NeuRMS: Unitwise RMS Preconditioning via Buffer Scaling for Fast and Accurate CPU Training

Anonymous for review

August 9, 2025

Abstract

We study a CPU-centric variant of adaptive preconditioning that avoids per-parameter state and preserves the standard two-GEMM training pipeline for fully-connected layers. The method, *NeuRMS*, maintains per-output-unit exponential moving averages (EMAs) of squared error signals and scales the corresponding columns of the backpropagation buffer prior to the weight-update GEMM. This realizes a left-diagonal (row-block) preconditioner with $O(d_{\text{out}})$ state and negligible overhead for large matrices. A factored extension, *NeuRMS-F*, additionally keeps per-input-feature activation EMAs and applies a temporary right-diagonal scaling to the activation buffer used in the update, yielding an effective $S^{(o)}GC$ transform without reconstructing per-parameter moments. We formalize the preconditioners, specify their placement relative to backprop, provide a convex-proxy convergence guarantee under bounded preconditioners, and give a dimension-explicit overhead model. On MLPs and small convs, accuracy–time Pareto curves against SGD/momentum, RMSProp, Adam, Adafactor, and LARS/LAMB demonstrate NeuRMS's competitive accuracy at lower memory and comparable or lower wall-clock, with microbench overheads decaying as $O(1/\min\{d_{\text{in}}, d_{\text{out}}\})$.

1 Introduction

Per-parameter adaptive optimizers (AdaGrad, RMSProp, Adam) improve robustness to curvature and scale but incur $O(\#\text{params})$ auxiliary memory and nontrivial CPU overhead due to scatter/gather and elementwise updates. Factorized approaches (e.g., Adafactor) reduce memory by using row/column moments, while second-order/Kronecker methods (K-FAC, Shampoo) exploit block structure at higher compute/memory cost.

We take a *systems-first* perspective: apply adaptivity to data already resident in CPU caches—the backprop error buffer $\Delta^{(o)}$ and the activation buffer H—so the fast GEMMs remain unchanged. NeuRMS scales *columns of* $\Delta^{(o)}$ to implement a unitwise left preconditioner; NeuRMS-F additionally scales *columns of* H for the update computation to realize a diagonal Kronecker factorization. Crucially, we do not form or store per-parameter moments nor change GEMM counts.

Contributions. (i) A precise formulation of *buffer-resident* unitwise (row-block) and factored (row/column) diagonal preconditioners that leave GEMMs intact. (ii) Two placement variants relative to backprop (update-only vs. backprop-propagating), with ablations. (iii) A convergence statement for diagonal preconditioning on convex proxies with explicit bounded-preconditioner conditions. (iv) A systems analysis showing overhead scales as $O(B(d_{\text{in}}+d_{\text{out}}))$ on top of $O(B\, d_{\text{in}}\, d_{\text{out}})$ GEMMs, yielding diminishing fractional cost with width. (v) A CPU implementation strategy compatible with BLAS or a custom backend, including cache-aware reductions and vectorizable scalings.

2 Preliminaries and Notation

Consider a fully-connected layer with parameters $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and per-output-unit bias $b \in \mathbb{R}^{d_{\text{out}}}$. For a batch of size B, let $H \in \mathbb{R}^{B \times d_{\text{in}}}$ denote activations, $\Delta^{(o)} = \partial L / \partial Z \in \mathbb{R}^{B \times d_{\text{out}}}$ the output-layer error signals (post loss), and $G = \frac{1}{B}(\Delta^{(o)})^\top H$ the gradient w.r.t. W. Standard SGD updates are

$$W \leftarrow W - \eta\, G, \qquad b \leftarrow b - \eta\, \frac{1}{B}(\Delta^{(o)})^\top \mathbf{1}. \tag{1}$$

We assume a forward/backward pipeline that computes H, $\Delta^{(o)}$, optionally $\Delta^{(h)} = \Delta^{(o)} W \odot \phi'(U)$, and then performs two GEMMs for updates.

3 NeuRMS: Buffer-Resident Preconditioning

3.1 Unitwise (row-block) scaling

Maintain EMA second moments over *columns* of $\Delta^{(o)}$:

$$v_i^{(o)} \leftarrow \beta v_i^{(o)} + (1-\beta) \cdot \frac{1}{B} \sum_{b=1}^{B} \left(\Delta_{b,i}^{(o)}\right)^2, \quad s_i^{(o)} = \left(\max(v_i^{(o)}, v_{\min})\right)^{-1/2}, \quad s_i^{(o)} \in [s_{\min}, s_{\max}], \tag{2}$$

and form $S^{(o)} = \operatorname{diag}(s_1^{(o)}, \ldots, s_{d_{\text{out}}}^{(o)})$. Scale delta columns in-place:

$$\Delta^{(o)} \leftarrow \Delta^{(o)} S^{(o)}. \tag{3}$$

Proposition 1 (Left preconditioning)**.** *With the above scaling,* $G' = \frac{1}{B}(\Delta^{(o)})^\top H = S^{(o)} G$. *Equivalently, on* $\operatorname{vec}(W)$ *the preconditioner is* $P_{row} = I_{d_{in}} \otimes S^{(o)}$.

Two placement variants are useful:

- **NeuRMS-U (update-only):** scale $\Delta^{(o)}$ only for the update computations in (1). Backpropagation to earlier layers uses the *unscaled* $\Delta^{(o)}$.
- **NeuRMS-B (backprop-propagating):** scale $\Delta^{(o)}$ before both the update and the upstream multiplication $\Delta^{(h)} = \Delta^{(o)} W \odot \phi'(U)$.

3.2 Factored (row/column) scaling

Additionally maintain EMA second moments over *columns* of H:

$$\hat{v}_j^{(H)} \leftarrow \beta \hat{v}_j^{(H)} + (1-\beta) \cdot \frac{1}{B} \sum_{b=1}^{B} H_{b,j}^2, \quad c_j = \left(\max(\hat{v}_j^{(H)}, v_{\min})\right)^{-1/2}, \quad c_j \in [c_{\min}, c_{\max}], \tag{4}$$

and $C = \operatorname{diag}(c_1, \ldots, c_{d_{\text{in}}})$. For the *update computation only*, temporarily scale activation columns:

$$H \leftarrow HC. \tag{5}$$

Proposition 2 (Diagonal Kronecker factorization)**.** *With both scalings,* $G' = \frac{1}{B}(\Delta^{(o)} S^{(o)})^\top (HC) = S^{(o)}\, G\, C$, *so the preconditioner on* $\operatorname{vec}(W)$ *is* $P_{fact} = C^\top \otimes S^{(o)}$.

Algorithm 1 NeuRMS (per layer, per batch)

Require: $H \in \mathbb{R}^{B\times d_{\text{in}}}$, $\Delta^{(o)} \in \mathbb{R}^{B\times d_{\text{out}}}$, states $v^{(o)}$, optional $\hat{v}^{(H)}$, lr η, hyperparams $\beta, \epsilon, v_{\min}, s_{\max}, c_{\max}$

$v_i^{(o)} \leftarrow \beta v_i^{(o)} + (1-\beta)\cdot \frac{1}{B}\sum_b (\Delta_{b,i}^{(o)})^2$ for $i = 1..d_{\text{out}}$

$s_i^{(o)} \leftarrow (\max(v_i^{(o)} + \epsilon, v_{\min}))^{-1/2}$; clip to $[s_{\min}, s_{\max}]$

if (NeuRMS-B) **then** $\Delta_{:,i}^{(o)} \leftarrow s_i^{(o)} \cdot \Delta_{:,i}^{(o)}$ **end if**

if (NeuRMS-F) **then**

  $\hat{v}_j^{(H)} \leftarrow \beta \hat{v}_j^{(H)} + (1-\beta)\cdot \frac{1}{B}\sum_b H_{b,j}^2$ for $j = 1..d_{\text{in}}$

  $c_j \leftarrow (\max(\hat{v}_j^{(H)} + \epsilon, v_{\min}))^{-1/2}$; clip to $[c_{\min}, c_{\max}]$

  $H_{:,j} \leftarrow c_j \cdot H_{:,j}$ % temporary for update GEMM only

$W \leftarrow W - \eta \cdot \frac{1}{B}(\Delta^{(o)})^\top H$ % unchanged GEMM

$b \leftarrow b - \eta \cdot \frac{1}{B}(\Delta^{(o)})^\top \mathbf{1}$ % per-unit biases

Assumption (faithfulness). NeuRMS-F best approximates row/column moment scaling when $\mathbb{E}\big[G_{ij}^2\big] \approx \mathbb{E}\big[(\Delta_i^{(o)})^2\big] \cdot \mathbb{E}\big[H_j^2\big]$. We will measure the correlation between G_{ij}^2 and $\Delta_i^{(o)2} H_j^2$ in practice.

3.3 Algorithmic reference

4 Convergence (Convex Proxy)

Let $f(W) = \mathbb{E}[\ell(W;\xi)]$ be smooth and convex, gradients $g_t = \nabla \ell(W_t;\xi_t)$ unbiased with bounded second moments. Suppose the NeuRMS (or NeuRMS-F) step can be written as $W_{t+1} = W_t - \eta_t P_t g_t$, where $P_t = \text{diag}(p_{t,k})$ satisfies $0 < p_{\min} \le p_{t,k} \le p_{\max} < \infty$ enforced by floors/clipping. With stepsizes $\sum_t \eta_t = \infty$, $\sum_t \eta_t^2 < \infty$, standard diagonal-preconditioned SGD analysis yields:

Theorem 1. *Under the above conditions,* $\min_{0\le t<T} \mathbb{E}\|\nabla f(W_t)\|^2 \to 0$ *as* $T \to \infty$, *and* $f(W_t)$ *converges to the minimum of* f.

Sketch. Lipschitz smoothness with bounded P_t gives a descent inequality with $\mathbb{E}[\langle \nabla f(W_t), P_t \nabla f(W_t)\rangle] \ge p_{\min}\mathbb{E}\|\nabla f(W_t)\|^2$. Summing and using Robbins–Monro conditions gives the claim. □

Remark. We do not claim stronger nonconvex guarantees; our use of EMAs and clipping is consistent with AdaGrad-Norm style arguments but we keep the statement conservative.

5 Complexity and Systems Analysis

Let $(B, d_{\text{in}}, d_{\text{out}})$ be the batch and layer dimensions. Per layer, SGD performs two GEMMs with cost $\Theta(B\, d_{\text{in}}\, d_{\text{out}})$. NeuRMS adds:

- Two column-wise reductions: $\Theta(B(d_{\text{out}} + d_{\text{in}}))$ if factored, else $\Theta(B d_{\text{out}})$.
- In-place column scalings with the same linear cost.

Thus the fractional overhead satisfies

$$\text{overhead fraction} \approx \tilde{O}\left(\frac{d_{\text{in}} + d_{\text{out}}}{d_{\text{in}} d_{\text{out}}}\right) = \tilde{O}\left(\frac{1}{\min\{d_{\text{in}}, d_{\text{out}}\}}\right),$$

so it decays with width. Practically, reductions and scalings are cache-friendly and vectorizable; avoid per-column BLAS calls (loop-fuse instead). Use OpenMP reductions across columns and FP64 accumulators for numerical stability if needed.

6 Implementation Notes (CPU)

GEMM pipeline. Keep forward/backward/update GEMMs unchanged. Implement reductions and scalings as fused row-major loops.
Biases. Use per-output-unit biases $b \in \mathbb{R}^{d_{\text{out}}}$ in both forward epilogue and gradient accumulation: $b \leftarrow b - \eta \frac{1}{B}(\Delta^{(o)})^\top \mathbf{1}$.
Placement choice. If adopting NeuRMS-B, scale $\Delta^{(o)}$ *before* computing $\Delta^{(h)}$; for NeuRMS-U, scale only within the update window and leave backprop deltas unchanged.
Momentum. If using momentum, scale the *gradient* before velocity updates; optionally scale the velocity by the same factor to reduce mismatch (ablate).
Numerics. Include ϵ in denominators and clip $s_i^{(o)}, c_j$ to avoid extreme steps for dead/rare units. An AGC-style guard on row updates $\Delta W_{i,:}$ is a robust safety valve.

7 Relation to Prior Work

Adafactor computes row/column moments of parameter gradients to reduce state; NeuRMS-F achieves a *similar form* by using moments of $\Delta^{(o)}$ and H buffers that already exist in backprop, avoiding reconstruction of per-parameter moments. K-FAC's Kronecker factors involve (co)variances of activations and backprop signals; NeuRMS-F can be viewed as the *diagonal* of those factors, implemented via buffer scaling. Shampoo performs higher-order tensor preconditioning with greater cost. LARS/LAMB perform layerwise trust-ratio scaling; NeuRMS is finer-grained (per unit/feature) yet remains cheap.

8 Experimental Protocol

Tasks. MNIST, Fashion-MNIST (MLPs), CIFAR-10 (small conv + MLP head).

Architectures. 1–2 hidden-layer MLPs (ReLU, 512/1024), small ResNet-20 for CIFAR-10.

Optimizers. SGD, SGD+momentum (0.9), RMSProp, Adam, Adafactor (matrix form), LARS, LAMB, and Shampoo on small models.

Schedules. Cosine decay with 200–400 warmup iterations; LR grid $\{3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}, 3\times10^{-2}\}$.

NeuRMS hyperparams. $\beta \in \{0.99, 0.995, 0.999\}$, $v_{\min} = 10^{-12}$, $\epsilon = 10^{-8}$, $s_{\max} \in \{10, 30, 100\}$; factored on/off; momentum coupling on/off.

Metrics. Accuracy at fixed wall-clock (1s, 5s), time/epoch, inference latency (unchanged by design), best-val accuracy, and state memory.

Statistical rigor. 5 seeds (mean±std); paired t-tests at fixed budgets.

8.1 Ablations (required)

1. **Placement:** NeuRMS-U vs. NeuRMS-B vs. NeuRMS-F.
2. **Independence check:** report correlation $\rho = \mathrm{corr}(G_{ij}^2, \Delta_i^{(o)2} H_j^2)$ across layers/epochs.
3. **Momentum coupling:** scale-grad only vs. scale-grad+velocity.
4. **Sensitivity:** grids over $(\beta, v_{\min}, s_{\max}, c_{\max})$; plot stability bands.
5. **Overhead scaling law:** sweep $(B, d_{\text{in}}, d_{\text{out}})$ on single/multi-thread, BLAS vs. custom; break epoch time into forward/backward/GEMM/NeuRMS-reduce/NeuRMS-scale.

9 Results (to be filled)

We will provide (i) Pareto frontiers of accuracy vs. time, (ii) overhead fraction vs. $\min\{d_{\text{in}}, d_{\text{out}}\}$, (iii) memory vs. optimizer, (iv) correlation diagnostics for NeuRMS-F faithfulness.

10 Limitations and Extensions

NeuRMS is isotropic within each unit's outgoing block; within-block anisotropy is unaddressed. NeuRMS-F partially compensates via a diagonal Kronecker factor; richer low-rank multi-factor variants ($k > 1$) remain future work. Extension to conv/attention is straightforward by mapping units to output channels/heads; we include smoke tests but defer large-scale models.

11 Reproducibility

We will release a minimal C99 implementation with Python bindings, unit tests checking analytical vs. finite-difference gradients, and scripts to reproduce plots. CPU microbenches include cache miss rates (L1/L2/LLC), vectorization ratios, and bandwidth.

Checklist for Reviewers (Response-Ready)

- **What is new?** Buffer-resident unitwise/factored diagonal preconditioning implemented via $\Delta^{(o)}/H$ scalings with no extra GEMMs and $O(d_{\text{out}}+d_{\text{in}})$ state.
- **Is it Adafactor?** Same row/column *form* at update-time if $\mathbb{E}[G_{ij}^2] \approx \mathbb{E}[\Delta_i^{(o)2}]\mathbb{E}[H_j^2]$, but computed from resident buffers without reconstructing per-parameter moments.
- **Does it converge?** Yes on convex proxies under bounded preconditioners; we do not overclaim for nonconvex.
- **Is the overhead really negligible?** It scales as $O(1/\min\{d_{\text{in}}, d_{\text{out}}\})$; we plot overhead across widths and batch sizes.

Acknowledgments

We thank colleagues and students for insights on cache-aware reductions, BLAS backends, and optimizer diagnostics.

References

[1] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[2] T. Tieleman and G. Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude. Coursera lecture notes, 2012.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[4] N. Shazeer and M. Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*, 2018.

[5] J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *ICML*, 2015.

[6] T. Gupta, A. Kumar, and M. R. et al. Shampoo: Preconditioned stochastic tensor optimization. arXiv:1802.09568, 2018.

[7] Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks with layer-wise adaptive rate scaling. *ICLR Workshop*, 2017.

[8] Y. You, Z. Li, S. Reddi, J. Hseu, et al. Large-batch optimization for deep learning: Training BERT in 76 minutes. In *ICLR*, 2019.

[9] R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *ICML*, 2019.

[10] Y. Ollivier. Riemannian metrics for neural networks I: feedforward networks. *Information and Inference*, 4(2):108–153, 2015.

Author Notes (Not for Review)

- Ensure per-output biases are vectors in code paths (no scalar shortcut).
- For NeuRMS-F, scale H only for the update GEMM; do not alter forward activations used for the next layer.
- Report $\rho = \mathrm{corr}(G_{ij}^2, \Delta_i^{(o)2} H_j^2)$; if low, discuss limits.