REVISITING SMALL BATCH TRAINING FOR DEEP NEURAL NETWORKS

Dominic Masters and Carlo Luschi
Graphcore Research
Bristol, UK
{dominicm,carlo}@graphcore.ai

ABSTRACT

Modern deep neural network training is typically based on mini-batch stochastic gradient optimization. While the use of large mini-batches increases the available computational parallelism, small batch training has been shown to provide improved generalization performance and allows a significantly smaller memory footprint, which might also be exploited to improve machine throughput.

In this paper, we review common assumptions on learning rate scaling and training duration, as a basis for an experimental comparison of test performance for different mini-batch sizes. We adopt a learning rate that corresponds to a constant average weight update per gradient calculation (i.e., per unit cost of computation), and point out that this results in a variance of the weight updates that increases linearly with the mini-batch size m.

The collected experimental results for the CIFAR-10, CIFAR-100 and ImageNet datasets show that increasing the mini-batch size progressively reduces the range of learning rates that provide stable convergence and acceptable test performance. On the other hand, small mini-batch sizes provide more up-to-date gradient calculations, which yields more stable and reliable training. The best performance has been consistently obtained for mini-batch sizes between $m = 2$ and $m = 32$, which contrasts with recent work advocating the use of mini-batch sizes in the thousands.

1 INTRODUCTION

The use of deep neural networks has recently enabled significant advances in a number of applications, including computer vision, speech recognition and natural language processing, and in reinforcement learning for robotic control and game playing (LeCun et al., 2015; Schmidhuber, 2015; Goodfellow et al., 2016; Arulkumaran et al., 2017).

Deep learning optimization is typically based on Stochastic Gradient Descent (SGD) or one of its variants (Bottou et al., 2016; Goodfellow et al., 2016). The SGD update rule relies on a stochastic approximation of the expected value of the gradient of the loss function over the training set, based on a small subset of training examples, or *mini-batch*.

The recent drive to employ progressively larger batch sizes is motivated by the desire to improve the parallelism of SGD, both to increase the efficiency of current processors and to allow distributed processing on a larger number of nodes (Dean et al., 2012; Das et al., 2016). In contrast, the use of small batch sizes has been shown to improve generalization performance and optimization convergence (Wilson & Martinez, 2003; LeCun et al., 2012; Keskar et al., 2016). The use of small batch sizes also has the advantage of requiring a significantly smaller memory footprint, which affords an opportunity to design processors which gain efficiency by exploiting memory locality. This motivates a fundamental trade-off in the choice of batch size.

Hoffer et al. (2017) have shown empirically that it is possible to maintain generalization performance with large batch training by performing the same number of SGD updates. However, this implies a computational overhead proportional to the mini-batch size, which negates the effect of improved hardware efficiency due to increased parallelism.

In order to improve large batch training performance for a given training computation cost, it has been proposed to scale the learning rate linearly with the batch size (Krizhevsky, 2014; Chen et al., 2016; Bottou et al., 2016; Goyal et al., 2017). In this work, we suggest that the discussion about the scaling of the learning rate with the batch size depends on how the problem is formalized, and revert to the view of Wilson & Martinez (2003), that considers the SGD weight update formulation based on the sum, instead of the average, of the gradients over the mini-batch. From this perspective, using a fixed learning rate keeps the expectation of the weight update per training example constant for any choice of batch size. At the same time, as will be discussed in Section 2, holding the expected value of the weight update per gradient calculation constant while increasing the batch size implies a linear increase of the variance of the weight update with the batch size.

To investigate these issues, we have performed a comprehensive set of experiments for a range of network architectures. The results provide evidence that increasing the batch size results in both a degradation of the test performance and a progressively smaller range of learning rates that allows stable training, where the notion of stability here refers to the robust convergence of the training algorithm.

The paper is organized as follows. Section 2 briefly reviews the main work on batch training and normalization. Section 3 presents a range of experimental results on training and generalization performance for the CIFAR-10, CIFAR-100 and ImageNet datasets. Previous work has compared training performance using batch sizes of the order of 128–256 with that of very large batch sizes of up to 4096 (Hoffer et al., 2017) or 8192 (Goyal et al., 2017), while we consider the possibility of using a wider range of values, down to batch sizes as small as 2 or 4. Section 4 provides a discussion of the main results presented in the paper. Finally, conclusions are drawn in Section 5.

2 Background: Batch Training and Batch Normalization

2.1 Stochastic Gradient Optimization

We assume a deep network model with parameters $\boldsymbol{\theta}$, and consider the non-convex optimization problem corresponding to the minimization of the loss function $L(\boldsymbol{\theta})$, with respect to $\boldsymbol{\theta}$. $L(\boldsymbol{\theta})$ is defined as the sample average of the loss per training example $L_i(\boldsymbol{\theta})$ over the training set,

$$L(\boldsymbol{\theta}) = \frac{1}{M} \sum_{i=1}^{M} L_i(\boldsymbol{\theta}) \,, \tag{1}$$

where M denotes the size of the training set. The above empirical loss is used as a proxy for the expected value of the loss with respect to the true data generating distribution.

Batch gradient optimization originally referred to the case where the gradient computations were accumulated over one presentation of the entire training set before being applied to the parameters, and stochastic gradient methods were typically online methods with parameter update based on a single training example. Current deep learning stochastic gradient algorithms instead use parameter updates based on gradient averages over small subsets of the full training set, or *mini-batches*, and the term *batch size* is commonly used to refer to the size of a mini-batch (Goodfellow et al., 2016).

Optimization based on the SGD algorithm uses a stochastic approximation of the gradient of the loss $L(\boldsymbol{\theta})$ obtained from a mini-batch $\mathcal{B}$ of m training examples, resulting in the weight update rule

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \, \Delta\boldsymbol{\theta}_k \tag{2}$$

$$\Delta\boldsymbol{\theta}_k = -\frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta}_k) \tag{3}$$

where η denotes the learning rate.

From (2), (3) the mean value of the SGD weight update is given by $\mathbb{E}\{\eta \, \Delta\boldsymbol{\theta}\} = -\eta \, \mathbb{E}\{\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})\}$. Therefore, for a given batch size m, the expected value of the weight update per unit cost of compu-

tation (per training example, i.e., per gradient calculation $\nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta})$) is proportional to η/m:

$$\frac{1}{m}\,\mathbb{E}\{\eta\,\Delta\boldsymbol{\theta}\} = -\frac{\eta}{m}\,\mathbb{E}\{\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})\}\,. \tag{4}$$

This implies that, when increasing the batch size, a linear increase of the learning rate η with the batch size m is required to keep the mean SGD weight update per training example constant.

This *linear scaling* rule has been widely adopted, e.g., in Krizhevsky (2014), Chen et al. (2016), Bottou et al. (2016), Smith et al. (2017) and Jastrzebski et al. (2017).

On the other hand, as shown in Hoffer et al. (2017), when $m \ll M$, the covariance matrix of the weight update $\mathrm{Cov}\{\eta\,\Delta\boldsymbol{\theta}\}$ scales linearly with the quantity η^2/m.

This implies that, adopting the linear scaling rule, an increase in the batch size would also result in a linear increase in the covariance matrix of the weight update $\eta\,\Delta\boldsymbol{\theta}$. Conversely, to keep the scaling of the covariance of the weight update vector $\eta\,\Delta\boldsymbol{\theta}$ constant would require scaling η with the square root of the batch size m (Krizhevsky, 2014; Hoffer et al., 2017).

2.2 A Different Perspective on Learning Rate Scaling

We suggest that the discussion about the linear or sub-linear increase of the learning rate with the batch size is the purely formal result of assuming the use of the *average* of the local gradients over a mini-batch in the SGD weight update.

As discussed in Wilson & Martinez (2003), current batch training implementations typically use a weight correction based on the average of the local gradients according to equation (3), while earlier work on batch optimization often assumed a weight correction based on the *sum* of the local gradients (Wilson & Martinez, 2003). Using the sum of the gradients at the point $\boldsymbol{\theta}_k$, the SGD parameter update rule can be rewritten as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \tilde{\eta} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta}_k) \tag{5}$$

and in this case, if the batch size m is increased, the mean SGD weight update per training example is kept constant by simply maintaining a constant learning rate $\tilde{\eta}$. This is equivalent to using the linear scaling rule.

In the SGD weight update formulation (5), the learning rate $\tilde{\eta}$ corresponds to the *base learning rate* that would be obtained from a linear increase of the learning rate η of (2), (3) with the batch size m, i.e.

$$\tilde{\eta} = \eta\,\frac{1}{m}\,. \tag{6}$$

At the same time, from Section 2.1 the variance of the weight update scales linearly with $\tilde{\eta}^2 \cdot m$. Therefore keeping the base learning rate $\tilde{\eta}$ constant implies a linear increase of the variance with the batch size m.

If we now consider a sequence of updates from the point $\boldsymbol{\theta}_k$ with a batch size m, from (5) the value of the weights at step $k+n$ is expressed as

$$\boldsymbol{\theta}_{k+n} = \boldsymbol{\theta}_k - \tilde{\eta} \sum_{j=0}^{n-1} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L_{i+jm}(\boldsymbol{\theta}_{k+j})\,, \tag{7}$$

while increasing the batch size by a factor n implies that the weights at step $k+1$ (corresponding to the same number of gradient calculations) are instead given by

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \tilde{\eta} \sum_{i=1}^{nm} \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta}_k)\,. \tag{8}$$

The comparison of (7), (8) highlights how, under the assumption of constant $\tilde{\eta}$, large batch training can be considered to be an approximation of small batch methods that trades increased parallelism for stale gradients.

From (7), (8), the end point values of the weights $\boldsymbol{\theta}_{k+n}$ and $\boldsymbol{\theta}_{k+1}$ after nm gradient calculations will generally be different, except in the case where one could assume

$$\nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta}_k) \approx \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta}_{k+j}) \,, \qquad j = 1, ..., n-1 \tag{9}$$

as also pointed out in Wilson & Martinez (2003) and Goyal et al. (2017). Therefore, for larger batch sizes, the update rule will be governed by progressively different dynamics for larger values of n, especially during the initial part of the training, during which the network parameters are typically changing rapidly over the non-convex loss surface (Goyal et al., 2017). It is important to note that (9) is a better approximation if the batch size and/or the base learning rate are small.

The purely formal difference of using the average of the local gradients instead of the sum has favoured the conclusion that using a larger batch size could provide more 'accurate' gradient estimates and allow the use of larger learning rates. However, the above analysis shows that, from the perspective of maintaining the expected value of the weight update per unit cost of computation, this may not be true. In fact, using smaller batch sizes allows gradients based on more up-to-date weights to be calculated, which in turn allows the use of higher base learning rates, as each SGD update has lower variance. Both of these factors potentially allow for faster and more robust convergence.

2.3 Effect of Batch Normalization

The training of modern deep networks commonly employs *Batch Normalization* (Ioffe & Szegedy, 2015). This technique has been shown to significantly improve training performance and has now become a standard component of many state-of-the-art networks.

Batch Normalization (BN) addresses the problem of *internal covariate shift* by reducing the dependency of the distribution of the input activations of each layer on all the preceding layers. This is achieved by normalizing activation x_i for each feature,

$$\hat{x}_i = \frac{x_i - \hat{\mu}_{\mathcal{B}}}{\sqrt{\hat{\sigma}^2_{\mathcal{B}} + \epsilon}}, \tag{10}$$

where $\hat{\mu}_{\mathcal{B}}$ and $\hat{\sigma}^2_{\mathcal{B}}$ denote respectively the sample mean and sample variance calculated over the batch $\mathcal{B}$ for one feature. The normalized values $\hat{x}_i$ are then further scaled and shifted by the learned parameters γ and β (Ioffe & Szegedy, 2015)

$$y_i = \hat{x}_i \cdot \gamma + \beta \,. \tag{11}$$

For the case of a convolutional layer, with a feature map of size $p \times q$ and batch size m, the sample size for the estimate of $\hat{\mu}_{\mathcal{B}}$ and $\hat{\sigma}^2_{\mathcal{B}}$ is given by $m \cdot p \cdot q$, while for a fully-connected layer the sample size is simply equal to m. For very small batches, the estimation of the batch mean and variance can be very noisy, which may limit the effectiveness of BN in reducing the covariate shift. Moreover, as pointed out in Ioffe (2017), with very small batch sizes the estimates of the batch mean and variance used during training become a less accurate approximation of the mean and variance used for testing. The influence of BN on the performance for different batch sizes is investigated in Section 3.

We observe that the calculation of the mean and variance across the batch makes the loss calculated for a particular example dependent on other examples of the same batch. This intrinsically ties the empirical loss (1) that is minimized to the choice of batch size. In this situation, the analysis of (7), (8) in Section 2.2 is only strictly applicable for a fixed value of the batch size used for BN. In some cases the overall SGD batch is split into smaller sub-batches used for BN. A common example is in the case of distributed processing, where BN is often implemented independently on each separate worker to reduce communication costs (e.g. Goyal et al. (2017)). For this scenario, the discussion relating to (7), (8) in Section 2.2 is directly applicable assuming a fixed batch size for BN. The effect of using different batch sizes for BN and for the weight update of the optimization algorithm will be explored in Section 3.6.

Different types of normalization have also been proposed (Ba et al., 2016; Salimans & Kingma, 2016; Arpit et al., 2016; Ioffe, 2017; Wu & He, 2018). In particular, *Batch Renormalization* (Ioffe, 2017) and *Group Normalization* (Wu & He, 2018) have reported improved performance for small batch sizes.

2.4 Other Related Work

Hoffer et al. (2017) have shown empirically that the reduced generalization performance of large batch training is connected to the reduced number of parameter updates over the same number of epochs (which corresponds to the same computation cost in number of gradient calculations). Hoffer et al. (2017) present evidence that it is possible to achieve the same generalization performance with large batch size, by increasing the training duration to perform the same number of SGD updates. Since from (3) or (5) the number of gradient calculations per parameter update is proportional to the batch size m, this implies an increase in computation proportional to m.

In order to reduce the computation cost for large batches due to the required longer training, Goyal et al. (2017) have investigated the possibility of limiting the training length to the same number of epochs. To this end, they have used larger learning rates with larger batch sizes based on the linear scaling rule, which from (5) is equivalent to keeping constant the value of $\tilde{\eta}$. As discussed in Section 2.2, this implies an increase of the variance of the weight update linearly with the batch size. The large batch training results of Goyal et al. (2017) report a reduction in generalization performance using the linear scaling rule with batch sizes up to 8192. However, Goyal et al. (2017) have found that the test performance of the small batch ($m = 256$) baseline could be recovered using a *gradual warm-up* strategy. We explore the impact of this strategy in Section 3.4 and find that it does not fully mitigate the degradation of the generalization performance with increased batch size.

Jastrzebski et al. (2017) claim that both the SGD generalization performance and training dynamics are controlled by a noise factor given by the ratio between the learning rate and the batch size, which corresponds to linearly scaling the learning rate. The paper also suggests that the invariance to the simultaneous rescaling of both the learning rate and the batch size breaks if the learning rate becomes too large or the batch size becomes too small. However, the evidence presented in this paper only shows that the linear scaling rule breaks when it is applied to progressively larger batch sizes.

Smith et al. (2017) have recently suggested using the linear scaling rule to increase the batch size instead of decreasing the learning rate during training. While this strategy simply results from a direct application of (4) and (6), and guarantees a constant mean value of the weight update per gradient calculation, as discussed in Section 2.2 decreasing the learning rate and increasing the batch size are only equivalent if one can assume (9), which does not generally hold for large batch sizes. It may however be approximately applicable in the last part of the convergence trajectory, after having reached the region corresponding to the final minimum. Finally, it is worth noting that in practice increasing the batch size during training is more difficult than decreasing the learning rate, since it may require, for example, modifying the computation graph.

3 Batch Training Performance

This section presents numerical results for the training performance of convolutional neural network architectures for a range of batch sizes m and base learning rates $\tilde{\eta}$.

The experiments have been performed on three different datasets: CIFAR-10, CIFAR-100 (Krizhevsky, 2009) and ImageNet (Krizhevsky et al., 2012). The CIFAR-10 and CIFAR-100 experiments have been run for different AlexNet and ResNet models (Krizhevsky et al., 2012; He et al., 2015a) exploring the main factors that affect generalization performance, including BN, data augmentation and network depth. Further experiments have investigated the training performance of the ResNet-50 model (He et al., 2015a) on the ImageNet dataset, to confirm significant conclusions on a more complex task.

3.1 Implementation Details

For all the experiments, training has been based on the standard SGD optimization. While SGD with momentum (Sutskever et al., 2013) has often been used to train the network models considered in this work (Krizhevsky et al., 2012; He et al., 2015a; Goyal et al., 2017), it has been shown that the optimum momentum coefficient is dependent on the batch size (Smith & Le, 2017; Goyal et al., 2017). For this reason momentum was not used, with the purpose of isolating the interaction between batch size and learning rate.

The numerical results for different batch sizes have been obtained by running experiments for a range of base learning rates $\tilde{\eta} = \eta/m$, from 2^{-12} to 2^{0}. For all learning rates and all batch sizes, training has been performed over the same number of epochs, which corresponds to a constant computational complexity (a constant number of gradient calculations).

For the experiments using the CIFAR-10 and CIFAR-100 datasets, we have investigated a reduced version of the AlexNet model of Krizhevsky et al. (2012), and the ResNet-8, ResNet-20 and ResNet-32 models as described in He et al. (2015a). The reduced AlexNet implementation uses convolutional layers with stride equal to 1, kernel sizes equal to $[5, 3, 3, 3, 3]$, number of channels per layer equal to $[16, 48, 96, 64, 64]$, max pool layers with 3×3 kernels and stride 2, and 256 hidden nodes per fully-connected layer. Unless stated otherwise, all the experiments have been run for a total of 82 epochs for CIFAR-10 and 164 epochs for CIFAR-100, with a learning rate schedule based on a reduction by a factor of 10 at 50% and at 75% of the total number of iterations. Weight decay has been also applied, with $\lambda = 5 \cdot 10^{-4}$ for the AlexNet model (Krizhevsky et al., 2012) and $\lambda = 10^{-4}$ for the ResNet models (He et al., 2015a). In all cases, the weights have been initialized using the method described in He et al. (2015b).

The CIFAR datasets are split into a 50,000 example training set and a 10,000 example test set (Krizhevsky, 2009), from which we have obtained the reported results. The experiments have been run with and without augmentation of the training data. This has been performed following the procedure of Lin et al. (2013) and He et al. (2015a), which pads the images by 4 pixels on each side, takes a random 32×32 crop, and then randomly applies a horizontal flip.

The ImageNet experiments were based on the ResNet-50 model of He et al. (2015a), with preprocessing that follows the method of Simonyan & Zisserman (2014) and He et al. (2015a). As with the CIFAR experiments, the weights have been initialized based on the method of He et al. (2015b), with a weight decay parameter $\lambda = 10^{-4}$. Training has been run for a total of 90 epochs, with a learning rate reduction by a factor of 10 at 30, 60 and 80 epochs, following the procedure of Goyal et al. (2017).

For all the results, the reported test or validation accuracy is the median of the final 5 epochs of training (Goyal et al., 2017).

3.2 Performance Without Batch Normalization

Figures 1–3 report the CIFAR-10 test performance of the reduced AlexNet, ResNet-8 and ResNet-20 models without BN or data augmentation, for different values of the batch size m and base learning rate $\tilde{\eta} = \eta/m$. The results show a clear performance improvement for progressively smaller batch sizes: for the reduced AlexNet model, the best test accuracy is obtained for batch sizes $m = 8$ or smaller, while for the ResNet models the best accuracy is obtained for batch sizes $m = 4$ and $m = 2$. This agrees with the expectation of Section 2.2.

The figures also indicate the presence of an optimum value of the base learning rate $\tilde{\eta} = \eta/m$ that delivers stable training only for batch sizes smaller than a critical value. Increasing the batch size with a constant base learning rate $\tilde{\eta}$ (equivalent to a linear scaling of η with the batch size m) also results in a reduction in generalization performance followed by a sharp decrease as training becomes unstable.

On the other hand, both the convergence robustness and the corresponding generalization performance consistently improve by reducing the batch size. From Figures 1–3 we observe that the use of small batch sizes corresponds to the largest range of learning rates that provides stable convergence. Comparing ResNet-8 and ResNet-20 it appears that that using smaller batch sizes reduces possible training difficulties associated with the use of deeper models. Not shown in the figures is

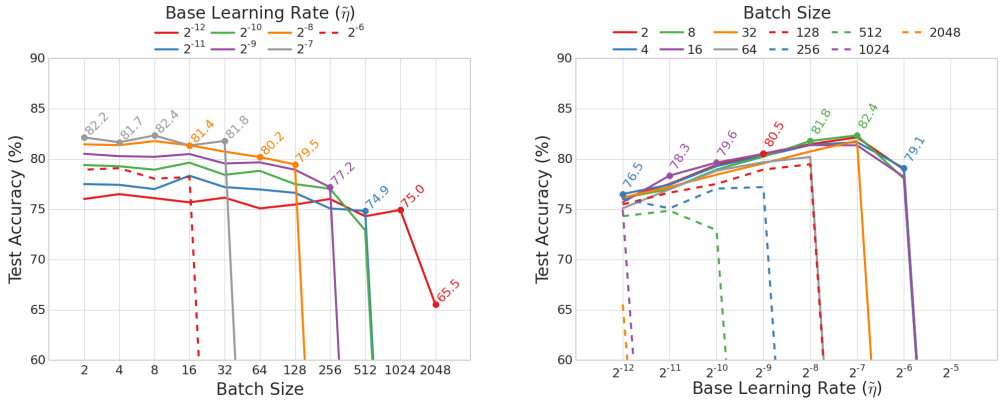


Figure 1: Test performance of reduced AlexNet model without BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-10 dataset without data augmentation.

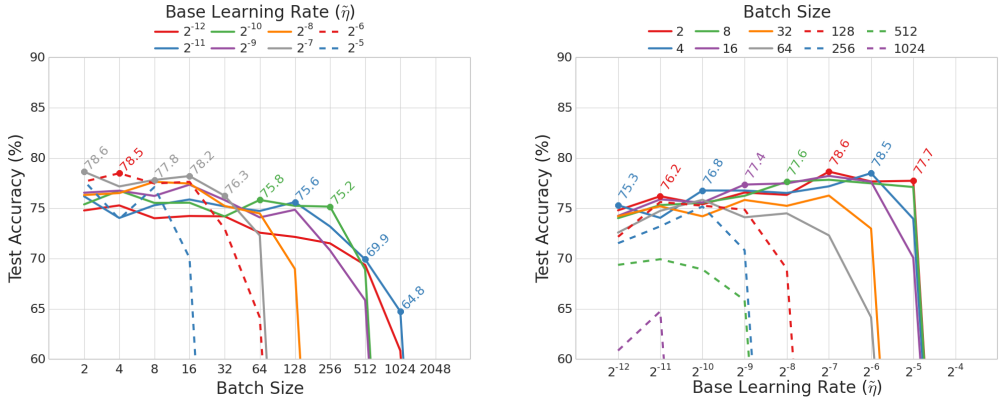


Figure 2: Test performance of ResNet-8 model without BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-10 dataset without data augmentation.

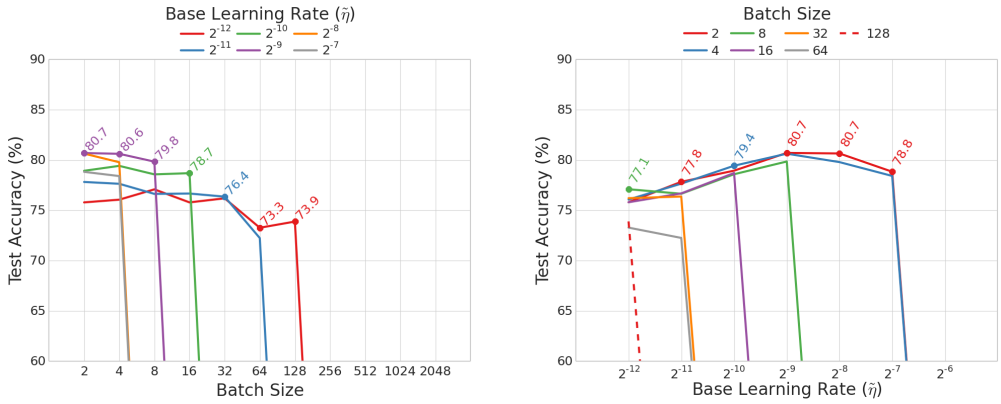


Figure 3: Test performance of ResNet-20 model without BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-10 dataset without data augmentation.

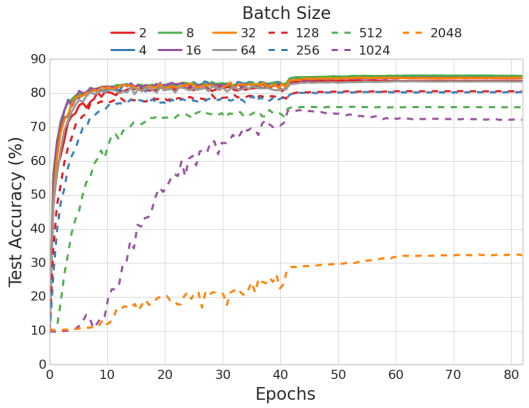


Figure 4: Convergence curves for ResNet-32 model with BN, for $\tilde{\eta} = 2^{-8}$ and for different values of the batch size. CIFAR-10 dataset without data augmentation.

the case of ResNet-32 without BN, for which all the CIFAR-10 experiments with base learning rate $\tilde{\eta}$ from 2^{-12} to 2^{0} and batch size from $m = 16$ to $m = 2048$ failed to converge, and learning was successful only for the cases of batch size $m = 2$, $m = 4$ and $m = 8$, with a best test performance of 76%.

3.3 Performance With Batch Normalization

We now present experimental results with BN. As discussed in Section 2.3, BN has been shown to significantly improve the training convergence and performance of deep networks.

Figure 4 shows the convergence curves for CIFAR-10, ResNet-32 training with BN, for different batch sizes. The curves have been obtained by keeping the base learning rate $\tilde{\eta}$ constant (which is equivalent to applying a linear scaling of η with the batch size m). From the plots, we observe similar convergence for some of the curves corresponding to the same value of $\tilde{\eta}$ with batch sizes in the range $4 \leq m \leq 64$, but with degraded training performance for larger batch sizes. Not shown here, analogous plots for higher learning rates show similar training divergence for large batch sizes.

Figures 5 and 6 show the CIFAR-10 test performance of the AlexNet model with BN but without data augmentation, for different values of batch size m and base learning rate $\tilde{\eta}$. BN is commonly applied to the activations in all convolutional and fully-connected layers. To assess the effect of BN on the different layers, here we have considered two different implementations of the reduced AlexNet model from Section 3.2: one with BN applied in both the convolutional and fully-connected layers, and a second with BN applied only in the convolutional layers. The results show that the use of BN provides a consistent improvement in test accuracy for a wide range of batch sizes, compared to the results without BN presented in Figure 1. As discussed in Section 2.3, while the sample size for the estimation of the batch statistics does reduce with the batch size, for the convolutional layers the batch samples are also taken over the feature maps height and width. For the fully-connected layers, the batch sample size for the calculation of the mean and variance is instead simply equal to the batch size – this can, unsurprisingly, cause issues for the smallest batch sizes investigated. The use of BN for the fully-connected layers improves the convergence for larger batch sizes. However, the best final performance is still obtained for batch sizes between $m = 8$ and $m = 64$, compared with best batch sizes between $m = 4$ and $m = 32$ for the case where BN is applied to only the convolutional layers (Figure 6). Both cases yield similar generalization performance.

Figures 7–9 report the training performance for CIFAR-10 (with and without data augmentation) and CIFAR-100 (with data augmentation) on the ResNet-32 model with BN, for different values of batch size m and base learning rate $\tilde{\eta}$. All results show a significant performance degradation for increasing values of the batch size, with the best results obtained for batch sizes $m = 4$ or $m = 8$.

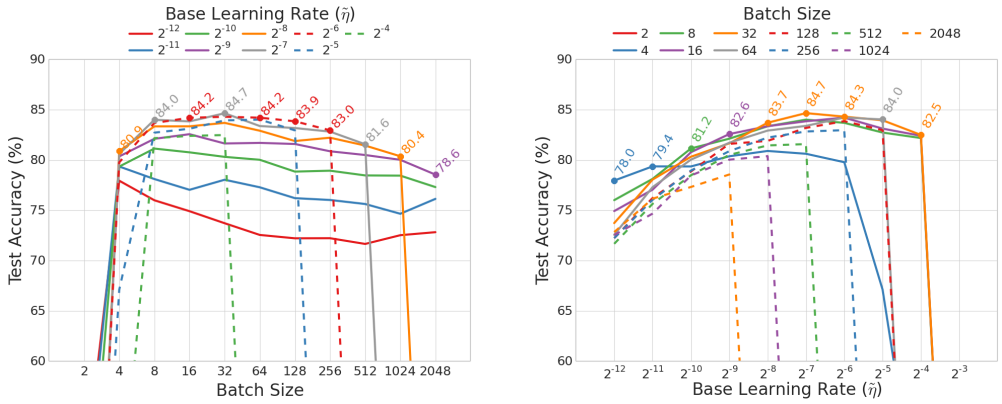


Figure 5: Test performance of reduced AlexNet model with BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-10 dataset without data augmentation.

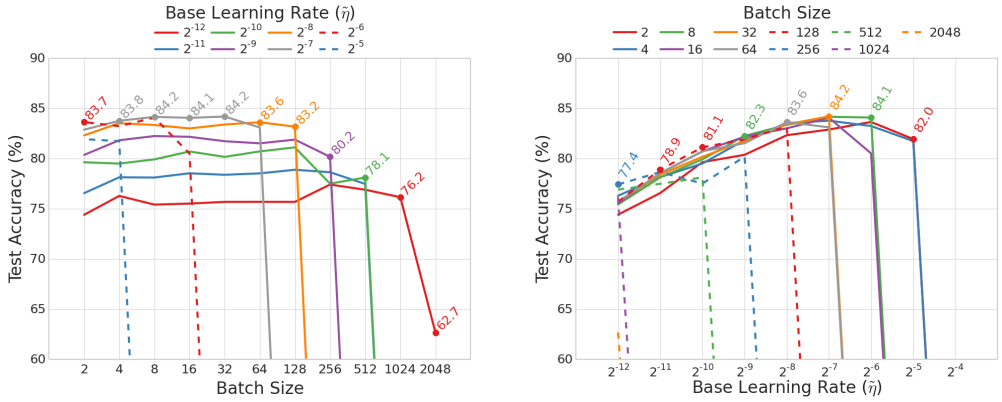


Figure 6: Test performance of reduced AlexNet model with BN only after the convolutions, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-10 dataset without data augmentation.

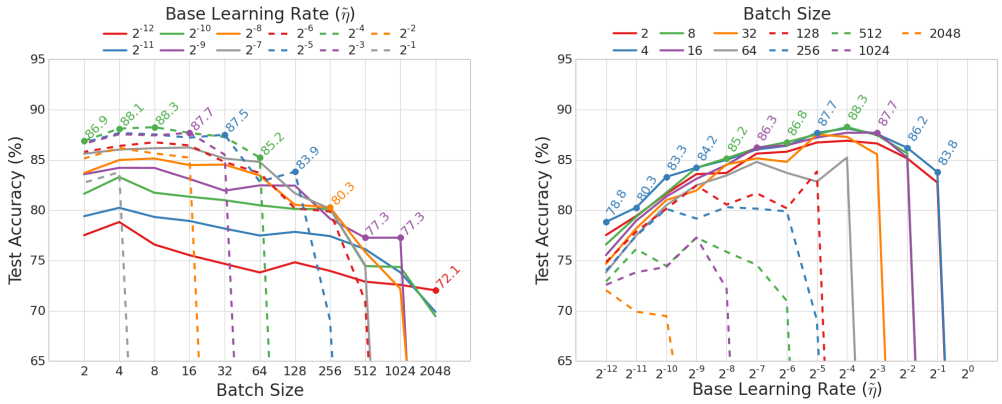


Figure 7: Test performance of ResNet-32 model with BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-10 dataset without data augmentation.

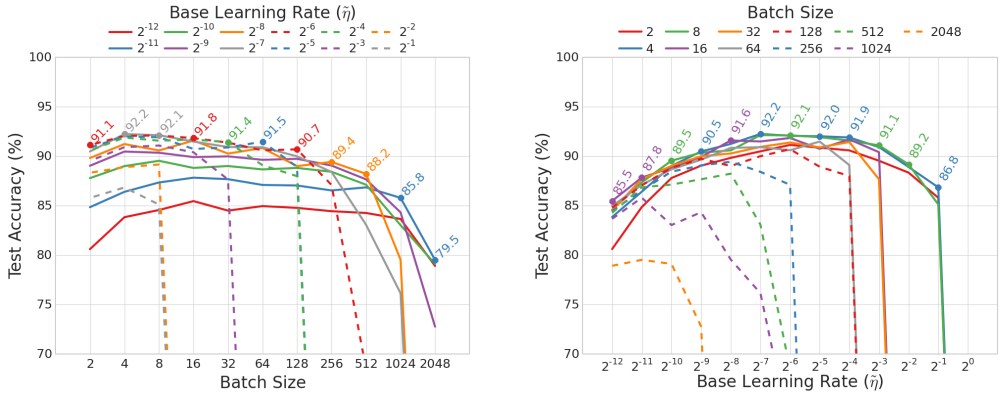


Figure 8: Test performance of ResNet-32 model with BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-10 dataset with data augmentation.

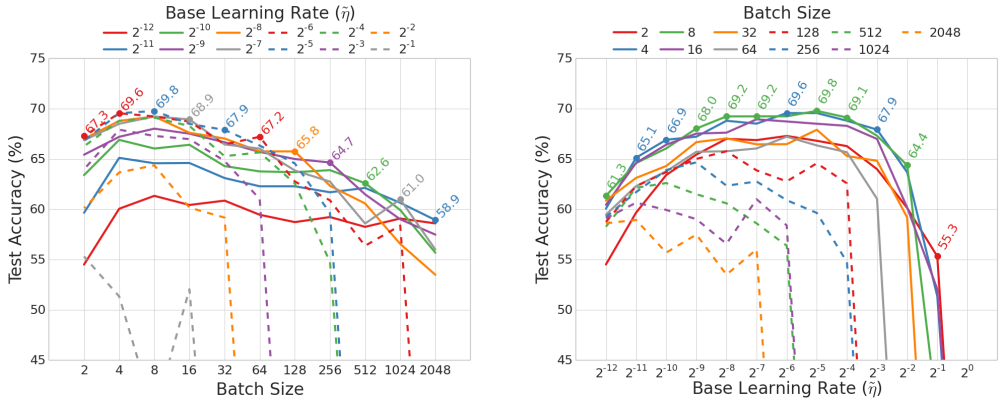


Figure 9: Test performance of ResNet-32 model with BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. CIFAR-100 dataset with data augmentation.

The results reported in Figures 7–9 indicate in each of the different cases a clear optimum value of base learning rate, which is only achievable for batch sizes $m = 16$ or smaller for CIFAR-10, and $m = 8$ or smaller for CIFAR-100.

3.4 Performance With Gradual Warm-up

As discussed in Section 2.4, warm-up strategies have previously been employed to improve the performance of large batch training (Goyal et al., 2017). The results reported in Goyal et al. (2017) have shown that the ImageNet validation accuracy for ResNet-50 for batch size $m = 256$ could be matched over the same number of epochs with batch sizes of up to 8192, by applying a suitable gradual warm-up procedure. In our experiments, the same warm-up strategy has been used to investigate a possible similar improvement for the case of small batch sizes.

Following the procedure of Goyal et al. (2017), the implementation of the gradual warm-up corresponds to the use of an initial learning rate of $\eta/32$, with a linear increase from $\eta/32$ to η over the first 5% of training. The standard learning rate schedule is then used for the rest of training. Here, the above gradual warm-up has been applied to the training of the ResNet-32 model, for both CIFAR-10 and CIFAR-100 datasets with BN and data augmentation. The corresponding performance results are reported in Figures 10 and 11.

As discussed in Section 2.4, the gradual warm-up helps to maintain consistent training dynamics in the early stages of optimization, where the approximation (9) is expected to be particularly weak. From Figures 10 and 11, the use of the gradual warm-up strategy proposed in Goyal et al. (2017) generally improves the training performance. However, the results also show that the best performance is still obtained for smaller batch sizes. For the CIFAR-10 results (Figure 10), the best test accuracy corresponds to batch sizes $m = 4$ and $m = 8$, although quite good results are maintained out to $m = 128$, while for CIFAR-100 (Figure 11) the best test performance is obtained for batch sizes between $m = 4$ and $m = 16$, with a steeper degradation for larger batch sizes.

3.5 ImageNet Performance

The ResNet-50 performance for the ImageNet dataset is reported in Figure 12, confirming again that the best performance is achieved with smaller batches. The best validation accuracy is obtained with batch sizes between $m = 16$ and $m = 64$.

The ImageNet results also show less predictable behaviour for larger batch sizes. In contrast to the CIFAR-10 and CIFAR-100 results, the performance degradation with larger batch sizes is observed to be particularly severe over a specific range of base learning rates $\tilde{\eta}$. Batch sizes $m = 16$ and $m = 32$ achieve the best performance over a continuous range of base learning rates, while the performance with larger batch sizes degrades precisely over the range of base learning rates that corresponds to the best overall performance. On the other hand, smaller batch sizes have shown stable and consistent convergence over the full range of learning rates.

Overall, the ImageNet results confirm that the use of small batch sizes provides performance advantages and reliable training convergence for an increased range of learning rates.

3.6 Different Batch Size for Weight Update and Batch Normalization

In the experimental results presented in Sections 3.2 to 3.5 (Figures 1–12), the same batch size has been used both for the SGD weight updates and for BN. We now consider the effect of using small sub-batches for BN, and larger batches for SGD. This is common practice for the case of data parallel distributed processing, where BN is often implemented independently on each individual worker, while the overall batch size for the SGD weight updates is the aggregate of the BN sub-batches across all workers.

Figure 13 shows the CIFAR-10 and CIFAR-100 results for different values of the overall SGD batch size and of the batch size for BN. Each curve corresponds to a fixed value of overall SGD batch size, where the experiments have been run using the base learning rate $\tilde{\eta}$ which delivered the best results in Section 3.3 (Figures 8 and 9) for the same SGD batch size.

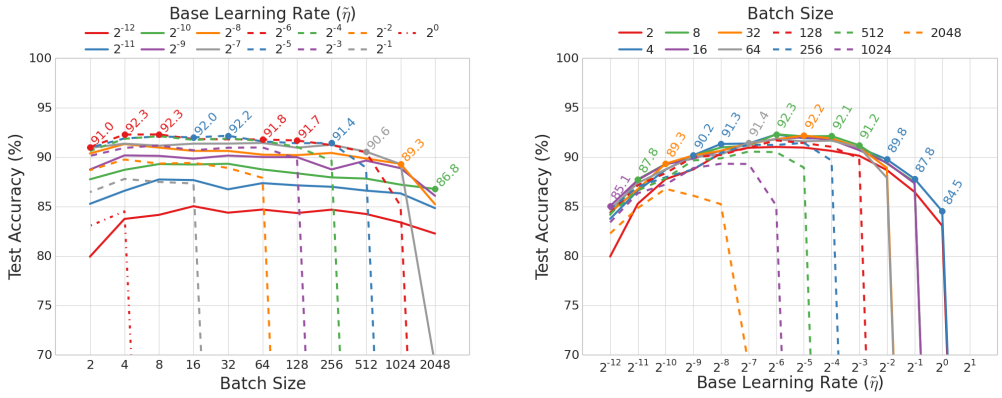


Figure 10: Test performance of ResNet-32 model with BN and with the warm-up strategy of Goyal et al. (2017). CIFAR-10 dataset with data augmentation.

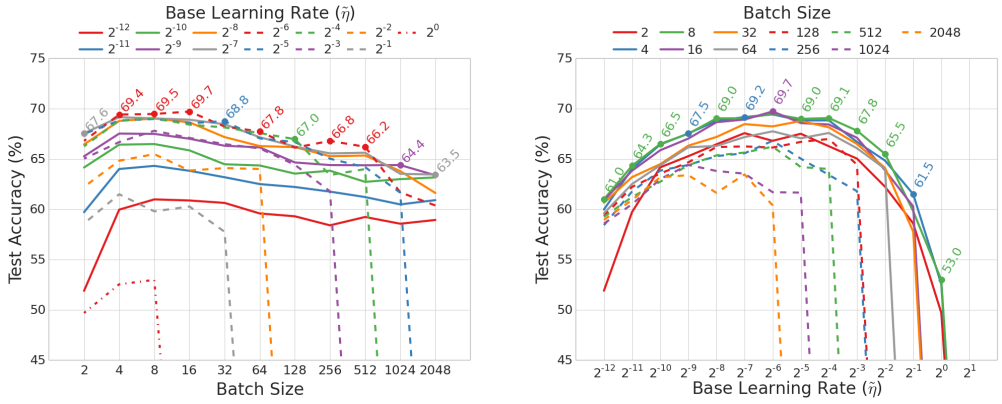


Figure 11: Test performance of ResNet-32 model with BN and with the warm-up strategy of Goyal et al. (2017). CIFAR-100 dataset with data augmentation.

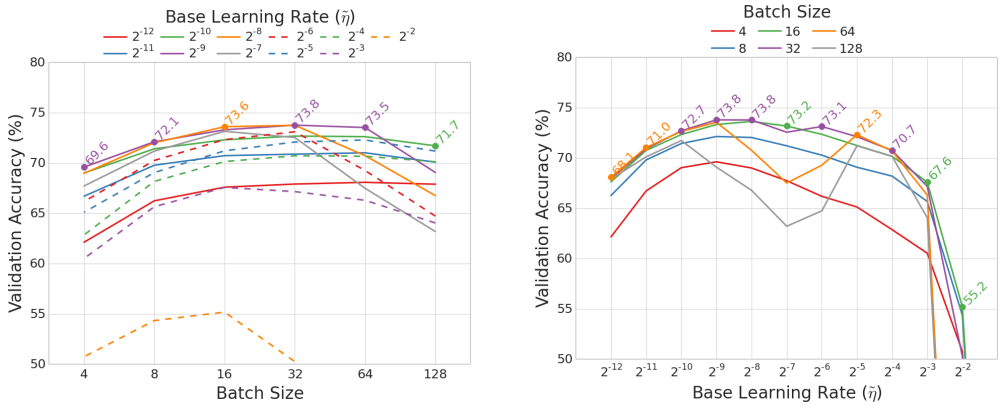
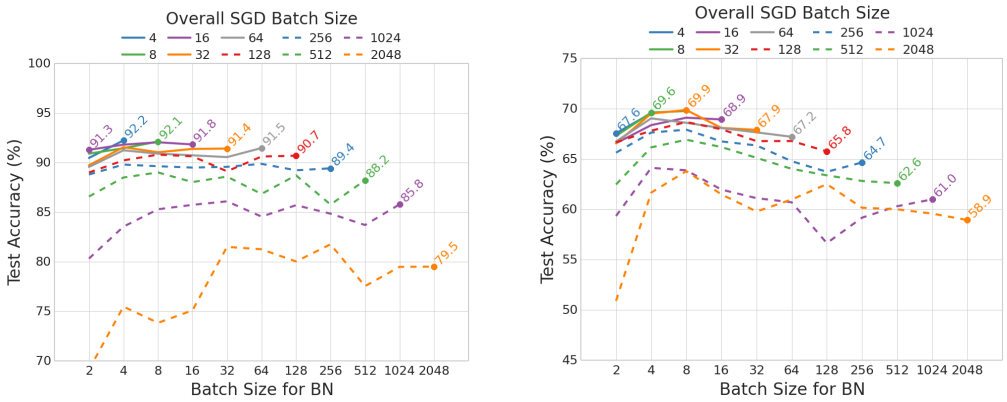


Figure 12: Top-1 validation performance of ResNet-50 model with BN, for different batch sizes and increasing values of $\tilde{\eta} = \eta/m$. ImageNet dataset with data augmentation. The reported results are the average of two runs.



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

Figure 13: Test performance of ResNet-32 model with BN, for different values of the overall SGD batch size and the batch size for BN. CIFAR-10 and CIFAR-100 datasets with data augmentation.

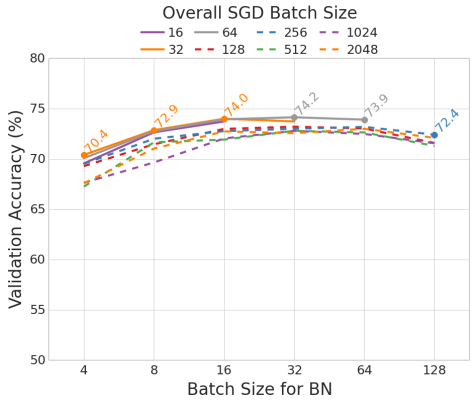


Figure 14: Top-1 validation performance of ResNet-50 model with BN, for different values of the overall SGD batch size and the batch size for BN. ImageNet dataset with data augmentation.

The results reported in Figure 13 show a general performance improvement by reducing the overall batch size for the SGD weight updates, in line with the analysis of Section 2. We also note that, for the CIFAR-100 results, the best test accuracy for a given overall batch size is consistently obtained when even smaller batches are used for BN. This is against the intuition that BN should benefit from estimating the normalization statistics over larger batches. This evidence suggests that, for a given overall batch size and fixed base learning rate, the best values of the batch size for BN are typically smaller that the batch size used for SGD. Moreover, the results indicate that the best values of the batch size for BN are only weakly related to the SGD batch size, possibly even independent of it. For example, a BN batch size $m = 4$ or $m = 8$ appears to give best results for all SGD batch sizes tested.

Figure 14 reports the corresponding ImageNet results for different values of the overall batch size for the SGD weight updates and of the batch size for BN. Again, each curve is for a fixed value of the SGD batch size, using the base learning rate $\tilde{\eta}$ which yielded the best results in Figure 12 with that batch size. For overall SGD batch sizes larger than 128, the best base learning rate has been chosen based on experiments with BN performed over batches of 128; the selected base learning rates were $\tilde{\eta} = 10^{-10}$ for overall batch sizes 256, 512 and 1024 and $\tilde{\eta} = 10^{-9}$ for overall batch size 2048.

The results presented in Figure 14 show that the performance with BN generally improves by reducing the overall batch size for the SGD weight updates. The collected data supports the conclusion

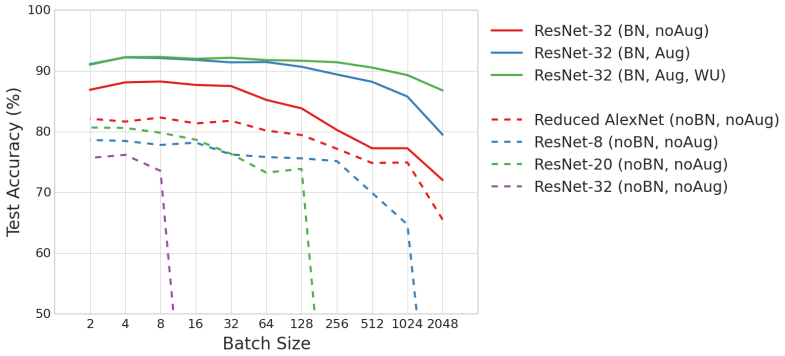


Figure 15: Summary of the best CIFAR-10 test performance achieved with different batch sizes, for the different network architectures presented in Section 3. (`BN`, `noBN`: with/without BN; `Aug`, `noAug`: with/without data augmentation; `WU`: with gradual warmup.)

that the possibility of achieving the best test performance depends on the use of a small batch size for the overall SGD optimization.

The plots also indicate that the best performance is obtained when smaller batches are used for BN in the range between 16 and 64, for all SGD batch sizes tested. From these results, the advantages of using a small batch size for both SGD weight updates and BN suggests that the best solution for a distributed implementation would be to use a small batch size per worker, and distribute *both* BN and SGD over multiple workers.

4 Discussion

The reported experiments have explored the training dynamics and generalization performance of small batch training for different datasets and neural networks.

Figure 15 summarizes the best CIFAR-10 results achieved for a selection of the network architectures discussed in Sections 3. The curves show that the best test performance is always achieved with a batch size below $m = 32$. In the cases without BN, the performance is seen to continue to improve down to batch size $m = 2$. This is in line with the analysis of Section 2.2, that suggests that, with smaller batch sizes, the possibility of using more up-to-date gradient information always has a positive impact on training. The operation with BN is affected by the issues highlighted in Section 2.3. However, the performance with BN appears to be maintained for batch sizes even as small as $m = 4$ and $m = 8$.

For the more difficult ImageNet training, Figure 12 shows that a similar trend can be identified. The best performance is achieved with batch sizes between $m = 16$ and $m = 64$, although for $m = 64$ SGD becomes acutely sensitive to the choice of learning rate. The performance for batch sizes $m \leq 8$ depends on the discussed effect of BN. As mentioned in Section 2.3, the issue of the BN performance for very small batch sizes can be addressed by adopting alternative normalization techniques like Group Normalization (Wu & He, 2018).

Goyal et al. (2017) have shown that the accuracy of ResNet-50 training on ImageNet for batch size 256 could be maintained with batch sizes of up to 8192 by using a gradual warm-up scheme, and with a BN batch size of 32 whatever the SGD batch size. This warm-up strategy has been also applied to the CIFAR-10 and CIFAR-100 cases investigated here. In our experiments, the use of a gradual warm-up did improve the performance of the large batch sizes, but did not fully recover the performance achieved with the smallest batch sizes.

Figure 16 gives an alternative view of the AlexNet and ResNet results presented in Section 3, showing the range of base learning rates $\tilde{\eta} = \eta/m$ that provide reliable convergence for each batch size. The results show that increasing the batch size progressively reduces the range of learning rates that provide stable convergence. This demonstrates how the increased variance in the weight update associated with the use of larger batch sizes can affect the robustness and stability of training, and

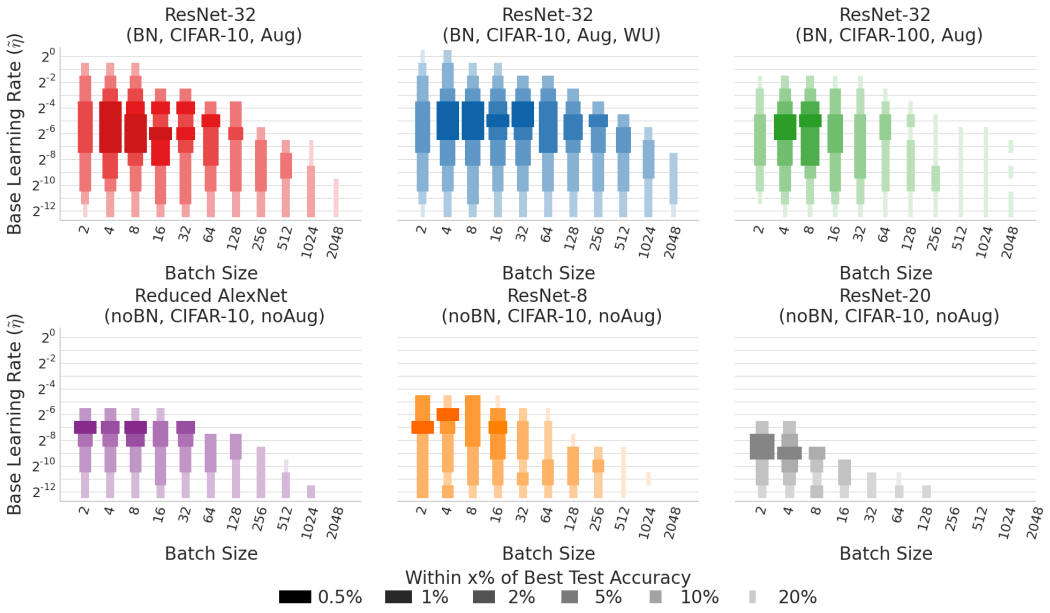


Figure 16: Summary of range of base learning rates $\tilde{\eta} = \eta/m$ that provide reliable convergence, for different network architectures. CIFAR-10 and CIFAR-100 datasets. (`BN`, `noBN`: with/without BN; `Aug`, `noAug`: with/without data augmentation; `WU`: with gradual warmup.)

highlights the limitations of the linear scaling rule. Furthermore, the range of results reported in Section 3 shows that, for each experiment, there is a clear optimal base learning rate $\tilde{\eta}$, but stable convergence can often only be achieved using this value of $\tilde{\eta}$ in combination with a small batch size. This suggests that performing a sweep of the learning rate for large batch sizes could easily lead to an incorrect conclusion on the optimal learning rate.

While the presented results advocate the use of small batch sizes to improve the convergence and accuracy, the use of small batch sizes also has the effect of reducing the computational parallelism available. This motivates the need to consider the trade-off between hardware efficiency and test performance.

We should also observe that the experiments performed in this paper refer to some of the most successful deep network architectures that have been recently proposed. These models have typically been designed under the presumption that large batches are necessary for good machine performance. The realisation that small batch training is preferable for both generalization performance and training stability should motivate re-examination of machine architectural choices which perform well with small batches; there exists at least the opportunity to exploit the small memory footprint of small batches. Moreover, it can be expected that small batch training may provide further benefits for model architectures that have inherent stability issues.

We have also investigated the performance with different batch sizes for SGD and BN, which has often been used to reduce communication in a distributed processing setting. The evidence presented in this work shows that in these circumstances the largest contributor to the performance was the value of the overall batch size for SGD. For the overall SGD batch sizes providing the best test accuracy, the best performance was achieved using values of batch size for BN between $m = 4$ and $m = 8$ for the CIFAR-10 and CIFAR-100 datasets, or between $m = 16$ and $m = 64$ for the ImageNet dataset. These results further confirm the need to consider the trade-off between accuracy and efficiency, based on the fact that to achieve the best test performance it is important to use a small overall batch size for SGD. In order to widely distribute the work, this implies that the best solution would be to distribute the implementation of both BN and SGD optimization over multiple workers.

5 CONCLUSIONS

We have presented an empirical study of the performance of mini-batch stochastic gradient descent, and reviewed the underlying theoretical assumptions relating training duration and learning rate scaling to mini-batch size.

The presented results confirm that using small batch sizes achieves the best training stability and generalization performance, for a given computational cost, across a wide range of experiments. In all cases the best results have been obtained with batch sizes $m = 32$ or smaller, often as small as $m = 2$ or $m = 4$.

With BN and larger datasets, larger batch sizes can be useful, up to batch size $m = 32$ or $m = 64$. However, these datasets would typically require a distributed implementation to avoid excessively long training. In these cases, the best solution would be to implement both BN and stochastic gradient optimization over multiple processors, which would imply the use of a small batch size per worker. We have also observed that the best values of the batch size for BN are often smaller than the overall SGD batch size.

The results also highlight the optimization difficulties associated with large batch sizes. The range of usable base learning rates significantly decreases for larger batch sizes, often to the extent that the optimal learning rate could not be used. We suggest that this can be attributed to a linear increase in the variance of the weight updates with the batch size.

Overall, the experimental results support the broad conclusion that using small batch sizes for training provides benefits both in terms of range of learning rates that provide stable convergence and achieved test performance for a given number of epochs.

REFERENCES

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 [stat.ML]*, 2016.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866 [cs.LG]*, 2017.

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

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838 [stat.ML]*, 2016.

Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous SGD. *arXiv preprint arXiv:1604.00981 [cs.LG]*, 2016.

Dipankar Das, Sasikanth Avancha, Dheevatsa Mudigere, Karthikeyan Vaidynathan, Srinivas Sridharan, Dhiraj Kalamkar, Bharat Kaul, and Pradeep Dubey. Distributed deep learning using synchronous stochastic gradient descent. *arXiv preprint arXiv:1602.06709 [cs.DC]*, 2016.

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, and Marc'Aurelio Ranzato. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems 25, NIPS 2012*, pp. 1223–1231, 2012.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677 [cs.CV]*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385 [cs.CV]*, 2015a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *arXiv preprint arXiv:1502.01852 [cs.CV]*, 2015b.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. *arXiv preprint arXiv:1705.08741 [stat.ML]*, 2017.

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *arXiv preprint arXiv:1702.03275 [cs.LG]*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of 32nd International Conference on Machine Learning, ICML15*, pp. 448–456, 2015.

Stanisław Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in SGD. *arXiv preprint arXiv:1711.04623 [cs.LG]*, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836 [cs.LG]*, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997 [cs.NE]*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25, NIPS 2012*, 2012.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pp. 9–48. Springer-Verlag, Berlin, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400 [cs.NE]*, 2013.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv preprint arXiv:1602.07868 [cs.LG]*, 2016.

Juergen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556 [cs.CV]*, 2014.

Samuel L. Smith and Quoc V. Le. A Bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451 [cs.LG]*, 2017.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489 [cs.LG]*, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *30th International Conference on Machine Learning, ICML 2013*, pp. 1139–1147, 2013.

D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

Tuxin Wu and Kaiming He. Group normalization. *arXiv preprint arXiv:1803.08494 [cs.CV]*, 2018.

A ADDITIONAL RESULTS WITH BATCH NORMALIZATION

Figures 17 and 18 show the effect of the training length in number of epochs on the results of Section 3.3. The CIFAR-10 ResNet-32 experiments with data augmentation have been repeated for half the number of epochs (41 epochs) and for twice the number of epochs (164 epochs). In both cases, the learning rate reductions have been implemented at 50% and at 75% of the total number of epochs.

The collected results show a consistent performance improvement for increasing training length. For lower base learning rates, the improvements are more pronounced, both for very small batch sizes and for large batch sizes. However, for the base learning rates corresponding to the best test accuracy, the benefits of longer training are important mainly for large batch sizes, in line with Hoffer et al. (2017). Even increasing the training length, and therefore the total computational cost, the performance of large batch training remains inferior compared to the small batch performance.

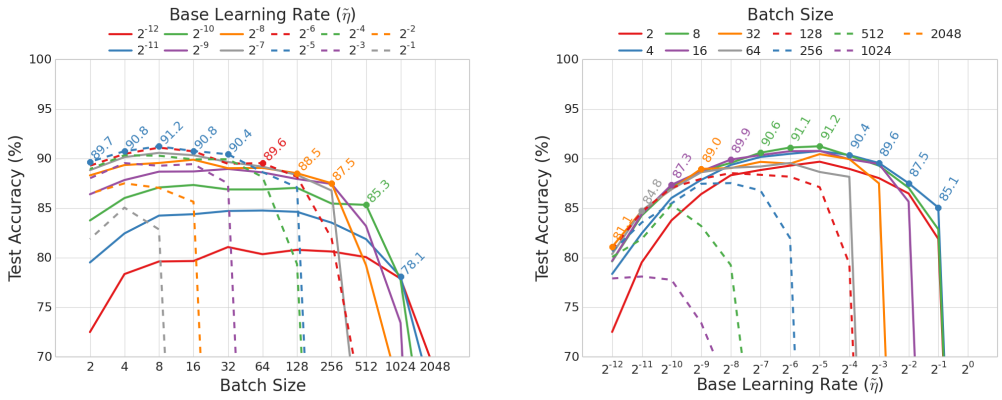


Figure 17: Test performance of ResNet-32 model with BN, for reduced training length in number of epochs. CIFAR-10 dataset with data augmentation.

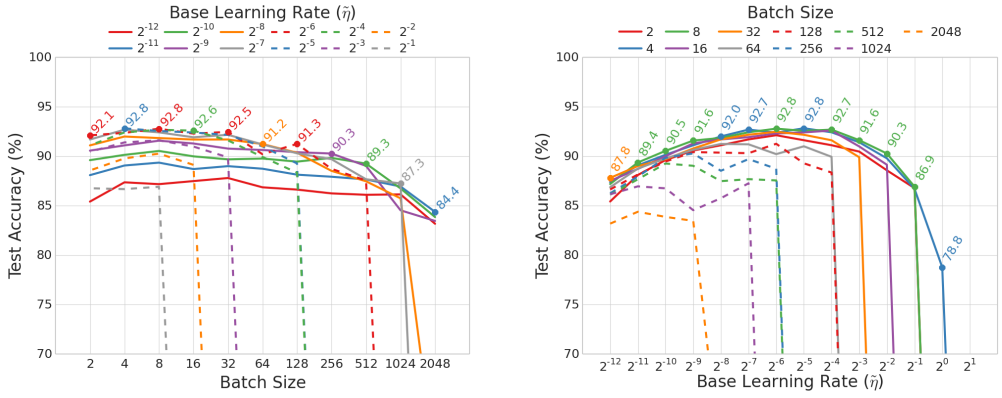


Figure 18: Test performance of ResNet-32 model with BN, for increased training length in number of epochs. CIFAR-10 dataset with data augmentation.