CMSC 5743

Efficient Computing of Deep Neural Networks

Mo04: Binary/Ternary Network

Bei Yu
CSE Department, CUHK
byu@cse.cuhk.edu.hk

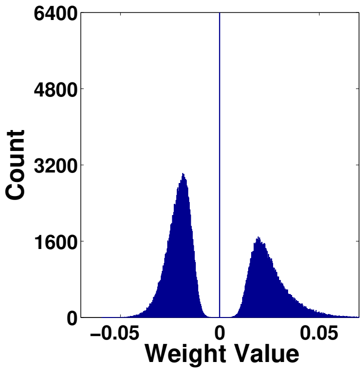
(Latest update: March 23, 2023)

Spring 2023

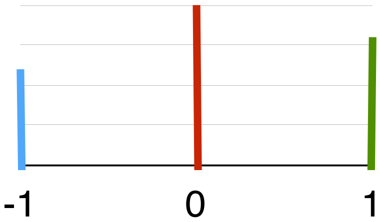
These slides contain/adapt materials developed by

- 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

Binary / Ternary Net: Motivation

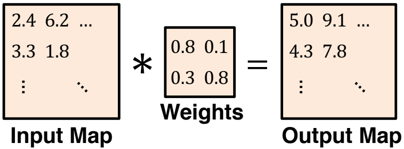


=>



Binarized Neural Networks (BNN)

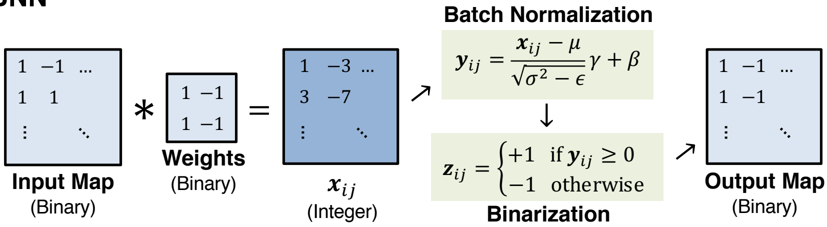
CNN



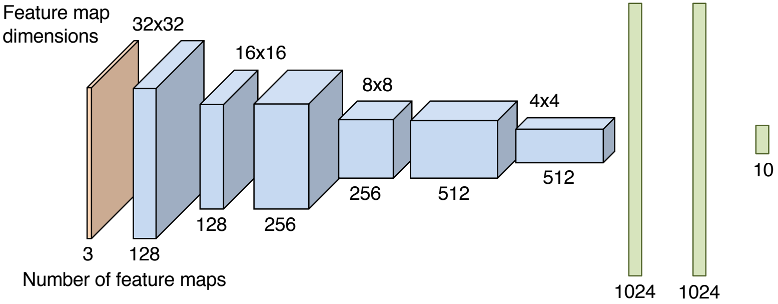
Key Differences

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

BNN



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.*

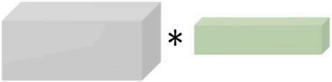
Overview

1. Minimize the Quantization Error
2. Reduce the Gradient Error

Overview

1. Minimize the Quantization Error

2. Reduce the Gradient Error

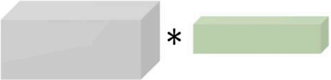
	Operations	Memory	Computation
$\mathbb{R} * \mathbb{R}$	$+ \ - \ \times$	1x	1x

Binary Weight Networks

XNOR-Networks

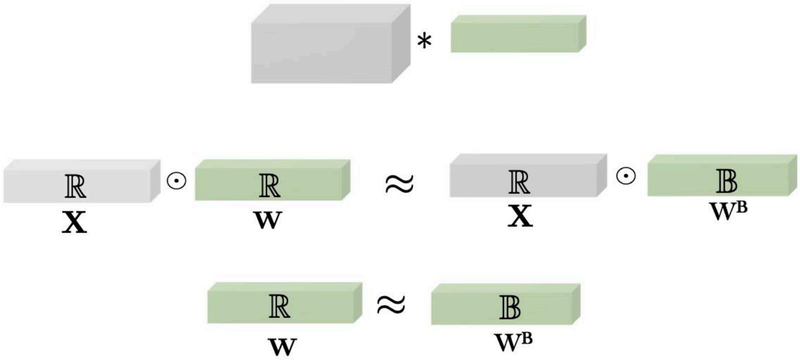
[1]

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

	Operations	Memory	Computation
ℝ * ℝ	+ − ×	1x	1x
ℝ * 𝔹	+ −	~32x	~2x
𝔹 * 𝔹	XNOR Bit-count	~32x	~58x

[1]

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



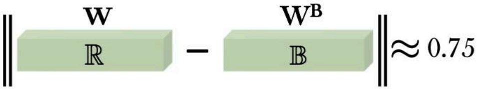
$$\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$$

[1]

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

Quantization Error

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



[1]

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

Optimal Scaling Factor

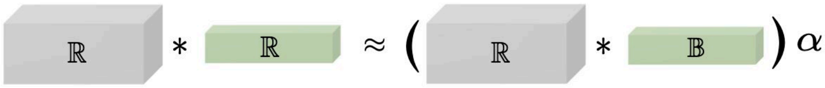
$$\alpha^*, \mathbf{W}^{\mathbf{B}*} = \arg\min_{\mathbf{W}^{\mathbf{B}},\alpha} \{||\mathbf{W} - \alpha\mathbf{W}^{\mathbf{B}}||^2\}$$

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

[1]

[1]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?



[1]

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

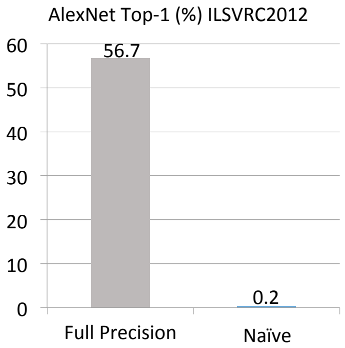
Training Binary Weight Networks

Naive Solution:

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

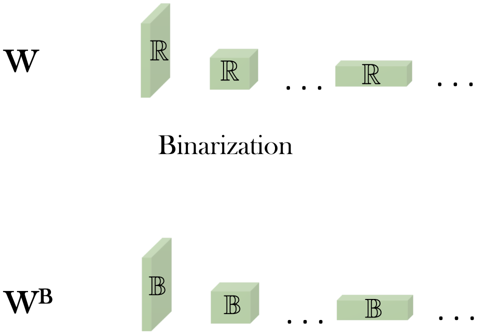
1

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



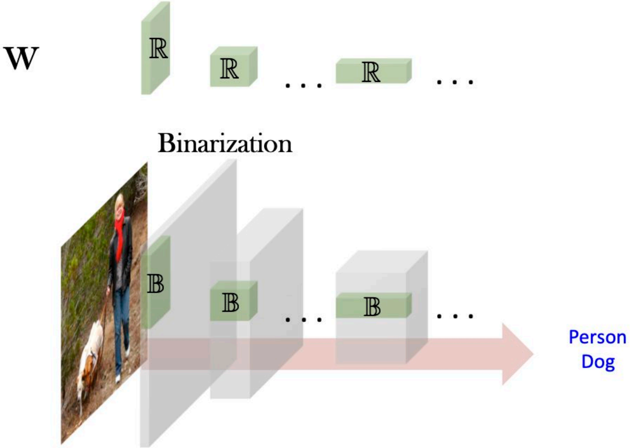
[1]

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



1

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



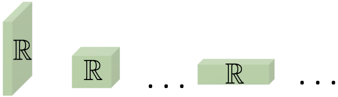
[1]

[1] 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 $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$



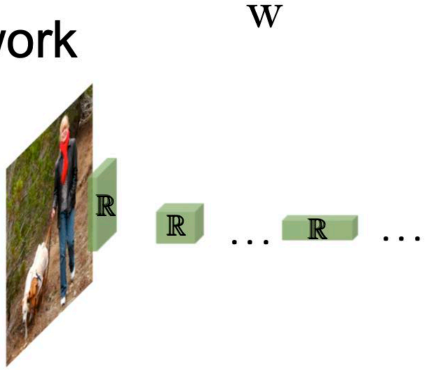
[1]

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

Binary Weight Network[1]

Train for binary weights:

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

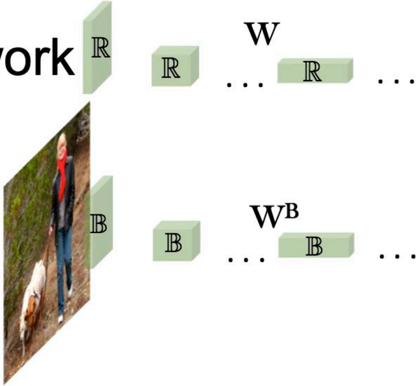


[1]Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Weight Network [1]

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 $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$

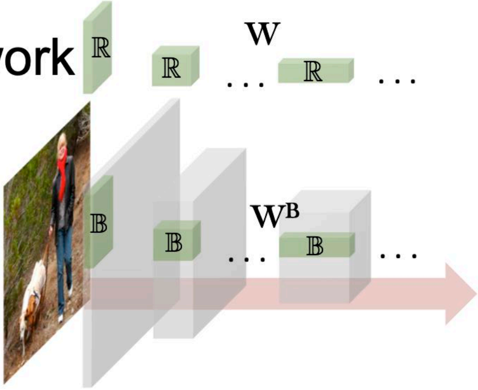


[1] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Weight Network

Train for binary weights:

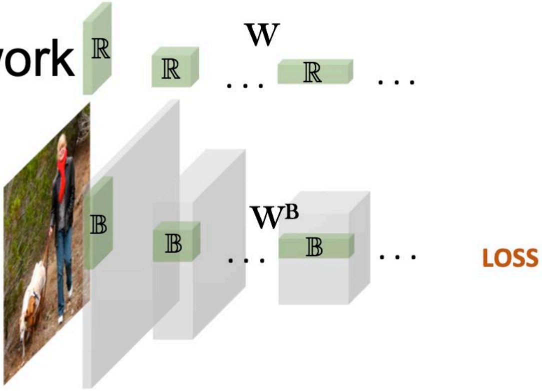
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W^B} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W^B}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial \mathbf{C}}{\partial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



[1]

[1]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 $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$

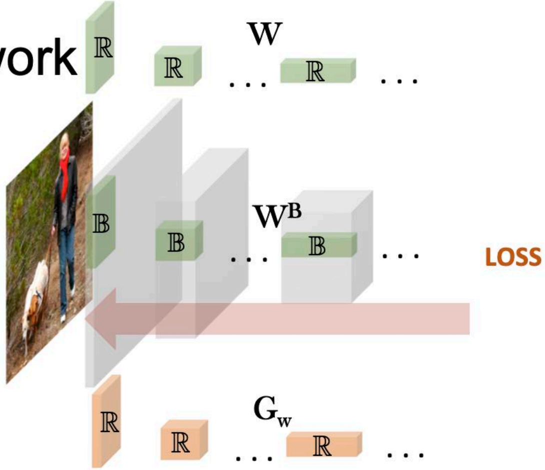
[1]

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

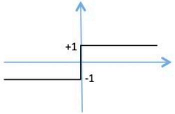
Binary Weight Network

Train for binary weights:

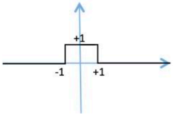
1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathbf{B}} = \text{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathbf{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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



sign(x) →



G_x →



[Hinton et al. 2012]

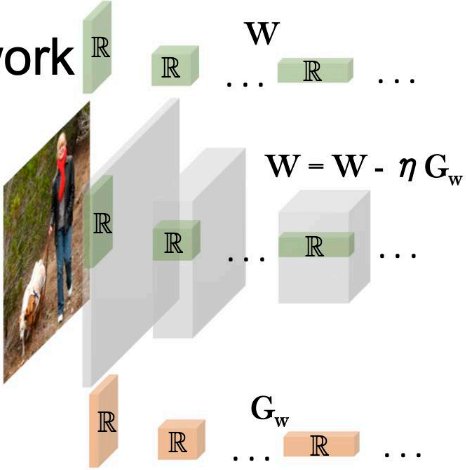
[1]

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

Binary Weight Network

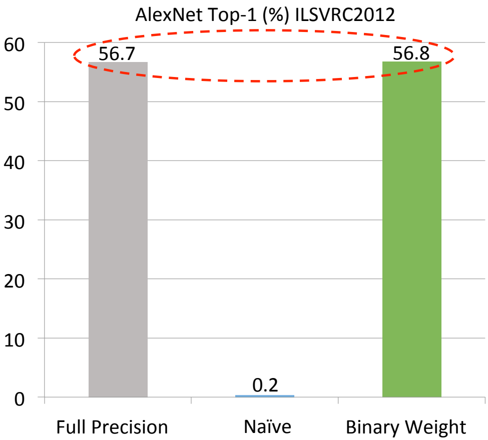
Train for binary weights:

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



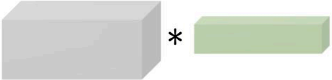
[1]

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



[1]

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



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

[1]

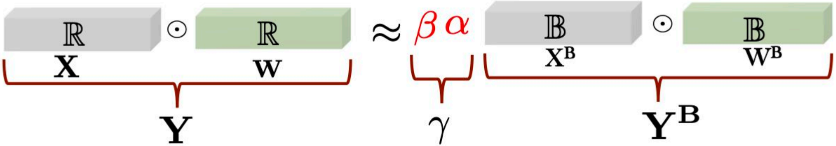
[1]Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Binary Input and Binary Weight (XNOR-Net)

[1]

[1]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)



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

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

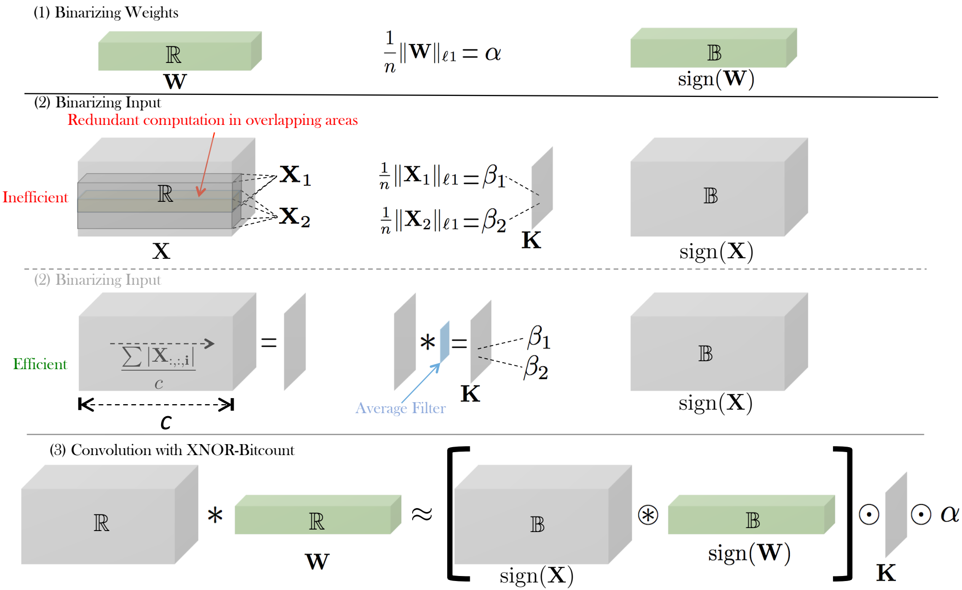
$$\mathbf{Y}^{\mathbf{B}*} = \text{sign}(\mathbf{Y}) \quad \gamma^* = \frac{1}{n}\|\mathbf{Y}\|_{\ell 1}$$

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

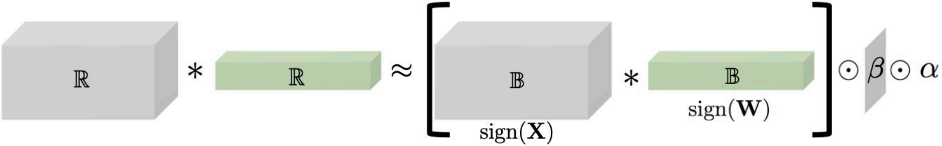
$$\alpha^* = \frac{1}{n}\|\mathbf{W}\|_{\ell 1} \qquad \beta^* = \frac{1}{n}\|\mathbf{X}\|_{\ell 1}$$

[1]

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



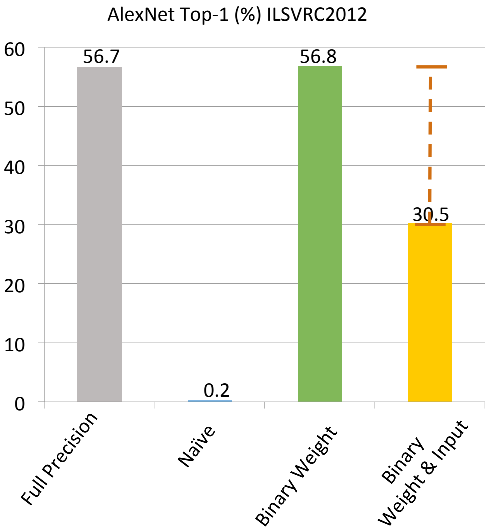
[1]Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



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 $\mathbf{W}$ $(\mathbf{W} = \mathbf{W} - \frac{\partial \mathbf{C}}{\partial \mathbf{W}})$

[1]

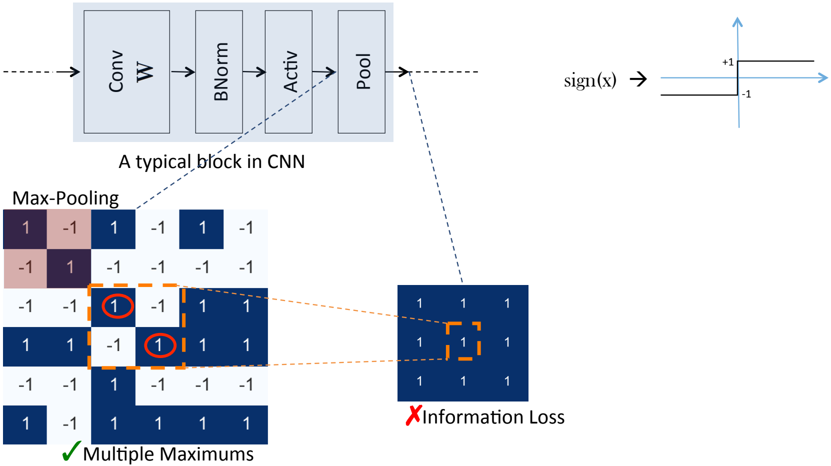
[1] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



1

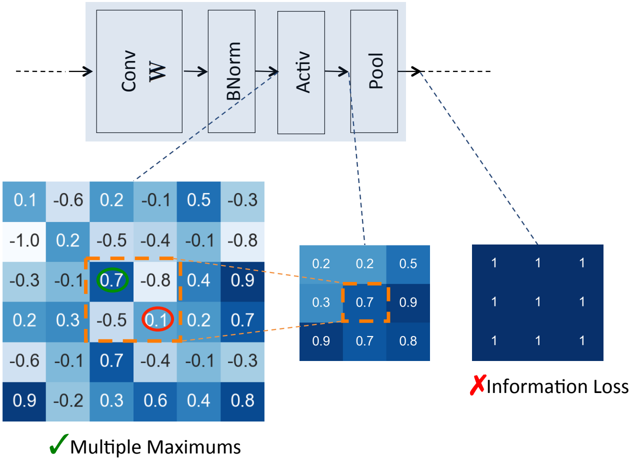
[1]Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Network Structure in XNOR-Networks[1]



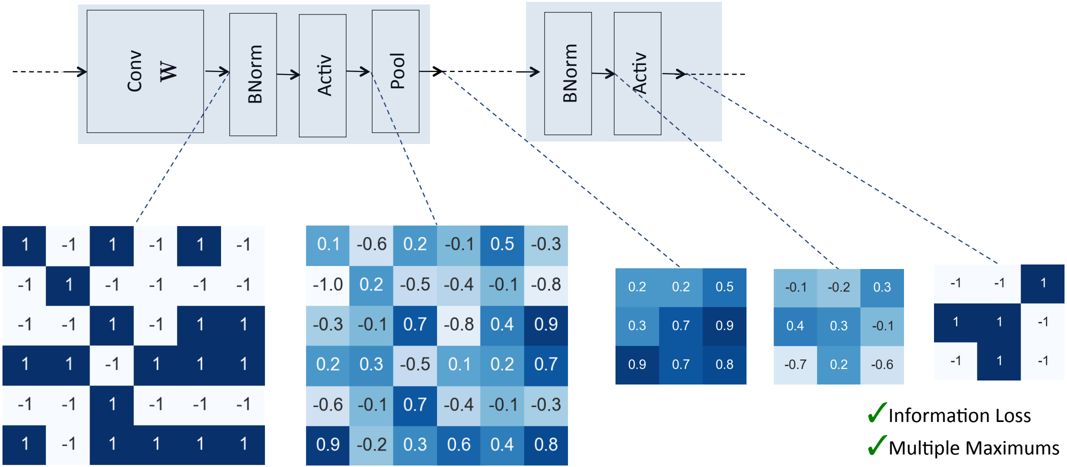
[1]Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Network Structure in XNOR-Networks[1]



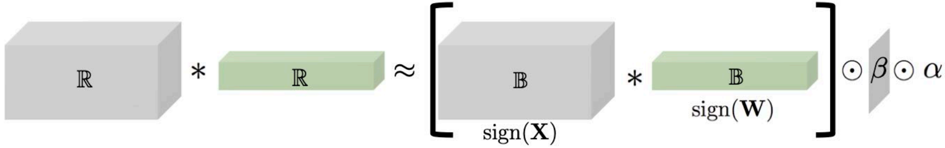
[1]Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.

Network Structure in XNOR-Networks

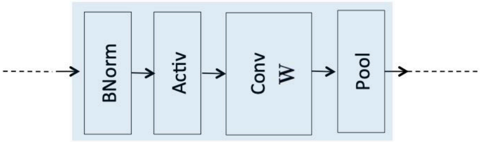


✓ Information Loss
✓ Multiple Maximums [1]

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

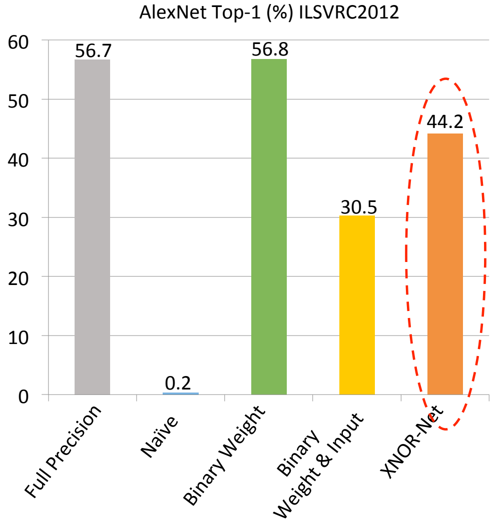


1. Randomly initialize $\mathbf{W}$
2. For $iter = 1$ to N
3. Load a random input image $\mathbf{X}$
4. $\mathbf{W}^{\mathrm{B}} = \mathrm{sign}(\mathbf{W})$
5. $\alpha = \frac{\|W\|_{\ell 1}}{n}$
6. Forward pass with $\alpha, \mathbf{W}^{\mathrm{B}}$
7. Compute loss function $\mathbf{C}$
8. $\frac{\partial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

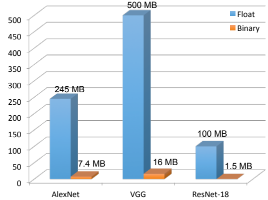


[1]

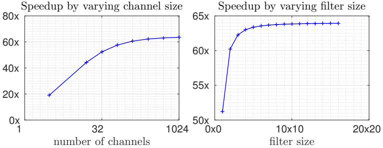
[1] Mohammad Rastegari et al. (2016). "XNOR-NET: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*, pp. 525–542.



✓ 32x Smaller Model



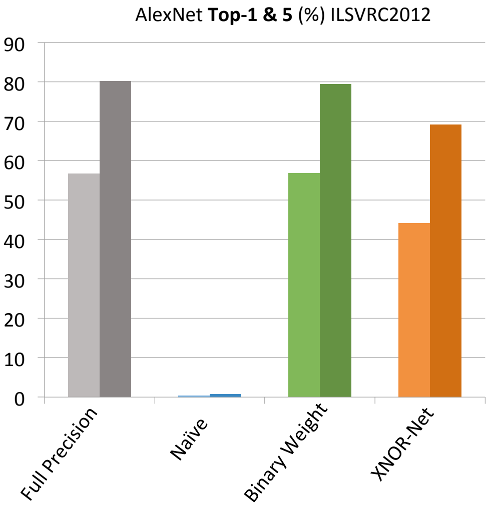
✓ 58x Less Computation



[1]

---

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



1

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

Motivation and Intuition

Motivation

- Naive methods (Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David (2015). "Binaryconnect: Training deep neural networks with binary weights during propagations". In: *Advances in neural information processing systems*, pp. 3123–3131, Matthieu Courbariaux, Itay Hubara, et al. (2016). "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1". In: *arXiv preprint arXiv:1602.02830*) suffer the accuracy loss

Intuition

- Quantized parameter should approximate the full precision parameter as closely as possible

Towards Accurate Binary Convolutional Neural Network

Contribution

- Approximate full-precision weights with the linear combination of multiple binary weight bases
- Introduce multiple binary activations

Weights Binarization

- Weights tensors in one layer: $W \in \mathbb{R}^{w \times h \times c_{in} \times c_{\text{out}}}$

$$B_1, B_2, \ldots, B_M \in \{-1, +1\}^{w \times h \times c_{in} x c_{out}}$$
$$W \approx \alpha_1 B_1 + \alpha_2 B_2 + \ldots + \alpha_M B_M$$
$$B_i = F_{u_i}(W) = \operatorname{sign}\left(\bar{W} + u_i \operatorname{std}(W)\right), i = 1, 2, \ldots, M$$

where $\bar{W} = W - mean(W)$, u_i is a shift parameter(e.g. $u_i = -1 + (i-1)\frac{2}{M-1}$)
α can be calculated via $\min_a J(\alpha) = \|W - B\alpha\|^2$

Forward and Backward

- Forward

$$B_1, B_2, \cdots, B_M = F_{u_1}(W), F_{w_2}(W), \cdots, F_{u,u}(W)$$
$$solve \min_a J(\alpha) = \|W - B\alpha\|^2 \; for \; \alpha$$
$$O = \sum_{m=1}^{M} \alpha_m \operatorname{Conv}(B_m, A)$$

- Backward

$$\frac{\partial c}{\partial W} = \frac{\partial c}{\partial O}\left(\sum_{m=1}^{M} \alpha_m \frac{\partial O}{\partial B_m}\frac{\partial B_m}{\partial W}\right) \overset{STE}{=} \frac{\partial c}{\partial O}\left(\sum_{m=1}^{M} \alpha_m \frac{\partial O}{\partial B_m}\right) = \sum_{m=1}^{M} \alpha_m \frac{\partial c}{\partial B_m}$$

Multiple Binary Activations

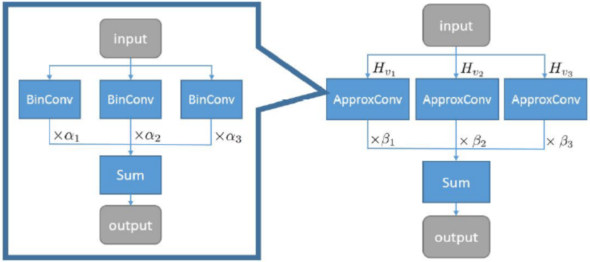
- Bounded Activation Function

$$h(x) \in [0,1]$$
$$h_r(x) = \text{clip}(x + v, 0, 1)$$
where v is a shift parameter

- Binarization Function

$$H_v(\boldsymbol{R}) := 2\mathbb{I}_{h_v(\boldsymbol{R}) \geq 0.5} - 1$$
$$A_1, A_2, \ldots, A_N = H_{v_1}(R), H_{v_2}(R), \ldots, H_{v_N}(R)$$
$$R \approx \beta_1 A_1 + \beta_2 A_2 + \ldots + \beta_N A_N$$
where R is the real-value activation

- $A_1, A_2, \ldots, A_N$ is the base to represent the real-valued activations



- ApproxConv is expected to approximate the conventional full-precision convolution with linear combination of binary convolutions
- The right part is the overall block structure of the convolution in ABC-Net.The input is binarized using different functions H_v1,H_v2,H_v3

$$\mathrm{Conv}(\boldsymbol{W}, \boldsymbol{R}) \approx \mathrm{Conv}\left(\sum_{m=1}^{M} \alpha_m \boldsymbol{B}_m, \sum_{n=1}^{N} \beta_n \boldsymbol{A}_n\right) = \sum_{m=1}^{M} \sum_{n=1}^{N} \alpha_m \beta_n \,\mathrm{Conv}\left(\boldsymbol{B}_m, \boldsymbol{A}_n\right)$$

Read the paper[2] if you want to learn the specific details of the algorithm

Towards Accurate Binary Convolutional Neural Network

Xiaofan Lin **Cong Zhao** **Wei Pan***
DJI Innovations Inc, Shenzhen, China
{xiaofan.lin, cong.zhao, wei.pan}@dji.com

[2] Xiaofan Lin, Cong Zhao, and Wei Pan (2017). “Towards accurate binary convolutional neural network”. In: *Advances in Neural Information Processing Systems*, pp. 345–353.

Overview

1. Minimize the Quantization Error
2. Reduce the Gradient Error

Motivation

- Although STE is often adopted to estimate the gradients in BP, there exists obvious gradient mismatch between the gradient of the binarization function
- With the restriction of STE, the parameters outside the range of $[-1 : +1]$ will not be updated.

Bi-real net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm

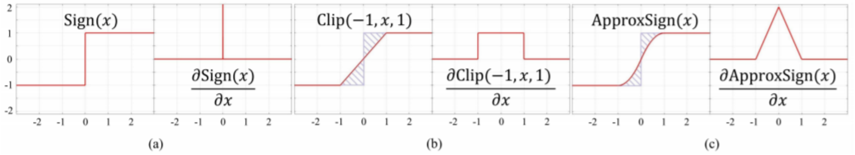
Naive Binarization Function

- Recall the partial derivative calculation in back propagation

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}_r^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial \mathbf{A}_b^{l,t}}{\partial \mathbf{A}_r^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial \operatorname{Sign}\left(\mathbf{A}_r^{l,t}\right)}{\partial \mathbf{A}_r^{l,t}} \approx \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial F\left(\mathbf{A}_r^{l,t}\right)}{\partial \mathbf{A}_r^{l,t}}$$

- *Sign* function is a non-differentiable function, so F is an approximation differentiable function of *Sign* function

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}_r^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial \mathbf{A}_b^{l,t}}{\partial \mathbf{A}_r^{l,t}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial \operatorname{Sign}\left(\mathbf{A}_r^{l,t}\right)}{\partial \mathbf{A}_r^{l,t}} \approx \frac{\partial \mathcal{L}}{\partial \mathbf{A}_b^{l,t}} \frac{\partial F\left(\mathbf{A}_r^{l,t}\right)}{\partial \mathbf{A}_r^{l,t}}$$



Approximation of *Sign* function

- Naive Approximation $F(x) = clip(x, 0, 1)$, see fig(b)
- More Precious Approximation in Bi-Real, see fig(c)

$$Approxsign(x) = \begin{cases} -1, & \text{if } x < -1 \\ 2x + x^2, & \text{if } -1 \leq x < 0 \\ 2x - x^2, & \text{if } 0 \leq x < 1 \\ 1, & \text{otherwise} \end{cases} \quad \frac{\partial Approxsign(x)}{\partial x} = \begin{cases} 2 + 2x, & \text{if } -1 \leq x < 0 \\ 2 - 2x, & \text{if } 0 \leq x < 1 \\ 0, & \text{otherwise} \end{cases}$$

Bi-Real

Read the paper[3] if you want to learn the specific details of the algorithm

Bi-Real Net: Enhancing the Performance of 1-bit CNNs With Improved Representational Capability and Advanced Training Algorithm

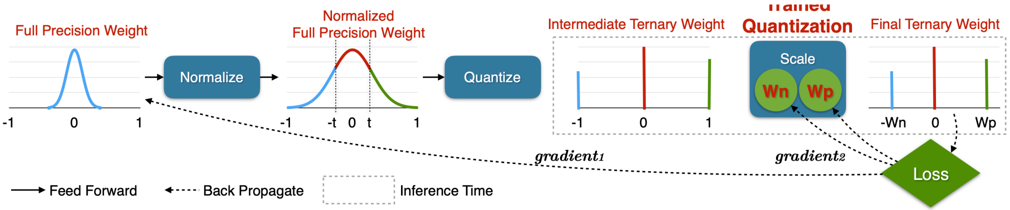
Zechun Liu[1], Baoyuan Wu[2], Wenhan Luo[2], Xin Yang[3]⋆, Wei Liu[2], and Kwang-Ting Cheng[1]

[1] Hong Kong University of Science and Technology
[2] Tencent AI lab
[3] Huazhong University of Science and Technology

[3]Zechun Liu et al. (2018). "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 722–737.

Trained ternary quantization

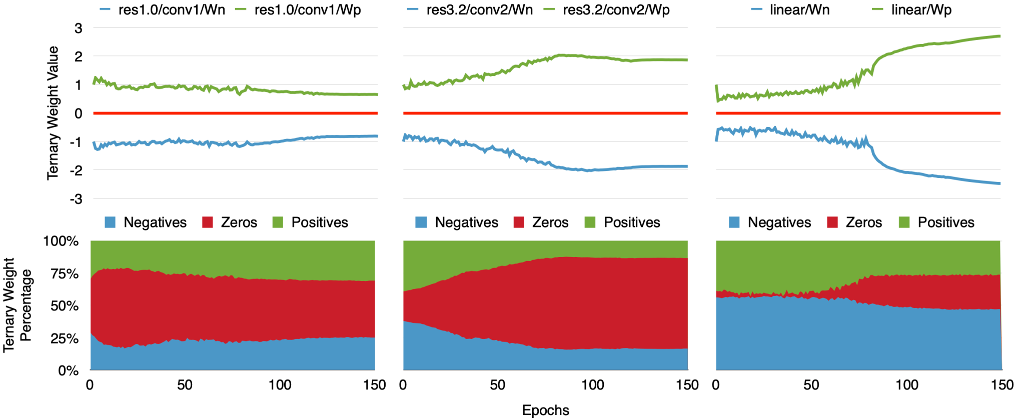
Trained Ternary Quantization[4]



Overview of the trained ternary quantization procedure.

[4]Chenzhuo Zhu et al. (2017). "Trained ternary quantization". In: *Proc. ICLR*.

Trained Ternary Quantization[4]



Ternary weights value (above) and distribution (below) with iterations for different layers of ResNet-20 on CIFAR-10.

[4]Chenzhuo Zhu et al. (2017). "Trained ternary quantization". In: *Proc. ICLR*.

Reading List

- Hyeonuk Kim et al. (2017). "A Kernel Decomposition Architecture for Binary-weight Convolutional Neural Networks". In: *Proc. DAC*, 60:1–60:6
- Jungwook Choi et al. (2018). "Pact: Parameterized clipping activation for quantized neural networks". In: *arXiv preprint arXiv:1805.06085*
- Dongqing Zhang et al. (2018). "Lq-nets: Learned quantization for highly accurate and compact deep neural networks". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 365–382
- Aojun Zhou et al. (2017). "Incremental network quantization: Towards lossless cnns with low-precision weights". In: *arXiv preprint arXiv:1702.03044*
- Zhaowei Cai et al. (2017). "Deep learning with low precision by half-wave gaussian quantization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5918–5926