On the Surprising Effectiveness of Masking Updates in LLM Training

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2026

Abstract

Training large language models (LLMs) relies almost exclusively on dense adaptive optimizers with increasingly sophisticated preconditioners. We challenge this by showing that randomly masking parameter updates can be highly effective, with a masked variant consistently outperforming the corresponding dense optimizer. Our analysis shows that random masking induces a curvature-dependent regularization term that penalizes sharp update directions. Motivated by this finding, we introduce Momentum-aligned gradient masking (Magma), which modulates the masked updates using momentum-gradient alignment. Across extensive LLM pre-training experiments up to 1B parameters, Magma improves a broad range of optimizers—including Adam, Lion, Muon, and SOAP—in 14 out of 16 cases for dense architectures, and successfully improves both Adam and Muon in sparse mixture-of-experts (MoE) architectures. Our analysis shows that alignment-based masking can widen the stable optimization regime by damping noisy, high-curvature blocks while preserving descent direction.

1. Introduction

This work focuses on *autoregressive LLM pre-training*, where the availability of dense gradients from a single backward pass [40] enables efficient simultaneous parameter updates. This efficiency has made dense adaptive optimizers like Adam [20] the de facto standard in this setting. In contrast, the reliance on dense gradients creates a structural mismatch with sparse update strategies such as coordinate descent [30, 32, 55]. Consequently, despite their strong performance on highly nonsmooth optimization problems common in LLM training, sparse methods are rarely used in this setting.

We challenge the prevailing dense-update paradigm with a counterintuitive empirical finding: *randomly masking gradient updates can substantially improve optimization performance*. We first study SkipUpdate, a masked variant of RMSProp described in Algorithm 1 that samples Bernoulli masks over parameter blocks, skips the corresponding parameter updates, and rescales the surviving updates while still updating moment estimates densely. Classical convergence analysis suggests that such masking should degrade performance by adding stochastic noise (cf. Appendix A.2). Yet, despite discarding half of the updates, SkipUpdate consistently outperforms dense optimizers across model scales. At the 1B scale on C4 [38], SkipUpdate—a masked version of RMSProp—reaches a perplexity of 13.71, outperforming Adam by 18% and SOAP [48] by 5%, although SOAP is a more sophisticated matrix-preconditioned optimizer designed to exploit richer update geometry (Figure 1).

One might attribute these gains to a reduced effective update dimension in the peculiar optimization landscapes of LLM pre-training. Indeed, Adafactor [42] and GaLore [62] also simplify dense optimizer trajectories through factored moment statistics or low-rank gradient projections. However, neither reproduces the gains from stochastic update masking: at the 1B scale, they reach perplexities

© .

Algorithm 1 SkipUpdate and Magma. Both methods randomly mask block-wise optimizer updates. SkipUpdate uses a fixed scaling factor, whereas Magma uses a momentum-gradient alignment score.

Input: Parameters $\{\theta_t^{(b)}\}_{b=1}^B$, Stochastic gradients $\{\boldsymbol{g}_t^{(b)}\}_{b=1}^B$, Updates from a base optimizer $\{\Delta_t^{(b)}\}_{b=1}^B$, First-moment estimates $\{\mu_t^{(b)}\}_{b=1}^B$
for Each parameter block $b \in [B]$ **do**

$s_t^{(b)} = 2$ **(SkipUpdate)**

$\tilde{s}_t^{(b)} = \mathrm{sigm}(\mathrm{cossim}(\mu_t^{(b)}, \boldsymbol{g}_t^{(b)})/\tau)$
$s_t^{(b)} = 0.9 s_{t-1}^{(b)} + 0.1 \tilde{s}_t^{(b)}$ **(Magma)**

$m_t^{(b)} \sim \mathrm{Bernoulli}(0.5)$
$\theta_{t+1}^{(b)} = \theta_t^{(b)} - s_t^{(b)} m_t^{(b)} \Delta_t^{(b)}$
end for

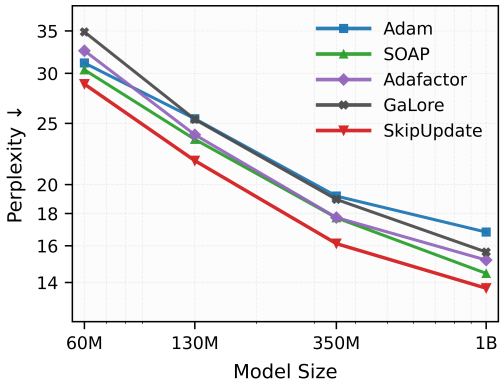

Figure 1: *Pre-training performance on C4.* Despite discarding half of updates, SkipUpdate (masked RMSProp) yields substantial improvements over dense optimizers.

of 15.00 and 15.64, respectively, far behind SkipUpdate's 13.71 (Figure 1). This suggests that the SkipUpdate's benefit comes from stochastic sparsification of the realized parameter update itself, rather than from constraining updates to a simpler subspace.

To explain the effectiveness of the random masking, we show in §2 that it induces an implicit curvature-dependent regularizer that penalizes updates aligned with sharp directions of the loss. This provides a mechanism by which masking can suppress sharp update directions and provides an implicit bias toward flatter regions of the loss landscape, which are associated with better generalization [14, 17, 19]. Importantly, this regularization emerges implicitly from stochastic noise in the update rule, rather than from explicit curvature computation.

We next ask whether SkipUpdate's homogeneous masking rule can be improved for LLM training. Transformer parameter blocks differ substantially in loss curvature and gradient statistics [33, 60], suggesting that masked updates can benefit from adapting to each block's optimization geometry. We use *momentum-gradient alignment* to modulate masked updates, since weak or negative alignment can indicate rapidly changing, noisy, or locally ill-conditioned directions [3, 36, 39]. To this end, **Momentum-aligned gradient masking (Magma)** replaces SkipUpdate's fixed scaling with a block-wise cosine similarity between the stochastic gradient and first-moment estimate (cf. Algorithm 1). A convergence analysis in Appendix A.2 reveals that alignment-based masking can widen the stable optimization regime by damping noisy high-curvature blocks while preserving descent direction. Empirically, Magma improves adaptive optimizers such as Adam and matrix-preconditioned optimizers such as Muon and SOAP (cf. §4) across model sizes up to 1B parameters and across both dense and sparse mixture-of-experts (MoE) architectures. Notably, the gains grow with model size, suggesting that larger models benefit more from this stronger regularization and stabilized optimization.

2. Update Masking as a Regularization

Notation. Let $\{\theta_t^{(b)}\}_{b=1}^B$ denote parameters at iteration t partitioned into B disjoint blocks. Let $\nabla_b l(\theta)$ denote the block gradient with respect to $\theta^{(b)}$, and assume l is three times continuously differentiable near the optimization trajectory. The stochastic gradient of $l(\cdot)$ at t is denoted by $\boldsymbol{g}_t \triangleq \{\boldsymbol{g}_t^{(b)}\}_{b=1}^B$. $\mathbf{H}_{bb'}(\theta_t)$ denotes the (b, b') block of $\nabla^2 l(\theta_t)$. We let $(\mathcal{F}_t)_{t\geq 0}$ be a filtration such that θ_t is $\mathcal{F}_t$-measurable and $\mathbb{E}_t[\cdot] \triangleq \mathbb{E}[\cdot|\mathcal{F}_t]$. We let $\Delta_t = \{\Delta_t^{(b)}\}_{b=1}^B$ be a block-wise update direction from a base optimizer, computed by $\boldsymbol{g}_t$ and optimizer state such as preconditioning statistics.

Main result. Given a block update direction from a base optimizer, SkipUpdate incorporates independent Bernoulli random variables $\{m_t^{(b)}\}_{b=1}^B$, where $m_t^{(b)} \sim \text{Bernoulli}(p)$ with survival probability $p \in (0, 1]$. Then, stochastic block-wise masking is applied to each update $\Delta_t^{(b)}$ as

$$\tilde{\Delta}_t^{(b)} = s_t^{(b)} m_t^{(b)} \Delta_t^{(b)}, \qquad b = 1, \ldots, B, \tag{1}$$

yielding $\tilde{\Delta}_t \triangleq (\tilde{\Delta}_t^{(1)}, \ldots, \tilde{\Delta}_t^{(B)})$. We set $s_t^{(b)} = 1/p$ to make the masked update unbiased; that is, $\mathbb{E}_t[\tilde{\Delta}_t^{(b)}] = \Delta_t^{(b)}$. Importantly, masking mechanisms only modify the realized update $\Delta_t^{(b)}$, leaving the base optimizer state, including dense moment estimates, unchanged. Thus, SkipUpdate and Magma can be applied as generic optimizer wrappers.

At first glance, randomly skipping the updates in (1) seems unlikely to explain the strong gains observed in Figure 1, since it appears to increase only gradient noise while maintaining the expected update direction. The key point, however, is that this noise is highly structured, and it changes the second-order behavior of the optimization dynamics.

As shown in Proposition 1 in Appendix A, SkipUpdate induces a curvature-dependent regularization term in the expected post-update loss, providing a mechanism that links the simple masking rule in (1) to the empirical gains in Figure 1. Specifically, the expected loss with $\tilde{\Delta}_t$ satisfies

$$\mathbb{E}_t\left[l(\theta_t - \tilde{\Delta}_t)\right] \approx l(\theta_t - \Delta_t) + \sum_{b=1}^{B} \frac{1-p}{2p} (\Delta_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t) \Delta_t^{(b)}. \tag{2}$$

The quadratic term $(\Delta_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t)\Delta_t^{(b)}$ acts as a *geometric regularizer*, measuring local curvature along the block update direction. Large positive values of $(\Delta_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t)\Delta_t^{(b)}$ indicate high local curvature along the update $\Delta_t^{(b)}$. Therefore, across training, the quadratic penalty term biases the optimization trajectory away from directions where small perturbations would lead to sharp loss increase. This connects to the flat-minima and sharpness-based perspectives studied in prior work [10, 14, 19]. In practice, we use block-wise masking by treating each individual parameter as a block, and discuss the effect of masking granularity on performance in Appendix F.2.

3. Momentum-Aligned Update Masking

Because transformer parameters exhibit substantial block-wise heterogeneity in curvature and gradient statistics [33, 60], we consider a block-adaptive alternative to SkipUpdate's homogeneous masking rule. In stochastic optimization, directions that remain consistent across iterations are more likely to carry reliable descent signal, whereas rapidly fluctuating directions often reflect noise [3, 36, 39]. Since momentum tracks such temporal consistency, weak or negative momentum-gradient alignment can indicate noisy, abrupt, or ill-conditioned block behavior. We therefore use *momentum-gradient alignment* to modulate potentially destabilizing updates during LLM training.

Based on this idea, we propose Magma, which uses block-wise momentum-gradient alignment to control the masking process. Specifically, Magma computes an alignment score $\tilde{s}_t^{(b)} \in (0, 1)$ as

$$\tilde{s}_t^{(b)} = \text{sigm}(\text{cossim}(\mu_t^{(b)}, \boldsymbol{g}_t^{(b)})/\tau), \quad \text{for each block } b \text{ at iteration } t, \tag{3}$$

where $\mu_t^{(b)} = \beta_1 \mu_{t-1}^{(b)} + (1-\beta_1)\boldsymbol{g}_t^{(b)}$ is the first-moment estimate, $\tau > 0$ is a temperature parameter, $\text{cossim}(\cdot, \cdot)$ is the cosine similarity, and $\text{sigm}(x) \triangleq 1/(1+\exp(-x))$ is the sigmoid function. Cosine

similarity is adopted for its scale-invariant property, which is particularly effective in LLM training where gradient norms vary substantially across both parameter blocks and iterations [15, 53].

Magma scales each masked block update using an EMA-smoothed alignment score, $s_t^{(b)} = 0.9s_{t-1}^{(b)} + 0.1\tilde{s}_t^{(b)}$, which results in a scaled random masked update

$$\tilde{\Delta}_t^{(b)} = s_t^{(b)} m_t^{(b)} \Delta_t^{(b)}, \quad \text{with } m_t^{(b)} \sim \text{Bernoulli}(p). \tag{4}$$

Magma maintains coherent updates with large $s_t^{(b)}$ and suppresses oscillatory updates with small $s_t^{(b)}$. Proposition 2 in Appendix A shows that Magma preserves the block-wise curvature-dependent regularization from stochastic masking, applied to the alignment-scaled direction $s_t^{(b)}\Delta_t^{(b)}$.

Although Magma introduces bias through alignment-based damping, the convergence analysis in Appendix A.2 suggests why this bias can be beneficial. In particular, Theorem 6 shows that Magma changes the effective block smoothness from $L^{(b)}$ to $\widetilde{L}_t^{(b)} = (\rho_t^{(b)}/p)L^{(b)}$, where $\rho_t^{(b)}$ captures how much alignment scaling reduces the block-wise gradient second moment. This is beneficial when weak alignment occurs in noisy or high-curvature blocks. For example, in a high-curvature block, small parameter changes can substantially rotate the local gradient, making $\boldsymbol{g}_t^{(b)}$ disagree with $\mu_t^{(b)}$; similarly, high-variance blocks can produce noisy $\boldsymbol{g}_t^{(b)}$ that are poorly aligned with $\mu_t^{(b)}$. In such cases, Magma reduces the corresponding curvature-weighted noise term, yielding a smaller error floor in stochastic optimization and a wider stable stepsize range. This provides one explanation for Magma's improved performance over dense and uniformly masked baselines, and is consistent with the broader stable learning-rate regime observed in Figure A5.

Closely related work. Momentum-gradient alignment has also been used in Cautious Optimizers [24] and MGUP [4], which attenuate updates through element-wise hard filtering: Cautious Optimizers keep only coordinates whose update and gradient agree in sign, while MGUP upweights top-ranked aligned coordinates and downweights the rest. RPROP [39] similarly adapts step sizes using temporal consistency of gradient signs. In contrast, Magma uses block-level soft alignment scores and couples them with structured stochastic masking, inducing the curvature-dependent geometric regularization in Proposition 2. Appendix E provides a broader discussion of stochastic perturbations, curvature-aware optimization, and LLM training stabilization.

4. Experiments

4.1. Pre-Training Llama

We present Llama 2 pre-training results on the C4 dataset [38] across model sizes of 60M, 130M, 350M, and 1B, following the standardized experimental setup by Zhao et al. [62] described in Appendix D.1. For all experiments, we use $\tau = 2$ and apply Magma only to attention and MLP layers; ablations in Appendix F show this setting is robust.

We examine effectiveness of Magma on four base optimizers, including diagonal adaptive optimizers (Adam and Lion [5]) and matrix-preconditioned optimizer (Muon and SOAP). In Table 1, across all 16 optimizer–scale settings, Magma improves over the corresponding base optimizer, and achieves the best result within its optimizer family in 14 out of 16 cases. For instance, at 1B scale, Magma improves Adam by 14.1% and Lion by 11.6%. Notably, Magma also improves Muon and SOAP, showing that it is compatible with matrix-preconditioned optimizers. While matrix

Table 1: Effect of Magma across optimizers and model scales; validation perplexity is reported for four model sizes, with lower values better. See Table A1 for full results.

Method	60M	130M	350M	1B
Adam	31.15	25.42	19.18	16.82
C-Adam	30.58	24.35	18.33	16.44
Adam+Magma	**30.55**	**23.04**	**16.98**	**14.45**
Lion	37.16	27.43	24.98	18.07
C-Lion	**33.41**	25.64	19.37	16.30
Lion+Magma	33.59	**25.26**	**19.00**	**15.97**
Muon	**28.43**	21.81	16.58	13.29
C-Muon	28.64	21.89	16.52	13.44
Muon+Magma	28.47	**21.76**	**16.21**	**13.22**
SOAP	30.38	23.58	17.72	14.47
SOAP+Magma	**28.65**	**22.12**	**16.57**	**14.07**

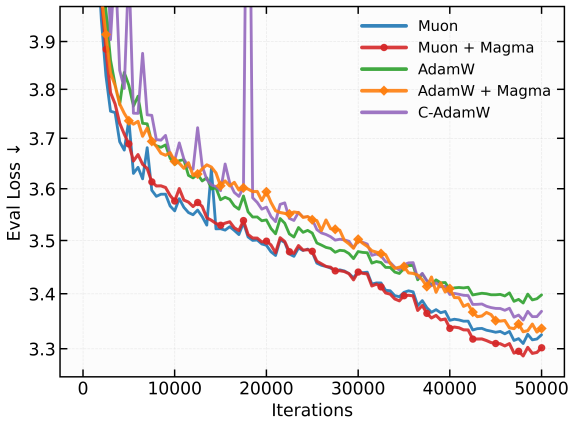

Figure 2: Optimization trajectories of pre-training the Nano MoE model on OpenWebText.

preconditioners refine the update geometry, Magma regularizes the realized block-wise updates, making it a complementary wrapper rather than a competing optimizer design. Magma also scales favorably with model size: averaged across all seven base optimizers, its relative perplexity reduction grows from 3.6% at 60M to 5.4% at 130M, 10.0% at 350M, and 9.8% at 1B. As larger models exhibit increasingly irregular and nonsmooth loss landscapes, this trend supports the interpretation of random masking as an effective form of geometric regularization.

Table 1 also compares Magma against Cautious optimizer [24] on Adam, Lion, and Muon, which shares the similar mechanism of momentum-gradient alignment. While Cautious (C-) optimizer variants generally provide reliable improvements over base algorithms, Magma demonstrates a distinct and more scalable advantage. In particular, at the 1B scale, Magma outperforms its cautious counterpart for every optimizer. The dominance of Magma over Cautious optimizer highlights the effects of the geometric regularization by structural random masking.

4.2. Pre-Training Nano MoE

We also evaluate Magma on sparse MoE pre-training using Nano MoE [54] on OpenWebText [11], following the standard setup described in Appendix D.2. MoE architectures, with dynamic load balancing, sparse token-to-expert routing, and non-uniform gradient flow, induce highly nonsmooth optimization landscapes [9, 43, 63], making them an effective testbed for optimizer robustness.

Figure 2 shows that Magma consistently improves performance of both Adam and Muon in the MoE setting. When applied to Adam, Magma exhibits slower convergence during intermediate training but ultimately achieves superior final performance. Consistent with Llama pre-training, Magma also significantly outperforms the Cautious Optimizer. The improvement on Muon is particularly encouraging, as it shows that Magma remains effective even when sparse MoE training is paired with strong matrix-preconditioned updates.

Robustness in LLM Pre-Training. Controlled experiments in Appendix C isolate two defining challenges of the LLM training regime—heavy-tailed gradient noise and heterogeneous blockwise eigenspectra—and demonstrate that Magma remains highly robust and effective under these conditions. These results show that Magma is particularly well matched to the distinctive demands of LLM pre-training.

References

[1] Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. Linear attention is (maybe) all you need (to understand transformer optimization). In *International Conference on Learning Representations*, 2024.

[2] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

[3] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 2018.

[4] Da Chang and Ganzhao Yuan. Mgup: A momentum-gradient alignment update policy for stochastic optimization. In *Advances in Neural Information Processing Systems*, 2025.

[5] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, 2023.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

[7] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, 2014.

[8] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

[9] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[10] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[11] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[12] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

[13] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[15] Tianjin Huang, Ziquan Zhu, Gaojie Jin, Lu Liu, Zhangyang Wang, and Shiwei Liu. Spam: Spike-aware adam with momentum reset for stable llm training. *arXiv preprint arXiv:2501.06842*, 2025.

[16] Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. NEFTune: Noisy embeddings improve instruction finetuning. In *International Conference on Learning Representations*, 2024.

[17] Stanislaw Jastrzkebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[18] Kaiqi Jiang, Dhruv Malik, and Yuanzhi Li. How does adaptive optimization impact local neural network geometry? In *Advances in Neural Information Processing Systems*, 2023.

[19] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

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

[21] Frederik Kunstner, Alan Milligan, Robin Yadav, Mark Schmidt, and Alberto Bietti. Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. In *Advances in Neural Information Processing Systems*, 2024.

[22] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, 2021.

[23] Tao Li, Pan Zhou, Zhengbao He, Xinwen Cheng, and Xiaolin Huang. Friendly sharpness-aware minimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[24] Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*, 2024.

[25] Qijun Luo, Hengxu Yu, and Xiao Li. Badam: A memory efficient full parameter optimization method for large language models. In *Advances in Neural Information Processing Systems*, 2024.

[26] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, 2015.

[27] Poorya Mianjy, Raman Arora, and Rene Vidal. On the implicit bias of dropout. In *International Conference on Machine Learning*, 2018.

[28] Maximilian Mueller, Tiffany Vlaar, David Rolnick, and Matthias Hein. Normalization layers are all that sharpness-aware minimization needs. In *Advances in Neural Information Processing Systems*, 2023.

[29] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[30] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[31] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

[32] Julie Nutini, Issam Laradji, and Mark Schmidt. Let's make block coordinate descent converge faster: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*, 2017.

[33] Antonio Orvieto and Robert Gower. In search of adam's secret sauce. *arXiv preprint arXiv:2505.21829*, 2025.

[34] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: layerwise importance sampling for memory-efficient large language model fine-tuning. In *Advances in Neural Information Processing Systems*, 2024.

[35] Yan Pan and Yuanzhi Li. Toward understanding why adam converges faster than sgd for transformers. *arXiv preprint arXiv:2306.00204*, 2023.

[36] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964.

[37] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf, 2019. [Online; accessed 28-January-2026].

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[39] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE International Conference on Neural Networks*, 1993.

[40] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[41] Yara Shamshoum, Nitzan Hodos, Yuval Sieradzki, and Assaf Schuster. Compact: Compressed activations for memory-efficient llm training. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

[42] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, 2018.

[43] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[45] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*, 2023.

[46] Hung-Yu Tseng, Yi-Wen Chen, Yi-Hsuan Tsai, Sifei Liu, Yen-Yu Lin, and Ming-Hsuan Yang. Regularizing meta-learning via gradient dropout. In *Asian Conference on Computer Vision*, 2020.

[47] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, 2023.

[48] Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

[49] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

[50] Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. In *International Conference on Machine Learning*, 2020.

[51] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

[52] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*, 2011.

[53] Ziqing Wen, Yanqi Shi, Jiahuan Wang, Ping Luo, Linbo Qiao, Dongsheng Li, and Tao Sun. SRON: State-free LLM training via row-wise gradient normalization, 2025. URL https://openreview.net/forum?id=BtQLBWr6zI.

[54] Cameron R. Wolfe. nanomoe: Minimal mixture-of-experts implementation, 2024. URL https://github.com/wolfecameron/nanoMoE. GitHub repository.

[55] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[56] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, 2020.

[57] Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *AAAI Conference on Artificial Intelligence*, 2021.

[58] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

[59] Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D Lee, Wotao Yin, Mingyi Hong, et al. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark. *arXiv preprint arXiv:2402.11592*, 2024.

[60] Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhiquan Luo. Why transformers need adam: A hessian perspective. In *Advances in Neural Information Processing Systems*, 2024.

[61] Zhongwang Zhang and Zhi-Qin John Xu. Implicit regularization of dropout. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6):4206–4217, 2024.

[62] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. In *International Conference on Machine Learning*, 2024.

[63] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Appendix A. Additional Theoretical Results

A.1. Geometric regularization in SkipUpdate and Magma

In Proposition 1, we show that SkipUpdate induces a curvature-dependent regularization term in the expected loss decrease.

Proposition 1 *Conditioned on $\mathcal{F}_t$ with $\Delta_t = (\Delta_t^{(1)}, \dots, \Delta_t^{(B)})$, the expected loss of SkipUpdate is*

$$\mathbb{E}_t\left[l(\theta_t - \tilde{\Delta}_t)\right] = l(\theta_t - \Delta_t) + \sum_{b=1}^{B} \frac{1-p}{2p} (\Delta_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t) \Delta_t^{(b)} + \mathcal{O}(\sum_{b=1}^{B} \|\Delta_t^{(b)}\|^3). \tag{5}$$

Proof Condition on $\mathcal{F}_t$, under which θ_t and Δ_t are deterministic. A second-order Taylor expansion of $l(\theta_t - \tilde{\Delta}_t)$ around θ_t yields

$$l(\theta_t - \tilde{\Delta}_t) = l(\theta_t) - \sum_{b=1}^{B} (\boldsymbol{g}_t^{(b)})^\top \tilde{\Delta}_t^{(b)} + \frac{1}{2} \sum_{b=1}^{B} \sum_{b'=1}^{B} (\tilde{\Delta}_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t) \tilde{\Delta}_t^{(b')} + R_2(\tilde{\Delta}_t), \tag{6}$$

where $\boldsymbol{g}_t = \nabla l(\theta_t)$, $\mathbf{H}_{bb'}(\theta_t)$ is the (b, b') Hessian block, and the remainder satisfies $|R_2(\tilde{\Delta}_t)| = O\left(\sum_{b=1}^{B} \|\tilde{\Delta}_t^{(b)}\|^3\right)$.

By construction of the masked update and conditioning on (θ_t, Δ_t),

$$\mathbb{E}[\tilde{\Delta}_t^{(b)} \mid \theta_t, \Delta_t] = \frac{1}{p} \mathbb{E}[m_t^{(b)}] \Delta_t^{(b)} = \Delta_t^{(b)}. \tag{7}$$

Moreover, using independence of $\{m_t^{(b)}\}_{b=1}^{B}$,

$$\mathbb{E}\left[(\tilde{\Delta}_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t) \tilde{\Delta}_t^{(b')} \mid \theta_t, \Delta_t\right] = \begin{cases} (\Delta_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t) \Delta_t^{(b')}, & b \neq b', \\ \frac{1}{p} (\Delta_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t) \Delta_t^{(b)}, & b = b'. \end{cases} \tag{8}$$

Taking conditional expectations in (6) and regrouping terms, we obtain

$$\mathbb{E}[l(\theta_t - \tilde{\Delta}_t) \mid \theta_t, \Delta_t] = l(\theta_t - \Delta_t) + \sum_{b=1}^{B} \frac{1-p}{2p} (\Delta_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t) \Delta_t^{(b)} + \mathbb{E}[R_2(\tilde{\Delta}_t) \mid \theta_t, \Delta_t]. \tag{9}$$

Finally, since $\mathbb{E}\|\tilde{\Delta}_t^{(b)}\|^3 = O(\|\Delta_t^{(b)}\|^3)$ for each b, the expected remainder term is of order $O\left(\sum_{b=1}^{B} \|\Delta_t^{(b)}\|^3\right)$, completing the proof. ■

Next, we show that the curvature-dependent regularization effect in Proposition 1 also holds for the scaled masking scheme used by Magma. The difference is that the regularizer is applied to the alignment-scaled update direction.

Proposition 2 *Conditioned on $\mathcal{F}_t$ with $\Delta_t = (\Delta_t^{(1)}, \dots, \Delta_t^{(B)})$, let $s_t = (s_t^{(1)}, \dots, s_t^{(B)})$ be the block-wise Magma alignment scores, and define the alignment-scaled update $\bar{\Delta}_t^{(b)} \triangleq s_t^{(b)} \Delta_t^{(b)}$, for $b = 1, \dots, B$. Then, the expected loss of Magma is*

$$\mathbb{E}_t\left[l(\theta_t - \tilde{\Delta}_t)\right] = l(\theta_t - S_t \Delta_t) + \sum_{b=1}^{B} \frac{1-p}{2p} (s_t^{(b)})^2 (\Delta_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t) \Delta_t^{(b)} + \mathcal{O}\left(\sum_{b=1}^{B} \frac{|s_t^{(b)}|^3}{p^2} \|\Delta_t^{(b)}\|^3\right). \tag{10}$$

Proof Condition on $\mathcal{F}_t$, θ_t, Δ_t, and s_t are deterministic, and the only remaining randomness is the Bernoulli mask. For notational simplicity, write $\bar{\Delta}_t^{(b)} = s_t^{(b)}\Delta_t^{(b)}$ and $\tilde{\Delta}_t^{(b)} = \frac{m_t^{(b)}}{p}\bar{\Delta}_t^{(b)}$.

Similar to Proposition 1, a second-order Taylor expansion of $l(\theta_t - \tilde{\Delta}_t)$ around θ_t gives

$$l(\theta_t - \tilde{\Delta}_t) = l(\theta_t) - \sum_{b=1}^{B}(\boldsymbol{g}_t^{(b)})^\top \tilde{\Delta}_t^{(b)} + \frac{1}{2}\sum_{b=1}^{B}\sum_{b'=1}^{B}(\tilde{\Delta}_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t)\tilde{\Delta}_t^{(b')} + R_2(\tilde{\Delta}_t), \tag{11}$$

where $\boldsymbol{g}_t = \nabla l(\theta_t)$, $\mathbf{H}_{bb'}(\theta_t)$ is the (b, b') Hessian block, and $|R_2(\tilde{\Delta}_t)| = \mathcal{O}\left(\sum_{b=1}^{B}\|\tilde{\Delta}_t^{(b)}\|^3\right)$.

First, the masked update is unbiased for the alignment-scaled update:

$$\mathbb{E}_t[\tilde{\Delta}_t^{(b)}] = \frac{1}{p}\mathbb{E}[m_t^{(b)}]\bar{\Delta}_t^{(b)} = \bar{\Delta}_t^{(b)}. \tag{12}$$

Therefore, the expected first-order term in (11) coincides with the first-order term of the deterministic update $\bar{\Delta}_t$.

Next, consider the second-order terms. If $b \neq b'$, independence of the Bernoulli masks gives

$$\mathbb{E}_t\left[(\tilde{\Delta}_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t)\tilde{\Delta}_t^{(b')}\right] = \frac{1}{p^2}\mathbb{E}\left[m_t^{(b)}m_t^{(b')}\right](\bar{\Delta}_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t)\bar{\Delta}_t^{(b')} \tag{13}$$

$$= (\bar{\Delta}_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t)\bar{\Delta}_t^{(b')}. \tag{14}$$

For the diagonal terms, using $(m_t^{(b)})^2 = m_t^{(b)}$, we obtain

$$\mathbb{E}_t\left[(\tilde{\Delta}_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t)\tilde{\Delta}_t^{(b)}\right] = \frac{1}{p^2}\mathbb{E}\left[(m_t^{(b)})^2\right](\bar{\Delta}_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t)\bar{\Delta}_t^{(b)} \tag{15}$$

$$= \frac{1}{p}(\bar{\Delta}_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t)\bar{\Delta}_t^{(b)}. \tag{16}$$

Now compare this with the Taylor expansion of the deterministic alignment-scaled update:

$$l(\theta_t - \bar{\Delta}_t) = l(\theta_t) - \sum_{b=1}^{B}(\boldsymbol{g}_t^{(b)})^\top \bar{\Delta}_t^{(b)} + \frac{1}{2}\sum_{b=1}^{B}\sum_{b'=1}^{B}(\bar{\Delta}_t^{(b)})^\top \mathbf{H}_{bb'}(\theta_t)\bar{\Delta}_t^{(b')} + R_2(\bar{\Delta}_t). \tag{17}$$

The first-order terms agree by (12). The off-diagonal second-order terms also agree. The only additional second-order contribution comes from the diagonal blocks, where masking inflates the second moment by a factor of $1/p$. Hence, we get

$$\mathbb{E}_t[l(\theta_t - \tilde{\Delta}_t)] = l(\theta_t - \bar{\Delta}_t) + \sum_{b=1}^{B}\frac{1}{2}\left(\frac{1}{p} - 1\right)(\bar{\Delta}_t^{(b)})^\top \mathbf{H}_{bb}(\theta_t)\bar{\Delta}_t^{(b)} + \mathbb{E}_t[R_2(\tilde{\Delta}_t)] - R_2(\bar{\Delta}_t). \tag{18}$$

Since $\frac{1}{2}\left(\frac{1}{p} - 1\right) = \frac{1-p}{2p}$, the desired curvature term follows.

It remains to bound the remainder. For each block, we have

$$\mathbb{E}_t\|\tilde{\Delta}_t^{(b)}\|^3 = \mathbb{E}\left[\left\|\frac{m_t^{(b)}}{p}\bar{\Delta}_t^{(b)}\right\|^3\right] = \frac{1}{p^3}\mathbb{E}\left[(m_t^{(b)})^3\right]\|\bar{\Delta}_t^{(b)}\|^3 = \frac{1}{p^2}\|\bar{\Delta}_t^{(b)}\|^3. \tag{19}$$

Because $\bar{\Delta}_t^{(b)} = s_t^{(b)} \Delta_t^{(b)}$, this becomes

$$\mathbb{E}_t \|\tilde{\Delta}_t^{(b)}\|^3 = \frac{|s_t^{(b)}|^3}{p^2} \|\Delta_t^{(b)}\|^3. \tag{20}$$

Also, we have $\|\bar{\Delta}_t^{(b)}\|^3 = |s_t^{(b)}|^3 \|\Delta_t^{(b)}\|^3$. Therefore, we get

$$\mathbb{E}_t[R_2(\tilde{\Delta}_t)] - R_2(\bar{\Delta}_t) = \mathcal{O}\left(\sum_{b=1}^{B} \frac{|s_t^{(b)}|^3}{p^2} \|\Delta_t^{(b)}\|^3 \right). \tag{21}$$

Substituting $\bar{\Delta}_t^{(b)} = s_t^{(b)} \Delta_t^{(b)}$ gives the equivalent expression in terms of Δ_t and s_t, completing the proof. ■

A.2. Convergence Analysis & Discussion

Setup. We analyze the effect of scaled random masking in Magma through the lens of classical optimization theory. Consider $\theta = (\theta^{(1)}, \dots, \theta^{(B)})$, where $\theta^{(b)} \in \mathbb{R}^{d'}$ and $Bd' = d$. We assume that the objective is lower bounded, i.e., $l_* \triangleq \inf_\theta l(\theta) > -\infty$. For a vector $\boldsymbol{g} = (\boldsymbol{g}^{(1)}, \dots, \boldsymbol{g}^{(B)})$, define the scaled masking operator $\mathcal{M}_t(\boldsymbol{g})^{(b)} \triangleq \frac{m_t^{(b)}}{p} \boldsymbol{g}^{(b)}$. For the block-wise alignment scores $\boldsymbol{s}_t = (s_t^{(1)}, \dots, s_t^{(B)})$, write $\boldsymbol{S}_t \triangleq \boldsymbol{s}_t \otimes \boldsymbol{I}_{d'}$ where $\otimes$ is the Kronecker product. For clarity, we analyze constant-stepsize SGD, $\theta_{t+1} = \theta_t - \eta \boldsymbol{g}_t$, and Magma $\theta_{t+1} = \theta_t - \eta \boldsymbol{S}_t \mathcal{M}_t(\boldsymbol{g}_t)$ as its masked counterpart. This isolates how alignment-dependent masking changes the descent and curvature terms that control stochastic optimization. Proofs are deferred to Appendix B.

We assume a block-wise smoothness condition and regularity conditions for stochastic gradients.

Assumption 3 *There exist constants $L^{(b)} \geq 0$ such that, for all θ and all block perturbations $\Delta = (\Delta^{(1)}, \dots, \Delta^{(B)})$, $l(\theta + \Delta) \leq l(\theta) + \sum_{b=1}^{B} \langle \nabla_b l(\theta), \Delta^{(b)} \rangle + \frac{1}{2} \sum_{b=1}^{B} L^{(b)} \|\Delta^{(b)}\|^2$. Also, for all θ and $b \in [B]$, $\mathbb{E}[\boldsymbol{g}^{(b)}(\theta)] = \nabla_b l(\theta)$ and $\mathbb{E}\|\boldsymbol{g}^{(b)}(\theta)\|^2 \leq \|\nabla_b l(\theta)\|^2 + \sigma_b^2$.*

The assumption captures the empirical heterogeneity of transformer loss landscapes, where different blocks may have substantially different local curvature. We write $\|\boldsymbol{g}\|_L^2 \triangleq \sum_{b=1}^{B} L^{(b)} \|\boldsymbol{g}^{(b)}\|^2$. Further, it assumes unbiased stochastic gradient with bounded block-wise second moment.

One-step descent. The following lemma shows how masking changes the standard descent lemma.

Lemma 4 *Under Assumption 3, Magma satisfies $\mathbb{E}_t[l(\theta_{t+1})] \leq l(\theta_t) - \eta \mathbb{E}_t[\boldsymbol{g}_t^\top \boldsymbol{S}_t \nabla l(\theta_t)] + \frac{\eta^2}{2p} \mathbb{E}_t[\|\boldsymbol{S}_t \boldsymbol{g}_t\|_L^2]$ for all t.*

In Lemma 4, $\mathbb{E}_t[\boldsymbol{g}_t^\top \boldsymbol{S}_t \nabla l(\theta_t)]$ measures how much descent is preserved after the alignment scaling, and $\mathbb{E}_t[\|\boldsymbol{S}_t \boldsymbol{g}_t\|_L^2]$ is the curvature penalty. To make the latter explicit, define the block-wise second-moment scaling factor $\rho_t^{(b)} \triangleq \frac{\mathbb{E}_t \|s_t^{(b)} \boldsymbol{g}_t^{(b)}\|^2}{\mathbb{E}_t \|\boldsymbol{g}_t^{(b)}\|^2}$. Then, we have $\frac{\eta^2}{2p} \mathbb{E}_t[\|\boldsymbol{S}_t \boldsymbol{g}_t\|_L^2] = \frac{\eta^2}{2} \sum_{b=1}^{B} \frac{\rho_t^{(b)}}{p} L^{(b)} \mathbb{E}_t \|\boldsymbol{g}_t^{(b)}\|^2$. Thus Magma replaces the block smoothness constants $L^{(b)}$ by the effective constants $\widetilde{L}_t^{(b)} \triangleq \frac{\rho_t^{(b)}}{p} L^{(b)}$. We also define $\|\boldsymbol{g}\|_{\widetilde{L}_t}^2 \triangleq \sum_{b=1}^{B} \widetilde{L}_t^{(b)} \|\boldsymbol{g}^{(b)}\|^2$ and $\sigma_{\widetilde{L}_t}^2 \triangleq \sum_{b=1}^{B} \widetilde{L}_t^{(b)} \sigma_b^2$.

Convergence analysis. The descent lemma depends on $\mathbb{E}_t[\boldsymbol{g}_t^\top \boldsymbol{S}_t \nabla l(\theta_t)]$. Since a stochastic block direction may occasionally have negative correlation with the true gradient, the lower bound must account for such events. The next lemma shows that alignment scaling preserves a conservative fraction of the first-order descent, up to a variance-level penalty.

Lemma 5 *Under Assumption 3, suppose $s_t^{(b)} \in [s_-, s_+]$ for all b and t (e.g., $s_- = \mathrm{sigm}(-1/\tau)$ and $s_+ = \mathrm{sigm}(1/\tau)$ for Magma). Then, $\mathbb{E}_t[\boldsymbol{g}_t^\top \boldsymbol{S}_t \nabla l(\theta_t)] \geq s_- \|\nabla l(\theta_t)\|^2 - \frac{s_+ - s_-}{4} \sum_{b=1}^B \sigma_b^2$.*

Combining Lemma 4 with Lemma 5 gives the following stationarity guarantee.

Theorem 6 *Under Assumptions 3 and a deterministic constant $\widetilde{L}_{\max}$ such that $\widetilde{L}_t^{(b)} \leq \widetilde{L}_{\max}$ for all t and b almost surely (e.g., $\widetilde{L}_{\max} = \frac{s_+^2}{p} \max_{b\in[B]} L^{(b)}$), it holds that*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla l(\theta_t)\|^2 \leq \frac{2(l(\theta_0) - l_*)}{\eta s_- T} + \frac{2\sigma_C^2}{s_-} + \frac{\eta \bar{\sigma}^2_{\widetilde{L}}}{s_-}, \quad \text{for stepsize } \eta \in (0, s_-/\widetilde{L}_{\max}], \tag{22}$$

where $\sigma_C^2 \triangleq \frac{s_+ - s_-}{4} \sum_{b=1}^B \sigma_b^2$ and $\bar{\sigma}^2_{\widetilde{L}} \triangleq \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\sigma^2_{\widetilde{L}_t}]$.

Theorem 6 has the standard form of a constant-stepsize nonconvex SGD bound. The important difference is that the stochastic error floor depends on the effective curvature-weighted noise $\bar{\sigma}^2_{\widetilde{L}}$. For vanilla SGD (i.e., $p = 1$ and $\boldsymbol{S}_t = \boldsymbol{I}_d$), the bound reduces to $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla l(\theta_t)\|^2 \leq \frac{2(l(\theta_0)-l_*)}{\eta T} + \eta\sigma_L^2$, for $\eta \leq \frac{1}{L_{\max}}$, where $L_{\max} \triangleq \max_b L^{(b)}$ and $\sigma_L^2 \triangleq \sum_{b=1}^B L^{(b)}\sigma_b^2$. The quantity $\sigma_L^2 = \sum_{b=1}^B L^{(b)}\sigma_b^2$ shows that the stationary neighborhood of SGD is controlled by *curvature-weighted gradient noise*. Hence blocks with large $L^{(b)}\sigma_b^2$ highly impacts the error floor. This observation is particularly relevant in ill-conditioned and heterogeneous transformer landscapes, where stability is often constrained by a relatively small set of high-curvature or high-variance blocks [1, 33, 35]. Therefore, it is highly beneficial to attenuate these blocks by masking or scaling. In that case, the effective constants $\widetilde{L}_t^{(b)}$ reduce both the effective noise term $\bar{\sigma}^2_{\widetilde{L}}$ and the effective smoothness envelope $\widetilde{L}_{\max}$, yielding a smaller stationary error floor and a larger admissible stepsize range.

SkipUpdate vs Magma. This perspective separates the roles of random masking and alignment-dependent scaling. Under SkipUpdate (i.e., $s_t^{(b)} = 1$ for all b), $\rho_t^{(b)} = 1$ and $\widetilde{L}_t^{(b)} = \frac{1}{p}L^{(b)}$. Consequently, the convergence bound becomes $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla l(\theta_t)\|^2 \leq \frac{2(l(\theta_0)-l_*)}{\eta T} + \frac{\eta}{p}\sigma_L^2$, for $\eta \leq \frac{p}{L_{\max}}$. Thus, in this worst-case analysis, the pure random masking does not reduce the curvature-weighted noise; rather, the scaled mask inflates it by $1/p$ and shrinks the certified stepsize range by a factor of p. Random masking may still provide useful regularization, but it does not, by itself, target the blocks that dominate the optimization bound.

However, Magma differs by making the effective smoothness alignment-dependent; that is, $\widetilde{L}_t^{(b)} = \frac{\rho_t^{(b)}}{p}L^{(b)} = \frac{\mathbb{E}_t\|s_t^{(b)}\boldsymbol{g}_t^{(b)}\|^2}{\mathbb{E}_t\|\boldsymbol{g}_t^{(b)}\|^2} \cdot \frac{L^{(b)}}{p}$. Its benefit therefore arises when the alignment score is small precisely on blocks with large curvature-weighted noise. The intuition from §3 suggests why this correlation is plausible. To see this, recall that momentum summarizes a recent history of stochastic gradients. If the block curvature is large, then even a small displacement over the momentum window can substantially change the local gradient: $\nabla_b \mathcal{L}(\theta_t) - \nabla_b \mathcal{L}(\theta_{t-k}) \approx H_{bb}(\theta_t)\big(\theta_t^{(b)} - \theta_{t-k}^{(b)}\big)$. Thus,

larger $L^{(b)}$ is likely make the current gradient less consistent with the historical gradients encoded in momentum. At the same time, large stochastic variance injects minibatch-specific components into $\boldsymbol{g}_t^{(b)}$ that may be orthogonal or opposite to the accumulated direction. Consequently, blocks with both sharp curvature and high gradient noise are exactly the blocks where momentum–gradient alignment is expected to be weak.

This explains the role of Magma in the convergence bound. Specifically, when $\boldsymbol{g}_t^{(b)}$ disagrees with the smoothed descent direction $\mu_t^{(b)}$, the score reduces $\rho_t^{(b)}$ and therefore attenuates that block's effective contribution $\widetilde{L}_t^{(b)}\sigma_b^2 = \frac{\rho_t^{(b)}}{p} L^{(b)}\sigma_b^2$. Therefore, unlike uniform masking in SkipUpdate, Magma can reduce the dominant curvature-weighted noise terms when low-alignment blocks coincide with large $L^{(b)}\sigma_b^2$. This explains how alignment-dependent masking can widen the stable optimization regime, being consistent to the wider stable learning-rate regime observed empirically in Figure A5.

Appendix B. Proofs of Claims

B.1. Proof of Lemma 4

Proof

Under Assumption 3, for any θ and any multi-block $\Delta = (\Delta^{(1)}, \cdots, \Delta^{(B)})$, it holds

$$l(\theta + \Delta) \leq l(\theta) + \Delta^\top \nabla l(\theta) + \frac{1}{2}\|\Delta\|_L^2. \tag{23}$$

Applying (23) with $\Delta = -\eta \boldsymbol{S}_t \mathcal{M}_t(\theta)$ yields

$$l(\theta_{t+1}) \leq l(\theta_t) - \eta(\boldsymbol{S}_t \mathcal{M}_t(\boldsymbol{g}_t))^\top \nabla l(\theta_t) + \frac{\eta^2}{2}\|\boldsymbol{S}_t \mathcal{M}_t(\boldsymbol{g}_t)\|_L^2. \tag{24}$$

By the independence between $m_t^{(b)}$ and $\boldsymbol{g}_t^{(b)}$, we have

$$\mathbb{E}_t[(\boldsymbol{S}_t \mathcal{M}_t(\boldsymbol{g}_t))^\top \nabla l(\theta_t)] = \mathbb{E}[\boldsymbol{g}_t^\top \boldsymbol{S}_t \nabla l(\theta_t)], \tag{25}$$

and

$$\mathbb{E}_t[\|\boldsymbol{S}_t \mathcal{M}_t(\boldsymbol{g}_t)\|_L^2] = \frac{1}{p}\mathbb{E}_t[\|\boldsymbol{S}_t \boldsymbol{g}_t\|_L^2]. \tag{26}$$

Therefore, taking conditional expectation $\mathbb{E}_t[\cdot]$ on both side of (24) yields the desired result. ■

B.2. Proof of Lemma 5

Proof We prove the result conditionally on $\mathcal{F}_t$. For each block $b \in [B]$, define $h_t^{(b)} \triangleq \nabla_b l(\theta_t)$ and $X_t^{(b)} \triangleq (\boldsymbol{g}_t^{(b)})^\top h_t^{(b)}$. Let $s_- \triangleq \operatorname{sgm}(-1/\tau)$ and $s_+ \triangleq \operatorname{sgm}(1/\tau)$. Since $s_t^{(b)} \in [s_-, s_+]$, for every real number x we have

$$s_t^{(b)} x \geq s_- x - (s_+ - s_-)x_-, \qquad x_- \triangleq \max\{-x, 0\}. \tag{27}$$

Therefore, we have

$$\mathbb{E}_t\left[s_t^{(b)}(\boldsymbol{g}_t^{(b)})^\top h_t^{(b)}\right] \geq s_- \mathbb{E}_t[X_t^{(b)}] - (s_+ - s_-)\mathbb{E}_t[(X_t^{(b)})_-]. \tag{28}$$

By conditional unbiasedness of the stochastic gradient, it holds that

$$\mathbb{E}_t[X_t^{(b)}] = \mathbb{E}_t[(\boldsymbol{g}_t^{(b)})^\top h_t^{(b)}] = \|h_t^{(b)}\|^2. \tag{29}$$

Now, it remains to bound the negative part of $X_t^{(b)}$. If $h_t^{(b)} = 0$, then $X_t^{(b)} = 0$ and the claim is immediate. Otherwise, let $A_b \triangleq \|h_t^{(b)}\|^2 > 0$. For every $x < 0$, it holds that

$$-x \le \frac{(x - A_b)^2}{4A_b}, \tag{30}$$

since this inequality is equivalent to $(x + A_b)^2 \ge 0$. Therefore, we get

$$(X_t^{(b)})_- \le \frac{(X_t^{(b)} - A_b)^2}{4A_b}, \tag{31}$$

and taking conditional expectations gives

$$\mathbb{E}_t[(X_t^{(b)})_-] \le \frac{\mathbb{E}_t[(X_t^{(b)} - A_b)^2]}{4A_b}. \tag{32}$$

Moreover, since we have $X_t^{(b)} - A_b = ((\boldsymbol{g}_t^{(b)} - h_t^{(b)}))^\top h_t^{(b)}$, applying Cauchy–Schwarz and Assumption 3 gives

$$\begin{aligned}
\mathbb{E}_t[(X_t^{(b)} - A_b)^2] &\le \|h_t^{(b)}\|^2 \mathbb{E}_t\|\boldsymbol{g}_t^{(b)} - h_t^{(b)}\|^2 && (33)\\
&= \|h_t^{(b)}\|^2 \left(\mathbb{E}_t\|\boldsymbol{g}_t^{(b)}\|^2 - \|h_t^{(b)}\|^2\right) && (34)\\
&\le \|h_t^{(b)}\|^2 \sigma_b^2. && (35)
\end{aligned}$$

Thus, we get

$$\mathbb{E}_t[(X_t^{(b)})_-] \le \frac{\sigma_b^2}{4}. \tag{36}$$

Combining (28), (29), and (36) yields

$$\mathbb{E}_t\left[s_t^{(b)}(\boldsymbol{g}_t^{(b)})^\top h_t^{(b)}\right] \ge s_-\|h_t^{(b)}\|^2 - \frac{s_+ - s_-}{4}\sigma_b^2. \tag{37}$$

Finally, summing over $b \in [B]$ gives the desired result:

$$\mathbb{E}_t[\boldsymbol{g}_t^\top \boldsymbol{S}_t \nabla l(\theta_t)] \ge \sum_{b=1}^{B} \left\{ s_-\|\nabla_b l(\theta_t)\|^2 - \frac{s_+ - s_-}{4}\sigma_b^2 \right\}. \tag{38}$$

■

B.3. Proof of Theorem 6

Proof By the definition of $\|\boldsymbol{S}_t\boldsymbol{g}_t\|_L^2$, we have

$$\frac{1}{p}\mathbb{E}_t[\|\boldsymbol{S}_t\boldsymbol{g}_t\|_L^2] = \sum_{b=1}^{B}\tilde{L}_t^{(b)}\mathbb{E}_t[\|\boldsymbol{g}_t^{(b)}\|^2] \leq \sum_{b=1}^{B}\tilde{L}_t^{(b)}(\|\nabla_b l(\theta_t)\|^2 + \sigma_b^2), \tag{39}$$

where the inequality comes from Assumption 3. Then, applying Lemma 5 to Lemma 4 yields

$$\mathbb{E}_t[l(\theta_{t+1})] \leq l(\theta_t) - \eta\mathbb{E}_t[\boldsymbol{g}_t^\top \boldsymbol{S}_t\nabla l(\theta_t)] + \frac{\eta^2}{2p}\mathbb{E}_t[\|\boldsymbol{S}_t\boldsymbol{g}_t\|_L^2] \tag{40}$$

$$\leq l(\theta_t) - \eta s_-\|\nabla l(\theta_t)\|^2 + \eta\sigma_C^2 + \frac{\eta^2}{2}\sum_{b=1}^{B}\tilde{L}_t^{(b)}\left(\|\nabla_b l(\theta_t)\|^2 + \sigma_b^2\right). \tag{41}$$

Rearranging the gradient terms gives

$$\mathbb{E}_t[l(\theta_{t+1})] \leq l(\theta_t) - \eta\sum_{b=1}^{B}\left(s_- - \frac{\eta}{2}\tilde{L}_t^{(b)}\right)\|\nabla_b l(\theta_t)\|^2 + \eta\sigma_C^2 + \frac{\eta^2}{2}\sigma_{\tilde{L}_t}^2. \tag{42}$$

Since $\tilde{L}_t^{(b)} \leq \tilde{L}_{\max}$ and $\eta \leq s_-/\tilde{L}_{\max}$, we have $s_- - \frac{\eta}{2}\tilde{L}_t^{(b)} \geq \frac{s_-}{2}$. Therefore, we have

$$\mathbb{E}_t[l(\theta_{t+1})] \leq l(\theta_t) - \frac{\eta s_-}{2}\|\nabla l(\theta_t)\|^2 + \eta\sigma_C^2 + \frac{\eta^2}{2}\sigma_{\tilde{L}_t}^2. \tag{43}$$

Taking total expectation and summing (43) from $t = 0$ to $T-1$ yields

$$\frac{\eta s_-}{2}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla l(\theta_t)\|^2] \leq l(\theta_0) - \mathbb{E}[l(\theta_T)] + \eta T\sigma_C^2 + \frac{\eta^2}{2}\sum_{t=0}^{T-1}\mathbb{E}[\sigma_{\tilde{L}_t}^2]. \tag{44}$$

Since $l(\theta_T) \geq l_*$, we get

$$\frac{\eta s_-}{2}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla l(\theta_t)\|^2] \leq l(\theta_0) - l_* + \eta T\sigma_C^2 + \frac{\eta^2}{2}\sum_{t=0}^{T-1}\mathbb{E}[\sigma_{\tilde{L}_t}^2], \tag{45}$$

and dividing by $\eta s_- T/2$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla l(\theta_t)\|^2] \leq \frac{2(l(\theta_0) - l_*)}{\eta s_- T} + \frac{2\sigma_C^2}{s_-} + \frac{\eta}{s_- T}\sum_{t=0}^{T-1}\mathbb{E}[\sigma_{\tilde{L}_t}^2]. \tag{46}$$

Finally, using the definition of $\bar{\sigma}_{\tilde{L}}^2$ completes the proof. ■

Appendix C. Additional Experiments

Table A1: Effect of Magma across optimizers and model scales. Validation perplexity is reported for four model scales (60M to 1B). Lower is better. Gray rows denote Magma variants. C- denotes Cautious Optimizer, evaluated only on a subset.

Method	60M	130M	350M	1B
Adafactor	29.84	23.52	17.71	15.00
C-Adafactor	29.65	23.15	16.99	14.67
Adafactor+Magma	**29.16**	**22.22**	**16.43**	**13.12**
Adam	31.15	25.42	19.18	16.82
C-Adam	30.58	24.35	18.33	16.44
Adam+Magma	**30.55**	**23.04**	**16.98**	**14.45**
LaProp	31.27	23.82	19.43	17.44
LaProp+Magma	**30.25**	**22.87**	**16.89**	**14.18**
Lion	37.16	27.43	24.98	18.07
C-Lion	**33.41**	25.64	19.37	16.30
Lion+Magma	33.59	**25.26**	**19.00**	**15.97**
RMSProp	30.28	23.46	17.48	14.69
RMSProp+Magma	**29.59**	**22.38**	**16.52**	**13.51**
Muon	**28.43**	21.81	16.58	13.29
C-Muon	28.64	21.89	16.52	13.44
Muon+Magma	28.47	**21.76**	**16.21**	**13.22**
SOAP	30.38	23.58	17.72	14.47
SOAP+Magma	**28.65**	**22.12**	**16.57**	**14.07**

C.1. Magma under Heavy-Tailed Gradient Noises

A salient feature of training autoregressive language models is the presence of heavy-tailed stochastic gradient noise [21, 58]. To isolate and study the effect of heavy-tailed noise on optimization dynamics, we evaluate Magma using the controlled benchmark proposed in Ahn et al. [1], which provides an abstraction of transformer training while faithfully reproducing key empirical phenomena observed in practice, including the performance gap between Adam and SGD for LLM pre-training. The benchmark considers training a simplified linear transformer to solve a random linear regression task in a meta learning format [2, 12, 47]. In this benchmark, we consider two data regimes, namely, *light-tailed* and *heavy-tailed* settings. Under the heavy-tailed setting, we modify the input sampling distribution to amplify tail behavior in the covariates, thereby inducing heavy-tailed stochastic gradient noise during optimization. We refer to Appendix D.4 for full experimental details.

Figure A1 (top) compares the optimization trajectories of Adam and Adam + Magma under normal and heavy-tailed covariates. While both methods perform similarly under normal noise, Magma significantly outperforms Adam in the heavy-tailed setting, offering insight into its strong empirical performance in LLM pre-training, where heavy-tailed data is ubiquitous [1, 21]. Notably, this result is appealing given Adam's known robustness to heavy-tailed distributions.

To further analyze this behavior, Figure A1 (bottom) reports the robust condition number along training trajectories [18]. Under heavy-tailed noise, Magma consistently attains substantially smaller

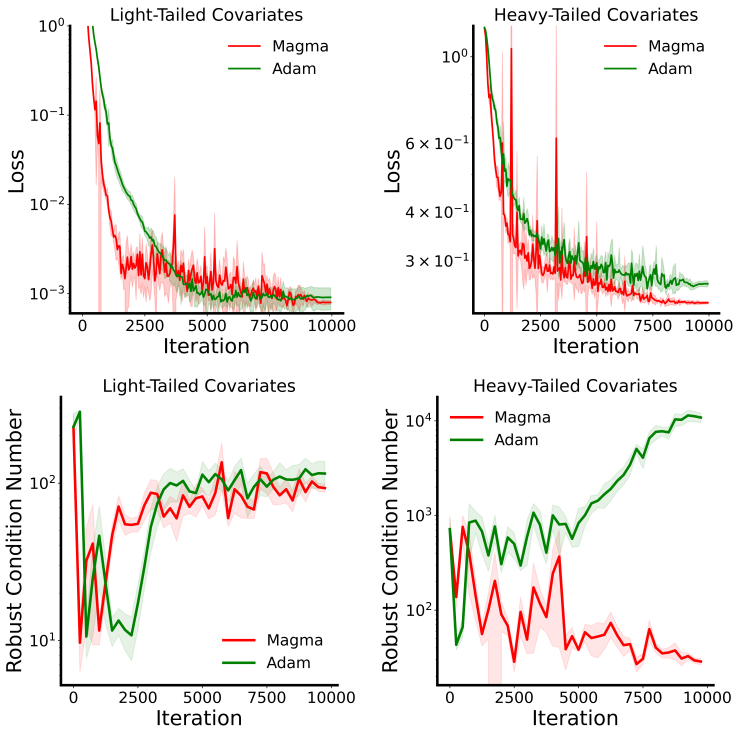

Figure A1: *Magma on light-tailed and heavy-tailed distributions.* ***Top:*** Optimization trajectories for Adam and Adam + Magma. ***Bottom:*** Robust condition number defined as the ratio between the maximum and median eigenvalues of the loss Hessian.

condition numbers, indicating that its updates remain confined to well-conditioned regions of the loss landscape. This reflects the curvature-dependent geometric regularization induced by scaled random masking, which selectively suppresses curvature- and noise-dominated directions, thereby enhancing robustness to extreme gradient fluctuations.

C.2. Magma on Heterogeneous Quadratics

To further validate the effectiveness of Magma in heterogeneous landscapes, we evaluate Magma on a controlled quadratic benchmark adopted in recent works [33, 60]. The benchmark consists of two quadratic objectives with *identical eigenspectra* but different block-wise curvature structure. In both cases, the loss takes the form $l(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{H}\boldsymbol{w}$, where $\boldsymbol{H}$ has eigenvalues spanning three orders of magnitude. The key distinction lies in how these eigenvalues are arranged: in the *homogeneous* setting, eigenvalues of similar scale are grouped within blocks, whereas in the *heterogeneous* setting, each block mixes eigenvalues with vastly different magnitudes, inducing strong curvature misalignment. Despite its simplicity, this heterogeneous structure qualitatively mimics the loss geometry observed in autoregressive transformers, in contrast to the more scale-separated curvature typical of CNNs. Full details are provided in Appendix D.3.

Figure A2 (top) presents optimization trajectories for both Hessian structures. On the homogeneous problem, AdamW + Magma and AdamW exhibit comparable performance, with Magma converging slightly faster in the early stages. However, in the heterogeneous problem, while AdamW is known to substantially outperform non-adaptive methods such as SGD [33, 60], AdamW + Magma achieves faster convergence and a lower final loss than AdamW. This mirrors the gains observed in

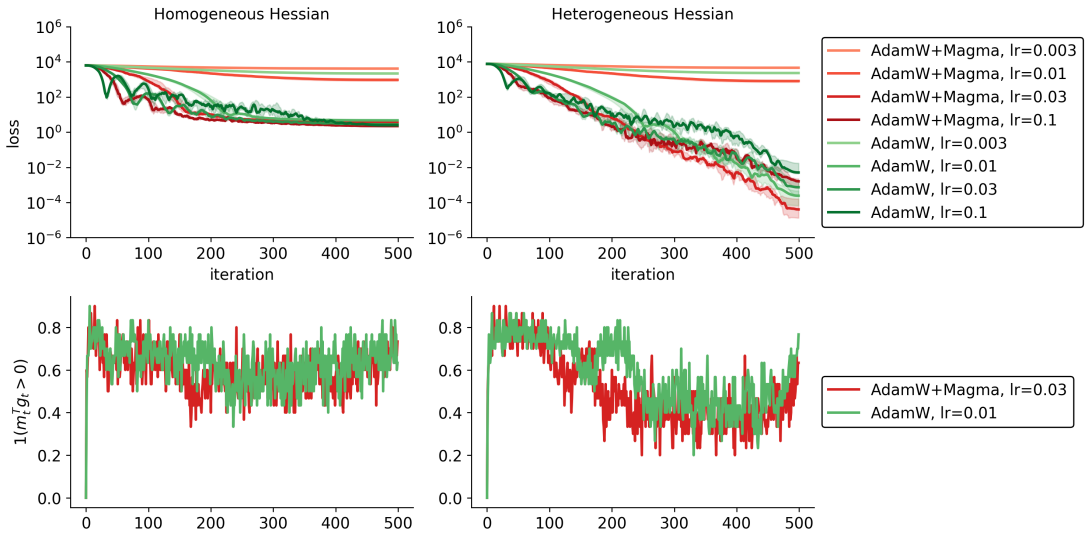

Figure A2: *Magma on homogeneous and heterogeneous quadratics.* ***Top:*** Optimization trajectories for AdamW and AdamW + Magma on quadratic objectives with identical eigenspectra but different block structures. ***Bottom:*** Average gradient–momentum alignment per block.

the pre-training experiments (cf. Table 1), suggesting that Magma is particularly effective in regimes where curvature is both ill-conditioned and misaligned across parameter subspaces. Consistent with the scope of this paper, we do not expect this advantage to transfer uniformly to architectures whose optimization geometry is closer to the homogeneous setting. For example, when applied to ResNet-50 on CIFAR-10 classification, Magma does not consistently improve over AdamW (94.46% vs. 93.82% test accuracy after 100 epochs under carefully tuned configurations). This supports our view that Magma is not a universal optimizer enhancement, but is most relevant to transformer-like regimes characterized by heterogeneous, ill-conditioned, and misaligned curvature.

Figure A2 (bottom) analyzes the average gradient–momentum alignment within each block for AdamW and AdamW + Magma. As expected, the homogeneous Hessian exhibits higher alignment across all blocks, reflecting its more benign conditioning. Notably, although Magma explicitly suppresses updates that conflict with accumulated momentum, it does not significantly increase the alignment score itself. This indicates that Magma's gains arise not from altering momentum statistics, but from enforcing consistency between instantaneous gradients and long-term descent directions.

Appendix D. Experimental Details

D.1. C4 Pre-Training Benchmark Setup

We follow the setup introduced in Zhao et al. [62]. Specifically, we use a fixed batch size of 512 and a max sequence length of 256 with a search grid of [5e-5, 1e-4, 3e-4, 5e-4, 1e-3, 3e-3, 5e-3, 1e-2, 3e-2, 5e-2, 1e-1] for the learning rate. We find that Magma typically performs best when its learning rate is 5–30× larger than the base learning rate due to the scaled random masking. For learning rate scheduling, we implement an initial warm-up for 10% of the total steps, followed by a cosine annealing that decays the learning rate to 10% of the peak value. We run 10K, 20K, 60K, and 100K iterations for the 60M, 130M, 350M, and 1B models, respectively, and report the final evaluation perplexity from the run with the best learning rate.

D.2. Nano MoE Pre-Training Benchmark Setup

We follow the default setup presented in Wolfe [54]. Specifically, we use a GPT2 [37]-style transformer with 124M number of parameters. Further for MoE configuration, the model uses 8 experts per MoE layer, with top-2 routing such that each token is dynamically dispatched to two experts. Further, the MoE layers are applied with a stride of 2, with dense and MoE layers alternating. The model also includes a number of training stabilization techniques, such as auxiliary load-balancing loss and switch transformer [9]-style initialization. Refer to Wolfe [54] for full details. Finally, the model is trained for 50K iterations on 8xA100 GPUs using the default configuration: a batch size of 12, gradient accumulation of 40, a sequence length of 1024 tokens, a minimum learning rate of 5e-6, a weight decay of 0.1, and a grad clip norm of 1.0.

D.3. Heterogeneous Quadratic Benchmark Setup

We provide full details of the quadratic benchmark used in Orvieto and Gower [33]. The setup consists of two quadratic optimization problems in $\mathbb{R}^9$, each defined by a Hessian matrix with an identical eigenspectrum and a 3×3 block-diagonal structure. Concretely, we consider losses of the form $L(w) = \frac{1}{2} w^\top H w$, where the eigenvalues of H are $\{1, 2, 3, 99, 100, 101, 4998, 4999, 5000\}$. While both problems share this eigenspectrums, they differ substantially in how the eigenvalues are arranged across blocks.

In the homogeneous Hessian, eigenvalues are grouped by scale within each block: $\{1, 2, 3\}$, $\{99, 100, 101\}$, $\{4998, 4999, 5000\}$. In contracst, the heterogeneous Hessian interleaves eigenvalues of vastly different magnitudes within each block: $\{1, 99, 4998\}, \{2, 100, 4999\}, \{3, 101, 5000\}$. This structural difference is intended to mimic qualitative distinctions between loss landscapes of autoregressive language models and those of more shallow architectures such as CNNs.

The Hessian H is constructed by first forming diagonal matrices with the specified eigenvalues for each 3×3 block and then applying an independent random rotation to each block. Stochasticity is introduced by defining a design matrix $X = H^{1/2}$ and, at each iteration, subsampling a random subset of rows of X to form a stochastic approximation of the loss and its gradients.

D.4. Heavy-Tailed Gradient Noise Benchmark Setup

We adopt the controlled linear-transformer benchmark introduced by Ahn et al. [1]. In this benchmark, each input sequence corresponds to a distinct linear regression task. Specifically, an input takes the form $z \triangleq \left(\begin{pmatrix} \boldsymbol{x}_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} \boldsymbol{x}_2 \\ y_2 \end{pmatrix}, \cdots, \begin{pmatrix} \boldsymbol{x}_n \\ y_n \end{pmatrix}, \begin{pmatrix} \boldsymbol{x}_{n+1} \\ 0 \end{pmatrix} \right)$, where $\boldsymbol{w}^\top \boldsymbol{x}_i = y_i$ for $i = 1, \ldots, n+1$ and the latent regression vector $\boldsymbol{w} \sim \mathcal{N}(0, I_d)$ is independently sampled for each sequence. The learning objective is to predict y_{n+1} from the context $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, and training minimizes the mean squared prediction error. Following Ahn et al. [1], we use dimension $d = 5$ and context length $n = 20$. This setting has been extensively used to study in-context learning and optimization dynamics of transformers in a simplified yet faithful regime [2, 12, 47].

To study the impact of heavy-tailed stochastic gradients, we consider two covariate distributions. In the *light-tailed* setting, covariates are sampled as $\boldsymbol{x}_i \sim \mathcal{N}(0, I_d)$. In the *heavy-tailed* setting, we sample $\boldsymbol{x}_i$ uniformly from the unit sphere $\mathbb{S}^{d-1}$ and scale each sample by an independent heavy-tailed random variable $\sqrt{\Gamma_{0.1,10}}$, where $\Gamma_{k,\theta}$ denotes the Gamma distribution with shape k and scale θ. This construction significantly amplifies tail behavior in the covariates, thereby inducing heavy-tailed

gradient noise during optimization and enabling a controlled evaluation of optimizer robustness under extreme stochastic fluctuations.

Appendix E. Literature Review

Stabilizing LLM training. A line of recent work addresses instability in LLM training by explicitly constraining optimizer updates in unstable regimes. The method most closely related to Magma is Cautious Optimizer where an update for each parameter is masked when its gradient has an opposite direction to its first-moment estimate for ensuring a descent direction update under a small step size. However, it lacks the structured random masking that promotes flatter optimization trajectories, unlike Magma. More broadly, peculiar optimization landscape of LLM training is known to be unstable due to loss spikes and gradient explosions [6]. Therefore, prior work has explored gradient clipping with a periodic momentum reset to address negative impacts of gradient spikes [15], initialization methods for stable training dynamics [31, 45], and architectural interventions that reshape gradient propagation [8, 49, 56]. Despite this wide exploration, the direction of inducing geometric regularization through stochastic perturbations of the optimization process remains relatively underexplored.

Geometry-aware and trust-region methods. A large body of work improves optimization by incorporating local geometry through curvature-aware updates [7, 13, 26, 57] or trust-region constraints [10, 22]. The former direction aims to approximate second-order structure by tractable preconditioners, recently extending to efficient eigen-decomposition for LLM training [48]. On the contrary, the latter trust-region–flavored methods, such as SAM and its recent variants [23, 28], bias optimization toward flatter regions by optimizing robustness to parameter perturbations, albeit at the cost of additional gradient evaluations. Instead of computing explicit curvature matrices or adversarial perturbation, Magma penalizes sharp curvature along its own block-wise update directions via stochastic masking, offering a lightweight alternative to flatness-seeking optimization.

Stochastic perturbation and noise injection. Stochastic perturbation is a well-established strategy in machine learning, ranging from random gradient masking in meta-learning [46] and federated learning [51] to the annealed Gaussian noise used in Bayesian inference [29, 52]. A prominent example is Dropout [44] that randomly masks hidden units, which has been shown to induce a data-dependent weight-space regularizer that promotes model stability [27, 50, 61]. In modern LLMs, this principle has evolved into embedding-space perturbations, injecting structured noise into token embeddings during instruction fine-tuning [16]. However, the impact of structured and stateful perturbations on optimization dynamics remains relatively under-explored. Magma operates directly on parameter updates and their alignment with local curvature for regularizing the optimization dynamics, highlighting a role for stochastic perturbation particularly suited to LLM training.

Appendix F. Ablation Studies

We conducted ablation studies on a 130M-parameter Llama model trained on the C4 dataset with RMSProp+Magma, following the setup in § 4.1.

F.1. Masking Component

We investigated the efficacy of Magma applied to specific transformer sub-modules (Attention vs. MLP) compared to a global masking strategy. As shown in Table A2, applying masking exclusively

Table A2: Validation perplexity (↓) for different masking components.

	Baseline	Attn Only	Attn + MLP	All
Eval Perplexity	22.64	21.92	21.65	21.94

Table A3: Validation perplexity (↓) for different masking granularities using Uniform vs. Momentum-Gradient Alignment-based sampling schemes (with and without damping). The baseline (RMSProp) excluding both sampling and damping achieves the perplexity of 22.64.

Method	Element	Row	Column	Block
Uniform Sampling	21.73	21.76	21.78	21.81
Damping	21.97	21.95	21.91	21.92
Uniform Sampling + Damping	21.58	21.62	21.61	21.65
Momentum-Gradient Alignment-based Sampling	21.77	21.78	21.75	21.78
Momentum-Gradient Alignment-based Sampling + Damping	21.63	21.60	21.61	21.65

to attention blocks improves model performance, reducing validation perplexity from a baseline of 22.64 to 21.92. We observe a synergistic effect when masking is applied to both attention and MLP simultaneously; this configuration achieves the lowest perplexity of 21.65, surpassing the most comprehensive setting (21.94). These findings indicate that targeted regularization of specific sub-modules yields superior performance compared to uniform masking strategies.

F.2. Masking Granularity

Table A3 summarizes the validation perplexity across four masking granularities (Element, Row, Column, and Block) using different sampling and damping configurations.

Robustness Across Granularities We found that changing the masking granularity has very little impact on stability. For example, with Uniform Sampling, the score varies only slightly—from 21.73 (Element) to 21.81 (Block). This indicates that while fine-grained masking provides a small edge, block masking is preferable for its efficiency—it saves significant memory for cosine similarities with minimal loss in accuracy. We conjecture that the near-equivalence reflects the limited ability of diagonal preconditioning in RMSProp to exploit dense within-block curvature, rendering finer-grained masking of marginal practical benefit.

Effectiveness of Damping and Sampling The results in Table A3 highlight the synergistic effect of combining sampling schemes with damping. While damping alone yields a perplexity improvement over the RMSProp baseline from 22.64 to around 21.92, it does not outperform standalone uniform sampling. However, integrating damping with sampling strategies consistently unlocks the lowest perplexity scores. Uniform sampling + damping achieves the minimum perplexity of 21.58 (Element-level masking), improving upon the standalone uniform sampling by approximately 0.15. On the

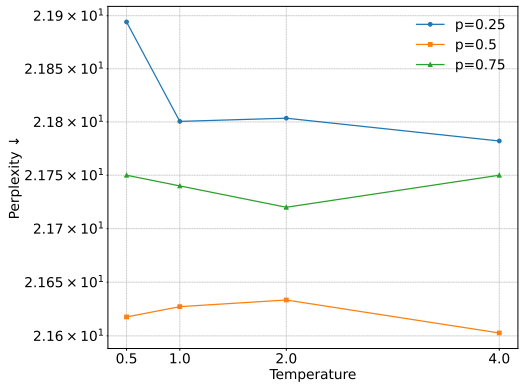

Figure A3: Comparison of eval perplexity for different values of sampling ratio p and damping temperature τ.

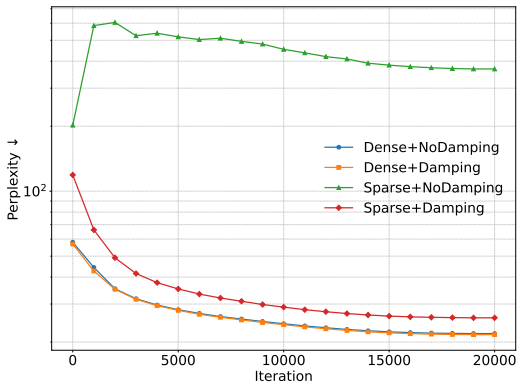

Figure A4: Comparison of training perplexity on the C4 dataset over 20,000 iterations for Dense and Sparse momentum updates, with and without damping.

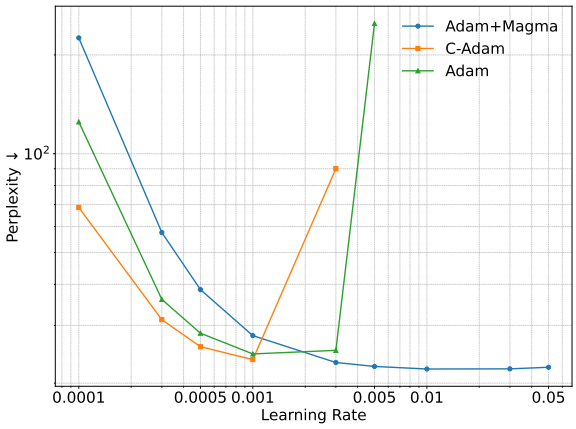

Figure A5: Sensitivity analysis of learning rate on evaluation perplexity. Unlike Adam and C-Adam, which exhibit narrow optimal windows, Adam+Magma maintains consistent performance, demonstrating stability across a significantly wider hyperparameter range.

other hand, momentum-gradient alignment-based sampling, which utilizes the masking probability, does not outperform uniform sampling.

F.3. Sampling Ratio and Damping Temperature

To identify the optimal sampling ratio p and damping temperature τ, we experimented with $p \in \{0.25, 0.5, 0.75\}$ and $\tau \in \{0.5, 1.0, 2.0, 4.0\}$. As Figure A3 demonstrates, $p = 0.5$ outperforms other values across all temperatures. On the other hand, the results are not very sensitive to temperature, so we proceed with $\tau = 2.0$ in all our experiments.

F.4. Sparse vs. Dense Momentum Update

The dense momentum update in SkipUpdate contrasts with subspace optimization methods [25, 34, 62], in which parameters and auxiliary states are updated only on selected coordinates. For example, GaLore [62] selects coordinates via leading gradient singular vectors and updates them over fixed batch intervals. While this reduces optimizer memory, it repeatedly optimizes a suboptimal coordinate subset, contrary to classical coordinate descent insights [30, 32]. Moreover, significant memory savings are not guaranteed in modern LLM training, where memory consumption is dominated by activations rather than parameters or optimizer states [41, 59].

Further, SkipUpdate effectively yields a variance-reduced estimator of the true momentum due to the lazy update scheme, resulting in more stable search directions. Consequently, as shown in Figure A4, the dense momentum updates yield greater stability with improved generalization, compared to the sparse momentum updates as in the memory-efficient subspace optimizers.

For empirical verification, we consider four different settings with a learning rate of 0.001. The results indicate a critical disparity between dense and sparse update methods. As shown in Figure A4, the dense baselines—regardless of damping—consistently maintain robust convergence and achieve the lowest perplexity. In contrast, the sparse momentum update without damping exhibits severe instability, characterized by a sharp increase in perplexity that remains high throughout the 20,000 iterations. Although the introduction of damping stabilizes the model, its trajectory still underperforms dense update baselines.

F.5. Sensitivity to Learning Rate

As Figure A5 demonstrates, Adam+Magma exhibits superior robustness to learning rate variations compared to baseline optimizers. While C-Adam and Adam are sensitive to the learning rate—with perplexity spiking when the learning rate deviates from approximately 0.001–0.003—Adam+Magma maintains stability across a broader spectrum. In particular, it remains effective at rates up to 0.05, a region where other optimizers fail to converge. This suggests Adam+Magma reduces the need for precise hyperparameter tuning, offering greater reliability for resource-constrained experimental setups.