Assessment of the Generalization Error of SGD in Deep Neural Network

Fayed Lynn, Maljković Marko
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland.

***Abstract*—Stochastic Gradient Descent (SGD) is an iterative optimization algorithm commonly used to train deep neural networks. Its stochastic nature surges from the use of a random sample/batch of the dataset to estimate the value of the gradient for model parameters update. The size of this batch, along with the learning rate, are two substantial hyperparameters dictating the convergence of SGD and its potential scalability. In this work, we evaluate how these two parameters are selected in the goal of achieving a low generalization error. Particularly, because the use of large batch sizes is convenient from a computational perspective albeit its poor generalization power, we empirically demonstrate how using both an adaptive batch size and learning rate is capable of resolving this problem.**

I. Introduction

SGD is one of the most popular algorithms used in machine learning to train deep neural networks. This algorithm is founded on an iterative process that computes the gradient of a loss function from a subset or batch of randomly sampled datapoints, hence its stochastic nature. Studies in the field demonstrated that the batch size is a major determinant of the degree of generalization of a trained neural network [1]. Generally speaking, the choice of a large batch size is associated with poorly scalable results whereas the selection of a small batch produces better generalizable outcomes. This is particularly an issue because small batch sizes – combined with the sequential nature of the SGD algorithm – limit the possibility of parallelizing the training process. Many factors substantiate the poor performance of large batch size such as the risk of over-fitting, the lack of exploration of the solution space or the high chances of getting trapped in sharp local minima [1]. To theoretically understand the impact of the batch size on the model outcome, we know that the update step of the model parameters is performed using the average gradient value of datapoints in the batch, and is given by

$$w_{t+1} = w_t - \mu \left(\frac{1}{B} \sum_{i \in \mathcal{B}} \nabla f_i(w_t) \right),$$

where w_t is the value of the model parameter at iteration t, μ is the learning rate, $\mathcal{B}$ is the sampled batch from the full training dataset such that $B = |\mathcal{B}|$, and ∇f is the gradient of the loss function [2]. Clearly, the gradient itself is a stochastic random variable whose variance is inversely proportional to the batch size. Therefore, opting for large batch sizes during training restricts the noise in the gradient, and subsequently its capacity to explore the full solution space. Because the loss function is non-convex, this leads the algorithm to converge to sharp local minima limiting hence the capacity of the model to generalize to new datasets. To remedy this issue, many approaches in the literature proposed an adaptive batch size framework that improves scalability while enhancing the parallelization potential [3]. Hence, the batch size represents a crucial parameter in SGD and should be carefully chosen.

Another major hyperparameter which, in conjunction with the batch size, largely influences the scalability of the SGD-trained models is the learning rate [4]. In fact, the higher the learning rate, the larger the gradient noise and hence the better the validation accuracy for large batch sizes. The learning rate is capable therefore of compensating for the poor generalization effect of the model trained with large batch sizes: since the noise is proportional to the ratio μ/B, increasing the learning rate enlarges the noise and helps achieving better generalization [4], [5].

The purpose of this work is to validate these observations by assessing the influences of batch size and learning rate on the validation accuracy and convergence speed. We also investigate the impact on scalability of having an adaptive rather than a fixed learning rate. Because large batch sizes are usually favored from a computational perspective, our goal is to also explore the effect of having both an adaptive batch size and a dynamic learning rate on the generalization error.

The remainder of this work is organized as follows. Section II provides a summary of the deep neural network architecture that we utilize and the dataset we use to train our model. In Section III, we present the accuracy and losses we obtain for training the neural network for different batch sizes and learning rates. We additionally provide a comprehensive analysis of the results we display. We finalize the discussion with a summary of our results in Section IV.

II. Models and Methods

For the purpose of assessing the influence of the batch size and the learning rate on the generalization error, we implement the AlexNet convolutional neural network architecture [6] on a dataset consisting of vegetable images [7]. The original dataset consists of a total of 21000 images divided into 15 classes. For the purpose of tractability, we reduce the size of the dataset to 10 classes containing each around 300 images for the training step, and circa 40 images for the validation step. The AlexNet model architecture is divided into convolutional feature layers and classification fully connected layers. We keep the feature extraction layers unaltered but we adapt the classification layers to our problem and train them from scratch.

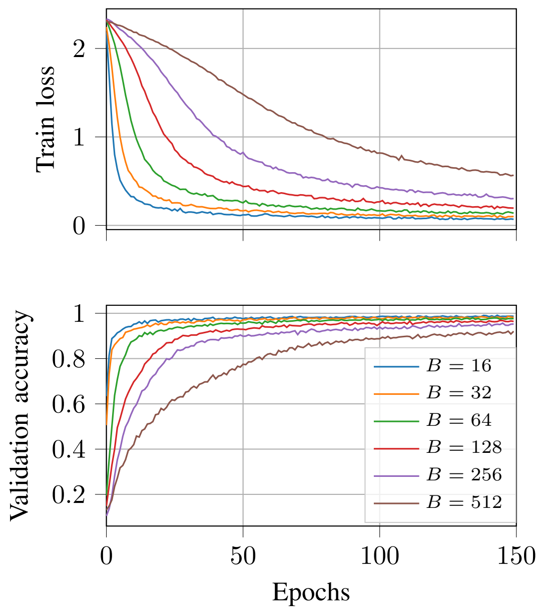

Fig. 1. Train loss and validation accuracy for $\mu = 0.0001$ and different B.

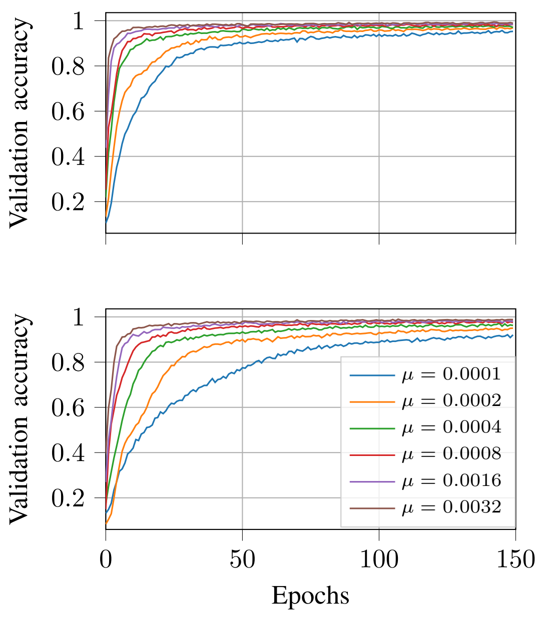

Fig. 2. The upper plot shows validation accuracy for $B = 256$ for different learning rates, whereas the lower plot shows for $B = 512$.

For the training phase, we first normalize the images according to their computed mean and standard deviation. We select SGD with a momentum value of 0.9 as the optimizer and the cross-entropy loss as the objective function. In SGD, the parameters are only updated once all datapoints in a batch are fed to the model. The model is trained for 150 epochs by first fixing the learning rate $\mu = 0.0001$ and only varying the batch size B where $B \in \{16, 32, 64, 128, 256, 512\}$. We then repeat the training, this time fixing the batch size to values equal to either 256 or 512, and we vary the learning rate such that $\mu \in \{0.0001, 0.0002, 0.0004, 0.0008, 0.0016, 0.0032\}$.

Once this is performed, we explore the generalization gain that we obtain in case we implement dynamic learning rate schemes where μ varies with the epoch. We evaluate the advantage of using a time-based decay, a step-decay, and exponential decay on the generalization error. The intuition behind this approach is to compensate for the limited exploratory nature of large batch sizes through the use of an adaptive learning rate. Similar to a simulated annealing procedure, having a large learning rate at the beginning strengthens the exploratory behavior of the SGD. As we move further in time however, decreasing the learning rate helps the minimizer to eventually converge. Let μ_k denote the value of the learning rate at epoch k and μ_0 denote the initial learning rate. In case of time-based decay, the learning rate is given by $\mu_{k+1} = \mu_k \cdot \frac{1}{1+kd}$, where $d = \frac{\mu_0}{N_{\text{epochs}}}$ is the decay rate and N_{epochs} is the total number of training epochs. The step-decay procedure is defined by $\mu_k = \mu_0 \cdot s^{g(k)}$, where $s < 1$ is the step decay, $g(k) = \left\lfloor \frac{k}{N_{\text{dec}}} \right\rfloor$ and N_{dec} is the number of consecutive epochs when the learning rate is constant. The exponential decay is given by $\mu_k = \mu_0 \cdot e^{-rk}$, where r is the decay rate.

As explained in Section I, having small batch sizes reduces the generalization error of SGD. Large batches however require less parameter updates, and are more effective from a computational point of view. For this reason, the final approach that we attempt consists of simultaneously adapting both the learning rate and batch size. By starting with $\mu_0 = 0.0001$, an initial batch size $B_0 = 16$, and following a step-decay of $s = 0.75$ for the learning rate update while simultaneously performing a step-ascend of $s_B = 2$ for the batch size update $B_k = \text{int}\left(B_0 \cdot s_B^{g(k)}\right)$ every $N_{\text{dec}} = 20$ epochs, we expect to achieve the same generalization error as if keeping a constant learning rate decay with $s = 0.375$. Since the SGD noise is dependent on the ratio μ/B, this approach yields the same generalization error without the need to use a very small batch size over all epochs. The results are displayed in the following section. All the code and data file descriptions to generate these results are provided in the Appendix.

III. Results and Discussion

In the following section, we investigate the influence of the batch size, the learning rate, or the influence of both simultaneously on the generalization error. We use classification accuracy as a metric to describe how well our model, trained using the SGD optimizer, performs on the validation set.

To delve into how growing the batch size influences the model generalization, we display the train loss and validation accuracy in Figure 1. For a fixed learning rate of 0.0001, the lowest batch size of 16 converges faster and achieves the best validation accuracy. As the batch size grows, not only does the algorithm converge much slowly, but it also converges to sharp minima. This is substantiated by a low variance of the gradient for large batch sizes restricting the noise, hence limiting the ability of the algorithm to explore areas of the loss function with better local minima. Therefore, albeit

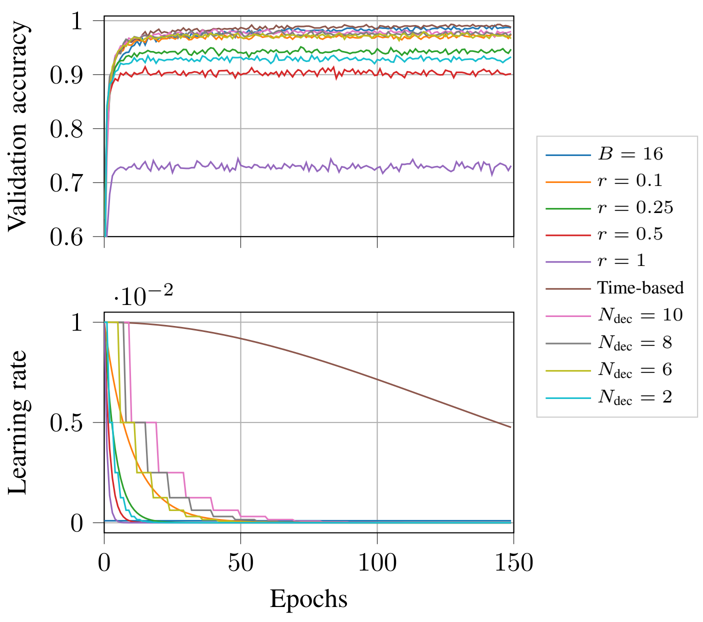

Fig. 3. Comparing the performance of the model trained with $B = 16$ and $\mu = 0.0001$ with the models trained with $B = 512$ and decaying μ. Initial value of the learning rate is $\mu_0 = 0.01$.

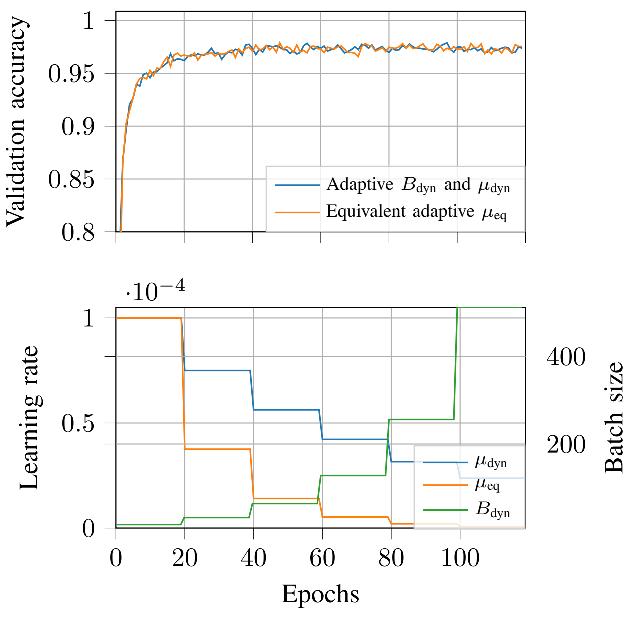

Fig. 4. Comparison of the model trained with batch size step-ascend $s_B = 2$ and step-decayed learning rate $s = 0.75$ with the model trained with only step-decayed learning rate $s = 0.375$.

being computationally efficient, large batch sizes yield poor generalization and poor performance on unseen data. Similarly in Figure 2, we evaluate the validation accuracy of the model trained with a batch size of 256 and 512 for different learning rates, and we observe the performance evolution as a function of the number of epochs. In accordance with the previous results, using smaller batch sizes during training results in a model with a lower validation error. Moreover, for large batch sizes, the larger the learning rate, the faster the convergence, and the higher the validation accuracy. This is justified by the noise emanating from the high learning rate, causing the SGD to explore better local minima during training. It is noteworthy to mention here that the convergence of the algorithm is comparable for both batch size values as long as the learning rate is relatively high. Our results are therefore conformable with the observations and conclusions in similar works.

Conversely to Figures 1 and 2, in Figure 3 we show results for dynamic learning rates defined in Section II when combined with the largest batch size $B = 512$. We test four different values of the parameters r and N_{dec} that dictate the speed of the learning rate decrease for the exponential and step-decay mechanisms respectively. Comparing the performance of the models with adaptive learning rate to the nominal model trained with $B = 16$ and $\mu = 0.0001$, we observe that for properly chosen μ_0 and decay parameters, it is possible to maintain the same convergence speed and achieve competitive validation accuracy even for large batch sizes. The time-based learning rate decay model was even able to outperform the nominal one. This is in accordance with the results shown in Figure 2 as the value of the learning rate stays high for the whole duration of the experiment, relative to the other dynamic mechanisms. Moreover, we observe that for large batch sizes, it is not necessary to run the whole experiment with a large value of the learning rate. The exponential and the step-decay dynamic mechanisms allow using large learning rates only during short initial periods. Figure 3 provides empirical evidence that there exist critical threshold values of the initial period length N_{cr} during which the learning rate has to be kept high in order to maintain certain validation accuracy. The decay rate is inversely proportional to N_{cr}, and from Figure 3 we can identify two values $N_{\text{cr}}^1 \approx 20$ and $N_{\text{cr}}^2 \approx 60$.

Finally, we show the effects of simultaneously reducing the learning rate and increasing the batch size during training. Figure 4 illustrates the equivalency between using the step-decay dynamic learning rate with $s = 0.375$ and simultaneously using the step-decay dynamic learning rate with $s = 0.75$ and step-ascend dynamic batch size with $s_B = 2$. The validation accuracy curves suggests almost perfect match in terms of convergence speed and generalization performance which solidifies the theoretical considerations in Section I.

IV. Summary

In the work we verified how to potentially reduce the generalization error for the SGD algorithm by adjusting two interrelated hyperparameters: the batch size and the learning rate. Low batch sizes and high learning rates achieve better scalability despite the approach being computationally exhaustive. One way around it is to adaptively alter the batch size and the learning rate to land over relatively better solution with lower number of gradient updates. Finally, exploring the influence of other hyperparamete, such as the momentum, on the scalability is another research line within this field.

References

[1] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *CoRR*, vol. abs/1609.04836, 2016.

[2] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, “Accurate, large minibatch sgd: Training imagenet in 1 hour,” 2017.

[3] S. Lee, Q. Kang, S. Madireddy, P. Balaprakash, A. Agrawal, A. Choudhary, R. Archibald, and W.-k. Liao, “Improving scalability of parallel cnn training by adjusting mini-batch size at run-time,” in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 830–839.

[4] S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. J. Storkey, “Three factors influencing minima in SGD,” *CoRR*, vol. abs/1711.04623, 2017.

[5] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: closing the generalization gap in large batch training of neural networks,” 2017. [Online]. Available: https://arxiv.org/abs/1705.08741

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” 2012.

[7] M. I. Ahmed, S. Mamun, and A. Asif, “Dcnn-based vegetable image classification using transfer learning: A comparative study,” 05 2021, pp. 235–243.

Appendix

The following section provides a summary of the list of submittals for this work in addition to this report.

- `SGD_main.ipnyb` jupyter notebook that contains code for all the steps we analyzed.
- `README.txt` file explaining how to reproduce the results.
- `Data` folder containing the reduced train and validation datasets.
- `Results` folder containing all the results presented in this project report.