Overcoming Generalization Gap in Large Mini-Batch Training of Deep Neural Networks

Satish Kumar Jaiswal[1,a)] Kenjiro Taura[1,b)]

Abstract: Large mini-batch training is inevitable for speeding up the training of deep neural networks. However, it has been empirically found that neural networks trained with large mini-batch generalize poorly. Recent observations suggest that the minima for large mini-batch training tend to be in sharper regions, which are known to have poor generalization ability. In this research, we propose a method to close this generalization gap by introducing Gaussian noise in the gradient during the parameter update of Stochastic Gradient Descent (SGD).

1. Introduction

The neural network models are becoming complex and the datasets are growing bigger. For instance, the number of layers increased from 8 (AlexNet [1]) to 152 (Residual Nets [2]) just over a period of 4 years. ResNet50, which has just 50 layers, has approximately 25 million parameters. Moreover, the size of the dataset has also increased dramatically. ImageNet [3] has 1.28 million training images and 150, 000 test images, all classified under 1000 classes. Training such complex networks with massive dataset takes days or weeks. For example, training GoogleNet by ImageNet dataset on one NVIDIA K20 GPU takes 21 days [4].

One of the most commonly used technique to speed up the training of the deep neural network (DNN) is large mini-batch training. Table 1 summarizes the speed-up reported by various teams for training ResNet50 on ImageNet dataset. Akita *et al.* [5] were able able to train in 15 minutes by using an extremely large mini-batch of size 32, 768. However, this is not even 3% of the total dataset size which is 1.28 million in case of ImageNet. On further increasing the mini-batch size, they found degradation in the performance of the model. In general, a model trained with large mini-batch has been found to lose its generalization ability even if trained "without any budget or limits, until the loss function ceased to improve [6]". This phenomenon of decrease in the performance is known as generalization gap.

Having an insight into the origin of generalization gap, and moreover, discovering ways to close the gap will have significant practical importance. Particularly, it will help to improve the parallelism of deep neural network which demands the usage of very large mini-batch.

The empirical findings of Keskar *et al.* [6] suggests that the origin of generalization gap can be attributed to sharp minima. They found that models trained with large mini-batch had sharp minima and generalized poorly while the models trained with small mini-batch had flat minima and generalized well.

The reason why models trained using large mini-batch end up in sharp minima is still unknown. However, much research has been done in order to prevent a model from falling into sharp minima [7], [8]. Hoffer *et al.* [7] suggests that generalization gap can be closed by adopting square-root learning rate scaling, Ghost Batch-Normalization and sufficient number of iterations. On the other hand, Jastrzebsk *et al.* [8] suggests that with larger value of $\frac{\eta}{B}$, where η and B are learning rate and mini-batch size respectively, we can steer a large mini-batch training into flat minima.

The increased number of iterations in the method proposed by Hoffer *et al.* [7] is a concern because it will lead to poor speed-up. Similarly, to obtain a higher value of $\frac{\eta}{B}$ ratio as proposed by Jastrzebsk *et al.* [8], we have to use a relatively large learning rate which might lead to divergence.

We propose a method to close the generalization gap by adding white noise to the gradients of the loss function. The idea is inspired from the fact that the covariance of the gradients decrease with increasing mini-batch size. We hypothesize that the addition of white noise will compensate the decrement in the covariance. We conducted preliminary experiments and found that the addition of noise helped to close the gap.

2. Background and Related Work

2.1 Training a DNN

The training of a DNN is an optimization problem in which we try to find an empirical optimal parameter $\theta^\star$ such that an empirical loss $L(\theta)$ [13] over the entire dataset is minimized as expressed in Equation 1.

$$\underset{\theta \in \mathbf{\Theta}}{\text{Minimize}} \quad L(\theta) = \frac{1}{N} \sum_{i=1}^{N} l(\theta, \mathbf{x}_i, \mathbf{y}_i), \tag{1}$$

where $l(\theta, \mathbf{x}_i, \mathbf{y}_i)$ is the loss incurred or discrepancy in predicting a single sample $(\mathbf{x}_i, \mathbf{y}_i)$ for a given parameter θ and N is the total

[1] The University of Tokyo 7-3-1 Hongo Bunkyo-ku, Tokyo 113–0033, Japan
[a)] ictish@gmail.com
[b)] tau@eidos.ic.i.u-tokyo.ac.jp

ⓒ 2018 Information Processing Society of Japan

Table 1 90 epoch training time and accuracy for ResNet50 on ImageNet reported by various teams

Team	Hardware	Software	Mini-batch size	Time	Accuracy
He *et al.* [9]	Tesla P100 x 8	Caffe	256	29 hr	75.3%
Goyal *et al.* [10]	Tesla P100 x 256	Caffe2	8,192	1 hr	76.3%
Codreanu *et al.* [11]	KNL 7250 x 720	Intel Caffe	11,520	62 min	75.0%
You *et al.* [12]	Xeon 8160 x 1600	Intel Caffe	16,000	31 min	75.3%
Akiba *et al.* [5]	Tesla P100 x 1024	Chainer	32,768	15 min	74.9%

number of samples in the dataset.

The stochastic gradient descent (SGD) is commonly used to optimize the problem in Equation 1. The parameter update in Stochastic Gradient Descent (SGD) is done according to Equation 2.

$$\theta_{t+1} = \theta_t - \frac{\eta}{B}\sum_{i=1}^{N}\nabla l(\theta_t, \mathbf{x}_i, \mathbf{y}_i), \qquad (2)$$

where B is the mini-batch size.

2.2 Large Mini-Batch Training

During the optimization of the DNN according to SGD, the size of the mini-batch B is usually very small as compared to the whole dataset. While the use of appropriately small mini-batch size gives better generalization ability, it leads to significantly slow training. In order to speed up the training, various teams, as tabulated in Table 1, have used large mini-batch.

One of the most significant important work in the field of large mini-batch training is from Goyal *et al.* [10] who were able to decrease the training time from 29 hours to 1 hour by increasing the mini-batch size from 256 to $8,192$. They suggested two important techniques for the large mini-batch training which have been adopted by various other following works. These two techniques are as follows:

(1) Linear Scaling of Learning Rate: This rule states that if we increase the mini-batch size k times, we should increase the learning rate by k times as well. The k times scaling of the learning rate compensates for k different parameter updates that would have been made with k smaller mini-batches.

(2) Gradual Warm-up: This rule states that we have to linearly increase the learning rate from a lower value to its scaled value over the first few epochs. The smaller value of the learning rate in the beginning of the training ensures stability.

The recent state-of-the-art large mini-batch training by Akiba *et al.* [5] were able to increase the mini-batch size to $32,768$ which is merely 3% of the whole dataset. On further increasing, the mini-batch size they reported a decrement in the generalization ability of the model.

2.3 Minima and Generalization Gap

2.3.1 Flat Minima and Sharp Minima

Figure 1 depicts a conceptual sketch of flat and sharp minima. A minima is said to be flat if the rate of change of the function in its neighbourhood is small, otherwise it is a sharp minima. More precisely, the eigenvalue of the Hessian of the function at a flat minima is smaller than that of a sharp minima.

In Figure 1, the solid line is the training function while the dotted line is the testing function. At sharp minima, the gap between the training function and testing function is bigger than that at the flat minima. This gap is the generalization gap.

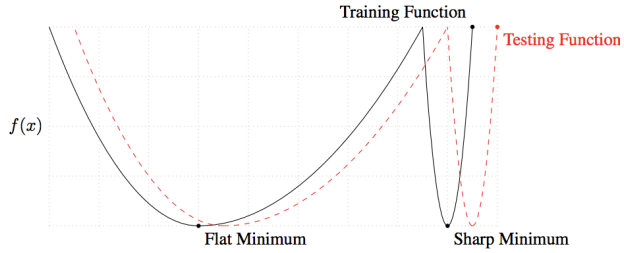


Fig. 1 A conceptual sketch of sharp and flat minima. The vertical axis is loss and horizontal axis is parameters. [6]

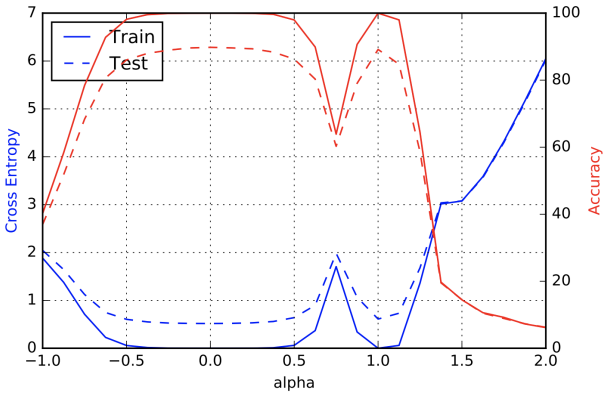


Fig. 2 A parametric plot to identify the nature of minima. $\alpha = 0$ corresponds to minima found by small mini-batch training while $\alpha = 1$ corresponds to minima found by large mini-batch training. [6]

2.3.2 Parameter Plot to Identify the Nature of Minima

A parametric plot as shown in Figure 2 can be used to analyze the nature of the minima in its neighbourhood. The parameter (θ) of the model which is used to obtain the cross entropy loss and accuracy in Figure 2 is calculated as $\theta = (1-\alpha)\theta_S + \alpha\theta_L, \alpha \in [-1, 2]$ where θ_S and θ_L represent model parameters for small mini-batch and large mini-batch training respectively. In Figure 2, we can clearly see that the rate of change is relatively high in the neighbourhood of large mini-batch training. This suggests that large mini-batch training ends up in sharp minima.

2.3.3 Origin of Generalization Gap

In general, a model trained with large mini-batch has been found to lose its generalization ability even if trained "without any budget or limits, until the loss function ceased to improve [6]". This phenomenon of decrease in the performance is known as generalization gap.

The insight into the origin of the generalization, and moreover, discovering ways to close this gap will have a significant practical importance, especially for large mini-batch training. Keskar *et al.* [6] empirically made the following observations:

(1) Models trained with large mini-batch ended up in sharp min-

ⓒ 2018 Information Processing Society of Japan

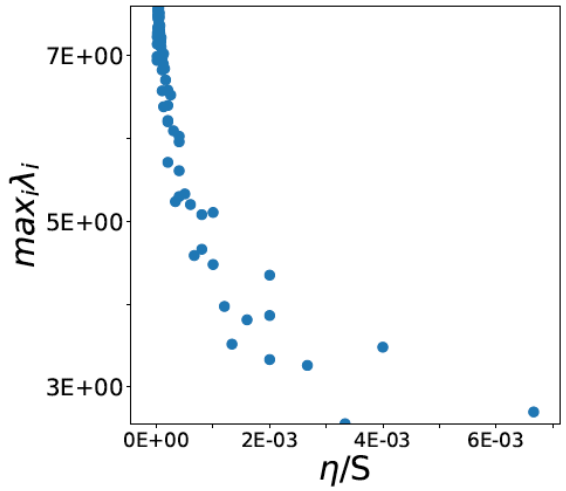


Fig. 3 Impact of SGD with ratio of learning rate η and batch-size S on flatness of final minima for a 4 layer ReLU MLP architecture on FashionMNIST dataset. For higher value of the ratio, SGD finds flat minima. [8]

ima while models trained with small mini-batch ended up in flat minima.

(2) Models trained with large mini-batch howed poor generalization ability than those trained with small mini-batch

Based on the above observations, they suggested that generalization gap can be attributed to sharp minima.

2.4 Related Work for closing Generalization Gap

Hoffer *et al.* [7] have proposed to use square root scaling of learning rate, Ghost Batch-Normalization and sufficient iterations to close the generalization gap. In their experiments conducted on MNIST, CIFAR10 and ImageNet dataset, they were able to decrease the generalization gap from 5% to $1\% - 2\%$. However, the increased number of iterations to overcome the generalization may not be useful to obtain speed-up from parallelization of large mini-batch training.

Jastrzebsk *et al.* [8] found that larger value of $\frac{\eta}{B}$ steers SGD into flat minima which are associated with better generalization ability. They measured the spectral norm (i.e. largest eigenvalue denoted by $max_j\lambda_j$, where λ_j are the eigenvalues of the Hessian) of the Hessian of the loss function at the minima for models trained with different values of $\frac{\eta}{B}$ as shown in Figure 3.

Increasing the ratio $\frac{\eta}{B}$ is not an easy task in practice. This is because to increase the ratio $\frac{\eta}{B}$, the scale factor for the learning rate has to be more than that of the batch size. The training of the DNN with larger learning rate is not stable and might not always result in convergence.

3. Proposed Method

3.1 Proposed Method

There is a trade-off between the number of increased iterations and speed-up in the method proposed by Hoffer *et al.* [7]. Similarly, the learning rate in the method proposed by Jastrzebsk *et al.* [8] may be large enough to converge to the minima or we might require additional efforts. Due to these limitations in the previous works, we propose a new method to close the generalization gap by adding white noise to the gradients. The idea of adding noise to the gradient itself is not new **?**, the application of the concept to overcome generalization gap is unique to our best knowledge. The proposed method can be put simply put into two steps:

(1) Linearly scale the learning rate by the same scale factor as that of the mini-batch.

(2) Add a white noise to the gradients calculated from the mini-batch according to the distribution $\mathcal{N}(\mathbf{0}, \sigma_t^2\mathbf{I})$ where $\sigma_{t^2} = \frac{\alpha}{(1+t)^\gamma}$ decays exponentially with time. α and γ are the hyper parameters which need to be tuned. In our experiments, we use $\alpha = 0.00001$ and $\gamma = 0.55$.

3.2 Motivation

Our main reason behind deliberately adding white noise to the gradient is that the covariance of the gradients decreases when the mini-batch size increases. The decrease in the covariance makes the gradients less stochastic. Since stochasticity in the gradient helps in the exploration of the parameter space, it is an important feature for the success of small mini-batch training. It might even help SGD with small mini-batch escape from sharp local minima. We add noise to the gradient to compensate for the decrement in the covariance.

3.3 Covariance of the Gradients of Mini-Batch

Let, $g_i(\theta_t)$ be the gradient of loss on the i^{th} sample, $g_B(\theta_t)$ be the average gradient of the mini-batch, $\bar{g}(\theta_t)$ and $\Sigma(\theta_t)$ be the average gradient and covariance of the entire dataset. For the sake of simplicity we just write them as g_i, g_B, $\bar{g}$ and Σ. Then the following equations will hold.

$$\bar{g} = \frac{1}{N}\sum_{i=1}^{N}\nabla l(\theta_t, \mathbf{x}_i, \mathbf{y}_i) \tag{3}$$

$$g_B = \frac{1}{B}\sum_{i=1}^{B}\nabla l(\theta_t, \mathbf{x}_i, \mathbf{y}_i) \tag{4}$$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(g_i - \bar{g})(g_i - \bar{g})^\top \tag{5}$$

Statistically the mini-batch will follow a sampling distribution. In case of large mini-batch, the central limit theorem will hold. As a result of which, g_B will follow a normal distribution with average $\bar{g}$ and covariance $\frac{\Sigma}{B}$.

$$g_B \sim \mathcal{N}(\bar{g}, \frac{\Sigma}{B}).$$

Alternately, we can also express the above expression as

$$g_B = \bar{g} + \frac{1}{\sqrt{B}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \Sigma), \tag{6}$$

where ϵ is the noise in the gradient which follows the distribution $\mathcal{N}(0, \Sigma)$. Then finally the parameter update in Equation 2 can be rewritten as

$$\theta_{t+1} = \theta_t - \eta_t\bar{g} + \frac{\eta_t}{\sqrt{B}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \Sigma). \tag{7}$$

The third term on the right hand side of Equation 7 is the noise

ⓒ 2018 Information Processing Society of Japan

term which makes SGD stochastic, and is responsible for the exploration of the parameter space or helps in escaping from the local minima. The two most important observations made from Equation 7 are as follows:

(1) When we increase the mini-batch size the noise decreases by square root term.

(2) This noise is time dependent since the covariance of the entire dataset is time dependent.

Based on the above two observations, we propose to add a white noise to compensate for the decrement in the noise due to decrement in the covariance when using large mini-batch. Similarly, we also propose that the covariance of the white noise should be time dependent and decay with time. The optimal value of the covariance of the noise itself is a hyperparamter that needs to be tuned.

4. Preliminary Experiment and Result

4.1 Objective

The goal of our experiment was to provide a proof of concept for our proposed method. Moreover, during the course of experiments, we also reproduced some of the experiments conducted in the related works.

4.2 Experimental Setup

In all of our experiments, we have used Caltech101 dataset to train ResNet50. ResNet50 is convolutional neural network (CNN) which consists of 50 layers and has been extensively used in state-of-the-art [5] training of large mini-batches. Similarly, Caltech101 is a small dataset of merely 9147 images but consists of untouched real-world images. It is very suitable for verifying complex ideas in short period of time. We split this dataset into training and validation dataset by 3 : 1 ratio. The training set consists of 6903 images. In most of our experiment, the training accuracy nearly reached 100%. This is due to the fact that over-parametrized CNN tend to overfit the data. This phenomenon was also reported by Jastrzebsk *et al.* [8].

We conducted our experiment on the *arcturus*-server of Taura laboratory. The details of this server are as follows:

- CPU: Intel Xeon E5-2699 v4 2.20GHz
- Memory: DIMM DDR4 2400 MHz 32GB x 16
- GPU: NVIDIA GP100GL Tesla P100 SMX2 16GB]x2

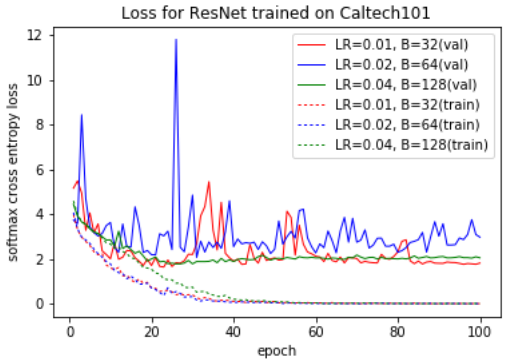
All of the codes for our experiments were written using chainer.

4.3 Scaling the Mini-Batch Size Without Noise

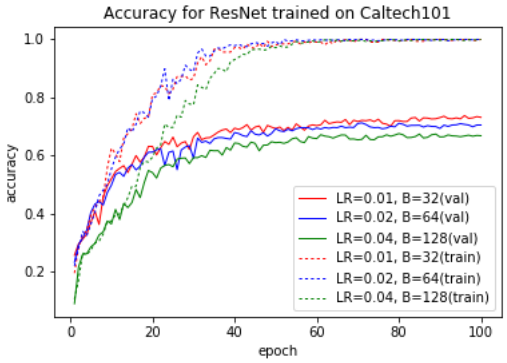
We trained ResNet50 on Caltech101 dataset for $\frac{\eta}{B} \in \{\frac{0.01}{32}, \frac{0.02}{64} \text{and } \frac{0.04}{128}\}$ by decaying the learning rates by 10% at 25^{th}, 40^{th} and every following 10^{th} epoch. Although the mini-batch size of 128 (1.85%) in this experiment appears to be small, it in fact is relatively large mini-batch as compared to the whole training dataset. Goyal *et al.* [10] had used a mini-batch of size 8192 which is just 0.66% of the entire ImageNet dataset.

The results of the experiment are as shown in Figure 4. The gap between the training (dotted curve) and the validation (solid curve) accuracy is huge 4(b). This leaves us with thinking if this is the case of overfitting. Had it been a case of overfitting, the validation loss curve 4(a) would have been increased. Since it stays

*[t]



(a) Loss curve.



(b) Accuracy curve. Generalization gap increases with increasing batch size. It is smallest for red curve (B=32) and biggest for green curve (B=64).

Fig. 4 Training ResNet50 on Caltech101 without noise. The solid and dotted curves are validation and training curves respectively.

the same without much fluctuations, we can say that this is not the case of overfitting. Morever, Jastrzebsk *et al.* [8] have also reported the same phenomenon.

One of the important observation from this result is that the generalization gap increases with increasing batch-size 4(b) from red curve to green curve. The accuracy for red, blue and green curves are 74.45%, 70.29% and 67.86% respectively.

Inspired by the work of Hoffer *et al.* [7], we trained the network for sufficient number of iterations to see if the generalization gap decreased. We obtained a result as shown in Figure 5. We state here that making a comparison from this experiment with the works of Hoffer *et al.* [7] is unfair because we used linear scaling rule instead of square root scaling and performed normal Batch-Normalization instead of Ghost Batch-Normalization. However, we can say that under our experiment conditions, the generalization did not improve even if we trained it for sufficient number of iterations.

In order to investigate the inherent cause of this generalization gap, we also performed an experiment to see what was happening around the minima. We made a parametric plot as explained in Section 2.3.2. The result of the plot is as shown in Figure 6. The plot suggests that minima in either cases are similar. This observation is in agreement with the works of Jastrzebsk *et al.* [8] who suggests that for the same value of $\frac{\eta}{B}$ the nature of the minima should be similar.

All of them probably may be sharp minima given a huge differ-

ⓒ 2018 Information Processing Society of Japan

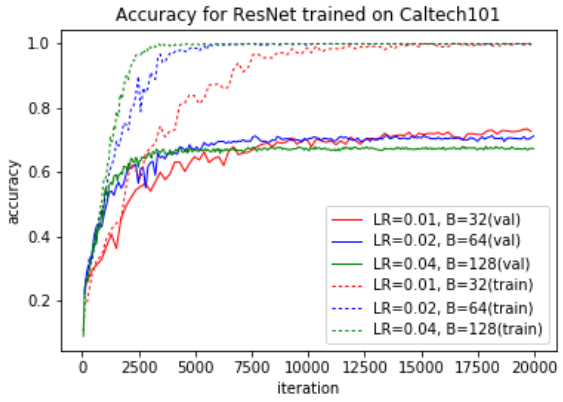


Fig. 5 Accuracy curve for ResNet50 trained on Caltech101. Even if we trained it for suffiecient number of iterations, we did not see any improvement in the performance.

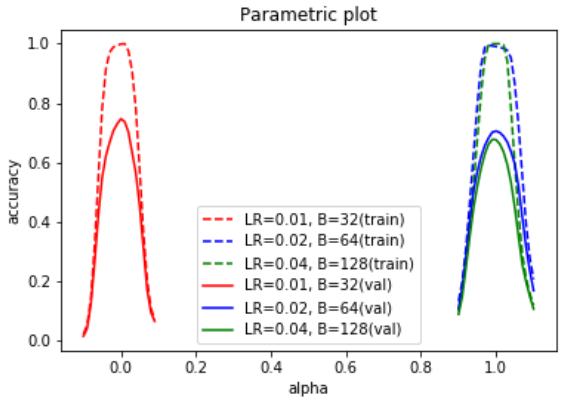


Fig. 6 Parametric plot. The solid and the dotted lines represent validation and train accuracy respectively. Red curve (B=32) is for small mini-batch at $\alpha = 0$, blue (B=64) and green (B=128) are for large mini-batch at $\alpha = 1$.

ence between training and validation accuracy. The point here is that even if the minima are the same they generalize differently. In the light of this result, we hypothesize that Goyal *et al.* [10] also might have ended up in the similar minima (because of proportionate scaling of learning rate and batch size) but with different generalization gap.

4.4 Scaling the Mini-Batch Size With Noise

From the experiment in the previous section, we found that we might get a generalization gap even if we ended up in the similar kind of minima. Adding a white noise to the gradients, we were able to close this gap.

In this experiment, we simply added a white noise from the distribution $\mathcal{N}(0, \frac{0.00001}{(1+t)^{0.55}})$ to the gradients before making an update to the parameter. All the other settings were similar to the experiment conducted in the previous section. We performed the experiment twice for $B = 128$ and once for $B = 64$.

Figure 7 shows the result of the experiment conducted with addition of noise to the gradient for the mini-batch size of 128. The black solid line is for the experiment conducted with noise and the green solid line is for the experiment conducted without noise. There is an overall improvement in the learning dynamics throughout all the epochs. It also surpasses the performance of

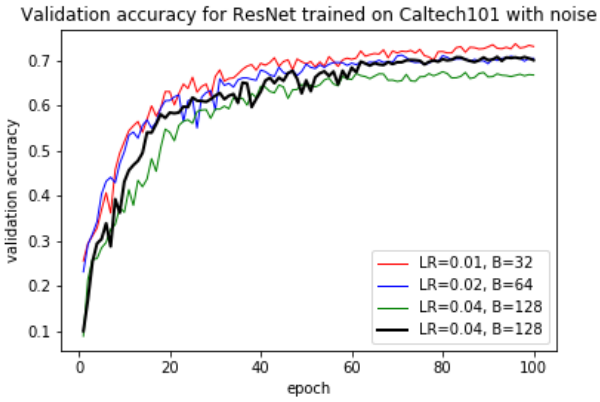


Fig. 7 Effect of noise on generalization gap. Noise helps to close the gap. The black curve is obtained from the noise experiment while the coloured curves from experiment without noise.

Table 2 Effect of noise in generalization gap

	B=64	B=128
No noise	70.29%	67.86%
Noise	72.07%	71.01 ± (0.42)%

$B = 64$. The results are tabulated in Table 2.

The result of this experiment is motivating because it is in agreement with our claim that the decrement in the covariance of the gradients with increasing mini-batch size could be compensated by the addition of white noise in the gradient. Motivated with this result, we would like to further investigate about why it works and how it can be applied for large mini-batch training.

5. Conclusion and Future Work

Large mini-batch training is inevitable for speeding up the training of DNN. However, it has been found empirically that the performance of the model decreases when large mini-batch is used for the training. Keskar *et al.* [6] empirically found that minima of the models trained with large mini-batch are sharp minima which are associated with poor generalization ability. Jastrzebsk *et al.* [8] suggested that we can avoid sharp minima during the training by increasing the ratio $\frac{\eta}{B}$.

In the linear learning rate rule for large mini-batch training [10], the ratio $\frac{\eta}{B}$ remains constant. We showed that the generalization gap in linear learning rate scaling, when the ratio $\frac{\eta}{B}$ remains constant and probably the nature of the minima are the same, can be decreased by adding noise to the gradients.

We are considering the following direction for our research in the future:

(1) Our proposed method is only verified against ResNet50 on Caltech101. We will verify its validity on a variety of other networks like MLP, GoogleNet and AlexNet on datasets like MNIST, CIFAR10, CIFAR100, Caltech256 and ImageNet.
(2) We will conduct a very large mini-batch training (8192, 16384, 32768) using ResNet50 on ImageNet-1k.
(3) The optimal amount of noise that needs to be added to the gradient for optimal training is another investigation for the future.

ⓒ 2018 Information Processing Society of Japan

References

[1] Krizhevsky, A., Sutskever, I. and Hinton, G. E.: Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, pp. 1097–1105 (2012).
[2] He, K., Zhang, X., Ren, S. and Sun, J.: Deep Residual Learning for Image Recognition, *CoRR*, Vol. abs/1512.03385 (online), available from ⟨http://arxiv.org/abs/1512.03385⟩ (2015).
[3] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L.: Imagenet: A large-scale hierarchical image database, *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, pp. 248–255 (2009).
[4] Iandola, F. N., Moskewicz, M. W., Ashraf, K. and Keutzer, K.: Firecaffe: near-linear acceleration of deep neural network training on compute clusters, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2592–2600 (2016).
[5] Akiba, T., Suzuki, S. and Fukuda, K.: Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes, *CoRR*, Vol. abs/1711.04325 (online), available from ⟨http://arxiv.org/abs/1711.04325⟩ (2017).
[6] 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, *CoRR*, Vol. abs/1609.04836 (online), available from ⟨http://arxiv.org/abs/1609.04836⟩ (2016).
[7] Hoffer, E., Hubara, I. and Soudry, D.: Train longer, generalize better: closing the generalization gap in large batch training of neural networks, *Advances in Neural Information Processing Systems*, pp. 1729–1739 (2017).
[8] Jastrz?bski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y. and Storkey, A.: Finding Flatter Minima with SGD, *ICLR 2018 Workshop* (2018).
[9] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (2016).
[10] Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. and He, K.: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, *CoRR*, Vol. abs/1706.02677 (online), available from ⟨http://arxiv.org/abs/1706.02677⟩ (2017).
[11] Codreanu, V., Podareanu, D. and Saletore, V.: Achieving deep learning training in less than 40 minutes on imagenet-1k., Coperatie SURF U.A. (online), available from ⟨https://blog.surf.nl/en/imagenet-1k-training-on-intel-xeon-phi-in-less-than-40-minutes/⟩ (accessed 2017-12-06).
[12] You, Y., Zhang, Z., Hsieh, C., Demmel, J. and Keutzer, K.: ImageNet training in minutes, *CoRR, abs/1709.05011* (2017).
[13] Murata, N. and Amari, S.-i.: Statistical analysis of learning dynamics, *Signal Processing*, Vol. 74, No. 1, pp. 3–28 (1999).
[14] Goossens, M., Mittelbach, F. and Samarin, A.: *The LaTeX Companion*, Addison Wesley, Reading, Massachusetts (1993).
[15] Lamport, L.: *A Document Preparation System LaTeX User's Guide & Reference Manual*, Addison Wesley, Reading, Massachusetts (1986).

ⓒ 2018 Information Processing Society of Japan