UVA CS 6316: Machine Learning : 2019 Fall
Course Project: Deep2Reproduce @
https://github.com/qiyanjun/deep2reproduce/tree/master/2019Fall

THE HIGH-DIMENSIONAL GEOMETRY OF BINARY NEURAL NETWORKS

Reproduced by: Rasool Sharifi, Sayed Ahmad Mansouri

12/05/2019

Motivation

- **NN are highly computationally intensive.**
  - AlexNet has 61 million parameters and executes 1.5 billion operations to classify one 224 by 224 image (30 thousand operations/pixel)
- **Usability challenge in:**
  - IoT systems
  - Small mobile devices
  - Embedded systems
  - Power/resource constrained platforms
- **BNN**
  - Offers similar capabilities of full precision DNN
  - BNNs execute **7 times faster** using a dedicated GPU kernel at test time.
  - Require at least a factor of **32 fewer memory accesses** at test time that should result in an even larger energy savings.

Related Work

Methods to make DNN smaller and faster without sacrificing accuracy:

- Channel-wise separable convolutions as a way to reduce the total number of weights
- **SqueezeNet:** Use fewer parameters for convolutional layers and maintain the same number of channels and kernel size
- **Network pruning:** Identify the neurons that do not contribute much to the network and remove from the network → sparse matrices, small network with fewer calculations

Related Work - Cont.

Network Quantization Techniques

- **Fixed point calculations:**
  - Gupta et al. show that reducing data type precision offers reduced model size with limited reduction in accuracy
  - **floating point arithmetic operation is highly optimized in GPUs and most CPUs**
- **Quantized value for gradients:**
  - Using higher precision in gradients is more useful than in activations
  - Using higher precision in activations is more useful than in weights

Related Work - Cont.

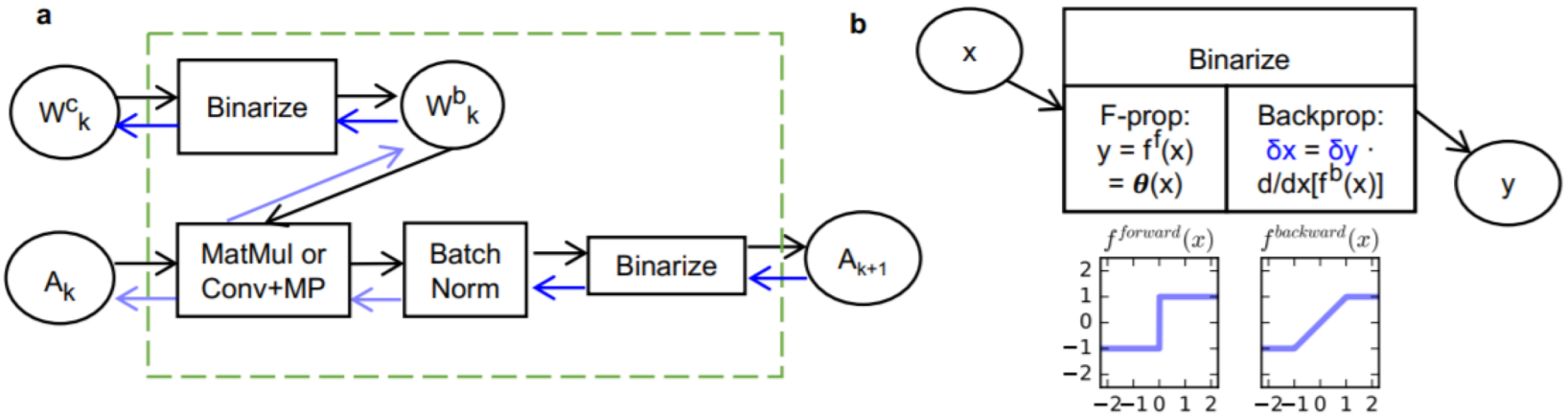
Extreme Quantization: Binarization

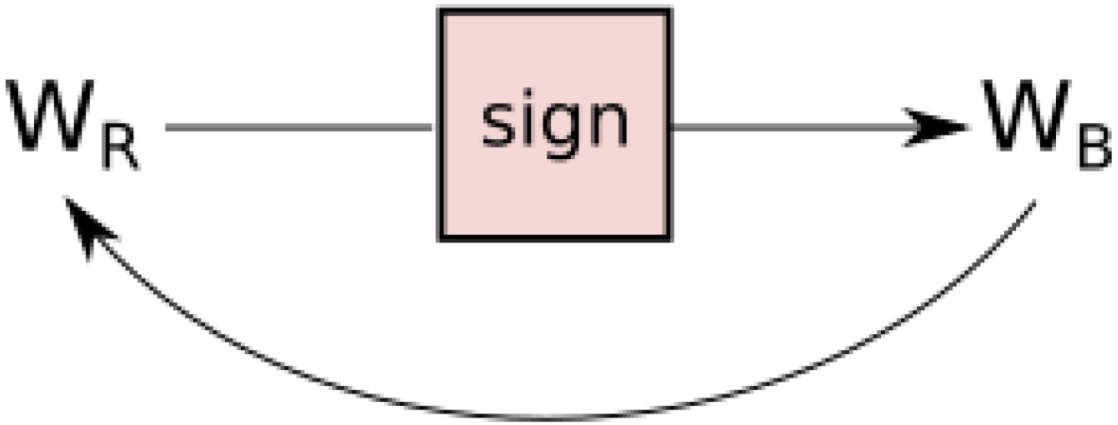
- **Ternary :** +1, 0 and -1
  - High level of compression and simple arithmatic
  - Still more complicated that BNN

- **Binary - Two possible values for values:** Usually +1 and -1

- **Early binary networks:**
  - Contained only single hidden layer
  - They point out that backpropagation and SGD cannot directly applied to these networks since weights cannot be updated in small increment

BNN: High-Level Architecture





BNN

- **Both the weights and activations are binarized**
  - Reduces the memory requirements
  - Reduces computational complexity through the use of bitwise operation

- **Train using binary weights using backpropagation with SGD**

- **How to update the weights?**
  - SGD methods make small change to the value of the weights
  - Uses set of real valued weights W_R
  - W_R are binerzied within the network to obtain binary weights
  - W_R can be updated through backprop
  - W_R is not needed during inference

BNN

- **Binarization**

  $W_B = \text{sign}(W_R)$

- **How to calculate gradient of the loss wrt. real valued weights while the sign function is not differentiable?**
  - ❖ Straight Through Estimator (STE):
    - ➢ approximates a gradient by bypassing the gradient of the layer in question
    - ➢ Cancels large gradient
    - ➢ Problematic gradient is simply turned into an identity function:

$$\frac{\partial L}{\partial W_R} = \frac{\partial L}{\partial W_B}$$

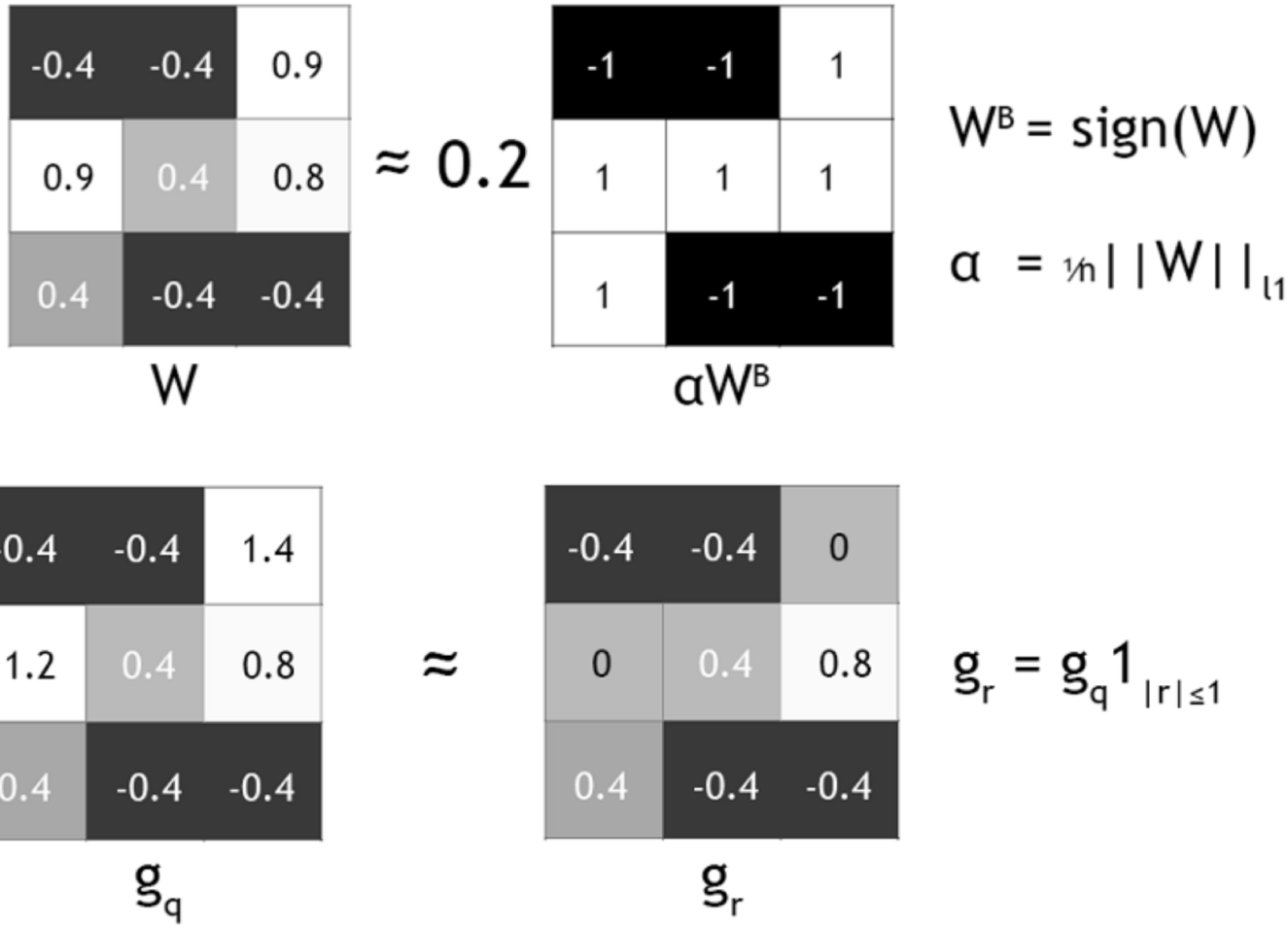
BNN

- **Bit-wise operation:**
  - Dot product between weights and activations is reduced to bitwise operation.
  - Using XNOR operation logical operation on the binary encoding is equivalent to multiplication on the binary values.

Encoding (Value)		XNOR (Multiply)
0 (−1)	0 (−1)	1 (+1)
0 (−1)	1 (+1)	0 (−1)
1 (+1)	0 (−1)	0 (−1)
1 (+1)	1 (+1)	1 (+1)

  - Multiplication is done by counting the ones in the result: **popcount** instruction
  - 23x speed up compared to real-value operation

BNN


-0.4 -0.4 0.9
0.9 0.4 0.8
0.4 -0.4 -0.4
W
≈ 0.2
-1 -1 1
1 1 1
1 -1 -1
αW^B
W^B = sign(W)
α = 1/n ||W||_l1
-0.4 -0.4 1.4
1.2 0.4 0.8
0.4 -0.4 -0.4
g_q
≈
-0.4 -0.4 0
0 0.4 0.8
0.4 -0.4 -0.4
g_r
g_r = g_q 1_|r|≤1

Claim / Target Task

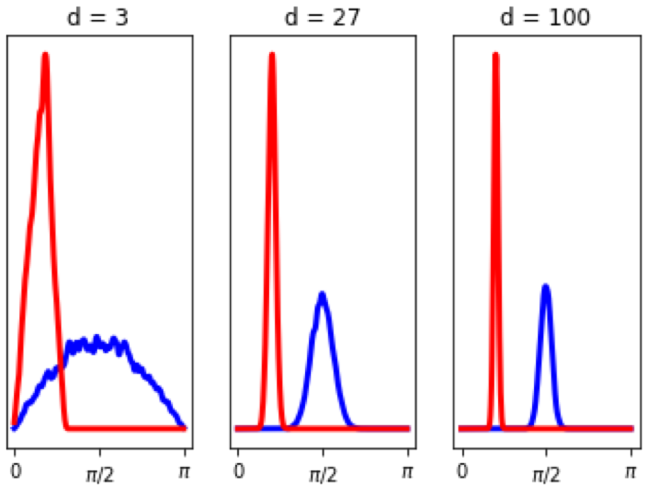
- **Angle Preservation Property:**
  - We demonstrate that binarization approximately preserves the direction of high-dimensional vectors.
- **Dot Product Proportionality Property:**
  - First, we empirically show that the weight-activation dot products, an important intermediate quantity in a neural network, are approximately proportional under the binarization of the weight vectors.
  - Second, we argue that if these weight activation dot products are proportional, then the continuous weights in the Courbariaux et al. (2016) method aren't just a learning artifact.

Implementation

- **In order to test the theory of the binarization**
  - A multilayer binary CNN is trained on CIFAR10 dataset and the weight vectors of that network are studied.
- **The CIFAR-10 convolutional neural network**
  - Six layers of convolutions, all of which have a 3 by 3 spatial kernel
  - The number of feature maps in each layer are 128, 128, 256, 256, 512, 512.
  - After the second, fourth, and sixth convolutions, there is a 2 by 2 max pooling operation.
  - Then there are two fully connected layers with 1024 units each.
  - Each layer has a batch norm layer in between.

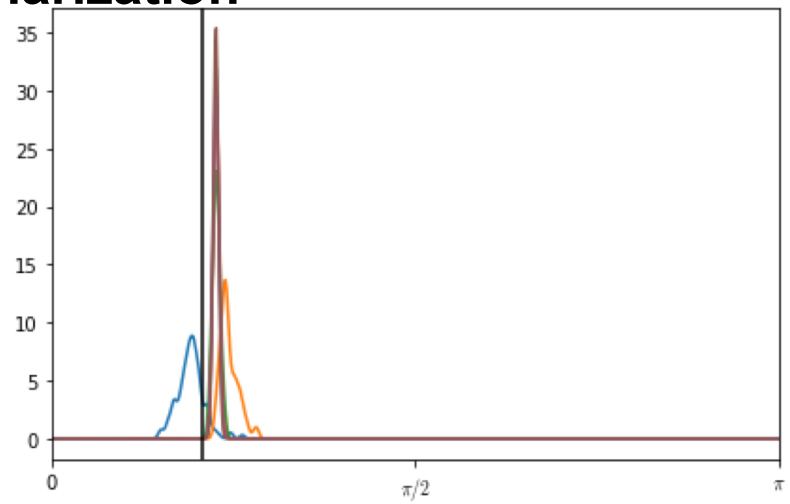
Experimental Results

- **Preservation of direction during binarization**



Binarizing a random continuous vector changes its direction by a small amount relative to the angle between two random vectors in moderately high dimensions

Figure above shows distribution of angles between two random vectors (blue curve), and between a vector and its binarized version (red curve)



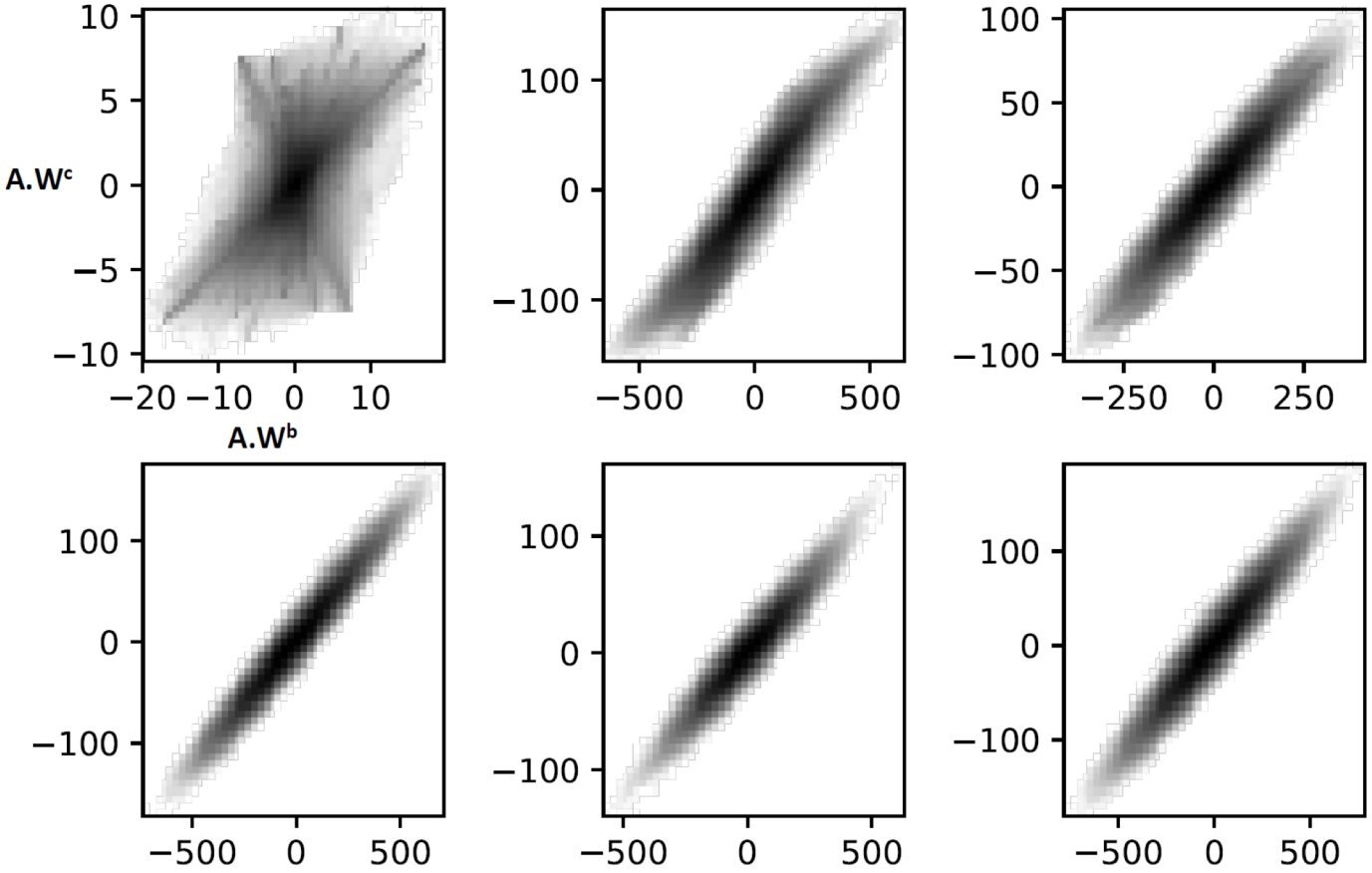
As shown in figure above remarkably, there is a close correspondence between the experimental results and the theory for the angles between the binary and continuous weights.

Experimental Results

❖ Dot Product Proportionality Property:

- **An important question to ask:**
  - Are continuous weights a learning artifact without a clear correspondence to the binary weights?
  - While we know that there are many continuous weights that map onto a particular binary weight vector. Which one do we find when we apply the method of Courbariaux et al. (2016)?
- **Answer:**
  - We get the continuous weight that preserves the dot products with the activation
  - Dot products of the activations with the pre-binarization and post-binarization weights are **highly correlated**
- $\mathbf{a} \cdot \mathbf{W_B} \sim \mathbf{a} \cdot \mathbf{W_R}$
  - In summary, the learning dynamics where we use ***g*** for the forward and backward passes (i.e. training the network with continuous weights) is approximately equivalent to the modified learning dynamics (***f*** on the forward pass, and ***g*** on the backward pass) when we have the DPP property

Experimental Results



- Binarization Preserves Dot Products: Each subplot shows a 2D histogram of the dot products between the binarized weights and the activations (horizontal axis) and the dot products between the continuous weights and the activations (vertical axis) for different layers of a network trained on CIFAR10

Conclusion

- Neural networks with binary weights and activations have similar performance to their continuous counterparts with substantially reduced execution time and power usage.
- We provided an experimentally verified theory for understanding how one can get away with such a massive reduction in precision based on the geometry of HD vectors
  - First, we showed that binarization of high-dimensional vectors preserves their direction
  - Second, we took the perspective of the network and show that binarization approximately preserves weight-activation dot products
  - Third, we discussed the impacts of the low effective dimensionality of the data on the first layer of the network
  - Finally, we show that neural networks with ternary weights and activations can also be understood with provided approach

References

- Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. 2015
- Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems, pp. 1135–1143, 2015b.
- Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding, 2015a
- Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. arXiv prepring arXiv:1607.04381, 2017
- Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In Advances in Neural Information Processing Systems, pp. 3123–3131, 2015.
- Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training neural networks with weights and activations constrained to +1 and -1. arXiv preprint arXiv:1602.02830, 2016.
- Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. arXiv preprint arXiv:1609.07061, 2016