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)
  $\rightarrow$ Classical requirements: tremendous computation power and storage limitations

- **Adversarial examples**: threaten networks' integrity
  $\rightarrow$ 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
  $\rightarrow$ 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

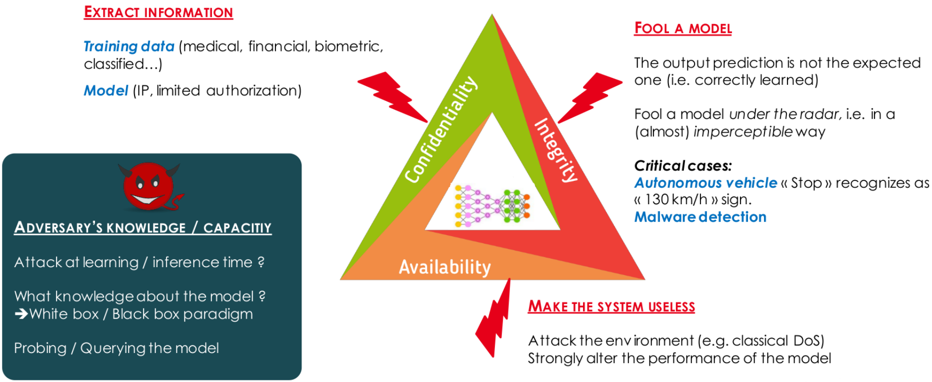


Figure: CIA threat model for a Machine Learning system

Security of Machine Learning Systems

Striking the ML pipeline

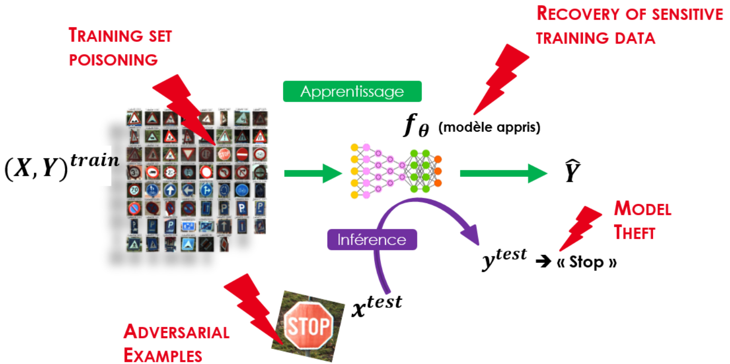


Figure: Illustration from Goodfellow et al., *Defense against the dark arts: An overview of adversarial example security research and future research directions.*, 2018

Adversarial Examples

Adversarial Examples

Adversarial Examples: Attacking Integrity (at inference time)

Principle: Craft maliciously modified examples to fool a model.

Adversarial example = Clean example + Adversarial perturbation

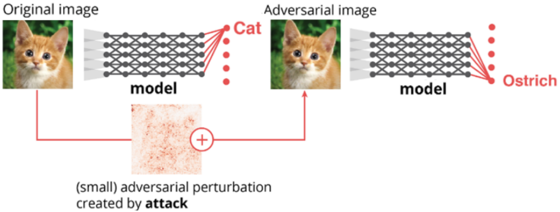


Figure: NIPS 2018 Adversarial Vision Challenge

- Classification errors
- Serious threat for critical decision systems

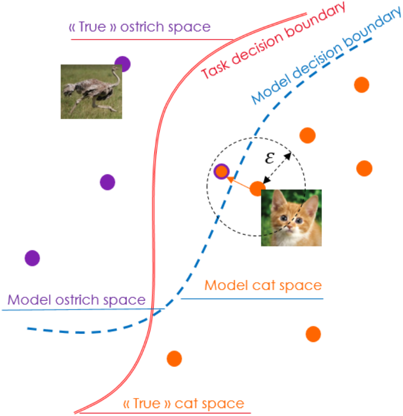
Adversarial perturbation: usually "imperceptible"... but not always!

Physical adversarial image

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

$$\underbrace{M_w}_{\textit{model}} : \underbrace{x}_{\textit{input}} \rightarrow \underbrace{f(x)}_{\textit{logits}} \rightarrow \underbrace{F(x)}_{\textit{softmax}} \rightarrow \underbrace{M(x)}_{\substack{\text{predicted} \\ \textit{label}}}$$

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 ?

$$x' = x + \alpha \text{ } (\alpha\text{: adversarial perturbation})$$

l_p norm-bounded adversarial examples: $\|\alpha\|_p \leq \epsilon$
$\rightarrow$ Classical attacks: l_2 or l_∞ (some l_0 attacks)

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

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

$\rightarrow$ 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, ...

FGSM Attack

Principle (gradient-based, one-step, l_∞): linearity approximation of $L(w, x, y)$ around x:

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

BIM Attack

Principle (gradient-based, iterative, l_∞), a multi-step version of FGSM:

$$x_0 = x, \; x_{n+1} = clip_{\mathcal{B}_\infty(x,\epsilon)}\left(x_n + \alpha sign(\frac{\partial L}{\partial x}(w, x_n, y))\right)$$

With $\mathcal{B}_\infty(x, \epsilon)$, the ϵ l_∞ ball around x and α, the step size.

CWl2 Attack

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

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

where :

$$K(x+\alpha, y) = max(f_{M(x)}(x+\alpha) - \max_{j \neq M(x)} f_j(x+\alpha), 0)$$

SPSA Attack

Principle (gradient-free, iterative, l_∞): a gradient-free attack

$$\min_{\alpha} \quad f_{M(x)}(x+\alpha) - \max_{j \neq M(x)} f_j(x')$$
$$s.t \, \|\alpha\|_\infty < \epsilon$$

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 CWl2 attack

Method:
Adam optimizer with discrete gradient approximation:

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

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?*

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

Remarks:

- Defense through obscurity (Uesato, *Adversarial Risk and the Dangers of Evaluating Against Weak Attacks*, 2018)
- An adversary can use a substitute model to circumvent it
- Gradient-free attacks, decision-based attacks, ...

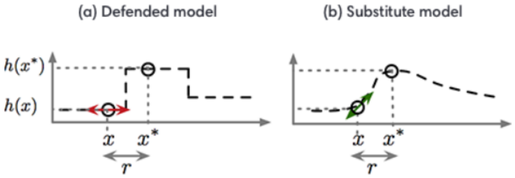


Figure: Goodfellow et al., *Attacking Machine Learning with Adversarial Examples*, openAI blog, 2017

Neural Networks Quantization

Motivation: Neural networks on embedded systems

$$\left.\begin{array}{l}\textbf{Memory footprint}\text{:}\\ \text{Parameters storage}\\ \\ \\ \textbf{Energy cost}\text{:}\\ \text{Efficient inference methods}\end{array}\right\} \rightarrow \textit{Quantization methods}$$

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...

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

Binary Neural Networks

SoA approaches: *Binary Net*, Courbariaux, Bengio et al. (2015 & 2016)

- weights and activations are binarized for the forward pass

$$w_b = sign(w),\ a_b^k = sign(a^k)$$

- 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

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial w_b}\frac{\partial w_b}{\partial w} \approx \frac{\partial L}{\partial w}\Big|_{w=w_b}\mathbf{1}_{|w|\leq 1}$$

Low bit-width Neural Networks

SoA approaches: *Dorefa Net*, Zhou et al. 2016

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

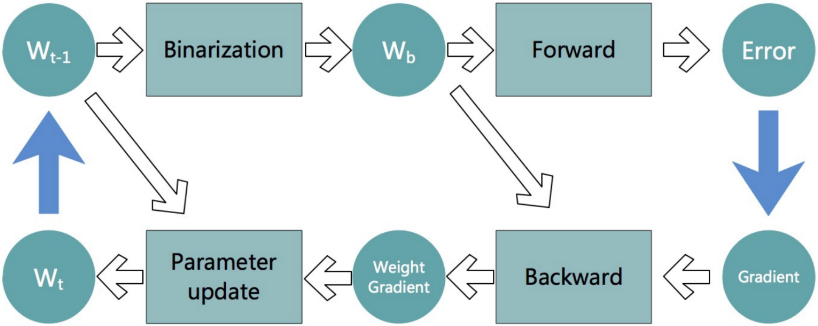


Figure: Guo et al., *A Survey on Methods and Theories of Quantized Neural Networks*, 2018

Previous Work

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*) $\rightarrow$ not scalable on big data sets

Experiments

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)

1. **Fast Gradient Sign Method (FGSM)**
2. **Basic Iterative Method (BIM)**: iterative **FGSM**
3. **Carlini-Wagner l_2 (CWl2)**
4. **SPSA**: Gradient free l_∞ attack
5. **ZOO**: Gradient-free version of **CWl2**

	FGSM	BIM	CWL2	SPSA	ZOO
Gradient-based	✓	✓	✓		
Gradient-free				✓	✓
one-step	✓				
iterative		✓	✓	✓	✓
$\boldsymbol{l_\infty}$	✓	✓		✓	
$\boldsymbol{l_2}$			✓		✓

Adversarial accuracy: accuracy of the model on adversarial examples

l_p adversarial **distortion**:

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

	CIFAR10				SVHN			
Full-precision	0.89				0.96			
Bitwidth	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Full quantization	0.79	0.87	0.88	0.88	0.89	0.95	0.95	0.95
Weight quantization	0.88	0.88	0.88	0.88	0.96	0.95	0.96	0.95

Table: Models accuracy on test set

	CIFAR10						SVHN					
	Float model *(32-bit)*			Binarized models *(1-bit)*			Float model *(32-bit)*			Binarized models *(1-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
FGSM	0.12	1.65	0.03	**0.66**	1.65	0.03	0.29	1.66	0.03	**0.78**	1.64	0.03
BIM	0.07	1.17	0.03	**0.66**	1.01	0.03	0.05	1.16	0.03	**0.79**	1.0	0.03
CWl2	0.03	0.58	0.04	**0.11**	0.78	0.08	0.02	0.64	0.66	**0.06**	1.02	0.1

1) Fully binarized neural networks:

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

$\rightarrow$ No additional robustness against gradient based attacks

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
BIM	0.07	1.17	0.03	**0.66**	1.01	0.03	0.05	1.16	0.03	**0.79**	1.0	0.03
				0.06	1.14	0.03				**0.11**	1.13	0.03
				0.11	1.17	0.03				**0.11**	1.13	0.03
				0.06	1.14	0.03				**0.1**	1.13	0.03
SPSA	0.0	1.37	0.03	**0.16**	1.31	0.03	0.01	1.38	0.03	**0.4**	1.32	0.03
				0.0	1.34	0.03				**0.14**	1.34	0.03
				0.0	1.36	0.03				**0.07**	1.35	0.03
				0.0	1.36	0.03				**0.04**	1.37	0.03

2) Fully quantized neural networks:
BIM (gradient-based, l_∞) less efficient than SPSA (gradient-free, l_∞)
$\rightarrow$ Gradient masking

Experiments

Direct attacks, fully-quantized models

	CIFAR10						SVHN					
	Float model (*32-bit*)			Quantized models (*1,2,3,4-bit*)			Float model (*32-bit*)			Quantized models (*1,2,3,4-bit*)		
	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞	acc	l_2	l_∞
CWl2	0.03	0.58	0.04	**0.11** **0.06** **0.09** **0.05**	**0.78** **0.6** **0.55** **0.5**	0.08 0.04 0.04 0.04	0.02	0.64	0.06	**0.06** **0.03** **0.02** **0.02**	**1.02** **0.67** **0.66** **0.68**	0.1 0.07 0.07 0.07
ZOO	0.0	0.72	0.09	**0.56** **0.83** **0.76** **0.73**	**0.1** **0.13** **0.24** **1.09**	0.05 0.06 0.07 0.14	0.0	0.91	0.11	**0.82** **0.93** **0.94** **0.93**	**0.07** **0.1** **0.11** **0.38**	0.05 0.06 0.05 0.1

3) Fully quantized neural networks:

- Quantization alters ZOO objective function ($\simeq 0$ or $>> 1$)
  $\rightarrow$ ZOO fails, CWl2 succeeds (thanks to STE)
- No effect from quantization
  $\rightarrow$ ZOO performs better (l_2 distortion)

$\rightarrow$ Gradient masking

Poor transferability capacities (particularly for CWl2)

	float	w_1a_1	w_1a_{32}	w_2a_2	w_2a_{32}	w_3a_3	w_3a_{32}	w_4a_4	w_4a_{32}
float	0.12	0.58	0.38	0.41	0.42	0.40	0.40	0.40	0.39
w_1a_1	0.82	0.66	0.82	0.81	0.81	0.72	0.81	0.80	0.80
w_1a_{32}	0.33	0.61	0.11	0.37	0.38	0.42	0.37	0.36	0.36
w_3a_3	0.49	0.62	0.51	0.43	0.44	0.17	0.44	0.43	0.43
w_3a_{32}	0.46	0.61	0.47	0.38	0.39	0.45	0.18	0.38	0.38

FGSM

	float	w_1a_1	w_1a_{32}	w_2a_2	w_2a_{32}	w_3a_3	w_3a_{32}	w_4a_4	w_4a_{32}
float	0.07	0.64	0.28	0.32	0.33	0.38	0.29	0.26	0.27
w_1a_1	0.88	0.66	0.86	0.86	0.86	0.79	0.86	0.86	0.85
w_1a_{32}	0.20	0.64	0.07	0.27	0.28	0.37	0.24	0.23	0.22
w_3a_3	0.38	0.67	0.46	0.35	0.37	0.11	0.34	0.33	0.33
w_3a_{32}	0.30	0.70	0.38	0.24	0.26	0.40	0.06	0.20	0.23

BIM

	float	w_1a_1	w_1a_{32}	w_2a_2	w_2a_{32}	w_3a_3	w_3a_{32}	w_4a_4	w_4a_{32}
float	0.03	0.74	0.74	0.75	0.76	0.69	0.74	0.73	0.72
w_1a_1	0.90	0.11	0.88	0.87	0.88	0.79	0.88	0.88	0.87
w_1a_{32}	0.71	0.79	0.05	0.75	0.77	0.76	0.74	0.73	0.73
w_3a_3	0.83	0.81	0.81	0.80	0.80	0.09	0.80	0.80	0.81
w_3a_{32}	0.79	0.80	0.80	0.72	0.73	0.78	0.05	0.69	0.72

CWl2

	float	w_1a_1	w_1a_{32}	w_2a_2	w_2a_{32}	w_3a_3	w_3a_{32}	w_4a_4	w_4a_{32}
float	0.00	0.64	0.29	0.28	0.37	0.38	0.22	0.23	0.22
w_1a_1	0.81	0.16	0.78	0.79	0.77	0.79	0.83	0.79	0.77
w_1a_{32}	0.29	0.68	0.00	0.34	0.38	0.41	0.24	0.30	0.33
w_3a_3	0.42	0.67	0.56	0.39	0.44	0.00	0.36	0.34	0.41
w_3a_{32}	0.32	0.68	0.42	0.27	0.30	0.40	0.00	0.26	0.30

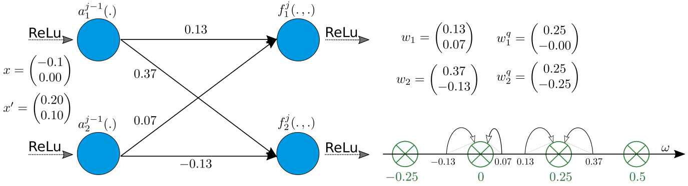
SPSA

Quantization Shift Phenomenon: Quantization ruins the adversarial effect

- activation shift: $a_1^j(x') > a_2^j(x') \rightarrow a_1^{j,\mathbf{q}}(x') = a_2^{j,\mathbf{q}}(x')$

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

- weight shift:



$f_1^j(x, w_1) > f_2^j(x, w_1) \;\; f_1^j(\mathbf{x'}, w_1) < f_2^j(\mathbf{x'}, w_1) \rightarrow f_1^j(\mathbf{x'}, w_1^{\mathbf{q}}) > f_2^j(\mathbf{x'}, w_1^{\mathbf{q}})$

Gradient misalignment:
Cosinus similarity values near 0 → 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?