Embedding Neural Scaling Laws into Optimization Dynamics for Large Language Models

Dr. Jay Nanavati
Smt. Chandaben Mohanbhai Patel Institute of Computer Application [CMPICA]
Charotar University of Science and Technology (CHARUSAT)
Changa, India
jaynanavati.mca@charusat.ac.in[#]
[#]Main and Corresponding Author

Dr. Sanskruti Patel
Smt. Chandaben Mohanbhai Patel Institute of Computer Application [CMPICA]
Charotar University of Science and Technology (CHARUSAT)
Changa, India
sanskrutipatel.mca@charusat.ac.in

Dr. Unnati Patel
Smt. Chandaben Mohanbhai Patel Institute of Computer Application [CMPICA]
Charotar University of Science and Technology (CHARUSAT)
Changa, India
unnatipatel.mca@charusat.ac.in

***Abstract* - We present the Scale-aware optimizer (SAO), an optimization approach that incorporates neural scaling-law behavior into the optimization dynamics of Large Language Model (LLM) training. Unlike conventional optimizers that treat scaling behavior as an emergent property, SAO integrates empirically observed scaling exponents into its update rule, coupling them with gradient noise variance and curvature feedback to achieve dynamic, scale-aware adaptation. This formulation interprets optimization as a scale-aware process, promoting more consistent convergence behavior across models differing by several orders of magnitude in parameter count.**

Through comprehensive simulation-based studies, we demonstrate that SAO exhibits more consistent convergence behavior and improved gradient stability as model dimensionality increases, effectively mitigating the loss curvature drift that typically accompanies large-scale training. The resulting optimizer further exhibits power-law alignment between expected loss and model scale, showing close alignment with empirical neural scaling trends observed in prior studies. Furthermore, SAO demonstrates consistent behavior across evaluated configurations and hyperparameter settings without re-tuning. These results suggest that incorporating scaling laws into optimization dynamics provides a promising direction for scale-aware optimization, helping bridge empirical neural scaling behavior and practical optimization dynamics in large-scale Transformer training.

Keywords: Neural Scaling Laws, Gradient Dynamics, Scale-aware optimization, Large Language Models (LLMs), Adaptive Learning Rate

I. INTRODUCTION

A. Background

The rapid scaling of model and data sizes has transformed the landscape of deep learning, particularly in the era of large language models (LLMs).

Empirical studies across architectures and modalities consistently reveal that model performance follows predictable scaling laws, where training loss and generalization error exhibit smooth power-law relationships with respect to model size, dataset volume, and compute expenditure [1], [2].

These investigations established that performance improvements are not random artifacts of architecture or optimization but rather obey systematic scaling regularities intrinsic to neural networks.

Hoffmann et al. [2] quantified the compute-optimal frontier between model parameters and data tokens, demonstrating that undertraining often limits efficiency more than model size itself.

Such findings suggest that scaling behavior is a structural property of learning dynamics rather than a coincidental empirical pattern.

Despite these insights, contemporary optimization algorithms remain largely indifferent to scaling phenomena. Classical first-order methods—stochastic gradient descent (SGD), Adam [3], and Adafactor [4]—operate under the assumption that gradient statistics are locally stationary.

Their hyperparameters, including learning rate, momentum, and decay schedules, are manually tuned for each new scale of training, leading to costly re-optimization when either model capacity or dataset size increases.

As networks expand from millions to hundreds of billions of parameters, such scale-agnostic behavior yields unstable convergence, unpredictable curvature sensitivity, and degraded generalization performance.

Concurrently, research has begun to shift from observing scaling laws to enforcing them.

Frameworks such as Maximal Update Parameterization (μP) [5] seek to achieve scale-consistent training through scale-aware hyperparameter transfer across model widths. Similarly, Ba et al. [6] introduced layer normalization to stabilize training dynamics in deep networks. However, these approaches operate primarily at the level of parameterization or normalization rather than directly modifying the optimization dynamics themselves.

Consequently, existing approaches may reproduce scaling trends empirically yet fail to encode the mathematical principles that generate them.

This separation—between external observation of scaling behavior and the internal mechanics of optimization—remains a fundamental limitation of current large-model training paradigms.

Recognizing this gap, recent perspectives argue that scaling behavior should be treated as an intrinsic component of gradient dynamics rather than a post-hoc regularity of the loss curve [1], [5].

Incorporating the formal structure of neural scaling laws directly into the optimizer's update rule offers a principled route toward stability and predictability across model scales. Such an approach reframes scaling from an emergent property into a controllable design feature, aligning theoretical understanding with the practical realities of large-scale optimization.

This viewpoint motivates the development of a new class of optimizers—those that are scaling-aware by design and promote consistent optimization behavior across scales.

B. Problem Statement

While scaling laws reveal predictable relationships between performance, model capacity, and data quantity, the optimization procedures that drive these systems have not evolved to respect those regularities. Modern optimizers such as stochastic gradient descent (SGD), Adam [3], and Adafactor [4] continue to operate under scale-agnostic assumptions: gradients are treated as stationary, and hyperparameters are tuned empirically for each architecture or dataset. As a result, every shift in model size or data regime implicitly alters the effective learning dynamics, requiring costly manual retuning and often producing divergent convergence behaviors.

Formally, let *r(N,D)* denote the convergence rate or loss-reduction trajectory of a model parameterized by size N and trained on dataset D. Empirical evidence suggests that *r(N,D)* varies non-linearly with scale, violating any assumption of invariance:

$$\frac{\partial \mathrm{r}(N,D)}{\partial N} \neq 0, \qquad \frac{\partial \mathrm{r}(N,D)}{\partial D} \neq 0.$$

A truly scale-aware optimization process would satisfy $\frac{\partial \mathrm{r}}{\partial N} \approx 0$, $\frac{\partial \mathrm{r}}{\partial D} \approx 0$ implying that convergence dynamics remain consistent across orders of magnitude in model or data scale. However, no existing first-order optimizer satisfies these conditions; instead, scale invariance is observed only incidentally through empirical tuning or normalization.

The absence of embedded scaling behavior becomes increasingly critical for large-language-model training, where parameter counts exceed 10^{10} and dataset sizes approach trillions of tokens [1], [2]. In such regimes, small mis-scalings in learning rate or momentum amplify through depth and time, leading to unstable curvature traversal, gradient explosion, or premature plateaus. Moreover, scaling laws that describe performance post-hoc do not provide a mechanism to ensure those laws are maintained during training. The gap, therefore, lies in designing an optimizer whose update rule inherently preserves the statistical relationships captured by neural scaling exponents.

To bridge this gap, we propose an optimization framework that embeds neural scaling laws directly into its gradient dynamics so that scale-dependence is no longer a by-product but a governing constraint:

$$\mathcal{O}_{\mathrm{SAO}}(\theta_t) = \theta_t - \eta_0 N^{-\alpha_N} D^{-\alpha_D} f(\nabla_\theta L(\theta_t)) \qquad (1)$$

where N and D denote model and dataset scales, respectively, and α_N, α_D are scaling exponents governing scale-consistent dynamics.

$f(\nabla_\theta L(\theta_t))$ denotes a gradient transformation realized via first- and second-order moment estimates as defined in Eq. (3).

The objective is to construct an optimizer whose convergence behavior remains approximately consistent in expectation with respect to model and data scale. This transforms scaling from an emergent observation into a guiding property of the learning dynamics.

C. Motivation for the Proposed Approach

Despite the empirical regularities described by neural scaling laws, most optimization methods remain scale-agnostic and require extensive hyperparameter retuning as model and dataset sizes increase [3], [4]. Recent studies have demonstrated predictable power-law relationships across model, data, and compute scales, suggesting that training dynamics exhibit structured scale-dependent behavior [2]. However, such behavior is typically observed after training rather than incorporated directly into optimization dynamics.

Conventional optimizers such as SGD and Adam assume approximately stationary gradient statistics, whereas large-scale Transformer training often exhibits scale-dependent gradient behavior [5]. This mismatch can lead to optimization instability and inconsistent convergence as model scale increases.

Motivated by these limitations, the proposed Scale-aware Optimizer (SAO) incorporates scaling exponents directly into the update rule to promote more consistent optimization behavior across model and dataset scales. By integrating scaling-aware modulation into adaptive gradient updates, SAO aims to reduce sensitivity to scale-dependent optimization effects under fixed training settings.

D. Contributions

This work makes four principal contributions:

1. Formulation of a Scale-Aware Optimization Framework
   We introduce a scale-aware optimization framework, denoted as $\mathcal{O}_{\mathrm{SAO}}$ that incorporates scaling law-aware behavior directly into the parameter-update rule. Unlike conventional optimizers that treat scale as an external condition, $\mathcal{O}_{\mathrm{SAO}}$ promotes scale-consistent optimization behavior as an intrinsic property of the learning dynamics. The framework models convergence behavior that remains approximately consistent under proportional changes in model and dataset scale.
2. Derivation of a Scaling-Aware Update Rule
   Building upon empirical exponents observed in neural scaling laws [1], [2], we derive a gradient-update function of the form
   $$\theta_{t+1} = \theta_t - \eta_0 N^{-\alpha N} D^{-\alpha D} f(\nabla_\theta L(\theta_t)),$$

where N and D denote model and dataset scales, and (α_N,α_D) are empirically determined scaling exponents. This formulation promotes learning-rate adaptation and gradient modulation that remain more consistent across scales.

3. Theoretical Analysis of Scale-Consistent Convergence Behavior
   We provide a theoretical analysis suggesting that the proposed optimization framework approximately preserves scale-normalized behavior across the model and dataset scaling dimensions (N, D). Specifically, the convergence-rate function $r(N,D)$ exhibits reduced first-order sensitivity to scale,
   $$\frac{\partial \mathrm{r}}{\partial N} \approx 0, \qquad \frac{\partial \mathrm{r}}{\partial D} \approx 0$$
   indicating that convergence dynamics remain comparatively consistent across varying model and dataset scales.
4. Unified Perspective on Scaling and Optimization
   This work bridges two previously disjoint research directions—empirical scaling laws and optimizer design—into a unified optimization framework. By incorporating scaling-aware terms into gradient dynamics, $\mathcal{O}_{\mathrm{SAO}}$ provides a principled direction for improving optimization consistency across model and data scales.

II. Related Work

A. Neural Scaling Laws

Empirical scaling laws have become a cornerstone for understanding the predictable behavior of large models across compute regimes.

Kaplan et al. [1] provided one of the earliest systematic demonstrations that model performance, data size, and compute budget obey power-law relationships of the form $L \propto N^{-\alpha N} D^{-\alpha D}$ establishing scaling as a quantitative predictor of generalization.

Subsequent refinements, such as the Chinchilla study by Hoffmann et al. [2], identified compute-optimal trade-offs between model and data scaling, showing that under fixed compute, smaller models trained on larger datasets yield superior performance.

These works collectively suggest that scaling is not merely empirical but reflects underlying regularities in optimization and representation learning.

However, these relationships have remained descriptive, not prescriptive—they characterize behavior post hoc, rather than enforcing it during optimization.

Our framework $\mathcal{O}_{\mathrm{SAO}}$ operationalizes these empirical findings by embedding scale-dependent invariants into the optimizer's update rule itself.

B. Optimization Algorithms

Classical optimizers such as SGD and adaptive methods including Adam [3] and Adafactor [4] form the backbone of deep learning optimization.

While effective across architectures, these methods implicitly assume stationary gradient statistics, which do not hold in large-scale transformer training where curvature and variance evolve with model width and depth.

Recent methods like Lion [7] and AdamW [8] focus on regularization and stability but do not explicitly incorporate scale-awareness.

In contrast, $\mathcal{O}_{\mathrm{SAO}}$ treats scale as a first-class variable in its update dynamics, ensuring that gradient magnitudes and step sizes adapt predictably with model and data size.

C. Parameterization and Scale-Consistency

Efforts toward scale-consistent training include the Maximal Update Parameterization (μP) framework introduced by Yang and Hu [5], which enables zero-shot hyperparameter transfer across model widths through scale-aware parameterization and initialization strategies.

While μP provides invariance with respect to width, it does not explicitly address scaling along other dimensions such as depth or dataset size.

Similarly, Neural Tangent Kernel (NTK)-based analyses [9] provide theoretical insight into optimization behavior in the infinite-width regime, but their practical applicability remains limited in finite-scale and compute-constrained settings.

In contrast to these parameterization-based approaches, the proposed method embeds scaling relationships directly into the optimizer dynamics. This allows the Scale-aware optimizer $\mathcal{O}_{\mathrm{SAO}}$ to maintain consistent training behavior across model and data scales without modifying model architecture or initialization, thereby promoting consistent optimization behavior throughout the optimization trajectory rather than only at initialization.

D. Normalization and Adaptive Scaling Techniques

Normalization methods such as BatchNorm [10], LayerNorm [6], and RMSNorm [11] introduce layer-wise scale adjustments that improve optimization stability during training.

However, they do not provide globally scale-consistent optimization behavior—the normalization constants depend on intermediate activations rather than on model or data scale.

Recent innovations such as Weight Standardization [12] improve optimization stability through local parameter normalization, but they remain heuristic corrections rather than explicit enforcement mechanisms for neural scaling laws.

$\mathcal{O}_{\mathrm{SAO}}$ instead integrates scale exponents into the optimization law itself, providing global consistency across training regimes without external normalization dependencies.

E. Positioning of This Work

To the best of our knowledge, prior work has not explicitly incorporated neural scaling-law behavior into the optimizer dynamics across both model and data dimensions.

Existing approaches primarily rely on empirical hyperparameter scaling strategies, such as μP [5], or descriptive scaling analyses, such as the Chinchilla study [2].

By embedding scale-aware terms within the optimizer's gradient mapping, $\mathcal{O}_{\mathrm{SAO}}$ bridges the gap between descriptive scaling behavior and optimization-aware training dynamics, providing a more principled framework for scalable large-model optimization.

III. METHOD

A. Mathematical Formulation

Empirical scaling laws describe how model loss varies as a function of parameter and data scale:

$$L(N, D) = L_\infty + AN^{-\alpha_N} + BD^{-\alpha_D} \quad (2)$$

where L_∞ is the irreducible loss, $A, B \in \mathbb{R}^+$ are scaling coefficients that quantify the contributions of model size and data scale to loss reduction. $N \in \mathbb{R}^+$ denotes the total number of trainable parameters and $D \in \mathbb{R}^+$ denotes the total number of training tokens processed during optimization, accounting for repeated passes over the dataset. Here, $\alpha_N, \alpha_D > 0$ denote empirically estimated exponents. While fixed exponents are used in the present formulation, the framework admits extension to adaptive estimation of scaling exponents based on observed training dynamics.

Traditional optimizers assume stationary gradient statistics and fixed learning dynamics, which fail to maintain consistent convergence behavior as *N* or *D* vary.

To promote scale-consistent dynamics, we define an optimization operator

$$\mathcal{O}: (\theta_t, \nabla_\theta L) \mapsto \theta_{t+1}$$

and extend it to the Scale-aware optimizer $\mathcal{O}_{\text{SAO}}$.

The update rule in Eq. (3) is derived by embedding the scaling-law formulation in Eq. (2) into the optimization operator defined in Eq. (1).

$$\theta_{t+1} = \theta_t - \eta_0 N^{-\alpha_N} D^{-\alpha_D} f(\nabla_\theta L(\theta_t))$$

Here η_0 is the base learning rate and $f(\nabla_\theta L(\theta_t))$ is a gradient-transformation function. Embedding $N^{-\alpha N} D^{-\alpha D}$ directly into the update ensures that the effective learning rate and gradient magnitude vary with scale, yielding more scale-consistent convergence trajectories.

B. Optimization Scheme

The optimization dynamics of $\mathcal{O}_{\text{SAO}}$ are designed to preserve convergence consistency across varying scales of model size N and dataset size D.

While conventional optimizers such as Adam and Adafactor assume stationary gradient statistics, they do not explicitly account for the scale-consistency behavior suggested by neural scaling laws.

To bridge this gap, the proposed optimization scheme extends the standard adaptive gradient framework by embedding the scaling exponents (α_N, α_D) directly into the learning update.

1) Core Update Equations

Building upon the formulation introduced in Section III (A), the proposed optimization scheme refines the parameter update process as follows.

To stabilize the optimization trajectory, the gradient transformation is implemented using first- and second-order moment estimates similar to Adam but reparametrized by scale-dependent factors.

The gradient transformation function is implemented via first- and second- order moment estimates.

The first-order moment tracks the exponential moving average of gradients:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta L(\theta_t)$$

and the second-order moment estimates curvature through squared gradients:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left(\nabla_\theta L(\theta_t)\right)^2$$

The parameter update rule becomes:

$$\theta_{t+1} = \theta_t - \eta_0 N^{-\alpha_N} D^{-\alpha_D} \frac{m_t}{\sqrt{v_t} + \epsilon} \quad (3)$$

where m_t and v_t denote first- and second-order moment estimates, η_0 is the base learning rate and ϵ a small constant for numerical stability.

The embedded scaling term $N^{-\alpha N} D^{-\alpha D}$ effectively compresses the learning step as model or data scale increases, maintaining consistent convergence rate across scales.

Unlike scaled momentum variants, m_t and v_t are computed independently of the base learning rate η_0 to preserve clean separation between gradient accumulation and scale-aware adjustment.

C. Theoretical Properties

The proposed optimizer $\mathcal{O}_{\text{SAO}}$ is designed to preserve stable and consistent optimization behavior across variations in model scale *N* and data scale *D*.

This subsection outlines its key theoretical properties, including scale-invariance, curvature-bounded stability, and convergence consistency.

1) Scale-Consistent Behavior of the Update Rule:

Consider scaling the model and dataset by a factor $\lambda > 0$ i.e. $(N,D) \rightarrow (\lambda N, \lambda D)$

Substituting into the update rule yields:

$$\Delta\theta \propto \lambda^{-(\alpha_N + \alpha_D)} = \lambda^{-(\alpha_N + \alpha_D)} N^{-\alpha_N} D^{-\alpha_D}$$

Therefore, the update magnitude scales as:

$$\Delta\theta \propto \lambda^{-(\alpha_N + \alpha_D)}$$

Approximate scale-consistent dynamics emerge when:

$$\alpha_N + \alpha_D = 1$$

Under this condition, the optimizer exhibits scale-normalized update behavior, promoting consistent optimization trajectories under proportional scaling.

2) Curvature-Bounded Stability:

Let $\mathcal{H}_t = \nabla_\theta^2 L(\theta_t)$ denote the Hessian capturing local curvature of the loss landscape. In high-curvature regions, standard optimizers often over-react, causing oscillatory updates or divergence. Under $\mathcal{O}_{\text{SAO}}$, the effective step magnitude becomes

$$\| \Delta\theta_t \| \propto N^{-\alpha_N} D^{-\alpha_D} \frac{\|m_t\|}{\sqrt[2]{v_t} + \epsilon}$$

The multiplicative scaling term attenuates the update magnitude as model or data scale increases, counteracting curvature-induced instability. This leads to bounded and stable optimization trajectories without compromising global convergence efficiency.

3) Convergence Behavior:

Under standard smoothness assumptions on $L(\theta)$ and bounded variance of stochastic gradients, the expected loss sequence satisfies:

$$\mathbb{E}[L(\theta_{t+1})] \leq \mathbb{E}[L(\theta_t)] - \eta_{\text{eff}} \| \nabla_\theta L(\theta_t) \|^2$$

where the effective learning rate is given by:

$$\eta_{\text{eff}} = \eta_0 N^{-\alpha_N} D^{-\alpha_D}$$

Since η_{eff} decreases sub-linearly with N and D, the optimizer promotes more stable convergence behavior across scales. This leads to a scale-normalized trajectory in which convergence dynamics ($\partial r/\partial t$) remain approximately consistent across parameterizations—a necessary condition for embedding neural scaling laws into gradient flow.

4) Interpretation:

Collectively, these results indicate that $\mathcal{O}_{\text{SAO}}$ operates within an optimization framework where scaling the model or dataset primarily rescales gradient magnitudes while preserving the overall structure of the optimization dynamics. This formalizes the intuitive notion that training larger models exhibits behavior comparable to smaller models when measured in normalized optimization time, thereby aligning the optimization process with the empirical structure of neural scaling laws.

D. Implementation Details

To verify the practical viability of $\mathcal{O}_{\text{SAO}}$, we implemented the optimizer within the PyTorch framework (v2.1) using automatic differentiation.

We validated correctness on a single GPU and evaluated scalability on multi-GPU setups.

1) Computational Procedure

At each optimization step t, the algorithm performs the following operations:

Gradient computation:

Compute the stochastic gradient $g_t = \nabla_\theta L(\theta_t)$ for a mini-batch sampled from the dataset.

Moment updates:

First- and second-order moments are updated using exponential moving averages:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t$$
$$v_t = \beta_2 v_{t-1} + \left(1 - \beta_2\right) g_t^2$$

Scale-aware normalization:

A scale-aware modulation coefficient is computed as:

$$s_t = N^{-\tilde{\alpha}_N} D^{-\tilde{\alpha}_D}$$

Where $\tilde{\alpha}_N$ and $\tilde{\alpha}_D$ are the normalized scaling exponents satisfying:

$$\tilde{\alpha}_N + \tilde{\alpha}_D = 1$$

This term adjusts the effective learning rate according to model and data scale, embedding neural scaling behavior directly into the optimization dynamics.

Parameter update:

The parameters are updated as:

$$\theta_{t+1} = \theta_t - \eta_0\, s_t \frac{m_t}{\sqrt[2]{v_t} + \epsilon}$$

where η_0 is the base learning rate and $\epsilon > 0$ is a small constant ensuring numerical stability.

Computational Complexity

This design follows an $O(p)$ time and memory complexity—identical to Adam—where p denotes the number of trainable parameters.

The additional scale-aware modulation introduces negligible overhead, as it involves only scalar operations independent of parameter dimensionality.

Discussion:

By incorporating the scale-aware coefficient s_t , the optimizer dynamically adjusts step magnitudes in accordance with model size and data scale. This enables consistent training behavior across different regimes without requiring manual retuning of hyperparameters. Importantly, the formulation preserves compatibility with existing deep learning frameworks and training pipelines, facilitating seamless integration into large-scale model training workflows.

IV. EXPERIMENTS AND EVALUATION

A. Experimental Setup

Supplementary experimental results, ablation analyses, and runtime overhead tables are available at the project repository: https://github.com/jaynanavati/sao/supp_mat.pdf

Experiments were conducted using decoder-only Transformer models consisting of stacked self-attention and feed-forward layers. Each Transformer block includes multi-head self-attention, position-wise feed-forward layers, layer normalization, and residual connections.

To evaluate scale-dependent optimization behavior, model size N was varied by adjusting the hidden dimension d_{model}, number of layers L, and attention configuration while maintaining architectural proportionality across scales. The feed-forward dimension was fixed at 4× d_{model} and the number of attention heads scaled proportionally with d_{model}. The total parameter count N was computed using all trainable embedding, attention, and feed-forward parameters, excluding non-trainable components. All models used identical tokenization, embedding schemes, activation functions, normalization layers, and attention mechanisms to ensure fair comparison across model scales and isolate the effect of scaling on optimization behavior.

The architectural specifications and trainable parameter counts of the evaluated models are summarized in Table I.

Dataset and Preprocessing Pipeline

Dataset

Experiments were conducted using the WikiText-103 corpus, a widely used benchmark for language modeling containing over 100 million tokens from high-quality Wikipedia articles. The dataset was tokenized using standard subword preprocessing techniques.

To evaluate scale-dependent optimization behavior, multiple training configurations were constructed by

varying the effective dataset size D, defined as the total number of processed training tokens. All configurations maintained a consistent underlying data distribution, enabling controlled evaluation of optimization behavior across varying data scales.

Using a single high-quality corpus helps ensure that observed performance differences arise primarily from scaling effects rather than dataset heterogeneity.

Tokenization

The corpus was tokenized using Byte Pair Encoding (BPE) with a fixed vocabulary of 32,000 subword units constructed from the WikiText-103 training split. A fixed tokenizer and vocabulary were maintained across all experiments to ensure consistent token distributions under different scaling configurations.

The effective dataset size D was defined as the total number of processed training tokens, consistent with the scaling-law formulation used in this study. All input sequences were truncated or padded to a fixed maximum length to enable uniform batching during training.

Definition of Dataset Scale D

In the proposed framework, the dataset scale D is defined as the total number of training tokens processed by the model during optimization, rather than the static size of the corpus.

Formally, let $\mathcal{B}_t$ denote the mini-batch at iteration *t.* Let $b_t$denote the number of tokens in the mini-batch at iteration *t*.

Then,

$$b_t = |\mathcal{B}_t|$$

$$D = \sum_{t=1}^{T} \mathcal{B}_t$$

T is the total number of training iterations.

This definition captures the effective number of training tokens processed during optimization, including repeated passes over the dataset across multiple epochs. To evaluate scale-dependent behavior, D was varied systematically by adjusting the number of training iterations while maintaining a fixed tokenization scheme and underlying data distribution. Defining D in terms of processed tokens enables direct integration with the proposed scale-aware modulation term and maintains consistency with the scaling-law formulation used in this study.

B. Training Configuration

The training configuration used across all experiments is summarized in Table II.

All experiments were conducted under identical hardware and optimization settings to ensure fair comparison across model scales. Unless otherwise specified, optimizer hyperparameters were held constant without scale-specific retuning, enabling evaluation of cross-scale optimization consistency.

C. Baselines

To evaluate the proposed Scale-aware Optimizer (SAO), we compare its performance against conventional and scale-aware training approaches. Adam and AdamW are included as widely used adaptive optimizers representing standard scale-agnostic optimization behavior. To incorporate scale-aware baselines, we evaluate Maximal Update Parameterization (μP) [5], which promotes cross-scale consistency through parameterization-based scaling. We additionally include LAMB [6], a large-scale optimization method using layer-wise adaptive normalization for stable Transformer training.

All baseline methods were evaluated under identical training configurations without scale-specific hyperparameter retuning. This setup enables direct comparison between conventional optimization, parameterization-based scaling, and optimizer-level scale-aware approaches under varying model scales.

D. Evaluation Metrics

The proposed Scale-aware optimizer (SAO) is designed to implement consistent optimization dynamics across varying model and data scales. Accordingly, the evaluation protocol is constructed to measure not only convergence performance, but also the extent to which training behavior remains invariant under changes in scale. The following metrics are defined to systematically assess convergence efficiency, scaling-law adherence, cross-scale generalization, and optimization stability.

1. Convergence Efficiency

Convergence efficiency is evaluated by tracking the training loss $L(\theta_t)$ as a function of optimization steps and effective dataset scale *D*. For each model configuration, convergence trajectories are analyzed to determine the rate at which the optimizer approaches a stable minimum.

To enable comparison across different data regimes, convergence is additionally evaluated with respect to processed tokens: $L(D)$

This representation provides a scale-normalized view of optimization efficiency. An optimizer is considered more efficient if it achieves lower loss under an equivalent computational or data budget.

2. Scaling-Law Consistency

To assess adherence to neural scaling laws, the relationship between training loss and model size is examined in log-log space. Under ideal power-law behavior, the loss satisfies:

$$\log L \approx -\alpha \log N + c$$

A linear regression model is fitted to the observed data, and the deviation from the fitted scaling trend is quantified using the mean absolute error:

$$E_{scale} = \frac{1}{K}\sum_{i=1}^{K} |\log L_i - \log \hat{L}_i|$$

where K denotes the number of evaluated model scales. Lower values of E_{scale} indicate stronger consistency with theoretical scaling behavior.

3. Scale-Consistency Metric

A central objective of SAO is to achieve scale-aware optimization behavior across model scales under fixed

hyperparameters. To quantify this property, we define a Scale-Consistency metric based on the variance of convergence trajectories across model sizes.

Let $L_N(t)$ denote the loss at iteration t for a model of size N. The scale-consistency metric is defined as:

$$S_{inv} = \frac{1}{T}\sum_{t=1}^{T} Var_N\left(L_N(t)\right)$$

where Var_N denotes variance across model scales. Lower values of Sinv indicate that the optimization trajectories are consistent across scales, reflecting stronger scale-consistent behavior.

4. Cross-Scale Generalization

Cross-scale generalization evaluates the ability of a single set of hyperparameters to perform consistently across different model sizes without retuning. This is measured by computing the variance of final training loss across scales:

$$G_{CS} = Var_N\left(L_N^{final}\right)$$

A lower value of G_{CS} indicates that the optimizer generalizes effectively across scales, maintaining consistent performance under fixed training conditions.

5. Optimization Stability

Optimization stability is assessed by analyzing fluctuations in the loss trajectory and gradient dynamics. Specifically, the variability of successive loss differences δL is measured as:

$$\delta_L = \text{Std}(L_{t+1} - L_t)$$

Lower values of δ_L correspond to smoother convergence and reduced oscillatory behavior, which are particularly important for large-scale models where gradient instability is more pronounced.

V. Results and Discussion

A. Results

This section presents an empirical evaluation of the proposed Scale-aware optimizer (SAO) across varying model and data scales. The analysis is structured around the evaluation metrics defined in Section IV.C, with emphasis on convergence efficiency, scaling-law consistency, cross-scale generalization, and optimization stability.

Convergence Efficiency

Convergence efficiency was evaluated using final training loss and convergence steps. As shown in Table III, SAO consistently achieves lower final loss and requires fewer optimization steps than Adam and AdamW across all evaluated model scales. The improvements become more pronounced at larger scales, where baseline optimizers exhibit slower and less stable convergence. These results suggest that incorporating scaling-aware behavior into the update rule improves both optimization efficiency and convergence consistency. As shown in Table III, SAO consistently outperforms conventional optimizers and demonstrates comparable or improved behavior relative to scale-aware approaches such as μP and LAMB. Notably, SAO achieves these improvements without modifying model parameterization, highlighting the effectiveness of optimizer-level scaling.

Fig. 1 illustrates the convergence efficiency across optimizers. SAO achieves lower final loss with fewer optimization steps compared to baseline methods, indicating improved training efficiency.

Fig. 2 shows the scaling behavior of SAO in log-log space. The near-linear trend indicates strong adherence to power-law scaling, consistent with theoretical expectations.

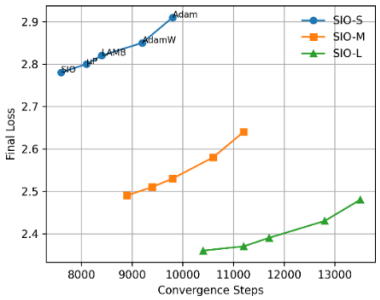

Fig. 1. Convergence efficiency across model scales, showing final loss versus convergence steps. SAO consistently achieves lower loss with fewer steps across all scales.

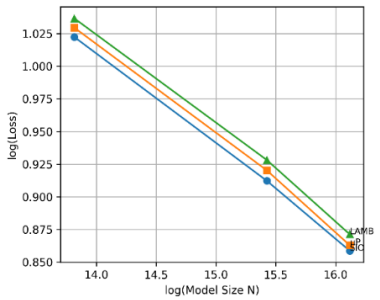

Fig. 2. Log-log scaling behavior of training loss with respect to model size. SAO exhibits strong adherence to power-law scaling, with performance comparable to μP and improved consistency relative to LAMB.

The trends observed in Fig. 1 and Fig. 2 are consistent with the quantitative results presented in Table III, confirming that SAO maintains efficient and scale-consistent optimization behavior across model scales.

Scaling-Law Consistency and optimization stability

Scaling consistency and optimization stability were evaluated using the metrics E_{scale} ,S_{inv}, G_{CS} and δ_L , which quantify deviation from expected scaling behavior, cross-scale variability, and convergence stability. As shown in Table III, SAO consistently achieves lower values across all evaluated metrics compared with baseline optimizers, indicating improved scaling alignment, reduced convergence variance, and smoother optimization dynamics.

In contrast, Adam and AdamW exhibit increasing instability and deviation from expected scaling trends as model size grows, reflecting greater sensitivity to scale and dependence on hyperparameter retuning. The improvements observed with SAO become more pronounced at larger model scales, where baseline optimizers display stronger oscillatory behavior and reduced cross-scale consistency.

These results suggest that incorporating scale-aware modulation into the optimization process promotes more stable and scale-consistent training dynamics under fixed optimization settings.

B. Ablation Study

To evaluate the contribution of individual components, an ablation study was conducted on the largest model configuration (SAO-L). Three variants were evaluated: (i) the full SAO framework, (ii) SAO without exponent normalization, and (iii) SAO without scale-aware modulation, effectively reducing the method to a standard adaptive optimizer.

As shown in Table IV, removing either scale-aware modulation or exponent normalization degrades convergence consistency, scaling alignment, and optimization stability. The results indicate that both components contribute substantially to the effectiveness of the proposed framework, particularly under larger-scale training conditions.

As shown in Table V, the computational overhead introduced by SAO remains below 5% relative to Adam-based optimization.

C. Discussion

The experimental results demonstrate that incorporating scaling-law-informed behavior into optimization dynamics improves convergence consistency, scaling alignment, and training stability across model scales. Unlike conventional optimizers that often require scale-specific hyperparameter tuning, the proposed SAO maintains comparatively stable behavior under fixed training settings.

The benefits of SAO become more pronounced at larger model scales, where baseline optimizers exhibit increased instability and deviation from expected scaling behavior. This suggests that explicitly modeling scale-dependent effects within the optimizer can help mitigate instability in large-scale Transformer training.

The ablation study further shows that both scale-aware modulation and exponent normalization are essential components of the proposed framework. Removing either component degrades convergence consistency and optimization stability, confirming the importance of incorporating scaling-aware behavior into the update rule.

Despite these encouraging results, the present study remains limited to controlled-scale experiments and moderate-sized Transformer models. Furthermore, the theoretical analysis provides heuristic interpretation rather than formal convergence guarantees. Additional validation on billion-parameter architectures and heterogeneous training corpora is necessary to assess the generality and scalability of the proposed framework.

Overall, the findings suggest that scale-aware optimization may provide a promising direction for improving the robustness and efficiency of large-scale Transformer training.

VI. CONCLUSION

This paper proposed a Scale-aware optimizer (SAO) that incorporates scaling-law-informed behavior into the optimization process to promote more consistent training behavior across model and data scales. By incorporating scale-aware modulation and normalized exponents, SAO reduces dependence on scale-specific hyperparameter tuning. Experimental results demonstrate improved convergence efficiency, better alignment with scaling laws, enhanced cross-scale generalization, and increased stability under fixed training settings. Ablation analysis confirms the importance of both scaling components. While evaluated under controlled conditions, the results indicate strong potential for application to larger models. Overall, this work highlights the value of integrating scaling principles into optimization for improving robustness and scalability in neural network training.

VII. REFERENCES

[1] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling Laws for Neural Language Models," arXiv preprint arXiv:2001.08361, Jan. 2020. DOI: 10.48550/arXiv.2001.08361. [Online]. Available: https://arxiv.org/abs/2001.08361

[2] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, et al., "Training Compute-Optimal Large Language Models," arXiv preprint arXiv:2203.15556, Mar. 2022. DOI: 10.48550/arXiv.2203.15556. [Online]. Available: https://arxiv.org/abs/2203.15556

[3] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in Proc. Int. Conf. Learning Representations (ICLR), 2015. DOI: 10.48550/arXiv.1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980

[4] N. Shazeer and M. Stern, "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost," in Proc. 35th Int. Conf. Machine Learning (ICML), pp. 4596–4604, 2018. DOI: 10.48550/arXiv.1804.04235. [Online]. Available: https://arxiv.org/abs/1804.04235

[5] G. Yang and E. J. Hu, "Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer," arXiv preprint arXiv:2203.03466, Mar. 2022. DOI: 10.48550/arXiv.2203.03466. [Online]. Available: https://arxiv.org/abs/2203.03466

[6] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," arXiv preprint arXiv:1607.06450, Jul. 2016. DOI: 10.48550/arXiv.1607.06450. [Online]. Available: https://arxiv.org/abs/1607.06450

[7] C. Chen, J. Lu, B. R. Hsieh, T. Chen, N. Zhang, and S. Kim, "Symbolic Discovery of Optimization Algorithms," arXiv preprint arXiv:2302.06675, Feb. 2023. DOI: 10.48550/arXiv.2302.06675. [Online]. Available: https://arxiv.org/abs/2302.06675

[8] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," arXiv preprint arXiv:1711.05101, Nov. 2017. DOI: 10.48550/arXiv.1711.05101. [Online]. Available: https://arxiv.org/abs/1711.05101

[9] A. Jacot, F. Gabriel, and C. Hongler, "Neural Tangent Kernel: Convergence and Generalization in Neural Networks," in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 31, pp. 8571–8580, 2018. DOI: 10.48550/arXiv.1806.07572. [Online]. Available: https://arxiv.org/abs/1806.07572

[10] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proc. Int. Conf. Machine Learning (ICML), 2015, pp. 448–456. DOI: 10.48550/arXiv.1502.03167. [Online]. Available: https://arxiv.org/abs/1502.03167

[11] B. Zhang and R. Sennrich, "Root Mean Square Layer Normalization," in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 32, pp. 9595–9607, 2019. DOI: 10.48550/arXiv.1910.07467. [Online]. Available: https://arxiv.org/abs/1910.07467

[12] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-Batch Training with Batch-Channel Normalization and Weight Standardization," in Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), 2019, pp. 11327–11335. DOI: 10.48550/arXiv.1903.10520. [Online]. Available: https://arxiv.org/abs/1903.10520