WaLeM: Weight-Aware Learnable Merging for Large Language Model Depth Compression

Anonymous authors
Paper under double-blind review

Abstract

The residual connections in modern Large Language Models (LLMs) can be formulated as forward-Euler discretizations of continuous-depth dynamical systems. From this Neural ODE perspective, heuristic depth-compression techniques—such as layer pruning and merging—mathematically correspond to step-size coarsening operations over discrete trajectories. However, existing methods implicitly apply uniform coarsening schemes that ignore the heterogeneous smoothness of distinct Transformer operators, thereby incurring unmitigated local truncation errors. To address this, we introduce WaLeM, a principled, smoothness-aware operator-split coarsening framework for LLM depth compression. Specifically, WaLeM estimates local trajectory smoothness via Centered Kernel Alignment (CKA) and employs dynamic programming to construct an adaptive temporal grid that minimizes a global truncation-error surrogate. Furthermore, it learns component-wise quadrature weights to strictly eliminate operator-coupling errors. Theoretically, we establish an end-to-end approximation error bound of $O(LM_f((1-\gamma)B+L_fB^2)e^{L_fL})$ and demonstrate that our scheduling strategy optimally minimizes this bound. Across five mainstream LLM families, WaLeM outperforms state-of-the-art pruning and merging baselines with or without post-compression recovery, isolating the gain to the coarsening itself rather than to subsequent fine-tuning. Notably, it exhibits exceptional robustness on long-chain reasoning tasks (e.g., GSM8K), where early-stage discretization errors typically compound exponentially. Our work formally bridges heuristic layer reduction with rigorous numerical analysis, establishing a unified design space for continuous-depth LLM compression.

1 Introduction

The rapid evolution of Large Language Models (LLMs), exemplified by the GPT (Brown et al., 2020; Achiam et al., 2023) and LLaMA (Touvron et al., 2023; Grattafiori et al., 2024) families, has revolutionized natural language processing. Scaled to billions of parameters, these models exhibit unprecedented capabilities across diverse downstream tasks, including complex reasoning (Wei et al., 2023; Yao et al., 2023), coding (Chen et al., 2021; Jimenez et al., 2024), and mathematical problem-solving (Cobbe et al., 2021; Lightman et al., 2023). However, the prohibitive computational overhead and extensive memory footprints associated with such massive scales present substantial bottlenecks for real-world deployment, particularly in resource-constrained environments like edge computing platforms. Consequently, network depth reduction has emerged as a promising, hardware-agnostic model compression paradigm. By removing sequential residual blocks from a trained network, it mitigates inference latency without requiring specialized hardware accelerators, and it is often the only recourse available, since the capability one needs may exist only in a model too large to deploy. Despite this potential, current depth reduction methodologies remain constrained by heuristic implementation strategies and a lack of rigorous theoretical foundations (Men et al., 2024; Gromov et al., 2025; Yang et al., 2024; Ma et al., 2023).

To develop a principled theoretical foundation for depth reduction, we revisit Transformer architectures from a Neural ODE perspective (Lu et al., 2019; Li et al., 2022). Specifically, the residual connections in Transformer-based architectures, formulated as $h^{(l+1)} = h^{(l)} + f_l(h^{(l)})$, can be naturally interpreted as a forward-Euler discretization of a continuous-depth dynamical system $d\mathbf{h}(t)/dt = f(\mathbf{h}(t), t)$ with a unit step size $\Delta t = 1$ (Chen et al., 2019). Under this continuous-time framework, we can further reconceptualize LLM depth reduction as an adaptive step-size control

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

problem in numerical ODE integration. From this perspective, the compression objective translates into consolidating multiple small Euler steps into fewer, larger adaptive steps while strictly bounding the Local Truncation Error (LTE)—a classical numerical analysis problem governed by well-established theoretical bounds.

Viewed through this numerical lens, current depth reduction paradigms exhibit profound structural flaws. The first paradigm, direct layer removal (Men et al., 2024; Song et al., 2024), typically relies on heuristic importance metrics (e.g. block influence (BI), perplexity, or angular distance) to remove redundant layers (Men et al., 2024; Song et al., 2024; Gromov et al., 2025). However, within a continuous dynamical framework, unconditionally dropping these forward-Euler steps equates to introducing severe discontinuities into the ODE trajectory. This forced state transition skips intermediate vector field transformations, thereby injecting an unbounded and mathematically unverified truncation error that cascades exponentially through the residual stream (Li et al., 2024; 2022).

While direct layer removal induces discontinuities by deleting Euler steps outright, the second paradigm, layer merging (Choudhry et al., 2025; Liu et al., 2025), seeks a seemingly smoother alternative by consolidating multiple layers into a single equivalent block. Yet, existing techniques predominantly impose uniform merging ratios through the network (Yang et al., 2024; Liu et al., 2025). From an integration standpoint, this coarse-grained approach enforces a homogeneous step-size modification across all parameter matrices, including Attention (Q/K/V/O) and FFN (up/gate/down) modules. This fundamentally ignores the intrinsic heterogeneity of Transformer blocks-specifically, the fact that these sub-operators possess highly disparate Lipschitz constants and govern distinct flow dynamics (Clark et al., 2019; Geva et al., 2021; Meng et al., 2023). Consequently, uniform merging implicitly assumes a monolithic vector field, inevitably leading to severe operator-coupling errors, over-compression of sensitive components, and ultimately catastrophic feature blurring.

Collectively, the fundamental limitation of the current literature lies in the absence of a unified, physics-informed framework to address two coupled imperatives: *where* the integration steps should be coarsened (macro-level scheduling), and *how* heterogeneous operators should be aggregated (micro-level quadrature). The reliance on ad-hoc metrics and greedy heuristics underscores the urgent need for a principled, numerical ODE-driven layer merging paradigm.

Motivated by this numerical diagnosis, we do not merely employ ODEs as a post-hoc theoretical justification; rather, we constructively derive an optimal step-coarsening scheme directly from first principles. We propose **WaLeM** (**W**eight-**a**ware **Le**arnable **M**erging), a unified framework that naturally maps classical numerical integration techniques to the architectural components of LLMs. Specifically, our design establishes three core mathematical mappings: (1) We utilize Centered Kernel Alignment (CKA) (Kornblith et al., 2019) as a tractable proxy to estimate the local Lipschitz continuity (trajectory smoothness) of the continuous vector field; (2) We employ a Dynamic Programming algorithm to search for the optimal discrete temporal grid, effectively functioning as smoothness-aware adaptive schedule to bound global LTE; (3) We introduce an operator-split coarsening strategy with learnable component-wise quadrature weights, explicitly eliminating operator-coupling errors.

In summary, our main contributions are as follows: (i) **Theoretical Formalization:** We mathematically formalize the heuristic practice of network depth reduction as a continuous-depth coarsening problem within the Neural-ODE framework, establishing a unified mathematical lens for layer compression. (ii) **Optimal Scheduling Bounds:** We theoretically prove the end-to-end approximation error bounds of our compression scheme, demonstrating that our dynamic programming strategy optimally identifies coarse integration intervals strictly bounded by local smoothness. (iii) **Operator Splitting Integration:** We derive learnable component-wise quadrature weights that strictly adhere to operator splitting principles, ensuring the independent and optimal preservation of Attention and MLP flow dynamics. (iv) **Empirical Excellence:** We conduct comprehensive evaluations on various benchmarks and LLM architectures. Our results demonstrate that WaLeM consistently outperforms existing heuristic pruning and uniform merging baselines, achieving superior performance retention.

2 Continuous-Depth Coarsening Framework

In this section, we formulate layer pruning in Transformers as a continuous-depth coarsening problem. We first demonstrate that layer compression in residual networks is mathematically equivalent to the

integration step size of an underlying neural ODE, which inevitably introduces Local Truncation Error (§2.1). To bound this error, we propose a smoothness-aware adaptive schedule that identifies the optimal coarse time grid via dynamic programming, using representation similarity as a tractable proxy for local Lipschitz continuity (§2.2). Based on this grid, we introduce WaLeM, an operator-split coarsening algorithm that learns component-wise weights to align the fine and coarse trajectories, followed by a deferred correction mechanism to absorb high-order discretization errors (§2.3).

2.1 Layer Pruning as Continuous-Depth Coarsening

Modern decoder-only large language models (LLMs) are fundamentally constructed from stacked residual blocks (Gromov et al., 2025). Let $\mathbf{h}^{(l)} \in \mathbb{R}^{n\times d}$ denote the hidden representation at layer l. The generic residual, $\mathbf{h}^{(l+1)} = \mathbf{h}^{(l)} + f_l(\mathbf{h}^{(l)}; \theta_l)$, exactly mirrors the forward-Euler discretization of a continuous-depth dynamical system $d\mathbf{h}(t)/dt = f(\mathbf{h}(t), t)$, with a unit step size $\Delta t = 1$ (Chen et al., 2019; E, 2017; Lu et al., 2019; Haber & Ruthotto, 2017; Tong et al., 2025). Under this continuous-depth perspective, layer pruning and merging (Fan et al., 2019; Men et al., 2024; Ashkboos et al., 2024; Liu et al., 2025) can be recast as step-size coarsening in numerical integration.

Consider an interval of B consecutive fine-grained layers. Compressing these layers implies substituting B unit-step Euler updates with a single coarse step ($\Delta t = B$). According to numerical analysis (Linge & Langtangen, 2016; Ascher & Petzold, 1998), this substitution incurs a Local Truncation Error (LTE) bounded by:

$$\mathrm{LTE} = O\left(B^2 \sup_{t\in[l,l+B]} \left\| \frac{\partial f}{\partial \mathbf{h}} \right\| \right). \tag{1}$$

This bound dictates that temporal coarsening is only well-posed in regions where the trajectory is locally smooth (i.e., a small local Lipschitz constant). Thus, the objective of layer compression formally translates to bounding this accumulated discretization error while minimizing the number of integration steps.

However, standard Neural ODE frameworks treat the layer transformation f as a monolithic vector field, which fails to capture the intrinsic heterogeneity of Transformer blocks (Dong et al., 2023; Geva et al., 2021; 2022). Formally, the parameters of a layer can be decomposed as $\theta_l = \{\theta_l^{(\tau)}\}_{\tau\in\mathcal{T}}$, where $\mathcal{T}$ denotes diverse architectural components (e.g., query/key projections, MLP gates). Different operators exhibit vast disparities in parameter sensitivity and Lipschitz continuity (Kim et al., 2021; Sun et al., 2024; Dasoulas et al., 2021). Uniform layer dropping or merging implicitly assumes that all heterogeneous sub-operators evolve under identical quadrature weights, inevitably injecting unmitigated truncation errors. A principled compression paradigm must therefore resolve two coupled challenges: (1) determining a non-uniform temporal grid based on trajectory smoothness to bound macro-level LTE, and (2) deriving operator-specific integration schemes within each interval to minimize micro-level approximation errors.

2.2 Smoothness-Aware Adaptive Coarsening

To address the first challenge, we must identify optimal intervals for coarse integration. While the LTE is bounded by the local Lipschitz constant L_f, explicitly computing high-dimensional Jacobians across LLM layers is computationally intractable. Alternatively, under the small-residual assumption, a high similarity between consecutive hidden states implies a slowly varying vector field, correlating with a lower L_f (Jastrzębski et al., 2018; Chang et al., 2018). We therefore leverage Centered Kernel Alignment (CKA) (Kornblith et al., 2019; Raghu et al., 2022) as a theoretically grounded, scalable proxy for local smoothness.

Let $\mathbf{H}_i \in \mathbb{R}^{N\times d}$ denote the activation matrix at layer i, aggregated from the hidden states $\mathbf{h}^{(i)}$ over a calibration set of N tokens. We define the local coarsening cost for an interval $[i, j]$ as:

$$\mathcal{E}(i, j) = \beta_i \cdot (j - i) \cdot (1 - \mathrm{CKA}(\mathbf{H}_i, \mathbf{H}_j)). \tag{2}$$

In this formulation, the raw trajectory deviation $(1 - \mathrm{CKA})$ is explicitly scaled by the interval size $(j - i)$ to bound error accumulation over long integration steps, and weighted by a depth-dependent coefficient β_i to mitigate the cascading effect of perturbations in shallow layers.

To optimally allocate a target budget of K retained blocks (Raposo et al., 2024), we aim to find an adaptive coarse temporal grid (schedule) $\mathcal{S} = \{s_0, s_1, \ldots, s_K\}$ satisfying $0 = s_0 < s_1 < \cdots < s_K = L$, which minimizes the global truncation error surrogate:

$$\min_{\mathcal{S}} \sum_{k=0}^{K-1} \mathcal{E}(s_k, s_{k+1}), \tag{3}$$

This global objective exhibits optimal substructure and can be efficiently solved via dynamic programming. Let $D[k, j]$ denote the minimum accumulated cost of partitioning the first j layers into k intervals. The recurrence is $D[k, j] = \min_{i<j}\{D[k-1, i] + \mathcal{E}(i, j)\}$. To ensure numerical stability, we constrain the search space by a maximum block size $j - i \leq B$ and a minimum smoothness threshold $\mathrm{CKA}(\mathbf{H}_i, \mathbf{H}_j) \geq \gamma$. The optimal schedule $\mathcal{S}^*$ is recovered by backtracking from $D[K, L]$, strategically allocating large-step integrations to smooth regions while preserving fine-grained resolution where representations evolve rapidly.

2.3 WaLeM: Operator-Split Coarsening with Trajectory Correction

Given the optimal schedule $\mathcal{S}^*$, we must now aggregate the heterogeneous operators within each selected interval $[s_k, s_{k+1}]$. A naive strategy assigns a shared merging coefficients across all components, implicitly forcing a homogeneous flow dynamic. To eliminate the resulting operator-coupling error, WaLeM employs an operator-split (Strang, 1968) coarsening strategy.

For each interval k and operator type $\tau \in \mathcal{T}$, we construct a merged operator $\widetilde{W}_k^\tau$:

$$\widetilde{W}_k^\tau = \sum_{l=s_k}^{s_{k+1}-1} \alpha_l^\tau W_l^\tau, \tag{4}$$

where W_l^τ is the fine-layer parameter matrix, and α_l^τ is the learnable, component-specific integration weight. To ensure numerical stability and bound parameter scaling, α_l^τ is parameterized on the probability simplex via a softmax transformation, ensuring the merged operator strictly resides within the convex hull of its fine-layer counterparts (Wortsman et al., 2022; Ilharco et al., 2023).

We optimize these operator-split weights via trajectory matching on a calibration set. Let Φ_L denote the terminal flow of the original fine-grained model, and $\widetilde{\Phi}_{K,\mathcal{S}^*,\alpha}$ denote the terminal flow of the coarsened model under schedule $\mathcal{S}^*$ and weight α. We optimize:

$$\min_{\alpha} \mathbb{D}\left(\Phi_L(\mathbf{h}_0), \widetilde{\Phi}_{K,\mathcal{S}^*,\alpha}(\mathbf{h}_0)\right). \tag{5}$$

We instantiate $\mathbb{D}$ as the token-level Kullback–Leibler divergence between the predictive distributions emitted at the final depth by the dense and the coarsened flow:

$$\mathbb{D}\left(\Phi_L(\mathbf{h}_0), \widetilde{\Phi}_{K,\mathcal{S}^*,\alpha}(\mathbf{h}_0)\right) = \frac{1}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} \mathrm{KL}\big(p_{\mathrm{dense}}(\cdot \mid x_{\leq t}) \,\big\|\, p_\alpha(\cdot \mid x_{\leq t})\big), \tag{6}$$

where $\mathcal{V}$ collects all non-padding positions in the calibration batch. Supervision is thus applied at every position rather than only at the sequence end, constraining the coarse solver to track the dense trajectory throughout the calibration rollout.

Despite optimal scheduling and operator-split weights, residual errors from high-order truncation and imperfect CKA estimation remain. Inspired by deferred correction in numerical analysis, WaLeM introduces a lightweight Low-Rank Adaptation (LoRA) (Hu et al., 2021) module to absorb these residuals: $\widetilde{\Phi}^{\mathrm{corr}} = \widetilde{\Phi} + \Delta_{\mathrm{LoRA}}$. Crucially, LoRA serves strictly as a high-order correction; the primary compression is achieved intrinsically through physics-informed coarsening. Appendix A.6

Table 1: Zero-shot performance of Llama models under different pruning strategies. "Dense" denotes the unpruned model and "PPL" the WikiText-2 perplexity; **best** and second-best pruned results are highlighted. Llama2-7B results are in Appendix A.4.

Model	Method	Pruning Rate	PPL (↓)	HellaSwag	PIQA	Winogrande	MMLU	ARC-e	ARC-c	RACE	GSM8K	Avg. Score
Llama3-8B	Dense	0.00%	7.44	79.11	80.96	73.40	62.12	77.65	54.10	40.38	49.89	66.82
	MKA	18.75%	1469.73	57.39	69.97	59.75	58.45	61.91	44.03	31.96	2.50	54.78
	ShortGPT	18.75%	39.72	68.31	72.80	70.17	60.55	59.47	43.94	34.74	5.61	58.57
	SLEB	18.75%	14.99	63.66	74.86	56.27	25.85	62.25	35.07	32.73	2.58	50.10
	LaCo	18.75%	3167.02	30.45	57.89	59.19	35.09	36.36	28.58	24.11	0.30	38.81
	LLMPruner	18.75%	29.50	69.71	76.17	67.09	**62.04**	70.16	**48.55**	38.76	23.96	61.78
	Ours	18.75%	**13.63**	**74.52**	**76.55**	**72.23**	61.28	**70.96**	47.95	**41.82**	**36.32**	**63.62**
	MKA	25.00%	3557.07	52.94	67.74	60.38	**62.20**	59.00	41.72	29.95	0.45	53.42
	ShortGPT	25.00%	-	31.41	61.15	55.01	34.59	39.65	33.11	24.69	0.23	39.94
	SLEB	25.00%	23.07	57.96	72.14	52.41	26.63	57.87	32.85	31.29	1.90	47.31
	LaCo	25.00%	89.61	60.01	67.90	65.04	60.58	52.27	38.48	32.25	2.12	53.79
	LLMPruner	25.00%	42.06	67.34	73.50	65.82	61.04	**65.99**	**45.56**	37.99	2.20	59.61
	Ours	25.00%	**17.19**	**70.63**	**73.67**	**70.72**	59.71	64.81	44.20	**40.67**	**22.82**	**60.63**
	MKA	31.25%	9999.44	48.19	63.87	60.85	**59.03**	52.90	39.76	29.86	0.00	50.64
	ShortGPT	31.25%	-	33.83	60.45	58.41	36.68	38.24	32.08	26.79	0.99	40.93
	SLEB	31.25%	34.24	52.88	69.26	53.51	26.90	52.36	31.31	29.67	2.35	45.13
	LaCo	31.25%	166.27	48.36	63.71	63.61	25.38	41.46	31.57	29.38	1.29	43.35
	LLMPruner	31.25%	59.94	63.31	71.38	64.40	54.27	**64.14**	**44.71**	36.36	**14.94**	56.94
	Ours	31.25%	**21.84**	**66.15**	**72.52**	**70.09**	58.78	61.49	42.75	**38.85**	5.53	**58.66**
Llama2-13B	Dense	0.00%	7.74	79.65	80.30	72.53	52.39	76.47	48.81	40.48	23.65	64.38
	MKA	20.00%	1411.24	55.44	69.64	62.83	50.69	58.33	40.53	32.15	0.00	52.80
	ShortGPT	20.00%	15.35	72.62	75.63	69.93	52.22	67.00	43.26	38.09	2.35	59.82
	SLEB	20.00%	11.77	70.69	76.33	63.69	30.02	70.62	43.00	37.70	2.81	56.01
	LaCo	20.00%	66.70	61.98	69.53	65.35	37.96	57.74	37.80	33.68	1.67	52.01
	LLMPruner	20.00%	38.53	72.09	76.82	66.30	52.59	71.09	46.42	37.42	11.45	60.39
	Ours	20.00%	**11.63**	**76.33**	**78.13**	**71.35**	**53.57**	**71.84**	**48.55**	**40.57**	**13.50**	**62.91**
	MKA	25.00%	4433.19	50.91	66.59	61.17	50.06	54.29	37.88	31.67	0.00	50.37
	ShortGPT	25.00%	40.40	68.08	72.74	69.14	50.10	60.10	41.89	37.32	0.76	57.05
	SLEB	25.00%	13.90	67.23	**76.66**	60.38	25.31	65.45	39.33	35.31	1.44	52.81
	LaCo	25.00%	130.90	55.80	66.59	64.25	34.45	54.17	35.75	30.53	0.30	48.79
	LLMPruner	25.00%	67.54	67.89	75.14	64.88	**52.27**	68.56	44.03	36.75	**6.82**	58.50
	Ours	25.00%	**13.44**	**74.34**	76.39	**71.82**	51.69	**69.61**	**45.56**	**40.10**	6.67	**61.36**
	MKA	30.00%	8453.01	47.57	64.58	59.75	51.44	52.23	36.09	31.29	0.00	48.99
	ShortGPT	30.00%	65.10	58.31	69.86	68.11	48.44	53.45	36.77	31.96	0.99	52.41
	SLEB	30.00%	16.54	64.49	74.59	61.09	23.37	63.09	36.60	34.45	1.67	51.10
	LaCo	30.00%	205.28	52.07	63.98	63.14	36.87	51.30	35.15	29.57	0.53	47.44
	LLMPruner	30.00%	85.14	65.53	72.85	64.01	51.05	65.74	**43.34**	36.27	3.79	56.97
	Ours	30.00%	**15.11**	**71.60**	**75.24**	**70.01**	**52.76**	**66.50**	43.26	**38.85**	**4.40**	**59.75**

verifies this claim directly, showing that WaLeM leads both when deferred correction is disabled for every method and when every method receives an identical recovery budget.

We now establish its theoretical properties (proofs deferred to Appendix A.3).

Proposition 1 (End-to-End Approximation Error)**.** *Assume the underlying vector field has a bounded local Lipschitz constant L_f. If the coarse block size is bounded by B, and every selected interval satisfies the smoothness threshold γ, the approximation error of WaLeM is bounded by:*

$$\left\| \Phi_L(\mathbf{h}_0) - \widetilde{\Phi}_{K,\mathcal{S}^*,\alpha}(\mathbf{h}_0) \right\| = O\left(L M_f ((1-\gamma)B + L_f B^2) e^{L_f L} \right). \tag{7}$$

Remark. This bound provides three core insights: (1) **Block Size (B):** The linear scaling with B mathematically explains the instability of aggressive, large-interval merging. (2) **Smoothness (γ):** High smoothness bounds strictly suppress the local truncation error, validating our CKA-guided scheduling. (3) **Trajectory Length (L):** Local errors amplify exponentially along the depth as $e^{L_f L}$, explaining why long-chain reasoning tasks are highly sensitive to layer pruning, as early-stage trajectory deviations compound throughout the residual stream.

3 Experimental Setup

We evaluate WaLeM on multiple open-source LLMs against representative layer pruning and merging baselines, investigating the compression-performance trade-off and model robustness under aggressive depth reduction and multi-step reasoning.

Models. We conduct experiments on five widely used LLMs spanning various scales and architectural designs: **Llama2-7B/13B** (Touvron et al., 2023), **Llama3-8B** (Grattafiori et al., 2024), and **Qwen2.5-7B/14B** (Qwen et al., 2025). This diverse selection verifies the generalizability of our continuous-depth formulation across different Transformer families. All models are evaluated in their dense forms and under multiple target depth-compression ratios.

Table 2: Zero-shot performance of Qwen models under different pruning strategies. "Dense" denotes the unpruned model and "PPL" the WikiText-2 perplexity; **best** and second-best pruned results are highlighted.

Model	Method	Pruning Rate	PPL (↓)	HellaSwag	PIQA	Winogrande	MMLU	ARC-e	ARC-c	RACE	GSM8K	Avg. Score
Qwen2.5-7B	Dense	0.00%	9.41	80.58	80.47	71.19	71.78	81.40	55.03	46.22	81.80	69.52
	MKA	21.43%	182964.77	45.12	61.81	58.09	**40.31**	40.40	35.49	28.61	1.14	44.26
	ShortGPT	21.43%	16.64	65.26	**76.93**	56.04	28.12	68.27	40.44	**37.42**	4.25	53.21
	LaCo	21.43%	797.66	53.05	61.97	53.59	26.48	38.22	30.97	25.26	1.21	41.36
	Ours	21.43%	**13.22**	**66.84**	76.39	**60.22**	39.78	**69.49**	**45.05**	37.32	**15.69**	**56.44**
	MKA	25.00%	607147.37	42.11	60.72	**57.38**	**37.10**	37.92	33.53	26.99	0.99	42.25
	ShortGPT	25.00%	17.96	61.02	74.59	53.99	27.86	63.72	36.43	35.12	2.96	50.39
	LaCo	25.00%	1357.73	48.24	60.77	50.04	23.07	35.14	30.29	22.58	1.21	38.59
	Ours	25.00%	**13.71**	**65.06**	**76.12**	57.06	30.63	**68.27**	**42.66**	**37.22**	**8.95**	**53.86**
	MKA	28.57%	1876698.27	38.60	59.14	56.35	23.00	34.01	31.91	25.36	1.59	38.34
	ShortGPT	28.57%	20.44	56.95	73.45	54.93	**26.58**	61.49	36.09	34.64	2.35	49.16
	LaCo	28.57%	1694.25	44.19	58.92	51.22	23.44	34.85	30.29	23.25	0.61	38.02
	Ours	28.57%	**17.95**	**62.12**	**74.97**	**56.43**	26.14	**65.87**	**41.38**	**35.22**	**4.17**	**51.73**
Qwen2.5-14B	Dense	0.00%	6.66	84.37	81.94	75.77	78.83	81.65	62.29	45.36	44.43	72.89
	MKA	20.83%	400283.16	45.90	65.02	58.96	**77.80**	58.33	41.38	31.39	0.00	54.11
	ShortGPT	20.83%	-	71.47	77.91	60.30	42.60	66.46	43.34	35.41	3.03	56.78
	LaCo	20.83%	1909.78	47.30	61.81	62.04	53.79	42.38	33.79	29.19	0.23	47.19
	Ours	20.83%	**11.95**	**72.31**	**78.18**	**63.22**	44.94	**72.90**	**49.23**	**38.28**	**26.16**	**59.87**
	MKA	25.00%	1289431.81	42.51	63.17	59.59	**53.32**	52.15	38.91	30.53	0.00	48.60
	ShortGPT	25.00%	-	67.00	75.14	56.20	32.97	62.79	37.20	34.55	3.11	52.26
	LaCo	25.00%	12723.14	44.42	62.35	**59.91**	42.08	39.65	35.58	30.53	0.15	44.93
	Ours	25.00%	**13.64**	**69.60**	**77.37**	58.56	39.96	**69.87**	**45.05**	**36.08**	**9.33**	**56.64**
	MKA	31.25%	7517022.80	36.18	57.94	57.85	**32.08**	39.52	34.30	26.32	0.00	40.60
	ShortGPT	31.25%	-	46.86	68.82	51.38	23.64	52.78	30.55	24.11	0.53	42.59
	LaCo	31.25%	775.68	40.32	62.84	56.27	22.99	35.52	30.72	25.26	0.68	39.13
	Ours	31.25%	**18.76**	**63.76**	**74.27**	**59.35**	31.84	**64.31**	**40.78**	**37.03**	**4.78**	**53.05**

Datasets. We utilize the **C4** (Raffel et al., 2023) corpus for calibration, sampling 500 examples to compute layer-wise CKA similarities and 10,000 examples to optimize the component-wise merging coefficients. To mitigate residual discretization errors, we perform lightweight deferred correction via LoRA-based post-pruning recovery on the **Alpaca** (Taori et al., 2023) dataset.

Evaluation Metrics. We evaluate compressed models along four axes. (1) **Language modeling quality**: perplexity on WikiText2 (Merity et al., 2016). (2) **Downstream accuracy**: seven likelihood-based benchmarks under the LM Evaluation Harness (Gao et al., 2024)—HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), WinoGrande (Sakaguchi et al., 2019), MMLU (Hendrycks et al., 2021), ARC-Easy/Challenge (Clark et al., 2018), and RACE (Lai et al., 2017). **Avg. Score** denotes the unweighted mean over exactly these seven tasks. (3) **Long-horizon generative robustness**: GSM8K (Cobbe et al., 2021), reported separately rather than folded into Avg. Score, as it is the only generative, multi-step, exact-match task in our suite and probes the error amplification predicted by Proposition 1. (4) **Compression efficiency**: the depth-reduction ratio, with induced parameter and KV-cache savings in Appendix A.11. All ratios denote *depth* reduction, i.e. the fraction of residual blocks removed, which differs from parameter reduction as embeddings and the LM head are unaffected.

Baselines. We compare WaLeM against representative structural compression baselines, encompassing both layer pruning and layer merging paradigms. For direct layer removal, we include **LLM-Pruner** (Ma et al., 2023), **ShortGPT** (Men et al., 2024), and **SLEB** (Song et al., 2024), which eliminate layers based on structural pruning criteria, block influence, and greedy perplexity minimization, respectively. For layer merging, we compare against **LaCo** (Yang et al., 2024) and **MKA** (Liu et al., 2025), which merge adjacent layers utilizing representation-based or manifold-based criteria.

Implementation Details. We compute the CKA similarity matrix on 500 C4 samples to extract non-overlapping coarse intervals via dynamic programming. Coarse block sizes are bounded to $(4, 6)$ ($(5, 7)$ for for Llama2-13B) with a 0.85 smoothness threshold. Given these intervals, we freeze the base model and optimize component-wise integration weights on 10K C4 samples for 2 epochs (LR 0.1, batch 4, 4 accumulation steps, temperature 1.0). For post-pruning recovery, we fine-tune via LoRA ($\gamma = 8$, $\alpha = 16$, dropout 0.05) on the Alpaca dataset for 2 epochs (LR 10^{-4}, batch size 4, 16 accumulation steps). Baselines follow official settings; notably, we adopt LLM-Pruner's layer-pruning configuration and terminate LaCo's iterative merging immediately upon reaching the target pruning ratio. Full per-model resolved configurations, calibration/evaluation sample indices, seeds and environment versions are documented in Appendix A.10 and released with the code.

Table 3: Perplexity of the proposed DP schedule and the ShortGPT Block-Influence (BI) schedule across depth-reduction ratios.

(a) Results on Llama3-8B

Strategy	Pruning Rate				
	18.75%	**21.88%**	**25.00%**	**28.13%**	**31.25%**
BI	**21.16**	76.44	100.54	190.62	363.86
DP	44.24	**59.33**	**89.27**	**142.42**	**263.25**

(b) Results on Llama2-13B

Strategy	Pruning Rate				
	15.00%	**20.00%**	**25.00%**	**30.00%**	**35.00%**
BI	11.54	13.86	28.21	**41.22**	60.39
DP	**10.90**	**13.53**	**18.05**	49.38	**59.22**

4 Experimental Results

To validate our continuous-depth coarsening perspective and demonstrate WaLeM's superior compression-performance trade-off, our evaluation addresses three core questions: (1) Does WaLeM outperform established baselines at practical pruning ratios? (2) What is the individual contribution of our proposed mechanisms? (3) Do empirical observations—such as CKA similarity and learned operator weights—corroborate the theoretical error bounds established in Section 2?

4.1 Main Results

Tables 1 and 2 report zero-shot results on the Llama and Qwen families. WaLeM attains the best compression–performance trade-off on all five backbones, and its margin widens under aggressive compression, where baselines either diverge in perplexity or lose capability outright. On Llama3-8B at 25% depth reduction it reaches the highest average score (60.63) at a perplexity of 17.19, whereas the strongest baseline by average score (LLM-Pruner, 59.61) is $2.4\times$ worse in perplexity (42.06) and the strongest by perplexity (SLEB, 23.07) gives up 13.32 average points; on Llama2-13B it leads on both metrics at every ratio, and at 20% it matches the depth of dense Llama2-7B (32 blocks) at 2.20 points higher average score. The Qwen results isolate the role of smoothness-aware coarsening: at 20.83% on Qwen2.5-14B, similarity-based merging destabilizes the forward trajectory entirely (MKA $400{,}283.16$, LaCo $1{,}909.78$), while WaLeM holds perplexity at 11.95 with a 59.87 average. Stable coarsening thus requires operator-wise quadrature weights and explicit residual correction, not merely the identification of similar layers.

On the long-horizon probe the separation grows by an order of magnitude: WaLeM retains 26.16 on GSM8K at 20.83% on Qwen2.5-14B, against 0.00, 0.23 and 3.03 for MKA, LaCo and ShortGPT; 22.82 against 2.20 on Llama3-8B at 25.00%; and 15.69 against 4.25 on Qwen2.5-7B at 21.43%. This is what Remark (3) of Proposition 1 predicts: generation traverses the residual stream repeatedly, amplifying an unbounded per-step truncation error as $e^{L_f L}$, whereas a single-pass likelihood score tolerates it. Only at the most aggressive Llama3-8B setting (31.25%) does LLM-Pruner retain more GSM8K (14.94 vs. 5.53), at $2.7\times$ worse perplexity and 1.72 fewer average points; beyond roughly 25% depth the probe degrades for every method (Appendix A.1).

4.2 Ablation Studies

Effect of Maximum Coarse Step Size B. In our framework (§2.1), B defines the maximum integration step ($\Delta t \leq B$). While Proposition 1 strictly bounds the Local Truncation Error (LTE) linearly with respect to B, simply minimizing B is suboptimal. An overly small B necessitates more discrete integration intervals, compounding residual mismatches along the depth trajectory; conversely, an overly large B incurs severe single-step LTE.

Figure 1 and Table 6 (Appendix A.5) confirm that intermediate step sizes are optimal. Llama3-8B peaks at $B \in (3, 5)$ and $(4, 6)$ (score 60.20, PPL 13.62), and Llama2-13B at $(3, 5)$ and $(5, 7)$. The sensitivity is most pronounced on Qwen2.5-14B, where an overly restrictive range $(2, 4)$ fragments the trajectory and degrades perplexity to 170.80, while relaxing it to $(4, 6)$ recovers 11.95. Consistent with §2.2, B must be matched to the model's intrinsic smoothness profile to balance integration stability against truncation error.

Effect of the Adaptive DP Schedule. Table 3 compares the DP schedule against the Block Influence (BI) heuristic under an identical learnable merging stage, so that only the temporal partition differs. DP wins in eight of ten settings, with the margin widening as coarsening becomes harder ($363.86 \rightarrow$

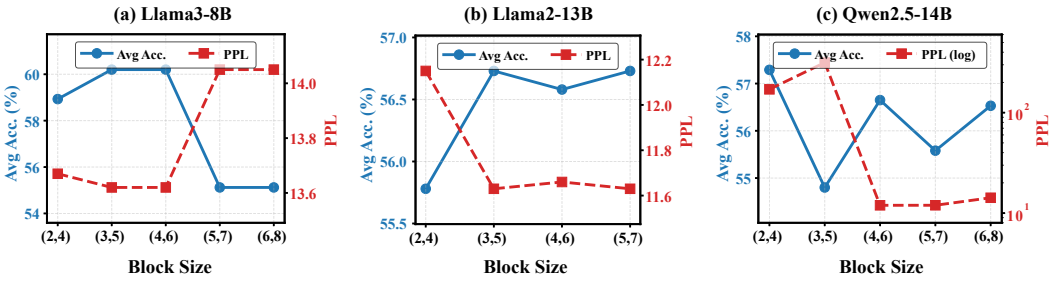

Figure 1: The impact of merging block size on model performance. Results are shown for (a) LLaMA3-8B, (b) LLaMA2-13B, and (c) Qwen2.5-14B. The tuples $(\min, \max)$ on the x-axis denote the minimum and maximum block size constraints used for layer merging.

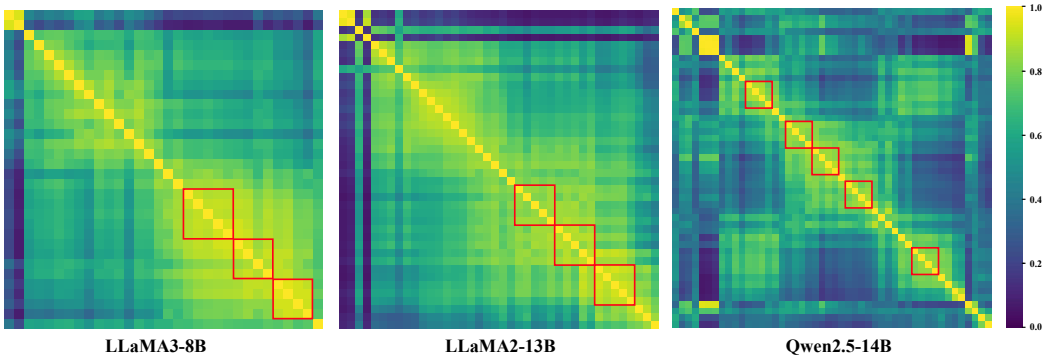

Figure 2: CKA similarity matrices for Llama3-8B, Llama2-13B, and Qwen2.5-14B. The axes represent layer indices arranged from shallow to deep. Red highlighted regions denote the layer fusion strategies adopted at an approximate pruning ratio of 30%.

263.25 on Llama3-8B at 31.25%; 28.21 → 18.05 on Llama2-13B at 25.00%) and BI leading only at the two mildest ratios. This is the behaviour Proposition 1 predicts: local ranking discards individually redundant layers while ignoring that early trajectory deviations compound exponentially with depth, whereas minimizing a global LTE surrogate retains fine-grained steps where the flow varies fastest—which matters precisely when the budget is binding. Appendix A.8 extends the comparison to non-adaptive fixed-interval schedules, over which DP gains +5.00 average points, and shows that low perplexity alone does not imply preserved capability: the strongest fixed schedule reaches perplexity 20.47 against DP's 37.01 while collapsing MMLU to near-chance (23.18 vs. 46.87).

Effect of Component-Wise Integration Weights. Table 4 isolates the impact of WaLeM's component-specific quadrature against baselines using shared merging coefficients. Under identical temporal grids, WaLeM consistently yields substantial perplexity reductions.

This supports the operator-split formulation of §2.3. A shared integration rule treats the Transformer block as a monolithic vector field and thereby imposes an overly restrictive quadrature scheme that introduces operator-coupling errors. Learning operator-specific coefficients instead captures the distinct flow dynamics of each component within the same coarse interval: the DP schedule bounds global truncation error, while component-wise quadrature controls the local coupling error.

Effect of Deferred Correction. To separate the contribution of coarsening from that of the LoRA correction stage, we re-evaluate all six methods with only the compression method varying: the dense backbone, calibration set and evaluation harness are shared, as are the LoRA data, token budget, steps, rank and seed wherever recovery is enabled. We contrast **No Recovery**, with deferred correction disabled for all methods, against **Matched Recovery**, which grants all methods the same budget (Appendix A.6).

Without recovery, WaLeM is the only method that keeps both downstream accuracy and language modelling healthy. Its average score of 53.53 is on par with the top entry (LaCo, 53.77), yet every

Table 4: Comparison of our learnable fusion strategy against similarity-based (MKA) and difference-based (LaCo) baselines under a fixed layer-merging scope.

Model	Method	Low Ratio (18.75% / 21.43%)		High Ratio (25.00%)	
		PPL (↓)	Avg. Score	PPL (↓)	Avg. Score
Llama3-8B	LaCo	494.73	37.00	393.57	38.30
	MKA	1469.73	48.19	3557.07	**46.94**
	Ours	**75.07**	**48.63**	**126.64**	46.69
Llama2-7B	LaCo	100.70	43.75	145.84	40.88
	MKA	1012.85	42.30	1837.25	39.67
	Ours	**72.35**	**44.58**	**102.55**	**41.30**
Qwen2.5-7B	LaCo	611424.10	34.99	478749.02	34.70
	MKA	190022.30	38.93	607147.37	**37.09**
	Ours	**3458.57**	**40.11**	**9401.84**	36.07

competitor with comparable accuracy carries a perplexity one to two orders of magnitude worse (LaCo 89.62; LLM-Pruner and MKA above 3000; ShortGPT diverges), and the only method with a lower perplexity (SLEB, 23.19) loses 6.41 accuracy points and collapses MMLU to near-chance (27.30). Smoothness-aware, operator-split coarsening thus yields a coarse trajectory that is already faithful to the original flow.

Under matched recovery, WaLeM ranks first on average score (59.59) and attains the lowest perplexity among the capability-preserving methods (20.25, ahead of LaCo's 20.30); SLEB's lower perplexity (15.55) again comes with an average of 52.22 and near-random MMLU (27.91). WaLeM therefore leads both before recovery and under an equal recovery budget: deferred correction amplifies but does not create its advantage, because the discretization it starts from already requires little correction.

4.3 Empirical Analysis of the Continuous-Depth View

CKA Similarity as a Smoothness Proxy. Figure 2 illustrates the Centered Kernel Alignment (CKA) similarity matrices for Llama3-8B, Llama2-13B, and Qwen2.5-14B. Diverging from the conventional view that treats these matrices strictly as indicators of representational redundancy, we reinterpret them as empirical estimators of trajectory smoothness. Specifically, high CKA similarity between adjacent layers implies a gradual evolution of hidden representations along the depth dimension. From a dynamical systems perspective, this suggests minimal local vector-field variation, thereby mitigating the approximation risks associated with coarser integration step sizes.

Learned Operator Weights Are Heterogeneous. As illustrated in Figures 3- 5, the learned merging coefficients exhibit distinct layer-wise trajectories across different operator types, rather than collapsing to a uniform scalar. This heterogeneity empirically validates our operator-split formulation: Transformer components do not constitute a homogeneous vector field amenable to shared interpolation; instead, they necessitate a data-driven quadrature rule tailored to heterogeneous operator dynamics. This component-wise flexibility drives WaLeM's performance gains over baselines like MKA and LaCo (Table 4). Whereas heuristic similarity metrics fail to capture individual operator dynamics, our learned coefficients explicitly minimize approximation errors, ensuring a higher-fidelity reconstruction of the original fine-grained flow.

5 Conclusion

This paper presents a continuous-depth perspective on LLM compression, fundamentally reformulating layer pruning and merging as a numerical coarsening problem. To address the truncation errors accumulated along depth and generation trajectories, we propose WaLeM, a smoothness-aware and operator-split coarsening framework. By integrating adaptive discretization with trajectory matching, WaLeM theoretically bounds coarsening errors and empirically outperforms heuristic baselines on Llama and Qwen models, especially in high-compression and complex reasoning regimes. Broadly, our findings advocate for a paradigm shift in LLM compression—from heuristic structural search to principled numerical solver design—inspiring future explorations in adaptive neural dynamics.

AI USE STATEMENT

In this work, we used generative AI tools for polishing author-written prose, namely grammar correction and rewording for clarity. We have not used generative AI tools for research ideation, for the formulation or proof, for the design or implementation of WaLeM, for the execution or analysis of the experiments, or for literature retrieval and citation generation, and the remaining required-disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. Every AI-edited passage was checked by the authors against the underlying experimental records, and all numerical values, claims and references were verified against the original results and sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies depth compression of publicly released pretrained language models. It involves no human subjects, no private or personally identifiable information, and no new data collection. All backbones (Llama2-7B/13B, Llama3-8B, Qwen2.5-7B/14B) and corpora (C4, Alpaca, WikiText-2 and the evaluation suite of §3) are publicly available and are used in accordance with their respective licenses and intended research use; we redistribute no model weights or data.

REPRODUCIBILITY STATEMENT

The method is specified in §2, with the discrepancy objective defined in Eq. equation 6 and a complete proof of Proposition 1, under its stated assumptions, in Appendix A.3. Models, datasets, metrics, baselines and hyperparameters are documented in §3, with per-model resolved configurations and the measurement protocol for all timing and memory figures in Appendix A.10. All models and datasets used in this work are publicly available. We will release the implementation, together with the calibration and evaluation sample indices, seeds and environment versions, upon acceptance.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1998. doi: 10.1137/1.9781611971392. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611971392. _eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611971392.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns, 2024. URL https://arxiv.org/abs/2401.15024.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models, 2019. URL https://arxiv.org/abs/1909.01377.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level residual networks from dynamical systems view, 2018. URL https://arxiv.org/abs/1710.10348.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019. URL https://arxiv.org/abs/1806.07366.

Arjun Choudhry, Chang Liu, Nina Żukowska, Yifu Cai, Mononito Goswami, and Artur Dubrawski. Layermerge: Modality-agnostic depth pruning for efficient foundation model deployment. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. URL https://openreview.net/forum?id=7jEz15AsVT.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention, 2019. URL https://arxiv.org/abs/1906.04341.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

George Dasoulas, Kevin Scaman, and Aladin Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks, 2021. URL https://arxiv.org/abs/2103.04886.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth, 2023. URL https://arxiv.org/abs/2103.03404.

Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5:1 – 11, 2017. URL https://api.semanticscholar.org/CorpusID:64849498.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout, 2019. URL https://arxiv.org/abs/1909.11556.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode: the world of jacobian and kinetic regularization, 2020. URL https://arxiv.org/abs/2002.02798.

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

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories, 2021. URL https://arxiv.org/abs/2012.14913.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space, 2022. URL https://arxiv.org/abs/2203.14680.

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

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2025. URL https://arxiv.org/abs/2403.17887.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, December 2017. ISSN 1361-6420. doi: 10.1088/1361-6420/aa9a90. URL http://dx.doi.org/10.1088/1361-6420/aa9a90.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023. URL https://arxiv.org/abs/2212.04089.

Stanisław Jastrzębski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference, 2018. URL https://arxiv.org/abs/1710.04773.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: Depth pruning for large language models with comparison of retraining methods, 2024. URL https://arxiv.org/abs/2402.02834.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention, 2021. URL https://arxiv.org/abs/2006.04710.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019. URL https://arxiv.org/abs/1905.00414.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations, 2017. URL https://arxiv.org/abs/1704.04683.

Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, JingBo Zhu, Xuebo Liu, and Min Zhang. ODE transformer: An ordinary differential equation-inspired model for sequence generation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8335–8351, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.571. URL https://aclanthology.org/2022.acl-long.571/.

Bei Li, Tong Zheng, Rui Wang, Jiahao Liu, Qingyan Guo, Junliang Guo, Xu Tan, Tong Xiao, Jingbo Zhu, Jingang Wang, and Xunliang Cai. Predictor-corrector enhanced transformers with exponential moving average coefficient learning, 2024. URL https://arxiv.org/abs/2411.03042.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Svein Linge and Hans Langtangen. Solving Ordinary Differential Equations. volume 15, pp. 95–160. July 2016. ISBN 978-3-319-32427-2. doi: 10.1007/978-3-319-32428-9_4.

Deyuan Liu, Zhanyue Qin, Hairu Wang, Zhao Yang, Zecheng Wang, Fangying Rong, Qingbin Liu, Yanchao Hao, Xi Chen, Cunhang Fan, Zhao Lv, Zhiying Tu, Dianhui Chu, Bo Li, and Dianbo Sui. Pruning via merging: Compressing llms via manifold alignment based layer merging, 2025. URL https://arxiv.org/abs/2406.16330.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving transformer from a multi-particle dynamic system point of view, 2019. URL https://arxiv.org/abs/1906.02762.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations, 2020. URL https://arxiv.org/abs/1710.10121.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models, 2023. URL https://arxiv.org/abs/2305.11627.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024. URL https://arxiv.org/abs/2403.03853.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023. URL https://arxiv.org/abs/2202.05262.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Zehua Pei, Hui-Ling Zhen, Xianzhi Yu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. From pruning to grafting: Dynamic knowledge redistribution via learnable layer fusion, 2025. URL https://arxiv.org/abs/2411.14507.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks?, 2022. URL https://arxiv.org/abs/2108.08810.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks, 2024. URL https://arxiv.org/abs/2402.09025.

Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5:506–517, 1968. URL https://api.semanticscholar.org/CorpusID:62800567.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models, 2024. URL https://arxiv.org/abs/2306.11695.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Anh Tong, Thanh Nguyen-Tang, Dongeun Lee, Duc Nguyen, Toan Tran, David Hall, Cheongwoong Kang, and Jaesik Choi. Neural ode transformers: Analyzing internal dynamics and adaptive fine-tuning, 2025. URL https://arxiv.org/abs/2503.01329.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022. URL https://arxiv.org/abs/2203.05482.

Yifei Yang, Zouying Cao, and Hai Zhao. Laco: Large language model pruning via layer collapse, 2024. URL https://arxiv.org/abs/2402.11187.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

A Appendix

A.1 Limitations

Four limitations delimit the present results. First, CKA is a monotone proxy for trajectory smoothness rather than a Lipschitz estimator: it tracks Jacobian-based smoothness closely over the intervals searched by DP ($r = 0.905$, Appendix A.7), and it can misrank intervals. Second, our evidence covers dense decoder-only Transformers with an ordered residual stack, for which a single depth trajectory and operator-aligned fusion are well defined; Mixture-of-Experts and encoder–decoder backbones violate this assumption—token-dependent routing and the distinct roles of self- and cross-attention would require route-conditioned costs and per-operator schedules, respectively—and we leave them to future work. Third, retention depends on stack geometry: the 48-layer Qwen2.5-14B degrades faster than the 32-layer backbones, which places the safe operating range at roughly 21–25% depth for the models studied. A plausible cause is that a fixed threshold γ admits proportionally more candidate intervals as L grows, making the schedule budget-bound rather than smoothness-bound; a depth-adaptive γ is a natural remedy. Finally, our bounds predict the *ordering* of fragility—long-horizon generation degrades before likelihood scoring (§4.1)—but are not numerical forecasts of downstream accuracy.

A.2 Related Work

The interpretation of residual networks as forward Euler discretizations of ordinary differential equations (ODEs) (E, 2017; Lu et al., 2020) has established a foundational link between deep learning and numerical analysis. This perspective culminated in Neural ODEs, which parameterize continuous dynamics to enable memory-efficient, adaptive computation (Chen et al., 2019), inspiring subsequent advances in implicit-depth models and accelerated solvers (Bai et al., 2019; Finlay et al., 2020). Recently, this dynamical-systems lens has been extended to Transformer architectures, where modeling hidden states as continuous trajectories facilitates adaptive fine-tuning and inference-time depth control (Lu et al., 2019). We uniquely leverage the ODE paradigm as a principled framework for model compression, conceptualizing layer pruning as the step-size coarsening of discrete trajectories. We import rigorous numerical-analysis tools—including adaptive grid selection, operator splitting, and local truncation error bounds—to systematically guide layer consolidation.

Identifying redundant layers constitutes a critical step in layer pruning. To address this, numerous studies have established metrics to quantify the importance of specific layers. For instance, ShortGPT (Men et al., 2024) introduces Block Influence (BI), which assesses the similarity between a layer's input and output. Similarly, MKA (Liu et al., 2025) employs manifold learning and the Normalized Pairwise Information Bottleneck to identify layers suitable for merging. Other strategies, such as SLEB (Song et al., 2024), leverage perplexity to evaluate layer importance. Furthermore, angular distance, Magnitude and first-order Taylor saliency has been used to detect less informative layers (Gromov et al., 2025; Kim et al., 2024). Collectively, these metrics guide the pruning process to ensure the preservation of functional integrity.

To mitigate the information loss inherent in direct layer removal, recent studies have pivoted towards layer merging, a paradigm that consolidates adjacent layer parameters rather than discarding them. Prominent approaches, such as MKA (Liu et al., 2025), LaCo (Yang et al., 2024), and LayerMerge (Choudhry et al., 2025), have advanced this direction by employing manifold learning, iterative parameter aggregation, and PCA-based activation analysis, respectively. Despite these innovations, existing techniques predominantly rely on coarse-grained merging strategies, which neglect the functional heterogeneity of components within Transformer layers. Concurrent with and independent of our work, FuseGPT (Pei et al., 2025) utilizing learnable low-rank matrices to graft pruned layers onto adjacent ones. Our approach differs in two key aspects: (1) Granularity: Instead of low-rank grafting, WaLeM performs weight-aware merging, assigning adaptive coefficients to specific functional components; (2) Efficiency: While FuseGPT relies on an iterative greedy search to handle rank shift, WaLeM utilizes Dynamic Programming to efficiently derive a globally optimal merging strategy in a single pass.

A.3 Proof of Proposition 1

We first restate Proposition 1 for completeness.

Proposition 1 (End-to-End Approximation Error). *Assume the underlying vector field $f(\cdot, t)$ has a bounded local Lipschitz constant L_f and uniformly bounded magnitude $\sup_{t,x} \|f(x,t)\| \le M_f$. If the maximum coarse block size is bounded by B, and every interval $[s_k, s_{k+1}]$ in the schedule $\mathcal{S}^*$ satisfies the smoothness condition $\mathrm{CKA}(H_{s_k}, H_{s_{k+1}}) \ge \gamma$, the end-to-end approximation error of WaLeM is bounded by:*

$$\left\| \Phi_L(h_0) - \tilde{\Phi}_{K,\mathcal{S}^*,\alpha}(h_0) \right\| \le M_f L \left[2(1-\gamma)B + \frac{1}{2} L_f B^2 \right] e^{L_f L}, \tag{8}$$

which can be compactly written as $O\big(L\, M_f \big((1-\gamma)B + L_f B^2\big)\, e^{L_f L}\big)$.

Proof. The proof proceeds in two steps: (I) bounding the per-block local truncation error (LTE), and (II) propagating the global error via the discrete Gronwall lemma.

Step I: Local Truncation Error. Let $h^{(l)}$ denote the hidden states of the fine-grained model, updated via $h^{(l+1)} = h^{(l)} + f_l(h^{(l)})$. Within the k-th block of size $B_k \triangleq s_{k+1} - s_k$, the cumulative fine-grained displacement is:

$$h^{(s_{k+1})} = h^{(s_k)} + \sum_{l=s_k}^{s_{k+1}-1} f_l(h^{(l)}). \tag{9}$$

For the coarsened model, the equivalent merged update function $\tilde{F}_k$ can be expressed as a weighted sum over the fine-layer operators evaluated at the coarse state:

$$\tilde{h}^{(s_{k+1})} = \tilde{h}^{(s_k)} + \tilde{F}_k(\tilde{h}^{(s_k)}), \quad \text{where} \quad \tilde{F}_k(x) \triangleq \sum_{l=s_k}^{s_{k+1}-1} w_l\, f_l(x), \tag{10}$$

with $w_l \ge 0$ and $\sum_l w_l = B_k$.[1] By construction, $\tilde{F}_k$ has a Lipschitz constant of $B_k L_f$.

We define the LTE ϵ_k as the one-step discrepancy starting from the exact state $h^{(s_k)}$. By applying the triangle inequality, we decompose ϵ_k into two components:

$$\begin{aligned} \epsilon_k &\triangleq \left\| \sum_{l=s_k}^{s_{k+1}-1} f_l(h^{(l)}) - \tilde{F}_k(h^{(s_k)}) \right\| \\ &\le \underbrace{\left\| \sum_{l=s_k}^{s_{k+1}-1} \Big[f_l(h^{(l)}) - f_l(h^{(s_k)}) \Big] \right\|}_{\text{(A): State-deviation error}} + \underbrace{\left\| \sum_{l=s_k}^{s_{k+1}-1} (1-w_l)\, f_l(h^{(s_k)}) \right\|}_{\text{(B): Quadrature error}}. \end{aligned} \tag{11}$$

Bounding (A): Using the field magnitude bound, we have $\|h^{(l)} - h^{(s_k)}\| \le (l - s_k) M_f$. Applying the Lipschitz condition yields:

$$\text{(A)} \le L_f \sum_{l=s_k}^{s_{k+1}-1} \|h^{(l)} - h^{(s_k)}\| \le L_f M_f \sum_{l=s_k}^{s_{k+1}-1} (l - s_k) = \frac{1}{2} L_f M_f B_k (B_k - 1) \le \frac{1}{2} L_f M_f B_k^2. \tag{12}$$

Bounding (B): Since $\sum_l (1 - w_l) = B_k - B_k = 0$, we can introduce the mean vector field $\bar{f}_k \triangleq \frac{1}{B_k} \sum_{l=s_k}^{s_{k+1}-1} f_l(h^{(s_k)})$ without altering the sum:

$$\text{(B)} = \left\| \sum_{l=s_k}^{s_{k+1}-1} (1-w_l) \Big[f_l(h^{(s_k)}) - \bar{f}_k \Big] \right\| \le \left(\sum_{l=s_k}^{s_{k+1}-1} |1 - w_l| \right) \sup_l \|f_l(h^{(s_k)}) - \bar{f}_k\|. \tag{13}$$

[1] In §2.3, the merged parameter matrix is defined as $\tilde{W}_k = \sum_l \alpha_l W_l$ with $\sum_l \alpha_l = 1$. Applied with a step size of B_k, the effective functional update is $B_k \sum_l \alpha_l f_l(x)$, which is mathematically identical to Eq. equation 10 by letting $w_l = B_k \alpha_l$.

The weight deviation factor is bounded by $2B_k$ (the maximum ℓ_1 distance between uniform weights and w_l). The vector-field variation is constrained by the smoothness assumption $\mathrm{CKA} \geq \gamma$; under standard small-residual conditions (Jastrzębski et al., 2018; Chang et al., 2018), the highly aligned adjacent representations imply $\sup_l \|f_l(h^{(s_k)}) - \bar{f}_k\| \leq (1-\gamma)M_f$. Thus, (B) $\leq 2B_k(1-\gamma)M_f$.

Combining (A) and (B), the per-block LTE is bounded by:

$$\epsilon_k \leq B_k M_f \left[\frac{1}{2} L_f B_k + 2(1-\gamma)\right]. \tag{14}$$

Step II: Global Error Propagation. Define the global error at the partition point as $e_k \triangleq \|\tilde{h}^{(s_k)} - h^{(s_k)}\|$, with $e_0 = 0$. Comparing the coarse update with the fine-grained sequence yields the recurrence:

$$e_{k+1} \leq \underbrace{\|\tilde{h}^{(s_k)} - h^{(s_k)}\| + \|\tilde{F}_k(\tilde{h}^{(s_k)}) - \tilde{F}_k(h^{(s_k)})\|}_{\leq (1+B_k L_f) e_k} + \underbrace{\epsilon_k}_{\text{LTE}}. \tag{15}$$

Using the inequality $1 + x \leq e^x$, we have $e_{k+1} \leq e^{B_k L_f} e_k + \epsilon_k$. Unrolling this recurrence via the discrete Gronwall lemma, and noting that $\sum_i B_i = L$, gives:

$$e_K \leq \sum_{k=0}^{K-1} \epsilon_k \exp\left(L_f \sum_{i=k+1}^{K-1} B_i\right) \leq e^{L_f L} \sum_{k=0}^{K-1} \epsilon_k. \tag{16}$$

Finally, we sum the local errors using Eq. equation 14. Since $\sum_{k=0}^{K-1} B_k = L$ and $\sum_{k=0}^{K-1} B_k^2 \leq B \sum_{k=0}^{K-1} B_k = BL$, we obtain:

$$\sum_{k=0}^{K-1} \epsilon_k \leq \frac{1}{2} L_f M_f B L + 2(1-\gamma) M_f L = M_f L \left[\frac{1}{2} L_f B + 2(1-\gamma)\right]. \tag{17}$$

Substituting Eq. equation 17 into Eq. equation 16 completes the proof. □

Discussion and Implications. The derived bound elucidates the theoretical advantage of the CKA-guided scheduling in WaLeM:

- **Suppression of Quadrature Error:** The DP schedule ensures every selected block satisfies $\mathrm{CKA} \geq \gamma$, directly minimizing the $(1-\gamma)$ term. As $\gamma \to 1$, the quadrature error vanishes, leaving only the higher-order $O(L_f B^2)$ state-deviation error.
- **Comparison with Naive Pruning:** Without smoothness constraints (e.g., arbitrary layer removal), the quadrature error lacks the $(1-\gamma)$ suppression, deteriorating to $O(BM_f)$ per block. For a typical $\gamma = 0.85$, WaLeM's schedule yields a nearly $6.7\times$ theoretical reduction in the quadrature penalty.
- **Error Accumulation Limits:** The linear scaling with depth $O(L)$ and the exponential Lipschitz amplification $e^{L_f L}$ reflect the fundamental integration properties of dynamical systems. This explains why controlling per-step error (via constrained block size B and high γ) is critical for maintaining performance on complex tasks (e.g., GSM8K), where early-stage approximation errors otherwise compound exponentially.

A.4 Extended Evaluation on LLaMA2-7B

Table 5 presents the performance comparison between WaLeM and baseline methods on Llama2-7B across various pruning rates. Our proposed WaLeM consistently outperforms existing baselines on the majority of tasks. We observe particularly significant improvements on HellaSwag, Winogrande, MMLU and RACE. Notably, WaLeM retains over 90% of the original performance on MMLU and RACE even with 31.25% of layers pruned.

Table 5: Zero-shot performance of Llama2-7B under different pruning strategies. "Dense" denotes the unpruned model and "PPL" the WikiText-2 perplexity; **best** and second-best pruned results are highlighted.

Model	Method	Pruning Rate	PPL (↓)	HellaSwag	PIQA	Winogrande	MMLU	ARC-e	ARC-c	RACE	GSM8K	Avg. Score
Llama2-7B	Dense	0.00%	8.76	76.19	78.73	69.46	41.71	73.82	44.97	40.10	13.42	60.71
	MKA	18.75%	1008.68	54.59	66.81	59.19	33.29	55.81	38.05	30.24	0.76	48.28
	ShortGPT	18.75%	27.04	65.18	72.03	64.64	25.90	57.45	38.31	33.49	1.74	51.00
	SLEB	18.75%	14.77	64.89	74.48	59.83	26.48	59.68	34.64	33.78	2.05	50.54
	LaCo	18.75%	100.73	58.21	70.46	62.43	31.68	57.95	37.71	31.77	0.45	50.03
	LLMPruner	18.75%	30.36	68.27	74.16	63.22	34.49	65.70	41.21	35.60	**6.37**	54.66
	Ours	18.75%	**14.69**	**71.52**	**75.79**	**67.25**	**40.14**	**66.50**	**41.89**	**39.43**	5.00	**57.50**
	MKA	25.00%	1842.92	49.81	65.72	57.70	25.06	52.10	36.26	30.05	0.38	45.24
	ShortGPT	25.00%	47.19	59.53	68.44	66.06	23.49	52.78	34.04	32.06	1.52	48.06
	SLEB	25.00%	20.33	60.91	71.60	58.56	27.66	54.21	32.08	31.67	1.52	48.10
	LaCo	25.00%	145.97	51.37	65.51	59.12	31.33	52.99	34.22	31.29	0.30	46.55
	LLMPruner	25.00%	40.57	65.63	**72.42**	62.35	25.14	61.83	**39.33**	35.12	**4.32**	51.69
	Ours	25.00%	**20.28**	**67.06**	72.09	**68.19**	**38.72**	**62.25**	38.57	**38.56**	3.26	**55.06**
	MKA	31.25%	3569.59	46.11	61.53	58.56	30.44	48.40	32.42	31.39	0.30	44.12
	ShortGPT	31.25%	119.99	50.84	63.11	63.54	37.22	45.03	32.17	32.63	0.53	46.36
	SLEB	31.25%	28.78	54.27	68.50	54.22	25.64	50.25	31.06	30.62	1.44	44.94
	LaCo	31.25%	117.32	46.32	61.10	60.54	34.39	37.92	32.08	31.48	1.06	43.40
	LLMPruner	31.25%	51.49	61.89	70.02	61.64	28.31	**56.82**	**35.92**	35.02	2.12	49.95
	Ours	31.25%	**26.88**	**62.04**	**70.18**	**66.22**	**37.91**	54.88	34.39	**37.03**	**2.20**	**51.81**

Table 6: Performance comparison of pruned models with different coarse step size at a 20% depth-reduction ratio. The "Block Indices" column details the specific layer indices contained within each merged block.

Model	Size Range	PPL (↓)	Avg. Score	Block Indices
Llama3-8B	2-4	13.67	58.93	{18-19}, {21-24}, {27-28}, {29-30}
	3-5	13.62	60.20	{21-24}, {27-30}
	4-6	13.62	60.20	{21-24}, {27-30}
	5-7	14.05	55.12	{18-24}
	6-8	14.05	55.12	{18-24}
Llama2-13B	2-4	12.15	55.78	{26-29}, {30-31}, {32-33}, {34-36}, {37-38}
	3-5	11.63	56.73	{27-31}, {32-36}
	4-6	11.66	56.58	{26-29}, {30-35}
	5-7	11.63	56.73	{27-31}, {32-36}
Qwen2.5-14B	2-4	170.80	57.29	{1-2}, {4-6}, {11-13}, {20-21}, {23-24}, {26-27}, {28-29], {42-43}
	3-5	315.61	54.80	{4-6}, {11-13}, {17-19}, {20-22}, {23-25}
	4-6	11.95	55.65	{11-14}, {20-24}, {26-29}
	5-7	11.96	55.58	{11-15}, {20-26}
	6-8	14.21	56.53	{20-25}, {36-41}

A.5 Detailed Results for Sensitivity to Coarse Step Size

Table 6 details the numerical results corresponding to Figure 1. We explored various block size constraints to determine the optimal trade-off between compression rate and model performance. Our experiments reveals an inverted-U performance trajectory. Small blocks (2 to 4 layers) fail to fully exploit redundancy between consecutive layers, leading to an increased number of blocks and merged layers. This fragmentation exacerbates error accumulation during forward propagation. Conversely, larger blocks (6 to 8 layers) degrade performance by over-compressing distinct feature representations. Empirically, a block size range of 3 to 6 offers the most effective granularity, yielding the highest average accuracy and lowest perplexity on Llama3-8B and Llama2-13B.

We formally demonstrate that small block sizes lead to higher error accumulation. Consider a network interval of 4 layers. We compare the Small Block Strategy (merging two 2-layer groups sequentially) against the Large Block Strategy (merging 4 layers directly).

A.6 Detailed Results for Effect of Deferred Correction

To isolate the efficacy of deferred correction, and more importantly to test whether WaLeM's advantage depends on it, we evaluate Llama3-8B at a 25% depth-reduction ratio under two settings: **No Recovery** (deferred correction disabled for every method) and **Matched Recovery** (every method receives an identical LoRA budget). The protocol is frozen across methods: the same dense backbone, the same calibration samples, the same evaluation harness and, when recovery is enabled, the same

Table 7: Recovery-controlled comparison on Llama3-8B at 25% depth reduction, under the shared protocol of §4.2.

Method	PPL (↓)	HellaSwag	PIQA	Winogrande	MMLU	ARC-e	ARC-c	RACE	Avg. Score
No Recovery — deferred correction disabled for every method									
ShortGPT	∞	31.44	61.04	55.09	34.60	39.77	33.11	23.35	39.77
MKA	3343.60	52.38	67.90	61.09	**61.61**	58.80	41.47	29.19	53.20
LLMPruner	3022.56	53.53	67.95	60.46	57.56	**60.10**	**42.83**	30.05	53.21
LaCo	89.62	60.11	67.79	65.59	60.55	51.98	38.48	31.87	**53.77**
SLEB	**23.19**	57.46	**72.09**	53.04	27.30	57.37	32.76	29.86	47.12
WaLeM (Ours)	37.01	**60.32**	69.70	**68.90**	46.86	55.89	38.14	**34.93**	53.53
Matched Recovery — every method receives the identical LoRA budget									
MKA	249.01	61.02	71.60	65.27	58.60	62.46	41.89	33.88	56.39
LLMPruner	251.92	62.05	71.71	63.85	56.72	61.70	42.32	33.21	55.94
SLEB	**15.55**	65.48	**75.79**	55.49	27.91	**66.08**	38.65	36.17	52.22
ShortGPT	25.12	**67.08**	72.09	71.11	49.95	66.16	**44.03**	36.08	58.07
LaCo	20.30	66.50	71.44	70.72	56.36	65.07	43.69	36.46	58.60
WaLeM (Ours)	20.25	66.15	73.34	**71.27**	**61.15**	63.85	43.94	**37.42**	**59.59**

LoRA data, token budget, optimization steps, rank and random seed. Only the compression method differs. We report both average downstream accuracy and perplexity, because either metric in isolation can hide a failure mode.

Without recovery (Table 7, upper block), baseline methods suffer catastrophic degradation: LLM-Pruner and MKA exceed perplexity 3000 and ShortGPT diverges outright, while SLEB is the only baseline with a healthy perplexity (23.19) and pays for it with an average of 47.12 and a near-random MMLU of 27.30. WaLeM is the only method that keeps both quantities healthy simultaneously: its average of 53.53 is statistically indistinguishable from the top entry (LaCo, 53.77), yet its perplexity of 37.01 is $2.4\times$ better than LaCo's 89.62 and two orders of magnitude better than LLM-Pruner's and MKA's. Since no method receives any correction in this condition, the advantage must be attributed to the coarsening itself.

Under matched recovery (lower block), WaLeM ranks first on average score (59.59, ahead of LaCo's 58.60 and ShortGPT's 58.07) and, among the capability-preserving methods, also attains the lowest perplexity (20.25 vs. LaCo's 20.30). SLEB again reaches a lower perplexity (15.55) only while its average falls to 52.22 and MMLU stays near random (27.91). The two conditions together give the decisive reading: WaLeM leads *before* recovery and remains strongest *after* recovery is equalized across methods. Deferred correction is therefore an optional refinement that absorbs residual high-order discretization error, not the origin of the reported gains. Rather than relying on post-hoc fine-tuning to rectify divergent trajectories, WaLeM establishes a numerically robust initialization that minimizes the correction burden, thereby facilitating optimal alignment of terminal distributions via lightweight adapters.

A.7 Validating CKA as a Smoothness Proxy

Our scheduler relies on the endpoint CKA gap $g_{i,j} = 1 - \mathrm{CKA}(\mathbf{H}_i, \mathbf{H}_j)$ as a tractable stand-in for local trajectory smoothness. Since this is the central modelling assumption of §2.2, we validate it against a direct Jacobian-based measurement.

On Llama3-8B we estimate the spectral norm $\|J_l\|_2$ of the Jacobian of each residual vector field $f_l(x) = \mathrm{Block}_l(x) - x$, using power iteration on the same calibration batch used to compute CKA. We then compare $g_{i,j}$ against two aggregations of these norms over each candidate interval $[i, j]$: the maximum, $\max_{l\in[i,j]}\|J_l\|_2$, which captures the least-smooth local step inside the interval, and the sum, $\sum_{l\in[i,j]}\|J_l\|_2$, which captures accumulated variation across the interval. The study covers all 84 candidate intervals searched by the DP recursion, i.e. exactly the set on which the scheduler operates.

Table 8 reports a strong and highly significant linear relationship with the interval maximum ($r = 0.905$) and with the interval sum ($r = 0.858$). At the finer adjacent-layer granularity ($n = 31$), $1 - \mathrm{CKA}(\mathbf{H}_l, \mathbf{H}_{l+1})$ also correlates significantly with $\|J_l\|_2$ (Pearson $r = 0.579$, $p = 6.5 \times 10^{-4}$; Spearman $\rho = 0.479$, $p = 6.4 \times 10^{-3}$). The interval-level result is the more relevant of the two, both

Table 8: Correlation between the endpoint CKA gap used by the scheduler and direct Jacobian-based smoothness measurements on Llama3-8B.

Direct smoothness measure	**Pearson** r (p)	**Spearman** ρ (p)
$\max_{l\in[i,j]}\|J_l\|_2$	**0.905** (4.0×10^{-32})	**0.673** (2.4×10^{-12})
$\sum_{l\in[i,j]}\|J_l\|_2$	0.858 (1.7×10^{-25})	0.656 (1.3×10^{-11})

because it spans the complete candidate set on which DP operates and because the significant *rank* correlations show that the proxy preserves the ordering that schedule selection actually depends on: DP needs to know which intervals are smoother than which, not their absolute Lipschitz constants.

A.8 DP versus Fixed-Interval Schedules and Ablation of the DP Cost Terms

Since the DP search is constrained to a block-size range such as 4 to 6, a natural question is whether the adaptive search contributes anything beyond simply merging evenly spaced blocks of a comparable size. We therefore compare DP against non-adaptive fixed-interval controls, and separately ablate the individual terms of the DP objective.

Fixed-interval controls. Let Δ denote the nominal number of adjacent layers in each merge block. Because removing eight layers may require a smaller final remainder block, we choose block sizes deterministically so that every control matches the same 25% reduction. Two placement rules are considered. **Endpoint-anchored** places the first and last blocks at the shallow and deep boundaries of the stack and divides the untouched layers into near-equal internal gaps. **Centered** uses the same block sizes but distributes the untouched layers across the two external and all internal gaps, keeping every block away from both boundaries. Neither control uses CKA and neither performs any search. All schedules, including DP, are followed by an identical fusion stage with deferred correction disabled.

Table 9: DP against non-adaptive fixed-interval schedules on Llama3-8B at 25% depth reduction. Identical fusion protocol for every row, with deferred correction disabled throughout. Avg. follows the seven-task definition of §3.

Schedule	**Merged intervals**	**PPL (↓)**	**Avg.**
Fixed $\Delta = 4$ (endpoint-anchored)	[0,3]; [14,17]; [29,31]	41919.74	28.25
Fixed $\Delta = 4$ (centered)	[5,8]; [15,18]; [24,26]	55.07	35.52
Fixed $\Delta = 5$ (endpoint-anchored)	[0,4]; [27,31]	139307.68	28.47
Fixed $\Delta = 5$ (centered)	[7,11]; [20,24]	**20.47**	41.98
Fixed $\Delta = 6$ (endpoint-anchored)	[0,5]; [28,31]	96250.90	28.11
Fixed $\Delta = 6$ (centered)	[7,12]; [21,24]	<u>19.60</u>	41.70
DP (ours)	**[19,23]; [24,28]**	37.01	**46.98**

DP improves average downstream accuracy by $+5.00$ **points** over the strongest fixed control (centered $\Delta = 5$), while every endpoint-anchored schedule diverges outright, with perplexity between 4×10^4 and 1.4×10^5. This confirms that boundary layers must not be coarsened and that a naive evenly spaced rule has no mechanism for discovering that. The comparison also furnishes the clearest demonstration in this paper that perplexity alone does not measure preserved capability: centered $\Delta = 5$ attains a *better* perplexity than DP (20.47 vs. 37.01) while its MMLU collapses to near-chance (23.18 vs. DP's 46.87). Any evaluation that ranked these two schedules by perplexity would reach exactly the wrong conclusion.

Cost-term ablation. We next remove individual terms from the DP objective $\mathcal{E}(i, j) = \beta_i \cdot (j - i) \cdot (1 - \mathrm{CKA}(\mathbf{H}_i, \mathbf{H}_j))$ of Eq. (2), keeping all other settings at their defaults.

The CKA gap is the decisive term: removing it changes the selected schedule and inflates perplexity from 37.01 to 123.00. At 25%, both the interval-length penalty and β_i appear inert, but this is *by construction* rather than evidence of redundancy. At that ratio only eight layers are removed and the DP optimum already lies entirely within the deep block $[19, 28]$, where $\beta_i \approx 1$; a depth guardrail cannot bind when nothing is pulling the schedule towards shallow layers. We therefore stress-test at

Table 10: Ablation of the DP cost terms on Llama3-8B. Upper block: at 25% depth reduction. Lower block: at 28%, where shallow layers enter the candidate optimum and the depth-dependent weight β_i becomes active.

Variant	Merged intervals	PPL (↓)	Avg.	ΔAvg.
At 25% depth reduction				
Full DP	[19,23]; [24,28]	37.01	46.98	—
w/o CKA gap	[21,26]; [27,30]	123.00	45.63	−1.35
w/o interval length	[19,23]; [24,28]	37.00	47.03	+0.05
w/o β_i	[19,23]; [24,28]	36.97	46.99	+0.01
At 28% depth reduction				
Full DP (with β_i)	[19,22]; [23,26]; [27,30]	133.20	**42.12**	—
w/o β_i	[10,13]; [21,24]; [27,30]	60.05	40.10	−2.02

28%, where the budget does create that pull. Without β_i, DP descends to merge the shallow interval $[10, 13]$ and loses 2.02 average points, degrading on six of the eight measured tasks (WinoGrande -9.24, ARC-c -3.67, MMLU -3.19); note that it simultaneously achieves a *lower* perplexity (60.05 vs. 133.20), which is again the pattern of perplexity improving while capability degrades. β_i thus functions as a depth guardrail that becomes active exactly when it is needed, consistent with the $e^{L_f L}$ amplification of Proposition 1.

A.9 Sensitivity to the Smoothness Threshold γ

The threshold γ acts as a hard feasibility guardrail: every selected merge interval must satisfy an endpoint CKA of at least γ. We swept γ exhaustively over $[0.85, 0.96]$ on Llama3-8B at 25% depth reduction with deferred correction disabled, including the exact CKA breakpoints at which the feasible interval set changes.

Table 11: Exhaustive sweep of the smoothness threshold γ on Llama3-8B at 25% depth reduction. The default $\gamma = 0.85$ lies in the first range.

γ range	Selected schedule	PPL (↓)	Avg.
$0.85 \le \gamma \le 0.8932$	[19,23]; [24,28]	37.01	46.98
$0.8932 < \gamma \le 0.8958$	[19,24]; [27,30]	88.39	43.72
$\gamma > 0.8958$	infeasible	—	—

Table 12: Compression cost on Llama3-8B at 25% depth reduction, excluding post-compression recovery.

Method	Wall-clock (s)	#GPU	GPU-h	Peak mem/GPU (GB)
LaCo	32.8	1	0.0091	16.78
LLM-Pruner	33.8	1	0.0094	15.93
SLEB	240.3	1	0.0668	17.11
MKA	374.2	1	0.1040	16.60
ShortGPT	2349.2	1	0.6526	17.55
WaLeM (ours)	**601.4**	**8**	**1.3363**	**39.00**

The response is piecewise constant, since γ eliminates a candidate interval only at the point where it crosses that interval's measured CKA. The entire range from 0.85 to 0.8932 therefore selects the same schedule and yields identical performance, leaving no tuning to be done inside this plateau. Only a narrow band of width 0.0026 at the feasibility boundary alters the schedule, degrading perplexity by 2.39× and average accuracy by 3.26 points, beyond which the requested reduction is infeasible. This makes γ a conservative safety constraint rather than a performance knob, although the width of the plateau is a property of the backbone's CKA spectrum: Appendix A.1 discusses why a fixed γ becomes less binding on deeper stacks.

A.10 Computational Cost, Hardware and Runtime Comparison

We quantify every stage of the pipeline separately, so that the required cost of coarsening can be distinguished from the cost of the optional recovery stage.

Measurement setup. We profile Llama3-8B at 25% depth compression ($32 \rightarrow 24$ layers) in bf16 on NVIDIA H20 GPUs. End-to-end wall-clock and memory are measured externally, with GPU memory sampled by `nvidia-smi` every $0.3\,\mathrm{s}$; stage timings and peak allocated memory are additionally recorded from inside the pipeline. The identical measurement and sampling protocol is applied to every method. We ran the five baselines directly from their official repositories without modifying their execution paths; those implementations expose single-GPU compression and therefore ran on one H20. WaLeM contains a training-based fusion stage for which we implemented 8-GPU data parallelism, and therefore ran on eight H20s. To make this difference in parallelism explicit rather than hidden by a wall-clock-only comparison, we report GPU count and GPU-hours alongside wall-clock and peak memory. Evaluation cost is excluded for all methods.

Table 13: Stage-wise cost breakdown for WaLeM on Llama3-8B at 25% depth reduction.

Stage	**Wall-clock (s)**	**Peak mem/GPU (GB)**
CKA collection + computation	129.6	1.63
DP schedule search	**0.03**	**0.03**
Fusion (operator-split optimization)	434.1	33.75
Export	18.9	16.38
Instrumented-stage subtotal	582.6	—
End-to-end compression (external profile)	**601.4**	**39.00**
LoRA deferred correction (optional)	1514.5	34.24
Full pipeline with recovery	**2115.9**	**39.00**

Table 14: Analytic parameter and KV-cache reduction induced by depth compression (bf16). "Layers" is the number of retained residual blocks.

Model	**Depth ratio**	**Layers**	**Params**	**ΔParams**	**KV (KB/token)**	**ΔKV**
Llama2-7B	0.00%	32	6.74B	—	512	—
	18.75%	26	5.52B	−18.0%	416	−18.8%
	25.00%	24	5.12B	−24.0%	384	−25.0%
	31.25%	22	4.71B	−30.0%	352	−31.2%
Llama2-13B	0.00%	40	13.02B	—	800	—
	20.00%	32	10.48B	−19.5%	640	−20.0%
	25.00%	30	9.84B	−24.4%	600	−25.0%
	30.00%	28	9.21B	−29.2%	560	−30.0%
Llama3-8B	0.00%	32	8.03B	—	128	—
	18.75%	26	6.72B	−16.3%	104	−18.8%
	25.00%	24	6.29B	−21.7%	96	−25.0%
	31.25%	22	5.85B	−27.2%	88	−31.2%
Qwen2.5-7B	0.00%	28	7.62B	—	56	—
	21.43%	22	6.22B	−18.4%	44	−21.4%
	25.00%	21	5.98B	−21.4%	42	−25.0%
	28.57%	20	5.75B	−24.5%	40	−28.6%
Qwen2.5-14B	0.00%	48	14.77B	—	192	—
	20.83%	38	12.02B	−18.6%	152	−20.8%
	25.00%	36	11.47B	−22.4%	144	−25.0%
	31.25%	33	10.64B	−28.0%	132	−31.2%

Results. The instrumented stages total $582.6\,\mathrm{s}$; the $18.8\,\mathrm{s}$ difference from the externally profiled end-to-end time is launcher and I/O overhead. Within the stage subtotal, CKA accounts for 22.2%,

DP for 0.005%, fusion for 74.5% and export for 3.2%. End-to-end compression consumes 1.3363 GPU-hours on 8 GPUs; the optional recovery stage adds 3.3656 GPU-hours, bringing compression plus recovery to 4.7019 GPU-hours.

WaLeM thus requires more total compute than the lightweight pruning and merging baselines, though less wall-clock than ShortGPT in this setup. The combinatorial component, however, is not the bottleneck: the $O(L^2)$ DP search contributes 0.03 s and 0.03 GB, so the adaptive scheduling worth $+5.00$ average points (Appendix A.8) is effectively free. The dominant required cost is the one-time offline fusion optimization; all costs are incurred once, offline, per target ratio.

A.11 Parameter and KV-Cache Reduction Induced by Depth Compression

The compression ratios quoted in this paper are *depth*-reduction ratios. Table 14 reports the induced reductions in parameter count and KV-cache footprint, which are analytic counts derived from the published architecture configurations under bf16 storage rather than measured latency. Parameter reduction trails depth reduction because the embedding matrices and the LM head are untouched, an effect most pronounced for large-vocabulary models such as Llama3-8B and the Qwen2.5 family, whereas the KV cache scales exactly with retained depth.

A.12 Detailed Results for Learned Operator Weights

We detail the optimization process of merging coefficients for Llama2-13B with a 30% pruning rate. Given that the merging strategy comprises three distinct blocks, Figure 3–5 depict the evolution of merging coefficients for specific weight types across different layers.

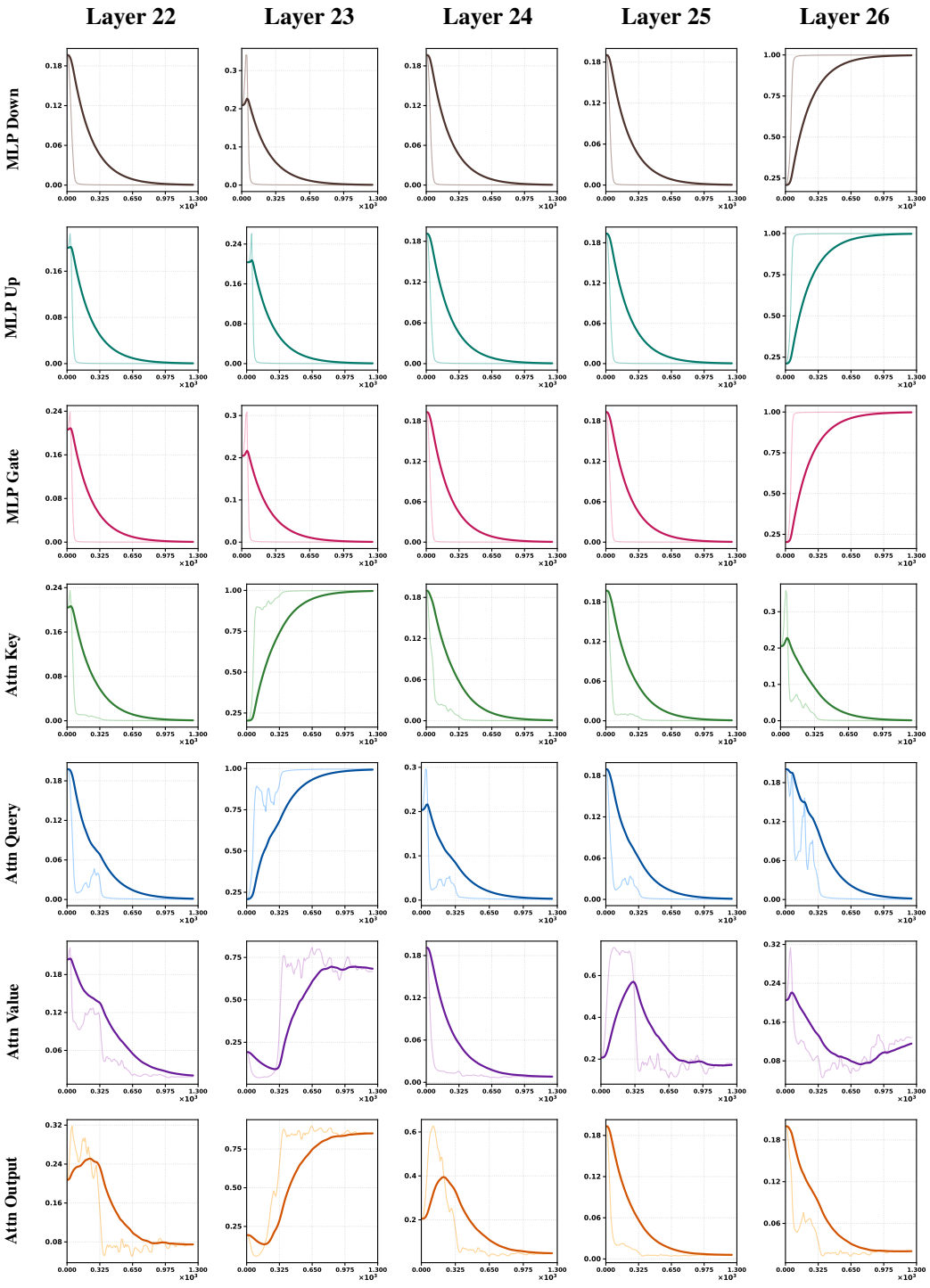

Figure 3: Visualization of the learning process of merging coefficients across layers within Block 1. **Left Axis:** Weight Types. **Top Axis:** Layer Index. Light lines represent raw data; bold lines represent smoothed trends.

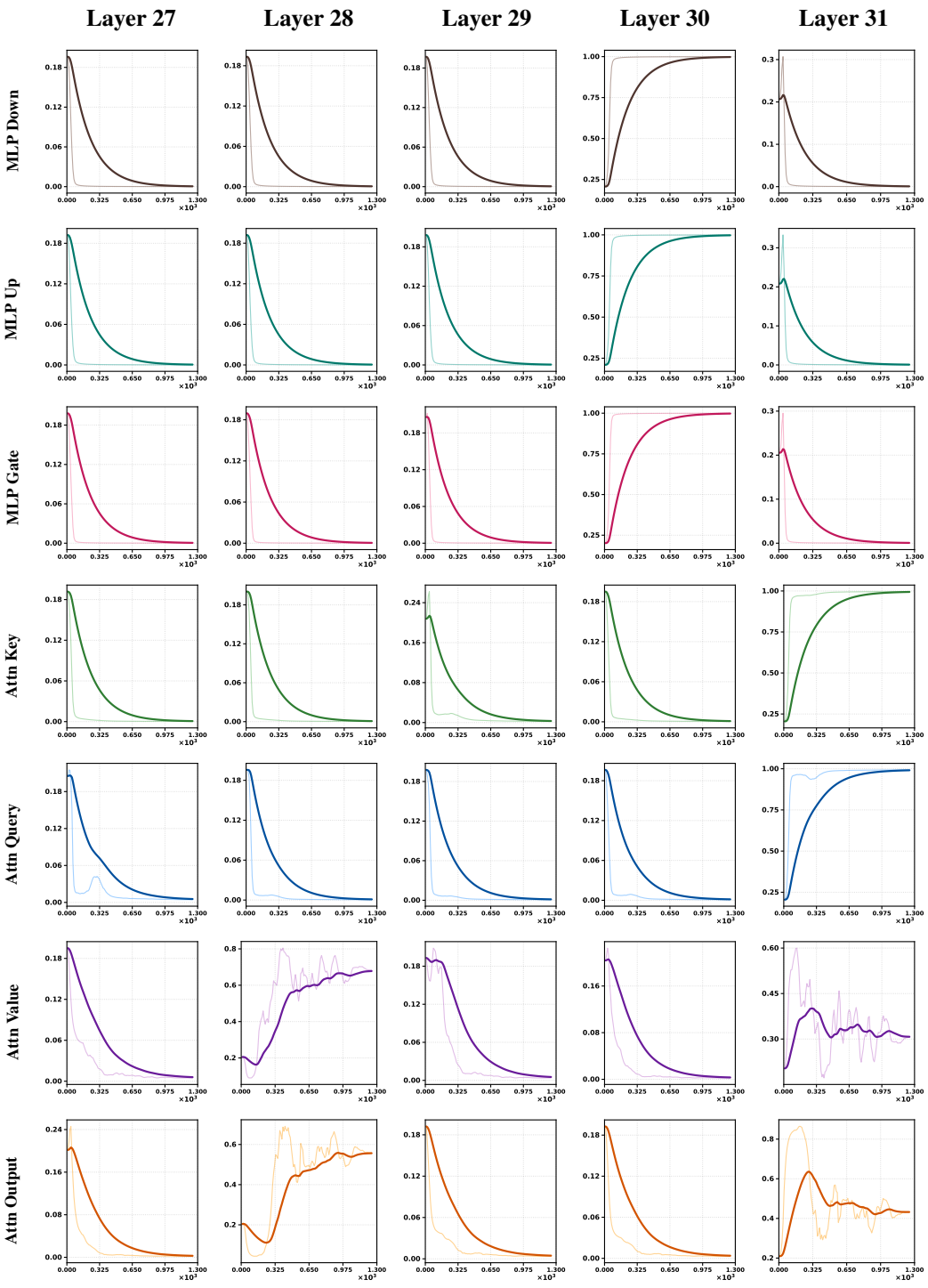

Figure 4: Visualization of the learning process of merging coefficients across layers within Block 2. **Left Axis:** Weight Types. **Top Axis:** Layer Index. Light lines represent raw data; bold lines represent smoothed trends.

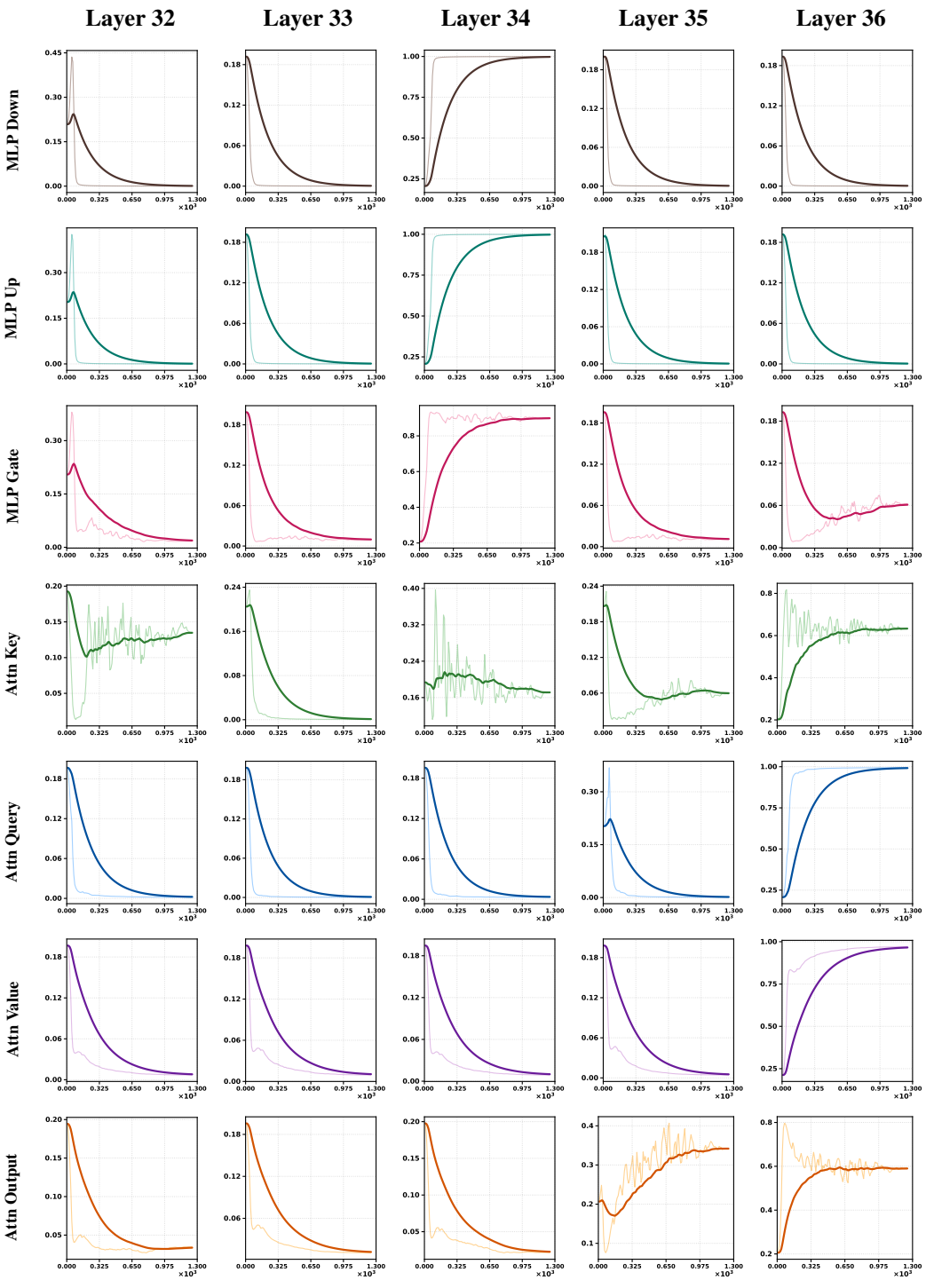

Figure 5: Visualization of the learning process of merging coefficients across layers within Block 3. **Left Axis:** Weight Types. **Top Axis:** Layer Index. Light lines represent raw data; bold lines represent smoothed trends.