WaLeM: Layer Pruning via Weight-Aware Learnable Merging for Large Language Model Compression

Anonymous ACL submission

Abstract

Layer pruning has emerged as a promising technique for compressing Large Language Models (LLMs) by reducing their depth. However, existing methods predominantly rely on direct layer removal or coarse-grained layer merging, where uniform merging coefficients are applied to entire layers. These approaches neglect the distinct functional roles of internal components, often leading to severe feature blurring and performance degradation. To address this, we propose Weight-aware Learnable Merging (WaLeM), a noval framework that transitions from coarse-grained, heuristic layer pruning to fine-grained, optimization-driven layer merging. In WaLeM, we first employs Centered Kernel Alignment (CKA) combined with dynamic programming to globally identify redundant layers for merging, ensuring structural consistency. Subsequently, we introduce a learnable merging mechanism that assigns adaptive, component-specific coefficients to Transformer's weight matrices, optimized via knowledge distillation. Extensive experiments on various models demonstrate that WaLeM significantly outperforms state-of-the-art baselines. Notably, WaLeM preserves complex reasoning capabilities on benchmarks, like GSM8K, even at high compression rates, offering a superior trade-off between efficiency and performance.

1 Introduction

The rapid evolution of Large Language Models (LLMs), represented by the GPT (Brown et al., 2020; Achiam et al., 2023) and Llama (Touvron et al., 2023; Grattafiori et al., 2024) series, has fundamentally transformed natural language processing. When scaled to billions of parameters, these models exhibit remarkable performance across a wide range of tasks, including complex reasoning (Wei et al., 2023; Yao et al., 2023), code generation (Chen et al., 2021; Jimenez et al., 2024), and mathematical problem solving (Cobbe et al., 2021; Lightman et al., 2023). However, the prohibitive computational costs and memory footprints associated with such massive scales pose significant impediments to their deployment, particularly in resource constrained environments such as edge computing platforms.

To address these challenges, substantial research efforts have been directed towards model compression (Cheng et al., 2020; Zhu et al., 2024; Wang et al., 2024). Generally, compression methodologies encompasses quantization (Liu et al., 2023; Dettmers et al., 2023), knowledge distillation (Hinton et al., 2015; Gu et al., 2024), and pruning (Ma et al., 2023; Zhu et al., 2024). Pruning can be further delineated into unstructured pruning, which sparsify individual weights (Frantar and Alistarh, 2023), and structured pruning that eliminate coherent components such as attention heads or entire layers (Ma et al., 2023). Layer pruning, a specific form of structured pruning, has gained prominence for its ability to reduce model depth directly. Unlike other pruning approaches, it delivers tangible reduction in inference latency while preserving architectural compatibility without requiring specialized hardware support (Ling et al., 2024). Capitalizing on these advantages, this study focus on the layer-wise pruning methodology.

Current research on layer pruning primarily addresses two fundamental challenges: selecting specific redundancy layers and determining the pruning execution strategy (Pei et al., 2025). For redundancy layer selection, various studies have introduced distinct metrics, including block influence, perplexity and angular distance (Men et al., 2024; Song et al., 2024; Gromov et al., 2025). Regarding determining the pruning execution strategy, existing methods generally fall into two categories: direct removal (Men et al., 2024; Song et al., 2024) and layer merging (Choudhry et al., 2025; Liu et al., 2025). Compared with layer merging, direct removal usually risks losing valuable

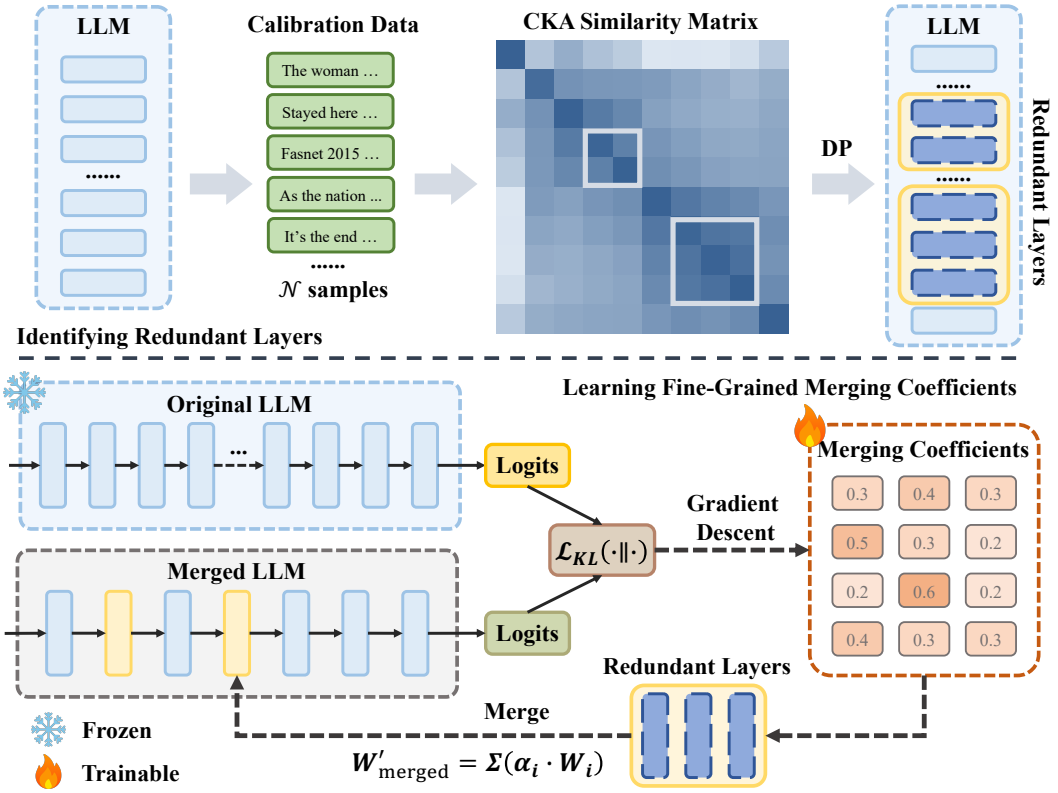

Figure 1: Overview of the Weight-aware Learnable Merging (WaLeM) framework. The framework involves two phases. (1) Identifying redundant layers, which employs Centered Kernel Alignment (CKA) to quantify representational similarity across all layers and applies dynamic programming (DP) to locate blocks with redundant layers; (2) Learning fine-grained merging coefficients, which treats the merging coefficients as learnable parameters and train them through a lightweight knowledge-distillation process.

model structures and interrupts the flow of semantic information (Liu et al., 2025). Consequently, we adopt layer merging strategy to condense multiple layers and preserve information. However, current merging-based approaches still exhibit a fundamental drawback: the merging coefficients are determined in an empirical or heuristic manner, which restricts these approaches to adopting a single, uniform coefficient at each layer-merging step (Yang et al., 2024; Liu et al., 2025). Such coarse-grained merging methods fail to account for the heterogeneous functional responsibilities of sub-modules inside each Transformer layer (Clark et al., 2019; Geva et al., 2021; Meng et al., 2023), inevitably causing feature blurring and functional confusion.

Therefore, we present **W**eight-**a**ware **Le**arnable **M**erging (**WaLeM**), a framework that transition model compression from coarse-grained, heuristic layer pruning to fine-grained, optimization-driven and information-preserving layer merging. In detail, WaLeM begins by using Centered Kernel Alignment (Kornblith et al., 2019) to quantify representational similarity across all layers. Then, a dynamic programming algorithm is employed to analyze this similarity and identify optimal merging blocks, which can bypass the sub-optimality of previous greedy methods (Liu et al., 2025; Pei et al., 2025). Finally, based on the identified optimal merge blocks, we implement fine-grained learnable merging by assigning independent coefficients to different Transformer's weight matrices. These coefficients are treated as learnable parameters and are determined via a lightweight knowledge distillation phase that minimizes Kullback–Leibler divergence against the original LLM to achieve alignment. Such a way allows the merging process to adaptively inherit information based on component sensitivity rather than relying on a fixed coefficient. Extensive experiments on five LLMs across diverse tasks demonstrate that WaLeM outperforms established baselines, maintaining robust capabilities even at high compression rates.

In summary, the main contributions of our work are as follows:

- We propose Weight-aware Learnable Merging (WaLeM), a novel framework designed to compress LLMs through information-

preserving layer merging. WaLeM employs a fine-grained, learnable mechanism that assigns adaptive coefficients to heterogeneous weight components, which allows for precise information inheritance.

- In WaLeM, we develop a globally optimal strategy for layer selection based on Centered Kernel Alignment (CKA) and dynamic programming. By rigorously quantifying the representational similarity across the network, our approach can identify redundant layers from a global perspective.
- We validate the efficacy of WaLeM through comprehensive experiments on multiple benchmarks and various large language models. The results demonstrate that our method consistently outperforms existing baselines by effectively compressing models while preserving their original performance.

2 Related Work

Metrics for Layer Redundancy. 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.

Pruning via Layer Merging. 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.

3 Methodology

The proposed WaLeM framework mainly proceeds in two phases: identifying redundant layers and learning the merging coefficients, which is illustrated in Figure 1.

3.1 Identifying Redundant Layers

We consider a LLM comprising L Transformer layers. Given a dataset $\mathcal{D} = \{x_i\}_{i=1}^{N}$, we extract the output of the l-th Transformer layer as the feature representation. Flattening that feature representation across the batch and sequence dimensions yields $\mathbf{X}^{(l)}$:

$$\mathbf{X}^{(l)} \in \mathbb{R}^{M \times d} \tag{1}$$

where M represents the token count and d is the hidden dimension.

To quantify geometric similarity between the l-th layer and the k-th layer, we employ Linear Centered Kernel Alignment (CKA). In detail, we first compute the Gram matrices:

$$\mathbf{K}^{(l)} = \mathbf{X}^{(l)}(\mathbf{X}^{(l)})^\top, \quad \mathbf{K}^{(k)} = \mathbf{X}^{(k)}(\mathbf{X}^{(k)})^\top \tag{2}$$

Subsequently, we calculate the Hilbert-Schmidt Independence Criterion (HSIC) between the two Gram matrices:

$$\text{HSIC}(\mathbf{K}^{(l)}, \mathbf{K}^{(k)}) = \frac{1}{(M-1)^2}\text{tr}(\mathbf{K}^{(l)}\mathbf{H}\mathbf{K}^{(k)}\mathbf{H}) \tag{3}$$

where $\mathbf{H} = \mathbf{I}_M - \frac{1}{M}\mathbf{1}\mathbf{1}^\top$ is the centering matrix used to remove the mean shift. The CKA similarity score s_{lk} between the l-th layer and k-th layer is defined as the normalized HSIC value:

$$\begin{aligned} s_{lk} &= \text{CKA}(\mathbf{X}^{(l)}, \mathbf{X}^{(k)}) \\ &= \frac{\text{HSIC}(\mathbf{K}^{(l)}, \mathbf{K}^{(k)})}{\sqrt{\text{HSIC}(\mathbf{K}^{(l)}, \mathbf{K}^{(l)}) \cdot \text{HSIC}(\mathbf{K}^{(k)}, \mathbf{K}^{(k)})}} \end{aligned} \tag{4}$$

Computing this for all layer pairs yields a symmetric CKA similarity matrix $\mathbf{S} \in \mathbb{R}^{L \times L}$, where $\mathbf{S}[l, k] = s_{lk}$, capturing the model's hierarchical evolution.

Building on the CKA similarity matrix $\mathbf{S}$, we further formulate identifying redundant layers as a constrained optimization problem on $\mathbf{S}$ to remove exactly K layers by merging non-overlapping diagonal blocks.

In detail, for a candidate block of size B ending at the k-th layer (spanning $[k - B + 1, k]$), we define the redundancy score $r(k, B)$ as:

$$r(k, B) = B^{\alpha} \cdot (\bar{s}_{k,B} - \gamma) \cdot \mathcal{W}_{pos}(\bar{l}) \tag{5}$$

where $\bar{s}_{k,B}$ is the average similarity within the upper triangle of the block's sub-matrix, α is a hyperparameter incentivizing larger block sizes, and γ represents the predefined similarity threshold. To prioritize deeper layers (Gromov et al., 2025), we apply a position-dependent weight based on the block's center $\bar{l} = k - \frac{B-1}{2}$:

$$\mathcal{W}_{pos}(\bar{l}) = 1 + \beta \cdot \frac{\bar{l}}{L} \tag{6}$$

where L is the total number of layers. This weighting mechanism ensures that blocks located in the latter stages of the network contribute more significantly to the overall score.

Finally, a dynamic programming algorithm is employed to derive the optimal strategy. Let $v_{k,j}$ denote the maximum redundancy score achievable within the first k layers, subject to a constraint that exactly j layers are removed. We define the state transition equation as:

$$v_{k,j} = \max \Bigg(v_{k-1,j}, \max_{B \in [B_{\min}, B_{\max}]} \{ v_{k-B, j-(B-1)} + r(k, B) \} \Bigg) \tag{7}$$

The first term corresponds to preserving the k-th layer (i.e., not merging it into a preceding block), where the state simply inherits the score from $v_{k-1,j}$. The second term considers the case where the k-th layer serves as the end of a merging block of size B. Since merging B layers into a single layer results in a reduction of $B - 1$ layers, we transition from the state $v_{k-B,j-(B-1)}$ and add the corresponding redundancy gain $r(k, B)$.

By leveraging a classical dynamic programming algorithm, the problem can be solved exactly and efficiently with an $\mathcal{O}(L^2)$ time complexity. Moreover, such a way enables the identification of redundant layers from a global, rather than local or heuristic, perspective.

3.2 Learning Fine-Grained Merging Coefficients

Once the redundant layers are identified, we determine merging coefficients via a learning phase. Consider a target redundant block of size B involving Transformer layers with indices $l \in [k - B + 1, k]$. A set of weight types $\mathcal{T}$ (e.g., $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o$ in attention) are defined to be merged.

For each block, we initialize a learnable coefficient matrix $\mathbf{F} \in \mathbb{R}^{|\mathcal{T}| \times B}$. To ensure normalized contributions, we apply a row-wise Softmax operation over the layer dimension. The merging coefficient $\alpha_{\tau,l}$ for a specific weight type $\tau \in \mathcal{T}$ and the l-th layer in the block is computed as:

$$\alpha_{\tau,l} = \frac{\exp(\mathbf{F}_{\tau,l})}{\sum_{j=1}^{B} \exp(\mathbf{F}_{\tau,j})} \tag{8}$$

The merged Transformer weight matrix $\widetilde{\mathbf{W}}_{\tau}$ is then obtained by the weighted aggregation of the original parameters $\mathbf{W}_{\tau}^{(l)}$:

$$\widetilde{\mathbf{W}}_{\tau} = \sum_{j=1}^{B} \alpha_{\tau,j} \cdot \mathbf{W}_{\tau}^{(k-B+j)} \tag{9}$$

To preserve the representational capacity of the original LLM $\mathcal{M}$, we align the merged LLM $\mathcal{M}'$ using a calibration dataset $\mathcal{D}_{cal}$ (Kovalev and Tikhomirov, 2025). We minimize the Kullback-Leibler (KL) divergence between the output probability distributions of the two models. The optimization objective is formulated as:

$$\mathcal{L} = \sum_{x \in \mathcal{D}_{cal}} D_{KL}\Big(P_{\mathcal{M}}(y|x) \parallel P_{\mathcal{M}'}(y|x) \Big) \tag{10}$$

where $P_{\mathcal{M}}$ and $P_{\mathcal{M}'}$ denote the softmax probabilities generated by the original model and the merged model, respectively. We update $\mathbf{F}$ via backpropagation while keeping other model parameters frozen.

Finally, consistent with prior studies (Ma et al., 2023), we implement a fast post-fusion recovery phase to mitigate performance degradation. We employ Low-Rank Adaptation (LoRA) (Hu et al., 2021) to efficiently heal the merged model. This lightweight fine-tuning step allows the compressed model to recover its capabilities with minimal computational overhead.

Note that, the comprehensive pseudocode of the WaLeM framework is shown in the Algorithm 1 at the Appendix.

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	64.70
	MKA	18.75%	1469.73	57.39	69.97	59.75	58.45	61.91	44.03	31.96	2.50	48.25
	ShortGPT	18.75%	39.72	68.31	72.80	70.17	60.55	59.47	43.94	34.74	5.61	51.95
	SLEB	18.75%	14.99	63.66	74.86	56.27	25.85	62.25	35.07	32.73	2.58	44.16
	LaCo	18.75%	3167.02	30.45	57.89	59.19	35.09	36.36	28.58	24.11	0.30	34.00
	LLMPruner	18.75%	29.50	69.71	76.17	67.09	**62.04**	70.16	**48.55**	38.76	23.96	57.06
	Ours	18.75%	**13.63**	**74.52**	**76.55**	**72.23**	61.28	**70.96**	47.95	**41.82**	**36.32**	**60.20**
	MKA	25.00%	3557.07	52.94	67.74	60.38	**62.20**	59.00	41.72	29.95	0.45	46.80
	ShortGPT	25.00%	-	31.41	61.15	55.01	34.59	39.65	33.11	24.69	0.23	34.98
	SLEB	25.00%	23.07	57.96	72.14	52.41	26.63	57.87	32.85	31.29	1.90	41.63
	LaCo	25.00%	89.61	60.01	67.90	65.04	60.58	52.27	38.48	32.25	2.12	47.33
	LLMPruner	25.00%	42.06	67.34	73.50	65.82	61.04	**65.99**	**45.56**	37.99	2.20	52.43
	Ours	25.00%	**17.19**	**70.63**	**73.67**	**70.72**	59.71	64.81	44.20	**40.67**	**22.82**	**55.90**
	MKA	31.25%	9999.44	48.19	63.87	60.85	**59.03**	52.90	39.76	29.86	0.00	44.31
	ShortGPT	31.25%	-	33.83	60.45	58.41	36.68	38.24	32.08	26.79	0.99	35.93
	SLEB	31.25%	34.24	52.88	69.26	53.51	26.90	52.36	31.31	29.67	2.35	39.78
	LaCo	31.25%	166.27	48.36	63.71	63.61	25.38	41.46	31,57	29.38	1.29	39.03
	LLMPruner	31.25%	59.94	63.31	71.38	64.40	54.27	**64.14**	**44.71**	36.36	**14.94**	51.69
	Ours	31.25%	**21.84**	**66.15**	**72.52**	**70.09**	58.78	61.49	42.75	**38.85**	5.53	**52.02**
Llama2-13B	Dense	0.00%	7.74	79.65	80.30	72.53	52.39	76.47	48.81	40.48	23.65	59.29
	MKA	20.00%	1411.24	55.44	69.64	62.83	50.69	58.33	40.53	32.15	0.00	46.20
	ShortGPT	20.00%	15.35	72.62	75.63	69.93	52.22	67.00	43.26	38.09	2.35	52.64
	SLEB	20.00%	11.77	70.69	76.33	63.69	30.02	70.62	43.00	37.70	2.81	49.36
	LaCo	20.00%	66.70	61.98	69.53	65.35	37.96	57.74	37.80	33.68	1.67	45.71
	LLMPruner	20.00%	38.53	72.09	76.82	66.30	52.59	71.09	46.42	37.42	11.45	54.27
	Ours	20.00%	**11.63**	**76.33**	**78.13**	**71.35**	**53.57**	**71.84**	**48.55**	**40.57**	**13.50**	**56.73**
	MKA	25.00%	4433.19	50.91	66.59	61.17	50.06	54.29	37.88	31.67	0.00	44.07
	ShortGPT	25.00%	40.40	68.08	72.74	69.14	50.10	60.10	41.89	37.32	0.76	50.02
	SLEB	25.00%	13.90	67.23	**76.66**	60.38	25.31	65.45	39.33	35.31	1.44	46.39
	LaCo	25.00%	130.90	55.80	66.59	64.25	34.45	54.17	35.75	30.53	0.30	42.73
	LLMPruner	25.00%	67.54	67.89	75.14	64.88	**52.27**	68.56	44.03	36.75	**6.82**	52.04
	Ours	25.00%	**13.44**	**74.34**	76.39	**71.82**	51.69	**69.61**	**45.56**	**40.10**	6.67	**54.52**
	MKA	30.00%	8453.01	47.57	64.58	59.75	51.44	52.23	36.09	31.29	0.00	42.87
	ShortGPT	30.00%	65.10	58.31	69.86	68.11	48.44	53.45	36.77	31.96	0.99	45.99
	SLEB	30.00%	16.54	64.49	74.59	61.09	23.37	63.09	36.60	34.45	1.67	44.92
	LaCo	30.00%	205.28	52.07	63.98	63.14	36.87	51.30	35.15	29.57	0.53	41.58
	LLMPruner	30.00%	85.14	65.53	72.85	64.01	51.05	65.74	**43.34**	36.27	3.79	50.32
	Ours	30.00%	**15.11**	**71.60**	**75.24**	**70.01**	**52.76**	**66.50**	43.26	**38.85**	**4.40**	**52.83**

Table 1: Zero-shot performance of Llama models with distinct pruning strategies. "Dense" denotes the unpruned models and "PPL" denotes the perplexity evaluated on Wikitext2. **Bold** fonts indicate the best results, while underlined values represent the sub-optimal performance. Results for Llama2-7B are detailed in Appendix A.3.

4 Experiments

4.1 Experimental Setups

LLMs. We perform experiments on five different-sized LLMs: Llama2-7B (Touvron et al., 2023), Llama2-13B (Touvron et al., 2023), Llama3-8B (Grattafiori et al., 2024), Qwen2.5-7B (Qwen et al., 2025) and Qwen2.5-14B (Qwen et al., 2025).

Datasets and Metrics. We utilize the C4 (Raffel et al., 2023) for identifying redundant layers (500 samples) and learning merging coefficients (10k samples), and utilize Alpaca (Taori et al., 2023) for performance recovery. Using LM Evaluation Harness (Gao et al., 2024), we evaluate on **Perplexity** (WikiText2 (Merity et al., 2016)), **Commonsense Reasoning** (HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), WinoGrande (Sakaguchi et al., 2019), MMLU (Hendrycks et al., 2021), ARC-easy (Clark et al., 2018), ARC-challenge (Clark et al., 2018), RACE (Lai et al., 2017)), and **Mathematical Reasoning** (GSM8K (Cobbe et al., 2021)).

Baselines. We benchmark WaLeM against five state-of-the-art methods. For layer removal, we compare it with LLM-Pruner (Ma et al., 2023), ShortGPT (Men et al., 2024), and SLEB (Song et al., 2024), which utilize depth-based, block influence, and greedy strategies, respectively. For layer merging, we compare it with LaCo (Yang et al., 2024) and MKA (Liu et al., 2025), which employ representation thresholds and manifold learning to guide merging. Specific implementation details are available in Appendix A.1.

4.2 Main Results

Following prior studies (Zhang et al., 2024; Zhong et al., 2025; Lu et al., 2025), we evaluate our proposed WaLeM framework across pruning ratios ranging from approximately 20% to 30%. As presented in Table 1 and Table 2, WaLeM demon-

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	71.06
	MKA	21.43%	182964.77	45.12	61.81	58.09	**40.31**	40.40	35.49	28.61	1.14	38.87
	ShortGPT	21.43%	16.64	65.26	**76.93**	56.04	28.12	68.27	40.44	**37.42**	4.25	47.09
	LaCo	21.43%	797.66	53.05	61.97	53.59	26.48	38.22	30.97	25.26	1.21	36.34
	Ours	21.43%	**13.22**	**66.84**	76.39	**60.22**	39.78	**69.49**	**45.05**	37.32	**15.69**	**51.35**
	MKA	25.00%	607147.37	42.11	60.72	**57.38**	**37.10**	37.92	33.53	26.99	0.99	37.09
	ShortGPT	25.00%	17.96	61.02	74.59	53.99	27.86	63.72	36.43	35.12	2.96	44.46
	LaCo	25.00%	1357.73	48.24	60.77	50.04	23.07	35.14	30.29	22.58	1.21	33.92
	Ours	25.00%	**13.71**	**65.06**	**76.12**	57.06	30.63	**68.27**	**42.66**	**37.22**	**8.95**	**48.25**
	MKA	28.57%	1876698.27	38.60	59.14	56.35	23.00	34.01	31.91	25.36	1.59	33.75
	ShortGPT	28.57%	20.44	56.95	73.45	54.93	**26.58**	61.49	36.09	34.64	2.35	43.31
	LaCo	28.57%	1694.25	44.19	58.92	51.22	23.44	34.85	30.29	23.25	0.61	33.35
	Ours	28.57%	**17.95**	**62.12**	**74.97**	**56.43**	26.14	**65.87**	**41.38**	**35.22**	**4.17**	**45.79**
Qwen2.5-14B	Dense	0.00%	6.66	84.37	81.94	75.77	78.83	81.65	62.29	45.36	44.43	69.33
	MKA	20.83%	400283.16	45.90	65.02	58.96	**77.80**	58.33	41.38	31.39	0.00	47.35
	ShortGPT	20.83%	-	71.47	77.91	60.30	42.60	66.46	43.34	35.41	3.03	50.07
	LaCo	20.83%	1909.78	47.30	61.81	62.04	53.79	42.38	33.79	29.19	0.23	41.32
	Ours	20.83%	**11.95**	**72.31**	**78.18**	**63.22**	44.94	**72.90**	**49.23**	**38.28**	**26.16**	**55.65**
	MKA	25.00%	1289431.81	42.51	63.17	59.59	**53.32**	52.15	38.91	30.53	0.00	42.52
	ShortGPT	25.00%	-	67.00	75.14	56.20	32.97	62.79	37.20	34.55	3.11	46.12
	LaCo	25.00%	12723.14	44.42	62.35	**59.91**	42.08	39.65	35.58	30.53	0.15	39.33
	Ours	25.00%	**13.64**	**69.60**	**77.37**	58.56	39.96	**69.87**	**45.05**	**36.08**	**9.33**	**50.73**
	MKA	31.25%	7517022.80	36.18	57.94	57.85	**32.08**	39.52	34.30	26.32	0.00	35.52
	ShortGPT	31.25%	-	46.86	68.82	51.38	23.64	52.78	30.55	24.11	0.53	37.33
	LaCo	31.25%	775.68	40.32	62.84	56.27	22.99	35.52	30.72	25.26	0.68	34.33
	Ours	31.25%	**18.76**	**63.76**	**74.27**	**59.35**	31.84	**64.31**	**40.78**	**37.03**	**4.78**	**47.02**

Table 2: Zero-shot performance of Qwen models with distinct pruning strategies. "Dense" denotes the unpruned models and "PPL" denotes the perplexity evaluated on Wikitext2. **Bold** fonts indicate the best results, while underlined values represent the sub-optimal performance.

(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**

Table 3: Perplexity comparison between the proposed Global Optimal Fusion strategy and ShortGPT Block Influence across various pruning ratios.

strates superior performance consistently over baseline methods across diverse zero-shot downstream tasks. Specifically, WaLeM achieves the highest average scores while maintaining exceptionally low perplexity on the Wikitext2 dataset. Notably, this advantage becomes increasingly pronounced at higher pruning ratios. While baseline methods often exhibit severe performance degradation or exploding perplexity, WaLeM preserves the model's linguistic coherence and knowledge retention.

Beyond standard multiple-choice benchmarks, we explicitly evaluate mathematical reasoning capabilities using the GSM8K dataset, which is often overlooked in previous pruning studies. WaLeM exhibits remarkable resilience in this challenging tasks. For instance, even with approximately 20% of parameter pruned, our method still retains significant reasoning abilities, particularly on Llama3-8B, achieving results that other baselines fail to match.

Furthermore, we visualize the CKA similarity matrices for Llama3-8B, Llama2-13B, and Qwen2.5-14B in Figure 2. These visualizations explicitly reveal the inherent redundancy present within the original LLMs. By annotating the specific merging strategies with red blocks on the similarity matrices, we demonstrate that our proposed WaLeM method can efficiently capture this redundancy. Subsequently, we present the CKA matrices of the pruned models using these strategies in Figure 3. Comparing this with Figure 2, we observe that the original redundancy in the deeper layers of Llama2-13B, as well as in the middle and deeper layers of Qwen2.5-14B, has been effectively mitigated. This suggests a more efficient information flow within the compressed model.

5 Analyses

5.1 Effectiveness of Identifying Redundant Layers

We investigate the effectiveness of the strategy of identifying redundant layers armed by our dy-

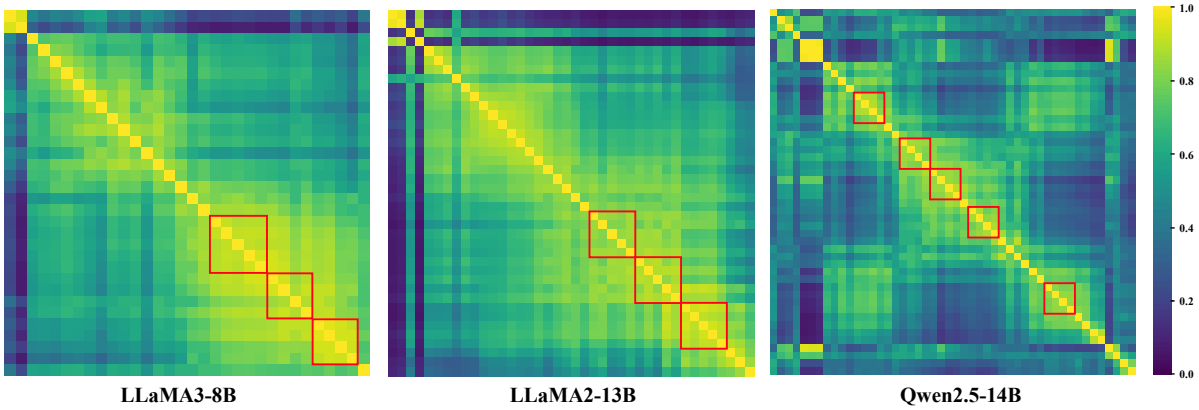

Figure 2: CKA similarity matrices for Llama3-8B, Llama2-13B, and Qwen2.5-14B. The axes represent layer indices arranged from shallow to deep. Red highlighted regions denote the layer fusion strategies adopted at an approximate pruning ratio of 30%.

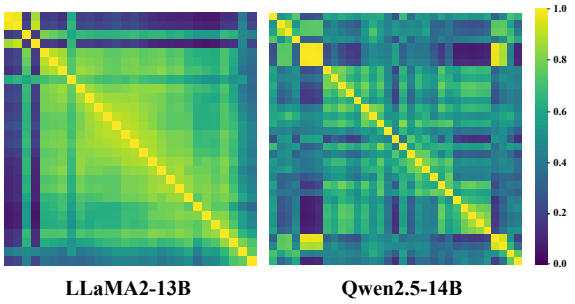

Figure 3: Post-pruning CKA similarity matrices for Llama2-13B and Qwen2.5-14B with an approximate 30% pruning ratio.

namic programming approach. We benchmark our method against the Block Influence metric proposed in ShortGPT (Men et al., 2024). Both strategies serve as the initialization for the subsequent fine-grained learnable merging training. We evaluate the pruned models based on perplexity using the Wikitext2 dataset. Results are presented in Table 3. As shown, our proposed strategy consistently outperforms Block Influence on Llama3-8B, achieving significant margins in most of the pruning ratios. Similar performance advantages are also observed on the Llama2-13B model.

5.2 Significance of Fine-Grained Merging

To validate the efficacy of our proposed strategy, we conduct a controlled experiment where merging is restricted to the final layers to ensure a fair comparison. In this setting, we benchmark our learnable merging coefficients against two baselines, MKA and LaCo. Specifically, MKA directly derives the merging coefficients via similarity, whereas LaCo integrates layer differences. As shown in Table 4, our method achieves significantly lower perplexity

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

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

compared to the baselines, often reducing it by an order of magnitude. This substantial improvement underscores the superiority of the learned coefficients, which is further evidenced by consistently higher average scores across downstream tasks.

5.3 Sensitivity to Merging Granularity

We further investigate the impact of block size on the layer merging strategy. In this experiment, with the pruning rate fixed at approximately 20%, we constrain the block size using minimum and maximum hyperparameters and evaluate the pruned model based on perplexity and downstream task performance. As illustrated in Figure 4, we observe

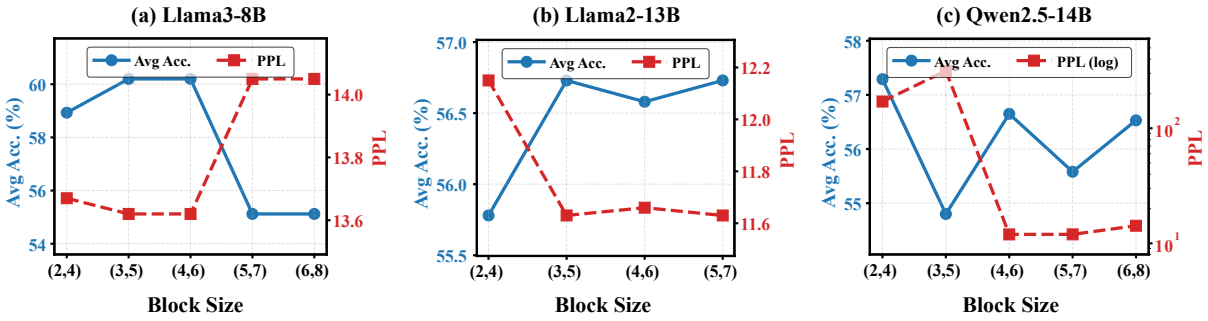

Figure 4: The impact of merging block size on model performance. Results are shown for (a) LLaMA3-8B, (b) LLaMA2-13B, and (c) Qwen2.5-14B. The tuples $(\min, \max)$ on the x-axis denote the minimum and maximum block size constraints used for layer merging.

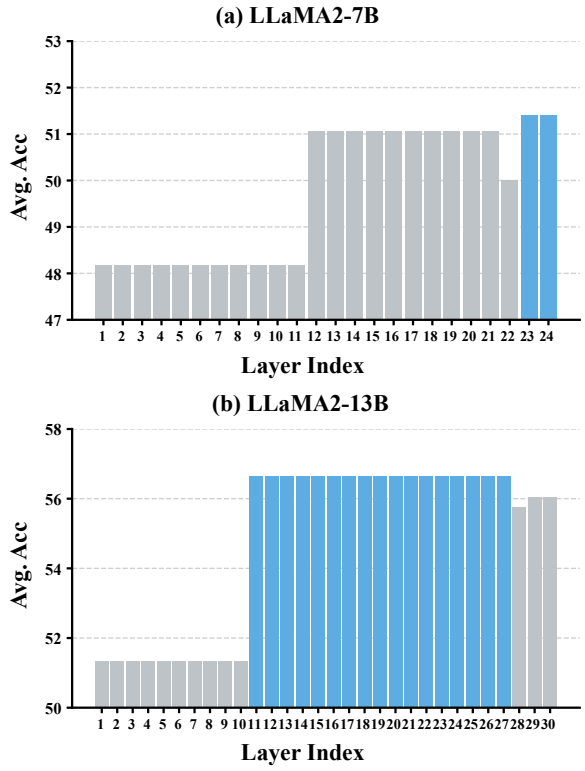

Figure 5: Impact of Merging Starting Depth. (a) and (b) show the results for LLaMA2-7B and LLaMA2-13B, respectively. The x-axis represents the starting layer index. Blue bars indicate the best performance achieved.

that a block size range of 3 to 6 maintains relatively higher performance and lower perplexity. We attribution this to a trade-off between merging efficiency and feature preservation. Excessively small block sizes result in a larger number of blocks dispersed across the model; this leads to more merged layers, causing error accumulation and amplification during forward propagation, which destabilizes performance. Conversely, overly large block sizes compress too many layers into a single layer, resulting in severe information loss and feature blurring. Consequently, we conclude that the block size range of (3, 6) is optimal, as it typically yields two merged blocks, each containing three to five layers, thereby effectively preserving essential features. Further details regarding Figure 4 can be found in Appendix A.4.

5.4 Impact of Merging Starting Depth

We also explore how the starting depth of merging affects model performance. Under a 20% pruning rate, we adjust the hyperparameter controlling the start point to influence the merging strategy discovery phase, effectively constraining the merging process to occur only after a designated layer. This design aligns with established observations (Cai et al., 2025; Xiao et al., 2024; Ma et al., 2025) that earlier layers in large language models are more crucial than later ones. The results in Figure 5 confirm this hypothesis, demonstrating that deferring merging to later layers results in superior efficacy on downstream tasks compared to compressing the earlier layers.

6 Conclusion

In this paper, we presented WaLeM, a framework designed to improve layer pruning by addressing the limitations of coarse-grained merging. By distinguishing the functional roles of weight matrices and integrating learnable merging coefficients, WaLeM enables more precise layer merging. Empirical results demonstrate that WaLeM consistently outperforms state-of-the-art baselines, particularly on reasoning-intensive benchmarks like GSM8K. Furthermore, our analysis underscores the critical impact of merging granularity and position, validating fine-grained layer merging as a robust solution for efficient LLM deployment.

Limitations

Despite WaLeM's promising performance in LLM compression, we acknowledge several limitations remain. First, unlike training-free approaches, WaLeM entails a lightweight optimization phase for merging coefficients. While significantly more efficient than full fine-tuning, this process introduces computational overhead and necessitates access to calibration data. Furthermore, the layer selection mechanism relies on Centered Kernel Alignment (CKA). Since CKA involves Gram matrix computation, its complexity scales quadratically with sequence length. This computational bottleneck may prohibit scaling to extremely long context windows without approximation techniques. Finally, constrained by computational resources, our empirical validation is currently restricted to models up to the 14B parameter scale. Verify the efficacy of WaLeM on larger-scale foundation models is a critical direction for future work.

Declaration of LLM Usage

Large language models were used in a limited, assistive role during the preparation of this manuscript, primarily to improve language and clarity. All scientific content, interpretations, and conclusions are the sole work of the authors, who reviewed and approved the final version. The literature review and reference curation were conducted independently by the authors using published and verifiable sources.

References

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv: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. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *Preprint*, arXiv:2406.02069.

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. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2020. A survey of model compression and acceleration for deep neural networks. *Preprint*, arXiv:1710.09282.

Arjun Choudhry, Chang Liu, Nina Żukowska, Yifu Cai, Mononito Goswami, and Artur Dubrawski. 2025. Layermerge: Modality-agnostic depth pruning for efficient foundation model deployment. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention. *Preprint*, arXiv:1906.04341.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv: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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *Preprint*, arXiv:2301.00774.

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. 2024. The language model evaluation harness.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. *Preprint*, arXiv:2012.14913.

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

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2025. The unreasonable ineffectiveness of the deeper layers. *Preprint*, arXiv:2403.17887.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: Depth pruning for large language models with comparison of retraining methods. *Preprint*, arXiv:2402.02834.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. *Preprint*, arXiv:1905.00414.

Grigory Kovalev and Mikhail Tikhomirov. 2025. Iterative layer-wise distillation for efficient compression of large language models. *Preprint*, arXiv:2511.05085.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *Preprint*, arXiv:1704.04683.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. 2024. Slimgpt: Layer-wise structured pruning for large language models. *Preprint*, arXiv:2412.18110.

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. 2025. Pruning via merging: Compressing llms via manifold alignment based layer merging. *Preprint*, arXiv:2406.16330.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. Llm-qat: Data-free quantization aware training for large language models. *Preprint*, arXiv:2305.17888.

Yao Lu, Yuqi Li, Wenbin Xie, Shanqing Yu, Qi Xuan, Zhaowei Zhu, and Shiping Wen. 2025. The structural scalpel: Automated contiguous layer pruning for large language models. *Preprint*, arXiv:2510.23652.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Preprint*, arXiv:2305.11627.

Zehong Ma, Shiliang Zhang, Longhui Wei, and Qi Tian. 2025. Efficient multi-modal long context learning for training-free adaptation. *Preprint*, arXiv:2505.19812.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *Preprint*, arXiv:2403.03853.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. Locating and editing factual associations in gpt. *Preprint*, arXiv:2202.05262.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Zehua Pei, Hui-Ling Zhen, Xianzhi Yu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. 2025. From pruning to grafting: Dynamic knowledge redistribution via learnable layer fusion. *Preprint*, arXiv: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. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *Preprint*, arXiv:2402.09025.

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

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

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *Preprint*, arXiv:2402.09748.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *Preprint*, arXiv:2402.11187.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. 2024. Finercut: Finer-grained interpretable layer pruning for large language models. *Preprint*, arXiv:2405.18218.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. 2025. Blockpruner: Fine-grained pruning for large language models. *Preprint*, arXiv:2406.10594.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Preprint*, arXiv:2308.07633.

A Appendix

A.1 Details of Implementations

In this section, we elaborate our experimental setup. We utilize the C4 dataset for the initial phases, randomly sampling 500 samples for CKA similarity matrix computation and 10,000 samples for learning the merging coefficients, with the random seed fixed at 42. Regarding the merging strategy, we constrain the block size between 4 and 6 for most models, while adjusting this range to 5 and 7 for Llama2-13B. We set the similarity threshold to 0.85 and α to 1.5 in Equation 5, and β to 0.3 in Equation 6. The merging start depth is initialized at layer 17 for Llama3-8B and Llama2-7B, and at layer 22 for Llama2-13B. Conversely, we set the start depth to 0 for Qwen models to accommodate their distinct layer-wise redundancy patterns observed in the CKA analysis. During the coefficients learning process, we train for 2 epochs with a batch size of 4, gradient accumulation of 4, a learning rate of 0.1, and a temperature of 1. For the post-pruning recovery phase, we fine-tune the model on the Alpaca dataset for 2 epochs using LoRA. The hyperparameters include a learning rate of 1×10^{-4}, LoRA rank $r = 8$, $\alpha = 16$, and a dropout of 0.05, with a batch size of 4 and gradient accumulation of 16. We adhere to the official configurations for all baseline methods with two exceptions. For LLM-Pruner, we adopt its layer pruning configuration; for LaCo, we terminate the iteration early once the desired pruning ratio is achieved.

A.2 Details of Datasets

A.2.1 Pruning Datasets

C4 (Colossal Clean Crawled Corpus) (Raffel et al., 2023) is a massive, cleaned version of the Common Crawl web corpus. A subset is sampled to compute the CKA similarity matrix, while a separate subset is employed to optimize the merging coefficients. This dataset serves as a standard benchmark for measuring perplexity on continuous text streams.

A.2.2 Post Pruning Recovery Datasets

Alpaca (Taori et al., 2023) To restore model performance after pruning, we employ the Alpaca dataset for instruction tuning. This dataset consists of synthetic instruction-response pairs designed to enhance the capability of the model to follow user commands and maintain response quality.

Algorithm 1 Weight-Aware Learnable Merging Framework

Input: Pre-trained LLM $\mathcal{M}$ with L layers, Calibration Dataset $\mathcal{D}$, Pruning Target K
Output: Compressed Model $\mathcal{M}'$

 # Phase 1: Identifying Redundant Layers
Sample small subset $\mathcal{D}_{sim} \subset \mathcal{D}$ where $|\mathcal{D}_{sim}| = 500$ {See Appendix A.1}
Extract hidden states $\mathbf{X}^{(l)}$ for all layers using $\mathcal{D}_{sim}$
for each pair (l, k) in layers **do**
  Compute Gram matrices $\mathbf{K}^{(l)}, \mathbf{K}^{(k)}$ based on $\mathcal{D}_{sim}$
  Compute CKA similarity s_{lk} via HSIC in Eq. 4
end for
Construct similarity matrix $\mathbf{S} \in \mathbb{R}^{L\times L}$
Initialize DP table V and solve for optimal blocks $\mathbb{B}$ using Eq. 7
Output: Set of non-overlapping blocks $\mathbb{B}$
 # Phase 2: Learning Merging Coefficients
Sample larger subset $\mathcal{D}_{train} \subset \mathcal{D}$ where $|\mathcal{D}_{train}| = 10,000$ {See Appendix A.1}
Initialize merging coefficients $\mathbf{F}_\tau \in \mathbb{R}^{|\mathcal{T}|\times B}$ for blocks in $\mathbb{B}$
Freeze original LLM parameters
while not converged **do**
  Sample batch x from $\mathcal{D}_{train}$
  for each block in $\mathbb{B}$ **do**
    Compute merging coefficients via Eq. 8: $\alpha_{\tau,l} = \text{Softmax}(F_{\tau,l})$
    Merge weights via Eq. 9: $\tilde{W}_\tau = \sum_{j=1}^{B} \alpha_{\tau,j} \cdot W_\tau^{(j)}$
  end for
  Forward pass to get logits $P_{\mathcal{M}}(x)$ and $P_{\mathcal{M}'}(x)$
  Compute Loss in Eq. 10: $\mathcal{L} = \mathcal{D}_{KL}(P_{\mathcal{M}}||P_{\mathcal{M}'})$
  Update $\mathbf{F}$ via gradient descent
end while
Apply learned α to permanently merge layers
Fine-tune $\mathcal{M}'$ using LoRA on instruction data
return $\mathcal{M}'$

A.2.3 Evaluation Datasets

We implement the LM Evaluation Harness (Gao et al., 2024) with default configurations to assess performance across various downstream tasks.

WikiText2 (Merity et al., 2016) This dataset evaluates language modeling capabilities on high-quality encyclopedic content. It retains original

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	54.80
	MKA	18.75%	1008.68	54.59	66.81	59.19	33.29	55.81	38.05	30.24	0.76	42.34
	ShortGPT	18.75%	27.04	65.18	72.03	64.64	25.90	57.45	38.31	33.49	1.74	44.84
	SLEB	18.75%	14.77	64.89	74.48	59.83	26.48	59.68	34.64	33.78	2.05	44.48
	LaCo	18.75%	100.73	58.21	70.46	62.43	31.68	57.95	37.71	31.77	0.45	43.83
	LLMPruner	18.75%	30.36	68.27	74.16	63.22	34.49	65.70	41.21	35.60	**6.37**	48.63
	Ours	18.75%	**14.69**	**71.52**	**75.79**	**67.25**	**40.14**	**66.50**	**41.89**	**39.43**	5.00	**50.94**
	MKA	25.00%	1842.92	49.81	65.72	57.70	25.06	52.10	36.26	30.05	0.38	39.64
	ShortGPT	25.00%	47.19	59.53	68.44	66.06	23.49	52.78	34.04	32.06	1.52	42.24
	SLEB	25.00%	20.33	60.91	71.60	58.56	27.66	54.21	32.08	31.67	1.52	42.28
	LaCo	25.00%	145.97	51.37	65.51	59.12	31.33	52.99	34.22	31.29	0.30	40.77
	LLMPruner	25.00%	40.57	65.63	**72.42**	62.35	25.14	61.83	**39.33**	35.12	**4.32**	45.77
	Ours	25.00%	**20.28**	**67.06**	72.09	**68.19**	**38.72**	**62.25**	38.57	**38.56**	3.26	**48.59**
	MKA	31.25%	3569.59	46.11	61.53	58.56	30.44	48.40	32.42	31.39	0.30	38.64
	ShortGPT	31.25%	119.99	50.84	63.11	63.54	37.22	45.03	32.17	32.63	0.53	40.63
	SLEB	31.25%	28.78	54.27	68.50	54.22	25.64	50.25	31.06	30.62	1.44	39.50
	LaCo	31.25%	117.32	46.32	61.10	60.54	34.39	37.92	32.08	31.48	1.06	38.11
	LLMPruner	31.25%	51.49	61.89	70.02	61.64	28.31	**56.82**	**35.92**	35.02	2.12	43.97
	Ours	31.25%	**26.88**	**62.04**	**70.18**	**66.22**	**37.91**	54.88	34.39	**37.03**	**2.20**	**45.61**

Table 5: Zero-shot performance of Llama2-7B model with distinct pruning strategies. "Dense" denotes the unpruned models and "PPL" denotes the perplexity evaluated on Wikitext2. **Bold** fonts indicate the best results, while underlined values represent the sub-optimal performance.

article structures to test the ability of the model to handle long-term dependencies.

HellaSwag (Zellers et al., 2019) As a benchmark for commonsense natural language inference, HellaSwag challenges models to select the most plausible continuation of a sentence. It employs adversarial filtering to ensure the task requires genuine reasoning rather than simple statistical matching.

PIQA (Bisk et al., 2019) Focusing on physical commonsense, PIQA presents questions regarding the interaction of physical objects. The model must identify the plausible solution to a specified goal, testing its grounding in real-world physics.

Winogrande (Sakaguchi et al., 2019) This dataset is designed to evaluate robust commonsense reasoning and pronoun resolution. It filters out biases prevalent in earlier datasets to ensure performance relies on reasoning capabilities rather than algorithmic artifacts.

MMLU (Hendrycks et al., 2021) MMLU serves as a comprehensive test of general world knowledge and problem-solving ability. It covers a wide array of subjects across STEM, the humanities, and social sciences, ranging from elementary to professional levels.

ARC (Clark et al., 2018) The AI2 Reasoning Challenge (ARC) evaluates grade-school scientific knowledge. We report results on both the Easy subset (ARC-e), which tests basic fact retrieval, and the Challenge subset (ARC-c), which requires deep reasoning to solve complex scientific scenarios.

RACE (Lai et al., 2017) Collected from English reading comprehension examinations, RACE requires diverse reasoning skills including summarization, inference, and detail extraction to select correct answers based on provided passages.

GSM8K (Cobbe et al., 2021) This benchmark assesses multi-step mathematical reasoning. It consists of high-quality grade school math word problems that require the model to generate a chain of thought to reach the correct numerical solution.

A.3 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.

A.4 Detailed Results for Sensitivity to Merging Granularity

Table 6 details the numerical results corresponding to Figure 4. 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

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}

Table 6: Performance comparison of pruned models with different merging block sizes at a 20% pruning ratio. "Block Indices" column details the specific layer indices contained within each merged block.

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).

Let ϵ denote the forward propagation error arising from merging 2 layers. Assuming the fusion error scales linearly with the number of layers, the error for a 4-layer merge is 2ϵ. Let λ represent the feature amplification factor of the network layers.

For the Small Block Strategy, the error from the first merged group is amplified by λ during forward propagation and accumulates with the error from the second group:

$$E_{small} = \lambda \cdot \epsilon + \epsilon = \epsilon(\lambda + 1) \tag{11}$$

In contrast, the Large Block Strategy introduces the fusion error only once:

$$E_{large} = 2\epsilon \tag{12}$$

Comparing the two strategies:

$$\frac{E_{small}}{E_{large}} = \frac{\epsilon(\lambda + 1)}{2\epsilon} = \frac{\lambda + 1}{2} \tag{13}$$

Since deep networks inevitably amplify input perturbations ($\lambda > 1$), we derive:

$$\frac{\lambda + 1}{2} > 1 \implies E_{small} > E_{large} \tag{14}$$

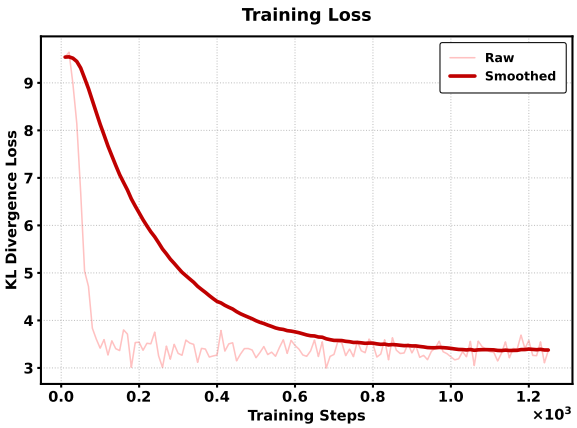

Figure 6: KL divergence loss during the merging coefficients learning phase on Llama2-13B with a 30% pruning rate.

This inequality theoretically validates that merging of small blocks results in greater total error compared to direct merging with larger blocks.

A.5 Pseudocode of WaLeM framework

We provide the comprehensive pseudocode for the WaLeM framework in Algorithm 1. This process encompasses the three distinct phases: global redundancy analysis using CKA, learnable merging training via knowledge distillation, and the final post-fusion recovery.

A.6 Details of Training Process

We detail the optimization process of merging coefficients for Llama2-13B with a 30% pruning rate. Figure 6 illustrates the KL divergence loss during this phase. Given that the merging strategy comprises three distinct blocks, Figure 7–9 depict

the evolution of merging coefficients for specific weight types across different layers.

A.7 Details of Software Environment

All experiments were implemented using Python 3.11.14 in a Conda environment. The major software components and their corresponding versions are detailed as follows:

Software	Version
PyTorch	2.9.1
Transformers	4.57.1
CUDA Libraries	12.8
PEFT	0.18.0
LM Eval	0.4.9.1
Accelerate	1.11.0
Datasets	4.4.1
Numpy	2.3.4
Pandas	2.3.3

Additional dependencies include scikit-learn (1.7.2), wandb (0.23.0), and various utilities for data processing and model optimization.

A.8 Case Study

Table 7 presents qualitative samples generated by Llama3-8B at a 31.25% pruning rate. We observe that our WaLeM method effectively mitigates the redundancy that typically leads to semantic drift or tangential digression in the original model. As a result, the pruned model generates text with improved discourse structure and logical coherence, even at high compression rate. Although minor numerical hallucinations persist, the pruning process demonstrates a significant de-noising effect: it appears to filter out low quality corpus patters, such as web crawl artifacts, and yields cleaner and more focused outputs.

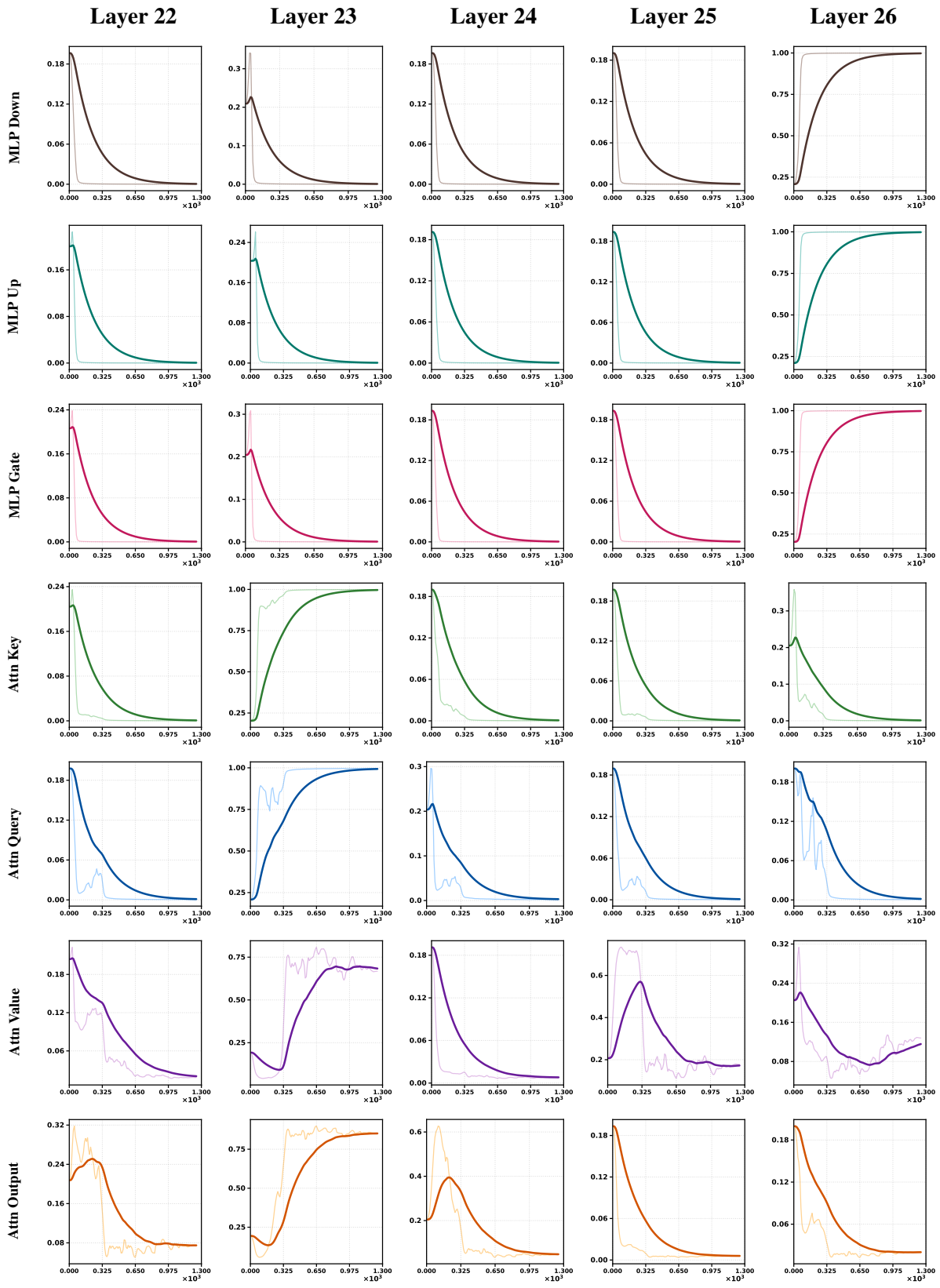

Figure 7: Visualization of the learning process of merging coefficients across layers within Block 1. **Left Axis:** Weight Types. **Top Axis:** Layer Index. Light lines represent raw data; bold lines represent smoothed trends.

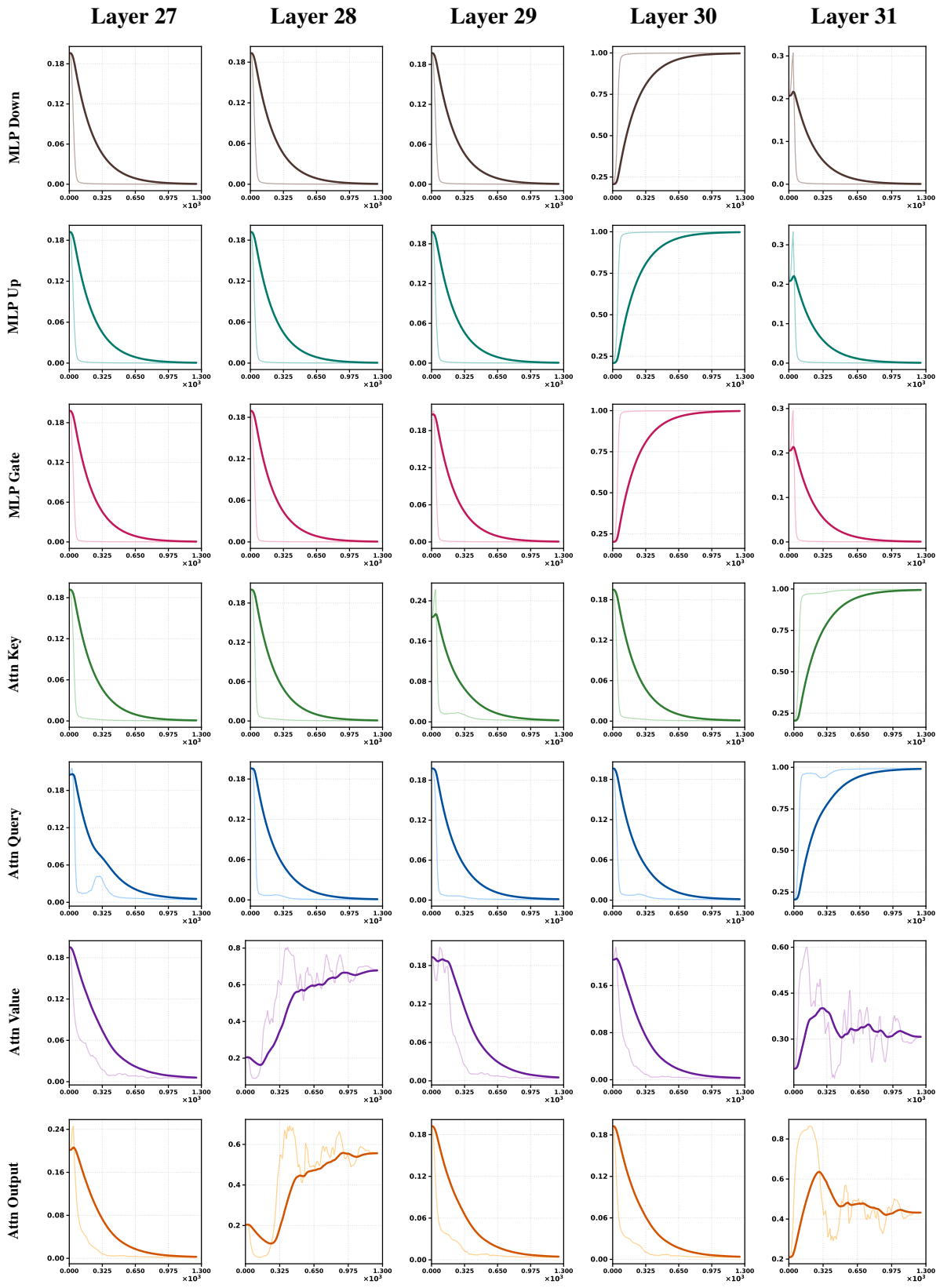

Figure 8: Visualization of the learning process of merging coefficients across layers within Block 2. **Left Axis:** Weight Types. **Top Axis:** Layer Index. Light lines represent raw data; bold lines represent smoothed trends.

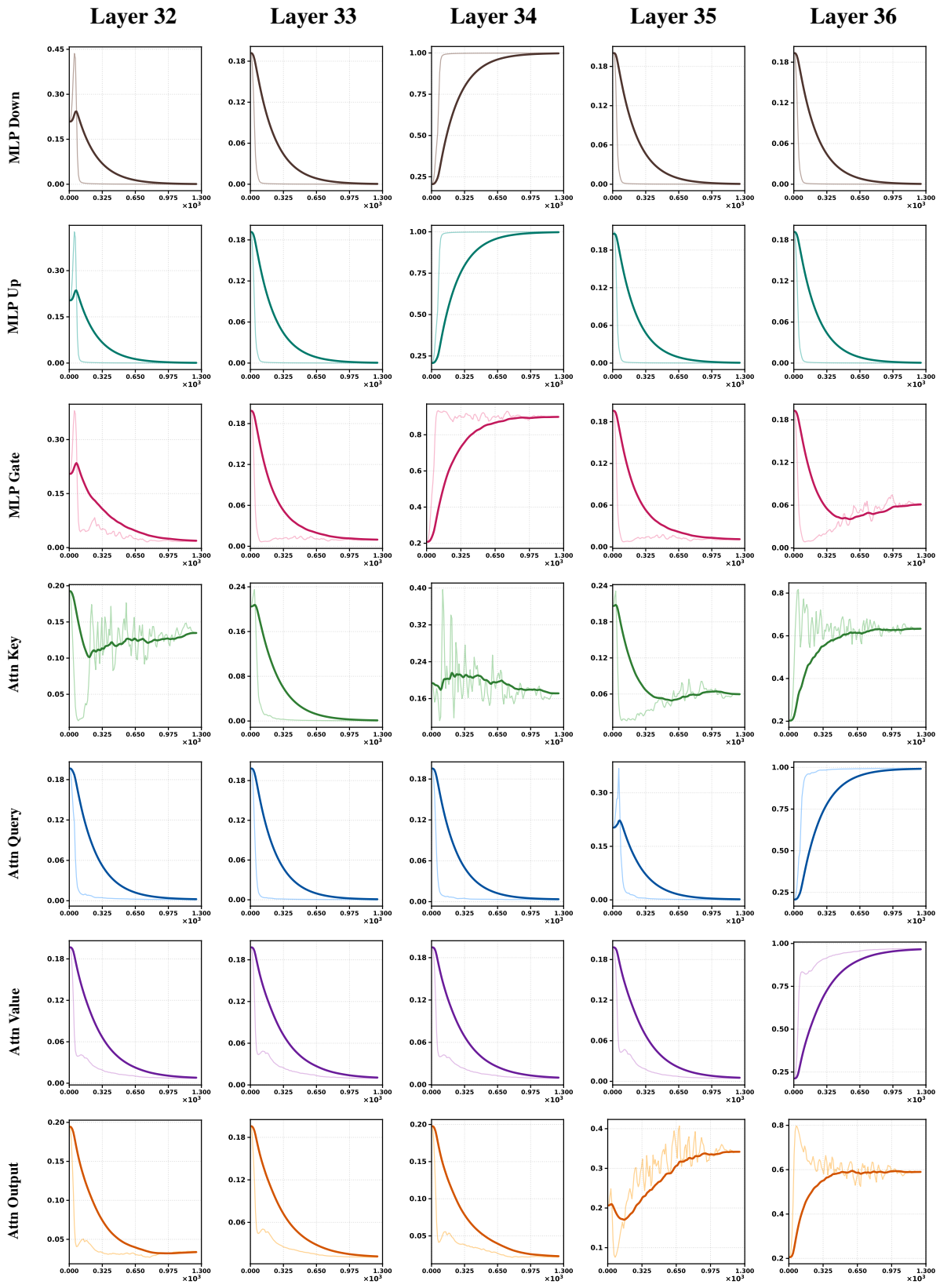

Figure 9: Visualization of the learning process of merging coefficients across layers within Block 3. **Left Axis:** Weight Types. **Top Axis:** Layer Index. Light lines represent raw data; bold lines represent smoothed trends.

Original	Creating a strong brand identity is essential for any modern business. While a logo is a good start, the most successful companies understand that branding is actually about more than just what you see and hear. The best brands also consider how they are perceived by their customers. When it comes to establishing trust in your audience, there's nothing like having an expert speak on behalf of your company. So why not use the opportunity? Hiring spokespersons can be expensive and time-consuming – but these days, video content has become increasingly important as well! In this article we'll discuss 5 benefits associated with hiring experts who will represent yourself or organization through various means including social media campaigns such as YouTube videos which offer great exposure at little cost when compared against traditional advertising methods (TV spots). 1. Expertise: Hiring someone else speaks volumes If you're looking for something specific, then chances are other people have found success already. For example if I want some help figuring out my taxes because tax season is coming up soon – would anyone think twice before going online searching "tax accountant near me"? Well no one really needs to since all sorts of websites exist dedicated solely towards finding answers from professionals within minutes rather than hours waiting around while trying things themselves without guidance first hand experience etcetera...
Pruned	Creating a strong brand identity is essential for any modern business. While a logo is a good start, the most successful companies understand that branding is actually about creating unique and memorable experiences for their customers. Here's how to create an effective brand image: 1. Identify your target audience: Before you can create anything from scratch it's important to know who you're targeting in terms of demographics such as age group or job titles etc 2. Define your value proposition: To have a clear idea which makes us different, we need to define our value proposition - what exactly do they pay for? 3. Create visual elements: After identifying your target audiences and defining values it becomes easy to choose colors, fonts, images & design style that resonates with them 4. Develop your digital presence: With more than 9 billion users worldwide on social media platforms and millions every second considering whether there is enough room left between two sides? Social media has become one of the easiest ways to reach out to potential buyers. So having presence at all possible platforms like Facebook, Instagram, Twitter, LinkedIn, YouTube helps make a short attention span user engaged for longer time longer period before turning away by providing engaging and informative information! 5. Engage with customer feedback: When you want people to buy your
Original	Explain the concept of 'Photosynthesis' to a 5-year-old child using a simple analogy. Photosynthesis is like cooking. Plants use sunlight as energy, take in carbon dioxide from air and water from soil, make their own food (glucose) which they need for growth, just like we cook our meals at home. You might also want to ask your kid how plants grow bigger every day? It's because it needs glucose or sugar that comes out of photosynthesis! So next time you are outside playing with them, tell them about this amazing process called Photosynthesis! Do share any other good examples if you have heard so far! - [](https://www.facebook.com/sharer/sharer.php?u=https://aakashdigitalsrv1.meritnation.com/ask-answer/qu) - [](/board/detail/board.do?gnbrd=&cmd=attach&atch_file_id=17007054) The following reaction represents photosynthesis: 6CO _{2} +12H _{2 } O $\rightarrow$ C _{6} H _{12} O _{6 } +6O _{2} We can represent it by considering each molecule on its side C O $\rightarrow$ C O CO CO CO OH HO
Pruned	Explain the concept of 'Photosynthesis' to a 5-year-old child using a simple analogy. Photosynthesis is like cooking. Plants use sunlight as an ingredient and feed it into their bodies, then produce oxygen which can be used by humans or animals that breathe it out at all time while we eat plants such as fruits, vegetables, rice, potatoes etc. So instead of us cooks directly from our food sources, plants act as intermediaries for their own nutritional needs. It's just like how people cook and consume different foods depending on what they taste or have preferences with regard to healthy diets and nutrition. The same applies when photosynthesis occurs within the plant itself. Like how things are put in order through the process of cooking so too do photosynthesis involve intricate processes involving light energy (sun), carbon dioxide air, water molecules being converted into glucose (sugar) and oxygen, essential nutrients for both human and plants alike.

Table 7: Generated samples from original Llama3-8B and the model pruned via WaLeM with a 31.25% pruning rate.