Train longer, generalize better: closing the generalization gap in large batch training of neural networks

Elad Hoffer[*] **Itay Hubara**[*] **Daniel Soudry**
Technion - Israel Institute of Technology, Haifa, Israel
`{elad.hoffer, itayhubara, daniel.soudry}@gmail.com`

Abstract

Background: Deep learning models are typically trained using stochastic gradient descent or one of its variants. These methods update the weights using their gradient, estimated from a small fraction of the training data. It has been observed that when using large batch sizes there is a persistent degradation in generalization performance - known as the "generalization gap" phenomenon. Identifying the origin of this gap and closing it had remained an open problem.

Contributions: We examine the initial high learning rate training phase. We find that the weight distance from its initialization grows logarithmically with the number of weight updates. We therefore propose a "random walk on a random landscape" statistical model which is known to exhibit similar "ultra-slow" diffusion behavior. Following this hypothesis we conducted experiments to show empirically that the "generalization gap" stems from the relatively small number of updates rather than the batch size, and can be completely eliminated by adapting the training regime used. We further investigate different techniques to train models in the large-batch regime and present a novel algorithm named "Ghost Batch Normalization" which enables significant decrease in the generalization gap without increasing the number of updates. To validate our findings we conduct several additional experiments on MNIST, CIFAR-10, CIFAR-100 and ImageNet. Finally, we reassess common practices and beliefs concerning training of deep models and suggest they may not be optimal to achieve good generalization.

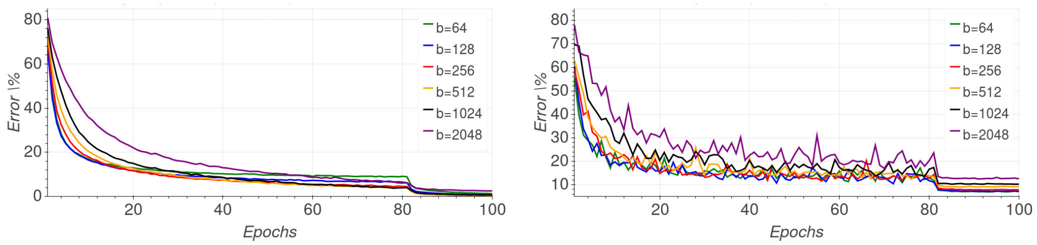
1 Introduction

For quite a few years, deep neural networks (DNNs) have persistently enabled significant improvements in many application domains, such as object recognition from images (He et al., 2016); speech recognition (Amodei et al., 2015); natural language processing (Luong et al., 2015) and computer games control using reinforcement learning (Silver et al., 2016; Mnih et al., 2015).

The optimization method of choice for training highly complex and non-convex DNNs, is typically stochastic gradient decent (SGD) or some variant of it. Since SGD, at best, finds a local minimum of the non-convex objective function, substantial research efforts are invested to explain DNNs ground breaking results. It has been argued that saddle-points can be avoided (Ge et al., 2015) and that "bad" local minima in the training error vanish exponentially (Dauphin et al., 2014; Choromanska et al., 2015; Soudry & Hoffer, 2017). However, it is still unclear why these complex models tend to generalize well to unseen data despite being heavily over-parameterized (Zhang et al., 2017).

A specific aspect of generalization has recently attracted much interest. Keskar et al. (2017) focused on a long observed phenomenon (LeCun et al., 1998a) – that when a large batch size is used while training DNNs, the trained models appear to generalize less well. This remained true even when the

[*] Equal contribution



(a) Training error

(b) Validation error

Figure 1: Impact of batch size on classification error

models were trained "without any budget or limits, until the loss function ceased to improve" (Keskar et al., 2017). This decrease in performance has been named the "generalization gap".

Understanding the origin of the generalization gap, and moreover, finding ways to decrease it, may have a significant practical importance. Training with large batch size immediately increases parallelization, thus has the potential to decrease learning time. Many efforts have been made to parallelize SGD for Deep Learning (Dean et al., 2012; Das et al., 2016; Zhang et al., 2015), yet the speed-ups and scale-out are still limited by the batch size.

In this study we suggest a first attempt to tackle this issue.
First,

- We propose that the initial learning phase can be described using a high-dimensional "random walk on a random potential" process, with an "ultra-slow" logarithmic increase in the distance of the weights from their initialization, as we observe empirically.

Inspired by this hypothesis, we find that

- By simply adjusting the learning rate and batch normalization the generalization gap can be significantly decreased (for example, from 5% to $1\% - 2\%$).
- In contrast to common practices (Montavon et al., 2012) and theoretical recommendations (Hardt et al., 2016), generalization keeps improving for a long time at the initial high learning rate, even without any observable changes in training or validation errors. However, this improvement seems to be related to the distance of the weights from their initialization.
- There is no inherent "generalization gap": large-batch training can generalize as well as small batch training by adapting the number of iterations.

2 Training with a large batch

Training method. A common practice of training deep neural networks is to follow an optimization "regime" in which the objective is minimized using gradient steps with a fixed learning rate and a momentum term (Sutskever et al., 2013). The learning rate is annealed over time, usually with an exponential decrease every few epochs of training data. An alternative to this regime is to use an adaptive per-parameter learning method such as Adam (Kingma & Ba, 2014), Rmsprop (Dauphin et al.) or Adagrad (Duchi et al., 2011). These methods are known to benefit the convergence rate of SGD based optimization. Yet, many current studies still use simple variants of SGD (Ruder, 2016) for all or part of the optimization process (Wu et al., 2016), due to the tendency of these methods to converge to a lower test error and better generalization.

Thus, we focused on momentum SGD, with a fixed learning rate that decreases exponentially every few epochs, similarly to the regime employed by He et al. (2016). The convergence of SGD is also known to be affected by the batch size (Li et al., 2014), but in this work we will focus on generalization. Most of our results were conducted on the Resnet44 topology, introduced by He et al. (2016). We strengthen our findings with additional empirical results in section 6.

Empirical observations of previous work. Previous work by Keskar et al. (2017) studied the performance and properties of models which were trained with relatively large batches and reported the following observations:

- Training models with large batch size increase the generalization error (see Figure 1).
- This "generalization gap" seemed to remain even when the models were trained without limits, until the loss function ceased to improve.
- Low generalization was correlated with "sharp" minima[2] (strong positive curvature), while good generalization was correlated with "flat" minima (weak positive curvature).
- Small-batch regimes were briefly noted to produce weights that are farther away from the initial point, in comparison with the weights produced in a large-batch regime.

Their hypothesis was that a large estimation noise (originated by the use of mini-batch rather than full batch) in small mini-batches encourages the weights to exit out of the basins of attraction of sharp minima, and towards flatter minima which have better generalization.In the next section we provide an analysis that suggest a somewhat different explanation.

3 Theoretical analysis

Notation. In this paper we examine Stochastic Gradient Descent (SGD) based training of a Deep Neural Network (DNN). The DNN is trained on a finite training set of N samples. We define $\mathbf{w}$ as the vector of the neural network parameters, and $L_n\left(\mathbf{w}\right)$ as loss function on sample n. We find $\mathbf{w}$ by minimizing the training loss.

$$L\left(\mathbf{w}\right) \triangleq \frac{1}{N}\sum_{n=1}^{N} L_n\left(\mathbf{w}\right) \ ,$$

using SGD. Minimizing $L\left(\mathbf{w}\right)$ requires an estimate of the gradient of the negative loss.

$$\mathbf{g} \triangleq \frac{1}{N}\sum_{n=1}^{N} \mathbf{g}_n \triangleq -\frac{1}{N}\sum_{n=1}^{N} \nabla L_n\left(\mathbf{w}\right)$$

where $\mathbf{g}$ is the true gradient, and $\mathbf{g}_n$ is the per-sample gradient. During training we increment the parameter vector $\mathbf{w}$ using only the mean gradient $\hat{\mathbf{g}}$ computed on some mini-batch B – a set of M randomly selected sample indices.

$$\hat{\mathbf{g}} \triangleq \frac{1}{M}\sum_{n\in B} \mathbf{g}_n \ .$$

In order to gain a better insight into the optimization process and the empirical results, we first examine simple SGD training, in which the weights at update step t are incremented according to the mini-batch gradient $\Delta\mathbf{w}_t = \eta\hat{\mathbf{g}}_t$. With respect to the randomness of SGD,

$$\mathbb{E}\hat{\mathbf{g}}_t = \mathbf{g} = -\nabla L\left(\mathbf{w}_t\right),$$

and the increments are uncorrelated between different mini-batches[3]. For physical intuition, one can think of the weight vector $\mathbf{w}_t$ as a particle performing a random walk on the loss ("potential") landscape $L\left(\mathbf{w}_t\right)$. Thus, for example, adding momentum term to the increment is similar to adding inertia to the particle.

Motivation. In complex systems (such as DNNs) where we do not know the exact shape of the loss, statistical physics models commonly assume a simpler description of the potential as a random process. For example, Dauphin et al. (2014) explained the observation that local minima tend to have

[2] It was later pointed out (Dinh et al., 2017) that certain "degenerate" directions, in which the parameters can be changed without affecting the loss, must be excluded from this explanation. For example, for any $c > 0$ and any neuron, we can multiply all input weights by c and divide the output weights by c: this does not affect the loss, but can generate arbitrarily strong positive curvature.

[3] Either exactly (with replacement) or approximately (without replacement): see appendix section A.

low error using an analogy between $L(\mathbf{w})$, the DNN loss surface, and the high-dimensional Gaussian random field analyzed in Bray & Dean (2007), which has zero mean and auto-covariance

$$\mathbb{E}\left(L\left(\mathbf{w}_1\right)L\left(\mathbf{w}_2\right)\right) = f\left(\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|^2\right) \tag{1}$$

for some function f, where the expectation now is over the randomness of the loss. This analogy resulted with the hypothesis that in DNNs, local minima with high loss are indeed exponentially vanishing, as in Bray & Dean (2007). Only recently, similar results are starting to be proved for realistic neural network models (Soudry & Hoffer, 2017). Thus, a similar statistical model of the loss might also give useful insights for our empirical observations.

Model: Random walk on a random potential. Fortunately, the high dimensional case of a particle doing a "random walk on a random potential" was extensively investigated already decades ago (Bouchaud & Georges, 1990). The main result of that investigation was that the asymptotic behavior of the auto-covariance of a random potential[4],

$$\mathbb{E}\left(L\left(\mathbf{w}_1\right)L\left(\mathbf{w}_2\right)\right) \sim \left\|\mathbf{w}_1 - \mathbf{w}_2\right\|^{\alpha} \ \ ,\ \alpha > 0 \tag{2}$$

in a certain range, determines the asymptotic behavior of the random walker in that range:

$$\mathbb{E}\left\|\mathbf{w}_t - \mathbf{w}_0\right\|^2 \sim \left(\log t\right)^{\frac{4}{\alpha}} \ . \tag{3}$$

This is called an "ultra-slow diffusion" in which, typically $\left\|\mathbf{w}_t - \mathbf{w}_0\right\| \sim \left(\log t\right)^{2/\alpha}$, in contrast to standard diffusion (on a flat potential), in which we have $\left\|\mathbf{w}_t - \mathbf{w}_0\right\| \sim \sqrt{t}$. The informal reason for this behavior (for any $\alpha > 0$), is that for a particle to move a distance d, it has to pass potential barriers of height $\sim d^{\alpha/2}$, from eq. (2). Then, to climb (or go around) each barrier takes exponentially long time in the height of the barrier: $t \sim \exp(d^{\alpha/2})$. Inverting this relation, we get eq. $d \sim (\log(t))^{2/\alpha}$. In the high-dimensional case, this type of behavior was first shown numerically and explained heuristically by Marinari et al. (1983), then rigorously proven for the case of a discrete lattice by Durrett (1986), and explained in the continuous case by Bouchaud & Comtet (1987).

3.1 Comparison with empirical results and implications

To examine this prediction of ultra slow diffusion and find the value of α, in Figure 2a, we examine $\left\|\mathbf{w}_t - \mathbf{w}_0\right\|$ during the initial training phase over the experiment shown in Figure 1. We found that the weight distance from initialization point increases logarithmically with the number of training iterations (weight updates), which matches our model with $\alpha = 2$:

$$\left\|\mathbf{w}_t - \mathbf{w}_0\right\| \sim \log t\,. \tag{4}$$

Interestingly, the value of $\alpha = 2$ matches the statistics of the loss estimated in appendix section B.

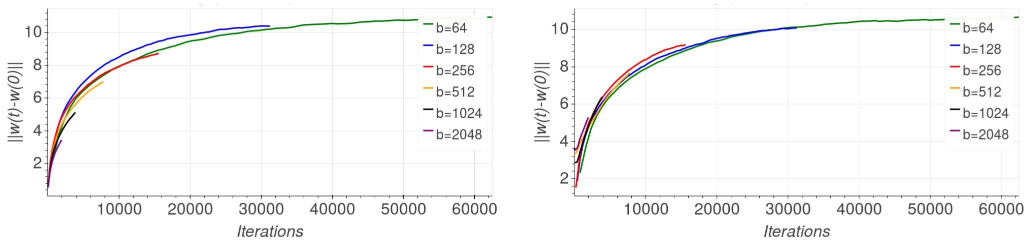
Moreover, in Figure 2a, we find that a very similar logarithmic graph is observed for all batch sizes. Yet, there are two main differences. First, each graph seems to have a somewhat different slope (i.e., it is multiplied by different positive constant), which peaks at $M = 128$ and then decreases with the mini-batch size. This indicates a somewhat different diffusion rate for different batch sizes. Second, since we trained all models for a constant number of epochs, smaller batch sizes entail more training iterations in total. Thus, there is a significant difference in the number of iterations and the corresponding weight distance reached at the end of the initial learning phase.

This leads to the following informal argument (which assumes flat minima are indeed important for generalization). During the initial training phase, to reach a minima of "width" d the weight vector $\mathbf{w}_t$ has to travel at least a distance d, and this takes a long time – about $\exp(d)$ iterations. Thus, to reach wide ("flat") minima we need to have the highest possible diffusion rates (which do not result in numerical instability) and a large number of training iterations. In the next sections we will implement these conclusions in practice.

4 Matching weight increment statistics for different mini-batch sizes

First, to correct the different diffusion rates observed for different batch sizes, we will aim to match the statistics of the weights increments to that of a small batch size.

[4] Note that this form is consistent with eq. (1), if $f(x) = x^{\alpha/2}$.



(a) Before learning rate adjustment and GBN

(b) After learning rate adjustment and GBN

Figure 2: Euclidean distance of weight vector from initialization

Learning rate. Recall that in this paper we investigate SGD, possibly with momentum, where the weight updates are proportional to the estimated gradient.

$$\Delta \mathbf{w} \propto \eta \hat{\mathbf{g}}\,, \tag{5}$$

where η is the learning rate, and we ignore for now the effect of batch normalization.

In appendix section A, we show that the covariance matrix of the parameters update step $\Delta \mathbf{w}$ is,

$$\operatorname{cov}\left(\Delta \mathbf{w}, \Delta \mathbf{w}\right) \approx \frac{\eta^2}{M}\left(\frac{1}{N}\sum_{n=1}^{N} \mathbf{g}_n \mathbf{g}_n^\top\right) \tag{6}$$

in the case of uniform sampling of the mini-batch indices (with or without replacement), when $M \ll N$. Therefore, a simple way to make sure that the covariance matrix stays the same for all mini-batch sizes is to choose

$$\eta \propto \sqrt{M}\,, \tag{7}$$

i.e., we should increase the learning rate by the square root of the mini-batch size.

We note that Krizhevsky (2014) suggested a similar learning rate scaling in order to keep the variance in the gradient expectation constant, but chose to use a linear scaling heuristics as it reached better empirical result in his setting. Later on, Li (2017) suggested the same.

Naturally, such an increase in the learning rate also increases the mean steps $\mathbb{E}\left[\Delta \mathbf{w}\right]$. However, we found that this effect is negligible since $\mathbb{E}\left[\Delta \mathbf{w}\right]$ is typically orders of magnitude lower than the standard deviation.

Furthermore, we can match both the first and second order statistics by adding multiplicative noise to the gradient estimate as follows:

$$\hat{\mathbf{g}} = \frac{1}{M}\sum_{n \in B}^{N} \mathbf{g}_n z_n\,,$$

where $z_n \sim \mathcal{N}\left(1, \sigma^2\right)$ are independent random Gaussian variables for which $\sigma^2 \propto M$. This can be verified by using similar calculation as in appendix section A. This method keeps the covariance constant when we change the batch size, yet does not change the mean steps $\mathbb{E}\left[\Delta \mathbf{w}\right]$.

In both cases, for the first few iterations, we had to clip or normalize the gradients to prevent divergence. Since both methods yielded similar performance [5] (due the negligible effect of the first order statistics), we preferred to use the simpler learning rate method.

It is important to note that other types of noise (e.g., dropout (Srivastava et al., 2014), dropconnect (Wan et al., 2013), label noise (Szegedy et al., 2016)) change the structure of the covariance matrix and not just its scale, thus the second order statistics of the small batch increment cannot be accurately matched. Accordingly, we did not find that these types of noise helped to reduce the generalization gap for large batch sizes.

Lastly, note that in our discussion above (and the derivations provided in appendix section A) we assumed each per-sample gradient $\mathbf{g}_n$ does not depend on the selected mini-batch. However, this ignores the influence of batch normalization. We take this into consideration in the next subsection.

[5] a simple comparison can be seen in appendix (figure 3)

Ghost Batch Normalization. Batch Normalization (BN) (Ioffe & Szegedy, 2015), is known to accelerate the training, increase the robustness of neural network to different initialization schemes and improve generalization. Nonetheless, since BN uses the batch statistics it is bounded to depend on the choosen batch size. We study this dependency and observe that by acquiring the statistics on small virtual ("ghost") batches instead of the real large batch we can reduce the generalization error. In our experiments we found out that it is important to use the full batch statistic as suggested by (Ioffe & Szegedy, 2015) for the inference phase. Full details are given in Algorithm 1. This modification by itself reduce the generalization error substantially.

Algorithm 1: Ghost Batch Normalization (GBN), applied to activation x over a large batch B_L with virtual mini-batch B_S. Where $B_S < B_L$.

```
Require: Values of $x$ over a large-batch: $B_L = \{x_{1...m}\}$ size of virtual batch $|B_S|$; Parameters to
    be learned: $\gamma$, $\beta$, momentum $\eta$
    Training Phase:
    Scatter $B_L$ to $\{X^1, X^2, ...X^{|B_L|/|B_S|}\} = \{x_{1...|B_S|}, x_{|B_S|+1...2|B_S|}...x_{|B_L|-|B_S|...m}\}$
    $\mu_B^l \leftarrow \frac{1}{|B_S|}\sum_{i=1}^{|B_S|} X_i^l$   for $l = 1, 2, 3 \ldots$   {calculate ghost mini-batches means}
    $\sigma_B^l \leftarrow \sqrt{\frac{1}{|B_S|}\sum_{i=1}^{|B_S|}(X_i^l - \mu_B)^2 + \epsilon}$   for $l = 1, 2, 3 \ldots$   {calculate ghost mini-batches std}
    $\mu_{run} = (1-\eta)^{|B_S|}\mu_{run} + \sum_{i=1}^{|B_L|/|B_S|}(1-\eta)^i \cdot \eta \cdot \mu_B^l$
    $\sigma_{run} = (1-\eta)^{|B_S|}\sigma_{run} + \sum_{i=1}^{|B_L|/|B_S|}(1-\eta)^i \cdot \eta \cdot \sigma_B^l$
    return $\gamma\frac{X^l - \mu_B^l}{\sigma_B^l} + \beta$
    Test Phase:
    return $\gamma\frac{X - \mu_{run}^l}{\sigma_{run}} + \beta$ {scale and shift}
```

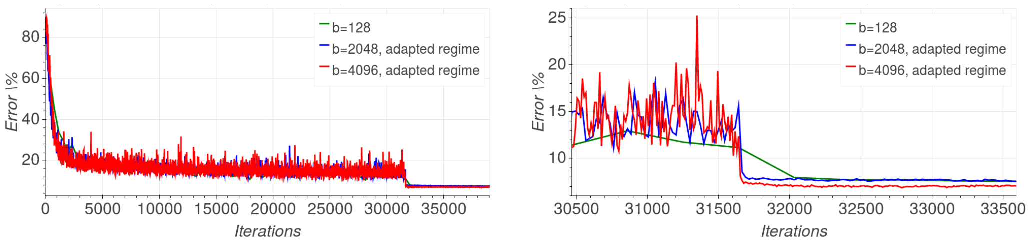
We note that in a multi-device distributed setting, some of the benefits of "Ghost BN" may already occur, since batch-normalization is often preformed on each device separately to avoid additional communication cost. Thus, each device computes the batch norm statistics using only its samples (i.e., part of the whole mini-batch). It is a known fact, yet unpublished, to the best of the authors knowledge, that this form of batch norm update helps generalization and yields better results than computing the batch-norm statistics over the entire batch. Note that GBN enables flexibility in the small (virtual) batch size which is not provided by the commercial frameworks (e.g., TensorFlow, PyTorch) in which the batch statistics is calculated on the entire, per-device, batch. Moreover, in those commercial frameworks, the running statistics are usually computed differently from "Ghost BN", by weighting each update part equally. In our experiments we found it to worsen the generalization performance.

Implementing both the learning rate and GBN adjustments seem to improve generalization performance, as we shall see in section 6. Additionally, as can be seen in Figure 6, the slopes of the logarithmic weight distance graphs seem to better matched, indicating similar diffusion rates. We also observe some constant shift, which we believe is related to the gradient clipping. Since this shift only increased the weight distances, we assume it does not harm the performance.

5 Adapting number of weight updates eliminates generalization gap

According to our conclusions in section 3, the initial high-learning rate training phase enables the model to reach farther locations in the parameter space, which may be necessary to find wider local minima and better generalization. Examining figure 2b, the next obvious step to match the graphs for different batch sizes is to increase the number of training iterations in the initial high learning rate regime. And indeed we noticed that the distance between the current weight and the initialization point can be a good measure to decide upon when to decrease the learning rate.

Note that this is different from common practices. Usually, practitioners decrease the learning rate after validation error appears to reach a plateau. This practice is due to the long-held belief that the optimization process should not be allowed to decrease the training error when validation error "flatlines", for fear of overfitting (Girosi et al., 1995). However, we observed that substantial improvement to the final accuracy can be obtained by continuing the optimization using the same



(a) Validation error

(b) Validation error - zoomed

Figure 3: Comparing generalization of large-batch regimes, adapted to match performance of small-batch training.

learning rate even if the training error decreases while the validation plateaus. Subsequent learning rate drops resulted with a sharp validation error decrease, and better generalization for the final model.

These observations led us to believe that "generalization gap" phenomenon stems from the relatively small number of updates rather than the batch size. Specifically, using the insights from Figure 2 and our model, we adapted the training regime to better suit the usage of large mini-batch. We "stretched" the time-frame of the optimization process, where each time period of e epochs in the original regime, will be transformed to $\frac{|B_L|}{|B_S|}e$ epochs according to the mini-batch size used. This modification ensures that the number of optimization steps taken is identical to those performed in the small batch regime. As can be seen in Figure 3, combining this modification with learning rate adjustment completely eliminates the generalization gap observed earlier [6].

6 Experiments

Experimental setting. We experimented with a set of popular image classification tasks:

- MNIST (LeCun et al., 1998b) - Consists of a training set of $60K$ and a test set of $10K$ 28×28 gray-scale images representing digits ranging from 0 to 9.
- CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) - Each consists of a training set of size 50K and a test set of size $10K$. Instance are 32×32 color images representing 10 or 100 classes.
- ImageNet classification task Deng et al. (2009) - Consists of a training set of size 1.2M samples and test set of size 50K. Each instance is labeled with one of 1000 categories.

To validate our findings, we used a representative choice of neural network models. We used the fully-connected model, F1, as well as shallow convolutional models C1 and C3 suggested by Keskar et al. (2017). As a demonstration of more current architectures, we used the models: VGG (Simonyan, 2014) and Resnet44 (He et al., 2016) for CIFAR10 dataset, Wide-Resnet16-4 (Zagoruyko, 2016) for CIFAR100 dataset and Alexnet (Krizhevsky, 2014) for ImageNet dataset.
In each of the experiments, we used the training regime suggested by the original work, together with a momentum SGD optimizer. We use a batch of 4096 samples as "large batch" (LB) and a small batch (SB) of either 128 (F1,C1,VGG,Resnet44,C3,Alexnet) or 256 (WResnet). We compare the original training baseline for small and large batch, as well as the following methods[7]:

- Learning rate tuning (LB+LR): Using a large batch, while adapting the learning rate to be larger so that $\eta_L = \sqrt{\frac{|B_L|}{|B_S|}}\eta_S$ where η_S is the original learning rate used for small batch, η_L is the adapted learning rate and $|B_L|, |B_S|$ are the large and small batch sizes, respectively.
- Ghost batch norm (LB+LR+GBN): Additionally using the "Ghost batch normalization" method in our training procedure. The "ghost batch size" used is 128.
- Regime adaptation: Using the tuned learning rate as well as ghost batch-norm, but with an adapted training regime. The training regime is modified to have the same number of

[6] Additional graphs, including comparison to non-adapted regime, are available in appendix (figure 2).
[7] Code is available at `https://github.com/eladhoffer/bigBatch`.

iterations for each batch size used - effectively multiplying the number of epochs by the relative size of the large batch.

Results. Following our experiments, we can establish an empirical basis to our claims. Observing the final validation accuracy displayed in Table 1, we can see that in accordance with previous works the move from a small-batch (SB) to a large-batch (LB) indeed incurs a substantial generalization gap. However, modifying the learning-rate used for large-batch (+LR) causes much of this gap to diminish, following with an additional improvement by using the Ghost-BN method (+GBN). Finally, we can see that the generalization gap completely disappears when the training regime is adapted (+RA), yielding validation accuracy that is good-as or better than the one obtained using a small batch. We additionally display results obtained on the more challenging ImageNet dataset in Table 2 which shows similar impact for our methods.

Table 1: Validation accuracy results, SB/LB represent small and large batch respectively. GBN stands for Ghost-BN, and RA stands for regime adaptation

Network	Dataset	SB	LB	+LR	+GBN	+RA
F1 (Keskar et al., 2017)	MNIST	98.27%	97.05%	97.55%	97.60%	98.53%
C1 (Keskar et al., 2017)	Cifar10	87.80%	83.95%	86.15%	86.4%	88.20%
Resnet44 (He et al., 2016)	Cifar10	92.83%	86.10%	89.30%	90.50%	93.07%
VGG (Simonyan, 2014)	Cifar10	92.30%	84.1%	88.6%	91.50%	93.03%
C3 (Keskar et al., 2017)	Cifar100	61.25%	51.50%	57.38%	57.5%	63.20%
WResnet16-4 (Zagoruyko, 2016)	Cifar100	73.70%	68.15%	69.05%	71.20%	73.57%

Table 2: ImageNet top-1 results using Alexnet topology (Krizhevsky, 2014), notation as in Table 1.

Network	LB size	Dataset	SB	LB[8]	+LR[8]	+GBN	+RA
Alexnet	4096	ImageNet	57.10%	41.23%	53.25%	54.92%	59.5%
Alexnet	8192	ImageNet	57.10%	41.23%	53.25%	53.93%	59.5%

7 Discussion

There are two important issues regarding the use of large batch sizes. First, why do we get worse generalization with a larger batch, and how do we avoid this behaviour? Second, can we decrease the training wall clock time by using a larger batch (exploiting parallelization), while retaining the same generalization performance as in small batch?

This work tackles the first issue by investigating the random walk behaviour of SGD and the relationship of its diffusion rate to the size of a batch. Based on this and empirical observations, we propose simple set of remedies to close down the generalization gap between the small and large batch training strategies: (1) Use SGD with momentum, gradient clipping, and a decreasing learning rate schedule; (2) adapt the learning rate with batch size (we used a square root scaling); (3) compute batch-norm statistics over several partitions ("ghost batch-norm"); and (4) use a sufficient number of high learning rate training iterations.

Thus, the main point arising from our results is that, in contrast to previous conception, there is no inherent generalization problem with training using large mini batches. That is, model training using large mini-batches can generalize as well as models trained using small mini-batches. Though this answers the first issues, the second issue remained open: can we speed up training by using large batch sizes?

Not long after our paper first appeared, this issue was also answered. Using a Resnet model on Imagenet Goyal et al. (2017) showed that, indeed, significant speedups in training could be achieved using a large batch size. This further highlights the ideas brought in this work and their importance to future scale-up, especially since Goyal et al. (2017) used similar training practices to those we

[8] Due to memory limitation those experiments were conducted with batch of 2048.

described above. The main difference between our works is the use of a linear scaling of the learning rate[9], similarly to Krizhevsky (2014), and as suggested by Bottou (2010). However, we found that linear scaling works less well on CIFAR10, and later work found that linear scaling rules work less well for other architectures on ImageNet (You et al., 2017).

We also note that current "rules of thumb" regarding optimization regime and explicitly learning rate annealing schedule may be misguided. We showed that good generalization can result from extensive amount of gradient updates in which there is no apparent validation error change and training error continues to drop, in contrast to common practice. After our work appeared, Soudry et al. (2017) suggested an explanation to this, and to the logarithmic increase in the weight distance observed in Figure 2. We show this behavior happens even in simple logistic regression problems with separable data. In this case, we exactly solve the asymptotic dynamics and prove that $\mathbf{w}(t) = \log(t)\hat{\mathbf{w}} + O(1)$ where $\hat{\mathbf{w}}$ is to the L_2 maximum margin separator. Therefore, the margin (affecting generalization) improves slowly (as $O(1/\log(t))$, even while the training error is very low. Future work, based on this, may be focused on finding when and how the learning rate should be decreased while training.

Conclusion. In this work we make a first attempt to tackle the "generalization gap" phenomenon. We argue that the initial learning phase can be described using a high-dimensional "random walk on a random potential" process, with a an "ultra-slow" logarithmic increase in the distance of the weights from their initialization, as we observe empirically. Following this observation we suggest several techniques which enable training with large batch without suffering from performance degradation. This implies that the problem is not related to the batch size but rather to the amount of updates. Moreover we introduce a simple yet efficient algorithm "Ghost-BN" which improves the generalization performance significantly while keeping the training time intact.

Acknowledgments

We wish to thank Nir Ailon, Dar Gilboa, Kfir Levy and Igor Berman for their feedback on the initial manuscript. The research leading to these results has received funding from the European Research Council under European Unions Horizon 2020 Program, ERC Grant agreement no. 682203 ”Speed-InfTradeoff”. The research was partially supported by the Taub Foundation, and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DoI/IBC) contract number D16PC00003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government.

References

Amodei, D., Anubhai, R., Battenberg, E., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Bouchaud, J. P. and Georges, A. Anomalous diffusion in disordered media: statistical mechanisms, models and physical applications. *Physics reports*, 195:127–293, 1990.

Bouchaud, J. P. and Comtet, A. Anomalous diffusion in random media of any dimensionality. *J. Physique*, 48: 1445–1450, 1987.

Bray, A. J. and Dean, D. S. Statistics of critical points of Gaussian fields on large-dimensional spaces. *Physical Review Letters*, 98(15):1–5, 2007.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The Loss Surfaces of Multilayer Networks. *AISTATS15*, 38, 2015.

Das, D., Avancha, S., Mudigere, D., et al. Distributed deep learning using synchronous stochastic gradient descent. *arXiv preprint arXiv:1602.06709*, 2016.

Dauphin, Y., de Vries, H., Chung, J., and Bengio, Y. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. corr abs/1502.04390 (2015).

Dauphin, Y., Pascanu, R., and Gulcehre, C. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, pp. 1–9, 2014.

Dean, J., Corrado, G., Monga, R., et al. Large scale distributed deep networks. In *NIPS*, pp. 1223–1231, 2012.

[9] e.g., Goyal et al. (2017) also used an initial warm-phase for the learning rate, however, this has a similar effect to the gradient clipping we used, since this clipping was mostly active during the initial steps of training.

Deng, J., Dong, W., Socher, R., et al. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Durrett, R. Multidimensional random walks in random environments with subclassical limiting behavior. *Communications in Mathematical Physics*, 104(1):87–102, 1986.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle points-online stochastic gradient for tensor decomposition. In *COLT*, pp. 797–842, 2015.

Girosi, F., Jones, M., and Poggio, T. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pp. 249–256, 2010.

Goyal, P., Dollár, P., Girshick, R., et al. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *ICML*, pp. 1–24, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

LeCun, Y., Bottou, L., and Orr, G. Efficient backprop in neural networks: Tricks of the trade (orr, g. and müller, k., eds.). *Lecture Notes in Computer Science*, 1524, 1998a.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998b.

Li, M. *Scaling Distributed Machine Learning with System and Algorithm Co-design*. PhD thesis, Intel, 2017.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670. ACM, 2014.

Luong, M.-T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Marinari, E., Parisi, G., Ruelle, D., and Windey, P. Random Walk in a Random Environment and 1f Noise. *Physical Review Letters*, 50(1):1223–1225, 1983.

Mnih, V., Kavukcuoglu, K., Silver, D., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Montavon, G., Orr, G., and Müller, K.-R. *Neural Networks: Tricks of the Trade*. 2 edition, 2012. ISBN 978-3-642-35288-1.

Ruder, S. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

Silver, D., Huang, A., Maddison, C. J., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Simonyan, K. e. a. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Soudry, D., Hoffer, E., and Srebro, N. The Implicit Bias of Gradient Descent on Separable Data. *ArXiv e-prints*, October 2017.

Soudry, D. and Hoffer, E. Exponentially vanishing sub-optimal local minima in multilayer neural networks. *arXiv preprint arXiv:1702.05777*, 2017.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. Regularization of neural networks using dropconnect. ICML'13, pp. III–1058–III–1066. JMLR.org, 2013.

Wu, Y., Schuster, M., Chen, Z., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

You, Y., Gitman, I., and Ginsburg, B. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 2017.

Zagoruyko, K. Wide residual networks. In *BMVC*, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Zhang, S., Choromanska, A. E., and LeCun, Y. Deep learning with elastic averaging sgd. In *NIPS*, pp. 685–693, 2015.

Appendix

A Derivation of eq. (6)

Note that we can write the mini-batch gradient as

$$\hat{\mathbf{g}} = \frac{1}{M}\sum_{n=1}^{N}\mathbf{g}_n s_n \text{ with } s_n \triangleq \begin{cases} 1 & \text{, if } n \in B \\ 0 & \text{, if } n \notin B \end{cases}$$

Clearly, $\hat{\mathbf{g}}$ is an unbiased estimator of $\mathbf{g}$, if

$$\mathbb{E}s_n = P\left(s_n = 1\right) = \frac{M}{N}\,.$$

since then

$$\mathbb{E}\hat{\mathbf{g}} = \frac{1}{M}\sum_{n=1}^{N}\mathbf{g}_n \mathbb{E}s_n = \frac{1}{N}\sum_{n=1}^{N}\mathbf{g}_n = \mathbf{g}\,.$$

First, we consider the simpler case of sampling with replacement. In this case it easy to see that different minibatches are uncorrelated, and we have

$$\begin{aligned}\mathbb{E}\left[s_n s_{n'}\right] &= P\left(s_n = 1\right)\delta_{nn'} + P\left(s_n = 1, s_{n'} = 1\right)\left(1-\delta_{nn'}\right) \\ &= \frac{M}{N}\delta_{nn'} + \frac{M^2}{N^2}\left(1-\delta_{nn'}\right)\,.\end{aligned}$$

and therefore

$$\begin{aligned}\mathrm{cov}\left(\hat{\mathbf{g}}, \hat{\mathbf{g}}\right) &= \mathbb{E}\left[\hat{\mathbf{g}}\hat{\mathbf{g}}^\top\right] - \mathbb{E}\hat{\mathbf{g}}\mathbb{E}\hat{\mathbf{g}}^\top \\ &= \frac{1}{M^2}\sum_{n=1}^{N}\sum_{n'=1}^{N}\mathbb{E}\left[s_n s_{n'}\right]\mathbf{g}_n\mathbf{g}_{n'}^\top - \mathbf{g}\mathbf{g}^\top \\ &= \frac{1}{M^2}\sum_{n=1}^{N}\sum_{n'=1}^{N}\left[\frac{M}{N}\delta_{nn'} + \frac{M^2}{N^2}\left(1-\delta_{nn'}\right)\right]\mathbf{g}_n\mathbf{g}_{n'}^\top - \mathbf{g}\mathbf{g}^\top \\ &= \left(\frac{1}{M} - \frac{1}{N}\right)\frac{1}{N}\sum_{n=1}^{N}\mathbf{g}_n\mathbf{g}_n^\top,\end{aligned}$$

which confirms eq. (6).

Next, we consider the case of sampling without replacement. In this case the selector variables are now different and correlated between different mini-batches (*e.g.*, with indices t and $t+k$), since we cannot select previous samples. Thus, these variables s_n^t and s_n^{t+k} have the following second-order statistics

$$\begin{aligned}\mathbb{E}\left[s_n^t s_{n'}^{t+k}\right] &= P\left(s_n^t = 1, s_{n'}^{t+k} = 1\right) \\ &= \frac{M}{N}\delta_{nn'}\delta_{k0} + \frac{M}{N}\frac{M}{N-1}\left(1-\delta_{nn'}\delta_{k0}\right)\,.\end{aligned}$$

This implies

$$\begin{aligned}&\mathbb{E}\left[\hat{\mathbf{g}}_t\hat{\mathbf{g}}_{t+k}^\top\right] - \mathbb{E}\hat{\mathbf{g}}_t\mathbb{E}\hat{\mathbf{g}}_{t+k}^\top \\ &= \mathbb{E}\left[\left(\frac{1}{M}\sum_{n=1}^{N}s_n^t\mathbf{g}_n\right)\left(\frac{1}{M}\sum_{n'=1}^{N}s_{n'}^{t+k}\mathbf{g}_{n'}^\top\right)\right] - \mathbf{g}\mathbf{g}^\top \\ &= \frac{1}{M^2}\sum_{n=1}^{N}\sum_{n'=1}^{N}\mathbb{E}\left[s_n^t s_{n'}^{t+k}\right]\mathbf{g}_n\mathbf{g}_{n'}^\top - \mathbf{g}\mathbf{g}^\top \\ &= \frac{1}{M^2}\sum_{n=1}^{N}\sum_{n'=1}^{N}\left[\left(\frac{M}{N} - \frac{M^2}{N^2-N}\right)\delta_{nn'}\delta_{k0} - \frac{M^2}{N^2-N}\right]\mathbf{g}_n\mathbf{g}_{n'}^\top - \mathbf{g}\mathbf{g}^\top\end{aligned}$$

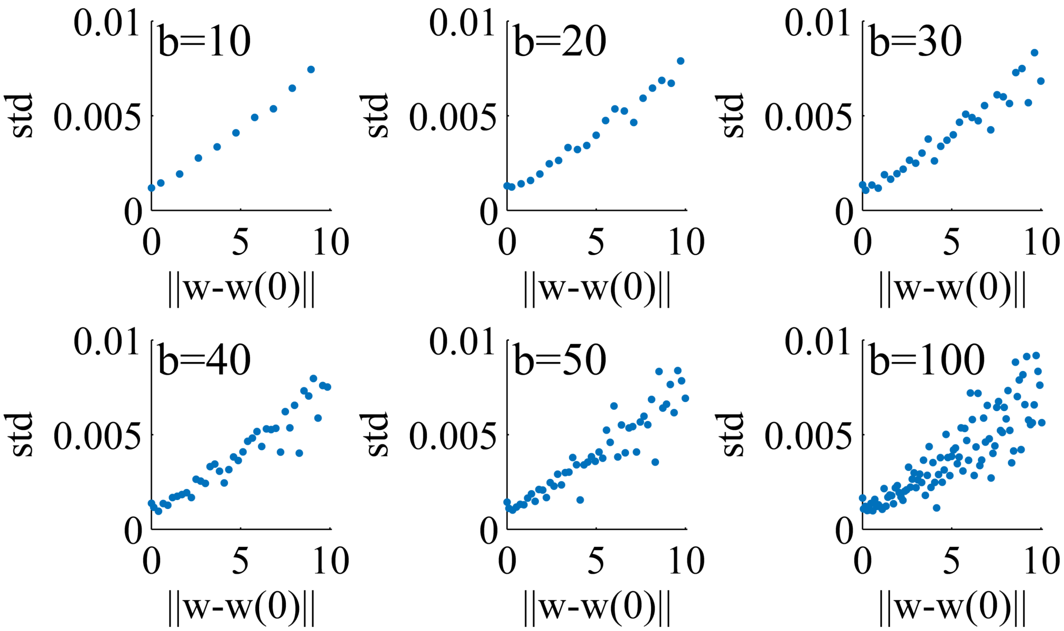


Figure 4: **The standard deviation of the loss shows linear dependence on weight distance (eq. 8) as predicted by the "random walk on a random potential" model with $\alpha = 2$ we found in the main paper.** To approximate the ensemble average in eq. 8 we divided the x-axis to b bins and calculated the empiric average in each bin. Each panel shows the resulting graph for a different value of b.

so, if $k = 0$ the covariance is

$$\mathbb{E}\left[\hat{\mathbf{g}}_t\hat{\mathbf{g}}_t^\top\right] - \mathbb{E}\hat{\mathbf{g}}_t\mathbb{E}\hat{\mathbf{g}}_t^\top = \left(\frac{1}{M} - \frac{1}{N-1}\right)\frac{1}{N}\sum_{n=1}^{N}\mathbf{g}_n\mathbf{g}_n^\top + \frac{1}{N-1}\mathbf{g}\mathbf{g}^\top$$

$$\overset{M\ll N}{\approx} \frac{1}{M}\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{g}_n\mathbf{g}_n^\top\right)$$

while the covariance between different minibatches ($k \neq 0$) is much smaller for $M \ll N$

$$\mathbb{E}\left[\hat{\mathbf{g}}_t\hat{\mathbf{g}}_{t+k}^\top\right] - \mathbb{E}\hat{\mathbf{g}}_t\mathbb{E}\hat{\mathbf{g}}_{t+k}^\top = \frac{1}{N-1}\mathbf{g}\mathbf{g}^\top$$

this again confirms eq. (6).

B Estimating α from random potential

The logarithmic increase in weight distance (Figure 2 in the paper) matches a “random walk on a random potential” model with $\alpha = 2$. In such a model the loss auto-covariance asymptotically increases with the square of the weight distance, or, equivalently (Marinari et al., 1983), the standard deviation of the loss difference asymptotically increases linearly with the weight distance

$$\text{std} \triangleq \sqrt{\mathbb{E}\left(L\left(\mathbf{w}\right) - L\left(\mathbf{w}_0\right)\right)^2} \sim \left\|\mathbf{w} - \mathbf{w}_0\right\| . \tag{8}$$

In this section we examine this behavior: in Figure we indeed find such a linear behavior, confirming the prediction of our model with $\alpha = 2$.

To obtain the relevant statistics to plot eq. 8 we conducted the following experiment on Resnet44 model (He et al., 2016). We initialized the model weights, $\mathbf{w}_0$, according to Glorot & Bengio (2010), and repeated the following steps a 1000 times, given some parameter c:

- Sample a random direction $\mathbf{v}$ with norm one.
- Sample a scalar z uniformly in some range $[0, c]$.
- Choose $\mathbf{w} = \mathbf{w}_0 + z\mathbf{v}$.
- Save $||\mathbf{w} - \mathbf{w}_0||$ and $L(\mathbf{w})$.

We have set the parameter c so that the maximum weight distance from initialization $||\mathbf{w} - \mathbf{w}_0||$ is equal to the same maximal distance in Figure 2 in the paper, *i.e.,* $c \approx 10$.

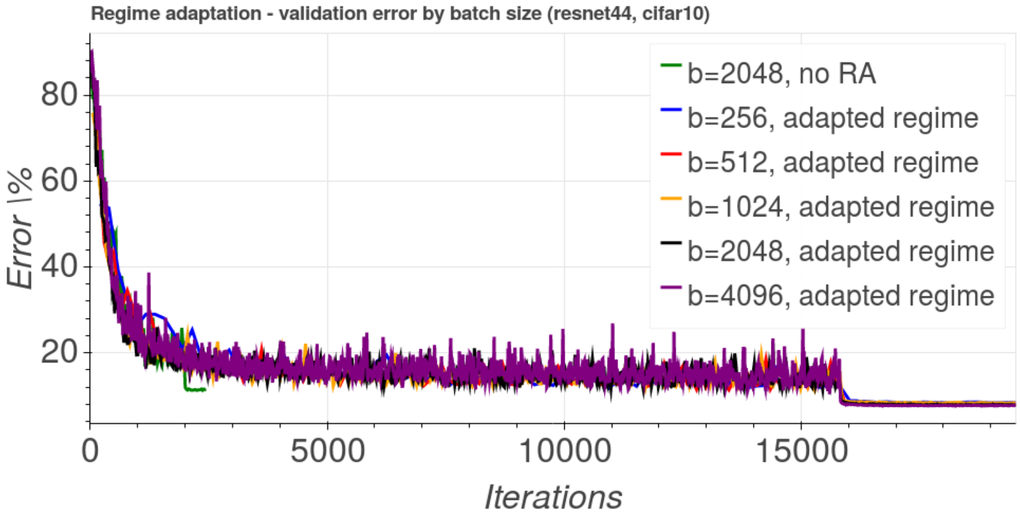


Figure 5: Comparing regime adapted large batch training vs. a 2048 batch with no adaptation.

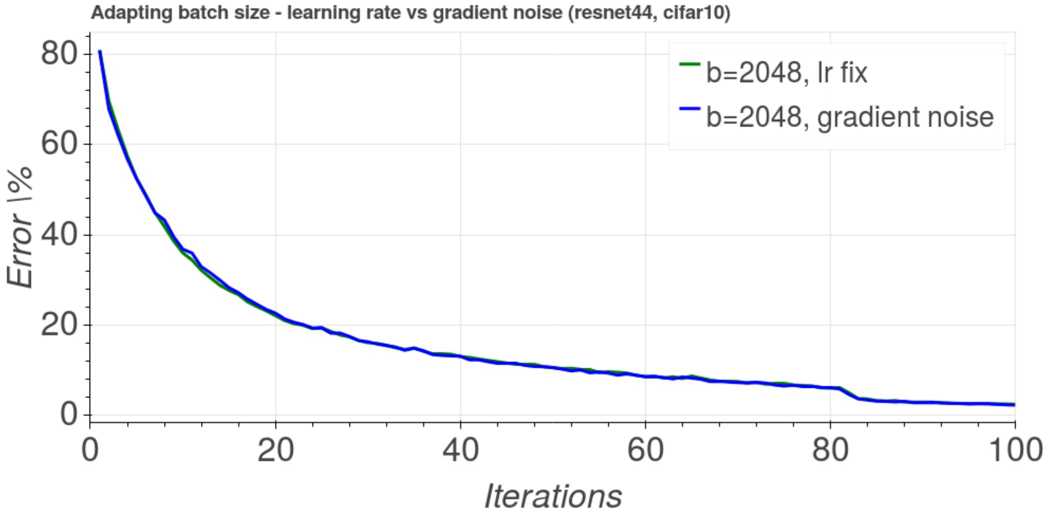


Figure 6: Comparing a learning scale fix for a 2048 batch, to a multiplicative noise to the gradient of the same scale

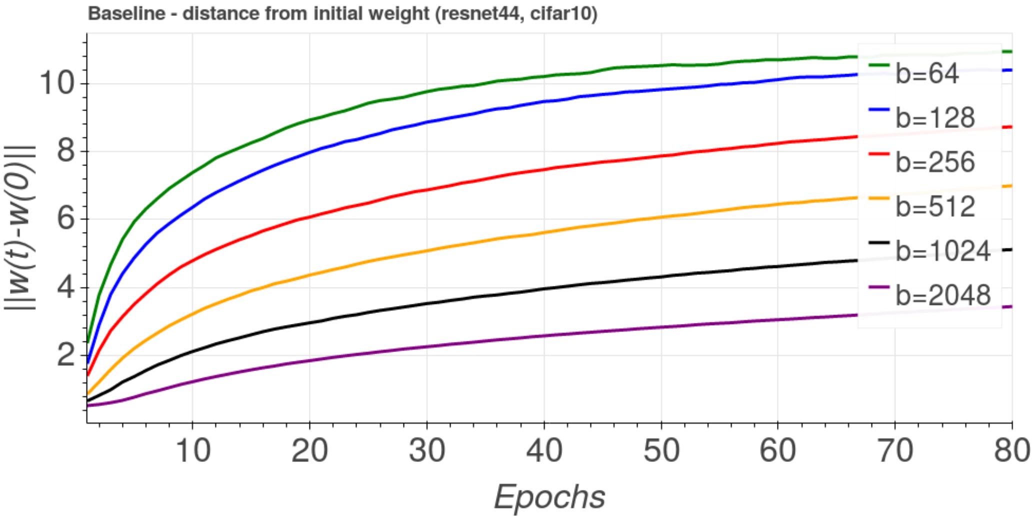


Figure 7: Comparing L_2 distance from initial weight for different batch sizes