Training Binary Neural Networks

11月8日

周争光

二值化

- 二值化
  - 浮点数→{-1,1}
- 权值二值化
  - 卷积乘法→加法与减法
  - 减少~32倍参数量
  - ~2倍理论加速
- 权值二值化+激活值二值化
  - 卷积乘法→位运算（XNOR，bitcount）
  - 减少~32倍参数量
  - ~58倍理论加速

BinaryConnect

- 权值二值化
  - 确定方式
  - 随机方式

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

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

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

- 反向传播与梯度计算
  - 保留*w*浮点数值
  - 梯度累加，将超出[-1, 1]的部分裁剪

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

[1] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In NIPS, 2015.

BinaryConnect

- 训练算法

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

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

Ensure: updated parameters w_t and b_t.

1. Forward propagation:

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

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

2. Backward propagation:

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

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

3. Parameter update:

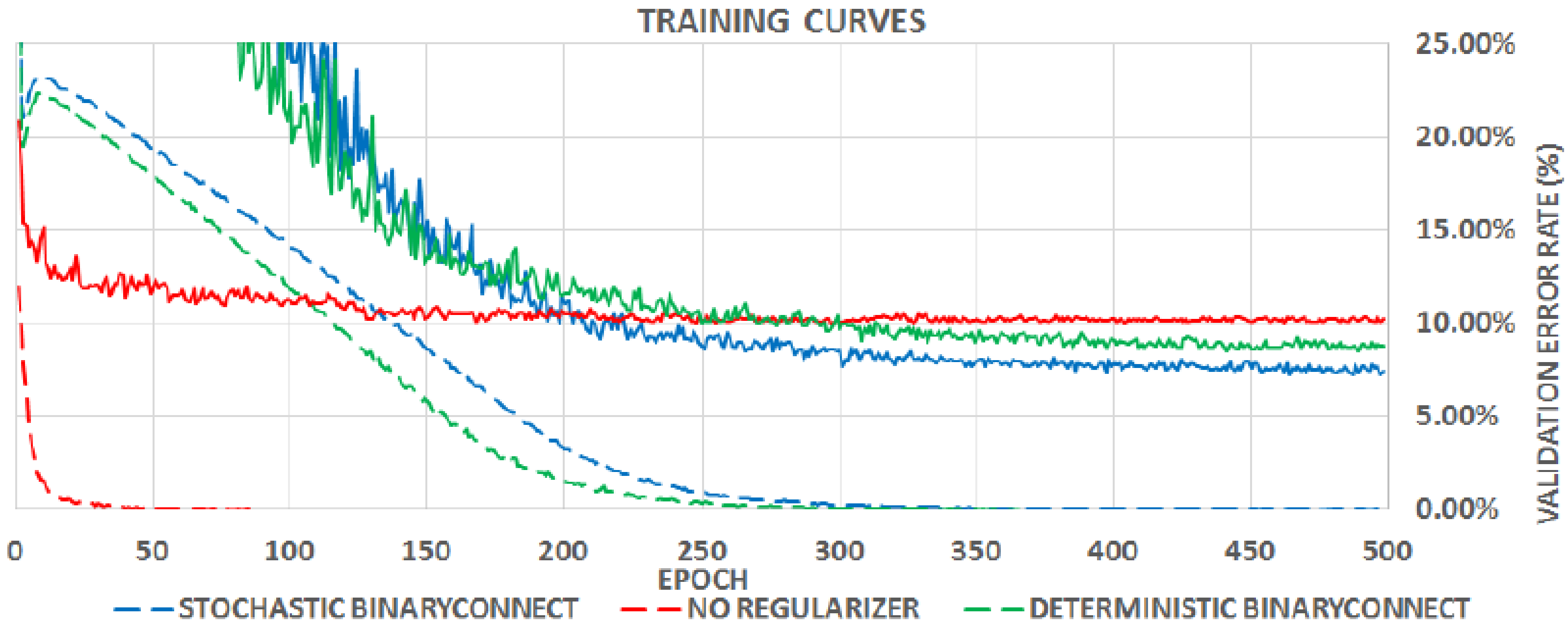
Compute $\frac{\partial C}{\partial w_b}$ and $\frac{\partial C}{d b_{t-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and a_{k-1}

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

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

BinaryConnect

- 实验结果



- 二值化相当于给权值添加了噪声，具有正则化作用，可以防止模型过拟合

Binarized Neural Networks (BNN)

- 量化函数（权值+激活值）

$$x^b = \mathrm{Sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise}, \end{cases}$$

  - 有确定和随机方式两种
  - 但随机形式需要生成随机数耗时，一般都采用确定形式的

- 对sign求导（“straight through estimator”）

$$q = \mathrm{Sign}(r),$$

$$g_r = g_q 1_{|r| \leq 1}.$$

$$\mathrm{Htanh}(x) = \mathrm{Clip}(x, -1, 1)$$

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016.

BNN

- 训练
- 运行

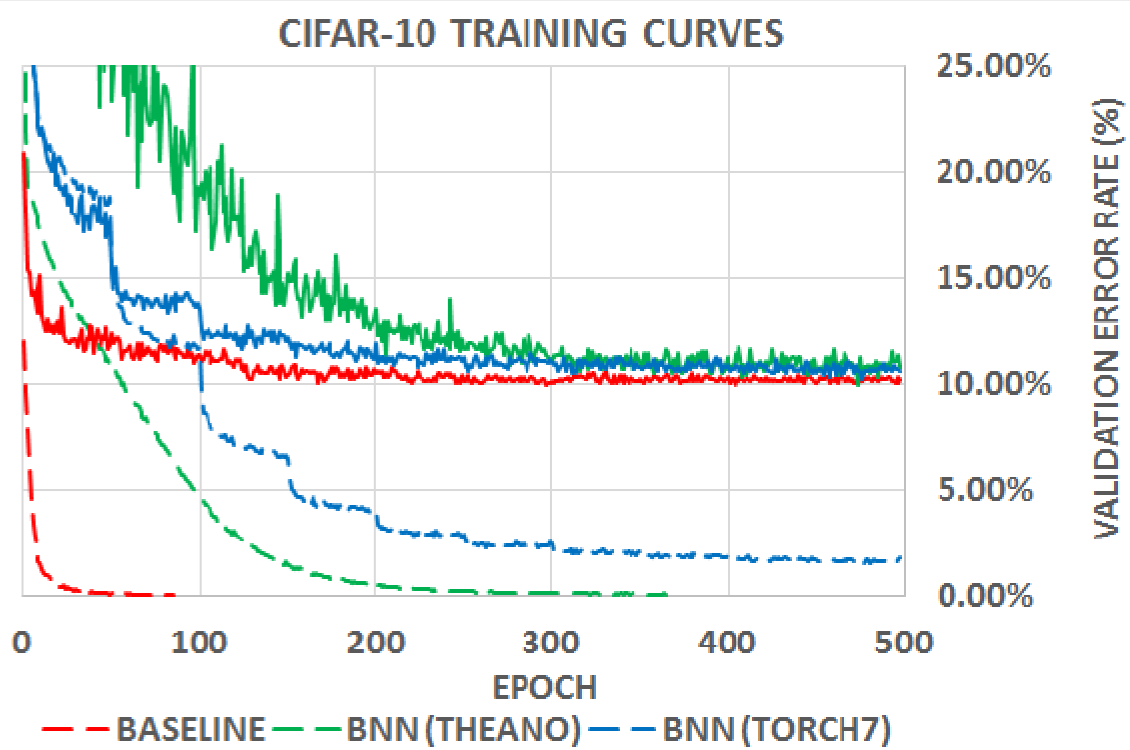
{1. Computing the parameters gradients:}
{1.1. Forward propagation:}
for $k = 1$ to L **do**
  $W_k^b \leftarrow \text{Binarize}(W_k)$
  $s_k \leftarrow a_{k-1}^b W_k^b$
  $a_k \leftarrow \text{BatchNorm}(s_k, \theta_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow \text{Binarize}(a_k)$
  end if
end for
{1.2. Backward propagation:}
{Please note that the gradients are not binary.}
Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*
for $k = L$ to 1 **do**
  if $k < L$ **then**
    $g_{a_k} \leftarrow g_{a_k^b} \circ 1_{|a_k| \leq 1}$
  end if
  $(g_{s_k}, g_{\theta_k}) \leftarrow \text{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$
  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
  $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$
end for
{2. Accumulating the parameters gradients:}
for $k = 1$ to L **do**
  $\theta_k^{t+1} \leftarrow \text{Update}(\theta_k, \eta, g_{\theta_k})$
  $W_k^{t+1} \leftarrow \text{Clip}(\text{Update}(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$
  $\eta^{t+1} \leftarrow \lambda \eta$
end for

Algorithm 5 Running a BNN. L is the number of layers.

Require: a vector of 8-bit inputs a_0, the binary weights W^b, and the BatchNorm parameters θ.
Ensure: the MLP output a_L.
  {1. First layer:}
  $a_1 \leftarrow 0$
  for $n = 1$ to 8 **do**
    $a_1 \leftarrow a_1 + 2^{n-1} \times \text{XnorDotProduct}(\mathrm{a}_0^{\mathrm{n}}, \mathrm{W}_1^{\mathrm{b}})$
  end for
  $a_1^b \leftarrow \text{Sign}(\text{BatchNorm}(a_1, \theta_1))$
  {2. Remaining hidden layers:}
  for $k = 2$ to $L - 1$ **do**
    $a_k \leftarrow \text{XnorDotProduct}(a_{k-1}^b, W_k^b)$
    $a_k^b \leftarrow \text{Sign}(\text{BatchNorm}(a_k, \theta_k))$
  end for
  {3. Output layer:}
  $a_L \leftarrow \text{XnorDotProduct}(a_{L-1}^b, W_L^b)$
  $a_L \leftarrow \text{BatchNorm}(a_L, \theta_L)$

BNN

□ 实验结果

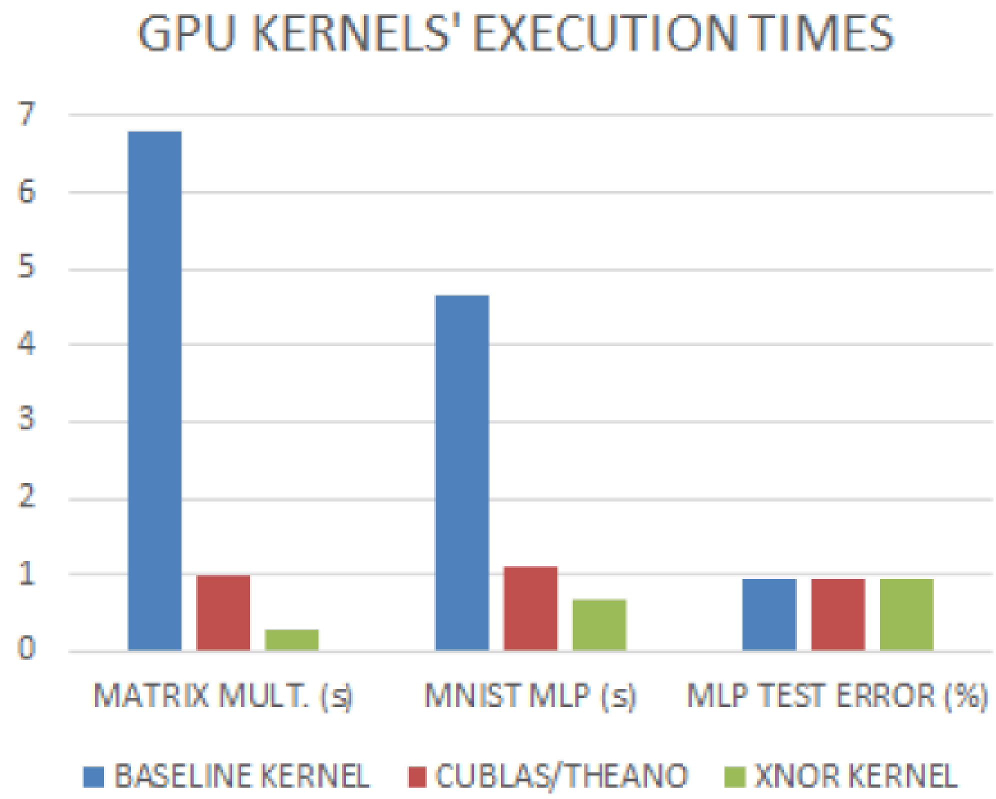

CIFAR-10 TRAINING CURVES
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
VALIDATION ERROR RATE (%)
0
100
200
300
400
500
EPOCH
BASELINE
BNN (THEANO)
BNN (TORCH7)

BNN

- 实现优化

$$a_1 += \text{popcount}(\text{xnor}(a_0^{32b}, w_1^{32b})),$$

  - 位操作使用SIMD指令加速，将32二值数合并在一个32位寄存器中，从而32个时钟现在可用1+4+1=6个时钟完成



XNOR-Net

- 提出了两种二值网络
  - Binary-Weight-Networks (BWN)：只二值化权值
  - XNOR-Networks (XNOR-Net)：二值化权值和激活值
  - 由于引入缩放因子，BWN和XNOR-Net不是纯粹的二值网络

	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

[3] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In ECCV, 2016.

BWN

- 近似

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

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

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

- 优化

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

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

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

- 求解

$$\mathbf{B}^* = \underset{\mathbf{B}}{\operatorname{argmax}}\{\mathbf{W}^\top \mathbf{B}\} \quad s.t. \;\; \mathbf{B} \in \{+1, -1\}^n$$

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

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

BWN

- 训练

$$\frac{\partial \text{sign}}{\partial r} = r 1_{|r| \leq 1}$$

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}.

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

XNOR-Net

- 优化

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

$$\gamma^*, \mathbf{C}^* = \underset{\gamma,\mathbf{C}}{\operatorname{argmin}} \|\mathbf{Y} - \gamma\mathbf{C}\|$$

- 求解同BWN

$$\mathbf{C}^* = \operatorname{sign}(\mathbf{Y}) = \operatorname{sign}(\mathbf{X}) \odot \operatorname{sign}(\mathbf{W}) = \mathbf{H}^* \odot \mathbf{B}^*$$

$$\gamma^* = \frac{\sum |\mathbf{Y}_i|}{n} = \frac{\sum |\mathbf{X}_i||\mathbf{W}_i|}{n} \approx \left(\frac{1}{n}\|\mathbf{X}\|_{\ell 1}\right)\left(\frac{1}{n}\|\mathbf{W}\|_{\ell 1}\right) = \beta^*\alpha^*$$

- 卷积实现

$$\mathbf{I} * \mathbf{W} \approx (\operatorname{sign}(\mathbf{I}) \circledast \operatorname{sign}(\mathbf{W})) \odot \mathbf{K}\alpha$$

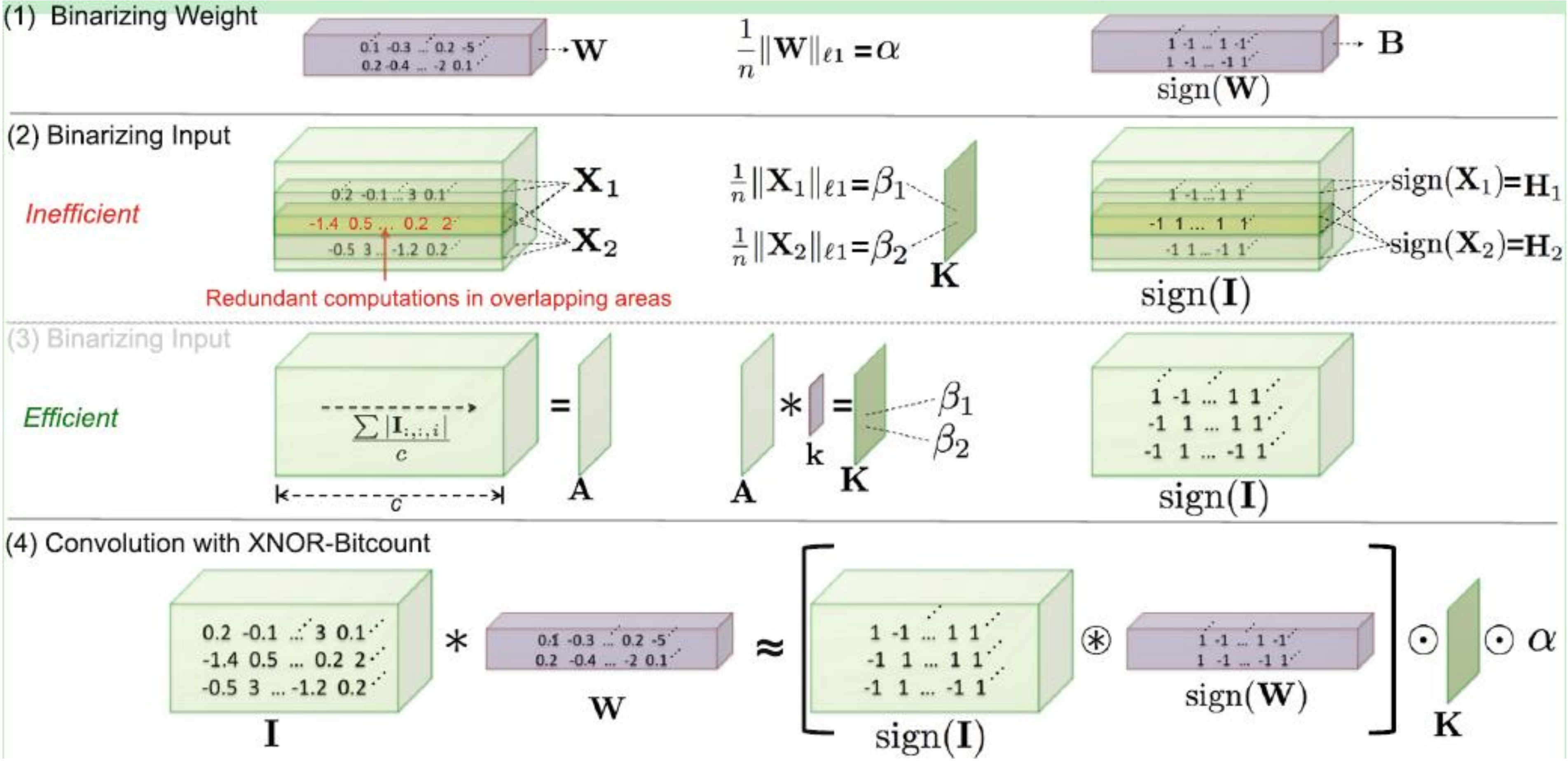
XNOR-Net

- 卷积实现

$$\mathbf{I} * \mathbf{W} \approx (\operatorname{sign}(\mathbf{I}) \circledast \operatorname{sign}(\mathbf{W})) \odot \mathbf{K}\alpha$$

$$\mathbf{A} = \frac{\sum |\mathbf{I}_{:,:,i}|}{c}$$

$$\mathbf{K} = \mathbf{A} * \mathbf{k}, \quad \forall ij \ \ \mathbf{k}_{ij} = \frac{1}{w \times h}.$$



XNOR-Net

网络结构

A typical block in CNN　　　　A block in XNOR-Net

性能分析

- 一个卷积含有 $cN_{\mathbf{W}}N_{\mathbf{I}}$ 个二值运算和 $N_{\mathbf{I}}$ 个非二值运算
- 一个CPU时钟可运行64个二值运算，因此计算加速为：

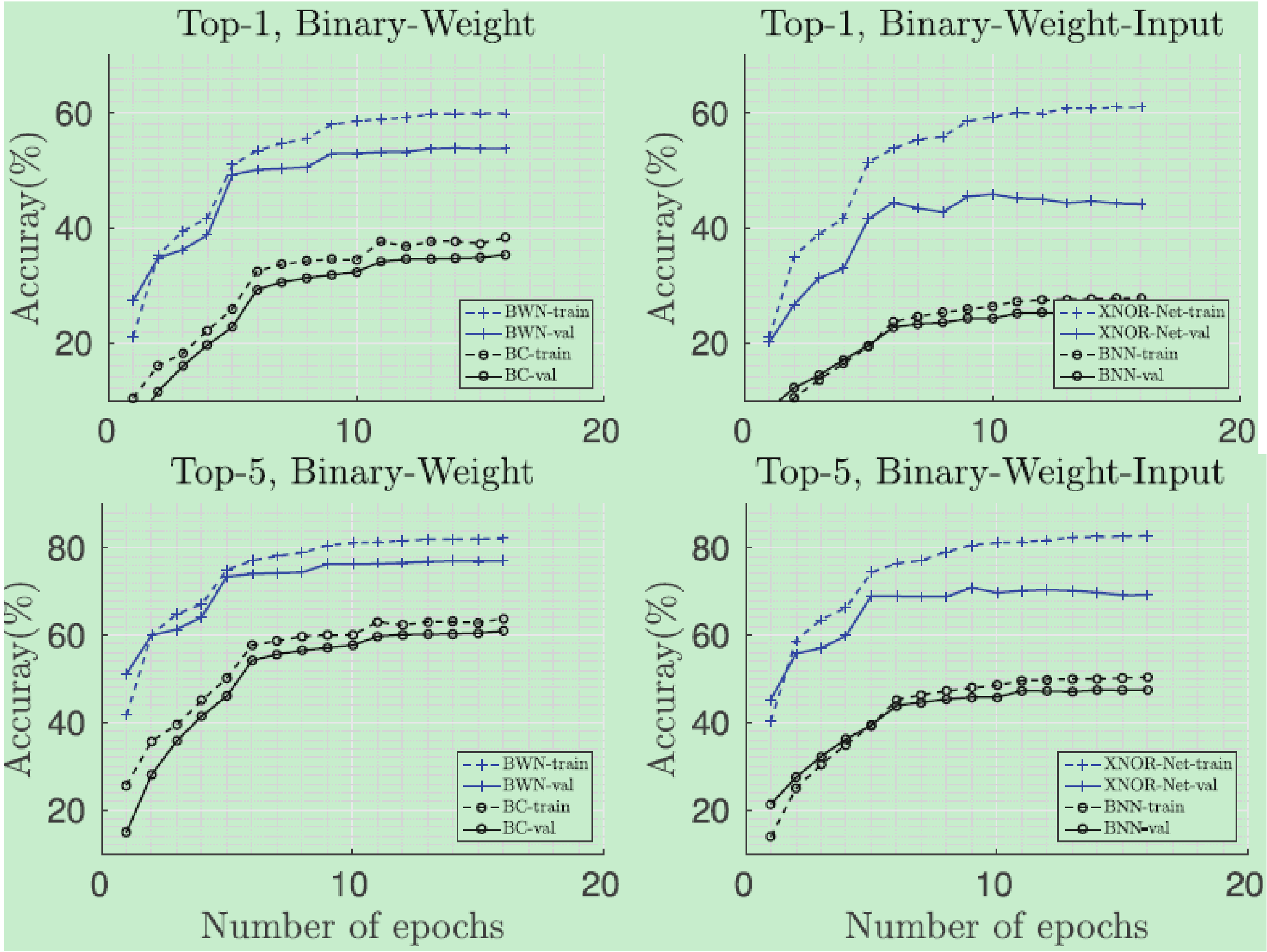
$$S = \frac{cN_{\mathbf{W}}N_{\mathbf{I}}}{\frac{1}{64}cN_{\mathbf{W}}N_{\mathbf{I}} + N_{\mathbf{I}}} = \frac{64cN_{\mathbf{W}}}{cN_{\mathbf{W}} + 64}.$$

- 获得62.27倍理论加速但实际只有58倍加速

$$c = 256,\ n_{\mathbf{I}} = 14^2 \text{ and } n_{\mathbf{W}} = 3^2$$

实验结果

- 与BC和BNN方法对比



Ternary weight networks

- 三值网络（-1,0,1）
- 优化

$$\begin{cases} \alpha^*, \mathbf{W}^{t*} = & \underset{\alpha, \mathbf{W}^t}{\arg\min}\, J(\alpha, \mathbf{W}^t) = \|\mathbf{W} - \alpha \mathbf{W}^t\|_2^2 \\ \text{s.t.} & \alpha \geq 0, \mathrm{W}_i^t \in \{-1, 0, 1\}, i = 1, 2, \ldots, n. \end{cases}$$

- 不好求解，近似

$$\mathrm{W}_i^t = f_t(\mathrm{W}_i|\Delta) = \begin{cases} +1, & \text{if } \mathrm{W}_i > \Delta \\ 0, & \text{if } |\mathrm{W}_i| \leq \Delta \\ -1, & \text{if } \mathrm{W}_i < -\Delta \end{cases}$$

$$\alpha^*, \Delta^* = \underset{\alpha \geq 0, \Delta > 0}{\arg\min} \left(|\mathbf{I}_\Delta| \alpha^2 - 2 \left(\sum_{i \in \mathbf{I}_\Delta} |\mathrm{W}_i| \right) \alpha + c_\Delta \right)$$

$$\alpha_\Delta^* = \frac{1}{|\mathbf{I}_\Delta|} \sum_{i \in \mathbf{I}_\Delta} |\mathrm{W}_i|.$$

$$\Delta^* = \underset{\Delta > 0}{\arg\max} \frac{1}{|\mathbf{I}_\Delta|} \left(\sum_{i \in \mathbf{I}_\Delta} |\mathrm{W}_i| \right)^2$$

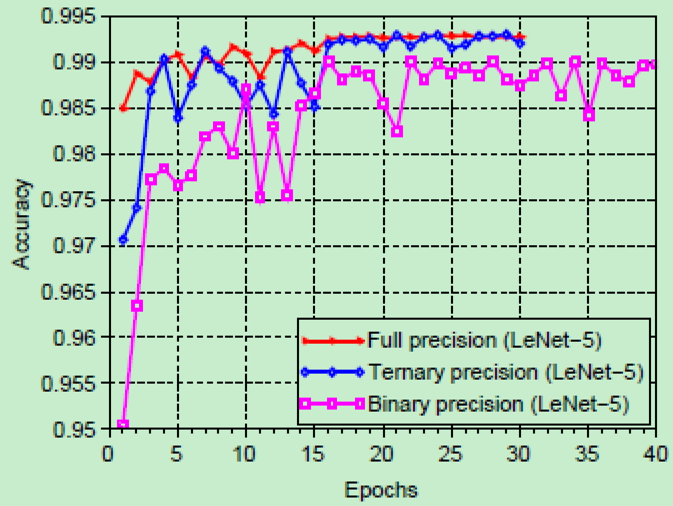
$$\Delta^* \approx 0.7 \cdot \mathrm{E}(|\mathbf{W}|) \approx \frac{0.7}{n} \sum_{i=1}^n |\mathrm{W}_i|$$

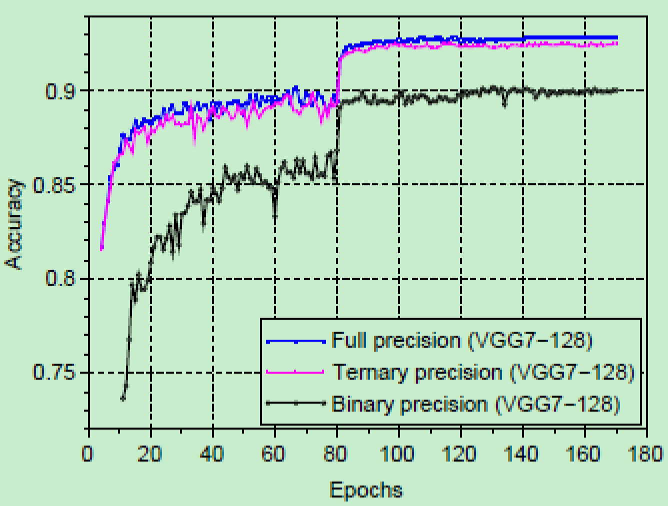
[4] F. Li, B. Zhang, and B. Liu. Ternary weight networks. arXiv preprint arXiv:1605.04711, 2016.

TWN

实验结果

	MNIST	CIFAR-10	ImageNet (top-1)	ImageNet (top-5)
TWNs	**99.35**	**92.56**	**61.8 / 65.3**	**84.2 / 86.2**
BPWNs	99.05	90.18	57.5 / 61.6	81.2 / 83.9
FPWNs	99.41	92.88	65.4 / 67.6	86.76 / 88.0
BinaryConnect	98.82	91.73	-	-
Binarized Neural Networks	88.6	89.85	-	-
Binary Weight Networks	-	-	60.8	83.0
XNOR-Net	-	-	51.2	73.2





TTQ

- Learn the ternary assignments and ternary values.
- 三值量化

$$w_l^t = \begin{cases} W_l^p : \tilde{w}_l > \Delta_l \\ 0 : |\tilde{w}_l| \le \Delta_l \\ -W_l^n : \tilde{w}_l < -\Delta_l \end{cases}$$

- 求导：权值和缩放因子

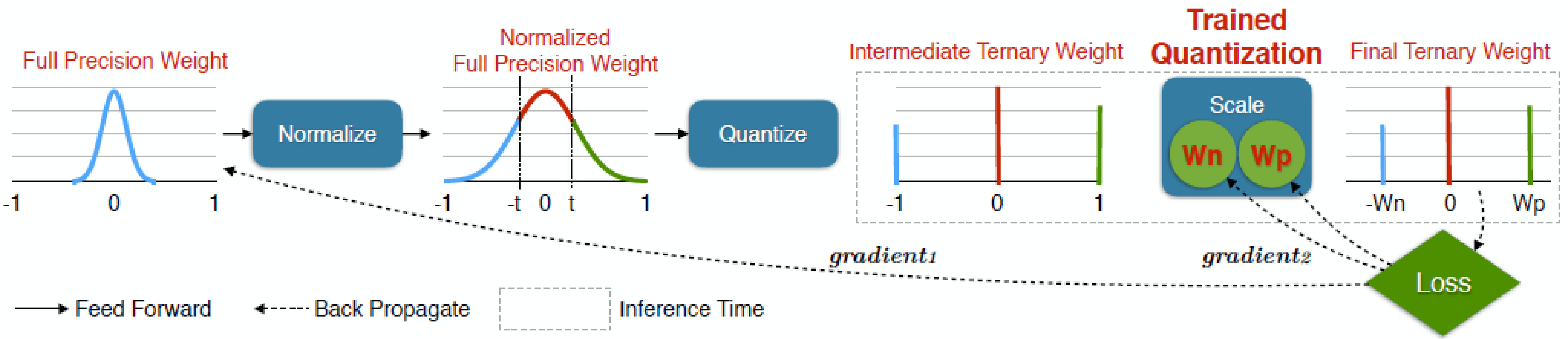
$$\frac{\partial L}{\partial W_l^p} = \sum_{i \in I_l^p} \frac{\partial L}{\partial w_l^t(i)}, \frac{\partial L}{\partial W_l^n} = \sum_{i \in I_l^n} \frac{\partial L}{\partial w_l^t(i)}$$

$$\frac{\partial L}{\partial \tilde{w}_l} = \begin{cases} W_l^p \times \frac{\partial L}{\partial w_l^t} : \tilde{w}_l > \Delta_l \\ 1 \times \frac{\partial L}{\partial w_l^t} : |\tilde{w}_l| \le \Delta_l \\ W_l^n \times \frac{\partial L}{\partial w_l^t} : \tilde{w}_l < -\Delta_l \end{cases}$$

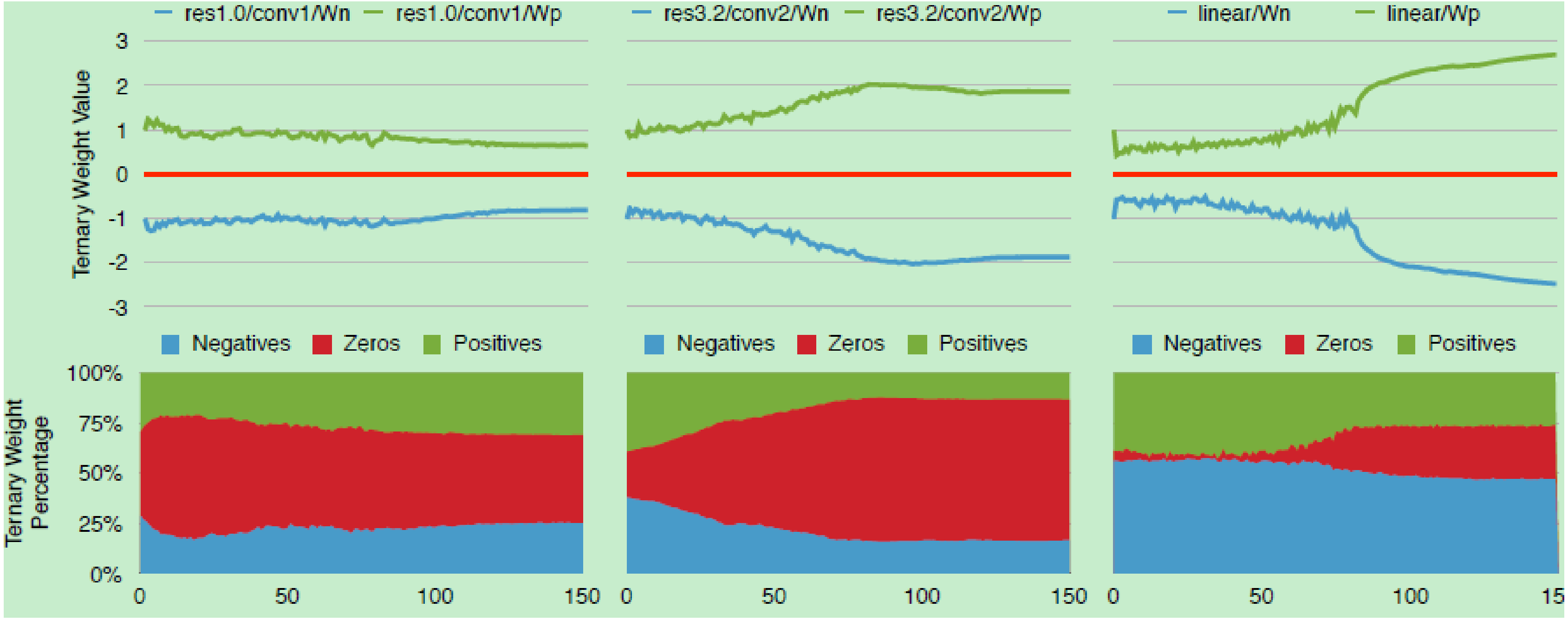
- 阈值选择
  - 依据该层最大权值 $\Delta_l = t \times \max(|\tilde{w}|)$
  - 保持一个固定的稀疏度r

[5] C. Zhu, S. Han, H. Mao, and W. Dally. Trained ternary quantization. ICLR, 2017.

TTQ



Full Precision Weight
Normalize
Normalized
Full Precision Weight
Quantize
Intermediate Ternary Weight
Trained
Quantization
Scale
Wn
Wp
Final Ternary Weight
-1
0
1
-t
t
-Wn
Wp
gradient1
gradient2
Loss
Feed Forward
Back Propagate
Inference Time



res1.0/conv1/Wn
res1.0/conv1/Wp
res3.2/conv2/Wn
res3.2/conv2/Wp
linear/Wn
linear/Wp
Ternary Weight Value
Negatives
Zeros
Positives
Ternary Weight Percentage
100%
75%
50%
25%
0%
0
50
100
150

TTQ

Error	Full precision	1-bit (DoReFa)	2-bit (TWN)	2-bit (Ours)
Top1	42.8%	46.1%	45.5%	**42.5%**
Top5	19.7%	23.7%	23.2%	**20.3%**

Table 2: Top1 and Top5 error rate of AlexNet on ImageNet

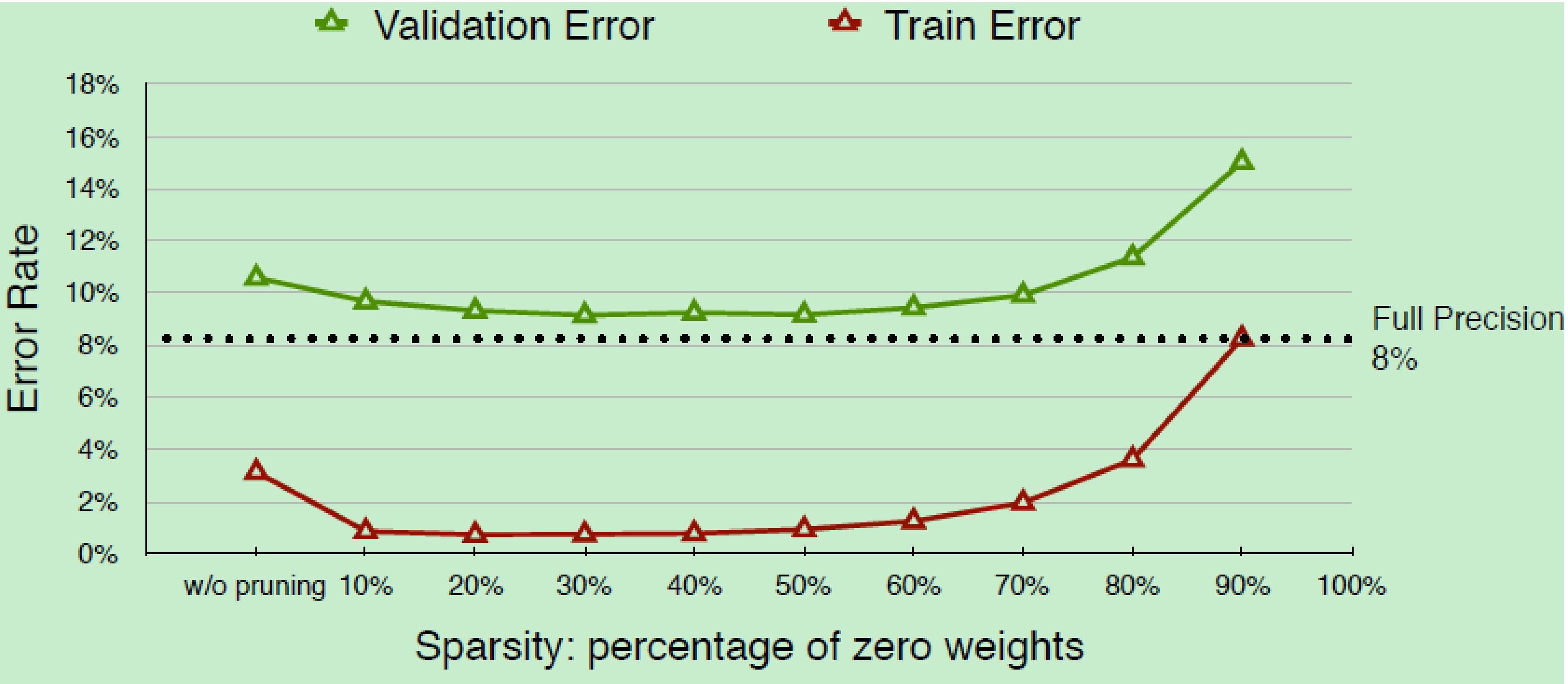


Figure 5: Accuracy v.s. Sparsity on ResNet-20

DoReFa-Net

- 在BNN和XNOR-Net中，梯度保持浮点型
- DoReFa-Net
  - 对权值、激活值、梯度均做量化
  - 量化到任意比特，

$$\textbf{Forward: } r_o = \frac{1}{2^k - 1}\operatorname{round}((2^k - 1)r_i)$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}.$$

  - STE直通估计：BNN，XNOR-Net

$$\textbf{Forward: } r_o = \operatorname{sign}(r_i)$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}\mathbb{I}_{|r_i|\le 1}.$$

$$\textbf{Forward: } r_o = \operatorname{sign}(r_i) \times \mathbf{E}_F(|r_i|)$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}.$$

[6] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160, 2016.

DoReFa-Net

- 二值化：一层只分配一个缩放因子

$$\textbf{Forward: } r_o = \operatorname{sign}(r_i) \times \mathbf{E}(|r_i|)$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}.$$

- k比特量化

$$\textbf{Forward: } r_o = f_\omega^k(r_i) = 2\,\mathrm{quantize_k}\left(\frac{\tanh(r_i)}{2\max(|\tanh(r_i)|)} + \frac{1}{2}\right) - 1.$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial r_o}{\partial r_i}\frac{\partial c}{\partial r_o} \quad \boxed{4}$$

- 激活值量化

1. $h(x) = \frac{\tanh(x)+1}{2}$
2. $h(x) = \operatorname{clip}(x, 0, 1)$
3. $h(x) = \min(1, |x|)$

$$f_\alpha^k(r) = \mathrm{quantize_k}(r).$$

DoReFa-Net

- 量化梯度（随机方式）

$$f_\gamma^k(\mathrm{d}r) = 2\max_0(|\mathrm{d}r|)\left[\mathrm{quantize}_k\left[\frac{\mathrm{d}r}{2\max_0(|\mathrm{d}r|)} + \frac{1}{2} + N(k)\right] - \frac{1}{2}\right].$$

$$N(k) = \frac{\sigma}{2^k-1} \text{ where } \sigma \sim Uniform(-0.5, 0.5).$$

- 合并非线性函数和量化函数

for $k = 1$ **to** L **do**
    $W_k^b \leftarrow f_\omega^W(W_k)$
    $\tilde{a}_k \leftarrow \texttt{forward}(a_{k-1}^b, W_k^b)$
    $a_k \leftarrow h(\tilde{a}_k)$
    if $k < L$ **then**
        $a_k^b \leftarrow f_\alpha^A(a_k)$
    end if

$$a_k^b = f_\alpha(h(a_k))$$

DoReFa-Net

Algorithm 1 Training a L-layer DoReFa-Net with W-bit weights and A-bit activations using G-bit gradients. Weights, activations and gradients are quantized according to Eqn. 9, Eqn. 11, Eqn. 12, respectively.

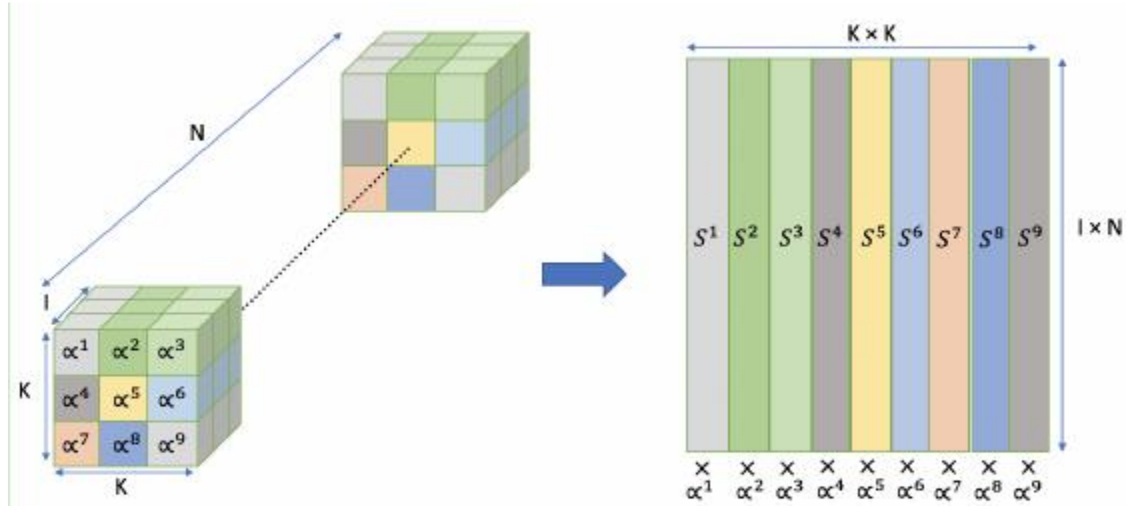
Require: a minibatch of inputs and targets (a_0, a^*), previous weights W, learning rate η
Ensure: updated weights W^{t+1}
{1. Computing the parameter gradients:}
{1.1 Forward propagation:}
for $k = 1$ **to** L **do**
    $W_k^b \leftarrow f_\omega^W(W_k)$
    $\tilde{a}_k \leftarrow \texttt{forward}(a_{k-1}^b, W_k^b)$
    $a_k \leftarrow h(\tilde{a}_k)$
    if $k < L$ **then**
        $a_k^b \leftarrow f_\alpha^A(a_k)$
    end if
    Optionally apply pooling
end for
{1.2 Backward propagation:}
Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*.
for $k = L$ **to** 1 **do**
    Back-propagate g_{a_k} through activation function h
    $g_{a_k}^b \leftarrow f_\gamma^G(g_{a_k})$
    $g_{a_{k-1}} \leftarrow \texttt{backward\_input}(g_{a_k}^b, W_k^b)$
    $g_{W_k^b} \leftarrow \texttt{backward\_weight}(g_{a_k}^b, a_{k-1}^b)$
    Back-propagate gradients through pooling layer if there is one
end for
{2. Accumulating the parameters gradients:}
for $k = 1$ **to** L **do**
    $g_{W_k} = g_{W_k^b} \frac{\partial W_k^b}{\partial W_k}$
    $W_k^{t+1} \leftarrow Update(W_k, g_{W_k}, \eta)$
end for

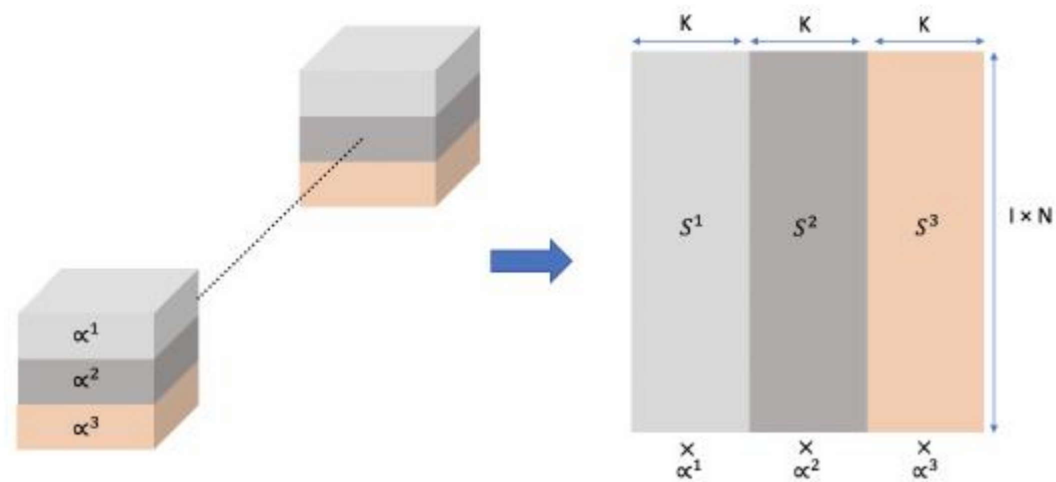
DoReFa-Net

W	A	G	Training Complexity	Inference Complexity	Storage Relative Size	AlexNet Accuracy
1	1	6	7	1	1	0.395
1	1	8	9	1	1	0.395
1	1	32	-	1	1	0.279 (BNN)
1	1	32	-	1	1	0.442 (XNOR-Net)
1	1	32	-	1	1	0.401
1	1	32	-	1	1	0.436 (initialized)
1	2	6	8	2	1	0.461
1	2	8	10	2	1	0.463
1	2	32	-	2	1	0.477
1	2	32	-	2	1	0.498 (initialized)
1	3	6	9	3	1	0.471
1	3	32	-	3	1	0.484
1	4	6	-	4	1	0.482
1	4	32	-	4	1	0.503
1	4	32	-	4	1	0.530 (initialized)
8	8	8	-	-	8	0.530
32	32	32	-	-	32	0.559

SYQ

- 不一样的缩放因子
  - Pixel-wise
  - Row-wise





[7] J. Faraone, N. Fraser, M. Blott, and P. H. Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In CVPR, 2018.

SYQ

- 权值量化

$$Q_l = sign(W_l) \odot M_l$$

$$M_{l_{i,j}} = \begin{cases} 1 & \text{if} \quad |W_{l_{i,j}}| \geq \eta_l \\ 0 & \text{if} \quad -\eta_l < W_{l_{i,j}} < \eta_l \end{cases}$$

$$\hat{w} = sign(w)$$

$$w_q = diag(\alpha)\hat{w}^T, diag(\alpha) \in \mathbb{R}^{d^2 \times d^2}$$

- 激活值量化同DoReFa-Net

$$\hat{a} = clip(a, 0, 1)$$

$$a_q = \frac{1}{2^k - 1} round((2^k - 1)\hat{a})$$

- BP阶段

$$\frac{\partial E}{\partial w^i} = \frac{\partial E}{\partial w_q^i} \frac{\partial w_q^i}{\partial w^i} = \alpha^i \frac{\partial E}{\partial w_q^i}$$

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial a_q} \frac{\partial \hat{a}}{\partial a}$$

$$\frac{\partial E}{\partial \alpha^i} = \sum_{j \in S^i} \frac{\partial E}{\partial w^j}$$

初始化

$$\alpha_{l_0}^i = \frac{\sum_{j \in S_l^i} |W_{l_{i,j}}|}{I \times N}$$

SYQ

Table 3. Comparison to previously published AlexNet results

Model	Weights	Act.	Top-1	Top-5
DoReFa-Net [33]	1	2	49.8	-
QNN [15]	1	2	51.0	73.7
HWGQ [2]	1	2	52.7	76.3
SYQ	**1**	**2**	**55.4**	**78.6**
DoReFa-Net [33]	1	4	53.0	-
SYQ	**1**	**4**	**56.2**	**79.4**
BWN [24]	1	32	56.8	79.4
SYQ	**1**	**8**	**56.6**	**79.4**
SYQ	**2**	**2**	**55.8**	**79.2**
FGQ [21]	2	8	49.04	-
TTQ [34]	2	32	57.5	79.7
SYQ	**2**	**8**	**58.1**	**80.8**

TBN

- Ternary-Binary Network

	Methods	Inputs	Weights	MACs	Binary operations	Speedup	Operations
	Full-precision	$\mathbb{R}$	$\mathbb{R}$	$n \times m \times q$	0	1×	+, x
Quantize Weights	TTQ [60]	$\mathbb{R}$	$\{-\alpha^n, 0 + \alpha^p\}$	$n \times m \times q$	0	$\sim 2\times$	+,-
	TWN [33]	$\mathbb{R}$	$\{-\alpha, 0, -\alpha\}$	$n \times m \times q$	0	$\sim 2\times$	+,-
	BWN [42]	$\mathbb{R}$	$\{-\alpha, +\alpha\}$	$n \times m \times q$	0	$\sim 2\times$	+,-
	BC [6]	$\mathbb{R}$	$\{-1, +1\}$	$n \times m \times q$	0	$\sim 2\times$	+,-
Quantize Inputs and Weights	TNN [1]	$\{-1, 0, 1\}$	$\{-1, 0, 1\}$	0	$8 \times n \times m \times q$	15×	AND, bitcount
	GXNOR [9]	$\{-1, 0, 1\}$	$\{-1, 0, 1\}$	0	$5 \times n \times m \times q$	15×	AND, bitcount
	BNN [7]	$\{-1, +1\}$	$\{-1, +1\}$	0	$2 \times n \times m \times q$	64×	XOR, bitcount
	XNOR [42]	$\{-\beta, +\beta\}$	$\{-\alpha, +\alpha\}$	$2 \times n \times m$	$2 \times n \times m \times q$	58×	XOR, bitcount
	HORQ [34]	$\{-\beta, +\beta\} \times 2$	$\{-\alpha, +\alpha\}$	$4 \times n \times m$	$4 \times n \times m \times q$	29×	XOR, bitcount
	DoReFa* [59]	$\{0, 1\} \times 2$	$\{0, 1\}$	0	$4 \times n \times m \times q$	30×	AND, bitcount
	TBN	$\{-1, 0, +1\}$	$\{-\alpha, +\alpha\}$	$n \times m$	$3 \times n \times m \times q$	40×	AND, XOR, bitcount

*We adopt DoReFa Network with 1-bit weight, 2-bit activation.

[8] Wan D, Shen F, Liu L, et al. TBN: Convolutional Neural Network with Ternary Inputs and Binary Weights, In ECCV, 2018.

TBN

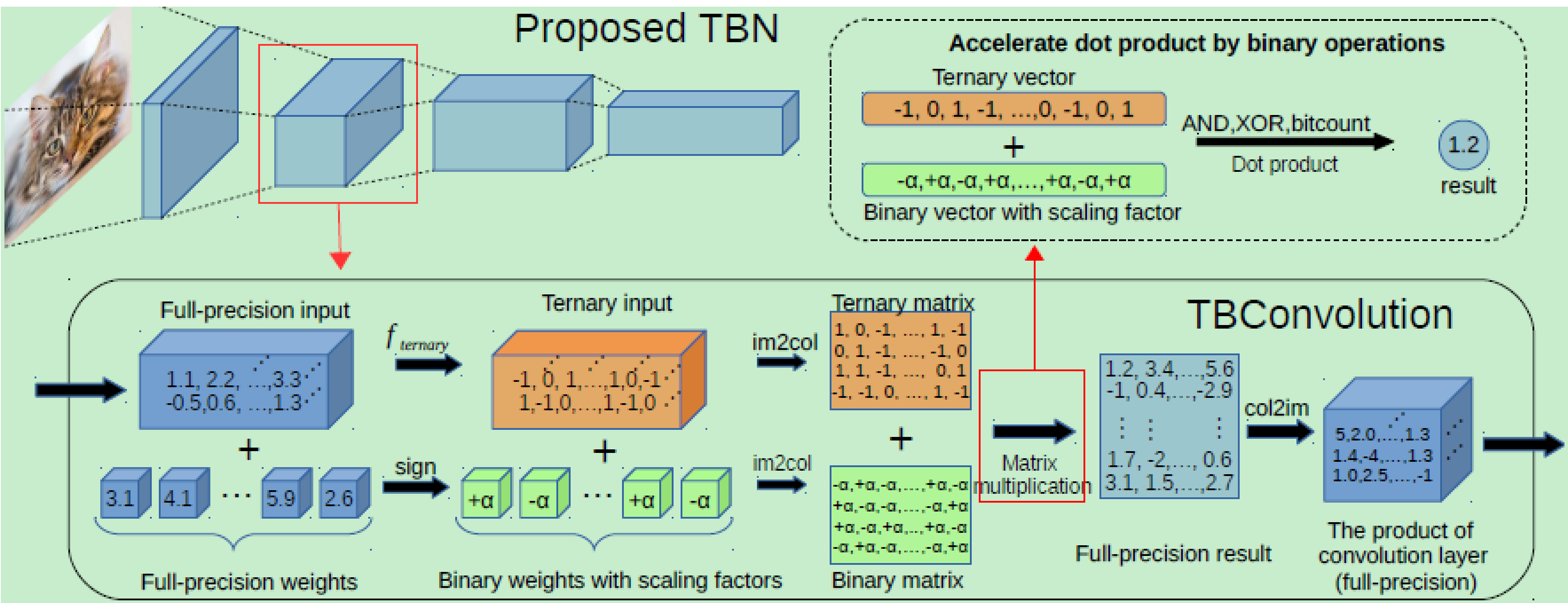
- 权值量化同XNOR-Net：
- 激活值量化像TWN：
- 求导同上：

$$\mathbf{B} = sign(\mathbf{W}), \ \alpha = \frac{1}{c \times h \times w} \|\mathbf{W}\|_1.$$

$$\mathbf{T}_i = f_{ternary}(\mathbf{I}_i, \Delta) = \begin{cases} +1, & \mathbf{I}_i > \Delta; \\ 0, & |\mathbf{I}_i| \le \Delta; \\ -1, & \mathbf{I}_i < -\Delta; \end{cases}$$

$$\frac{\partial sign}{\partial r} = \frac{\partial f_{ternary}}{\partial r} = 1_{|r|<1} = \begin{cases} 1, & |r| < 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\Delta = \delta \times \mathrm{E}(|\mathbf{I}|) \approx \frac{\delta}{c \times h_{in} \times w_{in}} \|\mathbf{I}\|_1$$



TBN

- 卷积为矩阵乘 $\mathbf{C} = mat2ten(\widetilde{\mathbf{W}}\widetilde{\mathbf{I}}), \widetilde{\mathbf{W}} = ten2mat(\mathcal{W}), \widetilde{\mathbf{I}} = ten2mat(\mathbf{I})$

  $(q = c \times h \times w)$, $\widetilde{\mathbf{W}} \in \mathbb{R}^{n \times q}$. $\widetilde{\mathbf{I}} \in \mathbb{R}^{q \times m}(m = h_{out} \times w_{out})$

  $\widetilde{\boldsymbol{W}}_i \approx \alpha \boldsymbol{b}, \boldsymbol{b} = ten2mat(\mathbf{B}) \in \{-1, 1\}^q$ $\boldsymbol{t} \in \{-1, 0, +1\}^q$

- 三值-二值向量点乘加速：

  $C_{ij} = \alpha(c_t - 2 \times \mathrm{bitcount}((\boldsymbol{b} \text{ XOR } \boldsymbol{t}') \text{ AND } \boldsymbol{t}''))$,

- 其中

  $$\boldsymbol{t}'_i = \begin{cases} 1, & \boldsymbol{t}_i = 1 \\ -1, & \text{otherwise} \end{cases}, \quad \boldsymbol{t}''_i = \begin{cases} 0, & \boldsymbol{t}_i = 0 \\ 1, & \text{otherwise} \end{cases}, \quad i = 1, \ldots, q$$

  $t_i = t'_i \times t''_i$. $c_t = \mathrm{bitcount}(\boldsymbol{t}'') = \|\boldsymbol{t}\|_1$

  - XOR、AND均为逻辑运算
  - 1在 $\boldsymbol{b}, \boldsymbol{t}', \boldsymbol{t}''$ 中视为逻辑真，0和-1则为逻辑假

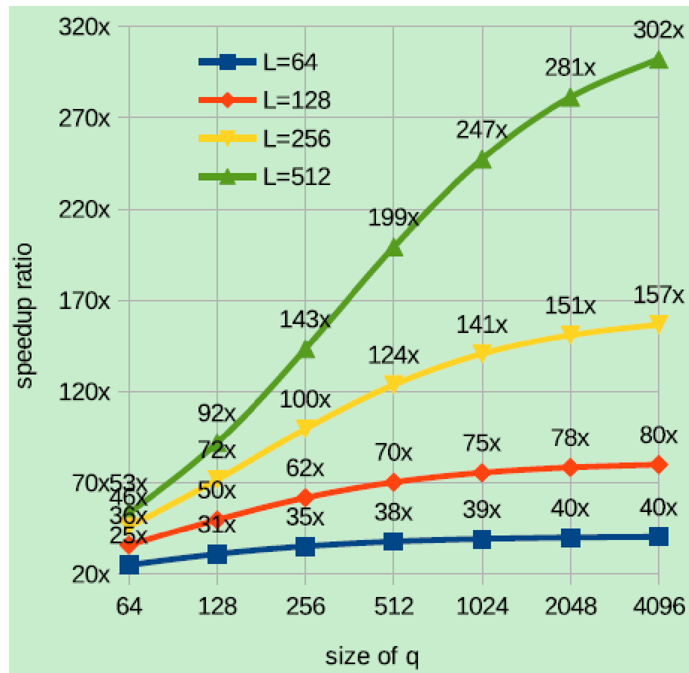
TBN

- 效率分析
  - 原浮点运算 $\tilde{\mathbf{C}} = \widetilde{\mathbf{W}}\tilde{\mathbf{I}},$ $n \times m \times q$
  - TBN运算： $n \times m$ 次浮点+ $n \times m \times q$ **AND, XOR** and **bitcount**
  - 一个时钟执行L=64位二值运算，设

$$\gamma = \frac{\text{average time required by MAC}}{\text{average time required by } L\text{-bits binary operation}}$$

  - 加速比为

$$S = \frac{\gamma nmq}{\gamma nm + 3nm\lceil \frac{q}{L} \rceil} = \frac{\gamma q}{\gamma + 3\lceil \frac{q}{L} \rceil}$$



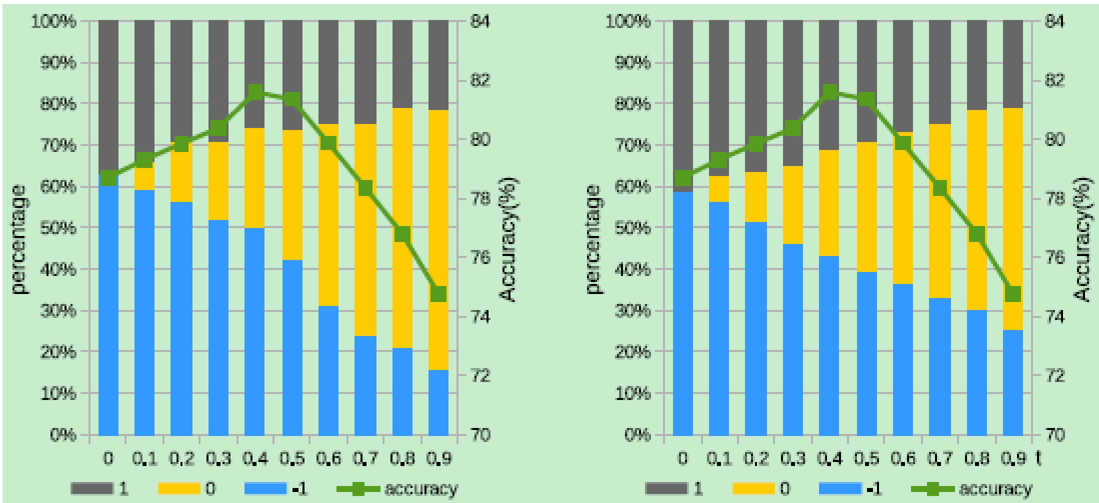
TBN

	Dataset Models	MNIST LeNet-5	CIFAR-10 VGG-7	SVHN VGG-7	ImageNet AlexNet	ImageNet ResNet-18	ImageNet ResNet-34
	Full-precision	99.48	92.88	97.68	57.2/80.2	69.3/89.2	73.3/91.4
Quantize Weights	BC [6]	98.82	91.73	97.85	35.5/61.0	-	-
	BWN [42]	99.38	92.58	97.46	56.8/79.4	60.8/83.0	-
	TWN [33]	99.38	92.56	-	54.5/76.8	65.3/86.2	-
	TTQ [60]	-	-	-	57.5/79.7	66.6/87.2	-
Other Methods	FFN [53]	-	-	-	55.5/79.0	-	-
	LCNN-fast [3]	-	-	-	44.3/68.7	51.8/76.8	-
	LCNN-accurate [3]	-	-	-	55.1/78.1	62.2/84.6	-
	LBCNN [24]	99.51	92.66	94.50	54.9/-	-	-
Quantize Inputs and Weights	TNN [1]	98.33	87.89	97.27	-	-	-
	GXNOR [9]	99.32	92.50	97.37	-	-	-
	BNN [7]	98.60	89.85	97.47	27.9/50.42	-	-
	DoReFa-Net* [59]	-	-	97.6	47.7/-	-	-
	BinaryNet [51]	-	-	-	46.6/71.1	-	-
	HORQ [34]	99.38	91.18	97.41	-	55.9/78.9	-
	XNOR-Network [42]	99.21	90.02	96.96	44.2/69.2	51.2/73.2	55.9/79.1
	TBN	99.38	90.85	97.27	49.7/74.2	55.6/79.0	58.2/81.0

*We adopt DoReFa-Net with 1-bit weight, 2-bit activation and 32-bit gradient for fair comparison.

TBN

- 稀疏度 $\delta = 0.4$



- 检测

method base network	full-precision VGG-16	full-precision ResNet-34	XNOR-Networks ResNet-34	**TBN** **ResNet-34**
Faster R-CNN	73.2	75.6	54.7	59.0
SSD 300	74.3	75.5	55.1	59.5

谢谢！