Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit? Supplementary Material

Shilin Zhu
UC San Diego
La Jolla, CA 92093
shz338@eng.ucsd.edu

Xin Dong
Harvard University
Cambridge, MA 02138
xindong@g.harvard.edu

Hao Su
UC San Diego
La Jolla, CA 92093
haosu@eng.ucsd.edu

1. Overview

In this document, we provide detailed analysis of BNN, DNN and BENN in Sec. 2. The training algorithm of BENN is provided in Sec. 3, and Sec. 4 presents network architectures used in our main paper.

2. Detailed Analysis on DNN, BNN, and BENN

Given a full-precision real valued DNN f_w with a set of parameters $w \sim N(0, \sigma_w^2)$, a BNN f_{w_b} with binarized parameters w_b, input vector $x \sim N(0, 1)$ (after Batch Normalization) and perturbation $\Delta x \sim N(0, \sigma^2)$, and a BENN $f_{w_{\text{benn}}}$ with K ensembles, we want to compare their robustness w.r.t. the input perturbation. Here we analyze the variance of output change before and after perturbation, which echoes Eq.1 in Sec.3 in the main paper. This is because the output change has zero mean and its variance reflects the distribution of output variation. More specifically, larger variance means increased variation of output w.r.t. input perturbation.

Assume $f_w, f_{w_b}, f_{w_{\text{benn}}}$ are outputs before non-linear activation function of a single neuron in an one-layer network, we have the output variation of real-value DNN:

$$f_w(x + \Delta x) - f_w(x) = w \odot (x + \Delta x) - w \odot x = w \odot \Delta x$$

whose distribution has variance $\sigma_r^2 = |w|\sigma_w^2\sigma^2$, where $|w|$ denotes number of input connections for this neuron and $\odot$ denotes inner product. This is because summation of multiple independent distributions (due to inner product $\odot$) has variance summed as well. Some modern non-linear activation function $g(\cdot)$ like ReLU will not change the inequality of variances (i.e., if $\sigma_{f_a}^2(x) > \sigma_{f_b}^2(x)$, then $\sigma_{g(f_a(x))}^2 > \sigma_{g(f_b(x))}^2$), thus we can omit them in the analysis to keep it simple.

2.1. Activation Binarization

Suppose w is real valued but only input binarized (denote as f_w^b), the activation binarization (-1 and +1) has threshold 0, then the output variation is:

$$f_w^b(x + \Delta x) - f_w^b(x) = w \odot \text{sign}(x + \Delta x) - w \odot \text{sign}(x)$$

whose distribution has variance $\sigma_{A_b}^2 = |w|\sigma_w^2\sigma_{\text{sign}(x+\Delta x)-\text{sign}(x)}^2$. This is because $\text{sign}(x) \in \{-1, +1\}$ so the inner product is just the summation of $|w|$ independent distributions, each having variance $\sigma_{\text{sign}(x+\Delta x)-\text{sign}(x)}^2$. Note that $\gamma = \text{sign}(x + \Delta x) - \text{sign}(x)$ only has three possible values, namely, 0, -2 and +2. We compute each of them as follows:

$$\begin{aligned}
\Pr(\gamma = 2) =& \Pr(x < 0 \text{ AND } x + \Delta x > 0) \\
=& \Pr(\Delta x > -x | x < 0)\Pr(x < 0) \\
=& \int_{-\infty}^{0^-} [\int_{-x}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\Delta x)^2}{2\sigma^2}} d(\Delta x)] \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
\Pr(\gamma = -2) =& \Pr(x > 0 \text{ AND } x + \Delta x < 0) \\
=& \Pr(\Delta x < -x | x > 0)\Pr(x > 0) \\
=& \int_{0^+}^{\infty} [\int_{-\infty}^{-x} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\Delta x)^2}{2\sigma^2}} d(\Delta x)] \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
\Pr(\gamma = 0) =& 1 - \Pr(\gamma = \pm 2)
\end{aligned}$$

and its variance can be computed by:

$$\begin{aligned}
\sigma_{A_b}^2 &= |w|\sigma_w^2 \left\{\mathbb{E}[\gamma^2] - \mathbb{E}^2[\gamma]\right\} \\
&= |w|\sigma_w^2 \left\{\mathbb{E}[\gamma^2]\right\}
\end{aligned}$$

since $\mathbb{E}[(\text{sign}(x + \Delta x) - \text{sign}(x))] = 0$. Unfortunately this integral is too complicated to be solved by analytical formula, thus we use numerical method to obtain $\Pr(\text{sign}(x + \Delta x) - \text{sign}(x) = \pm 2)$. Therefore, the variance is:

$$\sigma_{A_b}^2 = B|w|\sigma_w^2, \sigma_r^2 = R|w|\sigma_w^2$$

Table 1. Relation between B, R and σ

σ	B	R
1.5	**1.25**	2.25
1.0	1.0	1.0
0.5	0.59	**0.25**
0.1	0.13	**0.01**
0.01	0.013	**0.0001**
0.001	0.0013	**0.000001**

where $B(=\sigma^2_{\text{sign}(x+\Delta x)-\text{sign}(x)})$ and $R(=\sigma^2)$ can be found in Table 1. When $\sigma < 1$, robustness of BNN is worse than DNN's. As for BENN-Bagging with K ($K > 1$) ensembles, the output change has variance:

$$\sigma^2_{\text{benn}} = K\sigma^2_{A_b} \cdot \frac{1}{K^2} = \frac{\sigma^2_{A_b}}{K} = \frac{B}{K}|w|\sigma^2_w < \sigma^2_{A_b}$$

thus BENN-Bagging has better robustness than BNN. If $K > \frac{\sigma^2_{A_b}}{\sigma^2_r} = \frac{B}{R}$, then BENN-Bagging can have even better robustness than DNN.

2.2. Weight Binarization

If we binarize w to w_b but keeping the activation real-valued, the output variation follows:

$$f_{w_b}(x + \Delta x) - f_{w_b}(x) = \text{sign}(w) \odot \Delta x$$

with variance $\sigma^2_{W_b} = |w|\sigma^2_{\text{sign}(w)}\sigma^2 = |w|\sigma^2$. Thus whether weight binarization will hurt robustness or not depends on whether $\sigma^2_{\text{sign}(w)} = 1 > \sigma^2_w$ holds or not. In particular, the robustness will not decrease if $\sigma^2_w = 1$. BENN-Bagging has variance $\sigma^2_{\text{benn}} = \frac{1}{K}|w|\sigma^2$. So if $K > \frac{1}{\sigma^2_w}$, then BENN-Bagging is better than DNN.

2.3. Binarization of Both Weight and Activation

If both activation and weight are binarized (denote as $f^b_{w_b}$), the output variation:

$$f^b_{w_b}(x+\Delta x) - f^b_{w_b}(x) = \text{sign}(w) \odot [\text{sign}(x+\Delta x) - \text{sign}(x)]$$

has variance $\sigma^2_{E_b} = |w|\sigma^2_{\text{sign}(w)}(\sigma^2_{\text{sign}(x+\Delta x)-\text{sign}(x)}) = B|w|$ which is just the combination of Sec. 2.1 and Sec. 2.2. BENN-Bagging has variance $\sigma^2_{\text{benn}} = \frac{B}{K}|w|$, which is more robust than DNN when $K > \frac{\sigma^2_{E_b}}{\sigma^2_r} = \frac{B}{R\sigma^2_w}$.

The above analysis results in the following theorem:

Theorem 1 *Given a activation binarization, weight binarization or extreme binarization one-layer network introduced above, input perturbation is $\Delta x \sim N(0, \sigma^2)$, then the output variation obeys:*

1. *If only activation is binarized, BNN has worse robustness than DNN when perturbation $\sigma < 1$. BENN-Bagging is guaranteed to be more robust than BNN. BENN-Bagging with K ensembles is more robust than DNN when $K > \frac{B}{R}$.*
2. *If only weight is binarized, BNN has worse robustness than DNN when $\sigma_w < 1$. BENN-Bagging is guaranteed to be more robust than BNN. BENN-Bagging with K ensembles is more robust than DNN when $K > \frac{1}{\sigma^2_w}$.*
3. *If both weight and activation are binarized, BNN has worse robustness than DNN when $\sigma_w < 1$ and perturbation $\sigma < 1$. BENN-Bagging is guaranteed to be more robust than BNN. BENN-Bagging with K ensembles is more robust than DNN when $K > \frac{B}{R\sigma^2_w}$.*

2.4. Multiple Layers Scenario

All the above analysis is for one layer models before and after activation function. The same conclusion can be extended to multiple layers scenario with Theorem 2.

Theorem 2 *Given a activation binarization, weight binarization or extreme binarization L-layer network (without batch normalization for generalization) introduced above, input perturbation is $\Delta x \sim N(0, \sigma^2)$, then the accumulated perturbation of ultimate network output obeys:*

1. *For DNN, ultimate output variation is $\sigma^2_r \leq \sigma^2 \prod_{l=1}^{L} |w_l|\sigma^2_{w_l}$.*
2. *For activation binarization BNN, ultimate output variation is $\sigma^2_{A_b} \leq B \prod_{l=1}^{L} |w_l|\sigma^2_{w_l}$.*
3. *For weight binarization BNN, ultimate output variation is $\sigma^2_{W_b} \leq \sigma^2 \prod_{l=1}^{L} |w_l|$*
4. *For extreme binarization BNN, ultimate output variation is $\sigma^2_{E_b} \leq B \prod_{l=1}^{L} |w_l|$.*
5. *Theorem 1 holds for multiple layers scenario.*

People have not fully understood the effect of variance reduction in boosting algorithms and some debates still exist in literature [1, 3], given that classifiers are not independent with each other. However, our experiments show that BENN-boosting can also reduce variance in our situation, which is consistent with [2, 3]. The theoretical analysis on BENN-boosting is left for future work.

If we switch x and w, replace input perturbation Δx with parameter perturbation Δw in the above analysis, then the same conclusion holds for parameter perturbation (stability issue). To sum up, BNN often can be worse than DNN in terms of robustness and stability, and our method BENN can cure these problems.

3. Training Process of BENN

Algorithm 1: Training Process of BENN

Input: a full-precision neural net with L layers, f elements in convolution kernel and learning rate η, initial weight u_i for each training example i and number of ensemble rounds K. Initialize BNN with a pre-trained XNOR-Net model [5]. Retrain each BNN for maximally M epochs.
Ensemble Pass:
for *k=1 to K* **do**
  Sampling a new training set given weight u_i of each example i;
  for *epoch=1 to M* **do**
    Forward Pass:
    for l*=1 to* L **do**
      for *each filter in* l*-th layer* **do**
        $a^l = \frac{1}{f}||w_t^l||_{l1}$;
        $b^l = \text{Sign}(w_t^l)$;
        $w_b^l = a^l b^l$
      end
      Compute activation a_l based on binary kernel w_b^l and input a_{l-1};
    end
    Backward Pass:
    Compute gradient $\frac{\partial J}{\partial w_t}$ based on [5, 4];
    Parameter Update:
    Update w_t to w_{t+1} with any update rules (e.g., SGD or ADAM)
  end
  Ensemble Update:
  Pick the BNN when training converges;
  Use either bagging or boosting algorithm to update weight u_i of each training example i;
end
Return: K trained base classifiers for BENN;

4. Network Architectures Used in the Paper

In this section we provide network architectures used in the experiments of our main paper.

4.0.1 Self-Designed Network-In-Network (NIN)

Table 2. Self-Designed Network-In-Network (NIN)

Layer Index	Type	Parameters
1	Conv	Depth: 192, Kernel Size: 5x5, Stride: 1, Padding: 2
2	BatchNorm	ϵ: 0.0001, Momentum: 0.1
3	ReLU	-
4	BatchNorm	ϵ: 0.0001, Momentum: 0.1
5	Dropout	p: 0.5
6	Conv	Depth: 96, Kernel Size: 1x1, Stride: 1, Padding: 0
7	ReLU	-
8	MaxPool	Kernel: 3x3, Stride: 2, Padding: 1
9	BatchNorm	ϵ: 0.0001, Momentum: 0.1
10	Dropout	p: 0.5
11	Conv	Depth: 192, Kernel Size: 5x5, Stride: 1, Padding: 2
12	ReLU	-
13	BatchNorm	ϵ: 0.0001, Momentum: 0.1
14	Dropout	p: 0.5
15	Conv	Depth: 192, Kernel Size: 1x1, Stride: 1, Padding: 0
16	ReLU	-
17	AvgPool	Kernel: 3x3, Stride: 2, Padding: 1
18	BatchNorm	ϵ: 0.0001, Momentum: 0.1
19	Dropout	p: 0.5
20	Conv	Depth: 192, Kernel Size: 3x3, Stride: 1, Padding: 1
21	ReLU	-
22	BatchNorm	ϵ: 0.0001, Momentum: 0.1
23	Conv	Depth: 192, Kernel Size: 1x1, Stride: 1, Padding: 0
24	ReLU	-
25	BatchNorm	ϵ: 0.0001, Momentum: 0.1
26	Conv	Depth: 192, Kernel Size: 1x1, Stride: 1, Padding: 0
27	ReLU	-
28	AvgPool	Kernel: 8x8, Stride: 1, Padding: 0
29	FC	Width: 1000

4.0.2 AlexNet

Table 3. AlexNet

Layer Index	Type	Parameters
1	Conv	Depth: 96, Kernel Size: 11x11, Stride: 4, Padding: 0
2	ReLU	-
3	MaxPool	Kernel: 3x3, Stride: 2
4	BatchNorm	-
5	Conv	Depth: 256, Kernel Size: 5x5, Stride: 1, Padding: 2
6	ReLU	-
7	MaxPool	Kernel: 3x3, Stride: 2
8	BatchNorm	-
9	Conv	Depth: 384, Kernel Size: 3x3, Stride: 1, Padding: 1
10	ReLU	-
11	Conv	Depth: 384, Kernel Size: 3x3, Stride: 1, Padding: 1
12	ReLU	-
13	Conv	Depth: 256, Kernel Size: 3x3, Stride: 1, Padding: 1
14	ReLU	-
15	MaxPool	Kernel: 3x3, Stride: 2
16	Dropout	p: 0.5
17	FC	Width: 4096
18	ReLU	-
19	Dropout	p: 0.5
20	FC	Width: 4096
21	ReLU	-
22	FC	Width: 1000

4.0.3 ResNet-18

Table 4. ResNet-18

Layer Index	Type	Parameters
1	Conv	Depth: 64, Kernel Size: 7x7, Stride: 2, Padding: 3
2	BatchNorm	ϵ: 0.00001, Momentum: 0.1
3	ReLU	-
4	MaxPool	Kernel: 3x3, Stride: 2
5	Conv	Depth: 64, Kernel Size: 3x3, Stride: 1, Padding: 1
6	BatchNorm	ϵ: 0.00001, Momentum: 0.1
7	ReLU	-
8	Conv	Depth: 64, Kernel Size: 3x3, Stride: 1, Padding: 1
9	BatchNorm	ϵ: 0.00001, Momentum: 0.1
10	Conv	Depth: 64, Kernel Size: 3x3, Stride: 1, Padding: 1
11	BatchNorm	ϵ: 0.00001, Momentum: 0.1
12	ReLU	-
13	Conv	Depth: 64, Kernel Size: 3x3, Stride: 1, Padding: 1
14	BatchNorm	ϵ: 0.00001, Momentum: 0.1
15	Conv	Depth: 128, Kernel Size: 3x3, Stride: 2, Padding: 1
16	BatchNorm	ϵ: 0.00001, Momentum: 0.1
17	ReLU	-
18	Conv	Depth: 128, Kernel Size: 3x3, Stride: 1, Padding: 1
19	BatchNorm	ϵ: 0.00001, Momentum: 0.1
20	Conv	Depth: 128, Kernel Size: 1x1, Stride: 2
21	BatchNorm	ϵ: 0.00001, Momentum: 0.1
22	Conv	Depth: 128, Kernel Size: 3x3, Stride: 1, Padding: 1
23	BatchNorm	ϵ: 0.00001, Momentum: 0.1
24	ReLU	-
25	Conv	Depth: 128, Kernel Size: 3x3, Stride: 1, Padding: 1
26	BatchNorm	ϵ: 0.00001, Momentum: 0.1
27	Conv	Depth: 256, Kernel Size: 3x3, Stride: 2, Padding: 1
28	BatchNorm	ϵ: 0.00001, Momentum: 0.1
29	ReLU	-
30	Conv	Depth: 256, Kernel Size: 3x3, Stride: 1, Padding: 1
31	BatchNorm	ϵ: 0.00001, Momentum: 0.1
32	Conv	Depth: 256, Kernel Size: 1x1, Stride: 2
33	BatchNorm	ϵ: 0.00001, Momentum: 0.1
34	Conv	Depth: 256, Kernel Size: 3x3, Stride: 1, Padding: 1
35	BatchNorm	ϵ: 0.00001, Momentum: 0.1
36	ReLU	-
37	Conv	Depth: 256, Kernel Size: 3x3, Stride: 1, Padding: 1
38	BatchNorm	ϵ: 0.00001, Momentum: 0.1
39	Conv	Depth: 512, Kernel Size: 3x3, Stride: 2, Padding: 1
40	BatchNorm	ϵ: 0.00001, Momentum: 0.1
41	ReLU	-
42	Conv	Depth: 512, Kernel Size: 3x3, Stride: 1, Padding: 1
43	BatchNorm	ϵ: 0.00001, Momentum: 0.1
44	Conv	Depth: 512, Kernel Size: 1x1, Stride: 2
45	BatchNorm	ϵ: 0.00001, Momentum: 0.1
46	Conv	Depth: 512, Kernel Size: 3x3, Stride: 1, Padding: 1
47	BatchNorm	ϵ: 0.00001, Momentum: 0.1
48	ReLU	-
49	Conv	Depth: 512, Kernel Size: 3x3, Stride: 1, Padding: 1
50	BatchNorm	ϵ: 0.00001, Momentum: 0.1
51	AvgPool	-
52	FC	Width: 1000

References

[1] P. Bühlmann and T. Hothorn. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, pages 477–505, 2007. 2

[2] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Bari, Italy, 1996. 2

[3] J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000. 2

[4] G. Hinton. Neural networks for machine learning. In *Coursera*, 2012. 3

[5] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 3