Supplementary Materials: Delving into the Estimation Shift of Batch Normalization in a Network

Lei Huang[1]* Yi Zhou[2] Tian Wang[1] Jie Luo[1] Xianglong Liu[1]
[1]SKLSDE, Institute of Artificial Intelligence, Beihang University, Beijing, China
[2]MOE Key Laboratory of Computer Network and Information Integration, Southeast University, China

1. Investigation of Estimation Shift on MNIST

In Figure 2 of the main paper, we show the results of setup one where the training set $\mathbf{S}$ equals to the test set $\mathbf{S}'$. The experiments are conducted using a learning rate of 0.1 and an update factor $\alpha = 0.9$, trained on the 20-layer multi-layer perceptron (MLP) architecture.

Here, we provide the results under different configurations, including varying the learning rate (Figure 1), varying the update factor α (Figure 2 and 3), varying the depth of the network (Figure 4), and further experiments on convolutional neural networks (Figure 5). We have the similar observations as the ones shown in Figure 2 of the main paper: 1) there are significant gaps between the training and test errors in the first dozens of epochs, and these error gaps between training and test are mainly caused by the inaccurate estimation of the population statistics of batch normalization (BN) [5]; 2) the ESM_μ and ESM_σ of BN in deeper layers have potentially higher values during the first dozens of epochs.

Group normalization with different groups. In Figure 4 of the main paper, we show the results using group normalization (GN) [12] with group number $g = 4$. Here, we provide results of GN with group numbers $g = 1$ (Figure 6) and $g = 16$ (Figure 7). Note that GN with $g = 1$ is equivalent to layer normalization (LN) [1]. We have the similar observations as the ones shown in Figure 4 of the main paper: 1) the gaps of training and test errors of 'GNBN' are significantly reduced in the first 30 epochs; 2) the ESM_σ of BNs in each layer of 'GNBN' nearly are the same during training.

2. More Results on ImageNet Classification

In this section, we provide more results on large-scale ImageNet classification [10].

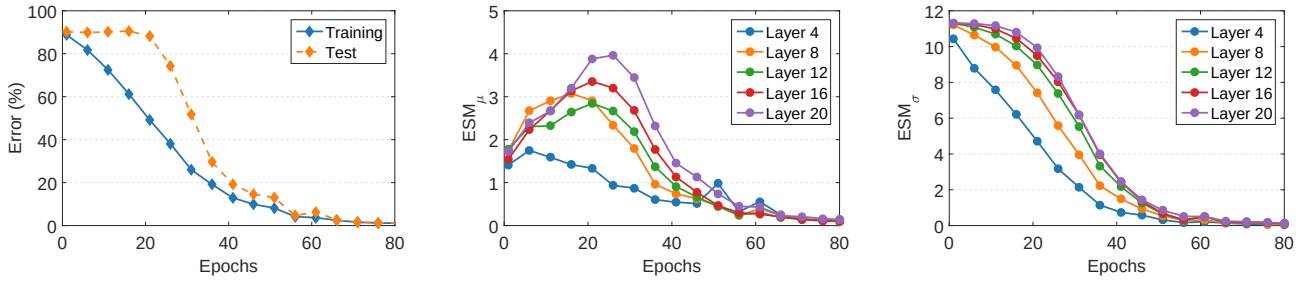
Different group number. In Section 5.2.1 of the main paper, we mention that we use GN with group number $g = 64$ as BFN in the XBNBlock. Here, we compare the results between GNs with $g = 64$ and $g = 32$, under different positions where a GN is used in an XBNBlock. Figure 8 shows the results. We observe that all the models with different group numbers outperform the baseline (BN) significantly. Besides, There are no remarkable differences in performance between the models using GN with $g = 64$ and $g = 32$.

Robustness to distribution shift. In Figure 8 of the main paper, we conduct experiments on model robustness to distribution shift. We show the results on ResNet-50 with its first six blocks being 'disturbed block'. Here, we provide more results on ResNet-50 with different blocks being 'disturbed block', and the results are shown in Figure 9. We obtain similar observations as the ones in Figure 8 of the main paper. Besides, we observe that BN has significant performance degeneration, even though only one BN's population statistics are disturbed (Figure 9 (a)), while XBNBlock$_{GN}$ and XBNBlock$_{IN}$ have no remarkable performance degeneration in this case (Figure 9 (a)).

Advanced training strategies. In Section 5.2.2 of the main paper, we conduct experiments using more advanced training strategies and show the results on ResNet-50 [2]. Here, we provide the results on ResNet-101 [2] and ResNext-50 [13] (Table 1). We also observe that XBNBlock consistently outperforms the baseline by a remarkable margin.

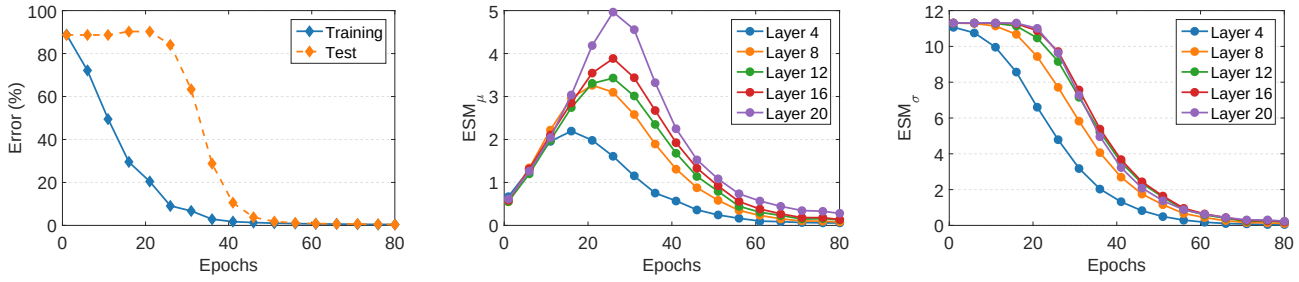
Towards whitening. We also apply the recently proposed group whitening (GW) [4] as a BFN in our XBNBlock, referred to as XBNBlock$_{GW}$. We use the released code provided in [4]. The results are shown in Table 2. By applying GW in our design, our XBNBlock outperforms the state-of-the-art whitening methods. *E.g.*, our method 'XBNBlock$_{GW}$-D4' obtains 79.18% top-1 accuracy on ResNet-101. Note that XBNBlock$_{GW}$-D4 has only 7% additional time cost.

*Corresponding author. E-mail: *huangleiAI@buaa.edu.cn*



(a) Training and test errors (b) ESM$_\mu$ of BN in different layers (c) ESM$_\sigma$ of BN in different layers

Figure 1. Experiments with training set $\mathbf{S}$ equaling to the test set $\mathbf{S}'$. We follow the same experimental setup as the one in Figure 2 of the main paper, except that we use a learning rate of 0.05.



(a) Training and test errors (b) ESM$_\mu$ of BN in different layers (c) ESM$_\sigma$ of BN in different layers

Figure 2. Experiments with training set $\mathbf{S}$ equaling to the test set $\mathbf{S}'$. We follow the same experimental setup as the one in Figure 2 of the main paper, except that we use an update factor of $\alpha = 0.1$ for the running average in calculate population statistics of BN.

2.1. Experiments on other Architectures

In Section 5.2 of the main paper, we show the results experimented on ResNet and ResNeXt architectures. Here, we provide the results on MobileNet-V2 [11] and ShuffleNet-V2 [7] architectures which are designed for more efficient computations. We conduct experiments using XBNBlock-P2 that replace the second BN of the original block of MobileNet-V2 (Figure 10) and ShuffleNet-V2 (Figure 11) with a BFN. We again use GN as BFN.

2.1.1 MobileNet-V2

Following the experimental setup shown in the main paper, we consider two training protocols:

(1) **Standard training protocol**: We apply stochastic gradient descent (SGD) using a mini-batch size of 256, momentum of 0.9 and weight decay of 0.0001. We train over 100 epochs. The initial learning rate is set to 0.1 and divided by 10 at 30, 60 and 90 epochs.

(2) **Advanced training protocol**: We train 150 epochs with cosine learning rate decay, and use a weight decay of 0.00004, under which the baseline model (MobileNet) obtains a better performance.

The results are shown in Table 3. We observe that the proposed XBNBlock consistently improves the performance of the original MobileNet-V2 architecture.

2.1.2 ShuffleNet-V2

Here, we also consider two training protocols:

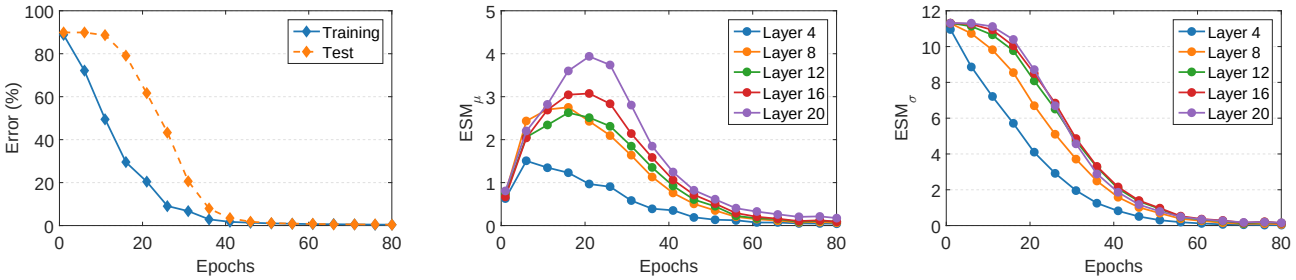
(1) **Standard training protocol**: We apply SGD using a mini-batch size of 256, momentum of 0.9 and weight decay of 0.0001. We train over 100 epochs. The initial learning rate is set to 0.1 and divided by 10 at 30, 60 and 90 epochs.

(2) **Advanced training protocol**: We train 150 epochs with cosine learning rate decay, and use a weight decay of 0.00004. We also further use the label smoothing with a smoothing factor of 0.1. The baseline (ShuffleNet-V2) has a better performance under this training protocol.

Table 3 show the results of ShuffleNet-V2 with different model sizes, including the '0.5×', '1.0×' and '1.5×' [7]. We observe that the proposed XBNBlock consistently improve the performance of the original ShuffleNet-V2 architecture, under different training protocols and model sizes.

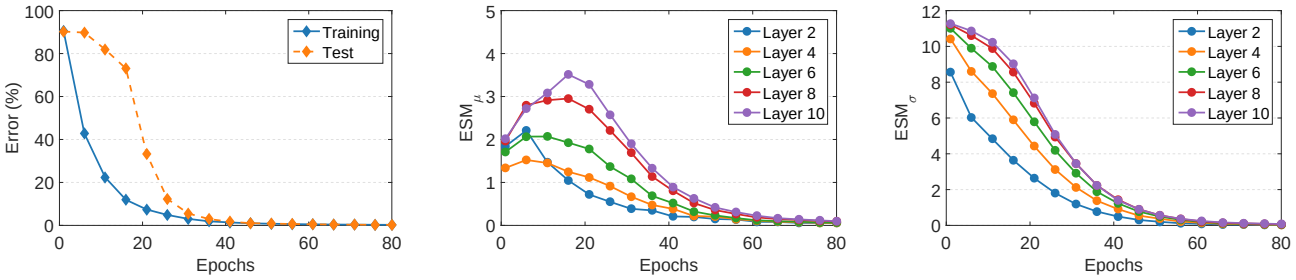
3. More Results of Detection and Segmentation on COCO

In the main paper, we only report the average precision (AP) for bounding box detection (AP^{bbox}) and instance segmentation (AP^{mask}) [6], due to space limit. Here, we report more COCO metrics including the results at different scales (AP$_{50}$, AP$_{75}$, AP$_s$, AP$_m$ and AP$_l$) for both bounding box detection and instance segmentation. Table 4 and Table 5 show



(a) Training and test errors (b) ESM$_\mu$ of BN in different layers (c) ESM$_\sigma$ of BN in different layers

Figure 3. Experiments with training set $\mathbf{S}$ equaling to the test set $\mathbf{S}'$. We follow the same experimental setup as the one in Figure 2 of the main paper, except that we use an update factor of $\alpha = 0.5$ for the running average in calculate population statistics of BN.



(a) Training and test errors (b) ESM$_\mu$ of BN in different layers (c) ESM$_\sigma$ of BN in different layers

Figure 4. Experiments with training set $\mathbf{S}$ equaling to the test set $\mathbf{S}'$. We follow the same experimental setup as the one in Figure 2 of the main paper, except that we train a 10-layer MLP.

the results using ResNet-50 and ResNeXt-101 respectively as the backbone.

Acknowledgement This work was partially supported by the National Key Research and Development Plan of China under Grant 2021ZD0112901, National Natural Science Foundation of China (Grant No. 62106012, 61972016 and 62106043).

References

[1] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[3] Lei Huang, Lei Zhao, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. An investigation into the stochasticity of batch whitening. In *CVPR*, 2020. 5

[4] Lei Huang, Yi Zhou, Li Liu, Fan Zhu, and Ling Shao. Group whitening: Balancing learning efficiency and representational capacity. In *CVPR*, 2021. 1, 5

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 5

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2

[7] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 2

[8] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: 09-26-2019. 6

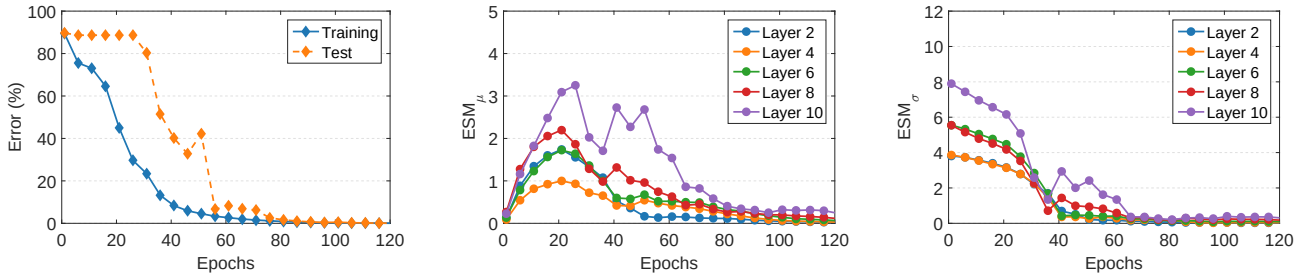
[9] Xingang Pan, Xiaohang Zhan, Jianping Shi, Xiaoou Tang, and Ping Luo. Switchable whitening for deep representation learning. In *ICCV*, 2019. 5

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[11] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 2

[12] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 1

[13] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1

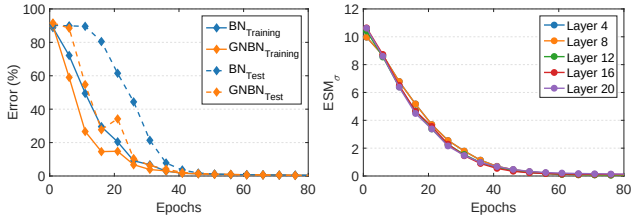


(a) Training and test errors (b) ESM$_\mu$ of BN in different layers (c) ESM$_\sigma$ of BN in different layers

Figure 5. Experiments with training set $\mathbf{S}$ equaling to the test set $\mathbf{S}'$. We train a 14-layer convolutional neural network (CNN) for MNIST classification. $\mathbf{S}$ and $\mathbf{S}'$ are the original test set of MNIST with 10,000 samples. We use full-batch gradient descent to train 120 epochs (iterations) with a learning rate of 0.1. The estimated population statistics of BN are calculated by the commonly used running average with update factor $\alpha = 0.9$.

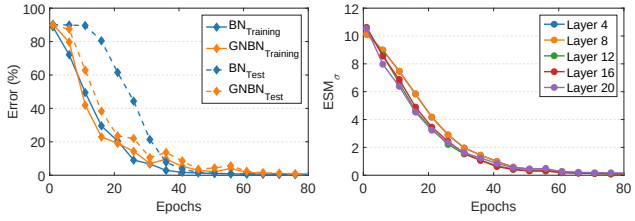
Training strategies	ResNet-101		ResNext-50	
	Baseline (BN)	XBNBlock$_{GN}$	Baseline (BN)	XBNBlock$_{GN}$
label smooth (LS)	78.25	**78.85**	77.83	**78.46**
MixUp	78.67	**79.14**	78.20	**78.73**
cosine learning (COS)	78.51	**78.91**	77.91	**78.31**
LS + MixUP + COS	79.10	**79.41**	78.84	**79.27**

Table 1. Top-1 accuracy (%) on ResNet-101 and ResNeXt-50 using advanced training strategies.



(a) Training and test errors (b) ESM$_\sigma$ of BN in different layers

Figure 6. Estimation shift experiments on a network with BN and GN mixed (referred to as 'GNBN'). We follow the same experimental setup as the one in Figure 4 of the main paper, except that we use GN with a group number of 1.



(a) Training and test errors (b) ESM$_\sigma$ of BN in different layers

Figure 7. Estimation shift experiments on a network with BN and GN mixed (referred to as 'GNBN'). We follow the same experimental setup as the one in Figure 4 of the main paper, except that we use GN with a group number of 16.

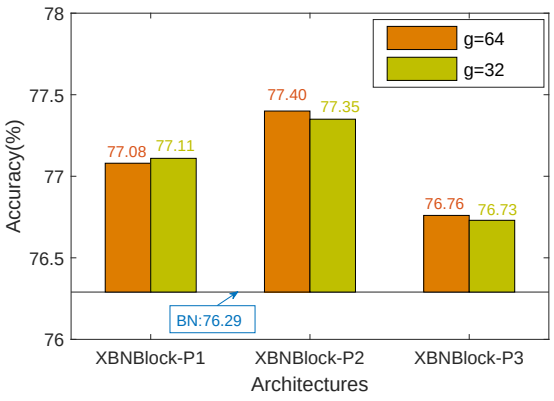


Figure 8. Comparison of the results (top-1 validation accuracy) between GNs with $g = 64$ and $g = 32$, under different positions where a GN is used in an XBNBlock. 'XBNBlock-P1', 'XBNBlock-P2' and 'XBNBlock-P3' indicate that the first, second and third BN in the bottleneck are replaced with GN, respectively.

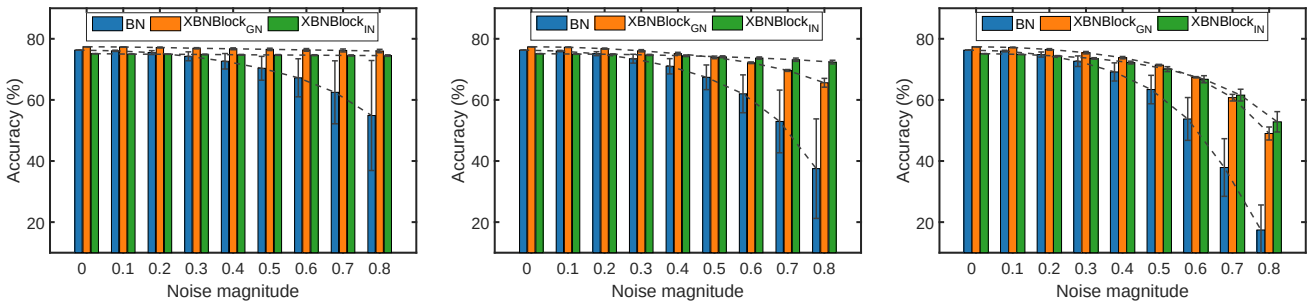
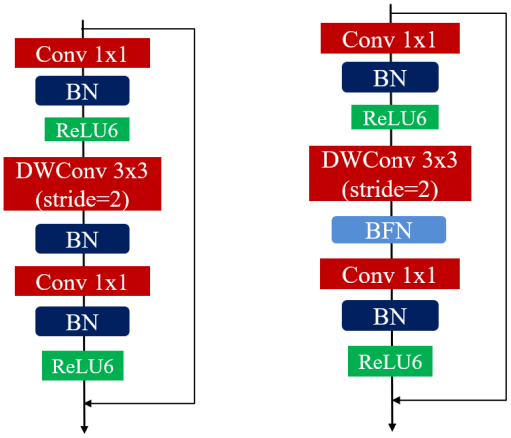


Figure 9. Top-1 validation accuracy with different noise magnitude imposed on the estimated population statistics. The results are averaged over 5 random seeds. We refer to a bottleneck/XBNBlock as 'disturbed block' if its first BN uses $\{\hat{\mu}_\delta, \hat{\sigma}^2_\delta\}$ for normalization during inference. (a) the first block of ResNet-50 is 'disturbed block'; (b) the first 6 blocks of ResNet-50 are 'disturbed block'; (c) the first 13rd blocks of ResNet-50 are 'disturbed block'.

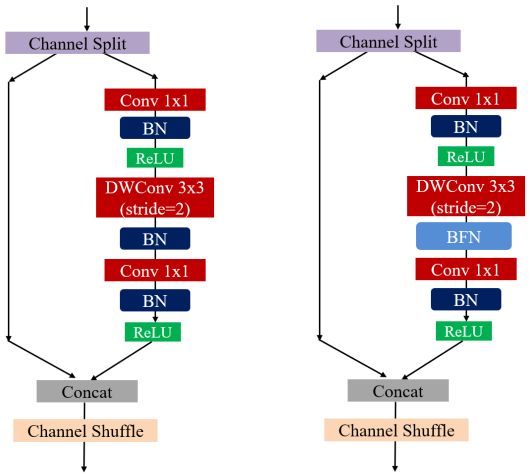
Method	ResNet-50	ResNet-101
Baseline (BN) [5]	76.29	77.65
BW [3]	77.21	78.27
SW [9]	**77.93**	79.13
GW [4]/XBNBlock$_{GW}$	77.72	78.71
XBNBlock$_{GW}$-D2 (ours)	77.89	79.12
XBNBlock$_{GW}$-D4 (ours)	77.56	**79.18**

Table 2. Comparison of top-1 validation accuracy (%) between methods with whitening module. We compare our method to batch whitening (BW) [3], switchable whitening (SW) [9] and group whitening (GW) [4]. Note that the ResNet-50 (ResNet-101) with GW used in paper [4] is equivalent to the ResNet-50 (ResNet-101) with our XBNBlock$_{GW}$ used.



(a) Inverted residual block (b) XBNBlock-P2

Figure 10. Inverted residual block of MobileNet-V2 *vs* our XBNBlock-P2 that replaces its second BN with a BFN. Note that 'DWconv' indicates the depth-wise convolutions.

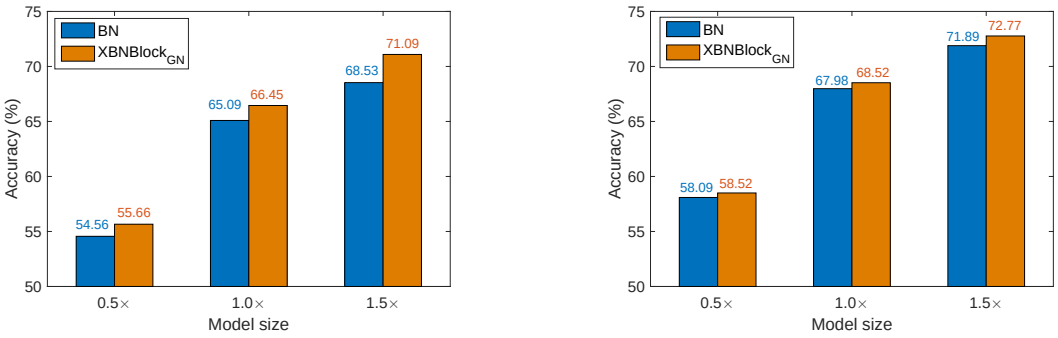


(a) Basic block of ShuffleNet-V2 (b) XBNBlock-P2

Figure 11. Basic block of ShuffleNet-V2 *vs* our XBNBlock-P2 that replaces its second BN with a BFN.

Method	Standard training	Advanced training
Baseline (BN)	66.59	70.83
XBNBlock$_{GN}$	**70.81**	**72.69**

Table 3. Top-1 accuracy (%) on MobileNet-V2 for ImageNet classification.



(a) Standard training protocol

(b) Advanced training protocol

Figure 12. Top-1 accuracy (%) on ShuffleNet-V2 for ImageNet classification.

2fc head box												
Method	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}_{s}	AP^{bbox}_{m}	AP^{bbox}_{l}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{s}	AP^{mask}_{m}	AP^{mask}_{l}
$BN^{\dagger}$	37.40	59.01	40.43	21.87	40.91	48.17	34.01	55.72	35.9	15.56	37.05	49.86
GN	37.55	59.36	40.87	21.99	40.49	48.43	34.06	55.97	35.76	15.58	36.86	49.61
XBNBlock$_{GN}$	**38.19**	**60.09**	**41.65**	**22.48**	**41.50**	**48.73**	**34.57**	**56.79**	**36.58**	**16.14**	**37.49**	**50.58**
4conv1fc head box												
Method	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}_{s}	AP^{bbox}_{m}	AP^{bbox}_{l}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{s}	AP^{mask}_{m}	AP^{mask}_{l}
$BN^{\dagger}$	37.51	58.26	40.65	21.88	40.72	48.00	33.68	54.90	35.62	15.44	36.60	48.88
GN	39.02	59.87	42.71	23.29	42.14	50.48	34.37	56.56	36.47	15.86	37.16	50.17
XBNBlock$_{GN}$	**39.57**	**60.64**	**42.93**	**24.15**	**42.32**	**51.12**	**34.86**	**57.06**	**36.95**	**17.12**	**37.41**	**50.65**

Table 4. Detection and segmentation results (%) on COCO using the Mask R-CNN framework implemented in [8]. We use ResNet-50 as the backbone, combined with FPN. All models are trained by 1x lr scheduling (90k iterations), with a batch size of 16 on eight GPUs.

2fc head box												
Method	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}_{s}	AP^{bbox}_{m}	AP^{bbox}_{l}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{s}	AP^{mask}_{m}	AP^{mask}_{l}
$BN^{\dagger}$	42.13	63.98	46.35	24.94	45.98	54.87	37.78	60.37	40.34	17.74	40.69	55.43
GN	41.47	63.54	44.76	25.36	45.33	53.20	37.17	60.23	39.31	17.92	40.24	54.12
XBNBlock$_{GN}$	**42.69**	**64.98**	**46.20**	**25.39**	**46.72**	**55.03**	**38.00**	**61.03**	**40.39**	**17.99**	**40.95**	**55.52**
4conv1fc head box												
Method	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}_{s}	AP^{bbox}_{m}	AP^{bbox}_{l}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	AP^{mask}_{s}	AP^{mask}_{m}	AP^{mask}_{l}
$BN^{\dagger}$	42.18	63.22	46.00	25.01	45.60	54.90	37.53	60.18	39.99	17.80	40.49	55.04
GN	42.24	63.00	46.19	25.27	45.76	54.94	37.53	59.82	39.96	18.00	40.42	54.52
XBNBlock$_{GN}$	**43.43**	**64.56**	**47.51**	**25.89**	**46.65**	**56.65**	**38.68**	**61.62**	**41.34**	**18.52**	**41.60**	**56.68**

Table 5. Detection and segmentation results (%) on COCO using the Mask R-CNN framework and using ResNeXt-101 as the backbone, combined with FPN. All models are trained by 1x lr scheduling (180k iterations), with a batch size of 8 on eight GPUs.