DLO: Dynamic Layer Operation for Efficient Vertical Scaling of LLMs

Anonymous ACL submission

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 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 stage, eliminating the need for resource-intensive Continual Pre-Training. 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://anonymous.4open.science/r/LLaMA-DLO.

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Team et al., 2023) have shown remarkable success across various natural language processing (NLP) tasks, leveraging their vast capacity to capture complex patterns in data. Traditional scaling of these models has predominantly focused on *horizontal* expansion, as seen in Mixture-of-Experts (MoE) architectures (Shazeer et al., 2017; Fedus et al., 2022b; Lepikhin et al., 2020), where the width of the model is increased by adding more experts. This approach primarily optimizes parameter usage and computational cost by activating a fixed portion of parameters conditioned on the given input (Fedus et al., 2022a).

However, the potential for *vertical* expansion remains underexplored. Inspired by how the human brain allocates more neurons for complex tasks and forms deeper neural chains (Baddeley, 1992; Koechlin et al., 2003), we propose focusing on vertical scaling. Our method dynamically expands, activates, or skips layers to optimize model depth and reduce redundancy, as shown in Figure 1 (a). Three critical challenges exist in vertically scaling LLMs: ❶ **Optimization Complexity.** Calculating optimal combinations of adding or pruning layers has been proved to be an NP-hard problem (Glorot and Bengio, 2010; Hestness et al., 2017), while an approximation method (Wang et al., 2023a) has shown compromised improvement. ❷ **Computation Cost.** The inherent computational cost is associated with processing deeper networks. Each additional layer contributes to the overall latency and resource consumption. ❸ **Feature Collapse.** Our analysis in Figure 1 (b) reveals that for a significant number

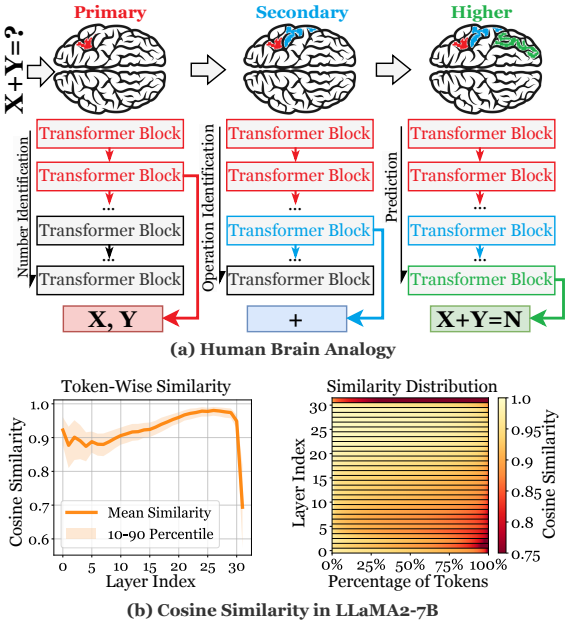

Figure 1: (a) DLO structure that ensembles human brain activities in a math problem example (Koechlin et al., 2003), where the primary neurons preceive numbers, secondary neurons understand operations, and high-order neurons calclulate the results. (b) Layer-wise token similarity and distribution.

of inputs, the representations across consecutive layers exhibit substantial similarity, suggesting that many layers may be redundant for certain samples.

To address these challenges, in this paper, we propose **D**ynamic **L**ayer **O**peration (**DLO**), that consists of three operations: (***i***) `expansion`, (***ii***) `activation`, and (***iii***) `skipping`, for dynamic vertical scaling of LLMs without a proportional increase in computational cost. Our specific designs are as follows: ❶ **`Expansion`**: *Additional layers* are dynamically expanded from existing ones, easing optimization complexity. ❷ **`Activation & Skip`**: *Feature Similarity* guides the activation and skipping of layers. We propose similarity-induced labels to train the router that controls these operations. ❸ **Adaptive FLOPs**: *Sparsity settings* vary for layers to facilitate adaptive FLOPs for different tokens, maintaining efficiency. ❹ **Enhanced Generalizability**: *Layer-specific learning rates*, based on sparsity, further improve the model's ability to generalize across tasks. ***Note*** that all modules are trained during the Supervised Fine-Tuning (SFT) stage, eliminating the need for Continual Pre-training (CPT) and simplifying the training process. Our primary contributions are as follows:

- ***Method.*** We introduce a novel method, **DLO**, for dynamically scaling LLMs vertically by dynamically expanding, activating, or skipping layers.
- ***Performance & Efficiency.*** Through rigorous experiments, we demonstrate that **DLO** not only surpasses the performance of the original unscaled models but also achieves comparable results to densely expanded models with significantly enhanced efficiency.
- ***Applicability.*** Fine-tuned on *language understanding*, *math*, and *coding* tasks, we manifest DLO's effectiveness across multiple NLP tasks.

2 Related Work

2.1 Mixture-of-Experts (MoE)

MoE architectures have emerged as a promising approach for enhancing the efficiency and scalability of LLMs (Shazeer et al., 2017). Traditional neural networks activate all parameters for every input, leading to significant computational overhead, particularly as models scale up. In contrast, MoE models activate only a subset of parameters for each input, optimizing computational resource usage and enabling models to scale to billions of parameters without a corresponding increase in computational cost per input (Lepikhin et al., 2020; Fedus et al., 2022b; Zoph et al., 2022; Zhu et al., 2024; Jiang et al., 2024). This selective activation makes MoE highly efficient for both training and inference by focusing on horizontal expansion and adding more experts. However, MoEs primarily aim to optimize width, potentially leaving layer redundancy unaddressed. Our Dynamic Layer Operation (DLO) approach complements MoE by focusing on vertical scaling through dynamic layer expansion and activation, targeting depth scalability and reducing potential feature redundancy.

2.2 Efficient Model Stacking

Model stacking is a common ensemble learning technique that improves predictive performance by combining multiple models to leverage their complementary strengths (TING, 1997; Chen et al., 2015). In the context of LLMs, stacking can involve integrating various models into a hierarchical structure, where outputs from one model serve as inputs to another, capturing a broader range of features and patterns (Dabre and Fujita, 2019; Chen et al., 2021a; Wang et al., 2023b; Kim et al., 2023).

Recent advancements have focused on progressively stacking pre-trained transformer or self-attention layers to create composite language models (Gong et al., 2019; Gu et al., 2020; Evci et al., 2022; Yao et al., 2023; Du et al., 2024; Wu et al., 2024). This approach reduces training costs by reusing pre-trained components. However, stacked models' increased depth and complexity lead to high inference latency.

To mitigate this issue, layer-skipping methods have been developed, allowing models to "early exit" using additional layer-wise classifiers, thereby reducing the number of layers processed during inference (Wang et al., 2022; Chen et al., 2023; Zhang et al., 2024). More recently, conditional computation techniques, such as MoD (Raposo et al., 2024), have been proposed to dynamically skip layers based on token-specific conditions, further enhancing efficiency. However, these methods often require modifications during the pre-training stage, adding computation complexity and limiting their application to existing pre-trained LLMs. In contrast, our DLO method focuses on efficiency and scalability through dynamic vertical scaling within a single model during the SFT stage. It provides a comprehensive, high-performance solution to scaling LLMs without the extensive computational demands associated with stacked ensembles.

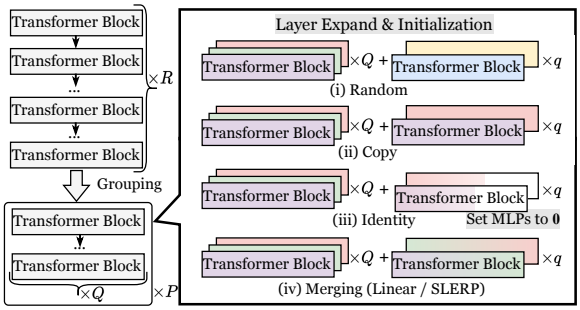

Figure 2: Layer extension with initialization strategies.

3 Methodology

In this section, we introduce the **D**ynamic **L**ayer **O**peration (**DLO**) framework for efficient vertical scaling of LLMs. DLO consists of three key operations: `expansion`, `activation`, and `skipping`. These operations dynamically adjust the model structure during the Supervised Fine-Tuning (SFT) phase to optimize computational efficiency and improve performance. A pseudo-code style description is included in Appendix E.

3.1 Layer Expansion

To facilitate dynamic depth adjustment, we introduce a group-based layer expansion strategy. Suppose the LLM has R transformer layers, which we group into P groups with Q layers each, such that $R = P \times Q$. Each group is expanded to $Q' = Q + q$ layers, where q is the number of additional layers introduced per group. The resulting number of layers will be $R' = P \times Q'$.

Let $\mathcal{G}_i$ denote the i-th group with layers $\mathcal{L}_{i1}, \mathcal{L}_{i2}, \ldots, \mathcal{L}_{iQ}$. The expanded group $\mathcal{G}'_i$ will contain layers $\mathcal{L}_{i1}, \mathcal{L}_{i2}, \ldots, \mathcal{L}_{i(Q+q)}$. The expanded layers are initialized using a policy Π, and we consider several initialization strategies:

- **Random Initialization** (Π_{rand}): Initialize the new layers' weights θ'_{ij} using Xavier initialization (Glorot and Bengio, 2010) [Figure 2 (i)].

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

  where $\forall j \in \{Q+1, Q+2, \ldots, Q+q\}$, $\mathcal{U}$ denotes the uniform distribution, n_{in} is the number of input neurons, and n_{out} is the number of output neurons in the layer.

- **Copy from Previous Layer** (Π_{copy}): Copy parameters from the preceding layer [Figure 2 (ii)].

$$\theta'_{ij} = \theta_{i(Q+q-1)}, \forall j \in \{Q+1, \ldots, Q+q\}.$$

- **Identity Initialization** (Π_{Identity}) (Wu et al., 2024): Copy from the preceding layer but set the output linear matrix of the multi-head self-attention (MHSA) to zero [Figure 2 (iii)].
  a. Copy the parameters of the previous layer: $\theta'_{ij} = \theta_{i(Q+q-1)}, \forall j \in \{Q+1, \ldots, Q+q\}$.
  b. Set the weights of the output linear layer W'_{out} in the MHSA to zero: $W'_{\text{out}} = 0$.

  In this way, the output of the expanded layers will preserve the features from the original layers.

- **Linear Merge** (Π_{linear}): Merge from the preceding τ layers via a linear function [Figure 2 (iv)].

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

- **Spherical Linear Interpolation (SLERP)** (Π_{slerp}) (Shoemake, 1985): Merge from the preceding τ layers using SLERP, shown in Figure 2 (iv). The SLERP method smoothly interpolates between two weight vectors on a unit sphere, maintaining constant velocity. The interpolation between two weight vectors $\mathbf{u}$ and $\mathbf{v}$ is defined:

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

  where Ω is the angle between $\mathbf{u}$ and $\mathbf{v}$:

$$\Omega = \arccos\left(\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right),$$

  and $\alpha \in [0, 1]$ is the interpolation parameter. In our context, for the new layer j, the parameters are initialized by interpolating between the weights of the previous layers $\theta_{i(Q+q-1)}$ and $\theta_{i(Q+q-\tau)}$:

$$\theta'_{ij} = \text{SLERP}(\theta_{i(Q+q-1)}, \theta_{i(Q+q-\tau)}, \alpha),$$

  where α controls the interpolation. For example, $\alpha = 0.5$ provides a midpoint interpolation, while $\alpha = 0$ or $\alpha = 1$ will take the exact values of $\theta_{i(Q+q-1)}$ or $\theta_{i(Q+q-\tau)}$, respectively. This ensures a smooth transition between layers, aiding in gradient flow and stable training.

We conduct comprehensive experiments on the choice of the policy Π in Section 4.3.

3.2 Layer Activation & Skipping

DLO dynamically skips the multi-layer perceptron (MLP) modules within the transformer layer $\mathcal{L}_i$ for input tokens. We observe significant performance

degradation if attention modules are skipped. To achieve this, we use a linear router to determine the activation of layers. Suppose we have the set of token embeddings in a sequence of length S for a given layer $\mathcal{L}_i$, that is $\mathbf{h}_i = \{\mathbf{h}_i^s | s \in \mathbb{N}^*, s \leq S\}$. In the following contents, we omit the superscript s for better readability.

Considering feature redundancy, we use the router weights W_i to process the input token $\mathbf{h}_i$ and obtain the decision score r_i, which is given by:

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

where σ is the sigmoid function, β and γ are hyper-parameters controlling the output range. During the inference stage, this score r_i determines whether layer $\mathcal{L}_i$ is active: the layer is activated if and only if $\mathcal{L}_i \geq \frac{\beta}{2}$, otherwise it's skipped. The final activated output for the layer is:

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

where $\mathcal{M}_i$, $\mathcal{A}_i$ are the MLP and the attention modules within layer $\mathcal{L}_i$, respectively. This activation & skipping mechanism aims to encourage the utilization of the most relevant layers, thus reducing unnecessary computation. In this paper, we initialize the router weights W_i as zeros and set $\beta = 2.0, \gamma = 0.05$, so that $r_i = 1.0$ on the first step and $r_i \in (0.975, 1.025)$ during training. This ensures benign initialization for activated tokens and avoids excessive disturbance on activated outputs brought by the decision scores.

3.3 Training and Integration

As shown in Figure 3. The training procedure includes a general language model loss for downstream tasks and a router skip loss for guiding the learning of the router, which controls the dynamic layer skipping. Details are as follows.

Similarity-induced Label & Router Skip Loss. Given a predefined overall sparsity ρ, we define a sparsity factor ρ_i for each layer $\mathcal{L}_i$ that controls the layer-wise activated tokens. To determine the status of token $\mathbf{h}_i^s$, we utilize predicted router labels $\hat{\Lambda}^s = \{\hat{\lambda}_1^s, \ldots, \hat{\lambda}_i^s, \ldots, \hat{\lambda}_{R'}^s\}$, which is obtained through the decision scores r_i^s:

$$\hat{\lambda}_i^s = \begin{cases} 1 & \text{if } r_i^s \in \text{Top}_{\lfloor(1-\rho_i)S\rfloor}\left(\{r_i^s\}_{s=1}^S\right), \\ 0 & \text{otherwise}. \end{cases} \tag{3}$$

where $\hat{\lambda}_i^s = 1$ indicates layer $\mathcal{L}_i$ is predicted to be activated, and vice versa. To train the routers, we utilize the supervised router labels $\tilde{\Lambda}^s = \{\tilde{\lambda}_1^s, \ldots, \tilde{\lambda}_i^s, \ldots, \tilde{\lambda}_{R'}^s\}$ to guide the learning of sparsity, *i.e.*, to skip or not. The router labels $\tilde{\lambda}_i^s$ of layer $\mathcal{L}_i$ at training step t is determined as:

1. The cosine feature similarity of layer i is calculated through features across MLP $\mathcal{M}_i$:

$$\mu_i^s = \frac{\mathcal{A}_i(\mathbf{h}_i^s) \cdot \mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i^s)}{\|\mathcal{A}_i(\mathbf{h}_i^s)\| \|\mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i^s)\|} \in [0, 1]. \tag{4}$$

2. The similarity is sorted over all the layers, and the similarity-induced label is given as follows:

$$\tilde{\lambda}_i^s = \begin{cases} 1 & \text{if } \mu_i^s \in \text{Bottom}_{\lfloor(1-\rho)R'S\rfloor}\left(\{\mu_i^s\}_{i,s=1}^{R',S}\right), \\ 0 & \text{otherwise}. \end{cases} \tag{5}$$

   where $\text{Bottom}_{\lfloor(1-\rho)R'S\rfloor}$ indicates the labels for tokens with the least $\lfloor(1-\rho)R'S\rfloor$ portion of cosine similarity are set to 1s, which are expected to be activated.

3. A skip loss $\mathcal{L}_{\text{skip}}$ based on the Binary-Cross-Entropy loss $\mathcal{L}_{\text{BCE}}$ is incorporated to guide the learning of the router attached to each layer:

$$\mathcal{L}_{\text{skip}} = \frac{1}{R'S} \sum_{i,s=1}^{R',S} \mathcal{L}_{\text{BCE}}(\sigma(\mathbf{h}_i^s W_i), \tilde{\lambda}_i^s). \tag{6}$$

Given the task-specific loss $\mathcal{L}_{\text{task}}$, and a weight parameter γ, the overall loss function for DLO is:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \epsilon \mathcal{L}_{\text{skip}}, \tag{7}$$

Skip Rate Dynamics. The redundancy exhibits an imbalanced distribution across layers, as is evidenced in Figure 1 (b). To this end, we adjust the next-step $\rho_{i,t+1}$ for each training step t over the total T steps, where the initial skip rate $\rho_{i,1} = \rho$. The layer-wise sparsity factor $\rho_{i,t+1}$ is calculated using the router labels as follows:

$$\rho_{i,t+1} = \frac{\sum_{s=1}^S \tilde{\lambda}_{i,t}^s}{S}, \tag{8}$$

where $\tilde{\lambda}_{i,t}^s$ is the supervised router label of the s-th token in layer $\mathcal{L}_i$ at step t. Additionally, we employ an annealing technique on the skip rate to ensure a warm start. During training, the overall skipping rate gradually increases from an initial low value $\bar{\rho}$ to the target sparsity level ρ over a predefined number of steps T'. The overall skip rate ρ^t at step t is given by:

$$\rho^t = \begin{cases} \bar{\rho} + (\rho - \bar{\rho})\frac{t}{T'} & \text{if } t \leq T', \\ \rho & \text{otherwise}, \end{cases} \tag{9}$$

where we set $\bar{\rho} = 0$. This annealing process helps the model progressively adapt to higher sparsities with smoother changes, leading to more stable training and convergence.

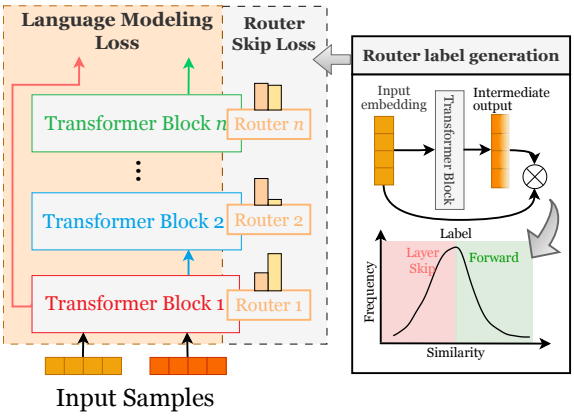

Figure 3: Training pipeline of **DLO**, consists of the traditional language modeling loss and an auxiliary router skip loss supervised by generated router labels.

Layer-Wise Learning Rates. DLO also employs layer-wise learning rates $\zeta_{i,t}$, adjusted based on sparsity to promote generalizability. The learning rate for each layer is:

$$\zeta_{i,t} = \bar{\zeta} \cdot \frac{1 - \rho_{i,t}}{1 - \rho^t}, \tag{10}$$

where $\bar{\zeta}$ is the base learning rate. It is noteworthy that **all DLO components are trained** during the Supervised Fine-Tuning (SFT) stage in an end-to-end manner, eliminating the need for Continual Pre-Training (CPT). DLO achieves dynamic vertical scaling, optimize model depth, and maintain high performance with reduced computation.

3.4 Adaptive Inference-Time FLOPs

During inference time, DLO uses layer-specific sparsity settings to maintain computational efficiency and ensure adaptive floating-point operations (FLOPs) for different tokens. In other words, the predicted sparsity $\hat{\rho}_i$ will be determined completely by the router based on Equation (1)-(2). The adaptive FLOPs are computed as follows:

$$\text{FLOPs}_i = \hat{\rho}_i \cdot \text{FLOPs}_{\text{full}}, \tag{11}$$

where $\text{FLOPs}_{\text{full}}$ represents the FLOPs for a fully active layer. Since $\hat{\rho}_i$ is predicted based on each specific token, DLO achieves adaptive FLOPs that entail better generalizability.

4 Experiments

4.1 Experimental Settings

Model Selection. We utilize LLaMA2-7B (Touvron et al., 2023) as the primary backbone due to its open-source availability and extensive usage. It consists of $R = 32$ original transformer layers, which we group into $P = 4$ clusters, each containing $Q = 8$ layers, akin to Wu et al. (2024). For expansion, we increase the group size to $Q' = 10$ layers, resulting in a dense model, LLaMA-DLO, with a total of 40 layers and 8 billion parameters. For comparison, we also employ LLaMA-Pro-8B (Wu et al., 2024), a competitive model trained with Continual Pre-Training (CPT) on specialized datasets. We demonstrate DLO achieves a favored balance between performance and computation cost in Section 4.4. DLO is model-agnostic. Due to the computational limits in academic labs, experiments for various LLMs in larger sizes are future works.

Fine-tuning Details. Following common practices (Wu et al., 2024), we fine-tune using a mixture of 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 replicated three times, totaling approximately 1.44 million instances. We use a batch size of 128 and a maximum sequence length of 4,096 tokens. The learning rate is set to $2e-5$ with a warmup ratio of 0.03 and cosine scheduling, and we utilize AdamW (Loshchilov and Hutter, 2017) as the optimizer. Flash Attention (Dao et al., 2022) and bfloat16 mixed-precision training are adopted to accelerate training. Fine-tuning LLaMA-DLO under different skip ratios yields the sparse models, with each training run taking approximately 36 hours on eight NVIDIA A100 GPUs. Hyperparameter tuning is explained in Appendix A with more details.

Evaluation Benchmarks. We assess the fine-tuned models using the 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)]. The primary metrics utilized are accuracy. More details are described in Appendix B.

4.2 Overall Performance

Table 1 summarizes the performance of the DLO framework across various datasets using LLaMA2-7B as the backbone. From the results, we draw several key observations:

	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$_{+\text{SFT}}$		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$_{+\text{SFT}}$		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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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$_{+\text{SFT}}$	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.

❶ Dense Models' Superiority: Dense models (•) generally outperform their sparse counterparts (○) across most datasets, especially for harder tasks like *Math* and *Code*. For instance, the best-performed dense model • LLaMA-DLO-8B+$_{\text{SFT}}$ achieve better average scores of 53.90, compared to the best-performed sparse model ○ LLaMA-DLO-8B+$_{\text{SFT}}$'s 53.34.

❷ Efficiency of Sparse Models: Sparse models (○) show a notable reduction in inference-time FLOPs while maintaining competitive accuracy. At 10% sparsity, the ☆ LLaMA-DLO-8B+$_{\text{SFT}}$ model achieves an average score of 53.34 with 34.2T FLOPs, compared to the dense counterpart's 53.90 with 36.5T FLOPs. This demonstrates the efficiency of DLO's `activation` and `skipping` operations in optimizing computational resources without significant performance drop. We also observe that smaller sparsities lead to minimal performance drop, while the performance degrades significantly if sparsity is too large (*e.g.*, $\rho = 30\%$).

❸ DLO-`Expansion` Advantages: Models expanded using DLO `expansion` with up to 8B parameters outperform the original LLaMA2-7B model across multiple metrics. For example, dense • LLaMA-DLO-8B+$_{\text{SFT}}$ achieves a higher average score of 53.90 compared to • LLaMA2-7B+$_{\text{SFT}}$'s 52.89, and another SOTA expansion method • LLaMA-Pro-8B+$_{\text{SFT}}$'s 53.53, highlighting the effectiveness of vertical scaling through layer expansion in improving model capacity.

❹ Balanced Performance of Sparse Models: Sparse models (○) with DLO's dynamic `activation` and `skipping` provide a well-balanced trade-off between performance and computational efficiency. At 30% sparsity, LLaMA-DLO models maintain strong performance on tasks like GLUE (73.0 vs. 2.8) and HumanEval (31.2 vs. 0.0) while significantly reducing inference FLOPs compared to their dense counterparts.

❺ General Observations: Overall, the DLO framework successfully balances performance and efficiency across various datasets and tasks. The adoption of both `expansion` and `skipping` strategies enables LLaMA-DLO to achieve robust performance improvements while maintaining lower computational costs, suggesting that DLO is a viable approach for scalable LLM deployment.

We conduct further analysis of each component, which is provided in the subsequent subsections.

4.3 Ablation Studies

Initialization Strategies for Expanded Layers. We explored the effectiveness of various layer initialization strategies for the expanded layers. Table 2 evaluates the impact of different initialization strategies on the performance of LLaMA-DLO models with 10% sparsity. The results show that the choice of initialization plays a critical role in determining model performance across tasks.

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 a wide

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 2: 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 3: 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.

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 4: 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 $\rho = 10\%$.

range of 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 model 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.

Zeros Router Initialization & Score Rescaling. In this experiment, we investigate the impact of zeros router initialization and score rescaling on performance degradation.

▷ *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 3 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 more 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 3, 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.

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 4, 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, showing the importance of fine-tuning in maximiz-

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, as described in Section 3.3.

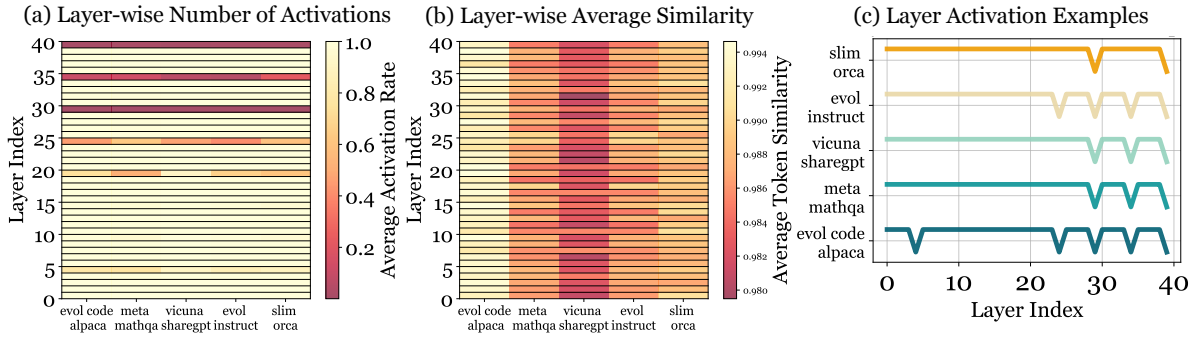

Figure 4: Visualization on different datasets of (a) Layer-Wise Number of Activations, (b) Layer-Wise Average Similarity, and (c) Token Activation Examples.

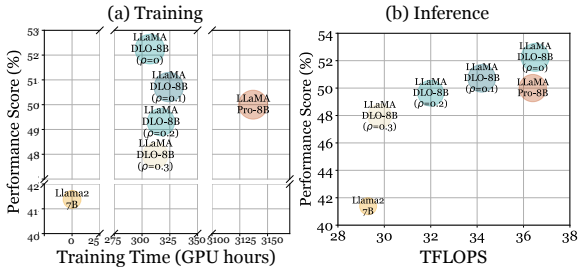

Figure 5: (a) Performance (↑) *v.s.* Training time (↓). LLaMA-Pro is reported in H800 GPU hours quoted from the original paper. The rest are reported in A100 GPU hours. (b) Performance (↑) *v.s.* Inference FLOPs (↓). DLO achieves the best trade-off between performance and training / inference costs.

ing the effectiveness of expansion.

Layer-Wise Skip Rates & Sparsity Allocation. This experiment evaluates the impact of using layer-specific skip rates on sparsity allocation.

▷ *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.

▷ *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.

4.4 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.

Limitation Discussions & Future Work

Disentanglement of Routing Decisions and Rescaling Scores. Currently, the routing decisions and rescaling scores in our framework are interdependent, which may impact the model's accuracy. For instance, the skipped outputs are optimized to match the original outputs primarily through cosine similarity, which does not account for the difference in L2 magnitude. This discrepancy could potentially be mitigated by applying an additional rescaling factor to the skipped outputs, ensuring a better match in magnitude and improving the overall performance.

Improved Supervision for Router Labels. The current method relies on cosine similarity for supervising router labels, which may not be the most effective approach. Exploring alternative supervision methods, such as task-specific metrics or direct gradients, could lead to more accurate router decisions. For example, drawing inspiration from works like Jiang et al. (2023) or employing gradient-based techniques similar to those used in network pruning, could enhance the router's ability to prioritize important tokens and improve performance.

Skipping Attention Modules. This work primarily focuses on skipping MLP layers due to the observed instability in decoding when skipping attention modules, such as excessive repetition and degraded accuracy (e.g., achieving 0% accuracy on GSM8K). Future work could explore strategies to stabilize the skipping of attention layers, potentially improving model efficiency without compromising output quality.

Ethical Statement

The development and deployment of large language models, including the LLaMA-DLO framework presented in this work, can raise important ethical considerations. Our research aims to enhance the efficiency and scalability of LLMs while maintaining high standards of responsibility and ethical practice. We recognize the potential impact of our work on various stakeholders and are committed to the following ethical principles:

Fairness and Bias Mitigation. We are aware that language models can inadvertently learn and propagate biases present in training data. Efforts have been made to ensure that LLaMA-DLO is trained on diverse and representative datasets to minimize the risk of bias. Effective ways to mitigate bias is a pressing problem worth further study.

Transparency and Accountability. We strive to maintain transparency in our research and development processes. Detailed documentation and open access to our methodologies and results will be provided to allow for scrutiny and reproducibility. Accountability mechanisms are in place to ensure that any adverse effects of our technology are promptly identified and addressed.

Social Impact. The potential societal impact of LLaMA-DLO is carefully evaluated to prevent misuse or harm. We are committed to the ethical deployment of our technology, ensuring it is used for beneficial purposes such as advancing research, improving accessibility, and enhancing communication. We actively discourage and take steps to prevent the use of our models for malicious activities, misinformation, or any application that could harm individuals or society.

Continuous Ethical Review. The ethical implications of our work are continually assessed to adapt to evolving norms and expectations. We engage with interdisciplinary experts and stakeholders to identify and address ethical concerns, ensuring that our research and its applications remain aligned with societal values and ethical standards.

By adhering to these principles, we aim to contribute positively to the field of artificial intelligence and ensure that the benefits of our research are realized in an ethical and responsible manner.

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*.

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

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

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

Daniel Mesafint Belete and Manjaiah D Huchaiah. 2022. 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.

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

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

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

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. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

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

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. *arXiv preprint arXiv:1803.05457*.

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

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

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

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

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

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

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

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. 2023. A framework for few-shot language model evaluation.

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

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

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

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

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

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. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

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

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

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

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

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

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

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

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*.

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

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

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

EvolCodeAlpaca Team. 2022. Evolcodealpaca dataset.

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

Self-Duplicate Stack Team. 2024. Self-duplicate stack.

ShareGPT Team. 2023a. Sharegpt dataset.

SlimOrca Team. 2023b. Slimorca dataset.

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

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*.

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

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

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

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

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

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

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

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

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *Preprint*, arXiv:2406.16554.

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

A 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 and 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, **1.0**, 2.0.
- Annealing steps T': 0, **1000**, 2000, 3000.

B 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.

C Acknowledgment of AI Assistance in Writing and Revision

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

D 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.

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

A pseudo-code style description of DLO is attached here.

F 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.

Algorithm 1 Dynamic Layer Operation (DLO)

Input: 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 $\bar{\zeta}$

Output: 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**
        $\hat{\lambda}_i^s \leftarrow \nVdash r_i^s \in \text{Top}_{\lfloor \rho_{i,t} S \rfloor}\left(\{r_i^s\}_{s=1}^{S}\right)$
      else
        $\hat{\lambda}_i^s \leftarrow \nVdash r_i > \frac{\beta}{2}$
      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:
      $\mu_i^s \leftarrow \cos(\mathcal{A}_i(\mathbf{h}_i^s), \mathcal{M}_i \circ \mathcal{A}_i(\mathbf{h}_i^s))$
      $\tilde{\lambda}_i^s \leftarrow \nVdash \mu_i^s \in \text{Bottom}_{\lfloor (1-\rho^t)R'S \rfloor}\left(\{\mu_i^s\}_{i,s=1}^{R',S}\right)$
      $\mathcal{L}_{\text{skip}} \leftarrow \mathcal{L}_{\text{skip}} + \mathcal{L}_{\text{BCE}}(\sigma(\mathbf{h}_i^s W_i), \tilde{\lambda}_i^s)$
    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