XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

POTAPOVA POLINA

Good results in object detection and recognition

+

Progress in smart wearable devices

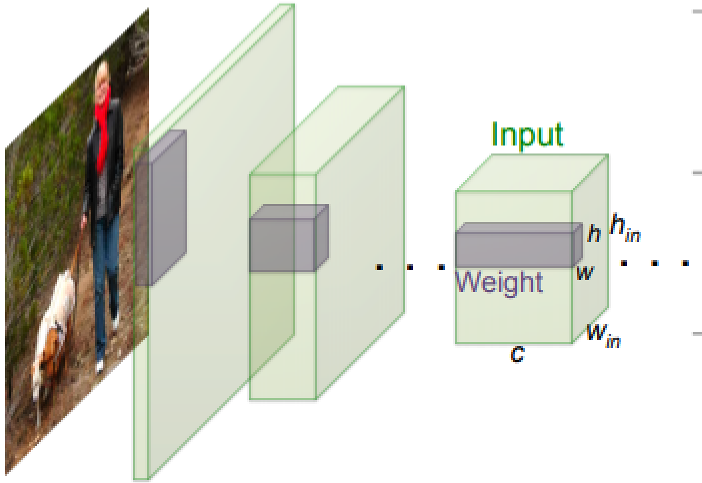
=

Good idea

Good results in object detection and recognition + Progress in smart wearable devices = Good idea

Problem: CNN need large amounts of memory and computational power. Small devices often don’t have this.

Binary-Weight-Networks and XNOR-Networks



	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights: 0.12 -1.2 ... 0.41 / -0.2 0.5 ... 0.68	+ , − , ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	+ , −	~32x	~2x	%56.8
BinaryWeight Binary Input (**XNOR-Net**)	Binary Inputs: 1 -1 ... -1 / -1 1 ... 1; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	XNOR , bitcount	~32x	~58x	%44.2

Existing approaches

- Shallow networks

Existing approaches

- Shallow networks
- Compressing pre-trained deep networks:

Existing approaches

- Shallow networks
- Compressing pre-trained deep networks
- Designing compact layers

Existing approaches

- Shallow networks
- Compressing pre-trained deep networks
- Designing compact layers
- Quantizing parameters

Existing approaches

- Shallow networks
- Compressing pre-trained deep networks
- Designing compact layers
- Quantizing parameters
- Network binarization

Binary Convolutional Neural Network

- Binary-Weight-Networks

$$\mathbf{W} \approx \alpha \mathbf{B}$$

$$\mathbf{I} * \mathbf{W} \approx (\mathbf{I} \oplus \mathbf{B})\, \alpha$$

$$\mathbf{W} \in \mathbb{R}^{c \times w \times h}$$

$$\mathbf{B} \in \{+1, -1\}^{c \times w \times h}$$

- Estimating binary weights

$$J(\mathbf{B}, \alpha) = \|\mathbf{W} - \alpha \mathbf{B}\|^2$$

$$\alpha^*, \mathbf{B}^* = \underset{\alpha, \mathbf{B}}{\operatorname{argmin}} J(\mathbf{B}, \alpha)$$

Binary Convolutional Neural Network

- Binary-Weight-Networks

$$\mathbf{W} \approx \alpha \mathbf{B}$$

$$\mathbf{I} * \mathbf{W} \approx (\mathbf{I} \oplus \mathbf{B})\,\alpha$$

$$\mathbf{W} \in \mathbb{R}^{c \times w \times h}$$

$$\mathbf{B} \in \{+1, -1\}^{c \times w \times h}$$

- Estimating binary weights

$$J(\mathbf{B}, \alpha) = \|\mathbf{W} - \alpha \mathbf{B}\|^2$$

$$\alpha^*, \mathbf{B}^* = \underset{\alpha, \mathbf{B}}{\operatorname{argmin}} J(\mathbf{B}, \alpha)$$

$$\alpha^* = \frac{\mathbf{W}^{\mathsf{T}} \operatorname{sign}(\mathbf{W})}{n} = \frac{\sum |\mathbf{W}_i|}{n} = \frac{1}{n}\|\mathbf{W}\|_{\ell 1}$$

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

Algorithm

Algorithm 1 Training an L-layers CNN with binary weights:

Input: A minibatch of inputs and targets $(\mathbf{I}, \mathbf{Y})$, cost function $C(\mathbf{Y}, \hat{\mathbf{Y}})$, current weight $\mathcal{W}^t$ and current learning rate η^t.

Output: updated weight $\mathcal{W}^{t+1}$ and updated learning rate η^{t+1}.

Binarizing weight filters:
for $l = 1$ to L **do**
  for k^{th} filter in l^{th} layer **do**
    $\mathcal{A}_{lk} = \frac{1}{n}\|\mathcal{W}^t_{lk}\|_{\ell 1}$
    $\mathcal{B}_{lk} = \text{sign}(\mathcal{W}^t_{lk})$
    $\widetilde{\mathcal{W}}_{lk} = \mathcal{A}_{lk}\mathcal{B}_{lk}$
$\hat{\mathbf{Y}} = \textbf{BinaryForward}(\mathbf{I}, \mathcal{B}, \mathcal{A})$ // standard forward propagation except that convolutions are computed using equation 1 or 11
$\frac{\partial C}{\partial \widetilde{\mathcal{W}}} = \textbf{BinaryBackward}(\frac{\partial C}{\partial \hat{\mathbf{Y}}}, \widetilde{\mathcal{W}})$ // standard backward propagation except that gradients are computed using $\widetilde{\mathcal{W}}$ instead of $\mathcal{W}^t$
$\mathcal{W}^{t+1} = \textbf{UpdateParameters}(\mathcal{W}^t, \frac{\partial C}{\partial \widetilde{\mathcal{W}}}, \eta_t)$ // Any update rules (*e.g.*,SGD or ADAM)
$\eta^{t+1} = \textbf{UpdateLearningrate}(\eta^t, t)$ // Any learning rate scheduling function

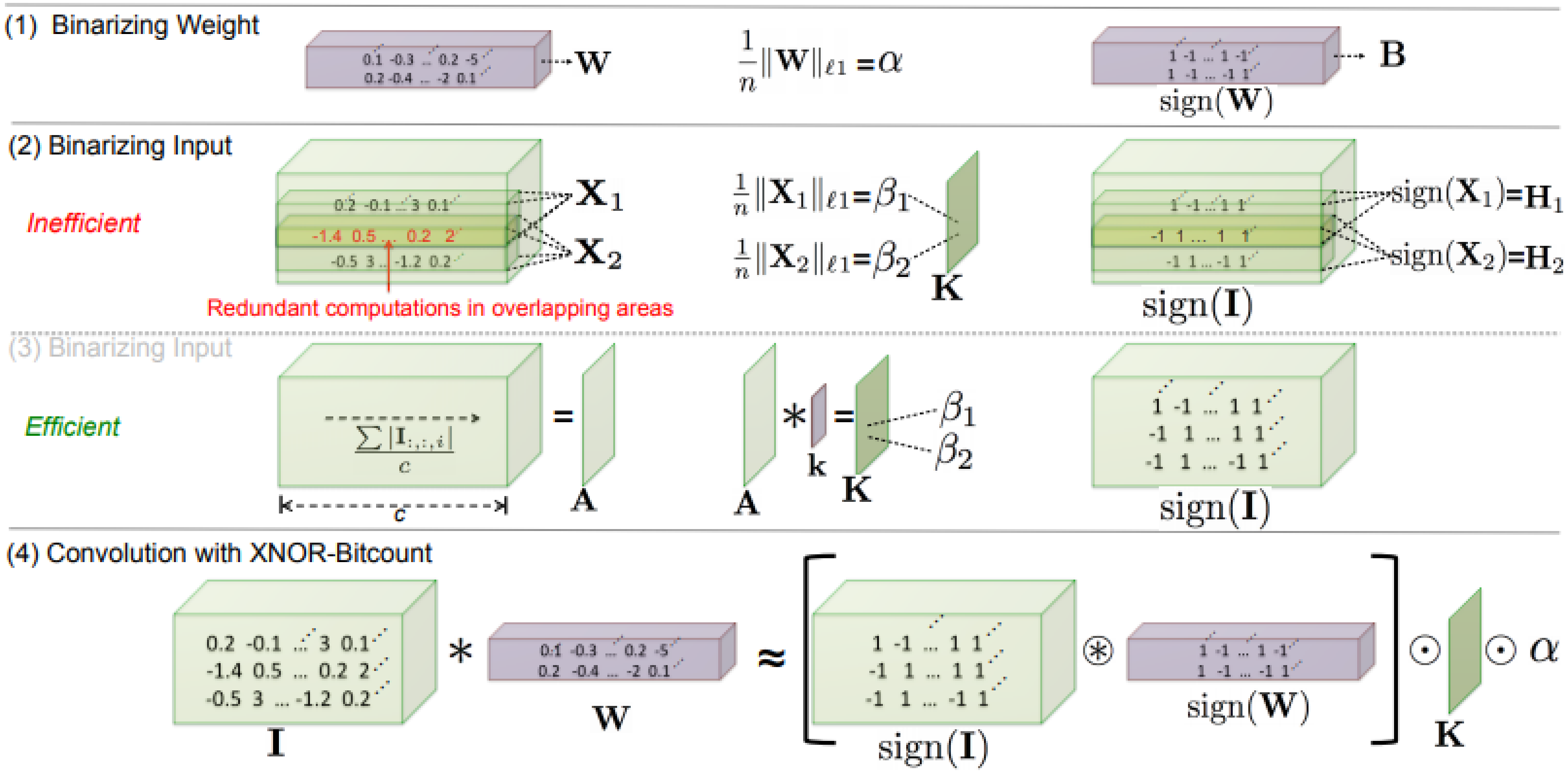
---

XNOR-Networks

- Binary Dot Product

$$\mathbf{X}^{\intercal}\mathbf{W} \approx \beta\mathbf{H}^{\intercal}\alpha\mathbf{B}, \text{ where } \mathbf{H}, \mathbf{B} \in \{+1, -1\}^n$$

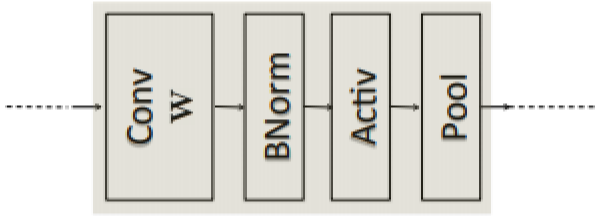
$$\alpha^*, \mathbf{B}^*, \beta^*, \mathbf{H}* = \underset{\alpha,\mathbf{B},\beta,\mathbf{H}}{\operatorname{argmin}} \|\mathbf{X} \odot \mathbf{W} - \beta\alpha\mathbf{H} \odot \mathbf{B}\|$$



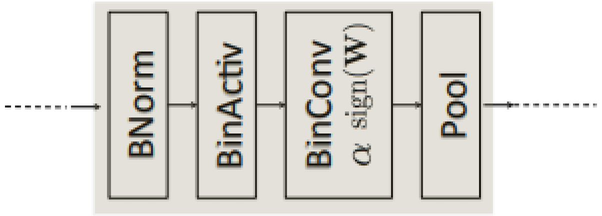
(1) Binarizing Weight
W
B
sign(W)
(2) Binarizing Input
Inefficient
Redundant computations in overlapping areas
sign(I)
(3) Binarizing Input
Efficient
A
k
K
sign(I)
(4) Convolution with XNOR-Bitcount
I
W
sign(I)
sign(W)
K

- Binary Convolution

$$\mathbf{I} * \mathbf{W} \approx (\operatorname{sign}(\mathbf{I}) \circledast \operatorname{sign}(\mathbf{W})) \odot \mathbf{K}\alpha \qquad \mathbf{K} = \mathbf{A} * \mathbf{k}, \text{ where } \forall ij \;\; \mathbf{k}_{ij} = \frac{1}{w \times h}$$

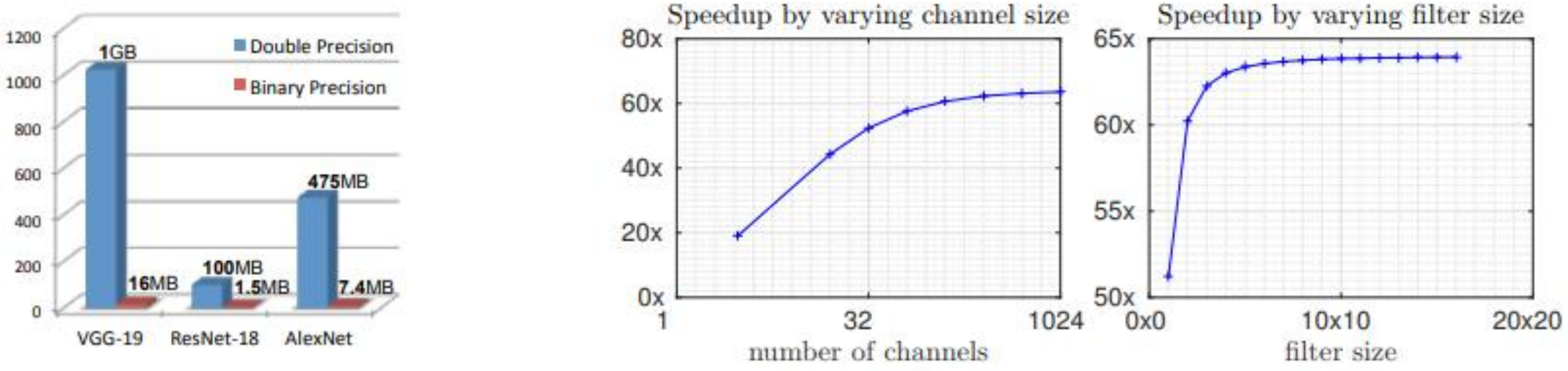


A typical block in CNN



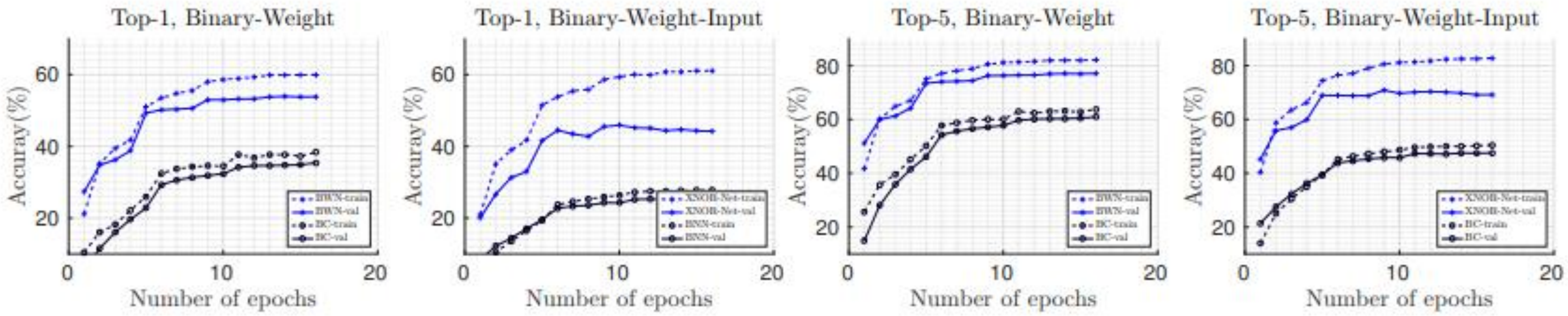
A block in XNOR-Net

Experiments

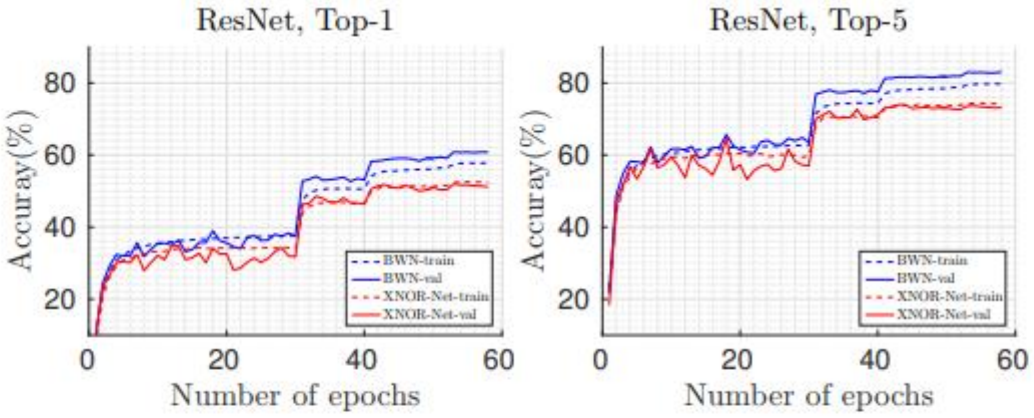


(a) (b) (c)

Experiments



Experiments



(a) (b)

Any question?