DynMuon: A Dynamic Spectral Shaping View of Muon

Fangzhou Wu FWU89@WISC.EDU
University of Wisconsin–Madison

Rikhav Shah† RDSHAH@MIT.EDU
MIT

Sandeep Silwal† SILWAL@CS.WISC.EDU
University of Wisconsin–Madison

Qiuyi (Richard) Zhang† RICHARD@ELORIAN.AI
Elorian AI

Abstract

In recent years, Muon has emerged as the dominant method for training large language models, and transformers more broadly. The essential difference, when compared to standard gradient descent methods, is to replace the usual update matrix $M = U\Sigma V^\top$ with its polar factor $UV^\top$. In this work, we consider a class of Muon-like updates, where we replace the update M with $U\Sigma^p V^\top$ for some parameter p. We call this a "spectral-shaping" operation, and develop a theory of how to pick p which depends on (a) local curvature of the loss function, (b) noise stemming from stochastic gradients and label noise, and (c) training stage. Our theory and experimentation reveal a previously overlooked behavior: positive p helps early by emphasizing high-curvature directions and accelerating signal contraction, while mildly negative p helps later by reallocating update strength toward low-curvature directions that still contain useful training signals. Building on the insight, we propose `DynMuon`, an efficient dynamic spectral shaping method that schedules p from positive to mildly negative over training. Extensive experiments across model sizes, architectures, and training settings show that `DynMuon` consistently achieves lower validation loss than Muon, while requiring **10.6–26.5%** fewer steps to reach the same target loss. Our code is available at https://github.com/fzwark/DynMuon.

1. Introduction

Muon has recently shown strong empirical performance for LLM training [11]. At a high level, for each matrix-valued model parameter, Muon forms a momentum-averaged gradient matrix $M = U\Sigma V^\top$ and replaces it with its polar factor $UV^\top$. The resulting update preserves the singular directions while "flattening" the singular values, and has been shown to improve convergence and training stability across model scales [1, 17]. This flattening of singular values naturally suggests studying more general spectral transformations, which we call *spectral shaping*, for matrix-valued optimization, and exploring how the relative weighting of spectral directions affects training dynamics.

Recent work has explored spectral shaping in a limited capacity by studying fixed power-law singular-value shapings [3, 28], or transformations that suppress dominant spectral subspaces [9, 37]. However, they still treat spectral shaping as static, seeking better fixed spectral transformations.

†. The remaining authors are listed alphabetically.

Thus, it is not clear whether a single shaping rule is desirable as training dynamics evolve [28]. Relatedly, prior work also lacks a training-dynamics model of spectral shaping, and therefore does not characterize how the relative influence of different spectral directions evolves across training [9, 37]. This leaves a central question unresolved, which is the main focus of our paper:

How should Muon-style spectral shaping adapt across training stages, if at all?

Our Results To answer this question, we generalize Muon as one point in a power-law family of what we call *spectral-shaping* operations, where p is the spectral exponent:

$$D^{(p)} := U\Sigma^p V^\top, \qquad \Sigma^p = \mathrm{diag}(\sigma_1^p, \ldots, \sigma_r^p), \tag{1}$$

for a matrix-valued update $M = U\Sigma V^\top$. This is demonstratively a very expressive family of operations: $p = -1$ gives an inverse-spectrum update, $p = 0$ recovers Muon, and $p = 1$ corresponds to the standard SGD-style updates. To understand training dynamics as a function of p, we develop a simple *noise-aware local modelling* that interprets spectral shaping as curvature-dependent reweighting of the update along *local curvature directions* of the loss landscape. Along these directions, it jointly tracks the *residual signal*, i.e., the remaining parameter distance to a nearby local optimum, and the stochastic gradient noise introduced by minibatch sampling. This mode-wise decomposition reveals an interesting signal–noise trade-off: decreasing p increases residual-signal contraction in "flat", lower-curvature directions, but also amplifies noise along these same directions. It further suggests that as training progresses, the remaining residual signal becomes less concentrated in high-curvature directions and relatively more prominent in small, flat curvature directions (Section B.1).

This leads to a stage-dependent finding: positive p helps early by emphasizing high-curvature directions and accelerating residual-signal contraction, whereas a mildly negative p helps later by refocusing the updates towards flat directions that still carry useful residual signal. Thus, we uncover a previously overlooked late-stage training behavior: dynamically shifting emphasis toward flat directions further improves optimization, a stage-dependent advantage that fixed transformations such as Muon cannot capture. Our empirical observations support the predictions of our modelling and show that adapting p across training improves performance (Sections B.2 to B.4).

Motivated by these observations, we propose `DynMuon`, a dynamic spectral shaping algorithm that adapts the spectral exponent p over training (Section 2). It leverages a simple decreasing logistic schedule for p, interpolating from positive values early in training to mildly negative values later. To realize our scheduled spectral shaping efficiently, `DynMuon` extends Newton–Schulz approximation to approximate $U\Sigma^p V^\top$ for varying values of p, avoiding full SVD and retaining the per-step cost as Muon. Extensive experiments across model sizes and architectures show that `DynMuon` consistently achieves lower validation loss than Muon across training settings, while reaching target losses with up to **26.5%** fewer training steps. To summarize, our main contributions are:

1. We introduce a dynamic spectral-shaping perspective for matrix-valued updates, reframing Muon-style optimization from a fixed spectral operation into the adaptive problem of choosing a suitable spectral exponent p as training dynamics evolve.
2. We develop a noise-aware local model (section B) that explains how the right choice for the spectral exponent changes across training stages through a trade-off between residual-signal reduction and stochastic-noise amplification across local curvature directions.
3. Guided by this model, we uncover a surprising stage-dependent regime: positive spectral exponents help early training, whereas (previously overlooked) negative exponents improve late-stage optimization by emphasizing flat directions that retain useful residual signal.

4. In Section 2, we propose DynMuon, a simple and efficient dynamic spectral-shaping algorithm that adapts the spectral exponent throughout training, yielding consistent improvements over Muon across training settings (Section 3).

2. DynMuon: Dynamic Spectral Shaping

The above analysis and observations motivate DynMuon (Algorithm 1 in Section E), which dynamically adapts spectral shaping by monotonically decreasing the spectral exponent from a positive early-stage value to a mildly negative late-stage value, while maintaining computational efficiency.

Logistic Scheduling of the Spectral Exponent. Although the residual-signal distribution across modes can guide the choice of p, estimating it online would require additional forward and backward passes. We therefore use a simple logistic schedule to approximate a smooth decreasing transition of p_t over training without this extra cost. Given the current training step t and total steps T, we set

$$u_t = \frac{t/T - \tau}{w}, \qquad a_t = \frac{1}{1+\exp(u_t)}, \qquad p_t = p_{\min} + a_t(p_{\max} - p_{\min}).$$

Here, τ controls the transition point and w controls the transition width, with a smaller w producing a sharper switch. We set $p_{\max} = 1$ and $p_{\min} = -0.25$, where $p_{\min} = -0.25$ is the best-performing negative exponent based on our observations in Section B.2 (we also ablate $p_{\min}$; see Section 3).

Efficient Updates. Given a scheduled exponent p_t, exact SVD can realize the corresponding fractional spectral shaping, but it is computationally expensive. Muon avoids SVD by approximating the $p = 0$ polar update with a small fixed number of Newton--Schulz (NS) iterations, but it does not directly support arbitrary exponents. To obtain an efficient implementation, DynMuon uses an equivalent factorization of the target spectral shaping (Appendix Algorithm 2). For the normalized input $X_n = X/\|X\|_F$ and $A := X_n X_n^\top$, we first write

$$U\Sigma^p V^\top = (X_n X_n^\top)^{\frac{p-1}{2}} X_n = (X_n X_n^\top)^{\frac{p}{2}} (X_n X_n^\top)^{-\frac{1}{2}} X_n = A^{\frac{p}{2}} A^{-\frac{1}{2}} X_n. \tag{2}$$

Since $Y_\mu = A^{-\frac{1}{2}} X_n$ corresponds to the Muon update, it can be efficiently approximated by NS. It remains to approximate the left correction $A^{\frac{p}{2}}$. When this factor acts as a mild spectral correction, it introduces only a small adjustment on top of the Muon update. We thus approximate it by a second-order Taylor expansion around the identity I. Letting $E = A - I, \delta = p/2$, we have $A^{\frac{p}{2}} \approx C = I + \delta E + \frac{1}{2}\delta(\delta-1)E^2$. The target spectral shaping is then given by $\widetilde{X} = \|X\|_F^p C Y_\mu$. For $X \in \mathbb{R}^{m\times n}$ with $m \le n$, DynMuon adds only one polynomial correction of cost $O(m^2 n + m^3)$ on top of NS computations. Thus, our method has the *same asymptotic complexity* as NS-based Muon. Figure 10 (right) shows that our approximation closely tracks exact SVD throughout training.

Stable Anchoring for Positive Exponents. DynMuon implements the spectral shaping scheduled exponent p_t through a simple stage-wise scheme. The main reason is stability: when p_t is positive and sufficiently large, the correction factor $A^{\frac{p}{2}}$ is no longer a mild adjustment to the Muon update, so the Taylor approximation around $A \approx I$ can become unreliable. To avoid this instability, DynMuon anchors the positive regime to two stable operators (lines 7–10 in Algorithm 1). DynMuon uses the original update when $p_t \ge 0.25$. For $p_t \in [0, 0.25)$, DynMuon applies standard NS orthogonalization, recovering the Muon-style update. For $p_t \in [p_{\min}, 0)$, the exponent is only mildly negative, so DynMuon uses the efficient spectral approximation described above for a continuous schedule.

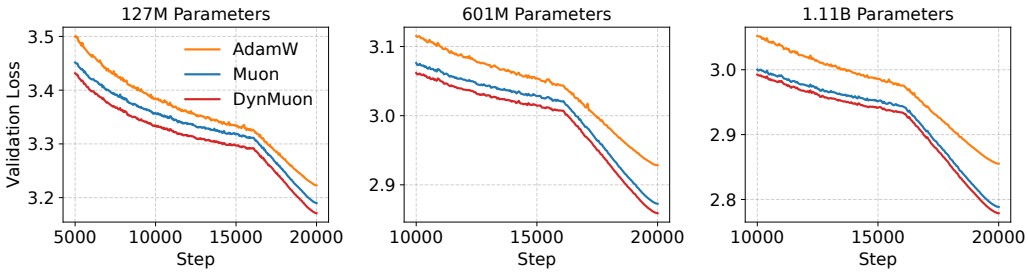

Figure 1: Validation loss trajectories across three model scales trained on 10B tokens. DynMuon consistently achieves the lowest validation loss across all three model scales.

Table 1: Performance and efficiency of DynMuon relative to Muon across GPT-style model scales. Steps to Target uses the validation loss reached by Muon at 80% of training as the target. Step Saving reports the relative step reduction, and Per-Step Time is the average ms/step.

Tokens	Method (Size)	Best Val. Loss (↓)	Steps to Target (↓)	Step Saving (↑)	Per-Step Time (ms)
10B	Muon (127M)	3.190	16000	0.0%	1142.4
	DynMuon (127M)	**3.171**	**12500**	**21.9%**	1150.3
	Muon (601M)	2.872	16000	0.0%	4121.7
	DynMuon (601M)	**2.858**	**13950**	**12.8%**	4200.1
	Muon (1.1B)	2.788	16000	0.0%	6883.3
	DynMuon (1.1B)	**2.776**	**14300**	**10.6%**	7055.8
20B	Muon (127M)	3.139	30400	0.0%	1137.3
	DynMuon (127M)	**3.124**	**22350**	**26.5%**	1151.8
	Muon (601M)	2.808	30400	0.0%	4126.2
	DynMuon (601M)	**2.797**	**25000**	**17.8%**	4184.8
	Muon (1.1B)	2.722	30400	0.0%	6889.77
	DynMuon (1.1B)	**2.713**	**26450**	**13.0%**	6910.1

3. Evaluation

Models and Datasets. We evaluate DynMuon on two decoder-only Transformer families: GPT-style models at multiple scales following modded-nanoGPT [10] and a Qwen-style model, with detailed configurations summarized in Tables 2 and 3. The GPT-style models use rotary position embeddings [34], RMSNorm, and squared ReLU MLPs [33]. The Qwen-style model uses pre-normalized Transformer blocks with RMSNorm, grouped-query attention, and gated SiLU MLPs. All models use sequence length 1024 and global batch size 512. Our main experiments use 10B tokens from FineWeb, and we additionally evaluate on FineWeb-Edu [26]. To study training-budget scaling, we vary the number of training tokens from 2.5B to 20B.

Baselines. We compare DynMuon against Muon [11] and AdamW [19]. Muon is our primary and most directly relevant baseline, while AdamW serves as a standard, widely used optimizer baseline. We further include NorMuon [16] as an additional Muon-variant baseline in Section G. Unless otherwise specified, the default learning rates are 0.01 for Muon and DynMuon, and 0.002 for AdamW. We also vary learning rates for all methods. By default, DynMuon uses $p_{\max} = 1$ and $p_{\min} = -0.25$. Additional evaluation setup details are provided in Section F.

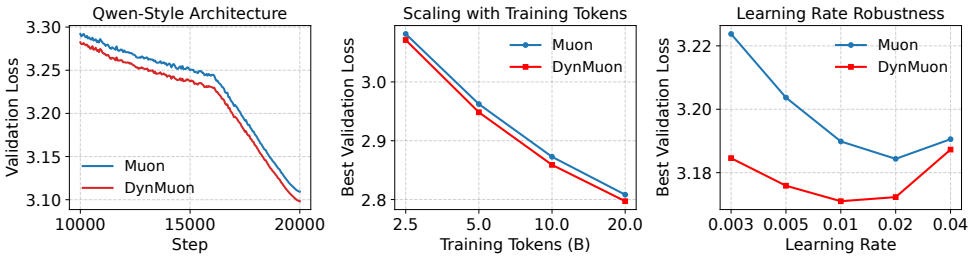

Figure 2: DynMuon outperforms Muon over architectures, training-token budgets, and learning rates.

Main Results. We train GPT-style models at three scales on FineWeb using both 10B and 20B token budgets. As shown in Figure 1, DynMuon consistently achieves the lowest validation loss across all three model scales compared with baselines. The improvement over Muon is clear in the late training stage, where scheduling the spectral exponent toward mildly negative values provides a stable advantage. Table 1 quantifies the practical significance of these gains in terms of both step efficiency and runtime overhead. For each model size and token budget, we define a fixed target as the validation loss reached by Muon at 80% of training, and record the first step at which DynMuon reaches it. Across model scales, DynMuon reaches the target **10.6–26.5%** earlier than Muon, requiring substantially fewer training steps to reach the same validation loss. Meanwhile, DynMuon has a per-step time ratio of only **1.003–1.025**$\times$ relative to Muon, indicating negligible additional runtime cost. Thus, DynMuon improves final performance and step efficiency with minimal per-step overhead.

Model Architecture. To evaluate transferability beyond GPT-style models, we train a 171M Qwen-style decoder-only Transformer using the configuration in Table 3. As shown in Figure 2 (left), DynMuon consistently achieves lower validation loss than Muon, suggesting that dynamic spectral shaping transfers across decoder-only architectures.

Training Token Scale. To evaluate robustness across training budgets, we vary the number of training tokens from 2.5B to 20B for the 601M model. As shown in Figure 2 (middle), DynMuon consistently achieves lower validation loss than Muon across all tested budgets, suggesting that the benefit of dynamic spectral shaping is not tied to a particular training horizon.

Learning Rate. We test learning-rate robustness on the 127M model under the 10B-token budget by sweeping Muon and DynMuon over learning rates from 0.003 to 0.04. As shown in Figure 2 (right), DynMuon outperforms Muon across all tested learning rates and has a flatter curve near its optimum, indicating lower sensitivity to learning-rate choice.

Additional Results Additional results in Section G, including comparisons with exact SVD and NorMuon, spectral-schedule ablations, sensitivity analyses of logistic schedule parameters, and robustness checks across seeds, show that DynMuon consistently achieves the best performance.

4. Conclusions

We studied a broader power-law spectral shaping family, $U\Sigma^p V^\top$. Our noise-aware local model reveals a stage-dependent signal-noise trade-off: early training benefits from positive p that emphasizes high-curvature directions, while late training benefits from mildly negative p that reallocates update strength towards flat directions. We proposed DynMuon, which efficiently schedules p from

positive to mildly negative values during training. Experiments across model scales, architectures, and training settings show that DynMuon consistently achieves lower validation loss than Muon.

References

[1] Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates, 2025. URL https://arxiv.org/abs/2504.05295.

[2] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173. URL https://doi.org/10.1137/16M1080173.

[3] Lizhang Chen, Jonathan Li, and qiang liu. Muon optimizes under spectral norm constraints. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=Blz4hjxLwU.

[4] Shenyang Deng, Boyao Liao, Zhuoli Ouyang, Tianyu Pang, Minhak Song, and Yaoqing Yang. Suspicious alignment of sgd: A fine-grained step size condition analysis, 2026. URL https://arxiv.org/abs/2601.11789.

[5] Sara Dragutinović and Rajesh Ranganath. To use or not to use muon: How simplicity bias in optimizers matters, 2026. URL https://arxiv.org/abs/2603.00742.

[6] Wenzhi Gao, Ya-Chi Chu, Yinyu Ye, and Madeleine Udell. Gradient methods with online scaling. In Nika Haghtalab and Ankur Moitra, editors, *Proceedings of Thirty Eighth Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pages 2192–2226. PMLR, 30 Jun–04 Jul 2025. URL https://proceedings.mlr.press/v291/gao25a.html.

[7] W Brier Glenn et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[8] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1842–1850. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/gupta18a.html.

[9] Zhendong Huang, Hengjie Cao, Fang Dong, Ruijun Huang, Mengyi Chen, Yifeng Yang, Xin Zhang, Anrui Chen, Mingzhi Dong, Yujiang Wang, Jinlong Hou, Qin Lv, Robert P. Dick, Yuan Cheng, Fan Yang, Tun Lu, and Li Shang. Spectra: Rethinking optimizers for llms under spectral anisotropy, 2026. URL https://arxiv.org/abs/2602.11185.

[10] Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024. URL https://github.com/KellerJordan/modded-nanogpt.

[11] 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/.

[12] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

[13] Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization, 2025. URL https://arxiv.org/abs/2503.12645.

[14] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/46a558d97954d0692411c861cf78ef79-Paper.pdf.

[15] Tim Tsz-Kit Lau, Qi Long, and Weijie Su. Polargrad: A class of matrix-gradient optimizers from a unifying preconditioning perspective, 2026. URL https://arxiv.org/abs/2505.21799.

[16] Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. Normuon: Making muon more efficient and scalable, 2025. URL https://arxiv.org/abs/2510.05491.

[17] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025. URL https://arxiv.org/abs/2502.16982.

[18] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[20] Binghang Lu, Jiahao Zhang, and Guang Lin. Muon with spectral guidance: Efficient optimization for scientific machine learning, 2026. URL https://arxiv.org/abs/2602.16167.

[21] Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8828–8836, May 2021. doi: 10.1609/aaai.v35i10.17069. URL https://ojs.aaai.org/index.php/AAAI/article/view/17069.

[22] Jianhao Ma, Yu Huang, Yuejie Chi, and Yuxin Chen. Preconditioning benefits of spectral orthogonalization in muon, 2026. URL https://arxiv.org/abs/2601.13474.

[23] James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL http://jmlr.org/papers/v21/17-678.html.

[24] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2408–2417, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/martens15.html.

[25] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[26] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

[27] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2Oqm2IzTy9.

[28] Xianbiao Qi, Marco Chen, Jiaquan Ye, Yelin He, and Rong Xiao. Delving into muon and beyond: Deep analysis and extensions, 2026. URL https://arxiv.org/abs/2602.04669.

[29] Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining, 2025. URL https://arxiv.org/abs/2509.01440.

[30] Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale, 2023. URL https://arxiv.org/abs/2309.06497.

[31] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018. URL https://arxiv.org/abs/1803.09820.

[32] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2018. URL https://arxiv.org/abs/1708.07120.

[33] David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Searching for efficient transformers for language modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6010–6022. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/2f3c6a4cd8af177f6456e7e51a916ff3-Paper.pdf.

[34] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[35] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

[36] Kaiyue Wen, David Leo Wright Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2J51qUZ0iG.

[37] Yujie Yang. Prism: Structured optimization via anisotropic spectral shaping, 2026. URL https://arxiv.org/abs/2602.03096.

[38] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

Appendix A. Extended Related Work

Muon and Spectral Shaping. Recent works have shown that orthonormalizing matrix-shaped momentum can substantially improve neural network / LLM training [11, 17]. Muon uses Newton–Schulz iterations to produce orthonormalized updates for hidden layers, and recent works explain its effectiveness through trust-region, norm-constrained, and spectral preconditioning perspectives [3, 13, 22, 27]. A growing line of work studies variants of matrix updates, including specific fixed positive choices of p within the $U\Sigma^p V^\top$, as well as spike-aware, anisotropic, or mode-guided transformations [9, 20, 28, 37]. More broadly, matrix-aware optimizers such as Shampoo, SOAP, and PolarGrad exploit non-diagonal geometry and spectral structure beyond coordinate-wise scaling [8, 15, 30, 35]. These methods establish spectral structure as an important design axis, but they largely use fixed or task-specific transformations. In contrast, this paper studies whether the preferred spectral bias should evolve across training stages, systematically developing a dynamic spectral shaping rule.

Orthonormalization-Based Optimizers for LLM Training. Recent work has shown that explicitly orthonormalizing matrix-shaped momentum can substantially improve optimization in large-scale neural network training [11, 17, 29, 36]. In particular, Muon applies Newton–Schulz-style iterations to transform matrix-valued momentum into an orthonormalized update direction, and was introduced as a structure-aware optimizer for hidden-layer matrices with strong empirical performance in neural network training [11]. Subsequent large-scale studies further demonstrate that Muon remains effective in language model pretraining when combined with appropriate weight decay and update scaling, supporting orthonormalization-based optimization as a practical paradigm beyond small-scale settings [17]. Beyond empirical scaling, recent theory has also begun to explain why orthonormalization-based updates can be effective, interpreting Muon through non-Euclidean trust-region optimization, norm-constrained optimization, and spectral preconditioning viewpoints [3, 13, 22, 27]. Taken together, these works establish orthonormalization as a promising matrix-aware optimization principle and motivate studying the spectral design of matrix-valued updates more directly.

Spectral Shaping Beyond Fixed Muon Updates. Recent work has begun to view Muon through a broader spectral lens, rather than treating it solely as a fixed orthonormalization rule. For instance, [28] places Muon within a family of spectral operators of the form $U\Sigma^p V^\top$ and studies how different fixed choices of positive p connect Muon-style updates to momentum and Adam-like normalization. Other recent extensions further explore richer but still largely fixed or task-specific forms of spectral shaping. For example, Spectra [9] argues that LLM training exhibits persistent spectral anisotropy with a dominant spike subspace and a long informative tail, and proposes spike-aware shaping that suppresses dominant directions without amplifying the noise-sensitive tail. PRISM [37] augments first-order spectral descent with low-rank quasi-second-order information to perform anisotropic spectral shaping, while SpecMuon [20] introduces mode-wise spectral guidance for scientific machine learning settings with stiff multi-scale dynamics. More broadly, beyond Muon-specific extensions, matrix-aware optimizers such as Shampoo, SOAP, and PolarGrad also exploit non-diagonal geometry and spectral structure to reshape updates beyond simple coordinate-wise scaling [8, 15, 30, 35]. These works establish spectral structure as an important design axis for matrix-valued optimization. However, they mainly study fixed or task-specific spectral transformations. In contrast, `DynMuon` asks whether the preferred spectral shaping should evolve across different training stages, and studies dynamic spectral shaping rather than a single fixed spectral rule. We primarily compare against Muon because `DynMuon` directly generalizes the Muon update: Muon corresponds to the fixed

exponent $p = 0$, whereas DynMuon dynamically varies p within the same spectral-shaping family. This provides a controlled comparison that isolates the effect of the spectral schedule while keeping the rest of the optimizer design fixed. Muon is also a strong and widely used baseline for LLM training, with recent work showing substantial efficiency gains. Matrix-preconditioned optimizers such as Shampoo, SOAP, and PolarGrad are better viewed as complementary to our study rather than as controlled baselines for isolating the effect of dynamic spectral shaping, since they also change the underlying preconditioner and update rule.

Dynamic Scheduling in Optimization. A long line of work improves training by dynamically scheduling optimizer hyperparameters across different phases of optimization, reflecting the broader principle that different stages of training often benefit from different update behaviors. Classic examples include learning-rate annealing and restart strategies such as SGDR [18], cyclical and one-cycle policies that jointly vary learning rate and momentum [31, 32], and warmup schedules that stabilize adaptive optimizers such as Adam in the early stage of training [21]. Such schedules remain central in modern large-batch and foundation-model training, where warmup and decay of scalar step sizes are often crucial for stable and efficient optimization [38]. More recent work has also begun to study iteration-dependent scaling more directly, for example, by learning online scaling matrices for gradient methods [6], while recent analyses of optimization under ill-conditioned objectives suggest that the relative importance of dominant and bulk subspaces can shift over the course of training [4]. Unlike prior schedules that mainly modulate scalar hyperparameters, DynMuon dynamically schedules the spectral shape of the update itself. This allows the preferred spectral bias to evolve across training rather than remain fixed for the entire run.

Appendix B. Motivating Dynamic Spectral Shaping with an Idealized Model

Although Muon-style methods [11, 17] have shown strong empirical gains from the orthonormalized update, recent analyses suggest that their effectiveness can vary across training conditions [5, 28]. This raises the question of whether fixed Muon-style spectral shaping remains optimal throughout training. Motivated by this, we systematically vary ***spectral component*** p in the spectral-shaping family $U\Sigma^p V^\top$ and study its effect on optimization. Before proposing our method, we first develop a simple noise-aware local model, using a standard local quadratic and gradient-noise approximation, to isolate how p controls the trade-off between useful training signal and gradient noise (Section B.1). In particular, our modelling predicts two stage-dependent regimes: positive p can benefit early training, whereas mildly negative p can improve late-stage training. We *empirically validate these predictions* in Sections B.2 to B.4, showing that our simplified modelling can serve as a useful and *predictive* guide for designing dynamic spectral shaping.

B.1. A Noise-Aware Local Model for Spectral Shaping

Our starting point is to consider one such weight matrix $W_t \in \mathbb{R}^{m\times n}$ at training step t, and take its stochastic gradient $G_t = U_t\Sigma_t V_t^\top$ as the update matrix. Applying spectral shaping with exponent p gives $D_t^{(p)} := U_t\Sigma_t^p V_t^\top$. With the learning rate η, the parameter update is

$$W_{t+1} = W_t - \eta D_t^{(p)} = W_t - \eta (G_t G_t^\top)^{\frac{p-1}{2}} G_t. \tag{3}$$

Let $L(W)$ denote the population loss, and let $W^\star$ be a nearby local minimizer with $\nabla L(W^\star) \approx 0$. We define the ***residual signal*** as $E_t := W_t - W^\star$, measuring the remaining error relative to $W^\star$. To analyze how the shaped update in Equation (3) affects optimization, we study the evolution of this

residual signal under the update. Since the update is driven by the gradient, we relate the gradient to the residual signal E_t by performing a locally linear approximation of the population gradient around $W^\star$ [25]. We further use a one-sided Kronecker-factored approximation to the local Hessian action, in the spirit of K-FAC [24]. We apply this approximation over a short training window in which the effective local curvature is empirically stable (see Figure 6 for empirical support):

$$\nabla L(W_t) \approx \nabla L(W^\star) + \nabla^2 L(W^\star)[E_t] \approx \kappa_t H E_t, \tag{4}$$

where $H = Q\Lambda Q^\top$ is a normalized effective local curvature matrix with $\Lambda = \mathrm{diag}(h_1, \ldots, h_m)$, $h_i \in (0, 1]$, and $\max_i h_i = 1$. The scalar $\kappa_t > 0$ captures the overall curvature scale. The eigenvectors of H define the *modes* of the local loss landscape, where each mode corresponds to one curvature direction and h_i measures the curvature along that direction. Modes with larger h_i are called ***strong modes*** and modes with smaller h_i are called ***flat modes***. To account for stochasticity in the actual training update, we decompose the stochastic gradient G_t into the population gradient and a zero-mean noise Ξ_t, following standard unbiased SGD assumptions [2]. Plugging into (4) gives:

$$G_t = \nabla L(W_t) + \Xi_t \approx \kappa_t H E_t + \Xi_t. \tag{5}$$

Since the shaped update depends on $(G_t G_t^\top)^{\frac{p-1}{2}}$, we relate the spectral structure of $G_t G_t^\top$ to the effective local curvature. Motivated by the common use of gradient second moments as Fisher-type proxies for local curvature [14, 23], we use a curvature-aligned surrogate: $(G_t G_t^\top)^{\frac{p-1}{2}} \approx \alpha_t^{\frac{p-1}{2}} H^{\frac{p-1}{2}}$, where $\alpha_t > 0$ is a scalar factor. This approximation models spectral shaping as a curvature-dependent reweighting. The underlying gradient-curvature alignment is empirically supported in Appendix Figure 6. Substituting this approximation into Equation (3), absorbing $\alpha_t^{\frac{p-1}{2}}$ into the effective learning rate η_t, and using Equation (5), the residual signal evolves as

$$E_{t+1} = E_t - \eta_t H^{\frac{p-1}{2}} (\kappa_t H E_t + \Xi_t) = \left(I - \eta_t \kappa_t H^{\frac{p+1}{2}}\right) E_t - \eta_t H^{\frac{p-1}{2}} \Xi_t. \tag{6}$$

Mode-Wise Signal–Noise Tradeoff. Based on the intuitive "idealized" analysis above, we now arrive at the main training dynamics equation that we focus on. Since p acts through powers of the curvature matrix $H = Q\Lambda Q^\top$ in Equation (6), its effect can vary across curvature directions. This motivates a mode-wise analysis, where we project the residual signal E_t and noise Ξ_t onto the eigenbasis of H. Define $\widetilde{E}_t := Q^\top E_t$ and $Z_t := Q^\top \Xi_t$. Let $\delta_{i,t}$ and $\xi_{i,t}$ denote the i-th coordinates of $\widetilde{E}_t$ and Z_t. Then mode i evolves as

$$\boxed{\delta_{i,t+1} = \left(1 - \eta_t \kappa_t h_i^{\frac{p+1}{2}}\right) \delta_{i,t} - \eta_t h_i^{\frac{p-1}{2}} \xi_{i,t}.} \tag{7}$$

We next consider the squared residual for each mode. Assume that the mode-wise noise is conditionally zero-mean with $\mathbb{E}[\xi_{i,t} \mid \delta_{i,t}] = 0$ and $\mathbb{E}[\xi_{i,t}^2 \mid \delta_{i,t}] = c_{i,t}$, where $c_{i,t}$ denotes the noise level of mode i at step t. Then Equation (7) gives

$$\boxed{\mathbb{E}[\delta_{i,t+1}^2 \mid \delta_{i,t}] = \left(1 - \eta_t \kappa_t h_i^{\frac{p+1}{2}}\right)^2 \delta_{i,t}^2 + \eta_t^2 h_i^{p-1} c_{i,t}.} \tag{8}$$

Thus, p induces a *mode-wise signal–noise trade-off*. The deterministic multiplier $1 - \eta_t \kappa_t h_i^{(p+1)/2}$ controls residual-signal contraction: within the stable range $(0, 1)$, larger $h_i^{(p+1)/2}$ makes this multiplier smaller and contracts $\delta_{i,t}$ faster. Increasing p therefore favors contraction in strong, high-curvature modes, whereas decreasing p increases the relative contraction strength in flat modes.

However, the stochastic term is scaled by h_i^{p-1}, so decreasing p also amplifies noise most strongly in flat modes. Without noise, $p = -1$ would maximize contraction in flat modes; with noise, the choice of p must balance residual-signal contraction against noise amplification. This decomposition also provides the basis for the mode-wise predictions tested in Section B.3, where we empirically estimate the residual-signal "energy" $\delta_{i,t}^2$ and noise level $c_{i,t}$ during training.

Takeaway: The spectral exponent p controls a mode-wise signal–noise tradeoff. Larger p accelerates residual signal contraction in strong modes, while smaller p shifts more emphasis toward flat modes but also amplifies noise along them.

B.2. Why a Slightly Negative Spectral Exponent Is Preferred in the Late Stage

Building on the mode-wise signal–noise tradeoff in Equation (8), we assess how different spectral exponents affect training performance. We focus on $p \in [-1, 1]$, covering the representative cases in **??**. At a high level, training improves when the update reduces residual signal in modes where substantial signal remains. Since varying p changes which modes are emphasized, the preferred choice of p should depend on how the residual signal is distributed across modes during training.

Why Residual Signal Concentrates in Flat Modes Late in Training. For $h_i \in (0, 1]$, the contraction strength $h_i^{\frac{p+1}{2}}$ increases with h_i. Thus, the residual signal in strong modes tends to decay earlier, whereas the signal in flat modes decays more slowly and can remain substantial later in training. This suggests that the residual signal becomes relatively more concentrated in flat modes as training progresses. This is also consistent with the local gradient approximation in Equation (4). Projecting the population gradient onto mode i gives $g_{i,t} \approx \kappa_t h_i \delta_{i,t}$, hence $\delta_{i,t} \approx \frac{g_{i,t}}{\kappa_t h_i}$. Thus, for comparable projected gradient magnitudes, a smaller curvature h_i corresponds to a larger residual signal.

When a Mildly Negative Exponent Helps in the Late Stage. Once the residual signal becomes more concentrated in flat modes in the late training stage, it can be beneficial to place relatively more emphasis on those modes rather than on strong modes whose residual signal has already decayed. Decreasing p achieves this effect by allocating relatively more contraction to flat modes. However, from Equation (8), lowering p also increases the noise level, especially in flat modes. Thus, a negative p can improve over $p = 0$ only when the residual signal in flat modes is large enough relative to the noise. This also explains why the exponent should be only mildly negative: otherwise, noise amplification can outweigh the benefit from reducing the residual signal and degrade optimization.

B.3. Validating Predictions of Our Model

We empirically examine whether the signal-noise conditions predicted by our model arise under Muon ($p = 0$). Using a GPT-style model with hidden dimension 768, we freeze all parameters except one target matrix in the final transformer block and estimate its mode-wise curvature, residual signal, and noise. Full setup and estimation details are provided in Section C.

Empirical Modes and Proxies. In our analysis, modes are curvature directions, which are expensive to track directly in practice. We therefore use the singular directions of the Muon update as empirical modes, since spectral shaping reweights singular values along these directions. For each empirical mode i at step t, we estimate local curvature $\hat{h}_{i,t}$, mode-wise noise level $\hat{c}_{i,t}$, and residual-signal energy $\hat{\delta}_{i,t}^2$. Specifically, $\hat{h}_{i,t}$ is estimated via Hessian-vector products, $\hat{c}_{i,t}$ from the variance of independent mini-batch gradient projections, and $\hat{\delta}_{i,t}^2$ from fixed-probe gradient projections using the local gradient approximation in Equation (4).

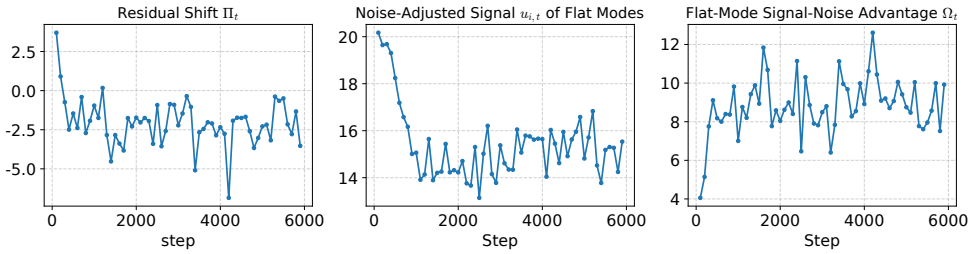

Figure 3: Validation of the mode-wise model predictions. **Left:** The residual-shift metric Π_t decreases during training and becomes negative, indicating that the residual signal shifts from strong, high-curvature modes toward flat modes. **Middle:** The noise-adjusted signal in flat modes remains substantially positive, indicating that flat modes retain the residual signal that is large relative to their noise level. **Right:** The flat-mode signal-noise advantage remains positive, showing that flat modes have a favorable signal-noise tradeoff relative to strong modes in the late stage.

Validating Residual-Signal Concentration in Flat Modes. To test whether the residual signal concentrates in flat modes, we sort modes by curvature $\hat{h}_{i,t}$ and define $\mathcal{S}$ and $\mathcal{F}$ as the top-8 highest-curvature and bottom-8 lowest-curvature modes, respectively. We summarize each bucket by the median $\log$ residual signal energy and define the *residual shift* as $\Pi_t = \mathrm{med}_{i\in\mathcal{S}} \log \hat{\delta}^2_{i,t} - \mathrm{med}_{i\in\mathcal{F}} \log \hat{\delta}^2_{i,t}$. Thus, $\Pi_t < 0$ indicates that flat modes have a larger residual signal than strong modes. Figure 3 (left) shows that Π_t steadily decreases and becomes mostly negative after roughly 500 steps, supporting the predicted late-stage concentration of residual signal in flat modes.

Validating Flat-Mode Signal After Accounting for Noise. Since a negative exponent also amplifies noise, we measure whether the flat-mode residual signal remains useful after accounting for noise. Therefore, we define the mode-wise *noise-adjusted signal* $u_{i,t} := \log \hat{\delta}^2_{i,t} - \log \hat{c}_{i,t}$, which measures the residual signal relative to the noise, and the *flat-mode signal-noise advantage* $\Omega_t = \mathrm{med}_{i\in\mathcal{F}} u_{i,t} - \mathrm{med}_{i\in\mathcal{S}} u_{i,t}$, which compares this noise-adjusted signal between flat and strong modes. Figures 3 (middle, right) show that the noise-adjusted signal $u_{i,t}$ in flat modes remains substantially positive and that Ω_t quickly increases and remains positive. This indicates that flat modes retain the residual signal large enough relative to noise, matching the condition under which a mildly negative exponent can improve over the Muon choice $p = 0$. We provide additional empirical analysis in Section D.

Training Performance for Mildly Negative p. We next test whether the mode-wise analysis translates into improved training performance by switching from $p = 0$ to a negative exponent at step 500, when the residual signal begins to concentrate in flat modes. We first compare negative exponents $p \in \{-0.1, -0.25, -0.5, -0.75, -1\}$ in both last-layer-only and full-model training. As shown in Figure 4 (left, middle), a mildly negative p provides the best performance: $p = -0.1$ and $p = -0.25$ outperform the Muon default $p = 0$, while more aggressive choices such as $p = -0.75$ and $p = -1$ become unstable and perform much worse. This agrees with our analysis: moderate emphasis on flat modes helps late-stage optimization, but overly negative p amplifies noise and degrades performance. Conversely, switching to a positive exponent ($p = 0.25$) at the same step degrades full-model training performance, as shown in Figure 4 (middle). This result further supports our prediction that late-stage gains specifically stem from emphasizing flat modes.

Figure 4: Training performance of stage-dependent spectral shaping. **Left/Middle:** Mildly negative exponents improve late-stage validation loss in both last-layer and full-model training, while overly negative and late-positive exponents degrade performance. **Right:** Early positive p improves full-model training, with the positive-to-negative schedule achieving the lowest validation loss. In contrast, simply doubling the early learning rate or reversing the schedule from negative to positive performs worse than Muon. The panels use capped y-axes; uncapped versions are shown in Appendix Figure 9.

Takeaway: In late training, residual signal shifts toward flat modes, making a slightly negative spectral exponent preferable because it allocates more contraction to these directions. However, this preference is mild: if p is too negative, noise amplification can outweigh the contraction benefit and hurt training. This matches our empirical results, where mildly negative values improve performance while more aggressive negative values degrade it.

B.4. Why a Positive Spectral Exponent Can Help in the Early Stage

Beyond the late-stage preference for a mildly negative exponent, we ask whether the Muon default $p = 0$ is also optimal early in training. Our analysis suggests otherwise: early residual signal can remain concentrated in high-curvature modes, and increasing p accelerates contraction of these strong-mode residual signal while reducing the noise-amplification factor $h_i^{\frac{p-1}{2}}$. Thus, before the residual distribution shifts toward flat modes, a positive exponent can accelerate early optimization by prioritizing the strong-mode residual signal with limited noise amplification.

Training Performance with an Early Positive Exponent. To test this prediction, we use a simple two-stage schedule that sets $p = 1$ for the first 500 steps, then switches to either $p = 0$ or $p = -0.1$. As Figure 4 (right) shows, both early-positive schedules achieve lower validation loss than the fixed Muon baseline, supporting the view that prioritizing strong modes early improves the optimization trajectory. Notably, the schedule transitioning from $p = 1$ to $p = -0.1$ yields the best performance. This validates our stage-dependent picture: a positive exponent accelerates early training, while a mildly negative exponent improves late-stage optimization as the residual signal becomes relatively more concentrated in flat modes. We further test two alternative explanations. First, to rule out the possibility that the early positive p merely acts as a larger effective learning rate, we test a Muon variant that doubles the learning rate during the first 500 steps while keeping $p = 0$. This variant performs worse than fixed Muon, indicating that the gain from an early positive p is distinct from simple step-size scaling. Second, we evaluate a reverse schedule that switches from $p = -0.1$ to $p = 1$ at step 500. This schedule also performs worse than fixed Muon, consistent with our analysis that positive p should be applied early, while mildly negative p is beneficial only later.

Model	d_{model} / Layers / Heads	Tokens/Step	Total Steps	Total Tokens
127M	512 / 24 / 8	0.524M	20K	10B
601M	1280 / 24 / 20	0.524M	20K	10B
1.11B	1792 / 24 / 28	0.524M	20K	10B

Table 2: Model scales for the GPT-style architecture used in our main experiments.

Model	d_{model} / Layers / Heads / KV Heads	MLP Dim.	Tokens/Step	Steps	Total Tokens
171M	512 / 24 / 8 / 2	2816	0.524M	20K	10B

Table 3: Model scale for the Qwen-style architecture used in our experiments.

Takeaway: A positive early-stage exponent helps reduce strong-mode residual signal before the remaining residual signal shifts toward flat modes, yielding lower validation loss than fixed Muon.

Appendix C. Validation Details in Section B

C.1. Validation Setup

To validate the predictions of our model, we run experiments on a GPT-style model with hidden dimension 768, 12 layers, and 6 attention heads [1], trained on FineWeb [26]. We consider two settings: last-layer-only training and full-model training. For last-layer-only training, we freeze all model parameters and optimize only one selected matrix-valued parameter in the final Transformer block. We train for 6000 steps in the last-layer-only setting and 10000 steps in the full-model setting, using up to 3B training tokens. For optimization, we use a learning rate of 0.01 for matrix-valued parameter updates under both Muon and stage-wise spectral shaping. For non-matrix parameter groups, we use AdamW with a learning rate of 0.001. We apply a weight decay of 0.01 to the main matrix parameter groups. The learning rate uses a linear warmup for the first 1% of training steps, followed by cosine decay with a final warmdown ratio of 0.2. For these validation experiments, we use exact SVD to compute spectral shaping for different exponents. For the experiments in Section B.2, we switch from $p = 0$ to a negative exponent at step 500 and compare $p \in \{-0.1, -0.25, -0.5, -0.75, -1\}$ in both last-layer-only and full-model training. For the validations in Section B.4, we use a simple two-stage schedule that sets $p = 1$ for the first 500 steps and then switches to either $p = 0$ or $p = -0.1$.

C.2. Empirical Modes and Proxies

In our analysis, modes are curvature directions, which are expensive to track directly in practice. We therefore use the singular directions of the Muon update as empirical modes, since spectral shaping reweights singular values along these directions. For efficiency, at each step t, we retain the top-k directions with $k = 256$ and represent each as $B_{i,t} := u_{i,t}v_{i,t}^\top$.

For each $B_{i,t}$, we estimate the empirical local curvature using a Hessian-vector product:

$$\hat{h}_{i,t} = \langle \nabla^2 L(W_t) B_{i,t}, B_{i,t} \rangle,$$

This gives the curvature scale along the empirical mode $B_{i,t}$.

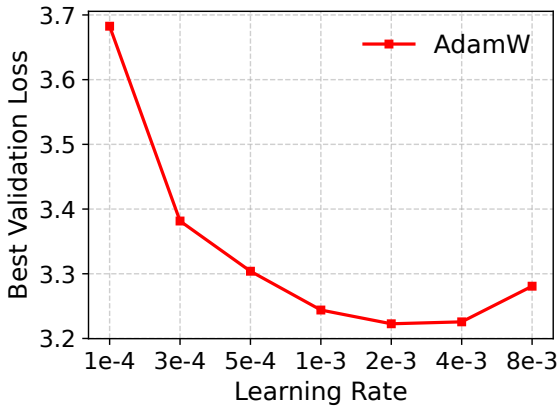

Figure 5: AdamW learning-rate sweep on the 127M GPT-style model, with the best validation loss achieved at 2×10^{-3}.

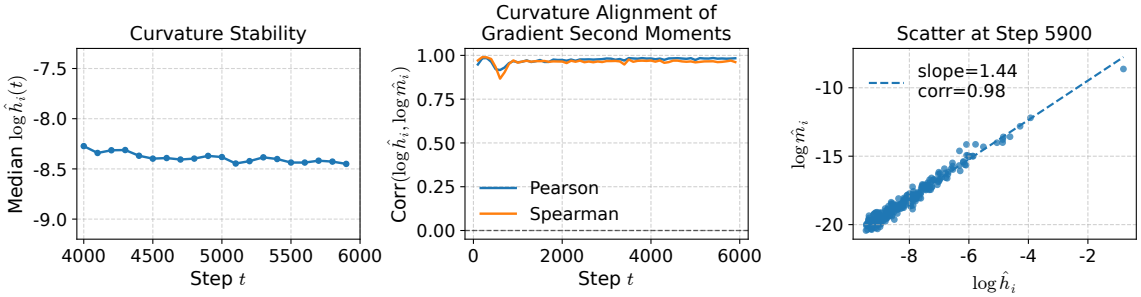

Figure 6: Empirical support for curvature stability and gradient-curvature alignment. **Left:** The median of $\log \hat{h}_{i,t}$ over the retained empirical modes changes only mildly within the 4k–6k step window, supporting our approximation that the effective local curvature can be treated as approximately fixed over short windows. **Middle:** Pearson (↑) and Spearman (↑) correlations across retained empirical modes between the local curvature proxy $\log \hat{h}_{i,t}$ and the gradient second-moment proxy $\log \hat{m}_{i,t}$, where $\hat{m}_{i,t} := \hat{c}_{i,t} + (g_{i,t}^{\text{probe}})^2$, remain strongly positive throughout training. **Right:** At a representative step ($t = 5900$), the scatter plot of $\log \hat{m}_{i,t}$ versus $\log \hat{h}_{i,t}$ shows a clear positive relationship. The left panel supports the local-stability assumption for the effective curvature. The middle and right panels support the gradient-curvature alignment used in the curvature-based approximation to the spectral preconditioner in Equation (6).

We estimate the mode-wise noise level by measuring how much the gradient projection varies across independent mini-batches. Specifically, for $n_b = 32$ mini-batches, we compute

$$g_{i,t}^{(b)} := \langle G_t^{(b)}, B_{i,t} \rangle,$$

and define the noise level proxy as the sample variance

$$\hat{c}_{i,t} := \text{Var}_{b \in [n_b]}[g_{i,t}^{(b)}].$$

To estimate residual signal energy, we use a separate fixed probe set of 8 mini-batches. Let $g_{i,t}^{\text{probe}}$ denote the gradient projection on this fixed probe set. Since the probe gradient is averaged over a

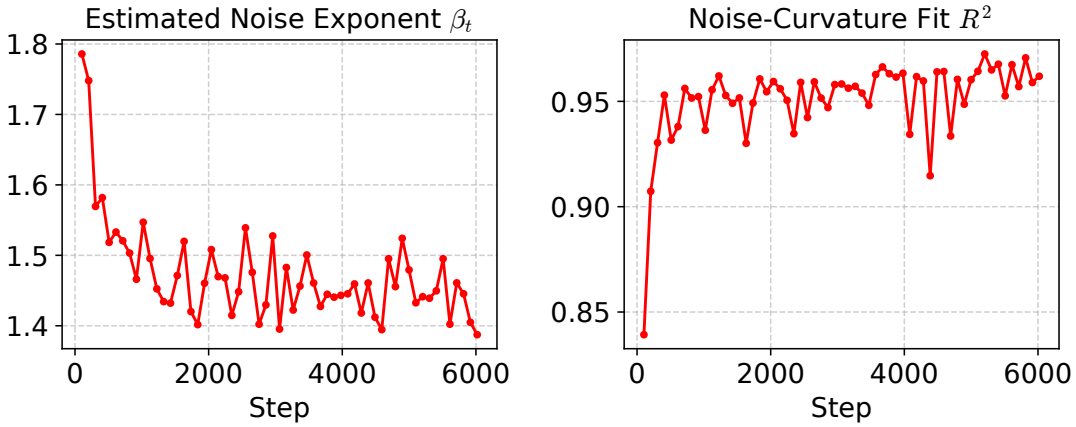

Figure 7: Trends in the estimated noise exponent β_t and the noise-curvature fit R^2 during training. The power-law relationship between noise and curvature remains stable and pronounced throughout training.

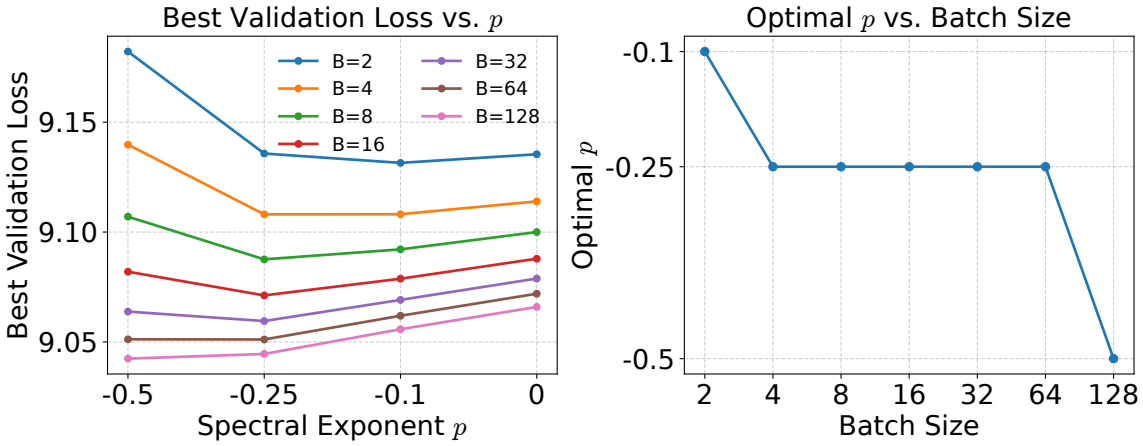

Figure 8: Impact of batch size on the preferred spectral exponent p. Left: best validation loss as a function of p under different batch sizes. Right: the optimal p selected by the best validation loss for each batch size. Smaller batch sizes induce higher gradient noise and favor mildly negative values of p closer to 0, consistent with our analysis that noise amplification limits the benefit of overly negative spectral exponents.

fixed batch set, we use it as a low-variance proxy for the population-gradient projection along $B_{i,t}$. Using the local relation $g_{i,t} \approx \kappa_t h_i \delta_{i,t}$, we estimate

$$\hat{\delta}^2_{i,t} := (g^{\text{probe}}_{i,t})^2 / \hat{h}^2_{i,t},$$

Here, $\hat{h}_{i,t}$ is the HVP-based empirical curvature and already includes the local curvature scale.

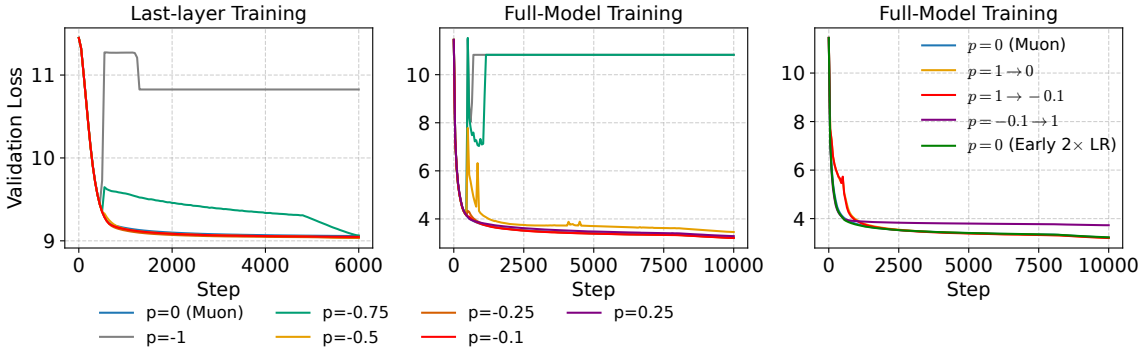

Figure 9: Training performance of stage-dependent spectral shaping. **Left/Middle:** Mildly negative exponents improve late-stage validation loss in both last-layer and full-model training, while overly negative and late-positive exponents degrade performance. **Right:** Early positive p improves full-model training, with the positive-to-negative schedule achieving the lowest validation loss. In contrast, simply doubling the early learning rate or reversing the schedule from negative to positive performs worse than Muon.

Algorithm 1 `DynMuon`

Input: Update matrix M, step t, total steps T, schedule parameters $(p_{\max}, p_{\min}, \tau, w)$
▷ /* Logistic Scheduling */
$u \leftarrow (t/T - \tau)/w$
$a \leftarrow 1/(1+\exp(u))$
$p_t \leftarrow p_{\min} + a(p_{\max} - p_{\min})$
▷ /* Positive Anchoring */
if $p_t \geq 1/4$ **then**
  return M
else if $p_t \geq 0$ **then**
  return `Newton--Schulz`(M)
else
  return `Fast--Spectral`(M, p_t)

Algorithm 2 `Fast--Spectral`

Input: Target matrix X, spectral exponent p
$X_n \leftarrow X/\|X\|_F$,
▷ /* Compute Muon Update */
$Y_\mu \leftarrow$ `Newton--Schulz`(X_n)
▷ /* Low-Order Correction */
$A \leftarrow X_n X_n^\top$
$E \leftarrow A - I$
$\delta \leftarrow p/2$
$C \leftarrow I + \delta E + \frac{1}{2}\delta(\delta-1)E^2$
▷ /* Rescaling */
$\widetilde{X} \leftarrow \|X\|_F^p C Y_\mu$
return $\widetilde{X}$

Appendix D. Additional Analysis in Section B

Stability of Effective Directional Curvatures. To examine the local-stability assumption in Equation (4), we track $\mathrm{median}_{i\in[k]} \log \hat{h}_{i,t}$ over the 4k–6k step window, where $\hat{h}_{i,t}$ denotes the empirical local curvature estimated along the corresponding direction $B_{i,t}$, as defined in Section B.3. As shown in Figure 6 (left), this median changes only mildly over this window, supporting our use of an approximately fixed effective local curvature over short training windows.

Gradient Second Moments Align with Curvature. To support the curvature-alignment approximation in Section B.1, we compare $\hat{m}_{i,t}$ with $\hat{h}_{i,t}$ over the retained empirical modes. For each retained empirical mode i, define

$$\hat{m}_{i,t} := \hat{c}_{i,t} + (g_{i,t}^{\mathrm{probe}})^2,$$

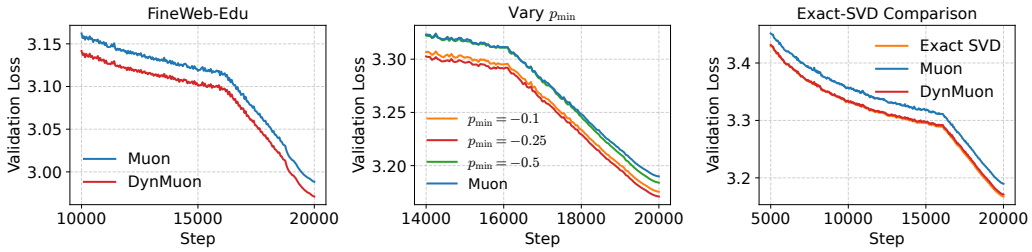

Figure 10: Additional experiments for DynMuon across corpora, $p_{\min}$ choices, and spectral-shaping implementations. **Left:** DynMuon outperforms Muon on FineWeb-Edu. **Middle:** mildly negative $p_{\min}$ values perform best. **Right:** our spectral shaping approximations closely tracks exact SVD.

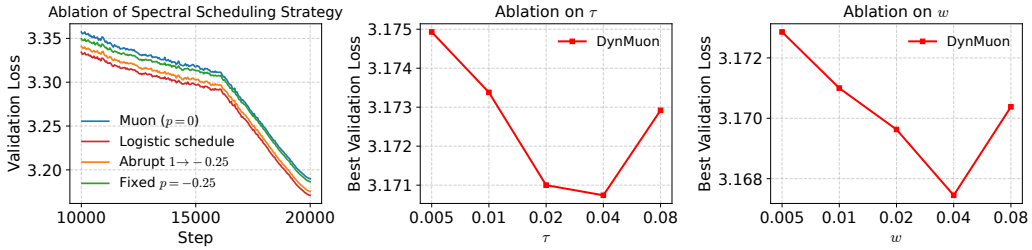

Figure 11: Ablation of spectral scheduling strategies and logistic schedule parameters (τ, w).

where $\hat{c}_{i,t}$ estimates the mode-wise noise level and $g_{i,t}^{\text{probe}}$ estimates the population-gradient projection. Thus, $\hat{m}_{i,t}$ estimates the gradient second moment along that empirical mode. At each saved step, we compute Pearson and Spearman correlations over the retained empirical modes between $\log \hat{m}_{i,t}$ and $\log \hat{h}_{i,t}$. As shown in Figure 6 (middle), these correlations remain strongly positive throughout training, providing empirical support for the approximation that gradient second moments are aligned with the local effective curvature. Figure 6 (right) further illustrates this alignment at a representative step ($t = 5900$), showing a strong positive relationship between $\log \hat{m}_{i,t}$ and $\log \hat{h}_{i,t}$ over the retained empirical modes.

Noise-Curvature Scaling. Motivated by prior work on the geometry of gradient noise, we examine how noise level varies with curvature across modes by fitting a power-law relation $\hat{c}_{i,t} \asymp N_t \hat{h}_{i,t}^{\beta_t}$. The exponent β_t describes the curvature dependence of noise, where a positive β_t means that the raw noise level is larger on high-curvature modes and smaller on flat modes. Figure 7 (left) shows that β_t remains stable around 1.4, while Figure 7 (right) shows consistently high R^2 values, indicating that this power-law relation provides a reliable description of the mode-wise noise structure. Thus, although decreasing p amplifies noise more strongly on flat modes, their raw noise level remains comparatively smaller, leaving room for slightly negative spectral shaping to exploit the flat-mode residual signal.

Impact of Gradient-Noise Amplification on the Preferred Spectral Exponent p. To investigate how gradient-noise amplification affects the preferred spectral exponent p in the late stage, we vary the batch size from 2^1 to 2^7 to induce different noise levels. We compare negative spectral exponents $p \in \{-0.1, -0.25, -0.5\}$ with Muon ($p = 0$) in last-layer training. As shown in Figure 8, smaller batch sizes, which induce higher gradient noise, favor mildly negative exponents closer to 0: the best exponent is $p = -0.1$ when the batch size is 2, and shifts to $p = -0.25$ when the batch size is 16. As the batch size further increases to 128, the preferred exponent becomes more negative, with $p = -0.5$ achieving the best validation loss. This trend is consistent with our analysis: negative

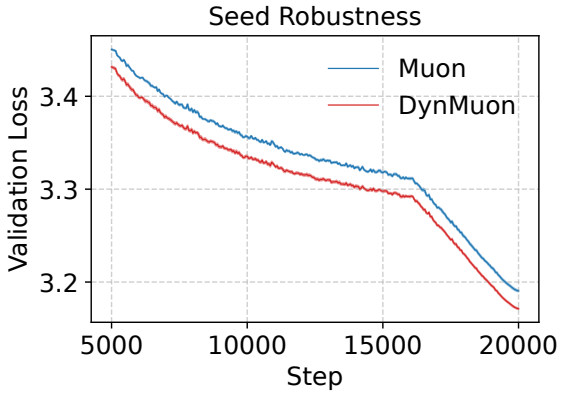

Figure 12: Mean validation loss across three random seeds. Shaded regions indicate one standard deviation, showing that DynMuon consistently outperforms Muon with very low seed variability.

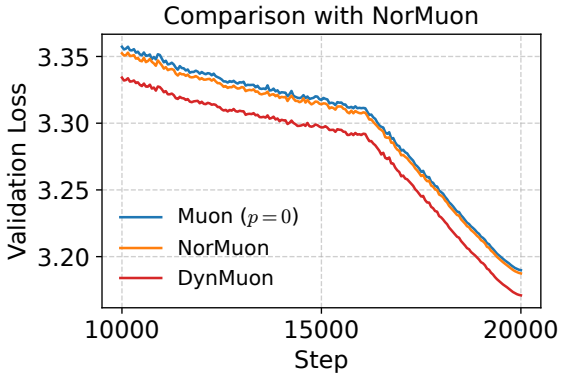

Figure 13: Comparison with NorMuon on the 127M model. DynMuon outperforms both Muon and NorMuon in validation loss.

spectral shaping can improve late-stage optimization by emphasizing flat modes, but overly negative exponents also amplify noise and can degrade performance, especially when the gradient-noise level is high. These results also suggest that reducing gradient noise, for example, through larger batch sizes, may make more negative spectral exponents beneficial.

Appendix E. DynMuon Algorithm Details

We provide the full and efficient implementations of DynMuon in Algorithms 1 and 2. To facilitate reproducibility, we release a lightweight implementation of these core algorithmic components, including the spectral-exponent schedule and the fast spectral-shaping approximation. The implementation is modular and plug-and-play: it only requires a matrix-valued update as input, applies the scheduled spectral-shaping transform, and returns the shaped update for use in existing training code.

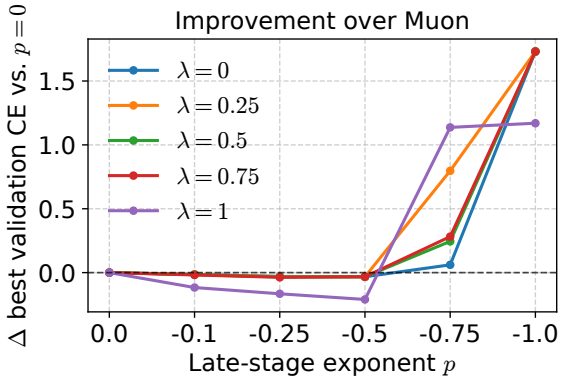

Figure 14: Robustness of mild negative spectral shaping across loss objectives. We plot the best validation CE relative to the corresponding $p = 0$ baseline for each λ. Negative values indicate improvement over Muon. Mildly negative exponents remain beneficial across the CE–Brier interpolation, whereas overly negative exponents degrade performance.

Appendix F. Experimental Details

Models. We consider two decoder-only Transformer families: a GPT-style architecture following the modded-nanoGPT setup [1], and a Qwen-style architecture. The GPT-style models use a GPT-2 tokenizer with a vocabulary size of 50304, rotary position embeddings [34], non-parametric RMSNorm, bias-free linear layers, and squared ReLU activations in the MLP blocks [33]. The Qwen-style models use pre-normalized residual Transformer blocks with grouped-query attention, rotary position embeddings, and a gated MLP with SiLU activation, followed by a final RMSNorm layer and a bias-free output head. As shown in Tables 2 and 3, we evaluate three GPT-style model scales and one Qwen-style model scale. The GPT-style models use $d_{\text{model}} = 512, 1280$, and 1792, while the Qwen-style model uses $d_{\text{model}} = 512$. Across all models, we use the same training setup, with a global batch size of 512, per-device batch size 64, and sequence length 1024.

Baselines. We compare `DynMuon` against two baselines: Muon [11] and AdamW [19]. For Muon and `DynMuon`, we use a default learning rate of 0.01, and for AdamW, we use a default learning rate of 0.002. We use a weight decay of 0.01 for the main matrix parameter groups. Since Muon and `DynMuon` are designed for matrix-valued parameter updates, we use AdamW as the default scalar optimizer with a learning rate of 0.001 for parameters outside the main matrix groups, including the embedding table and output head. We tune the learning rate over $\{0.003, 0.005, 0.01, 0.02, 0.04\}$ for Muon and `DynMuon`, and over $\{10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 10^{-3}, 2\times10^{-3}, 4\times10^{-3}, 8\times10^{-3}\}$ for AdamW. As shown in Figure 5, AdamW achieves its best validation loss at learning rate 2×10^{-3}. We use this tuned AdamW baseline when comparing against Muon and `DynMuon` in the main experiments. For `DynMuon`, we use $w = 0.01$, $\tau = 0.02$, $p_{\max} = 1$, and $p_{\min} = -0.25$ by default. For all methods, we use a linear warmup for the first 0.01 of training steps, followed by a cosine decay over the remaining steps, with a final warmdown ratio of 0.2.

Dataset. We train our models on pretraining datasets FineWeb and FineWeb-Edu [26]. Unless otherwise specified, our main setting uses a training budget of 10B tokens of the FineWeb dataset. To study the effect of data scale, we vary the training budget from 2.5B to 20B tokens, corresponding to 5K, 10K, 20K, and 38K training steps, respectively.

Devices. We use NVIDIA H200 GPUs for all experiments.

Appendix G. Additional Experiments

Training Dataset. We test corpus robustness by replacing FineWeb with FineWeb-Edu on the 127M model while keeping all other settings unchanged. As shown in Figure 10 (left), DynMuon consistently outperforms Muon on FineWeb-Edu, with the advantage becoming more pronounced in the late stage. This suggests that the benefit of dynamic spectral shaping is robust to changes in the training corpus.

Ablation on $p_{\min}$. We vary the scheduling endpoint $p_{\min}$ on the 127M model. As shown in Figure 10 (middle), mildly negative choices outperform Muon, with $p_{\min} = -0.25$ achieving the best validation loss. More aggressive negative choices, e.g., $p_{\min} = -0.5$, perform worse, consistent with our analysis that overly negative exponents can degrade training performance.

Comparison with Exact Spectral Operations. We compare DynMuon with an exact-SVD implementation of the same dynamic spectral schedule. As shown in Figure 10 (right), DynMuon closely matches exact SVD in validation loss, and both outperform Muon. Since exact SVD is roughly $3\times$ slower, DynMuon captures the benefit of dynamic spectral shaping at much lower cost.

Ablation on Spectral Scheduling, Logistic Parameters (τ, w). We ablate both the scheduling strategy and the logistic schedule parameters (τ, w). As shown in Figure 11 (left), our default logistic schedule outperforms standard Muon and two ablations: an abrupt switch from $p = 1$ to $p = -0.25$ at step 500, and a fixed negative schedule with $p = -0.25$ throughout training. The abrupt schedule underperforms the logistic schedule, suggesting that a smooth transition between spectral shaping is more effective than a sharp switch. The fixed negative schedule performs substantially worse, showing that negative shaping throughout training is insufficient and supporting our stage-dependent design. Figures 11 (middle, right) further show that DynMuon is reasonably robust to the transition point τ and transition width w, with the best performance observed around $\tau = 0.04$ and $w = 0.04$.

Seed Robustness. To assess the robustness of DynMuon to training randomness, we run Muon and DynMuon with three different seeds $\{0, 1, 42\}$. Figure 12 reports the mean validation loss with one-standard-deviation bands across seeds. DynMuon consistently outperforms Muon, and the very small across-seed variance suggests that the improvement is robust to training randomness rather than seed-specific effects.

Comparison with NorMuon. We further compare DynMuon with NorMuon [16], a recent Muon variant that augments Muon orthogonalization with neuron-wise normalization based on second-moment statistics. We implement NorMuon following its original algorithmic design in our controlled setting. As shown in Figure 13, DynMuon consistently achieves lower validation loss than both Muon and NorMuon.

Appendix H. Discussion on Robustness Across Loss Objectives

The preferred spectral exponent may depend not only on the optimizer but also on the loss objective, since different losses induce different gradient and noise structures. Cross-Entropy (CE) emphasizes poorly predicted target tokens through the $-\log \hat{y}_c$ term, whereas squared-error-like probability losses impose smoother penalties on probability errors. These differences can alter the distribution of the residual signal and stochastic noise across spectral directions. Since p controls the relative update

strength across these directions, we investigate whether the mild negative regime identified above is specific to CE or remains beneficial under other probability-space objectives.

To investigate this question, we use a simple parameterized family of probability-space losses that interpolates between CE and a squared-error-like objective:

$$L_\lambda(y, \hat{y}) = (1 - \lambda)\mathrm{CE}(y, \hat{y}) + \lambda\mathrm{Brier}(y, \hat{y}), \tag{9}$$

where $\hat{y}$ denotes the predicted probability distribution and $\mathrm{Brier}(y, \hat{y}) = \sum_i (y_i - \hat{y}_i)^2$ is the squared error on predicted probabilities [7]. The endpoint $\lambda = 0$ recovers the standard CE objective, while $\lambda = 1$ gives the Brier score, a probability-space analogue of MSE. By varying λ, we systematically change how the loss weights prediction errors and test whether the preferred mild-negative spectral regime remains robust.

We use the same last-block target-matrix setting as in Section B.2, but train for 3000 steps. For each loss interpolation parameter $\lambda \in \{0, 0.25, 0.5, 0.75, 1.0\}$, we sweep late-stage exponents $p \in \{0, -0.1, -0.25, -0.5, -0.75, -1.0\}$. As before, we use $p = 0$ before the switch step and apply the target exponent afterward. Because the training objective changes with λ, objective values are not directly comparable across different loss choices. We therefore evaluate all runs using validation CE as a common metric, since it is the standard language-modeling measure of next-token predictive quality and is directly tied to perplexity [12].

For each configuration (λ, p), we report the difference between its best validation CE and that of the corresponding $p = 0$ baseline under the same λ. Results in Figure 14 show that mildly negative exponents consistently improve upon the Muon baseline across the CE–Brier interpolation, while more aggressive negative exponents such as $p = -0.75$ and $p = -1$ degrade performance. This suggests that the late-stage benefit of mild negative spectral shaping is not specific to standard cross-entropy, but persists across this family of probability-space loss objectives.

Parallel Training. `DynMuon` preserves the matrix-wise update structure of Muon and only changes the local spectral shaping rule through the scheduled exponent p_t. It therefore introduces no additional cross-layer or cross-device coupling beyond Muon, and should be compatible with existing Muon-style parallel training implementations.

Appendix I. Broader Impact

This work proposes a general-purpose optimization method for training LLMs. Its main potential benefits are improved training efficiency and performance, which may reduce compute costs and energy usage. As a technical contribution, the method does not pose direct societal risks. We believe it promotes LLM training, and future work can further address potential concerns by incorporating responsible use guidelines.