EvalNorm: Estimating Batch Normalization Statistics for Evaluation

Saurabh Singh
Google Research
saurabhsingh@google.com

Abhinav Shrivastava*
University of Maryland, College Park
abhinav@cs.umd.edu

Abstract

Batch normalization (BN) has been very effective for deep learning and is widely used. However, when training with small minibatches, models using BN exhibit a significant degradation in performance. In this paper we study this peculiar behavior of BN to gain a better understanding of the problem, and identify a cause. We propose 'EvalNorm' to address the issue by estimating corrected normalization statistics to use for BN during evaluation. EvalNorm supports online estimation of the corrected statistics while the model is being trained, and does not affect the training scheme of the model. As a result, EvalNorm can also be used with existing pre-trained models allowing them to benefit from our method. EvalNorm yields large gains for models trained with smaller batches. Our experiments show that EvalNorm performs 6.18% (absolute) better than vanilla BN for a batchsize of 2 on ImageNet validation set and from 1.5 to 7.0 points (absolute) gain on the COCO object detection benchmark across a variety of setups.

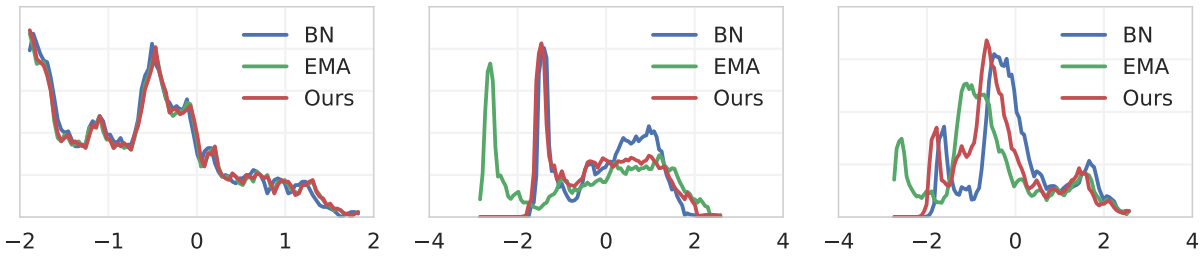
1. Introduction

Batch Normalization (BN) [11] has significantly contributed to the wide application and success of deep learning. It has enabled training of larger and deeper models [7, 8, 22, 23] leading to significant advances in computer vision. However, a well-acknowledged drawback of BN is that it requires sufficiently large minibatches [7, 10, 23], and results in drastically reduced model performance for smaller mini-batches. In this paper we study this peculiar behavior of BN to gain a better understanding of the source of the problem. We identify a cause and propose *EvalNorm* to address the issue. EvalNorm estimates corrected normalization statistics to use for BN during evaluation only. EvalNorm supports online estimation of the corrected statistics while the model is being trained and does not affect the training scheme of the model. As a result, EvalNorm can also be used with existing pre-trained models allowing them to benefit from our method without retraining. For pre-trained models EvalNorm suggests: 1) a rule of thumb that is trivial to implement and, 2) an offline estimation method that requires a pass through data (but doesn't update the model). The ability to estimate corrected statistics online for new models helps avoid an otherwise two step process. The model is still trained as it would be without our method. Note that proposing a new normalization technique is not a goal of this paper. Instead, the goal is to gain a better understanding of the behavior of BN for small minibatches. We limit ourselves to explorations that only affect the evaluation of a model trained with BN and refer to our method as 'Eval Normalization' (EvalNorm or EN).

Reliance of BN on large minibatches is prohibitive in several settings due to the hardware memory limitations. For example, applications requiring high-resolution inputs (object detection, segmentation, medical image analysis, etc.) or high-capacity (deeper and wider) networks are constrained to using smaller minibatches and thus take a performance hit. As a result, several normalization techniques, such as Group Normalization [25] and Batch Re-normalization [10], have been proposed to address the problem due to smaller minibatches. However, these approaches don't shed any light on the source of the problem with BN. Instead, they propose alternatives that make changes to the model and require retraining. As a result, many already trained and deployed models that use BN, where retraining is not possible, can not benefit from these alternatives. Our experimental evaluation of EN shows that it addresses the shortcomings of BN for small batches and provides a feasible alternative when retraining isn't an option.

Training and evaluation discrepancy in BN: During training, BN normalizes each channel for an example using the mean and variance of that channel aggregated across the full minibatch. This aggregation across the minibatch introduces a dependency on other minibatch samples. However, during evaluation each example is evaluated by itself and thus an approximation of the minibatch statistics is required. Typically, an exponential moving average (EMA) of minibatch moment statistics is maintained during training and used as a substitute during evaluation. Normalization using EMA

*Work done while at Google.



(a) 128 samples/minibatch (b) 2 samples/minibatch (c) 2 samples/minibatch

Figure 1: **Discrepancy in train vs. test distributions:** In each plot we show normalized activation distributions using different normalization operations and different normalization minibatch sizes for an arbitrary channel of a ResNet-20 trained on CIFAR-100. 'BN' denotes result of batch normalization during *training*, 'EMA' denotes normalization used during *evaluation* using EMA, and 'Ours' is using our EvalNorm adjustment during *evaluation*. Notice that for larger minibatches (128 samples) in (a), all three normalization operation result in similar distributions. However, for small minibatches (2 samples) in (b) and (c), 'Ours' using test time EN is much closer to the train time 'BN' than the standard 'EMA' statistics.

statistics is assumed to provide an accurate approximation to normalization observed during training. While reasonable for larger minibatches, we demonstrate that this assumption is erroneous for small minibatches (Section 3.2). This is because the mean and variance used to normalize a sample in a minibatch during training depend on that sample itself. For small minibatches this dependency is significant but ignored by the default method of using EMA. We qualitatively illustrate the discrepancy in train and test normalization in Figure 1, where we plot the distribution of normalized features at train ('BN') and test ('EMA') time. Each plot shows smoothed histograms of activations for an arbitrary channel of an arbitrary layer (in a ResNet-20 model) using different methods. For larger minibatches (Figure 1a), we observe that the distribution of normalized activations during training and testing match well. However, for small minibatches (Figures 1b and 1c), normalized feature distributions are quite different. Lastly, note that our method ('Ours') reduces this discrepancy and brings the distribution of normalized activations during testing closer to that during training ('BN').

To summarize our contributions, we: 1) quantify the dependence of the normalization statistics used by BN on a particular minibatch sample leading to an insight into the source of poor small minibatch performance, 2) propose two methods for estimating a correction without retraining existing models that use BN, and 3) propose a method to estimate corrections online during training of new models without affecting the training. We have included an extensive experimental section that analyzes and validates our insight and demonstrates that our insight leads to an improved evaluation performance of BN on a variety of common benchmarks.

2. Related work

Normalization of data for training is regularly used in machine learning. For example, it's a common practice to normalize features (scale or whiten) before learning classifiers like SVM. Similarly, for deep networks, many techniques have been proposed to normalize both inputs and intermediate representations, which make the training better and faster [6, 13]. Batch Normalization or BatchNorm (BN) is one such technique which aims to stabilize latent feature distributions in a deep network. BN normalizes features using the statistics computed from a minibatch; and has been shown to ease the learning problem and enable fast convergence of very deep network architectures. However, with BN's widespread adoption, it has been observed that models using BN exhibit severe degradation in performance when trained with smaller minibatches (as discussed in Section 1).

To address the stochasticity due to small minibatches and bias due to non-iid samples, Ioffe [10] introduced batch renormalization (Renorm) which constrains the minibatch moments to a specific range. This limits the variation in minibatch statistics during training. Ioffe [10] also introduced hyperparameters to prevent drift in the EMA statistics due to the variability of small minibatches. However, Renorm is still dependent on minibatch statistics with small minibatches, leading to worse performance. For tasks where small minibatches are standard (e.g., object detection), another approach is to engineer systems that can circumvent the issue. Peng et al. [17] proposed to perform synchronized computation of BN statistics across GPUs (Cross-GPU BN) to obtain better statistics. However, since images/GPU for object detection is small (1-2 images), this approach requires $\sim$128 GPUs to compute reasonable BN estimates. Further, this does not address the original problem of BN with small minibatches. Moreover, the need for synchronized computation prohibits the use of asynchronous training, which is a standard and practical tool for large-scale problems.

Instead of dealing with the small minibatch problem, several normalization techniques have been proposed that do not utilize the construct of a 'minibatch.' Instance Normalization [24] performs normalization similar to BN but only for a single sample and was shown to be effective on image style

transfer applications. Similarly, Layer Normalization [2] utilizes the entire layer (all channels) to estimate the normalization statistics. These approaches [2, 24] have not shown benefits on image recognition tasks, which is the application we focus on. Instead of normalizing the activations, Weight Normalization [20] reparameterizes the weights in the neural network to accelerate convergence. Normalization Propagation [1] uses data independent moment estimates in every layer, instead of computing them from minibatches during training. Group Normalization (GN) [25] divides the channels into groups and, within each group, computes the moments for normalization. GN alleviates the small minibatch problem to some extent, but it performs worse than BN for larger minibatches. Ren et al. [18] provide a unifying view of the different normalization approaches by characterizing them as the same transformation but along different dimensions (layers, samples, filters, etc.).

Unlike many approaches discussed above, the proposed EN does not modify the training scheme of batch normalization. It only estimates a different set of evaluation time statistics. The parameters used in EN are independent of the deep network, and training the parameters does not impact the network's training in any way. We conjecture that EN may be complementary to some of the above-mentioned approaches and may be used in conjunction with them to normalize across batch dimension. However, we consider this beyond the focus of this study.

3. Eval Normalization (EN)

We first briefly describe the relevant aspects of batch normalization and then present our method.

3.1. Batch Normalization (BN)

BN normalizes a particular channel of a sample in a minibatch using the mean and variance of that channel computed across the whole minibatch. Since the normalization of the channels is decoupled, we analyze the normalization of a single (but arbitrary) channel. Consider the set of activations $\mathcal{B} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_B}\}$ of a particular channel in an arbitrary layer for a minibatch of size B. Typically, $\boldsymbol{x_i}$ is a two dimensional field of scalars. During training, BN uses the minibatch mean $\mu_\mathcal{B}$ and variance $\sigma^2_\mathcal{B}$ to compute the normalized activations $\hat{\boldsymbol{x}}_\boldsymbol{i}$ as

$$\hat{\boldsymbol{x}}_\boldsymbol{i} = \frac{(\boldsymbol{x_i} - \mu_\mathcal{B})}{\sigma_\mathcal{B}}. \tag{1}$$

This has two notable ramifications during training: 1) For activations $\boldsymbol{x_i}$ the normalization statistics $\mu_\mathcal{B}$ and $\sigma^2_\mathcal{B}$ are a function of the statistics of activations $\boldsymbol{x_i}$ themselves, 2) the normalized activations $\hat{\boldsymbol{x}}_\boldsymbol{i}$ for a particular sample are stochastic as they depend on the statistics of the other samples in a stochastic minibatch.

During evaluation randomness of $\hat{\boldsymbol{x}}_\boldsymbol{i}$ due to stochastic dependency on other minibatch elements poses a difficulty for BN. To keep the evaluation deterministic and remove the dependency on other test samples BN substitutes $\hat{\boldsymbol{x}}_\boldsymbol{i}$ by an estimate of its expected value $\mathbb{E}[\hat{\boldsymbol{x}}_\boldsymbol{i}]$. This is done by treating $\mu_\mathcal{B}$ and $\sigma^2_\mathcal{B}$ encountered during training as random variables and substituting them by an estimate of their expected values as $\mu_\mathcal{E} \approx \mathbb{E}[\mu_\mathcal{B}]$ and $\sigma^2_\mathcal{E} \approx \mathbb{E}[\sigma^2_\mathcal{B}]$ to construct a first order approximation of $\mathbb{E}[\hat{\boldsymbol{x}}_\boldsymbol{i}]$ as

$$\mathbb{E}[\hat{\boldsymbol{x}}_\boldsymbol{i}] \approx \frac{(\boldsymbol{x_i} - \mathbb{E}[\mu_\mathcal{B}])}{\sqrt{\mathbb{E}[\sigma^2_\mathcal{B}]}} \approx \frac{(\boldsymbol{x_i} - \mu_\mathcal{E})}{\sqrt{\sigma^2_\mathcal{E}}}. \tag{2}$$

The estimates $\mu_\mathcal{E}$ and $\sigma^2_\mathcal{E}$ are typically maintained as exponential moving averages (EMA) during training.

Note that BN also employs a learned affine transform after normalization. We omit this affine transform in the text as it is not relevant to the discussion of normalization.

3.2. Source of stochasticity in $\hat{\boldsymbol{x}}_\boldsymbol{i}$

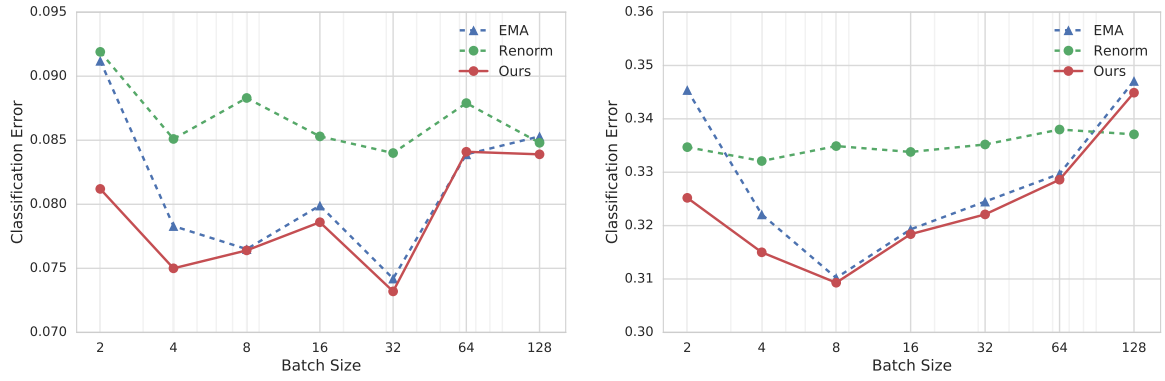
As noted earlier, $\mu_\mathcal{B}$ and $\sigma^2_\mathcal{B}$ are a function of $\boldsymbol{x_i}$. However, eq. (2) ignores this dependency entirely. Let us first make this dependency explicit. Let μ_i and σ^2_i be the mean and variance respectively of $\boldsymbol{x_i}$. Denote $\mathcal{C} = \mathcal{B} - \boldsymbol{x_i}$ as the set of $B-1$ mini-batch elements excluding $\boldsymbol{x_i}$ with combined mean $\mu_\mathcal{C}$ and variance $\sigma^2_\mathcal{C}$. The full minibatch mean $\mu_\mathcal{B}$ and variance $\sigma^2_\mathcal{B}$ can be expressed in terms of the above with the combined population mean and variance formulae. Let $\alpha = 1/B$, then

$$\mu_\mathcal{B} = \alpha\mu_i + (1-\alpha)\mu_\mathcal{C} \tag{3}$$

$$\sigma^2_\mathcal{B} = \alpha\sigma^2_i + (1-\alpha)\sigma^2_\mathcal{C} + \alpha(1-\alpha)(\mu_i - \mu_\mathcal{C})^2. \tag{4}$$

First note that $0 < \alpha \leqslant 1$, since $B \geqslant 1$. Therefore, α can be viewed as interpolating between the contributions from the sample being normalized and the other minibatch samples. For normalized activations $\hat{\boldsymbol{x}}_\boldsymbol{i}$ it is evident that the true source of stochasticity are $\mu_\mathcal{C}$ and $\sigma^2_\mathcal{C}$ due to randomized minibatch. Therefore, we assert that $\mathbb{E}[\hat{\boldsymbol{x}}_\boldsymbol{i}]$ in eq. (2) for the minibatch sample $\boldsymbol{x_i}$ should be approximated using eqs. (3) and (4) as opposed to approximating $\mu_\mathcal{B}$ and $\sigma^2_\mathcal{B}$ directly.

Note that for large minibatches $\alpha \approx 0$, resulting in a nominal contribution of μ_i and σ^2_i to the normalizing statistics (i.e., $\mathbb{E}[\mu_\mathcal{B}] \approx \mathbb{E}[\mu_\mathcal{C}]$ and $\mathbb{E}[\sigma^2_\mathcal{B}] \approx \mathbb{E}[\sigma^2_\mathcal{C}]$). Therefore, $\mathbb{E}[\mu_\mathcal{B}] \approx \mathbb{E}[\mu_\mathcal{C}] \approx \mu_\mathcal{E}$ and $\mathbb{E}[\sigma^2_\mathcal{B}] \approx \mathbb{E}[\sigma^2_\mathcal{C}] \approx \sigma^2_\mathcal{E}$ (that is, using the EMA statistics for normalization) are reasonable approximations. However, for small minibatches the contributions of μ_i and σ^2_i in eqs. (3) and (4) can not be ignored and these approximations are not accurate. We argue that this inaccuracy is the primary reason for observed distribution mis-matches between BN and EMA in Figure 1, and poor performance of BN for small minibatches.



(a) CIFAR-10 (b) CIFAR-100

Figure 2: **Classification error on CIFAR-10 and CIFAR-100**. EN consistently outperforms both EMA and Renorm for all minibatch sizes. Notice that the error does not decrease monotonically with increasing minibatch sizes. One intuition behind this is that at the inflection point, the stochasticity of the estimates act as a good regularizer (similar trends were observed by Masters and Luschi [16]).

3.3. Approximating $\mathbb{E}[\hat{x}_i]$ for small minibatches

Simple approximation: At first glance a simple solution would be to directly use $\alpha = 1/B$ and approximate $\mathbb{E}[\mu_\mathcal{B}]$ and $\mathbb{E}[\sigma^2_\mathcal{B}]$ using eqs. (3) and (4). More specifically, $\mathbb{E}[\mu_\mathcal{B}] \approx \alpha\mu_i + (1-\alpha)\mu_\mathcal{E}$ and $\mathbb{E}[\sigma^2_\mathcal{B}] \approx \alpha\sigma^2_i + (1-\alpha)\sigma^2_\mathcal{E} + \alpha(1-\alpha)(\mu_i - \mu_\mathcal{E})^2$. These can then be substituted in eq. (2) for normalization. Note that we have made the additional approximations of $\mathbb{E}[\mu_\mathcal{C}] \approx \mu_\mathcal{E}, \mathbb{E}[\sigma^2_\mathcal{C}] \approx \sigma^2_\mathcal{E}$. However, for small minibatches $\mu_\mathcal{C}$ and $\sigma^2_\mathcal{C}$ exhibit high variance and these approximations may be inaccurate. Unfortunately, higher order approximations to account for the variance either require tracking higher order moments, whose estimates would themselves be unreliable, or making strong assumptions about the distributions of $\boldsymbol{x_i}$. Nevertheless, we explore this alternative in experiments where it turns out that $\alpha = 1/B$ is less than ideal. Experimenting with a few heuristic choices for the value of α leads to a simple *rule-of-thumb* of $\alpha = 1/B^2$ (Figure 4 and Table 1) that is found to work well empirically.

Estimated approximation: Instead of using a fixed α, we propose to turn α into an estimated variable. We instantiate two separate copies of α, namely $\hat{\alpha}$ and $\hat{\beta}$, and construct the following approximations to eqs. (3) and (4)

$$\mathbb{E}[\mu_\mathcal{B}] \approx \hat{\alpha}\mu_i + (1-\hat{\alpha})\mu_\mathcal{E} \tag{5}$$

$$\mathbb{E}[\sigma^2_\mathcal{B}] \approx \hat{\beta}\sigma^2_i + (1-\hat{\beta})\sigma^2_\mathcal{E} + \hat{\beta}(1-\hat{\beta})(\mu_i - \mu_\mathcal{E})^2. \tag{6}$$

Our method estimates $\hat{\alpha}$ and $\hat{\beta}$ (Section 3.4) such that normalized activations using above approximations match the normalized activations using minibatch statistics. Note that, decoupled $\hat{\alpha}$ and $\hat{\beta}$ lead to slightly better empirical results than a single value.

3.4. Estimating $\hat{\alpha}$ and $\hat{\beta}$

Offline estimation: Given a trained model, we propose to estimate $\hat{\alpha}$ and $\hat{\beta}$ for a particular layer by minimizing an auxiliary loss $\mathcal{L}_\text{aux}$ as below. Let $\langle\cdot\rangle$ represent a `stop_gradient` operation that does not allow flow of gradients to its argument in automatic differentiation frameworks. Then $\mathcal{L}_\text{aux}$ is computed as

$$\hat{\mu} = \hat{\alpha}\langle\mu_i\rangle + (1-\hat{\alpha})\mu_\mathcal{E} \tag{7}$$

$$\hat{\sigma}^2 = \hat{\beta}\langle\sigma_i\rangle^2 + (1-\hat{\beta})\sigma^2_\mathcal{E} + \hat{\beta}(1-\hat{\beta})(\langle\mu_i\rangle - \mu_\mathcal{E})^2 \tag{8}$$

$$\mathcal{L}_\text{aux} = \left\| \left\langle \frac{\boldsymbol{x_i} - \mu_\mathcal{B}}{\sigma_\mathcal{B}} \right\rangle - \frac{\langle \boldsymbol{x_i} \rangle - \hat{\mu}}{\hat{\sigma}} \right\|_1 \tag{9}$$

Minimizing this objective allows us to estimate $\hat{\alpha}, \hat{\beta}$ such that evaluation time normalization produces activations that are similar to those observed using minibatch statistics for normalization. Further, the `stop_gradient` operation prevents the estimation from affecting the model parameters.

Online estimation: For training a new model a naïve approach would be to first train the model as usual with BN and then estimate $\hat{\alpha}$ and $\hat{\beta}$ in a second pass while freezing rest of the parameters using $\mathcal{L}_\text{aux}$ above. However, use of `stop_gradient` as above allows us to collapse the two steps into a single one without affecting the training of the model parameters.

4. Experiments

We evaluate our approach on two tasks and four datasets: image classification on CIFAR-10, CIFAR-100 [12], and ImageNet [3], and object detection on COCO [14]. We use CIFAR-100 as a test-bed to perform ablation experiments and study various aspects of our method. The results on ImageNet and COCO demonstrate the utility of using EvalNorm

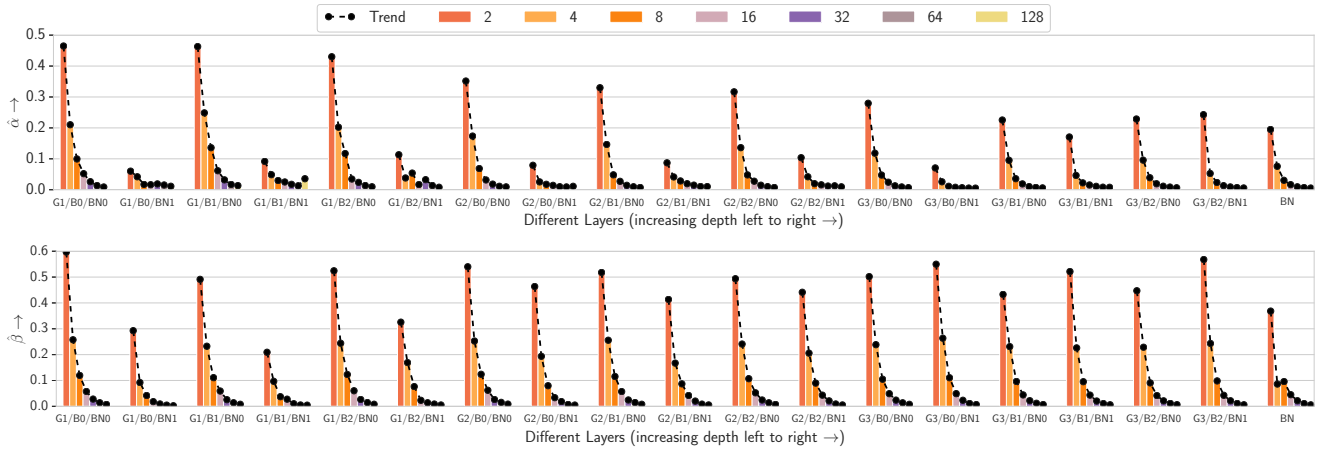


Figure 3: Estimated $\hat{\alpha}$ (top) and $\hat{\beta}$ (bottom) values for different batchsizes on CIFAR-100. Each set of bars corresponds to a BN operation (e.g. G1/B0 represents group 1, block 0; BN0 and BN1 represent first and second BN operation) and each color corresponds to a batch size (increasing left to right in a set). Both $\hat{\alpha}$ and $\hat{\beta}$ take larger values for smaller batchsizes indicating that individual statistics need to be included in the normalization statistics and weighted in accordance with the batchsize.

(EN), especially for models trained with small mini-batches. Note that, unless explicitly stated, $\hat{\alpha}$ and $\hat{\beta}$ in all the experiments are estimated online for ease of experimentation.

4.1. Image Classification on CIFAR-10/100

Experimental setup. CIFAR-10 and CIFAR-100 contain images from 10 and 100 classes respectively. For both CIFAR-10 and CIFAR-100, we train on the 50000 training images and evaluate on the 10000 test images. Unless specified otherwise, we use a 20 layer ResNetv2 CIFAR variant [8] for both datasets. The base CIFAR variant contains three groups of residual blocks with widths $\{16, 32, 64\}$ respectively. Wider variants use multiples of these sizes. All networks are trained using SGD with momentum for 128k updates, with an initial learning rate of 0.1 and a cosine decay schedule [15] unless otherwise specified.

Small minibatches. In this section we study the effect of minibatch size used to compute the normalization statistics. For fair comparison all models need to be trained for the same number of epochs. However, this leads to smaller minibatch models getting more gradient updates. Moreover, gradients from small minibatches have higher variance in comparison to larger minibatches requiring an adjustment in learning rate. To isolate the impact of small minibatches on normalization from these confounding factors, we use the same minibatch size (128 samples) for gradient updates and vary the number of samples used for normalization (from 2 to 128) (refer to the "microbatch" setup in [10]).

Figures 2a and 2b compare our method (EN) with the default EMA statistics used by standard BN [11] and batch re-normalization (Renorm) [10]. For Renorm, we set $r_{\max} = 2.0$ and $d_{\max} = 1.0$. For larger minibatch sizes (64 and 128),

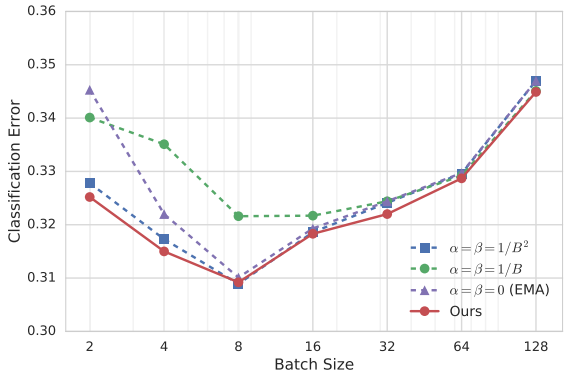
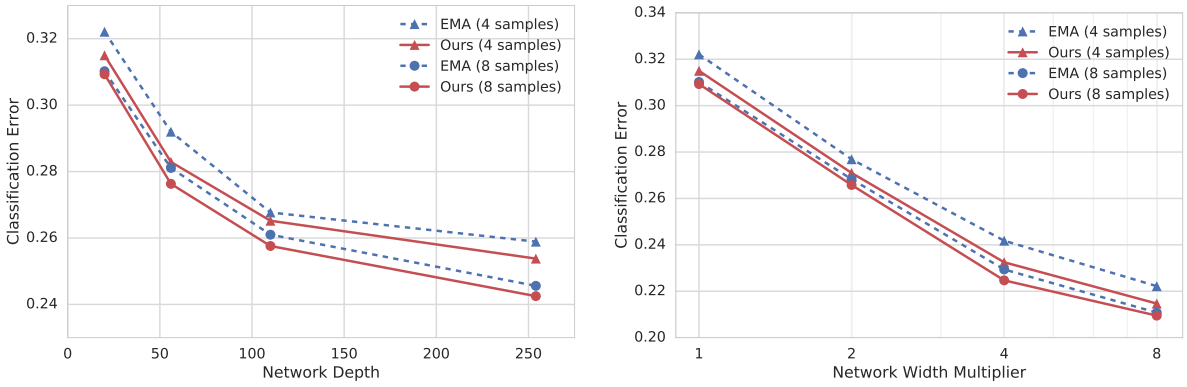


Figure 4: Comparison of a set of hand picked schedules for $\hat{\alpha}, \hat{\beta}$ as a function of batch size B. Although, the estimated values using our method perform the best, $\hat{\alpha} = \hat{\beta} = 1/B^2$ appears to be a good rule of thumb.

we notice that all methods are comparable, but for smaller minibatch sizes (2 and 4) using EN statistics leads to a lower classification error compared to both BN and Renorm. Even though Renorm is less sensitive to varying minibatch sizes, it also lead to worse performance.

Interestingly, the performance of BN and EN does not decrease monotonically with the decreasing minibatch size used for normalization. Instead, it seems to improve before becoming worse. Similar trends were reported by [16]. In depth study of this is beyond the scope of this paper (refer to [16] for an empirical study). One intuition is that smaller minibatches introduce stochasticity that acts as a regularizer before it starts to hurt performance.

Visualization of estimated $\hat{\alpha}, \hat{\beta}$. Figure 3 visualizes the estimated $\hat{\alpha}, \hat{\beta}$ across different layers for various batch sizes.



(a) Depth (b) Width

Figure 5: Performance of EN for deeper and wider network variants on CIFAR-100. EN consistently leads to lower classification error compared to EMA statistics.

Larger value of $\hat{\alpha}, \hat{\beta}$ for smaller batches confirms our insight that statistics of the sample being normalized need to be included and weighted in accordance with batchsize. Both $\hat{\alpha}, \hat{\beta}$ follow the trend of taking on smaller values for larger batches indicating a decreasing influence of individual statistics for larger batches in agreement with eqs. (3) and (4).

Simple approximation for $\hat{\alpha}, \hat{\beta}$. Figure 4 compares a set of hand picked schedules for $\hat{\alpha}, \hat{\beta}$ as a function of batch size B. Although, the estimated values using our method perform the best, $\hat{\alpha} = \hat{\beta} = 1/B^2$ appears to be a good rule of thumb. Note that this is counter intuitive to the more natural seeming $\hat{\alpha} = \hat{\beta} = 1/B$. We also validate this on ImageNet in Table 1.

Deeper and wider networks. Figure 5 compares EN and EMA in deeper (20, 56, 110, and 254 layers) and wider (1, 2, 4, and 8 $\times$ the original width) ResNetv2 models, with normalization minibatch size of 4 and 8 on CIFAR-100. EN consistently leads to lower classification error compared to using EMA statistics indicating its general applicability.

4.2. Image Classification on ImageNet

We report results on the ImageNet classification dataset [3] with 1000 classes. We train on $\sim$1.28M training images and evaluate on 50k validation images. All methods in this section use a 50 layer ResNetv2 [8] network architecture. We resize all images to 229×229 and use training time data augmentation from [23].

As is standard practice [7, 8], we use synchronous SGD across 8 GPUs to train all models in this section. We compute the BN and EN statistics per GPU, whereas gradients are averaged across all GPUs. Therefore, the effective minibatch size for gradient computation (SGD 'batchsize') is $8\times$ the per GPU normalization minibatch size ('samples/GPU'). This is the standard synchronized multi-GPU training setup in popular libraries, such as PyTorch and Tensorflow.

We study normalization minibatch sizes (samples/GPU) of 2, 4, 8, 16, and 32, leading to an effective SGD batchsize (N) of 16, 32, 64, 128, and 256 respectively. All models are trained for 60 epochs with an initial learning rate of $0.1 \times N/256$ and a cosine decay schedule. Other implementation details (such as initialization) follows [8]. We report results on two image classification metrics: 'Prec@1' and 'Recall@5'. Prec@1 measures the classification accuracy of the top-1 prediction, and Recall@5 measures the recall accuracy for top-5 predictions.

We evaluate using EN and EMA statistics for inference time normalization and report the results in Table 1. We observe that using EN statistics *consistently* perform better than EMA statistics across all minibatch sizes. For larger minibatches, where EMA statistics provide an accurate approximation, the difference in performance is marginal. However, for small minibatches EN statistics have a significant impact. For example, for 2 samples/GPU, Prec@1 metric improves by more than 8 points and Recall@5 metric improves by 6.27 points. Also note that the performance gap between small and large minibatch size is much lower with EN. For example, reducing samples/GPU from 32 to 4, Prec@1 for EMA drops from 75.98 to 71.40 (-4.58 points), whereas for EN, it only drops by 2.7 points.

Next, in Table 1 we also compare our method with two recent normalization methods namely Group normalization [25] and Batch renormalization [10] that circumvent minibatch dependence. Note that these methods change the model or the training method itself. EN performs similar to or better than alternatives that normalize across batch (EMA and Renorm), indicating that it properly accounts for individual sample contributions. For large batch sizes EN outperforms other methods while for small batch sizes Group normalization tends to perform better. However, the gain from retraining using GroupNorm is reduced from 10% to 3.75% (for batchsize 2), and EN does not suffer the side

Table 1: **ImageNet classification results** for ResNet-50 [8] using EMA and EN statistics for different normalization minibatch sizes (samples/GPU). We show the classification accuracy for top-1 prediction (Prec@1) and the recall for top-5 prediction (Recall@5). Δ represents improvement due to EN over EMA. EN consistently performs better than EMA across all minibatch sizes, and provides significant gains for small minibatches. See section 4.2 for details.

	samples/GPU →	32	16	8	4	2
Prec@1	Renorm	76.04	75.78	74.21	73.33	70.87
	Groupnorm	75.87	75.73	75.75	75.61	74.78
	EMA	75.98	75.26	73.68	71.40	64.80
	EN [**ours**]	76.20	75.89	74.87	73.49	70.98
	Δ	+0.22	+0.63	+1.19	+2.09	**+6.18**
	$\alpha = \beta = 1/B^2$ [**ours**]	76.28	75.77	74.52	73.17	70.12
	EN-Offline [**ours**]	76.18	75.85	74.92	73.48	71.03
Recall@5	Renorm	92.81	92.65	92.18	91.68	90.03
	Groupnorm	92.59	92.31	92.28	92.10	91.81
	EMA	92.88	92.61	92.17	90.70	86.13
	EN [**ours**]	92.97	92.78	92.38	91.75	90.18
	Δ	+0.09	+0.17	+0.21	+1.05	**+4.05**
	$\alpha = \beta = 1/B^2$ [**ours**]	92.96	92.81	92.17	91.46	89.51
	EN-Offline [**ours**]	92.93	92.79	92.30	91.81	90.25

effect of reduced performance for large batch sizes.

Table 1 also reports the performance using the suggested rule-of-thumb ($\alpha = \beta = 1/B^2$) as well as offline estimation. Rule-of-thumb significantly improves over EMA but underperforms EN. Nevertheless, it is an attractive and easier alternative to estimation at the cost of performance. Offline estimation performs similar to online estimation as expected.

4.3. Object Detection on COCO

Finally, we evaluate our method on the task of object detection, and demonstrate consistent improvements when using EN statistics as opposed to EMA. Object detection frameworks are typically trained with high resolution inputs, and hence use small SGD minibatch sizes. As a result object detection is a good benchmark to evaluate the differences between EN and EMA.

Experimental setup. All experiments in this section use the COCO dataset [14] with 80 object classes. We train using the trainval35k set [4] with ∼75k images and evaluate on a held out minival set [4] with 5k images. We report the standard COCO evaluation metrics for mean average prevision with varying IoU thresholds (AP, AP^{50}, AP^{75}) (see [14] for details).

We use the Faster R-CNN (FRCN) [9, 19] object detection framework, built with a 50 layer ResNetv1 [7] (ResNet-50). The FRCN system has three components: 1) backbone feature extractor (base network), which operates on high resolution images (we resize images to 600×600 for all experiments), 2) region proposal network (RPN), which proposes regions of interest (ROIs), and 3) region classification network (RCN), which classifies regions proposed by RPN using cropped features from the base network. All but the last residual blocks from ResNet-50 are used as the base network and the last residual block is used in the RCN network. Refer to [9, 19, 21] for architecture details.

We study the SGD minibatch sizes (images/GPU or N) of 2, 4, and 8. As is standard practice [9], we use minibatch size of 300 and $64N$ ROIs for the RPN and RCN respectively. Note that this results in different normalization minibatches for different BN layers in the network (N for base network and $64N$ for RCN). All models are trained for 64 epochs (in terms of images and not ROIs) with an initial learning rate of $0.015 \times N/8$ and a cosine decay schedule. All methods use asynchronous SGD with 11 parallel GPU workers. We use the publicly released code from [9].

We investigate two different training paradigms: training from a random initialization and transfer learning (or fine-tuning). The random initialization setup is similar to the experiments reported on image classification in Sections 4.1 and 4.2. The transfer learning setup is more common in practice because it generally leads to higher performance. Next, we discuss the trade-offs and training flexibility in each paradigm and report results in Table 2 (blocks (a)-(d)).

Random initialization. We study the impact of EN by training a ResNet-50 Faster R-CNN model with all parameters initialized randomly. The results for using both EMA and EN statistics are reported Table 2(a). We observe that when using 4 and 8 images/minibatch, both methods are on-par; but when training with 2 images/minibatch, we see consistent and significant gains of more than 1.5 points on all AP metrics. Note that in this setup, the SGD and normalization minibatch sizes for the base network are same; unlike the ImageNet setup, where we average gradients across 8 GPUs resulting in SGD minibatch size being 8× the normalization minibatch size.

Transfer learning. The standard paradigm of training object detection models (like Faster R-CNN) is to use transfer learning or fine-tuning [4, 5, 9, 19]. In this setup, parameters from a model trained on ImageNet classification are transferred to the detection model, which is then trained on object detection. We use the model checkpoint from [7] as is standard practice. The transfer learning paradigm allows us to study the impact of EN statistics compared to EMA in a variety of settings.

First, we investigate which method is able to better estimate BN statistics in the absence of a prior. For this, we use the ImageNet trained model to initialize the network parameters (except BN parameters), but we use initial values of BN parameters from [11]; and both sets of parameters

Table 2: **Object detection results on COCO** using Faster R-CNN [19] with ResNet50 [7]. We report EMA and EN performance for different minibatch sizes (images/batch). EN performs better across all AP metrics and all minibatch sizes, specially for 2 images/minibatch, where we see significant gains. Different blocks (a)-(d) are described in Section 4.3. ($\Delta > 1$ point are shown in bold)

	Network Weights		BN EMA Stats			AP			AP^{50}			AP^{75}		
	Init	Train?	Init	Train?	images/batch →	8	4	2	8	4	2	8	4	2
(a)	Random	✓	Random	✓	EMA	25.2	25.6	22.0	40.9	41.2	36.1	26.7	27.5	22.4
					EN [**ours**]	25.2	25.8	23.5	41.0	41.3	38.1	26.8	27.6	25.0
					Δ	0.0	+0.1	**+1.5**	+0.1	+0.1	**+2.0**	+0.1	+0.1	**+2.6**
(b)	ImageNet	✓	Random	✓	EMA	30.4	29.9	27.6	48.4	47.2	43.5	32.8	31.9	29.6
					EN [**ours**]	30.7	30.5	29.6	48.5	47.9	46.5	33.0	32.8	31.3
					Δ	+0.2	+0.6	**+2.0**	+0.1	+0.7	**+3.1**	+0.2	+0.9	**+1.7**
(c)	ImageNet	✓	ImageNet	×	EMA	27.5	28.8	28.2	45.1	47.3	46.6	28.9	30.5	30.5
					EN [**ours**]	30.2	30.3	30.5	48.2	48.6	48.7	32.2	32.1	32.7
					Δ	**+2.7**	**+1.5**	**+2.3**	**+3.1**	**+1.3**	**+2.1**	**+3.3**	**+1.6**	**+2.2**
(d)	ImageNet	✓	ImageNet	✓	EMA	31.7	30.1	24.8	50.6	47.6	40.0	33.9	32.2	26.1
					EN [**ours**]	31.9	31.6	30.2	50.7	49.6	46.9	34.1	34.0	31.0
					Δ	+0.2	**+1.5**	**+5.4**	+0.2	**+2.1**	**+7.0**	+0.2	**+1.8**	**+4.9**

are trained simultaneously. We report the results for EN and EMA in Table 2(b). We notice that EN is consistently better than EMA across all minibatch sizes and AP metrics. In fact, as opposed to random initialization, we see gains for both 2 and 4 images/minibatch, with the improvements for the smaller minibatch being much higher.

Next, we study the standard setup [9, 19] of initializing both, the network and the BN EMA parameters, using the ImageNet model. Since standard BN performs poorly with smaller minibatches [7, 10, 25], the BN parameters are not updated during training in this setup. The results for this setup are reported in Table 2(c). Notice that EN performs significantly better than EMA across all minibatch sizes and AP metrics. EN allows adjustment of statistics by using the learned features (as opposed to 'stale' features from initialization). Since this is the standard training paradigm used by almost all detection systems, these consistent and significant improvements are doubly important.

Finally, to demonstrate the effectiveness of EN in adjusting statistics, we initialize both network and BN parameters from ImageNet, and fine-tune **both** for object detection. In practice, this setup performs poorly because BN is unable to train for small minibatches, and is generally not used (for example, [25] ignore this variant because of significant drop in performance). We also notice this in Table 2(c) and (d), where the EMA performance for 2 images/minibatch drops by 3.4 AP, 6.6 AP^{50}, and 4.4 AP^{75}. Because of these drastic drops in performance, methods do not fine-tune BN parameters. Compare this to using the proposed EN statistics, which improves over EMA by **5.4** AP, **7.0** AP^{50}, and **4.9** AP^{75}. Therefore, using EN allows us to train BN parameters for smaller minibatches yielding significant improvements.

In this section, we showed that for the object detection task, which is generally trained with small minibatches, using EN statistics performs markedly better across all training setups. In Table 2, notice that we get a healthy performance improvement for 2 images/minibatch across training setups (ranging from **+1.5** points to **+7** points). In the standard paradigm (Table 2(c)), we improve for all minibatch sizes. Moreover, when training BN parameters for small minibatches, (Table 2(c) and (d)), we improve both 2 and 4 images/minibatch.

5. Conclusion

Our goal in this work was to gain a better understanding of the problem with BN for small minibatches. We found that for models trained with small minibatches, normalization using EMA statistics during evaluation provides inaccurate approximation for normalization using minibatch statistics during training. This leads to a discrepancy between training and evaluation and is the main reason of performance degradation of BN for small batch sizes. We proposed EvalNorm, which provides a corrected normalization term for use at evaluation. EN is fully compatible with the existing pre-trained models using BN and yields large gains for models trained with smaller batches.

Acknowledgement. We would like to thank Chen Sun, Sergey Ioffe, and Rahul Sukthankar for helpful discussions and comments; and Ishan Misra and Larry Davis for feedback on the draft.

References

[1] Devansh Arpit, Yingbo Zhou, Bhargava U Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. *arXiv preprint arXiv:1603.01431*, 2016. 3

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009. 4, 6

[4] Ross Girshick. Fast R-CNN. *Computer Vision and Pattern Recognition*, 2015. 7

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection a nd semantic segmentation. In *Computer Vision and Pattern Recognition*, pages 580–587. IEEE, 2014. 7

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer vision and pattern recognition*, pages 770–778, 2016. 1, 6, 7, 8

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 1, 5, 6, 7

[9] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *Computer Vision and Pattern Recognition*, 2017. 7, 8

[10] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in Neural Information Processing Systems*, pages 1942–1950, 2017. 1, 2, 5, 6, 8

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 1, 5, 7

[12] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10/100 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014. 4

[13] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998. 2

[14] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 4, 7

[15] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[16] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018. 4, 5

[17] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Computer Vision and Pattern Recognition*, pages 6181–6189, 2018. 2

[18] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H Sinz, and Richard S Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. *arXiv preprint arXiv:1611.04520*, 2016. 3

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015. 7, 8

[20] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016. 3

[21] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Computer Vision and Pattern Recognition*, pages 761–769, 2016. 7

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer vision and pattern recognition*, pages 1–9, 2015. 1

[23] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017. 1, 6

[24] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2, 3

[25] Yuxin Wu and Kaiming He. Group normalization. In *Computer Vision and Pattern Recognition*, 2018. 1, 3, 6, 8