Impact of Quantization for Embedded Neural Network Models on the Adversarial Robustness

Rémi BERNHARD (CEA Tech)  
Pierre-Alain MOELLIC (CEA Tech)  
Jean-Max DUTERTRE (MSE)

Laboratoire de Sécurité des Architectures et des Systèmes,  
Centre CMP, Equipe Commune CEA-Tech Mines Saint-Etienne,  
F-13541 Gardanne France

May 16, 2019
Overview – Context

- **Neural networks**: state-of-the-art performances in various complex tasks (e.g., image recognition, speech translation)
  → Classical requirements: tremendous computation power and storage limitations

- **Adversarial examples**: threaten networks’ integrity
  → Malicious perturbations which aim at fooling a model
    - Szegedy et al., *Intriguing properties of Neural Networks*, 2013
    - Goodfellow et al., *Explaining and harnessing adversarial examples*, 2015
Overview – Context: ubiquitous A.I.

- **Major trend**: Massive efforts for models deployment and embedded ML-systems
  → Mobile phones, Internet of things, ...

- **Major constraints**: Energy/Memory/Precision depending on the platform (from typical microcontroller to complex SoC):
  - Inference: keep high speed inference (no latency issue, user-friendly apps, ...)
  - Training and storage: memory footprint, duration and efficiency

E.g.: advanced STM32F4, Cortex M4, 180 MHz, 384 KBytes SRAM, 2MBytes of Flash memory

What is the impact of quantization on adversarial examples?
Outline

1. Security of Machine Learning systems
2. Adversarial Examples
3. Adversarial Attacks
4. Neural network quantization
5. Experiments
6. Conclusion and future work
Security of Machine Learning Systems
Security of Machine Learning systems

Threat Model

**Figure**: CIA threat model for a Machine Learning system

**Extract Information**

*Training data* (medical, financial, biometric, classified...)

*Model* (IP, limited authorization)

**Adversary’s Knowledge / Capacity**

Attack at learning / inference time?

What knowledge about the model?

→ White box / Black box paradigm

Probing / Querying the model

**Fool a Model**

The output prediction is not the expected one (i.e. correctly learned)

Fool a model *under the radar*, i.e. in a (almost) imperceptible way

**Critical cases:**

*Autonomous vehicle* « Stop » recognizes as « 130 km/h » sign.

*Malware detection*

**Make the System Useless**

Attack the environment (e.g. classical DoS)

Strongly alter the performance of the model
Security of Machine Learning Systems

Striking the ML pipeline

**Figure:** Illustration from Goodfellow et al., *Defense against the dark arts: An overview of adversarial example security research and future research directions.*, 2018
Adversarial Examples
Adversarial Examples: Attacking **Integrity** (at inference time)

**Principle:** Craft maliciously modified examples to fool a model.

\[
\text{Adversarial example} = \text{Clean example} + \text{Adversarial perturbation}
\]

**Figure:** NIPS 2018 Adversarial Vision Challenge

- Classification errors
- Serious threat for critical decision systems
Adversarial Examples

**Adversarial perturbation**: usually "imperceptible"... but not always!

Physical adversarial image

![Physical adversarial image](image-url)

**Figure**: Eykholt et al., *Robust Physical-World Attacks on Deep Learning Visual Classification*, 2018
Many hypothesis (and a lot of open questions...):

- Linearity hypothesis
- Boundary tilting perspective
- Different manifolds
- Data intrinsic dimension
- Statistical assumption
- Non-robust / Robust features
- ...

See: Serban et al, *Adversarial Examples – A complete Characterisation of the Phenomenon*, 2019
Adversarial Examples

Notations

\(C\): number of labels

\(M_w\): target classifier

\((x, y) \in \mathbb{R}^d \times \{1, \ldots, C\}\): observation with ground-truth label

\(M(x) \in \{1, \ldots, C\}\): predicted label of \(x\) by \(M\)

\(F(x) \in \mathbb{R}^C\): output probabilities (softmax) for \(x\)

\(f(x) \in \mathbb{R}^C\): pre-softmax (logits) for \(x\)

\(L(w, x, y) \in \mathbb{R}\): loss function of \(M\)

Pipeline:

\[ M_w : \underbrace{x}_{\text{model}} \rightarrow \underbrace{f(x)}_{\text{input}} \rightarrow \underbrace{F(x)}_{\text{logits}} \rightarrow \underbrace{M(x)}_{\text{softmax}} \rightarrow \underbrace{M(x)}_{\text{predicted label}} \]
**Adversarial Examples**

**Threat model**

**Adversarial goal**: Fool a model at inference time

From \((x, M(x))\) with \(M(x) = y\) (true label), craft \((x', M(x'))\) with
- \(M(x') \neq M(x)\) : *untargeted* attack
- \(M(x') = t\) : *targeted* attack towards label \(t\)

**Adversarial capabilities**: How much can the adversary alter \(x\)?

\[ x' = x + \alpha \] (\(\alpha\): adversarial perturbation)

\(l_p\) norm-bounded adversarial examples: \(\|\alpha\|_p \leq \epsilon\)

→ Classical attacks: \(l_2\) or \(l_\infty\) (some \(l_0\) attacks)
Adversarial Examples

Threat model

**Adversarial knowledge**: What does the adversary know about the target model $M$?

- **White-box** setting: model’s architecture and parameters
  $\rightarrow$ Derivatives of $L$, $F$ and $f$ available

- **Black-box** setting: model’s outputs only
  $\rightarrow$ no knowledge of the gradients
  $\rightarrow$ can query $M$, with/without restriction
Adversarial Examples
Transferability

**Principle:**
An adversarial example crafted to fool classifier $M_1$ may fool a classifier $M_2$

→ For the adversary, a very powerful property

**Remarks:**
- Inter and Intra-techniques transferability (many types of classifiers involved: SVM, decision trees, neural networks, etc.)
- Need to train a substitute model (architecture, training data, ...)
- Many influence factors: model architecture, test set accuracy, depth, ...
Adversarial Attacks
Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM)

**FGSM Attack**
**Principle** (gradient-based, one-step, $l_{\infty}$): linearity approximation of $L(w, x, y)$ around $x$:

$$x' = x + \epsilon \text{sign} \left( \frac{\partial L}{\partial x}(w, x, y) \right)$$

**BIM Attack**
**Principle** (gradient-based, iterative, $l_{\infty}$), a multi-step version of FGSM:

$$x_0 = x, \quad x_{n+1} = \text{clip}_{B_{\infty}(x, \epsilon)}(x_n + \alpha \text{sign}(\frac{\partial L}{\partial x}(w, x_n, y)))$$

With $B_{\infty}(x, \epsilon)$, the $\epsilon$ $l_{\infty}$ ball around $x$ and $\alpha$, the step size.
Adversarial Attacks
Carlini-Wagner $l_2$ (CWl2)

**CWl2 Attack**

**Principle** (gradient-based, iterative, $l_2$): known as one of the most powerful ($l_2$) attacks.

\[
\min_{\alpha} \|\alpha\|_2 + c K(x + \alpha, y) \\
\text{s.t} \quad x + \alpha \in [0, 1]
\]

where:

\[
K(x + \alpha, y) = \max(f_M(x)(x + \alpha) - \max_{j \neq M(x)} f_j(x + \alpha), 0)
\]
Adversarial Attacks
Simultaneous Perturbation Stochastic Approximation (SPSA)

**SPSA Attack**

**Principle** (gradient-free, iterative, $l_\infty$): a gradient-free attack

$$\min_{\alpha} \ f_M(x)(x + \alpha) - \max_{j \neq M(x)} f_j(x')$$

$$s.t \ |\alpha|_\infty < \epsilon$$

**Method:**
Adam optimizer with discrete gradient approximation:

$$g'(x)_i \simeq \frac{(g(x + \delta v) - g(x - \delta v))v_i^{-1}}{2\delta}$$

with $v \sim \{-1, 1\}^d$
**ZOO Attack**

**Principle** *(gradient-free, iterative, $l_2$):* Gradient-free softmax version of the CW$l_2$ attack

**Method:**
Adam optimizer with discrete gradient approximation:

$$g'(x)_i \approx \frac{g(x + he_i) - g(x - he_i)}{2h}$$

with $e_i$ vector with $i^{th}$ component valued 1
Two major defense strategies:

- **Proactive defenses**: adversarial training, pruning at inference, . . .
- **Reactive defenses**: detection mechanism, input preprocessing, . . .

→ Lack of certified and scalable defenses

→ Very hot topic in the ML community with numerous open questions:
  - *Properly define adversarial robustness*
  - *Lay a common benchmark for comparisons*
  - *How to evaluate it? MNIST or not MNIST?*
Adversarial Robustness
Gradient masking: a false Sense of Security

**Principle of Gradient Masking:**
Make gradients useless to craft adversarial examples

**Remarks:**
- An adversary can use a substitute model to circumvent it
- Gradient-free attacks, decision-based attacks, ...

**Figure:** Goodfellow et al., *Attacking Machine Learning with Adversarial Examples*, openAI blog, 2017
Neural Networks Quantization
Motivation: Neural networks on embedded systems

- **Memory footprint**: Parameters storage
- **Energy cost**: Efficient inference methods

→ Quantization methods
Neural networks quantization
Quantization post-training

Several tools have been recently proposed to map full precision pre-trained models to quantized models for inference purpose:

- Android NN API
- TensorFlow Lite
- ARM-NN, CMSIS-NN
- STMCubeMX. A.I.

→ Coarsely quantizing (some) weights into – usually – no more than INT8.

More advanced methods propose clustering methods, information theoretical vector quantization methods...
Neural networks quantization
Quantization-aware training

**Principle:**
Learn a model with quantized weights and/or activation values during the training process

**Issues:**
- Manage non-differentiability issues of quantization function during backward pass
- Training can be difficult
Neural networks quantization
Quantization-aware training

**Binary Neural Networks**

- weights and activations are binarized for the forward pass
  \[ w_b = \text{sign}(w), \ a_b^k = \text{sign}(a^k) \]
- Inference: only *bitcount* and *xnor* operations
- Binarization is not differentiable. Trick: use of a *Straight Through Estimator*: (STE, Bengio et al., 2013) at the backward pass
  \[
  \frac{\partial L}{\partial w} = \frac{\partial L}{\partial w_b} \frac{\partial w_b}{\partial w} \approx \frac{\partial L}{\partial w} \bigg|_{w = w_b} 1_{|w| \leq 1}
  \]
Neural networks quantization

Quantization-aware training

Low bit-width Neural Networks


$n$-bit width quantization

- Train neural networks with low-bitwidth:
  1. weights
  2. activations
  3. gradients
- STE for the backward pass
- Inference: usage of a bit convolution kernel
Neural networks quantization

Quantization-aware training

Figure: Guo et al., A Survey on Methods and Theories of Quantized Neural Networks, 2018
Massive research efforts on the topic (both attacks and defenses) with associated benchmarks and competitions (*NIPS Adversarial Vision Challenge*) but almost only on full-precision models.

**Existing works:**

- Galloway, 2017 (*Attacking binarized neural networks*): claims natural robustness with binarization. But, MNIST only, stochastic quantization
- Lin, 2019 (*Efficiency Meets Robustness*): FGSM attack only, white-box setting only (no transferability analysis)
- Khalil, 2018 (*Combinatorial attacks on binarized networks*) → not scalable on big data sets
Experiments
Experiments

Setup

Data sets:
- SVHN (training/test: 73,257/26,032)
- CIFAR10 (training/test: 50,000/10,000)

Models:
One full-precision (32-bit float) model for each data set (same CNN architecture as in Courbariaux et al., 2016)
Quantized models:
- Activation and Weight / Weight quantization: 1, 2, 3, 4 bits
- Techniques: Courbariaux et al. (2015, 2016), Zhou et al. (2016)

Computing environment:
- CPU: Intel Xeon, 2.1 GHz (12 cores)
- GPU: NVIDIA GTX 1080 Ti (11 Gb, 3584 CUDA cores)
Experiments

Attacks

1. Fast Gradient Sign Method (FGSM)
2. Basic Iterative Method (BIM): iterative FGSM
3. Carlini-Wagner $l_2$ (CWL2)
4. SPSA: Gradient free $l_\infty$ attack
5. ZOO: Gradient-free version of CWL2

<table>
<thead>
<tr>
<th></th>
<th>FGSM</th>
<th>BIM</th>
<th>CWL2</th>
<th>SPSA</th>
<th>ZOO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient-based</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Gradient-free</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>one-step</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iterative</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$l_\infty$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$l_2$</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Experiments

Metrics

**Adversarial accuracy**: accuracy of the model on adversarial examples

**$l_p$ adversarial distortion**:

$$||x' - x||_p = \left( \sum_{i=1}^{m} |x'_i - x_i|^p \right)^{\frac{1}{p}}$$

<table>
<thead>
<tr>
<th></th>
<th>CIFAR10</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full-precision</strong></td>
<td>0.89</td>
<td>0.96</td>
</tr>
<tr>
<td><strong>Bitwidth</strong></td>
<td>1  2  3  4</td>
<td>1  2  3  4</td>
</tr>
<tr>
<td><strong>Full quantization</strong></td>
<td>0.79  0.87  0.88  0.88</td>
<td>0.89  0.95  0.95  0.95</td>
</tr>
<tr>
<td><strong>Weight quantization</strong></td>
<td>0.88  0.88  0.88  0.88</td>
<td>0.96  0.95  0.96  0.95</td>
</tr>
</tbody>
</table>

**Table**: Models accuracy on test set
1) Fully binarized neural networks:

- Apparent robustness against FGSM and BIM attacks
- No robustness increase against CWl2 attack

→ No additional robustness against gradient based attacks
Experiments
Direct attacks, fully-quantized models

<table>
<thead>
<tr>
<th></th>
<th>CIFAR10</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Float model</td>
<td>Quantized models</td>
</tr>
<tr>
<td></td>
<td>(32-bit)</td>
<td>(1,2,3,4-bit)</td>
</tr>
<tr>
<td>acc</td>
<td>l₂</td>
<td>lₐ∞</td>
</tr>
<tr>
<td>BIM</td>
<td>0.07</td>
<td>1.17</td>
</tr>
<tr>
<td>SPSA</td>
<td>0.0</td>
<td>1.37</td>
</tr>
</tbody>
</table>

2) Fully quantized neural networks:
BIM (gradient-based, lₐ∞) less efficient than SPSA (gradient-free, lₐ∞)
→ Gradient masking
Experiments
Direct attacks, fully-quantized models

3) Fully quantized neural networks:

- Quantization alters ZOO objective function ($\approx 0$ or $>> 1$)
  $\rightarrow$ ZOO fails, CWl2 succeeds (thanks to STE)

- No effect from quantization
  $\rightarrow$ ZOO performs better ($l_2$ distortion)

$\rightarrow$ Gradient masking
Experiments
Transfer attacks, CIFAR10

Poor transferability capacities (particularly for CWI2)

<table>
<thead>
<tr>
<th></th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.12</td>
<td>0.58</td>
<td>0.38</td>
<td>0.41</td>
<td>0.42</td>
<td>0.40</td>
<td>0.40</td>
<td>0.40</td>
<td>0.39</td>
<td>0.12</td>
<td>0.58</td>
<td>0.38</td>
<td>0.41</td>
<td>0.42</td>
</tr>
<tr>
<td>$w_{1a1}$</td>
<td>0.82</td>
<td>0.66</td>
<td>0.82</td>
<td>0.81</td>
<td>0.81</td>
<td>0.72</td>
<td>0.81</td>
<td>0.80</td>
<td>0.80</td>
<td>0.82</td>
<td>0.66</td>
<td>0.82</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>$w_{1a32}$</td>
<td>0.33</td>
<td>0.61</td>
<td>0.11</td>
<td>0.37</td>
<td>0.38</td>
<td>0.42</td>
<td>0.37</td>
<td>0.36</td>
<td>0.36</td>
<td>0.33</td>
<td>0.61</td>
<td>0.11</td>
<td>0.37</td>
<td>0.38</td>
</tr>
<tr>
<td>$w_{3a3}$</td>
<td>0.49</td>
<td>0.49</td>
<td>0.47</td>
<td>0.43</td>
<td>0.44</td>
<td>0.17</td>
<td>0.44</td>
<td>0.43</td>
<td>0.36</td>
<td>0.49</td>
<td>0.49</td>
<td>0.47</td>
<td>0.43</td>
<td>0.44</td>
</tr>
<tr>
<td>$w_{3a32}$</td>
<td>0.46</td>
<td>0.61</td>
<td>0.47</td>
<td>0.38</td>
<td>0.39</td>
<td>0.45</td>
<td>0.18</td>
<td>0.38</td>
<td>0.38</td>
<td>0.46</td>
<td>0.61</td>
<td>0.47</td>
<td>0.38</td>
<td>0.39</td>
</tr>
</tbody>
</table>

FGSM

<table>
<thead>
<tr>
<th></th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.07</td>
<td>0.64</td>
<td>0.28</td>
<td>0.32</td>
<td>0.33</td>
<td>0.38</td>
<td>0.29</td>
<td>0.26</td>
<td>0.27</td>
<td>0.07</td>
<td>0.64</td>
<td>0.28</td>
<td>0.32</td>
<td>0.33</td>
</tr>
<tr>
<td>$w_{1a1}$</td>
<td>0.88</td>
<td>0.66</td>
<td>0.86</td>
<td>0.86</td>
<td>0.79</td>
<td>0.86</td>
<td>0.86</td>
<td>0.85</td>
<td>0.88</td>
<td>0.66</td>
<td>0.86</td>
<td>0.86</td>
<td>0.79</td>
<td>0.86</td>
</tr>
<tr>
<td>$w_{1a32}$</td>
<td>0.20</td>
<td>0.64</td>
<td>0.07</td>
<td>0.27</td>
<td>0.28</td>
<td>0.37</td>
<td>0.24</td>
<td>0.23</td>
<td>0.22</td>
<td>0.20</td>
<td>0.64</td>
<td>0.07</td>
<td>0.27</td>
<td>0.28</td>
</tr>
<tr>
<td>$w_{3a3}$</td>
<td>0.38</td>
<td>0.67</td>
<td>0.46</td>
<td>0.35</td>
<td>0.37</td>
<td>0.11</td>
<td>0.34</td>
<td>0.33</td>
<td>0.33</td>
<td>0.38</td>
<td>0.67</td>
<td>0.46</td>
<td>0.35</td>
<td>0.37</td>
</tr>
<tr>
<td>$w_{3a32}$</td>
<td>0.30</td>
<td>0.70</td>
<td>0.38</td>
<td>0.24</td>
<td>0.26</td>
<td>0.40</td>
<td>0.06</td>
<td>0.20</td>
<td>0.23</td>
<td>0.30</td>
<td>0.70</td>
<td>0.38</td>
<td>0.24</td>
<td>0.26</td>
</tr>
</tbody>
</table>

BIM

<table>
<thead>
<tr>
<th></th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.03</td>
<td>0.74</td>
<td>0.74</td>
<td>0.75</td>
<td>0.76</td>
<td>0.69</td>
<td>0.74</td>
<td>0.73</td>
<td>0.72</td>
<td>0.03</td>
<td>0.74</td>
<td>0.74</td>
<td>0.75</td>
<td>0.76</td>
</tr>
<tr>
<td>$w_{1a1}$</td>
<td>0.90</td>
<td>0.11</td>
<td>0.88</td>
<td>0.87</td>
<td>0.79</td>
<td>0.88</td>
<td>0.88</td>
<td>0.87</td>
<td>0.90</td>
<td>0.11</td>
<td>0.88</td>
<td>0.87</td>
<td>0.79</td>
<td>0.88</td>
</tr>
<tr>
<td>$w_{1a32}$</td>
<td>0.71</td>
<td>0.79</td>
<td>0.05</td>
<td>0.75</td>
<td>0.77</td>
<td>0.76</td>
<td>0.74</td>
<td>0.73</td>
<td>0.73</td>
<td>0.71</td>
<td>0.79</td>
<td>0.05</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>$w_{3a3}$</td>
<td>0.83</td>
<td>0.81</td>
<td>0.81</td>
<td>0.80</td>
<td>0.80</td>
<td>0.09</td>
<td>0.80</td>
<td>0.80</td>
<td>0.81</td>
<td>0.83</td>
<td>0.81</td>
<td>0.81</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>$w_{3a32}$</td>
<td>0.79</td>
<td>0.80</td>
<td>0.80</td>
<td>0.72</td>
<td>0.73</td>
<td>0.78</td>
<td>0.05</td>
<td>0.69</td>
<td>0.72</td>
<td>0.79</td>
<td>0.80</td>
<td>0.80</td>
<td>0.72</td>
<td>0.73</td>
</tr>
</tbody>
</table>

CWI2

<table>
<thead>
<tr>
<th></th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
<th>float</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.00</td>
<td>0.64</td>
<td>0.29</td>
<td>0.28</td>
<td>0.37</td>
<td>0.38</td>
<td>0.22</td>
<td>0.23</td>
<td>0.22</td>
<td>0.00</td>
<td>0.64</td>
<td>0.29</td>
<td>0.28</td>
<td>0.37</td>
</tr>
<tr>
<td>$w_{1a1}$</td>
<td>0.81</td>
<td>0.16</td>
<td>0.78</td>
<td>0.79</td>
<td>0.77</td>
<td>0.79</td>
<td>0.83</td>
<td>0.79</td>
<td>0.77</td>
<td>0.81</td>
<td>0.16</td>
<td>0.78</td>
<td>0.79</td>
<td>0.77</td>
</tr>
<tr>
<td>$w_{1a32}$</td>
<td>0.29</td>
<td>0.68</td>
<td>0.00</td>
<td>0.34</td>
<td>0.38</td>
<td>0.41</td>
<td>0.24</td>
<td>0.30</td>
<td>0.33</td>
<td>0.29</td>
<td>0.68</td>
<td>0.00</td>
<td>0.34</td>
<td>0.38</td>
</tr>
<tr>
<td>$w_{3a3}$</td>
<td>0.42</td>
<td>0.67</td>
<td>0.56</td>
<td>0.39</td>
<td>0.44</td>
<td>0.00</td>
<td>0.36</td>
<td>0.34</td>
<td>0.41</td>
<td>0.42</td>
<td>0.67</td>
<td>0.56</td>
<td>0.39</td>
<td>0.44</td>
</tr>
<tr>
<td>$w_{3a32}$</td>
<td>0.32</td>
<td>0.68</td>
<td>0.42</td>
<td>0.27</td>
<td>0.30</td>
<td>0.40</td>
<td>0.00</td>
<td>0.26</td>
<td>0.30</td>
<td>0.32</td>
<td>0.68</td>
<td>0.42</td>
<td>0.27</td>
<td>0.30</td>
</tr>
</tbody>
</table>

SPSA
Experiments

Conclusions on transferability: Quantization Shift Phenomenon

Quantization Shift Phenomenon: Quantization ruins the adversarial effect

- activation shift: $a^j_1(x') > a^j_2(x') \rightarrow a^{j,q}_1(x') = a^{j,q}_2(x')$

2 different activation values are mapped to the same quantization bucket.

- weight shift:

$$f^j_1(x, w_1) > f^j_2(x, w_1) \quad f^j_1(x', w_1) < f^j_2(x', w_1) \rightarrow f^j_1(x', w^{q}_1) > f^j_2(x', w^{q}_1)$$
Experiments
Conclusions on transferability: Gradient misalignment

**Gradient misalignment:**
Cosinus similarity values near 0 $\rightarrow$ near orthogonal gradients
Hard to transfer from/to fully binarized networks
Conclusion and future work
Conclusion and future work

Take-away:

- Complete study of quantized models vulnerabilities against adversarial examples, under various threat models
- Detection of some gradient masking issue
- Quantization is not a robust "natural" defense when facing advanced attacks
- But, interestingly, gradient misalignment issues and quantization shift phenomenon cause poor transferability

Future & ongoing works:

- How to improve robustness of quantized models specifically?