# ProPolar: Progressive Polar Decomposition for Implicit Neural Representations

**Pureum Kim**
Korea University
pureum_kim@korea.ac.kr

**Younggeon Ryu**
Korea University
alphayoung8@korea.ac.kr

**Dongyoon Lee**
Korea University
dongyoon001@gmail.com

**Hae Beom Lee**[†]
Korea University
haebeomlee@korea.ac.kr

**Kyong Hwan Jin**[†]
Korea University
kyong_jin@korea.ac.kr

## Abstract

Implicit neural representations (INRs) capture global low-frequency structure during the early stage of training and then refine localized high-frequency details, afterwards. However, standard optimizers are agnostic to the coarse-to-fine learning behavior of INRs. Such optimizers accumulate a gradient matrix in momentum buffer and retain components misaligned with the dominant low-frequency directions in the early stage. We propose a progressively growing rank scheduler and a rank-aware learning-rate scheduler for enhancing a subspace-based momentum optimization, where a momentum matrix is updated within a low-rank subspace. The rank schedule expands the subspace during training, and the scheduler holds the peak learning rate through rank growth so that the added subspace contributes to fine-detail recovery. Experiments on image fitting, single-image super-resolution (SISR), and neural radiance field optimization show that the proposed method improves convergence and peak reconstruction quality over competitive optimizers. The largest gains appear in the high-frequency refinement stage, mirroring the coarse-to-fine progression that the rank schedule targets. A significant study on hyperparameter optimization (HPO) on NeRF further confirms that these gains persist under matched search budgets. https://github.com/pur3um/ProPolar

## 1 Introduction

Implicit neural representations (INRs) represent discrete signals as continuous functions parameterized by coordinate-based neural networks [4, 37–39]. Their training exhibits the coarse-to-fine progression in which global low-frequency structure forms first and high-frequency details are refined later [12, 28, 43, 46, 47]. Such a stage-dependent process suggests that requested capacity increases proportional to training iterations [46, 49, 50]: low-frequency reduction in the early stage benefits from a low-rank subspace, whereas high-frequency recovery in the later stage benefits from a high-rank subspace [22, 33, 44, 50]. INR optimization therefore benefits from regulating the update structure throughout training.

Recent matrix-aware optimizers address a regulating structure of updating by decomposing layer-wise gradients or exploiting a low-rank structure of optimizer states [7, 14, 41, 42, 45, 50]. Such designs suggest the importance of regulating update-space structure during optimization. However, most optimizers retain a fixed updating representation throughout training. Full-rank optimizers use the same full-dimensional representation throughout training [22, 44, 48, 50]. A momentum matrix accumulates a full-sized gradient, including components misaligned with dominant updates during

[†]Co-corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

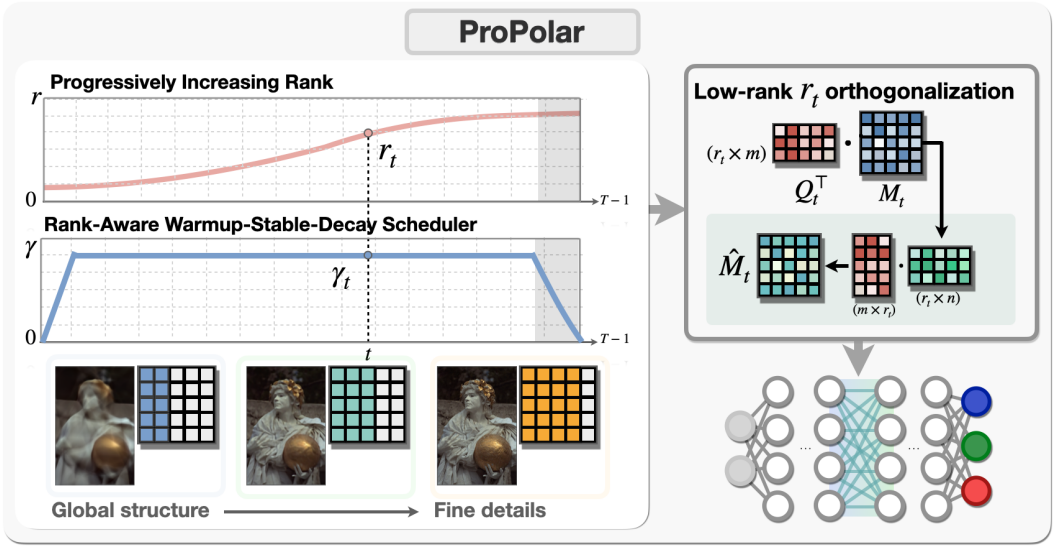



Figure 1: **Overview.** ProPolar introduces a progressive active-rank schedule $r_t$ that expands the momentum subspace from dominant low-rank directions to higher-rank directions during INR training. For the current rank budget, the orthonormal basis $Q_t \in \mathbb{R}^{m \times r_t}$ defines the active subspace used for projected polar orthogonalization. The rank-aware WSD schedule $\gamma_t$ maintains the peak step size during rank growth, allowing newly activated directions to contribute to fine-detail recovery before decay begins.

the early stage. Fixed-rank optimizers assign capacity to less informative directions during the early stage and constrain the update subspace during the later detail-refinement stage [30, 50]. In contrast, the coarse-to-fine behavior of INR training suggests that a fixed-dimensional update representation is misaligned with stage-dependent reconstruction demands [28, 34, 46]. The alignment issue motivates an optimizer with a progressively increasing rank that updates parameters in the original parameter space. Prior matrix-aware optimizers exploit compact update subspaces through low-rank projection or orthogonalization, but they are not designed for the coarse-to-fine training dynamics of INRs [11, 50].

In this paper, we propose ProPolar, an optimizer that projects matrix-valued momentum onto a progressively expanding subspace to match the coarse-to-fine dynamics of INR training. As illustrated in Figure 1, early updates concentrate on a small set of dominant directions associated with low-frequency structures, while the active subspace broadens later in training as fine-detail recovery proceeds. To preserve the contribution of these later directions, we pair the rank schedule with a learning-rate schedule that maintains the peak step size during rank growth. This combination aligns the optimizer state with the coarse-to-fine learning pattern of INRs [29, 46].

Experiments on image fitting, single-image super-resolution (SISR), and neural radiance field optimization show that the proposed method improves convergence and reconstruction quality over prior alternatives. Since INR training can be sensitive to optimization hyperparameters, we further conduct a unified tree-structured parzen estimator (TPE)-based HPO comparison on NeRF, where Adam, Muon, and the proposed optimizer are tuned under the same search budget [1, 3, 5]. The proposed optimizer remains the top performer under this tuned comparison, indicating that its gains are not merely due to hyperparameter mismatch.

Our contributions are summarized as follows:

- We analyze INR optimization through stage-dependent update, identifying the misalignment between fixed-rank representations and the progressive increase in reconstruction complexity as an obstacle.
- We propose an adaptive-rank momentum optimizer that represents momentum matrices in progressively increasing subspaces. In the early training phase, our optimizer concentrates on low-rank polar decomposed components and adds finer update directions as training progresses.
- We show consistent reconstruction improvements across image fitting, single-image super-resolution, and neural radiance field optimization, and further verify that the gains persist under a unified TPE-based HPO comparison in NeRF experiments.

## 2 Related Work

**Implicit Neural Representations.** Implicit neural representations (INRs) parameterize discrete signals as continuous coordinate-to-value functions [37]. Standard coordinate-based MLPs exhibit spectral bias and learn low-frequency components earlier than high-frequency details [17]. Fourier features and positional encodings mitigate this bias and enable high-frequency learning [38]. Periodic activations such as SIREN improve signal reconstruction and derivative fidelity [37]. FINER develops this line by introducing variable-periodic activations [20]. Several INR methods exploit the coarse-to-fine annealing behavior. SAPE progressively unmasks higher-frequency components over time and space [12]. PINs learn multi-scale implicit representations with progressive levels of detail [16]. BARF applies a similar strategy to NeRF optimization and registration [19]. These results indicate that frequency content and reconstruction detail are not treated uniformly throughout training, which motivates optimizer designs that account for stage-dependent fitting dynamics.

**INR Optimization and Matrix-Aware Optimizers.** Optimization affects INR reconstruction quality beyond representation design. Training-side methods address spectral bias through Fourier reparameterization or gradient adjustment [35, 36]. Rank-based INR studies relate reconstruction fidelity to stable rank, rank collapse, and rank restoration [23, 51]. These results suggest that update-space structure and rank-related dynamics are important considerations in high-fidelity INR fitting. Matrix-aware optimizers exploit the structure of matrix-valued parameters rather than treating updates as unstructured vectors. Muon applies orthogonalized updates to 2D hidden-layer parameters and Adam-style updates to non-matrix parameters [14]. Related low-rank and low-dimensional methods use projected gradients, compressed optimizer states, or factorized momentum to reduce optimizer-state cost while retaining full-parameter updates [10, 22, 30, 50]. Prior work has shown that gradient-based training tends to evolve in a low-dimensional subspace of the full parameter space [8, 18]. We specialize this observation to INR optimization by representing matrix-valued momentum in a subspace whose rank progressively increases, paired with a learning-rate schedule that preserves the step size during rank growth.

## 3 Methodology

### 3.1 INR Optimization Setup

We consider an $L$-layer coordinate MLP $f_\theta : \mathcal{X} \to \mathcal{Y}$ trained by minimizing

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \|f_\theta(x_i) - y_i\|_2^2, \tag{1}$$

where $\theta$ denotes the trainable parameters of $f_\theta$ and $\{(x_i, y_i)\}_{i=1}^N$ are coordinate-value observations of the target signal. Let $\mathcal{H}$ denote the hidden-layer weight matrices of $f_\theta$, where each $W \in \mathcal{H}$ has shape $m \times n$. Throughout the methodology, the terms *gradient matrix* and *momentum matrix* refer exclusively to the gradient and momentum associated with hidden-layer weight matrices. At each iteration $t$, the optimizer maintains a Nesterov momentum matrix $M_t \in \mathbb{R}^{m \times n}$ for each $W \in \mathcal{H}$.

Section 5.1 shows that the spectra of $M_t$ and updated momentum matrix retain a dominant low-rank structure throughout INR training, while their effective rank grows over training. This stage-dependent spectral behavior matches the coarse-to-fine learning pattern of INRs [28, 46].

**Momentum Orthogonalization.** Rank collapse in INR weight spectra reduces reconstruction fidelity [23, 51]. Matrix-aware optimizers mitigate rank collapse by orthogonalizing updates from $M_t$, yet full-space orthogonalization amplifies tail singular directions misaligned with the dominant update subspace [14, 23, 27].

### 3.2 Progressive Polar Decomposition

Orthogonalizing a matrix—computing its nearest semi-orthogonal matrix—yields the steepest-descent direction under the spectral norm [13, 14]. Applied to the momentum matrix $M_t$, however, full-rank orthogonalization amplifies every singular vector equally, including uninformative directions [27, 50]. In INR training, the spectrum of a momentum term remains approximately low-rank throughout most of training (Section 5.1), and the tail singular vectors are misaligned with the dominant subspace (Section 5.2). Inspired by [11], we project $M_t$ onto its progressively increasing low-rank ($r_t$) subspace using a Gaussian random matrix $\Omega_t \in \mathbb{R}^{n \times r_t}$ and obtain an orthonormal column basis as follows:

$$Q_t = \mathrm{qf}(M_t \Omega_t) \in \mathbb{R}^{m \times r_t} \tag{2}$$

where $\mathrm{qf}(\cdot)$ returns the $Q$-factor of a thin QR decomposition. Standard randomized-SVD analysis guarantees that $Q_t Q_t^\top M_t$ approximates the best rank-$r_t$ truncation of $M_t$ [9]. Let $\mathcal{P}(\cdot)$ denote the orthogonal/polar factor of the polar decomposition which is approximated by 5-step Newton–Schulz iterations [14]. The proposed update is calculated through a $r_t \times n$ reduced matrix:

$$\hat{M}_t = Q_t \mathcal{P}(Q_t^\top M_t), \qquad \hat{M}_t \in \mathbb{R}^{m \times n}. \tag{3}$$

The details about progressive polar decomposition based on the low-rank subspace orthogonalization [11] are described in Algorithm 1.

### 3.3 Progressively Increasing Rank

We run optimization for $T$ iterations, indexed by $t \in \{0, \ldots, T-1\}$. For each hidden-layer matrix $W \in \mathcal{H}$, let $d = \min(m, n)$. We choose $T_0$, $T_g$, $r^{\max}$, and an energy threshold $\rho$ such that

$$0 < T_0 < T, \qquad 2 \le T_{\mathrm{g}} \le T - T_0, \qquad 1 \le r^{\max} \le d, \qquad \rho \in (0, 1].$$

A static rank in [14] is efficient during early-phase training but becomes restrictive when INRs represent fine details. Conversely, using a large rank from the beginning can include low-energy tail singular directions in the polar decomposition [27]. We therefore estimate a data-dependent starting rank during a short initialization phase and then increase the rank by a cosine schedule.

**Initialization phase ($0 \le t < T_0$).** During the first $T_0$ iterations, we choose the rank $r_t$ from the singular values of the momentum matrix $M_t$. Let $\sigma_{t,1} \ge \sigma_{t,2} \ge \cdots \ge \sigma_{t,d}$ denote the singular values of $M_t$. For $M_t \neq 0$ and $k \in \{1, \ldots, d\}$, define

$$E_t(k) = \frac{\sum_{i=1}^{k} \sigma_{t,i}^2}{\sum_{i=1}^{d} \sigma_{t,i}^2}, \qquad r_t = \min\left\{r^{\max}, \min\left\{k \in \{1, \ldots, d\} : E_t(k) \ge \rho\right\}\right\}. \tag{4}$$

If $M_t = 0$, we set $r_t = 1$. At the end of the initialization phase, we set $r^{\mathrm{init}} = \lceil \mathrm{median}(r_0, \ldots, r_{T_0-1}) \rceil$ as the initial rank for the cosine phase.

**Cosine growth phase ($T_0 \le t < T$).** After initialization, the rank grows from $r^{\mathrm{init}}$ to $r^{\max}$ over a rank-growth length $T_{\mathrm{g}}$. For $t \ge T_0$, the rank used in the projected polar decomposition is

$$r_t = \left\lceil r^{\mathrm{init}} + (r^{\max} - r^{\mathrm{init}}) \frac{1 - \cos(\pi s_t)}{2} \right\rceil, \qquad s_t = \min\left\{1, \frac{t - T_0}{T_{\mathrm{g}} - 1}\right\}. \tag{5}$$

Thus $r_t = r^{\mathrm{init}}$ at $t = T_0$, reaches $r^{\max}$ at $t = T_0 + T_g - 1$, and remains at $r^{\max}$ thereafter. This schedule biases early training toward dominant directions and gradually admits higher-rank components for fine-detail recovery.

### 3.4 Rank-Aware Warmup-Stable-Decay Scheduler

The increasing-rank schedule in Eq. (5) admits additional subspace directions over training. If learning-rate decay begins before the rank reaches its target, these later directions are updated with smaller steps and can contribute less to fine-detail recovery. The scheduler acts only through the learning-rate multiplier; the rank trajectory itself is determined by Eq. (5).

Let $\tau_{\mathrm{sat}} = \min\{t : r_t = r^{\max}\}$ denote the first iteration at which the rank reaches its target. With layer-wise rank states, $\tau_{\mathrm{sat}}$ is the first iteration at which all tracked hidden-layer groups are saturated. We set the decay-start and decay-end iterations to

$$T_d = \min\left\{T - 1, \min\{t \ge T_0 : r_t = r^{\max}\} + \Delta_{\mathrm{sat}}\right\}. \tag{6}$$

Let $\gamma_{\min} \in [0, 1]$ be the minimum learning-rate ratio and $\Delta_{\mathrm{sat}} \ge 0$ be an optional delay after rank saturation. The learning-rate multiplier $\gamma_t \in [0, 1]$ is defined by Eq. (7)

$$\gamma_t = \begin{cases} \dfrac{t+1}{T_0}, & 0 \le t < T_0, \\ 1, & T_0 \le t < T_d, \\ \gamma_{\min} + (1 - \gamma_{\min}) \dfrac{T - 1 - t}{\max\{1, T - 1 - T_d\}}, & T_d \le t < T. \end{cases} \tag{7}$$

The same multiplier is applied to each parameter group $g$: $\eta_{t,g} = \gamma_t \eta_g^{\max}$. By maintaining the peak learning rate until rank saturation, rank-aware WSD prevents newly admitted components in the projected polar decomposition from being immediately suppressed by learning-rate decay. The final linear decay is applied only after the rank has saturated at $r^{\max}$.

**Algorithm 1** Progressive $r_t$ Polar Decomposition

**Require:** base lr $\eta_{\max}$, momentum coeff. $\beta$, weight decay $\lambda$; initial rank $r_0 = 1$, lr ratio $\gamma_t$ (Eq. (7))

1: $\eta_t \leftarrow \gamma_t \, \eta_{\max}$ ▷ rank-aware Warmup-Stable-Decay
2: Compute $\nabla_W \mathcal{L}_t$
3: **for** each $l \in \mathcal{H}$ **do**
*(layer index suppressed below)*
4: $M_t = \beta M_{t-1} + \nabla_W \mathcal{L}_t \, (W_{t-1} - \beta M_{t-1})$ ▷ Update momentum matrix
5: $r_t$ calculation from $M_t$ (Eqs. (4)–(5))
6: $\Omega_t \sim \mathcal{N}(0,1)^{n \times r_t}$
7: $Q_t \leftarrow \mathrm{qf}(M_t \, \Omega_t)$ ▷ thin QR
8: $\hat{M}_t \leftarrow Q_t \, \mathcal{P}(Q_t^\top M_t)$ ▷ Eq. (3)
9: $W_{t+1} \leftarrow (1 - \eta_t \lambda) \, W_t - \eta_t \, \hat{M}_t$
10: **end for**
11: Update non-matrix parameters ($W \notin \mathcal{H}$) with Adam

Table 1: **Inputs to Algorithm 1.** Symbols defined in Section 3 ($\eta_{\text{Adam}}, \beta, r_t, \gamma_t$) follow that section's definitions, while $(\beta_1, \beta_2)$ are the first- and second-moment EMA coefficients of the auxiliary Adam introduced only for the non-matrix update on Line 11.

| Task | $\eta_{\text{Adam}}$ | $\beta$ | $(\beta_1, \beta_2)$ |
|---|---|---|---|
| Image | 1e−4 | 0.95 | (0.9, 0.999) |
| SDF | | | |
| SISR | | | |
| NeRF | 4e−3 | | (0.9, 0.95) |

## 4 Experiments

In this section, we evaluate the proposed optimizer across INR reconstruction tasks and analyze the effect of rank-adaptive projected polar decomposition. The experiments assess fine-scale structure recovery and reconstruction quality. Representative INR tasks cover direct signal fitting and inverse problems. Each experiment compares the proposed method with task-specific baselines under consistent architectural and training settings. Table 1 provides the hyperparameter values used in all experiments. Unless otherwise stated, all experiments use a single NVIDIA GeForce RTX 3090 GPU, and table captions specify the hardware for experiments on other GPUs. The Appendix provides additional experiments, schedule hyperparameters, HPO details, and the cost reduction from basis reuse.

### 4.1 Signal Representation

**Overfitting of Image Signals.** We evaluate image representation performance on the 24-image Kodak dataset [6]. Table 2 reports quantitative results. Figure 2 shows representative reconstructions with black diagonal bands denoting pixel-wise absolute differences from the ground-truth images. The proposed optimizer improves reconstruction quality across all evaluated architectures. The results support the proposed method as an effective update rule for INR fitting. Appendix A.1 extends this comparison to WIRE [32] and FINER [20].

**Signed Distance Field Overfitting.** We evaluate neural signed distance field (SDF) fitting. SDF fitting requires consistency among the zero-level surface, the distance field, and spatial gradients. Following the public baseline [37], we use a ReLU MLP on the Armadillo and Dragon models from the Stanford 3D Scanning Repository. Figure 3 shows representative geometry reconstructions. The proposed optimizer captures finer surface details than Adam, as highlighted in the zoomed regions.

Table 2: **Overfitting performance on Kodak24.** Quantitative comparison on ReLU MLP and SIREN. Each entry reports the mean over 24 images. Bold values indicate the best result.

| Model | PSNR ↑ | | SSIM ↑ | | LPIPS ↓ | |
|---|---|---|---|---|---|---|
| | Adam | **Ours** | Adam | **Ours** | Adam | **Ours** |
| ReLU MLP | 22.55 | **28.70** | 0.5404 | **0.7855** | 0.5835 | **0.3417** |
| SIREN [37] | 33.67 | **35.05** | 0.9104 | **0.9320** | 0.0754 | **0.0751** |

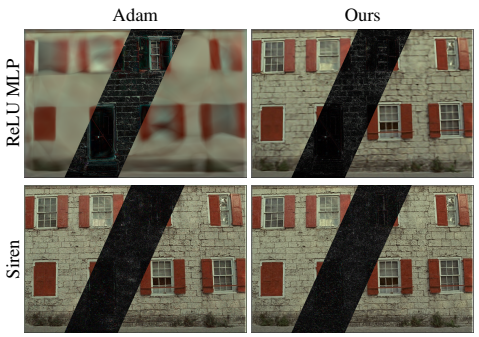



Figure 2: **Qualitative comparison on image overfitting.** Reconstructions from Adam and the proposed optimizer for ReLU MLP and SIREN.

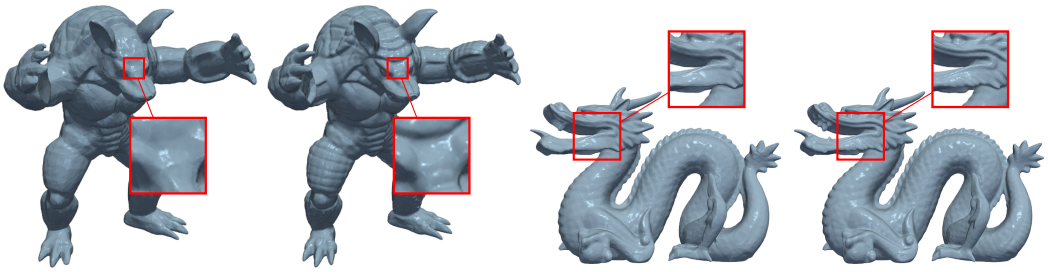



Figure 3: **Qualitative comparison for SDF overfitting.** Geometry reconstructions and zoomed regions on Armadillo and Dragon.

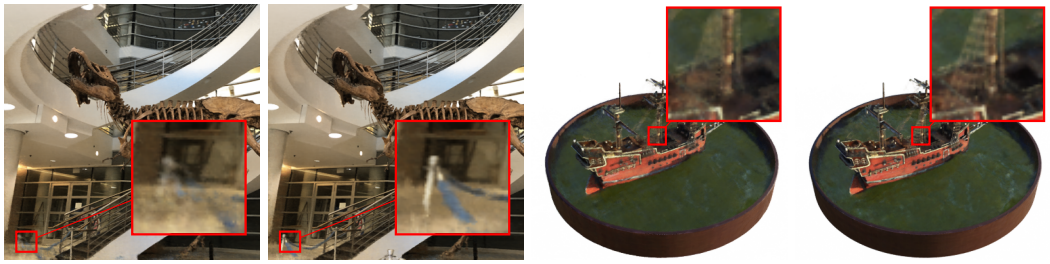



Figure 4: **Qualitative comparison on NeRF [25] scenes.** On both the real-world *Trex* (LLFF) and the synthetic *Ship* (Blender) scenes, our method produces sharper fine-grained details than Muon [14], as highlighted in the zoom-in.

### 4.2 Inverse Problem

**Single Image Super-Resolution.** We evaluate single-image super-resolution (SISR) on the DIV2K dataset [40] with an upsampling factor of 4. Table 3 reports quantitative results across INR architectures. The high-resolution image is represented as a continuous coordinate-based function. This formulation requires the optimizer to recover high-frequency details from downsampled observations. The proposed optimizer achieves the best or tied-best PSNR across the evaluated architectures. The results support the proposed method as an effective update rule for INR-based inverse reconstruction.

Table 3: **Quantitative comparison.** The best-performing optimizer for each architecture is highlighted in bold.

| Model | PSNR ↑ | | |
|---|---|---|---|
| | Adam [15] | Muon [14] | **Ours** |
| ReLU MLP | 19.65 | <u>22.01</u> | **23.47** |
| SIREN | 24.32 | **24.73** | **24.73** |
| WIRE | 23.97 | <u>24.62</u> | **24.68** |
| FINER | <u>25.23</u> | 25.19 | **25.45** |

**Neural Radiance Fields (NeRF).** We evaluate the proposed optimizer on novel view synthesis with neural radiance fields [2, 25, 26]. NeRF represents a scene as a continuous volumetric function and renders images from queried camera rays. This task requires optimization over geometry and appearance from multi-view observations. As shown in Table 4, we conduct experiments on the Synthetic NeRF [25] and LLFF datasets [24]. Each NeRF experiment uses the default hyperparameter configuration of its corresponding baseline implementation, and the proposed optimizer is matched to the same configuration as Adam. For validating progressive rank update, MoFaSGD [22] (fixed & low-rank subspace update) and Muon [14] (full-rank subspace update) are compared.

Figure 4 shows that our method recovers finer details. Table 5 reports reconstruction metrics on Mip-NeRF. Table 4 reports scene-level results for vanilla NeRF on the LLFF and Synthetic NeRF datasets at 100k iterations; results at the original 200k iteration budget are in Appendix A.2. Table 6 reports reconstruction metrics on Instant-NGP under the 1-second training budget setting reported in [26]. For Tables 4–7, bold and underlined values indicate the best and second-best results, respectively. Gray cells report average scores, and blue-shaded cells denote the proposed method.

On Mip-NeRF, the proposed optimizer improves reconstruction quality across synthetic scenes. Table 5 shows higher average scores than AdamW on Synthetic NeRF scenes. Instant-NGP results show consistent gains across all synthetic scenes. These results indicate that the proposed optimizer improves reconstruction quality across volumetric scene representations.

Table 4: **Quantitative comparison on vanilla NeRF [25].** Scene-level PSNR and SSIM are reported on the LLFF and Synthetic NeRF datasets. Experiments use the original hyperparameter configuration with 100k iterations.

| Metric | Method | LLFF | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Fern | Flower | Fortress | Horns | Leaves | Orchids | Room | T-Rex | Avg. |
| PSNR ↑ | Adam [15] | 26.38 | 27.47 | 31.74 | 27.79 | 21.92 | 21.10 | 31.96 | 26.90 | 26.91 |
| | MoFaSGD [22] | 26.24 | 27.30 | 31.42 | 27.32 | 21.56 | 20.98 | 32.01 | 26.71 | 26.69 |
| | Muon [14] | 26.84 | 28.12 | 32.21 | 28.60 | 22.16 | 21.42 | 33.17 | 27.79 | 27.53 |
| | Ours | **27.22** | **28.68** | **32.82** | **29.02** | **22.54** | **21.80** | **33.72** | **28.44** | **28.03** |
| SSIM ↑ | Adam [15] | 0.8322 | 0.8566 | 0.9036 | 0.8702 | 0.7737 | 0.7135 | 0.9515 | 0.9000 | 0.8501 |
| | MoFaSGD [22] | 0.8200 | 0.8435 | 0.8946 | 0.8489 | 0.7454 | 0.6934 | 0.9481 | 0.8884 | 0.8352 |
| | Muon [14] | 0.8507 | 0.8777 | 0.9184 | 0.8934 | 0.7880 | 0.7318 | 0.9604 | 0.9175 | 0.8672 |
| | Ours | **0.8584** | **0.8880** | **0.9270** | **0.8984** | **0.8014** | **0.7508** | **0.9629** | **0.9280** | **0.8768** |

| Metric | Method | Synthetic | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Chair | Drums | Ficus | Hotdog | Lego | Materials | Mic | Ship | Avg. |
| PSNR ↑ | Adam [15] | 32.57 | 25.08 | 28.02 | 35.40 | 30.23 | 28.32 | 32.04 | 28.47 | 30.02 |
| | MoFaSGD [22] | 31.57 | 24.76 | 27.45 | 34.77 | 29.90 | 28.51 | 31.02 | 28.17 | 29.26 |
| | Muon [14] | 33.22 | 25.22 | 28.28 | 36.23 | 30.95 | 29.19 | 32.64 | 28.87 | 30.58 |
| | Ours | **34.28** | **25.72** | **29.30** | **37.36** | **31.94** | **29.95** | **33.59** | **29.57** | **31.46** |
| SSIM ↑ | Adam [15] | 0.9702 | 0.9217 | 0.9566 | 0.9757 | 0.9536 | 0.9483 | 0.9742 | 0.8576 | 0.9447 |
| | MoFaSGD [22] | 0.9600 | 0.9165 | 0.9497 | 0.9719 | 0.9488 | 0.9501 | 0.9688 | 0.8569 | 0.9374 |
| | Muon [14] | 0.9747 | 0.9242 | 0.9602 | 0.9790 | 0.9622 | 0.9585 | 0.9775 | 0.8683 | 0.9506 |
| | Ours | **0.9791** | **0.9304** | **0.9669** | **0.9817** | **0.9681** | **0.9628** | **0.9800** | **0.8773** | **0.9558** |

Table 5: **Quantitative comparison on Mip-NeRF [2].** PSNR and SSIM on Synthetic NeRF scenes, trained for 200k iterations.

| Metric | Method | Synthetic | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Chair | Drums | Ficus | Hotdog | Lego | Materials | Mic | Ship | Avg. |
| PSNR ↑ | AdamW [21] | 35.24 | 25.64 | 29.50 | 37.75 | 33.07 | 30.55 | 34.35 | 29.95 | 32.01 |
| | Muon [14] | 34.68 | 25.51 | 29.17 | 37.63 | 32.45 | 30.25 | 34.30 | 29.78 | 31.72 |
| | Ours | **35.58** | **25.71** | **29.67** | **38.25** | **33.56** | **31.12** | **34.91** | **30.28** | **32.39** |
| SSIM ↑ | AdamW [21] | 0.9829 | **0.9292** | 0.9675 | 0.9831 | 0.9743 | 0.9653 | 0.9814 | 0.8784 | 0.9578 |
| | Muon [14] | 0.9799 | 0.9239 | 0.9658 | 0.9818 | 0.9693 | 0.9620 | 0.9807 | 0.8746 | 0.9548 |
| | Ours | **0.9839** | 0.9289 | **0.9698** | **0.9848** | **0.9770** | **0.9691** | **0.9838** | **0.8866** | **0.9605** |

Table 6: **Quantitative comparison on Instant-NGP [26].** PSNR and SSIM are reported on Synthetic NeRF scenes.

| Metric | Method | Synthetic | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Chair | Drums | Ficus | Hotdog | Lego | Materials | Mic | Ship | Avg. |
| PSNR ↑ | Adam [15] | 29.41 | 21.90 | 26.95 | 32.19 | 28.24 | 25.35 | 29.69 | 24.27 | 27.25 |
| | Muon [14] | 28.79 | 21.73 | 21.91 | 32.37 | 27.02 | 25.80 | 24.13 | 25.04 | 25.84 |
| | Ours | **30.04** | **23.08** | **27.01** | **33.83** | **29.05** | **27.51** | **30.28** | **26.69** | **28.44** |
| SSIM ↑ | Adam [15] | 0.9373 | 0.8598 | 0.9317 | 0.9629 | 0.9284 | 0.8861 | 0.9642 | 0.8311 | 0.9127 |
| | Muon [14] | 0.9270 | 0.8813 | 0.8730 | 0.9555 | 0.9013 | 0.9095 | 0.9472 | 0.8101 | 0.9006 |
| | Ours | **0.9420** | **0.8980** | **0.9405** | **0.9646** | **0.9338** | **0.9286** | **0.9689** | **0.8412** | **0.9272** |

### 4.3 Hyperparameter Optimization (HPO)

In the NeRF comparison, all optimizers are evaluated under the same training protocol with fixed hyperparameters. Coordinate-based neural representations such as NeRF [25] are sensitive to optimization settings [39], and empirical optimizer comparisons depend on the tuning protocol [5]. A performance gap under a single fixed configuration is therefore partly attributable to hyperparameter mismatch rather than the optimizer design alone. To reduce this source of bias, we conduct an HPO comparison for the NeRF experiments under a unified search budget.

Each HPO trial trains NeRF for 100K iterations under a sampled optimizer configuration and uses the validation-set PSNR as the objective. Since each trial requires a full training run, we adopt the Tree-structured Parzen Estimator (TPE) [3] implemented in Optuna [1], which models the densities of high- and low-performing trials and samples candidates likely to improve validation performance. Each optimizer receives the same trial budget, and the configuration with the highest validation PSNR is evaluated once on the test set using PSNR and SSIM. For Adam, the search variables are the learning rate and learning-rate decay. For Muon and the proposed optimizer, the search variables are Adam-branch learning rate, learning-rate decay, and the Muon/projected-polar learning rate.

The resulting HPO comparisons are reported in Table 7. After tuning, the proposed optimizer maintains the best average reconstruction quality, suggesting that the gain is driven by the optimizer design rather than unfavorable baseline hyperparameters. Search spaces are provided in Appendix D.

Table 7: **NeRF hyperparameter tuning performance at 100K iterations.** Comparison of Adam [15], Muon [14], and Ours under fixed-hyperparameter and TPE-tuned [3] settings on Synthetic NeRF scenes.

| Evaluation | | Synthetic | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | HPO setting | Chair | | | Drums | | | Ficus | | | Hotdog | | |
| | | Adam | Muon | Ours | Adam | Muon | Ours | Adam | Muon | Ours | Adam | Muon | Ours |
| PSNR ↑ | Fixed HP | 32.57 | 33.22 | **34.28** | 25.08 | 25.22 | **25.72** | 28.02 | 28.28 | **29.30** | 35.40 | 36.23 | **37.36** |
| | TPE | 32.93 | 34.19 | **34.37** | 25.20 | **25.73** | 25.61 | 28.33 | **29.30** | **29.30** | 37.35 | 37.40 | **37.48** |
| SSIM ↑ | Fixed HP | 0.9702 | 0.9747 | **0.9791** | 0.9217 | 0.9242 | **0.9304** | 0.9566 | 0.9602 | **0.9669** | 0.9757 | 0.9790 | **0.9817** |
| | TPE | 0.9702 | 0.9773 | **0.9793** | 0.9259 | **0.9317** | 0.9293 | 0.9568 | 0.9656 | **0.9666** | 0.9775 | **0.9821** | **0.9821** |
| Metric | HPO setting | Lego | | | Materials | | | Mic | | | Ship | | |
| | | Adam | Muon | Ours | Adam | Muon | Ours | Adam | Muon | Ours | Adam | Muon | Ours |
| PSNR ↑ | Fixed HP | 30.23 | 30.95 | **31.94** | 28.32 | 29.19 | **29.95** | 32.04 | 32.64 | **33.59** | 28.47 | 28.87 | **29.57** |
| | TPE | 30.61 | 31.66 | **32.03** | 28.37 | 29.55 | **29.95** | 32.62 | 33.64 | **33.71** | 28.51 | 29.48 | **29.64** |
| SSIM ↑ | Fixed HP | 0.9536 | 0.9622 | **0.9681** | 0.9483 | 0.9585 | **0.9628** | 0.9742 | 0.9775 | **0.9800** | 0.8576 | 0.8683 | **0.8773** |
| | TPE | 0.9552 | 0.9651 | **0.9688** | 0.9485 | 0.9597 | **0.9627** | 0.9750 | 0.9802 | **0.9804** | 0.8618 | **0.8813** | 0.8774 |

# 5 Analysis

In this section, we analyze the rank dynamics of the momentum matrix $M_t$ and the parameter update $\Delta W_t = W_{t+1} - W_t$, and then examine how different components of the polar factor of $M_t$ affect frequency-band residual reduction during INR training. In Section 5.1, we first measure the normalized entropy effective rank of $M_t$ and $\Delta W_t$, and compare rank-schedule configurations to examine whether increasing the rank over training better matches this observed behavior. In Section 5.2, we decompose the full polar factor $\bar{M}_t$ into head and tail components and evaluate their effects on residual energy in low-, mid-, and high-frequency bands during image overfitting.

## 5.1 Effective Rank and Progressively Increasing Rank

To examine whether the effective rank [31] of the momentum matrices increases during training, we monitor their normalized entropy effective rank. This quantity is defined in Figure 5. A low value indicates that a small number of leading singular values account for a large fraction of the sum of singular values. Figure 5 presents the evolution of the normalized entropy effective rank of the momentum matrices $M_t$ and the parameter updates $\Delta W_t$. For most of training, both effective-rank measures remain low, then increase rapidly in the later iterations. This behavior suggests that, in the early stage of training, a low-rank approximation is sufficient for the momentum matrices and parameter updates. In later iterations, more singular values become non-negligible, supporting the use of a higher rank.

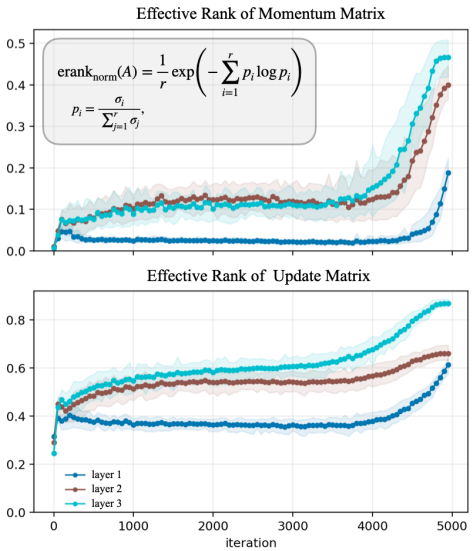



Figure 5: **Effective-rank dynamics during INR training.** Normalized entropy effective rank of the momentum matrix $M_t$ (top) and the parameter update $\Delta W_t = W_{t+1} - W_t$ (bottom). The normalized effective rank increases in later iterations, indicating that more singular components become non-negligible as training proceeds.

Figure 6 compares four configurations of the rank schedule in the image overfitting setting: increasing rank, full rank, fixed rank, and decreasing rank. Table 8 reports the detailed quantitative results. The increasing-rank schedule achieves the best reconstruction quality, whereas schedules with a static or decreasing rank yield lower performance. This result supports the proposed increasing rank schedule, which increases rank as optimization proceeds instead of keeping it fixed throughout training.

## 5.2 Frequency-Band Analysis on Head and Tail matrices

Let $M_t = U_t \Sigma_t V_t^\top$ be the singular value decomposition of the momentum matrix, and let $\bar{M}_t = U_t V_t^\top$ denote its full polar factor. We take the left and right singular vectors associated with the $k$ largest singular values, collect them as $U_{t,k}$ and $V_{t,k}$, and define $\bar{M}_{t,k} = U_{t,k} V_{t,k}^\top$.




Ground Truth | Increasing rank | Fixed rank | Full rank | Decreasing rank

Figure 6: **Rank-configuration comparison.** Qualitative comparison of increasing-rank, fixed-rank, decreasing-rank and full-rank schedules on image overfitting. Further details are provided in Appendix C.

We refer to $\bar{M}_{t,k}$ as the rank-$k$ head component. The corresponding tail component is defined as the remaining part of the polar factor, $\bar{M}_t - \bar{M}_{t,k}$. We measure the effect of the head and tail components of $\bar{M}_t$ on residual energy in low-, mid-, and high-frequency bands during image overfitting. For a frequency band $B$, we define the mean residual power as $E_t(B) = \frac{1}{|B|C} \sum_{\omega \in B} \sum_{c=1}^{C} |\mathcal{F}(R_t^{(c)})(\omega)|^2$, where $R_t$ is the residual image, $C$ is the number of color channels, and $\mathcal{F}$ denotes the two-dimensional Fourier transform. For visualization, we report the relative residual energy $\widetilde{E}_t(B) = E_t(B)/E_0(B)$.

Table 8: **Rank schedule ablation.** Mean $\pm$ std over the 24 images of Kodak24 under the same learning-rate scheduler. Best results are in bold. PSNR at intermediate checkpoints is reported in Appendix C.

| Method | PSNR ↑ | SSIM ↑ |
|---|---|---|
| Decreasing rank | $25.53 \pm 6.45$ | $0.7014 \pm 0.1136$ |
| Full rank | $25.22 \pm 7.08$ | $0.7018 \pm 0.1444$ |
| Fixed rank | $25.25 \pm 5.13$ | $0.7095 \pm 0.1234$ |
| **Increasing rank** | $\mathbf{27.33 \pm 5.37}$ | $\mathbf{0.7363 \pm 0.1210}$ |

**Frequency-band trajectories.** Figure 7 reports residual energy in three frequency bands under the head-only and tail-only settings for $k = 128$. The head term reduces low-frequency residual energy by orders of magnitude and continues to reduce mid- and high-frequency residuals. In contrast, the tail term decreases slowly in the low-frequency band and stays near its initial level in the other bands. These results suggest that the update components responsible for fine-detail recovery are concentrated in the head term of $\bar{M}_t$, rather than in the tail. Thus, the tail term alone does not account for high-frequency reconstruction.

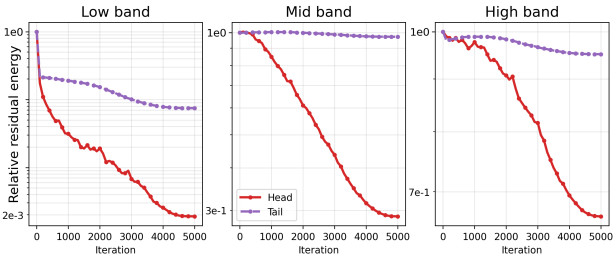



Figure 7: **Frequency-band residual energy analysis of the head and tail terms.** We compare the head and tail terms with $k = 128$ during 5000 iterations of image overfitting on Kodak24. The plots show relative residual energy in the low- (left), mid- (center), and high-frequency (right) bands.

**Ablation study.** We ablate the rank-aware WSD scheduler introduced in Section 3.4. For each optimizer, we compare training with and without the WSD scheduler under the same training budget. Figure 8 shows that the WSD scheduler consistently improves the proposed optimizer across the learning-rate range compared with its non-WSD counterpart. This behavior indicates that WSD is effective when coupled with the proposed optimizer: by maintaining the peak step size during the rank-growth phase, the scheduler matches the learning-rate schedule to the progressive expansion of the retained leading left and right subspaces. This coupling is a distinguishing characteristic relative to the fixed-rank baseline.

**Rank-growth schedule.** We compare cosine, logistic, and piecewise rank-growth schedules under the same learning-rate scheduler and rank bounds in Table 9. The three schedules yield similar PSNR and SSIM. Together with Table 8, this result suggests that the increase in rank contributes more to reconstruction quality than the shape of the growth curve. We adopt the cosine schedule since it requires no additional shape hyperparameter.

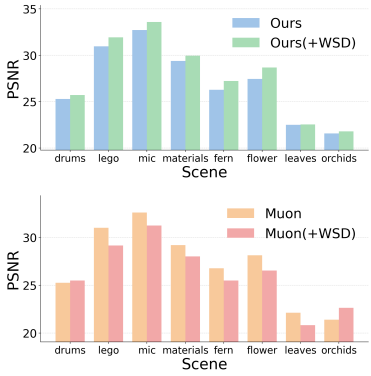



Figure 8: **Effect of the rank-aware WSD schedule.** PSNR across scenes with and without the WSD schedule, applied to the proposed optimizer (top) and the baseline (bottom), evaluated on synthetic NeRF scenes.

Table 9: **Rank-growth schedule.** Mean ± std over the 24 images of Kodak24 at 3,000 iterations. Experiments use a single NVIDIA RTX PRO 6000 GPU.

| Schedule | PSNR ↑ | SSIM ↑ |
|---|---|---|
| Cosine | 27.33 ± 5.38 | 0.7363 ± 0.1210 |
| Logistic | 27.32 ± 4.15 | 0.7355 ± 0.1492 |
| Piecewise | 27.32 ± 4.18 | 0.7358 ± 0.1402 |

Table 10: **Training time comparison on the Synthetic NeRF Lego scene.** Iter and Step denote the time per iteration and the `optimizer.step()` time, averaged over 10K iterations.

| Method | Params | Iter (ms) | Step (ms) |
|---|---|---|---|
| Adam | 1.19M | 96.63 | 5.23 |
| Muon | | 107.22 | 19.75 |
| Ours | | 112.70 | 25.45 |

## 6 Limitations

**Training time.** Similar to Muon [14], our method incurs additional computation relative to Adam due to the polar decomposition step. Table 10 reports the time per iteration and the optimizer-step time on the Lego scene after 50 warm-up iterations. In our preliminary experiments, periodically reusing the projection basis substantially reduced optimizer-step time with no noticeable loss of quality, suggesting a promising direction for further reducing this overhead.

**Gain under tuning.** In Table 7, our method reaches near its tuned performance under the default configuration, and TPE tuning yields an additional gain of 0.05 dB in average PSNR on Synthetic NeRF. This gain is narrow compared with those of the other optimizers.

**Scope of evaluation.** Our evaluation covers MLP-based coordinate INRs trained from scratch, and other recent INR architectures remain unexplored. Under a warm start from an initial encoding stage, the coarse structure is already fitted, and the benefit of the rank schedule is expected to decrease relative to its overhead.

## 7 Conclusion

In this paper, we proposed Progressive Polar Decomposition, a robust optimizer designed for INR training. The proposed method projects matrix-valued momentum onto a low-rank orthonormal basis, applies polar normalization in the compressed coordinates, and reconstructs the update in the original parameter space, with an energy-calibrated cosine schedule that progressively expands the subspace rank during training. This design aligns the optimizer state with the coarse-to-fine learning behavior of INRs. We evaluated the proposed method on image fitting, single-image super-resolution, and neural radiance field optimization, where it consistently improved reconstruction quality across the evaluated INR architectures, with PSNR gains of up to 6 dB on image fitting. For NeRF, we further conducted a unified HPO comparison to reduce the influence of optimizer-specific hyperparameter choices, and the proposed optimizer remained strongest under the tuned setting.

## Acknowledgments and Disclosure of Funding

This work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2024-00335741), Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS2025-25442405 Development of a Self-Learning World Model-Based AGI System for Hyperspectral Imaging), This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2026-25499511), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2025-02263080, Development of PIM Memory Technology for a High-Speed Interface-Based Programmable Computing Architecture).

## References


[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021.

[3] James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[4] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021.

[5] Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

[6] Rich Franzen. Kodak lossless true color image suite. http://r0k.us/graphics/kodak/, 1999. Online; accessed 5 May 2026.

[7] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[8] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

[9] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[10] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. *arXiv preprint arXiv:2402.03293*, 2024.

[11] Chuan He, Zhanwang Deng, and Zhaosong Lu. Low-rank orthogonalization for large-scale matrix optimization with applications to foundation model training. *arXiv preprint arXiv:2509.11983*, 2025.

[12] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Sape: Spatially-adaptive progressive encoding for neural optimization. *Advances in Neural Information Processing Systems*, 34: 8820–8832, 2021.

[13] Nicholas J. Higham. *Functions of Matrices*. Society for Industrial and Applied Mathematics, 2008.

[14] 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.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Zoe Landgraf, Alexander Sorkine Hornung, and Ricardo Silveira Cabral. Pins: progressive implicit networks for multi-scale neural representations. *arXiv preprint arXiv:2202.04713*, 2022.

[17] John Lazzari and Xiuwen Liu. Understanding the spectral bias of coordinate based mlps via training dynamics. *arXiv preprint arXiv:2301.05816*, 2023.

[18] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

[19] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5741–5751, 2021.

[20] Zhen Liu, Hao Zhu, Qi Zhang, Jingde Fu, Weibing Deng, Zhan Ma, Yanwen Guo, and Xun Cao. Finer: Flexible spectral-bias tuning in implicit neural representation by variable-periodic activation functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2713–2722, 2024.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[22] Pouria Mahdavinia and Mehrdad Mahdavi. Low-rank momentum factorization for memory efficient training. *arXiv preprint arXiv:2507.08091*, 2025.

[23] Julian McGinnis, Florian A Hölzl, Suprosanna Shit, Florentin Bieder, Paul Friedrich, Mark Mühlau, Björn Menze, Daniel Rueckert, and Benedikt Wiestler. Optimizing rank for high-fidelity implicit neural representations. *arXiv preprint arXiv:2512.14366*, 2025.

[24] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65 (1):99–106, 2021.

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

[27] Tianyu Pang, Yujie Fang, Zihang Liu, Shenyang Deng, Lei Hsiung, Shuhua Yu, and Yaoqing Yang. Htmuon: Improving muon via heavy-tailed spectral correction. *arXiv preprint arXiv:2603.10067*, 2026.

[28] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.

[29] Reza Rezaeian, Moein Heidari, Reza Azad, Dorit Merhof, Hamid Soltanian-Zadeh, and Ilker Hacihaliloglu. Sl2a-inr: Single-layer learnable activation for implicit neural representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26065–26074, 2025.

[30] Thomas Robert, Mher Safaryan, Ionut-Vlad Modoranu, and Dan Alistarh. Ldadam: Adaptive optimization from low-dimensional gradient statistics. *arXiv preprint arXiv:2410.16103*, 2024.

[31] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pages 606–610, 2007.

[32] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G. Baraniuk. Wire: Wavelet implicit neural representations, 2023.

[33] Steffen Schotthöfer, Timon Klein, and Jonas Kusch. A geometric framework for momentum-based optimizers for low-rank training. *arXiv preprint arXiv:2506.17475*, 2025.

[34] Steffen Schotthöfer, H Lexie Yang, and Stefan Schnake. Dynamical low-rank compression of neural networks with robustness under adversarial attacks. *arXiv preprint arXiv:2505.08022*, 2025.

[35] Kexuan Shi, Hai Chen, Leheng Zhang, and Shuhang Gu. Inductive gradient adjustment for spectral bias in implicit neural representations. *arXiv preprint arXiv:2410.13271*, 2024.

[36] Kexuan Shi, Xingyu Zhou, and Shuhang Gu. Improved implicit neural representation with fourier reparameterized training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 25985–25994, 2024.

[37] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[38] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33: 7537–7547, 2020.

[39] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2846–2855, 2021.

[40] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

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

[42] Junjie Wang, Pan Zhou, Yiming Dong, Huan Li, Jia Li, Xun Zhou, Qicheng Lao, Cong Fang, and Zhouchen Lin. Conda: Column-normalized adam for training large language models faster. *arXiv preprint arXiv:2509.24218*, 2025.

[43] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3295–3306, 2023.

[44] Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Taming momentum: Rethinking optimizer states through low-rank approximation. *arXiv preprint arXiv:2602.24283*, 2026.

[45] Ruihan Xu, Jiajin Li, and Yiping Lu. On the width scaling of neural optimizers under matrix operator norms i: Row/column normalization and hyperparameter transfer. *arXiv preprint arXiv:2603.09952*, 2026.

[46] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

[47] Zhi-Qin John Xu, Lulu Zhang, and Wei Cai. On understanding and overcoming spectral biases of deep neural network learning methods for solving pdes. *Journal of Computational Physics*, 530:113905, 2025.

[48] Xin Yu, Yujia Wang, Jinghui Chen, and Lingzhou Xue. Altlora: Towards better gradient approximation in low-rank adaptation with alternating projections. *arXiv preprint arXiv:2505.12455*, 2025.

[49] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

[50] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

[51] Jianqiao Zheng, Hemanth Saratchandran, and Simon Lucey. The inlet rank collapse in implicit neural representations: Diagnosis and unified remedy. *arXiv preprint arXiv:2602.01526*, 2026.

# A Additional Experiments

## A.1 Unified evaluation across architectures and tasks

We compare Adam, Muon, and ProPolar on ReLU MLP, SIREN, WIRE, and FINER under identical settings across optimizers. All experiments in this section use a single NVIDIA RTX PRO 6000 GPU, and absolute numbers differ from those in the main paper.

**Image overfitting.** Table A.1 reports PSNR on Kodak24. ProPolar achieves the highest mean PSNR on all four architectures.

**SDF fitting.** Table A.2 reports two errors on Armadillo and Dragon for WIRE and FINER. SDF denotes the mean absolute error between the predicted and ground-truth signed distances over all query points, and Near-surf. denotes the same error over query points within a distance of $\epsilon$ from the surface. ProPolar attains the lowest SDF error in three of the four settings, while Near-surf. shows no consistent winner. This result suggests that the gain of ProPolar on SDF fitting comes from reducing errors away from the surface.

Table A.1: **Image overfitting on Kodak24.** PSNR as mean $\pm$ std over 24 images. Best per architecture in bold.

| Model | PSNR ↑ | | |
|---|---|---|---|
| | Adam [15] | Muon [14] | **Ours** |
| ReLU MLP | $22.53 \pm 2.85$ | $24.92 \pm 2.91$ | $\mathbf{27.33 \pm 5.38}$ |
| SIREN | $37.68 \pm 2.29$ | $40.86 \pm 2.29$ | $\mathbf{41.22 \pm 3.51}$ |
| WIRE | $29.41 \pm 1.61$ | $36.23 \pm 2.73$ | $\mathbf{36.51 \pm 2.89}$ |
| FINER | $38.16 \pm 1.98$ | $42.27 \pm 3.03$ | $\mathbf{43.05 \pm 3.02}$ |

Table A.2: **SDF fitting on Armadillo and Dragon.** Mean absolute error over all query points and near-surface points. Best per architecture and shape in bold.

| Model | Optimizer | Armadillo | | Dragon | |
|---|---|---|---|---|---|
| | | SDF ↓ | Near-surf. ↓ | SDF ↓ | Near-surf. ↓ |
| WIRE | Adam | 0.03917 | **0.00092** | 0.03186 | 0.02890 |
| | Muon | 0.02181 | 0.00249 | **0.03157** | **0.02861** |
| | Ours | **0.02174** | 0.00155 | 0.03175 | 0.02869 |
| FINER | Adam | 0.08490 | **0.00558** | 0.04006 | 0.00537 |
| | Muon | 0.08070 | 0.00562 | 0.03761 | **0.00511** |
| | Ours | **0.07580** | 0.00587 | **0.03688** | 0.00556 |

## A.2 Additional NeRF Results at 200K Iterations

This appendix provides additional quantitative results for vanilla NeRF trained for 200K iterations in Table A.3.

Table A.3: **Additional 200K quantitative comparison on vanilla NeRF [25].** Scene-level PSNR and SSIM are reported on the LLFF and Synthetic NeRF datasets. The best result for each scene is highlighted in bold.

| Metric | Method | LLFF | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Fern | Flower | Fortress | Horns | Leaves | Orchids | Room | T-Rex | Avg. |
| PSNR ↑ | Adam [15] | 26.91 | 28.25 | 32.55 | 28.87 | 22.43 | 21.34 | 32.76 | 27.83 | 27.62 |
| | Muon [14] | 27.20 | 28.86 | 33.03 | 29.57 | 22.62 | 21.57 | 34.17 | 28.46 | 28.18 |
| | Ours | **27.42** | **28.96** | **33.17** | **29.66** | **22.72** | **21.72** | **34.21** | **28.92** | **28.35** |
| SSIM ↑ | Adam [15] | 0.8588 | 0.8879 | 0.9235 | 0.8978 | 0.8179 | 0.7509 | 0.9556 | 0.9208 | 0.8767 |
| | Muon [14] | 0.8665 | 0.8969 | 0.9332 | **0.9172** | 0.8135 | 0.7546 | **0.9666** | 0.9342 | 0.8853 |
| | Ours | **0.8746** | **0.9058** | **0.9357** | 0.9163 | **0.8308** | **0.7733** | 0.9649 | **0.9368** | **0.8923** |

| Metric | Method | Synthetic | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Chair | Drums | Ficus | Hotdog | Lego | Materials | Mic | Ship | Avg. |
| PSNR ↑ | Adam [15] | 33.97 | 25.55 | 28.99 | 36.69 | 31.65 | 29.35 | 33.18 | 29.22 | 31.08 |
| | Muon [14] | 34.30 | 25.67 | 29.19 | 37.20 | 31.93 | 29.85 | 33.46 | 29.49 | 31.39 |
| | Ours | **35.04** | **25.93** | **29.80** | **37.90** | **32.61** | **30.34** | **34.07** | **29.81** | **31.94** |
| SSIM ↑ | Adam [15] | 0.9775 | 0.9295 | 0.9655 | 0.9791 | 0.9651 | 0.9578 | 0.9785 | 0.8717 | 0.9531 |
| | Muon [14] | 0.9790 | 0.9320 | 0.9658 | 0.9823 | 0.9686 | 0.9623 | 0.9812 | 0.8842 | 0.9541 |
| | Ours | **0.9826** | **0.9346** | **0.9707** | **0.9839** | **0.9729** | **0.9660** | **0.9826** | **0.8846** | **0.9597** |

## B Schedule Hyperparameters

We set $T_g = 0.8T - T_0$ so that the rank saturates at 80% of training for any budget $T$, after which the learning rate decays linearly ($\Delta_{\text{sat}} = 0$). This ensures the rank schedule finishes before the learning rate anneals. All values were fixed a priori without per-experiment tuning.

Table B.1: **Rank schedule hyperparameters**, shared across all experiments.

| Symbol | Value | Role |
|---|---|---|
| $\rho$ | 0.90 | Energy threshold for rank initialization |
| $T_0$ | 8 | Initialization phase length |
| $T_g$ | $0.8T - T_0$ | Rank-growth phase length |
| $\Delta_{\text{sat}}$ | 0 | Delay before LR decay |

## C Rank Schedule Ablation over Training

We extend the rank schedule ablation in Table 8 by tracking PSNR during training. The increasing-rank schedule attains the highest mean PSNR at every checkpoint from 1,000 iterations onward. The margin over the next-best schedule stays at or below 0.22 dB until 2,000 iterations and grows to 1.80 dB at 3,000 iterations, which is consistent with the late increase in effective rank in Figure 5.

Table C.1: **PSNR on Kodak24 at intermediate checkpoints.** Mean $\pm$ std over 24 images, measured every 500 iterations on an NVIDIA RTX PRO 6000 GPU. All variants follow the setting of Table 8 except the rank trajectory. Best mean at each checkpoint in bold.

| Method | Iteration | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 500 | 1,000 | 1,500 | 2,000 | 2,500 | 3,000 |
| Decreasing rank | $7.34 \pm 1.77$ | $22.16 \pm 2.31$ | $22.69 \pm 2.80$ | $21.13 \pm 7.97$ | $21.37 \pm 5.34$ | $22.96 \pm 4.33$ | $25.53 \pm 6.45$ |
| Full rank | $7.34 \pm 1.77$ | $19.60 \pm 2.22$ | $20.48 \pm 5.08$ | $20.93 \pm 4.62$ | $21.57 \pm 5.00$ | $22.50 \pm 5.38$ | $25.22 \pm 7.08$ |
| Fixed rank | $7.34 \pm 1.77$ | $\mathbf{22.19 \pm 2.26}$ | $22.44 \pm 3.32$ | $22.51 \pm 3.21$ | $22.83 \pm 3.18$ | $22.70 \pm 5.33$ | $25.25 \pm 5.13$ |
| Increasing rank | $7.34 \pm 1.77$ | $22.14 \pm 2.30$ | $\mathbf{22.78 \pm 2.28}$ | $\mathbf{22.64 \pm 3.61}$ | $\mathbf{23.05 \pm 3.70}$ | $\mathbf{24.36 \pm 4.23}$ | $\mathbf{27.33 \pm 5.37}$ |

## D Hyperparameter Optimization Details

**TPE-based HPO Details.** We use Optuna's Tree-structured Parzen Estimator (TPE) for the NeRF HPO comparison. Each sampled configuration is trained for 100K iterations and evaluated by validation-set mean PSNR. All optimizers are tuned with 20 trials under the same search budget, and the final configuration is selected based on validation PSNR.

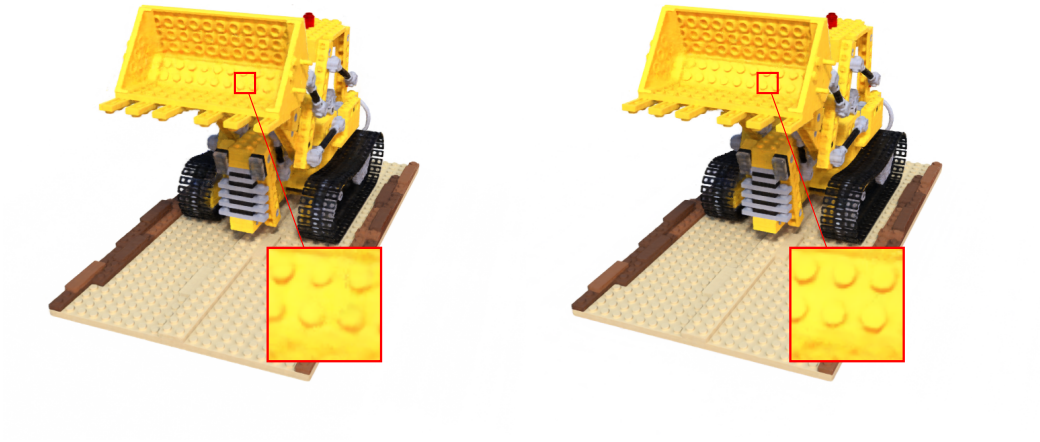
Table D.1: **TPE search spaces for NeRF HPO.** Each optimizer uses 20 trials with 100K training iterations per trial, and the best configuration is selected by validation PSNR.

| Method | Tuned hyperparameters | Search space | Sampling |
|---|---|---|---|
| Adam | Adam learning rate ($\eta_{\text{Adam}}$) | $\eta_{\text{Adam}} \in [3\text{e}{-4}, 3\text{e}{-3}]$ | log-uniform |
| | Learning-rate decay | Decay $\in [100, 500]$ | integer uniform |
| Muon | Adam branch learning rate ($\eta_{\text{Adam}}$) | $\eta_{\text{Adam}} \in [3\text{e}{-4}, 3\text{e}{-3}]$ | log-uniform |
| | Learning-rate decay | Decay $\in [100, 500]$ | integer uniform |
| | Muon learning rate ($\eta_{\text{Muon}}$) | $\eta_{\text{Muon}} \in [3\text{e}{-4}, 5\text{e}{-3}]$ | log-uniform |
| Ours | Adam branch learning rate ($\eta_{\text{Adam}}$) | $\eta_{\text{Adam}} \in [3\text{e}{-4}, 3\text{e}{-3}]$ | log-uniform |
| | Learning-rate decay | Decay $\in [100, 500]$ | integer uniform |
| | Projected-polar learning rate ($\eta_{\text{polar}}$) | $\eta_{\text{polar}} \in [3\text{e}{-4}, 5\text{e}{-3}]$ | log-uniform |

## E Additional Applications

### E.1 Model Fine-tuning

Figure E.1 examines NeRF fine-tuning on the Lego scene after the coarse reconstruction stage. The comparison targets the refinement stage, where optimizer updates must recover localized high-frequency residuals from an already recovered coarse structure. The matched crop centers on brick boundaries, which serve as a sensitive probe of local detail recovery. Muon yields smoother boundary transitions in the magnified region. The proposed optimizer preserves sharper edges and more regular brick contours. The proposed method preserves finer details more clearly than the baseline, even at a lower iteration count.



Muon Ours

Figure E.1: **Fine-tuning comparison on the Lego scene.** Zoomed crops highlight brick boundaries during refinement. The proposed optimizer recovers sharper details.

### E.2 Image denoising

Image denoising penalizes fitting high-frequency content, and we use it to test whether ProPolar overfits noise. Each INR fits an image corrupted by Poisson noise at $\tau = 30$ and Gaussian readout noise at an SNR of 2 dB, and PSNR is measured against the clean image. All methods share the architecture, iteration budget, and weight decay for each INR, and the full settings are provided in the released code.

Table E.1 shows that our method attains the highest PSNR on all four architectures. The gain over the best baseline ranges from 0.20 dB on WIRE to 1.20 dB on FINER and exceeds the seed-to-seed standard deviation of each architecture. This result suggests that the benefit of our method is not limited to tasks that reward fitting high-frequency detail.

Table E.1: **Image denoising.** PSNR against the clean image as mean $\pm$ std over 10 random seeds. Best per architecture in bold.

| **Model** | **PSNR** $\uparrow$ | | |
|---|---|---|---|
| | Adam [15] | Muon [14] | **Ours** |
| ReLU MLP | $22.04 \pm 0.21$ | $26.97 \pm 0.12$ | $\mathbf{28.09 \pm 0.09}$ |
| SIREN | $25.87 \pm 0.05$ | $27.06 \pm 0.05$ | $\mathbf{27.79 \pm 0.04}$ |
| WIRE | $28.56 \pm 0.06$ | $28.46 \pm 0.05$ | $\mathbf{28.76 \pm 0.05}$ |
| FINER | $28.24 \pm 0.06$ | $28.62 \pm 0.04$ | $\mathbf{29.82 \pm 0.05}$ |

# F Implementation of Progressive Rank and Rank-Aware Warmup-Stable-Decay Scheduler

**Algorithm 2** Code: Progressively Increasing Rank

```
def cosine_rank(step, rank_start, rank_end, rank_warmup):
    if rank_warmup <= 1:
        return rank_end
    if step <= 1:
        return rank_start
    if step >= rank_warmup:
        return rank_end

    tau = (step - 1) / (rank_warmup - 1)
    tau = 0.5 * (1 - cos(pi * tau))
    rank = rank_start + tau * (rank_end - rank_start)
    return round(rank)


def update_progressive_rank(group, grad=None):
    group.step += 1

    # Optional: estimate initial rank from early gradient energy.
    if group.auto_rank_start and group.step <= group.probe_steps:
        group.rank_start = EstimateRankStart(grad)

    rank = cosine_rank(
        group.step,
        group.rank_start,
        group.rank_end,
        group.rank_warmup
    )

    group.current_rank = max(group.rank_min, rank)
    group.target_rank = max(group.rank_min, group.rank_end)

    group.rank_saturated = group.current_rank >= group.target_rank
    group.rank_progress = Progress(
        group.current_rank,
        group.rank_start,
        group.target_rank
    )

    return group.current_rank
```

**Algorithm 3** Code: Rank-Aware Warmup-Stable-Decay Scheduler

```
def lr_ratio(step, warmup, decay_start, decay_end, min_ratio):
    if warmup > 0 and step < warmup:
        return (step + 1) / warmup

    if decay_start is None or step < decay_start:
        return 1.0

    if step >= decay_end:
        return min_ratio

    tau = (step - decay_start) / max(1, decay_end - decay_start)
    return 1.0 - (1.0 - min_ratio) * tau


def rank_aware_wsd(step, optimizer):
    status = ReadRankMetadata(optimizer)

    # Progressive rank: keep peak LR until rank saturation.
    if decay_window_is_unresolved():
        if status.has_progressive_rank:
            if status.rank_saturated:
                decay_start = step + patience
                decay_end = total_steps - 1
            else:
                decay_start = None
                decay_end = None

        # Full-rank or fixed-rank Muon does not wait for late decay.
        elif status.is_full_rank_or_fixed_rank:
            decay_start = round(full_start_frac * total_steps)
            decay_end = round(full_end_frac * total_steps)

        else:
            decay_start = round(default_start_frac * total_steps)
            decay_end = round(default_end_frac * total_steps)

    ratio = lr_ratio(
        step,
        warmup_steps,
        decay_start,
        decay_end,
        min_lr_ratio
    )

    lr_adam = base_lr_adam * ratio
    lr_muon = base_lr_muon * ratio

    return lr_adam, lr_muon, status
```

**Algorithm 4** Code: Progressive Rank Optimizer with Rank-Aware WSD

```
for step in range(total_steps):
    loss = model(batch)
    loss.backward()

    # Scheduler reads rank metadata from optimizer groups.
    lr_adam, lr_muon, status = rank_aware_wsd(step, optimizer)
    SetLearningRate(optimizer, lr_adam, lr_muon)

    for group in optimizer.muon_groups:
        rank = update_progressive_rank(group, grad=group.grad)
        update = LowRankMuonUpdate(group.grad, rank)
        group.params -= lr_muon * update

    for group in optimizer.adam_groups:
        update = AdamUpdate(group.grad)
        group.params -= lr_adam * update

    optimizer.zero_grad()
```

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are supported by prior work, experimental results, and analysis. Qualitative and quantitative results demonstrate the performance of the proposed method and reflect the scope of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the work. The scope of the claims is described in the method and algorithm sections, and limitations compared with other models are analyzed through experiments and reported in supplementary tables.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All theorems, formulas, and proofs are numbered and supported by relevant references. The claims of the proposed method are further supported by experiments and analysis.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The code and datasets used in the experiments will be publicly released. The experimental section describes the settings, and the hyperparameters are summarized in tables. Baseline execution details are provided in the supplemental material. The proposed method is described with equations and will be released as open-source code on GitHub.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides the exact commands and environment needed to reproduce the results. The release follows the NeurIPS code submission guidelines, and the code will be provided as open-source software on GitHub.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper includes the hyperparameters, hyperparameter selection procedure, and optimizer settings in the main text. Detailed settings are provided in the supplemental material and the released code to support reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper provides statistical information using the assumptions and ranges defined for the experiments. The analysis keeps outliers in the reported results and reports the experimental behavior without selective removal.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The supplemental material reports the compute resources used in the experiments, including memory usage and execution speed.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors reviewed the NeurIPS Code of Ethics and the research conforms to the guidelines. The submission preserves anonymity during the review process.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The work is foundational optimizer research and is not tied to a specific deployed application. The study uses no personal information and has no direct connection to high-risk decision-making systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer:[N/A]

Justification: The paper does not release pre-trained language models, image generators, scraped datasets, or other assets with high misuse risk. The proposed optimizer has low misuse risk.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All code and datasets used in the experiments are properly credited. The paper respects the licenses and terms of use of the existing assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The submitted materials provide the code and model in a structured form. The assets are anonymized for the review process.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The study does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The study does not involve crowdsourcing or research with human subjects, so IRB approval or an equivalent review is not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for writing assistance and grammar checking. LLMs were not used as an important, original, or non-standard component of the core method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.