WHEN DOES SECOND-ORDER OPTIMIZATION SPEED UP TRAINING?

Satoki Ishikawa & Rio Yokota
Department of Computer Science, Tokyo Institute of Technology
{riverstone, rioyokota}@rio.gsic.titech.ac.jp

ABSTRACT

While numerous second-order optimization methods have been proposed to accelerate training in deep learning, they are seldom used in practice. This is partly due to a limited understanding of the conditions under which second-order optimization outperforms first-order optimization. This study aims to identify these conditions, particularly in terms of batch size and dataset size. We find empirically that second-order optimization outperforms first-order optimization when the batch size is large and the data set size is not too large.

1 INTRODUCTION

Second-order optimization utilizes gradients preconditioned by a second-order information matrix(Pascanu & Bengio, 2013). Various Second-order optimization methods have been proposed, differing in the type of second-order information matrix and the method of matrix approximation.

K-FAC is a major second-order optimization method proposed in Martens & Grosse (2015) as an approximation of natural gradient descent(Amari, 1998). K-FAC has been studied in numerous papers because of its effectiveness in many deep learning optimizations(Grosse & Martens, 2016; Martens et al., 2018; Izadi et al., 2020; Eschenhagen et al., 2023; Zhang et al., 2022). Inspired by the fast convergence of K-FAC, Benzing (2022) proposed FOOF, hypothesizing that the fast convergence of K-FAC might stem from local learning rather than from natural gradient descent. Shampoo represents another notable advancement in second-order optimization(Gupta et al., 2018). It serves as an approximation to adaptive gradient descent (Duchi et al., 2011). Other second-order optimization methods, such as K-BFGS (Goldfarb et al., 2020) and PSGD (Li, 2017; 2018), estimate the curvature matrix through iterative processes. Additionally, second-order optimization methods like SENG (Yang et al., 2020) employ the Sherman-Morrison-Woodbury (SMW) formulas in inverse matrix computations to facilitate second-order optimization.

In this study, we initially examine the batch size characteristics of second-order optimization methods across various existing approaches. Some studies suggest that the large batch problem, where accuracy deteriorates when training with larger batch sizes, is less likely to occur with second-order optimization methods (Zhang et al., 2019), and numerous studies have applied second-order optimization to large-batch learning (Osawa et al., 2019b; Ueno et al., 2020; Anil et al., 2021). However, comprehensive research comparing different methods is still lacking. Additionally, in the Appendix.A.2, we investigate the relationship between batch size and hyperparameters and find some interesting proportional relationships. Next, we observe the behavior when the dataset size is increased while keeping the batch size constant. Surprisingly, we find that in training with larger dataset sizes, such as in language modeling, the advantage of second-order optimization over first-order optimization becomes less evident. This indicates that not only the batch size but also the dataset size is an important factor in determining the superiority of second-order optimization.

2 SECOND ORDER OPTIMIZATION IS GOOD FOR LARGE BATCH SIZES

Second-order optimization is known to be more advantageous than first-order optimization when the size of the mini-batch is large and the specific mini-batch closely resembles the entire dataset(Zhang et al., 2019; Osawa et al., 2019b; Anil et al., 2021). Fig.1 shows that for various architectures,

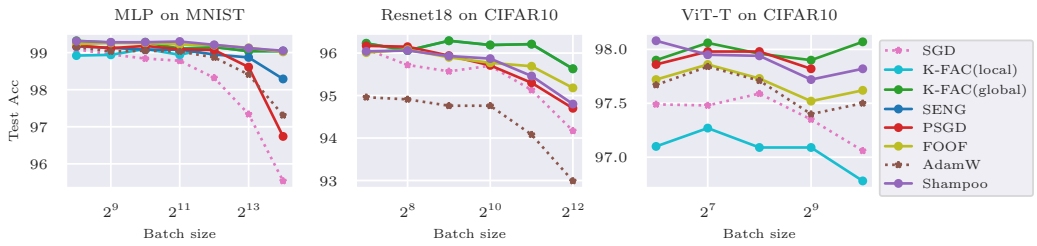

Figure 1: **Second-order optimization performs better than first-order optimization for large batch training** Second-order optimization methods perform better with larger mini-batch sizes. However, for PSGD, test accuracy decreases similarly to SGD as batch size increases.

second-order optimization methods, including K-FAC and Shampoo, outperform first-order optimization methods for larger batch sizes. Note that the accuracy of PSGD decreases rapidly as the batch size increases. This may occur because PSGD computes the curvature matrix iteratively. See Appendix A.1 for more details. In addition, when the mini-batch size is small, K-FAC(local) without an exponential moving average does not perform as well as K-FAC(global) with an exponential moving average. The relationship between batch size and appropriate hyperparameter settings is described in detail in Appendix A.2.

3 Second Order Optimization is Good for Small Dataset Size

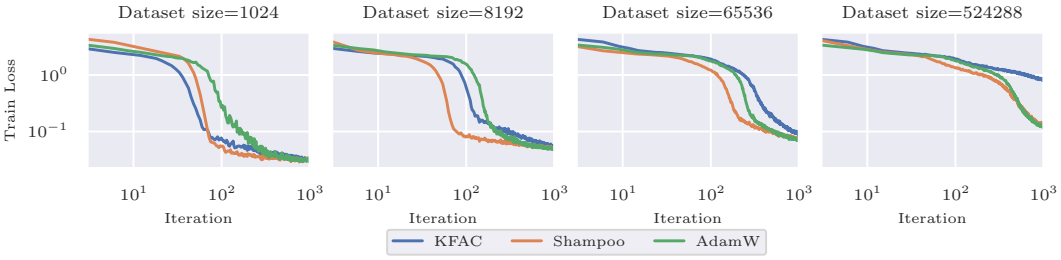

Figure 2: **Train Curve Comparison of optimizers for different dataset sizes in training character-level language modeling.** We trained minGPT on the BookCorpus dataset for character-level language modeling. When the dataset size is small, K-FAC and Shampoo converge faster than AdamW. However, as the dataset size increases, the convergence rate of K-FAC deteriorates.

In training language models, the dataset size is typically huge, and as a result, usually only about one epoch is trained. This implies that LLM is trained with relatively small mini-batch sizes, making it challenging to observe the advantages of second-order optimization compared to first-order optimization. As shown in Fig.2, when the dataset size is small, K-FAC and Shampoo achieve faster convergence than Adam. However, as the data set size increases, the final convergence destination of K-FAC becomes worse, and the speed of convergence of Shampoo and Adam coincides. This suggests that second-order optimization may not be a better method than first-order optimization when the data set size is too large.

4 Conclusion and Discussion

We investigated the convergence properties of second-order optimization methods concerning batch size and dataset size. Our findings indicate that, with few exceptions, second-order optimization generally surpasses first-order optimization when dealing with larger batch sizes. However, as the size of the dataset becomes excessively large, the convergence rate of second-order optimization becomes comparable to, or may even be less effective than, that of first-order optimization. These findings suggest that the advantages of second-order optimization over first-order optimization might not be as significant in language model optimization. This is because language modeling frequently involves very large datasets and relatively small batch sizes in comparison to the size of the dataset.

ACKNOWLEDGEMENTS

The authors thank Kazuki Osawa for fruitful discussions and support with this research. We also thank Yuji Iguchi for his help in running the program on a computer cluster.

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

S. Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10:251–276, 1998.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable Second Order Optimization for Deep Learning. *arXiv:2002.09018*, 2021.

Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In *International Conference on Machine Learning*, pp. 1817–1853. PMLR, 2022.

Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nhKHA59gXz.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

Runa Eschenhagen, Alexander Immer, Richard E Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. *arXiv preprint arXiv:2311.00636*, 2023.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks. *Advances in Neural Information Processing Systems*, pp. 2386–2396, 2020.

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*, 2017.

Roger Grosse and James Martens. A Kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2016.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade: Second Edition*, pp. 599–619. Springer, 2012.

Mohammad Rasool Izadi, Yihao Fang, Robert Stevenson, and Lizhen Lin. Optimization of graph neural networks with natural gradient descent. In *2020 IEEE international conference on big data (big data)*, pp. 171–179, 2020.

Dayal Singh Kalra and Maissam Barkeshli. Phase diagram of early training dynamics in deep neural networks: effect of the learning rate, depth, and width. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Al9yglQGKj.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint5997*, 2014.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.

Xi-Lin Li. Preconditioner on matrix lie group for sgd. *arXiv preprint*, 2018.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019a.

Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-scale distributed second-order optimization using Kronecker-factored approximate curvature for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12359–12367, 2019b.

Kazuki Osawa, Satoki Ishikawa, Rio Yokota, Shigang Li, and Torsten Hoefler. ASDL: A unified interface for gradient preconditioning in PyTorch. *arXiv:2305.04684*, 2023.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint*, 2013.

Yuichiro Ueno, Kazuki Osawa, Yohei Tsuji, Akira Naruse, and Rio Yokota. Rich information is affordable: A systematic performance analysis of second-order optimization using K-FAC. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2145–2153, 2020.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Minghan Yang, Dong Xu, Zaiwen Wen, Mengyun Chen, and Pengxiang Xu. Sketchy empirical natural gradient methods for deep learning. *arXiv preprint*, 2020.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.

Guodong Zhang, Aleksandar Botev, and James Martens. Deep learning without shortcuts: Shaping the kernel with tailored rectifiers. In *International Conference on Learning Representations*, 2022.

Appendices

A SECOND-ORDER OPTIMIZATION AND BATCH SIZE

A.1 PSGD ON LARGE BATCH TRAINING

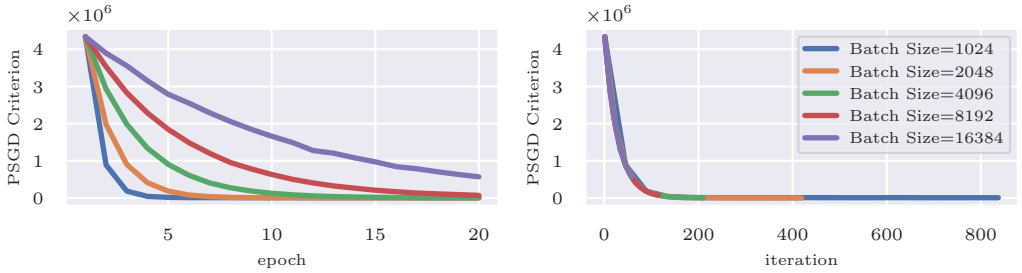

Figure 3: **The quality of curvature is determined by the number of Iterations** According to the criterion, the quality of PSGD's curvature matrix improves as iteration increases. Therefore, the quality of PSGD curvature in iterations is independent of the mini-batch size, but in epochs, the quality of PSGD curvature depends on the mini-batch size.

The accuracy of PSGD may decrease rapidly as the batch size increases. This may occur because PSGD computes the curvature matrix iteratively. To investigate the cause of this deterioration, we use the following criterion, which measures the quality of the Preconditioner matrix $\boldsymbol{P}$ as employed in the PSGD paper(Li, 2017; 2018).

$$c(\boldsymbol{P}) = \mathbb{E}_{\delta\boldsymbol{\theta}}\left[\delta\hat{\boldsymbol{g}}^\top \boldsymbol{P}\delta\hat{\boldsymbol{g}} + \delta\boldsymbol{\theta}^\top \boldsymbol{P}^{-1}\delta\boldsymbol{\theta}\right] \tag{1}$$

where $\delta\boldsymbol{\theta}$ represents the perturbation of parameters and $\delta\hat{\boldsymbol{g}}$ represents the perturbation of gradients. A smaller PSGD criterion value indicates a higher quality of the curvature matrix. Figure.3 illustrates the PSGD criterion as the batch size changes. When the horizontal axis represents epochs, the PSGD criterion increases with the batch size. However, when the horizontal axis represents iterations, the decrease in the criterion appears to be independent of the batch size. This suggests that the quality of the curvature matrix may not improve sufficiently with larger batch sizes due to the reduced number of iterations compared to the same number of epochs.

A.2 EFFECT OF BATCH SIZE ON OTHER HYPERPARAMETERS

In first-order optimization methods, an increase in batch size requires a proportional increase in the learning rate (Hinton, 2012; Krizhevsky, 2014; Goyal et al., 2017). This trend is also observed in second-order optimization. Fig.4 illustrates that second-order optimization methods, including K-FAC, FOOF, and Shampoo, demand proportionally larger damping as the batch size increases.

In addition to the learning rate, second-order optimization methods incorporate a hyperparameter known as the damping term, which is also contingent on the batch size. Specifically, it is observed in Fig.4 that the damping should be reduced as the batch size increases. This may be attributed to the fact that in second-order optimization, the ratio of learning rate to damping effectively represents the actual learning rate. When both the learning rate and damping are concurrently adjusted for specific batch size, higher accuracies are achieved when the learning rate-to-damping ratio remains constant.

A.3 TRAIN ACCURACY IN LARGE BATCH LEARNING OF SECOND ORDER OPTIMIZATION

Fig. 5 shows that second-order optimization methods performs better than first-order optimization methods even in the Training accuracy. In terms of Training Accuracy, AdamW shows almost the same accuracy as other methods in small batch training of Resnet18 on CIFAR10. However, given its poor performance in Test Accuracy, we can say that in this setting AdamW does not generalize as well as other optimization methods.

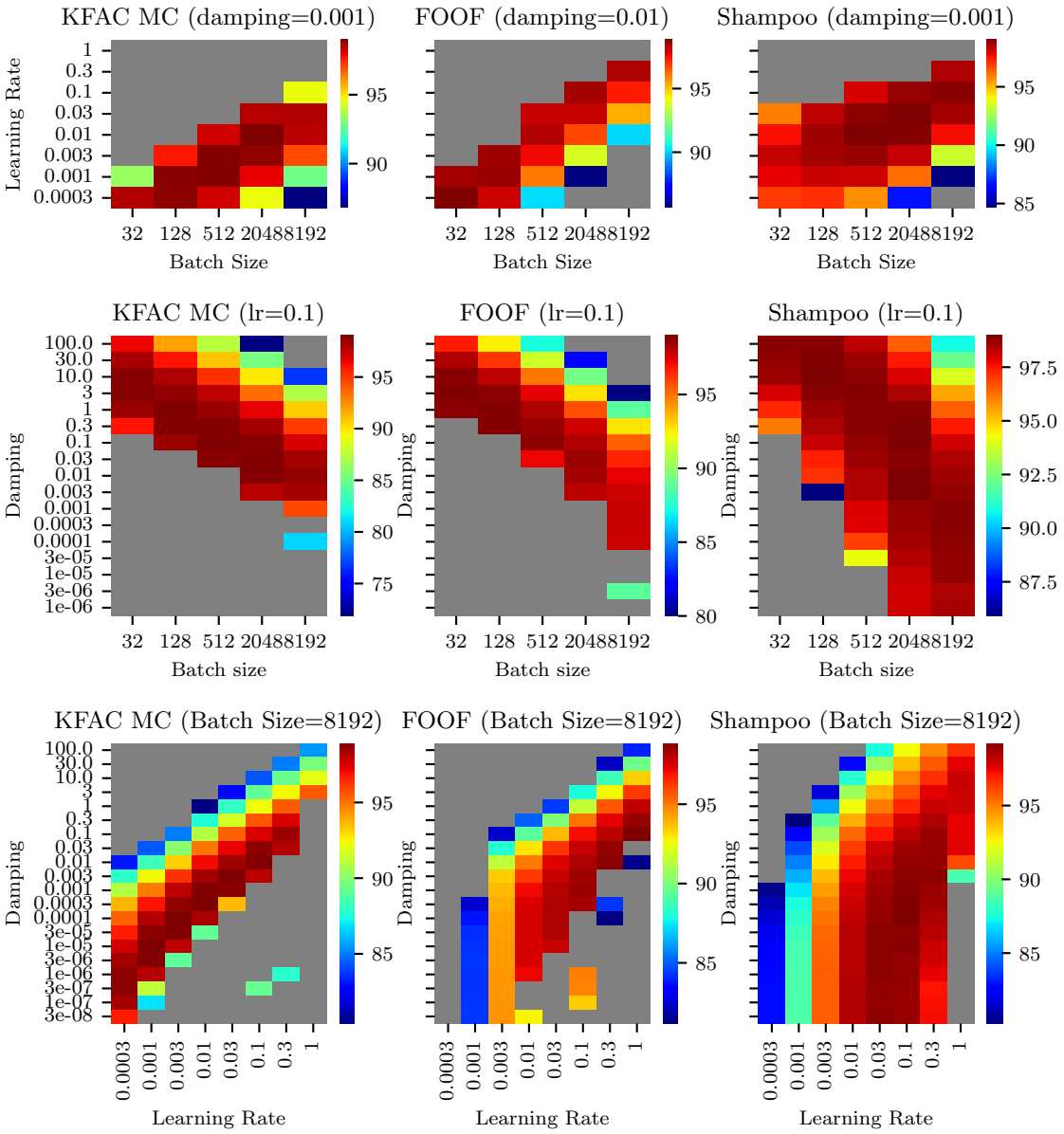

Figure 4: **Learning rate, damping, and batch size are interrelated** We trained a 3-layer MLP on MNIST. We see that the optimal learning rate and damping depend on the batch size.

A.4 Relationship between data set size and critical batch size

When dealing with large datasets, the mini-batch size is often small relative to the entire dataset. Consequently, the benefits of second-order optimization, which are more evident with larger batch sizes, can be minimal or negligible. Figure 6 illustrates how the performance of first-order versus second-order optimization varies with the batch-to-dataset size ratio. With a low ratio, the performance difference between the two optimization methods becomes negligible.

A.5 Large Batch Training in LLM

The results for large-batch learning of GPT2 are shown in Table. 1. The dataset is OpenWebText. Here, we are comparing by fixing the number of samples used for training at 4915200000. In cases of small batch sizes ($\leq$ 1152), AdamW reduces the validation loss more than the second-order

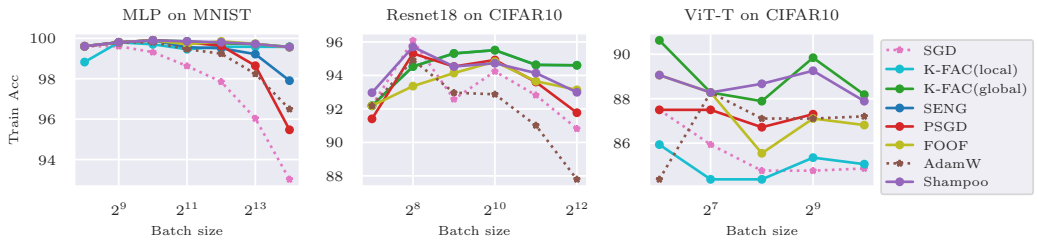

Figure 5: **Second-order optimization performs better than first-order optimization for large batch training even in the Training Accuracy.**

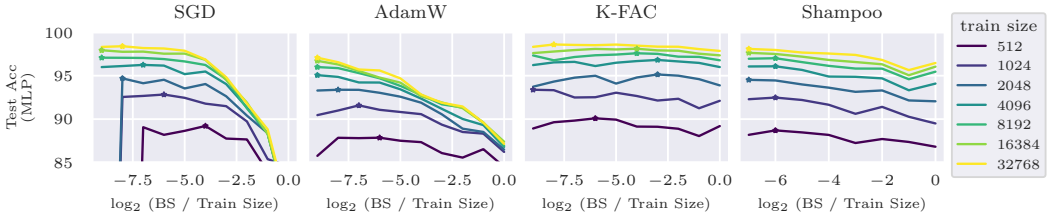

Figure 6: **If the ratio of Batch size to Train size is not sufficiently large, the benefits of second-order optimization are not apparent.** The second-order optimization achieves high accuracy even when the batch size / train size is large. The difference in performance between first-order optimization and second-order optimization depends on the batch size / train size, and its performance gap is negligible when the train size is large. We trained an MLP on MNIST.

optimization method, Shampoo. However, with larger batch sizes ($\geq$ 1536), Shampoo shows a greater reduction in validation loss compared to AdamW. This indicates that in Large Language Models (LLMs), Shampoo, a second-order optimization method, excels in large-batch learning. It is also observed that the batch sizes commonly used are very small, where Shampoo does not perform as effectively.

	Batch Size				
	384	768	1152	1536	1920
AdamW	**3.29**	**3.62**	**3.85**	4.68	5.61
Shampoo	3.43	3.72	3.95	**4.62**	**5.43**

Table 1: **Shampoo reduces validation loss more effectively than AdamW in Large Batch LLM training.** We trained GPT2 on openwebtext with AdamW and Shampoo. Its block size is 1024 and hyperparameters (learning rate and β_2) are tuned using grid search.

B AT WHAT POINT DURING TRAINING IS SECOND-ORDER INFORMATION IMPORTANT?

The curvature matrix calculated by the second-order information matrix is known to fluctuate greatly at the early stage of training, but then stabilizes and changes very littleFort et al. (2020); Kalra & Barkeshli (2023); Damian et al. (2023). In this case, the curvature matrix should be frequently updated in the early stages of learning, but once training has progressed to a certain extent, the intervals for calculating the curvature matrix can be extended, and we may be able to reuse the curvature matrixOsawa et al. (2019a) [1].

[1] In second-order optimization, there are two additional calculations beyond those in first-order optimization: the calculation of the curvature matrix and the calculation of the preconditioner matrix, which is the inverse of the curvature matrix. This section mainly focuses on minimizing the computational cost of the inverse calculation, as the calculation of the curvature matrix is relatively small.

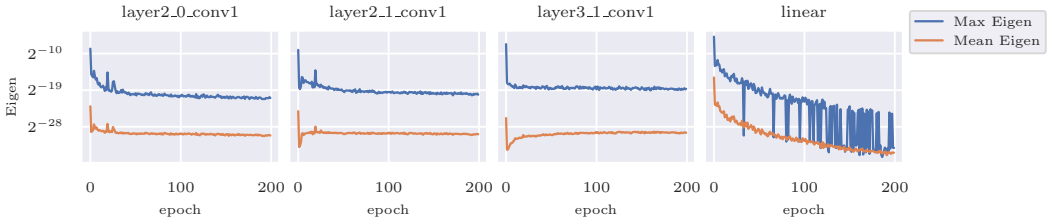

Figure 7: **The curvature matrix is stale during training except for final layer.** The eigenvalues of the curvature in Shampoo are stable for almost all layers except the final linear layer. This graph shows that the curvature matrices for all layers except the final layer hardly need to be updated after the middle stage of training. We trained ResNet18 trained with CIFAR10.

		Percentage of frequent updates (Preconditioner Interval = 10)							
		0%	0.01%	0.03%	0.1%	0.3%	1%	3%	10%
Preconditioner Interval	300	1.3	0.9	79.88	79.75	79.82	79.83	79.98	79.83
	1000	0.82	0.75	79.94	79.52	79.90	79.99	79.89	79.98
	3000	0.48	0.32	0.51	79.50	79.64	80.12	80.08	79.92
	10000	0.33	0.51	0.58	79.69	79.85	79.87	80.06	79.87
	30000	0.12	0.12	0.17	79.34	80.07	80.02	79.99	80.19

Table 2: **In ViT-Pretraining, we can reduce the frequency of computing the inverse matrix.** We trained Deit-small on ImageNet using Shampoo for 300epochs. We can see that if we update the curvature matrix only during 1 epoch (= 0.3% = 2500 iterations), we need not to update the curvature matrix after that. We compute statistics for every 10 iterations.

Figure.7 observes the eigenvalues for each layer during training by Shampoo. The eigenvalues decrease rapidly in the early stages of learning (for 3 5 epochs), while they remain stable after that. Note that this trend is not observed in the final layer. Therefore, the frequency of Preconditioner calculations during training can be greatly reduced after the midpoint of training. Table 2 illustrates the impact on accuracy when Vision Transformers (ViT) are trained using Shampoo, with updates occurring more frequently during the initial stages of training and the frequency of updates diminishing during the middle and later stages. The results indicate that, after the curvature has been adequately calculated for approximately one epoch, the frequency of updates can be significantly reduced thereafter.

C EXPERIMENTAL SETTINGS

We implemented a second-order optimization method based on the ASDL library(Osawa et al., 2023). We indicate the search range for grid search by curly braces {}.

Figure.1 (MLP) We trained a 3-layer MLP on MNIST. We considered a 3-layer multilayer perceptron (MLP) with ReLU activation. The MLP models do not include bias. The width of the middle layer is set to 2048. The hyperparameters are as follows. Grid search was used to find the optimal parameters. Table.3 shows its hyper parameters.

Parameter	Values
Learning Rate	{3e-1, 1e-1, 3e-2, 1e-2, 3e-3, 1e-3}
Momentum	0.9
Epochs	20
Curvature Update Interval for second-order optimization	1
Damping for second-order optimization	{1, 1e-3, 1e-6, 1e-9}
Gradient Clipping Norm	10
Weight Decay	1e-5
Label Smoothing	0.1

Table 3: **Hyper-parameters for MLP on MNIST in Figure.1**

Figure.1 (ResNet18) We trained a ResNet18 on CIFAR10. We used the existing implementation[2] for training CIFAR10. The hyperparameters are as follows. Grid search was used to find the optimal parameters. Table.4 shows its hyper parameters.

Parameter	Values
Learning Rate	{3e-1, 1e-1, 3e-2, 1e-2, 3e-3, 1e-3}
Momentum	0.9
Epochs	100
Curvature Update Interval for second-order optimization	{10, 100}
Damping for second-order optimization	{1, 1e-3, 1e-6, 1e-9}
Gradient Clipping Norm	10
Weight Decay	5e-4
Label Smoothing	0.1

Table 4: **Hyper-parameters for ResNet18 on CIFAR10 in Figure.1**

Figure.1 (ViT-Tiny) We trained a ViT-Tiny pretrained by ImageNet on CIFAR10. We used models in Pytorch Image Models(Wightman, 2019). The hyperparameters are as follows. Grid search was used to find the optimal parameters. Table.5 shows its hyper parameters. Missing point (PSGD with batch size = 1024) is due to the memory consumption.

Parameter	Values
Learning Rate	{3e-1, 1e-1, 3e-2, 1e-2, 3e-3, 1e-3}
Momentum	0.9
Epochs	20
Curvature Update Interval for second-order optimization	{10, 100}
Damping for second-order optimization	{1, 1e-3, 1e-6, 1e-9}
Gradient Clipping Norm	10
Weight Decay	1e-5
Label Smoothing	0.1

Table 5: **Hyper-parameters for ViT-Tiny on CIFAR10 in Figure.1**

[2] https://github.com/uoguelph-mlrg/Cutout

Figure.2 We trained minGPT[3] on the BookCorpus dataset for character-level language modeling. The hyperparameters for training are as follows. Grid search was used to find the optimal parameters. Table.6 shows its hyper parameters.

Parameter	**Values**
Learning Rate	{1, 3e-1, 3e-3, 1e-3}
Momentum	0.9
Batch Size	64
Iterations	1000
Curvature Update Interval (K-FAC, Shampoo)	1
Damping (K-FAC, Shampoo)	1e-3
Dropout	{0, 0.2}
Weight Decay	1e-5
Warmup Iterations	{1, 100}

Table 6: **Hyper-parameters for minGPT on Bookcorpus in Figure.2**

[3]We used this repository : `https://github.com/karpathy/minGPT`. And we used gpt-mini in this repository.