Qeios

Router-Tuning: A Simple and Effective Approach for Enabling Dynamic-Depth in Transformers

Shwai He[1] **Tao Ge**[2] **Guoheng Sun**[1] **Bowei Tian**[1]
Xiaoyang Wang[2] **Ang Li**[1] **Dong Yu**[2]
[1]University of Maryland, College Park [2]Tencent AI Lab, Bellevue, WA
shwaihe@umd.edu

Abstract

Traditional transformer models often allocate a fixed amount of computational resources to every input token, leading to inefficient and unnecessary computation. To address this, the Mixture of Depths (MoD) was introduced to dynamically adjust the computational depth by skipping less important layers. Despite its promise, current MoD approaches remain under-explored and face two main challenges: (1) *high training costs due to the need to train the entire model along with the routers that determine which layers to skip*, and (2) *the risk of performance degradation when important layers are bypassed*. In response to the first issue, we propose Router-Tuning, a method that fine-tunes only the router on a small dataset, drastically reducing the computational overhead associated with full model training. For the second challenge, we propose *MindSkip*, which deploys *Attention with Dynamic Depths*. This method preserves the model's performance while significantly enhancing computational and memory efficiency. Extensive experiments demonstrate that our approach delivers competitive results while dramatically improving the computation efficiency, e.g., 21% speedup and only a 0.2% performance drop. The code is released at https://github.com/CASE-Lab-UMD/Router-Tuning.

1 Introduction

Transformer-based large language models have shown promising performance across various domains (OpenAI et al., 2024; Team, 2024). However, the ever-increasing model size leads to substantial computational costs in real-world applications, making computation reduction a critical research focus for the efficiency of large language models (Sun et al., 2024; Lin et al., 2024).

A promising approach to address this issue is the Mixture of Depths (MoD) (Raposo et al., 2024), which dynamically allocates computational resources based on input complexity. Rather than applying the entire model uniformly to each input, MoD activates only a subset of the model's layers, skipping those deemed less important. This selective activation substantially reduces computation costs. Despite its potential, current MoD methods MoD methods are still underexplored and face several critical challenges. On the one hand, the involvement of additional router networks, which decide which layers to skip, often requires training the entire model from scratch (Raposo et al., 2024) or performing costly continual pretraining (Tan et al., 2024). This creates a significant barrier to efficiently integrating MoD with existing LLMs. Furthermore, most prior MoD implementations have been applied primarily to transformer blocks and MLP layers, which are sensitive to skipping, leading to performance degradation when important components are omitted (He et al., 2024b).

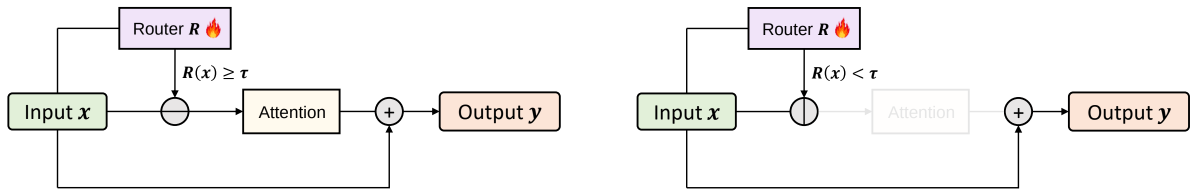
These challenges prompt us to consider two key questions: (1) *How can we implement dynamic depth in a way that reduces training costs and time?* (2) *How can we enhance model efficiency without sacrificing performance, i.e., achieving faster inference while maintaining competitive task performance?*

To tackle the first challenge, we introduce *Router-Tuning*, a novel method that fine-tunes only the router network without updating the backbone model's parameters. As each router network is a lightweight, single-layer projector that accounts for less than 0.01% of the total parameters, the training overhead is minimal and even significantly lower than that of parameter-efficient finetuning methods like LoRA (Hu et al., 2022). Router-tuning requires only a small-scale dataset and fewer training steps, eliminating the need for large-scale pretraining or extensive continual training.

To address the second challenge, we propose *Attention with Dynamic Depths (MindSkip)*, which applies dynamic depth selectively to the attention

https://doi.org/10.32388/KYQECV

Figure 1: **Overview of MindSkip.** For simplicity, LayerNorm before Attention is omitted. Unlike traditional Attention, MindSkip processes the input only when the routing score $R(x) \geq \tau$. During Router-Tuning, only the Router is trainable to enable dynamic depth.



layers. Prior research has highlighted attention redundancy, showing that many attention layers can be skipped without a significant loss in performance (He et al., 2024b). By focusing on these layers, MindSkip not only preserves model accuracy but also alleviates computational and memory bottlenecks—such as the quadratic complexity in attention mechanisms and memory-intensive operations like KV caching.

Through extensive experiments, we demonstrate the effectiveness of our approach across multiple open-source language models, including Llama (Touvron et al., 2023), Mistral (Jiang et al., 2023), and Qwen (Bai et al., 2023). Router-tuning takes less than half an hour on an Nvidia RTX A6000, making it much faster than DLO (Tan et al., 2024). By fine-tuning MindSkip, our approach preserves a high percentage of the original model's performance while significantly reducing memory usage and speeding up inference. In contrast, applying MindSkip to MLP layers or transformer blocks results in notable performance degradation, underscoring the effectiveness of focusing on attention layers for dynamic computation allocation.

2 Methodology

Motivation The Mixture of Depths (MoD) framework dynamically adjusts the depth of layers in a network based on the input, optimizing computational efficiency by skipping less important layers (Raposo et al., 2024). Initially, MoD was designed to be integrated during the pretraining phase, where transformer models are trained from scratch with MoD-enabled layers. More recently, (Tan et al., 2024) applied MoD to pretrained Llama models (Touvron et al., 2023) using continual training. While these methods have shown promise, they are computationally expensive and time-consuming, limiting their scalability and practical use. A more efficient alternative is to apply MoD to existing pretrained models and perform small-scale fine-tuning on a subset of model parameters (Houlsby et al., 2019; Hu et al., 2022), significantly reducing both the computational overhead and training time.

MoD has typically been implemented at the transformer block. However, skipping entire transformer blocks has shown to be suboptimal, resulting in notable performance drops. This is because certain layers within a block are more critical than others. Specifically, skipping Attention layers leads to only minor performance degradation, while skipping MLP layers causes a significant performance drop, comparable to skipping entire blocks (He et al., 2024b). Moreover, Attention layers are computationally expensive, scaling quadratically with sequence length and consuming additional memory to maintain the KV cache. These observations motivate us to focus on Attention layers as the primary target for dynamic depth adjustments, using Block and MLP layers as baselines for comparison.

MindSkip: Attention with Dynamic Depth As shown in Figure 1, an Attention layer with dynamic depth incorporates an additional router that determines whether to skip the layer. Given an input $\boldsymbol{x} \in \mathbb{R}^{s \times d}$, the router first computes an importance score for the input:

$$\boldsymbol{R}(\boldsymbol{x}) = \text{sigmoid}(\boldsymbol{W}\bar{\boldsymbol{x}}), \quad \bar{\boldsymbol{x}} = \frac{1}{s}\sum_{i}^{s} \boldsymbol{x}_i, \quad (1)$$

$$\boldsymbol{M} = \begin{cases} 1, & \text{if } \boldsymbol{R}(\boldsymbol{x}) \geq \tau \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

where $\boldsymbol{R}$ is a scoring function that assesses the importance score of the input, $\boldsymbol{M}$ is the binarized mask used to ensure stable output (Tan et al., 2024), and τ is the threshold that determines whether to skip the layer. Note we consider dynamic depth at the sequence level rather than the token level to avoid an unbalanced number of tokens across different sequences.

To make the binary decision differentiable and trainable, we employ the straight-through estimator

Table 1: **Experimental results of MindSkip deployed at different granularities**. While MindSkip is primarily applied to Attention layers, we also evaluate its performance on Block and MLP layers for comparison. The number of skippable layers is constrained to 16, and the overall capacity of MindSkip is 50%.

Llama-3-8B											
Method	Granularity	Speedup	ARC-C	BoolQ	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Avg.
Baseline	–	1.00×	58.1	81.3	82.1	65.3	45.0	80.5	67.2	77.7	69.7
MindSkip	Block	**1.27**×	44.5	78.0	62.6	64.6	34.2	70.3	65.3	71.2	61.3
	MLP	1.06×	45.1	77.7	65.4	62.4	33.4	71.6	66.4	72.1	61.8
	Attn	**1.21**×	**56.6**	**80.5**	**80.7**	**65.1**	**44.6**	**80.5**	**69.7**	**77.7**	**69.4**
Llama-3-8B-Instruct											
Method	Granularity	Speedup	ARC-C	BoolQ	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Avg.
Baseline	–	1.00×	62.1	83.2	78.8	65.7	42.8	78.7	67.5	75.9	69.3
MindSkip	Block	**1.27**×	44.7	81.2	54.5	60.6	32.4	64.6	67.1	64.8	58.7
	MLP	1.06×	41.8	75.1	59.3	64.5	31.2	68.2	66.7	68.8	59.5
	Attn	**1.21**×	**60.4**	**83.3**	**76.9**	**65.7**	**43.0**	**78.2**	**68.2**	**76.9**	**69.1**

(STE) (Bengio et al., 2013), allowing gradients to propagate through the binary decision-making process. The output of MindSkip is then computed as follows:

$$\boldsymbol{y} = \text{Attention}(\boldsymbol{x}) \cdot \boldsymbol{M} + \boldsymbol{x}. \tag{3}$$

where $\boldsymbol{y}$ is the output. This formulation ensures that the router is fully trainable through the following gradient calculations:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{R}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{M}} \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{R}} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}. \tag{4}$$

During inference, we further optimize computational efficiency by bypassing the Attention computation entirely for skipped inputs:

$$\boldsymbol{y} = \begin{cases} \text{Attention}(\boldsymbol{x}) + \boldsymbol{x}, & \text{if } \boldsymbol{R}(\boldsymbol{x}) \geq \tau \\ \boldsymbol{x}, & \text{otherwise} \end{cases}. \tag{5}$$

This dynamic routing mechanism ensures that Attention computation only occurs when necessary, i.e., when the score $\boldsymbol{R}(\boldsymbol{x})$ meets or exceeds the threshold τ, improving both computational and memory efficiency.

Router-Tuning Given the large size of LLMs, training the entire model is often computationally prohibitive. Our goal is to implement Dynamic Depth efficiently in terms of computational costs and time overhead. To this end, we only finetune the router to avoid a costly training budget.

Specifically, we employ two training objectives: task-specific performance and MindSkip capacity (the proportion of non-skipped inputs). On one hand, the model with MindSkip should maintain the performance of the original model. On the other hand, reducing capacity decreases computational cost and speeds up inference. Based on this, the training objective is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \cdot \mathcal{L}_{\text{MoD}}, \quad \mathcal{L}_{\text{MoD}} = ||M||_0. \tag{6}$$

where $\mathcal{L}$ is the standard loss function, e.g., cross-entropy, while $\mathcal{L}_{\text{MoD}}$ is the l_0-norm regularization term that lowers the MindSkip capacity and λ is the scale factor.

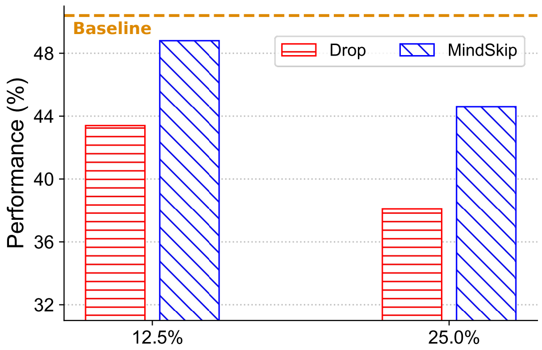
3 Experiments

3.1 Main Results

Competitive Performance of MindSkip We first compare the application of dynamic depth to different modules, e.g., Block, MLP, and Attention, with the implementation details in Appendix A. Based on the observation that deeper layers are more redundant than shallow layers (Gromov et al., 2024; He et al., 2024b), we focus on applying MindSkip to the deepest layers except the last one, leaving other layers unchanged. In Table 1, we transform the last half of the attention layers into MindSkip. While previous work primarily focused on applying dynamic depth to Block and MLP layers, this approach significantly degrades performance. In contrast, applying dynamic depth to Attention layers preserves nearly the same performance as the original models, e.g., 69.4% v.s. 69.7 in Llama-3-8B. These findings reinforce our motivation to focus on Attention with Dynamic Depth.

Comparison with Attention Drop Compared to statically dropping attention layers, MindSkip adapts dynamically to the input, which enhances its potential for improved performance. As

Figure 2: **Comparison with Attention Drop** under the same skipping ratios.



shown in Figure 2, we compare MindSkip with Attention Drop (He et al., 2024b) under the same computation budget — for instance, dropping 4 layers versus applying MindSkip to 8 layers with 50% capacity. When applying the same skipping ratios, MindSkip significantly outperforms Attention Drop (He et al., 2024b) on the GSM8K benchmark (Cobbe et al., 2021), e.g. 6.5% when the skipping ratio is 25.0%.

Training Efficiency The training efficiency of our method lies in two perspectives: trainable parameters and time consumption. Since the router projects the input from dimension d to 1, the number of trainable parameters is d per layer, and the total number of trainable parameters is fewer than 0.01% of the whole model. Furthermore, router-tuning on a single Nvidia A6000 GPU only takes less than 15 minutes, which is over 1000 times faster than DLO (Tan et al., 2024) that requires 36 hours on Nvidia RTX A100 GPUs.

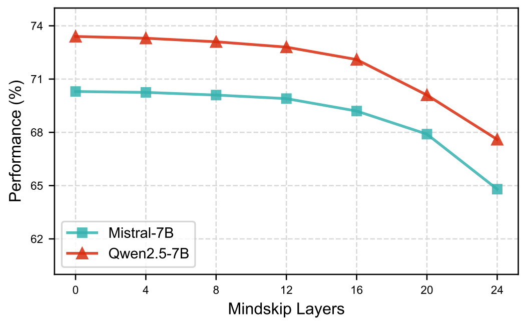
Inference Speedup We also evaluate the runtime speed improvements achieved with MindSkip. The inference speed is measured throughout the entire generation process, from the initial input prompt to the generation of the final token. To ensure that our results accurately reflect the performance gains, we adhere to two key principles: (1) all operations are performed on a single Nvidia RTX A6000 Ada GPU, eliminating any communication overhead from multi-GPU setups; and (2) we increase batch sizes to fully utilize the GPU for each model. As shown in Table 1, MindSkip achieves a 21% speedup in inference when applied to half of the layers.

KV Cache The KV cache stores intermediate representations of attention layers, which accelerates inference by preventing redundant computations but has a significant memory cost. Our approach significantly reduces the size of the KV cache—for example, an 8GB reduction when processing an input sequence of length 2048 with a batch size of 64 on Llama-3-8B. In contrast, DLO only applies to MLP layers and retains the full KV cache.

3.2 Ablation Study

Pretrained Models Since MoD-Attention can be seamlessly integrated into pretrained language models, we extend our evaluation to other mainstream models, specifically Mistral-7B and Qwen2.5-7B. In Figure 3, we experiment with various numbers of MindSkip layers, keeping the total MindSkip capacity at 50%. Our results show that applying MindSkip to half of the attention layers maintains model performance. However, when increasing the number of MindSkip layers, performance starts to degrade. We believe this decline occurs because important shallow layers are being transformed, negatively impacting overall performance (Men et al., 2024; He et al., 2024a,b).

Figure 3: Effectiveness across language models.



Training Dataset MindSkip requires a dataset to fine-tune the routers. In this section, we examine the impact of using different training datasets in Table 3. We consider a variety of datasets, including Alpaca (Taori et al., 2023), Evol-Instruct (Xu et al., 2023), ShareGPT (Zheng et al., 2023), and Llama-Pro (Wu et al., 2024). Since MindSkip only fine-tunes the routers while keeping the backbone of the language models intact, changes in the training dataset do not significantly impact performance. However, Llama-Pro, which incorporates diverse training data from various domains, provides slightly better performance due to its broader data coverage. On the other hand, due to the small number of trainable parameters, MindSkip does not require a large training dataset. As shown in Table 3, using just 5K training samples is sufficient to train the router effectively.

Table 2: Effectiveness across different training datatsets.

Dataset	HellaSwag	MMLU	OBQA	WinoGrande	Avg.
Baseline	82.1	65.3	45.0	77.7	<u>67.5</u>
Alpaca	79.8	62.2	43.8	77.4	<u>65.8</u>
Evol-Instruct	80.4	64.0	44.4	77.6	<u>66.6</u>
ShareGPT	80.6	63.3	**45.4**	76.7	<u>66.5</u>
Llama-Pro	**80.7**	**65.1**	44.6	**77.7**	<u>**67.0**</u>

Table 3: Ablation study on training samples.

Sample	HellaSwag	MMLU	OBQA	WinoGrande	Avg.
0	78.8	65.7	42.8	75.9	<u>65.8</u>
1K	74.3	64.5	41.6	74.6	<u>63.8</u>
2K	75.6	65.4	42.8	75.7	<u>64.9</u>
5K	**76.9**	65.7	**43.0**	**76.9**	<u>**65.6**</u>
10K	76.5	**65.8**	42.6	75.6	<u>65.1</u>

4 Conclusion

In this work, we explore the dynamic depth mechanism in both design and training. First, we propose MindSkip, which significantly enhances efficiency without compromising performance. Second, we introduce Router-Tuning which tunes a small number of parameters in just a few steps to implement dynamic depth. These improvements will provide valuable insights for the NLP community.

Limitations

Despite the progress we have made, our work still has limitations. First, while we have advanced MoD with Router-Tuning, other sophisticated training methods may exist that could further improve performance, warranting future exploration. Second, due to computational resource constraints, our experiments have been limited to Llama-3-8B, Mistral-7B, and Qwen2.5-7B on a small set of tasks. Expanding this approach to other models and a broader range of tasks would be highly valuable for understanding its full potential.

References

2019. Winogrande: An adversarial winograd schema challenge at scale.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *Preprint*, arXiv:1308.3432.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *Preprint*, arXiv:1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *Preprint*, arXiv:2403.17887.

Shwai He, Daize Dong, Liang Ding, and Ang Li. 2024a. Demystifying the compression of mixture-of-experts through a unified framework. *Preprint*, arXiv:2406.02500.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2024b. What matters in transformers? not all attention is needed. *Preprint*, arXiv:2406.15786.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *ArXiv*, abs/1902.00751.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. *Preprint*, arXiv:2306.00978.

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

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. *Preprint*, arXiv:1809.02789.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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. *Preprint*, arXiv:2404.02258.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. *Preprint*, arXiv:2306.11695.

Zhen Tan, Daize Dong, Xinyu Zhao, Jie Peng, Yu Cheng, and Tianlong Chen. 2024. Dlo: Dynamic layer operation for efficient vertical scaling of llms. *Preprint*, arXiv:2407.11030.

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

Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

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

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Appendix

Models We conduct experiments on Llama-3 (Touvron et al., 2023), Qwen (Bai et al., 2023), and Mistral (Jiang et al., 2023), given their competitive performance and wide usage.

Datasets For the training dataset, we used Llama-Pro (Wu et al., 2024), given it spanning general instruction, math, and code for the SFT process and offering a wealth of instruction data with varying complexity levels. To evaluate model performance, we report normalized zero-shot or few-shot accuracy on the LM-Harness benchmark. The number of shots for each task is detailed in Table 4, which includes multiple tasks: ARC-C (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2019), RTE (Wang et al., 2019), and WinoGrande (ai2, 2019). The evaluation code is based on EleutherAI's LM Harness framework (Gao et al., 2023).

Table 4: **Experimental settings for evaluation tasks.** "Norm" refers to the normalization performed with respect to the length of the input.

Task	Number of few-shot	Metric
BoolQ	0	Accuracy
RTE	0	Accuracy
OBQA	0	Accuracy (Norm)
PIQA	0	Accuracy (Norm)
MMLU	5	Accuracy
WinoGrande	5	Accuracy
GSM8K	5	Exact Match
HellaSwag	10	Accuracy (Norm)
ARC-C	25	Accuracy (Norm)

Hyperparameters We set τ as 0.5, which corresponds to the midpoint of the sigmoid function. To ensure that training starts from dense models, we initialize $\boldsymbol{W}$ to zero, ensuring that $\boldsymbol{R}(\boldsymbol{x}) \geq \tau$ initially, i.e., training from dense models. To achieve the desired MindSkip capacity, we perform a grid search over the learning rate from {1e-5, 2e-5, 5e-5, 1e-4, 2e-4} and the scale factor λ from {0, 0.1, 0.01, 0.001}, respectively.

To evaluate the performance of the model, we report the results of the following tasks: BoolQ (Clark et al., 2019), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2019), RTE (Wang et al., 2019), ARC-C (Clark et al., 2018), HellaSwag(Zellers et al., 2019), MMLU (Hendrycks et al., 2021), WinoGrande (ai2, 2019) and GSM8K (Cobbe et al., 2021). Please refer to Table 4 for detailed information. The evaluation code is based on EleutherAI LM Evaluation Harness (Gao et al., 2023).