DOREFA-NET: TRAINING LOW BITWIDTH CONVOLUTIONAL NEURAL NETWORKS WITH LOW BITWIDTH GRADIENTS

- Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, Yuheng Zou
- tensorflow
- 2016

Overview

- Bitwidth weights and activations/gradients

- 允许使用不同bit数的weight/activation/gradient
- 解决了前几篇论文中update parameters必须用real-valued value的问题

DOREFA-NET

- quantize **weights, activations** and **gradients** to low bitwidth numbers.

DoReFa-Net. E.g., training a network using 1-bit weights, 1-bit activations and 2-bit gradients can lead to 93% accuracy on SVHN dataset. In our experiments, gradients in general

- **Gradient采用与weight不同的bit数**
- 两个1-bit数运算：$\mathbf{x} \cdot \mathbf{y} = N - 2 \times \text{bitcount}(\text{xnor}(\mathbf{x}, \mathbf{y})), x_i, y_i \in \{-1, 1\} \forall i.$
- k-bit m-bit数运算：$\mathbf{x} = \sum_{m=0}^{M-1} c_m(\mathbf{x}) 2^m \quad \mathbf{y} = \sum_{k=0}^{K-1} c_k(\mathbf{y}) 2^k$

the computation complexity is O(MK)

where $(c_m(\mathbf{x}))_{m=0}^{M-1}$ and $(c_k(\mathbf{y}))_{k=0}^{K-1}$ are bit vectors,

$$\mathbf{x} \cdot \mathbf{y} = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} 2^{m+k} \text{ bitcount}[\text{and}(c_m(\mathbf{x}), c_k(\mathbf{y}))], \tag{3}$$

$$c_m(\mathbf{x})_i, c_k(\mathbf{y})_i \in \{0, 1\} \forall i, \ m, \ k. \tag{4}$$

STE

- Hence, in our experiments, we **use a constant scalar to scale all filters** instead of doing channel-wise scaling.

In XNOR-Net, the scaling factor $\mathbf{E}_F(|r_i|)$ is the mean of absolute value of each output channel of weights. The rationale is that introducing this scaling factor will increase the value range of weights,

1-bit时：

$$\textbf{Forward: } r_o = \text{sign}(r_i) \times \mathbf{E}(|r_i|) \tag{7}$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{8}$$

k-bit时：

In case we use k-bit representation of the weights with $k > 1$, we apply the STE f_ω^k to weights as follows:

$$\textbf{Forward: } r_o = f_\omega^k(r_i) = 2\,\text{quantize}_\text{k}\left(\frac{\tanh(r_i)}{2\max(|\tanh(r_i)|)} + \frac{1}{2}\right) - 1. \tag{9}$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial r_o}{\partial r_i}\frac{\partial c}{\partial r_o} \tag{10}$$

Note here we use $tanh$ to limit the value range of weights to $[-1, 1]$ before quantizing to k-bit. By construction, $\frac{\tanh(r_i)}{2\max(|\tanh(r_i)|)} + \frac{1}{2}$ is a number in $[0, 1]$, where the maximum is taken over all weights in that layer. $quantize_k$ will then quantize this number to k-bit fixed-point ranging in $[0, 1]$. Finally an affine transform will bring the range of $f_\omega^k(r_i)$ to $[-1, 1]$.

LOW BITWIDTH QUANTIZATION OF ACTIVATIONS

- In BNN and XNOR-Net, activations are binarized in the same way as weights. …… Hence instead, we **apply an STE on input activations r of each weight layer**. Here we assume the output of the previous layer has passed through a bounded activation function h , which ensures r in [0, 1].

quantization of activations r to k-bit is simply:

$$f_\alpha^k(r) = \text{quantize}_\text{k}(r). \tag{11}$$

- 在这里允许在activation时做k-bit量化（clip 然后 quantization）

LOW BITWIDTH QUANTIZATION OF GRADIENTS

- To quantize gradients to low bitwidth, it is important to note that **gradients are unbounded and may have significantly larger value range than activations**.
- 相比之前对于activation的操作，gradient的更新不存在nonlinear function，所以文中作了专门设计：

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

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

The noise therefore has the same magnitude as the possible quantization error. We find that the artificial noise to be critical for achieving good performance.

$$f_\gamma^k(\mathrm{d}r) = 2\max_0(|\mathrm{d}r|)\left[\mathrm{quantize}_k\left[\frac{\mathrm{d}r}{2\max_0(|\mathrm{d}r|)} + \frac{1}{2} + N(k)\right] - \frac{1}{2}\right]. \quad (12)$$

- 最终：

The quantization of gradient is done on the backward pass only. Hence we apply the following STE on the output of each convolution layer:

$$\textbf{Forward: } r_o = r_i \quad (13)$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = f_\gamma^k\left(\frac{\partial c}{\partial r_o}\right). \quad (14)$$

THE ALGORITHM FOR DO REFA-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

- FIRST AND THE LAST LAYER
  - 第一层input压缩损失信息太多；最后一层输出的是one-hot不需要压缩（但从最后一层起的backward-pragataion就开始做bianry）

- REDUCING RUN -TIME MEMORY FOOTPRINT BY FUSING NONLINEAR FUNCTION AND ROUNDING
  - 为了减少存储 full-precision带来资源消耗，将3 4 6步骤（Alogrithm中）合在一起、11 12步骤合在一起

Experiments

Table 3: Control experiments for investigation on theh degredation cost by quantizing the first convolutional layer and the last FC layer to low bitwidth. The row with "(1, 2, 4)" stands for the baseline case of $(W, A, G) = (1, 2, 4)$ and not quantizing the first and last layers. "+ first" means additionally quantizing the weights and gradients of the first convolutional layer (outputs of the first layer are already quantized in the base "(1,2,4)" scheme). "+ last" means quantizing the inputs, weights and gradients of the last FC layer. Note that outputs of the last layer do not need quantization.

Scheme	Model A Accuracy	Model B Accuracy	Model C Accuracy
(1, 2, 4)	0.975	0.969	0.939
(1, 2, 4) + first	0.972	0.963	0.932
(1, 2, 4) + last	0.973	0.969	0.927
(1, 2, 4) + first + last	0.971	0.961	0.928

Table 1: Comparison of prediction accuracy for SVHN with different choices of Bit-width in a DoReFa-Net. W, A, G are bitwidths of weights, activations and gradients respectively. When bitwidth is 32, we simply remove the quantization functions.

W	A	G	Training Complexity	Inference Complexity	Storage Relative Size	Model A Accuracy	Model B Accuracy	Model C Accuracy	Model D Accuracy
1	1	2	3	1	1	0.934	0.924	0.910	0.803
1	1	4	5	1	1	0.968	0.961	0.916	0.846
1	1	8	9	1	1	0.970	0.962	0.902	0.828
1	1	32	-	-	1	0.971	0.963	0.921	0.841
1	2	2	4	2	1	0.909	0.930	0.900	0.808
1	2	3	5	2	1	0.968	0.964	0.934	0.878
1	2	4	6	2	1	0.975	0.969	0.939	0.878
2	1	2	6	2	2	0.927	0.928	0.909	0.846
2	1	4	10	2	2	0.969	0.957	0.904	0.827
1	2	8	10	2	1	0.975	0.971	0.946	0.866
1	2	32	-	-	1	0.976	0.970	0.950	0.865
1	3	3	6	3	1	0.968	0.964	0.946	0.887
1	3	4	7	3	1	0.974	0.974	0.959	0.897
1	3	6	9	3	1	0.977	0.974	0.949	0.916
1	4	2	6	4	1	0.815	0.898	0.911	0.868
1	4	4	8	4	1	0.975	0.974	0.962	0.915
1	4	8	12	4	1	0.977	0.975	0.955	0.895
2	2	2	8	4	1	0.900	0.919	0.856	0.842
8	8	8	-	-	8			0.970	0.955
32	32	32	-	-	32	0.975	0.975	0.972	0.950

Table 2: Comparison of prediction accuracy for ImageNet with different choices of bitwidth in a DoReFa-Net. W, A, G are bitwidths of weights, activations and gradients respectively. Single-crop top-1 accuracy is given. Note the BNN result is reported by (Rastegari et al., 2016), not by original authors. We do not quantize the first and last layers of AlexNet to low bitwidth, as BNN and XNOR-Net do.

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