Effect of Batch Size Varying by Train Set Size in Mini-Batch Gradient Descent

Seungil Lee
seungil.lee@epfl.ch

Yujun Kim
kyujun02@kaist.ac.kr

June 16, 2022

Abstract The mini-batch gradient descent is a commonly used optimization method in machine learning. A smaller batch size is known to result in better performance, but slowdowns the training procedure. However, it has not been studied much how data set size affects these tendencies. To figure out this, we investigate some computer vision benchmark data sets. We conclude that the correlation between batch size and performance is higher when the train set size is large, while it is hard to figure out the relation when data size is small. [1]

1 Introduction

The gradient descent (GD) method is a general paradigm for optimizing objective function by navigating to the direction of descent. However, GD does not work well for non-convex problems as it may converge to the local optimum. The stochastic gradient descent(SGD) has come to rise to handle the problem in a situation where the objective function is represented as a sum of functions.

$$\min_{x \in S} f(x),\ f(x) = \frac{1}{M} \Sigma_{i=1}^{M} f_i(x)$$

SGD randomly picks one of these functions and navigates through the direction to optimize the selected function. Unfortunately, SGD lacks in the speed of computation as we calculate the gradient of a single function at a time.

[1] https://github.com/ChoiIseungil/OptMLProject.git

Mini-batch gradient descent (Batch GD) is the variant of SGD which tackles the efficiency problem. Batch GD selects batch size number of functions to calculate gradient to take advantage of computing gradients in parallel. In general, with a fixed number of the epoch, we get better accuracy as we decrease the batch size[1]. However, the computational time increase as well with the smaller batch size. Hence, it is important to select an adequate batch size to balance performance and time efficiency.

To solve this problem, we investigate how the performance with respect to batch size tendency changes according to the size of the training data set. We expect this work to be used as a reference for researchers to select the appropriate batch size to get desired performance while keeping the efficiency.

2 Experimental Settings

We experiment with computer vision problems that are being studied most actively in machine learning. We work with two different models and two different data sets to assure the result is independent of models and data. We choose the most commonly used computer vision benchmark data sets, **CIFAR-10** and **MNIST**, to solve the image classification problem.

2.1 Data Set and Neural Model

CIFAR-10 The CIFAR-10[2] data set consists of 50000 training images and 10000 test images in 10 classes. The classes are completely mutually exclu-

sive, so the training batches contain exactly 5000 images from each class. All training samples are padded, randomly horizontally flipped then normalized. ResNet18[3] and VGG-19[4], which are neural models that perform well in image classification tasks, are exploited.

MNIST The MNIST[5] is a large database of handwritten digits consisting of 60000 training images and 10000 test images. The digits have been size-normalized and centered in a fixed-size image. The images are normalized only, without any other augmentation. ResNet18 and simple Fully Connected Network(FCN) are used for MNIST. 10000 test sets are used as they are, and some of the 50000 train sets are used for training and validating.

2.2 Batch Size

The train set has been varied on uniform-logarithmic space between $2^{10} \sim 2^{15}$. For each train set size, we examine the accuracy by varying the batch size ratio ranging between $\frac{1}{2^8}$ and $\frac{1}{2^3}$ compared to the train set size. For instance, if train set size is 2^15 and batch size ratio is $\frac{1}{2^8}$, then the batch size is $\frac{2^{15}}{2^8} = 2^7 = 128$.

2.3 Details

$1e-2$ is chosen as learning rate for ResNet and VGG, while $1e-1$ is chosen for FCN. The learning rate scheduler and momentum are not adopted to investigate the effect of vanilla SGD. A TITAN RTX 3090, which has 24GB memory is used through all experiments. To prevent the model to be overfitted, a validation set whose size is equal to $\frac{1}{4}$ of the train set size is used for validating. The training process is early stopped if the validation loss increases for 5 consecutive epochs in series. The number of epochs is set to be $100 \times 2^{15}/(\text{train set size})$. It is kept negatively related to the train set size to expose train data to the same number in total.

3 Results

For accuracy and loss table corresponding to four different setup of the experiment, refer to Figure 2-9 in 5 Appendix: Figures.

3.1 CIFAR-10

Figure 2-5 shows the accuracy and validation loss with respect to both train set size and batch size. By comparing Figure 2, 3 and Figure 4, 5 respectively, we can check negative correlation between loss and accuracy.

The accuracy of the test gets higher with a greater number of train set size and smaller batch size which is highly expected. Consider each row in Figure 2 and 4. With more train set, the network optimizes in a way to be adapted in diverse test cases resulting in higher accuracy. Also, by considering each column in the same figures, we conclude that in general optimization works better with smaller batch size. This is resulted by higher tendency of randomness in the propagation of optimization with smaller batch size.

The interesting part is that when train set is not large enough, the correlation between batch size and test accuracy is not obvious. The optimal batch ratio is not with the smallest value, but rather is in between the smallest and the largest value. In VGG with train set size 2^{11}, it even reaches optimal accuracy with the largest batch ratio. This is interesting because it is known to have better performance with a smaller batch size. However, we find that it is not always optimal to adopt a small batch size if the train data is insufficient. Thus, we can still get optimal performance in time efficient way. We call this phenomenon *the break of optimality* with a smaller batch size in case of insufficient training data set. This will be further discussed in Section 4

The data set size is a more dominant factor than the batch ratio towards accuracy. In Figure 2 and 4, the maximum accuracy among the column of the smallest train set size is less than the minimum accuracy among the column of the largest train set size. Also, considering the diagonal of two accuracy tables, $(i, i)^{th}$ element increase with respect to i except the section $i = 5, 6$. This means the effect of reducing

batch size is less dominant than the effect of train set size, when train set size is fairly small. However, the effect of batch size gets gradually bigger with the increase of train size.

3.2 MNIST

Figure 6-9 show the accuracy and validation loss with respect to both train set size and batch size. The overall tendency of the tables is very similar to the result of CIFAR-10.

CIFAR-10 and MNIST are two different data sets in the perspective of classifying object images and number images respectively. Nevertheless, the section of train set size, where reducing batch size does not unconditionally benefit accuracy, is ranging similar between $2^{10} \sim 2^{11}$. In other words, the break of optimality with a smaller batch size occurs at a similar train set size for both CIFAR-10 and MNIST data sets.

ResNet table tendency for MNIST matches the one for CIFAR-10 which shows that the optimization tendency is independent of the data.

However, FCN shows quite different behavior than the other nets. Looking at the diagonal of the accuracy table in Figure 8 no longer gives a monotone increase in $(i, i)^{th}$ element with respect to i. This means FCN is more sensitive to batch ratio than other models. As FCN does not contain a convolutional layer, this might be a candidate for the unique behavior of FCN.

4 Discussion

The last two paragraphs of Section 3.1 tell a thing in common: Reducing batch ratio is less important when the train size is small, and gets gradually important when the train size gets larger.

Figure 1 further visualise Figure 2 as a graph. Each line in the graph represents the accuracy with respect to the batch ratio for each train set size. In this graph, we can further quantify the sensitivity of accuracy with respect to batch ratio. The graph differs a bit by data sets and model, and by converting Figure 4, 6 and 8 in the same manner can also give a deeper understanding of the tendency.

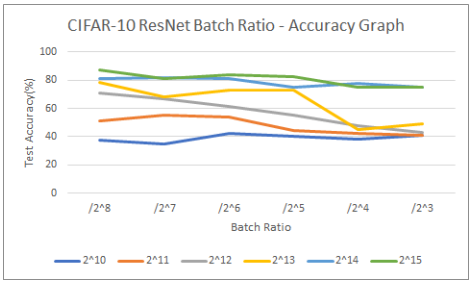

Figure 1: CIFAR-10 ResNet Accuracy Graph

We now discuss how the research can be further used in optimizing the area. Suppose a situation of an image classification problem. If we don't have enough image data for the objects we aim to classify, like the case where the object is newly created or the object is not commonly used, it is sometimes hard to get a sufficient train set. In those situations, to optimize the model with optimal value, it is not always good to keep a small batch size. Recall that the accuracy formed concave function with respect to the batch ratio when the train set is small. Rather, we may increase batch size gradually from 1 until accuracy begins to drop.

In the case where we have enough training data, cycling one epoch even takes a reasonable amount of time. Thus, we sometimes want to trade off between the accuracy and training time by increasing batch size a bit. In this situation, these data can be used as a reference guide for researchers to choose the right number of batch size to speed up while keeping the desired accuracy.

However, the research still lacks in cases where we desire really high performance. In this case, the train set size will be really large. Nevertheless, we may not want to trade-off any accuracy for training time, like when the problem is related to the safety problem like obstacle detection and classification problem in autonomous driving for example. To cover these parts, we can conduct similar experiments with much larger data sets, than those with CIFAR-10 and MNIST, to check how much data is needed to reach really high accuracy.

5 Appendix: Figures

5.1 CIFAR-10

CIFAR-10 accuracy and loss results in figures.

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	37.639	51.01	70.69	78.63	81.319	87.399
/2^7	34.939	55.57	66.979	68.47	81.909	81.349
/2^6	42.589	54.039	61.529	72.769	81.009	83.439
/2^5	40.629	44.359	54.949	73.059	74.619	82.179
/2^4	38.409	42.229	47.649	45.209	77.549	75.239
/2^3	41.01	40.8095	42.9256	48.9515	74.832	74.874

Figure 2: CIFAR-10 ResNet Accuracy

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	1.6413	1.4268	0.87	0.6928	0.5965	0.4312
/2^7	1.7083	1.3203	1.0455	0.9014	0.5715	0.5436
/2^6	1.5565	1.245	1.0984	0.8654	0.5957	0.4836
/2^5	1.4873	1.5394	1.284	0.845	0.7476	0.5404
/2^4	1.6592	1.5594	1.4587	1.4857	0.6461	0.6811
/2^3	1.7688	1.6655	1.4847	0.8908	1.2872	1.0409

Figure 3: CIFAR-10 ResNet Loss

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	20.039	12.079	53.159	74.699	79.999	82.979
/2^7	12.559	24.529	58.789	70.069	80.699	84.52
/2^6	20.479	25.45	42.589	72.639	80.199	82.019
/2^5	24.12	18.079	64.209	69.529	76.199	80.959
/2^4	21.759	34.779	57.589	65.459	75.629	78.319
/2^3	13.3	45.269	38.779	67.019	54.379	66.659

Figure 4: CIFAR-10 VGG Accuracy

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	1.6413	1.4268	0.87	0.6928	0.5965	0.4312
/2^7	1.7083	1.3203	1.0455	0.9014	0.5715	0.5436
/2^6	1.5565	1.245	1.0984	0.8654	0.5957	0.4836
/2^5	1.4873	1.5394	1.284	0.845	0.7476	0.5404
/2^4	1.6592	1.5594	1.4587	1.4857	0.6461	0.6811
/2^3	1.7688	1.6655	1.4847	0.8908	1.2872	1.0409

Figure 5: CIFAR-10 VGG Loss

5.2 MNIST

MNIST accuracy and loss results in figures.

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	96.419	97.2	98.469	98.96	99.269	99.439
/2^7	96.84	97.679	98.189	98.949	99.149	99.339
/2^6	95.699	98.049	98.259	98.78	98.989	99.219
/2^5	96.819	97.489	98.259	98.489	98.819	99.079
/2^4	95.98	97.139	97.859	98.579	98.749	99.109
/2^3	95.829	97.189	97.659	98.339	98.739	98.999

Figure 6: MNIST ResNet Accuracy

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	0.085	0.0867	0.0579	0.0555	0.0187	0.0229
/2^7	0.2346	0.1358	0.0794	0.043	0.0355	0.0204
/2^6	0.1504	0.0439	0.1096	0.0507	0.0336	0.0338
/2^5	0.1109	0.0989	0.0714	0.0523	0.0458	0.039
/2^4	0.1066	0.1673	0.0957	0.0546	0.0555	0.0269
/2^3	0.1688	0.1167	0.1023	0.0485	0.0487	0.0316

Figure 7: MNIST ResNet Loss

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	88.949	91.549	92.419	94.9	95.969	94.909
/2^7	82.019	91.439	93.129	94.9	94.679	92.779
/2^6	89.139	91.269	91.899	94.099	91.99	84.81
/2^5	82.459	91.2	91.13	85.559	84.429	83.419
/2^4	80.909	90.369	91.31	90.489	83.459	76.679
/2^3	88.589	83.039	90.759	83.199	68.509	41.439

Figure 8: MNIST FCN Accuracy

Batch Ratio \ Train Set Size	2^10	2^11	2^12	2^13	2^14	2^15
/2^8	1.5818	1.5582	1.5449	1.519	1.5037	1.5213
/2^7	1.6645	1.5802	1.5353	1.5233	1.5183	1.5449
/2^6	1.5762	1.5443	1.5668	1.528	1.5502	1.6228
/2^5	1.6306	1.5625	1.5725	1.6298	1.6255	1.653
/2^4	1.6467	1.5729	1.5629	1.5822	1.6541	1.7888
/2^3	1.5773	1.6454	1.5795	1.6664	1.8304	2.1804

Figure 9: MNIST FCN Loss

6 References

References

[1] Robert Kleinberg, Yuanzhi Li, and Yang Yuan. *An Alternative View: When Does SGD Escape Local Minima?* 2018. arXiv: 1802.06175.

[2] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *University of Toronto* (May 2012).

[3] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[4] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*. 2015.

[5] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.