LEARNABLE ADAPTIVE SPECTRAL REPARAMETERIZATION FOR IMPROVING LLM PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances suggest that flatter and better-conditioned weight spectra are closely associated with improved LLM pretraining and generalization, motivating increasing interest in shaping the spectral geometry of model parameters. However, existing approaches typically rely on the prescribed spectral transformations, whose predefined mappings impose strong priors on spectral evolution and may restrict the flexibility required during training. Moreover, different layers and parameter types can favor distinct spectral geometries, making a uniform transformation potentially suboptimal across the model. To address this limitation, we propose Learnable Adaptive Spectral Reparameterization (LASR), which enables individual weight matrices to learn their preferred spectral transformations from the training objective. Specifically, LASR constructs a learnable family of spectral mappings through a truncated K-th-order expansion of the reciprocal square-root function, with trainable coefficients controlling the contributions of different polynomial bases. This parameterization spans diverse feasible spectral amplification behaviors while ensuring bounded spectral reshaping. By jointly optimizing these coefficients with the model, each weight matrix adaptively learns its spectral transformation according to its training dynamics. Consequently, LASR turns spectral reshaping from a prescribed, model-wide operation into a weight-specific learnable mechanism. Extensive experiments across pretraining, downstream evaluations, and spectral analyses consistently demonstrate the effectiveness of LASR.

1 INTRODUCTION

Recent advances in large language model (LLM) training have drawn increasing attention to the geometry of model parameters (Yang et al., 2023; Loshchilov et al., 2025). In particular, the singular-value spectrum of weight matrices provides a direct characterization of how linear transformations amplify or suppress signals along different directions, thereby influencing forward propagation, backward optimization, and ultimately model generalization. A growing body of empirical and theoretical evidence suggests that flatter and better-conditioned spectra are often associated with more stable optimization and improved downstream performance. These observations have motivated a variety of approaches that explicitly regulate or reshape weight spectra during training, ranging from spectral normalization (Miyato et al., 2018) and orthogonalization (Huang et al., 2020) to spectral clipping and polynomial weight preconditioning (Newhouse et al., 2026; Wang et al., 2026).

Despite their different formulations, existing spectral-shaping methods largely follow the same design paradigm: a desired spectral transformation is specified in advance and then an corresponding matrix operation is constructed to approximate it (Grishina et al., 2025; Newhouse et al., 2026; Wang et al., 2026). For example, orthogonalization methods progressively drive nonzero singular values toward unity, spectral capping suppresses singular values beyond a predefined threshold, and recent polynomial preconditioning methods fit low-degree polynomials to prescribed soft-shaping targets. From a unified perspective, these methods determine beforehand a scalar mapping that dictates how each singular value should evolve. While such prescriptions provide effective control over weight geometry, they also impose a strong prior on the spectrum throughout training.

This naturally raises a more fundamental issue: *whether a prescribed spectral transformation can adequately accommodate the heterogeneous roles and optimization dynamics of different weight matrices*. Such heterogeneity is inherent in modern architectures, where different weight matri-

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

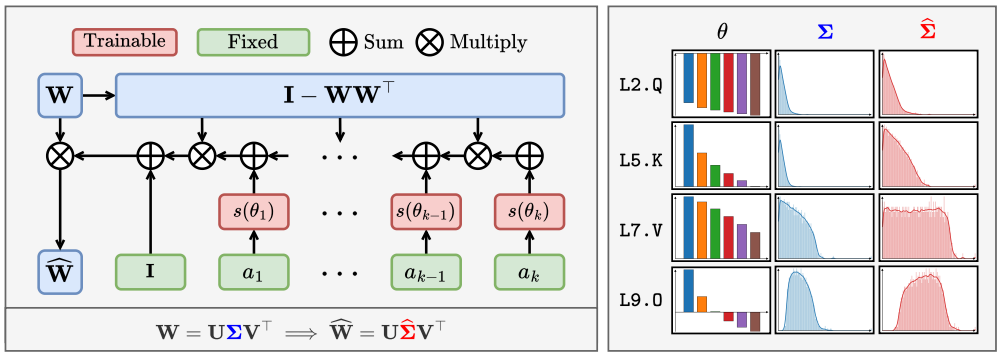


Figure 1: Left: Illustration of the LASR framework for learnable adaptive spectral reshaping, where a and θ denote the fixed and trainable coefficients respectively. $s(\cdot)$ denotes the sigmoid function and Σ denotes the all normalized singular values. More details are introduced in Section 3. Right: Different layers and parameters adaptively learn their preferred spectral reshaping functions.

ces participate in distinct functional pathways and exhibit different training behaviors. Indeed, our empirical analyses reveal substantial heterogeneity in spectral evolution across both layers and parameter types, with different weights favoring markedly different degrees and patterns of spectral amplification during training. These observations suggest that spectral reshaping should not merely be controllable, but adapt to the specific characteristics of each weight matrix.

Motivated by this perspective, we propose Learnable Adaptive Spectral Reparameterization (LASR), which recasts spectral reshaping as a weight-specific learning problem rather than a prescribed transformation. Figure 1 illustrates the computational workflow and effects of LASR. We define a principled feasible region, $\sigma_i \leq g(\sigma_i) \leq 1$, where σ_i denotes the normalized singular value, spanning the original spectrum to complete spectral equalization. To parameterize transformations within this region, we exploit the positive power-series expansion associated with the upper amplification envelope $1/\sigma_i$, truncated to K polynomial basis terms. Each term is modulated by a bounded learnable coefficient $\mathrm{sigmoid}(\theta_k)$, yielding a structured family of spectral mappings while ensuring bounded amplification and avoiding irregular coefficient cancellations. By learning these coefficients independently for each weight matrix under the language-modeling objective, LASR allows different weights to adapt their spectral transformations to their own training dynamics.

We extensively evaluate LASR across different model backbones and optimizers, covering both pretraining dynamics and zero-shot generalization on a diverse suite of downstream tasks. Across these settings, LASR consistently improves optimization efficiency, achieving approximately $1.4\times$ iteration reduction in reaching the same pretraining loss, while improving average zero-shot accuracy by approximately 2%. Beyond aggregate performance, we further analyze the learned spectral transformations across layers and parameter types, revealing pronounced heterogeneity in how different weights reshape their spectra. In particular, weight `Q` and `K` projections typically exhibit relatively mild spectral adaptation, whereas other projections undergo more active reshaping; similarly, shallower layers tend to remain more conservative, while deeper layers learn substantially stronger spectral transformations. These weight- and layer-dependent behaviors provide direct empirical evidence that different components of the model favor distinct spectral responses.

We summarize our contributions as follows:

- We identify spectral transformation flexibility as a missing dimension in existing spectral reparameterization methods, which largely rely on globally prescribed mappings despite substantial heterogeneity across weight matrices.
- We introduce LASR, a weight-specific learnable spectral reparameterization framework that learns bounded spectral transformations from the training objective through a structured positive polynomial parameterization.
- We provide extensive empirical evidence that adaptive spectral reshaping improves both optimization and generalization, and further uncover strong heterogeneity across parameter types and network depths, validating the need for weight-specific spectral adaptation..

2 Related Work

Weight Normalization and Spectral Control. Normalization is widely used to stabilize neural-network training, from activation-based methods such as BatchNorm (Ioffe & Szegedy, 2015), LayerNorm (Ba et al., 2016), and RMSNorm (Zhang & Sennrich, 2019) to parameter-space schemes including Weight Normalization and Weight Standardization (Salimans & Kingma, 2016; Huang et al., 2017; Qiao et al., 2019). Beyond local statistics or parameter magnitudes, matrix-level methods explicitly regulate spectral properties. Spectral-norm regularization and normalization constrain the largest singular value for generalization, robustness, and Lipschitz control (Yoshida & Miyato, 2017; Miyato et al., 2018; Gouk et al., 2021; Cisse et al., 2017), while singular-value bounding and orthogonality-based methods further shape the spectrum (Jia et al., 2017; Huang et al., 2018; Bansal et al., 2018; Li et al., 2019). Closely related studies on orthogonal initialization and dynamical isometry connect well-conditioned spectra to stable signal propagation and optimization (Saxe et al., 2013; Pennington et al., 2017; Xiao et al., 2018), with recent work extending spectral control to Transformer (Newhouse et al., 2026). These studies motivate regulating the full weight spectrum rather than only its scale, highlighting the need for efficient spectral transformations during training.

Polynomial Spectral Transformations. Matrix orthogonalization has long been studied through iterative polar-decomposition methods (Kovarik, 1970; Björck & Bowie, 1971; Higham & Schreiber, 1990; Kenney & Laub, 1992; Nakatsukasa & Higham, 2012; Nakatsukasa & Freund, 2016). Newton–Schulz iterations are particularly attractive for deep learning because they rely primarily on matrix multiplications. Muon applies such iterations to momentum matrices (Jordan et al., 2024), followed by increasingly efficient polynomial constructions: Polar Express studies optimal polynomial compositions (Amsel et al., 2026), optimized Newton–Schulz coefficients improve practical efficiency (Cesista et al.), CANS derives Chebyshev-type polynomials (Grishina et al., 2025), and IFNSO compresses iterative transformations into a unified polynomial optimized through a separate scalar approximation objective (Hu et al., 2026). Related norm-aware optimization and large-scale Muon studies further support this direction (Bernstein & Newhouse, 2024; Pethick et al., 2025; Liu et al., 2025). These methods can be viewed as polynomial spectral mappings, but predominantly target prescribed polar or near-orthogonal transformations that drive singular values toward unity. More broadly, classical preconditioning reshapes spectra to improve conditioning (Benzi, 2002; Chen, 2005), an idea recently transferred to neural-network weights. PC Layer performs low-degree polynomial spectrum shaping without enforcing exact orthogonality (Wang et al., 2026), while complementary methods study weight equilibration, attention conditioning, spectrum-preserving reparameterization, and explicit spectral constraints or clipping (Saratchandran et al., 2024; Saratchandran & Lucey, 2026; Qiu et al., 2026; Xie et al., 2026). Despite broadening spectral control beyond orthogonalization, existing spectral-shaping methods largely prescribe their mappings or constraints a priori (Grishina et al., 2025; Amsel et al., 2026), predefined spectral caps (Jiang et al., 2026), or predetermined shaping functions. In contrast, our Learnable Adaptive Spectral Reparameterization (LASR) jointly learns the spectral transformation with model parameters under the task objective, enabling weight-specific adaptive spectral shaping with efficient matrix-polynomial computation.

3 Methodology

3.1 Preliminary: Polynomial Preconditioning

Polynomial preconditioning is a fundamental technique in numerical linear algebra (Johnson et al., 1983) and has recently shown promising results in improving the spectral geometry of weight matrices in LLMs. In deep networks, polynomial preconditioning is typically applied through the Gram matrix, enabling direct reshaping of the singular-value spectrum of a weight matrix. Specifically, for $\mathbf{W} \in \mathbb{R}^{m\times n}$, the reparameterized weight can be written as $\widehat{\mathbf{W}} = p(\mathbf{W}\mathbf{W}^\top)\mathbf{W}$, or, equivalently, $\widehat{\mathbf{W}} = \mathbf{W}p(\mathbf{W}^\top\mathbf{W})$, where $p(\cdot)$ is a low-degree polynomial preconditioner. In practice, the formulation involving the smaller Gram matrix can be adopted for computational efficiency: $\mathbf{W}\mathbf{W}^\top$ when $m \le n$, and $\mathbf{W}^\top\mathbf{W}$ when $n < m$. We focus on polynomial preconditioners for which the induced spectral mapping $g(\sigma) = p(\sigma^2)\sigma$ is non-negative and non-decreasing over σ. These conditions ensure that the transformed singular values remain non-negative and preserve their original ordering, allowing the preconditioner to reshape the spectrum without altering the associated singular subspace. Its spectral effect can be characterized by the following proposition.

Proposition 1 (Wang et al., 2026). *For a matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ with $m \le n$, let its compact singular value decomposition be $\mathbf{W} = \mathbf{U}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_m)\mathbf{V}^\top$, where $\sigma_1 \ge \cdots \ge \sigma_m \ge 0$. Under the reparameterization $\widehat{\mathbf{W}} = p(\mathbf{W}\mathbf{W}^\top)\mathbf{W}$, the resulting weight admits the singular value decomposition $\widehat{\mathbf{W}} = \mathbf{U}\,\mathrm{diag}(\widehat{\sigma}_1, \ldots, \widehat{\sigma}_m)\,\mathbf{V}^\top$, where $\widehat{\sigma}_i = g(\sigma_i) = p(\sigma_i^2)\sigma_i$.*

Proposition 1 shows that polynomial preconditioning can be interpreted as a purely spectral transformation: it preserves the singular subspaces of $\mathbf{W}$ while rescaling each singular value according to the scalar mapping $g(\sigma) = p(\sigma^2)\sigma$. Hence, although the preconditioner operates on a matrix-valued parameter, its spectral effect is completely characterized by how $g(\cdot)$ reshapes the individual singular values. In particular, preferentially amplifying smaller singular values can reduce the spectral spread and improve the conditioning of the reparameterized weight. This observation reduces matrix-level preconditioner design to the construction of a scalar spectral mapping, providing a direct mechanism for reshaping the spectrum while preserving its underlying singular subspaces.

Existing Challenges and Motivation. Existing polynomial preconditioning methods typically prescribe a target spectral mapping and approximate it with a fixed low-degree polynomial, whose coefficients are determined beforehand through numerical procedures and remain unchanged throughout training (Huang et al., 2020; Grishina et al., 2025; Newhouse et al., 2026; Hu et al., 2026; Wang et al., 2026). While effective at improving conditioning, such a fixed transformation implicitly enforces a universal spectral preference across all weight matrices. Yet spectral structures can vary substantially across layers, parameter types, and training stages, making a uniform reshaping rule inherently inflexible. Consequently, the central challenge is no longer simply how to construct a polynomial that improves conditioning, but how to determine how strongly each individual spectrum should be reshaped while retaining sufficient freedom for the model to adapt. This motivates a transition from prescribed spectral transformations to learnable spectral reparameterization, allowing each weight matrix to adaptively determine its preferred spectral geometry during training.

3.2 Learnable Adaptive Spectral Reparameterization (LASR)

In this part, we elaborate on the rationale underlying our design. We begin by identifying the mathematical properties that an ideal learnable spectral transformation should satisfy, and then translate these principles into a concrete and practical formulation.

As discussed above, spectral reparameterization essentially reduces to constructing an effective transformation that reshapes the spectrum of a weight matrix $\mathbf{W}$. In the training, we usually estimate its spectral norm using a few steps of power iteration and denote the resulting normalization factor by $s(\mathbf{W}) \approx \sigma_{\max}(\mathbf{W}) = \|\mathbf{W}\|_2$. We then normalize the weight as $\widetilde{\mathbf{W}} = \mathbf{W}/s(\mathbf{W})$, such that its normalized singular values are approximately confined to $(0, 1]$. For notational simplicity, we henceforth use σ_i to denote the normalized singular value $\sigma_i/s(\mathbf{W})$. After performing spectral reparameterization in the normalized space, we restore the overall weight magnitude using the corresponding scalar factor $s(\mathbf{W})$ with stop-gradient.

Feasible Spectral Region. Previous approaches typically prescribe fixed spectral mappings that push normalized singular values toward unity. To retain sufficient flexibility for learning, we instead seek a relaxed spectral amplification mapping $g(\cdot)$ satisfying

$$\sigma_i \le g(\sigma_i) \le 1, \qquad \sigma_i \in (0, 1], \tag{1}$$

where the lower bound corresponds to preserving the original spectrum, while the upper bound corresponds to complete spectral equalization, yielding a (semi-)orthogonal structure when attained by all non-zero singular values. By end-to-end optimizing the mapping g, the model can adaptively learn the spectral amplification that is most favorable for each weight matrix. According to equation 1, the expected amplification p should satisfy $1 \le p\left(\sigma_i^2\right) \le \sigma_i^{-1}$.

Structured Learnable Coefficients. Having established the feasible range of spectral amplification, the remaining question is how to parameterize $p(\cdot)$ so that sufficient flexibility can be introduced without sacrificing well-behaved spectral reshaping. A straightforward choice is to directly learn the coefficients of a generic polynomial. However, unconstrained signed coefficients may introduce cancellations across different orders, leading to irregular or even oscillatory amplification patterns and making the contribution of individual coefficients strongly entangled. We therefore seek a structured parameterization based on a non-negative expansion, such that increasing any learnable component has a consistent directional effect on spectral amplification.

LASR Method. The above desiderata naturally lead to our Learnable Adaptive Spectral Reparameterization (LASR) method. The key observation is that the upper amplification envelope σ_i^{-1} admits a power-series expansion with strictly positive coefficients, providing exactly the structured basis desired above. Specifically, letting $r_i = 1 - \sigma_i^2 \in [0, 1)$, we have

$$\sigma_i^{-1} = (1 - r_i)^{-1/2} = 1 + \sum_{k=1}^{\infty} a_k r_i^k, \qquad a_k = \frac{\binom{2k}{k}}{4^k} > 0.$$

This positive expansion suggests a natural way to turn the fixed upper envelope into a learnable family of spectral mappings. We truncate the series to order K and modulate each positive basis term with a bounded learnable coefficient. Specifically, we define

$$p(\sigma_i^2) = 1 + \sum_{k=1}^{K} a_k c_k (1 - \sigma_i^2)^k, \qquad c_k = \mathrm{sigmoid}(\theta_k) \in (0, 1), \tag{2}$$

where θ_k is the learnable parameter and c_k controls the contribution of the k-th basis term to spectral amplification. K controls the approximation accuracy to the upper amplification envelope σ_i^{-1}, with larger K retaining more series terms and providing a tighter approximation. We further study the sensitivity to K in the ablation experiments and find that a small K is already sufficient in practice.

Properties of LASR. The structured parameterization of LASR naturally satisfies several desirable properties. Since $a_k > 0$, $c_k \in (0, 1)$, and $1 - \sigma_i^2 \geq 0$, the amplification factor is bounded by $1 \leq p(\sigma_i^2) \leq 1 + \sum_{k=1}^{K} a_k (1 - \sigma_i^2)^k \leq \sigma_i^{-1}$, and consequently the transformed singular value satisfies $\sigma_i \leq g(\sigma_i) = \sigma_i p(\sigma_i^2) \leq 1$, exactly respecting the feasible amplification range defined above. Moreover, $p(\sigma_i^2)$ decreases with σ_i, implying that smaller singular values naturally receive stronger multiplicative amplification, while the spectral boundary remains fixed since $p(1) = 1$. Meanwhile, each learnable coefficient has a consistent effect on the mapping, as $\partial p / \partial c_k = a_k (1 - \sigma_i^2)^k \geq 0$; increasing c_k therefore strengthens only the contribution of the corresponding basis term without introducing sign cancellation across different orders. Importantly, this parameterization constrains only how each coefficient affects spectral amplification, while the training objective remains free to change c_k through end-to-end optimization by the final cross-entropy loss.

According to equation 2, the scalar spectral mapping can be directly lifted to the matrix form. Let $\widetilde{\mathbf{W}} = \mathbf{W}/s(\mathbf{W})$ denote the normalized weight matrix. For the left-preconditioned form, we have

$$p\Big(\widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^\top\Big) = \mathbf{I} + \sum_{k=1}^{K} \mathrm{sigmoid}(\theta_k) \cdot \frac{\binom{2k}{k}}{4^k} \cdot \Big(\mathbf{I} - \widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^\top\Big)^k. \tag{3}$$

To stabilize the training (Wang et al., 2026), the final effective weight used for training is

$$\widehat{\mathbf{W}} = \gamma \, \mathrm{sg}\big(s(\mathbf{W})\big) \, p\Big(\widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^\top\Big) \, \widetilde{\mathbf{W}}, \tag{4}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient and γ is a learnable scalar initialized to 1, allowing the model to adapt the overall magnitude independently of the relative spectral reshaping.

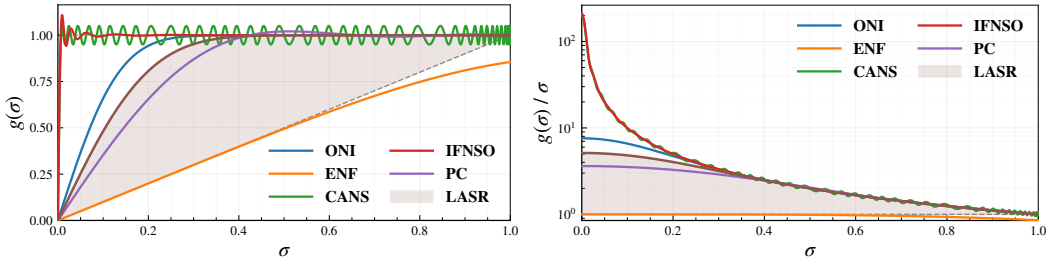
3.3 Comparison with Existing Spectral Shaping Methods

A growing line of work reshapes matrix spectra through polynomial transformations without explicitly computing the SVD. Early orthogonalization methods, such as ONI (Huang et al., 2020), use Newton–Schulz iterations to progressively drive singular values toward unity. CANS (Grishina et al., 2025) improves this process by optimizing polynomial coefficients under a prescribed orthogonalization objective, while IFNSO (Hu et al., 2026) compresses iterative Newton–Schulz transformations into a unified polynomial whose coefficients are optimized to approximate the unity mapping. Beyond orthogonalization, ENF (Newhouse et al., 2026) controls the upper end of the spectrum by approximating a predefined clipping function, whereas PC (Wang et al., 2026) employs fixed low-degree polynomials fitted to prescribed soft spectrum-shaping targets.

Despite their different formulations, these approaches largely share a common design principle: the desired spectral behavior is specified before model training, and the polynomial transformation is subsequently constructed or optimized to realize that target. Consequently, the resulting spectral shaping rule is not directly adapted to the preference of each individual weight during training.

Table 1: Comparison of recent advanced methods. $\mathbf{G}$ denotes the Gram matrix $\mathbf{W}\mathbf{W}^\top$ or $\mathbf{W}^\top\mathbf{W}$. The coefficients a_k are determined offline to approximate a prescribed spectral mapping g and remain fixed throughout model training. The MatMul cost is reported under the assumption $m \le n$.

Method	Spectral Region	Transformation $p(\mathbf{G})$	MatMul Cost	Learnable?
ONI	$g^{\circ T}(\sigma) \to 1$	$1.5\mathbf{I} - 0.5\mathbf{G}$	$2Tm^2n$	✗
CANS	$g(\sigma) \to 1 \pm \delta$	$\sum_{k=1}^{K} a_k \mathbf{G}^{k-1}$	Km^2n	✗
IFNSO	$g(\sigma) \to 1$	$\mathbf{I} + \sum_{k=1}^{K} a_k \left(\mathbf{I} - \mathbf{G}\right)^{2^{k-1}}$	$m^2n + Km^3$	✗
ENF	$g(\sigma) \le \sigma_{\text{th}}$	$(\mathbf{I} - a\mathbf{G}) + a\mathbf{G}(\mathbf{I} - a\mathbf{G})^3$	$4m^2n$	✗
PC	$g(\sigma) \to \min(\sigma/\sigma_{\text{th}}, 1)$	$\sum_{k=0}^{K} a_k \mathbf{G}^k$	Km^2n	✗
LASR	$\sigma \le g(\sigma) \le 1$	$\mathbf{I} + \sum_{k=1}^{K} \text{sigmoid}(\theta_k) \cdot a_k \left(\mathbf{I} - \mathbf{G}\right)^k$	Km^2n	✓



(a) Transformed Spectrum $g(\sigma)$.

(b) Amplification Ratio $g(\sigma) \,/\, \sigma$.

Figure 2: Comparison of the spectral amplification induced by different reparameterization methods.

LASR takes a different perspective. Rather than prescribing a specific target mapping, we specify only a feasible spectral region, $\sigma_i \le g(\sigma_i) \le 1$, and expose the mapping within this region to end-to-end learning. The positive-series parameterization further guarantees bounded amplification while allowing each weight matrix to learn its own spectral response through the task objective. Table 1 summarizes the key distinctions. To further illustrate these differences, Figure 2 visualizes the spectral mappings of different methods. CANS and IFNSO approach more aggressively drive singular values toward unity, whereas other methods retain certain spectral variations during reshaping. In contrast, LASR does not prescribe a fixed amplification profile, but adaptively learns an optimal spectral response **within the feasible region** shown in Figure (2) through end-to-end optimization. This provides greater flexibility in heterogeneous requirements of different model parameters.

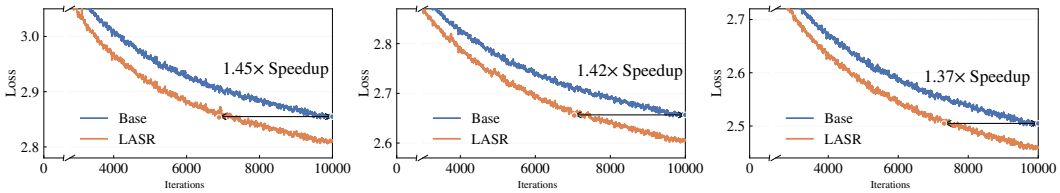
4 EXPERIMENTS

In this section, we comprehensively evaluate LASR in terms of zero-shot generalization and domain-specific capabilities after pretraining. We further analyze its learned spectral preferences across layers and parameters, study the effect of applying spectral scaling at different locations, and examine its stability and sensitivity under different setups. More details are stated in Appendix.

Setups and Hyperparameters. In experiments, we use Megatron-LM (Shoeybi et al., 2019) as the training framework and pretrain all models on the C4-en dataset (Raffel et al., 2020). We adopt LLaMA3 (Grattafiori et al., 2024) and OLMoE (Muennighoff et al., 2025) as the dense and MoE backbones, respectively, and proportionally scale their architectural configurations to obtain models of different sizes. We adopt a global batch size of 1024 for 350M and 2048 for others, a sequence length of 2048, and a warm-up budget of 1B tokens. We evaluate on both AdamW and Muon optimizers. The 350M models are pretrained for approximately 21B tokens, with a maximum learning rate of 5e-4 for AdamW and 1e-3 for Muon. For Muon, we enable Nesterov momentum, use 5 inner iterations, and set the scale factor to 0.2. Other models are pretrained for approximately 42B tokens with a maximum learning rate of 2.5e-4 for AdamW and 5e-4 for Muon. The learning rate schedule follows WSM (Tian et al., 2026), maintaining a constant after warm-up. Following its merging principle, we randomly select 5 checkpoints from the final stage and average them as output.

Table 2: Zero-shot generalization on the LLaMA-3 and OLMoE models. We report the mean ± standard deviation of accuracy. Bold indicates the best accuracy in each column.

	MMLU	ARC-E	BoolQ	HellaS.	PIQA	SCIQ	WinoG.	Avg.
LLaMA3 - 350M								
BASE	23.00$_{\pm 0.03}$	46.14$_{\pm 0.12}$	53.26$_{\pm 0.18}$	31.83$_{\pm 0.14}$	66.78$_{\pm 0.10}$	71.18$_{\pm 0.20}$	49.54$_{\pm 0.08}$	48.82$_{\pm 0.03}$
ONI	22.96$_{\pm 0.02}$	46.27$_{\pm 0.08}$	51.94$_{\pm 0.14}$	32.31$_{\pm 0.13}$	66.62$_{\pm 0.17}$	71.82$_{\pm 0.21}$	50.16$_{\pm 0.04}$	48.87$_{\pm 0.04}$
ENF	22.93$_{\pm 0.03}$	46.29$_{\pm 0.12}$	53.16$_{\pm 0.16}$	32.03$_{\pm 0.18}$	67.33$_{\pm 0.14}$	72.54$_{\pm 0.13}$	50.05$_{\pm 0.10}$	49.19$_{\pm 0.05}$
CANS	**23.01**$_{\pm 0.04}$	46.01$_{\pm 0.16}$	53.04$_{\pm 0.11}$	32.94$_{\pm 0.23}$	66.19$_{\pm 0.13}$	73.16$_{\pm 0.15}$	51.14$_{\pm 0.17}$	49.36$_{\pm 0.08}$
IFNSO	22.98$_{\pm 0.06}$	46.25$_{\pm 0.12}$	52.87$_{\pm 0.21}$	33.10$_{\pm 0.15}$	67.08$_{\pm 0.16}$	72.88$_{\pm 0.14}$	50.88$_{\pm 0.20}$	49.43$_{\pm 0.04}$
PC	22.97$_{\pm 0.03}$	47.77$_{\pm 0.13}$	51.89$_{\pm 0.24}$	33.20$_{\pm 0.29}$	**68.39**$_{\pm 0.21}$	73.00$_{\pm 0.15}$	**51.62**$_{\pm 0.22}$	49.83$_{\pm 0.03}$
LASR	22.93$_{\pm 0.02}$	**48.23**$_{\pm 0.22}$	**56.22**$_{\pm 0.10}$	**33.51**$_{\pm 0.27}$	68.30$_{\pm 0.19}$	**75.17**$_{\pm 0.14}$	51.23$_{\pm 0.13}$	**50.80**$_{\pm 0.05}$
LLaMA3 - 1.3B								
BASE	23.10$_{\pm 0.03}$	50.59$_{\pm 0.09}$	53.55$_{\pm 0.11}$	36.52$_{\pm 0.09}$	69.04$_{\pm 0.15}$	75.07$_{\pm 0.12}$	54.04$_{\pm 0.10}$	51.70$_{\pm 0.04}$
ONI	23.01$_{\pm 0.06}$	50.84$_{\pm 0.17}$	53.27$_{\pm 0.13}$	36.58$_{\pm 0.22}$	68.55$_{\pm 0.25}$	75.62$_{\pm 0.21}$	53.78$_{\pm 0.10}$	51.66$_{\pm 0.08}$
ENF	23.05$_{\pm 0.03}$	51.13$_{\pm 0.19}$	53.65$_{\pm 0.14}$	36.47$_{\pm 0.08}$	69.12$_{\pm 0.12}$	75.79$_{\pm 0.12}$	54.11$_{\pm 0.07}$	51.90$_{\pm 0.02}$
CANS	23.07$_{\pm 0.06}$	52.66$_{\pm 0.08}$	52.34$_{\pm 0.09}$	37.18$_{\pm 0.16}$	70.02$_{\pm 0.15}$	76.83$_{\pm 0.11}$	53.26$_{\pm 0.12}$	52.19$_{\pm 0.04}$
IFNSO	23.03$_{\pm 0.02}$	52.95$_{\pm 0.11}$	52.11$_{\pm 0.11}$	37.02$_{\pm 0.16}$	70.45$_{\pm 0.19}$	77.32$_{\pm 0.12}$	53.59$_{\pm 0.08}$	52.35$_{\pm 0.05}$
PC	**23.13**$_{\pm 0.04}$	53.32$_{\pm 0.13}$	53.12$_{\pm 0.17}$	37.06$_{\pm 0.13}$	**70.62**$_{\pm 0.07}$	77.16$_{\pm 0.12}$	54.06$_{\pm 0.08}$	52.64$_{\pm 0.05}$
LASR	22.97$_{\pm 0.05}$	**54.47**$_{\pm 0.13}$	**57.24**$_{\pm 0.11}$	**37.58**$_{\pm 0.13}$	69.36$_{\pm 0.15}$	**78.01**$_{\pm 0.16}$	**54.22**$_{\pm 0.10}$	**53.41**$_{\pm 0.06}$
OLMoE - 7B - A1B								
BASE	23.08$_{\pm 0.05}$	55.47$_{\pm 0.13}$	48.81$_{\pm 0.16}$	41.26$_{\pm 0.18}$	72.09$_{\pm 0.22}$	78.62$_{\pm 0.16}$	53.67$_{\pm 0.19}$	53.29$_{\pm 0.11}$
ONI	23.06$_{\pm 0.07}$	54.85$_{\pm 0.22}$	51.67$_{\pm 0.25}$	41.04$_{\pm 0.17}$	72.17$_{\pm 0.23}$	77.95$_{\pm 0.15}$	54.26$_{\pm 0.14}$	53.57$_{\pm 0.10}$
ENF	23.12$_{\pm 0.11}$	54.93$_{\pm 0.13}$	52.54$_{\pm 0.11}$	41.15$_{\pm 0.15}$	72.26$_{\pm 0.20}$	78.16$_{\pm 0.21}$	54.19$_{\pm 0.11}$	53.76$_{\pm 0.10}$
CANS	23.16$_{\pm 0.03}$	54.92$_{\pm 0.12}$	53.31$_{\pm 0.14}$	40.92$_{\pm 0.12}$	72.47$_{\pm 0.19}$	79.51$_{\pm 0.18}$	53.17$_{\pm 0.15}$	53.92$_{\pm 0.05}$
IFNSO	23.02$_{\pm 0.06}$	54.29$_{\pm 0.11}$	53.27$_{\pm 0.18}$	41.03$_{\pm 0.18}$	**72.80**$_{\pm 0.14}$	78.97$_{\pm 0.24}$	53.28$_{\pm 0.28}$	53.81$_{\pm 0.09}$
PC	23.14$_{\pm 0.08}$	**55.77**$_{\pm 0.14}$	53.73$_{\pm 0.15}$	41.39$_{\pm 0.13}$	72.20$_{\pm 0.19}$	77.73$_{\pm 0.23}$	**54.85**$_{\pm 0.29}$	54.12$_{\pm 0.07}$
LASR	**23.19**$_{\pm 0.04}$	55.30$_{\pm 0.15}$	**57.49**$_{\pm 0.21}$	**41.64**$_{\pm 0.14}$	72.49$_{\pm 0.13}$	**79.85**$_{\pm 0.23}$	53.43$_{\pm 0.09}$	**54.77**$_{\pm 0.06}$



(a) LLaMA3 - 350M (b) LLaMA3 - 1.3B (c) OLMoE - 7B - A1B

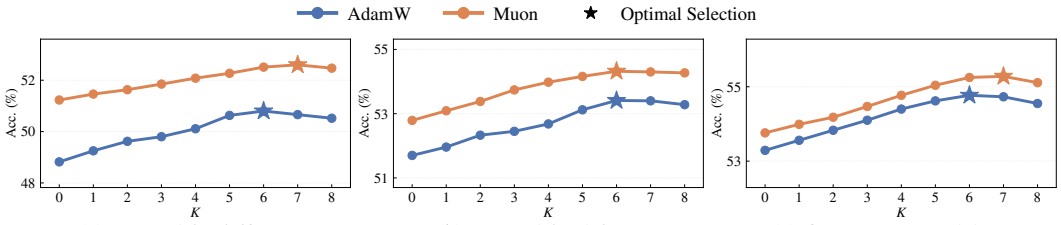
Figure 3: Training loss acceleration on different models with the LASR method.

Downstream Generalization. Table 2 reports the zero-shot performance across different model scales and architectures. LASR consistently achieves the best average accuracy across all three settings, improving the base models from 48.82% to 50.80%, 51.70% to 53.41%, and 53.29% to 54.77% on LLaMA3-350M, LLaMA3-1.3B, and OLMoE-7B-A1B, respectively. The gains are particularly clear on BoolQ, where LASR improves the base model by 2.96%, 3.69%, and 8.68%, while consistent improvements are also observed on SciQ and HellaSwag. Overall, LASR achieves the strongest aggregate performance across downstream evaluations on both backbones and different scales. Due to space constraints, the corresponding Muon results are reported in Appendix C.2, where similar trends are observed across all three experimental settings.

Training Acceleration. We further evaluate the training efficiency of LASR across different model scales and architectures. As shown in Fig. 3, LASR consistently exhibits faster loss reduction than the standard training baseline throughout pretraining. At the same training iteration, LASR maintains a noticeably lower loss, indicating more efficient optimization progress. Equivalently, it requires substantially fewer training iterations to reach a comparable loss level achieved by the baseline. Across all three model settings, this results in an approximately 1.4× convergence speedup, demonstrating that LASR can consistently accelerate optimization across different model scales and architectures.

Table 3: Breakdown of the runtime profiles within the Weight Transformer (WT) module for different methods, where all reported times are in milliseconds (ms). WT denotes the total weight-transformation time, and Ratio denotes the fraction of each component in the total WT time. Power, Fwd, Proxy, GradAcc, and VJP denote power-iteration, forward weight transformation, proxy construction, effective-gradient accumulation, and vector–Jacobian-product computation, respectively.

	WT	Power	Ratio	Fwd	Ratio	Proxy	Ratio	GradAcc	Ratio	VJP	Ratio
ONI	1123.1	5.0	0.4%	299.2	26.6%	7.6	0.7%	136.3	12.1%	675.0	60.1%
ENF	678.7	5.5	0.8%	165.5	24.4%	8.5	1.3%	143.7	21.2%	355.5	52.4%
CANS	1407.0	4.9	0.3%	383.9	27.3%	7.9	0.6%	135.2	9.6%	875.2	62.2%
IFNSO	1328.8	5.3	0.4%	362.6	27.3%	8.2	0.6%	150.3	11.3%	802.4	60.4%
PC	1143.8	432.1	37.8%	184.5	16.1%	8.1	0.7%	138.7	12.1%	380.5	33.3%
LASR	1283.9	457.5	35.6%	216.9	16.9%	8.5	0.7%	140.1	10.9%	460.9	35.9%



(a) LLaMA3 - 350M (b) LLaMA3 - 1.3B (c) OLMoE - 7B - A1B

Figure 4: Sensitivity to the Polynomial Degree K in LASR. We test on both dense and MoE models with the AdamW and Muon optimizer. $K = 0$ represents the vanilla base training.

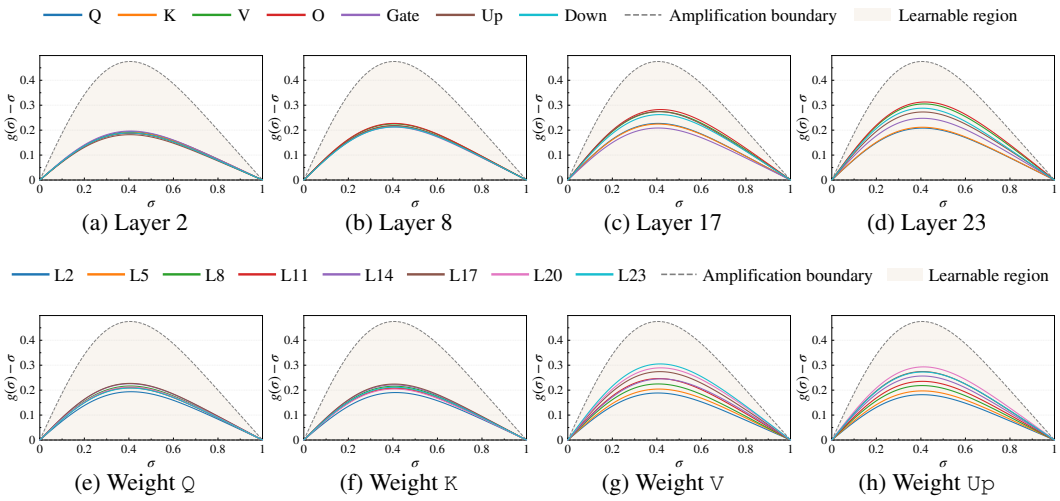
Training Profiles. Table 3 shows the runtime profile of training the 350M model. The spectral methods exhibit noticeably different cost distributions within the Weight Transformer module. For ONI, ENF, CANS, and IFNSO, most of the WT cost comes from VJP computation, which accounts for 52.4%–62.2% of the runtime, while the forward transformation contributes another 24.4%–27.3%. By contrast, PC and LASR additionally perform power-iteration-based spectral-norm estimation, which shifts part of the cost toward power iteration and leads to a more balanced runtime composition. Despite the learnable transformation introduced by LASR, its profile remains close to that of PC: power iteration and VJP together account for 71.5% of the WT runtime in LASR, compared with 71.1% in PC, while the remaining components contribute only modest differences.

Sensitivity to the Degree K. We report the sensitivity to the polynomial degree K across different model scales and optimizers in Figure 4. The degree K controls the flexibility of the learnable spectral transformation: higher-order polynomial bases progressively extend its influence toward smaller singular values while leaving the medium and large spectrum less affected. Increasing K therefore initially improves spectral adaptability and performance. Beyond a moderate degree, however, the useful spectral region is largely covered, and additional flexibility mainly affects very small singular values, where low-energy or noisy directions may be amplified. As a result, performance tends to saturate and may slightly degrade for large K. This trend is consistent across model and optimizer settings, suggesting that a moderate degree provides a robust trade-off. We therefore use $K = 6$ by default. Further analysis of the learnable spectral region is provided in Appendix C.3.

Learned Spectral Amplification. Figure 5 visualizes the spectral amplification patterns learned by LASR across different layers and weights. The learned patterns indicates that different weights require distinct forms of spectral shaping. From a layer-wise perspective, shallow layers show relatively consistent amplification patterns across weight types, whereas the differences become increasingly pronounced in deeper layers, particularly near the end of the model. From a weight-wise perspective, the patterns learned for weights `Q` and `K` remain largely consistent across layers, suggesting relatively stable spectral requirements for attention-score formation. In contrast, weights `V`, `O`, and MLP projections `Gate`, `Up`, and `Down` exhibit much stronger layer-wise variation, consistent with their layer-dependent roles in feature transformation and representation refinement. These observations underscore the necessity of adaptive spectral shaping. Due to space limitations, we show more results of training dynamics of θ and spectral comparisons in Appendix C.1 C.4, and C.5.

Table 4: Ablations on reparameterization flexibility of layer-wise and weight-wise patterns. $--$ (the first line) denotes the case of directly adopting the initialization of θ without changing. Loss is the averaged loss of last 10 iterations and Mean Acc. is the averaged accuracy on downstream tasks.

Reparameterization Pattern		LLaMA3 - 350M		LLaMA3 - 1.3B		OLMoE - 7B - A1B	
Layer-shared	**Weight-shared**	**Loss**	**Mean Acc.**	**Loss**	**Mean Acc.**	**Loss**	**Mean Acc.**
-	-	2.834	49.33	2.635	52.17	2.477	53.68
✓	✓	2.829	49.64	2.631	52.45	2.475	53.95
✓	✗	2.819	50.09	2.624	52.86	2.463	54.12
✗	✓	2.820	50.15	2.621	52.82	2.469	54.33
✗	✗	2.809	50.78	2.613	53.37	2.452	54.70



(a) Layer 2 (b) Layer 8 (c) Layer 17 (d) Layer 23

(e) Weight `Q` (f) Weight `K` (g) Weight `V` (h) Weight `Up`

Figure 5: Learned amplification patterns across layers and weight types on the 350M model. (a–d) compare different weights within each layer, while (e–h) compare each weight type across layers. The learned amplification patterns exhibit substantial variation across both layers and weight types.

Ablation on Different Patterns. In Table 4, we compare five settings: using the initialized mapping without learning, a single mapping shared by all weights, mappings shared within each layer, mappings shared across the same weight type, and fully weight-specific mappings. Learning the spectral transformation consistently improves over the fixed initialization, confirming that the prescribed starting point can be further adapted to the training objective. Beyond this, finer-grained parameterization brings additional gains: relaxing either layer-wise or type-wise sharing improves performance, while fully weight-specific mappings achieve the strongest results across all model scales. These observations indicate that the preferred spectral reshaping varies substantially across both layers and parameter types, thereby enabling stronger local adaptivity.

5 Conclusion

In this work, we introduced LASR, a learnable adaptive spectral reparameterization framework that enables individual weight matrices to learn their preferred spectral transformations during training. By replacing prescribed spectral mappings with weight-specific learnable transformations, LASR provides greater flexibility in shaping spectral geometry. Experiments across different model scales and architectures show consistent improvements over standard training and existing spectral methods, while analysis reveals distinct spectral behaviors across layers and parameter types. Moreover, LASR achieves these gains with runtime overhead comparable to existing spectral reparameterization approaches. Overall, our results highlight adaptive spectral shaping as a promising direction for improving both optimization and generalization efficiency for the large-scale models.

REFERENCES

Noah Amsel, David Persson, Christopher Musco, and Robert M Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. In *International Conference on Learning Representations*, volume 2026, pp. 138323–138360, 2026.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? *Advances in neural information processing systems*, 31, 2018.

Michele Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics*, 182(2):418–477, 2002.

Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning. *arXiv preprint arXiv:2410.21265*, 2024.

Åke Björck and Clazett Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971.

Franz Louis Cesista, Jiacheng You, and Keller Jordan. Squeezing 1–2% efficiency gains out of muon by optimizing the newton-schulz coefficients, 2025. *URL http://leloykun. github. io/ponder/muon-opt-coeffs/. Cited on*, pp. 24.

Ke Chen. Matrix preconditioning techniques and applications. *Cambridge monographs on applied and computational mathematics*, 19, 2005.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International conference on machine learning*, pp. 854–863. PMLR, 2017.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ekaterina Grishina, Matvey Smirnov, and Maxim Rakhuba. Accelerating newton-schulz iteration for orthogonalization via chebyshev-type polynomials. *arXiv preprint arXiv:2506.10935*, 2025.

Nicholas J Higham and Robert S Schreiber. Fast polar decomposition of an arbitrary matrix. *SIAM Journal on Scientific and Statistical Computing*, 11(4):648–655, 1990.

Chen Hu, Qianxi Zhao, Xiaochen Yuan, Hong Zhang, Ding Yuan, Yanbin Wu, and Xiying Li. Ifnso: Iteration-free newton-schulz orthogonalization. *arXiv preprint arXiv:2602.02500*, 2026.

Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2822–2830. IEEE, 2017.

Lei Huang, Xianglong Liu, Bo Lang, Adams Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Lei Huang, Li Liu, Fan Zhu, Diwen Wan, Zehuan Yuan, Bo Li, and Ling Shao. Controllable orthogonalization in training dnns. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6428–6437. IEEE, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Kui Jia, Dacheng Tao, Shenghua Gao, and Xiangmin Xu. Improving training of deep neural networks via singular value bounding. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3994–4002. IEEE, 2017.

Xiaowen Jiang, Andrei Semenov, and Sebastian U Stich. Enhancing llm training via spectral clipping. *arXiv preprint arXiv:2603.14315*, 2026.

Olin G Johnson, Charles A Micchelli, and George Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM Journal on Numerical Analysis*, 20(2):362–376, 1983.

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*, 6(3):4, 2024.

Charles Kenney and Alan J Laub. On scaling newton's method for polar decomposition and the matrix sign function. *SIAM Journal on Matrix Analysis and Applications*, 13(3):688–706, 1992.

Zdislav Kovarik. Some iterative methods for improving orthonormality. *SIAM Journal on Numerical Analysis*, 7(3):386–389, 1970.

Shuai Li, Kui Jia, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. ngpt: Normalized transformer with representation learning on the hypersphere. In *International Conference on Learning Representations*, volume 2025, pp. 74014–74038, 2025.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. In *International Conference on Learning Representations*, volume 2025, pp. 62061–62121, 2025.

Yuji Nakatsukasa and Roland W Freund. Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of zolotarev's functions. *siam REVIEW*, 58(3):461–493, 2016.

Yuji Nakatsukasa and Nicholas J Higham. Backward stability of iterations for computing the polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 33(2):460–479, 2012.

Laker Newhouse, Preston Hess, Franz Louis Cesista, Andrii Zahorodnii, Jeremy Bernstein, and Phillip Isola. Training transformers with enforced lipschitz bounds, 2026. URL `https://openreview.net/forum?id=GKr4GKrn6o`.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30, 2017.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

Zeju Qiu, Simon Buchholz, Tim Xiao, Maximilian Dax, Bernhard Schölkopf, and Weiyang Liu. Reparameterized llm training via orthogonal equivalence transformation. *Advances in Neural Information Processing Systems*, 38:140775–140821, 2026.

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.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Hemanth Saratchandran and Simon Lucey. Spectral conditioning of attention improves transformer performance. *Advances in Neural Information Processing Systems*, 38:59223–59256, 2026.

Hemanth Saratchandran, Thomas X Wang, and Simon Lucey. Weight conditioning for smooth optimization of neural networks. In *European Conference on Computer Vision*, pp. 310–325. Springer, 2024.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

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

Changxin Tian, Jiapeng Wang, Qian Zhao, Kunlong Chen, Jia Liu, Ziqi Liu, Jiaxin Mao, Xin Zhao, Zhiqiang Zhang, and Jun Zhou. Wsm: decay-free learning rate schedule via checkpoint merging for llm pre-training. In *International Conference on Learning Representations*, volume 2026, pp. 134767–134795, 2026.

Senmiao Wang, Tiantian Fang, Haoran Zhang, Yushun Zhang, Kunxiang Zhao, Alex Schwing, and Ruoyu Sun. Pc layer: Polynomial weight preconditioning for improving llm pre-training. *arXiv preprint arXiv:2606.06470*, 2026.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International conference on machine learning*, pp. 5393–5402. PMLR, 2018.

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, et al. Controlled llm training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026.

Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

A AI USE STATEMENT

AI tools were used only to improve sentence clarity, check spelling and grammar, and assist with debugging code. They were not used to generate ideas, conduct analysis, interpret results, or produce other substantive parts of the work. All other aspects of the work were completed by the authors.

B REPRODUCIBILITY DETAILS

We provide detailed experimental settings to facilitate reproducibility. Table 5 summarizes the detailed architectural configurations of all model scales.

Table 5: Model Hyperparameters.

	Experts	Layers	Attention heads	Embedding dim	Hidden size
LLaMA3 - 350M	1 / 1	24	16	1024	2384
LLaMA3 - 1.3B	1 / 1	24	16	2048	5504
OLMoE - 7B - A1B	8 / 64	16	16	2048	1024

Backbone Details. All experiments are conducted in BF16. We use rotary positional embeddings (Su et al., 2024), RMSNorm (Zhang & Sennrich, 2019), SwiGLU activations (Shazeer, 2020), and disable dropout during pretraining. Unless otherwise specified, all methods share the same model architecture, data pipeline, training budget, optimizer, and learning-rate schedule to ensure a controlled comparison.

Training Setups and Hyperparameters. All models are pretrained with Megatron-LM (Shoeybi et al., 2019) on C4-en (Raffel et al., 2020), using LLaMA3 (Grattafiori et al., 2024) and OLMoE (Muennighoff et al., 2025) as the dense and MoE backbones. Across all experiments, we use a global batch size of 1024 for the 350M model and 2048 for others, a sequence length of 2048, and approximately 1B warm-up tokens. The 350M models are trained for about 21B tokens with maximum learning rates of 5×10^{-4} for AdamW and 1×10^{-3} for Muon, while the remaining models are trained for about 42B tokens with learning rates of 2.5×10^{-4} and 5×10^{-4}, respectively. For Muon, we use Nesterov momentum, 5 inner iterations, and a scale factor of 0.2. Following WSM (Tian et al., 2026), the learning rate is kept constant after warm-up, and the final model is obtained by averaging five randomly selected checkpoints from the final training stage.

LASR Configuration. Unless otherwise specified, we apply LASR to the selected attention and MLP projection matrices in each Transformer block. We use $K = 6$ as the default polynomial order. Following PC Layer, the largest singular value is estimated using 10 steps of power iteration. To account for the approximation error of power iteration, we scale the estimated spectral norm by a safety factor of 1.02 before normalization, which helps keep the normalized spectrum within the interval $[0, 1]$. All learnable spectral parameters are jointly optimized with the model parameters during pretraining, and θ is initialized to 0.

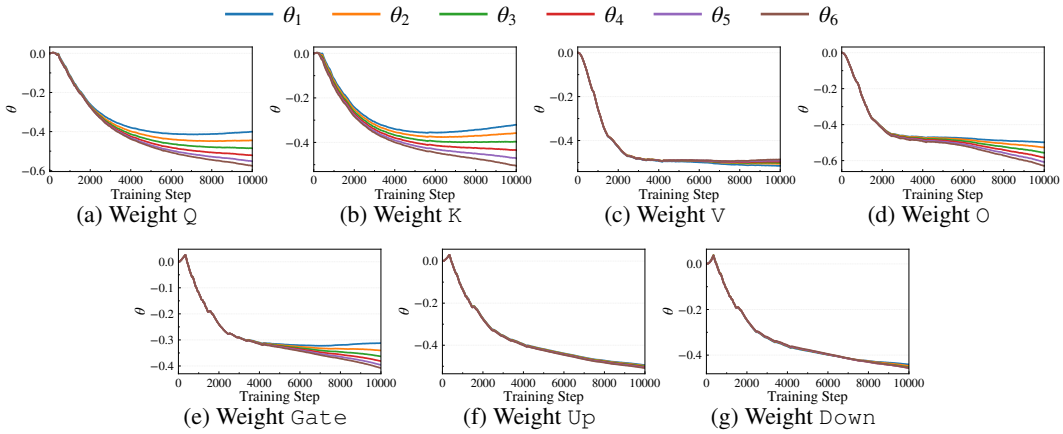
Hardware and Software Environment. Experiments are conducted on NVIDIA A100 80GB GPUs under Ubuntu 22.04 and CUDA 12.9, using PyTorch 2.11, Megatron-LM 0.20.0, and Transformer Engine 2.17.1. We use data parallelism for the reported dense-model experiments. Pretraining a single LLaMA3-350M, LLaMA3-1.3B, and OLMoE-7B-A1B model requires approximately 100, 2000, and 2400 GPU hours, respectively.

Evaluation. We evaluate downstream performance using the LM Evaluation Harness (Gao et al., 2024). For each benchmark, we fix the random seed to ensure consistent evaluation conditions across methods. To quantify evaluation variability, we evaluate three different output models trained from different random seeds for each configuration and report the mean and standard deviation across the resulting scores.

C Additional Experiments

C.1 Training Dynamics of θ

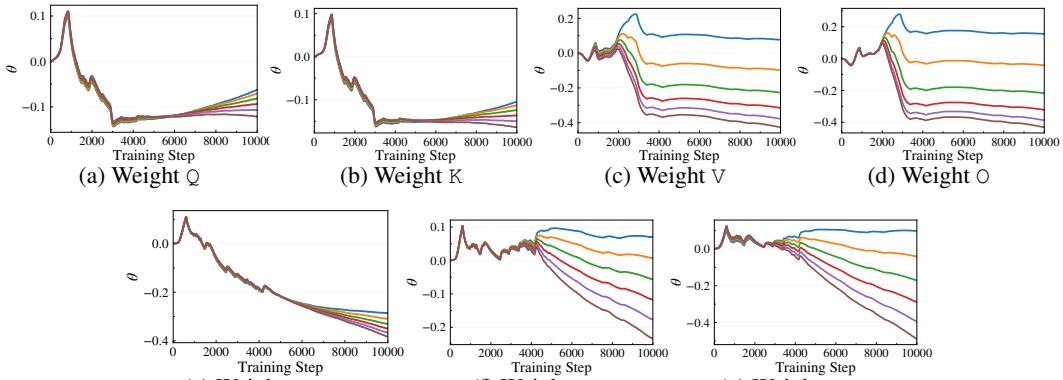
To further investigate the adaptive behavior of LASR, we visualize the evolution of θ throughout training across different layers and parameter groups. These trajectories provide a direct view of how the spectral reparameterization function is progressively adjusted during optimization.



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

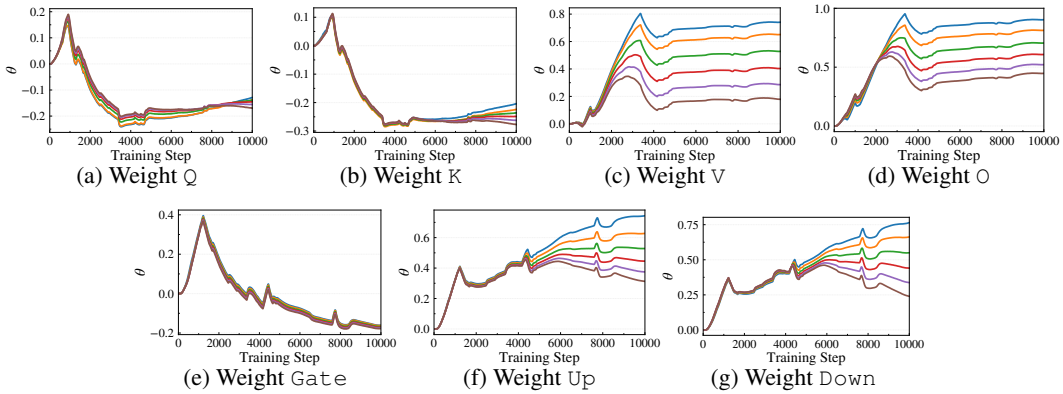
Figure 6: θ training curves on layer-2 of LLaMA3-350M.



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

Figure 7: θ training curves on layer-12 of LLaMA3-350M.



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

Figure 8: θ training curves on layer-22 of LLaMA3-350M.

Figures 6 7 and 8 show the evolution of the learnable coefficients and reveal clear layer- and weight-dependent specialization in the spectral reparameterization. At the early stage of training, the six coefficients often evolve in a highly synchronized manner, indicating that different spectral orders initially share a similar optimization direction. As training proceeds, however, distinct order-dependent trajectories gradually emerge. This differentiation is particularly evident for the query and key projections in shallow layers and for several attention projections in intermediate layers, where the coefficients progressively separate and form stable ordered patterns. In contrast, some weights maintain largely overlapping trajectories throughout training, suggesting that a relatively uniform spectral adjustment is sufficient for these components.

More importantly, the timing of this specialization also varies across layers and parameter types. Some projections establish their relative coefficient structure relatively early and remain stable afterwards, indicating that their characteristic spectral transformation is determined during the first half of training. Other parameters, especially several MLP projections in intermediate and deep layers, continue to separate even in the later stages of optimization, showing that they preserve substantial spectral flexibility over a much longer training horizon. These heterogeneous dynamics suggest that different network components not only prefer different spectral transformations, but also adapt them at different rates. This further supports the layer-wise and weight-wise flexibility of LASR, allowing each component to independently determine both the form and the training dynamics of its spectral reparameterization.

C.2 More Experiments on Muon

Table 6: Zero-shot generalization on the LLaMA-3 and MoE models with Muon.

	MMLU	ARC-E	BoolQ	HellaS.	PIQA	SCIQ	WinoG.	Avg.
LLaMA3 - 350M								
Base	23.03	47.55	59.24	33.86	67.45	74.20	53.25	51.23
Oni	23.05	47.52	58.21	33.86	67.83	74.90	52.42	51.11
Enf	22.98	47.82	58.50	33.35	67.37	74.80	52.73	51.08
Cans	23.04	47.14	58.52	33.85	66.96	75.50	53.54	51.22
Ifnso	23.02	47.85	57.33	**34.55**	67.39	75.40	53.26	51.26
Pc	23.00	48.55	57.34	34.04	**68.82**	77.10	52.66	51.64
Lasr	**23.08**	**49.15**	**61.09**	34.54	68.65	**77.30**	**53.78**	**52.51**
LLaMA3 - 1.3B								
Base	23.04	54.93	48.20	40.80	71.71	77.80	53.02	52.79
Oni	23.03	54.36	51.11	40.61	71.76	77.10	53.56	53.08
Enf	23.07	54.42	51.96	40.70	71.90	77.30	53.50	53.26
Cans	23.13	54.35	52.76	40.52	72.04	78.70	52.41	53.42
Ifnso	22.98	53.83	52.67	40.61	**72.45**	78.20	52.54	53.33
Pc	23.11	**55.33**	53.10	41.00	71.83	76.80	**54.13**	53.61
Lasr	**23.17**	54.87	**56.95**	**41.26**	72.16	**79.10**	52.74	**54.32**
OLMoE - 7B - A1B								
Base	23.06	56.65	47.25	43.18	73.31	79.70	53.20	53.76
Oni	23.08	56.20	50.75	42.90	73.58	78.50	54.05	54.15
Enf	23.14	56.26	51.72	43.10	73.50	78.80	53.82	54.33
Cans	23.20	55.70	53.52	42.55	73.34	**80.30**	52.75	54.48
Ifnso	23.01	54.72	53.55	42.77	74.00	79.30	52.78	54.30
Pc	23.15	**56.85**	53.65	43.22	72.85	77.60	**54.78**	54.59
Lasr	**23.30**	56.45	**56.32**	**43.50**	**74.85**	79.30	53.02	**55.25**

Table 6 further reports the zero-shot generalization results when using Muon as the optimizer. Overall, Lasr consistently achieves the best average performance across all three model settings, improving the corresponding base models by 1.28%, 1.53%, and 1.49% on LLaMA3-350M, LLaMA3-1.3B, and OLMoE-7B-A1B, respectively. Similar improvements can also be observed on several individual benchmarks, particularly BoolQ and HellaSwag, although the best performance on some tasks is achieved by other spectral reparameterization methods. These results suggest that the effectiveness of Lasr is not restricted to a specific optimizer and remains consistent when the models are trained with Muon. Together with the results using AdamW, this provides additional evidence for the robustness of the proposed learnable spectral reparameterization across different optimization settings.

C.3 Effect of the Polynomial Degree

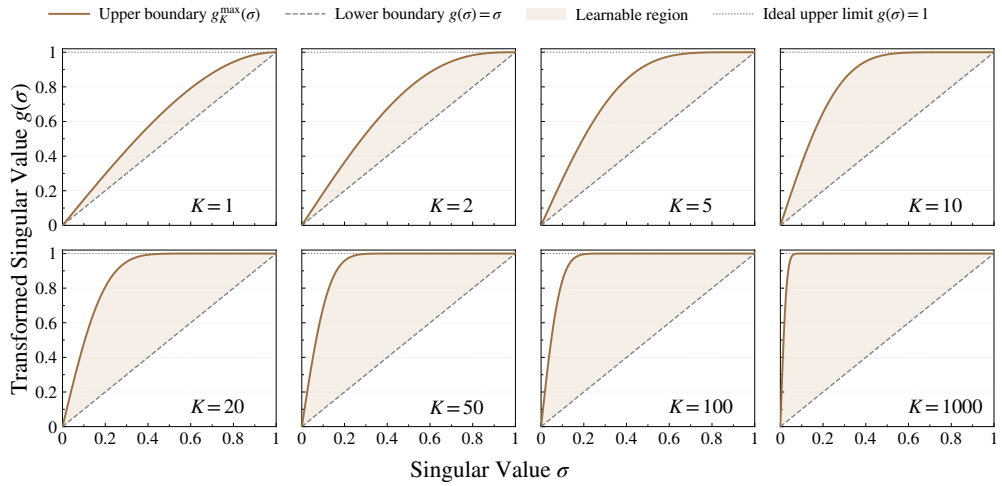


Figure 9: The final impact of different degree K in LASR.

We study how the polynomial degree K affects the spectral transformation in LASR. Recall that

$$g_K(\sigma) = \sigma \left[1 + \sum_{k=1}^{K} a_k c_k (1-\sigma^2)^k \right], \tag{5}$$

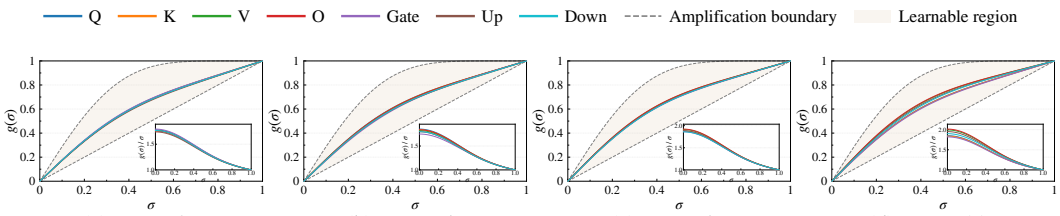
where $a_k = \binom{2k}{k}/4^k$ and $c_k \in [0,1]$ is learnable. For a fixed K, setting all $c_k = 0$ gives the identity mapping $g(\sigma) = \sigma$, while setting all $c_k = 1$ gives the largest amplification allowed by the parameterization. The corresponding pointwise learnable regions are shown in Figure. 9.

The main role of K is to control how far the transformation can extend into the small-singular-value region. The k-th basis is weighted by $(1-\sigma^2)^k$, which decays quickly with k when σ is large but remains close to one when σ is small. Higher-order terms therefore have little effect on the leading part of the spectrum and mainly change the spectral tail. For small σ, $(1-\sigma^2)^k \approx \exp(-k\sigma^2)$, so the effective scale of a high-order component moves toward smaller singular values as k increases. This is also visible in Figure. 9: increasing K mainly shifts the rising part of the upper boundary toward the origin, while the medium- and large-singular-value regions become nearly unchanged.

The slope of the mapping is $g'_K(0) = 1 + \sum_{k=1}^{K} a_k c_k$, so a larger K allows a steeper response near the spectral tail. In other words, increasing K gives the model more freedom to adjust small singular values. This does not, however, imply that a larger K should always lead to a better solution. Once the transformation already has enough capacity over the useful part of the spectrum, newly added high-order terms mainly act on very small singular values. This distinction matters because the tail of the spectrum contains low-energy directions, which can also be more sensitive to stochastic noise or weakly informative variations. Raising these singular values can make the spectrum more balanced, but an overly aggressive correction may amplify such components together with useful signals. We therefore do not expect the best spectral geometry to be the flattest one permitted by the parameterization. This is consistent with what we observe in practice: even for relatively large K, the learned transformation does not fully follow the maximal amplification boundary near $\sigma = 0$. The model still preserves a visible spectral tail instead of pushing all singular values toward one.

There is also little benefit in continuing to increase K once this region is sufficiently covered. In the small-σ regime, neighboring high-order bases become increasingly similar; for $k\sigma^2 \ll 1$, $(1-\sigma^2)^k \approx 1 - k\sigma^2$. As a result, very large K introduces additional degrees of freedom that are largely concentrated in the same narrow tail region. These extra components can be redundant and make the learned transformation more sensitive to optimization noise without providing much additional freedom over the rest of the spectrum.

C.4 Learned Amplification across Weights



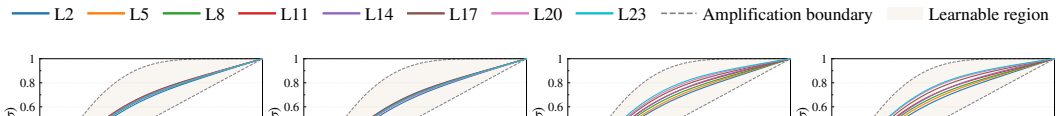
(a) Layer 2 (b) Layer 5 (c) Layer 8 (d) Layer 11

(e) Layer 14 (f) Layer 17 (g) Layer 20 (h) Layer 23

Figure 10: Learned amplification $g(\cdot)$ of **different weights in each layer** on LLaMA3-350M.

Figure 10 illustrates the spectral amplification functions $g(\cdot)$ adaptively learned by LASR ($K = 6$) for different parameter types within the same network layer. We observe distinct responses to spectral signals across network depths. In shallow layers, different parameters tend to learn relatively similar amplification patterns, whereas in deeper layers, their learned spectral responses become increasingly differentiated. This depth-dependent divergence highlights the importance of adaptively modeling the spectral characteristics of different parameters.

C.5 Learned Amplification across Layers



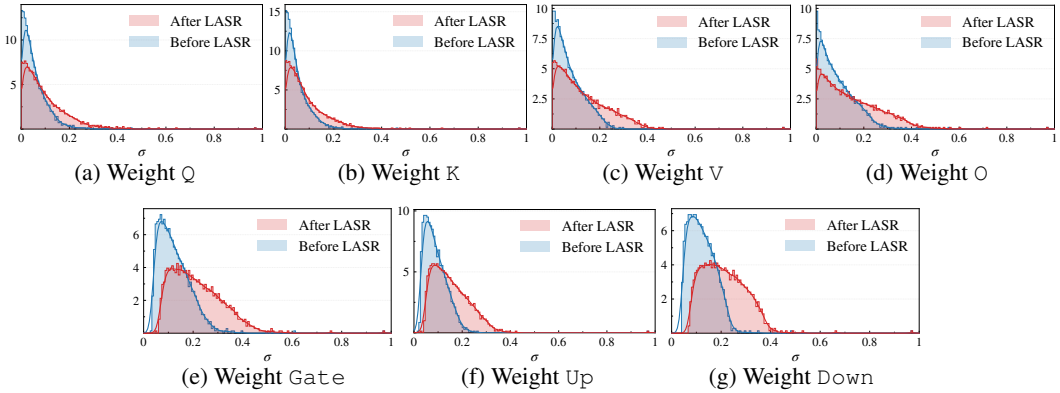
(a) Weight `Q` (b) Weight `K` (c) Weight `V` (d) Weight `O`

(e) Weight `Gate` (f) Weight `Up` (g) Weight `Down`

Figure 11: Learned amplification $g(\cdot)$ of **each weight across different layers** on LLaMA3-350M.

Figure 11 illustrates the spectral amplification functions $g(\cdot)$ adaptively learned by LASR ($K = 6$) for different parameter types across network layers. **Different parameter types exhibit distinct responses to spectral signals across layers**. In particular, the learned amplification patterns for `Q`, `K`, and `Gate` remain relatively stable across depth, with only minor differences between shallow and deep layers. In contrast, `V`, `O`, `Up`, and `Down` show pronounced layer-wise variations, indicating stronger depth-dependent spectral adaptation.

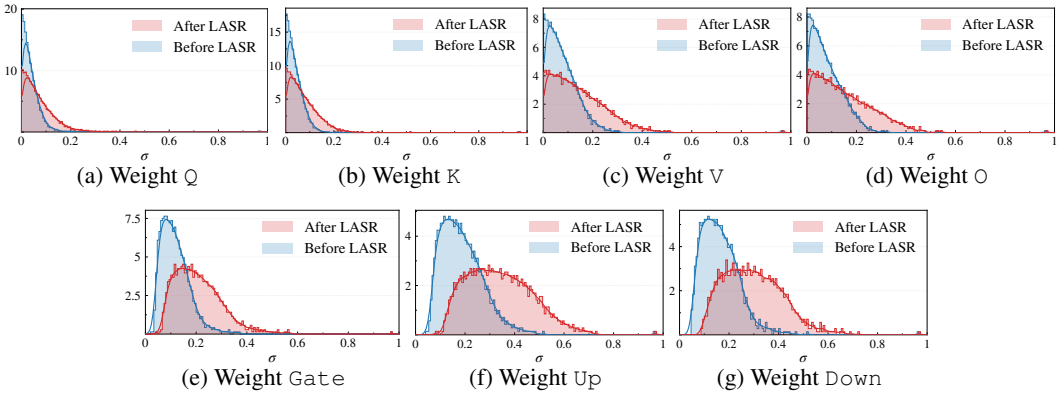
C.6 SPECTRAL AMPLIFICATION IN LASR: $\mathbf{W}$ VS. $\widehat{\mathbf{W}}$ (BEFORE AND AFTER LASR)



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

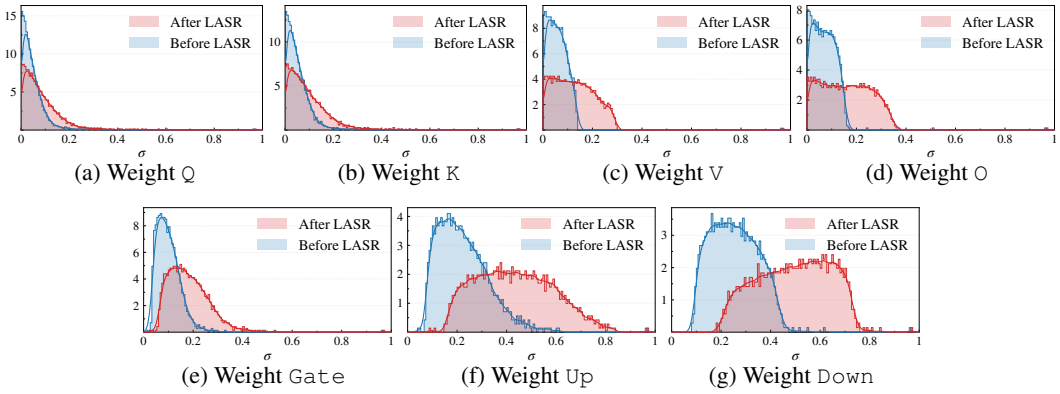
Figure 12: Distribution comparisons **before** and **after** LASR on layer-2 of LLaMA3-350M.



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

Figure 13: Distribution comparisons **before** and **after** LASR on layer-12 of LLaMA3-350M.



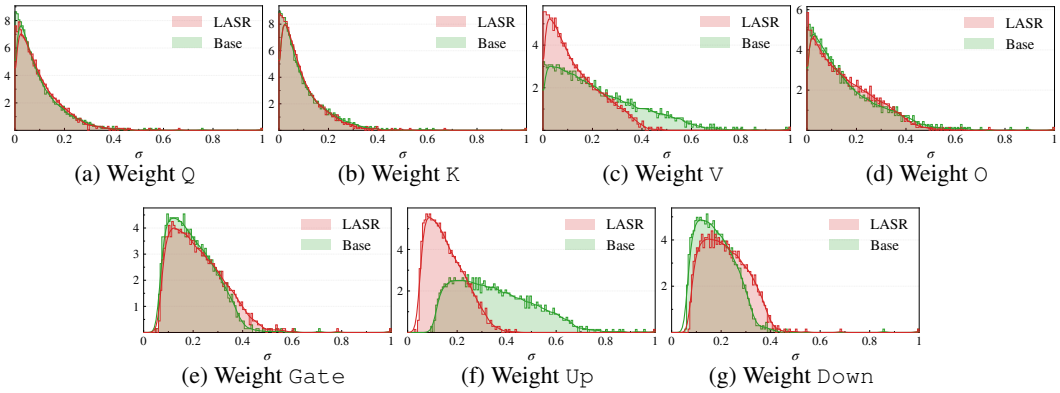
(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

Figure 14: Distribution comparisons **before** and **after** LASR on layer-22 of LLaMA3-350M.

Figures 12 13 and 14 present the spectral distributions of different weight matrices before and after LASR transformation ($K = 6$) at Layers 2, 12, and 22 of the 350M model. The results consistently show that LASR amplifies the singular-value spectrum across different layers and parameter types. More notably, we observe a pronounced parameter-wise heterogeneity in the learned spectral amplification: **different weight matrices exhibit substantially different reshaping behaviors**. This finding directly supports the central motivation of our method. Instead of prescribing a universal spectral equalization strength for all parameters, LASR allows each weight matrix to adaptively learn its preferred degree of spectral amplification under the task objective.

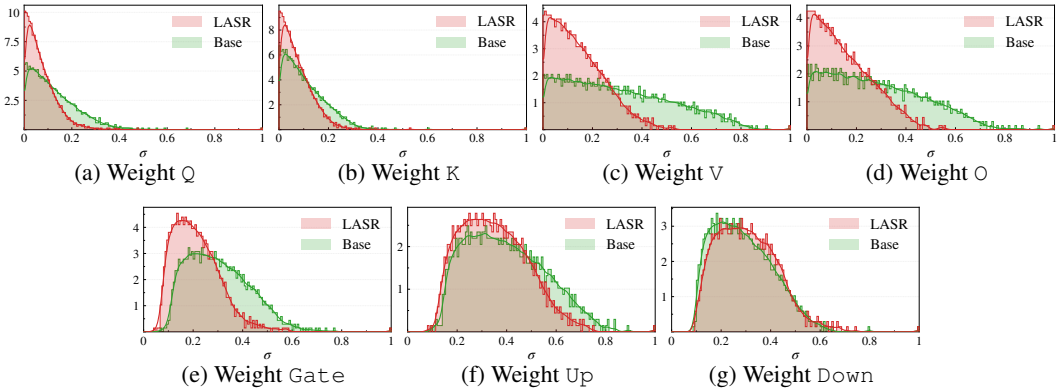
C.7 SPECTRAL COMPARISON: $\mathbf{W}_{\text{Base}}$ VS. $\widehat{\mathbf{W}}_{\text{Lasr}}$



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

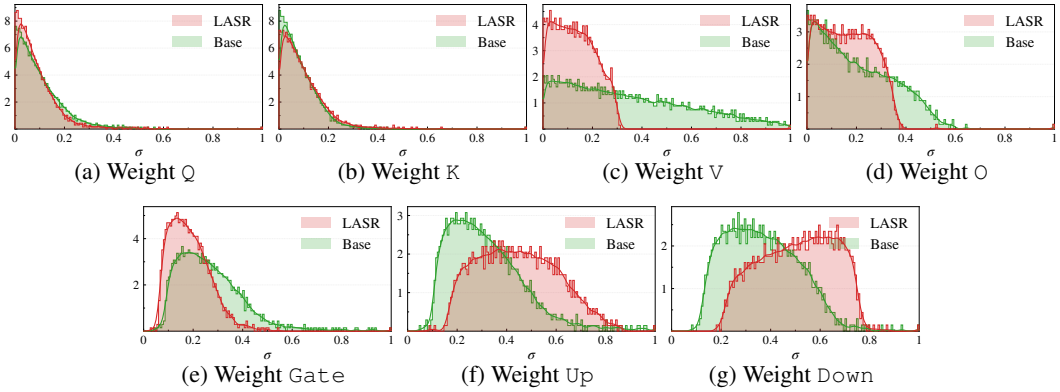
Figure 15: Distribution comparisons LASR and Vanilla pretraining on layer-2 of LLaMA3-350M.



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

Figure 16: Distribution comparisons LASR and Vanilla pretraining on layer-12 of LLaMA3-350M.



(a) Weight Q (b) Weight K (c) Weight V (d) Weight O

(e) Weight Gate (f) Weight Up (g) Weight Down

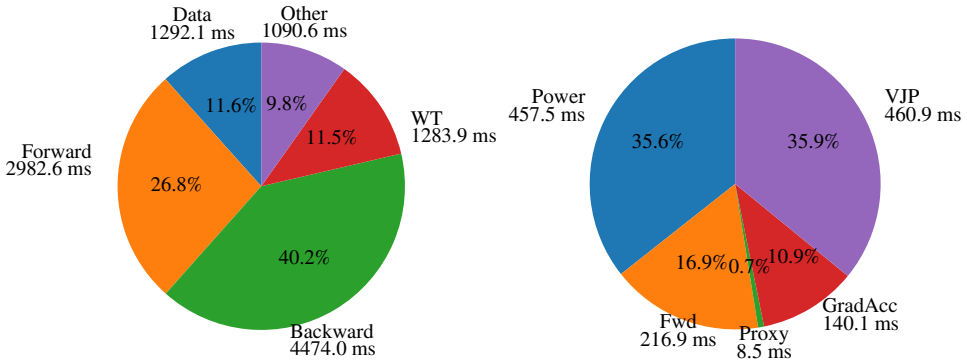
Figure 17: Distribution comparisons LASR and Vanilla pretraining on layer-22 of LLaMA3-350M.

Figures 15 16 and 17 compare the spectral distributions of different weight matrices at Layers 2, 12, and 22 of the 350M model under vanilla training and LASR ($K = 6$). The spectral changes vary substantially across layers and parameter types, further supporting our motivation that each weight matrix should adaptively determine its preferred degree of spectral reshaping rather than follow a uniform transformation. Moreover, compared with vanilla training, LASR generally produces more balanced and concentrated spectra across weights except for Q and K weights, indicating improved spectral conditioning. This observation is consistent with recent findings that better-conditioned weight spectra are associated with more favorable training and generalization behavior.

C.8 TRAINING PROFILES

Table 7: Comparison of overall training profiles. Total reports the average end-to-end training time per iteration with standard deviation. Overhead denotes the relative increase in iteration time over BASE. WT denotes the profiled Weight Transformer time, and Ratio denotes its fraction of the total iteration time.

	Overall (ms)	**Overhead**	**WT (ms)**	**Ratio of Overall**	**Memory (G)**	**Ratio to Base**
BASE	$9761.2_{\pm 412.5}$	–	110.4	-	194.7	-
ONI	$11010.8_{\pm 334.7}$	12.8%	1123.1	10.2%	195.2	1.003×
ENF	$10722.8_{\pm 643.2}$	9.9%	678.7	6.3%	196.5	1.010×
CANS	$11823.4_{\pm 289.6}$	21.1%	1407.0	11.9%	196.4	1.009×
IFNSO	$11675.3_{\pm 303.2}$	19.6%	1328.8	11.4%	195.8	1.006×
PC	$10728.4_{\pm 341.0}$	9.9%	1143.8	10.7%	195.3	1.003×
LASR	$11123.2_{\pm 342.8}$	14.0%	1283.9	11.5%	195.5	1.004×



(a) Overall Training Runtime Profile of LASR (b) WT Runtime Breakdown of LASR

Figure 18: Runtime profiling of LASR at the overall training and Weight Transformer levels.

Table 7 compares the overall training efficiency of different spectral reparameterization methods on LLaMA3-350M, while Fig. 18 further visualizes the runtime composition of LASR at both the overall training and Weight Transformer levels. We use BASE, which follows the same Weight Transformer execution pipeline without applying a non-trivial spectral transformation, as the reference. Compared with BASE, the spectral methods introduce moderate additional computational costs, with iteration-time overheads ranging from 9.9% to 21.1%. LASR requires 11123.2 ms per iteration, corresponding to a 14.0% overhead. This remains comparable to existing spectral baselines: LASR is only 3.7% slower than PC and 1.0% slower than ONI, while being 5.9% and 4.7% faster than CANS and IFNSO, respectively. As shown in Fig. 18, the WT module accounts for only 11.5% of the overall runtime, indicating that the additional spectral transformation does not dominate the end-to-end training cost.

Importantly, this additional flexibility introduces almost no extra memory burden. The peak memory consumption of all spectral methods remains within 1.01× that of BASE, and LASR uses only 195.5 GB compared with 194.7 GB for BASE, corresponding to merely a 1.004× memory footprint. Therefore, the performance improvements brought by the learnable spectral adaptation are achieved with negligible additional memory consumption and only moderate runtime overhead, preserving an efficiency profile comparable to existing spectral reparameterization methods.