Towards Scalable and Efficient Deep Learning: Structured Optimization Across Training and Fine-tuning

Mingzhe Yu [1] Osamu Tatebe [2]

[1]Graduate School of Science and Technology, University of Tsukuba [2]Center for Computational Sciences, University of Tsukuba, Tsukuba, Ibaraki, Japan

Thesis Question

Deep learning on supercomputing systems faces computation, memory, communication, and reliability constraints. This thesis asks how controlled relaxation can reduce their cost:

Which mathematical structure is worth preserving? → Which strict constraints can be relaxed? → What lightweight compensation keeps training stable?

Core claim: efficient learning comes from controlled relaxation: preserve the signal-carrying structure, relax expensive execution or parameterization constraints, and add minimal mechanisms that prevent drift.

Research Questions and Contributions

RQ	Constraint studied	Contribution
RQ1	Distributed K-FAC under delay, heterogeneity, and failure	AD-KFAC
RQ2	Dense-factor memory, compute, and synchronization cost	BD-KFAC
RQ3	Low-rank adapter geometry during parameter-efficient fine-tuning	RRLoRA

Shared system configuration. All experiments were conducted on Miyabi supercomputer [13]. Miyabi uses one NVIDIA GH200 Grace Hopper Superchip per node and InfiniBand NDR200 links to a full-bisection fat-tree network for inter-node communication and file sharing.

Scale. AD-KFAC: 16 nodes; BD-KFAC: 8 nodes; RRLoRA: 1 node (GLUE), up to 16 nodes (others).

RQ1: AD-KFAC

Problem. Distributed K-FAC [1] is powerful, but synchronous barriers make one slow or failed worker delay the entire curvature path.

Design. AD-KFAC separates model, curvature, and control state. Workers continue local training with versioned curvature caches, exchange factors through peer-to-peer RPC, average models in a decentralized manner, and reassign curvature tasks after load imbalance or failure.

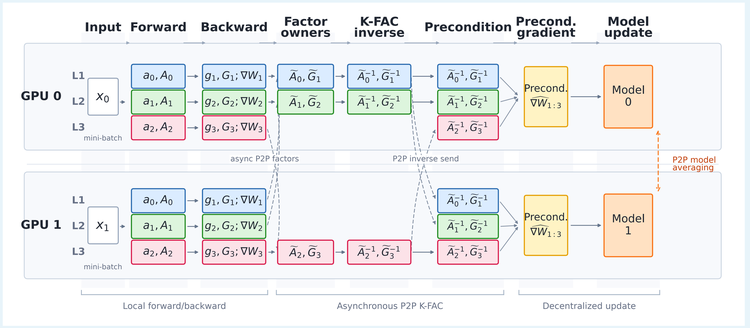


Figure 1. AD-KFAC decouples local training, factor exchange, eigensystems, and model averaging. Adapted from [P1].

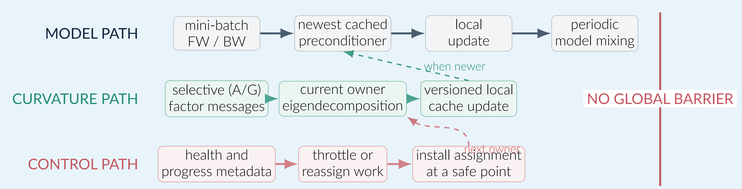


Figure 2. Three State Paths Keep the Foreground Moving. Model updates stay on the foreground path; curvature and control run asynchronously, so workers proceed without a global barrier. Adapted from [P1].

RQ1 Results: ResNet-34 / CIFAR-10

Normal Execution

92.73% final accuracy in **364 s** over 60 epochs.
1.41× speedup and 29.32% less time than KAISA (our synchronous K-FAC baseline) [2].
Curvature remains useful without synchronization.

Injected Latency

With a 90 ms random-worker delay, epoch time rises only ≈ 1 **s**, versus **12.37 s** for KAISA.
Final accuracy remains **92.69%**.

Fault-Tolerant Second-Order State

In fault-injection experiments, workers can temporarily stop computation and communication. AD-KFAC keeps training stable because curvature ownership is recoverable state: cached eigensystems remain usable, and future K-FAC tasks can be reassigned to active workers.

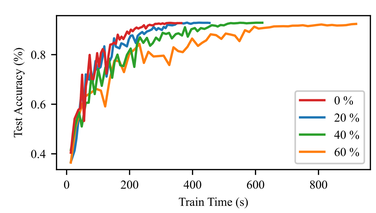


Figure 3a. Test accuracy with respect to training time for AD-KFAC under different fault probabilities. Adapted from [P1].

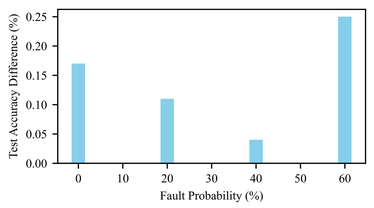


Figure 3b. Difference in final test accuracy between AD-KFAC and SWIFT (our asynchronous first-order baseline) [3] under different fault probabilities. Adapted from [P1].

Mechanism validated: versioned caches + recoverable K-FAC task ownership + decentralized model averaging

RQ2: BD-KFAC

Problem. Standard K-FAC keeps dense Kronecker factors. For wide layers, these factors are expensive to store, decompose, communicate, and apply.

Design. BD-KFAC keeps the layer-wise Kronecker view but replaces each dense factor with block-diagonal covariance:

$$A \approx \mathrm{blockdiag}(A_{S_1}, \ldots, A_{S_m}), \qquad G \approx \mathrm{blockdiag}(G_{T_1}, \ldots, G_{T_n}).$$

The number of blocks interpolates between full-factor K-FAC and diagonal-factor methods.

Precomputed Block-Wise Inverses. At each curvature refresh, BD-KFAC forms damped inverse factors

$$A_i^{-1}(\gamma) = Q_{A,i}(\Lambda_{A,i} + \gamma I)^{-1} Q_{A,i}^\top, \qquad G_j^{-1}(\gamma) = Q_{G,j}(\Lambda_{G,j} + \gamma I)^{-1} Q_{G,j}^\top.$$

It stacks these cached blocks into $\widehat{A}^{-1}$ and $\widehat{G}^{-1}$, then reuses them across optimizer steps:

$$\Delta W_l = -\widehat{G}^{-1} \nabla_{W_l} \mathcal{L}\, \widehat{A}^{-1}.$$

This amortizes repeated spectral transforms and communication between curvature refreshes.

RQ2 Experimental Results

We compare final test accuracy after adjusting every baseline to the same wall-clock training budget.

	CIFAR-10	CIFAR-100						
Optimizer	ResNet-18	ResNet-34	CCT	DenseNet	FocalNet	ResNet-50	Swin-S	Swin-B
BD-KFAC	**94.53**	**77.32**	56.49	**78.11**	57.86	**77.88**	**61.47**	**63.10**
AdamW [4]	94.40	76.33	44.33	76.28	56.56	77.26	59.44	62.61
Shampoo [6]	94.02	76.68	36.34	77.59	**59.93**	77.58	61.33	62.33
AdaFisherW [5]	94.00	75.92	56.59	77.23	47.65	76.85	56.24	57.45
K-FAC [1]	93.70	76.49	55.64	77.04	53.77	76.95	60.84	62.65
Adafactor [7]	85.25	53.66	**57.47**	51.95	40.95	52.07	43.87	47.10

Block-count ablation: ResNet-50 / CIFAR-100; 50 epochs, three seeds.

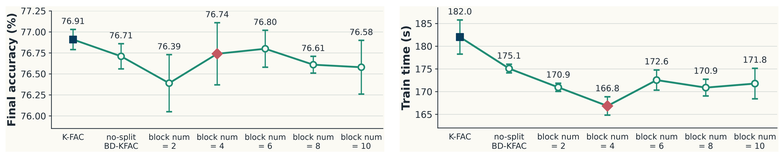


Figure 4. Four blocks: **8.3% less training time** (182.03 to 166.85 s), with a 0.17-point accuracy decrease versus K-FAC. Error bars: standard deviation over three seeds. Post-publication results using an improved BD-KFAC implementation.

References

[1] J. Martens and R. Grosse, "Optimizing Neural Networks with Kronecker-Factored Approximate Curvature," *ICML*, pp. 2408–2417, 2015.
[2] J. G. Pauloski et al., "KAISA: An Adaptive Second-Order Optimizer Framework for Deep Neural Networks," *SC*, 2021.
[3] M. Bornstein et al., "SWIFT: Rapid Decentralized Federated Learning via Wait-Free Model Communication," *ICLR*, 2023.
[4] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *ICLR*, 2019.
[5] D. M. Gomes et al., "AdaFisher: Adaptive Second Order Optimization via Fisher Information," *ICLR*, 2025.
[6] V. Gupta, T. Koren, and Y. Singer, "Shampoo: Preconditioned Stochastic Tensor Optimization," *ICML*, pp. 1842–1850, 2018.
[7] N. Shazeer and M. Stern, "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost," *ICML*, pp. 4603–4611, 2018.

RQ3: RRLoRA

Problem. LoRA [8] reduces trainable parameters, but the factorized update BA creates an optimization geometry whose orientation, scale, and conditioning can make adaptation slow.

Design. Inspired by Muon [9] and LoRA-GA [10], RRLoRA refactorizes adapters at a few early restart boundaries:

$$M = BA \approx U\Sigma V^\top, \qquad B \leftarrow U, \quad A \leftarrow V^\top.$$

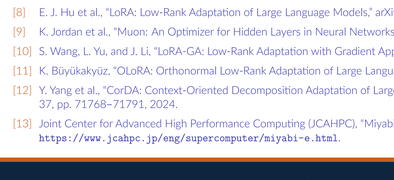
It preserves subspace directions, flattens singular values, resets optimizer moments, and restarts the learning rate.

RQ3 Experimental Results

Benchmark	Backbone / scope	Best RRLoRA vs. best non-RRLoRA	Margin
Commonsense170K	Four LLMs	**89.20** (+Kaiming) vs. 88.94 (OLoRA [11])	+0.26 pt
GLUE	RoBERTa-Large	**89.21** (+Gaussian) vs. 89.11 (orthogonal)	+0.10 pt
MetaMathQA	Qwen3-1.7B SmolLM3-3B	**59.26** (+orthogonal) vs. 59.06 (CorDA [12]) **59.83** (+OLoRA) vs. 58.41 (orthogonal)	+0.20 pt +1.42 pt
ChartQA	Three-VLM avg.	**68.57** vs. 67.02	+1.55 pt
Kvasir-VQA-x1	Three-VLM avg.	**9.1910** vs. 9.1585	+0.0325 pt

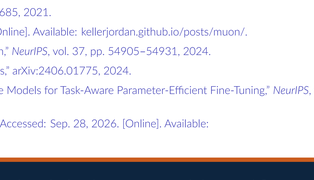
Metrics: accuracy for Commonsense170K, MetaMathQA, and ChartQA; GLUE aggregate score; Kvasir-VQA-x1 LLM judge (0–10); matched budgets and three-seed means.

Figure 5a.



Refactorization drives most of the gain. Across matched runs, **mean runtime overhead is 0.31%**. Adapted from [P3].

Figure 5b.



Blue: standard decay Orange: RRLoRA restarts. Adapted from [P3].

Kvasir-VQA-x1 isolation (LLM judge 0–10; three-seed mean)

Setting	LFM2.5-VL	Ministral
LoRA Kaiming	9.0049	9.0581
Restart-only	9.0264	9.0801
Refactor-only	9.1320	9.1181
Full RRLoRA	**9.1592**	**9.1661**

Takeaways: Controlled Relaxation

Method	Relaxed constraint	Preserved signal	Compensation
AD-KFAC	global curvature freshness	K-FAC preconditioning	versions, caches, reassignment
BD-KFAC	dense within-factor covariance	local coordinate coupling	block-wise inverses
RRLoRA	uninterrupted LoRA factor trajectory	learned low-rank subspace	SVD refactorization + restart

Preserve useful structure; relax costly constraints. scalable optimizers should preserve the structure that carries optimization signal, relax the part that creates unnecessary system cost, and make the relaxation observable and recoverable.

Limitations. Evidence covers the reported fault-injection, CIFAR-10/100, language, reasoning, and vision–language settings; broader scales and failure modes remain open.

Research Artifacts and Acknowledgements[AI]

Publications supporting the dissertation

[P1] M. Yu and O. Tatebe, "AD-KFAC: Asynchronous Decentralized Distributed K-FAC with Dynamic Load Balancing and Fault Tolerance," in *2025 10th International Conference on Machine Learning Technologies (ICMLT)*, pp. 373–382, 2025, doi: 10.1109/ICMLT65785.2025.11193392.
[P2] M. Yu and O. Tatebe, "Block-Diagonal K-FAC: A Trade-off Between Curvature Information and Resource Efficiency," in *OPT 2025: Optimization for Machine Learning*, 2025, [Online]. Available: OpenReview.
[P3] M. Yu and O. Tatebe, "RRLoRA: Refactorized Low-Rank Adaptation With Learning-Rate Restarts for Efficient Fine-Tuning," *IEEE Access*, vol. 14, pp. 84922–84942, 2026, doi: 10.1109/ACCESS.2026.3699357.

Acknowledgements. This work is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and was supported by JSPS KAKENHI Grant Number JP22H00509, JST CREST JPMJCR24R4, the Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba, and Fujitsu Limited. This work is also supported by JST SPRING Grant Number JPMJSP2124.

AI disclosure. OpenAI Codex (GPT-5.6 sol) assisted with English editing and layout of this canvas. The authors reviewed the canvas and take responsibility for the accuracy and integrity of its content.

[AI] OpenAI, "Codex," GPT-5.6 sol model, accessed Aug. 6, 2026. [Online]. Available: openai.com/codex.

References

[8] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685, 2021.
[9] K. Jordan et al., "Muon: An Optimizer for Hidden Layers in Neural Networks," 2024. [Online]. Available: kellerjordan.github.io/posts/muon/.
[10] S. Wang, L. Yu, and J. Li, "LoRA-GA: Low-Rank Adaptation with Gradient Approximation," *NeurIPS*, vol. 37, pp. 54905–54931, 2024.
[11] K. Büyükakyüz, "OLoRA: Orthonormal Low-Rank Adaptation of Large Language Models," arXiv:2406.01775, 2024.
[12] Y. Yang et al., "CorDA: Context-Oriented Decomposition Adaptation of Large Language Models for Task-Aware Parameter-Efficient Fine-Tuning," *NeurIPS*, vol. 37, pp. 71768–71791, 2024.
[13] Joint Center for Advanced High Performance Computing (JCAHPC), "Miyabi (OFP-2)," Accessed: Sep. 28, 2026. [Online]. Available: `https://www.jcahpc.jp/eng/supercomputer/miyabi-e.html`.