Performance Guaranteed Network Acceleration via High-Order Residual Quantization

2018-10-21

Overview

- 2017
- HORQ可以看做是XNOR的改进版，HORQ和XNOR都包含对weight和input做二值化，weight二值化方面基本一样，接下来主要介绍对input的二值化。 将CNN网络层的输入 进行高精度二值量化，从而实现高精度的二值网络计算。
- 参考博客： https://cloud.tencent.com/developer/article/1011974

HORQ Network

3.1. XNOR Network Revisited

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

Input: A minibatch of inputs and targets (I, Y), cost function $C(Y, \hat{Y})$, 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 $k = 1$ to c_{out} **do**
    $A_{lk} = \frac{1}{n}\|\mathcal{W}^t_{lk}\|_{lk}$
    $B_{lk} = sign(\mathcal{W}^t_{lk})$
    $\widetilde{\mathcal{W}}_{lk} = A_{lk}B_{lk}$
$\hat{Y}$ =BinaryForward(I, B, A)
$\frac{\partial C}{\partial \widetilde{\mathcal{W}}}$ =BinaryBackward$(\frac{\partial C}{\partial \hat{Y}}, \widetilde{\mathcal{W}})$
$\mathcal{W}^{t+1}$ =UpdateParameters$(\mathcal{W}^t, \frac{\partial C}{\partial \widetilde{\mathcal{W}}}, \eta_t)$
η^{t+1} =UpdateLearningrate(η^t, t)

$$\alpha^*, B^*, \beta^*, H^* = \underset{\alpha,B,\beta,H}{\operatorname{argmin}} \|X \odot W - \alpha\beta H \odot B\|^2 \quad (4)$$

is:

$$\begin{cases} \beta^* H^* = \frac{1}{n}\|X\|_{l_1} sign(H) \\ \alpha^* B^* = \frac{1}{n}\|W\|_{l_1} sign(W) \end{cases} \quad (5)$$

HORQ Network

3.2. High-Order Residual Quantization

$$X \approx \beta_1 H_1 \tag{6}$$

$$\begin{aligned}\beta_1^*, H_1^* &= \underset{\beta_1, H_1}{\operatorname{argmin}} J(\beta_1, H_1) \\ &= \underset{\beta_1, H_1}{\operatorname{argmin}} \|X - \beta_1 H_1\|^2\end{aligned} \tag{7}$$

The analytical solution to this problem is:

$$\begin{cases} H_1^* = sign(X) \\ \beta_1^* = \dfrac{1}{n}\|X\|_{l_1} \end{cases} \tag{8}$$

$$R_1(X) = X - \beta_1 H_1 \tag{9}$$

$$R_1(X) \approx \beta_2 H_2 \tag{10}$$

$$X = \beta_1 H_1 + R_1(X) \approx \beta_1 H_1 + \beta_2 H_2 \tag{11}$$

$$\beta_2^*, H_2^* = \underset{\beta_2, H_2}{\operatorname{argmin}} \|R_1(X) - \beta_2 H_2\|^2 \tag{12}$$

$$\begin{cases} H_2^* = sign(R_1(X)) \\ \beta_2^* = \dfrac{1}{n}\|R_1(X)\|_{l_1} \end{cases} \tag{13}$$

$$\begin{aligned} R_2(X) &= X - \beta_1 H_1 - \beta_2 H_2 \\ &= R_1(X) - \beta_2 H_2 \end{aligned} \tag{14}$$

Notice that H_2^* and β_2^* minimize $\|R_1(X) - \beta_2 H_2\|^2$, therefore:

$$\begin{aligned} &\|R_2(X)|_{\beta_2=\beta_2^*, H_2=H_2^*}\|^2 \\ =&\|(R_1(X) - \beta_2 H_2)\|^2|_{\beta_2=\beta_2^*, H_2=H_2^*} \\ =&\|(R_1(X) - \beta_2 H_2)\|^2_{min} \\ \leqslant&\|(R_1(X) - \beta_2 H_2)\|^2|_{\beta_2=0} \\ =&\|R_1(X)\|^2 \end{aligned} \tag{15}$$

这一部分定义了XNOR-Net量化过程中Residual Quantization（Order-Two）

HORQ Network

3.2. High-Order Residual Quantization

It's straightforward to develop the Order-Two Residual Quantization using Equation 11 into a Order-K Residual Quantization:

$$X \approx \sum_{i=1}^{K} \beta_i H_i \tag{16}$$

where

$$\begin{cases} R_0(X) = X & \\ R_{i-1}(X) = X - \sum_{j=1}^{i-1} \beta_j H_j & i = 2, 3, ..., K \\ H_i = sign(R_{i-1}(X)) & i = 1, 2, ..., K \\ \beta_i = \frac{1}{n} \| R_{i-1}(X) \|_{l_1} & i = 1, 2, ..., K \end{cases} \tag{17}$$

推广至Order-K

HORQ Network

3.3. The HORQ Network

Tensor Reshape

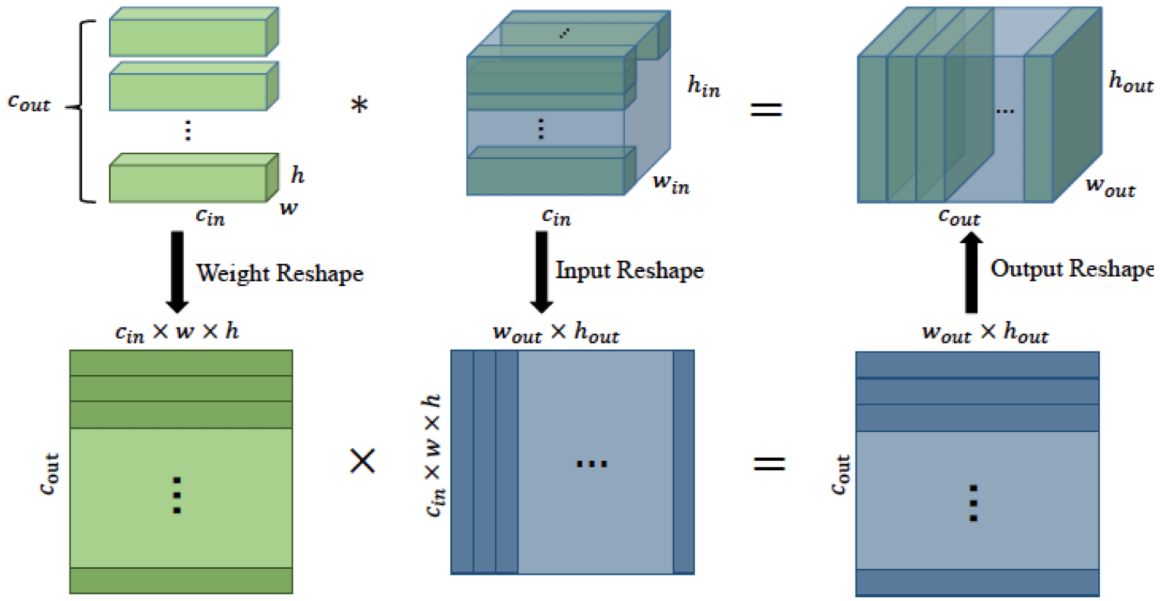

Figure 2. This figure shows the tensor reshape process.

HORQ Network

3.3. The HORQ Network

$$W_{r(i)} \approx \alpha_i B_i \quad (i = 1, 2, ..., c_{out}) \tag{18}$$

$$\begin{cases} B_i = sign(W_{r(i)}) \\ \alpha = \dfrac{1}{c_{in} \times w \times h} \|W_{r(i)}\|_{l_1} \end{cases} \tag{19}$$

where $W_{r(i)}$ is the i-th row of W_r; $W_{r(i)}, B_i \in \mathbb{R}^{1 \times (c_{in} \times w \times h)}$; $\alpha \in \mathbb{R}$.

首先对weight进行量化

Convolution Using Order-Two Residual Quantization

$$X_{r(i)} \approx \beta_{1(i)} H_{1(i)} + \beta_{2(i)} H_{2(i)} \quad (i = 1, 2, ..., w_{out} \times h_{out}) \tag{20}$$

$$\begin{cases} H_{1(i)} = sign(X_{r(i)}) \\ \beta_{1(i)} = \dfrac{1}{c_{in} \times w \times h} \|X_{r(i)}\|_{l_1} \\ R_1(X_{r(i)}) = X_{r(i)} - \beta_{1(i)} H_{1(i)} \\ H_{2(i)} = sign(R_1(X_{r(i)})) \\ \beta_{2(i)} = \dfrac{1}{c_{in} \times w \times h} \|R_1(X_{r(i)})\|_{l_1} \end{cases} \tag{21}$$

where $X_{r(i)}$ is the i-th column of X_r; $X_{r(i)}, H_{1(i)}, H_{2(i)} \in \mathbb{R}^{(c_{in} \times w \times h) \times 1}$; $\beta_{1(i)}, \beta_{2(i)} \in \mathbb{R}$. Thus we can compute the binary convolution via Algorithm 2.

之后对input量化

HORQ Network

3.3. The HORQ Network

Convolution Using Order-Two Residual Quantization

Algorithm 2 OrderTwoBinaryConvolution(X, W)

Input: Input tensor $X \in \mathbb{R}^{c_{in} \times w_{in} \times h_{in}}$, Weight tensor $W \in \mathbb{R}^{c_{out} \times c_{in} \times w \times h}$ and convolutional parameters include pad and stride.

Output: The convolutional result Y using method of second-order binary approximation.

Reshape weight tensor and input tensor:
W_r=ReshapeWeight(W)
X_r=ReshapeInput(X, W)
Binarizing weight matrix:
for $k = 1$ to c_{out} **do**
  $A_k = \frac{1}{c_n \times w \times h} \|W_{r(k)}(t)\|_{l1}$
  $M_k = sign(W_{r(k)})$
  $\widetilde{W}_{r(k)} = A_k M_k$
Binarizing input matrix:
for $k = 1$ to $w_{out} \times h_{out}$ **do**
  $B_{1k} = \frac{1}{c_n \times w \times h} \|X_{r(k)}\|_{l1}$
  $N_{1k} = sign(X_{r(k)})$
  $R_1(X_{r(k)}) = X_{r(k)} - B_{1k} N_{1k}$
  $B_{2k} = \frac{1}{c_n \times w \times h} \|R_1(X_{r(k)})\|_{l1}$
  $N_{2k} = sign(R_1(X_{r(k)}))$
  $\widetilde{X}_{r(k)} = B_{1k} N_{1k} + B_{2k} N_{2k}$
Y_r =BinaryProduction$(\widetilde{X}_{r(k)}, \widetilde{W}_{r(k)})$
Y =ReshapeOutput(Y_r)

Training HORQ Network

Algorithm 3 Traning an L-layers HORQ network:

Input: A minibatch of inputs and targets (X, Y), cost function $L(Y, \hat{Y})$, current weight $\mathcal{W}(t) = \{W^l(t)\}_{l=1,...,L}$ and current learning rate $\eta(t)$

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

for $l = 1$ to L **do**
  $\hat{Y}^l(t)$=OrderTwoBinaryConvolution$(X^l(t), W^l(t))$
  $\frac{\partial L}{\partial \widetilde{\mathcal{W}}}$ =BinaryBackward$(\frac{\partial L}{\partial \hat{Y}}, \widetilde{\mathcal{W}})$
  $\mathcal{W}(t+1)$ =UpdateParameters$(\mathcal{W}(t), \frac{\partial L}{\partial \widetilde{\mathcal{W}}}, \eta(t))$
  $\eta(t+1)$ =UpdateLearningrate$(\eta(t), t)$

Experiments

Method	Speedup ratio
Order-One Residual Quantization(XNOR)	58×
Order-Two Residual Quantization	30×
Order-Three Residual Quantization	20×
Order-Four Residual Quantization	15×

Table 2. This table shows speedup ratio using HORQ method in different orders. XNOR-Net can be considered as Order-One Residual Quantization.

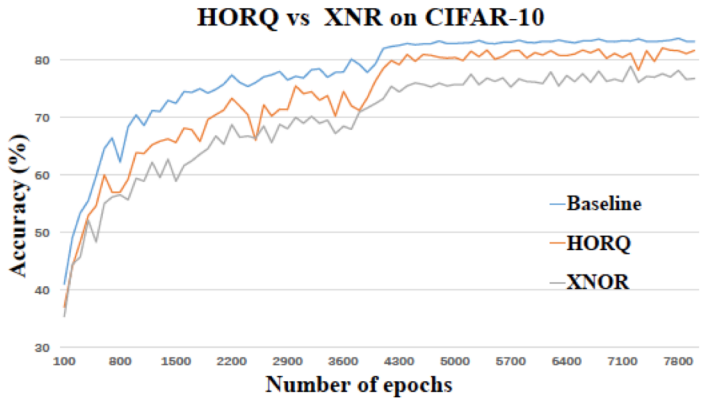

Figure 5. This figure shows the classification accuracy of HORQ-Network and XNOR-Network on CIFAR-10 on a shallow CNN.

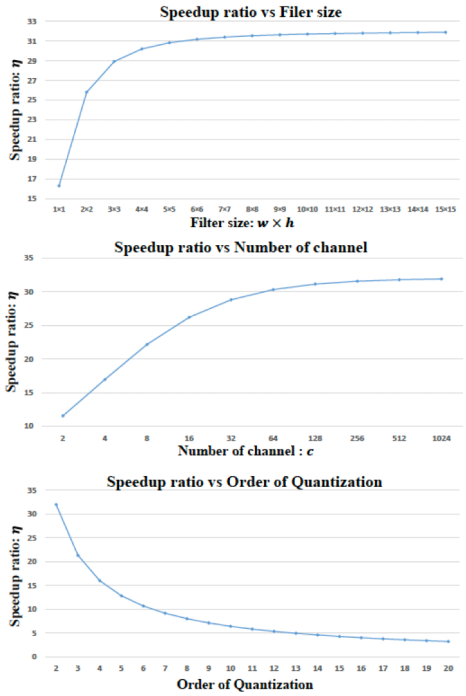

Figure 8. This figure shows the relationship between (a)speedup ratio and filter size, (b)speedup ratio and channels, (c)speedup ratio and order of quantization.

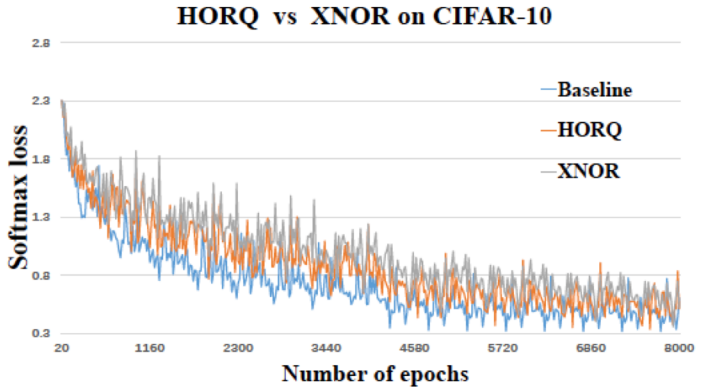

Figure 6. This figure shows the softmax loss of HORQ-Network and XNOR-Network on CIFAR-10 on a shallow CNN.

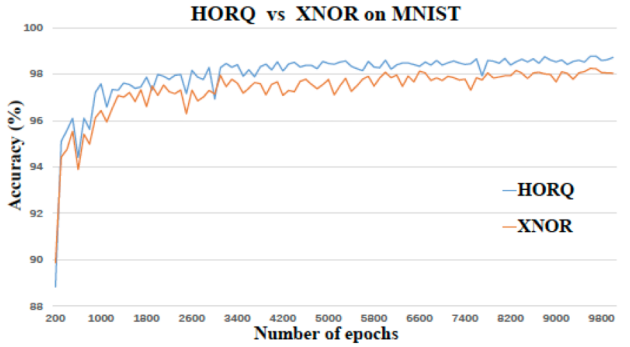

Figure 3. This figure shows the classification accuracy of HORQ-Network and XNOR-Network on MNIST.

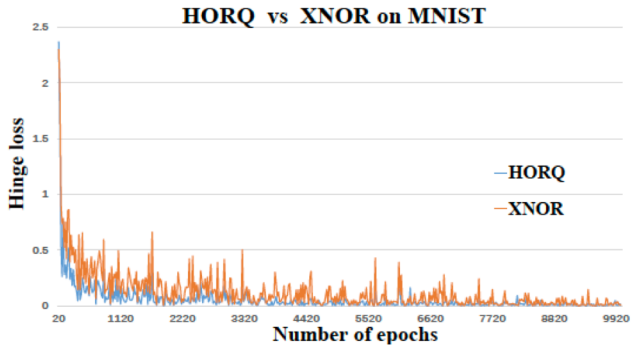

Figure 4. This figure shows the hinge loss of HORQ-Network and XNOR-Network on MNIST.

Method	Binary Input	Binary Weight	Test error
BEB	No	Yes	2.12%
BC	No	Yes	1.18%
BN	No	Yes	0.96%
BNN	Yes	Yes	1.33%
XNOR	Yes	Yes	1.96%
HORQ	Yes	Yes	1.25%

Table 1. This Table shows the Test error rate of different binary method on MNIST: BEB (Binary expectation backpropagation [2]), BC (BinaryConnect [4]), BN (BinaryNet [5]), BNN (Bitwise Neural Networks [14]), XNOR (XNOR-Networks [21]), HORQ (This work).