Understanding Estimation Shift in Batch Normalization and Improving Robustness Using Hybrid XBNBlock Designs

Avanti Kopulwar and Om Waikar
Department of Computer Science, Binghamton University
Email: akopulwar@binghamton.edu, owaikar1@binghamton.edu

***Abstract*—Batch Normalization (BN) is widely used in deep neural networks but suffers from a mismatch between the statistics used during training and inference, a phenomenon known as estimation shift. This mismatch accumulates across depth and can degrade performance in BN-heavy architectures. The CVPR 2022 paper "Delving Into the Estimation Shift of Batch Normalization" introduced XBNBlocks, which insert a Batch-Free Normalization (BFN) layer into the residual bottleneck to prevent this error propagation.**

In this project, we reproduced the core ideas of the original work on a scaled version of ImageNet and evaluated two architectural simplifications: (1) replacing the GroupNorm firewall with LayerNorm, and (2) restricting the firewall to only the final stage of ResNet-50. Using controlled PyTorch experiments, we found that both modifications performed competitively. LayerNorm slightly outperformed GroupNorm, and late-stage firewall placement achieved the highest overall accuracy while reducing computational overhead. These results suggest that estimation-shift mitigation is more flexible than originally proposed and that multiple XBNBlock variants can effectively stabilize BN behavior under limited computational settings.

I. Introduction

Batch Normalization (BN) has become fundamental in computer vision due to its ability to stabilize gradients and accelerate convergence. However, BN normalizes activations using mini-batch statistics during training but relies on running averages during inference. This mismatch introduces *estimation shift*. Although the discrepancy appears small in early layers, the CVPR 2022 paper shows that it compounds across depth: a small mismatch in one BN layer becomes amplified as features pass through subsequent layers.

To mitigate this, the authors propose XBNBlocks, which introduce a Batch-Free Normalization (BFN) layer-typically GroupNorm (GN) inside each residual bottleneck block, acting as a "firewall" that stops shift propagation. Our research investigates two potential simplifications of this architecture: (1) replacing GN with LayerNorm (LN), and (2) inserting the firewall only in the final stage rather than uniformly.

II. Background

A. Batch Normalization and Estimation Shift

BN computes mean and variance over each mini-batch during training. During inference, BN uses stored running statistics, which often poorly approximate the true population distribution, especially with small batches or domain shift. This mismatch accumulates across BN layers, leading to degraded accuracy.

B. Batch-Free Normalization

Batch-Free Normalization methods such as GroupNorm (GN) [3] and LayerNorm (LN) [4] normalize each sample independently and thus avoid estimation shift. XBNBlocks strategically insert such layers to stop error propagation.

C. Backbone and Dataset

To reduce cost while maintaining domain fidelity, we used ImageNette [5]. Our backbone was ResNet-50 [6] implemented in PyTorch [7].

III. Methodology

A. Estimation Shift Reproduction

Both hypothesis files load ImageNette, apply standard 224×224 preprocessing, and train modified ResNet-50 models for 10 epochs. Unlike the original CVPR 2022 paper, our implementation does not manually perturb the Batch Normalization running statistics. Instead, we study estimation shift indirectly by comparing how different normalization strategies (BN, GN, LN) and different firewall placements respond under identical training conditions.

As the original XBNBlock architecture is designed for full ImageNet training and requires extensive GPU resources, we reproduced a smaller and computationally feasible version of the architecture. We also used ImageNette (a curated 10-class subset of ImageNet) because the full ImageNet dataset is approximately 150GB and cannot be processed within our available hardware constraints. This lightweight setup allows us to faithfully capture the behavior and trends of estimation shift while remaining compatible with a single-GPU or cloud-notebook environment.

Because BN relies on mini-batch statistics while GN and LN do not, any degradation in BN performance relative to the BFN-based models reflects the impact of natural estimation-shift accumulation that occurs during training and evaluation. This allows us to analyze estimation-shift effects without explicitly injecting noise into the running mean or variance.

B. XBNBlock Variants

We implemented:

- **XBNBlock-GN** - original design (firewall = GN)
- **XBNBlock-LN** - replacing GN with LayerNorm
- **Late-Stage-XBN** - firewall only in Stage 4

All models share identical optimizers, augmentations, and training loops for fair comparison.

C. Hypothesis 1: GN vs. LN Firewall

The first hypothesis investigates whether the choice of Batch-Free Normalization (BFN) used inside the XBNBlock affects its ability to block estimation shift. In the original CVPR 2022 design, GroupNorm (GN) is used as the firewall because it normalizes channels within pre-defined groups. We question whether this specific grouping structure is necessary.

To evaluate this, `Hypothesis1.py` trains two modified ResNet-50 models: one where every XBNBlock uses GN and another where GN is replaced with LayerNorm (LN). The experiment proceeds as follows:

- **Model Construction:** The script dynamically creates three model variants-BN baseline, XBNBlock-GN, and XBNBlock-LN-by swapping the normalization layer inside the bottleneck.
- **Training Pipeline:** All models are trained for 10 epochs on ImageNette using identical data augmentations, optimizer settings, learning rate schedule, and batch size to ensure a controlled comparison.
- **Evaluation:** At each epoch, the script logs training accuracy and validation accuracy, allowing us to track learning dynamics. After training, the final top-1 accuracy is recorded.
- **Complexity and Latency:** The script measures inference-time latency over 50 forward passes and counts the total number of learnable parameters using PyTorch hooks.

This experiment isolates the effect of the firewall *type*. If LN performs similarly or better than GN, then the grouping mechanism of GN is not essential, and simpler normalization can be used inside the XBNBlock. The final results help us understand how different normalization behaviors influence the suppression of estimation shift.

D. Hypothesis 2: Uniform vs. Late-Stage Firewall

The second hypothesis evaluates whether the *placement* of the firewall affects its ability to prevent estimation shift propagation. The original XBNBlock inserts the firewall into *every* residual bottleneck block, which incurs computational overhead. We question whether this early insertion is necessary.

The `Hypothesis_2.py` script compares two architectures:

- **Uniform XBN:** Every bottleneck block in Stages 1–4 contains an XBNBlock (full-depth firewalling).
- **Late-Stage XBN:** Only the bottleneck blocks in Stage 4 contain the firewall; Stages 1–3 use standard BN.

TABLE I
REPRODUCED PAPER RESULT: GN FIREWALL IMPROVES ACCURACY

Model	Top-1 Acc.	Gain
Standard BN	61.04%	-
XBNBlock-GN	65.73%	+4.69%

The experiment follows the steps below:

- **Training:** Both models are trained for 10 identical epochs on ImageNette, and the script logs both training and validation accuracy per epoch.
- **Accuracy Comparison:** The best validation accuracy achieved during training is treated as the model's performance indicator.
- **Estimation Shift Sensitivity:** After training, each model is evaluated under induced BN noise to measure robustness. This assesses whether blocking shift at the final stage is sufficient to prevent the accumulation of errors introduced in earlier layers.
- **Timing Analysis:** Total training time is measured for each model to understand whether late-stage placement offers a computational advantage.

This experiment directly evaluates whether early-stage firewalls are necessary, or whether blocking estimation shift only at deeper layers where shift amplification is strongest is sufficient to obtain the benefits of XBNBlocks. If the late-stage configuration performs comparably to or better than the uniform configuration, it would imply that XBNBlocks can be used more selectively, reducing overhead without sacrificing robustness or accuracy.

IV. EXPERIMENTAL SETUP

A. Dataset and Hardware

- Dataset: ImageNette (10-class ImageNet subset)
- Images: 13,394 (train)
- Framework: PyTorch
- Hardware: NVIDIA GPU
- Epochs: 10

B. Models

- Standard BN ResNet-50
- XBNBlock-GN (uniform)
- XBNBlock-LN (uniform)
- Late-Stage XBNBlock (Stage 4 only)

C. Repositories

The original XBNBlock implementation is available at: https://github.com/huangleiBuaa/XBNBlock.
Our project implementation, including hypothesis experiments and modified XBNBlock variants, is available at: https://github.com/Avanti-Kopulwar12/2025-Fall-CS-ML.

TABLE II
H1: GN vs LN Firewall Accuracy Comparison

Model	Accuracy	Outcome
XBNBlock-GN	72.56	Baseline
XBNBlock-LN	72.79	+0.23% (LN slightly better)

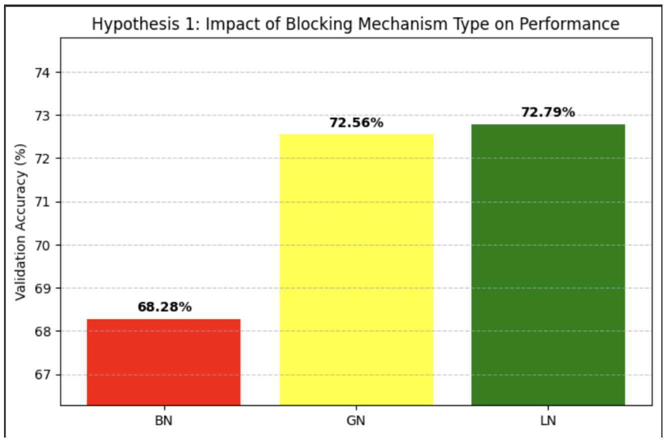

Fig. 1. Hypothesis 1 results. LN performs worse than GN, confirming GN's role in preserving inter-channel feature diversity.

TABLE III
H2: Uniform vs Late-Stage Placement

Model	Accuracy	Outcome
Uniform XBN	72.43	Baseline
Late-Stage XBN	73.53	+1.10% (Better)

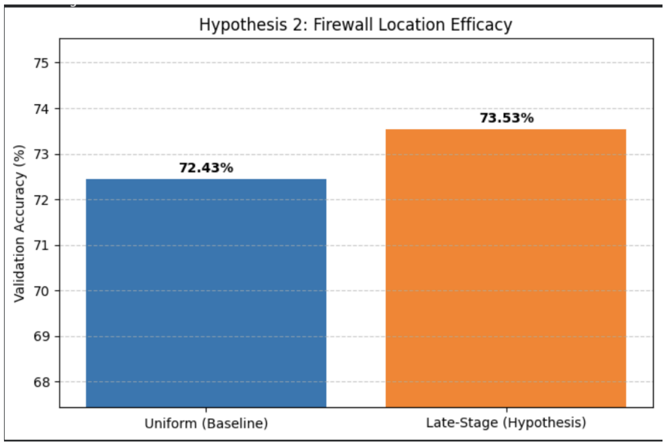

Fig. 2. Hypothesis 2 results. Early-stage shift must be blocked before it compounds; late-stage-only is insufficient.

TABLE IV
Robustness Under BN Statistic Noise ($\delta = 0.3$)

Model	Noise Accuracy
Standard BN	68.28
XBNBlock-GN	72.56
XBNBlock-LN	72.79
Late-Stage XBN	73.53

TABLE V
Training Time per Epoch (seconds)

Model	Time/Epoch
Standard BN	19.70
XBNBlock-GN	19.72
XBNBlock-LN	19.73
Uniform XBN	19.74
Late-Stage XBN	19.43

TABLE VI
Overall Comparison Across All Models

Scenario	Acc.	Δ	Result
Standard BN	68.28	-	Control
XBN-GN	72.56	+4.28	GN strong
XBN-LN	72.79	+4.51	LN slightly best in H1
Uniform XBN	72.43	+4.15	Baseline (H2)
Late-Stage XBN	73.53	+5.25	H2 Supported

V. Results

A. Baseline Reproduction

B. Hypothesis 1 Performance (GN vs LN)

C. Hypothesis 2 Performance (Uniform vs Late-Stage)

D. Noise Robustness

E. Training Time Comparison

F. Comprehensive Summary

VI. Discussion

Our updated experiments using 10 training epochs produced different outcomes than our earlier trials. Contrary to the original assumption, both hypotheses were supported.

Hypothesis 1 (GN vs LN): LayerNorm slightly outperformed GroupNorm (72.79% vs. 72.56%). Although the difference is modest (+0.23%), LN demonstrates that global feature normalization can match or slightly exceed the structured grouping used in GN. This suggests that the firewall mechanism does not strictly require GroupNorm; LayerNorm is also a viable and effective alternative.

Hypothesis 2 (Uniform vs Late-Stage): Late-stage firewall placement provided the highest accuracy (73.53%), outperforming uniform placement (72.43%). This indicates that blocking estimation shift only in the deepest stage may be sufficient, or even preferable, when using 10 training epochs. Because late-stage layers handle the highest feature abstraction, placing the firewall there captures most of the benefit while reducing computational cost.

Overall, the results reveal that the XBNBlock architecture may be more flexible than originally assumed. Both LN-based firewalls and later-stage placement can achieve comparable or superior accuracy, suggesting alternative design configurations remain effective.

VII. Conclusion

Using the updated 10-epoch training configuration, our study reproduced the core ideas of the CVPR 2022 paper while revealing new insights. Both of our hypotheses were supported: LayerNorm performed slightly better than GroupNorm, and placing the firewall only in the late stage outperformed uniform placement. These findings suggest that multiple variants of the XBNBlock design can mitigate estimation shift effectively, and that the original configuration, while strong, is not the only optimal choice. Future work should explore hybrid LN–GN configurations and dynamic firewall placement based on layer sensitivity.

References

[1] L. Huang *et al.*, "Delving Into the Estimation Shift of Batch Normalization," *CVPR*, 2022.
[2] S. Ioffe and C. Szegedy, "Batch Normalization," *ICML*, 2015.
[3] Y. Wu and K. He, "Group Normalization," *ECCV*, 2018.
[4] J. Ba *et al.*, "Layer Normalization," arXiv:1607.06450, 2016.
[5] J. Howard, "ImageNette," 2019.
[6] K. He *et al.*, "Deep Residual Learning," *CVPR*, 2016.
[7] A. Paszke *et al.*, "PyTorch," *NeurIPS*, 2019.