香港中文大學
The Chinese University of Hong Kong

CENG5030

Part 2-4: CNN Inaccurate Speedup-2
—– Quantization

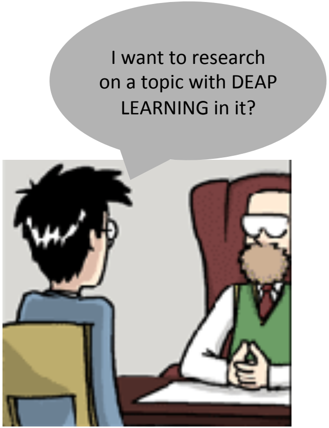
Bei Yu

(Latest update: March 25, 2019)

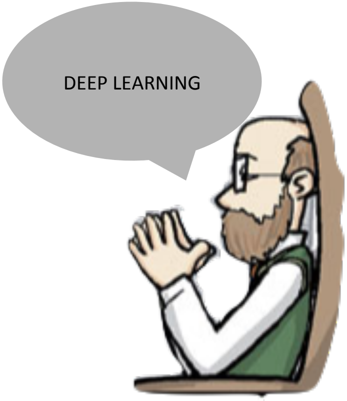
Spring 2019

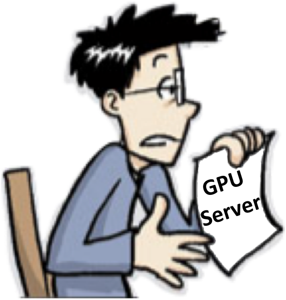
These slides contain/adapt materials developed by

- Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746
- Ritchie Zhao et al. (2017). "Accelerating binarized convolutional neural networks with software-programmable FPGAs". In: *Proc. FPGA*, pp. 15–24
- Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542


I want to research on a topic with DEAP LEARNING in it?


What should I learn to do well in computer vision research?

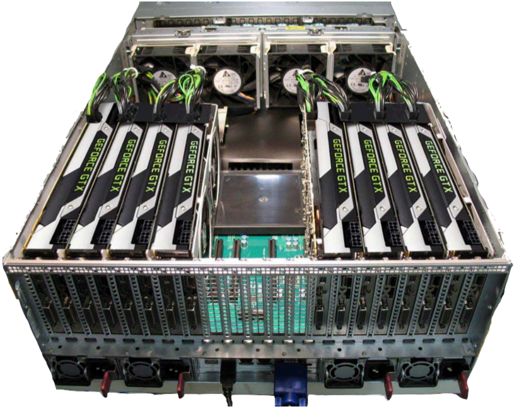

DEEP LEARNING


GPU
Server


Ohhh No!!!

State of the art recognition methods

- Very Expensive
  - Memory
  - Computation
  - Power



Overview

Fixed-Point Representation

Binary/Ternary Network

Reading List

Overview

Fixed-Point Representation

Binary/Ternary Network

Reading List

Fixed-Point v.s. Floating-Point

Fixed-Point v.s. Floating-Point

Fixed Point

$\boxed{6.5}$ 10.75

8 4 2 1 . .5 .25 .125 .0625

0 1 1 0 . 1 0 0 0

6 . 5

8 4 2 1 . .5 .25 .125 .0625

1 0 1 0 . 1 1 0 0

1 0 1 0 . 1 1 0 0

range accuracy

Fixed-Point v.s. Floating-Point

Fixed Point

$\boxed{6.5}$ 10.75

8 4 2 1 . .5 .25 .125 .0625
0 1 1 0 . 1 0 0 0

6 . 5

8 4 2 1 . .5 .25 .125 .0625
1 0 1 0 . 1 1 0 0

1010.1100

range

accuracy

Floating Point

Mantissa 5

Exponent 3

4 2 1
0.1010[011]
3

2s Comp.

8 4 2 1 .5
0101.0

5.0

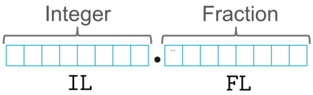
+2
0.1010[010]

4 2 1 .5 .25
010.10

2.5

Fixed-Point Arithmetic

Number representation $\langle \mathtt{IL}, \mathtt{FL} \rangle$



Word Length $\mathtt{WL} = \mathtt{IL} + \mathtt{FL}$

Granularity $2^{-\mathtt{FL}}$

Range $\left[-2^{\mathtt{IL}-1}, 2^{\mathtt{IL}-1} - 2^{-\mathtt{FL}}\right]$

$$Convert\left(x, \langle \mathtt{IL}, \mathtt{FL} \rangle\right) =$$

$$\begin{cases} -2^{\mathtt{IL}-1} & \text{if } x \leq -2^{\mathtt{IL}-1} \\ 2^{\mathtt{IL}-1} - 2^{-\mathtt{FL}} & \text{if } x \geq 2^{\mathtt{IL}-1} - 2^{-\mathtt{FL}} \\ Round(x, \langle \mathtt{IL}, \mathtt{FL} \rangle) & \text{otherwise} \end{cases}$$

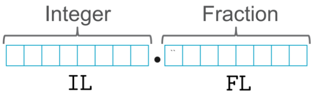
7

1

[1] Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746.

Fixed-Point Arithmetic

Number representation $\langle \mathtt{IL}, \mathtt{FL} \rangle$



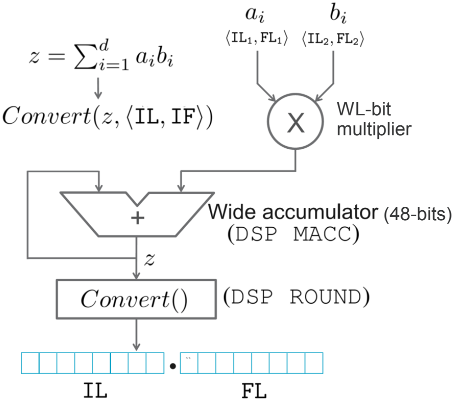
Word Length $\mathtt{WL} = \mathtt{IL} + \mathtt{FL}$

Granularity $2^{-\mathtt{FL}}$

Range $\left[-2^{\mathtt{IL}-1}, 2^{\mathtt{IL}-1} - 2^{-\mathtt{FL}}\right]$

$$Convert\left(x, \langle \mathtt{IL}, \mathtt{FL} \rangle\right) = \begin{cases} -2^{\mathtt{IL}-1} & \text{if } x \leq -2^{\mathtt{IL}-1} \\ 2^{\mathtt{IL}-1} - 2^{-\mathtt{FL}} & \text{if } x \geq 2^{\mathtt{IL}-1} - 2^{-\mathtt{FL}} \\ Round(x, \langle \mathtt{IL}, \mathtt{FL} \rangle) & \text{otherwise} \end{cases}$$

*Multiply-and-**ACC**umulate*

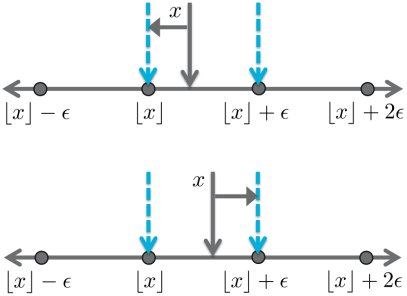


8

[1] Suyog Gupta et al. (2015). “Deep learning with limited numerical precision”. In: *Proc. ICML*, pp. 1737–1746.

Fixed-Point Arithmetic: Rounding Modes

Round-to-nearest

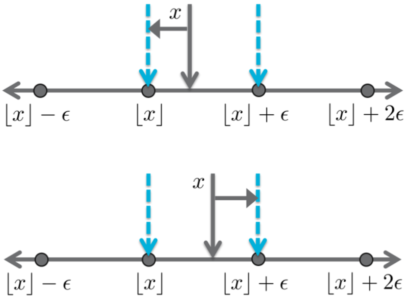


9

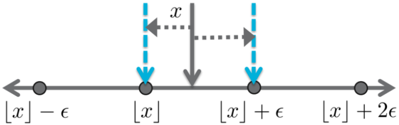
[1] Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746.

Fixed-Point Arithmetic: Rounding Modes

Round-to-nearest



Stochastic rounding



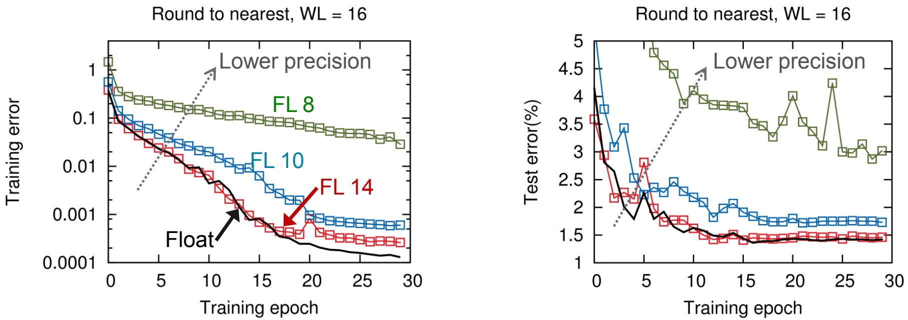
$$Round\left(x, \langle \mathtt{IL}, \mathtt{FL} \rangle\right) = \begin{cases} \lfloor x \rfloor & \text{w.p. } 1 - \frac{x - \lfloor x \rfloor}{\epsilon} \\ \lfloor x \rfloor + \epsilon & \text{w.p. } \frac{x - \lfloor x \rfloor}{\epsilon} \end{cases}$$

- Non-zero probability of rounding to either $\lfloor x \rfloor$ or $\lfloor x \rfloor + \epsilon$
- Unbiased rounding scheme: expected rounding error is zero

10 [1]

[1] Suyog Gupta et al. (2015). “Deep learning with limited numerical precision”. In: *Proc. ICML*, pp. 1737–1746.

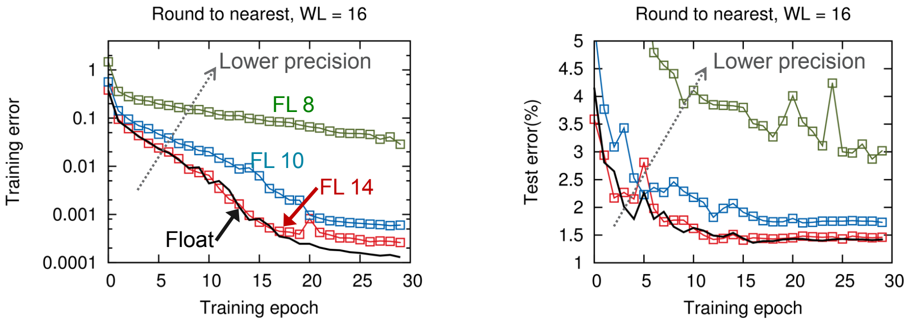
MNIST: *Fully-connected DNNs*



11

[1] Suyog Gupta et al. (2015). “Deep learning with limited numerical precision”. In: *Proc. ICML*, pp. 1737–1746.

MNIST: *Fully-connected DNNs*



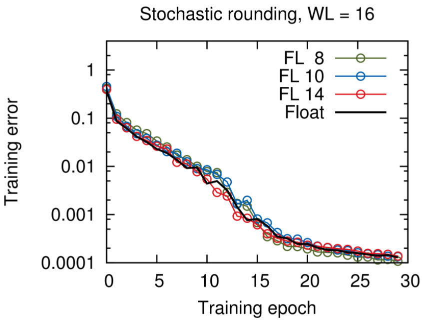
- For small fractional lengths (FL < 12), a large majority of weight updates are rounded to zero when using the round-to-nearest scheme.
  - Convergence slows down
- For FL < 12, there is a noticeable degradation in the classification accuracy

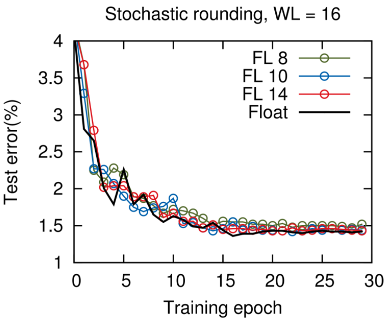
12

1

[1] Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746.

MNIST: *Fully-connected DNNs*

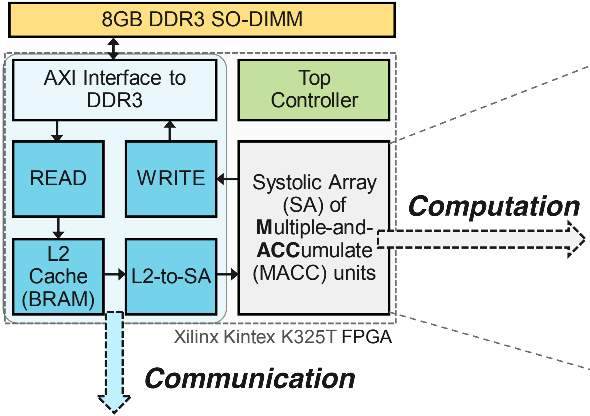




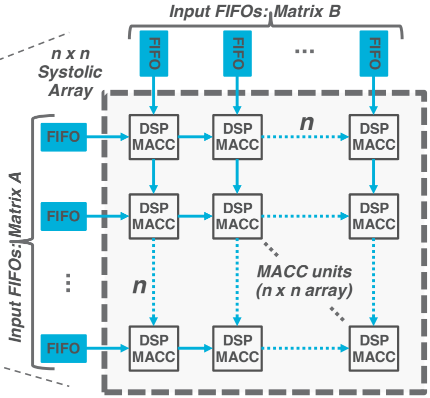
- Stochastic rounding preserves gradient information (statistically)
  - No degradation in convergence properties
- Test error nearly equal to that obtained using 32-bit floats

[1] Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746.

FPGA prototyping: GEMM with stochastic rounding



Top-level controller and memory hierarchy designed to maximize data reuse

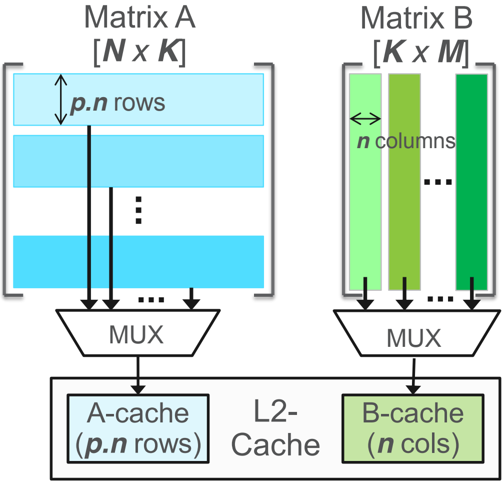


Wavefront systolic array for computing matrix product ***AB***. Arrows indicate dataflow

21 [1]

[1] Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746.

Maximizing data reuse



Inner Loop:

Cycle through columns of Matrix B (***M/n*** iterations)

Outer Loop:

Cycle through rows of Matrix A (***K/p.n*** iterations)

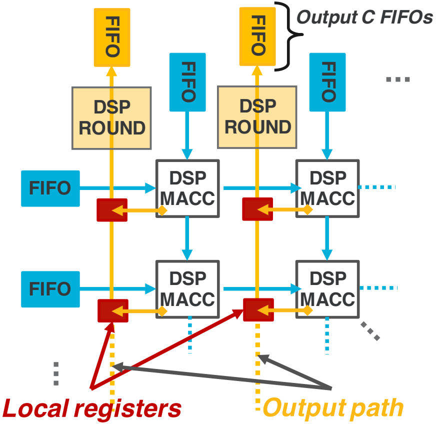
Re-use factor for Matrix A: ***M*** times
Re-use factor for Matrix B: ***p.n*** times

n : dimension of the systolic array
p : parameter chosen based on available BRAM resources

[1]

[1] Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746.

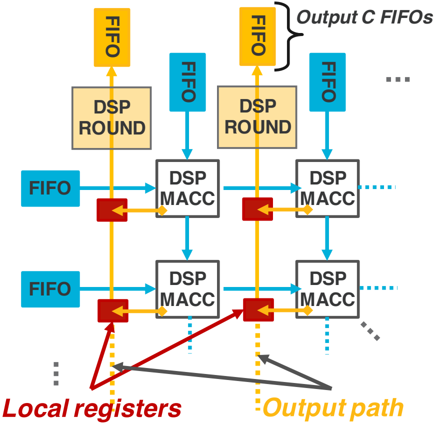
Stochastic rounding

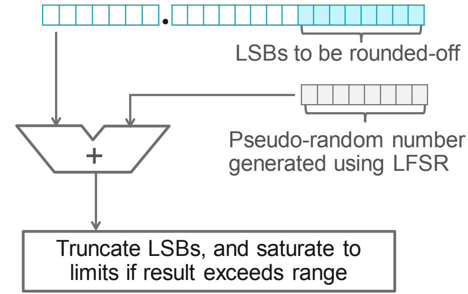


1

Suyog Gupta et al. (2015). “Deep learning with limited numerical precision”. In: *Proc. ICML*, pp. 1737–1746.

Stochastic rounding





These operations can be implemented efficiently using a single DSP unit

[1] Suyog Gupta et al. (2015). "Deep learning with limited numerical precision". In: *Proc. ICML*, pp. 1737–1746.

Overview

Fixed-Point Representation

Binary/Ternary Network

Reading List

Binarized Neural Networks (BNN)

CNN

$$\begin{bmatrix} 2.4 & 6.2 & \cdots \\ 3.3 & 1.8 & \\ \vdots & & \ddots \end{bmatrix} * \begin{bmatrix} 0.8 & 0.1 \\ 0.3 & 0.8 \end{bmatrix} = \begin{bmatrix} 5.0 & 9.1 & \cdots \\ 4.3 & 7.8 & \\ \vdots & & \ddots \end{bmatrix}$$

Input Map **Weights** **Output Map**

Key Differences

1. Inputs are binarized (–1 or +1)
2. Weights are binarized (–1 or +1)
3. Results are binarized after **batch normalization**

BNN

$$\begin{bmatrix} 1 & -1 & \cdots \\ 1 & 1 & \\ \vdots & & \ddots \end{bmatrix} * \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -3 & \cdots \\ 3 & -7 & \\ \vdots & & \ddots \end{bmatrix}$$

Input Map (Binary) **Weights** (Binary) $\boldsymbol{x}_{ij}$ (Integer)

Batch Normalization

$$\boldsymbol{y}_{ij} = \frac{\boldsymbol{x}_{ij} - \mu}{\sqrt{\sigma^2 - \epsilon}}\gamma + \beta$$

$$\downarrow$$

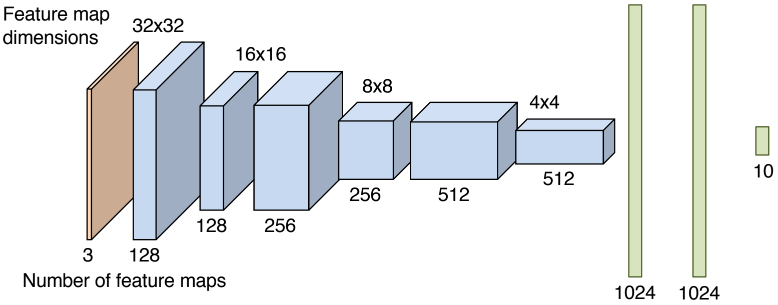
$$\boldsymbol{z}_{ij} = \begin{cases} +1 & \text{if } \boldsymbol{y}_{ij} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Binarization

$$\begin{bmatrix} 1 & -1 & \cdots \\ 1 & -1 & \\ \vdots & & \ddots \end{bmatrix}$$

Output Map (Binary)

BNN CIFAR-10 Architecture [2]



- 6 conv layers, 3 dense layers, 3 max pooling layers
- All conv filters are 3x3
- First conv layer takes in floating-point input
- **13.4 Mbits total model size** (after hardware optimizations)

[2] M. Courbariaux et al. **Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1**. *arXiv:1602.02830*, Feb 2016.

Advantages of BNN

1. Floating point ops replaced with binary logic ops

b_1	b_2	$b_1 \times b_2$
+1	+1	+1
+1	−1	−1
−1	+1	−1
−1	−1	+1

b_1	b_2	b_1 XOR b_2
0	0	0
0	1	1
1	0	1
1	1	0

- Encode {+1,−1} as {0,1} → multiplies become XORs
- Conv/dense layers do dot products → XOR and popcount
- Operations can map to LUT fabric as opposed to DSPs

2. Binarized weights may reduce total model size

- Fewer bits per weight may be offset by having more weights

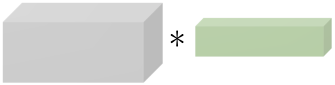
BNN vs CNN Parameter Efficiency

Architecture	Depth	Param Bits (Float)	Param Bits (Fixed-Point)	Error Rate (%)
ResNet [3] (CIFAR-10)	164	51.9M	13.0M*	11.26
BNN [2]	9	-	13.4M	11.40

* Assuming each float param can be quantized to 8-bit fixed-point

- **Comparison:**
  - Conservative assumption: ResNet can use 8-bit weights
  - BNN is based on VGG (less advanced architecture)
  - BNN seems to hold promise!

[2] M. Courbariaux et al. **Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1**. *arXiv:1602.02830*, Feb 2016.
[3] K. He, X. Zhang, S. Ren, and J. Sun. **Identity Mappings in Deep Residual Networks.** *ECCV 2016.*



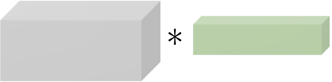
	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	$+ \; - \; \times$	1x	1x

Binary Weight Networks

XNOR-Networks

2

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

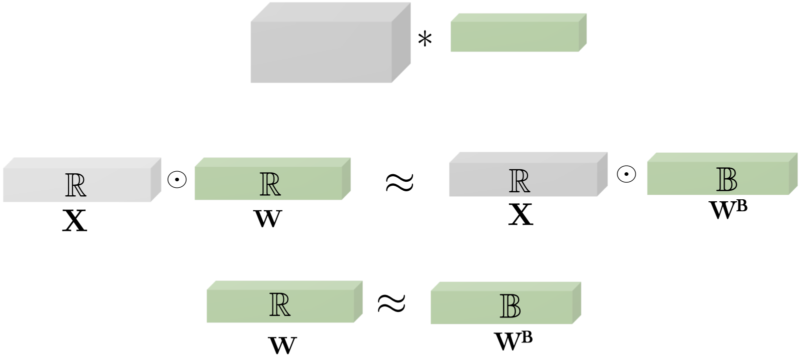


	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	$+ \; - \; \times$	1x	1x
$\mathbb{R} * \mathbb{B}$	$+ \; -$	~32x	~2x
$\mathbb{B} * \mathbb{B}$	XNOR Bit-count	~32x	~58x

[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

[2]



$$\mathbf{W}^{\mathbf{B}} = \mathrm{sign}(\mathbf{W})$$

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Quantization Error

$$\mathrm{W}^{\mathrm{B}} = \mathrm{sign}(\mathrm{W})$$

$$\left\| \underset{\mathbb{R}}{\mathbf{W}} - \underset{\mathbb{B}}{\mathbf{W}^{\mathbf{B}}} \right\| \approx 0.75$$

[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Optimal Scaling Factor

$$\underset{\mathbf{W}}{\mathbb{R}} \approx \alpha \underset{\mathbf{W^B}}{\mathbb{B}}$$

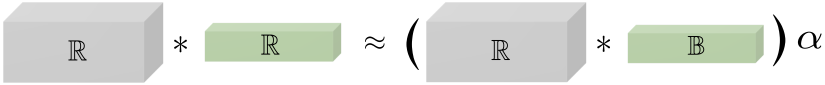
$$\alpha^*, \mathbf{W^B}^* = \arg\min_{\mathbf{W^B},\alpha} \{||\mathbf{W} - \alpha\mathbf{W^B}||^\mathbf{2}\}$$

$$\boxed{\begin{aligned} \mathbf{W^B}^* &= \operatorname{sign}(\mathbf{W}) \\ \alpha^* &= \frac{1}{n}\|\mathbf{W}\|_{\ell 1} \end{aligned}}$$

[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

How to train a CNN with binary filters?



[2]

---

[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.

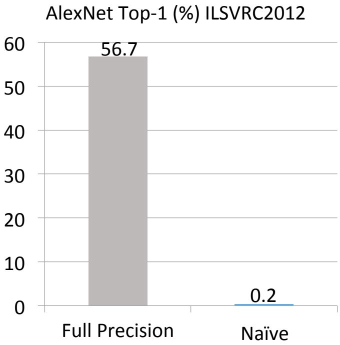
Training Binary Weight Networks

Naive Solution:

1. Train a network with real value parameters
2. Binarize the weight filters

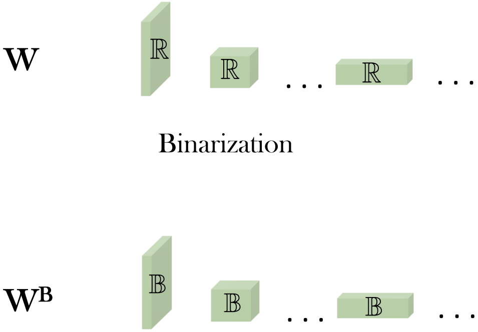
[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



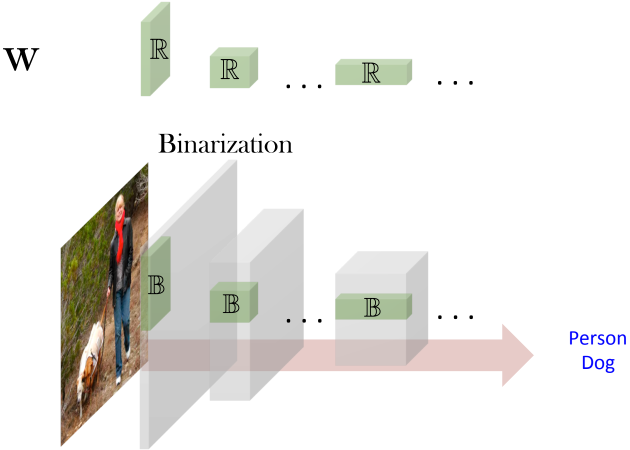
2

[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.



2

[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.



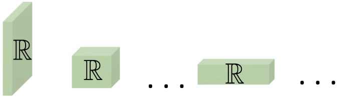
[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Weight Network

Train for binary weights:

1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$



2

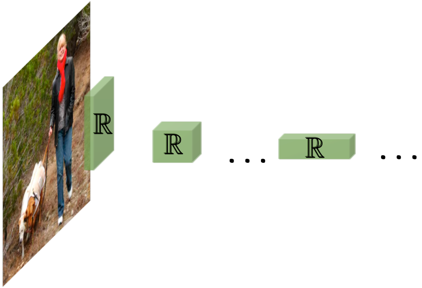
[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.

Binary Weight Network

$\mathbf{W}$

Train for binary weights:

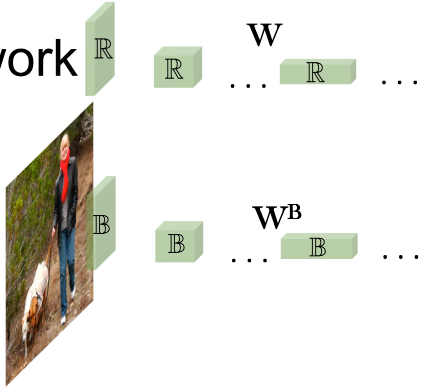
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $\left(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}}\right)$



2

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Weight Network



Train for binary weights:

1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $\left(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}}\right)$

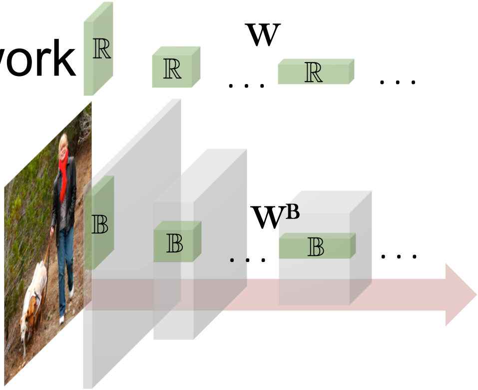
2

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Weight Network

Train for binary weights:

1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$



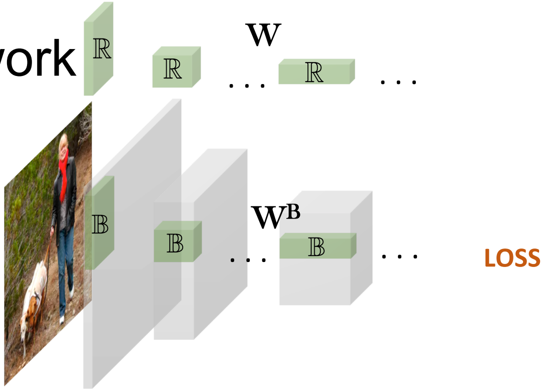
2

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Weight Network[2]

Train for binary weights:

1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$



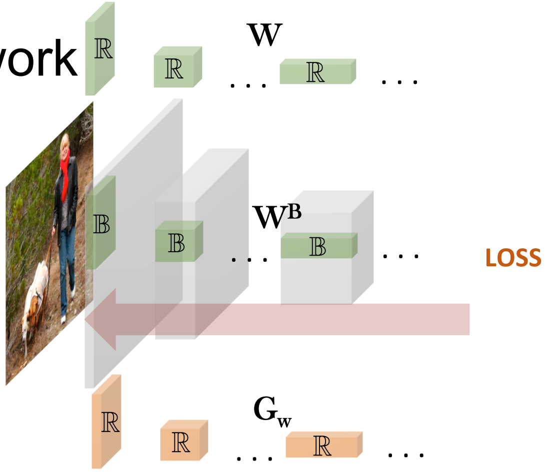
2

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

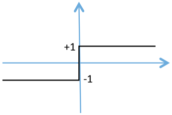
Binary Weight Network

Train for binary weights:

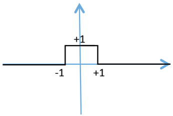
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $\left(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}}\right)$



sign(x) →



G_x →



[Hinton et al. 2012]

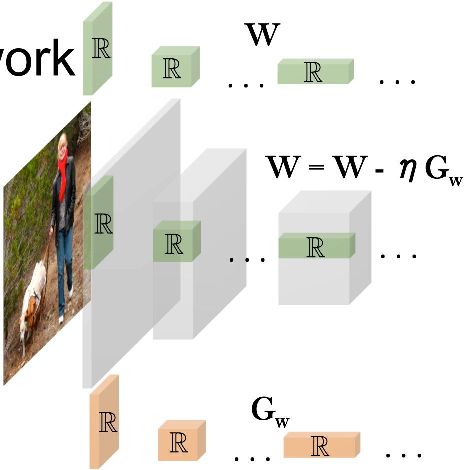
2

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

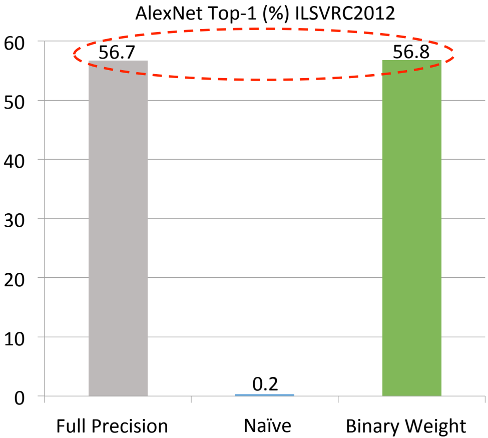
Binary Weight Network[2]

Train for binary weights:

1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \operatorname{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $\left(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}}\right)$

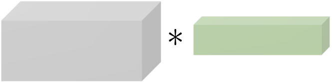


[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.



2

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	$+ \; - \; \times$	1x	1x
$\mathbb{R} * \mathbb{B}$	$\boxed{+ \; -}$	$\boxed{\sim 32x}$	$\boxed{\sim 2x}$
$\mathbb{B} * \mathbb{B}$ XNOR-Networks	XNOR Bit-count	~32x	~58x

[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Input and Binary Weight (XNOR-Net)

$$\underset{\mathbf{X}}{\mathbb{R}} \odot \underset{\mathbf{W}}{\mathbb{R}} \approx {\color{red}\beta} \underset{\mathbf{X^B}}{\mathbb{B}} \odot {\color{red}\alpha} \underset{\mathbf{W^B}}{\mathbb{B}}$$

[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Input and Binary Weight (XNOR-Net)

$$\underbrace{\underset{\mathbf{X}}{\mathbb{R}} \odot \underset{\mathbf{W}}{\mathbb{R}}}_{\mathbf{Y}} \approx \underbrace{\beta\,\alpha}_{\gamma} \underbrace{\underset{\mathbf{X}^{\mathbf{B}}}{\mathbb{B}} \odot \underset{\mathbf{W}^{\mathbf{B}}}{\mathbb{B}}}_{\mathbf{Y}^{\mathbf{B}}}$$

$$\mathbf{Y} \approx \gamma\ \mathbf{Y}^{\mathbf{B}}$$

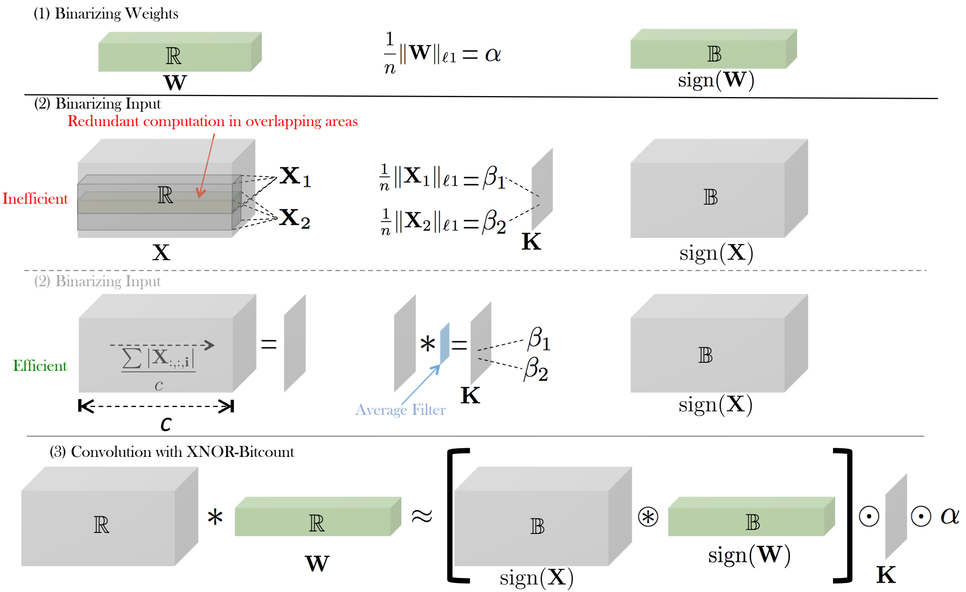
$$\mathbf{Y}^{\mathbf{B}^*}, \gamma^* = \arg\min_{\mathbf{Y}^{\mathbf{B}}, \gamma} \|\mathbf{Y} - \gamma \mathbf{Y}^{\mathbf{B}}\|^{\mathbf{2}}$$

$$\boxed{\mathbf{Y}^{\mathbf{B}^*} = \operatorname{sign}(\mathbf{Y}) \quad \gamma^* = \frac{1}{n}\|\mathbf{Y}\|_{\ell 1}}$$

$$\boxed{\mathbf{X}^{\mathbf{B}^*} = \operatorname{sign}(\mathbf{X}) \quad \mathbf{W}^{\mathbf{B}^*} = \operatorname{sign}(\mathbf{W})} \qquad \boxed{\alpha^* = \frac{1}{n}\|\mathbf{W}\|_{\ell 1} \qquad \beta^* = \frac{1}{n}\|\mathbf{X}\|_{\ell 1}}$$

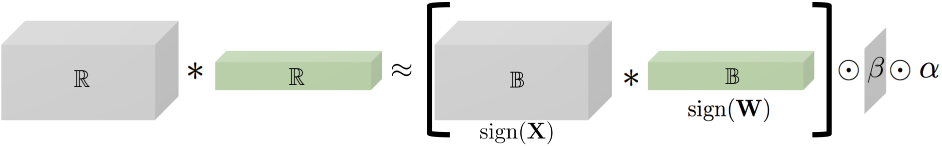
[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



2

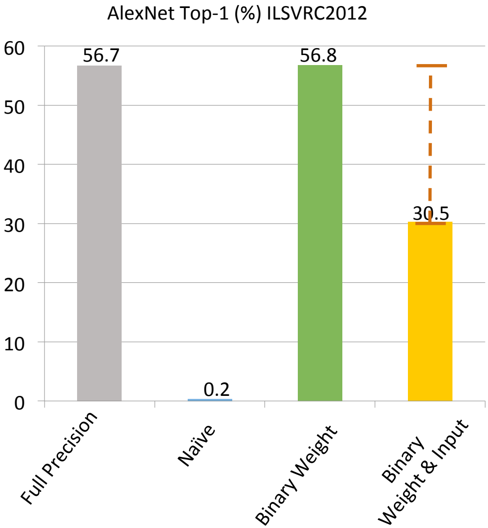
[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
9. Update $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$

[2]

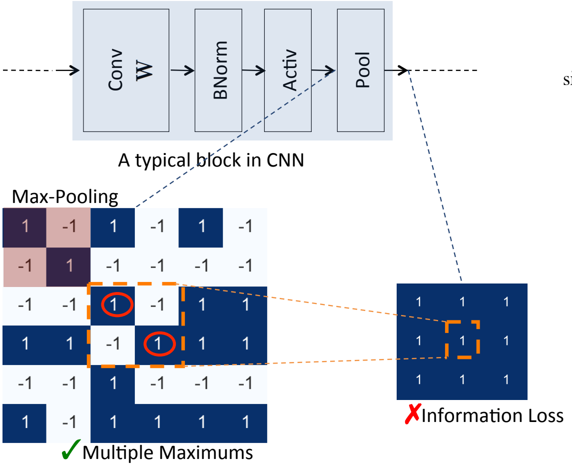
[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.

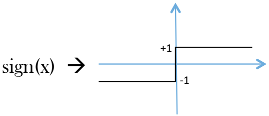


[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Network Structure in XNOR-Networks

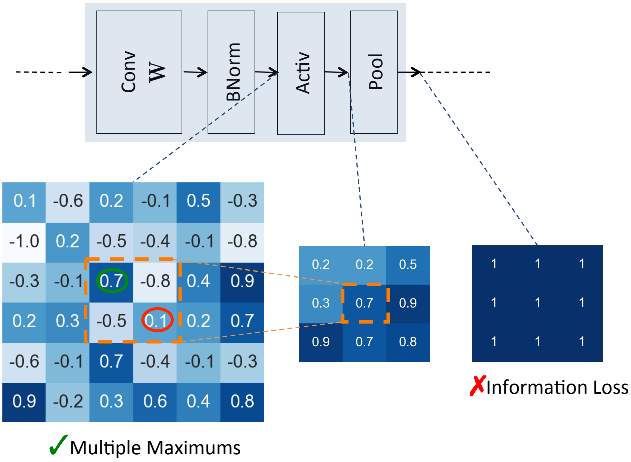




[2]

[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

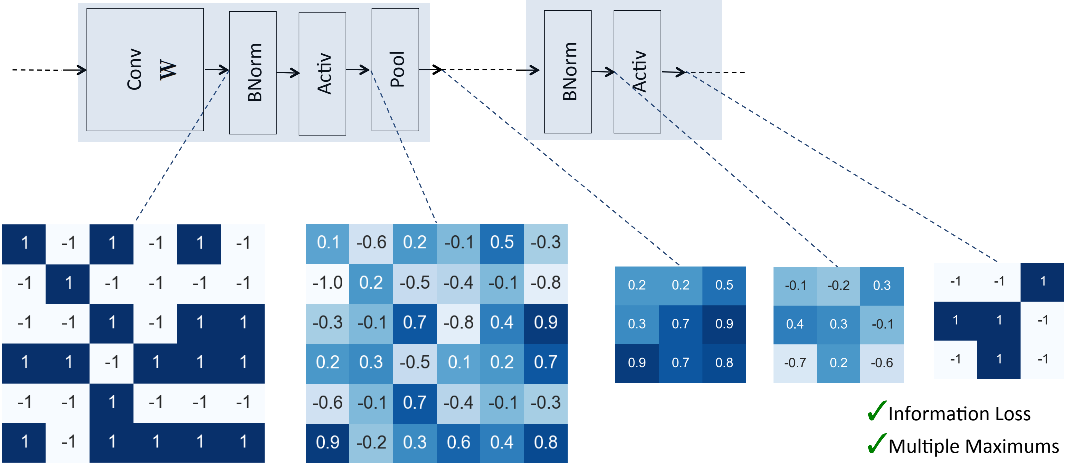
Network Structure in XNOR-Networks



[2]

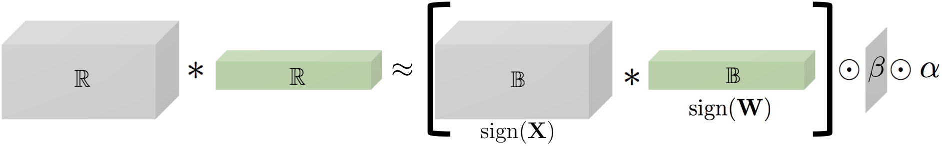
[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Network Structure in XNOR-Networks

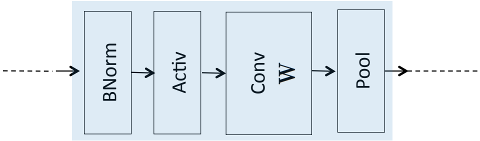


✓Information Loss
✓Multiple Maximums [2]

[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.

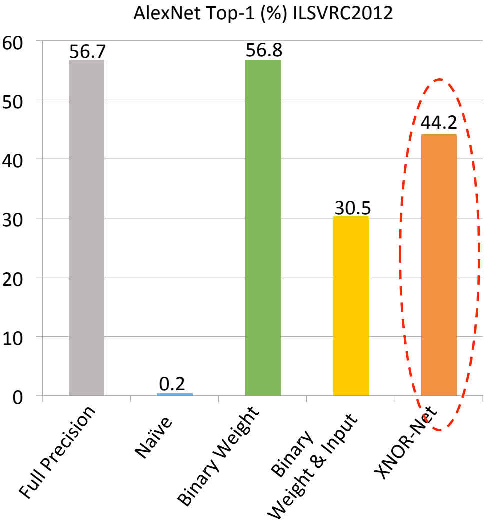


1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W^B} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W^B}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial \mathbf{W}} =$ Backward pass with $\alpha, \mathbf{W^B}$
9. Update $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$

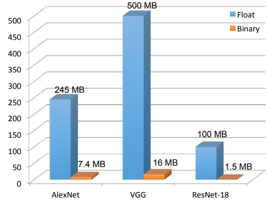


[2]

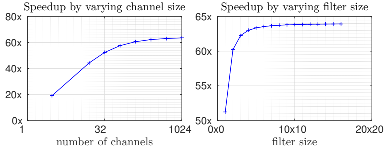
[2] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



✓ 32x Smaller Model

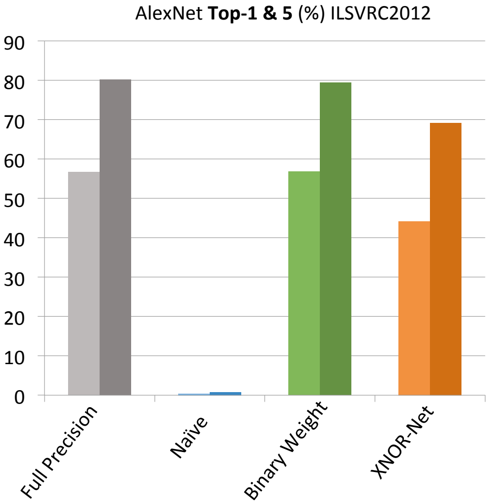


✓ 58x Less Computation



2

[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.



[2]

[2] Mohammad Rastegari et al. (2016). “XNOR-NET: Imagenet classification using binary convolutional neural networks”. In: *Proc. ECCV*, pp. 525–542.

Overview

Fixed-Point Representation

Binary/Ternary Network

Reading List

Further Reading List

Fixed-Point Representation:

- Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy (2016). "Fixed point quantization of deep convolutional networks". In: *Proc. ICML*, pp. 2849–2858
- Soroosh Khoram and Jing Li (2018). "Adaptive quantization of neural networks". In: *Proc. ICLR*

Binary/Ternary Network:

- Hyeonuk Kim et al. (2017). "A Kernel Decomposition Architecture for Binary-weight Convolutional Neural Networks". In: *Proc. DAC*, 60:1–60:6
- Chenzhuo Zhu et al. (2017). "Trained ternary quantization". In: *Proc. ICLR*