

# Quantizing neural networks

## Optimizing AI - Session 2



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

## Sessions

- 1 Deep Learning and Transfer Learning,
- 2 Quantization,
- 3 Pruning,
- 4 Factorization,
- 5 Distillation,
- 6 Operators and Architectures,
- 7 Embedded Software and Hardware for DL.
- 8 Presentations for challenge.

## Sessions

- 1 Deep Learning and Transfer Learning,
- 2 Quantization,
- 3 Pruning,
- 4 Factorization,
- 5 Distillation,
- 6 Operators and Architectures,
- 7 Embedded Software and Hardware for DL.
- 8 Presentations for challenge.

# Today's Summary

## 1 Objectives

## 2 Quantization : Basics

- Floating Point
- Integers, Fixed Point
- Quantization

## 3 Quantization : Neural Networks

- Quantization Post Training
- Quantization Aware Training

## 4 Quantization in Pytorch

## 1 Objectives

## 2 Quantization : Basics

- Floating Point
- Integers, Fixed Point
- Quantization

## 3 Quantization : Neural Networks

- Quantization Post Training
- Quantization Aware Training

## 4 Quantization in Pytorch

# Motivation

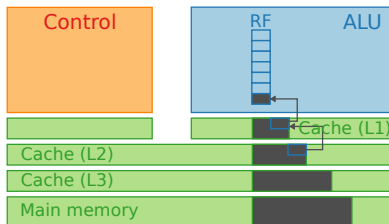
- 1 Reduce model size
  - Fewer bits → Reduced memory footprint
- 2 Decrease memory access
  - GPU & CPU : reduce Cache usage
- 3 Computational complexity

**Table:** Performance on the ImageNet dataset and complexities

Network	Alexnet	Inceptionv1	ResNet50	ResNet152
Top-5 error	16.4%	6.7%	5.25%	4.49%
Num. Weights	61M	7M	25.5M	63.75M
Num. MAC	724M	1.43G	3.9G	11.31G

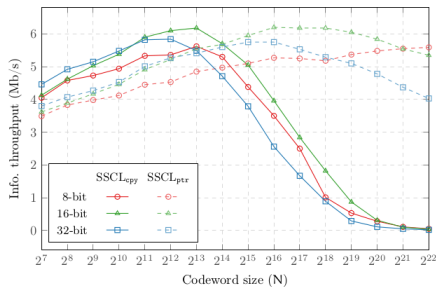
# Motivation

- 1 Reduce model size
  - Fewer bits → Reduced memory footprint
- 2 Decrease memory access
  - GPU & CPU : reduce Cache usage
- 3 Computational complexity



# Motivation

- 1 Reduce model size
  - Fewer bits  $\rightarrow$  Reduced memory footprint
- 2 Decrease memory access
  - GPU & CPU : reduce Cache usage
- 3 Computational complexity





# Motivation

- 1 Reduce model size
  - Fewer bits → Reduced memory footprint
- 2 Decrease memory access
  - GPU & CPU : reduce Cache usage
- 3 Computational complexity

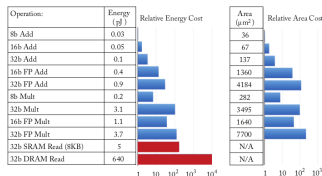
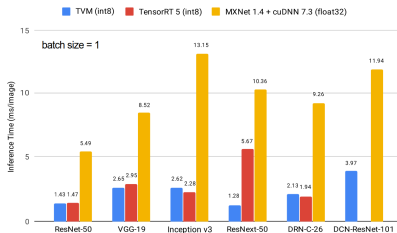


Figure 7.1: The area and energy cost for additions and multiplications at different precision, and memory accesses in a 45 nm process. The area and energy scale different for multiplication and addition. The energy consumption of data movement (red) is significantly higher than arithmetic operations (blue). (Figure adapted from [121].)

# Motivation

- 1 Reduce model size
  - Fewer bits → Reduced memory footprint
- 2 Decrease memory access
  - GPU & CPU : reduce Cache usage
- 3 Computational complexity



From : <https://github.com/vinx13/tvm-cuda-int8-benchmark/>

## 1 Objectives

## 2 Quantization : Basics

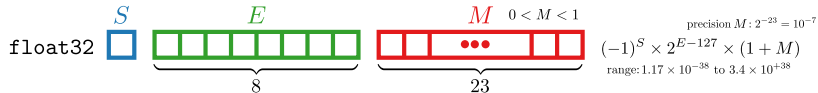
- Floating Point
- Integers, Fixed Point
- Quantization

## 3 Quantization : Neural Networks

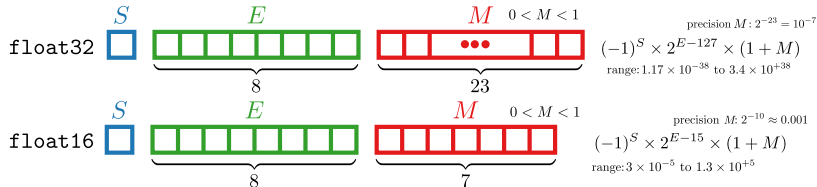
- Quantization Post Training
- Quantization Aware Training

## 4 Quantization in Pytorch

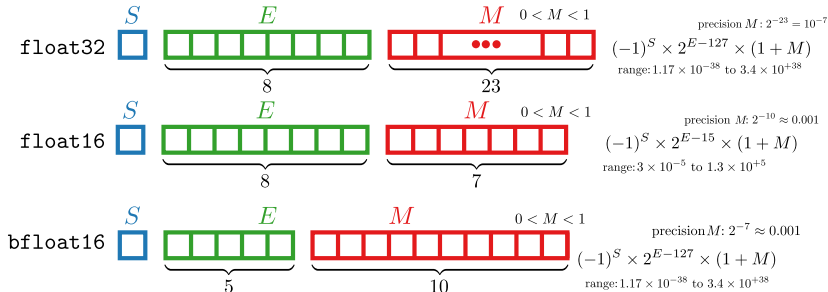
# Floating Point



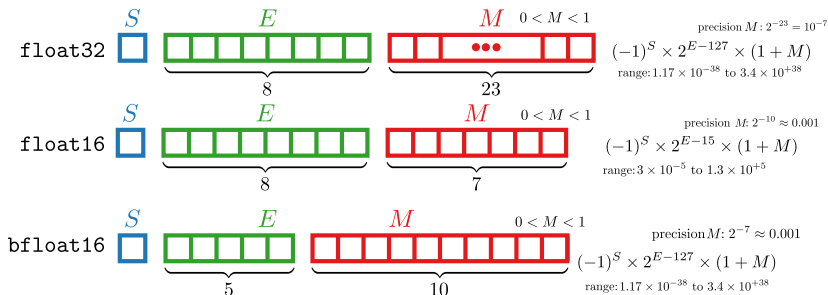
# Floating Point



# Floating Point



# Floating Point



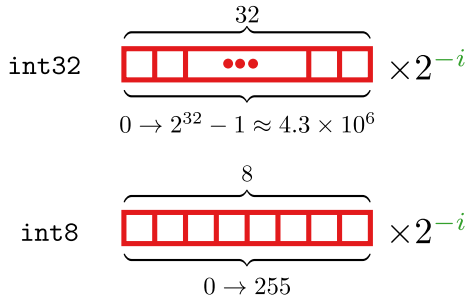
## ■ To add to FP numbers:

- Shift  $M$  according to  $E$  (int shift  $n_E$  bits)
- Add  $M$  (int add  $n_M$  bits)
- Normalize ( $0 < M < 1$ )

## ■ To multiply to FP numbers:

- Multiply  $M$  (int mult  $n_M$  bits)
- Add  $E$  (int mult  $n_E$  bits)
- Normalize ( $0 < M < 1$ )

# Integers, fixed point



- Fixed point ( $-i$ )
- Short range
- Simple computation



# Quantizing

In practice, data/weights are often random and may have high variation ranges (ex: Gaussian distribution). Quantizing with full precision requires too many bits  $\rightarrow$  need approximations to reduce the number of bits:

- **Remove  $l$  the least significant bits**  $\rightarrow$  residual errors, requires **round** operations to improve the performance.  
$$X_{\text{round}} = 2^l \text{round}(2^{-l} \times X)$$
- **Remove  $q$  the most significant bits**  $\rightarrow$  risk of overflow, requires **saturation** mechanism.  
$$X_{\text{saturate}} = \min(X, 2^{n-q} - 1) \text{ (for } n \text{ bits unsigned)}$$

Exemple:  $\mathbf{D} = \{-47, 64, 3, -26\}$  requires 9 bits for full precision. But the value 64 is closed to the value 63.

Therefore  $\hat{\mathbf{D}} = \{-47, 63, 3, -26\}$  can be used instead  $\rightarrow$  1 MSB removed + saturation at  $2^6 - 1$ , slight loss of precision.

# Estimating the impact of quantization

## Impact on weights

Signal-to-Quantization Noise Ratio metric.

$W_k$ : weight number index  $k$  in the set.

$\hat{W}_k$ : quantized weight index  $k$  in the set.

$L$ : number of element in the set.

$$\text{SQNR}(\hat{W}) = \frac{\sum_{k=0}^{L-1} |W_k|^2}{\sum_{k=0}^{L-1} \underbrace{|W_k - \hat{W}_k|^2}_{\text{quantization error}}}$$

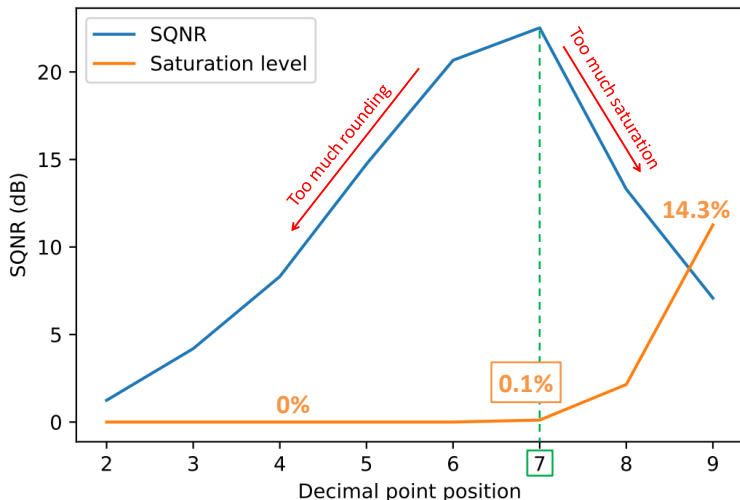
Generally expressed in dB:  $\text{SQNR}_{\text{dB}} = 10\log_{10}(\text{SQNR})$

## Impact on network performance

Directly measure the accuracy of the network. For instance: Top-1 or Top-5 errors.

# Quantizing trained networks: example

Fully connected network with 1 hidden layer, trained on MNIST.  
6 bits weights (hidden layer)  $\rightarrow$  SQNR optimal at  $\times 2^{-7}$  (see figure).



## 1 Objectives

## 2 Quantization : Basics

- Floating Point
- Integers, Fixed Point
- Quantization

## 3 Quantization : Neural Networks

- Quantization Post Training
- Quantization Aware Training

## 4 Quantization in Pytorch

# Quantization Post Training : Weights

Start by considering weights with a few number of bits  $n$ .  
Find the point position ( $2^{-v}$ ) which maximize the SQNR for each trained weight sets  $\hat{W}(v)$ :

$$\hat{W}_k(v) = 2^v \underbrace{\max(\min(\text{round}(2^{-v} \times W_k), 2^{n-1} - 1), -2^{n-1})}_{\text{saturation}}$$

Quantize  $\rightarrow$  measure accuracy  $\rightarrow$  increase the number of bits and repeat.

Different weight sets can be considered

- Whole network,
- per layer,
- per neuron.

Finer sets segmentation  $\rightarrow$  better accuracy.

Depends on how weights are stored in hardware (parallel accesses).

# Quantization Post Training : Activation

Start by considering **activations** with a few number of bits  $n$ .  
Find the point position ( $2^{-v}$ ) which maximize the SQNR for each **activation sample set**  $\hat{W}(v)$ :

$$\hat{W}_k(v) = 2^v \underbrace{\max(\min(\text{round}(2^{-v} \times W_k), 2^{n-1} - 1), -2^{n-1})}_{\text{saturation}}$$

Quantize  $\rightarrow$  measure accuracy  $\rightarrow$  increase the number of bits and repeat.

## Also different strategies

- Whole network,
- per layer,
- per neuron.

Finer sets segmentation  $\rightarrow$  better accuracy.

Depends on how **activations** are stored (parallel accesses).

# Quantization Aware Training

- Quantize Forward
- Quantize Backward & Forward
- Weights
- Weights & Activations

Quantization Aware Techniques yield way better accuracy

# Quantization while Learning - Binary Connect

---

**Algorithm 1** SGD training with BinaryConnect.  $C$  is the cost function for minibatch and the functions  $\text{binarize}(w)$  and  $\text{clip}(w)$  specify how to binarize and clip weights.  $L$  is the number of layers.

---

**Require:** a minibatch of (inputs, targets), previous parameters  $w_{t-1}$  (weights) and  $b_{t-1}$  (biases), and learning rate  $\eta$ .

**Ensure:** updated parameters  $w_t$  and  $b_t$ .

**1. Forward propagation:**

$w_b \leftarrow \text{binarize}(w_{t-1})$

For  $k = 1$  to  $L$ , compute  $a_k$  knowing  $a_{k-1}$ ,  $w_b$  and  $b_{t-1}$

**2. Backward propagation:**

Initialize output layer's activations gradient  $\frac{\partial C}{\partial a_L}$

For  $k = L$  to 2, compute  $\frac{\partial C}{\partial a_{k-1}}$  knowing  $\frac{\partial C}{\partial a_k}$  and  $w_b$

**3. Parameter update:**

Compute  $\frac{\partial C}{\partial w_b}$  and  $\frac{\partial C}{\partial b_{t-1}}$  knowing  $\frac{\partial C}{\partial a_k}$  and  $a_{k-1}$

$w_t \leftarrow \text{clip}(w_{t-1} - \eta \frac{\partial C}{\partial w_b})$

$b_t \leftarrow b_{t-1} - \eta \frac{\partial C}{\partial b_{t-1}}$

---

Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David.

"Binaryconnect: Training deep neural networks with binary weights during propagations." Advances in neural information processing systems. 2015.

<https://arxiv.org/pdf/1511.00363.pdf>

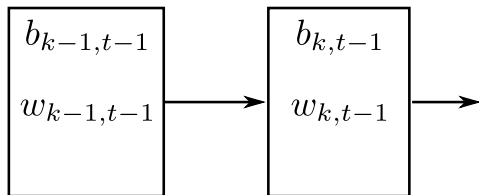


# Quantization while Learning - Binary Connect

## 1. Forward propagation:

$w_b \leftarrow \text{binarize}(w_{t-1})$

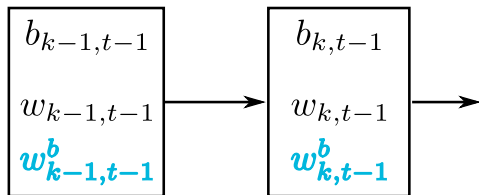
For  $k = 1$  to  $L$ , compute  $a_k$  knowing  $a_{k-1}$ ,  $w_b$  and  $b_{t-1}$



# Quantization while Learning - Binary Connect

## 1. Forward propagation:

$$w_b \leftarrow \text{binarize}(w_{t-1})$$



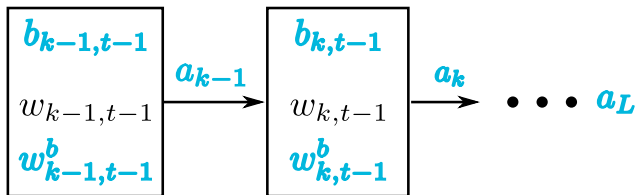
$$w^b = \begin{cases} +1, & \text{if } w \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

# Quantization while Learning - Binary Connect

## 1. Forward propagation:

$w_b \leftarrow \text{binarize}(w_{t-1})$

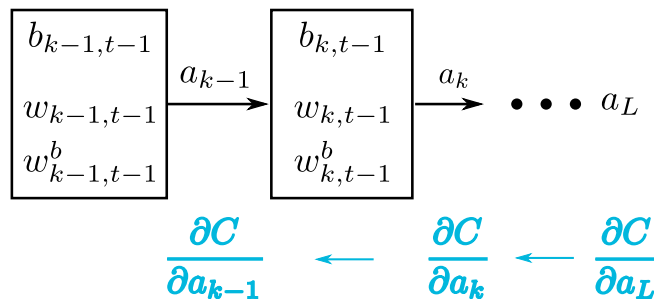
For  $k = 1$  to  $L$ , compute  $a_k$  knowing  $a_{k-1}$ ,  $w_b$  and  $b_{t-1}$



## 2. Backward propagation:

Initialize output layer's activations gradient  $\frac{\partial C}{\partial a_L}$

For  $k = L$  to 2, compute  $\frac{\partial C}{\partial a_{k-1}}$  knowing  $\frac{\partial C}{\partial a_k}$  and  $w_b$



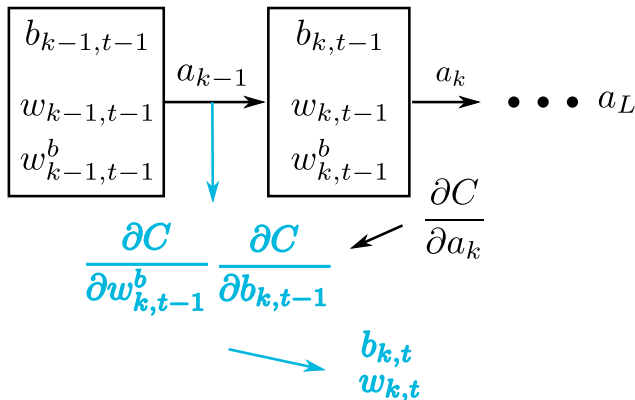
# Quantization while Learning - Binary Connect

## 3. Parameter update:

Compute  $\frac{\partial C}{\partial w_b}$  and  $\frac{\partial C}{\partial b_{t-1}}$  knowing  $\frac{\partial C}{\partial a_k}$  and  $a_{k-1}$

$$w_t \leftarrow \text{clip}(w_{t-1} - \eta \frac{\partial C}{\partial w_b})$$

$$b_t \leftarrow b_{t-1} - \eta \frac{\partial C}{\partial b_{t-1}}$$



# Binarization : Stochastic vs Deterministic

## ■ Deterministic

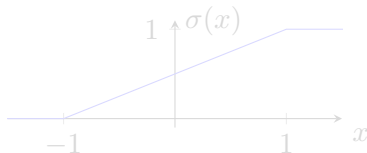
$$w_b = \begin{cases} +1, & \text{if } w \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

## ■ Stochastic

$$w_b = \begin{cases} +1, & \text{with probability } p = \sigma(w) \\ -1, & \text{with probability } 1 - p \end{cases}$$

avec

$$\sigma(x) = \text{clip}\left(\frac{x+1}{2}, 0, 1\right) = \max\left(0, \min\left(1, \frac{x+1}{2}\right)\right)$$



# Binarization : Stochastic vs Deterministic

## ■ Deterministic

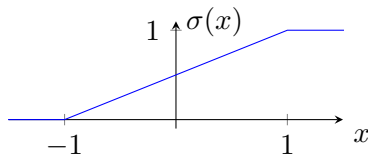
$$w_b = \begin{cases} +1, & \text{if } w \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

## ■ Stochastic

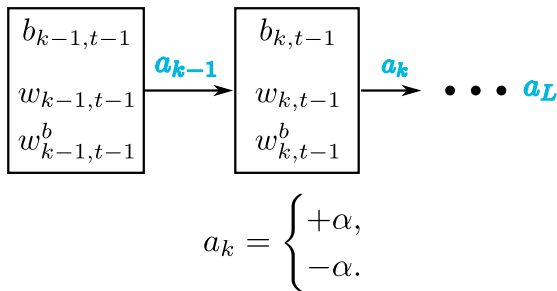
$$w_b = \begin{cases} +1, & \text{with probability } p = \sigma(w) \\ -1, & \text{with probability } 1 - p \end{cases}$$

avec

$$\sigma(x) = \text{clip}\left(\frac{x+1}{2}, 0, 1\right) = \max\left(0, \min\left(1, \frac{x+1}{2}\right)\right)$$




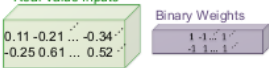

# Quantization while Learning - Binary Weighted network (XNOR-NET)



Rastegari, Mohammad, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks." European conference on computer vision. Springer, Cham, 2016. <https://arxiv.org/pdf/1603.05279.pdf>



# Quantization while Learning - Binary Weighted network (XNOR-NET)

	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution		$+, -, \times$	1x	1x	%56.7
Binary Weight		$+, -$	$\sim 32x$	$\sim 2x$	%56.8
BinaryWeight Binary Input (XNOR-Net)		XNOR, bitcount	$\sim 32x$	$\sim 58x$	%44.2

Rastegari, Mohammad, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks." European conference on computer vision. Springer, Cham, 2016. <https://arxiv.org/pdf/1603.05279.pdf>

## 1 Objectives

## 2 Quantization : Basics

- Floating Point
- Integers, Fixed Point
- Quantization

## 3 Quantization : Neural Networks

- Quantization Post Training
- Quantization Aware Training

## 4 Quantization in Pytorch

# Quantization in Pytorch