LayerMerge: Modality-Agnostic Depth Pruning for Efficient Foundation Model Deployment

Arjun Choudhry[*1,2] **Chang Liu**[*1] **Nina Zukowska**[1] **Yifu Cai**[1]
Mononito Goswami[1] **Artur Dubrawski**[1]

[1]Auton Lab, Carnegie Mellon University [2]Georgia Institute of Technology

achoudhry32@gatech.edu

Abstract

Large foundation models face deployment challenges in resource-constrained environments. While width pruning typically outperforms depth pruning, we introduce LayerMerge, a simple modality-agnostic depth pruning technique that closes the performance gap with width pruning while providing linear reductions in inference time and memory. Extensive benchmarks show LayerMerge preserves emergent abilities under aggressive compression, maintaining most of the original performance while reducing model depth by up to 90%.

1 Introduction

The increasing size of Large Language Models (LLMs) makes deployment computationally prohibitive in resource-constrained environments. While various pruning methods exist, they often require extensive retraining or complex optimization. Existing depth pruning approaches are rarely evaluated beyond 50% compression, limiting understanding of their potential under extreme sparsity.

We introduce LayerMerge, a simple depth pruning approach that preserves model capabilities through strategic layer merging rather than elimination. LayerMerge analyzes contextual similarity through activation patterns, identifying representations that can be efficiently combined without semantic loss. Unlike approaches that remove layers entirely, LayerMerge preserves learned knowledge by combining compatible layer blocks using Principal Component Analysis, using only forward passes. LayerMerge offers distinct advantages: linear reductions in inference time and memory, hardware-agnostic acceleration, and simplified architectures maintaining dense operations without specialized sparse kernels. We evaluate LayerMerge on Llama-2 and Llama-3.1, demonstrating improved performance compared to other depth pruning methods with aggressive compression up to 90%. We also evaluate LayerMerge on MOMENT[1], a recent time series foundation model, to verify the modality-agnostic compression capability. Our contributions are: 1. simple yet effective layer merging with superior computational benefits; 2. systematic evaluation of emergent abilities under extreme compression; and 3. modality-agnostic validation establishing broader applicability. We open-source our code in https://anonymous.4open.science/r/SimpleDepthPruning-45EE/.

2 Related Work

Model pruning offers a practical strategy for scaling LLMs down for deployment. Early attempts focus on width pruning, which sparsifies LLMs by removing weights within layers. For example, Wanda[2] removes weights with smallest magnitudes multiplied by corresponding input activations. SparseGPT[3] prunes LLMs one-shot, treating pruning as large-scale sparse regression. More recently, depth pruning provides an alternative by removing entire layers, offering more uniform and hardware-agnostic acceleration. ShortGPT[4] selects layers based on importance scores measuring similarity between each layer's input and output. Shortened Llama[5] removes layers based on their influence on model output perplexity using calibration examples. LaCo [6] merges multiple

[*]Equal Contribution

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

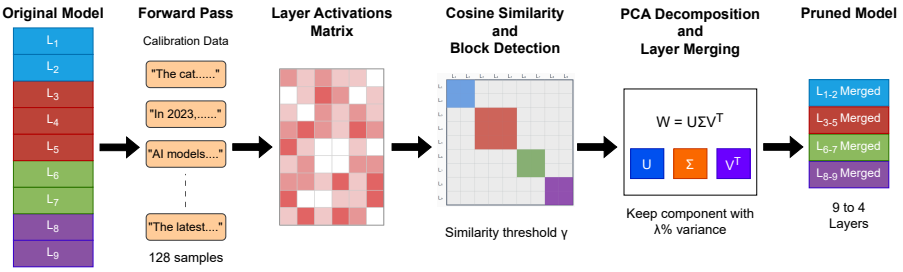

Figure 1: Overview of our `LayerMergeAct` approach.

subsequent layers into a single preceding layer, adding parameter differences iteratively guided by output similarity. However, these methods use different benchmark sets, hindering direct comparison. Limited comparisons show width pruning's performance advantages and depth pruning's efficiency advantages [7]. Our work proposes simple depth pruning approaches that retain depth pruning benefits without significant performance drops compared to width pruning.

3 Methodology

We present simple depth pruning techniques for efficient LLM compression. Our approach targets transformer decoder layers while preserving auxiliary components (embedding, normalization layers) to maintain architectural integrity. Let $\{L_1, L_2, \cdots, L_m\}$ denote the original decoder sequence. Our objective is constructing a compressed model with layers $\{\tilde{L}_1, \tilde{L}_2, \cdots, \tilde{L}_n\}$ where $n < m$. We investigate aggressive pruning where $n \leq \frac{m}{2}$, enabling substantial parameter reduction.

Our framework supports selective pruning within any contiguous subset $\{L_a, L_{a+1}, \cdots, L_b\}$ where $[a, b] \subseteq [1, m]$. The architecture preserves boundary layers $\{\tilde{L}_1, \cdots, \tilde{L}_{a-1}\}$ and $\{\tilde{L}_{b+1}, \cdots, \tilde{L}_m\}$ when non-empty, ensuring compatibility with existing architectures. Our methods operate on module-specific parameter tensors within each transformer layer. We define module type set T =[`self_attn.k_proj`, `self_attn.v_proj`, `self_attn.q_proj`, `self_attn.o_proj`, `mlp.gate_proj`, `mlp.up_proj`, `mlp.down_proj`, `input_layernorm`, `post_attention_layernorm`], encompassing all learnable components. For module type $t \in T$ in layer L_i, we denote parameters as W_i^t.

`LayerCluster`: K-Medoids Clustering We formulate layer selection as clustering in parameter space. Given target range $[a, b]$ and module type t, we apply k-medoids clustering to $[W_a^t, W_{a+1}^t, \cdots, W_b^t]$, yielding k cluster centers at indices $\{j_1, \cdots, j_k\} \subseteq \{a, a+1, \cdots, b\}$, representing the most representative layers. The compressed architecture retains $\{L_1, \cdots, L_{a-1}, L_{j_1}, \cdots, L_{j_k}, L_{b+1}, \cdots, L_m\}$.

`LayerMerge`: PCA-Based Merging We propose dimensionality reduction that synthesizes new layers from existing parameter distributions. For each module type t within $[a, b]$, we stack $[W_a^t, W_{a+1}^t, \cdots, W_b^t]$ and perform PCA. We retain the first k components to generate k synthetic layers $\tilde{L}_{j_1}, \cdots, \tilde{L}_{j_k}$ substituting the original set, where $\tilde{L}_{j_r}$ parameters correspond to the r-th component across all module types. The architecture becomes $\{L_1, \cdots, L_{a-1}, \tilde{L}_{j_1}, \cdots, \tilde{L}_{j_k}, L_{b+1}, \cdots, L_m\}$.

`LayerMergeAct`: Activation-Guided Adaptive Merging Unlike parameter-only approaches, `LayerMergeAct` incorporates activation patterns using calibration data $\mathcal{D}$ of 128 random C4 samples [8]. We execute forward passes for each $x \in \mathcal{D}$, capturing activations $\{z_1^x, z_2^x, \cdots, z_m^x\}$. We calculate cosine similarity between pairs (z_i^x, z_j^x) creating similarity matrix $S(x)$. The aggregated matrix is $S = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} S(x)$. Using threshold γ, we iteratively identify diagonal blocks in S where: (i) all values exceed γ, and (ii) minimum value is maximized among qualifying blocks. Details are in Algorithm 1. After extracting indices $\mathcal{I} = \{(s_1, e_1), (s_2, e_2), \cdots, (s_r, e_r)\}$, we apply `LayerMerge` to each block with modification: global hyperparameter λ retains components preserving $\lambda\%$ variance instead of fixed count. Parameters γ and λ jointly control sparsity.

4 Experimentation and Results

We evaluate `LayerMerge` on multiple architectures and modalities to demonstrate its generalizability.

To evaluate how depth scaling affects key emergent abilities, we utilize the `Open LLM Leaderboard v2` (implemented in Language Model Evaluation Harness [9], MIT license), which evaluates reasoning across *mathematics and logic*, *multistep soft reasoning*, *natural language understanding*, and *graduate level domain knowledge*.

Accuracy / Sparsity / Method	0.5 k=16	0.53 k=15	0.5625 k=14	0.625 k=12	0.7188 k=9	0.7813 k=7	0.8438 k=5	0.9062 k=3
Wanda	0.3126	0.3098	0.3101	**0.3091**	**0.3173**	**0.3121**	0.2751	0.3088
SparseGPT	**0.3195**	**0.3205**	**0.3155**	0.3036	0.2974	0.2907	**0.3019**	**0.3101**
ShortGPT	0.2920	0.3029	**0.3118**	**0.3088**	**0.3064**	0.2922	0.2949	0.2999
Shortened Llama	**0.2967**	0.2994	0.2915	0.3014	0.2793	0.2915	0.3105	0.2999
LaCo	0.2900	0.3061	0.3108	0.2997	0.3004	0.2850	0.2888	0.2798
LayerCluster	0.2793	0.2907	0.2689	0.2843	0.2870	0.2728	0.2822	0.2711
LayerMerge	0.2768	0.2793	0.2810	0.2885	0.2987	**0.3073**	**0.3113**	0.2967
LayerMergeAct	0.2897	**0.3091**	0.2987	0.3014	0.2999	0.2924	0.2987	**0.3062**

Table 1: Performance of pruning methods on `Leaderboard-Lite` at different sparsity levels using `llama-2-7b-hf` (accuracy 0.3152). "k" denotes decoder layers in depth-pruned models. For each sparsity, **bold** marks best depth pruning method, underlined marks second-best depth pruning, and **bold** marks best width pruning method.

For rapid evaluation, we curate `Leaderboard-Lite`, a subset of 15 `Open LLM Leaderboard v2` challenges (Table 5) with highest performance variance when running `LayerMergeAct` under 4 configurations yielding 50% sparsity (half original depth) on `llama-2-7b-hf`.

We compare our simple pruning methods with state-of-the-art width pruning (`Wanda` [2], `SparseGPT` [3]) and depth pruning methods (`ShortGPT` [4], `Shortened Llama` [5], `LaCo` [6]) using `llama-2-7b-hf` and `llama-3.1-8b` on a single NVIDIA Tesla V100-SXM2-32GB GPU.

Accuracy / Sparsity / Method	0.5 k=16	0.53 k=15	0.5625 k=14	0.625 k=12	0.7188 k=9	0.7813 k=7	0.8438 k=5	0.9062 k=3
Wanda	0.3158	0.3168	**0.3203**	**0.3185**	0.3029	**0.3021**	**0.3143**	0.2843
SparseGPT	**0.3277**	**0.3230**	0.3061	0.3106	**0.3051**	0.2964	0.2987	**0.3131**
ShortGPT	0.2838	0.2992	0.2982	**0.3091**	0.2934	**0.3049**	0.2894	**0.3004**
Shortened Llama	0.2927	0.2937	0.2885	0.2758	0.2920	0.2833	0.2919	**0.3004**
LaCo	0.3041	0.2982	**0.3071**	0.3021	0.2969	0.2967	0.2885	0.2862
LayerCluster	0.2880	**0.2994**	0.2867	0.2880	0.2751	0.2860	0.2537	0.2602
LayerMerge	0.2835	0.2820	0.2817	0.2830	0.2872	0.2865	0.2897	0.2984
LayerMergeAct	**0.3071**	0.2574	0.2897	0.2731	**0.3054**	0.2922	**0.2974**	0.2825

Table 2: Performance of pruning methods on `Leaderboard-Lite` at different sparsity levels using `llama-3.1-8b` (accuracy 0.3850). "k" denotes decoder layers in depth-pruned models. For each sparsity, **bold** marks best depth pruning method, underlined marks second-best depth pruning, and **bold** marks best width pruning method.

	FLOPs*	Memory
LayerMerge	18.47 TFLOPs	3.92 GB
SoTA	62.94 TFLOPs	12.68 GB

Table 3: Computational efficiency comparison between `LayerMerge` and state-of-the-art width pruning methods (Wanda, SparseGPT) on `llama-2-7b` at 50% sparsity. *Computed using a single C4 sample.

As shown in Tables 1, 2 and 3, our simpler pruning methods achieve comparable or superior `Leaderboard-Lite` performance to baselines while providing significant computational advantages, reducing FLOPs by 70% and memory usage by 69% compared to width pruning methods. For `llama-2-7b-hf`, our methods rank first among depth pruning methods at the three highest sparsities (k = 3, 5, 7) and achieve performance comparable to the base model. However, depth pruning methods generally lag slightly behind width pruning methods for both base models.

We examine how emergent abilities are affected when aggressively scaling model depth to 3 layers. Pruning strategies affect emergent abilities differently (Figure 3): While both width pruning methods outperform depth pruning methods and the base model on *multistep soft reasoning*, `Wanda` excels in *graduate level domain knowledge* and `SparseGPT` leads in *mathematics and logic*. For depth pruning, layer selection methods (`ShortGPT`, `Shortened Llama`) slightly outperform the base model in *mathematics and logic*.

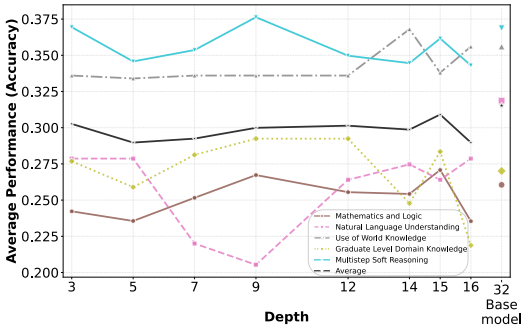

Figure 2: Emergent abilities of `LayerMergeAct`-pruned models with different depth (using `llama-2-7b-hf`).

Our `LayerMergeAct`, achieving best overall `Leaderboard-Lite` performance among depth pruning methods (Table 1), most successfully retains *graduate level domain knowledge* and *multistep soft reasoning* abilities. While all methods show noticeable drops in *natural language understanding*, with largest loss in 'disambiguation qa', `LayerMergeAct` best retains this ability.

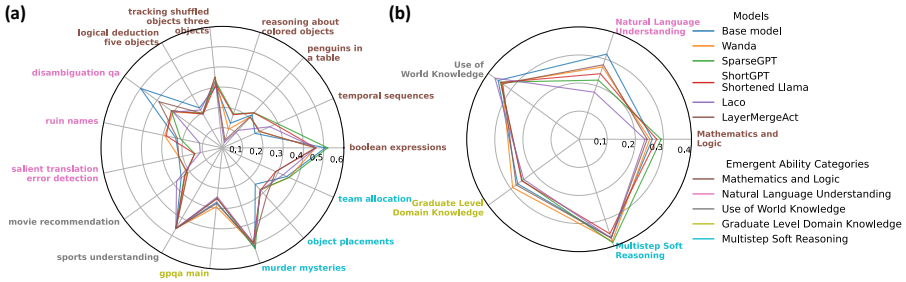

Figure 3: Performance of pruning methods on `Leaderboard-Lite` (`llama-2-7b-hf`) at highest sparsity (k=3). **(a)** shows task-wise performance while **(b)** aggregates by emergent ability category. `ShortGPT` and `Shortened Llama` produced identical models.

We find no significant trend between depth and emergent ability retention (Figure 2). For `LayerMergeAct`, 3 layers achieves superior or comparable performance to deeper configurations, suggesting aggressive depth scaling can meet both resource and performance requirements.

To validate modality-agnostic efficacy, we evaluate `LayerMerge` for time series forecasting using `MOMENT` [1] across four ETT datasets and four prediction horizons {96, 192, 336, 720 time steps}.

Table 4 shows averaged MAE performance across all horizons. `LayerMerge` achieves competitive results, with pruned 3-layer models best on ETTh2 and pruned 12-layer models best on ETTm2. Pruned models consistently outperform shallow baselines trained from scratch, demonstrating effective preservation of learned representations. Results show minimal performance differences (typically <0.01 MAE) between pruned and original models, confirming `LayerMerge` achieves substantial parameter reduction with negligible accuracy loss across modalities.

5 Discussion

In this paper, we presented three simple and direct pruning methods that reduce LLM complexity for deployment. We conducted a unified, systematic evaluation of state-of-the-art pruning methods. Our evaluation showed that our methods, while aggressively shrinking model size, can effectively retain key emergent abilities that empower LLMs, suggesting a practical pathway for managing LLM deployment and their evolving lifecycle. Our validation on `MOMENT` for time series forecasting further confirms the modality-agnostic efficacy of LayerMerge, with pruned models outperforming shallow baselines while achieving minimal performance loss. LayerMerge also provides substantial computational advantages, reducing FLOPs by 70% and memory usage by 69% compared to state-of-the-art width pruning methods, enabling efficient deployment in resource-constrained environments.

Dataset	Average MAE				
	Shallow (3)	Pruned (3)	Shallow (12)	Pruned (12)	Original (24)
ETTh1	0.446	0.439	0.443	0.437	**0.436**
ETTh2	0.402	**0.389**	0.406	0.394	0.395
ETTm1	0.380	0.382	0.385	0.382	**0.379**
ETTm2	0.321	0.317	0.324	**0.316**	0.318

Table 4: Average MAE performance across all prediction horizons. Shallow (k) models are k-layer `MOMENT` models trained from scratch; Pruned (k) models are original `MOMENT` models pruned to k layers using `LayerMerge`. Complete results in Appendix C.

Although our methods achieve comparable or superior performance to baselines, all methods suffer from larger gaps when using `llama-3.1-8b` (Table 2), while pruned model performance under the same sparsity does not improve. This suggests `llama-3.1-8b` may be less "prunable" than `llama-2-7b-hf`. As more advanced LLMs are developed, they may paradoxically become harder to scale down for deployment. Furthermore, while all evaluated methods lack trends between performance and depth retention beyond 50% pruning (Tables 1 and 2, Figure 2), this phenomenon may reflect that task-agnostic pruning methods may not retain specific emerging capabilities.

In future work, we will develop targeted, task-specific methods for scaling down LLMs and expect greater compression with lower performance loss. We will also develop more systematic benchmarks evaluating emergent abilities at finer resolutions to carefully guide such method development.

Acknowledgements

This work has been partially supported by the National Science Foundation (awards 2427948, 2406231 and 2530752) and Defense Advanced Research Projects Agency (award HR00112420329).

References

[1] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: a family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[2] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[3] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR, 2023.

[4] 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. *arXiv preprint arXiv:2403.03853*, 2024.

[5] Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11:1, 2024.

[6] Yifei Yang, Zouying Cao, and Hai Zhao. Laco: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417, 2024.

[7] Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, Chenhan Yu, Wei-Chun Chen, Hayley Ross, Oluwatobi Olabiyi, Ashwath Aithal, Oleksii Kuchaiev, Daniel Korzekwa, Pavlo Molchanov, Mostofa Patwary, Mohammad Shoeybi, Jan Kautz, and Bryan Catanzaro. Llm pruning and distillation in practice: The minitron approach, 2024.

[8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[9] 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.

A Algorithm for finding diagonal blocks

Algorithm 1 Recursive Block Finder

Input:
Raw similarity matrix $S \in \mathbb{R}^{n \times n}$
Binarized matrix M where $M_{ij} = 1$ if $S_{ij} \geq \gamma$ and 0 otherwise, for a threshold γ.
Output:
A list of non-overlapping block indices $\mathcal{I} = [(s_1, e_1), (s_2, e_2), \dots]$

procedure FINDBLOCKS(S, M)
  $n \leftarrow$ number of rows in M
  if $n = 0$ **then return** []
  end if
  if $n = 1$ **then return** $[(0, 0)]$
  end if
  $B \leftarrow []$ ▷ List to store potential blocks and their scores
  for $s \leftarrow 0$ to $n - 1$ **do**
    if $M[s, s] = 1$ **then**
      $e \leftarrow s + 1$
      while $e < n$ and submatrix $M[s : e, s : e]$ is all ones **do**
        $e \leftarrow e + 1$
      end while
      $s_{\min} \leftarrow \min(S[s : e - 1, s : e - 1])$
      Append $((s, e - 1), s_{\min})$ to B
    end if
  end for
  Sort B descending by block size $(e - s)$, then by $s_{\min}$
  $(s^*, e^*) \leftarrow$ indices of the first block in sorted list B
  $M_{\text{before}} \leftarrow M[0 : s^* - 1, 0 : s^* - 1]$
  $M_{\text{after}} \leftarrow M[e^* + 1 : n - 1, e^* + 1 : n - 1]$
  $\mathcal{I}_{\text{before}} \leftarrow$ FindBlocks(S, M_{before})
  $\mathcal{I}_{\text{after}} \leftarrow$ FindBlocks(S, M_{after})
  Shift all indices (i_s, i_e) in $\mathcal{I}_{\text{after}}$ by $e^* + 1$
  return $\mathcal{I}_{\text{before}} \cup [(s^*, e^*)] \cup \mathcal{I}_{\text{after}}$
end procedure

B Challenges in `Leaderboard-Lite`

C Complete Results for `LayerMerge` Applied to `MOMENT`

Domain	Challenge

Mathematics and Logic	`leaderboard_bbh_boolean_expressions`
	`leaderboard_bbh_temporal_sequences`
	`leaderboard_bbh_penguins_in_a_table`
	`leaderboard_bbh_reasoning_about_colored_objects`
	`leaderboard_bbh_tracking_shuffled_objects_three_objects`
	`leaderboard_bbh_logical_deduction_five_objects`

Natural Language Understanding	`leaderboard_bbh_disambiguation_qa`
	`leaderboard_bbh_ruin_names`
	`leaderboard_bbh_salient_translation_error_detection`

Use of World Knowledge	`leaderboard_bbh_movie_recommendation` `leaderboard_bbh_sports_understanding`
Graduate Level Domain Knowledge	`leaderboard_gpqa_main`

Multistep Soft Reasoning	`leaderboard_musr_murder_mysteries`
	`leaderboard_musr_object_placements`
	`leaderboard_musr_team_allocation`

Table 5: Challenges in `Leaderboard-Lite` grouped by the measured emergent ability of the LLM.

Dataset	**Horizon**	**MAE**				
		Shallow (3)	**Pruned (3)**	**Shallow (12)**	**Pruned (12)**	**Original**
ETTh1	96	0.409	**0.400**	0.412	0.401	0.410
	192	0.430	0.426	0.434	**0.424**	0.426
	336	0.462	0.446	0.466	0.442	**0.437**
	720	0.483	0.485	**0.461**	0.480	0.472
ETTh2	96	0.351	**0.338**	0.350	0.341	0.345
	192	0.392	**0.380**	0.390	0.384	0.386
	336	0.414	**0.405**	0.422	0.409	0.408
	720	0.452	**0.434**	0.463	0.441	0.439
ETTm1	96	0.350	0.354	0.351	0.352	**0.349**
	192	0.369	0.370	0.372	0.370	**0.368**
	336	0.386	0.387	0.392	0.388	**0.384**
	720	0.416	0.416	0.424	0.417	**0.416**
ETTm2	96	0.262	0.260	0.269	**0.258**	0.260
	192	0.299	0.297	0.306	**0.295**	0.297
	336	0.334	0.329	0.334	**0.328**	0.328
	720	0.390	**0.382**	0.388	0.383	0.387

Table 6: MAE performance comparison across different datasets and prediction horizons