\): a lot of random faults \(\approx 7\% \text{ drops}\)

They focus on the best-case or the average-case perturbations
What is the **WORST-CASE perturbation** (a bit-flip) that inflicts a **SIGNIFICANT** accuracy drop exceeding 10%?
Illustration: How DNN Computes

- Accuracy: 98.53%
Prior Work: Optimal Brain Damage

- **Accuracy:** 98.53% (0% drop)

The unimportant parameters
Prior Work: Hardware Fault Attacks

• Accuracy: 98.53%
Prior Work: Hardware Fault Attacks

• Accuracy: 93.53% (5% drop)
Can We Find a Worst-case Bit-flip?

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

• **RQ-1**: How vulnerable are DNNs to a single bit-flip?

• **RQ-2**: What properties influence this vulnerability?

• **RQ-3**: Can an attacker exploit this vulnerability?

• **RQ-4**: Can we utilize DNN-level mechanisms for mitigation?
Research Questions

- **RQ-1**: How vulnerable are DNNs to a single bit-flip?
  
  - **RQ-2**: What properties influence this vulnerability?
  
  - **RQ-3**: Can an attacker exploit this vulnerability?
  
  - **RQ-4**: Can we utilize DNN-level mechanisms for mitigation?
RQ-1: How Vulnerable are DNNs to a Bit-flip?

• Metric
  – Relative Accuracy Drop \([\text{RAD}]\) = \frac{(\text{acc}\_\text{clean} - \text{acc}\_\text{corrupted})}{\text{acc}\_\text{clean}}

• Methodology
  – Flip \((0\rightarrow1\text{ and }1\rightarrow0)\) each bit in all parameters of a model
  – Measure the RAD over the entire validation set, each time
  – **Achilles bit**: when the bit flips, the flip inflicts \(\text{RAD} > 10\%\)

• Vulnerability
  – **Max RAD**: the maximum RAD that an Achilles bit can inflict
  – **Ratio**: the percentage of vulnerable parameters in a model
### RQ-1: Vulnerability Analysis in MNIST

<table>
<thead>
<tr>
<th>Network</th>
<th>Acc.</th>
<th># Params</th>
<th>Max RAD</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>B(ase)</td>
<td>95.71</td>
<td>21,840</td>
<td>98 %</td>
<td>50%</td>
</tr>
<tr>
<td>B-Wide</td>
<td>98.46</td>
<td>85,670</td>
<td>99 %</td>
<td>50%</td>
</tr>
<tr>
<td>B-PReLU</td>
<td>98.13</td>
<td>21,843</td>
<td>99 %</td>
<td>99%</td>
</tr>
<tr>
<td>B-Dropout</td>
<td>96.86</td>
<td>21,840</td>
<td>99 %</td>
<td>49%</td>
</tr>
<tr>
<td>B-DP-Norm</td>
<td>97.97</td>
<td>21,962</td>
<td>99 %</td>
<td>51%</td>
</tr>
<tr>
<td>L(eNet)5</td>
<td>98.81</td>
<td>61,706</td>
<td>99 %</td>
<td>47%</td>
</tr>
<tr>
<td>L5-Dropout</td>
<td>98.72</td>
<td>61,706</td>
<td>99 %</td>
<td>45%</td>
</tr>
<tr>
<td>L5-D-Norm</td>
<td>99.05</td>
<td>62,598</td>
<td>98 %</td>
<td>49%</td>
</tr>
</tbody>
</table>

- Maximum RAD ≈ 98% in all models
- > 45% of params are vulnerable in all the MNIST models
RQ-1: How Vulnerable Are Larger Models?

• Metric
  – Relative Accuracy Drop [RAD] = \( \frac{\text{acc}_{\text{clean}} - \text{acc}_{\text{corrupted}}}{\text{acc}_{\text{clean}}} \)

• Methodology
  – Flip (0→1 and 1→0) each bit in all parameters of a model
  – Measure the RAD over the entire validation set, each time
    [e.g. VGG16-ImageNet: examine 138M parameters ≈ 942 days]
RQ-1: How Vulnerable Are Larger Models?

• Metric
  – Relative Accuracy Drop [RAD] = \( \frac{\text{acc}_{\text{clean}} - \text{acc}_{\text{corrupted}}}{\text{acc}_{\text{clean}}} \)

• Methodology
  – Flip (0 → 1 and 1 → 0) each bit in all parameters of a model
  – Measure the RAD over the entire validation set, each time

• Speed-up heuristics
  – Sampled validation set (SV): use 10% of the validation set
  – Inspect only specific bits (SB): the exponents or their MSBs
  – Sampled parameters (SP): uniformly sample 20k parameters
### RQ-1: Vulnerability Analysis in Large Models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Network</th>
<th>Acc.</th>
<th># Params</th>
<th>SV</th>
<th>SB</th>
<th>SP</th>
<th>Max RAD</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>B(ase)</td>
<td>83.74</td>
<td>776K</td>
<td>✓</td>
<td>✓_{exp}</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B-Slim</td>
<td>82.19</td>
<td>197K</td>
<td>✓</td>
<td>✓_{exp}</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B-Dropout</td>
<td>81.18</td>
<td>776K</td>
<td>✓</td>
<td>✓_{exp}</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B-D-Norm</td>
<td>80.17</td>
<td>778K</td>
<td>✓</td>
<td>✓_{exp}</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AlexNet</td>
<td>83.96</td>
<td>2.5M</td>
<td>✓</td>
<td>✓_{exp}</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VGG16</td>
<td>91.34</td>
<td>14.7M</td>
<td>✓</td>
<td>✓_{exp}</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageNet</td>
<td>AlexNet</td>
<td>79.07</td>
<td>61.1M</td>
<td>✓</td>
<td>✓_{31st}</td>
<td>✓</td>
<td>(20K)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VGG16</td>
<td>90.38</td>
<td>138.4M</td>
<td>✓</td>
<td>✓_{31st}</td>
<td>✓</td>
<td>(20K)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ResNet50</td>
<td>92.86</td>
<td>25.6M</td>
<td>✓</td>
<td>✓_{31st}</td>
<td>✓</td>
<td>(20K)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DenseNet161</td>
<td>93.56</td>
<td>28.9M</td>
<td>✓</td>
<td>✓_{31st}</td>
<td>✓</td>
<td>(20K)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>InceptionV3</td>
<td>88.65</td>
<td>27.2M</td>
<td>✓</td>
<td>✓_{31st}</td>
<td>✓</td>
<td>(20K)</td>
<td></td>
</tr>
</tbody>
</table>
## RQ-1: Vulnerability Analysis in Large Models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Network</th>
<th>Acc.</th>
<th># Params</th>
<th>SV</th>
<th>SB</th>
<th>SP</th>
<th>Max RAD</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>B(ase)</td>
<td>83.74</td>
<td>776K</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>94 %</td>
<td>46.8%</td>
</tr>
<tr>
<td></td>
<td>B-Slim</td>
<td>82.19</td>
<td>197K</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>93 %</td>
<td>46.7%</td>
</tr>
<tr>
<td></td>
<td>B-Dropout</td>
<td>81.18</td>
<td>776K</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>94 %</td>
<td>40.5%</td>
</tr>
<tr>
<td></td>
<td>B-D-Norm</td>
<td>80.17</td>
<td>778K</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>97 %</td>
<td>45.9%</td>
</tr>
<tr>
<td></td>
<td>AlexNet</td>
<td>83.96</td>
<td>2.5M</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>96 %</td>
<td>47.3%</td>
</tr>
<tr>
<td></td>
<td>VGG16</td>
<td>91.34</td>
<td>14.7M</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>99 %</td>
<td>46.2%</td>
</tr>
<tr>
<td>ImageNet</td>
<td>AlexNet</td>
<td>79.07</td>
<td>61.1M</td>
<td>✓</td>
<td>✓</td>
<td>✓ (20K)</td>
<td>100 %</td>
<td>47.3%</td>
</tr>
<tr>
<td></td>
<td>VGG16</td>
<td>90.38</td>
<td>138.4M</td>
<td>✓</td>
<td>✓</td>
<td>✓ (20K)</td>
<td>99 %</td>
<td>42.1%</td>
</tr>
<tr>
<td></td>
<td>ResNet50</td>
<td>92.86</td>
<td>25.6M</td>
<td>✓</td>
<td>✓</td>
<td>✓ (20K)</td>
<td>100 %</td>
<td>47.8%</td>
</tr>
<tr>
<td></td>
<td>DenseNet161</td>
<td>93.56</td>
<td>28.9M</td>
<td>✓</td>
<td>✓</td>
<td>✓ (20K)</td>
<td>100 %</td>
<td>49.0%</td>
</tr>
<tr>
<td></td>
<td>InceptionV3</td>
<td>88.65</td>
<td>27.2M</td>
<td>✓</td>
<td>✓</td>
<td>✓ (20K)</td>
<td>100 %</td>
<td>40.8%</td>
</tr>
</tbody>
</table>
Research Questions

• **RQ-1**: How vulnerable are DNNs to a single bit-flip?

• **RQ-2**: What properties influence this vulnerability?

• **RQ-3**: Can an attacker exploit this vulnerability?

• **RQ-4**: Can we utilize DNN-level mechanisms for mitigation?
RQ-2: Properties that Influence the Vulnerability

- (Network-level) DNN-properties
- (Parameter-level) Bitwise representation
RQ-2: Impact of the Common Techniques

- (Network-level) DNN-properties
  - The dropout and batch-norm do not affect the vulnerability

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Network</th>
<th>Base acc.</th>
<th># Params</th>
<th>SV</th>
<th>SB</th>
<th>SP</th>
<th>Max RAD</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINIST</td>
<td>L(eNet)5</td>
<td>98.81</td>
<td>61,706</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>99 %</td>
<td>47%</td>
</tr>
<tr>
<td></td>
<td>L5-Dropout</td>
<td>98.72</td>
<td>61,706</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>99 %</td>
<td>45%</td>
</tr>
<tr>
<td></td>
<td>L5-D-Norm</td>
<td>99.05</td>
<td>62,598</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>98 %</td>
<td>49%</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>B(ase)</td>
<td>83.74</td>
<td>776 K</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>94 %</td>
<td>47%</td>
</tr>
<tr>
<td></td>
<td>B-Dropout</td>
<td>81.18</td>
<td>776 K</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>94 %</td>
<td>41%</td>
</tr>
<tr>
<td></td>
<td>B-D-Norm</td>
<td>80.17</td>
<td>778 K</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>97 %</td>
<td>46%</td>
</tr>
</tbody>
</table>
RQ-2: Impact of the Other DNN Properties

• (Network-level) DNN-properties
  – The dropout and batch-norm cannot reduce the vulnerability
  – The vulnerability increases proportionally with the width
  – The activation with negative values doubles the vulnerability
  – The vulnerability is consistent across 19 DNNs’ architectures
    • [8 MNIST, 5 CIFAR-10, and 5 ImageNet architectures]
RQ-2: Impact of the Parameter Sign

• (Parameter-level) Bitwise representation
  – Flip the MSB of the exponents mostly lead to \([\text{RAD} > 10\%]\)
  – The only \((0 \rightarrow 1)\) flip direction leads to \([\text{RAD} > 10\%]\)
  – The positive parameters are likely to be vulnerable to bit-flips than the negative parameters
Research Questions

• **RQ-1**: How vulnerable are DNNs to a single bit-flip?

• **RQ-2**: What properties influence this vulnerability?

• **RQ-3**: Can an attacker exploit this vulnerability?

• **RQ-4**: Can we utilize DNN-level mechanisms for mitigation?
RQ-3: Threat Model – Attacker’s Capability

• Capability
  – **Surgical**: can cause a bit-flip at an intended location
  – **Blind**: cannot control the location of a bit-flip
RQ-3: Threat Model – Attacker’s Knowledge

• Capability
  – **Surgical**: can cause a bit-flip at an intended location
  – **Blind**: cannot control the location of a bit-flip

• Knowledge:
  – **White-box**: knows the victim model internals
  – **Black-box**: has no knowledge of the victim model
RQ-3: Threat Model – Single Bit Adversary

Strongest attacker

\[ P(\text{hit an Achilles bit}) \approx 100\% \]

Weakest attacker

\[ P(\text{hit an Achilles bit}) \approx \varepsilon \]

[VGG16: 42.1% / 32-bits \approx 1.32\%]
RQ-3: Practical Weapon – Rowhammer

- Rowhammer attacks
  - **Single-bit corruption primitives** at DRAM-level
  - **Software-induced** hardware fault attacks
    [The attacker only requires a user-level access to memory]

[Double-sided Rowhammer attack]
RQ-3: Practical Weapon – Rowhammer

- Rowhammer attacks
  - **Single-bit corruption primitives** at DRAM-level
  - **Software-induced** hardware fault attacks
    [The attacker only requires a user-level access to memory]

```
0 1 1 0 1
```

*Double-sided Rowhammer attack*
RQ-3: Threat Model (Re-visited)

Weakest attacker
\[ P(\text{hit an Achilles bit}) \approx \varepsilon \]
[VGG16: 42.1% / 32-bits \approx 1.32%]

Strongest attacker
\[ P(\text{hit an Achilles bit}) \approx 100% \]
RQ-3: If Our Adversary Can Flip Multiple-Bits

Black-box

White-box

Surgical

Blind

Strongest attacker
\[ P(\text{hit an Achilles bit}) \approx 100\% \]

(Weakest) Stronger attacker
\[ \sum P(\text{hit an Achilles bit}) \gg 1.3\% \]

Sanghyun Hong, http://hardwarefail.ml
**RQ-3: The Weakest Attacker with Rowhammer**

- **Evaluation**
  - **MLaaS scenario**: a VM runs under the Rowhammer pressure
    - A Python process that constantly queries the VGG16 ImageNet model
    - Make bit-flips to the process memory: both on the code and data
      - **Consequences**: RAD > 10%, process crash, or RAD <= 10%

- **Method**: Hammertime\(^1\) DB
  - Explore Rowhammer effects systematically in 12 different DRAM chips
    - **Vulnerability of DRAM**: based on the number of bits subjected to flip

- **Experiments**
  - 25 experiments for each of 12 different DRAM chips
  - 300 cumulative bit-flip attempts for each experiment

---

\(^1\)Tatar et al., *Defeating Software Mitigations against Rowhammer: a Surgical Precision Hammer*, RAID’18
RQ-3: The Weakest Attacker with Rowhammer

• Blind attack results
  – The attacker can inflict the Terminal Brain Damage (RAD > 10%) to the victim model, effectively
    • On average, 62% (15.6/25) of the experiments were successful
    • With the most vulnerable DRAM chip, 96% (24/25) successes
    • With the least vulnerable DRAM chip, 4% (1/25) successes
  – It is Challenging to Detect the blind attacker
    • Only 6 crashes observed over the entire 7.5k bit-flip attempts

Blind Rowhammer attack is practical against DNN models
Research Questions

• **RQ-1**: How vulnerable are DNNs to single bit-flips?

• **RQ-2**: What properties influence this vulnerability?

• **RQ-3**: Can an attacker exploit this vulnerability?

• **RQ-4**: Can we utilize DNN-level mechanisms for mitigation?
RQ-4: Rowhammer Defenses

- Hardware-supported defenses to fault attack
  - ECC: Error correcting code in memory\(^1\)
  - Detection based on hardware performance counters\(^2\)

- System-level defenses to fault attack
  - CATT: Memory isolation of the kernel and user space\(^3\)
  - ZebRAM: Software-based isolation of every DRAM row\(^4\)

They require infrastructure-wide changes, or they are not effective against other hardware faults

---

\(^1\)Kim et al., *Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM* ..., ACM SIGARCH’14
\(^2\)Aweke et al., *Anvil: Software-based Protection against Next-generation Rowhammer attacks*, ACM SIGPLAN’16
\(^3\)Brasser et al., *Can’t Touch This: Software-only Mitigation against Rowhammer Attacks* ..., USENIX’17
\(^4\)Konoth et al., *Zebram: Comprehensive and Compatible Software Protection against Rowhammer Attacks*, OSDI’18
RQ-4: Can We Mitigate this Vulnerability?

- Investigate DNN-level defenses:
  - Restrict activation magnitudes: Tanh or ReLU6
  - Use low-precision numbers: quantization or binarization
RQ-4: Pros and Cons of Our Defenses

• Pros
  – Both the directions reduce the # of vulnerable parameters

• Cons
  – Require to re-train a whole model from scratch
RQ-4: Pros and Cons of Our Defenses

• Pros
  – Both directions reduce the # of vulnerable parameters
  – Substitute activation functions *without re-training*

• Cons
  – Require to re-train a whole model from scratch
  – *Expect the accuracy drop* of a model without re-training

Sanghyun Hong, http://hardwarefail.ml
Summary of Our Results

• **RQ-1**: How vulnerable are DNNs to single bit-flips?
  All DNNs have a bit whose flip causes RAD up to 100%
  40-50% of all parameters in a model are vulnerable

• **RQ-2**: What properties influence this vulnerability?
  The vulnerability is consistent across multiple DNNs

• **RQ-3**: Can an attacker exploit this vulnerability?
  Blind Rowhammer attacker can exploit this practically

• **RQ-4**: Can we utilize DNN-level mechanisms for mitigation?
  We reduce the vulnerable parameters in a model; but
  ours degrade the performance or require the re-training
Conclusions and Implications

• DNNs are not resilient to worst-case parameter perturbations
  – Re-examine techniques relying on graceful degradations with security lens

• The vulnerability of DNNs to \( \mu \)-arch. attacks is under-studied
  – Explore and evaluate new attacks, particularly thought hard
  – These attacks may be inflicted with weak attackers, e.g. blind Rowhammer

• For AI systems, system-level defenses are not sufficient
  – Consider additional model-level defenses that account for DNN properties
Thank you!

Sanghyun Hong
shhong@cs.umd.edu
http://hardwarefail.ml