COREM: COSINE-RELATION MOMENTUM RESHAPING WITH STATEFUL WRITEBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Matrix-valued optimizer states may contain relational structure that is not captured by treating their entries independently. We study whether relations within matrix-valued optimizer states can be exploited to improve optimization. To this end, we introduce a unit–relation–transform abstraction and instantiate it as COREM, a Cosine-Relation Momentum Reshaping method with stateful writeback. COREM partitions the momentum state into update units, computes cosine relations among them, and uses these relations to reshape the momentum before writing the transformed state back to the optimizer. This stateful mechanism allows the reshaped momentum to affect not only the current update but also future optimization dynamics. We evaluate COREM on CIFAR-10 with an MLP and on enwik8 with a Transformer. Compared with Muon, COREM shows lower early-stage step efficiency but stronger improvement in the mid-to-late stages of training, achieving better final validation performance on CIFAR-10 and comparable final performance on enwik8. Spectral diagnostics on enwik8 show that COREM consistently increases entropy effective rank and reduces the concentration of singular energy in dominant modes, while preserving an anisotropic spectrum. For square matrix updates, COREM requires approximately 13.3% of the transformation FLOPs of Muon with five Newton–Schulz iterations.

1 INTRODUCTION

Modern optimization methods increasingly exploit structure beyond coordinate-wise rescaling when transforming parameter updates. Adaptive methods such as AdaGrad (Duchi et al., 2011) and Adam (Kingma & Ba, 2015) use statistics of past gradients to modulate updates, while structured approaches such as K-FAC (Martens & Grosse, 2015), Shampoo (Gupta et al., 2018), and SOAP (Vyas et al., 2025) exploit richer matrix- or tensor-level structure through curvature approximation or preconditioning. These developments demonstrate that structure within an optimizer state can provide useful information for determining how updates are transformed.

A complementary source of structure lies in the relations among subcomponents of a structured optimizer state, which we refer to as update units. Explicit relation modeling has proved useful in relational reasoning (Santoro et al., 2017), self-attention (Vaswani et al., 2017; Devlin et al., 2019; Dosovitskiy et al., 2021; Carion et al., 2020), and non-local computation (Wang et al., 2018), where interactions among units condition subsequent aggregation or transformation. This motivates a central question: can relations among optimizer-state units similarly be used to determine how the state itself should be transformed?

To study this question, we introduce a **unit–relation–transform abstraction**,

$$\boxed{\text{units} \;\rightarrow\; \text{relations} \;\rightarrow\; \text{relation-induced transformation.}}$$

The abstraction separates three modeling choices: the update units, the relation function, and the transformation induced by the resulting relation structure.

We instantiate this abstraction for matrix-valued momentum states as **COREM: Cosine-Relation Momentum Reshaping with stateful writeback**. COREM constructs signed cosine relations among momentum units and uses the resulting relation operator to reshape the state. COREM does not impose a fixed target geometry such as exact orthogonality, while its transformation is

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

conditioned on the relational structure of the current momentum state. The transformed momentum is then written back into the optimizer state, allowing the reshaped state to influence subsequent momentum accumulation.

Muon (Jordan et al., 2024) provides a natural reference because it also transforms matrix-valued momentum, but through an orthogonalization-oriented mechanism. Across CIFAR-10 and enwik8, COREM achieves competitive final performance while exhibiting distinct optimization dynamics and substantially lower transformation cost. Spectral diagnostics further indicate that COREM redistributes singular-mode energy while preserving nontrivial spectral anisotropy.

Our contributions are threefold:

1. We introduce a **unit–relation–transform abstraction** for constructing optimizer-state transformations from relations among update units.
2. We instantiate this abstraction as **COREM**, which uses signed cosine relations and stateful writeback to reshape matrix-valued momentum without enforcing a fixed target geometry.
3. We evaluate **COREM** on CIFAR-10 and enwik8, where it achieves competitive final performance and exhibits distinct optimization dynamics relative to Muon. We further characterize its spectral reshaping behavior through optimizer-state diagnostics.

2 Related Work

2.1 Adaptive and Structured Optimizers

Adaptive methods such as AdaGrad (Duchi et al., 2011) and Adam (Kingma & Ba, 2015) use statistics of past gradients to modulate updates, typically through coordinate-wise adaptation. Adafactor (Shazeer & Stern, 2018) further reduces optimizer-state memory through factored second-moment estimates for matrix-valued parameters.

A complementary line of work exploits matrix- or tensor-level structure. Shampoo (Gupta et al., 2018) uses structured preconditioning across tensor dimensions, while SOAP (Vyas et al., 2025) combines Shampoo-style preconditioning with Adam-like adaptive updates in an evolving structured basis. These methods move beyond independent scalar coordinates by introducing richer moment or preconditioning structure.

COREM approaches structured optimization from a different direction. Rather than estimating a richer preconditioner, it constructs a transformation from relations among update units within the current optimizer state. Its modeling object is therefore the relational geometry among update units rather than a diagonal or matrix-valued estimate of update scale or curvature.

2.2 Matrix-Valued Momentum and Spectral Transformations

Another line of optimizer design directly reshapes matrix-valued updates. Matrix-function methods, including Newton–Schulz iterations, provide practical ways to modify matrix geometry without explicitly computing a full singular value decomposition (Higham, 2008).

Muon (Jordan et al., 2024) applies an orthogonalization-oriented Newton–Schulz transformation to matrix-valued momentum. COREM also reshapes matrix-valued momentum, but constructs its transformation from pairwise directional relations rather than from a prescribed orthogonalization objective. Although both can involve cubic terms of the form $XX^\top X$, this similarity is algebraic rather than algorithmic: Newton–Schulz iterations approximate a matrix function associated with orthogonalization, whereas COREM uses a dynamically normalized relation operator to reshape the optimizer state. Muon therefore provides a natural reference point for comparing two distinct mechanisms for transforming matrix-valued momentum.

2.3 Relation-Based Computation

Relation-conditioned computation is widely used in representation learning. Relation Networks (Santoro et al., 2017), self-attention (Vaswani et al., 2017; Devlin et al., 2019; Dosovitskiy et al., 2021; Carion et al., 2020), Graph Attention Networks (Veličković et al., 2018), and non-local

neural networks (Wang et al., 2018) all construct input-dependent interactions among units and use those interactions to control aggregation or transformation.

A broad class of such operations can be written schematically as

$$X' = R(X)X,$$

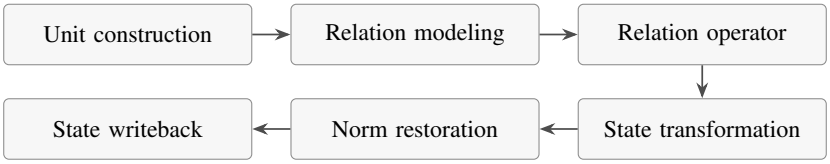
where $R(X)$ is a relation operator constructed from the current input. COREM shares this general relation-conditioned computation pattern, but applies it to optimizer-state units rather than feature representations. Its relation operator is built from signed cosine similarities and enters a residual momentum transformation rather than a softmax-style feature aggregation rule.

The connection to attention and graph-based computation is therefore structural rather than architectural. Taken together, these works motivate COREM as a relation-conditioned optimizer-state transformation that is distinct from both structured preconditioning and orthogonalization-oriented update transforms.

3 METHOD

3.1 OVERVIEW

COREM transforms a momentum state through dynamically constructed relations among update units. It follows the pipeline



Let G_t be the gradient and M_{t-1} the previously written-back momentum state. The momentum candidate, COREM transformation, and parameter update are

$$V_t = \mu M_{t-1} + G_t, \qquad M_t = \mathcal{T}_{\mathrm{COREM}}(V_t), \qquad \theta_t = \theta_{t-1} - \alpha\, M_t.$$

Crucially, M_t is retained: $V_{t+1} = \mu M_t + G_{t+1}$. Thus the state recurrence is $M_t = \mathcal{T}_{\mathrm{COREM}}(\mu M_{t-1} + G_t)$, coupling spatial relations among units with temporal momentum memory.

3.2 UNIT–RELATION–TRANSFORM ABSTRACTION

Given update units $\mathcal{U} = \{u_1, \ldots, u_n\}$, a generic unit–relation–transform construction represents each unit as $z_i = \phi(u_i)$, defines pairwise relations $R_{ij} = \mathcal{R}(z_i, z_j)$, and constructs a relation-induced operator $A = \mathcal{A}(R)$ that produces $\widetilde{\mathcal{U}} = \mathcal{T}_A(\mathcal{U})$. This separates the choices of update units, relation functions, and relation-induced transformations.

For a matrix-valued momentum candidate, COREM first chooses a canonical orientation in which the shorter dimension indexes update units. The matrix is transposed when necessary so that

$$V_t = [v_{t,1}^\top; \ldots; v_{t,n}^\top] \in \mathbb{R}^{n \times d}, \qquad n \le d.$$

For notational simplicity, V_t denotes this oriented representation throughout the transformation; the original orientation is restored before the transformed state is written back.

COREM represents each update unit by its normalized direction:

$$r_{t,i} = \max(\|v_{t,i}\|_2, \epsilon), \qquad \widehat{v}_{t,i} = \frac{v_{t,i}}{r_{t,i}}, \qquad \widehat{V}_t = \mathrm{RowNormalize}(V_t).$$

where $\epsilon > 0$ is a small numerical constant used to avoid division by zero.

3.3 COSINE-RELATION CONSTRUCTION

COREM constructs signed cosine relations among the normalized update units and removes self-relations:

$$C_t = \mathrm{OffDiag}(\widehat{V}_t\widehat{V}_t^\top) = \widehat{V}_t\widehat{V}_t^\top - I, \qquad \rho_t = \|C_t\|_\infty = \max_i \sum_j |(C_t)_{ij}|, \qquad \overline{C}_t = \frac{C_t}{\rho_t + \epsilon}.$$

Thus, C_t retains both positive and negative cross-unit directional relations, while ρ_t normalizes their maximum absolute row-sum scale.

Because C_t is symmetric, $\|C_t\|_2 \le \sqrt{\|C_t\|_1\|C_t\|_\infty} = \rho_t$. Hence, $\|\overline{C}_t\|_2 \le \rho_t/(\rho_t + \epsilon) < 1$ for $\rho_t > 0$ (and $\overline{C}_t = 0$ when $\rho_t = 0$), so the eigenvalues of $\overline{C}_t$ lie in $(-1, 1)$.

3.4 RELATION-NORMALIZED MOMENTUM RESHAPING AND SPECTRAL CONTROL

The normalized momentum state is reshaped by the relation operator:

$$\widetilde{V}_t = \widehat{V}_t - \eta\overline{C}_t\widehat{V}_t = (I - \eta\overline{C}_t)\widehat{V}_t.$$

Since the eigenvalues $\lambda_k(\overline{C}_t)$ of the normalized relation operator lie in $[-1, 1]$, the corresponding relation eigenmode is scaled by

$$g_{t,k} = 1 - \eta\lambda_k(\overline{C}_t) \in [1 - \eta,\, 1 + \eta].$$

Thus, relation normalization bounds the mode-wise effect of the transformation; in particular, when $0 \le \eta < 1$, all mode gains remain positive and no relation eigenmode changes sign.

After relation filtering, COREM restores the Frobenius norm $\|V_t\|_F$ and returns the transformed momentum to its original orientation, yielding M_t.

3.5 CONNECTIONS AND COMPUTATIONAL COST

The relation term has the generic form $R(X)X$, resembling the relation-conditioned aggregation pattern used in attention-like operations. COREM instantiates this pattern differently: its relation matrix is signed and non-softmax, and the relation term enters through the residual transformation $X' = X - \eta R(X)X$. Ignoring the numerical safeguard, relation normalization, and norm restoration, $C = \widehat{V}\widehat{V}^\top - I$ gives

$$\widetilde{V} = (1 + \eta)\widehat{V} - \eta\widehat{V}\widehat{V}^\top\widehat{V}.$$

At $\eta = \frac{1}{2}$, this has the cubic form of a Newton–Schulz polar iteration. This connection is algebraic only: the full COREM method normalizes the relation operator, restores the Frobenius norm, and writes the transformed state back into the momentum dynamics.

For $V \in \mathbb{R}^{n\times d}$, the two dominant matrix products $\widehat{V}\widehat{V}^\top$ and $\overline{C}\widehat{V}$ cost approximately $4n^2d$ FLOPs. For a square state, this is $4n^3$ FLOPs. A quintic Newton–Schulz iteration uses three dominant products, or approximately $6n^3$ FLOPs; five iterations therefore cost approximately $30n^3$. Hence, for square states, one COREM transform requires $4/30 \approx 13.3\%$ of the transformation FLOPs of a five-iteration Muon Newton–Schulz transform (Jordan et al., 2024).

4 EXPERIMENTS

We evaluate COREM on two complementary benchmarks, CIFAR-10 image classification (Krizhevsky, 2009) and enwik8 character-level language modeling (Hutter, 2006), and compare it with Muon (Jordan et al., 2024). Non-matrix parameters are optimized using SGD with momentum. Detailed experimental settings and optimizer hyperparameters are provided in Appendix A.

4.1 EXPERIMENTAL SETUP

For CIFAR-10, we train a two-hidden-layer MLP with hidden dimensions 256×256 and compare COREM against Muon. All CIFAR-10 experiments are run on a MacBook using the Apple MPS

Table 1: CIFAR-10 validation results over three random seeds. Values denote mean $\pm$ sample standard deviation.

Method	Final Val. Acc.	Best Val. Acc.	Final Val. Loss	Best Val. Loss
Muon	$59.74 \pm 1.01\%$	$60.67 \pm 0.44\%$	1.2157 ± 0.0322	1.1713 ± 0.0131
COREM	$61.36 \pm 0.19\%$	$61.87 \pm 0.23\%$	1.1450 ± 0.0084	1.1282 ± 0.0040

Table 2: enwik8 validation results after $64{,}000$ steps over three random seeds. Values denote mean $\pm$ sample standard deviation.

Method	Final Val. Loss	Final Val. BPB	Final Val. Accuracy
Muon	1.095263 ± 0.004475	1.580130 ± 0.006457	0.678172 ± 0.001344
COREM	1.096513 ± 0.002470	1.581933 ± 0.003563	0.678525 ± 0.001018

backend. For enwik8, we train a decoder-only Transformer (Vaswani et al., 2017) with hidden dimension 192, four layers, six attention heads, and a context length of 256 for $64{,}000$ optimization steps. COREM and Muon are evaluated under the same model and data configuration. All enwik8 experiments are run on an NVIDIA GeForce RTX 3080 Ti.

4.2 Main Results

According to Table 1, COREM achieves higher final and best validation accuracy on CIFAR-10, together with lower final and best validation loss and lower cross-seed variation. Table 2, in contrast, shows comparable final performance on enwik8: Muon obtains slightly lower validation loss and BPB, whereas COREM attains slightly higher validation accuracy with lower cross-seed variability.

4.3 Optimization Dynamics

The final metrics in Section 4.2 do not fully reflect the different optimization trajectories of COREM and Muon. As shown in Figure 1, Muon progresses faster early in training, whereas COREM exhibits relatively stronger late-stage improvement.

4.3.1 CIFAR-10

On CIFAR-10, Muon reaches 50% and 55% validation accuracy after approximately 5.3 and 16.3 epochs on average, compared with 8.0 and 20.7 epochs for COREM. In contrast, over the final 25 epochs, COREM averages 61.20% validation accuracy, compared with 60.00% for Muon. Thus, Muon reaches moderate accuracy levels earlier, while COREM attains stronger late-stage validation performance.

4.3.2 enwik8

A similar pattern appears on enwik8. Muon reaches validation loss ≤ 1.2 after approximately 7.2k steps, compared with 9.7k for COREM, and reaches BPB ≤ 1.6 after 40.7k versus 48.4k steps. Between 50k and 64k steps, however, COREM reduces mean validation BPB from 1.600102 to 1.581933, compared with 1.592999 to 1.580130 for Muon, substantially narrowing the remaining gap by the end of training.

4.4 Ablation Studies

4.4.1 Stateful Momentum Writeback

We evaluate the role of stateful momentum writeback on CIFAR-10 by comparing standard COREM with a no-writeback variant, in which the transformed momentum is used for the current parameter update but the untransformed momentum is retained for subsequent accumulation. Both variants are trained for 200 epochs over three random seeds using independently tuned operating points.

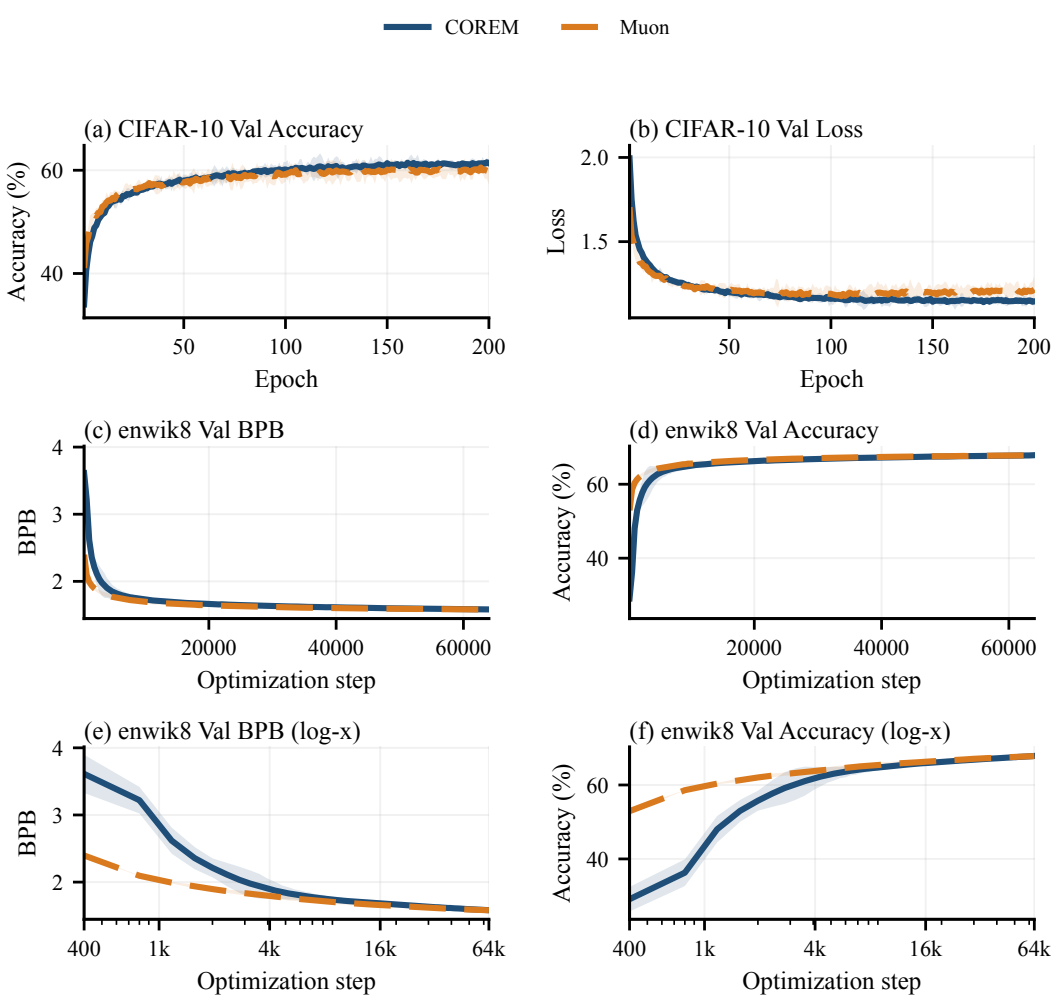


Figure 1: Validation trajectories for COREM and Muon on CIFAR-10 and enwik8. The first two rows use linear optimization time, while the third row shows enwik8 trajectories with logarithmic optimization steps. Shaded regions denote the three-seed uncertainty bands.

As shown in Table 3, stateful writeback improves both final and best validation performance. The corresponding trajectories in Figure 2 show that the writeback variant also reaches the higher-performance regime earlier and maintains a stronger late-stage validation level. Because the two variants use independently tuned configurations ($\alpha = 0.01, \eta = 1.2$ with writeback and $\alpha = 0.0125, \eta = 1.0$ without writeback), the comparison should be interpreted as one between their tuned operating points rather than as a strict single-hyperparameter causal isolation. Nevertheless, the no-writeback variant does not recover the performance of stateful COREM after retuning, supporting writeback as an important component of the method.

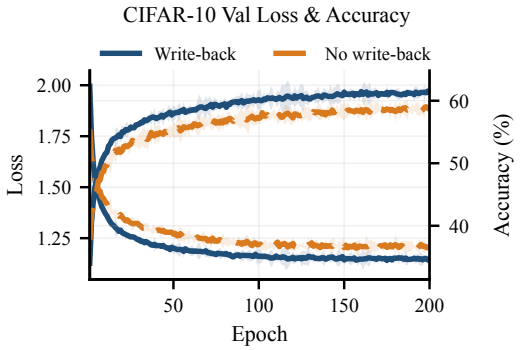


Figure 2: CIFAR-10 validation trajectories for the tuned writeback and no-writeback variants. Shaded regions denote the three-seed uncertainty bands.

Table 3: Effect of stateful momentum writeback on CIFAR-10 over three random seeds. Values denote mean $\pm$ sample standard deviation.

Variant	Final Acc.	Best Acc.	Final Loss	Best Loss
No writeback	$58.61 \pm 0.44\%$	$59.47 \pm 0.13\%$	1.2130 ± 0.0027	1.1859 ± 0.0035
Writeback	$61.36 \pm 0.19\%$	$61.87 \pm 0.23\%$	1.1450 ± 0.0084	1.1282 ± 0.0040

4.4.2 Relation Normalization

We examine the role of relation-operator normalization by removing the max-row-ℓ_1 scaling and performing a 4k-step enwik8 sweep over

$$\alpha \in \{0.02, 0.05, 0.08, 0.10\}, \qquad \eta \in \{0.05, 0.10, 0.20, 0.30, 0.50, 0.60, 0.80\}.$$

As shown in Figure 3, only 7 of the 28 tested configurations remain numerically stable. In particular, all configurations with $\alpha \geq 0.08$ or $\eta \geq 0.50$ become unstable.

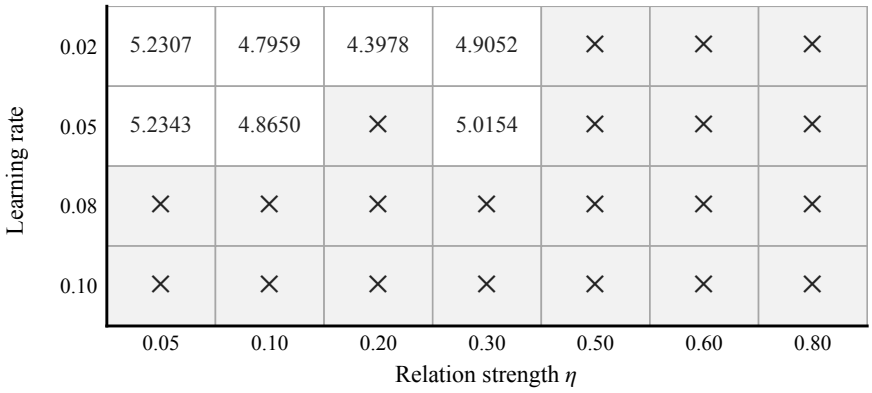


Figure 3: Stability map of the unnormalized COREM variant on enwik8 over a 4k-step learning-rate/relation-strength sweep. Each finite configuration is annotated with its final validation BPB, while $\times$ denotes a numerically unstable or diverged run.

Among the finite runs, the best final validation BPB is 4.3978, compared with 1.8556 for the normalized reference ($\alpha = 0.10, \eta = 0.50$). Because the two variants are evaluated over different hyperparameter grids, this comparison is used primarily to characterize their stable operating regions rather than as a pointwise matched ablation. The result is consistent with the spectral-control motivation for relation-operator normalization developed in Section 3.4.

4.5 Analysis of Relation and Spectral Dynamics

We next examine how COREM reshapes the geometry of the optimizer state during training. The analysis uses diagnostic snapshots from the 64k-step enwik8 run at nine checkpoints, 400, 800, 2k, 4k, 8k, 16k, 32k, 49.6k, and 64k, covering four Transformer layers and six matrix-valued parameter groups. For each layer-step pair, we record the instantaneous gradient G_t, the pre-transformation momentum $V_t = \mu M_{t-1} + G_t$, and the transformed momentum $M_t = \mathcal{T}_{\mathrm{COREM}}(V_t)$. Because writeback is enabled, V_t already contains the previously transformed state M_{t-1}; it should therefore be interpreted as an inherited stateful momentum geometry rather than a COREM-free baseline.

4.5.1 Relation-Scale Heterogeneity

We examine the raw relation scale $\rho_t = \|C_t\|_\infty$ throughout training. Across the 216 observations, ρ_t varies substantially across parameter groups, Transformer layers, and optimization steps, ranging from approximately 15.42 to 72.02. As shown in Figure 4, W_v generally exhibits larger relation

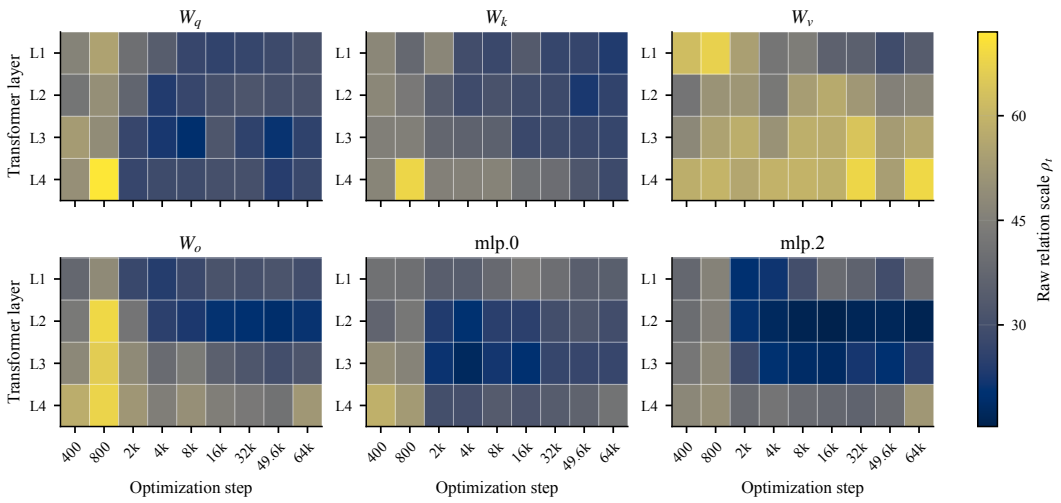


Figure 4: Raw relation-scale dynamics across parameter groups, Transformer layers, and sampled optimization steps on enwik8. Each panel corresponds to one matrix-valued parameter group; rows denote Transformer layers, columns denote sampled optimization steps, and color indicates $\rho_t = \|C_t\|_\infty$. W_q, W_k, W_v, and W_o denote the attention projection matrices, while `mlp.0` and `mlp.2` denote the first and second linear projections in the Transformer MLP block. All panels share the same color scale.

scales, while the MLP projections tend to operate at lower levels. This variation is consistent with the reduced stability observed in the unnormalized ablation: without max-row-ℓ_1 scaling, a fixed η acts on relation operators with substantially different raw scales.

4.5.2 Stateful Spectral Maintenance

We characterize spectral structure using entropy effective rank, top-10 singular-energy concentration, and a robust 10–90 spectral condition ratio:

$$r_{\text{eff}}(X) = \exp\left(-\sum_i p_i \log p_i\right), \quad p_i = \frac{\sigma_i^2(X)}{\sum_j \sigma_j^2(X)},$$

$$C_{10}(X) = \frac{\sum_{i=1}^{10} \sigma_i^2(X)}{\sum_j \sigma_j^2(X)},$$

$$\kappa_{\text{robust}}(X) = \frac{\sigma_{\lceil 0.1r \rceil}(X)}{\sigma_{\lceil 0.9r \rceil}(X) + \epsilon}.$$

Here, r denotes the number of singular values of X.

Across all 216 paired observations, transformed momentum M_t exhibits higher effective rank and lower top-10 singular-energy concentration than the pre-transformation state V_t. In contrast, the robust condition ratio changes comparatively little from V_t to M_t, while both states exhibit substantially lower robust condition ratios than the instantaneous gradient G_t.

Because $V_t = \mu M_{t-1} + G_t$, its spectral geometry already contains previously transformed momentum propagated through stateful writeback and should not be interpreted as a COREM-free baseline. The observed pattern is therefore consistent with stateful propagation of the reshaped momentum, together with additional current-step spectral redistribution from V_t to M_t.

5 Conclusion

We introduced a unit–relation–transform abstraction for optimizer-state transformation and instantiated it as COREM, a Cosine-Relation Momentum Reshaping method with stateful writeback for

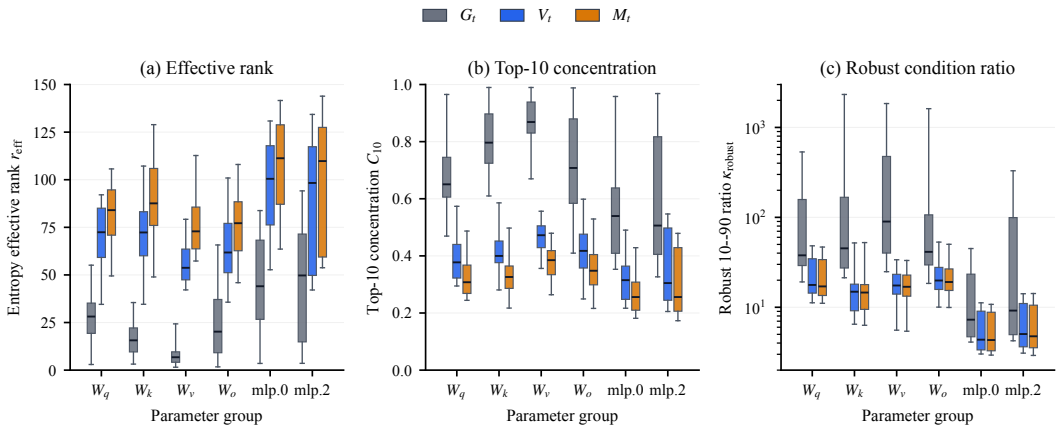


Figure 5: Spectral diagnostics across optimizer states and parameter groups on enwik8. Each panel compares G_t, V_t, and M_t over the 36 layer–checkpoint observations for each parameter group. Panels show entropy effective rank, top-10 singular-energy concentration, and the robust 10–90 spectral condition ratio. Boxes denote medians and interquartile ranges, with whiskers spanning the observed range; the condition-ratio panel uses a logarithmic y-axis.

matrix-valued momentum states. COREM derives its transformation from relations among update units rather than from a prescribed target geometry, and integrates the resulting reshaped state into subsequent momentum dynamics. Across the evaluated image and language modeling settings, COREM remains competitive with Muon while exhibiting distinct optimization dynamics. The ablation results support the roles of stateful writeback and relation-operator normalization in the proposed design. These findings provide an initial validation of relation-conditioned optimizer-state transformation and motivate future exploration of alternative update units, relation functions, and transformation rules beyond the current matrix-valued setting.

6 Generative AI Usage Disclosure

Generative AI tools were used at several stages of the research and writing process. The overall method and its core research idea were developed by the authors. During the development of one specific component of the method, generative AI suggested relation-operator normalization as a mechanism for bounding the mode-wise effect of the relation-induced transformation and assisted with the associated mathematical derivation. Generative AI also assisted with the implementation of the proposed method.

For empirical analysis, generative AI was used, under author-specified criteria and guidance, to analyze hyperparameter sweep results and assist in identifying an appropriate operating range. Generative AI assisted in developing plotting scripts for the figures in the paper. In addition, for Figure 3, it provided suggestions on the choice of visualization format.

During manuscript preparation, generative AI was used to draft parts of an initial version of the paper. It was also used for brainstorming and literature discovery during the development of the related-work discussion, including suggesting K-FAC and relational reasoning as potentially relevant prior work.

All AI-assisted outputs were reviewed by the authors before inclusion in the paper. The authors take responsibility for the final content of this work, including the mathematical claims, method implementation, experimental analysis, text, citations, and artifacts produced with the assistance of generative AI.

7 Reproducibility Statement

The proposed method is specified in Section 3, including the optimizer-state transformation and stateful writeback procedure. Detailed experimental settings, training configurations, optimizer hyperparameters, and random seeds for CIFAR-10 and enwik8 are provided in Appendix A. Code required to reproduce the reported experiments is submitted together with the paper as supplementary material.

References

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, pp. 213–229. Springer, 2020. doi: 10.1007/978-3-030-58452-8_13.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

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

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850, 2018.

Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.

Marcus Hutter. Human knowledge compression prize, 2006. URL `https://www.hutter1.net/prize/`.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417. PMLR, 2015.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4596–4604, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, 2025.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

A Experimental Details

A.1 CIFAR-10

A.1.1 Experimental Setup

All CIFAR-10 experiments use the same data pipeline, model architecture, and evaluation protocol across optimizer variants. The main experimental settings are summarized in Table 4.

Table 4: Experimental setup for CIFAR-10.

Setting	**Configuration**
Data partition	Standard CIFAR-10 training and test partitions. The test partition is used for evaluation throughout the experiments.
Preprocessing	`ToTensor` followed by channel-wise normalization with mean $(0.4914, 0.4822, 0.4465)$ and standard deviation $(0.2470, 0.2435, 0.2616)$.
Data augmentation	Training additionally uses `RandomCrop`(32, padding=4) and `RandomHorizontalFlip`.
Model	MLP with architecture $3072 - 256 - 256 - 10$, with ReLU activations after both hidden layers and no dropout.
Batch size	512
Training duration	200 epochs
Evaluation	Loss and accuracy are evaluated after every training epoch in evaluation mode without gradient computation.
Random seeds	63, 101, and 2025
Hardware	MacBook using the Apple MPS backend.

A.1.2 Training and Optimization Hyperparameters

Table 5 summarizes the configurations used for the 200-epoch CIFAR-10 comparison. COREM and Muon use separately tuned hyperparameter configurations for the final experiments.

Table 5: Training and optimization hyperparameters for the CIFAR-10 200-epoch experiments.

Setting	**COREM**	**Muon**
Learning rate	0.01	0.006
Momentum	0.9	0.9
Weight decay	0	0
Learning-rate schedule	Fixed	Fixed
Relation strength η	1.2	–
Relation normalization	max-row-ℓ_1	–
Relation-normalization stability constant	$\epsilon_{\text{rel-norm}} = 10^{-8}$	–
Newton–Schulz iterations	–	5
Newton–Schulz stability constant	–	$\epsilon_{\text{NS}} = 10^{-7}$
Orthogonal-update scaling	–	Enabled
Batch size	512	512
Training duration	200 epochs	200 epochs
Random seeds	63, 101, 2025	63, 101, 2025

A.2 ENWIK8

A.2.1 EXPERIMENTAL SETUP

All enwik8 experiments use the same byte-level causal language modeling setup, model architecture, data partition, and evaluation protocol across optimizer variants. The experimental configuration is summarized in Table 6.

Table 6: Experimental setup for enwik8.

Setting	**Configuration**
Data partition	The first $90{,}000{,}000$ bytes are used for training, followed by $5{,}000{,}000$ bytes for validation and $5{,}000{,}000$ bytes for testing. The three partitions are non-overlapping.
Tokenization	Byte-level encoding with a vocabulary size of 256.
Sequence construction	Each training example contains 256 input bytes with the corresponding next-byte prediction targets. No data augmentation is applied.
Model	Decoder-only Transformer with embedding dimension 192, four layers, six attention heads, MLP ratio 4, context length 256, no dropout, and tied input/output embeddings.
Batch size	64 sequences per optimization step
Training duration	$64{,}000$ optimization steps
Evaluation	Validation loss, bits per byte (BPB), and accuracy are evaluated every 400 optimization steps. Training metrics are logged every 50 steps.
Random seeds	63, 101, and 2025
Data-loader workers	4
Hardware	NVIDIA GeForce RTX 3080 Ti

A.2.2 TRAINING AND OPTIMIZATION HYPERPARAMETERS

Table 7 summarizes the configurations used for the 64k-step enwik8 comparison. COREM and Muon use the same model, data configuration, matrix-valued parameter scope, and training budget, while their optimizer-specific hyperparameters are tuned separately.

Table 7: Training and optimization hyperparameters for the enwik8 64k-step experiments.

Setting	**COREM**	**Muon**
Learning rate	0.10	0.02
Momentum	0.9	0.9
Weight decay	0	0
Learning-rate schedule	Fixed	Fixed
Warmup	None	None
Relation strength η	0.5	–
Relation normalization	max-row-ℓ_1	–
Relation-normalization stability constant	$\epsilon_{\text{rel-norm}} = 10^{-8}$	–
Matrix transform scope	Attention and MLP matrices	Attention and MLP matrices
Newton–Schulz iterations	–	5
Newton–Schulz stability constant	–	$\epsilon_{\text{NS}} = 10^{-7}$
Orthogonal-update scaling	–	Enabled
Batch size	64	64
Training duration	$64{,}000$ steps	$64{,}000$ steps
Random seeds	63, 101, 2025	63, 101, 2025