International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 6 Issue 11, November - 2022, Pages: 22-30

A Directional Gradient Acceleration Method In Mini-Batch Gradient Descent

Sungjae Ahn
St. Johnsbury Academy Jeju, Jeju, Republic of Korea
4x493x1@gmail.com

***Abstract*—*The time needed to train a large-scale neural network with a large-scale dataset is immense. There were many studies that attempted to solve this problem using large batches, but bigger batch sizes decrease the average value of the gradient, resulting in a slower training speed. Bigger learning rates are therefore used to address this side effect but finding the optimal learning rate consumes a large amount of time as well. To assist the training of the model, this paper proposes the method of Direction Accelerated Stochastic Gradient Descent—a method that adds the average of the previous and present gradient to the present gradient if the direction of the two gradients is equal. The experiment compared the accuracy rate of vanilla SGD and DA-SGD with three different datasets, and the results show that DA-SGD yields greater accuracy rates for batch sizes bigger than 32. Further research of comparing DA-SGD with momentum optimizers, which are like the proposed method from the point that they take the previous gradients into consideration for updating the weights, is needed.*

Keywords—Batch size, Gradient Descent, Direction Accelerated Stochastic Gradient Descent

1. INTRODUCTION

DNN (Deep Neural Network) is being used in various fields while showing performance superior to that of humans. Representatively, there is speech recognition [1][2], natural language understanding [3][4], computer vision [5][6][7] and reinforcement learning [8][9].

With the development of computers, the number of computations per hour is steadily increasing, and the increase of data acquisition speed with the development of the Internet and communication is evolving into big data. Advances in these technologies enable the use of large-scale neural networks and large-scale datasets. However, training large networks and large datasets takes a lot of time.

Learning is the process of defining the loss function as the difference between the target values and the result values calculated based on the current weights and updating the weights in the direction that the loss value decreases. The direction in which the loss value decreases is found through gradient descent. If the target accuracy rate is reached quickly, the number of updates can be reduced, and consequently the training time can be reduced. Transfer learning is a method of shortening the weight update process required at the beginning of learning by using pre-trained weights rather than random initial weight values [5][7][10]. In other words, transfer learning has the effect of advancing the starting line. It is a practical method for large networks and large datasets, but it is known to be effective only when the pretrained weights are based on similar datasets. In addition, the appropriate initial values of the weights [11], the characteristics of the activation function used in the process of calculating the gradient [12], and batch normalizing the error values [13] also affect the learning time.

Besides transfer learning, the learning rate and batch size are factors that have a big influence on the learning time. A large batch size is preferred in terms of learning speed because a large batch size can have an advantage in parallel processing, but it is known that increasing the batch size reduces the test accuracy rate [14][15]. It is known that there is a positive correlation between batch size and learning rate. That is, when the batch size is increased, a prior study has shown that a proportional increase in the learning rate can give better results empirically [15][16]. As the batch size increases, the fluctuation due to noise will decrease because the number of samples used for weight update increases. Since many samples are used, it is logical to use a relatively large learning rate in terms of heuristics.

In addition, previous studies are using a method of reducing the learning rate according to the progression of the learning time. This is based on the heuristic that a large step of learning is advantageous at the beginning of learning, but it is better to decrease the amount of learning in the second half. On the other hand, Samuel L. Smith et al (2018) argues in his study that increasing the batch size according to the learning time can give the same effect as decaying learning rate [16]. Since increasing the batch size is advantageous to reducing the learning rate in terms of computations such as parallel processing, it is better to increase the batch size if the two methods have the same effect.

The positive correlation between batch size and learning rate can reduce the search time to find the optimal combination between two hyper-parameters. However, even with a proportional relationship, it is difficult to generalize which initial value combination between two hyper-parameters is the best starting point. Empirically, the batch size is fixed, and the optimal learning rate is found

through repeated experiments, and thereafter, the learning rate is also increased as the batch size increases. The learning rate increment suitable for the increment of the batch size is determined by the experimenter as a kind of hyper-parameter.

Increasing the batch size increases the number of samples used for gradient calculation. Inter-sample averaging can reduce the negative effects of noise or outliers, but on the other hand, increasing the number of samples dilutes representativeness by averaging. Representing 256 samples (batch size = 8 * 32) with 32 averages is more effective in terms of accuracy than representing 256 samples (batch size = 256) with one average. As a result, as the batch size increases, the representativeness decreases due to averaging, so the accuracy rate decreases.

As the batch size increases, the size of the gradient values may decrease due to being averaged, but the representativeness of the gradient increases. This is because the direction of 256 samples is more representative than the direction of 8 samples.

In this study, we propose the acceleration of the learning amount by using the directionality of the gradient. Weight update using gradient locally applies only the change amount of the loss function at a specific point in time. When the direction of the previous gradient and the current gradient are the same, the average value between the two gradients is used as an additional weight for acceleration, and when the direction of the gradient changes, no weight is applied. It is designed to act like general SGD at the inflection point where the direction of the gradient changes, and like inertia when the gradient proceeds in the same direction.

As a result of the experiment, it was applied to the dataset of fashion mnist, kuzushiji mnist, and quickdraw 10, and when the batch size was large, better performance could be obtained than a general mini batch using vanilla gradient. In the Fashion mnist and the kuzushiji mnist, the vanilla gradient showed better performance when the batch size was less than 8, but in the case of quickdraw 10, similar performance was shown between the vanilla gradient and the method proposed in this study. On the other hand, as the batch size increased, the proposed method performed better than the vanilla gradient, and it was possible to obtain a better accuracy rate as well as a faster learning speed through acceleration using the directionality of the gradient.

2. PROPOSED METHOD

The mini batch averages the gradient values of several samples and updates the weights. Here, the gradient for each weight has a value in the direction in which the loss value decreases. In order to find the value at which the loss value becomes the smallest, the correlation between the weights must be considered, but due to the complexity of the calculation, the gradient is calculated for each weight under the assumption that all weights are independent. Therefore, the average gradient in a mini batch becomes the average of the gradient vector values that have a one-dimensional relationship with each weight. In other words, it is the average between gradients of positive or negative values for each weight.

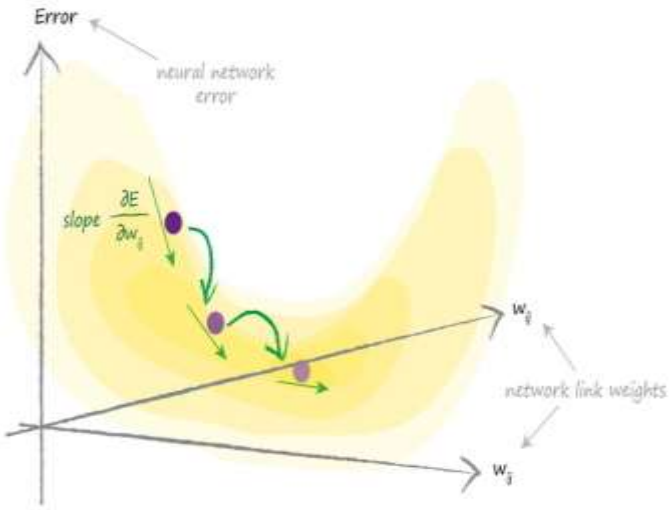


Figure 1. Gradient Descent (Tariq, Make your own neural network, p.92)

Therefore, as the batch size increases, the average value of the gradient decreases, and the speed of convergence to the local minimum is slowed down by the slow rate of change of the weights. A commonly used solution to offset this decrease in the sizes of the gradient values is to increase the size of the learning rate as the batch size increases. However, the learning rate is a hyper parameter that must be directly specified by a person, and it takes a lot of time to find the optimal learning rate value for a fixed batch size.

To solve this problem, we focused on the mini batch's representativeness of the direction of the gradients. As the batch size increases, the average value of the gradient becomes smaller, but since the gradient values for more samples are considered, the representativeness of the direction of weight movement increases. The method presented in this paper offsets the small average value of the gradient by adding to the current gradient a booster, which is calculated according to the direction—the sign—of the previous and current gradients. This process is expressed as a formula as follows.

$$\omega_{t+1} = \omega_t - \alpha \cdot (\nabla_{\omega_t} L(\omega) + \rho_t) \quad (1)$$

$$\rho_t = \begin{cases} \mathbf{avg}\,(\boldsymbol{\nabla_{\omega_t} L(\omega)} + \boldsymbol{\nabla_{\omega_{t-1}} L(\omega)})\,, & when\ sign(\nabla_{\omega_t} L(\omega)) = sign(\nabla_{\omega_{t-1}} L(\omega)) \\ \mathbf{0}, & when\ sign(\nabla_{\omega_t} L(\omega)) \neq sign(\nabla_{\omega_{t-1}} L(\omega)) \end{cases}$$

In the equation above, Pt is the booster added to the gradient. This booster is calculated as the average of the previous and current gradients if their signs are equal and calculated as 0 if their signs are different.

Vanilla SGD only uses the gradient locally computed at a specific point on the loss function and does not use the concept of acceleration that continuously increases or decreases the gradient. On the other hand, the proposed method has an acceleration factor that offsets the slow rate of change of the weight, which can perform better than vanilla SGD at large batch sizes.

3. EXPERIMENT

Three datasets were used to measure the effect of the direction acceleration method proposed in this paper. The datasets used are: the fashion mnist composed of clothing photos, the kuzushiji mnist composed of Japanese characters, and the quickdraw 10 composed of simple hand drawings. Each sample in all the datasets consists of a 28 x 28 gray image, where the fashion mnist and kuzushiji mnist consist of 60K training and 10K testing samples. In the case of Quickdraw 10, it provides 80K training samples and 20K testing samples.

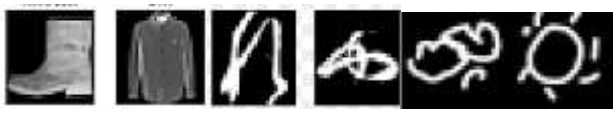


Figure 2. Samples of Fashion mnist, Kuzushiji mnist, and Quckdraw10

The DNN network used in the experiment consists of 784 input nodes, 128 hidden nodes, 2 hidden layers, and 10 output nodes. Because all 3 datasets consisted of 28 by 28-pixel pictures, which had to be converted into a 784 elements-long one-dimensional array, 784 input nodes were used, and 10 output nodes were used to distinguish the 10 types of answer labels. A bias node was added to all layers except the output layer, and a sigmoid function was used for the activation function. This DNN network was constructed with Python and NumPy.

The experiment compared vanilla SGD with the proposed method, Direction Accelerated Stochastic Gradient Descent (DA-SGD). The models were trained with the DNN network settings, and the epoch was set to 200. The (batch size, learning rate) sets used in the experiment are as follows: (1, 0.001), (2, 0.001), (4, 0.001), (8, 0.002), (16, 0.003), (32 , 0.004), (64, 0.005), (128, 0.006), (256, 0.007). As the batch size increased, the learning rate value was also increased, but in the case of batch sizes 1, 2, and 4, the same learning rate of 0,001 was used. This was because the convergence speed could be slow if the learning rate was too small.

The accuracy rate was measured by changing the learning rate from 0.001 to 0.01 and epoch from 1 to 200 with a fixed batch size of 8. As shown in Figure 3, the learning rate of 0.002 at epoch 200 shows better performance than 0.001. Therefore, in the combination of (batch size, learning rate), (8, 0.002) was used as a reference point, and when the batch size was smaller than 8, the learning rate was decreased to 0.001, and when the batch size was larger than 8, the learning rate was increased by 0.001.

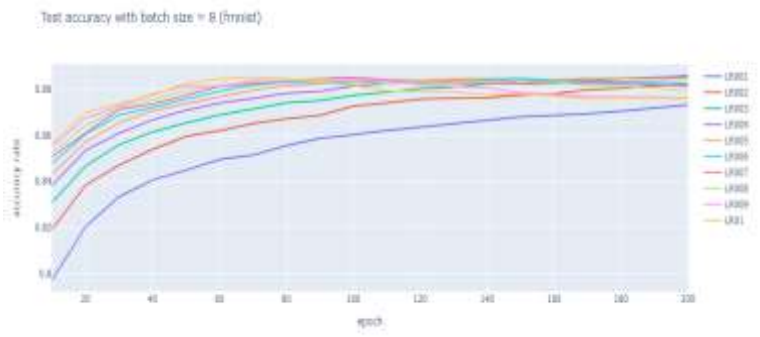
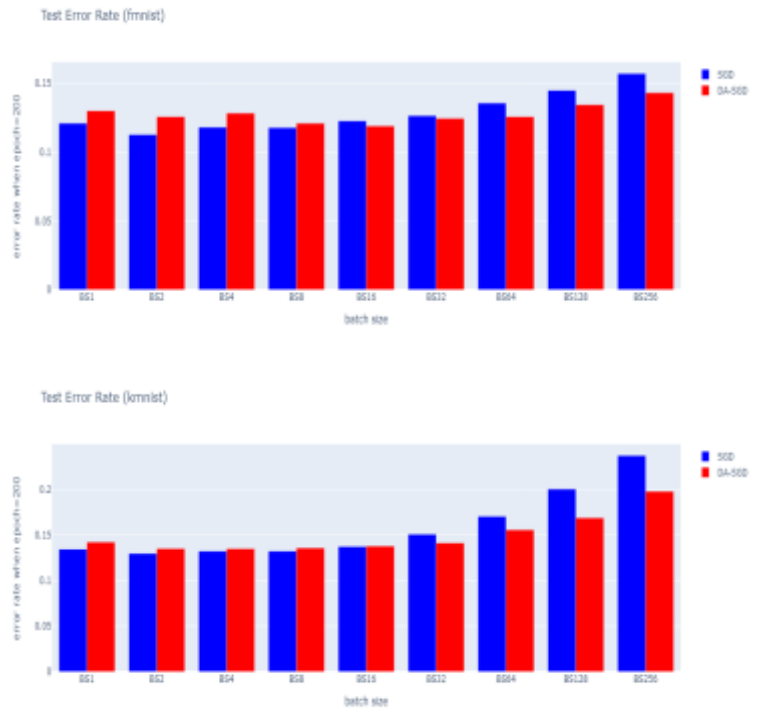


Figure 3. Test accuracy rate according to various learning rates

Figure 4 shows the error rate for the combinations of (batch size, learning rate) at epoch 200. Batch sizes 1 and 2 show different results for each dataset, but vanilla SGD generally performs better with batch sizes smaller than 32. However, as expected in Chapter 2, the proposed method shows better performance for big batch sizes that are, where in this experiment, equal to or bigger than 32. This performance difference is more pronounced when the batch size is larger. Looking at Figure 4, you can see a bigger difference in performance for batch sizes 64, 128, and 256 than batch size 32.



International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 6 Issue 11, November - 2022, Pages: 22-30

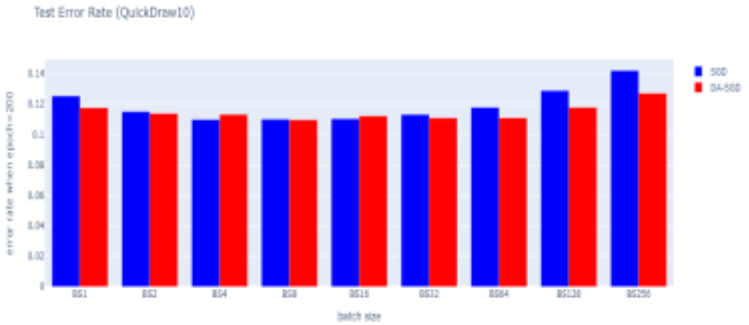
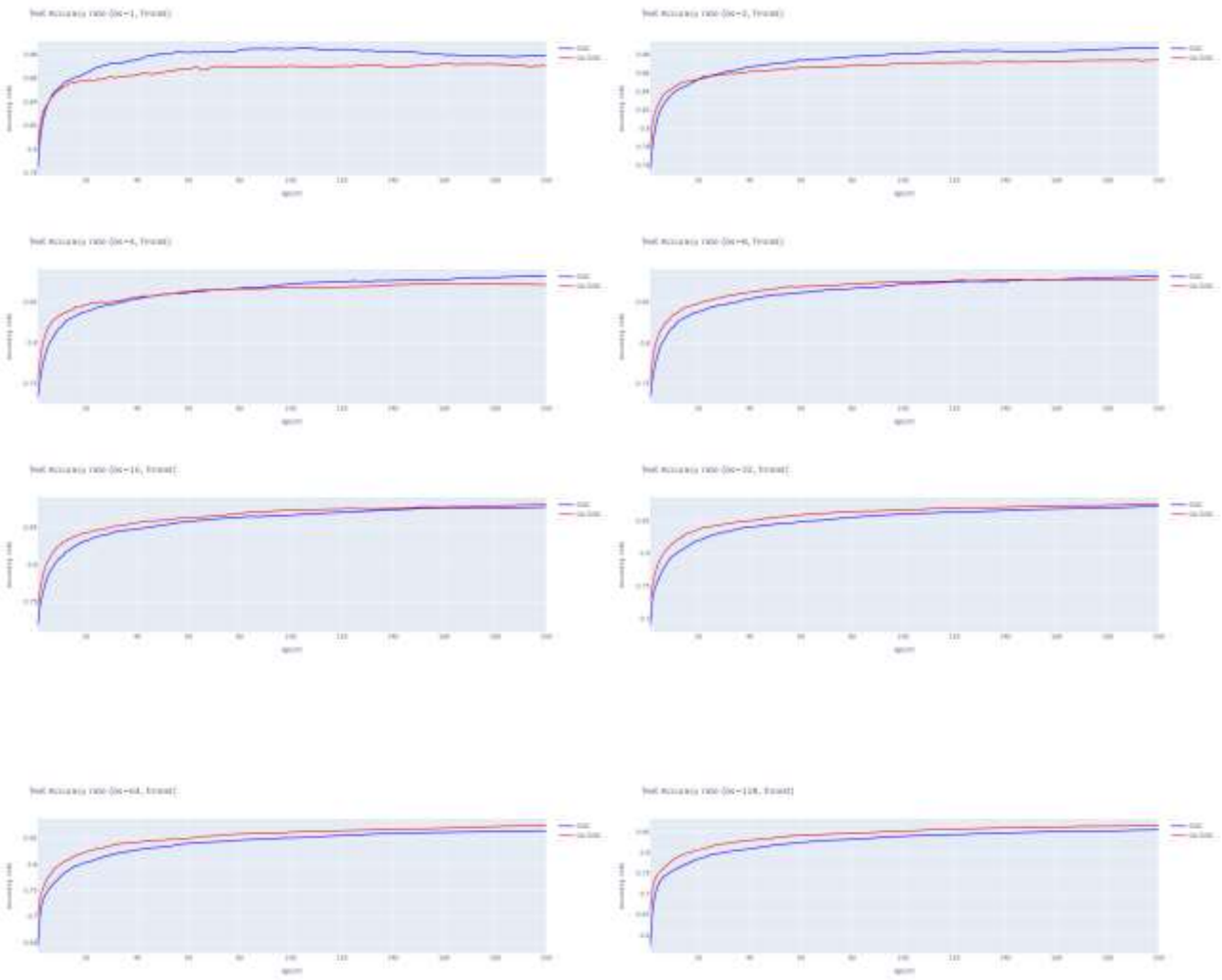
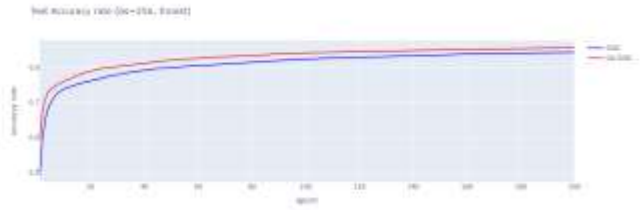
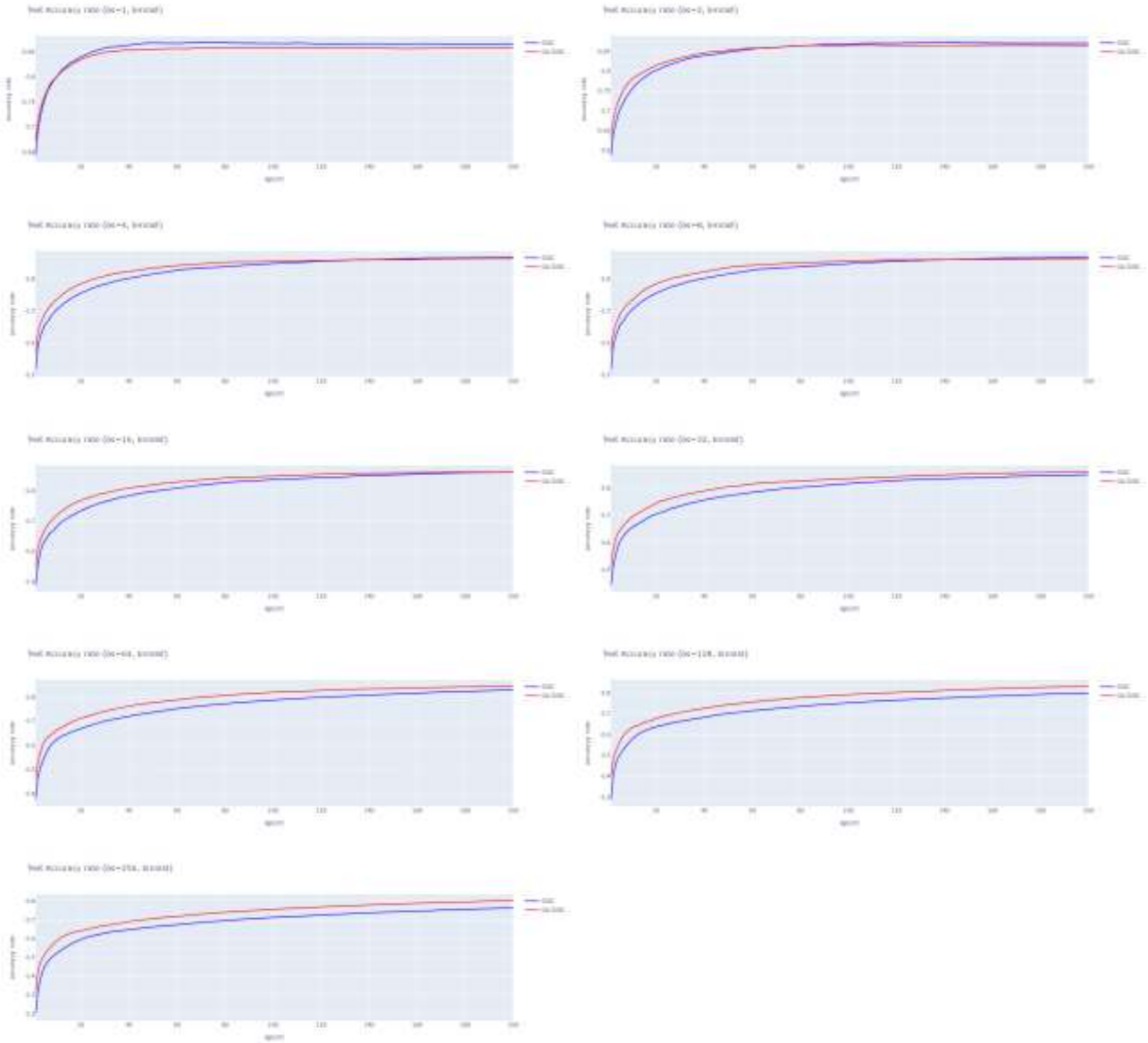


Figure 4. Test error rate between SGD(vanilla) and DA-SGD

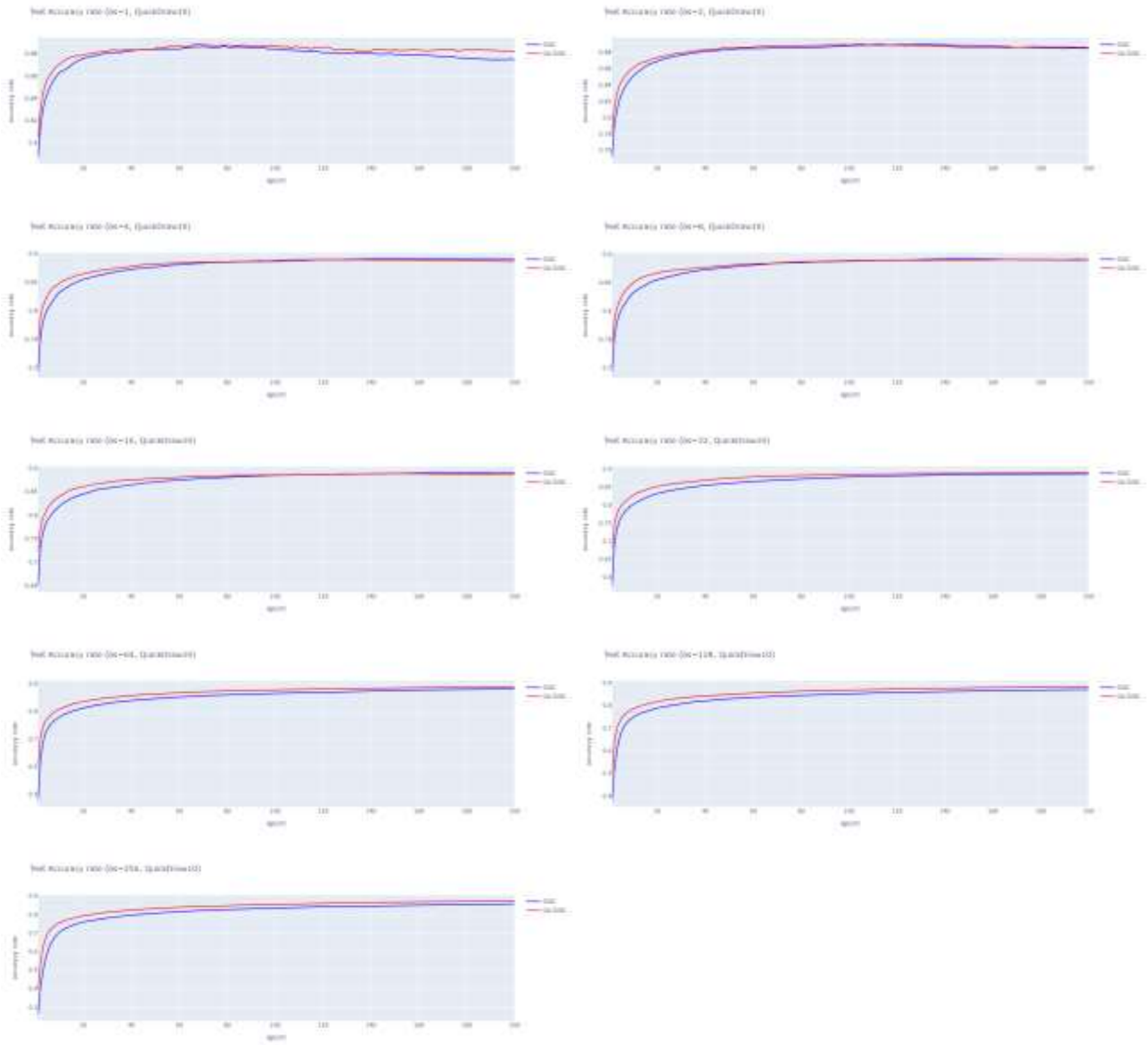




(a) Fashion mnist with batch size = 1 to 256



(a) Kuzushiji mnist with batch size = 1 to 256



(a) Kuzushiji mnist with batch size = 1 to 256

Figure 5. Test accuracy rate between SGD (vanilla) and DA-SGD

4. CONCLUSION

When using large-scale neural networks and large-scale datasets, training time becomes a real problem. There have been previous studies about the use of large mini batches to shorten the learning time. However, if the batch size is increased, the number of samples involved in calculating the gradient increases, which decreases the size of the average gradient that results in the slow change of the weights. As a solution to this problem, large learning rates are used, but finding the optimal learning rate is a time-consuming task.

To address this problem, this paper paid attention to the characteristic of large mini batches: strong representation of gradient directions. When the batch size is small, it can be said that the mini batch weakly represents the direction of the gradients. Therefore, applying the values of the previous gradient to the current gradient can result in a benefit but it can also result in a loss. On the other hand, as the batch size increases, the mini batch better represents the direction of the gradients. Therefore, when using a large batch size, it can be analyzed that there is a high probability of benefiting from using the previous gradient as an acceleration component if the previous gradient and the current gradient point in the same direction. In the experiment conducted in this paper, DA-SGD showed better performance than vanilla SGD when the batch size was large (batch size 32 or more), as expected.

The proposed method is like the concept of DNN optimizers from the point that they both accelerate the change of weights according to the direction of the previous gradient. In particular, it is similar to the momentum-type optimizer. However, while momentum uses the gradient accumulated on a global scale throughout the entire learning process, the proposed method is conceptually different in the sense that it uses an acceleration factor calculated based on a local scale, where it is only applied when the direction of the previous and current gradient is in the same direction. As an additional study, a comparison between the proposed method and the momentum optimizer is needed to determine whether the global scale calculation of the acceleration factor is better or local scale calculation of the acceleration factor is better.

.

5. REFERENCES

[1] Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." IEEE Signal processing magazine 29.6 (2012): 82-97.

[2] Xiong, Wayne, et al. "Achieving human parity in conversational speech recognition." arXiv preprint arXiv:1610.05256 (2016).

[3] Collobert, Ronan, et al. "Natural language processing (almost) from scratch." Journal of machine learning research 12.ARTICLE (2011): 2493-2537.

[4] Wu, Yonghui, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." arXiv preprint arXiv:1609.08144 (2016).

[5] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[6] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[7] He, Kaiming, et al. "Deep residual learning for image recognition. CVPR. 2016." arXiv preprint arXiv:1512.03385 (2016).

[8] Dayan, Peter, and Yael Niv. "Reinforcement learning: the good, the bad and the ugly." Current opinion in neurobiology 18.2 (2008): 185-196.

[9] François-Lavet, Vincent, et al. "An introduction to deep reinforcement learning." Foundations and Trends® in Machine Learning 11.3-4 (2018): 219-354.

[10] Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." arXiv preprint arXiv:1312.6229 (2013).

[11] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010.

[12] Xu, Bing, et al. "Empirical evaluation of rectified activations in convolutional network." arXiv preprint arXiv:1505.00853 (2015).

[13] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. PMLR, 2015.

[14] 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, 2016.

[15] Priya Goyal, Piotr Dollar, 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, 2017.

[16] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. arXiv preprint arXiv:1710.06451, 2017.

International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 6 Issue 11, November - 2022, Pages: 22-30

Authors

Sungjae Ahn, St. Johnsbury Academy Jeju