DLO: DYNAMIC LAYER OPERATION FOR EFFICIENT VERTICAL SCALING OF LLMS

Zhen Tan*& Daize Dong*
School of Computing, and Augmented Intelligence, Arizona State University
Shanghai Artificial Intelligence Laboratory
{ztan36}@asu.edu, {dongdaize.d}@pjlab.org.cn

Xinyu Zhao, Jianing Cai & Jie Peng
Department of Computer Science, University of North Carolina at Chapel Hill
Department of Computer and Information Science, University of Pennsylvania
School of Artificial Intelligence and Data Science, University of Science and Technology of China
{xinyu}@cs.unc.edu, {cjianing}@seas.upenn.edu, {pengjieb}@mail.ustc.edu.cn

Yu Cheng & Tianlong Chen
Department of Computer Science, Chinese University of Hong Kong
Department of Computer Science, University of North Carolina at Chapel Hill
{chengyu}@cse.cuhk.edu.hk, {tianlong}@cs.unc.edu

ABSTRACT

In this paper, we introduce Dynamic Layer Operations (DLO), a novel approach for vertically scaling transformer-based Large Language Models (LLMs) by dynamically expanding, activating, or skipping layers using a sophisticated routing policy based on layerwise feature similarity. Unlike traditional Mixture-of-Experts (MoE) methods that focus on extending the model width, our approach targets model depth, addressing the redundancy observed across layer representations for various input samples. Our framework is integrated with the Supervised Fine-Tuning (SFT) stage, eliminating the need for resource-intensive Continual Pre-Training (CPT). Experimental results demonstrate that DLO not only outperforms the original unscaled models but also achieves comparable results to densely expanded models with significantly improved efficiency. Our work offers a promising direction for building efficient yet powerful LLMs. Our code is available at https://github.com/DaizeDong/LLaMA-DLO.git.

1 INTRODUCTION

Large Language Models (LLMs) (Achiam et al., 2023; Team et al., 2023) excel in NLP tasks by capturing complex patterns in data. Traditional scaling focuses on *horizontal* expansion, as in Mixture-of-Experts (MoE) architectures (Shazeer et al., 2017; Fedus et al., 2022b; Lepikhin et al., 2020), which optimize parameter usage by activating subsets of parameters (Fedus et al., 2022a). However, *vertical* expansion remains underexplored.

Inspired by brains' selective neural activation for complex tasks (Baddeley, 1992; Koechlin et al., 2003), we propose DLO, a method for dynamic vertical scaling that expands, activates, or skips layers based on feature similarity. Vertically scaling LLMs faces three challenges: ❶ **Optimization Complexity.** Selecting optimal layer configurations is NP-hard (Glorot & Bengio, 2010; Hestness et al., 2017), with existing approximations showing limited improvement (Wang et al., 2023a). ❷ **Computation Cost.** Deeper networks increase latency and resource usage. ❸ **Feature Collapse.** Figure 1 (b) shows significant similarity in consecutive layers, suggesting redundancy.

To address these, we propose **D**ynamic **L**ayer **O**peration (**DLO**), consisting of (***i***) `expansion`, (***ii***) `activation`, and (***iii***) `skipping` operations for vertical scaling without proportional com-

*Equal contribution.

putational costs. Key designs include: ❶ **`Expansion`**: Dynamically adds layers to simplify optimization. ❷ **`Activation & Skip`**: Uses similarity-induced labels to activate or skip layers. ❸ **Adaptive FLOPs**: Varies sparsity settings for tokens, improving efficiency. ❹ **Enhanced Generalizability**: Adjusts layer-specific learning rates based on sparsity. All modules are trained during the Supervised Fine-Tuning (SFT) stage, avoiding the costly Continual Pre-training (CPT).

2 Related Work

Mixture-of-Experts (MoE). MoE architectures enhance LLM efficiency and scalability by activating only a subset of parameters per input, significantly reducing computational overhead compared to traditional networks that activate all parameters (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022b; Zoph et al., 2022; Zhu et al., 2024; Jiang et al., 2024). This selective activation enables scaling to billions of parameters without proportional computational costs. While MoE optimizes width through horizontal expansion, it often overlooks layer redundancy. Our Dynamic Layer Operation (DLO) complements MoE by focusing on vertical scaling, dynamically expanding and activating layers to enhance depth scalability and reduce redundancy.

Efficient Model Stacking. Model stacking is a common ensemble technique that improves performance by combining models to leverage complementary strengths (TING, 1997; Chen et al., 2015). In LLMs, stacking integrates models hierarchically, where outputs from one model serve as inputs to another, capturing diverse features (Dabre & Fujita, 2019; Chen et al., 2021a; Wang et al., 2023b; Kim et al., 2023).

Recent advancements stack pre-trained transformer layers to create composite models, reducing training costs but increasing inference latency (Gong et al., 2019; Gu et al., 2020; Evci et al., 2022; Yao et al., 2023; Du et al., 2024; Wu et al., 2024). To address this, layer-skipping methods enable early exits via additional classifiers, minimizing processed layers (Wang et al., 2022; Chen et al., 2023; Zhang et al., 2024). Conditional computation techniques further enhance efficiency by dynamically skipping layers based on token-specific conditions (Ainslie et al., 2023; Raposo et al., 2024), but often require pre-training modifications, adding complexity.

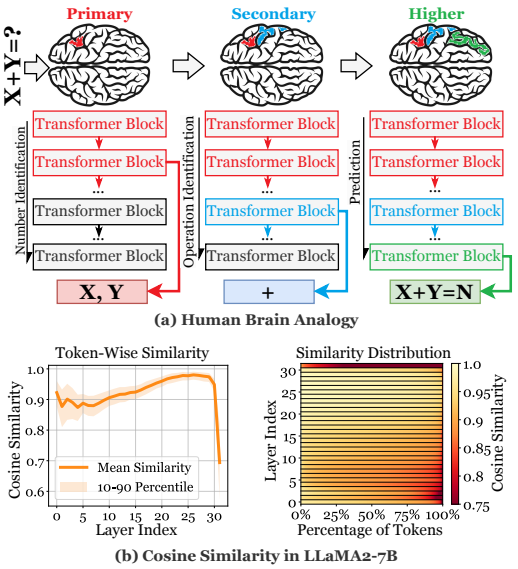

Figure 1: (a) DLO structure inspired by human brain activities in a math problem (Koechlin et al., 2003), where primary neurons perceive numbers, secondary neurons understand operations, and higher-order neurons calculate results. (b) Layer-wise token similarity and distribution.

In contrast, our DLO method ensures efficiency and scalability through dynamic vertical scaling during the SFT stage, offering a high-performance solution without the computational demands of stacked ensembles.

3 Methodology

We introduce the **D**ynamic **L**ayer **O**peration (**DLO**) framework, designed to optimize the vertical scaling of large language models (LLMs) by dynamically adjusting their depth during the supervised fine-tuning (SFT) phase. DLO comprises three core operations: `layer expansion`, `layer activation`, and `dynamic skipping`. Algorithm 1 provides a high-level overview, while details are provided in Appendix A.

3.1 Layer Expansion

DLO divides the R transformer layers into P groups of Q layers each and dynamically expands each group by q layers. This results in a total of $R' = P \times (Q + q)$ layers. The new layers are initialized

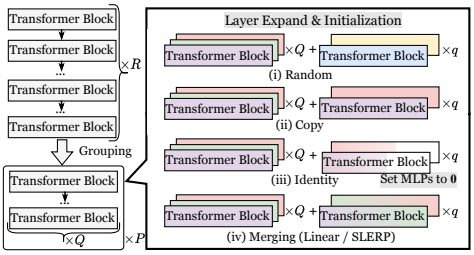

Figure 2: Layer extension strategies.

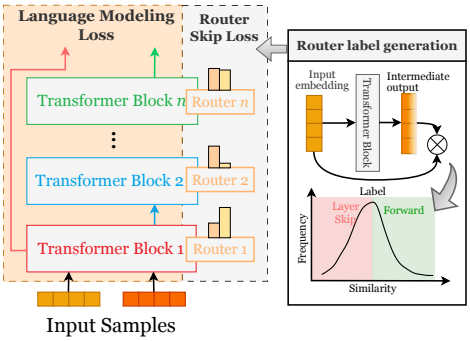

Figure 3: Training pipeline of DLO

using strategies such as Xavier initialization, parameter copying, or spherical linear interpolation (SLERP). See Appendix A.1 for mathematical details and initialization strategies.

3.2 Layer Activation and Skipping

During training and inference, DLO selectively activates or skips the MLP module within each transformer layer based on a decision score computed by a linear router. The router dynamically evaluates token embeddings to determine layer activation. Key steps include computing the decision score, token-specific activation, and the dynamic update of router parameters. Mathematical formulations for score calculation and dynamic activation are provided in Appendix A.2.

3.3 Training and Integration

DLO optimizes both task-specific and router-related objectives through an integrated training process. The skip loss $\mathcal{L}_{\text{skip}}$, based on binary cross-entropy (BCE), penalizes incorrect router predictions. The overall loss is defined as $\mathcal{L} = \mathcal{L}_{\text{task}} + \epsilon\mathcal{L}_{\text{skip}}$, where ϵ controls the influence of the skip loss. We employ a skip rate annealing strategy and layer-wise learning rates to progressively increase sparsity and enhance training stability (see Appendix A.3).

3.4 Adaptive Inference-Time FLOPs

At inference time, DLO applies token-specific sparsity settings to optimize floating-point operations (FLOPs). For each layer $\mathcal{L}_i$, the FLOPs are computed as: $\text{FLOPs}_i = \hat{\rho}_i \cdot \text{FLOPs}_{\text{full}}$, where $\hat{\rho}_i$ is the sparsity predicted by the router, and $\text{FLOPs}_{\text{full}}$ is the FLOPs for a fully active layer. See Appendix A.4 for more details.

4 Experiments

4.1 Experimental Settings

Model Selection. We use LLaMA2-7B (Touvron et al., 2023) as the primary backbone for its open-source availability and extensive usage. It comprises $R = 32$ transformer layers grouped into $P = 4$ clusters of $Q = 8$ layers (Wu et al., 2024). For expansion, we increase the group size to $Q' = 10$, creating LLaMA-DLO, a dense model with 40 layers and 8 billion parameters. We also compare against LLaMA-Pro-8B (Wu et al., 2024), trained with Continual Pre-Training (CPT). Section 4.3 shows DLO balances performance and computation cost effectively. While DLO is model-agnostic, experiments on larger models are left as future work due to academic lab constraints.

Fine-tuning Details. We fine-tune using five instruction tuning datasets: ShareGPT (Team, 2023a), EvolInstruct (Luo et al., 2023), SlimOrca (Team, 2023b), MetaMath (Yu et al., 2023), and Evol-CodeAlpaca (Team, 2022), with ShareGPT tripled, totaling 1.44 million instances (Wu et al., 2024). Training uses a batch size of 128, sequence length of 4,096 tokens, and a learning rate of $2e-5$ with cosine scheduling and AdamW (Loshchilov & Hutter, 2017). Flash Attention (Dao et al., 2022)

	Model	FLOPs ↓	Language ↑							Math ↑		Code ↑		Avg. ↑
			ARC-C	GLUE	MMLU	OBQA	PIQA	TruthfulQA	WinoGrande	GSM8K	MathQA	HumanEval	MBPP	
0%	•LLaMA2-7B	29.3T	53.1	40.6	46.9	44.2	79.0	38.8	74.0	14.5	28.3	21.8	29.0	42.75
	•LLaMA2-7B$_{+SFT}$		54.0	72.4	53.0	44.4	78.8	**40.8**	**74.2**	56.6	30.8	57.3	30.5	**52.89**
	•LLaMA-Pro-8B	36.4T	**54.1**	40.7	47.9	41.6	78.2	39.0	74.0	17.9	29.5	28.7	**33.2**	44.07
	•LLaMA-Pro-8B$_{+SFT}$		51.0	71.0	53.0	**45.0**	79.0	38.0	73.6	**58.6**	30.8	**58.4**	30.5	53.53
	•LLaMA-DLO-8B$_{+SFT}$	36.5T	53.2	**75.5**	**53.2**	43.7	**79.0**	38.7	74.0	57.4	**31.0**	57.0	30.2	**53.90**
5%	○ LLaMA2-7B$_{+SFT}$	28.4T	**51.0**	**74.4**	50.4	41.6	77.4	38.8	**72.6**	48.9	30.0	50.1	28.7	51.26
	○ LLaMA-DLO-8B$_{+SFT}$	35.3T	50.8	73.2	**51.6**	**43.0**	**77.8**	**39.0**	70.2	**53.4**	**30.5**	**56.7**	**28.9**	**52.28**
10%	○ LLaMA2-7B$_{+SFT}$	27.5T	51.0	72.5	51.1	40.2	78.0	39.3	71.0	53.4	29.6	49.7	28.3	51.28
	☆ LLaMA-DLO-8B$_{+SFT}$	34.2T	**52.5**	**75.4**	**51.4**	**43.6**	**78.7**	**41.0**	**73.4**	**55.0**	**31.4**	**55.3**	**29.1**	**53.34**
15%	○ LLaMA2-7B$_{+SFT}$	26.7T	44.2	71.8	50.6	38.6	75.4	36.9	55.3	17.0	26.2	3.9	0.0	38.17
	○ LLaMA-DLO-8B$_{+SFT}$	33.1T	**48.7**	**74.0**	**51.2**	**43.0**	**77.4**	**39.0**	**72.8**	**46.8**	**28.1**	**56.6**	**28.7**	**51.48**
20%	○ LLaMA2-7B$_{+SFT}$	25.8T	33.0	69.9	50.4	35.0	65.6	36.9	54.2	1.0	24.5	0.0	0.0	33.68
	○ LLaMA-DLO-8B$_{+SFT}$	32.0T	**51.2**	**73.3**	**50.8**	**43.2**	**78.2**	**38.9**	**73.2**	**50.1**	**30.5**	**57.6**	**28.2**	**52.29**
25%	○ LLaMA2-7B$_{+SFT}$	25.0T	29.5	67.4	47.1	37.8	57.6	35.4	58.3	0.0	22.6	0.0	0.0	32.34
	○ LLaMA-DLO-8B$_{+SFT}$	31.9T	**46.8**	**73.1**	**51.6**	**42.6**	**77.6**	**39.6**	**70.9**	**42.8**	**30.4**	**50.9**	**26.7**	**50.27**
30%	○ LLaMA2-7B$_{+SFT}$	24.1T	28.1	2.8	47.1	35.0	53.8	37.9	52.2	0.0	21.6	0.0	0.0	25.32
	○ LLaMA-DLO-8B$_{+SFT}$	29.8T	**44.6**	**73.0**	**50.1**	**41.2**	**77.1**	**37.1**	**63.2**	**46.9**	**28.1**	**31.2**	**6.0**	**45.32**

Table 1: Performance comparison of DLO (our approach) on various datasets using LLaMA2-7B as the backbone. Models marked with • are dense models, either original or those expanded using DLO `expansion`. Models with 8B parameters indicate expansion via LLaMA-Pro or our DLO. Models marked with ○ are sparse models incorporating DLO `activation` and `skipping` operations. Inference FLOPs are counted with a sequence with 2,048 tokens. The proposed • LLaMA-DLO-8B with 0% spasity signifies that no layer is skipped and all the original and expanded layers are activated. ○ LLaMA2-7B with non-zero sparsity equals LLaMA-DLO without expanding layers. ☆ indicates the sparsity leads to the best performance with DLO.

and bfloat16 mixed-precision training are employed to accelerate training. Fine-tuning LLaMA-DLO under varying skip ratios produces sparse models, with each run taking 36 hours on eight NVIDIA A100 GPUs. Additional hyperparameter details are in Appendix C.

Evaluation Benchmarks. We evaluate models using EleutherAI LM Harness (Gao et al., 2023) and BigCode Harness (Ben Allal et al., 2022) across three domains: ❶ *Language* [ARC-C (Clark et al., 2018), GLUE (Wang et al., 2018), MMLU (Hendrycks et al., 2020), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), TruthfulQA (Lin et al., 2021), WinoGrande (Sakaguchi et al., 2021)], ❷ *Math* [GSM8K (Cobbe et al., 2021), MathQA (Amini et al., 2019)], and ❸ *Code* [HumanEval (Chen et al., 2021b), MBPP (Austin et al., 2021)]. Accuracy is the primary metric, with additional details in Appendix D.

4.2 Overall Performance

Table 1 summarizes DLO's performance across various datasets using LLaMA2-7B as the backbone. Key observations include:

❶ **Dense Models' Superiority:** Dense models (•) generally outperform sparse models (○), particularly on harder tasks like *Math* and *Code*. For example, dense LLaMA-DLO-8B+$_{SFT}$ achieves an average score of 53.90, compared to the sparse model's 53.34.

❷ **Efficiency of Sparse Models:** Sparse models (○) significantly reduce inference FLOPs while maintaining competitive accuracy. At 10% sparsity, sparse LLaMA-DLO-8B+$_{SFT}$ achieves 53.34 with 34.2T FLOPs, compared to the dense counterpart's 53.90 with 36.5T FLOPs. However, higher sparsities (*e.g.*, $\rho = 30\%$) lead to performance degradation.

❸ **DLO-`Expansion` Advantages:** Models with DLO `expansion` outperform the original LLaMA2-7B and SOTA methods. For instance, dense LLaMA-DLO-8B+$_{SFT}$ achieves a higher score (53.90) than LLaMA2-7B+$_{SFT}$ (52.89) and LLaMA-Pro-8B+$_{SFT}$ (53.53).

❹ **Balanced Performance of Sparse Models:** Sparse models offer an excellent balance between performance and efficiency. At 30% sparsity, LLaMA-DLO maintains strong performance on GLUE (73.0 vs. 2.8) and HumanEval (31.2 vs. 0.0) while significantly reducing FLOPs.

❺ **General Observations:** DLO effectively balances performance and efficiency, leveraging `expansion` and `skipping` to enhance scalability and reduce computational costs, making it a robust solution for LLM deployment.

Further ablation analysis of DLO's components, including experiments on initialization strategies, router score rescaling, sparsity allocation, and performance comparisons with baseline methods, is provided in Appendix B.

Method	ARC-C	MMLU	TruthfulQA	WinoGrande	GSM8K	Avg. ↑
SOLAR	24.8	24.8	38.8	50.7	2.2	28.3
SD-Stack	23.5	23.4	36.0	51.1	2.5	27.3
DLO	**52.5**	**51.4**	**41.0**	**73.4**	**55.0**	**54.7**

Table 2: Comparison with different expansion methods. We extend ○ LLaMA-DLO layers using different strategies and fine-tune the expanded models with DLO under overall skip rate $\rho = 10\%$.

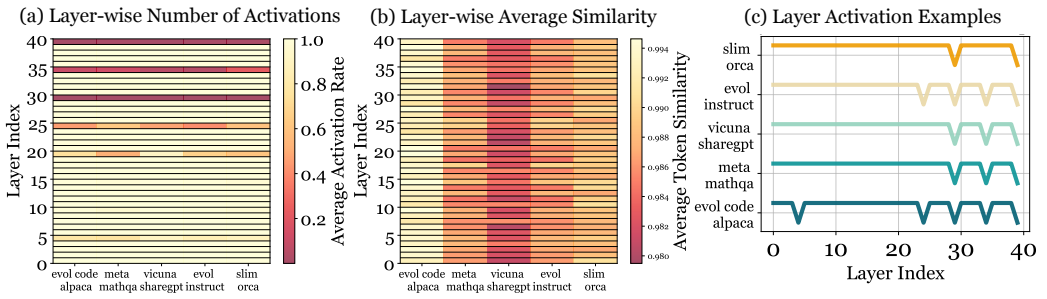

Figure 4: Visualization on different datasets of (a) Layer-Wise Number of Activations, (b) Layer-Wise Average Similarity, and (c) Token Activation Examples.

Efficient Expansion. In addition to the high-cost LLaMA-Pro approach (studied in Table 1), we compare our expansion method with two state-of-the-art efficient vertical expansion baselines: SOLAR Kim et al. (2023) and Self-Duplicate Stack (SD-Stack) Team (2024). These two methods duplicate blocks of transformer layers and stack them together in a training-free manner.

As shown in Table 2, the proposed DLO-`expansion` significantly outperforms both SOLAR and SD-Stack by a considerable margin. This highlights the critical role of SFT in adapting the expanded layers effectively. Unlike training-free approaches, DLO-`expansion` achieves a superior balance between training cost and performance, demonstrating the importance of fine-tuning in maximizing the effectiveness of layer expansion.

▷ *Layer-Wise Skip Rates.* Adjusting skip rates for each layer aims to selectively activate or skip layers based on their contribution to task performance, which is measured by layer-wise similarity. This method helps focus computational resources on more critical layers. Results in Table 5 suggest that this approach can lead to more efficient sparsity allocation with less impact on model performance. Figure 4 also shows that DLO can skip layers that have high layer-wise similarity.

▷ *Layer-Wise Skip Rates.* Adjusting skip rates based on layer-wise similarity selectively activates critical layers, optimizing computational resources. Table 5 shows this approach improves sparsity allocation with minimal impact on performance. Figure 4 illustrates that DLO effectively skips layers with high similarity.

▷ *Sparsity Allocation.* Tailoring skip rates by layer helps distribute sparsity more effectively, potentially reducing computational overhead. As indicated in Table 5, models with layer-wise skip rates tend to maintain performance while achieving better computational efficiency.

4.3 Scalability

The proposed LLaMA-DLO model surpasses the performance of the original dense LLaMA while also achieving comparable results to the dense LLaMA-Pro. Notably, it does so at a significantly lower training cost without the need for expensive CPT. Additionally, LLaMA-DLO facilitates efficient inference through adaptively reduced FLOPs, making it a cost-effective choice for both training and deployment. Figure 5 illustrates the trade-off between model performance and both training and inference costs. LLaMA-DLO emerges as the best solution, achieving a favored balance across these metrics. This demonstrates the model's scalability, ensuring that high performance is maintained while keeping computational costs relatively manageable.

5 CONCLUSION

This paper presents LLaMA-DLO, a framework for efficient vertical scaling of LLMs that dynamically expands, activates, and skips layers to optimize computational resources. Our experiments demonstrate that LLaMA-DLO achieves performance on par with expensive dense expansion model like LLaMA-Pro, while significantly reducing training costs and enhancing inference efficiency. These results highlight LLaMA-DLO's potential as a cost-effective solution for scaling LLMs in various NLP tasks, offering a balanced approach between model performance and resource management.

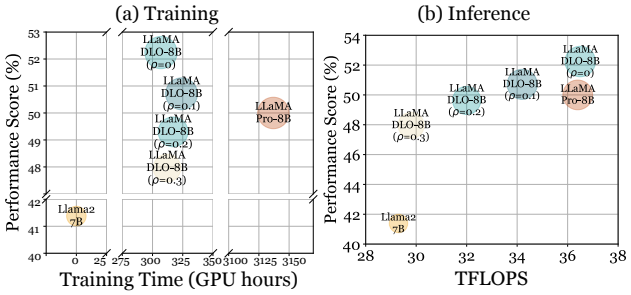

Figure 5: (a) Performance (↑) *v.s.* Training time (↓). LLaMA-Pro is reported in H800 GPU hours quoted from the original paper. The rest are reported in A100 GPU hours. (b) Performance (↑) *v.s.* Inference FLOPs (↓). DLO achieves the best trade-off between performance and training / inference costs.

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.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. Colt5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*, 2023.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Alan Baddeley. Working memory. *Science*, 255(5044):556–559, 1992.

Daniel Mesafint Belete and Manjaiah D Huchaiah. Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results. *International Journal of Computers and Applications*, 44(9):875–886, 2022.

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. A framework for the evaluation of code generation models. `https://github.com/bigcode-project/bigcode-evaluation-harness`, 2022.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, number 05, pp. 7432–7439, 2020.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2bert: Towards reusable pretrained language models. *arXiv preprint arXiv:2110.07143*, 2021a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. *arXiv preprint arXiv:2312.04916*, 2023.

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. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Raj Dabre and Atsushi Fujita. Recurrent stacking of layers for compact neural machine translation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 01, pp. 6292–6299, 2019.

T Dao, DY Fu, S Ermon, A Rudra, and C Flashattention Ré. Fast and memory-efficient exact attention with io-awareness. *URL https://arxiv. org/abs/2205.14135*, 2022.

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. *arXiv preprint arXiv:2405.15319*, 2024.

Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Max Vladymyrov, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information. *arXiv preprint arXiv:2201.05125*, 2022.

William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022a.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022b.

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, d Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of bert by progressively stacking. In *International conference on machine learning*, pp. 2337–2346. PMLR, 2019.

Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. On the transformer growth for progressive bert training. *arXiv preprint arXiv:2010.12562*, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*, 2023.

Etienne Koechlin, Chrystele Ody, and Frédérique Kouneiher. The architecture of cognitive control in the human prefrontal cortex. *Science*, 302(5648):1181–1185, 2003.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

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

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

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. *arXiv preprint arXiv:2404.02258*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 245–254. ACM, 1985.

EvolCodeAlpaca Team. Evolcodealpaca dataset, Feb 2022. URL `https://www.kaggle.com/code/mpwolke/evol-codealpaca`.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Self-Duplicate Stack Team. Self-duplicate stack, May 2024. URL `https://huggingface.co/mlabonne/Meta-Llama-3-120B-Instruct`.

ShareGPT Team. Sharegpt dataset, Apr 2023a. URL `https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered/tree/main`.

SlimOrca Team. Slimorca dataset, Dec 2023b. URL `https://www.kaggle.com/datasets/thedevastator/open-orca-slimorca-gpt-4-completions`.

WK TING. Stacking bagged and dagged models. In *Proceedings of ICML'97*, pp. 367–375. Morgan Kaufmann, 1997.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Jue Wang, Ke Chen, Gang Chen, Lidan Shou, and Julian McAuley. Skipbert: Efficient inference with shallow layer skipping. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7287–7301, 2022.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. *arXiv preprint arXiv:2303.00980*, 2023a.

Yite Wang, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. Lemon: Lossless model expansion. *arXiv preprint arXiv:2310.07999*, 2023b.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*, 2024.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. Masked structural growth for 2x faster language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Xingjian Zhang, Jiaxi Tang, Yang Liu, Xinyang Yi, Li Wei, Lichan Hong, Qiaozhu Mei, and Ed H Chi. Conditional transformer fine-tuning by adaptive layer skipping. In *5th Workshop on practical ML for limited/low resource settings*, 2024.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. Llama-moe: Building mixture-of-experts from llama with continual pre-training, 2024. URL https://arxiv.org/abs/2406.16554.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A Methodology Details

This section provides a comprehensive explanation of the Dynamic Layer Operation (DLO) framework, including its key components: layer expansion, activation, skipping, training integration, and inference-time efficiency.

A.1 Layer Expansion

To enable dynamic depth adjustment, DLO divides R transformer layers into P groups of Q layers each, such that $R = P \times Q$. Each group is expanded by q additional layers, resulting in $R' = P \times (Q + q)$ layers after expansion. Let $\mathcal{G}_i$ denote the i-th group consisting of layers $\mathcal{L}_{i1}, \ldots, \mathcal{L}_{iQ}$. The expanded group $\mathcal{G}'_i$ includes layers $\mathcal{L}_{i(Q+1)}, \ldots, \mathcal{L}_{i(Q+q)}$.

The new layers are initialized using one of the following strategies:

- **Random Initialization (Π_{rand})**: Parameters θ'_{ij} are initialized using Xavier initialization (Glorot & Bengio, 2010):

$$\theta'_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{in} + n_{out}}}, \sqrt{\frac{6}{n_{in} + n_{out}}}\right), \tag{1}$$

  where n_{in} and n_{out} represent the input and output dimensions of the layer.

- **Copy Initialization (Π_{copy})**: Parameters are copied from the previous layer:

$$\theta'_{ij} = \theta_{i(Q+q-1)}. \tag{2}$$

- **Identity Initialization (Π_{identity})** (Wu et al., 2024): Parameters are copied from the preceding layer, with the output projection matrix of the multi-head self-attention (MHSA) set to zero.
- **Linear Merge (Π_{linear})**: The new layer's parameters are a weighted sum of multiple preceding layers:

$$\theta'_{ij} = \sum_{k=1}^{\tau} \alpha_k \theta_{i(Q+q-k)}, \quad \sum_{k=1}^{\tau} \alpha_k = 1. \tag{3}$$

- **Spherical Linear Interpolation (SLERP) (Π_{slerp})** (Shoemake, 1985): SLERP smoothly interpolates between two parameter vectors $\mathbf{u}$ and $\mathbf{v}$ on a unit sphere:

$$\text{SLERP}(\mathbf{u}, \mathbf{v}, \alpha) = \frac{\sin((1-\alpha)\Omega)}{\sin(\Omega)}\mathbf{u} + \frac{\sin(\alpha\Omega)}{\sin(\Omega)}\mathbf{v}, \tag{4}$$

  where $\Omega = \arccos\left(\frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right)$ is the angle between the vectors, and $\alpha \in [0, 1]$ is the interpolation parameter.

A.2 Layer Activation and Skipping

During training and inference, DLO selectively activates or skips the MLP module within each transformer layer based on a decision score computed by a linear router. Let $\mathbf{h}_i$ denote the token embeddings at layer $\mathcal{L}_i$. The router computes the decision score as:

$$r_i = \frac{\beta + (2\sigma(\mathbf{h}_i W_i) - 1)\gamma}{2}, \tag{5}$$

where σ is the sigmoid function, and β, γ are hyperparameters controlling the score range.

Activation Condition: During inference, layer $\mathcal{L}_i$ is activated if $r_i \geq \frac{\beta}{2}$. The output is computed as:

$$\mathbf{h}_{i+1} = \begin{cases} r_i \cdot \mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i), & \text{if } r_i \geq \frac{\beta}{2}, \\ \mathcal{A}_i(\mathbf{h}_i), & \text{otherwise.} \end{cases} \tag{6}$$

A.3 Training and Integration

DLO optimizes both task-specific and router-related objectives through a combined loss function. The skip loss $\mathcal{L}_{\text{skip}}$ is defined as:

$$\mathcal{L}_{\text{skip}} = \frac{1}{R'S} \sum_{i,s=1}^{R',S} \mathcal{L}_{\text{BCE}}(\sigma(\mathbf{h}_i^s W_i), \tilde{\lambda}_i^s), \tag{7}$$

where $\tilde{\lambda}_i^s$ are supervised labels generated based on cosine similarity between token embeddings:

$$\mu_i^s = \frac{\mathcal{A}_i(\mathbf{h}_i^s) \cdot \mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i^s)}{\|\mathcal{A}_i(\mathbf{h}_i^s)\|\|\mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i^s)\|}. \tag{8}$$

Tokens with the lowest similarity scores are labeled for activation:

$$\tilde{\lambda}_i^s = \begin{cases} 1, & \text{if } \mu_i^s \in \text{Bottom}_{\lfloor(1-\rho)R'S\rfloor}(\{\mu_i^s\}), \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The overall training objective is:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \epsilon\mathcal{L}_{\text{skip}}, \tag{10}$$

where ϵ controls the contribution of the skip loss.

Skip Rate Annealing: The sparsity factor ρ^t is annealed over T' steps:

$$\rho^t = \begin{cases} \bar{\rho} + (\rho - \bar{\rho})\frac{t}{T'}, & \text{if } t \leq T', \\ \rho, & \text{otherwise.} \end{cases} \tag{11}$$

Layer-Wise Learning Rates: To enhance generalization, DLO employs layer-wise learning rates:

$$\zeta_{i,t} = \bar{\zeta} \cdot \frac{1 - \rho_{i,t}}{1 - \rho^t}. \tag{12}$$

A.4 Adaptive Inference-Time FLOPs

During inference, DLO applies token-specific sparsity to reduce floating-point operations (FLOPs). For each layer $\mathcal{L}_i$, the FLOPs are computed as:

$$\text{FLOPs}_i = \hat{\rho}_i \cdot \text{FLOPs}_{\text{full}}, \tag{13}$$

where $\hat{\rho}_i$ is the predicted sparsity and $\text{FLOPs}_{\text{full}}$ is the FLOPs for a fully active layer. This mechanism improves inference efficiency while maintaining performance.

B Ablation Studies

Initialization Strategies for Expanded Layers.

❶ The identity and copy initialization strategies demonstrate the most consistent and high-performing results, suggesting that leveraging existing layer information is beneficial for stabilizing and enhancing model performance. These methods help maintain coherence in the model's internal representations, leading to robust results across tasks, including GLUE and HumanEval.

❷ Interestingly, while linear and SLERP initializations were expected to offer smoother transitions and potentially enhance performance, their results were only moderately effective. This indicates that while sophisticated initialization techniques can offer benefits, they may not always outperform simpler strategies like identity and copy initialization, which directly utilize pre-existing structures.

❸ Random initialization yields the lowest performance. The variability in task performance with this method highlights the challenges of using non-specific weights, which can lead to unstable and suboptimal model behavior, particularly in complex tasks like math and coding.

Overall, the findings emphasize that initialization strategies that leverage prior information from existing layers tend to provide a better foundation for training expanded models, leading to improved performance. We thus choose $\Pi_{identity}$ as the default initialization strategy.

Method	**Language** ↑							**Math** ↑		**Code** ↑		**Avg.** ↑
	ARC-C	GLUE	MMLU	OBQA	PIQA	TruthfulQA	WinoGrande	GSM8K	MathQA	HumanEval	MBPP	
Random	25.1	41.4	24.4	26.8	50.3	37.7	49.3	0.3	20.3	0.0	1.3	25.17
Identity	**52.5**	**75.4**	51.4	**43.6**	**78.7**	**41.0**	**73.4**	**55.0**	**31.4**	**55.3**	**29.1**	**53.34**
Copy	52.0	71.3	52.3	43.2	78.6	40.8	73.0	52.2	29.0	41.1	26.7	50.93
Linear	48.6	70.7	**53.4**	39.4	72.9	38.7	57.5	29.7	25.7	24.0	0.7	41.94
Slerp	42.2	71.6	49.6	39.8	76.4	36.2	63.0	43.0	27.1	40.4	4.9	44.93

Table 3: Experiments on the effectiveness of different initialization strategies for the expanded blocks. For this study, we evaluate ○ LLaMA-DLO-8B with 10% sparsity.

Method	**Language** ↑							**Math** ↑		**Code** ↑		**Avg.** ↑
	ARC-C	GLUE	MMLU	OBQA	PIQA	TruthfulQA	WinoGrande	GSM8K	MathQA	HumanEval	MBPP	
DLO	**52.5**	**75.4**	51.4	43.6	**78.7**	**41.0**	73.4	**55.0**	**31.4**	55.3	**29.1**	**53.34**
w/o Zero Init	51.9	73.0	**52.1**	**44.6**	77.7	40.3	**74.3**	**55.0**	30.5	**56.1**	28.2	53.06
w/o Rescaling	47.4	71.3	49.4	35.8	75.3	39.6	67.5	35.0	24.5	55.6	28.2	48.15

Table 4: Ablation Study on the effectiveness of zeros router initialization & score rescaling. For this evaluation, we deploy ○ LLaMA-DLO-8B with 10% sparsity for the experiment.

▷ *Zeros Router Initialization.* Initializing the router parameters to zero aims to start the model from a neutral state, avoiding any initial bias toward layer activation or skipping. This method allows the model to learn activation patterns from scratch without being influenced by predefined weights. Results in Table 4 indicate that this approach helps maintain balanced training dynamics and mitigates premature convergence, as reflected in the performance stability observed.

▷ *Score Rescaling.* Score rescaling adjusts the routing scores to maintain them within a specific range, typically 0 to 1. This adjustment is intended to preserve gradient flow and prevent extreme activations, ensuring that the model remains responsive to training signals. Our findings suggest that score rescaling helps avoid over-activation of layers, leading to efficient use of the model's capacity.

The combined use of zeros router initialization and score rescaling appears to prevent performance degradation effectively. As shown in Table 4, models with these techniques generally achieve more consistent accuracy and efficiency across various tasks. These results suggest that careful initialization and rescaling strategies are beneficial for maintaining robust performance during adaptation.

Method	**Language ↑**							**Math ↑**		**Code ↑**		**Avg. ↑**
	ARC-C	GLUE	MMLU	OBQA	PIQA	TruthfulQA	WinoGrande	GSM8K	MathQA	HumanEval	MBPP	
Uniform	36.0	61.7	48.7	36.6	58.4	36.8	68.2	25.4	21.7	23.7	2.2	38.13
Layer-Wise	**52.5**	**75.4**	**51.4**	**43.6**	**78.7**	**41.0**	**73.4**	**55.0**	**31.4**	**55.3**	**29.1**	**53.34**

Table 5: Performance of the fine-tuned ○ LLaMA-DLO-8B with 10% sparsiy and different sparsity distribution strategies. "Uniform" represents all layers use the same sparsity $\rho_i = \rho$ during training. "Layer-Wise" denotes the model maintains different skip rates for different layers.

C Hyperparameter Tuning

We list the key hyperparameters searched in our experiments and mark the adopted values as **bold** below. We select combination of hyperparameters giving best performance via grid search Belete & Huchaiah (2022).

- Base learning rate $\bar{\zeta}$: $4e-5$, $\mathbf{2e-5}$, $1e-5$.
- Batch size: 64, **128**, 256.
- Weight of skip loss ϵ: 0.001, 0.01, 0.1, $\mathbf{1.0}$, 2.0.
- Annealing steps T': 0, **1000**, 2000, 3000.

D Evaluation Details

We list the number of shots and the metric used for each dataset below. Our options of metrics are akin to common practices of previous works Touvron et al. (2023); Wu et al. (2024)

- ARC-C: 25 shots, normalized accuracy.
- GLUE: 0 shot, accuracy.
- MMLU: 5 shots, normalized accuracy.
- PIQA: 0 shot, normalized accuracy.
- OBQA: 0 shot, normalized accuracy.
- TruthfulQA: 0 shot, accuracy.
- WinoGrande: 5 shots, accuracy.
- GSM8K: 5 shots, accuracy.
- MathQA: 0 shot, normalized accuracy.
- HumanEval: 200 rounds, pass@100.
- MBPP: 15 rounds, pass@10.

E Acknowledgment of AI Assistance in Writing and Revision

We utilized ChatGPT-4 for revising and enhancing wording of this paper.

F Reproducibility

The implmentation of our framework, including code for model construction, data preprocessing, and experiments, is released at https://anonymous.4open.science/r/LLaMA-DLO.

G Discussion and Future Work

Due to the computational limits in academic labs, we primarily experiment with the widely used LLaMA2-7B. It is noteworthy that the proposed framework is model-agnostic and is compatible to general transformer-based LLMs. We save the experiments for LLMs in larger sizes or different architectures as future works.

H Pseudo-Code Style Description of Dynamic Layer Operation (DLO)

A pseudo-code style description of DLO is attached in the next page.

Algorithm 1 Dynamic Layer Operation (DLO)

Require: Pre-trained LLM with R layers, group size Q, expansion size q, target overall sparsity ρ, training steps T, annealing steps T', base learning rate $\bar{\zeta}$
Ensure: Optimized LLM with dynamic scaling

Initialize: $P \leftarrow R \,/\, Q$, $Q' \leftarrow Q + q$, $\rho_{i,1} \leftarrow \rho$
// Layer Expansion:
for group $i \leftarrow 1$ to P **do**
  for layer $j \leftarrow Q+1$ to $Q+q$ **do**
    Initialize θ'_{ij} using Π:
    if $\Pi =$ 'Xavier' **then**
      $\theta'_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right)$
    else if $\Pi =$ 'Copy' **then**
      $\theta'_{ij}\mathcal{L}_{\text{skip}}\theta_{i(Q+q-1)}$
    else if $\Pi =$ 'Identity' **then**
      $\theta'_{ij}\mathcal{L}_{\text{skip}}\theta_{i(Q+q-1)}$, $W'_{\text{out}} = 0$
    else if $\Pi =$ 'Linear Merge' **then**
      $\theta'_{ij}\mathcal{L}_{\text{skip}} \sum_{k=1}^{\tau} \alpha_k \theta_{i(Q+q-k)}$
    else if $\Pi =$ 'SLERP' **then**
      $\Omega = \arccos\left(\frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right)$
      $\theta'_{ij}\mathcal{L}_{\text{skip}} \frac{\sin((1-\alpha)\Omega)}{\sin(\Omega)}\mathbf{u} + \frac{\sin(\alpha\Omega)}{\sin(\Omega)}\mathbf{v}$
    end if
  end for
end for
// Layer Activation and Skipping:
for step $t \leftarrow 1$ to T **do**
  // Skip Rate Annealing:
  if $t \leq T'$ **then**
    $\rho^t \leftarrow \bar{\rho} + (\rho - \bar{\rho})\frac{t}{T'}$
  else
    $\rho^t \leftarrow \rho$
  end if
  // Training and Integration:
  $\mathcal{L}_{\text{skip}} \leftarrow 0$
  for layer $i \leftarrow 1$ to R' **do**
    for token s in sequence **do**
      // Dynamic Skip:
      $r_i^s \leftarrow \frac{1}{2}(\beta + (2\sigma(\mathbf{h}_i^s W_i) - 1)\gamma)$
      if training **then**
        $\hat{\lambda}_i^s \leftarrow \mathbb{1}\!\!\!\!\mathbb{1} r_i^s \in \text{Top}_{\lfloor \rho_{i,t} S \rfloor}\left(\{r_i^s\}_{s=1}^{S}\right)$
      else
        $\hat{\lambda}_i^s \leftarrow \mathbb{1}\!\!\!\!\mathbb{1} r_i > \frac{\beta}{2}$
      end if
      if $r_i^s = 1$ **then**
        $\mathbf{h}_{i+1}^s \leftarrow r_i \cdot \mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i^s)$
      else
        $\mathbf{h}_{i+1}^s \leftarrow \mathcal{A}_i(\mathbf{h}_i^s)$
      end if
      // Skip Loss:
      $\mu_i^s \leftarrow \cos(\mathcal{A}_i(\mathbf{h}_i^s), \mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i^s))$
      $\tilde{\lambda}_i^s \leftarrow \mathbb{1}\!\!\!\!\mathbb{1} \mu_i^s \in \text{Bottom}_{\lfloor (1-\rho^t) R' S \rfloor}\left(\{\mu_i^s\}_{i,s=1}^{R',S}\right)$
      $\mathcal{L}_{\text{skip}} \leftarrow \mathcal{L}_{\text{skip}} + \mathcal{L}_{\text{BCE}}(\sigma(\mathbf{h}_i^s W_i), \tilde{\lambda}_i^s)$
    end for
    // Layer-Wise Skip Rate:
    $\rho_{i,t+1} \leftarrow \sum_{s=1}^{S} \tilde{\lambda}_i^s \,/\, S$
  end for
  $\mathcal{L}_{\text{skip}} \leftarrow \mathcal{L}_{\text{skip}} \,/\, R'S$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{skip}}$
  Adjust learning rate $\zeta_{i,t} \leftarrow \bar{\zeta} \cdot \frac{1-\rho_{i,t}}{1-\rho^t}$
end for
return Optimized LLM