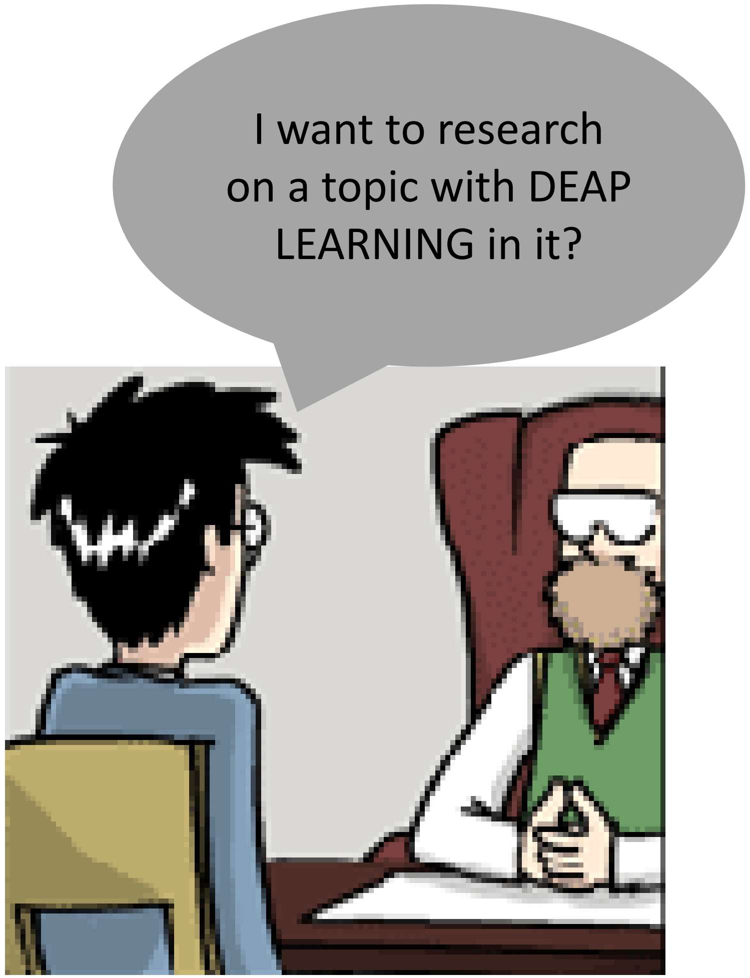
XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

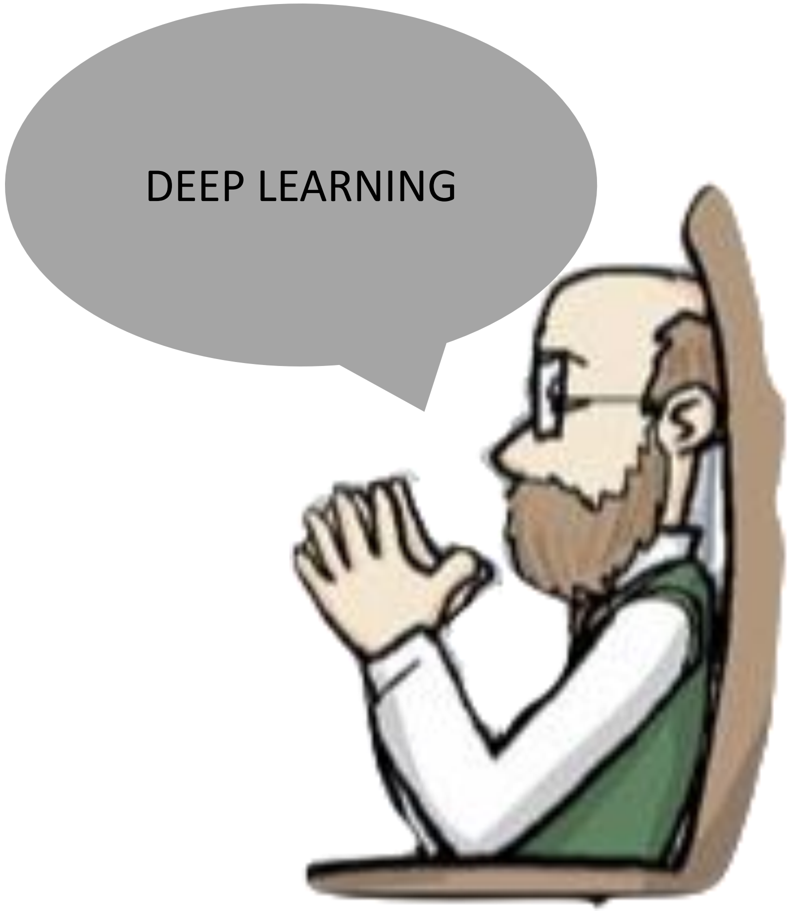
Mohammad Rastegari
Vicente Ordonez
Joseph Redmon
Ali Farhadi

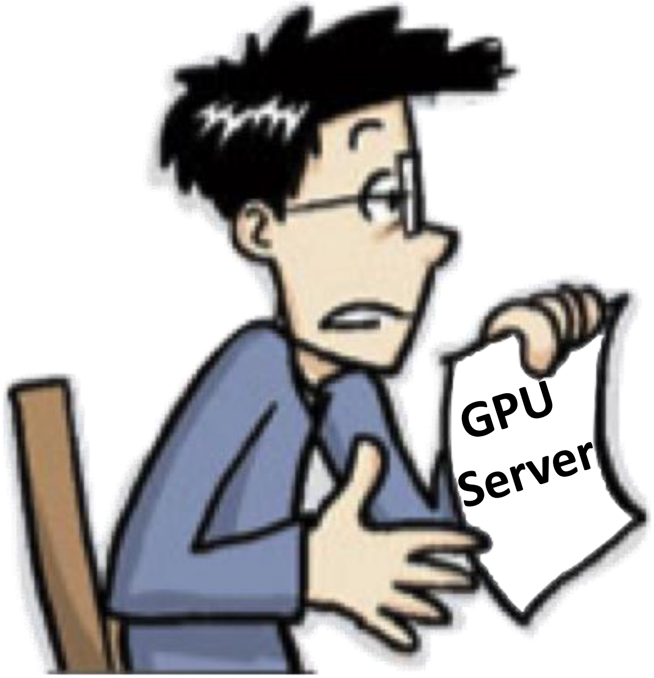


What should I learn to do well in computer vision research?


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!!!

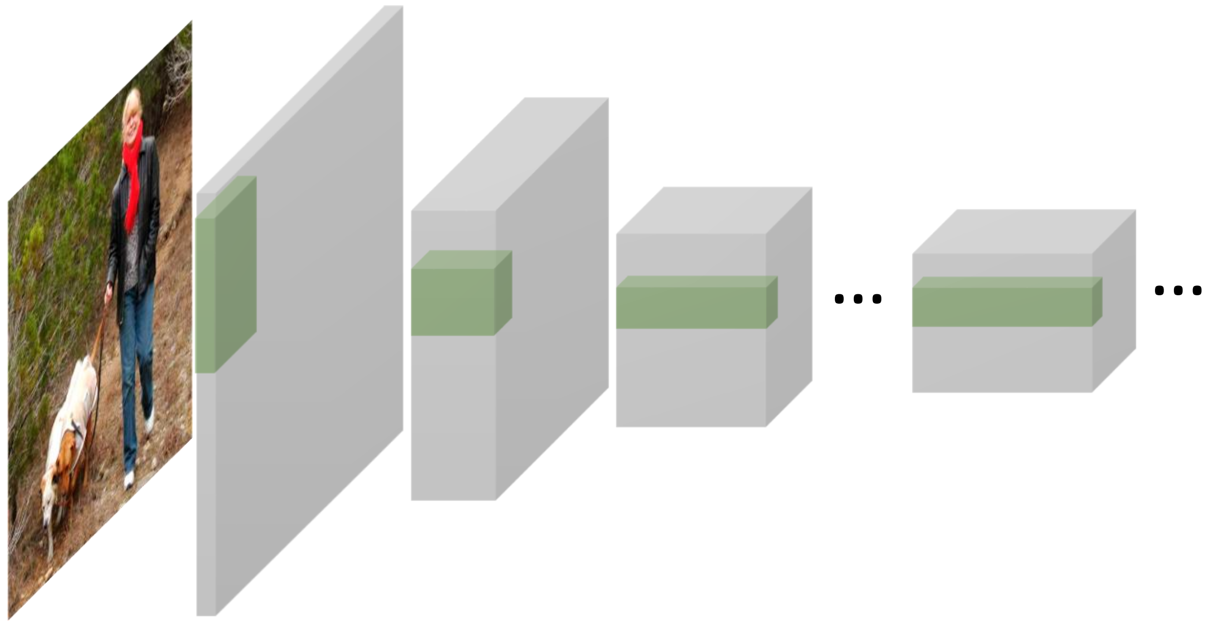
State of the art recognition methods

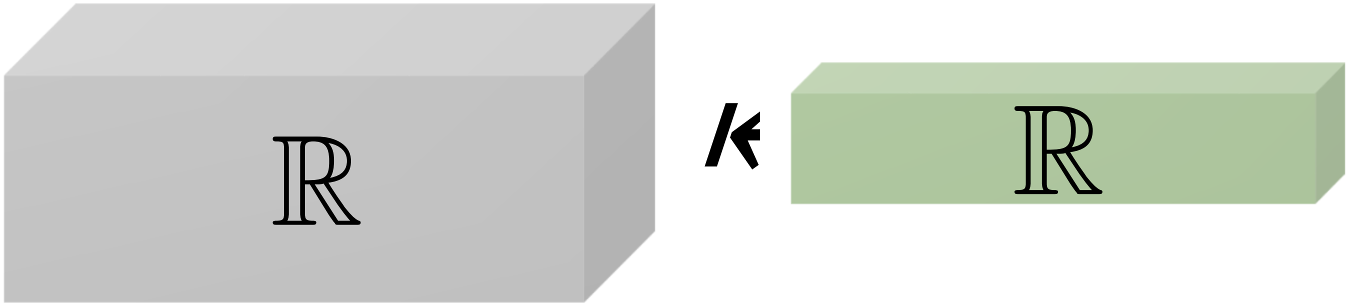
State of the art recognition methods

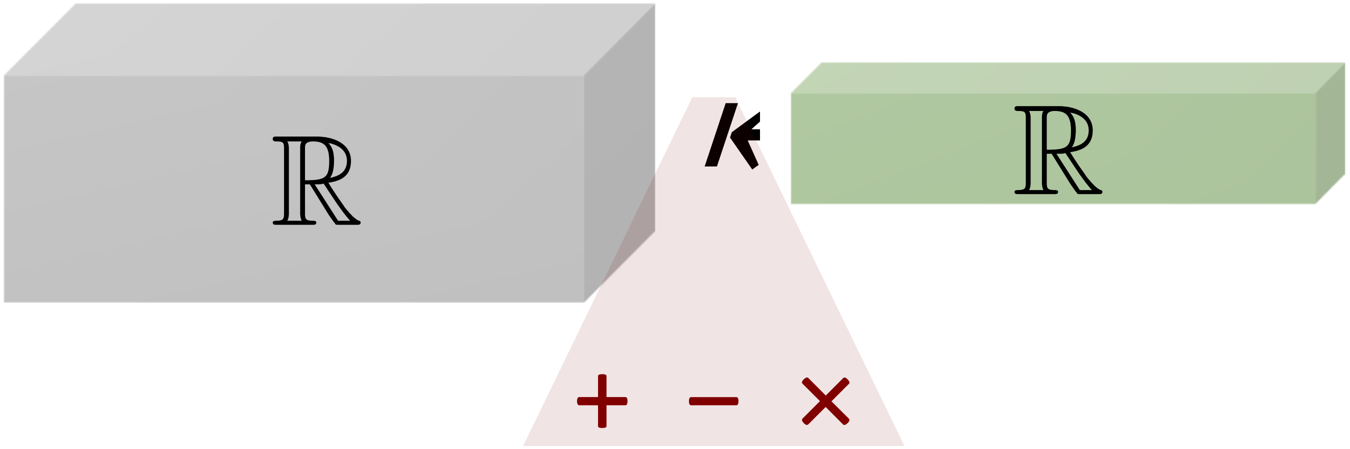
- Very Expensive
  - Memory
  - Computation
  - Power


World
Make America
Great Again

Convolutional Neural Networks




ℝ
ℝ



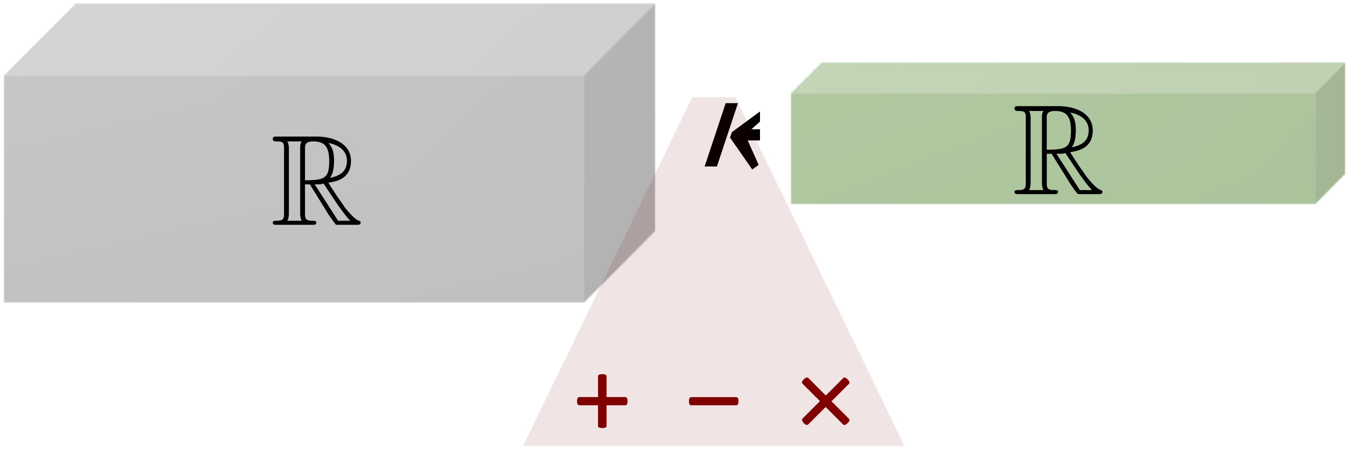
Number of Operations :

- AlexNet →1.5B FLOPs
- VGG → 19.6B FLOPs

Inference time on CPU :

- AlexNet →~3 fps
- VGG → ~0.25 fps

GPU !



Number of Operations :

- AlexNet →1.5B FLOPs
- VGG → 19.6B FLOPs

Inference time on CPU :

- AlexNet →~3 fps
- VGG → ~0.25 fps

Lower Precision

$\mathbb{R}$

32-bit

Lower Precision

Reducing Precision

- Saving Memory
- Saving Computation

$$\underset{\text{32-bit}}{\mathbb{R}} \longrightarrow \underset{\text{8-bit}}{\mathbb{I}}$$

Lower Precision

Reducing Precision

- Saving Memory
- Saving Computation

$$\mathbb{R} \longrightarrow \mathbb{I} \longrightarrow \mathbb{B}$$

32-bit 8-bit 1-bit

Lower Precision

Reducing Precision

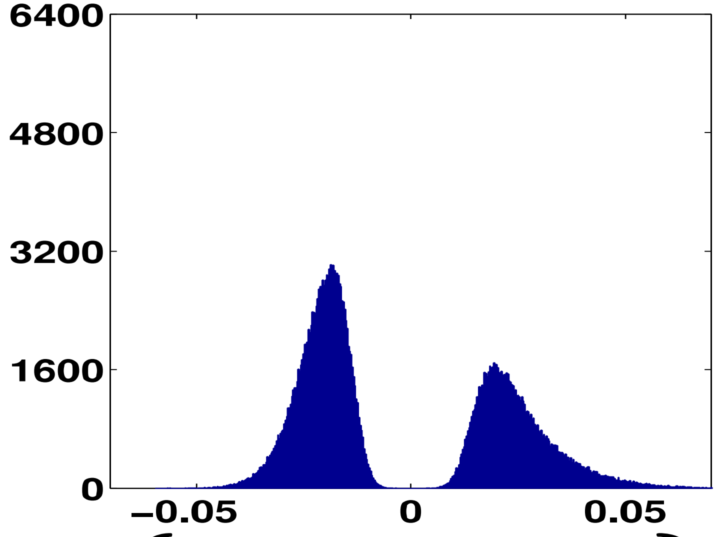
- Saving Memory
- Saving Computation

$$\mathbb{R} \longrightarrow \mathbb{I} \longrightarrow \mathbb{B} \in \{-1, +1\}$$

$\mathbb{R}$	$\mathbb{I}$	$\mathbb{B} \in \{-1, +1\}$
32-bit	8-bit	1-bit

Lower Precision

[Han et al. 2016]



Reducing Precision

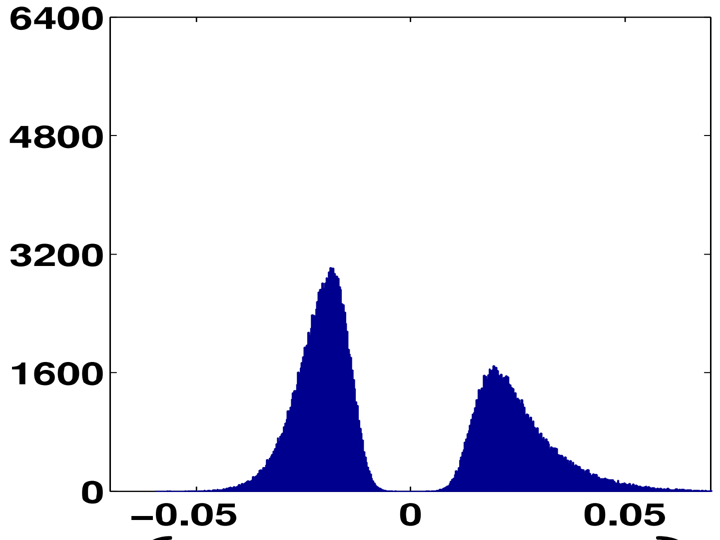
- Saving Memory
- Saving Computation

$$\mathbb{R} \rightarrow \mathbb{I} \rightarrow \mathbb{B} \in \{-1, +1\}$$

32-bit → 8-bit → 1-bit

Lower Precision

[Han et al. 2016]



Reducing Precision

- Saving Memory
- Saving Computation

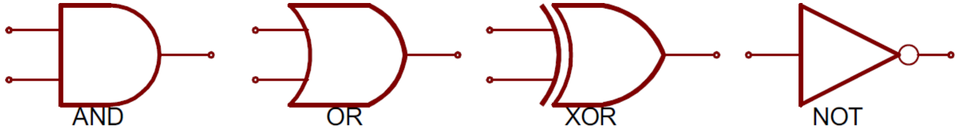
$$\mathbb{R} \longrightarrow \mathbb{I} \longrightarrow \mathbb{B} \in \{-1, +1\}$$

32-bit → 8-bit → 1-bit

{-1,+1}	{0,1}
MUL	XNOR
ADD, SUB	Bit-Count (popcount)

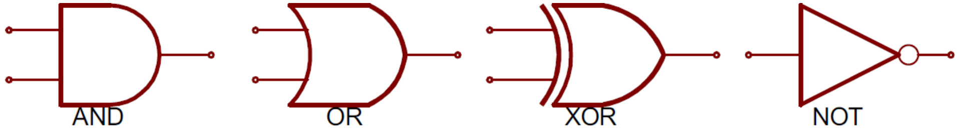
Why Binary?

- Binary Instructions
  - AND, OR, XOR, XNOR, PoPCount (Bit-Count)

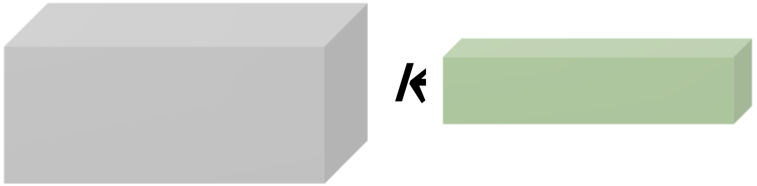


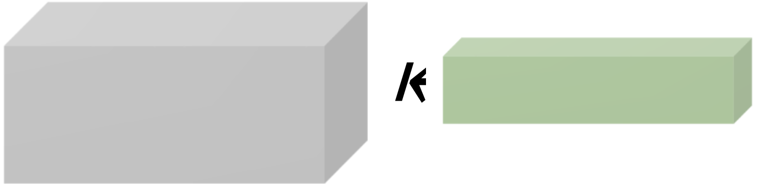
Why Binary?

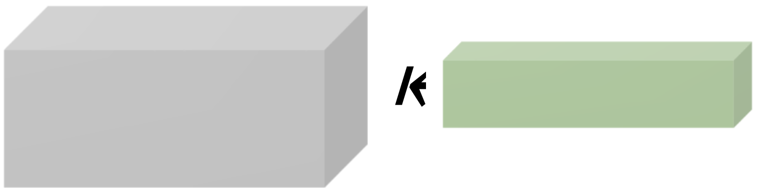
- Binary Instructions
  - AND, OR, XOR, XNOR, PoPCount (Bit-Count)



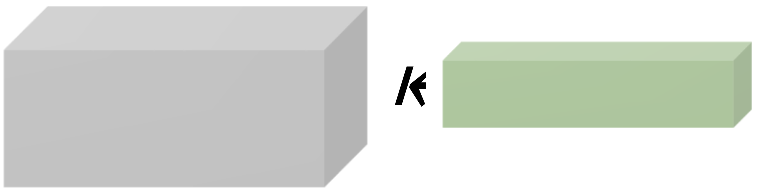
- Low Power Device

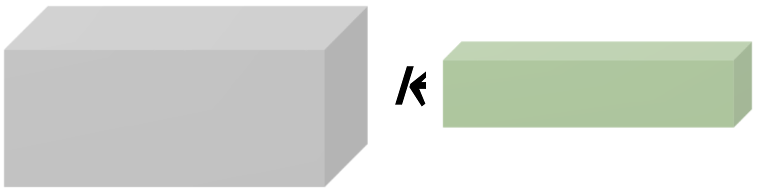
	Operations	Memory	Computation
$\mathbb{R}$ k $\mathbb{R}$	+ − ×	1x	1x

	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	+ − ×	1x	1x
$\mathbb{R} * \mathbb{B}$	+ −	~32x	~2x

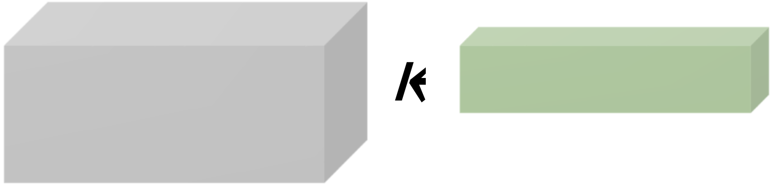
 $*$

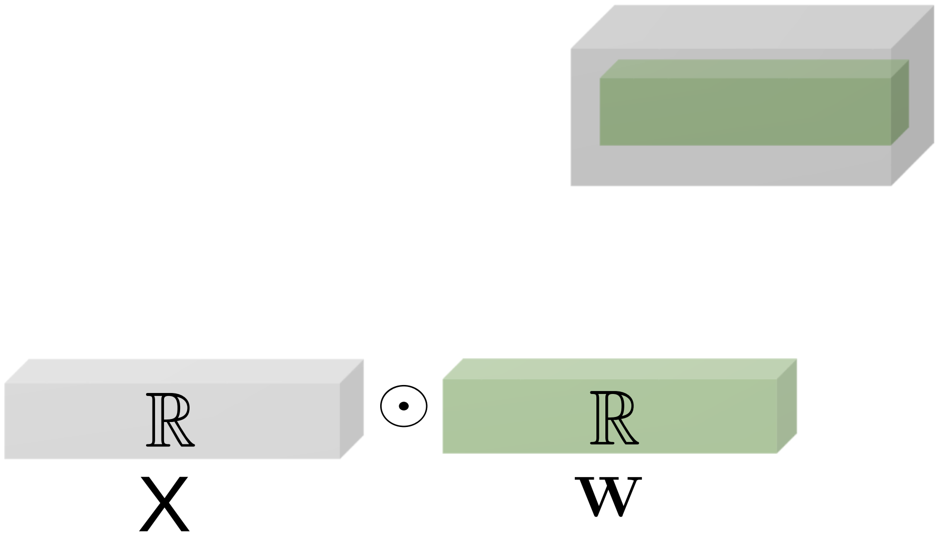
	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	+ − ×	1x	1x
$\mathbb{R} * \mathbb{B}$	+ −	~32x	~2x
$\mathbb{B} * \mathbb{B}$	XNOR Bit-count	~32x	~58x

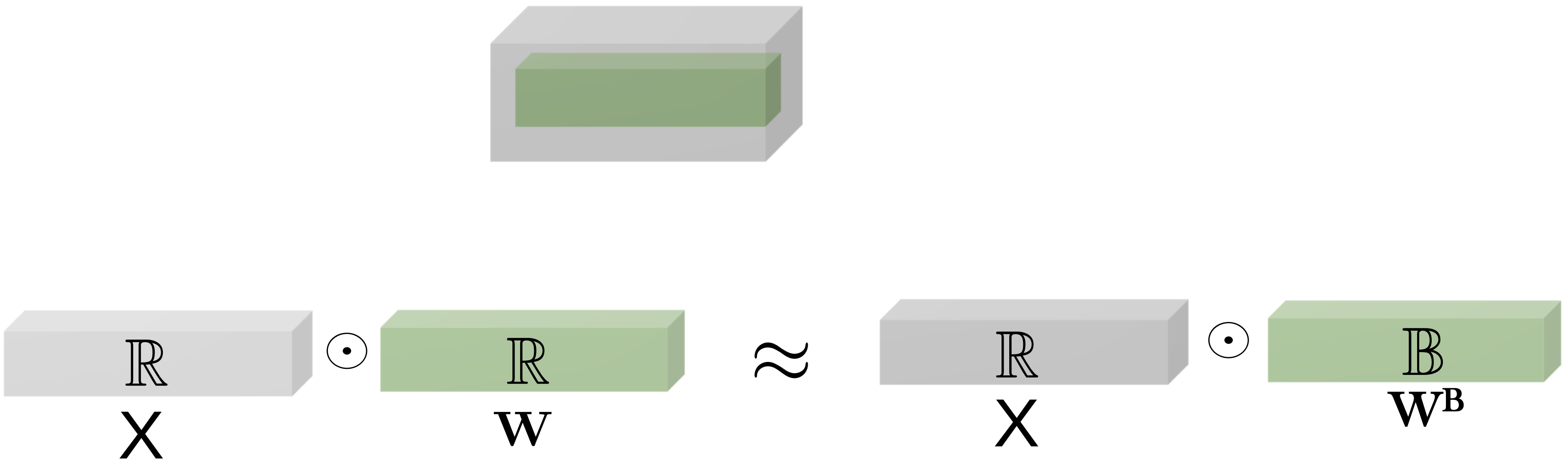
	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	+ − ×	1x	1x
$\mathbb{R} * \mathbb{B}$ Binary Weight Networks	+ −	~32x	~2x
$\mathbb{B} * \mathbb{B}$ XNOR-Networks	XNOR Bit-count	~32x	~58x



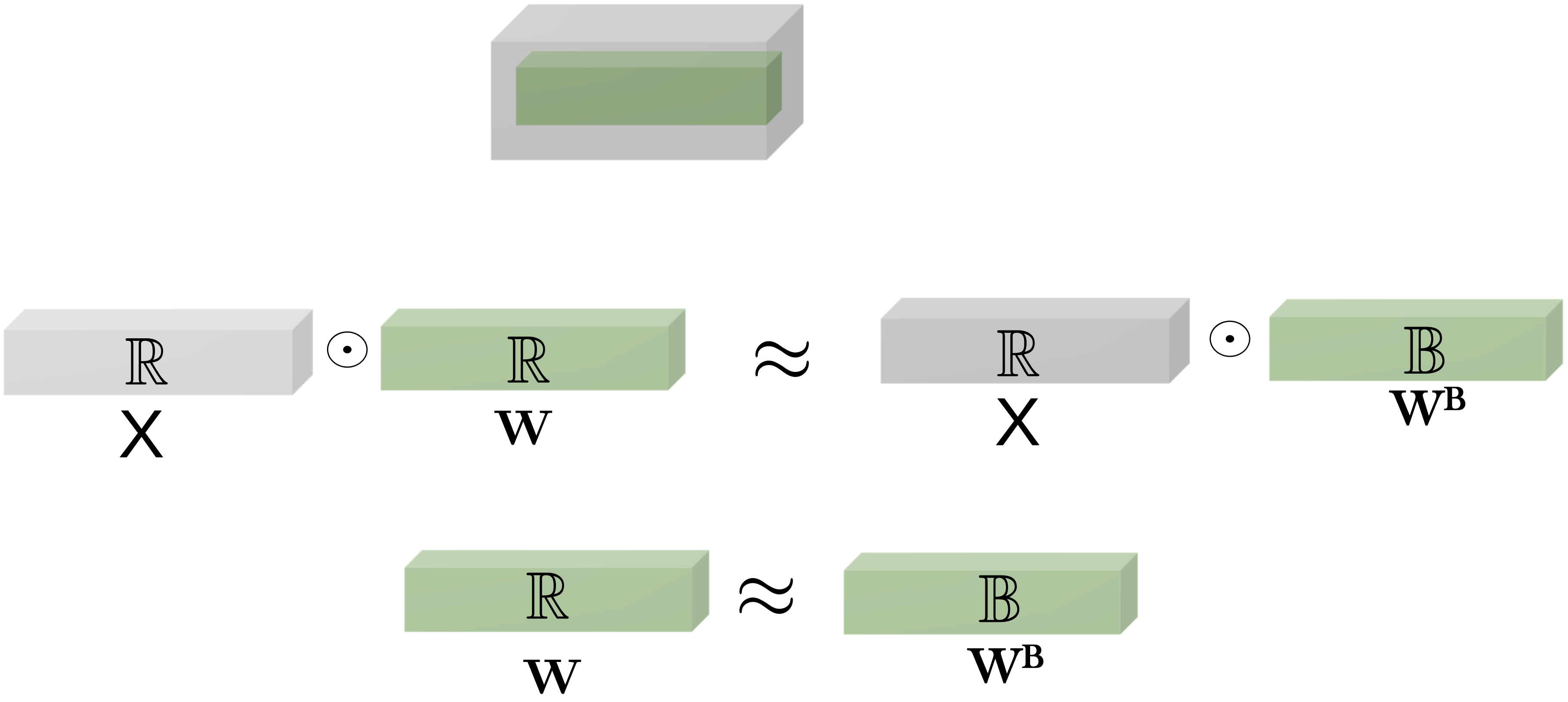
	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	$+ \; - \; \times$	1x	1x
$\mathbb{R} * \mathbb{B}$	$+ \; -$	~32x	~2x
$\mathbb{B} * \mathbb{B}$	XNOR Bit-count	~32x	~58x

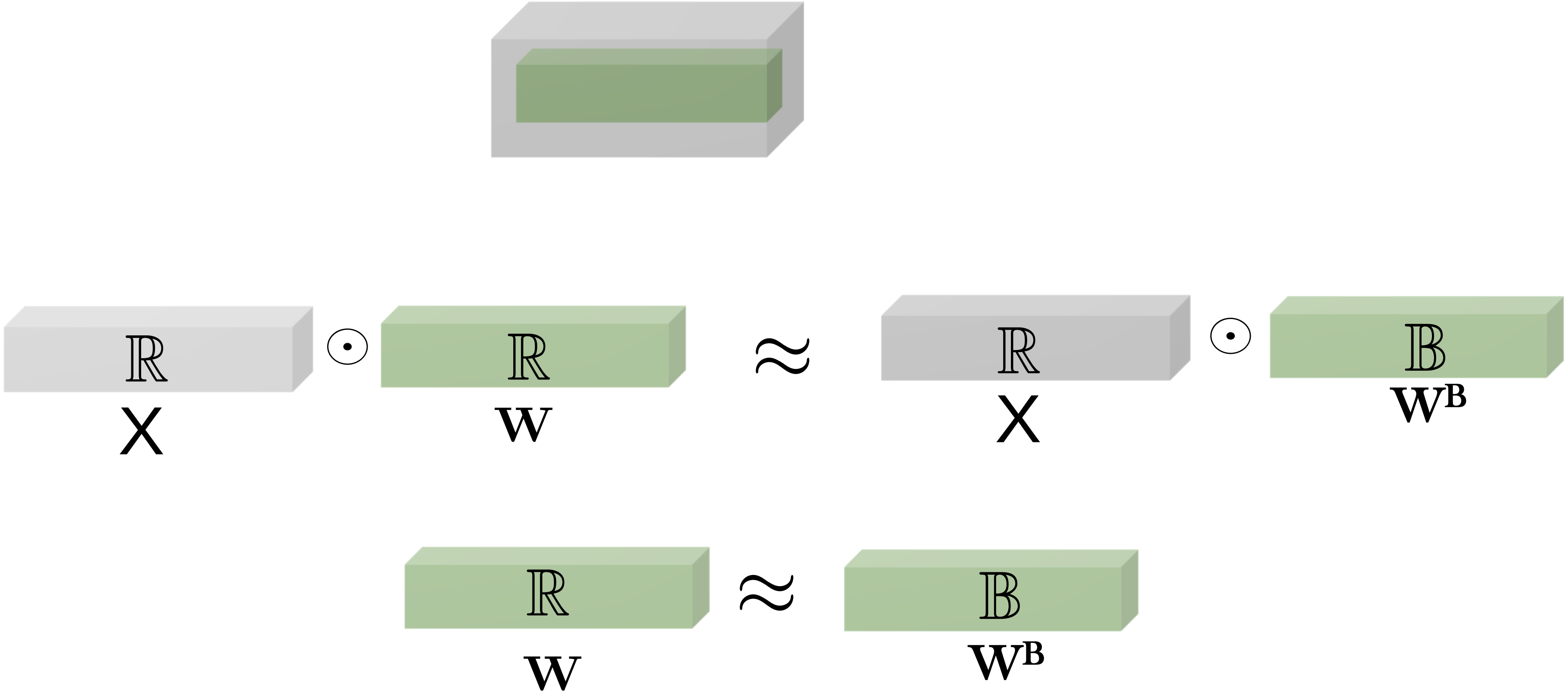

k


ℝ
⊙
ℝ
X
w



ℝ
⊙
ℝ
≈
ℝ
⊙
𝔹
X
W
X
W^B

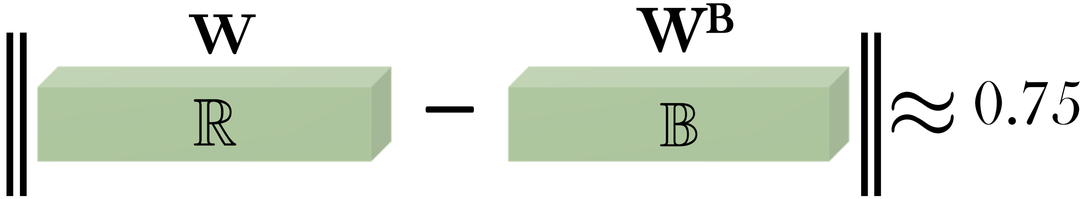

ℝ
X
⊙
ℝ
W
≈
ℝ
X
⊙
𝔹
W^B
ℝ
W
≈
𝔹
W^B



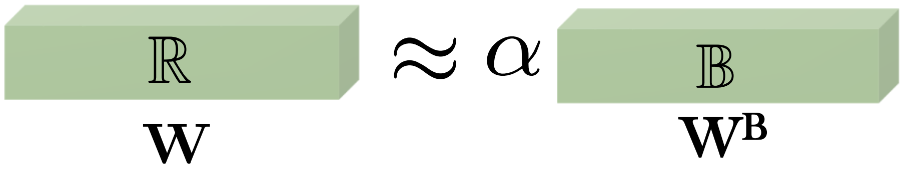
$$\mathbf{W}^{\mathbf{B}} = \operatorname{sign}(\mathbf{W})$$

Quantization Error

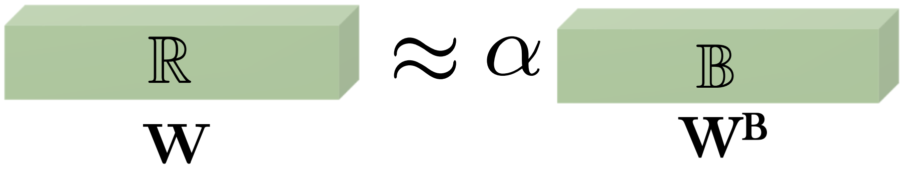
$$\mathbf{W}^{\mathrm{B}} = \operatorname{sign}(\mathbf{W})$$



Optimal Scaling Factor

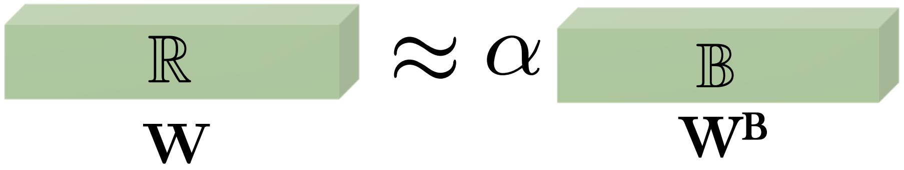


Optimal Scaling Factor



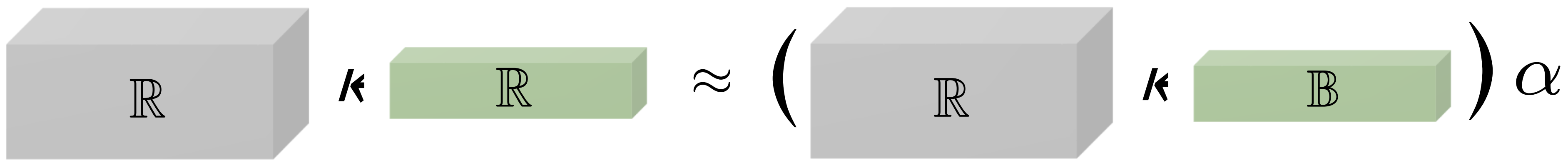
$$\alpha^*, \mathbf{W}^{\mathbf{B}^*} = \arg\min_{\mathbf{W}^{\mathbf{B}},\alpha} \{||\mathbf{W} - \alpha\mathbf{W}^{\mathbf{B}}||^{\mathbf{2}}\}$$

Optimal Scaling Factor

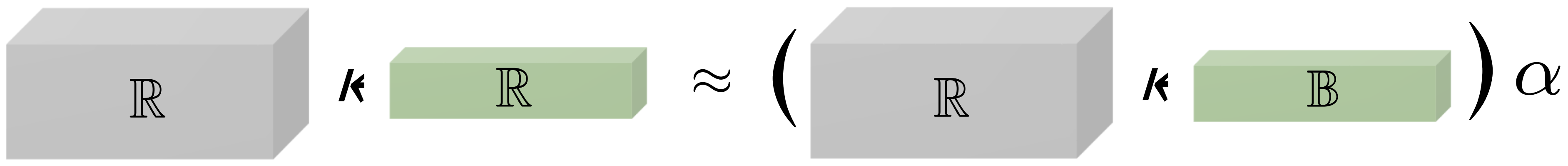


$$\alpha^*, \mathbf{W}^{\mathbf{B}^*} = \arg\min_{\mathbf{W}^{\mathbf{B}},\alpha} \{||\mathbf{W} - \alpha\mathbf{W}^{\mathbf{B}}||^{\mathbf{2}}\}$$

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


ℝ
ℝ
≈
ℝ
𝔹
α

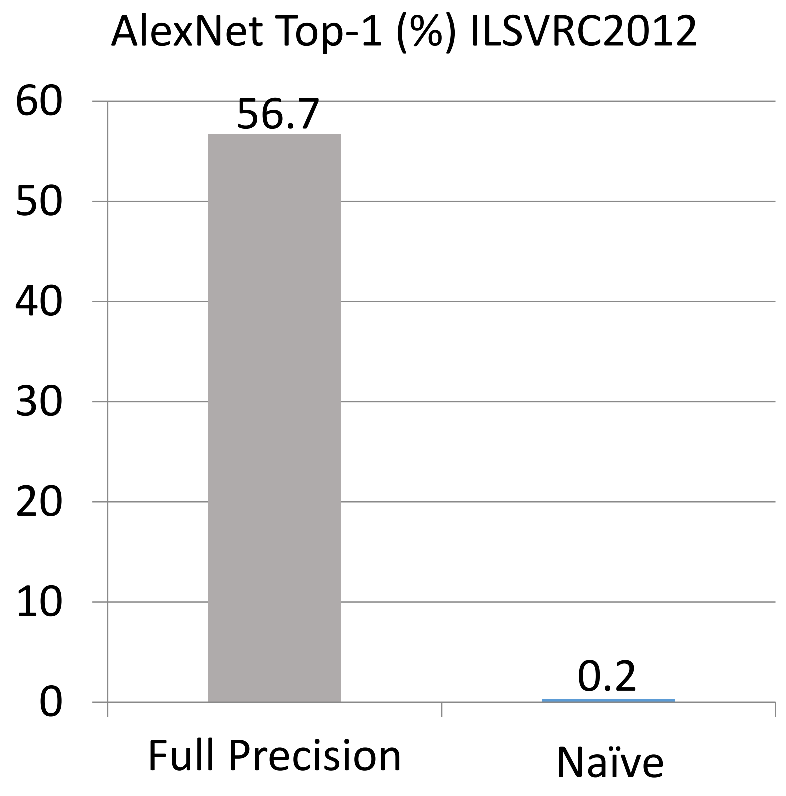
How to train a CNN with binary filters?



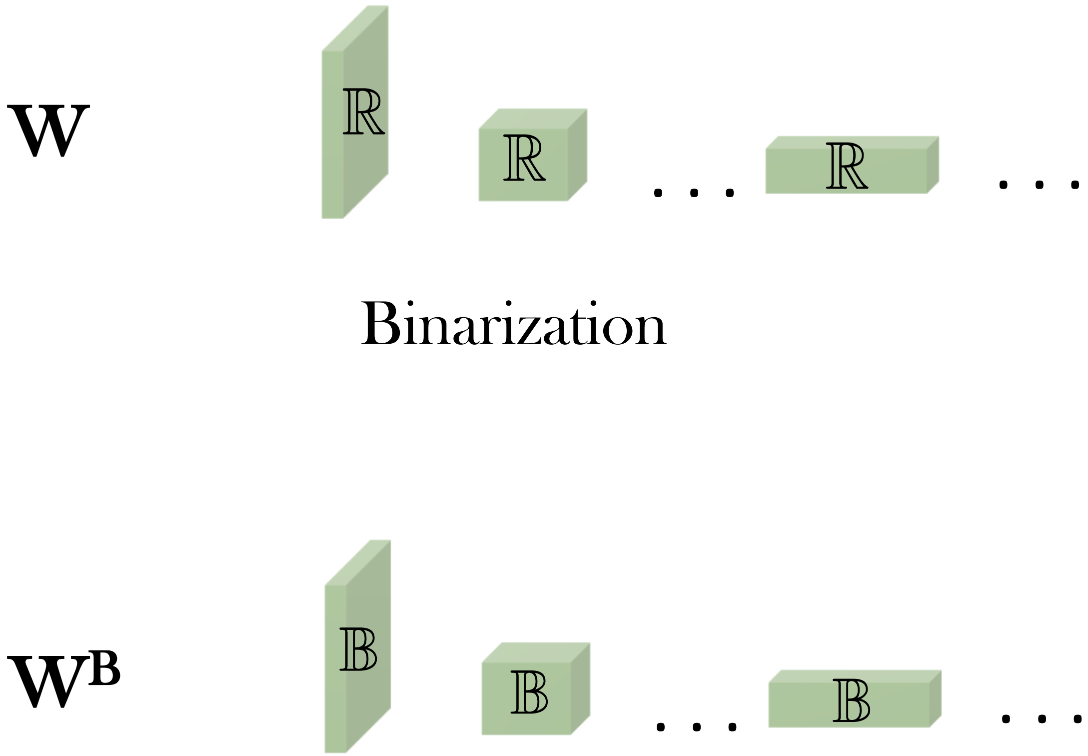
Training Binary Weight Networks

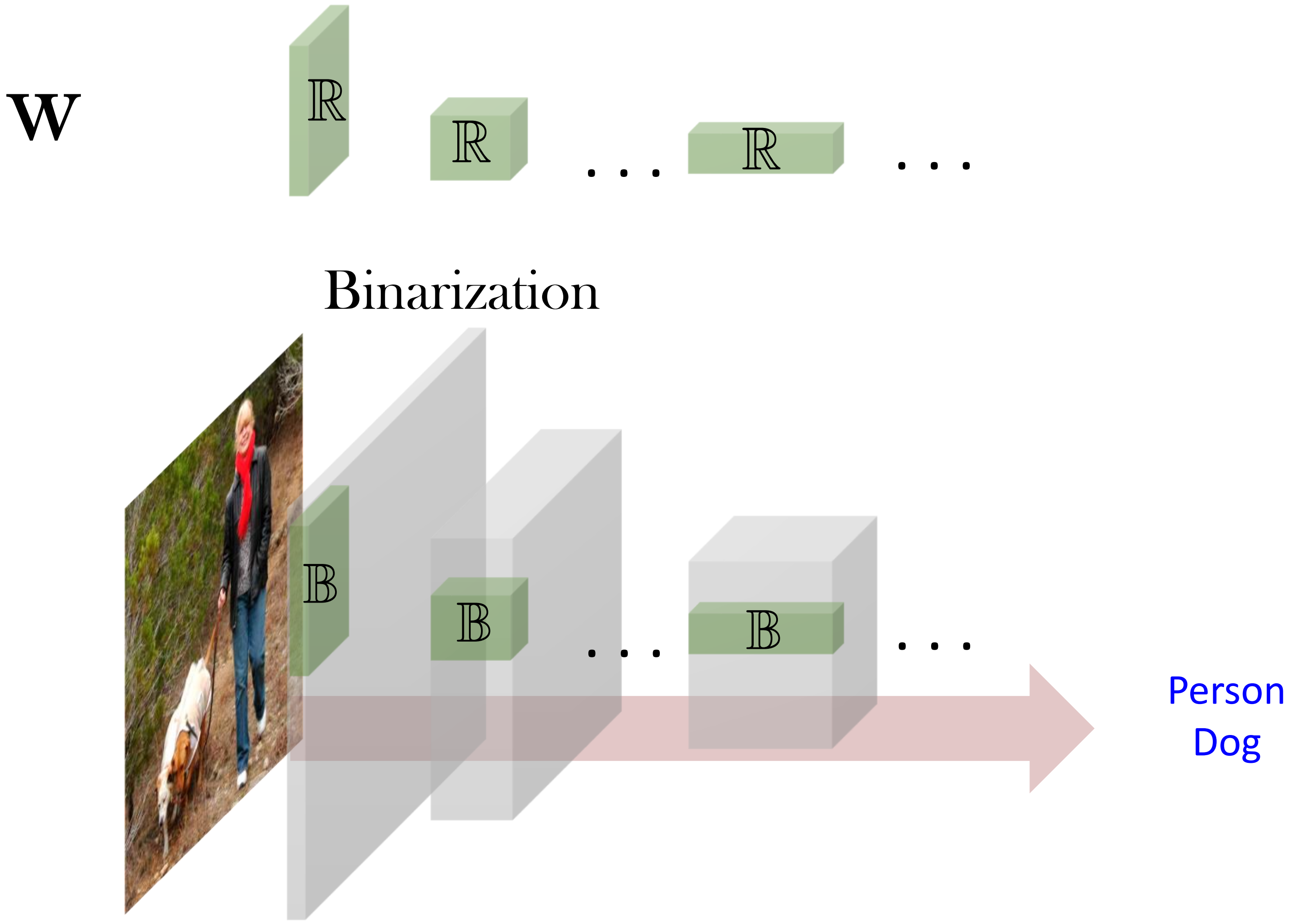
Naive Solution:

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


AlexNet Top-1 (%) ILSVRC2012
60
50
40
30
20
10
0
56.7
0.2
Full Precision
Naïve

W


W
ℝ
ℝ
. . .
ℝ
. . .
Binarization
W^B
𝔹
𝔹
. . .
𝔹
. . .

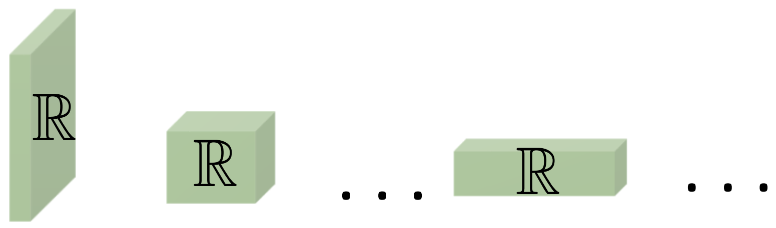


W
ℝ
ℝ
. . .
ℝ
. . .
Binarization
𝔹
𝔹
. . .
𝔹
. . .
Person
Dog

Binary Weight Network

Train for binary weights:

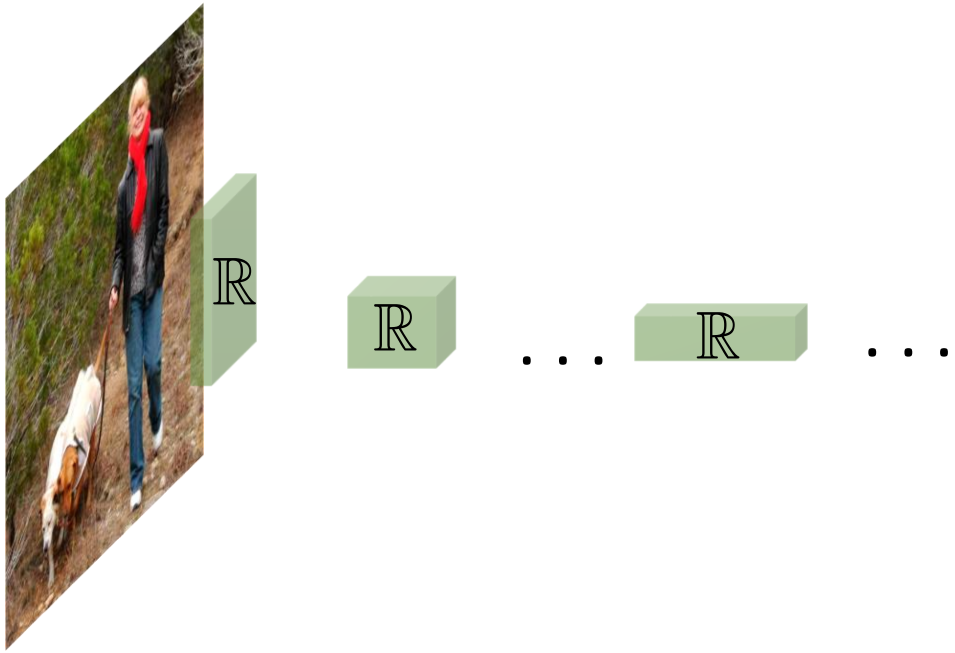
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \mathrm{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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



Binary Weight Network

Train for binary weights:

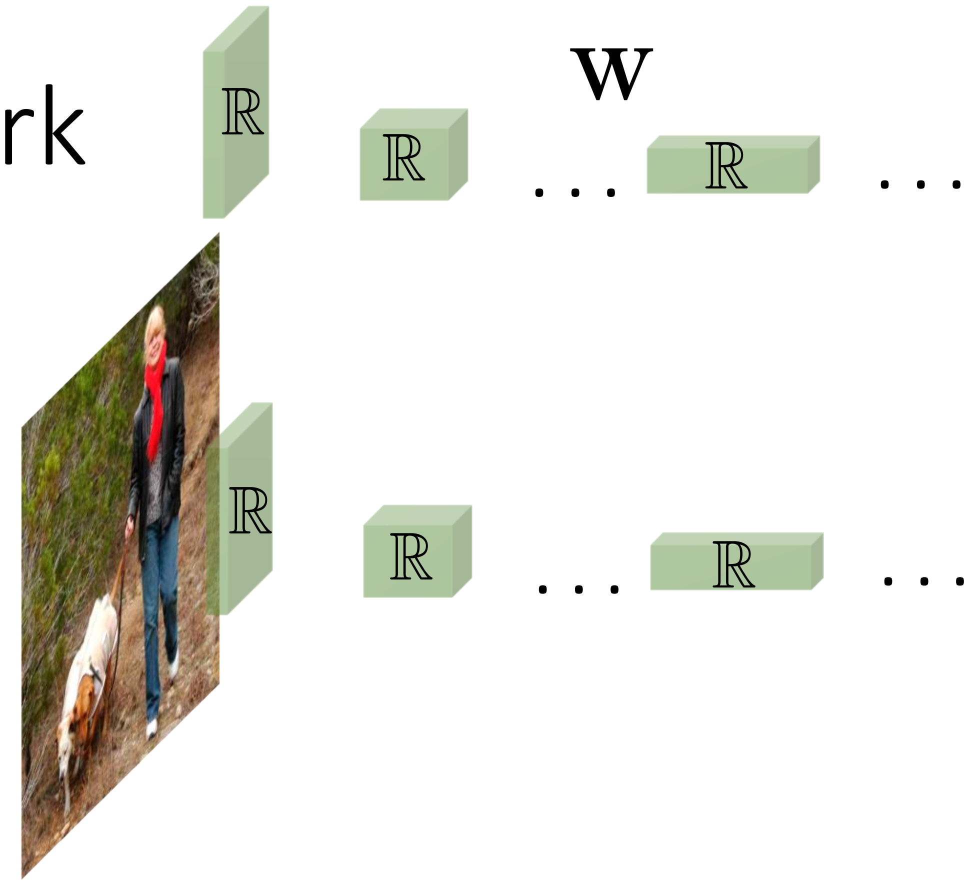
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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



Binary Weight Network

Train for binary weights:

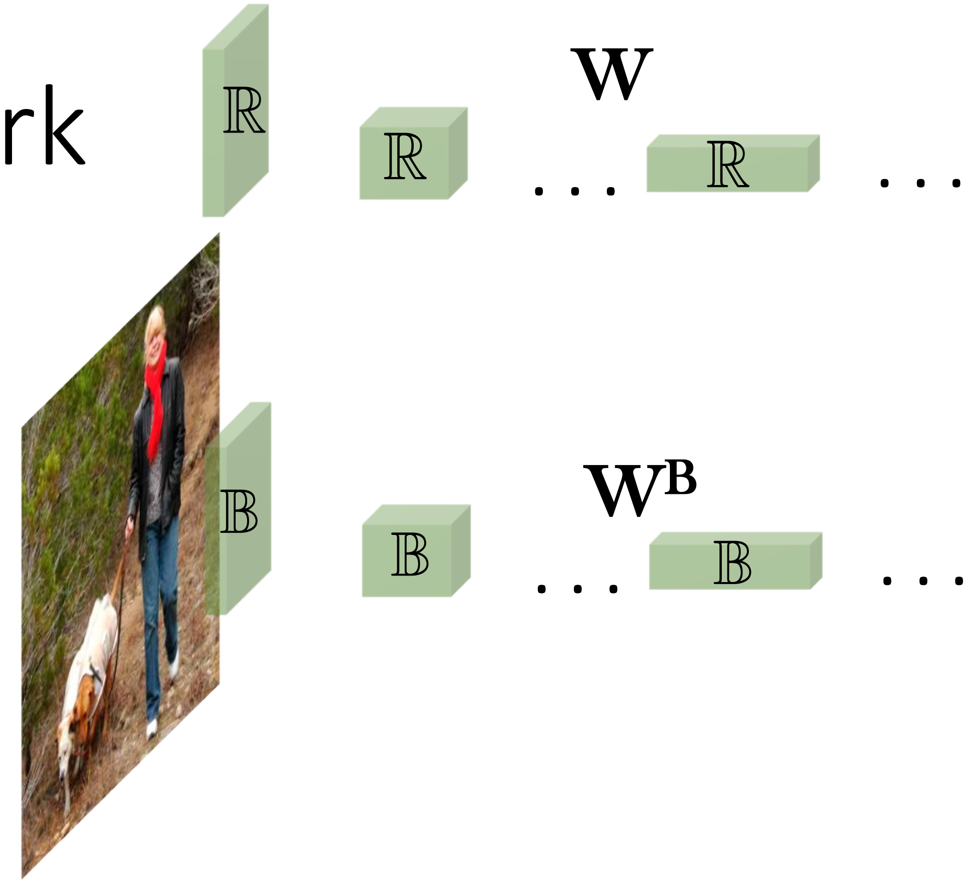
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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



Binary Weight Network

Train for binary weights:

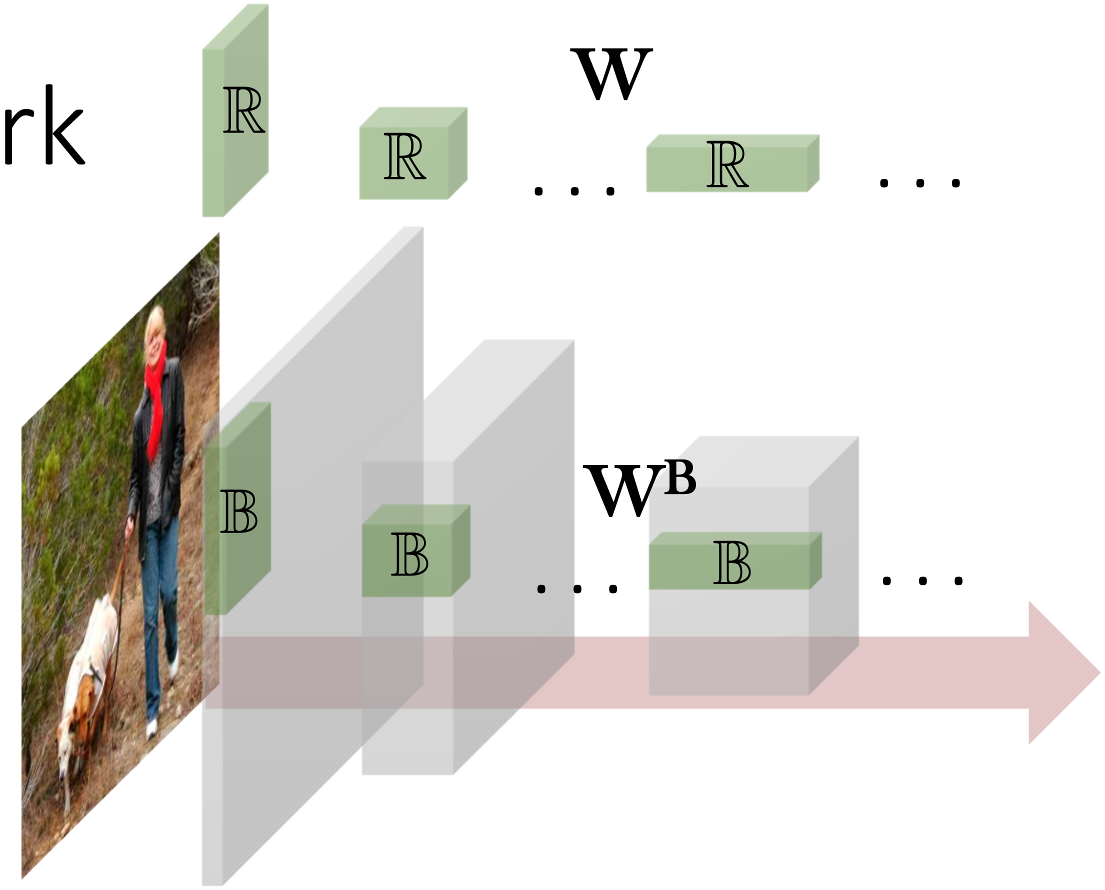
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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



Binary Weight Network

Train for binary weights:

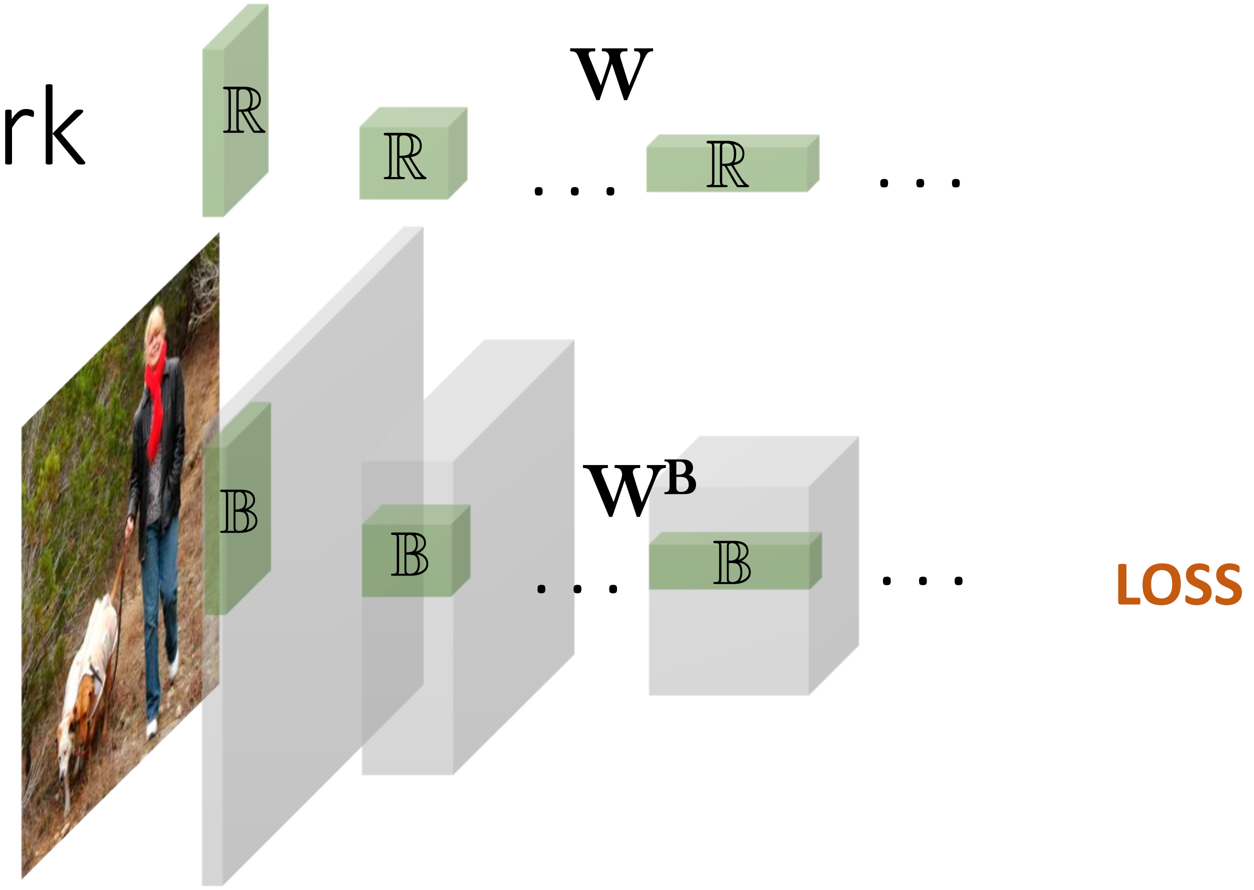
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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



Binary Weight Network

Train for binary weights:

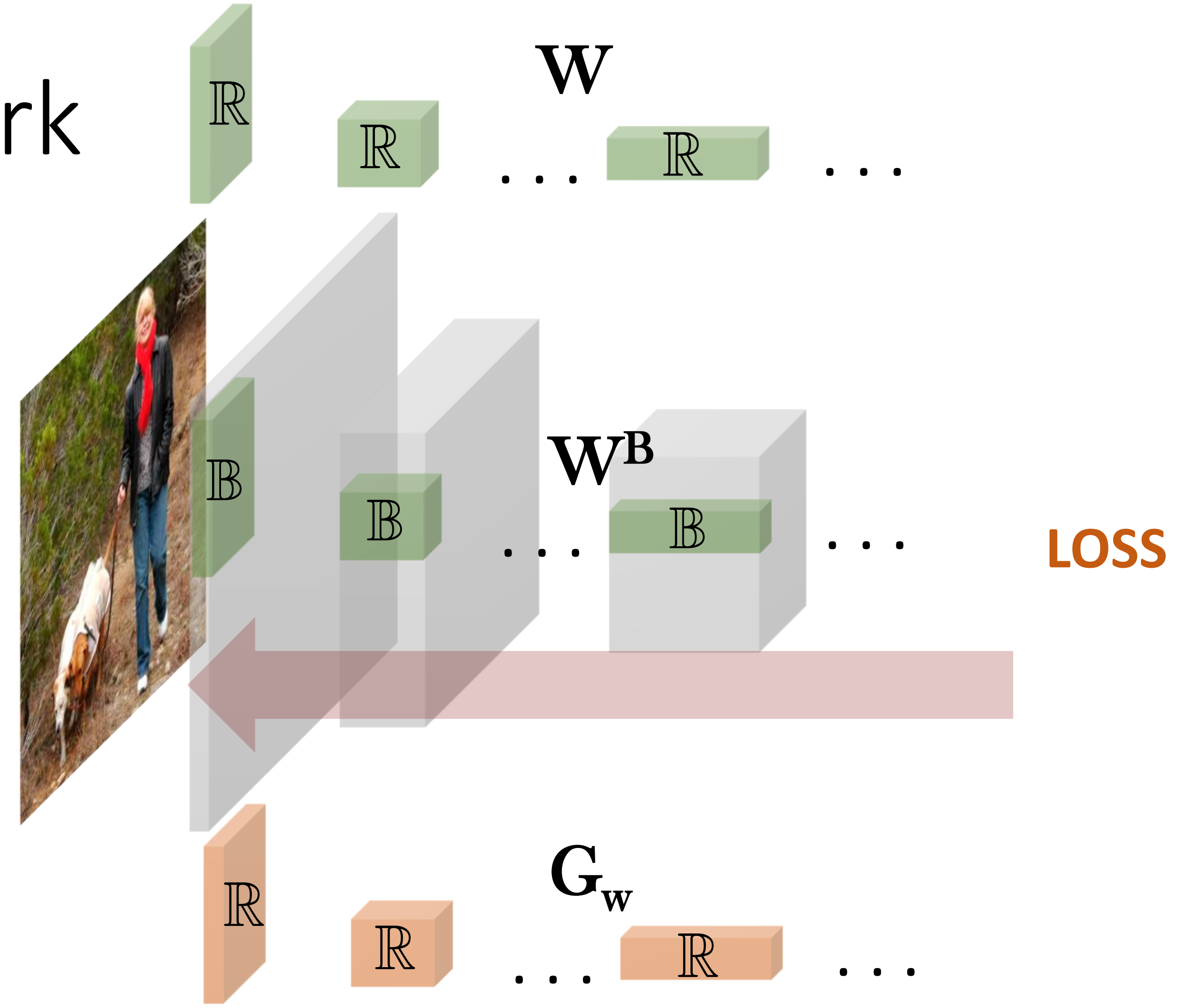
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W^B} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W^B}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial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



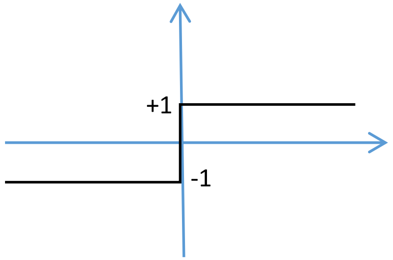
Binary Weight Network

Train for binary weights:

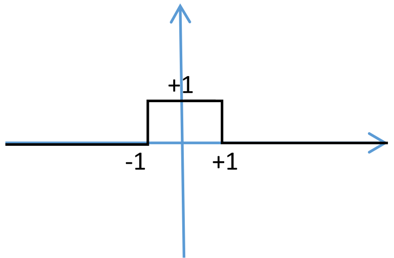
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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



sign(x) →



G_x →

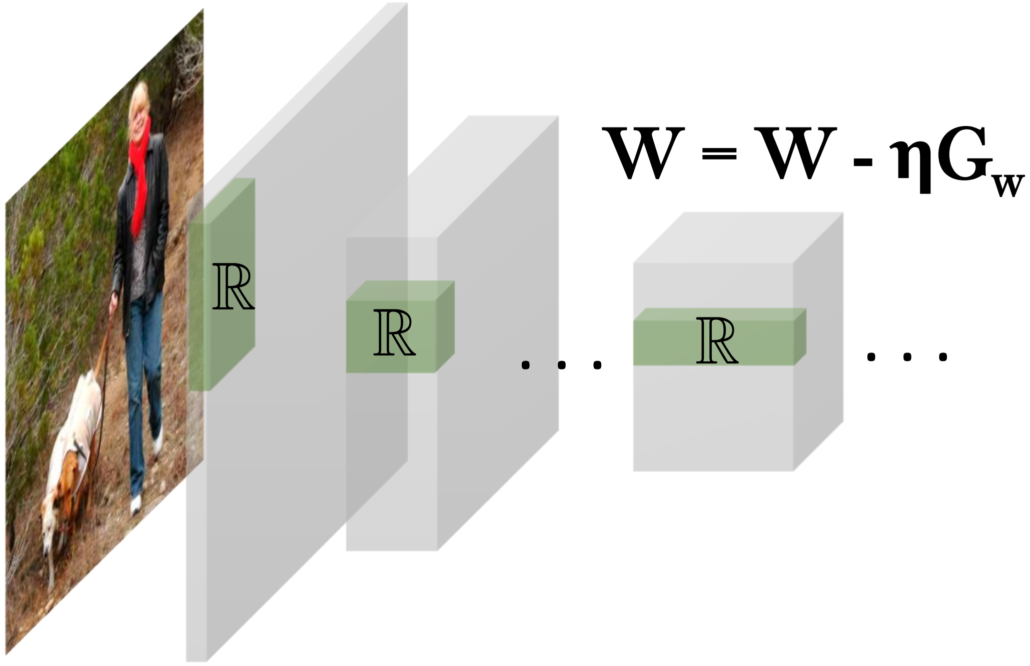


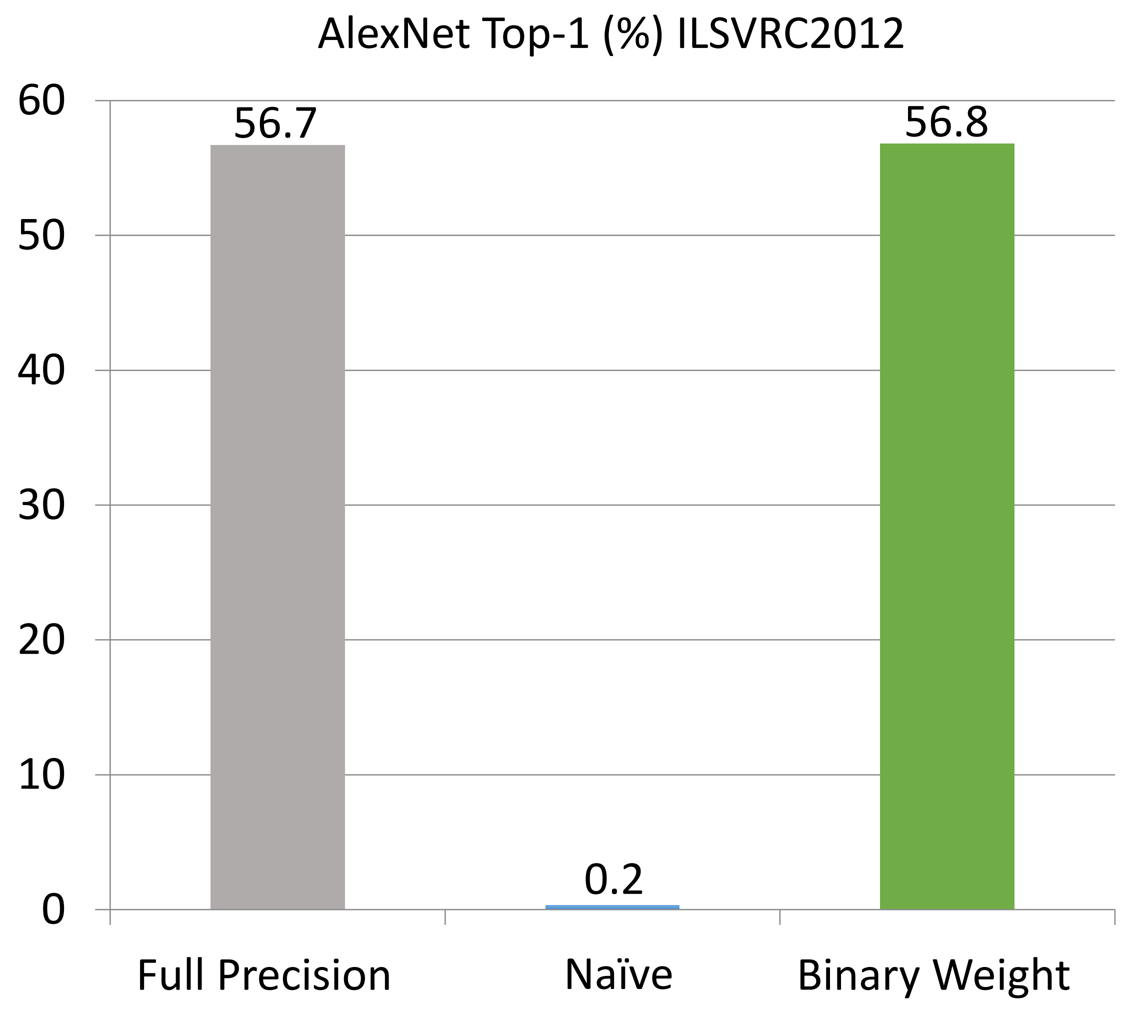
[Hinton et al. 2012]

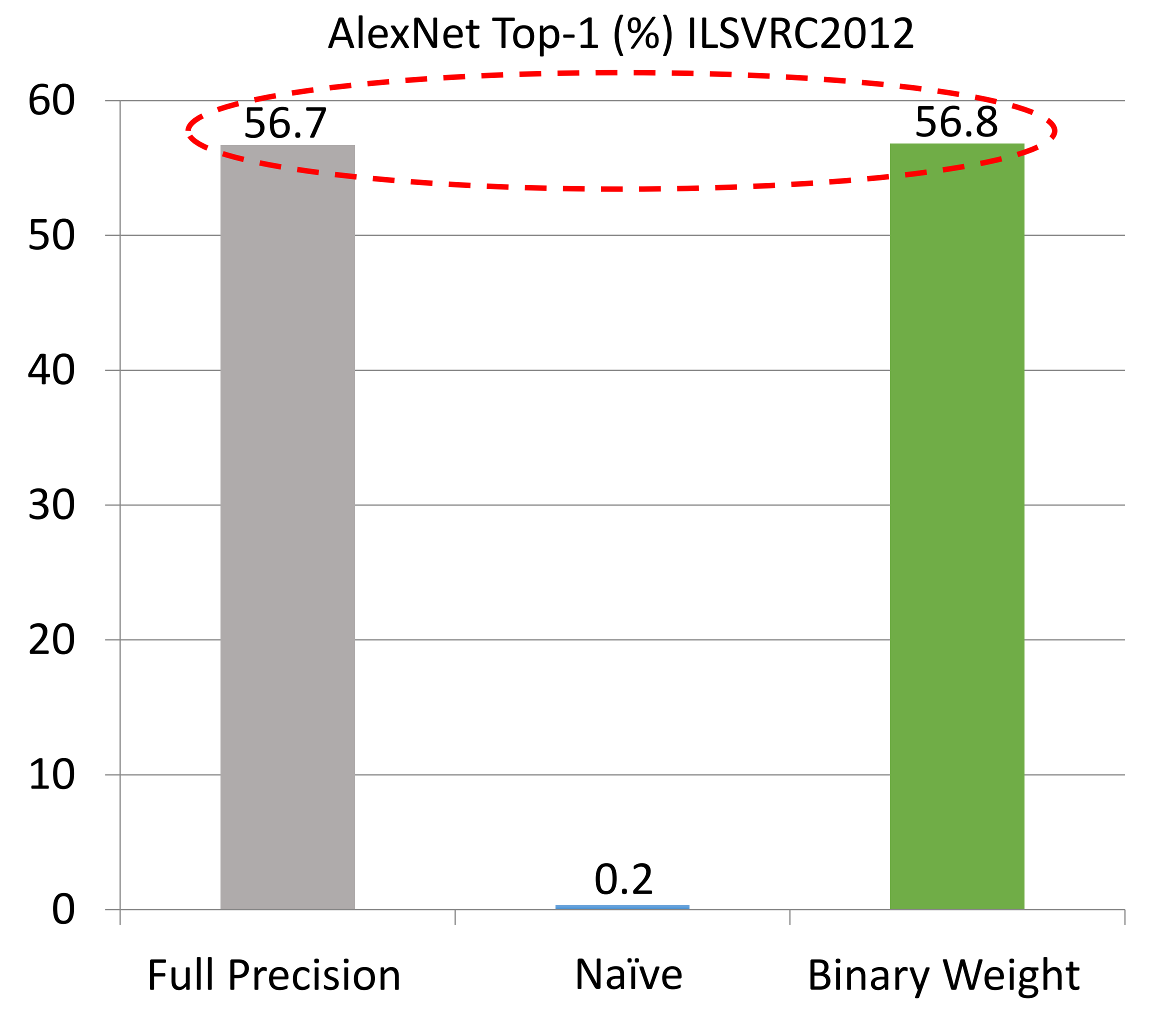
Binary Weight Network

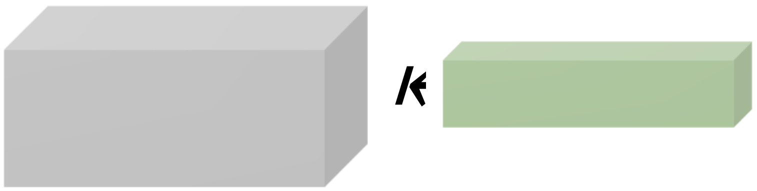
Train for binary weights:

1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W^B} = \text{sign}(\mathbf{W})$
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 $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$

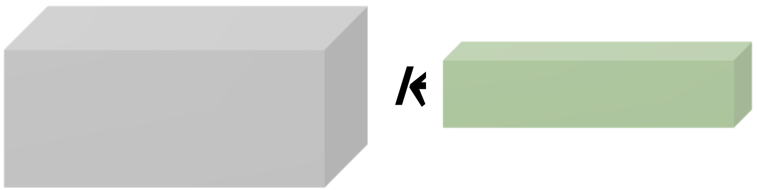


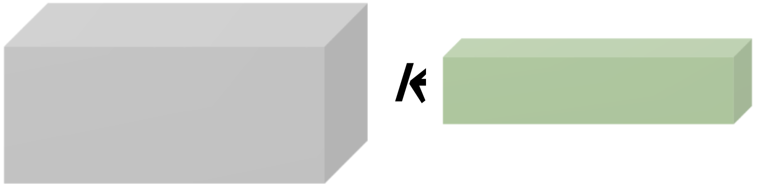

AlexNet Top-1 (%) ILSVRC2012
60
50
40
30
20
10
0
56.7
0.2
56.8
Full Precision
Naïve
Binary Weight


AlexNet Top-1 (%) ILSVRC2012
60
50
40
30
20
10
0
56.7
0.2
56.8
Full Precision
Naïve
Binary Weight

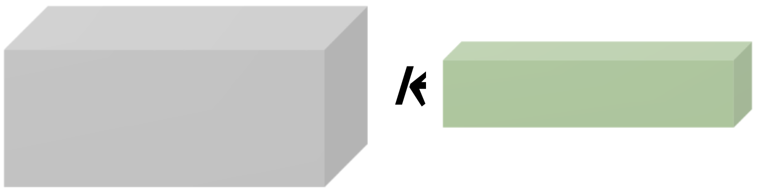


	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	+ − ×	1x	1x
$\mathbb{R} * \mathbb{B}$	+ −	~32x	~2x
$\mathbb{B} * \mathbb{B}$	XNOR Bit-count	~32x	~58x

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



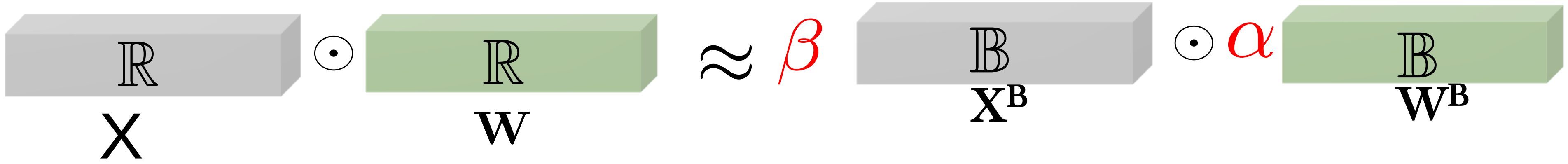
	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	+ − ×	1x	1x
$\mathbb{R} * \mathbb{B}$	+ −	~32x	~2x
$\mathbb{B} * \mathbb{B}$	XNOR Bit-count	~32x	~58x



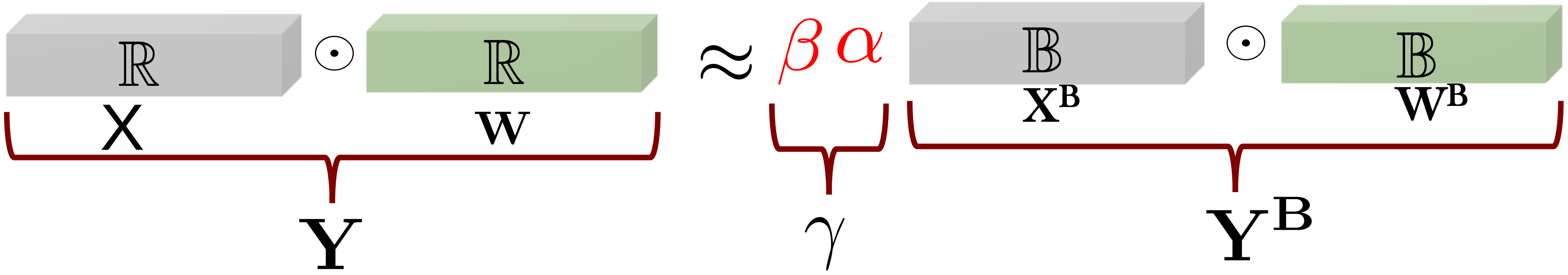
	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	$+ \; - \; \times$	1x	1x
$\mathbb{R} * \mathbb{B}$	$+ \; -$	~32x	~2x
$\mathbb{B} * \mathbb{B}$ XNOR-Networks	XNOR Bit-count	~32x	~58x

Binary Input and Binary Weight (XNOR-Net)

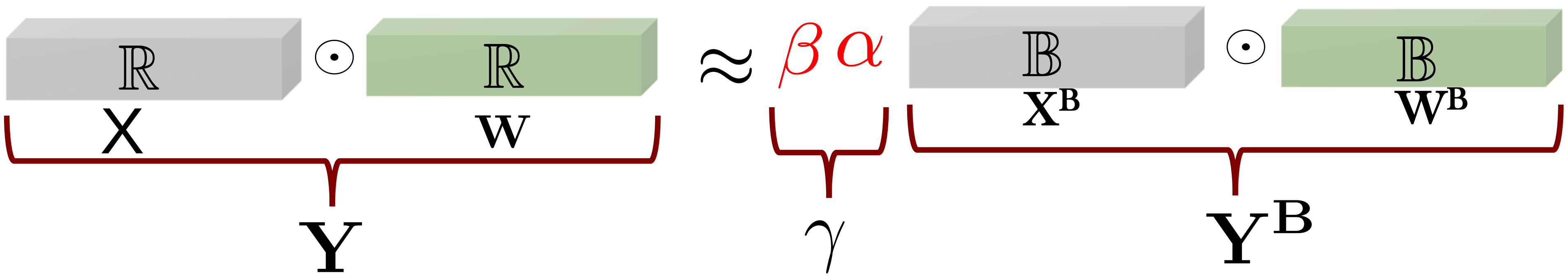
Binary Input and Binary Weight (XNOR-Net)



Binary Input and Binary Weight (XNOR-Net)



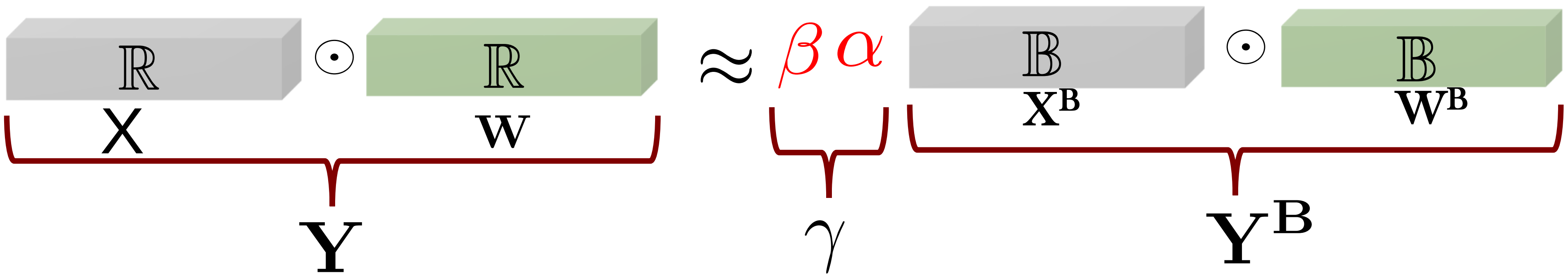
Binary Input and Binary Weight (XNOR-Net)



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

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

Binary Input and Binary Weight (XNOR-Net)

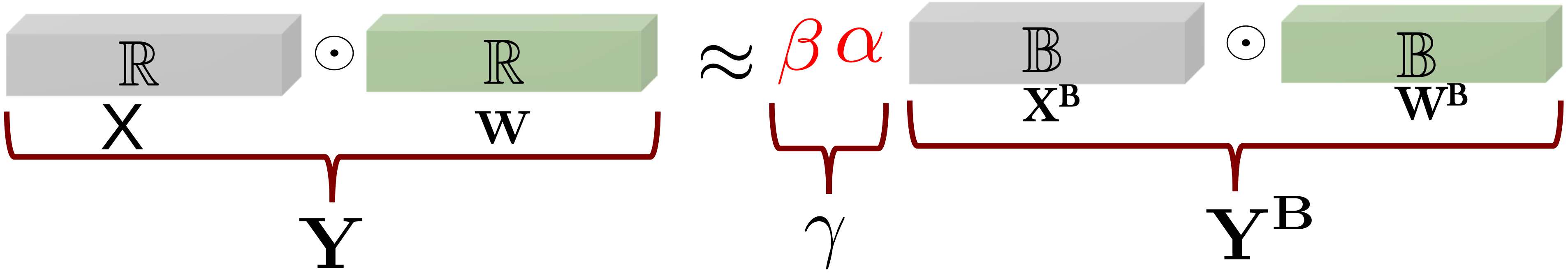


$$\mathbf{Y} \approx \gamma \, \mathbf{Y^B}$$

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

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

Binary Input and Binary Weight (XNOR-Net)



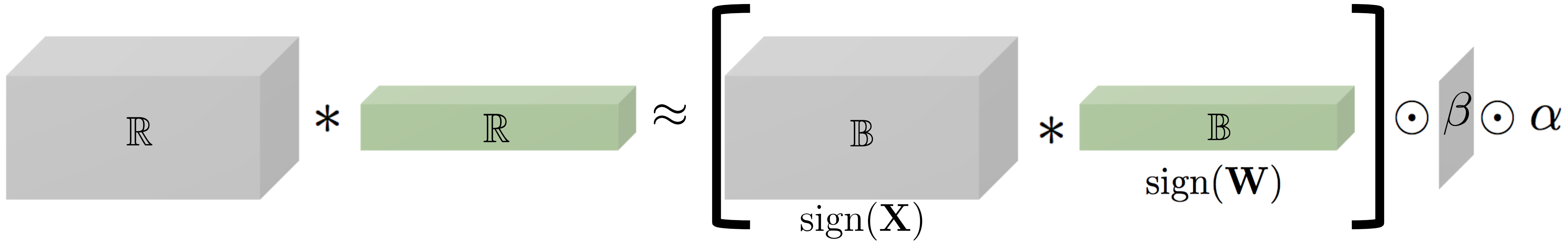
$$\mathbf{Y} \approx \gamma \, \mathbf{Y^B}$$

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

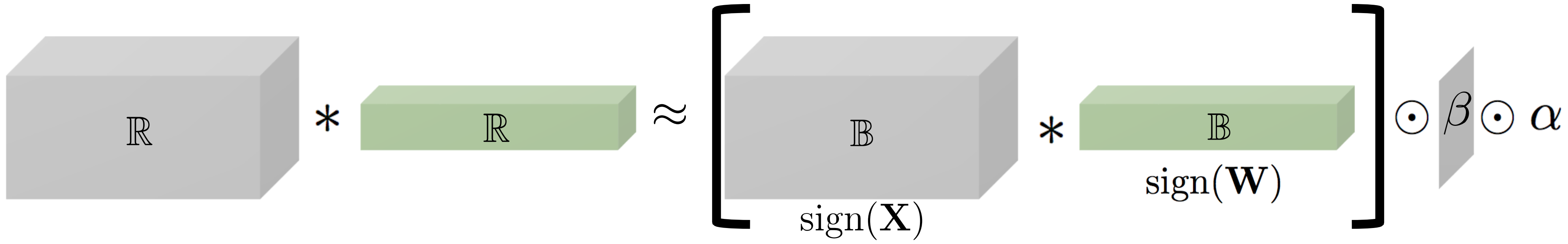
$$\mathbf{Y^B}^* = \text{sign}(\mathbf{Y}) \quad \gamma^* = \frac{1}{n}\|\mathbf{Y}\|_{\ell 1}$$

$$\mathbf{X^B}^* = \text{sign}(\mathbf{X}) \quad \mathbf{W^B}^* = \text{sign}(\mathbf{W})$$

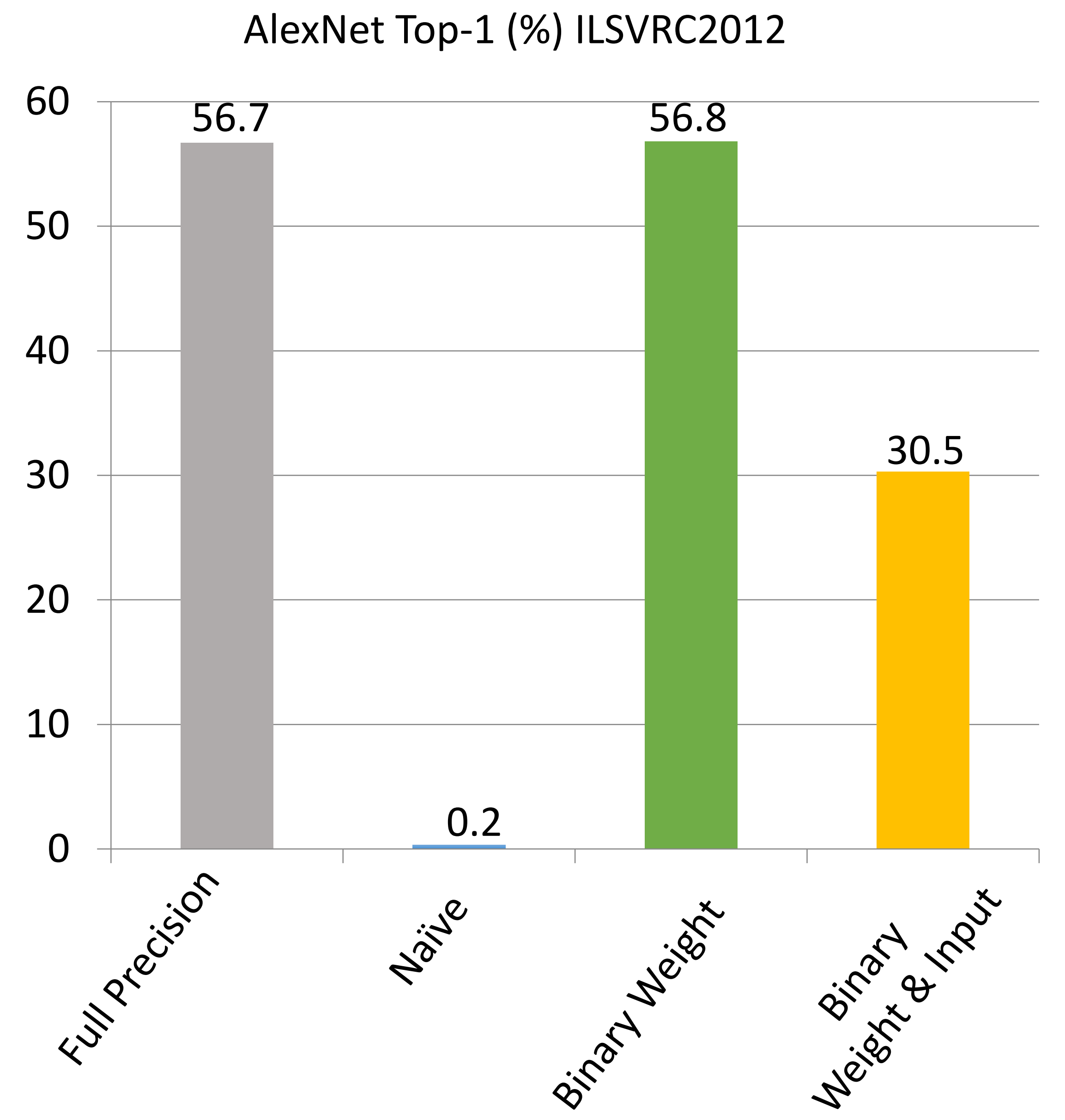
$$\alpha^* = \frac{1}{n}\|\mathbf{W}\|_{\ell 1} \qquad \beta^* = \frac{1}{n}\|\mathbf{X}\|_{\ell 1}$$

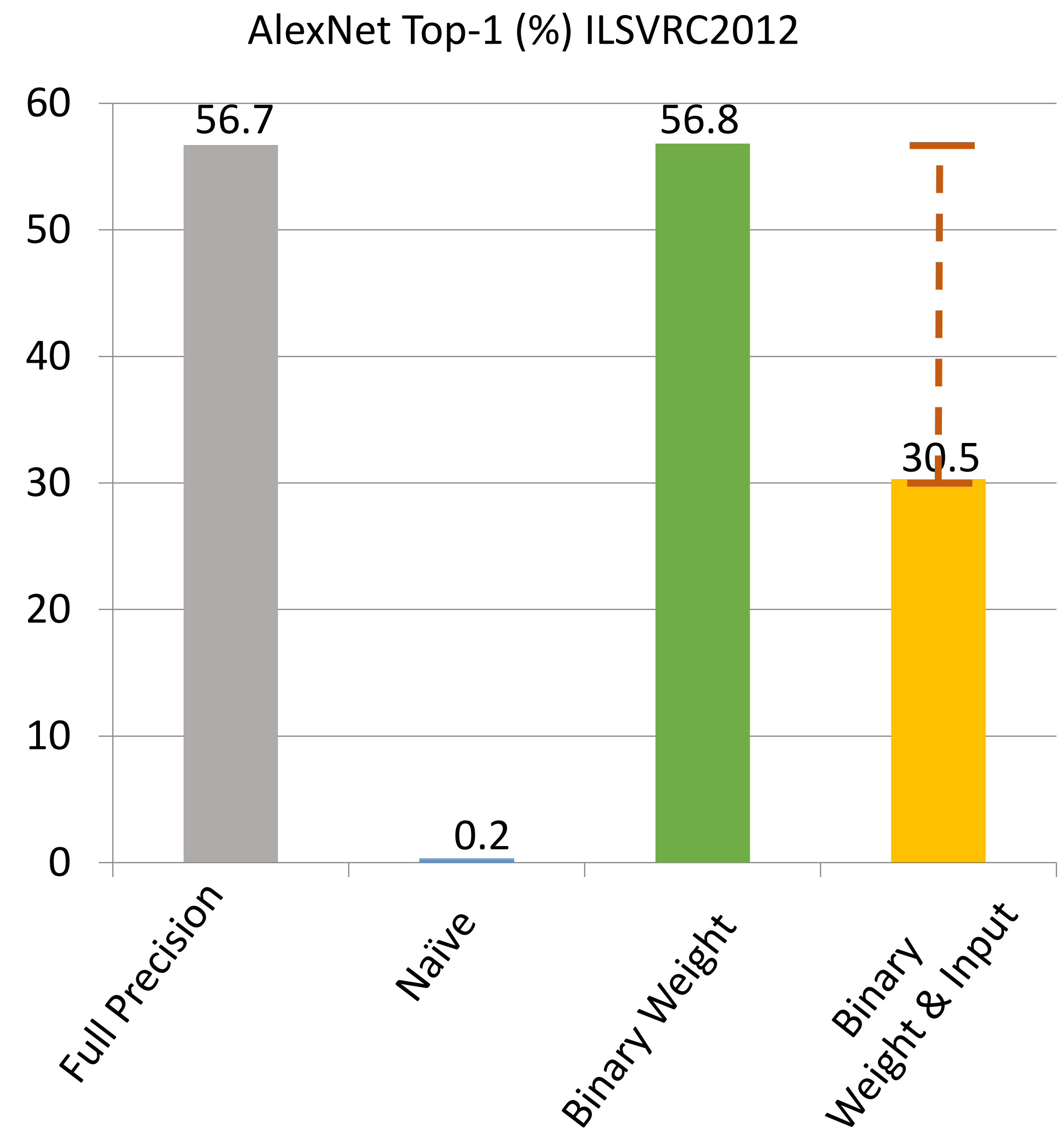


ℝ
*
ℝ
≈
𝔹
*
𝔹
sign(X)
sign(W)
⊙ β ⊙ α

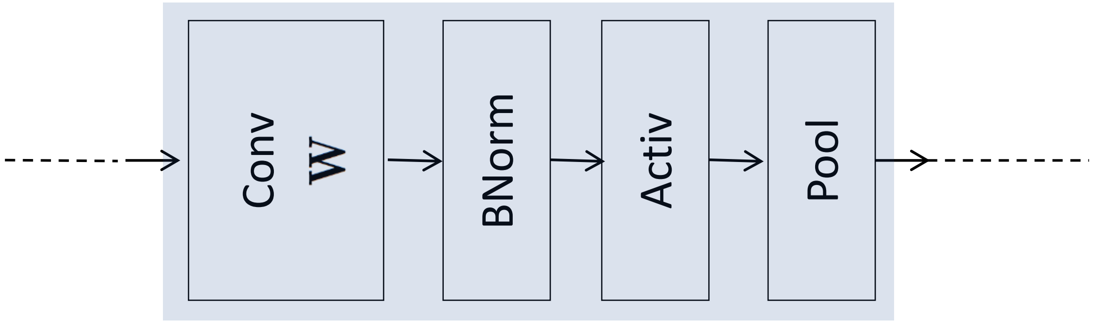


1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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


AlexNet Top-1 (%) ILSVRC2012
60
50
40
30
20
10
0
56.7
0.2
56.8
30.5
Full Precision
Naïve
Binary Weight
Binary
Weight & Input

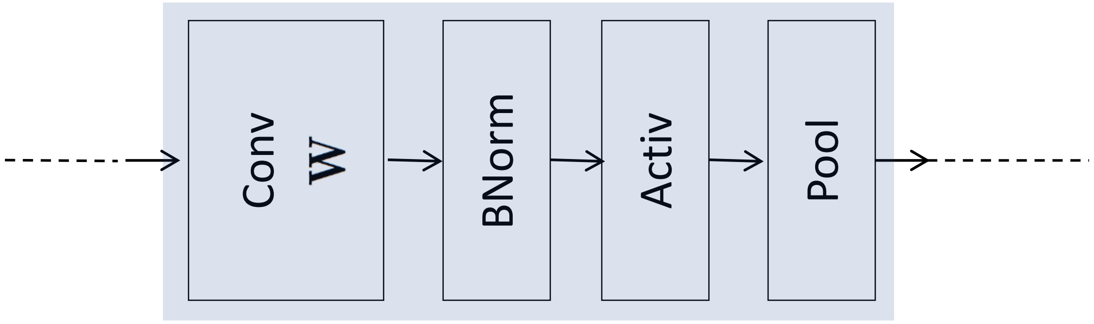

AlexNet Top-1 (%) ILSVRC2012
60
50
40
30
20
10
0
56.7
0.2
56.8
30.5
Full Precision
Naïve
Binary Weight
Binary
Weight & Input

Network Structure in XNOR-Networks

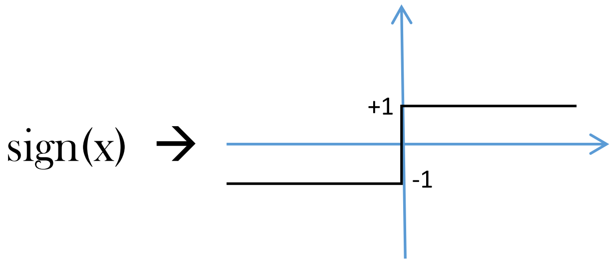


A typical block in CNN

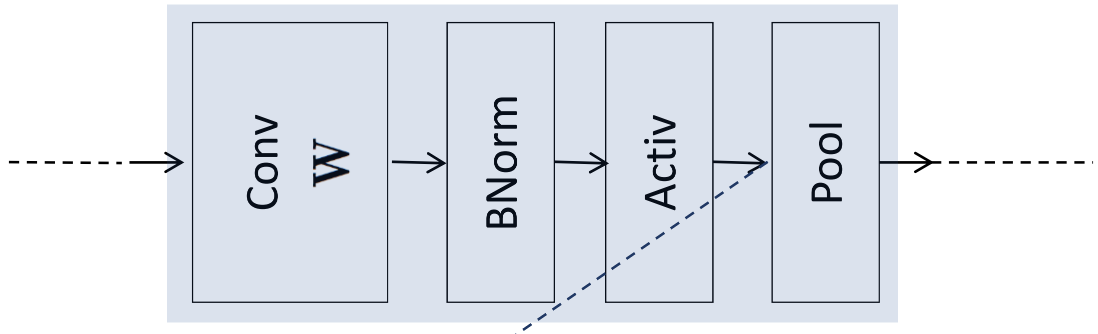
Network Structure in XNOR-Networks



A typical block in CNN

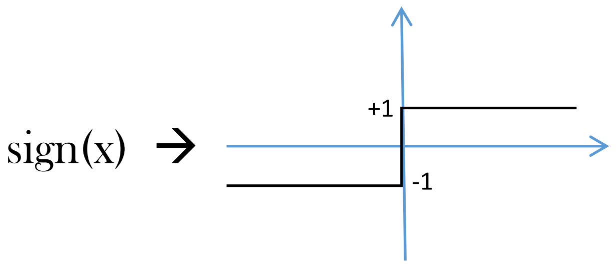


Network Structure in XNOR-Networks

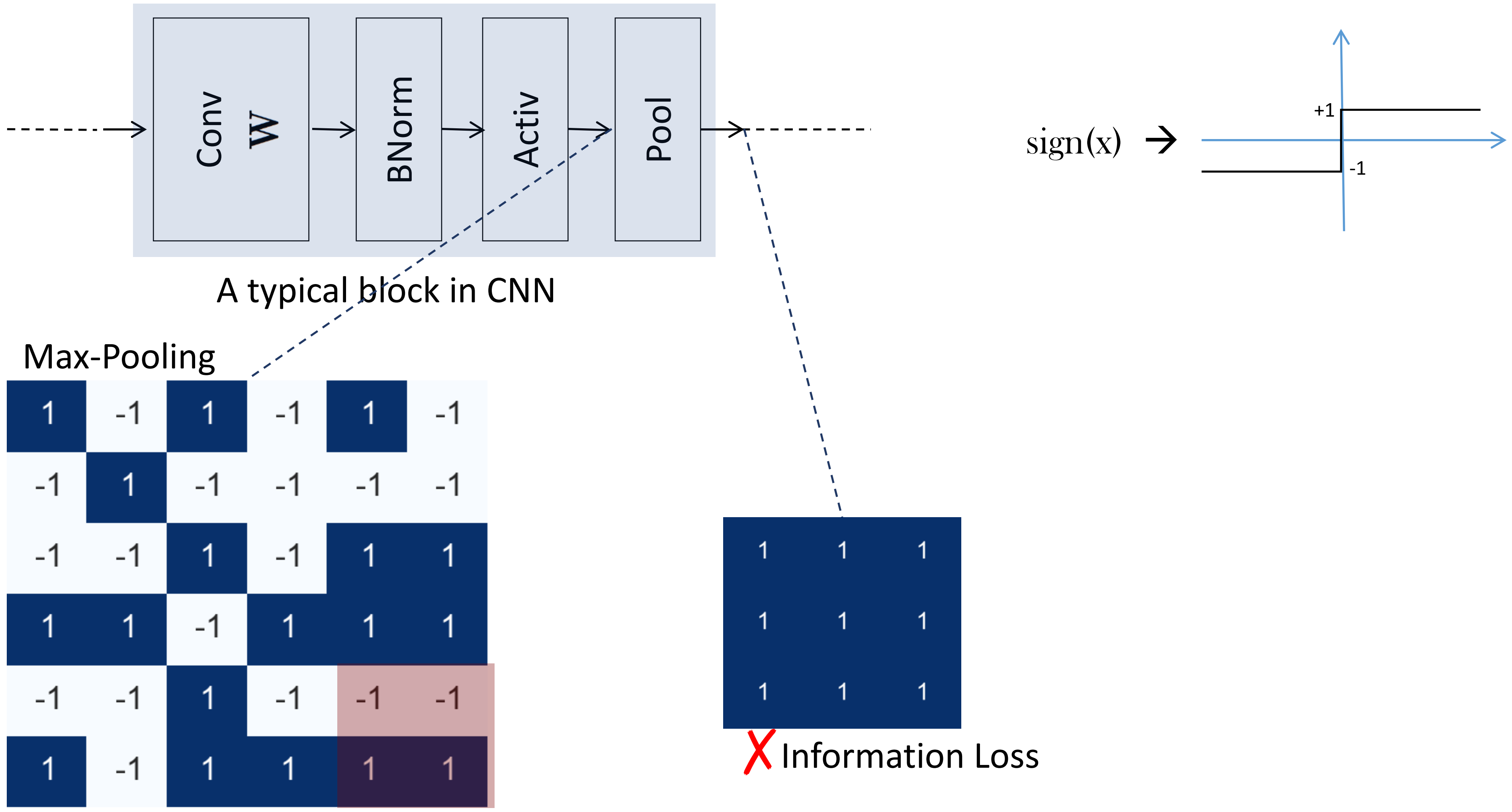


A typical block in CNN

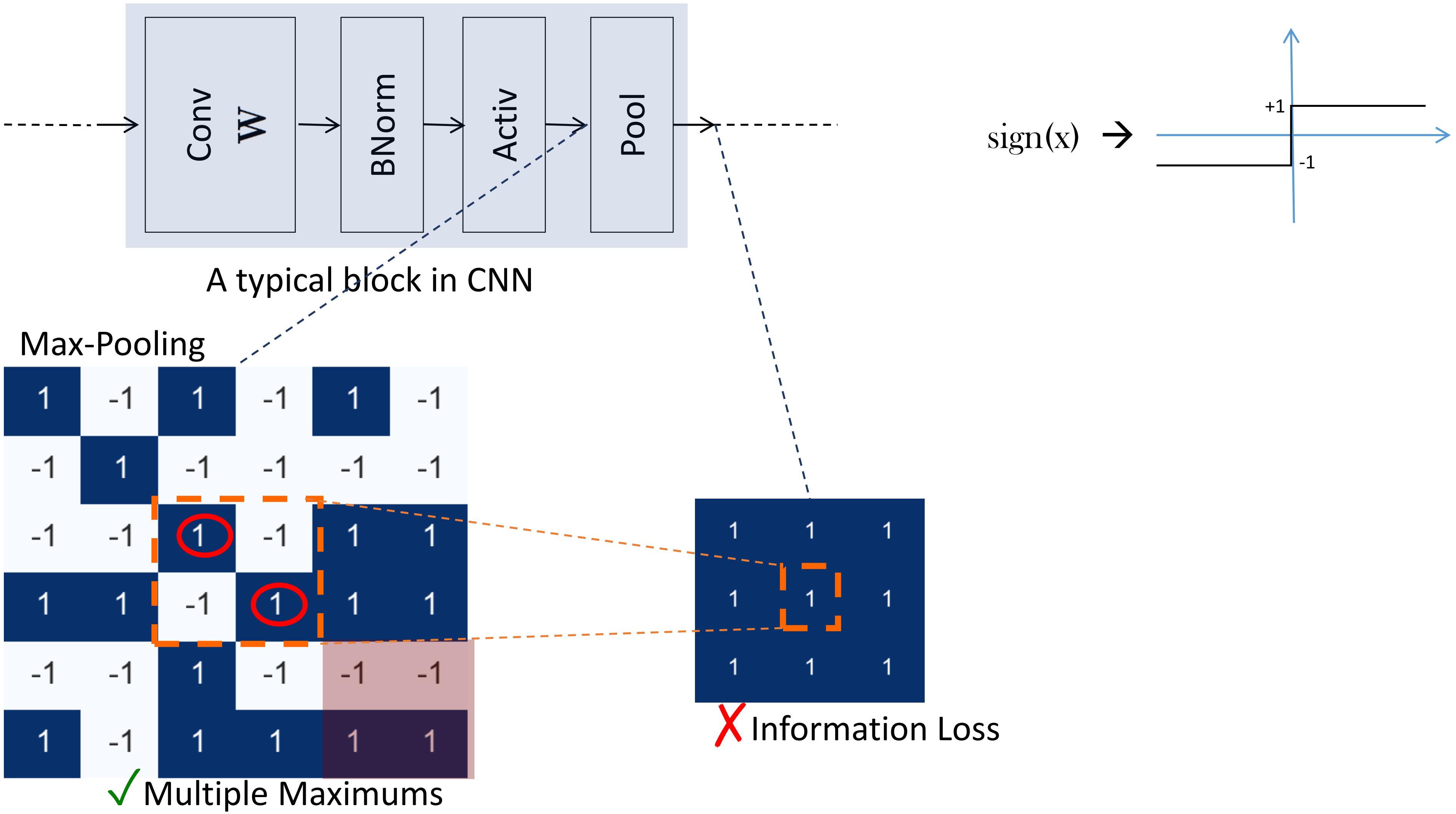
1	-1	1	-1	1	-1
-1	1	-1	-1	-1	-1
-1	-1	1	-1	1	1
1	1	-1	1	1	1
-1	-1	1	-1	-1	-1
1	-1	1	1	1	1



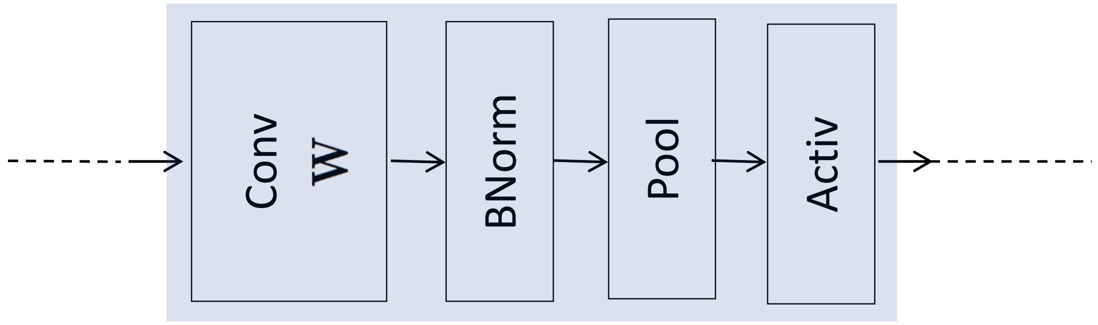
Network Structure in XNOR-Networks



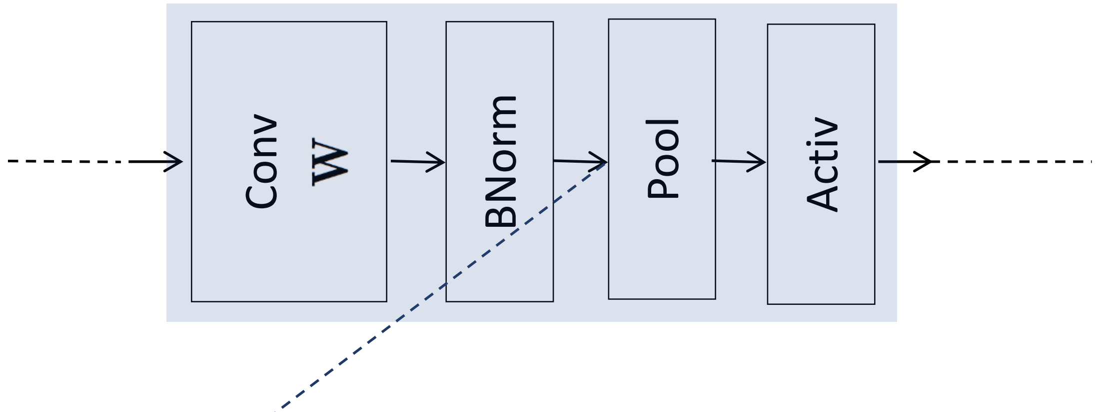
Network Structure in XNOR-Networks



Network Structure in XNOR-Networks

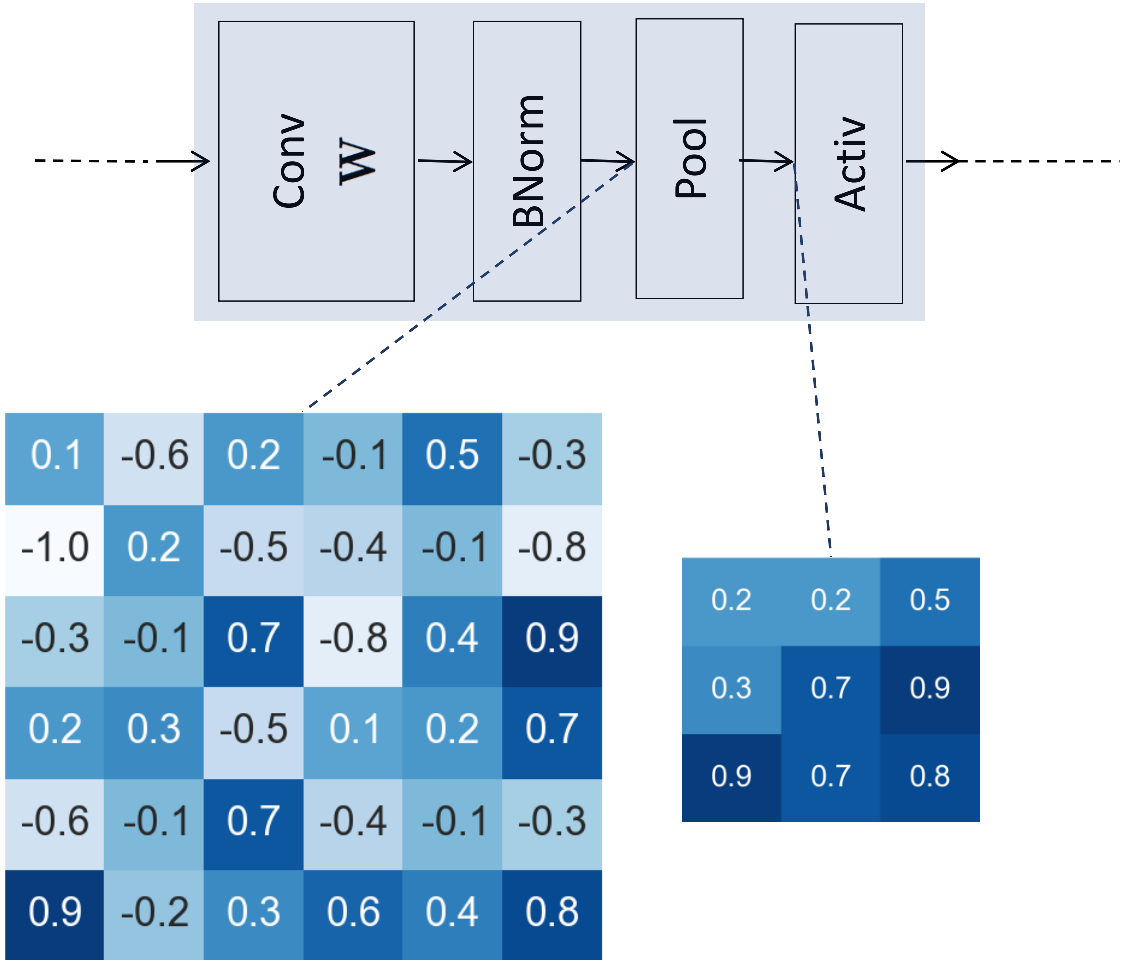


Network Structure in XNOR-Networks

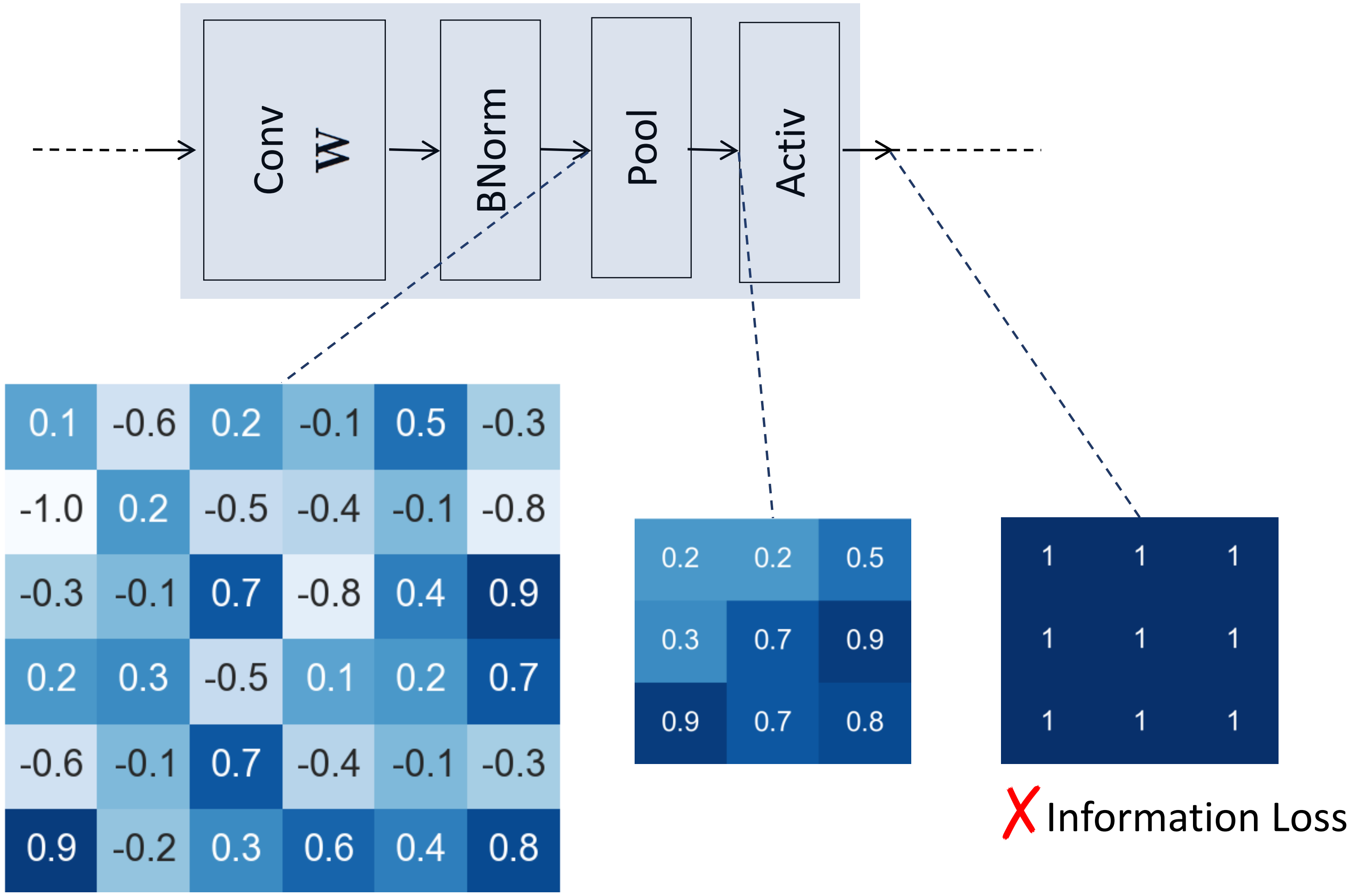


0.1	-0.6	0.2	-0.1	0.5	-0.3
-1.0	0.2	-0.5	-0.4	-0.1	-0.8
-0.3	-0.1	0.7	-0.8	0.4	0.9
0.2	0.3	-0.5	0.1	0.2	0.7
-0.6	-0.1	0.7	-0.4	-0.1	-0.3
0.9	-0.2	0.3	0.6	0.4	0.8

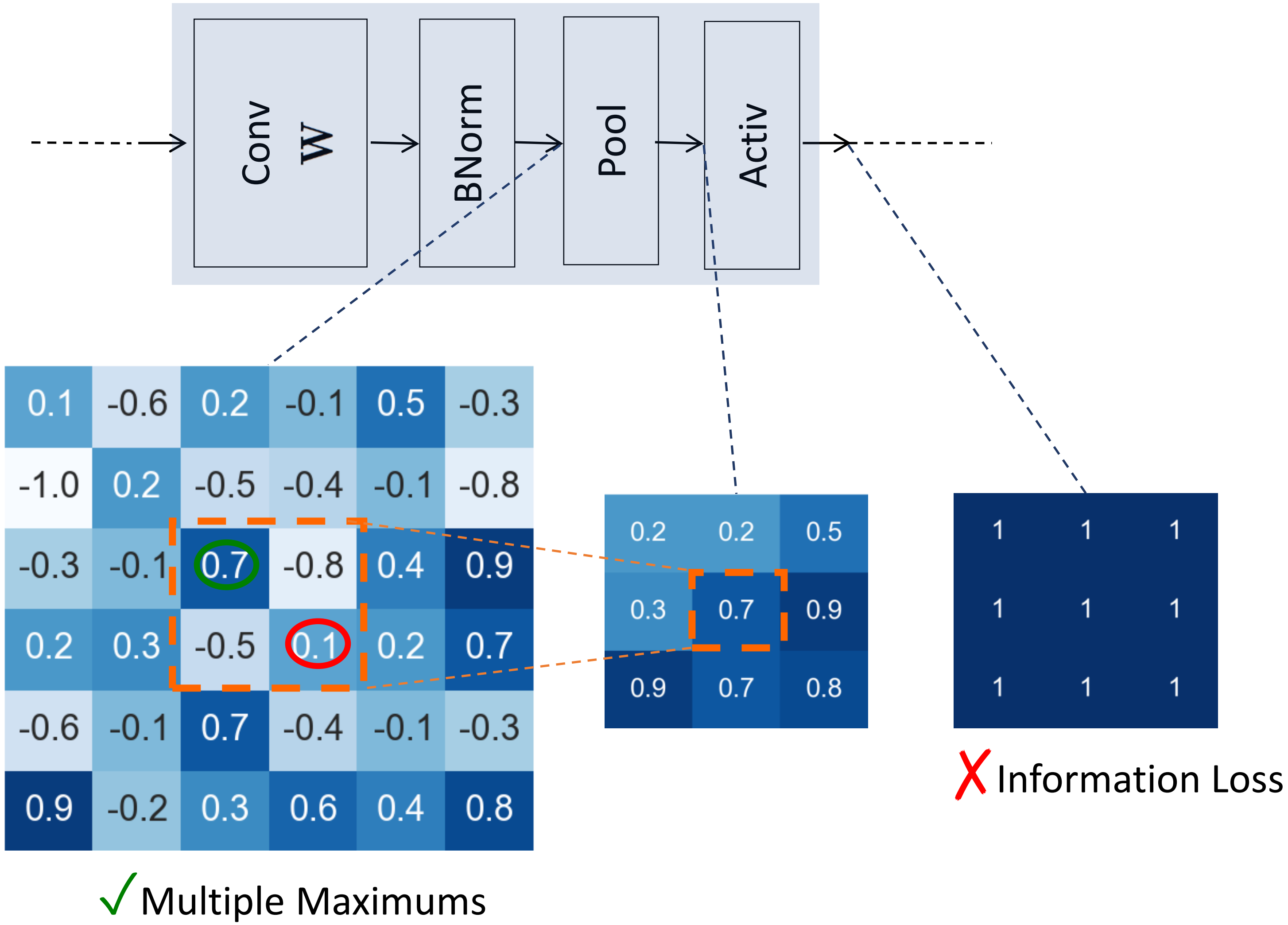
Network Structure in XNOR-Networks



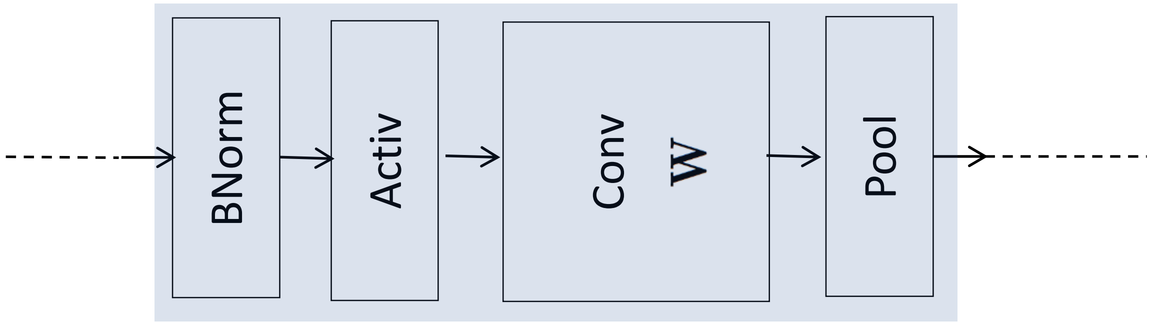
Network Structure in XNOR-Networks



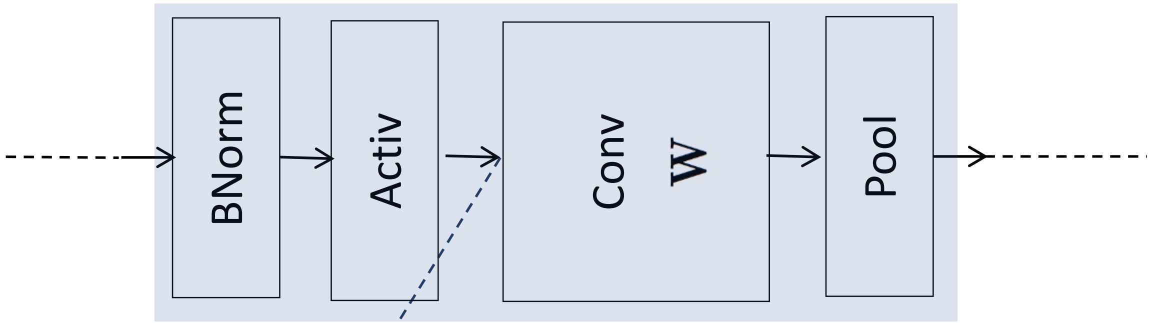
Network Structure in XNOR-Networks



Network Structure in XNOR-Networks

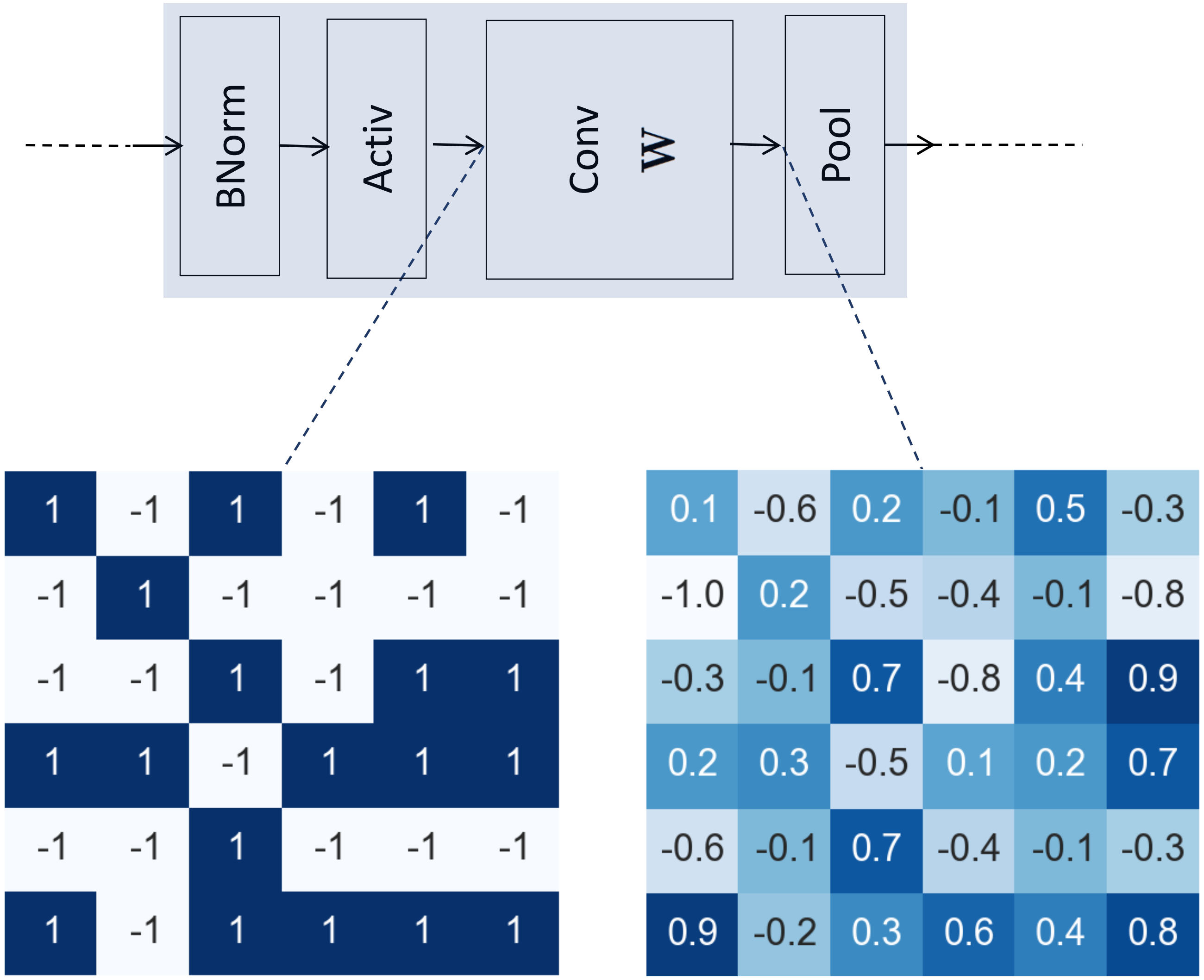


Network Structure in XNOR-Networks

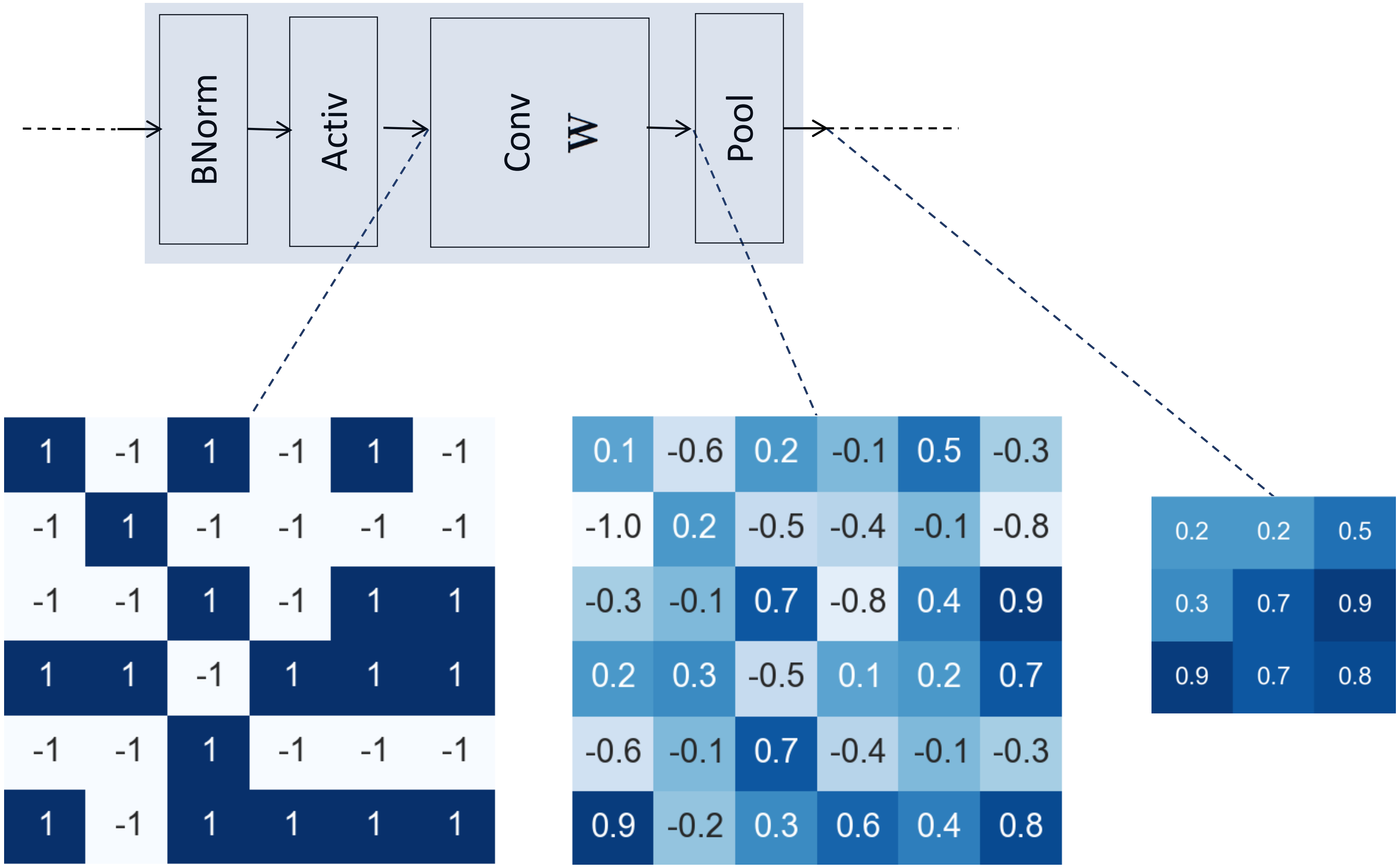


1	-1	1	-1	1	-1
-1	1	-1	-1	-1	-1
-1	-1	1	-1	1	1
1	1	-1	1	1	1
-1	-1	1	-1	-1	-1
1	-1	1	1	1	1

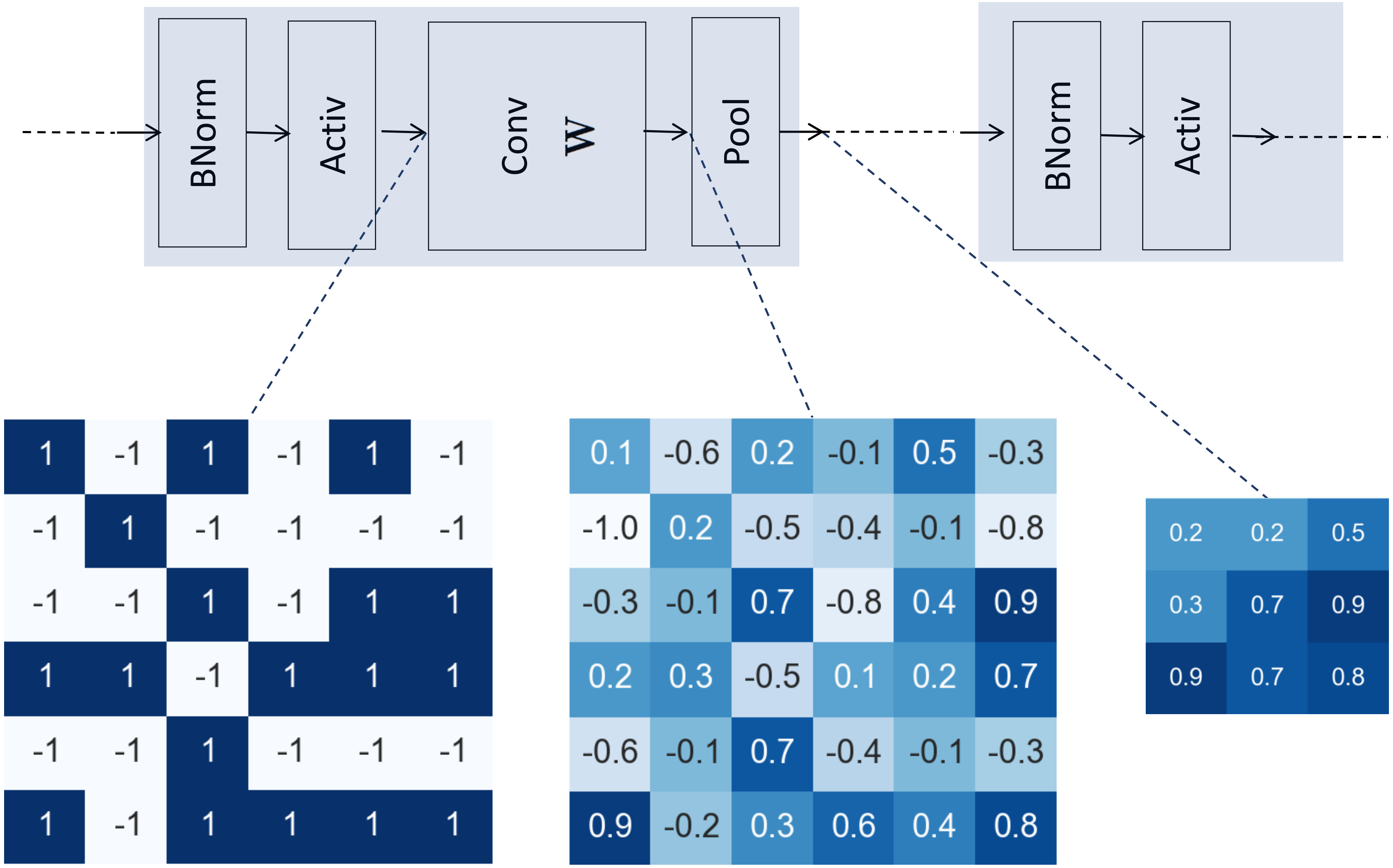
Network Structure in XNOR-Networks



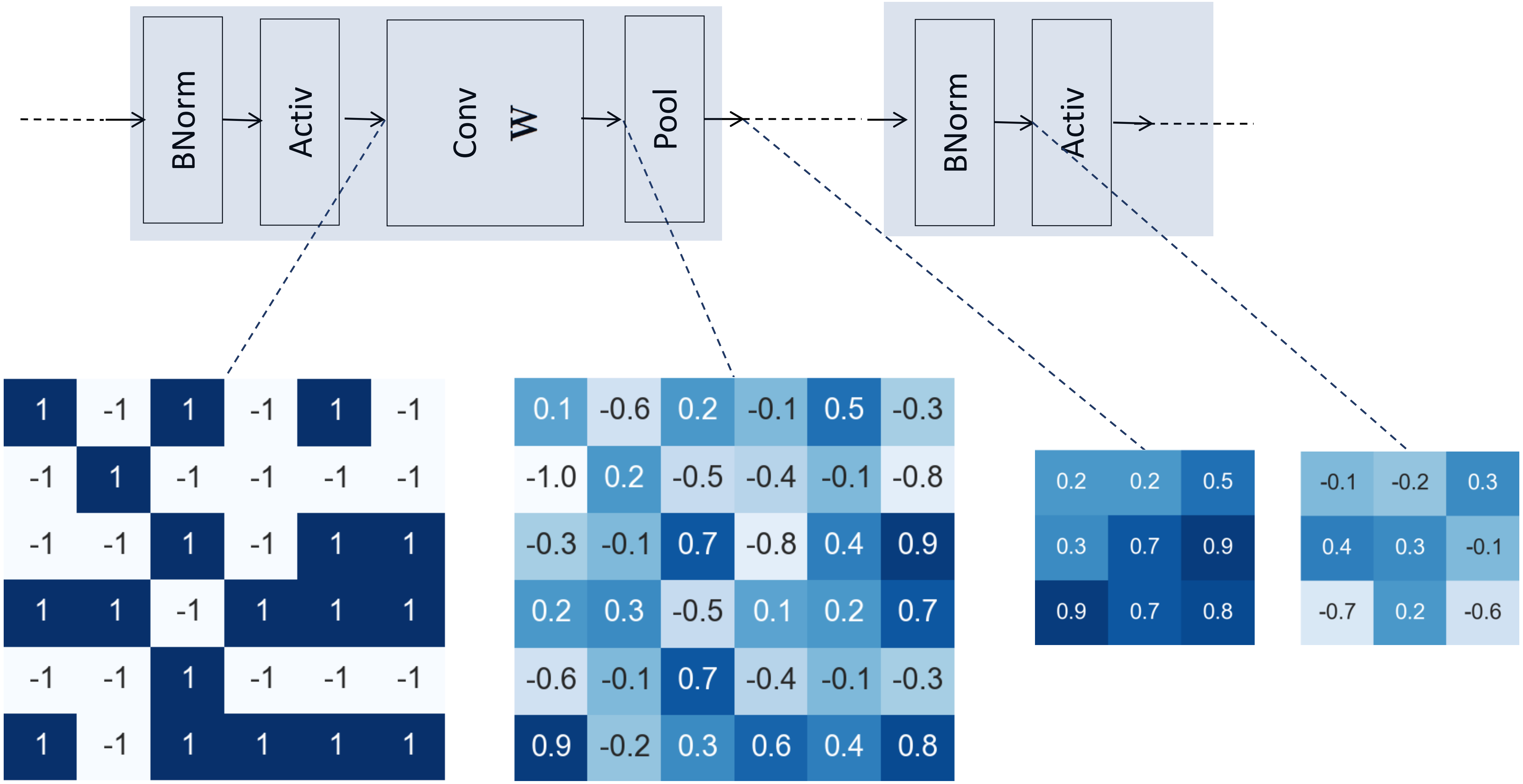
Network Structure in XNOR-Networks



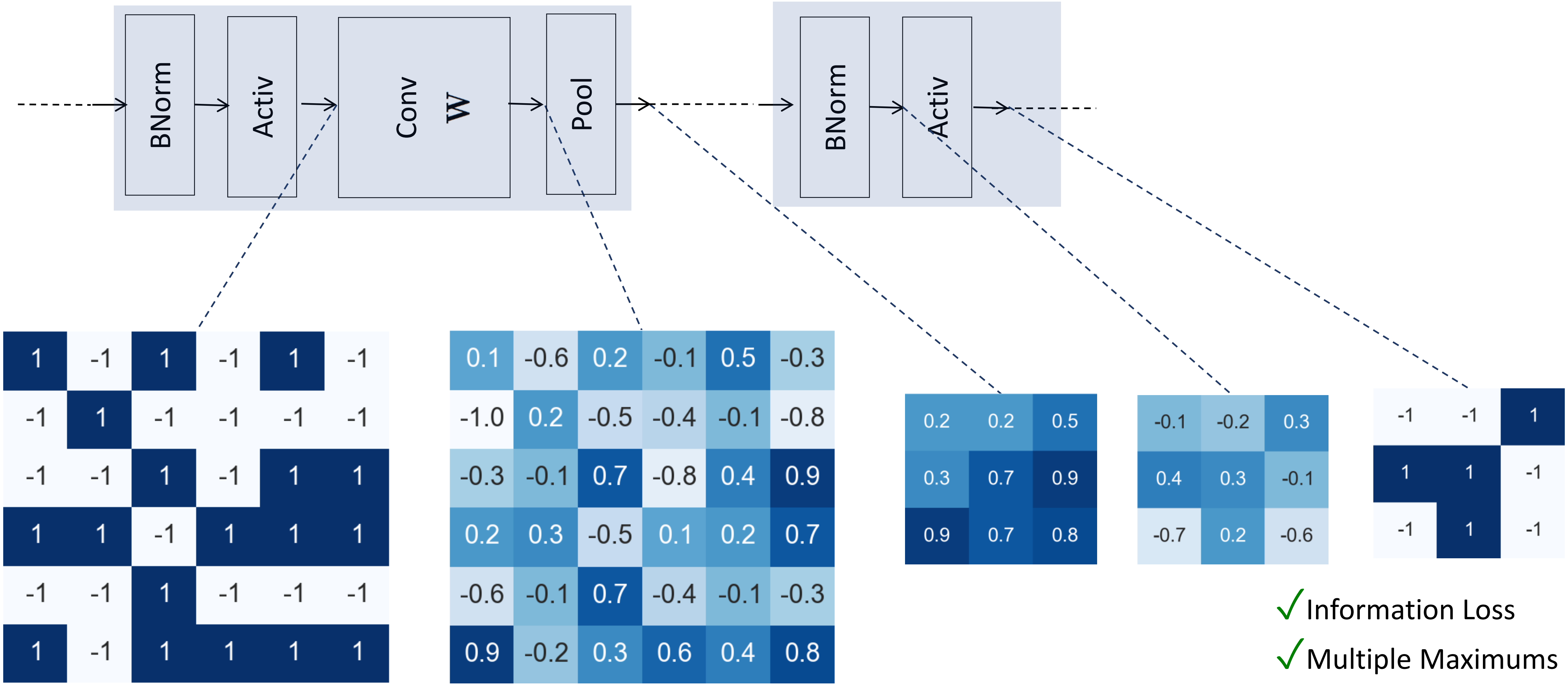
Network Structure in XNOR-Networks



Network Structure in XNOR-Networks

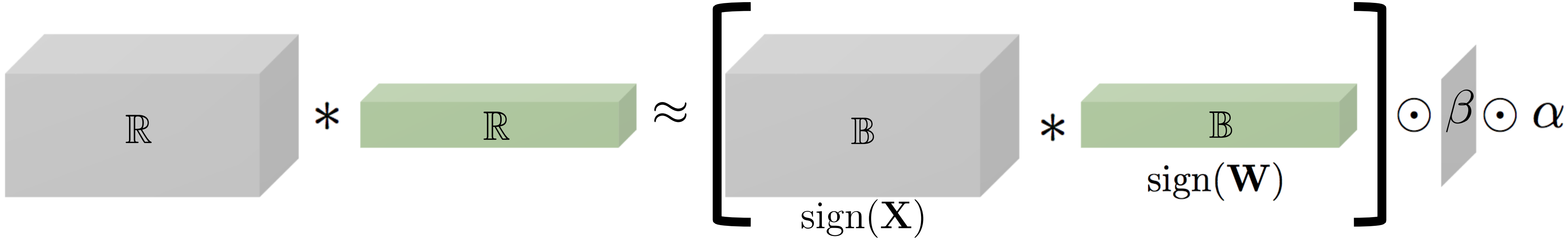


Network Structure in XNOR-Networks

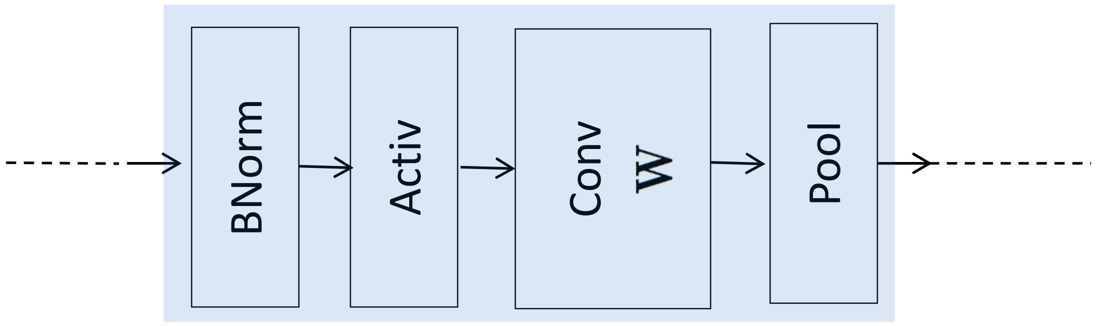


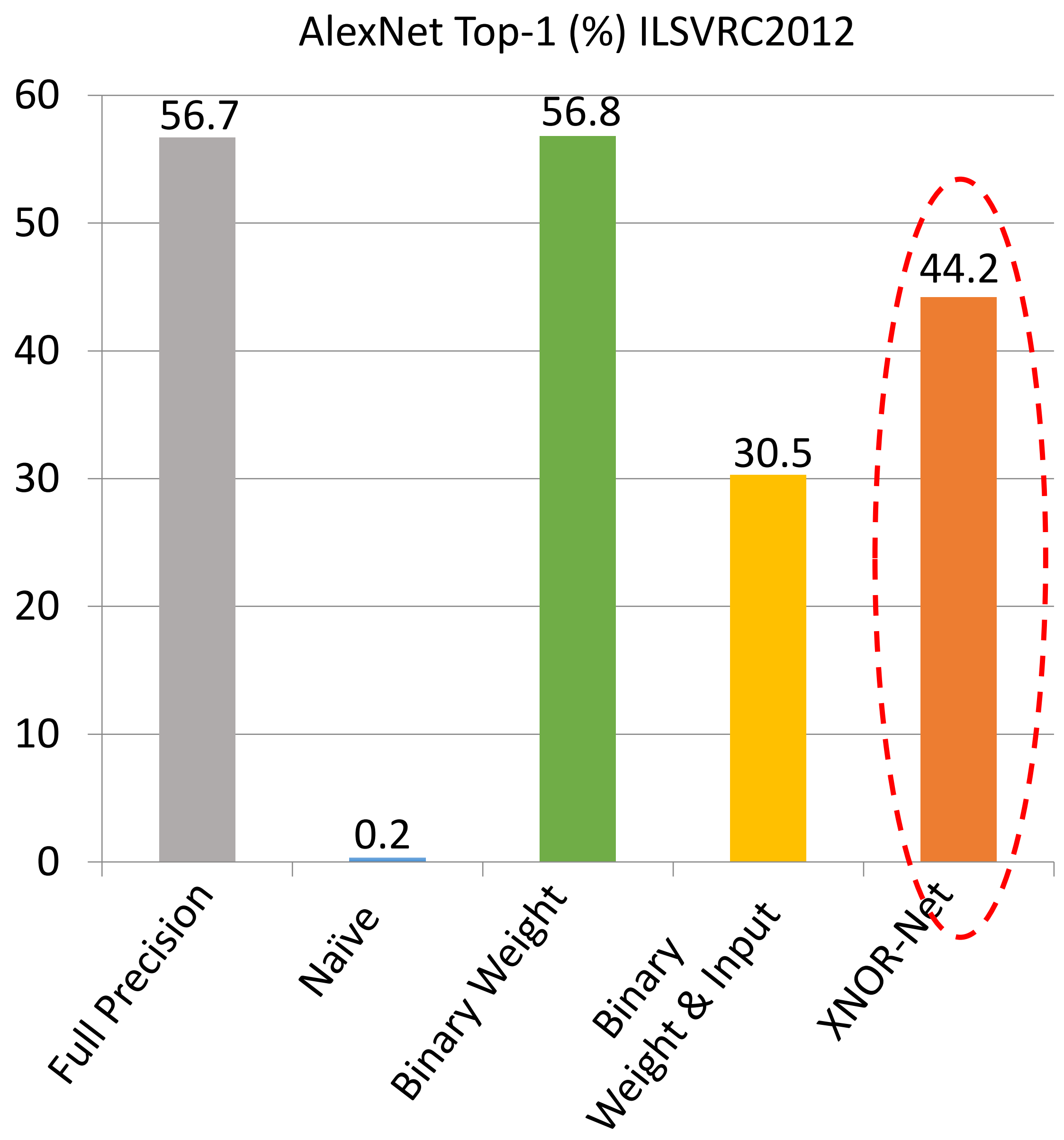
✓Information Loss

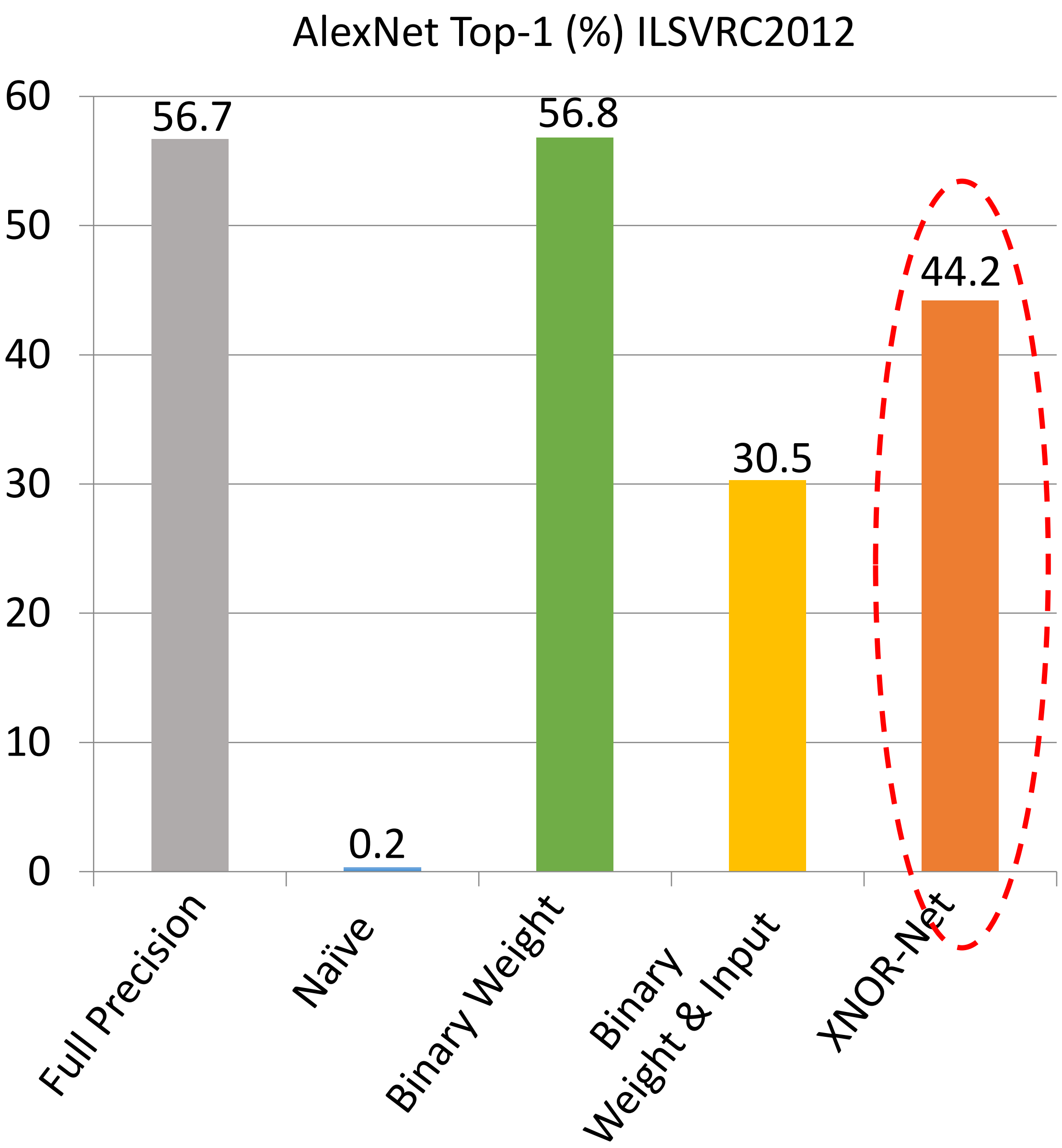
✓Multiple Maximums



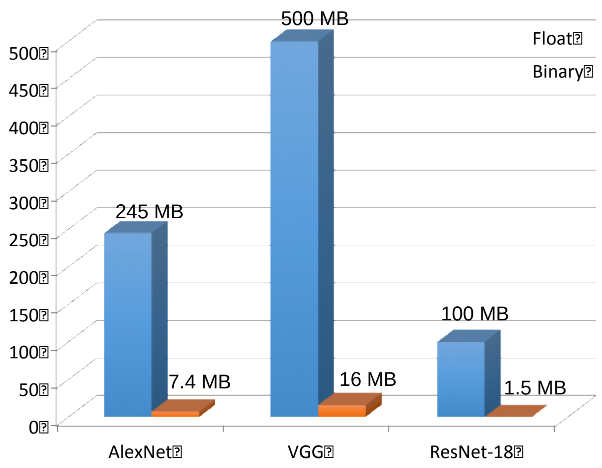
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W^B} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W^B}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial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

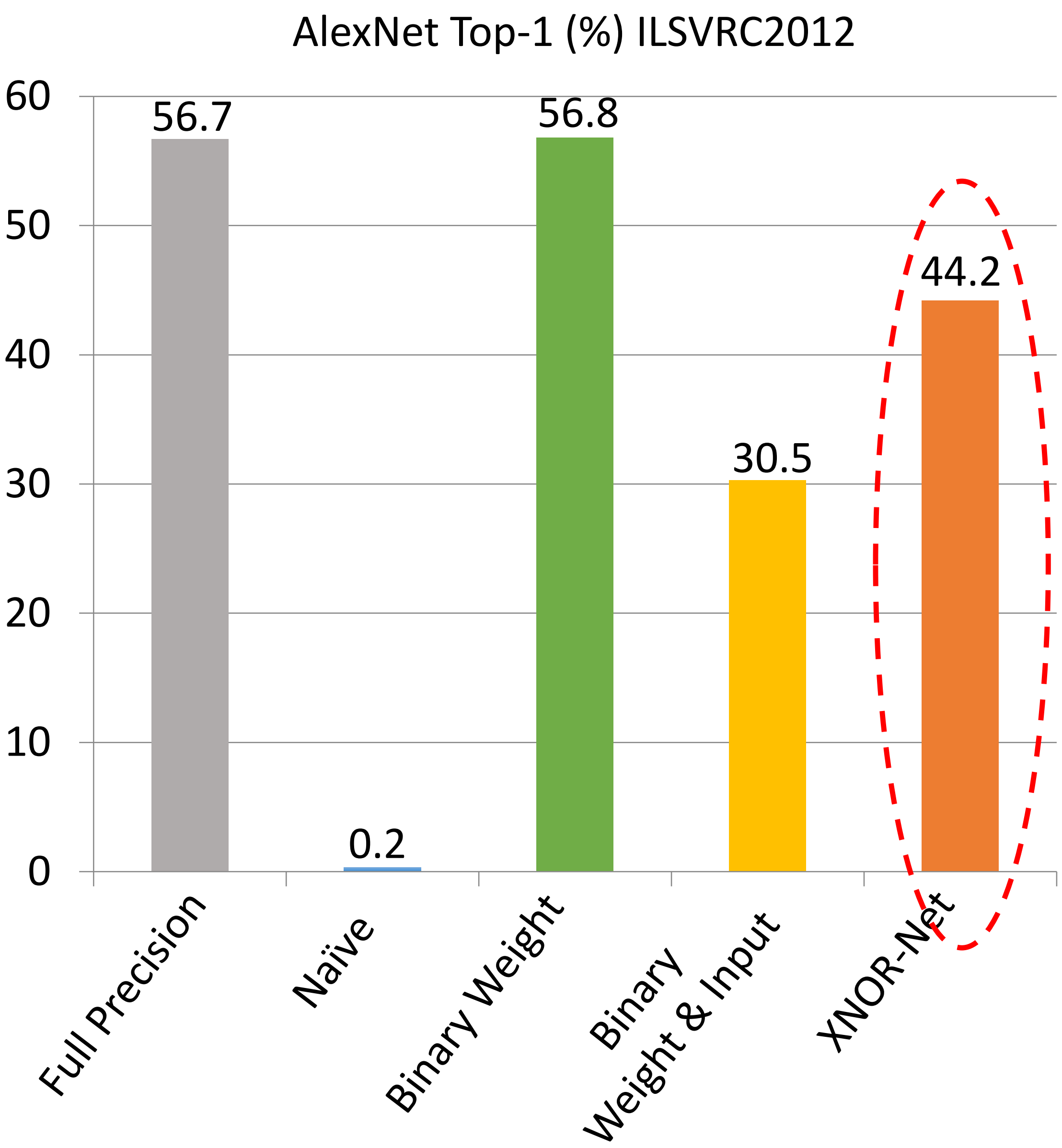



AlexNet Top-1 (%) ILSVRC2012
60
50
40
30
20
10
0
56.7
0.2
56.8
30.5
44.2
Full Precision
Naïve
Binary Weight
Binary Weight & Input
XNOR-Net

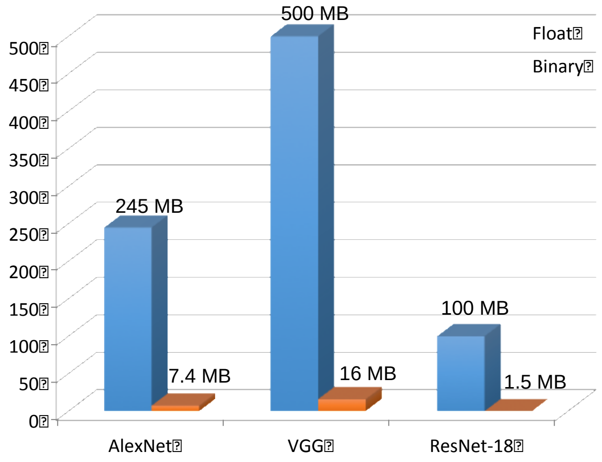


✓ 32x Smaller Model

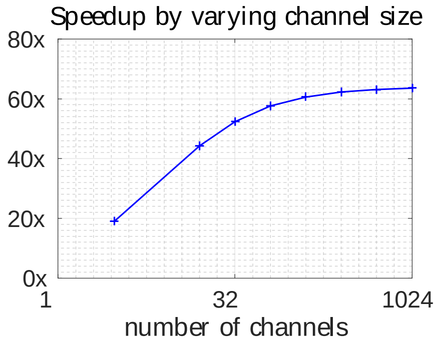


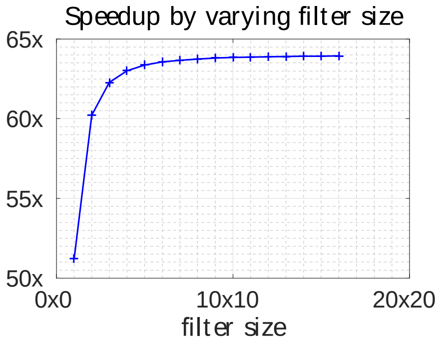


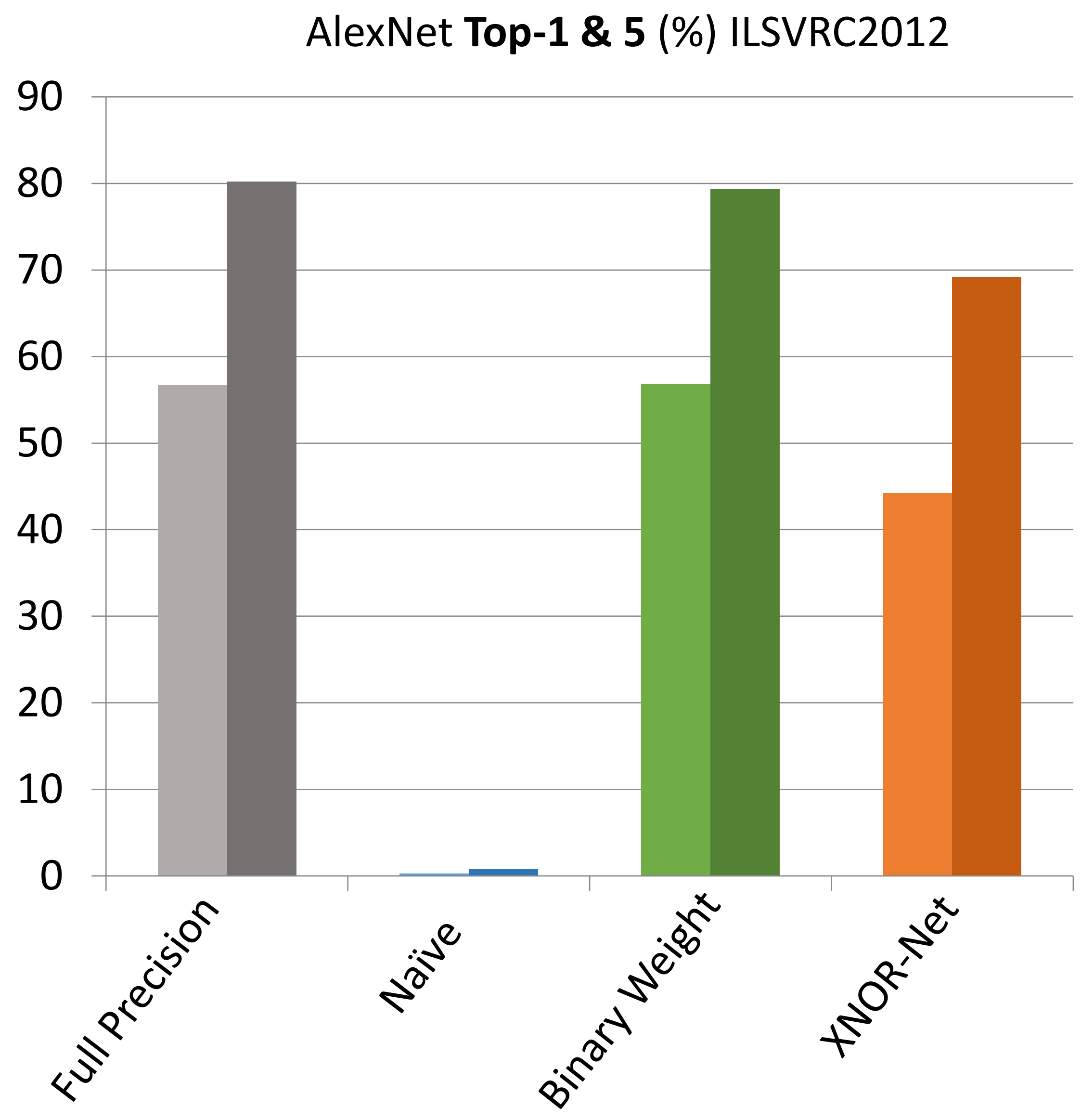
✓ 32x Smaller Model

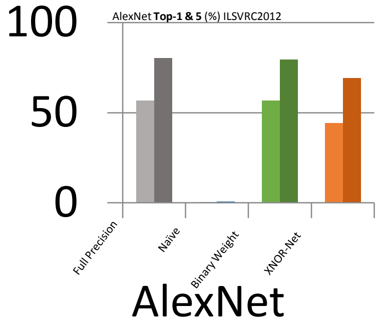


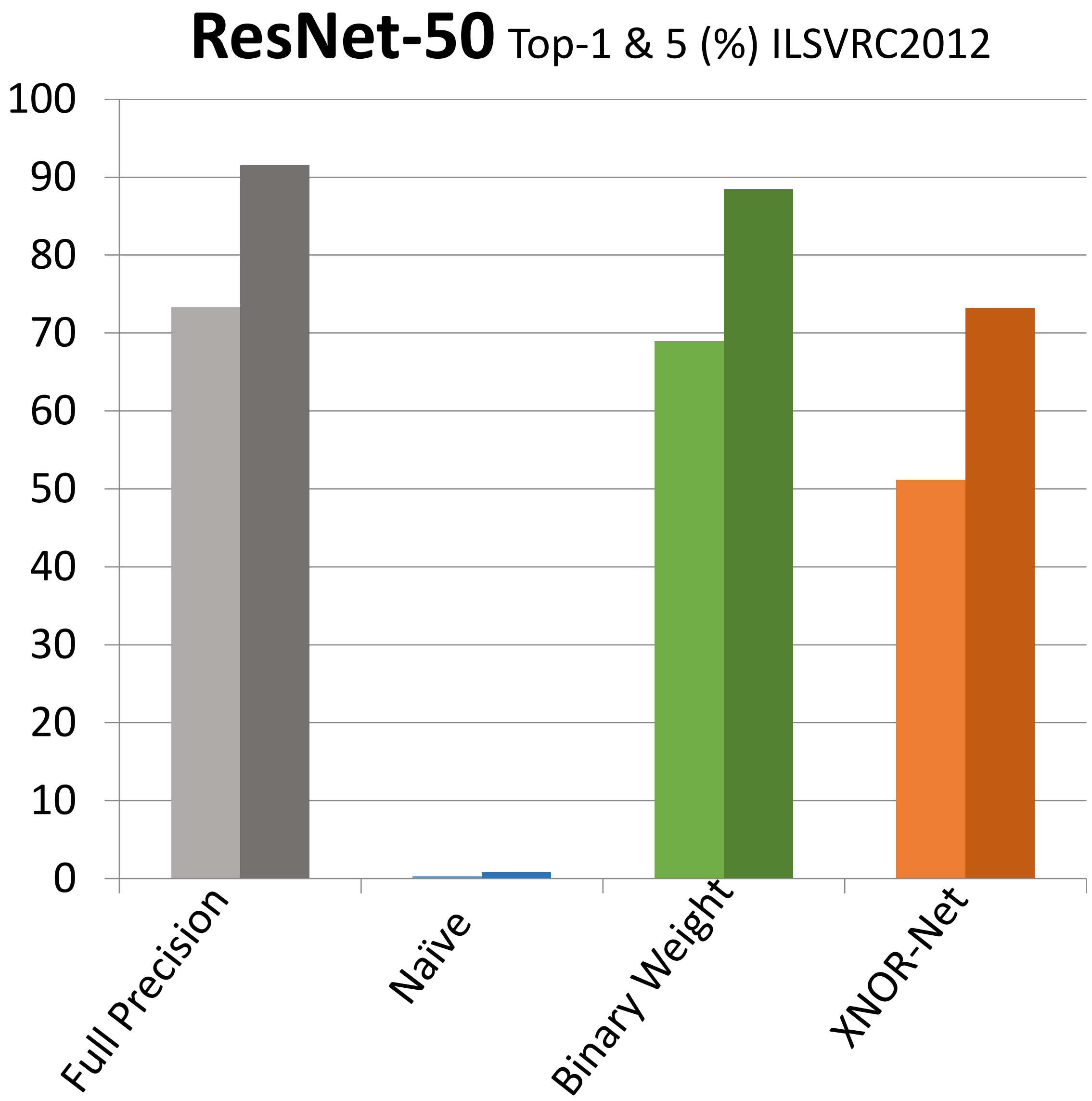
✓ 58x Less Computation

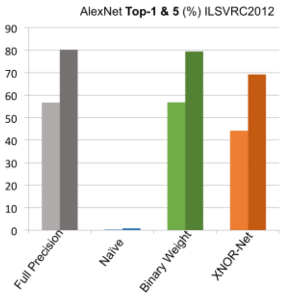




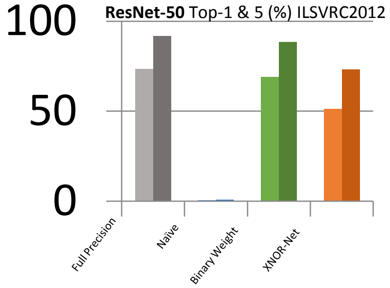

AlexNet Top-1 & 5 (%) ILSVRC2012
90
80
70
60
50
40
30
20
10
0
Full Precision
Naïve
Binary Weight
XNOR-Net

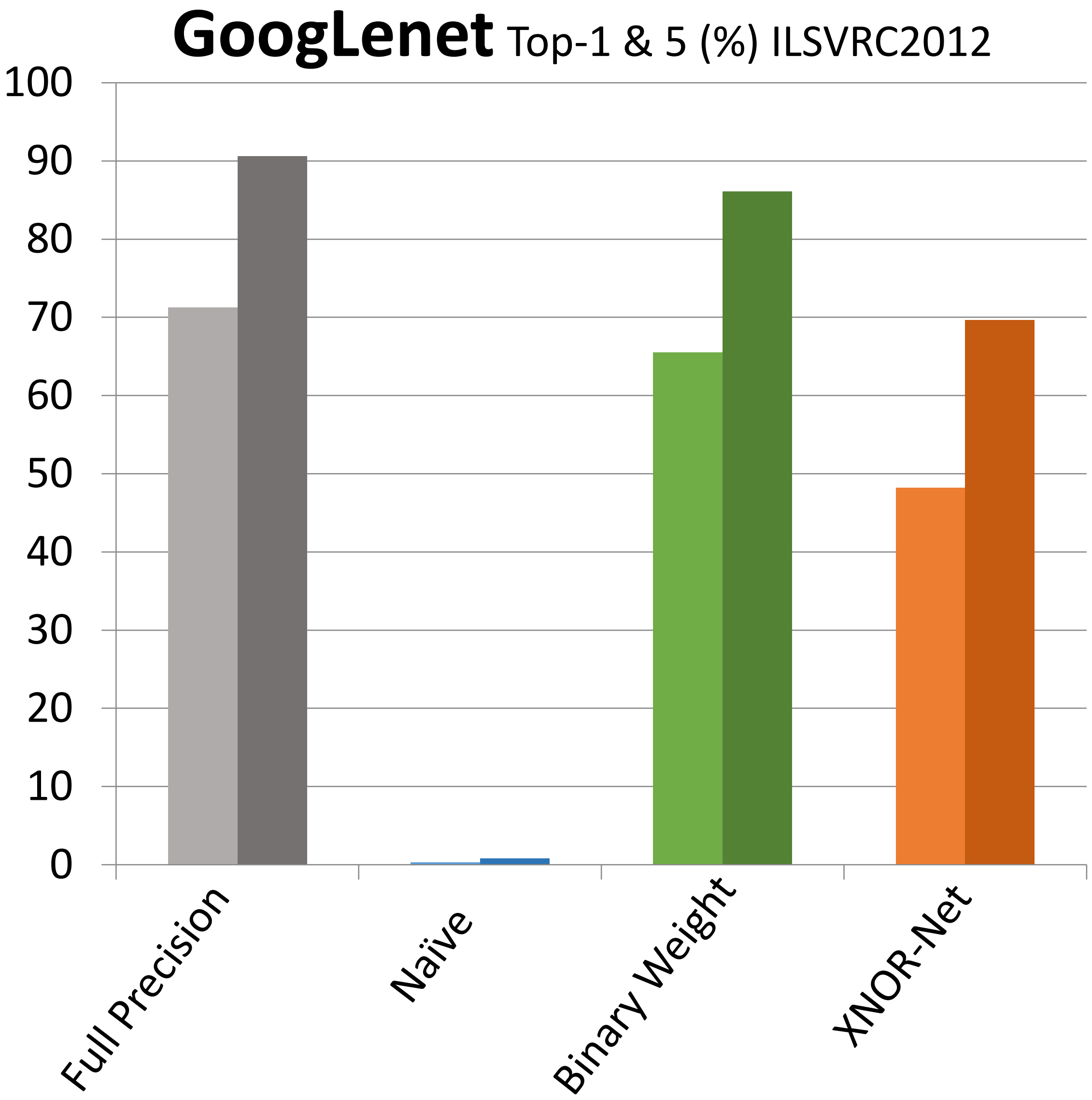

100
AlexNet Top-1 & 5 (%) ILSVRC2012
50
0
Full Precision
Naïve
Binary Weight
XNOR-Net
AlexNet


ResNet-50 Top-1 & 5 (%) ILSVRC2012
100
90
80
70
60
50
40
30
20
10
0
Full Precision
Naïve
Binary Weight
XNOR-Net

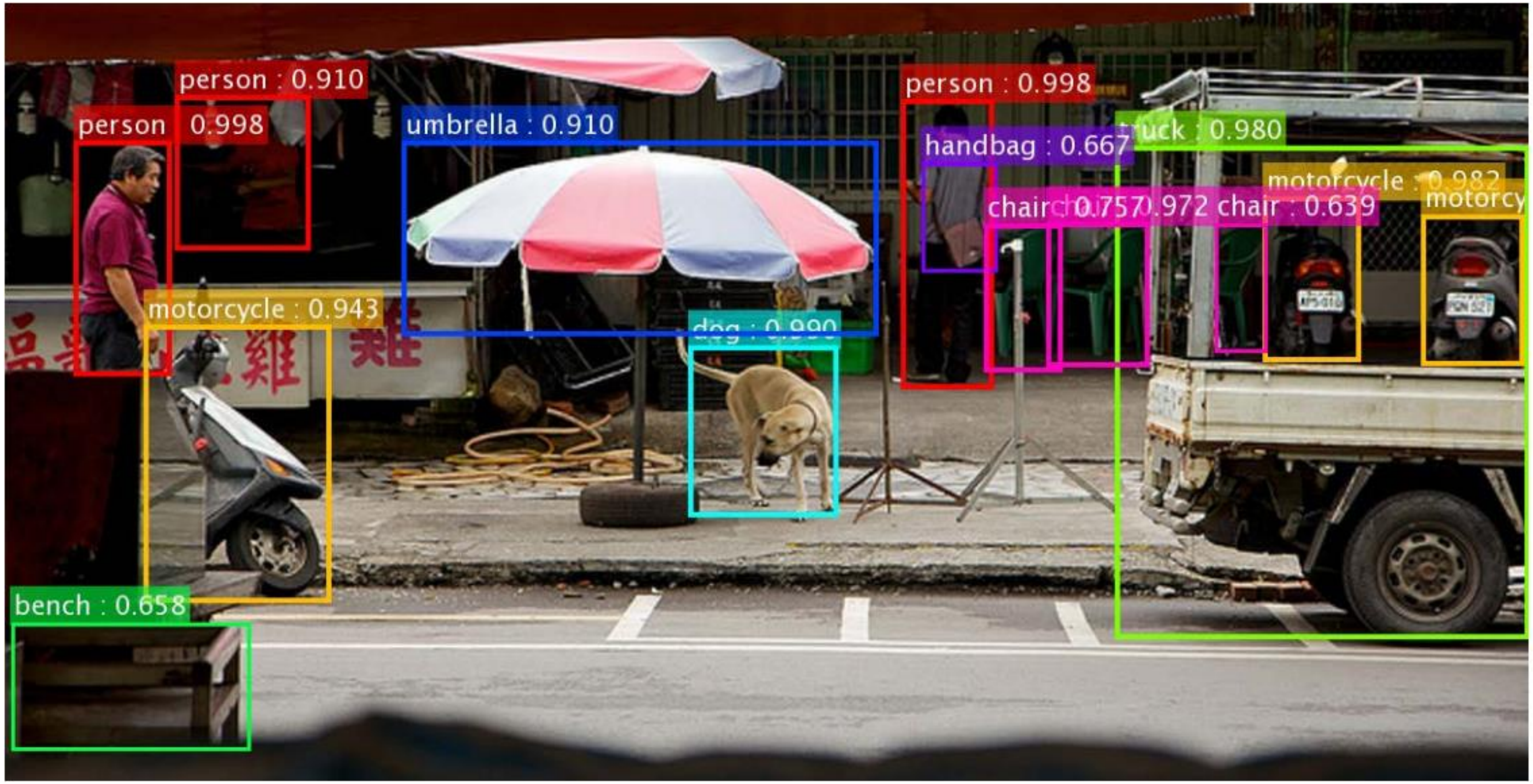

AlexNet Top-1 & 5 (%) ILSVRC2012
90
80
70
60
50
40
30
20
10
0
Full Precision
Naïve
Binary Weight
XNOR-Net

AlexNet


ResNet-50 Top-1 & 5 (%) ILSVRC2012
100
50
0
Full Precision
Naïve
Binary Weight
XNOR-Net


GoogLenet Top-1 & 5 (%) ILSVRC2012
100
90
80
70
60
50
40
30
20
10
0
Full Precision
Naïve
Binary Weight
XNOR-Net

Object Detection



[He et al, 2015]

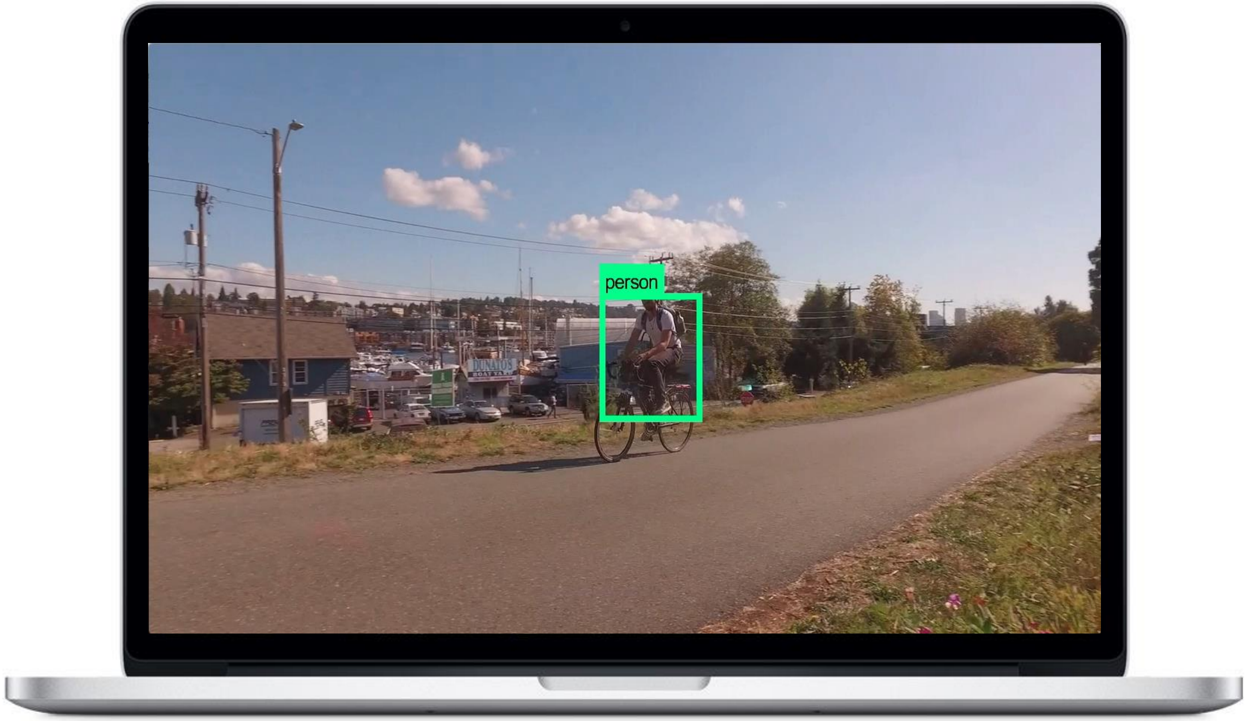
YOLO: Fastest Object Detector

[Redmon et al. CVPR 2016]

YOLO on CPU (NOT GPU)

Fastest Object Detector

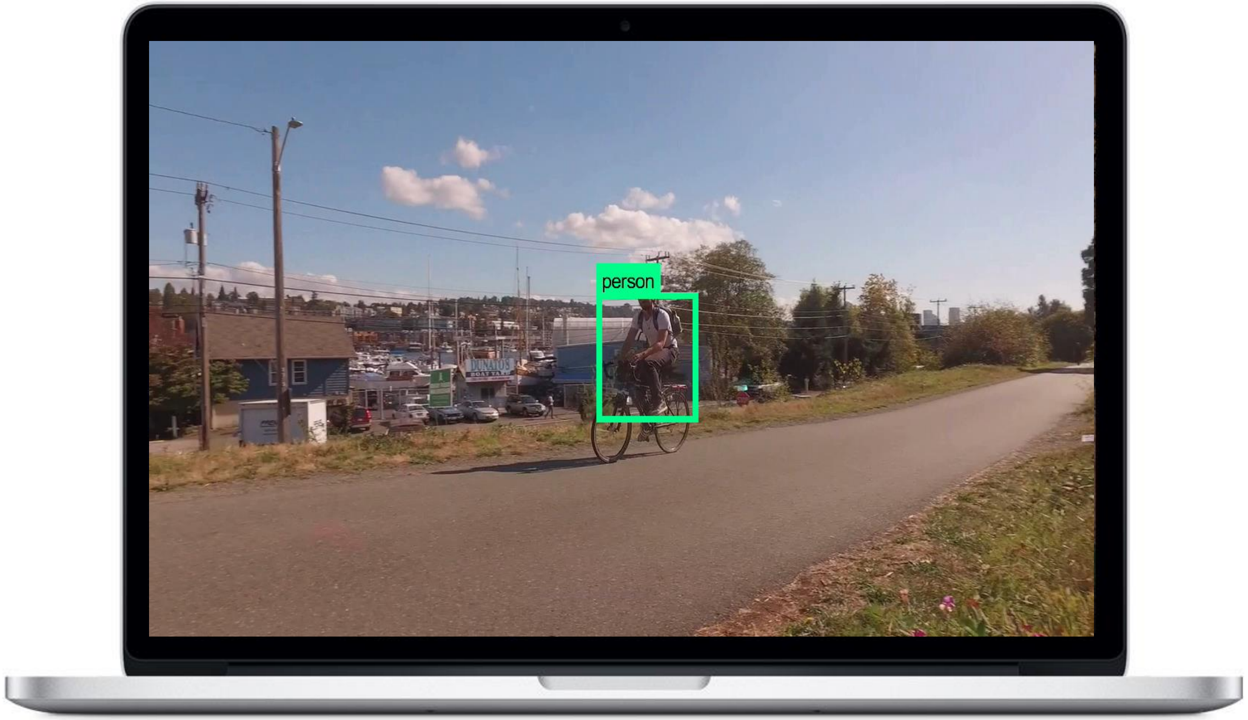
[Redmon et al. CVPR 2016]



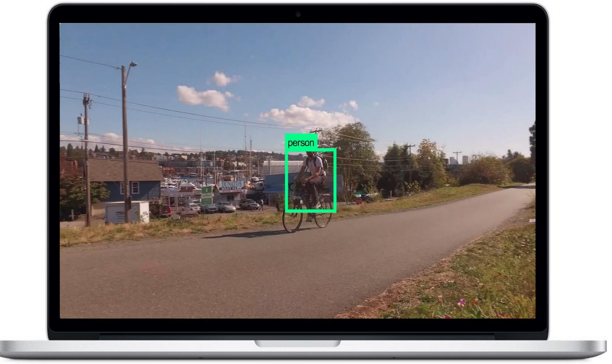
YOLO on CPU (NOT GPU)

Fastest Object Detector

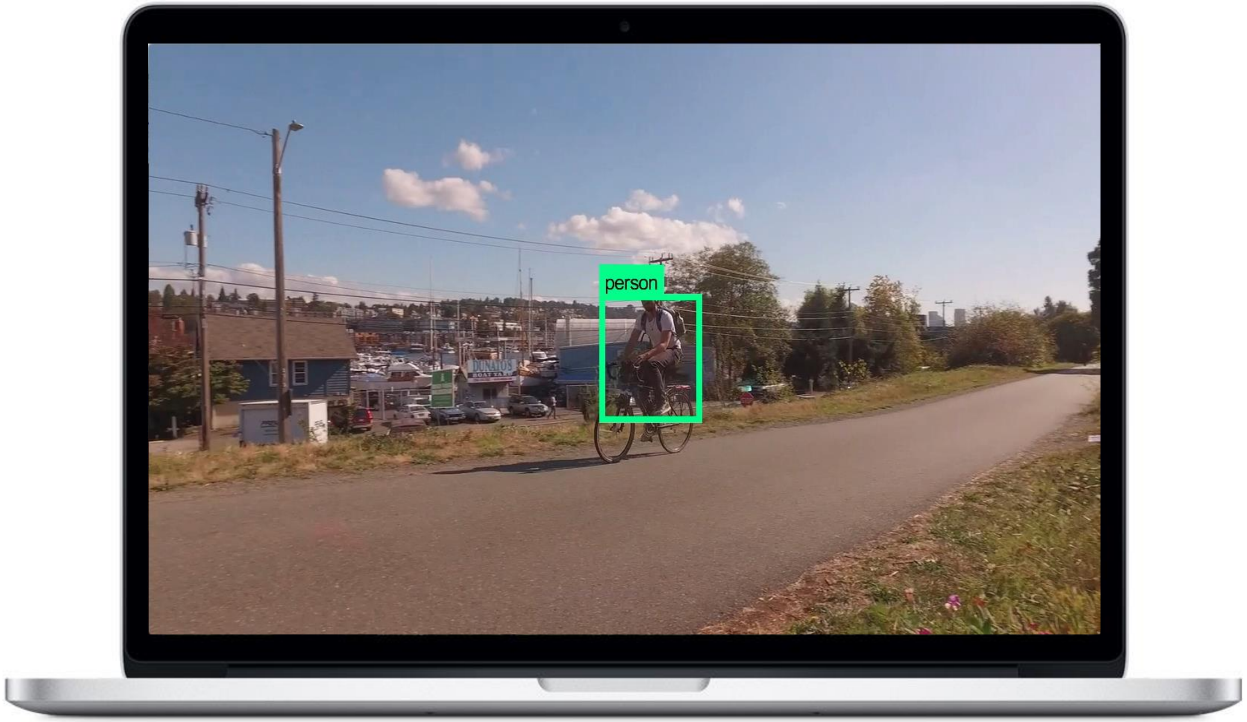
[Redmon et al. CVPR 2016]



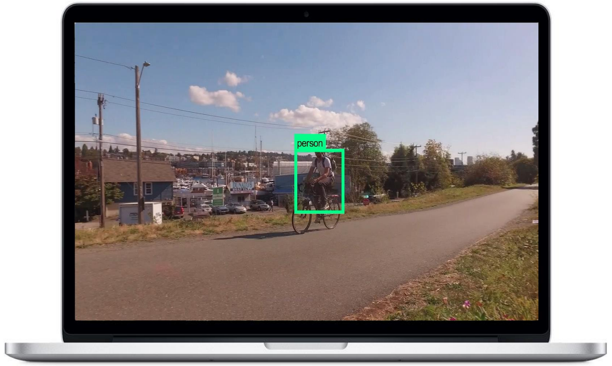
YOLO on CPU(NOT GPU)
Fastest Object Detector
[Redmon et al. CVPR 2016]



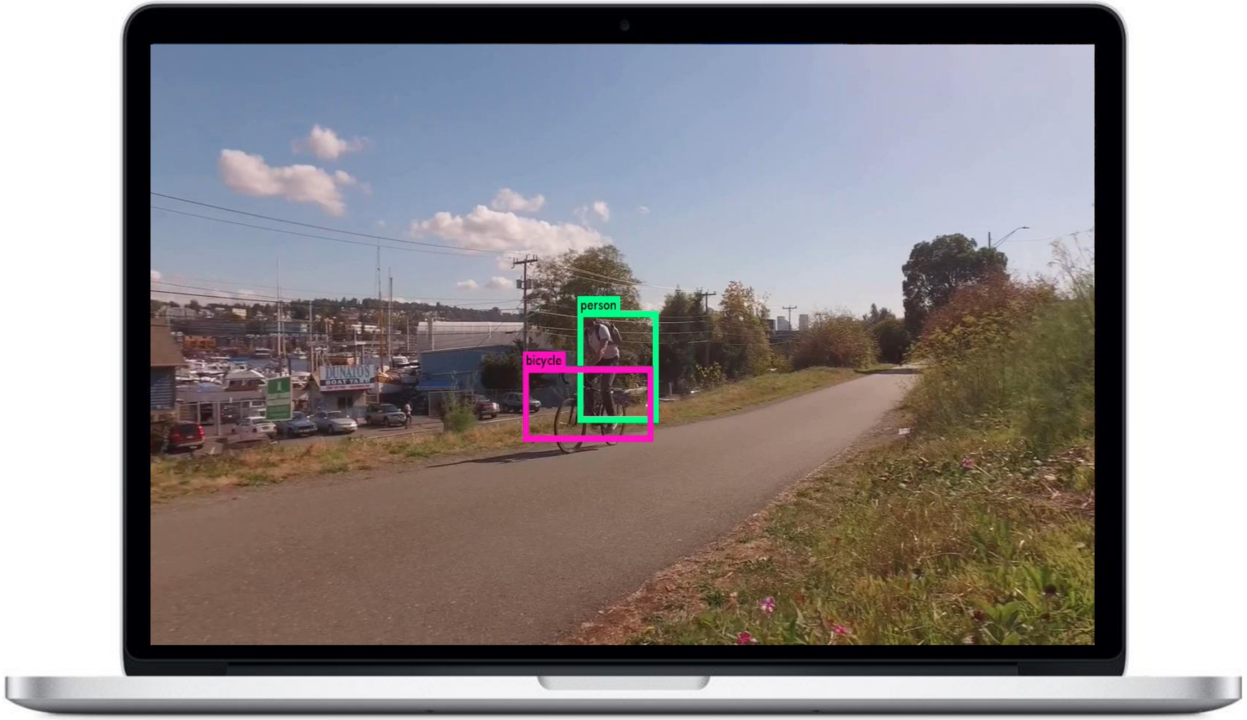
Our Method



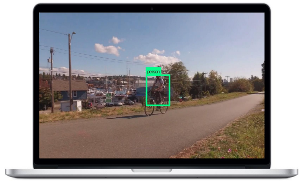
YOLO
Fastest Object Detector
[Redmon et al. CVPR 2016]



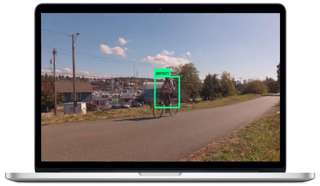
Our Method
On CPU (NOT GPU)



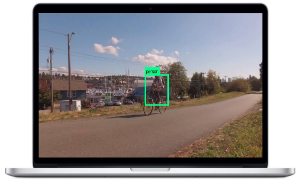
YOLO
Fastest Object Detector
[Redmon et al. CVPR 2016]



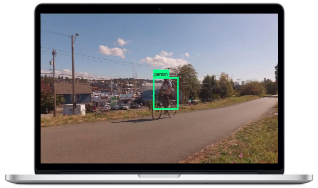
Our Method
On CPU (NOT GPU)



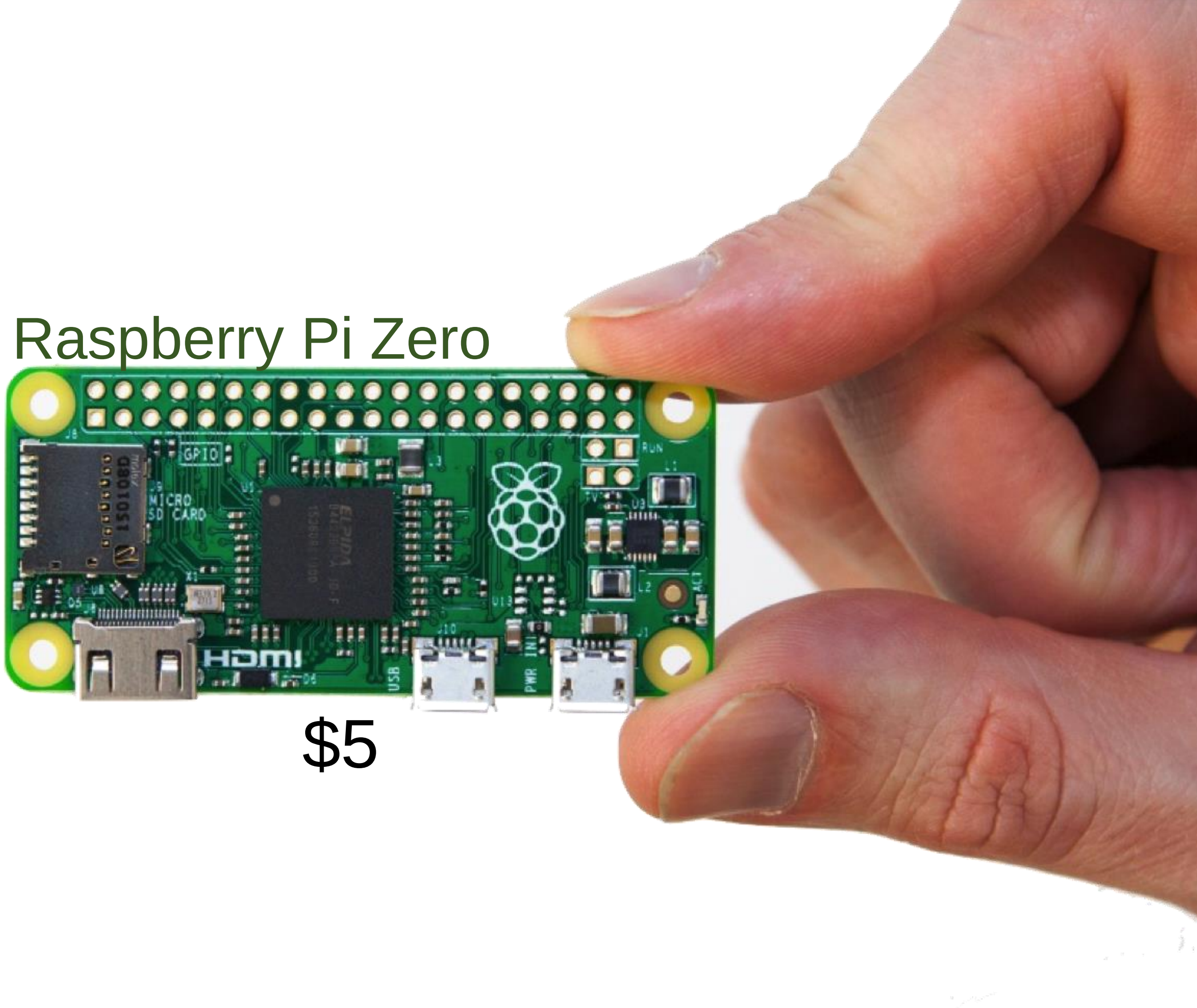
YOLO
Fastest Object Detector
[Redmon et al. CVPR 2016]



Our Method

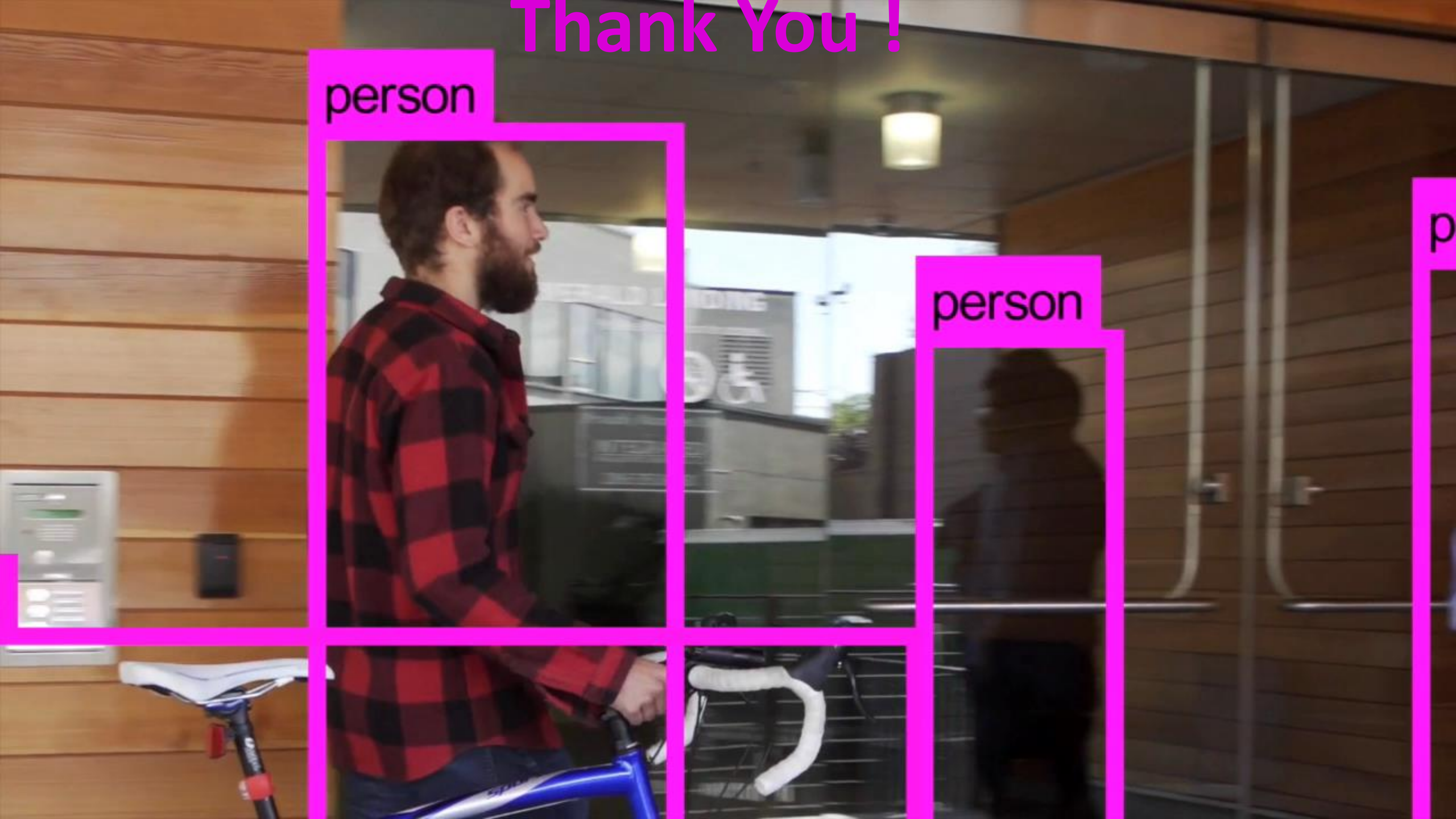


Raspberry Pi Zero



$5

Thank You !



Thank You !

Carlo C. del Mundo

Dmitry Belenko

GitHub
https://github.com/allenai/XNOR-Net