Ghost Noise for Regularizing Deep Neural Networks

Atli Kosson, Dongyang Fan, Martin Jaggi
EPFL, Switzerland
atli.kosson@epfl.ch, dongyang.fan@epfl.ch, martin.jaggi@epfl.ch

Abstract

Batch Normalization (BN) is widely used to stabilize the optimization process and improve the test performance of deep neural networks. The regularization effect of BN depends on the batch size and explicitly using smaller batch sizes with Batch Normalization, a method known as Ghost Batch Normalization (GBN), has been found to improve generalization in many settings. We investigate the effectiveness of GBN by disentangling the induced "Ghost Noise" from normalization and quantitatively analyzing the distribution of noise as well as its impact on model performance. Inspired by our analysis, we propose a new regularization technique called Ghost Noise Injection (GNI) that imitates the noise in GBN without incurring the detrimental train-test discrepancy effects of small batch training. We experimentally show that GNI can provide a greater generalization benefit than GBN. Ghost Noise Injection can also be beneficial in otherwise non-noisy settings such as layer-normalized networks, providing additional evidence of the usefulness of Ghost Noise in Batch Normalization as a regularizer.

1 Introduction

The adoption of normalization techniques has been a prevailing trend in the field of deep learning (Ioffe and Szegedy 2015; Ba, Kiros, and Hinton 2016; Wu and He 2018), notably following the seminal introduction of Batch Normalization (BN) (Ioffe and Szegedy 2015) in 2015. Batch Normalization operates on each feature independently, normalizing it by subtracting its mean and dividing by its standard deviation. The mean $\boldsymbol{\mu}$ and standard deviation (std) $\boldsymbol{\sigma}$ with elements corresponding to the different features in the input tensor. During training, these statistics are computed over the batch dimension, causing the network output for a given sample to depend on the others in the batch. This cross-sample dependency is undesirable at inference time, as data can arrive in small correlated batches or even one sample at a time. Consequently, we utilize population-level statistics during inference, instead of computing the mean and variance for individual batches. The population statistics are typically approximated by an exponential moving average of the mean and variance values observed during the training phase.

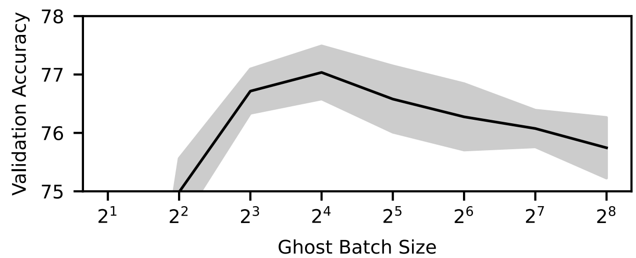

Figure 1: Impact of ghost batch size on ResNet-18 accuracy on CIFAR-100. Mean (solid) and std (shaded) for 3 runs.

The use of different forms of statistics in training and inference can lead to a *detrimental* train-test discrepancy resulting in lower test performance. One major difference is that during training each sample contributes to its normalization statistics, biasing the result. The smaller the batch size, the greater the impact of each sample on the statistics, thus a larger bias. We refer to this effect as the **self-dependency** of batch normalization. The population statistics used during inference are identical for all samples and therefore not biased in the same way. The distribution of normalized activations during testing can therefore be significantly different than during training, leading to worse generalization performance. This effect is studied by Singh and Shrivastava (2019) and Summers and Dinneen (2020) which propose modifying inference to reduce the train-test discrepancy. The use of correlated samples in a batch can similarly lead to a bias causing a different form of train-test discrepancy (Wu and Johnson 2021).

Several alternative methods such as Layer Normalization (Ba, Kiros, and Hinton 2016), Instance Normalization (Huang and Belongie 2017), Group Normalization (Wu and He 2018), Online Normalization (Chiley et al. 2019), Batch Renormalization (Ioffe 2017) and Weight Normalization (Salimans and Kingma 2016) have been proposed to overcome some of the issues with Batch Normalization. These methods either normalize across other dimensions of the tensor $\boldsymbol{X}$, apply normalization to the weights instead of activations, or make use of the history over prior batches to augment the effective size of the batch.

Aside from the train-test discrepancy, the dependency of the batch normalization statistics on the elements in the batch can also have a *beneficial* regularization effect, which is frequently observed but somewhat poorly understood. From a geometrical point of view, Keskar et al. (2017) empirically found that training with small batches arrives at flatter minima, which they argue contributes to better generalization. Hoffer, Hubara, and Soudry (2017) proposed **Ghost Batch Normalization (GBN)**, followed by observations where performing batch normalization over smaller subgroups of the batch could improve generalization. These smaller batches were termed *ghost batches* by them. Summers and Dinneen (2020) also observed this effect and recommended the use of GBN. The regularization effect arises from the noise in $\boldsymbol{\mu}, \boldsymbol{\sigma}$ computed over a batch (typically randomly sampled) compared to the corresponding statistics for the full dataset. This noise, which we will refer to as **Ghost Noise**, arises from the **cross-sample dependency** and increases for smaller batches, potentially explaining the improved generalization with GBN. Figure 1 shows an example of how the ghost batch size can affect performance.

In this work, we analyze the effects of ghost noise and train-test discrepancy in BN. The strength of both effects increases with smaller batch sizes, creating *a trade-off between the regularization from the ghost noise and the detrimental effect of train-test discrepancy*. Section 3 constitutes the main part of our analysis and proposed methods. We first propose **Exclusive Batch Normalization (XBN)**. This method mitigates the train-test discrepancy in GBN by simply excluding a sample from the computation of its normalization statistics. This way, self-dependency is eliminated during training, as in inference time. However, it requires moderately sized batches for stable training. Following that, we introduce a novel technique named **Ghost Noise Injection (GNI)**. GNI decouples ghost noise from the normalization process, allowing increased regularization from ghost noise without actually performing normalization using small batches. This method avoids the detrimental effects associated with small batches and can be extended to BN-free settings, such as Layer Normalization, where inherent noise is absent.

Experimentally, we find that GNI can deliver a stronger regularization effect than GBN, resulting in improved test performance across a wide range of training settings. Our ablation study suggests that both the noise in $\boldsymbol{\mu}$ (shifting noise) and in $\boldsymbol{\sigma}$ (scaling noise) contribute to the regularization effect of GNI. In Convolutional Neural Networks (CNNs), the noise magnitude can vary between channels and layers depending on their distribution, in particular, how much of their variance comes from the spatial extent of a single sample compared to the variance between samples. This "adaptivity" may be important, as we find that simpler i.i.d. (independent and identically distributed) noise methods are unable to match the regularization effect of GNI. This includes several dropout variants as well as i.i.d. noise based on our analysis of the ghost noise distribution.

Summary of Contributions:

- We examine the impact of smaller batch sizes in GBN, focusing on the potentially useful regularization effect from ghost noise and the harmful train-test discrepancy, offering insights into the trade-off between the two.
- We introduce Exclusive Batch Normalization (XBN) to mitigate the train-test discrepancy. Our findings also reveal that inherent self-dependency contributes to stability during small-batch training.
- By deconstructing GBN into two constituent components, we devise a novel method termed Ghost Noise Injection (GNI). This technique allows us to specifically investigate the noise effects associated with small batch sizes.
- Extensive experimentation highlights the utility of GNI as an effective regularization method across diverse architectures and training scenarios. This underscores the significance of the intrinsic noise within BN.

2 Ghost Batch Normalization

In this section we provide an overview of Ghost Batch Normalization (GBN) from Hoffer, Hubara, and Soudry (2017), followed by a novel modeling of GBN as a composition of GBN and BN, separating the normalization and noise effects.

2.1 Overview

In short, GBN is performed by applying Batch Normalization on disjoint subsets of a batch, i.e. ghost batches. By intentionally operating on smaller batch sizes, GBN increases the stochasticity in the normalization statistics compared to standard batch normalization. This has been found to give a beneficial regularization effect in certain settings, see e.g. Figure 1. Our primary goal in this work is to explore and isolate the effect of Ghost Batch Normalization.

A batch $\boldsymbol{X} \in \mathbb{R}^{B \times C \times H \times W}$ can be divided into $G = B/N$ ghost batches $\{\boldsymbol{X}_1, \boldsymbol{X}_2, ...\boldsymbol{X}_G\}$, of size $N \times C \times H \times W$ each. The output of GBN is obtained by performing batch normalization on each ghost batch. For ghost batch $\boldsymbol{X}_g$ with elements $X_{n,c,h,w}$, this can be expressed as:

$$\widetilde{\boldsymbol{X}}_g = \frac{(\boldsymbol{X}_g - \boldsymbol{\mu}_g)}{\sqrt{\boldsymbol{\sigma}_g^2 + \varepsilon}} \tag{1}$$

where $\boldsymbol{\mu}_g, \boldsymbol{\sigma}_g^2 \in \mathbb{R}^{1 \times C \times 1 \times 1}$, $\boldsymbol{\mu}_g = \frac{1}{NHW} \sum_{n,h,w} X_{n,c,h,w}$, $\boldsymbol{\sigma}_g^2 = \frac{1}{NHW} \sum_{n,h,w} (X_{n,c,h,w} - \boldsymbol{\mu}_g)^2$ and we use broadcasted operations similar to PyTorch (Paszke et al. 2019).

Here we note that the term "batch" is highly overloaded, encompassing different batch sizes of interest. *Accelerator batch size* (denoted B) is the number of samples each worker (e.g. GPU) uses during a single forward / backward pass through the network. *Normalization batch size* or ghost batch size (denoted N) is the number of samples over which normalization statistics are calculated.

2.2 Modeling GBN as Double Normalization

In this section, we present a novel perspective on GBN aimed at separating noise effects from normalization effects. The key insight lies in the fact that applying standard batch normalization before ghost batch normalization preserves the output of GBN. For a batch $\boldsymbol{X}$, where batch normalization is denoted as BN and ghost batch normalization as GBN, this relationship can be expressed as:

$$\mathrm{GBN}(\boldsymbol{X}) = \mathrm{GBN}(\mathrm{BN}(\boldsymbol{X})) \tag{2}$$

which follows from the fact that normalization is invariant to affine transformations of the inputs.

This decomposition of GBN into two successive normalization operations lets us isolate the differences between standard batch normalization and ghost batch normalization. Ignoring ε, the double normalization can be formulated as:

$$\hat{\boldsymbol{X}} := \mathrm{BN}(\boldsymbol{X}) = \frac{\boldsymbol{X} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} = [\hat{\boldsymbol{X}}_1, .., \hat{\boldsymbol{X}}_G] \tag{3}$$

$$\widetilde{\boldsymbol{X}}_g := \mathrm{GBN}(\hat{\boldsymbol{X}})_g = \frac{\hat{\boldsymbol{X}}_g - \hat{\boldsymbol{\mu}}_g}{\hat{\boldsymbol{\sigma}}_g} \tag{4}$$

where we split the normalized batch into G ghost batches and show the following GBN for the g-th subgroup. Following the equivalence of Equation 1 and Equation 4, we have:

$$\frac{\boldsymbol{X}_g - \boldsymbol{\mu}_g}{\boldsymbol{\sigma}_g} = \frac{1}{\hat{\boldsymbol{\sigma}}_g}\left(\frac{\boldsymbol{X}_g - \boldsymbol{\mu}}{\boldsymbol{\sigma}} - \hat{\boldsymbol{\mu}}_g\right) \tag{5}$$

We can now write the $\boldsymbol{\mu}_g$ and $\boldsymbol{\sigma}_g$ of the original GBN($\boldsymbol{X}$) setup in Equation 1 as:

$$\boldsymbol{\mu}_g = \boldsymbol{\mu} + \boldsymbol{\sigma}\hat{\boldsymbol{\mu}}_g, \qquad \boldsymbol{\sigma}_g = \boldsymbol{\sigma}\hat{\boldsymbol{\sigma}}_g \tag{6}$$

Standard GBN would be equivalent to BN if $\hat{\boldsymbol{\mu}}_g = 0$ and $\hat{\boldsymbol{\sigma}}_g = 1$. This happens when the ghost batches are identical, but generally, they contain different samples resulting in random variations in $\hat{\boldsymbol{\mu}}_g$ and $\hat{\boldsymbol{\sigma}}_g$. These fluctuations induce noise in two components, additive (or shifting) noise from $\hat{\boldsymbol{\mu}}_g$ and multiplicative (or scaling) noise from $\hat{\boldsymbol{\sigma}}_g$. The increased train-test discrepancy of GBN arises from the dependency of $\hat{\boldsymbol{\mu}}_g$ and $\hat{\boldsymbol{\sigma}}_g$ on a specific sample (self-dependency).

3 Methods and Analysis

Within this section, we introduce two innovative approaches designed to enhance the generalization performance of Deep Neural Networks (DNNs): 1) Exclusive Batch Normalization, which addresses the issue of train-test discrepancy from self-dependency; 2) Ghost Noise Injection, a method that enhances training-time stochasticity by incorporating noise from ghost batches. We provide a detailed and clear analysis of this technique.

3.1 Exclusive Batch Normalization (XBN)

We first consider a straightforward technique in which all elements within a ghost batch are utilized except for the element under consideration, for the calculation of normalization statistics. This aims to mitigate the excessive reliance on a single sample during training. We term this approach **Exclusive Batch Normalization (XBN)**. Note that XBN applies a separate set of normalization statistics for each sample. Consequently, the training-time normalization statistics become less dependent on the specific sample itself, aligning more closely with the conditions at test time. The mean and variance of the k-th sample in the g-th ghost batch of XBN is computed as:

$$\boldsymbol{\mu}_{g,k} = \frac{1}{(N-1)HW}\sum_{n\neq k,h,w} X_{n,c,h,w} \tag{7}$$

$$\boldsymbol{\sigma}^2_{g,k} = \frac{1}{(N-1)HW}\sum_{n\neq k,h,w} (X_{n,c,h,w} - \boldsymbol{\mu}_{g,k})^2 \tag{8}$$

Importantly, XBN maintains a similar level of noise as GBN while reducing the discrepancy between the train and test time. However, eliminating the self-dependency of normalization comes with a significant drawback: it does not guarantee a bounded output range. We present a detailed analysis in Appendix B[1]. The absence of an output range constraint can lead to training instability. We empirically observe this behavior for smaller batch sizes. Yet, when the batch size is sufficiently large to support stable training, we do observe an increase in test accuracy, as demonstrated in Figure 3. This increase is likely attributed to the reduced train-test discrepancy.

XBN offers a way to address the train-test discrepancy in GBN which would in theory also allow us to obtain increased ghost noise by just decreasing the ghost batch size. However, the instability of XBN for smaller batch sizes prevents this so we seek an improved method that does not suffer from this drawback. Despite this XBN remains a valuable baseline for performance comparison.

3.2 Ghost Noise Injection (GNI)

Building upon our findings in Section 2.2, we can identify $\hat{\boldsymbol{\mu}}_g$ and $\hat{\boldsymbol{\sigma}}_g$ in Equation 6 as the source of the ghost noise and the train-test discrepancy arising from self-dependency. We propose replacing the GBN in double normalization with:

$$\widetilde{\boldsymbol{X}} = \frac{\hat{\boldsymbol{X}} - \hat{\mathbf{m}}}{\hat{\mathrm{s}}} \tag{9}$$

where the elements of $\hat{\mathbf{m}}, \hat{\mathrm{s}} \in \mathbb{R}^{N\times C\times 1\times 1}$ for a given input sample are computed as the mean and standard deviation of a **randomly sampled** (with replacement) ghost batch from $\hat{\boldsymbol{X}}$. This departs from the methodology of GBN, where computations are based on the corresponding ghost batch. Since the ghost batch from which the normalization statistics are calculated is randomly sampled, the current sample itself is not necessarily included in the ghost batch. The self-dependency is thus reduced during training time. We refer to the size of this subset as the ghost batch size, similar to GBN. The resulting $\mathbf{m}$ and s are treated as pure noise, and *we do not perform backpropagation through their computation*.

Equation 9 assumes the inputs $\hat{\boldsymbol{X}}$ are batch normalized. In general we may want to apply noise in more settings e.g. where no normalization is present. This leads us to propose a more general form —– Ghost Noise Injection (GNI):

$$\frac{\boldsymbol{X} - (\mathbf{m} - \boldsymbol{\mu})}{\mathrm{s}/\boldsymbol{\sigma}} \tag{10}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are computed like in batch normalization, and $\mathbf{m}$ and s are mean and standard deviation from a randomly sampled ghost batch of the *unnormalized* input $\boldsymbol{X}$. $\mathbf{m} - \boldsymbol{\mu}$ is treated as **shift noise** and $\mathrm{s}/\boldsymbol{\sigma}$ is treated as **scale noise**. In Appendix C we give a derivation of Equation 10 by "undoing" the effect of the first normalization in double normalization. A minimal implementation of GNI is shown in Figure 2. GNI offers two key advantages over GBN:

- Each sample is unlikely to contribute strongly to its own normalization statistics, reducing the train-test discrepancy for a given level of noise (similar to XBN).

[1]See arXiv:2305.17205 for an extended version with appendices.

```python
import torch
def ghost_noise_injection(X: torch.Tensor, N: int, eps: float=1e-3):
    B, C, _, _ = X.shape  # Shape: Batch (B), Channels (C), Height (H), Width (W)
    with torch.no_grad():
        batch_var, batch_mean = torch.var_mean(X, dim=(0,2,3), correction=0)
        ghost_var = torch.zeros(size=(B,C), device=X.device)
        ghost_mean = torch.zeros(size=(B,C), device=X.device)
        for idx in range(B):
            ghost_sample = torch.randint(high=B, size=(N,))
            ghost_var[idx], ghost_mean[idx] = torch.var_mean(
                X[ghost_sample], dim=[0,2,3], correction=0)
        shift = (ghost_mean - batch_mean)
        scale = torch.sqrt((ghost_var + eps)/(batch_var + eps))
    return (X - shift.view(B, C, 1, 1)) / scale.view(B, C, 1, 1)
```

Figure 2: A minimal PyTorch implementation of Ghost Noise Injection for convolutional activations maps without any performance optimizations. A small epsilon value helps with the numerical stability of the computation (1e-3 in all experiments).

- The choice of normalization batch size N is not limited to being a divisor of the accelerator batch size.

GNI comes at a slight computation and memory overhead which we discuss further in Appendix F.

3.3 Analytical Form of Ghost Noise Injection

We further offer an analytical version of GNI, by modeling the distribution of the shift and scale noises. The estimated mean $\hat{\boldsymbol{\mu}}_g$ and standard deviation $\hat{\boldsymbol{\sigma}}_g$ from the g-th ghost batch, can be interpreted as bootstrapped statistics derived from the empirical distribution $P_{\hat{\boldsymbol{X}}}$. Following the normalization in Equation 3, the variable $\hat{\boldsymbol{X}}$ exhibits a zero mean and unit variance. Assuming that the individual elements of $\hat{\boldsymbol{X}}$ are normally distributed allows us to derive an analytical distribution for the mean and variance, giving us additional insights into the workings of GBN. For this section we focus on the distribution for a single channel c in the g-th ghost batch, which we denote $\hat{\mu}_{gc}$ and $\hat{\sigma}_{gc}$.

Case 1: Fully Connected Layers Assuming that the output of BN is independent and normally distributed, the normalization statistics $\hat{\boldsymbol{\mu}}_g$ and $\hat{\boldsymbol{\sigma}}_g^2$ are computed over a ghost batch $\hat{\boldsymbol{X}}_g = [\hat{\boldsymbol{x}}_1, .., \hat{\boldsymbol{x}}_N] \in \mathbb{R}^{N \times C}$ of N elements sampled i.i.d. from $\mathcal{N}(0, 1)$. We can then derive the distribution of the sample mean, and therefore the shift noise, as:

$$\hat{\mu}_{gc} = \frac{1}{N} \sum_{n=1}^{N} \hat{x}_{nc} \sim \mathcal{N}\left(0, \frac{1}{N}\right) \tag{11}$$

The sum of the square of independent standard normally distributed variables follows a Chi-squared distribution:

$$\hat{\sigma}_{gc}^2 = \frac{1}{N} \sum_{n=1}^{N} (\hat{x}_{nc} - 0)^2 \sim \frac{1}{N} \chi^2(N) \tag{12}$$

This clearly shows the dependency of the noise on the ghost batch size. Larger ghost batch sizes correspond to less noise, explaining their reduced generalization benefit as observed in e.g. Figure 1.

- **Analytical Ghost Noise Injection:** Based on the preceding analysis, an alternative approach to computing ghost statistics from sampled batches is to directly sample $\hat{\boldsymbol{\mu}}_g$ and $\hat{\boldsymbol{\sigma}}_g^2$ from the respective analytical distributions for each channel. Let $\hat{\boldsymbol{X}}$ denote $\boldsymbol{X}$ post batch normalization, as previously defined. Subsequently, for each channel c, we can then inject noise using:

$$\frac{\hat{\boldsymbol{X}}_c - \mu_c}{\sigma_c} \qquad \text{with } \mu_c \sim P_{\hat{\mu}_{gc}}, \sigma_c \sim P_{\hat{\sigma}_{gc}^2} \tag{13}$$

  $P_{\hat{\mu}_{gc}}$ is given in Equation 11 and $P_{\hat{\sigma}_{gc}^2}$ is given in Equation 12. The hyperparameter N is used to vary the amount of noise, corresponding to different ghost batch sizes.
- **Comparison to Gaussian Dropout:** It is noteworthy to highlight the similarity between the scaling noise and Gaussian Dropout, which is written as:

$$\hat{\boldsymbol{X}} \cdot \mathbf{t} \qquad \text{with } \mathbf{t} \sim \mathcal{N}(1, \tfrac{p}{1-p}) \tag{14}$$

  where $\mathbf{t}$ is sampled element-wise from the Gaussian distribution and p is interpreted like the drop probability in standard dropout ($p = 0$ for no dropout). Both the scaling noise and Gaussian dropout are multiplicative with similar yet slightly different distributions.

Case 2: Convolutional Layers In CNNs the batch statistics are computed across both the batch and spatial dimensions (e.g. the height and width of an image). As a result, the data can exhibit variability in two distinct manners: variation among samples along the batch dimension, termed *inter-sample variance*, and variation within a single sample across the spatial dimension, referred to as *intra-sample variance*. This stands in contrast to the fully connected scenario, where all variance emerges along the batch dimension, and the concept of intra-sample variance is absent.

To gain insight into how the inter- and intra-sample variances could affect the noise distributions we will present and analyze a simplified model. We will assume that $\boldsymbol{X} \in \mathbb{R}^{B \times C \times I}$, where I is the spatial dimension e.g. $I = H \times W$ where H is the height and W the width for image data. We further assume the true mean of b-th batch in a specific channel c is sampled i.i.d. from a normal distribution, i.e. $\mu_{bc} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{Bc}^2)$, where σ_{Bc}^2 denotes the

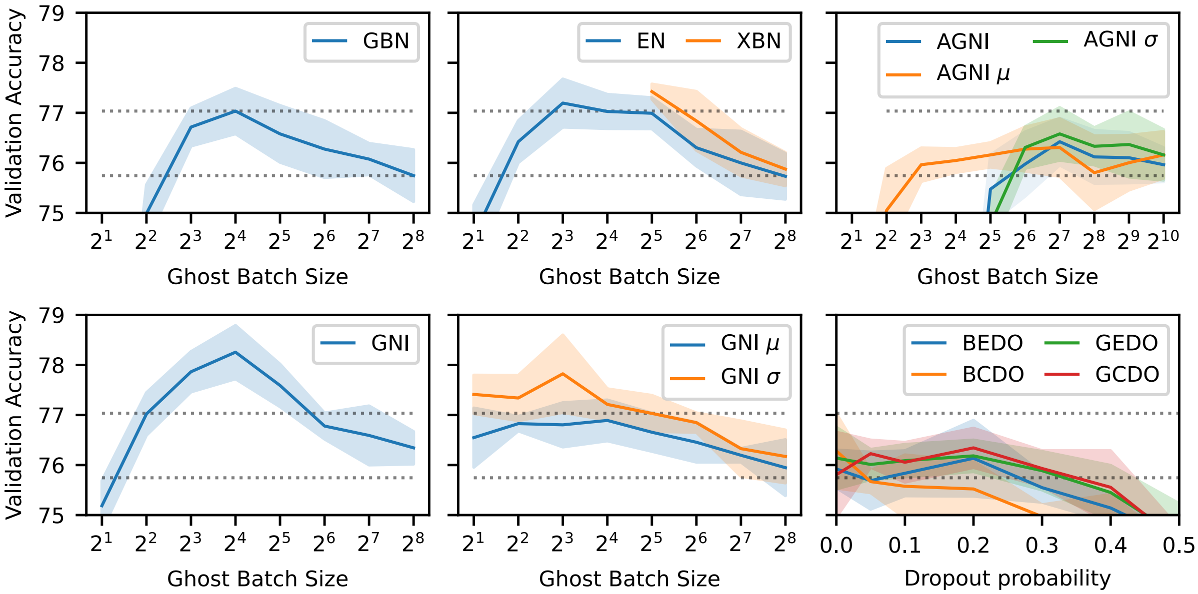

Figure 3: CIFAR-100 ResNet-18 validation accuracy versus ghost batch size and dropout probability for different methods. Each line is the average of five runs and the shaded area shows the standard deviation. The dotted lines show the standard batch normalization performance and the maximum for ghost batch normalization. For dropout (DO, lower right panel), B=Bernoulli, G=Gaussian, E=elementwise and C=channelwise. See Experiments section for further details and discussion.

inter-sample variance and only varies across c, i.e. it is constant across B. We model the i-th spatial location in the b-th batch as being sampled from $x_{bci} \overset{i.i.d.}{\sim} \mathcal{N}(\mu_{bc}, \sigma_{Ic}^2)$, with an intra-sample variance σ_{Ic}^2 that does not vary across I. After batch normalization, we always have unit variance, which means $\sigma_{Bc}^2 + \sigma_{Ic}^2 = 1$, for any c. Now assume we sample a random ghost batch of size N. As the sample mean is an average of all samples, we still have it following a normal distribution. We defer the calculation details of the distributions to Appendix D. The resulting expressions for the distributions of the scale and shift noise in channel c are:

$$\hat{\mu}_{gc} \sim \mathcal{N}\left(0, \frac{1}{NI}\sigma_{Ic}^2 + \frac{1}{N}\sigma_{Bc}^2\right) \tag{15}$$

$$\hat{\sigma}_{gc}^2 \sim \frac{\sigma_{Ic}^2}{NI}\chi^2(NI) + \frac{\sigma_{Bc}^2}{N}\chi^2(N) \tag{16}$$

When $\sigma_{Ic}^2 = 0$, the outcomes align with the 1D scenario. However, the analysis in the 2D case affords us greater flexibility to distinguish between inter-sample variance (σ_{Bc}^2) and intra-sample variance (σ_{Ic}^2). An additional noteworthy insight pertains to channel dependency. Given that both σ_{Bc}^2 and σ_{Ic}^2 are specific to individual channels, a distinct noise effect is introduced for each channel. This phenomenon will be empirically explored in Figure 4.

4 Experiments and Discussions

The setup of our experiments is detailed in Appendix A.

4.1 Comparison of Methods

Baseline - GBN The top left panel of Figure 3 shows how the ghost batch size affects the final validation accuracy when using Ghost Batch Normalization. Using a ghost batch size $N = 256$ is equivalent to standard batch normalization. We see that the accuracy increases for smaller batch sizes up to a certain extent, after which it goes down again. The initial improvement could be attributed to heightened regularization, while the subsequent decrease might arise from excessive regularization or the accentuated bias in normalization statistics, potentially leading to a discrepancy between training and testing. The optimal $N = 16$ gives an accuracy boost of just over 1% on both the validation and the test sets (Table 1).

XBN The upper-middle panel of Figure 3 delves into the efficacy assessment of Exclusive Batch Normalization (XBN) and EvalNorm (Singh and Shrivastava 2019) (EN). The primary objective of both techniques is the enhancement of test performance through the mitigation of the train-test discrepancy. XBN accomplishes this by altering sample interdependence during training to align more closely with inference conditions, while EvalNorm adopts the reverse approach. XBN can be unstable for small batch sizes (as discussed in Section 3.1), in this case batch sizes smaller than 32. Despite

	BN	GBN-16	GNI-16	XBN	EN
Mean	77.10	78.20	**78.84**	78.33	78.33
Std	0.26	0.10	0.09	0.40	0.27

Table 1: CIFAR-100 test accuracy (mean±std% for 3 runs).

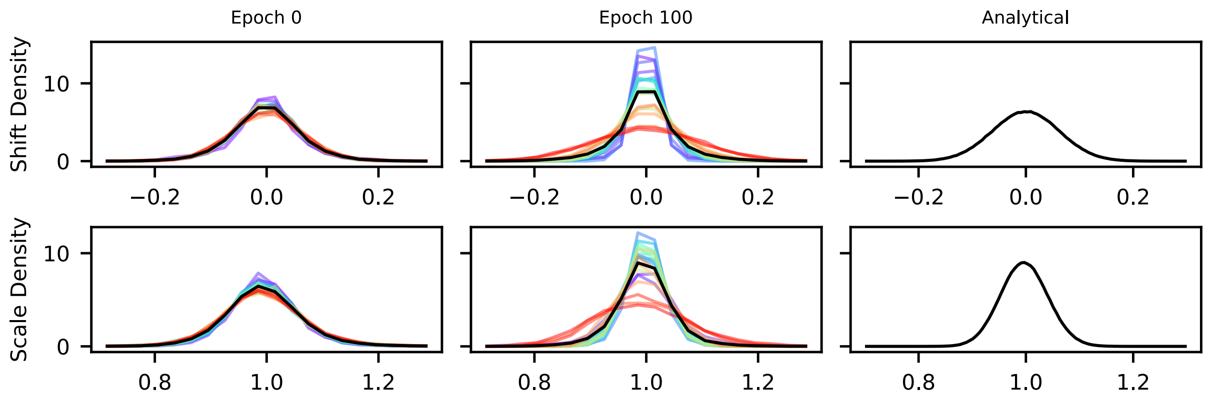

Figure 4: Measured GNI noise distributions for ResNet18 on CIFAR100 with a Ghost Batch Size of 32 along with analytical fully connected distributions for batch size 256. Each GNI line is an average over a layer, the distributions also vary between channels inside each layer. The lines are plotted with a standard color spectrum (rainbow) from violet (first layer) to red (last layer).

this, both XBN and EN appear to effectively alleviate certain aspects of the train-test discrepancy, yielding a marginal accuracy increase in comparison to GBN. Specifically, XBN exhibits slightly superior performance on the validation set, while producing comparable outcomes on the test set (as indicated in Table 1).

Sample-based GNI Within Figure 3, the lower-left panel presents the performance evaluation of Ghost Noise Injection (GNI) across varying ghost batch sizes. Remarkably, we note a substantial enhancement—approximately double that of GBN. At higher ghost batch sizes, GNI performs similarly to XBN. This observation suggests that GNI effectively mitigates the train-test discrepancy while stabilizing training with smaller batch sizes, allowing us to benefit from their increased regularization effect. Notably, among the explored methodologies, GNI emerges as the most successful, showcasing superior performance on the test set (Table 1. In the lower middle panel of Figure 3 we investigate the effect of the different noise components of GNI. GNI $\boldsymbol{\mu}$ only includes the shift and GNI $\boldsymbol{\sigma}$ only the scaling term. Evidently, each component independently delivers a significant performance enhancement. However, their individual impacts fall short of matching the comprehensive benefits of GNI. This strongly implies that both constituents of Ghost Noise collectively yield advantageous outcomes within this context.

Ghost Noise Distribution Figure 4 shows measured distributions of both the scale and shift components of the noise generated by GNI. The observed average distributions are similar to our derived distribution for the fully connected case, but the batch size parameter must be adjusted to account for the intra-sample variance component, as elaborated in Equation 16. The depicted distributions exhibit notable variations across layers and channels (though not visually presented here), particularly in the later stages of training. This variability could potentially be attributed to fluctuations in the intra-sample variance. In the top right panel of Figure 3 we apply our analytical GNI **(AGNI)** to training. Although we observe some improvements, it is unable to match the performance of the sample-based GNI. This suggests a potential significance associated with the *channel-specific* fluctuations. In Appendix E we further measure the distribution of ghost noise for a fully connected network.

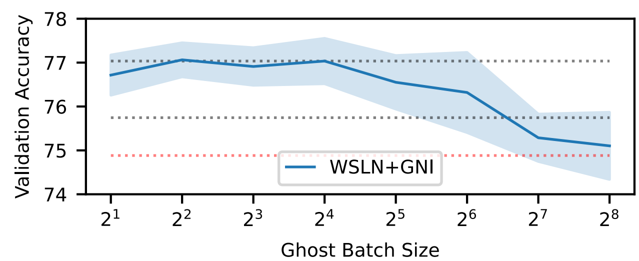

Figure 5: Accuracy impact of Ghost Noise Injection on Weight Standardized Layer Normalized (WSLN) ResNet18 trained on CIFAR100. The two upper black dotted lines correspond to Figure 3 (BN baseline, best GBN performance) and the bottom red dotted line to WSLN without GNI.

GNI vs Dropout GNI is a regularizer and a potential alternative to Dropout. In Figure 3 (bottom right) we compare four variants of dropout (Bernoulli/Gaussian and Elementwise/Channel-wise) to the other methods. We apply the dropout after the second normalization layer on each branch, as was done in Wide Residual Networks (Zagoruyko and Komodakis 2017). We find that dropout performs similarly to AGNI but is unable to match GBN or sample-based GNI.

4.2 Wider Applicabilities of GNI

Applicability to other Normalizers So far we have applied GNI on top of Batch Normalization. However, GNI is not limited to this use case. Figure 5 shows the CIFAR-100 validation accuracy of a ResNet-18 where batch normalization has been replaced by weight-standardization (Qiao et al. 2020) and layer normalization (Ba, Kiros, and Hinton 2016). We find that GNI can also provide a significant accuracy boost

Setup	CIFAR-10 RN-20	ImageNet-1k RN-50
BN-256	92.22±0.05	77.43
GBN-16	92.65±0.07	76.90
GBN-32	92.53±0.33	77.14
GBN-64	92.50±0.13	77.48
GNI-16	**93.01**±0.28	77.62
GNI-32	92.76±0.11	77.70
GNI-64	92.56±0.12	**77.98**

Table 2: Test accuracy for ResNet-20 CIFAR-10 (mean±std% for 3 runs) and ResNet-50 ImageNet-1k for different normalization and noise setups. GNI is performed on top of the standard BN-256 for additional regularization.

Network	Aug	Baseline	GNI (N)
NF ResNet	✗	84.31±0.86	86.18±0.58 (32)
NF ResNet	✓	93.60±0.21	94.19±0.24 (32)
Simple ViT	✗	71.45±0.12	74.36±0.27 (4)
Simple ViT	✓	88.19±0.08	89.55±0.05 (12)
ConvMixer	✗	91.18±0.24	91.57±0.08 (192)
ConvMixer	✓	93.29±0.17	93.66±0.12 (128)

Table 3: Test Accuracy (mean±std% for 3 runs) on CIFAR-10 with/without data augmentation (Aug) and with/without GNI (using ghost batch size N). See Appendix A for details.

in this setting, bringing the performance with this noise-free normalization method in line with that of GBN.

Generalization to other Datasets Table 2 shows that GNI can also regularize the training of ResNet-50 on ImageNet-1k and ResNet-20 on CIFAR-10. In both cases, GNI provides a decent boost in accuracy and outperforms GBN.

Extension to other Architectures To investigate the adaptability of GNI to diverse architectures, we perform additional experiments on CIFAR-10 using a Normalization Free ResNet (Brock et al. 2021), a Simple Visual Transformer (Beyer, Zhai, and Kolesnikov 2022; Dosovitskiy et al. 2021; Cordonnier, Loukas, and Jaggi 2020) and a ConvMixer (Trockman and Kolter 2023). We explore two data augmentation settings, no augmentation and the ResNet style augmentation which consists of a random flip, padding and a random crop. Further details about the networks and other experimental details are given in Appendix A. In each case, GNI can improve the test accuracy showing that it can generalize to other architectures. We typically observe a greater performance boost when weaker data augmentation is used. Excessive data augmentation may already prevent a network of a given size that is trained for limited time from fitting the training data, reducing the benefit of additional regularization. The ghost batch size N was tuned on a validation set and varies considerably between the settings.

5 Related Work

Mitigating Train-test Discrepancy Singh and Shrivastava (2019) proposed EvalNorm to estimate corrected normalization statistics to use for BN during evaluation, such that training and testing time normalized activation distributions become closer. Summers and Dinneen (2020) incorporated example statistics during inference, which reduces the output range of a network with Batch Normalization during testing. While both works try to mitigate train-test discrepancy by alternating test time normalization statistics, XBN shows that it is also possible to address the same issue by reducing sample dependency during training time.

Noise Injection Liu et al. (2022) proposed a composition of two trainable scalars and a zero-centered noise injector for regularizing normalization-free DNNs. Camuto et al. (2020) analyzed Gaussian Noise Injection from a theoretical point of view and found injected noise particularly regularises neural network layers closer to the output. We compare the GNI to these methods in Appendix E. Compared to these two, Ghost Noise Injection more closely imitates the noise of batch normalization, accounting for the channel and layer-wise differences in the distribution. Shekhovtsov and Flach (2019) explored the noise in batch normalization from a Bayesian perspective, obtaining similar results to Equations 11-12, but did not analyze the layer and channel dependencies.

Dropout Dropout was first proposed by Srivastava et al. (2014) as a simple regularization approach to prevent units from co-adapting too much. Wei, Kakade, and Ma (2020) demonstrates the explicit and implicit regularization effects from dropout and found out that the implicit regularization effect is analogous to the effect of stochasticity in small mini-batch stochastic gradient descent. Hou and Wang (2019); He et al. (2022) applied channel-wise dropout and experimentally showed consistent benefits in DNNs with convolutional structures. Further, it is observed that channel-wise dropout encourages its succeeding layers to minimize the intra-class feature variance (He et al. 2022).

6 Conclusion

In this study, we explored an important aspect of batch normalization – the generalization benefit from smaller batch sizes. By formulating GBN as a series of two normalization operations, we are able to analyze the impact of smaller batch sizes on the noise and train-test discrepancy. Our analysis of the noise component unveiled its channel-dependent nature, comprising two distinct facets—shifting and scaling—both instrumental in the overall effectiveness. Furthermore, we demonstrated that the train-test discrepancy could be alleviated by preventing a sample from contributing to its own normalization statistics during training. Building on these insights, we introduced Ghost Noise Injection (GNI), a novel technique that elevates ghost noise levels without necessitating normalization over smaller batches, thus diminishing the train-test mismatch. Empirical investigations across various scenarios, including those without batch normalization, showcased GNI's beneficial impact on generalization, underscoring ghost noise's significance as a potent source of regularization in batch normalization.

Acknowledgments

We thank Maksym Andriushchenko for his feedback on the manuscript, which has enhanced its quality and clarity.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Beyer, L.; Zhai, X.; and Kolesnikov, A. 2022. Better plain ViT baselines for ImageNet-1k. *arXiv preprint arXiv:2205.01580*.

Brock, A.; De, S.; Smith, S. L.; and Simonyan, K. 2021. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, 1059–1071. PMLR.

Camuto, A.; Willetts, M.; Simsekli, U.; Roberts, S. J.; and Holmes, C. C. 2020. Explicit regularisation in gaussian noise injections. *Advances in Neural Information Processing Systems*, 33: 16603–16614.

Chiley, V.; Sharapov, I.; Kosson, A.; Koster, U.; Reece, R.; Samaniego de la Fuente, S.; Subbiah, V.; and James, M. 2019. Online normalization for training neural networks. *Advances in Neural Information Processing Systems*, 32. ArXiv:1905.05894.

Cordonnier, J.-B.; Loukas, A.; and Jaggi, M. 2020. On the Relationship between Self-Attention and Convolutional Layers. In *International Conference on Learning Representations*.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, M.; Zhang, J.; Shan, S.; Liu, X.; Wu, Z.; and Chen, X. 2022. Locality-Aware Channel-Wise Dropout for Occluded Face Recognition. *IEEE Transactions on Image Processing*, 31: 788–798.

Hoffer, E.; Hubara, I.; and Soudry, D. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30.

Hou, S.; and Wang, Z. 2019. Weighted Channel Dropout for Regularization of Deep Convolutional Neural Network. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press. ISBN 978-1-57735-809-1.

Huang, X.; and Belongie, S. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, 1501–1510.

Ioffe, S. 2017. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *Advances in neural information processing systems*, 30.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. pmlr.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. arXiv:1609.04836.

Krizhevsky, A.; and Hinton, G. 2009. CIFAR-100 (Canadian Institute For Advanced Research). *Dataset available from https://www.cs.toronto.edu/ kriz/cifar.html*.

Liu, C.; Yang, Y.; Ding, Y.; and Lu, H. 2022. NoMorelization: Building Normalizer-Free Models from a Sample's Perspective. arXiv:2210.06932.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Qiao, S.; Wang, H.; Liu, C.; Shen, W.; and Yuille, A. 2020. Micro-Batch Training with Batch-Channel Normalization and Weight Standardization. arXiv:1903.10520.

Salimans, T.; and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29.

Shekhovtsov, A.; and Flach, B. 2019. Stochastic normalizations as bayesian learning. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part II 14*, 463–479. Springer.

Singh, S.; and Shrivastava, A. 2019. Evalnorm: Estimating batch normalization statistics for evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3633–3641.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958.

Summers, C.; and Dinneen, M. J. 2020. Four Things Everyone Should Know to Improve Batch Normalization. arXiv:1906.03548.

Trockman, A.; and Kolter, J. Z. 2023. Patches Are All You Need? *Transactions on Machine Learning Research*. Featured Certification.

Wang, P. 2023. vit-pytorch. https://github.com/lucidrains/vit-pytorch. Accessed: 2023-08-09 (commit 3e5d1be).

Wei, C.; Kakade, S.; and Ma, T. 2020. The implicit and explicit regularization effects of dropout. In *International conference on machine learning*, 10181–10192. PMLR.

Wightman, R. 2019. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models. Accessed: 2023-03-24 (commit d196fa53).

Wu, Y.; and He, K. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, 3–19.

Wu, Y.; and Johnson, J. 2021. Rethinking" batch" in batchnorm. *arXiv preprint arXiv:2105.07576*.

Zagoruyko, S.; and Komodakis, N. 2017. Wide Residual Networks. arXiv:1605.07146.