——国防科大2020年高性能评测与优化课程小组讨论

P8-Binarized Neural Networks

- Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1

汇报人：夏泽宇 19020121

段志敏 19020130

邵明天 19020048

指导：龚春叶、甘新标、杨博

P8-Binarized Neural Networks

Contents

需求分析

传统的深度神经网络如**DNN**，一般都需要在一个或者多个**GPU**上进行训练，这对训练设备的计算能力和存储容量都有很高的要求。这就导致在一些嵌入式或者移动场景很难运行深度神经网络，这大大限制了深度神经网络的应用。研究人员们希望能够对神经网络进行压缩，降低深度神经网络的训练成本，使神经网络能够在专用或者通用设备上运行。为了解决上述问题，二值化神经网络（**BNN**）应运而生

研究动机

BNN将权重和每层的激活值进行二值化带来了以下好处：

- 存储空间降低
- 运算速度加快
- 功耗降低
- 训练效果提高

研究动机

BNN的限制条件：

- 二值化不可避免的导致严重的信息损失
- 二值化的函数不连续

技术方案

二值化：

- 决定式：

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

- 随机二值化：

$$x^b = \begin{cases} +1 & \text{with probability } p = \sigma(x) \\ -1 & \text{with probability } 1 - p. \end{cases}$$

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

技术方案

前向传播、梯度下降、权值更新:

Algorithm 1 Training a BNN. C is the cost function for minibatch, λ the learning rate decay factor and L the number of layers. $\circ$ indicates element-wise multiplication. The function Binarize() specifies how to (stochastically or deterministically) binarize the activations and weights, and Clip() how to clip the weights. BatchNorm() specifies how to batch normalize the activations, using either batch normalization (Ioffe & Szegedy, 2015) or its shift-based variant we describe in Algorithm 2. BackBatchNorm() specifies how to backpropagate through the normalization. Update() specifies how to update the parameters knowing their gradient, using either ADAM (Kingma & Ba, 2014) or the shift-based AdaMax we describe in Algorithm 3.

Require: a minibatch of inputs and targets (a_0, a^*), previous weights W, previous BatchNorm parameters θ, weights initialization coefficients from (Glorot & Bengio, 2010) γ, and previous learning rate η.

Ensure: updated weights W^{t+1}, updated BatchNorm parameters θ^{t+1} and updated learning rate η^{t+1}.

{1. Computing the parameters' gradient:}
{1.1. Forward propagation:}
for $k = 1$ to L **do**
  $W_k^b \leftarrow \mathrm{Binarize}(W_k)$
  $s_k \leftarrow a_{k-1}^b W_k^b$
  $a_k \leftarrow \mathrm{BatchNorm}(s_k, \theta_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow \mathrm{Binarize}(a_k)$
  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
  $(g_{s_k}, g_{\theta_k}) \leftarrow \mathrm{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$
  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
  $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$
end for
{2. Accumulating the parameters' gradient:}
for $k = 1$ to L **do**
  $\theta_k^{t+1} \leftarrow \mathrm{Update}(\theta_k, \eta, g_{\theta_k})$
  $W_k^{t+1} \leftarrow \mathrm{Clip}(\mathrm{Update}(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$
  $\eta^{t+1} \leftarrow \lambda \eta$
end for

技术方案

前向传播过程：

- 先使用Sign(W_k)的到二值化后的权重W_k^b
- 用上一层的激活值α_{k-1}^b与W_k^b做乘积得到s_k(s_k也是二值化参数)
- 将s_k送入BatchNorm层，由于BatchNorm层的参数θ_k是实数型，所以经过BatchNorm层得到的α_k也是实数型的
- 使用Sign(α_k)得到α_k^b

{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

技术方案

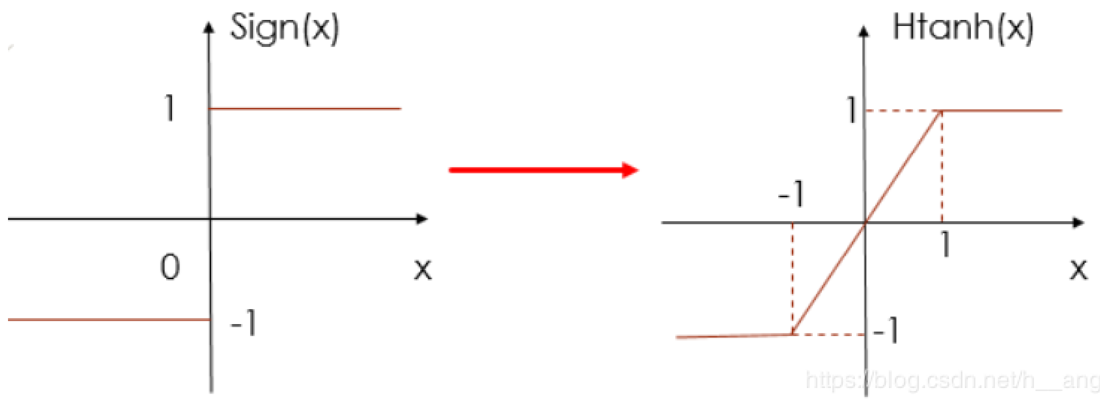
反向传播过程：

- 对所有层倒序循环，如果**k**不是第一层，则计算梯度g_{a_k}
- 基于链式法则，求解实数型激活值的梯度g_{a_k}和BatchNorm参数的梯度g_{θ_k}
- 求出二值化的权值梯度$g_{W_k^b}$和前一层的二值化激活值梯度$g_{\alpha_{k-1}^b}$

{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
  $(g_{s_k}, g_{\theta_k}) \leftarrow \mathrm{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$
  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
  $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$
end for

技术方案

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



□二值化操作函数： $q = \mathrm{Sign}(r)$

$$g_r = g_q 1_{|r| \le 1}$$

技术方案

权值更新

{2. Accumulating the parameters' gradient:}

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

技术方案

乘法优化：

Shift based Batch Normalization

Algorithm 2 Shift based Batch Normalizing Transform, applied to activation x over a mini-batch. $AP2(x) = \text{sign}(x) \times 2^{\text{round}(\log 2|x|)}$ is the approximate power-of-2 [3], and $\ll\gg$ stands for **both** left and right binary shift.

Require: Values of x over a mini-batch: $B = \{x_{1 \dots m}\}$;
Parameters to be learned: γ, β

Ensure: $\{y_i = \text{BN}(x_i, \gamma, \beta)\}$

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ {mini-batch mean}

$C(x_i) \leftarrow (x_i - \mu_B)$ {centered input}

$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(C(x_i) \ll\gg AP2(C(x_i)))$ {apx variance}

$\hat{x}_i \leftarrow C(x_i) \ll\gg AP2((\sqrt{\sigma_B^2 + \epsilon})^{-1})$ {normalize}

$y_i \leftarrow AP2(\gamma) \ll\gg \hat{x}_i$ {scale and shift}

Algorithm 3 Shift based Batch Normalizing Transform, applied to activation (x) over a mini-batch. Where AP2 is the approximate power-of-2 and $\ll\gg$ stands for **both** left and right binary shift.

Require: Values of x over a mini-batch: $B = \{x_{1 \dots m}\}$;
Parameters to be learned: γ, β

Ensure: $\{y_i = \text{BN}(x_i, \gamma, \beta)\}$

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ {mini-batch mean}

$C(x_i) \leftarrow (x_i - \mu_B)$ {centered input}

$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(C(x_i) \ll\gg AP2(C(x_i)))$ {apx variance}

$\hat{x}_i \leftarrow C(x_i) \ll\gg AP2((\sqrt{\sigma_B^2 + \epsilon})^{-1})$ {normalize}

$y_i \leftarrow AP2(\gamma) \ll\gg \hat{x}_i$ {scale and shift}

技术方案

BN层的前向传播:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

技术方案

First Layer：

在**BNN**中，某一层的输出就是下一层的输入，除了第一层之外所有层的输入都是二值化的，但是这存在两点问题：

- 第一层的通道数**x**相比于内部层来说很少**(**对于彩色图片来说，就是**3)**，因此不管是从计算量还是参数量来说，第一层都是最小的卷积层运算速度加快
- 将输入层的像素点用**m**位定点数表示，举个例子，一般用**8**位的定点数表示一个像素点，那么我们可以用下面的公式进行优化

$$s = x \cdot w^b$$

$$s = \sum_{n=1}^{8} 2^{n-1}(x^n \cdot w^b),$$

技术方案

各层的计算方法：

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**

  $a_1 \leftarrow a_1 + 2^{n-1} \times \mathrm{XnorDotProduct}(\mathrm{a}_0^{\mathrm{n}}, \mathrm{W}_1^{\mathrm{b}})$

end for

$a_1^b \leftarrow \mathrm{Sign}(\mathrm{BatchNorm}(a_1, \theta_1))$

{2. Remaining hidden layers:}

for $k = 2$ to $L - 1$ **do**

  $a_k \leftarrow \mathrm{XnorDotProduct}(a_{k-1}^b, W_k^b)$

  $a_k^b \leftarrow \mathrm{Sign}(\mathrm{BatchNorm}(a_k, \theta_k))$

end for

{3. Output layer:}

$a_L \leftarrow \mathrm{XnorDotProduct}(a_{L-1}^b, W_L^b)$

$a_L \leftarrow \mathrm{BatchNorm}(a_L, \theta_L)$

技术方案

为什么可以用xnor代替乘法：

- **+1，-1**的乘法运算真值表，和**Xnor**（同或）真值表如下：

Original multiplication			Affine transformed		
$a_{\langle -1,1\rangle}$	$b_{\langle -1,1\rangle}$	$a \cdot b_{\langle -1,1\rangle}$	$a_{\langle 0,1\rangle}$	$b_{\langle 0,1\rangle}$	$a \cdot b_{\langle 0,1\rangle}$
1	1	1	1	1	1
1	−1	−1	1	0	0
−1	1	−1	0	1	0
−1	−1	1	0	0	1

- 不难发现，假如用**0**表示**-1**，那么原来的二值乘法运算，与**Xnor**的真值表，是一致的。如果，用数学表达式，描述这种转换关系的话，可以这样写

$$\mathbf{A}_{\langle 0,1\rangle} = \frac{\mathbf{A}_{\langle -1,1\rangle} + \mathbf{A}_{\langle 1\rangle}}{2}$$

效果分析

□准确率：

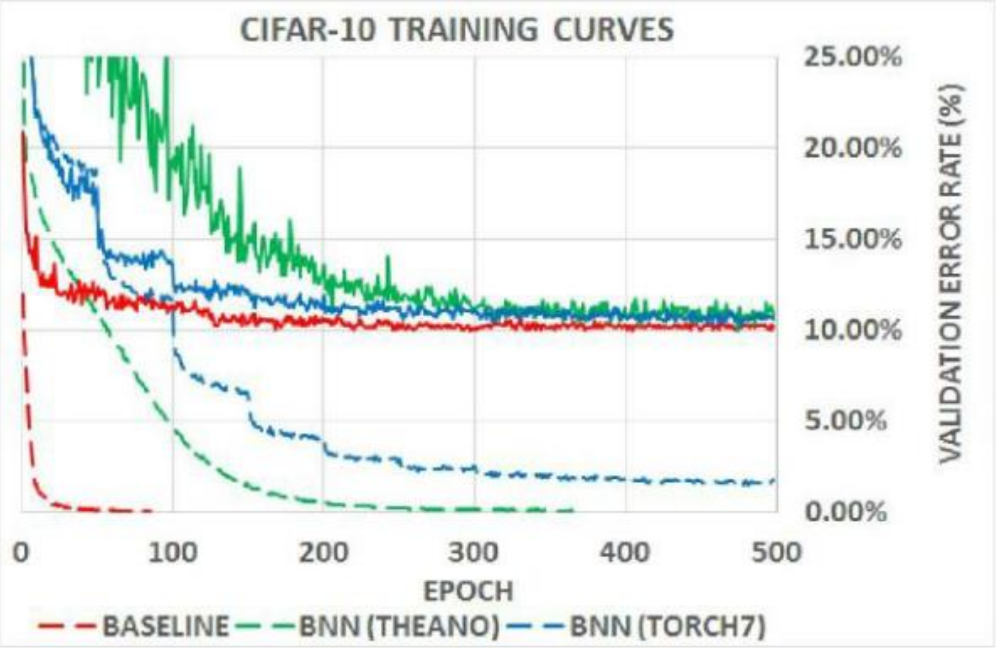
Table 1. Classification test error rates of DNNs trained on MNIST (MLP architecture without unsupervised pretraining), CIFAR-10 (without data augmentation) and SVHN.

Data set	MNIST	SVHN	CIFAR-10
Binarized activations+weights, during training and test			
BNN (Torch7)	1.40%	2.53%	10.15%
BNN (Theano)	0.96%	2.80%	11.40%
Committee Machines' Array (Baldassi et al., 2015)	1.35%	-	-
Binarized weights, during training and test			
BinaryConnect (Courbariaux et al., 2015)	1.29± 0.08%	2.30%	9.90%
Binarized activations+weights, during test			
EBP (Cheng et al., 2015)	2.2± 0.1%	-	-
Bitwise DNNs (Kim & Smaragdis, 2016)	1.33%	-	-
Ternary weights, binary activations, during test			
(Hwang & Sung, 2014)	1.45%	-	-
No binarization (standard results)			
Maxout Networks (Goodfellow et al.)	0.94%	2.47%	11.68%
Network in Network (Lin et al.)	-	2.35%	10.41%
Gated pooling (Lee et al., 2015)	-	1.69%	7.62%

效果分析

- 训练时间：

Figure 1. Training curves of a ConvNet on CIFAR-10 depending on the method. The dotted lines represent the training costs (square hinge losses) and the continuous lines the corresponding validation error rates. Although BNNs are slower to train, they are nearly as accurate as 32-bit float DNNs.



效果分析

- Filter数目：不同的卷积核的数目可以降低为原来的42%
- 内存使用：能源使用可以减少31/32
- 能耗：

Table 2. Energy consumption of multiply-accumulations (Horowitz, 2014)

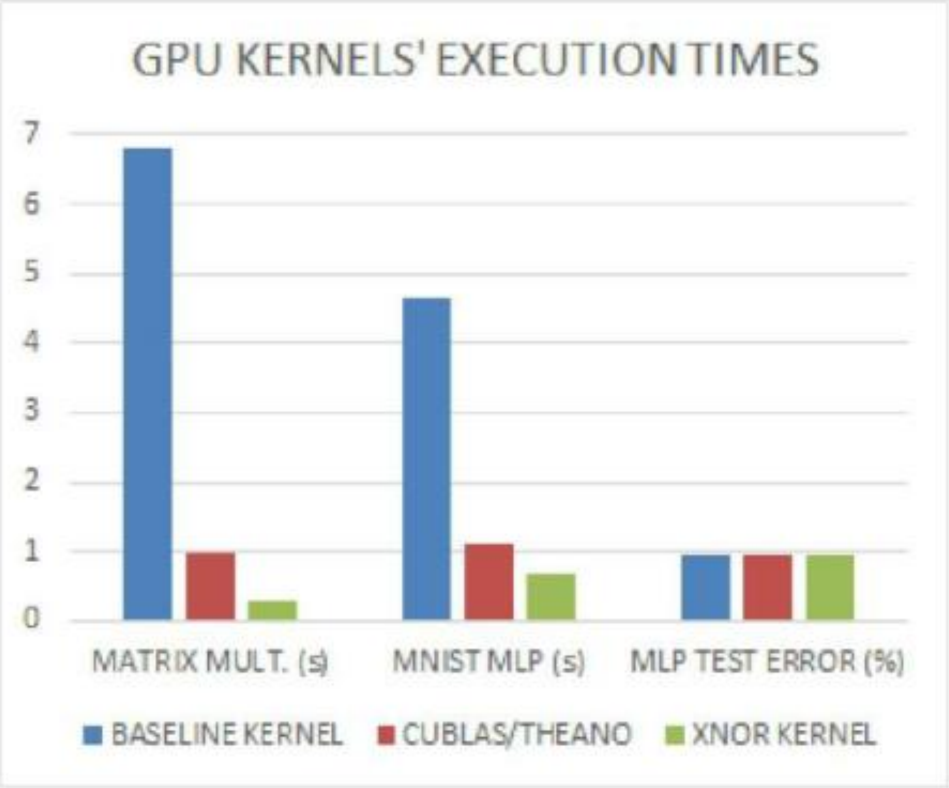
Operation	MUL	ADD
8bit Integer	0.2pJ	0.03pJ
32bit Integer	3.1pJ	0.1pJ
16bit Floating Point	1.1pJ	0.4pJ
32tbit Floating Point	3.7pJ	0.9pJ

Table 3. Energy consumption of memory accesses (Horowitz, 2014)

Memory size	64-bit memory access
8K	10pJ
32K	20pJ
1M	100pJ
DRAM	1.3-2.6nJ

效果分析

□GPU Run-Time：



未来展望

BNNs是在**BinaryConnect**的基础上，同时将权重和激活值量化到**1bit**，不仅从实验角度证明了量化算法的可行，还分析针对低**bit**如何进行更有效的计算，整理出了同时量化权重和激活值到**1bit**的算法流程，且针对内部的硬件计算，给出了具体实现，例如**Shift-based Batch Normalization**、**XNOR-Count**，最终训练能减少**60%**的时间，**32**倍的存储空间等等。

未来展望

BNN存在的问题和局限性

- 在训练过程中，从**Figure1**中可以明显看出**BNN**的收敛比**DNN**要慢，训练时间这里应该有改进的空间
- **BNN**在**MNIST**、**CIFAR10**和**SVHN**上实现了和普通**DNN**类似的精度，那么**BNN**能否在更复杂的数据集如**ImageNet**上也实现和**DNN**类似的精度并保持效果上的优势？这是一个需要继续验证的问题，并且最好也要和**RNN**作比较

P8-Binarized Neural Networks

谢 谢