COLLEGE OF Letters & Science
UNIVERSITY OF WISCONSIN-MADISON

电子科技大学
University of Electronic Science and Technology of China

Using Binarized Neural Network to Compress DNN

Xianda (Bryce) Xu

xxu373@wisc.edu

November 7th

Why is model compression so important ?

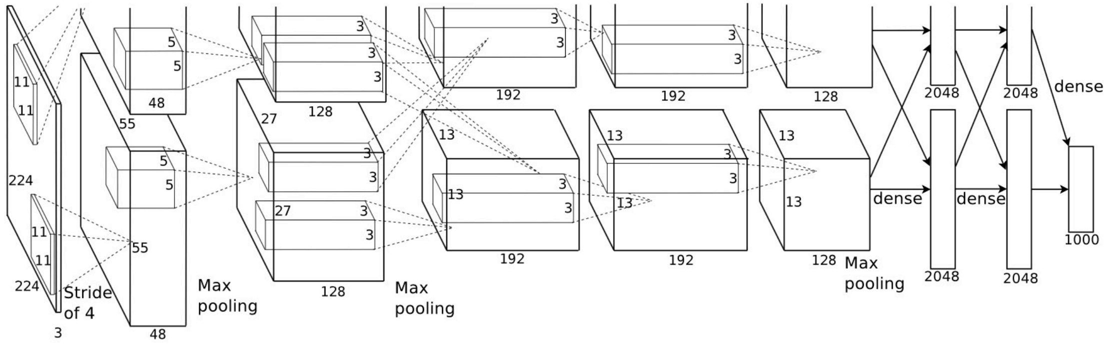


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Figure 1. AlexNet architecture

(ImageNet Large Scale Visual Recognition Challenge)

Top-1 accuracy: 57.1%

Top-5 accuracy: 80.2%

~ 60 M Parameters !

Problem 1: Computation Cost

$$A = \sigma(X \bullet W^T + B)$$

Multiplication is energy & time consuming !

Problem 2: Memory Cost

If float-32

60 M Parameters = 240 MB Memory !

However,

The energy and memory are limited on mobile devices and embedded devices !

How can we compress DNN ?

Parameters!

- **Pruning** — Removing unimportant parameters
- **Decomposition** — Apply Singular Value Decomposition on W
- **Distillation** — Use the combined output of several large networks to train a simpler model
- **Quantization** — Finding efficient representation of each parameters

What is BNN ?

In brief, it binarizes parameters and activations to +1 and -1

Why should we choose BNN ?

-- Reduce memory cost

Full-precision parameter takes 32 bits

Binary parameter takes 1 bit

Compress network by 32-X theoretically

-- Save energy and speed up

Full-precision multiplication: $\bullet$

Binary multiplication: $\odot$ (XOR)

Multiply-accumulations can be replaced by XOR and bit-count

How do we implement BNN ?

Problem 1: How to binarize ?

-- Stochastic Binarization

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

where σ is the "hard sigmoid" function:

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

-- Deterministic Binarization ☺

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

Though Stochastic Binarization seems more reasonable, we prefer deterministic binarization for its efficiency.

Problem 2: When to binarize ?

-- Forward propagation

1. First Layer:

   We do not binarize the input but binarize the Ws and As

2. Hidden Layers:

   We binarize all the Ws and As

3. Output Layer:

   We binarize Ws and only binarize the output in training

-- Back-propagation

We do not binarize gradients in back-propagation
But we have to clip weights when we update them

How do we implement BNN ?

Problem 3: How to do back-propagation ?

Recall: We calculate the gradients of the loss function L with respect to I_l, the input of the l layer.

$$g_l = \frac{\partial L}{\partial I_l}$$

The layer activation gradients:

$$g_{l-1} = g_l W_l^T$$

The layer weight gradients:

$$g_{W_l} = g_l I_{l-1}^T$$

But since we use Binarizing function, gradients are all zero !

⇒ **Straight-Through Estimator !**

Straight-Through Estimator (STE)

Yoshua Bengio etc. Estimating or propagating gradients through stochastic neurons for conditional computation (15 Aug 2013).

Adapted, for hidden layers:

For the last layer: $g_{a_L} = \dfrac{\partial C}{\partial a_L}$

For hidden layers: (map sign(x))

$g_{a_k} = g_{a_k^b} 1_{|a_k| \le 1}$ STE

$g_{s_k} = BN(g_{a_k})$ Back Batch Norm

$g_{a_{k-1}^b} = g_{s_k} W_k^b$ Activation gradients

$g_{W_k^b} = g_{s_k}^T a_{k-1}^b$ Weight gradients

STE = The gradient on Htanh(x)

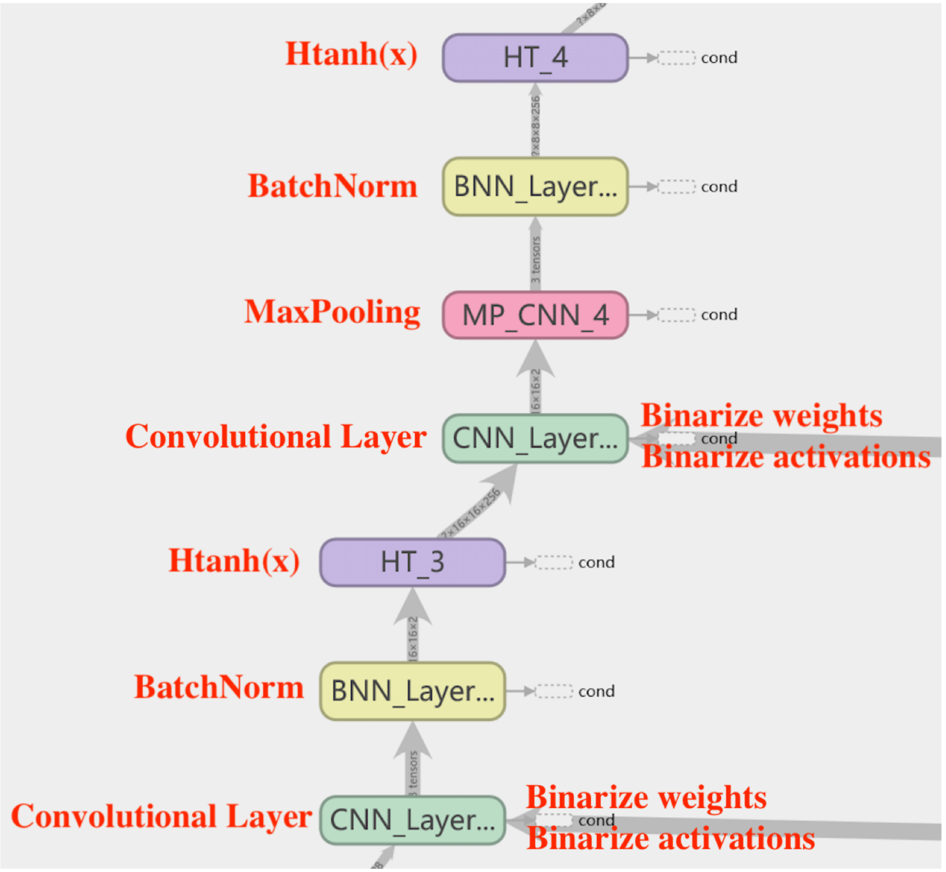
So, we use Htanh(x) as our activation function

$$H\tanh(x) = Clip(x, -1, 1)$$

How was my experiment ?

The architecture of BNN in this paper (fed by Cifar-10).

The block



In this paper:

The validation accuracy
89%

My experiment:

The training accuracy
95%

The validation accuracy
84%

https://github.com/brycexu/BinarizedNeuralNetwork/tree/master/SecondTry

```
AlexNet
Input
    Cifar-10:32x32x3
SpatialConvolution (Weight Only) 'VALID'
    32x32x3 -> 32x32x128
BatchNormalization
HardTanh
SpartialConvolution_1
    32x32x128 -> 32x32x128
MaxPooling
    32x32x128 -> 16x16x128
BatchNormalization
HardTanh
SpartialConvolution_2
    16x16x128 -> 16x16x256
BatchNormalization
HardTanh
SpartialConvolution_3
    16x16x256 -> 16x16x256
MaxPooling
    16x16x256 -> 8x8x256
BatchNormalization
HardTanh
SpartialConvolution_4
    8x8x256 -> 8x8x512
BatchNormalization
HardTanh
SpartialConvolution_5
    8x8x512 -> 8x8x512
MaxPooling
    8x8x512 -> 4x4x512
BatchNormalization
HardTanh
FC_1
    8192 -> 1024
BatchNormalization
HardTanh
FC_2
    1024 -> 1024
BatchNormalization
HardTanh
FC_3
    1024 -> 10
BatchNormalization
```

The problems about the current BNN model ?

Accuracy Loss ! Possible reasons ?

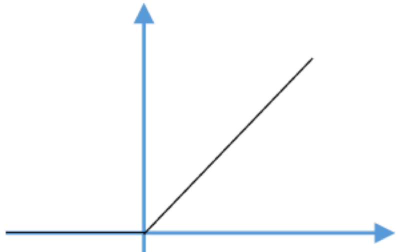
Problems 1: Robustness Issue

BNN always has larger output change which makes them more susceptible to input perturbation.

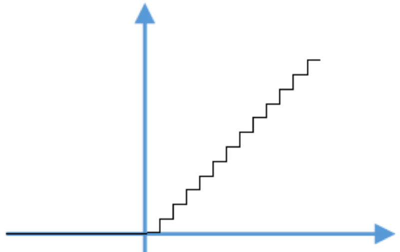
Problems 2: Stability Issue

BNN is hard to optimize due to problems such as gradient mismatch. This is because of the non-smoothness of the whole architecture.

Gradient mismatch:



(a) $g(\cdot)$



(b) $g_q(\cdot)$

The effective activation function in a fixed point network is a non-differentiable function in a discrete point network

That is why we cannot apply ReLU in BNN !

DarryI D. Lin etc. Overcoming challenges in challenges in fixed point training of deep convolutional networks. 8 Jul 2016.

The potential ways to optimize BNN model ?

Robustness Issue

1. Adding more bits ?

-- Ternary model (-1,0,+1)

-- Quantization

Research shows that having more bits at activations improve model' robustness.

3. Adding more weights ?

-- WRPN

Stability Issue ?

1. Better activation function ?

2. Better back-propagation methods ?

2. Weakening learning rate ?

Research shows that higher learning rate can cause turbulence inside the model, so BNN needs finer tuning.

4. Modifying the architecture ?

-- AdaBoost (BENN)

-- Recursively using binarization

Table 4: Comparison with state-of-the-arts on ImageNet using AlexNet (W-weights, A-activation)

Method	W	A	Top-1
Full-Precision[34, 45]	32	32	56.6%
XNOR-Net [45]	1	1	44.2%
DoReFa-Net[58]	1	1	43.6%
BinaryConnect[10, 45]	1	32	35.4%
BNN[27, 45]	1	1	27.9%
BENN-Bag-5 (ours)	1	1	54.56%
BENN-Boost-5 (ours)	1	1	57.28%
BENN-Bag-8 (ours)	1	1	55.81%
BENN-Boost-8 (ours)	1	1	**58.34%**

More bits per network ?

More networks per bit ?

Thank you !

xxu373@wisc.edu