Application of Operations Research in Training Binary Neural Networks

ZhangXian

***Abstract*—In this paper, a binary neural network training (BNNs) method, namely in the runtime binary weight and activation functions of neural network. Inspired by training the network method solving 0-1 integer programming problems in operations research. In training, the binary weight and activation parameters used to compute the gradient. In the process of the forward pass, BNN greatly reduces the memory size and access, and will be most arithmetic for bitwise operations, it is expected to significantly improve energy efficiency.**

***Index Terms*—binary neural network, BNN, weight update, multiplication optimization**

I. INTRODUCTION

Deep neural network (DNN) has greatly promoted the development of artificial intelligence (AI) in a wide range of tasks, including but not limited to image object recognition, speech recognition, statistical machine translation, Atari and Go games, and even abstract art. Today, DNN almost only accepts training on one or more very fast and energy-intensive graphics processing units (GPUs). Therefore, running DNN on low-power devices is usually a challenge. A lot of research work has been invested in general and special computer hardware to accelerate the operation of DNN. On the basis of CNN, the binary neural network binarizes the weights and activation values (eigenvalues), that is, the value is+1 or - 1. The structure of BNN network is the same as that of CNN, and some optimizations are mainly made on gradient descent, weight update and convolution operation.

II. BINARY NEURAL NETWORK

A. *Binary*

The so-called binarization means that the weight value and the activation value can only take+1 or - 1. It is indicated here that there are two methods for binarization: ①Deterministic; ②Stochastic (statistical method).

Deterministic: greater than or equal to 0, take+1; Otherwise, take - 1.

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

Stochastic (statistical method): with a certain probability, take+1, or - 1.

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

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

Although Stochastic (statistical method) is more reasonable, it is difficult to use hardware to generate random numbers in actual operation, so the deterministic method is used here.

In the forward propagation process, the weight value and activation value of the real number type can be quantized into+1 or - 1 through the Sign function of Deterministic. When used for prediction, the parameter value is only+1 or - 1, which can reduce the memory occupation and access amount of the parameter; However, during training, it is still necessary to calculate the gradient of the real weight and activation value, and update the weight value accordingly.

B. *Weight update*

Before the weight update, you need to understand the specific process of forward propagation. In the figure below, the algorithms of forward propagation, gradient descent and weight update are given.

①Algorithm: train a BNN. C is the cost function of small batches, λ Is the learning rate decay factor, and L is the number of layers. Represents the multiplication rule at the element level. *Binarize()* specifies how binarization is activated and assigned (randomly or definitively), and *Clip()* specifies how weights are trimmed. *BatchNorm()* specifies how to use batch standardization for batch normalization activation.*BackBatchNorm()* specifies how to back propagation through normalization. *Update()* specifies how to use ADAM to update parameters when their gradient is known.

②Requirements: a minibatch input and target (a_0, a^*), previous weight W, previous BatchNorm parameter θ , Weight initialization coefficient γ, And previous learning rate η.

③ Ensure:updated weight W_{t+1}, updated batch specification parameters θ_{T+1} and updated learning rate η_{t+1}.

{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

Figure 1

The weight parameter W in the training of binary network must contain real number parameters, and then binarize the real number weight parameter to get the binary weight parameter, namely

$$W_k^b \leftarrow \text{Binarize}\left(W_k\right)$$

Then, the binary parameters are used to calculate the real intermediate vector, which is then used to obtain the real hidden layer activation vector through the Batch Normalization operation. If it is not the output layer, the vector will be binarized.

Because the gradient of the Sign (x) function is almost zero everywhere, which is obviously not conducive to reverse propagation, so the straight-through estimator strategy is adopted here, that is:

$$q = \text{Sign}(r)$$

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

Where gq is the gradient of partial C/partial q, and C is the loss function. Indicates that when | r |<=1, the gradient of Sign (r) is equal to 1; Otherwise, all are 0. It can be seen that this processing not only preserves the gradient information, but also cancels the gradient when r is too large to accelerate the convergence of the network. This is equivalent to replacing Sign (x) with HTanh (x):

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

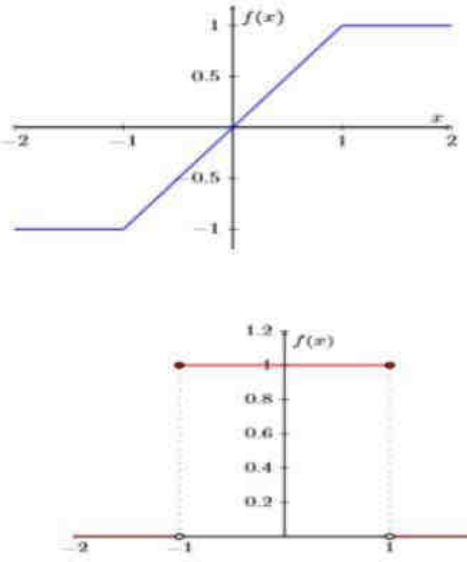

Figure 2

The figure above shows the *HTanh (x)* function, and the figure below shows the *HTanh (x)* gradient.

Then, based on the chain rule, first calculate the gradient g_{ak} of the real activation value, then the real intermediate vector gradient g_{sk} before BatchNorm, and the gradient g of the BatchNorm parameter θ_k. Then calculate the binary weight gradient $g_{w_{bk}}$, and the binary activation value gradient of the previous layer $g_{a_{k-1}^b}$ (note: when calculating the gradient of the weight value of the binary network, it is to calculate the gradient of the binary weight value, rather than the gradient of the real weight value before the binary value. This is because the weight value before the binary value does not really participate in the forward propagation process of the network)

Weight update during training: When calculating the weight (W) gradient, it is to calculate the gradient of the binary weight, but when updating the weight, it is necessary to update the real weight according to the known binary weight gradient, plus the learning rate.

$$W_k^{t+1} \leftarrow \text{Clip}\left(\text{Update}\left(W, \gamma\eta, g_{W_k^b}\right), -1, 1\right)$$

III. MULTIPLICATION OPTIMIZATION

The most important multiplication optimization in binary networks is the multiplication optimization of the output of hidden layer multiplied by the weight W in forward propagation, that is, convolution optimization. The following figure shows the forward propagation algorithm for BNN testing. It should be noted that only the input layer data is 8-bit binary, while the elements in the activation vector and weight matrix are all 1-bit binary. Therefore, in order to unify, the multiplication between the input layer and the first hidden layer is also split into "Xnor" operation mode, namely

①Algorithm: Run a BNN. L is the number of layers.
②Requirements: an 8-bit input vector a_0, binary weight Wb and batch specification parameter θ.
③Make sure that MLP outputs a_L.

{1. First layer:}
$a_1 \leftarrow 0$
for $n = 1$ to 8 **do**
  $a_1 \leftarrow a_1 + 2^{n-1} \times \mathrm{XnorDotProduct}(\mathrm{a_0^n}, \mathrm{W_1^b})$
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:}
$a_L \leftarrow \mathrm{XnorDotProduct}(a_{L-1}^b, W_L^b)$
$a_L \leftarrow \mathrm{BatchNorm}(a_L, \theta_L)$

Figure 3

IV. CONCLUSION

In general, the binary neural network BNN can:

①Reduce the memory occupation: obviously, after the weight value and activation value are binarized, they can be expressed by only 1 bit, which greatly reduces the memory occupation.

②Reduce power consumption: because of binarization, the original 32-bit floating point number only needs 1 bit to represent, and the amount of operations to access memory is reduced; In fact, the power consumption of access memory is far greater than the power consumption of calculation, so compared with 32-bit DNN, the memory occupation and access operations of BNN are reduced by 32 times, greatly reducing the power consumption.

③Reduce the area cost of hardware for deep learning: XNOR replaces multiplication, and 1-bit XNOR-gate replaces the original 32-bit floating-point multiplication, which has a great impact on hardware for deep learning. For example, FPGA originally required about 200 Slices to complete a 32-bit floating-point multiplication, while 1-bit Xnor-gate only required one Slice, which directly reduced the hardware area cost by 200 times.

④Fast speed: A GPU core of binary multiplication matrix is specially written here, which can ensure that the operation speed is increased by 7 times without loss of accuracy.

REFERENCES

[1] Peisong Wang, Xiangyu He ,Jian Cheng. Toward Accurate Binarized Neural Networks With Sparsity for Mobile Application. IEEE transactions on neural networks and learning systems.50(4): 511–515, 2005.
[2] Ruilin Li,Guoqing Zhou,Pin-Qiang Mo,Matthew R.Hall,Jun Chen,D-aqing Chen,Shangyue Cai.Behaviour of granular matter under gravity-induced stress gradient: A two-dimensional numerical investigation.International Journal of Mining Science and Technology,2021,31(03):439-450.
[3] Xu Xiangbin, Li Zhipeng. Application of reinforcement learning in operational research: research progress and prospect. Operational Research and Management, 2020,29 (5): 227-239
[4] Litvinenko A.,Kucherov D.,Glybovets M.. Decomposition Method for Calculating the Weights of a Binary Neural Network[J]. Cybernetics and Systems Analysis,2023,58(6).
[5] Che Bo,Li Xinyi,Chen Zhi,He Qi. Trainable Communication Systems Based on the Binary Neural Network[J]. Frontiers in Communications and Networks,2022.
[6] Lu Zhilin,Wang Jintao,Song Jian. Binary Neural Network Aided CSI Feedback in Massive MIMO System[J]. IEEE WIRELESS COMMUNICATIONS LETTERS,2021,10(6).

ZhangXian Male,Shijiazhuang,Hebei Province,Graduate Student,Tianjin University of Technology school of Computer Science and Technology, Tianjin Xiqing District Guest Water, 399 West Road, 300387, China.