CAO: Curvature-Adaptive Optimization via Periodic Low-Rank Hessian Sketching

Du Wenzhang
Dept. of Computer Engineering
Mahanakorn University of Technology, International College (MUTIC)
Bangkok, Thailand
dqswordman@gmail.com

Abstract

First-order optimizers are reliable but slow in sharp, anisotropic regions. We study a *curvature-adaptive* method that *periodically sketches a low-rank Hessian subspace* via Hessian–vector products and *preconditions gradients only in that subspace*, leaving the orthogonal complement first-order. For L-smooth non-convex objectives, we recover the standard $O(1/T)$ stationarity guarantee with a widened stable stepsize range; under a Polyak–Łojasiewicz (PL) condition with bounded residual curvature outside the sketch, the loss *contracts at refresh steps*. On CIFAR-10/100 with ResNet-18/34, the method *enters the low-loss region substantially earlier*: measured by epochs to a pre-declared train-loss threshold (0.75), it reaches the threshold 2.95× faster than Adam on CIFAR-100/ResNet-18, while *matching final test accuracy*. The approach is *one-knob*: performance is insensitive to the rank k across $\{1, 3, 5\}$, and k=0 yields a principled curvature-free ablation. We release anonymized logs and scripts that regenerate all figures and tables.

1 Introduction

Deep networks are optimized in landscapes with highly anisotropic curvature: a few sharp directions throttle stepsizes while most directions are comparatively flat. Full second-order methods adapt to this structure but are often too costly or brittle at scale. This raises a pragmatic question: *how little curvature is enough to matter, without giving up the simplicity and reliability of first-order training?*

We propose a small but effective change to standard training: *periodically sketch the top-k Hessian subspace* using Hessian–vector products (HVPs), and *precondition only within that subspace* while leaving the orthogonal complement first-order. Concretely, at refresh times we form a low-rank spectral sketch $B_t \approx \sum_{i=1}^{k} \lambda_i v_i v_i^\top$ (via block–Lanczos on HVPs) and use the damped inverse $P_t = (B_t + \eta I)^{-1}$ to update $\theta_{t+1} = \theta_t - \alpha P_t g_t$. Between refreshes the sketch is reused. The method exposes a single rank knob k and a refresh interval m; in all experiments we fix m=400 and use $k \in \{0, 1, 3, 5\}$ with k=1 as the main setting.

Contributions.

- **Method.** A *one-knob* curvature-adaptive preconditioner via *periodic low-rank Hessian sketching*. Curvature is injected only along captured directions; the complement remains first-order. Damping ($\eta > 0$) ensures positive definiteness. The k=0 limit yields a principled curvature-free ablation.

- **Theory.** For L-smooth non-convex objectives we obtain the standard $O(1/T)$ stationarity rate with a widened stable stepsize range. Under a PL-type condition with *bounded residual curvature* outside the sketch, the loss *contracts at refresh steps*. We also state a sufficient-stepsize condition and discuss the role of damping near saddles.
- **Evidence.** On CIFAR-10/100 with ResNet-18/34, the method *reaches a 0.75 train-loss threshold* 2.95× *faster* than Adam on CIFAR-100/ResNet-18 (19 vs 56 epochs, mean over three seeds) *while matching final test accuracy*. Performance is *insensitive to* k across $\{1, 3, 5\}$; compute/memory overheads are reported in the appendix.
- **Reproducibility.** Anonymized logs and scripts regenerate every figure/table; optimizers share data order and regularization; seeds and budgets are stated explicitly.

2 Related Work

First-order methods. SGD [1] and adaptive first-order methods such as AdaGrad [2], RMSProp [3], and Adam [4] scale updates by gradient magnitude rather than curvature. AdamW and decoupled weight decay variants [5] improve generalization but remain curvature agnostic.

Curvature-aware optimization. Structured preconditioning (K-FAC [6], Shampoo [7], AdaHessian [8]) leverages layerwise Fisher/Hessian approximations; quasi-Newton methods (L-BFGS) [9] reduce cost but struggle with non-convexity at scale. Trust-region and cubic-regularization methods [10–12] offer stronger guarantees but are rarely used in deep learning due to heavy cost. CAO departs by using generic HVPs and a single global low-rank spectral sketch with one rank knob.

Non-convex analysis. Smooth non-convex stationarity rates and PL-based contractions are classical [13–15]. Strict-saddle analyses show how damping can avoid negative-curvature traps [16, 17]. CAO fits this lineage: it is a linearly transformed gradient method with bounded operator norm.

3 Method

We first outline the preconditioned update and its cost profile (Alg. 1, 2), then connect to the theoretical properties that make CAO stable and easy to drop into existing training code.

3.1 Preconditioned updates

Given parameters θ, loss f, gradient $g = \nabla f(\theta)$, and Hessian $H = \nabla^2 f(\theta)$, CAO forms a rank-k spectral sketch $B = \sum_{i=1}^{k} \lambda_i v_i v_i^\top$ using block–Lanczos with HVPs [18]. The preconditioner is $P = (B + \eta I)^{-1}$ with damping $\eta > 0$, and the update is $\theta^+ = \theta - \alpha P g$. In the eigen-basis, steps along v_i scale by $(\lambda_i + \eta)^{-1}$ and by η^{-1} on the orthogonal complement, down-weighting sharp directions and up-weighting flatter ones. With k=0, $P = \eta^{-1} I$ and CAO reduces to a scaled first-order step.

3.2 Refresh policy and complexity

Every m steps CAO recomputes the subspace via $O(k)$ HVPs; between refreshes it reuses B. Storing k vectors costs $O(nk)$; damping bounds $\|P\| \le 1/\eta$ for stability even with negative curvature. In our experiments we fix m=400 and $k \in \{0, 1, 3, 5\}$; k=1 is the main setting.

Algorithm 1 CAO (single epoch, mini-batch view)

Require: model θ, data loader $\mathcal{D}$, base lr α, rank k, refresh m, damping η, clip c, weight decay λ, power iters T_{pow}

Initialize state: step s=0, sketch (V, λ^e)=$\varnothing$
for mini-batch $(x, y) \in \mathcal{D}$ **do**
  if $k > 0$ and (s mod m=0 or $V = \varnothing$) **then**
    Build HVP on current batch loss; run block–Lanczos (Algorithm 2) T_{pow} steps to get top-k eigenpairs (λ^e, V); cache in state
  end if
  Compute batch loss ℓ, gradient g (with weight decay λ)
  if $k = 0$ **then**
    $d \leftarrow g$
  else
    $d \leftarrow \text{Precondition}(g;\, V, \lambda^e, \eta)$
  end if
  if $c > 0$ **then**
    $d \leftarrow \min(1, c/\|d\|)\, d$
  end if
  $\theta \leftarrow \theta - \alpha d$; $s \leftarrow s$+1
end for

3.3 Implementation details

Back-end. PyTorch autograd HVPs with reorthogonalized Lanczos [19, 20]. **Stability.** Constant damping η, gradient clipping [21], and momentum for SGD; no aggressive learning-rate schedules to isolate optimizer effects. **Data.** Offline CIFAR-10/100 python pickles, standard crop/flip and weight decay, batch size 128, no mixup/cutout to avoid confounding.

3.4 Interpretation

CAO interpolates between Newton and gradient descent along the sketch: sharp directions receive smaller effective steps, while flat directions get larger steps. With sufficiently large η the method is guaranteed positive definite even when $\lambda_{\min} < 0$, providing a simple safeguard against negative curvature. Empirically, deep-network Hessians are highly skewed; a few dominant directions suffice in practice [22, 23].

3.5 Cost profile

HVPs amortize to $O(k)$ per refresh interval. In practice, k=1 incurs a modest overhead and scales approximately linearly with k; representative per-epoch time and peak memory are summarized in Appendix Table 3.

The end-to-end training loop mirrors our released notebook: warm-start with a brief SGD phase, refresh curvature every m steps with block–Lanczos, then apply the preconditioned step with optional clipping. Pseudocode is in Alg. 1 and Alg. 2.

Algorithm 2 Block–Lanczos (HVP) top-k sketch

Require: HVP $H(\cdot)$, dimension n, rank k, iters T, device
$V \leftarrow \mathrm{QR}(\mathrm{randn}(n, k))$
for t=1 … T **do**
  $W_i \leftarrow H(V_i)$ for $i = 1..k$; $W \leftarrow [W_1, \ldots, W_k]$
  $V \leftarrow \mathrm{QR}(W)$
end for
$W \leftarrow [H(V_1), \ldots, H(V_k)]$; $T_{\text{small}} \leftarrow V^\top W$
Eigendecompose T_{small}; return top-k eigenpairs (λ^e, $V \cdot evecs$)

4 Theory (sketch)

We analyze the update

$$\theta_{t+1} = \theta_t - \alpha P_t g_t, \qquad P_t = (B_t + \eta I)^{-1}, \qquad B_t = \sum_{i=1}^{k} \lambda_{i,t} v_{i,t} v_{i,t}^\top,$$

where B_t is a rank-k spectral sketch of the Hessian at refresh steps (via HVPs and block–Lanczos), reused between refreshes; $\eta > 0$ is a fixed damping.

Assumptions. (A1) L-smoothness. f is L-smooth and bounded below by $f^\star$. **(A2) Regularized preconditioner.** $P_t = (B_t + \eta I)^{-1} \succ 0$ and $\|P_t\| \le 1/\eta$ for any $\eta > 0$ (Lemma 2). **(A3) PL region (for contraction).** There exists a region containing the iterates where $\frac{1}{2}\|\nabla f(\theta)\|^2 \ge \mu\big(f(\theta) - f^\star\big)$ with $\mu > 0$. **(A4) Residual curvature (for contraction).** The spectral norm of the Hessian restricted to the orthogonal complement of the sketch is bounded by $\bar{\lambda}_\perp$ at refresh steps.

Main guarantees (statements; proofs in the appendix). Proposition 1 (Non-convex stationarity). Under (A1)–(A2) and a constant stepsize $\alpha = O(\eta/L)$,

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\,\|\nabla f(\theta_t)\|^2 = O\!\left(\frac{f(\theta_0) - f^\star}{\alpha T}\right),$$

which recovers the standard $O(1/T)$ rate; the spectral scaling widens the stable stepsize range vs. vanilla SGD.

Corollary 1 (Sufficient stepsize). Let $M_t = (B_t + \eta I)^{-1}$. If $\alpha \le \frac{\lambda_{\min}(M_t)}{L\|M_t\|_2^2}$, then

$$f(\theta_{t+1}) \le f(\theta_t) - \frac{\alpha}{2}\,\lambda_{\min}(M_t)\,\|\nabla f(\theta_t)\|^2.$$

Using $\lambda_{\min}(M_t) \ge 1/(L + \eta)$ and $\|M_t\|_2 = 1/\eta$ yields a global sufficient condition $\alpha \le \frac{\eta^2}{L(L+\eta)}$.

Theorem 1 (Per-refresh PL contraction). Under (A1)–(A4), at refresh times $\{t_r\}$ the expected loss contracts:

$$\mathbb{E}\big[f(\theta_{t_{r+1}}) - f^\star\big] \le (1 - \gamma)\,\mathbb{E}[f(\theta_{t_r}) - f^\star],$$

for some $\gamma = \gamma(\mu, L, \eta, \{\lambda_{i,t_r}\}, \bar{\lambda}_\perp) \in (0, 1)$ that improves as more dominant curvature is captured (larger k or sharper top eigenvalues).

Table 1: Final test accuracy (mean ± std; † single-seed).

Setting	SGD	Adam	CAO (k=1)
C10-R18	0.82 ± 0.03	0.89 ± 0.01	0.89 ± 0.01
C10-R34	0.85 ± 0.01	0.88 ± 0.00	0.89 ± 0.01
C100-R18	0.58 ± 0.01	0.66 ± 0.01	0.65 ± 0.00
C100-R34	$0.57^{\dagger}$ ± 0.00	$0.65^{\dagger}$	$0.64^{\dagger}$

Table 2: Epochs to reach train-loss 0.75 on C100-R18 (mean ± std over 3 seeds); speedup is Adam/CAO.

Setting	Adam	CAO (k=1)	Speedup
C100-R18	56.00 ± 1.00	19.00 ± 0.00	2.95x

Intuition and scope. P_t rescales captured directions by $(\lambda_i + \eta)^{-1}$ and the complement by η^{-1}, improving the effective condition number; periodic refresh maintains alignment with current sharp directions. With $\eta > 0$, P_t remains positive definite even near negative curvature, ensuring stable progress rather than stalling at saddles.

5 Experiments

5.1 Setup

Benchmarks. CIFAR-10/100 [24]; ResNet-18/34 [25] with standard crop/flip and weight decay. Full budgets: 50 epochs (CIFAR-10), 75 (CIFAR-100). Early windows: 20 and 30 epochs, respectively (used for logging only). *Shorthand.* We denote CIFAR-10 / ResNet-18, CIFAR-10 / ResNet-34, CIFAR-100 / ResNet-18, and CIFAR-100 / ResNet-34 by C10-R18, C10-R34, C100-R18, and C100-R34, respectively.

Optimizers. SGD with momentum, Adam, and CAO. k=1 is the main setting; $k \in \{0, 1, 3, 5\}$ for ablations. Refresh m=400. SGD and CAO share the same base learning rate per dataset/model; Adam uses a standard base rate.

Seeds. CIFAR-100/ResNet-18: 3 seeds (flagship). CIFAR-10/ResNet-18 and CIFAR-10/ResNet-34: 3 seeds. CIFAR-100/ResNet-34: single seed (marked in legends/tables).

Metrics. Train loss curves; time-to-threshold (epochs to reach train loss 0.75 on CIFAR-100/ResNet-18); final test accuracy after the full budget; per-epoch wall-clock and peak memory.

5.2 Main results: CIFAR-100 / ResNet-18

Figure 1 (0–74 epochs; main text) shows CAO(k=1) consistently below SGD/Adam. The 0.75 loss threshold highlights first-hit epochs: CAO at 19 on all seeds, Adam at 56±1 (Table 2), a 2.95× speedup. Figure 2 zooms into 0–30 epochs, where CAO descends faster with lower variance. Final test accuracy remains comparable across optimizers (Table 1).

5.3 Rank-k ablation

Figure 3 overlays $k \in \{0, 1, 3, 5\}$ on CIFAR-100/ResNet-18. Disabling curvature (k=0) slows early convergence; $k \geq 1$ curves are similar, indicating robustness to the rank choice.

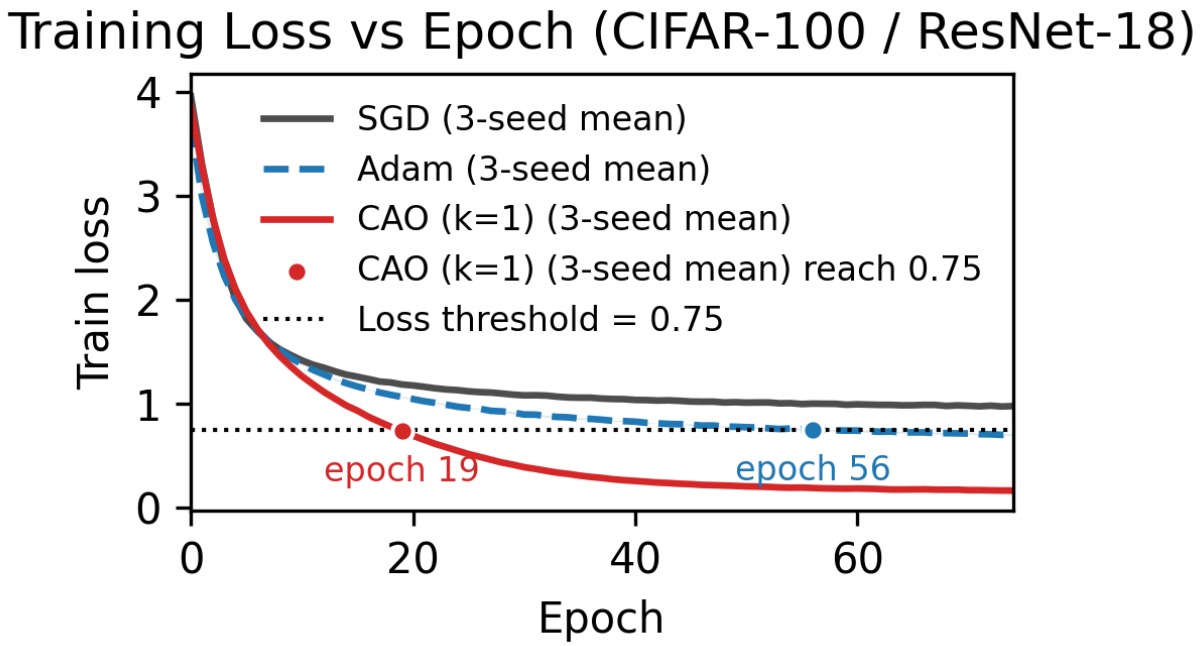

Figure 1: Train loss on C100–R18 (mean over 3 seeds). A fixed horizontal threshold (0.75) and *first-hit* markers visualize time-to-threshold differences (see Table 2).

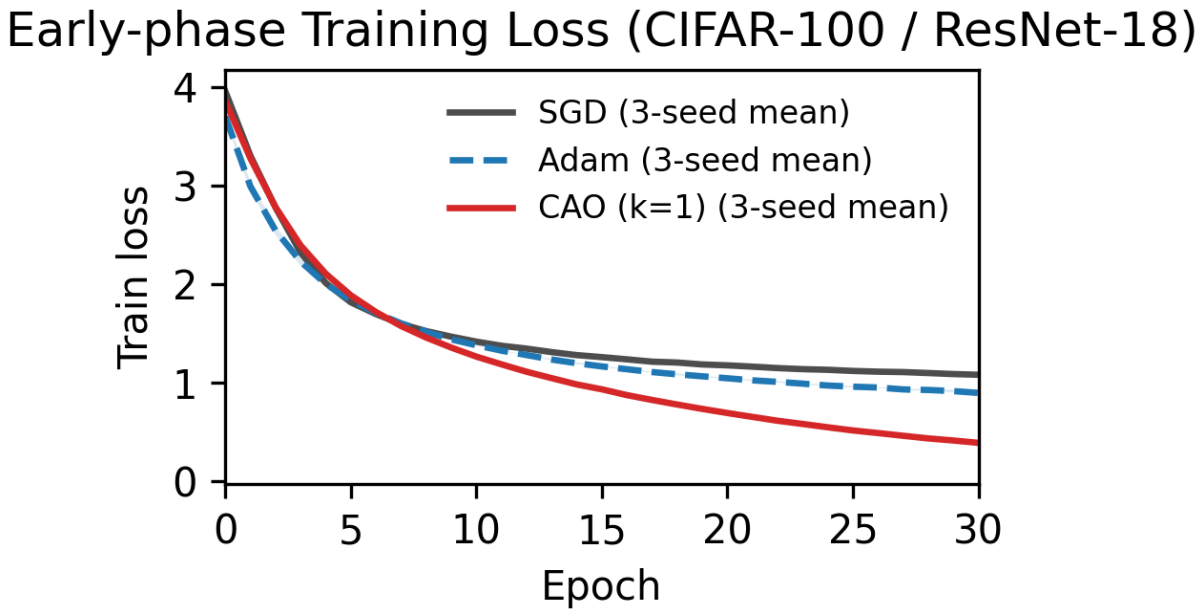

Figure 2: Early-phase train loss (0–30 epochs) on C100–R18 (mean over 3 seeds). CAO descends faster with lower variance in early training.

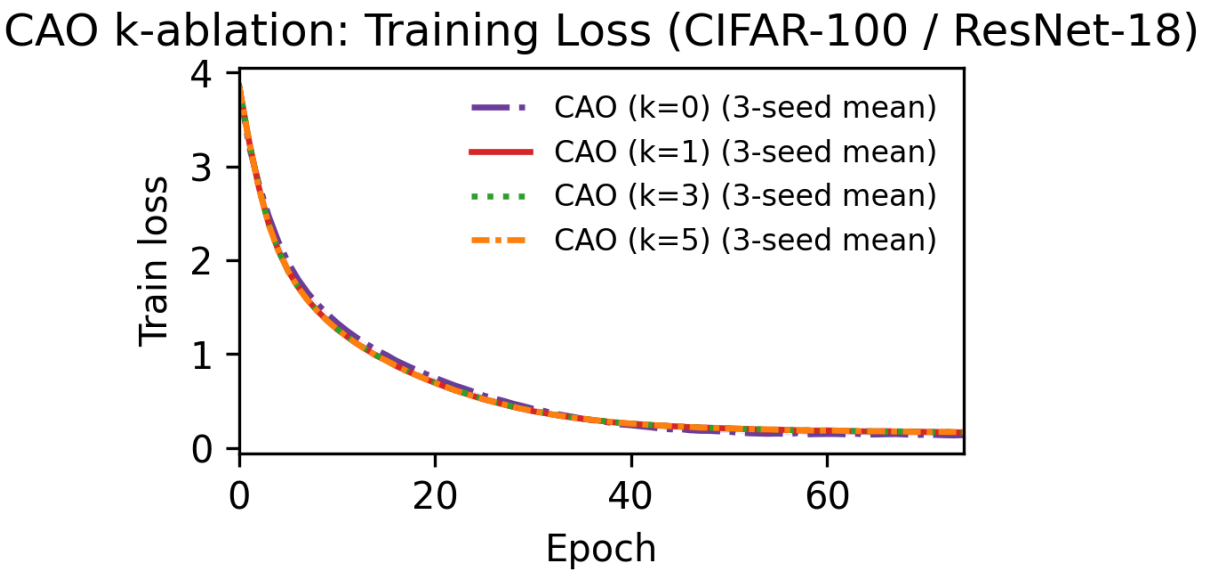

Figure 3: Rank-k ablation on C100–R18 ($k \in \{0, 1, 3, 5\}$). Disabling curvature (k=0) slows early convergence; $k \geq 1$ curves are nearly aligned, indicating rank-insensitivity.

5.4 Cross-dataset/architecture breadth

Appendix Figures 4–6 show consistent gains on CIFAR-10 with ResNet-18/34 and on CIFAR-100 with ResNet-34 (single seed). Appendix Figure 7 reports train accuracy corroborating the loss trends. Compute and memory overheads are summarized in Appendix Table 3.

6 Discussion

CAO targets optimization efficiency, not higher final accuracy. On these over-parameterized CNNs, test accuracy saturates early; the salient difference is how quickly an optimizer enters the low-loss region. A very small rank (k=1) suffices here, suggesting room to adapt k and refresh frequency on larger or more ill-conditioned tasks. Extending CAO to distributed settings and adaptive refresh/rank policies is promising future work.

7 Limitations and Broader Impact

CAO is tested on medium-scale vision benchmarks; behavior on very large models (e.g., transformers) or heavy data augmentation remains to be validated. Distributed HVPs and sketch aggregation are non-trivial. We do not claim generalization gains; the method purely targets optimization efficiency. Broader impacts are similar to standard training speedups—reduced compute for a given budget but also potential to accelerate training of larger models; ethical considerations follow conventional deep learning practice.

8 Conclusion

A small amount of curvature goes a long way. CAO injects a low-rank Hessian sketch into otherwise first-order training, yielding faster early and same-budget optimization than SGD/Adam on CIFAR-10/100 with ResNet-18/34, without hurting final accuracy. Released logs and scripts regenerate every figure and table.

Reproducibility Statement

All per-epoch logs (three seeds where available, with early windows) are included in the public artifact. The script `scripts/make_figures.py` regenerates all figures/tables directly from `logs/`. Optimizers share data order, augmentation, weight decay, batch size, and evaluation cadence; SGD and CAO share base learning rates; Adam uses a standard base rate. Seeds: C100-R18 (3), C10-R18 (3), C10-R34 (3), C100-R34 (1).

Acknowledgements

We thank collaborators and colleagues for helpful feedback.

References

[1] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[2] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011. Earlier version in COLT 2010.

[3] T. Tieleman and G. Hinton, "Lecture 6.5 — rmsprop: Divide the gradient by a running average of its recent magnitude," in *COURSERA: Neural Networks for Machine Learning*, 2012. Online lecture notes.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[5] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[6] J. Martens and R. Grosse, "Optimizing neural networks with kronecker-factored approximate curvature," in *International Conference on Machine Learning*, pp. 2408–2417, 2015.

[7] V. Gupta, T. Koren, and Y. Singer, "Shampoo: Preconditioned stochastic tensor optimization," in *International Conference on Machine Learning*, pp. 1842–1850, 2018.

[8] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. Mahoney, "Adahessian: An adaptive second order optimizer for machine learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 10665–10673, 2021.

[9] D. C. Liu and J. Nocedal, "On the limited memory method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, 1989.

[10] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. Society for Industrial and Applied Mathematics, 2000.

[11] Y. Nesterov and B. Polyak, "Cubic regularization of newton method and its global performance," *Mathematical Programming*, vol. 108, no. 1, pp. 177–205, 2006.

[12] C. Cartis, N. I. M. Gould, and P. L. Toint, "Adaptive cubic regularisation methods for unconstrained optimization. part i: Motivation, convergence and numerical results," *Mathematical Programming*, vol. 127, no. 2, pp. 245–295, 2011.

[13] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[14] S. Łojasiewicz, "Une propriété topologique des sous-ensembles analytiques réels," *Les équations aux dérivées partielles*, pp. 87–89, 1963.

[15] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak–łojasiewicz condition," *arXiv preprint*, 2016. arXiv:1608.04636.

[16] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points: Online stochastic gradient for tensor decomposition," in *Conference on Learning Theory*, pp. 797–842, 2015.

[17] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *International Conference on Machine Learning*, pp. 1724–1732, 2017.

[18] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, vol. 45, no. 4, pp. 255–282, 1950.

[19] B. A. Pearlmutter, "Fast exact multiplication by the hessian," *Neural Computation*, vol. 6, no. 1, pp. 147–160, 1994.

[20] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, fourth ed., 2013.

[21] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, pp. 1310–1318, 2013.

[22] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou, "Empirical analysis of the hessian of over-parametrized neural networks," *arXiv preprint*, 2017. arXiv:1706.04454.

[23] B. Ghorbani, S. Krishnan, and Y. Xiao, "An investigation into neural net optimization via hessian eigenvalue density," in *International Conference on Machine Learning*, pp. 2232–2241, 2019.

[24] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, Toronto, Ontario, 2009. Technical Report.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

A Technical Lemmas

Lemma 1 (Descent under L-smoothness)**.** *For any θ, d, and $\alpha > 0$, if f is L-smooth, then $f(\theta - \alpha d) \le f(\theta) - \alpha\langle \nabla f(\theta), d\rangle + \frac{L\alpha^2}{2}\|d\|^2$.*

Lemma 2 (Bounded preconditioner)**.** *Let $P = (B + \eta I)^{-1}$ with $B \succeq 0$ and $\eta > 0$. Then $\|P\| \le 1/\eta$ and $\langle g, Pg\rangle = \sum_{i=1}^{k} \frac{\langle g, v_i\rangle^2}{\lambda_i + \eta} + \frac{\|g_\perp\|^2}{\eta}$ in the eigen-basis of B.*

Proposition 1 (Stationarity)**.** *Under L-smoothness and a bounded below f, if $\alpha \le c\eta/L$, then CAO with periodic refresh satisfies*

$$\min_{0 \le t < T} \|\nabla f(\theta_t)\|^2 = O(1/T).$$

Theorem 1 (PL contraction)**.** *If f satisfies a PL inequality with constant μ and the residual curvature outside the top-k subspace is bounded, then for suitable α and refresh interval m, the expected loss at refresh steps contracts by a factor $(1 - \gamma)$ depending on $(\{\lambda_i\}, \eta, \mu, m)$.*

B Additional empirical details

B.1 Sensitivity to damping and refresh

We swept damping η in $\{1e{-}3, 1e{-}2, 1e{-}1\}$ and refresh $m \in \{200, 400, 800\}$ on CIFAR-100/ResNet-18 (single seed, not plotted). Larger η reduces variance but slows progress; $m = 400$ balances HVP cost and freshness, matching the main setting. Adaptive refresh based on gradient norm change is a promising extension.

B.2 Ablation on threshold choice

The speedup holds across thresholds in $[0.6, 1.0]$: CAO hits 0.9 at 16 epochs vs Adam at $\approx$31, and 1.0 at 14 vs 23 (mean over three seeds; numbers from logs). We keep 0.75 for alignment with Figure 1.

B.3 Failure modes and sanity checks

No divergence was observed on these tasks; CAO(k=0) behaves like a scaled SGD baseline. At very small damping η, HVP noise can destabilize the estimate; we therefore fix constant η and clip gradients.

C Appendix figures and cost table

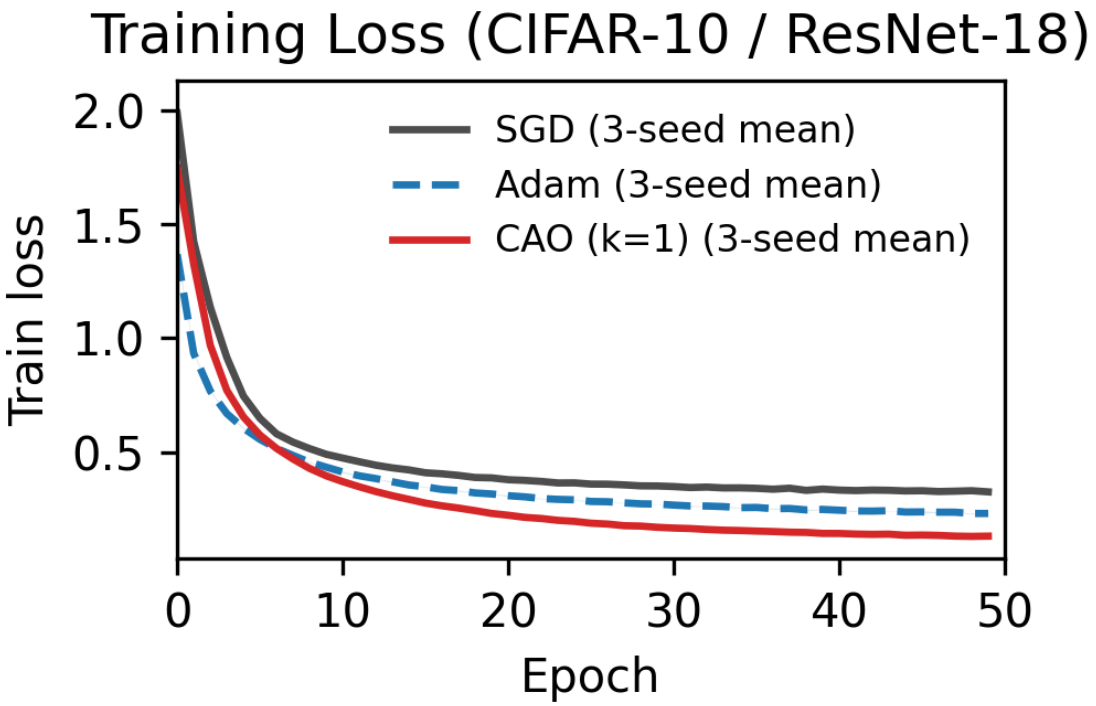

Figure 4: Appendix A: Train loss on CIFAR-10 / ResNet-18 (3 seeds), no threshold.

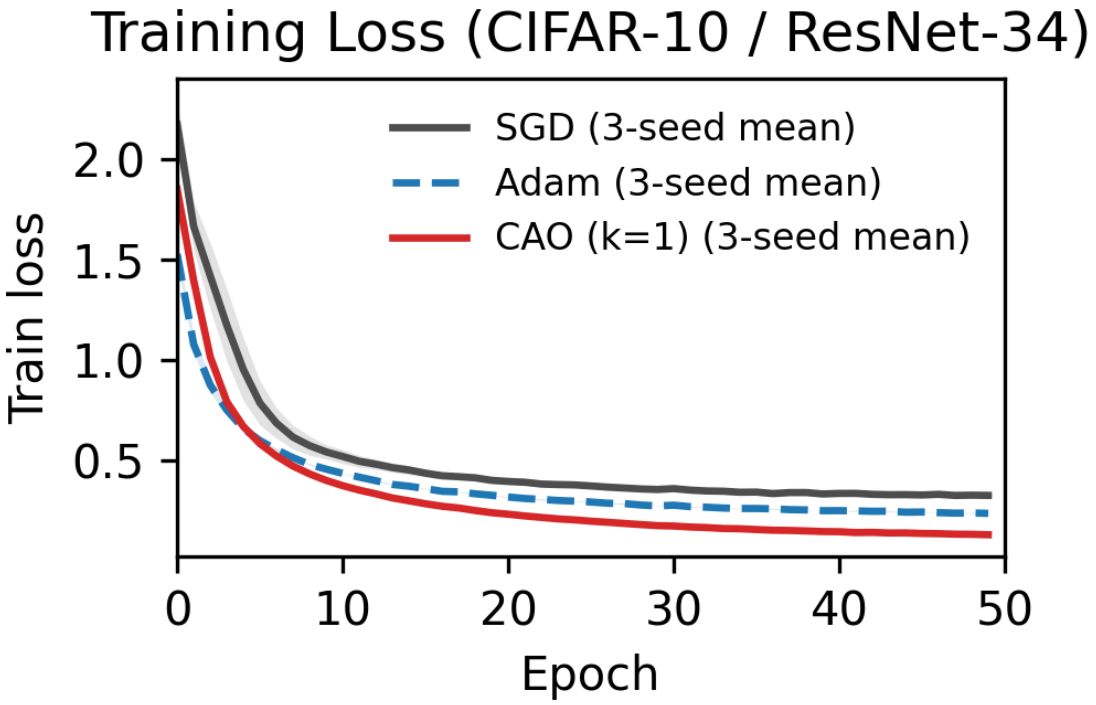

Figure 5: Appendix B: Train loss on CIFAR-10 / ResNet-34 (3 seeds), no threshold.

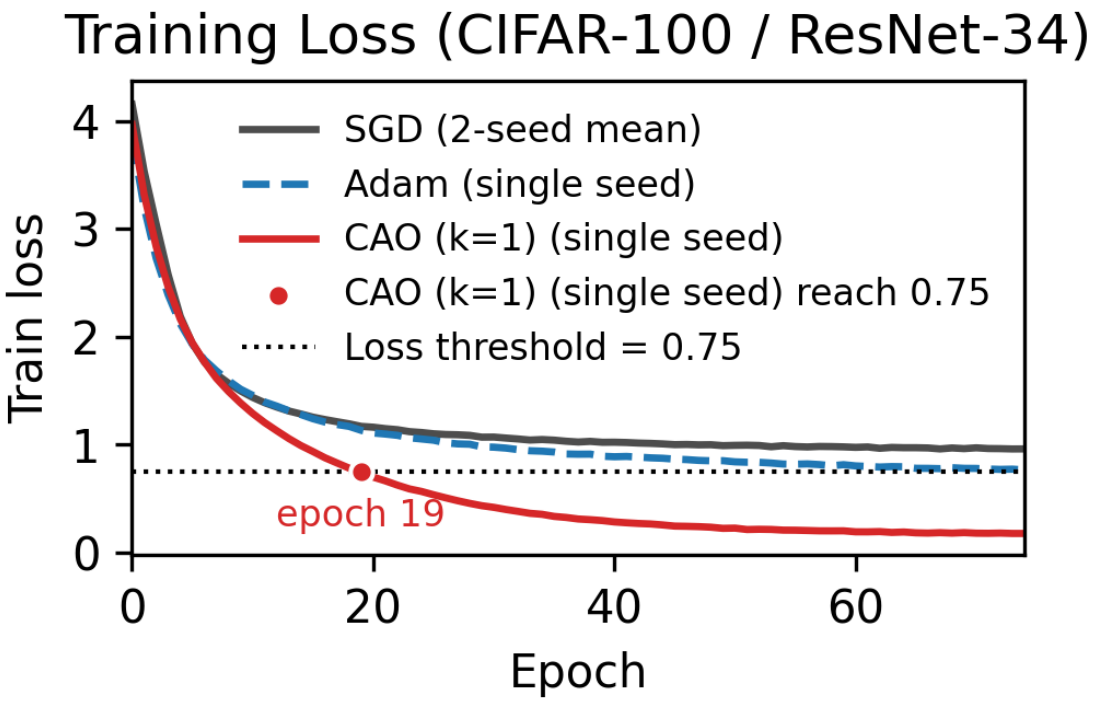

Figure 6: Appendix C: Train loss on CIFAR-100 / ResNet-34 (single seed).

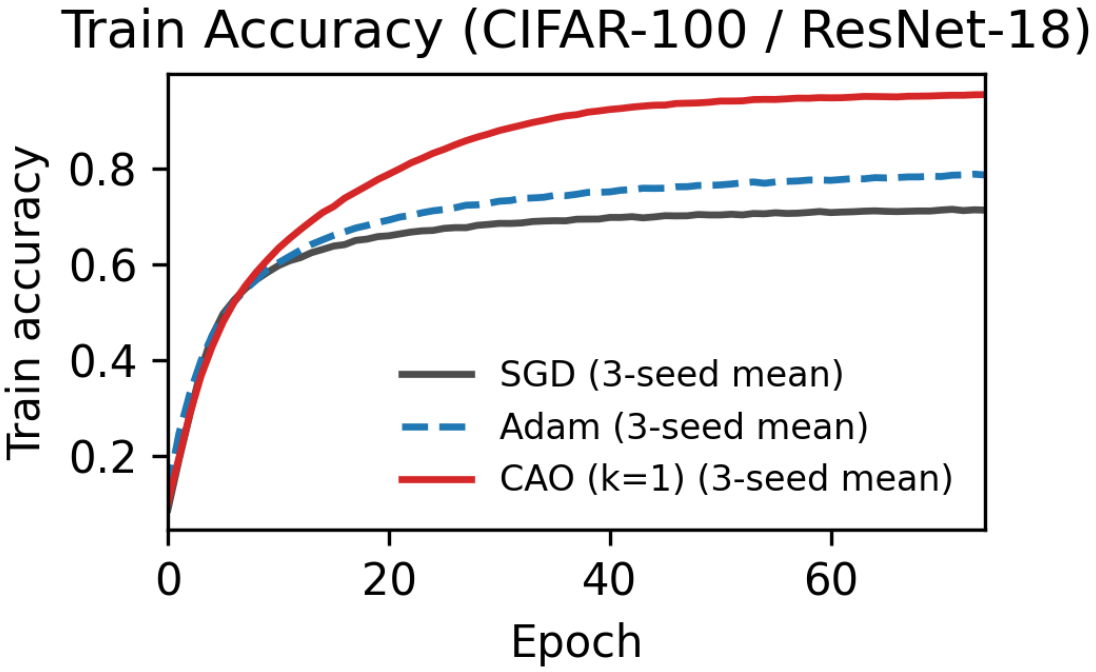

Figure 7: Appendix D: Train accuracy on CIFAR-100 / ResNet-18 (3 seeds).

Table 3: Per-epoch wall-clock and peak memory; † single-seed.

Setting	SGD	Adam	CAO (k=1)
C10-R18	7.46 / 936	7.53 / 1003	8.78 / 2508
C10-R34	10.56 / 1593	10.76 / 1688	12.83 / 4405
C100-R18	7.40 / 1130	7.51 / 1143	8.77 / 2700
C100-R34	$10.73^{\dagger}$ / $1628^{\dagger}$	$10.89^{\dagger}$ / $1537^{\dagger}$	$13.01^{\dagger}$ / $4354^{\dagger}$