

E³-Pruner: Towards Efficient, Economical, and Effective Layer Pruning for Large Language Models

Tao Yuan¹ Haoli Bai^{1*} Yinfei Pan¹ Xuyang Cao²
Tianyu Zhang¹ Lu Hou¹ Ting Hu¹ Xianzhi Yu¹

¹Huawei Technologies ²Tsinghua Shenzhen International Graduate School
{yuantao44, baihaoli}@huawei.com

Abstract

With the increasing size of large language models, layer pruning has gained increased attention as a hardware-friendly approach for model compression. However, existing layer pruning methods struggle to simultaneously address key practical deployment challenges, including performance degradation, high training costs, and limited acceleration. To overcome these limitations, we propose E³-PRUNER, a task-Effective, training-Economical and inference-Efficient layer pruning framework. E³-PRUNER introduces two key innovations: (1) a differentiable mask optimization method using a Gumbel-TopK sampler, enabling efficient and precise pruning mask search; and (2) an entropy-aware adaptive knowledge distillation strategy that enhances task performance. Extensive experiments over diverse model architectures and benchmarks demonstrate the superiority of our method over state-of-the-art approaches. Notably, E³-PRUNER achieves 96% accuracy, a mere 0.8% drop from the original model (96.8%) on MATH-500 when pruning 25% layers of Qwen3-32B, outperforming existing SOTA (95%), with a 1.33 $\times$ inference speedup by consuming merely 0.5B tokens (0.5% of the post-training data volume).

1 Introduction

In recent years, guided by scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), Large Language Models (LLMs) (Touvron et al., 2023; Grattafiori et al., 2024; Yang et al., 2025b,a; DeepSeek-AI et al., 2025b; Team et al., 2025a) have demonstrated substantial advances in performance, achieving unprecedented results across a wide range of tasks and signaling the advent of the artificial intelligence era. However, this performance gain has been accompanied by an exponential growth in model parameters up to trillions

*Corresponding author.

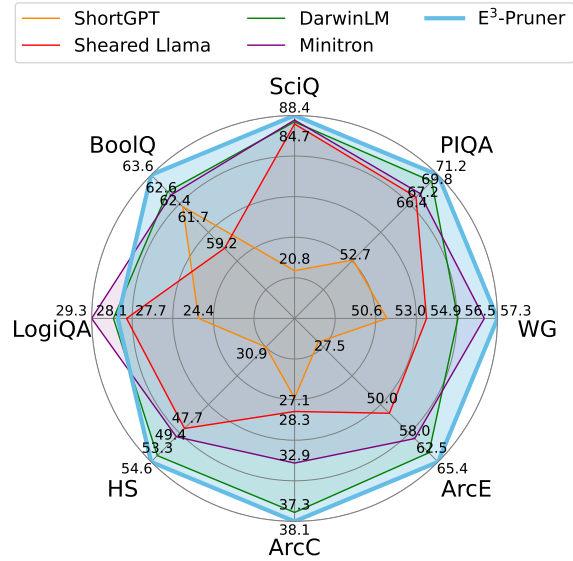


Figure 1: Comparisons between E³-PRUNER and current state-of-the-art structural pruning models in LLaMA-2-7B with 60% pruning ratio.

of parameters (Team et al., 2025b), which poses increasing challenges for the deployment. To obtain compact LLMs, various model compression methods have been explored, such as pruning (Han et al., 2015; Frantar and Alistarh, 2023; Sun et al., 2024; Chen et al., 2025b), quantization (Frantar et al., 2023; Lin et al., 2024; Xiao et al., 2023; Sun et al., 2025), and knowledge distillation (Hsieh et al., 2023; Hinton et al., 2015).

Among these methods, pruning effectively reduces model size by directly eliminating parameters. Currently, pruning approaches can be categorized primarily by granularity. While unstructured pruning (Frantar and Alistarh, 2023; Sun et al., 2024; Frankle and Carbin, 2019) minimizes accuracy loss through weight-level sparsity, its irregular patterns deliver limited practical acceleration. Structured pruning (Ma et al., 2023; Xia et al., 2024; Muralidharan et al., 2024; Tang et al., 2025) operates on components such as attention heads or

channels, making it more hardware-friendly. However, the irregularity in the resulting pruned architectures still limits achievable speedups. In contrast, layer pruning, which removes entire Transformer blocks, provides scalable inference acceleration. Nevertheless, current layer pruning methods face two critical hurdles: significant accuracy degradation and the prohibitive computational cost of mask searching and fine-tuning. These challenges necessitate a more refined strategy that balances performance, efficiency, and economy.

In this paper, we introduce E^3 -PRUNER, a task-Effective, training-Economical and inference-Efficient layer pruning approach. E^3 -PRUNER operates in two distinct phases: a *searching stage* in which we employ a differentiable Gumbel-TopK sampler to replace the traditional deterministic top- k operator. This enables gradient-based optimization, allowing for precise and efficient mask learning via backpropagation. And a *fine-tuning stage*, where an adaptive knowledge distillation strategy is employed, incorporating token-wise re-weighting to emphasize critical token representations and improve knowledge transfer from the teacher model. Extensive experiments on various architectures and benchmarks validate the superiority of E^3 -PRUNER. As shown in Figure 1, when pruning 60% of the layers from LLaMA-2-7B (Touvron et al., 2023), E^3 -PRUNER maintains remarkable higher accuracy. More remarkably, in the challenging scenario of pruning 25% of Qwen3-32B layers (Yang et al., 2025a), E^3 -PRUNER is the only approach capable of limiting performance degradation to less than 1% on the MATH-500 (Lightman et al., 2023) while requiring only 0.5% of original post-training data, demonstrating its exceptional ability to preserve reasoning abilities.

We summarize our contributions as follows:

- We identify practical challenges in layer pruning and propose E^3 -PRUNER, which establishes an efficient, economical and effective paradigm for layer pruning.
- We propose a differentiable Gumbel-TopK sampler for precise mask optimization, complemented by an adaptive KD strategy with token-wise re-weighting to ensure robust performance recovery.
- Through comprehensive evaluations and ablation studies, we demonstrate E^3 -PRUNER’s state-of-the-art performance and its efficiency across diverse LLM architectures, scaling ef-

fectively even to a massive 671B-parameter MoE model.

2 Preliminaries

2.1 LLM Pruning

Notations. We consider a standard Transformer architecture in modern LLMs. Let $f(\mathbf{X}_l; \theta_l)$ denote the l -th Transformer layer with parameters θ_l and input activations $\mathbf{X}_l$. Following the pre-norm convention, the input to layer $l + 1$ is defined as:

$$\mathbf{X}_{l+1} = \mathbf{X}_l + f(\mathbf{X}_l, \theta_l). \quad (1)$$

This work focuses on layer pruning, as it provides deterministic inference speedups proportional to the number of layers removed. When layer l is pruned, layer $l + 1$ directly bypasses it and takes the output $f(\mathbf{X}_{l-1}, \theta_{l-1})$ of the $(l - 1)$ -th layer as its own input, i.e.:

$$\mathbf{X}_{l+1} = \mathbf{X}_{l-1} + f(\mathbf{X}_{l-1}, \theta_{l-1}). \quad (2)$$

Prior Works. A key research in layer pruning is to determine which layer to discard. Prior studies have investigated a variety of methods, based on either different pruning metrics (Men et al., 2024; Chen et al., 2024) or the neural architecture search (Xia et al., 2024; Tang et al., 2025). Below we introduce three representative works and more related works can be found in Appendix A:

- **ShortGPT** (Men et al., 2024) is a training-free method that measures layer redundancy via the cosine similarity between activations on a calibration dataset $\mathcal{D}$. Layers with the highest similarity to their successors are prioritized for removal: $l^* = \arg \max_l \mathbb{E}_{i \in \mathcal{D}} \frac{\mathbf{X}_{i,l} \cdot \mathbf{X}_{i,l+1}}{\|\mathbf{X}_{i,l}\| \|\mathbf{X}_{i,l+1}\|}$.
- **DarwinLM** (Tang et al., 2025) employs evolutionary search to optimize layer depth, MLP widths, and attention heads. Each candidate architecture undergoes lightweight training to guide the selection of the next generation.
- **Sheared-LLaMA** (Xia et al., 2024) utilizes differentiable pruning via learnable soft masks and sparsity regularization. The final pruned structure is determined by a deterministic top- k selection of these masks.

2.2 Practical Challenges for Layer Pruning

Practical LLM pruning demands high inference speedup without sacrificing task accuracy, all

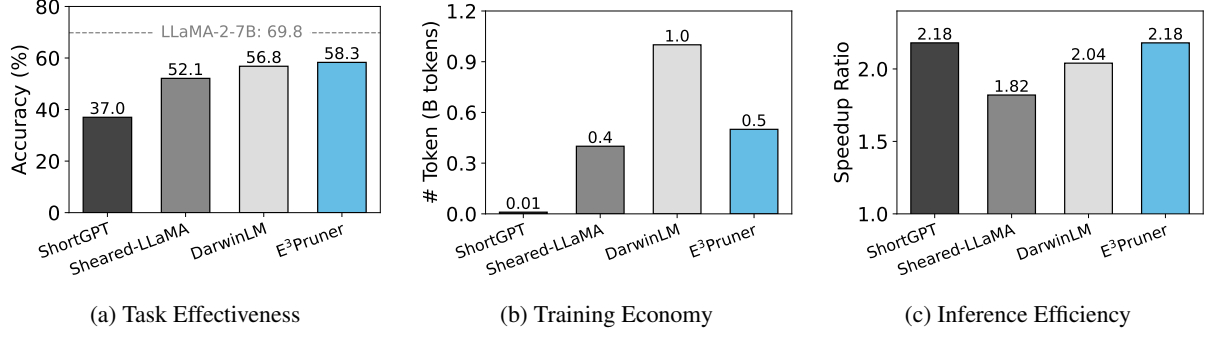


Figure 2: The comparisons of task effectiveness, training economy and inference efficiency among existing layer pruning methods. All experiments are done on LLaMA-2-7B model under 60% sparsity.

while operating under strict computational budgets. Based on these three dimensions, we compare the above pruning methods, and an overview is shown in Figure 2.

Task Effectiveness. The effectiveness of layer pruning (i.e., the task accuracy) is always of the highest priority in practice. As shown in Figure 2a, the training-free ShortGPT suffers from significant performance degradation, highlighting the necessity of post-pruning recovery. Furthermore, the performance gap among training-based methods underscores that identifying an accurate pruning mask is critical to preserving model capabilities.

Training Economy. Performance recovery must be computationally efficient. We use the number of training tokens as the metric for economy. As shown in Figure 2b, while ShortGPT requires zero training, DarwinLM’s evolutionary search is costly, consuming approximately 1B tokens. In contrast, differentiable search methods like Sheared-LLaMA and our E³-PRUNER require significantly fewer tokens (0.5B), demonstrating a more scalable path for practical deployment.

Inference Efficiency. Hardware-aligned acceleration is the ultimate goal of pruning. Figure 2c compares the wall-time speedup of different methods. Layer pruning (ShortGPT and E³-PRUNER) consistently outperforms uniform and non-uniform structured pruning (Sheared-LLaMA and DarwinLM), as removing entire blocks offers the most direct reduction in computational overhead.

In summary, While current state-of-the-art methods excel in specific areas, none simultaneously satisfy the requirements of high effectiveness, training economy, and inference efficiency. This gap motivates the design of E³-PRUNER.

3 Methods

3.1 Formulation

In this paper, we introduce E³-PRUNER, a task-Effective, training-Economical and inference-Efficient layer pruning approach for large language models. Inspired by neural architecture search, E³-PRUNER is empowered by an efficient differentiable mask optimization (Section 3.2) in joint effort with an enhanced variant of knowledge distillation (Section 3.3). An overview of E³-PRUNER is presented in Figure 3, and the algorithmic workflow is in Algorithm 1. Specifically, given the l -th transformer layer, we assign a pruning mask $m_l \in \{0, 1\}$ for the associated parameter θ_l . The forward computation of the l -th layer is thus $\mathbf{X}_{l+1} = \mathbf{X}_l + m_l \cdot f(\mathbf{X}_l, \theta_l)$.

Algorithm 1 E³-PRUNER

Require: LLM Θ , total iterations T , mask searching iterations T_M , layer scores $\mathbf{S}$, total layers L and pruning layers k .

- 1: **for** $t = 1$ to T **do**
 - 2: $\tau = 1 - \beta \cdot t/T, k' = L - \lceil (L - k) \cdot t/T \rceil$
 - 3: **if** $t \leq T_M$:
 - 4: $\mathbf{M} = \text{Gumbel-TopK}(\mathbf{S}, \tau, k')$
 - 5: $\mathbf{S} = \mathbf{S} - \lambda \cdot \partial \mathcal{L}(\mathbf{x}, \Theta, \mathbf{M}) / \partial \mathbf{S}$
 - 6: $\Theta = \Theta - \lambda \cdot \partial \mathcal{L}(\mathbf{x}, \Theta, \mathbf{M}) / \partial \Theta$
 - 7: **end for**
 - 8: **return** Pruned model Θ^*
-

The proposed E³-PRUNER consists of two stages: the *searching stage* and the *fine-tuning stage*. The former aims to identify the pruning masks, while the latter fine-tunes the pruned LLMs for further performance recovery. Specifically, in the *searching stage*, as pruning masks are inherently associated with the LLM param-

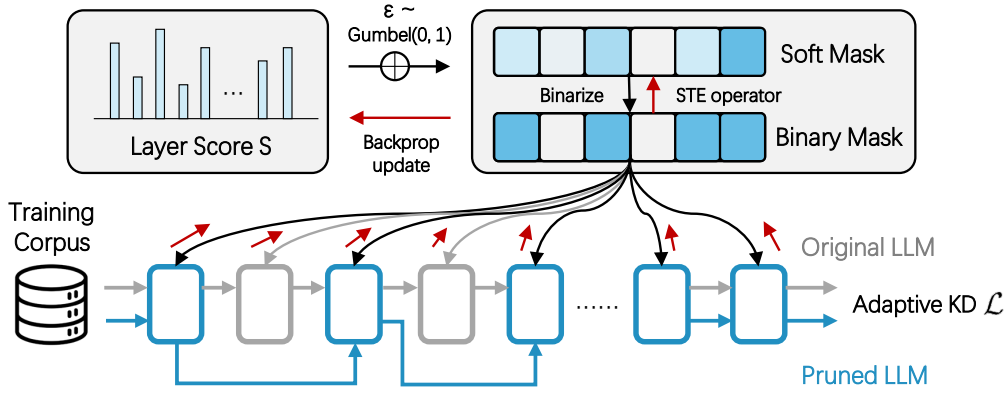


Figure 3: The framework of E³-PRUNER. We employ KL divergence to initialize proposed Gumbel-TopK sampler, which searches for the optimal mask in a differentiable way. Subsequently, the pruned model undergoes efficient adaptive knowledge distillation to restore its performance.

eters, E³-PRUNER optimizes both the model parameters $\Theta = \{\theta_1, \dots, \theta_L\}$ and the pruning masks $\mathbf{M} = \{m_1, \dots, m_L\}$ w.r.t. the training objective $\mathcal{L}$:

$$\{\Theta^*, \mathbf{M}^*\} = \min_{\Theta, \mathbf{M}: \|\mathbf{M}\|_0=k} \mathcal{L}(x; \Theta, \mathbf{M}), \quad (3)$$

where $x \in \mathcal{C}$ denote the training data from corpus $\mathcal{C}$, and k is the number of effective layers. In the *fine-tuning stage*, we keep the pruning mask $\mathbf{M}^*$ and continue to optimize model parameters Θ^* w.r.t. the objective $\mathcal{L}$. We propose adaptive knowledge distillation as the training objective, i.e., a variant of knowledge distillation (KD) that re-weights training tokens, as will be detailed in Section 3.3.

While the formulation in Equation 3 looks promising, the discrete nature of pruning masks $\mathbf{M}$ makes the optimization infeasible. To solve this, prior studies (Xia et al., 2024) adopt Gumbel-Softmax tricks with sparsity regularization (Louizos et al., 2017). However, these methods cannot exactly control the pruning rate, with additional training overhead up to 5 times slower.

3.2 Differentiable Mask Optimization with Gumbel-TopK Sampler

Instead of directly optimizing the pruning masks $\mathbf{M}$, we introduce layer importance $\mathbf{S} = \{s_1, \dots, s_L\}$ as auxiliary variables, where $s_l \in \mathbb{R}$ is the real value indicating the importance of the l -th layer. The pruning mask m_l can be thus obtained via the TopK selection, i.e., $m_l = 1$ if $s_l \in \text{TopK}(\mathbf{S}, k)$ and 0 otherwise. However, the discrete operator $\text{TopK}(\mathbf{S}, k)$ is non-differentiable, and we sort to Gumbel-TopK Sampler (Plötz and Roth, 2018) as a continuous relaxation.

Gumbel-TopK Sampler. The $\text{TopK}(\mathbf{S}, k)$ operator can be equivalently translated to repeatedly picking k largest elements from $\mathbf{S}$. In particular, the Gumbel-Softmax trick first sample Gumbel noise $g \sim \text{Gumbel}(0, 1)$, and add it over $\mathbf{S}$, i.e., $\mathbf{S}_1 = \mathbf{S} + g$. Then, for the i -th iteration ($i \geq 1$),

$$\begin{aligned} \tilde{\mathbf{M}}_{i+1} &= \tilde{\mathbf{M}}_i + \sigma(\mathbf{S}_i/\tau), \\ \mathbf{S}_{i+1} &= \mathbf{S}_i + \log(1 - \sigma(\mathbf{S}_i/\tau)), \end{aligned} \quad (4)$$

where $\tilde{\mathbf{M}}_i$ is the soft i -hot mask variable, $\sigma(\cdot/\tau)$ is the softmax function with temperature τ . (Plötz and Roth, 2018) has shown that as $\tau \rightarrow 0$, $\tilde{\mathbf{M}}_k \rightarrow \text{TopK}(\mathbf{S}, k)$. In practice, we initially choose a moderate temperature, and linearly anneal it with the training iterations t as $\tau = 1 - \beta \cdot (t/T)$, where β is a coefficient. To ensure pruning in the forward pass, we discretize the soft mask $\tilde{\mathbf{M}}_k$ as $\mathbf{M}$, and adopt the straight-through estimator (STE) (Bengio et al., 2013) for the backward pass to ensure the differentiability. The whole workflow of Gumbel-TopK sampler can be found in Algorithm 2. Notably, all operations—noise injection, softmax, and mask updates—introduce negligible computational overhead and are executed only once per forward pass, thus preserving training economy.

Initialization of Layer Importance. High-quality initialization of layer importance scores $\mathbf{S}$ is crucial for reducing training costs and preserving performance. We initialize the score s_l for the l -th layer as the Kullback-Leibler (KL) divergence between the original model Θ and the model Θ_{-l} with layer l removed, i.e., $\mathbb{E}_{x \sim \mathcal{C}} [D_{\text{KL}}(f(\mathbf{x}; \Theta_{-l}) \| f(\mathbf{x}; \Theta))]$.

Algorithm 2 Gumbel-TopK

Require: Layer score $\mathbf{S}$, temperature τ , number of retained layers k .

```
1:  $\mathbf{S}_1 = \mathbf{S} + \text{Gumbel}(0, 1)$ 
2:  $\tilde{\mathbf{M}}_1 = \mathbf{0}$ 
3: for  $i = 1$  to  $k$  do
4:    $\mathbf{S}_{i+1} = \mathbf{S}_i + \log(\mathbf{1} - \sigma(\frac{\mathbf{S}_i}{\tau}))$ 
5:    $\tilde{\mathbf{M}}_{i+1} = \tilde{\mathbf{M}}_i + \sigma(\frac{\mathbf{S}_i}{\tau})$ 
6: end for
7:  $\mathbf{M} = \text{STE}(\text{TopK}(\tilde{\mathbf{M}}_{k+1}, k))$ 
8: return Pruning mask  $\mathbf{M}$ 
```

Progressive Layer Pruning. To facilitate a smooth transition from the original to the pruned model, we adopt a curriculum schedule that gradually increases the pruning ratio (Frankle and Carbin, 2019). Specifically, the number of layers to retain, k' , is linearly decreased from the total count L to the target k over T iterations: $k' = L - \lceil (L - k) \cdot t/T \rceil$.

3.3 Adaptive Knowledge Distillation

Now we are ready to introduce adaptive knowledge distillation, a compelling training objective to enhance the task effectiveness of E³-PRUNER. The key idea behind this technique is to transfer the knowledge from the original LLM to the pruned model, where each token is re-weighted adaptively based on its entropy. A practical challenge in implementing knowledge distillation is the memory bottleneck, as it requires simultaneously hosting both the teacher and student model within the limited memory. Therefore, we pre-compute and save the top- K logits of the teacher model for each token. During training, these logits are loaded back to compute the KL divergence with the pruned model:

$$\mathcal{L}_{\text{token}} = \sum_{c \in \mathcal{S}_i^K} \sigma(\mathbf{z}_t^{(i)})_c \cdot \log(\sigma(\mathbf{z}_t^{(i)})_c / \sigma(\mathbf{z}_s^{(i)})_c),$$
$$\mathcal{L}_{\text{KL-TopK}} = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_{\text{token}}. \quad (5)$$

where N is the total number of tokens, $\mathbf{z}_t^{(i)}$ and $\mathbf{z}_s^{(i)}$ denote the pre-softmax logits of the teacher and student models for the i -th token, $\mathcal{S}_i^K$ contains the indices of the top- K values in the teacher’s distribution. By pre-storing teacher logits, we can bypasses the memory bottleneck of concurrent model loading; under a 32K sequence length, it incurs a negligible GPU memory overhead of only 2.6 MB. This

drastic reduction in hardware requirements makes the entire recovery training process significantly more economical and efficient.

Based on Equation 5, we argue that different tokens contribute unequally to model performance and should be treated accordingly. Existing works (Wang et al., 2025) show that token entropy $\mathcal{H}$ serves as an effective metric to identify these informative tokens. This finally leads to the objective of adaptive knowledge distillation in Equation 3:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \mathcal{H}(\sigma(\mathbf{z}_t^{(i)})_c) \cdot \mathcal{L}_{\text{token}}. \quad (6)$$

This formulation prioritizes high-entropy tokens, which typically signify uncertain or critical reasoning steps (Wang et al., 2025; Chen et al., 2025a).

4 Experiments

4.1 Setup

Models and Datasets. We evaluate E³-PRUNER across diverse architectures and scales to demonstrate its generalizability. Our primary analysis focuses on Qwen2.5-14B-Instruct (Yang et al., 2025b), Qwen3-32B (Yang et al., 2025a), and DeepSeek-R1 (DeepSeek-AI et al., 2025a). Additional results for LLaMA-2-7B (Touvron et al., 2023) and openPangu-Ultra-MoE (Ascend Tribe, 2025) are provided in Appendix B due to space constraints. For the training corpus, we utilize the open-source Fineweb-Edu (Lozhkov et al., 2024) dataset for Qwen2.5-14B-Instruct (and LLaMA-2-7B in the appendix). For the Qwen3-32B and DeepSeek-R1, we use the AM-DeepSeek-R1-Distilled-1.4M dataset (Zhao et al., 2025), which provides more reasoning content in the text.

Baselines. We perform systematic comparisons between E³-PRUNER and state-of-the-art pruning approaches: ShortGPT (Men et al., 2024), Sheared-LLaMA (Xia et al., 2024), DarwinLM (Tang et al., 2025), Minitron (Muralidharan et al., 2024), LLM-Streamline (Chen et al., 2024), Shortened LLaMA (Kim et al., 2024) and SLEB (Song et al., 2024). Where available, official model checkpoints are used to reproduce reported results. Otherwise, each baseline is carefully re-implemented by strictly following the pruning metrics reported in their original literature. To ensure a fair comparison, we adopt the single-iteration recipe for Minitron. All baselines are integrated into a unified evaluation framework with identical training data, distillation

Table 1: Comparison of Qwen2.5-14B-Instruct results with and without fine-tuning. E³-PRUNER achieves best in performance and speedup. *: We use the officially released checkpoints to reproduce the reported results. †: We use the single iteration recipe for Minitron, which is equivalent to ShortGPT with Adaptive KD.

Method	Param.	SciQ	PIQA	WG	ArcE	ArcC(25)	HS(10)	LogiQA	BoolQ	MMLU(5)	Avg ↑	Speedup ↑	# Token ↓
Original	14.8B	94.0	78.1	69.0	76.3	54.2	78.7	30.4	77.7	45.9	69.8	1.0×	-
<i>w/o KD fine-tuning</i>													
ShortGPT	8.2B	31.6	56.7	50.3	30.5	27.6	34.0	24.9	53.8	24.6	37.1	1.63×	-
Shortened LLaMA	8.2B	59.8	53.9	51.4	34.4	22.6	26.1	25.5	45.5	25.6	38.3	1.63×	-
SLEB	8.2B	75.0	64.2	51.5	50.9	24.9	42.1	25.5	58.4	24.6	46.3	1.63×	-
E ³ -PRUNER (stage 1)	8.2B	88.3	72.0	54.5	69.2	40.9	57.0	26.9	62.0	28.4	55.5	1.63×	-
<i>w/ KD fine-tuning</i>													
LLM-Streamline	8.2B	89.2	73.5	52.6	68.7	36.0	55.9	25.5	59.9	26.9	54.2	1.63×	0.5B
Sheared-LLaMA	8.2B	93.4	74.8	60.4	71.3	44.4	64.1	26.0	65.9	32.1	59.2	1.63×	0.5B
Shortened LLaMA	8.2B	91.0	75.6	59.6	74.7	47.5	64.8	27.3	60.8	31.6	59.2	1.63×	0.5B
DarwinLM*	8.4B	84.3	73.8	59.0	75.8	49.1	53.6	28.9	67.2	42.8	59.4	1.43×	1.0B
SLEB	8.2B	93.0	76.8	61.3	75.4	47.8	66.6	28.6	65.1	33.7	60.9	1.63×	0.5B
Minitron†	8.2B	91.9	73.5	62.0	72.1	44.9	64.5	28.4	70.4	32.8	60.1	1.63×	0.5B
E ³ -PRUNER (stage 2)	8.2B	93.7	76.9	63.0	76.0	47.9	67.2	30.0	66.5	36.5	61.9	1.63×	0.5B

Table 2: Results on Qwen3-32B and DeepSeek-R1 with KD fine-tuning. †: We test the results based on LiveCodeBench-v5, with questions spanning from 2024.08 to 2025.01.

Method	Param.	MATH-500	AIME'24	AIME'25	GPQA-Diamond	LiveCode-Bench†	MMLU-Pro	ArenaHard	Avg ↑	Speedup ↑	# Token ↓
<i>Qwen3-32B</i>											
Original	32.8B	96.8	79.6	71.3	64.7	66.9	76.5	93.5	78.5	1.0×	
Sheared-LLaMA	28.9B	93.2	60.0	55.4	52.0	56.3	65.1	78.6	65.8		
Minitron	28.9B	95.4	72.1	64.2	61.6	62.9	74.8	92.3	74.8	1.13×	0.5B
E ³ -PRUNER	28.9B	96.2	76.3	65.8	61.1	65.1	74.5	91.4	75.8		
Sheared-LLaMA	25.0B	92.4	58.8	49.6	40.9	36.0	55.8	67.4	57.3		
Minitron	25.0B	95.0	68.3	61.7	59.1	57.7	70.5	85.0	71.0	1.33×	0.5B
E ³ -PRUNER	25.0B	96.0	71.7	64.2	60.1	60.3	70.1	87.3	72.8		
<i>DeepSeek-R1</i>											
Original	671B	97.2	77.9	67.1	73.7	65.4	82.8	96.3	80.1	1.0×	
Sheared-LLaMA	601B	96.6	71.7	57.5	66.7	55.5	80.0	93.0	74.4		
Minitron	601B	97.0	75.0	62.9	71.2	65.1	81.8	95.0	78.3	1.13×	0.5B
E ³ -PRUNER	601B	97.0	76.3	60.8	73.2	67.3	81.2	96.2	78.9		
Sheared-LLaMA	509B	92.6	59.6	42.1	61.1	49.3	74.4	84.1	66.2		
Minitron	509B	95.6	68.8	55.4	69.2	59.2	80.2	89.8	74.0	1.33×	0.5B
E ³ -PRUNER	509B	95.2	65.8	60.0	69.7	59.9	79.8	90.0	74.3		

methods, pruning ratios and a fixed budget of 0.5B tokens. The gradual mask search occupies the first 10% of the budget and then we switch to the adaptive knowledge distillation to preserve task effectiveness. The process is highly efficient: recovery training for Qwen2.5-14B-Instruct completes in 1.5 hours using 64 NPUs, while the memory-intensive DeepSeek-R1 is finalized within 24 hours using LoRA (Hu et al., 2021). Complete pruning and hyperparameters specifications are in Appendix C.

Evaluation Details. For Qwen2.5-14B-Instruct, we follow established evaluation protocols by con-

ducting zero-shot assessments on SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2020), Wino-Grande (Sakaguchi et al., 2021), ARC-easy (Clark et al., 2018), LogiQA (Liu et al., 2020) and BoolQ (Clark et al., 2019), along with few-shot evaluations on HellaSwag (Zellers et al., 2019) (10-shot), ARC Challenge (Clark et al., 2018) (25-shot) and MMLU (Hendrycks et al., 2021) (5-shot), utilizing the standardized lm-evaluation-harness framework (Gao et al., 2024)¹ to ensure consistency. For rea-

¹<https://github.com/EleutherAI/lm-evaluation-harness/releases/tag/v0.4.8>

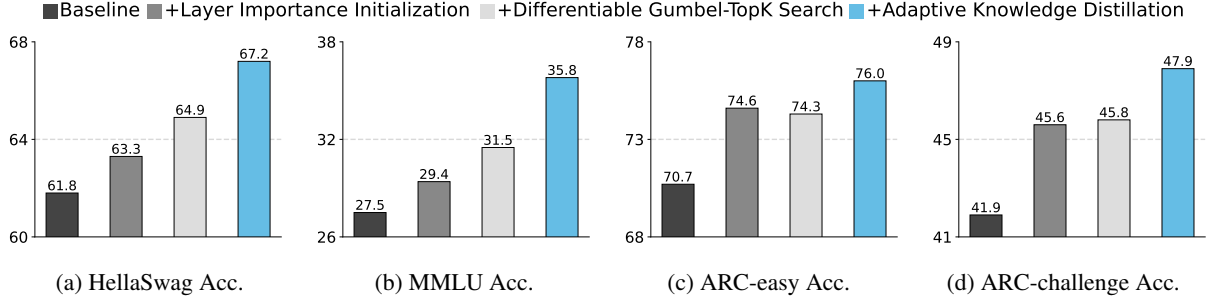


Figure 4: Performance improvement breakdown. By applying proposed E^3 -PRUNER, we achieve significant performance improvements across diverse benchmarks on Qwen2.5-14B-Instruct.

Table 3: Ablation of adaptive KD. Compared to normal KD, adaptive KD effectively improve the performance.

KD Type	SciQ	PIQA	WG	ArcE	ArcC(25)	HS(10)	LogiQA	BoolQ	MMLU(5)	Avg ↑
Offline KD	94.0	76.2	62.5	75.2	48.0	65.6	29.0	63.8	32.8	60.8
Adaptive KD	93.7	76.9	63.0	76.0	47.9	67.2	30.0	66.5	35.8	61.9

soning models, we evaluate models on complex reasoning tasks including MATH-500 (Lightman et al., 2023), AIME-2024/2025, GPQA-Diamond (Rein et al., 2023), LiveCodeBench (Jain et al., 2024) and a 2000-question subset of MMLU-Pro (Wang et al., 2024) for efficient yet comprehensive assessment. Given the limited size of AIME benchmarks, we conduct 8 independent evaluation runs and report mean accuracy to reduce measurement variance.

4.2 Main results

Results on Qwen2.5-14B-Instruct. We prune 24 of 48 layers for Qwen2.5-14B-Instruct and report the accuracy on downstream tasks in Table 1. Our approach consistently outperforms all baseline methods, both with and without KD fine-tuning. In the initial stage without KD, E^3 -PRUNER achieves a remarkable average accuracy of 55.5%, significantly surpassing other pruning strategies such as ShortGPT and SLEB. The integration of KD further elevates performance to a peak of 61.9%, solidifying a new state-of-the-art by surpassing existing frameworks such as Minitron and DarwinLM. Beyond task accuracy, E^3 -PRUNER also delivers a consistent $1.63\times$ speedup. These results demonstrate that E^3 -PRUNER effectively preserves the capabilities of the original model while offering a superior Pareto-optimal trade-off between computational efficiency and task accuracy.

Results on Qwen3-32B. To evaluate E^3 -PRUNER on reasoning-focused models, we conduct experiments with Qwen3-32B, as sum-

marized in Table 2. When 1/8 of the layers are removed, E^3 -PRUNER achieves an average accuracy of 75.8%, surpassing Minitron (74.8%) and Sheared-LLaMA (65.8%). This advantage persists under more aggressive pruning (1/4 layers removed), where our method maintains an accuracy of 72.8%, compared to 71.0% for Minitron and 57.3% for Sheared-LLaMA. Notably, on MATH-500, E^3 -PRUNER exhibits a degradation of less than 1% even at the 1/4 pruning ratio, underscoring its robustness and practical viability for scenarios demanding both inference efficiency and mathematical reasoning capability.

Results on DeepSeek-R1. We further evaluate the scalability of E^3 -PRUNER on DeepSeek-R1, a 671B mixture-of-experts model. As shown in Table 2, E^3 -PRUNER outperforms baselines in both 8-layer and 16-layer pruning configurations. When pruning 8 layers, our approach achieves an overall accuracy of 78.9%, exceeding Minitron (78.3%) and Sheared-LLaMA (74.4%), while maintaining near-original performance on most tasks. Under more aggressive pruning with 16 layers removed, the resulting model retains an accuracy of 74.3%, demonstrating the robustness of our method in large-scale model compression with only marginal performance degradation. These results collectively validate the efficacy of E^3 -PRUNER in effectively compressing large-scale MoE models.

4.3 Analysis

Ablation Study. We provide a detailed breakdown of the performance gains across multiple

Table 4: Ablation of searching budget. E³-PRUNER consistently outperform state-of-the-art methods across various mask search budget on Qwen2.5-14B-Instruct, demonstrating its robustness.

Searching Budget	SciQ	PIQA	WG	ArcE	ArcC(25)	HS(10)	LogiQA	BoolQ	MMLU(5)	Avg ↑
0.05	92.0	76.6	62.8	75.6	48.6	67.0	28.0	66.5	29.0	60.7
0.1	93.7	76.9	63.0	76.0	47.9	67.2	30.0	66.5	35.8	61.9
0.2	91.9	75.8	59.9	75.9	49.7	67.8	28.6	66.4	37.1	61.5
0.5	91.0	75.9	59.7	73.7	48.3	65.3	28.3	67.0	34.3	60.4

Table 5: Robustness of annealing temperature schedules. E³-PRUNER is insensitive to the specific annealing trajectory (linear, exponential, or cosine), while maintaining superior performance over state-of-the-art methods.

Annealing Schedule	SciQ	PIQA	WG	ArcE	ArcC(25)	HS(10)	LogiQA	BoolQ	MMLU(5)	Avg ↑
Linear Schedule	93.7	76.9	63.0	76.0	47.9	67.2	30.0	66.5	36.5	61.9
Exponential Schedule	93.7	75.8	62.5	74.6	48.4	65.5	27.8	66.7	36.9	61.3
Cosine Schedule	93.8	76.0	62.8	75.1	48.3	67.0	29.2	69.5	37.6	62.1

benchmarks to confirm the effectiveness of each component. As shown in Figure 4, each integrated module contributes to a consistent upward trend in accuracy across all evaluated benchmarks. Specifically, our initialization strategy and Gumbel-TopK search facilitate the generation of superior pruning masks, resulting in steady performance improvements. Furthermore, the introduction of Adaptive Knowledge Distillation yields a significant performance boost across all downstream tasks. More detailed analysis of the training losses and the optimization dynamics of Gumbel scores can be found in Appendix D and Appendix F.

Effect of Adaptive Knowledge Distillation. To demonstrate the superiority of our proposed adaptive KD loss over the conventional KD loss, we conduct a comparative experiment on Qwen2.5-14B-Instruct. We prune 50% layers and keep all other hyperparameters unchanged. Results in Table 3 show that while E³-PRUNER with conventional KD loss already achieves competitive performance, adopting the adaptive KD loss further improves the accuracy from 60.8% to 61.9%, demonstrating its effectiveness in identifying and emphasizing the critical tokens that contribute more significantly. Furthermore, our empirical analysis reveals that KD recovery training yields superior model consistency compared to standard fine-tuning. Detailed comparative results are provided in Appendix E.

The Balance between Searching and Fine-tuning. We further examine the influence of the searching budget, the fraction of total training steps allocated to the *searching stage*, on the perfor-

mance of our method. As shown in Table 4, E³-PRUNER consistently outperforms existing state-of-the-art approaches across various configurations. This indicates that our method exhibits strong robustness with respect to the searching budget. In all other experiments in this paper, a searching budget of 10% of the total training steps is adopted.

Robustness of the Annealing Schedule. In the *searching stage*, we employ a linear annealing schedule for the Gumbel-TopK temperature (τ). To evaluate the sensitivity of our method to the temperature trajectory, we conducted an ablation study on Qwen2.5-14B-Instruct, comparing the default linear schedule against exponential and cosine decay alternatives. Experimental results in Table 5 indicate that our differentiable Gumbel-TopK search is highly robust to the choice of annealing schedule, with all variants consistently outperforming existing pruning baselines. Given its computational simplicity and stable performance, we adopt the linear schedule as the default configuration.

Effect of Different Token Budgets. We additionally report an extended-budget experiment on DeepSeek-R1 below, with 8 layers pruned, and include the unpruned DeepSeek-R1 as an upper bound (original) reference. As shown in Table 6, even at 0.5B our pruned model is already within 1.2 avg points of the upper bound; increasing the budget yields further gains, but the improvements show a clear saturation trend toward this ceiling. We therefore keep 0.5B as the default setting in our main experiments to preserve training economy.

Table 6: Comparison of different token budget in DeepSeek-R1. Extended token budget leads to higher accuracy but shows a clear saturation trend toward ceiling.

Token Budget	MATH500	AIME'24	AIME'25	GPQADiamond	LiveCodeBench	MMLUPro	ArenaHard	Avg $\uparrow$
0.5B	97.0	76.3	60.8	73.2	67.3	81.2	96.2	78.9
1.5B	96.8	77.1	64.2	73.7	66.5	81.5	96.3	79.4
Upper bound (original)	97.2	77.9	67.1	73.7	65.4	82.8	96.3	80.1

Table 7: DeepSeek-R1 speedup results. Our 8-layer pruned model delivers a consistent $1.13\times$ speedup, demonstrating practical deployability.

BS	Prefill TTFT (ms)		Decode TPOT(ms)	
	Orig / Prun	Speedup	Orig / Prun	Speedup
1	7014 / 6187	$1.13\times$	70.6 / 62.0	$1.14\times$
2	10488 / 9231	$1.14\times$	69.7 / 62.0	$1.12\times$
8	12463 / 10858	$1.15\times$	71.0 / 62.9	$1.13\times$
64	13156 / 11451	$1.15\times$	73.2 / 63.8	$1.15\times$
128	18973 / 16459	$1.15\times$	74.4 / 66.2	$1.12\times$

Speedup Results on DeepSeek-R1. We evaluate the speedup ratio achieved by our layer-pruned DeepSeek-R1 model using the vLLM (Kwon et al., 2023) framework in a realistic 64-card deployment environment. As summarized in Table 7, under the quantization of w8a8 and with input and output lengths configured to 2K and 1K tokens, respectively, the 8-layer pruned model consistently demonstrates a $1.13\times$ speedup in both Time to First Token (TTFT) and Time per Output Token (TPOT) across a range of batch sizes. These results affirm the practical viability and deployment efficiency of the proposed method.

Generalizability to Iterative Schemes In main experiments, we primarily adopt a single-stage pruning-then-distillation strategy, prioritized for its suitability for practical deployment. However, we emphasize that the proposed E^3 -PRUNER framework is inherently extensible to iterative refinement protocols (e.g., the pruning-distillation-pruning cycles utilized in Minitron) whenever the computational budget permits. To evaluate the compatibility of our method with multi-stage schedules, we provide a comprehensive comparison between E^3 -PRUNER and Minitron across 1, 2, and 4 iterations. As summarized in Table 8, E^3 -PRUNER consistently outperforms Minitron in all corresponding settings. Notably, even our one-shot (1-iteration) variant achieves superior average performance compared to Minitron’s 4-iteration counterpart. This

significant margin underscores the efficacy of our differentiable Gumbel-TopK sampler in identifying high-quality pruning masks, which reduces the reliance on costly iterative retraining.

Table 8: Comparison of E^3 -PRUNER and Minitron across 1, 2, and 4 iterations. Even in a one-shot setting (1 iter), our method surpasses the 4-iteration Minitron baseline, demonstrating superior mask selection.

Metric (Acc %)	Minitron			E^3 -PRUNER		
	1-iter	2-iter	4-iter	1-iter	2-iter	4-iter
SciQ	91.9	93.2	93.6	93.7	94.1	93.4
PIQA	73.5	76.2	74.8	76.9	75.5	75.5
WGA	62.0	63.7	62.6	63.0	62.9	62.7
Arc-E	72.1	73.9	75.3	76.0	75.0	74.9
Arc-C	44.9	45.7	46.5	47.9	47.5	51.1
HS	64.5	65.4	66.1	67.2	66.0	66.7
LogiQA	28.4	30.1	29.0	30.0	30.0	30.3
BoolQ	70.4	65.2	67.2	66.5	68.1	69.1
MMLU	32.8	35.1	37.7	35.8	40.0	40.1
AVG	60.1	60.9	61.4	61.9	62.1	62.6

5 Conclusion

In this paper, we first review existing pruning methods and observe that they struggle to simultaneously fulfill three critical requirements for practical deployment: task effectiveness, training economy, and inference efficiency. To overcome this limitation, we propose E^3 -PRUNER, an effective, economical, and efficient layer pruning approach. E^3 -PRUNER incorporates a differentiable Gumbel-TopK sampling mechanism to enable efficient and precise optimization of layer masks, along with an adaptive knowledge distillation strategy to improve performance and maintain consistency. Extensive experiments conducted on diverse model architectures and benchmarks demonstrate the superiority of our approach. The results not only confirm the efficacy of our method, but also show that it achieves the best trade-off to date across the challenges for practical deployment.

6 Limitations

Despite its effectiveness, E³-PRUNER has limitations that offer avenues for future research. First, our study focuses on layer-level pruning. While the Gumbel-TopK mechanism proves robust here, its generalizability to other pruning paradigms, such as width pruning in LLMs or expert pruning in Mixture-of-Experts (MoE) architectures, remains to be explored. These granularities may introduce distinct optimization challenges regarding stability and sparsity patterns. Second, our recovery stage relies on standard, off-the-shelf pre-training or SFT datasets. We have yet to investigate the potential of model-generated (synthetic) data, which might provide better performance due to higher distributional alignment with the model’s internal representations. We leave the exploration of self-generated data for performance recovery to future work. Third, despite that we have conducted a thorough evaluation across a diverse set of benchmarks to ensure generalization, further analysis on multilingual and low-resource benchmarks remains unexplored.

References

- Ascend Tribe. 2025. openpangu-ultra-moe-718b-model. <https://ai.gitcode.com/ascend-tribe/openpangu-ultra-moe-718b-model>.
- Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. *SliceGPT: Compress large language models by deleting rows and columns*. *Preprint*, arXiv:2401.15024.
- 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. 2020. *Piqa: Reasoning about physical commonsense in natural language*. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.
- Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2024. *Flextron: Many-in-one flexible large language model*. *Preprint*, arXiv:2406.10260.
- Jierun Chen, Tiezheng Yu, Haoli Bai, Lewei Yao, Jiannan Wu, Kaican Li, Fei Mi, Chaofan Tao, Lei Zhu, Manyi Zhang, and 1 others. 2025a. The synergy dilemma of long-cot sft and rl: Investigating post-training techniques for reasoning vlms. *arXiv preprint arXiv:2507.07562*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. *Evaluating large language models trained on code*. *Preprint*, arXiv:2107.03374.
- Xiaodong Chen, Yuxuan Hu, and Jing Zhang. 2024. Compressing large language models by streamlining the unimportant layer. *arXiv preprint arXiv:2403.19135*.
- Xinrui Chen, Haoli Bai, Tao Yuan, Ruikang Liu, Kang Zhao, Xianzhi Yu, Lu Hou, Tian Guan, Yonghong He, and Chun Yuan. 2025b. *A simple linear patch revives layer-pruned large language models*. *Preprint*, arXiv:2505.24680.
- 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*.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shiro Ma, Peiyi Wang, and 1 others. 2025a. *Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning*. *Preprint*, arXiv:2501.12948.
- DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, and 1 others. 2025b. *Deepseek-v3 technical report*. *Preprint*, arXiv:2412.19437.
- Jonathan Frankle and Michael Carbin. 2019. *The lottery ticket hypothesis: Finding sparse, trainable neural networks*. *Preprint*, arXiv:1803.03635.
- Elias Frantar and Dan Alistarh. 2023. *SparseGPT: Massive language models can be accurately pruned in one-shot*. *Preprint*, arXiv:2301.00774.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. *GPTQ: Accurate post-training quantization for generative pre-trained transformers*. *Preprint*, arXiv:2210.17323.
- 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, and 5 others. 2024. [The language model evaluation harness](#).
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Yuxian Gu, Qinghao Hu, Shang Yang, Haocheng Xi, Junyu Chen, Song Han, and Han Cai. 2025. [Jet-nemotron: Efficient language model with post neural architecture search](#). *Preprint*, arXiv:2508.15884.
- Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). *Preprint*, arXiv:2009.03300.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#). *Preprint*, arXiv:1503.02531.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. [Training compute-optimal large language models](#). *Preprint*, arXiv:2203.15556.
- Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. [Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes](#). *Preprint*, arXiv:2305.02301.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. [Lora: Low-rank adaptation of large language models](#). *Preprint*, arXiv:2106.09685.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. [Live-codebench: Holistic and contamination free evaluation of large language models for code](#). *Preprint*, arXiv:2403.07974.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *Preprint*, arXiv:2001.08361.
- Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. [Shortened llama: A simple depth pruning for large language models](#). *arXiv preprint arXiv:2402.02834*, 11.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. [Let’s verify step by step](#). *Preprint*, arXiv:2305.20050.
- 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 on-device llm compression and acceleration](#). In *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100.
- Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. [Logiqa: A challenge dataset for machine reading comprehension with logical reasoning](#). *Preprint*, arXiv:2007.08124.
- Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.
- Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. [Fineweb-edu: the finest collection of educational content](#).
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. [Llm-pruner: On the structural pruning of large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc.
- Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>.
- 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.
- Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. [Compact language models via pruning and knowledge distillation](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 41076–41102. Curran Associates, Inc.

- Tobias Plötz and Stefan Roth. 2018. [Neural nearest neighbors networks](#). In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. [Gpqa: A graduate-level google-proof q&a benchmark](#). *Preprint*, arXiv:2311.12022.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. [Winogrande: an adversarial winograd schema challenge at scale](#). *Commun. ACM*, 64(9):99–106.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.
- Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. [Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks](#). *arXiv preprint arXiv:2402.09025*.
- 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.
- Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. 2025. [Flatquant: Flatness matters for llm quantization](#). *Preprint*, arXiv:2410.09426.
- Shengkun Tang, Oliver Sieberling, Eldar Kurtic, Zhiqiang Shen, and Dan Alistarh. 2025. [Darwinlm: Evolutionary structured pruning of large language models](#). *Preprint*, arXiv:2502.07780.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, and 1 others. 2025a. [Gemini: A family of highly capable multimodal models](#). *Preprint*, arXiv:2312.11805.
- Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025b. [Kimi k2: Open agentic intelligence](#). *Preprint*, arXiv:2507.20534.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, and 1 others. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. [Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning](#). *Preprint*, arXiv:2506.01939.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. 2024. [Mmlu-pro: A more robust and challenging multi-task language understanding benchmark](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290. Curran Associates, Inc.
- Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. [Crowdsourcing multiple choice science questions](#). *Preprint*, arXiv:1707.06209.
- Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. [Sheared llama: Accelerating language model pre-training via structured pruning](#). *Preprint*, arXiv:2310.06694.
- Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. [SmoothQuant: Accurate and efficient post-training quantization for large language models](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2025b. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [Hellaswag: Can a machine really finish your sentence?](#) *Preprint*, arXiv:1905.07830.
- Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xiangang Li. 2025. [1.4 million open-source distilled reasoning dataset to empower large language model training](#). *Preprint*, arXiv:2503.19633.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. [Instruction-following evaluation for large language models](#). *Preprint*, arXiv:2311.07911.

A Related Work

Structured Pruning. Structured pruning methods aim to reduce model size by removing entire substructures—such as attention heads, MLP channels, or hidden dimensions—directly from the network. Prior research has proposed various criteria for identifying the least important components. For instance, LLM-Pruner (Ma et al., 2023) identifies coupled structural groups and evaluates their collective importance to guide pruning decisions. SliceGPT (Ashkboos et al., 2024) applies principal component analysis to weight matrices and removes the least significant components, thereby reducing the dimensionality of the weight matrices. Although these methods are computationally lightweight, they often lead to non-negligible performance degradation. More recent approaches, such as Sheared-LLaMA (Xia et al., 2024), achieve state-of-the-art results by incorporating differentiable mask search and dynamic batching during retraining to maintain performance across diverse tasks. Meanwhile, DarwinLM (Tang et al., 2025) and Minitron (Muralidharan et al., 2024) adopt evolutionary search strategies to identify optimal pruning masks, followed by recovery training. However, these techniques typically require iterative training and evaluation of candidate architectures, which incur high computational costs and hinder practical deployment.

Layer Pruning. Layer pruning has recently gained attention as an effective strategy for compressing large language models. In contrast to width pruning, which may result in irregular and hardware-unfriendly structures, layer pruning removes entire Transformer layers—including both attention and feed-forward modules—thereby preserving the regularity of the model and facilitating efficient deployment. Recent studies, including ShortGPT (Men et al., 2024), SLEB (Song et al., 2024), Shortened LLaMA (Kim et al., 2024), and LinearPatch (Chen et al., 2025b), have demonstrated the viability of layer pruning in significantly reducing model depth while maintaining competitive performance. Despite achieving notable inference speedups, existing calibration-based layer pruning methods still struggle to fully preserve the original model’s capabilities, which remains a major obstacle to their widespread adoption in deployment.

Neural Architecture Search. Neural Architecture Search (NAS) automates the design of network architectures by searching for optimal sub-networks within a predefined super-net. Typical NAS frameworks focus on optimizing sampling strategies to identify high-performing and efficient architectures. For example, Flextron (Cai et al., 2024) introduces an integrated super-net that encapsulates multiple sub-networks, enabling flexible and adaptive deployment. Similarly, Jet-Nemotron (Gu et al., 2025) utilizes a PostNAS pipeline to design efficient linear attention blocks. These NAS-based approaches primarily target efficient model design and are orthogonal to our work, which focuses on developing a effective, economical, and efficient pruning strategy tailored for real-world deployment scenarios.

B Additional Results

Results on LLaMA-2-7B. As shown in Table 9, E³-PRUNER achieves superior performance on most downstream tasks after removing 60% of the layers of LLaMA-2-7B, attaining an average accuracy of 58.3%—surpassing both Minitron (55.3%) and DarwinLM (56.8%). Although Sheared-LLaMA also employs learnable masks, E³-PRUNER exhibits significantly better results, attributable to its unique equivalence-preserving property between mask search and pruning. Moreover, E³-PRUNER delivers a highest speedup of 2.18× with a modest token budget of 0.5B, highlighting its efficiency inference and economy in training.

Results on openPangu-Ultra-MoE. We evaluate E³-PRUNER on recently released openPangu-Ultra-MoE-718B (Ascend Tribe, 2025), a large-scale mixture-of-experts language model that incorporates depth-scaled Sandwich-Norm and EP-group load balancing. During both the mask search and recovery phases, we adopt a LoRA fine-tuning paradigm with a rank of 64 applied to all linear projection weights. The model is trained on 1.5 billion tokens sampled from the AM-DeepSeek-R1-Distilled-1.4M corpus (Zhao et al., 2025). Training is conducted over 6,000 iterations using a cosine learning rate scheduler that decays from 1e-4 to 1e-5. For E³-PRUNER, we prune a total of 8 layers, comprising 6 MoE layers and 2 dense layers. In the case of Minitron (Muralidharan et al., 2024), only 6 MoE layers are pruned, as the dense layers exhibit higher importance scores. The results are

Table 9: Results on LLaMA-2-7B. E³-PRUNER achieves best in performance and speedup. We omit the MMLU results for LLaMA-2-7B as they are close to random for all methods. *: We use the officially released checkpoints to reproduce the reported results.

Method	Param.	SciQ	PIQA	WG	ArcE	ArcC(25)	HS(10)	LogiQA	BoolQ	Avg ↑	Speedup ↑	# Token ↓
Original	6.7B	94.0	78.1	69.0	76.3	54.2	78.7	30.4	77.7	69.8	1.0×	
ShortGPT*	2.7B	20.8	52.7	50.6	27.5	27.1	30.9	24.4	61.7	37.0	2.18×	-
Sheared-LLaMA*	2.7B	84.7	66.4	53.0	50.0	28.3	47.7	27.7	59.2	52.1	1.82×	0.5B
DarwinLM*	2.7B	85.6	69.8	54.9	62.5	37.3	53.3	28.3	62.6	56.8	2.04×	1.0B
Minitron	2.7B	86.4	67.2	56.5	58.0	32.9	49.4	29.3	62.4	55.3	2.18×	0.5B
E ³ -PRUNER	2.7B	88.4	71.2	57.3	65.4	38.1	54.6	28.1	63.6	58.3	2.18×	0.5B

Table 10: Results on openPangu-Ultra-MoE-718B. †: We test the results based on LiveCodeBench-v5, with questions spanning from 2024.08 to 2025.01.

Method	Param.	MATH-500	AIME'24	AIME'25	GPQA-Diamond	LiveCode-Bench†	MMLU-Pro	ArenaHard	Avg ↑	Speedup ↑	# Token ↓
Dense	718B	97.8	82.5	76.7	80.8	69.5	80.2	97.5	83.6	1.0×	
Minitron	644B	96.6	74.6	64.6	73.2	62.5	80.1	93.7	77.9	1.13×	1.5B
E ³ -PRUNER	641B	96.8	81.3	72.9	78.8	68.8	80.5	96.2	82.2	1.13×	1.5B

summarized in Table 10. Although E³-PRUNER prunes two additional layers compared to Minitron, it achieves significantly better performance, with an average performance degradation of only 1.4%, in contrast to the 5.7% performance loss observed with Minitron. These findings further validate the superiority of the proposed E³-PRUNER.

C Implementation Details

We employ distinct pruning strategies that are tailored to different model architectures. For LLaMA-2-7B and Qwen2.5-14B-Instruct, we adopt target pruning ratios of 60% and 50%, respectively, consistent with established practices in prior research. For the reasoning models Qwen3-32B and DeepSeek-R1, we implement layer removal configurations of 12.5% / 25% layers based on empirical performance thresholds. The pruning process begins with a layer importance estimation using 40 calibration samples, followed by a Gumbel-TopK mask searching and subsequent recovery training using 0.5B tokens. The gradual mask search occupies the first 10% of the training budget. For the remaining stages, we switch to the adaptive knowledge distillation to preserve task effectiveness. In a 64 NPUs setting, the pruning for LLaMA-2-7B can be done in 0.5 hours, while it takes about 1.5 hours for Qwen2.5-14B-Instruct. For the memory-intensive DeepSeek-R1, we employ LoRA (Hu et al., 2021) fine-tuning. Specifi-

cally, we extend the PEFT (Mangrulkar et al., 2022) library to support LoRA adaptation of DeepSeek-R1’s MoE architecture and integrate it with Megatron’s (Shoeybi et al., 2019) tensor parallelism (TP) and expert parallelism (EP). This integration enables full-parameter LoRA fine-tuning across all model components while staying within memory limits. With a LoRA rank of 64 and TP=8/EP=8, the trainable parameters amount to less than 3% of the full model. As a result, the entire pruning and fine-tuning pipeline for DeepSeek-R1 completes in under 24 hours, representing a significant engineering advance for ultra-large MoE models. Complete specifications of pruning configurations and training hyperparameters are provided in Table 11.

D Training Losses Analysis

Figure 5a shows that our initialization effectively distinguishes important layers for Qwen2.5-14B-Instruct in the initial stage (excluding the first and last layers, which are not pruned). The training curves in Figures 5b and 5c indicate that both the proposed initialization and the Gumbel-Topk search contribute to superior pruning mask, leading to lower losses during either supervised fine-tuning or knowledge distillation.

Table 11: Hyperparameters details for pruning mask search and recovery training on LLaMA-2-7B, Qwen2.5-14B-Instruct, Qwen3-32B, DeepSeek-R1.

Hyperparameters	LLaMA-2-7B	Qwen2.5-14B-Instruct	Qwen3-32B	DeepSeek-R1
Pruning ratio	60%	50%	12.5% / 25%	12.5% / 25%
Learning rate	1e-4	1e-4	1e-5	1e-4
LR decay scheduler	Cosine	Cosine	Cosine	Cosine
LR warm-up steps	60	60	60	120
Training steps	1200	1200	1200	2400
Mask search steps	120	120	120	240
LoRA rank	-	-	-	64
LoRA alpha	-	-	-	128
Global batch size	128	32	32	64
Context length	4096	16384	16384	4096
Overall tokens	0.5B	0.5B	0.5B	0.5B
Distillation logits	Top 10	Top 10	Top 10	Top 10
Anneal coefficient β	0.9	0.9	0.9	0.9

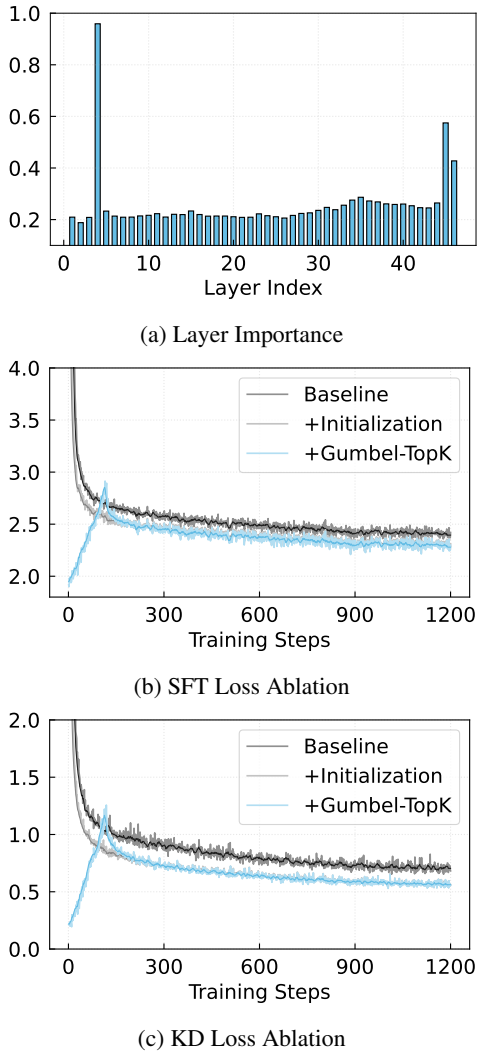


Figure 5: Ablations on Qwen2.5-14B-Instruct (a): Our initialization identifies important layers at the beginning. (b): Proposed components contribute to lower loss in SFT training. (c): Proposed components contribute to lower loss in KD training.

E Consistency Comparison between SFT and KD.

We demonstrate that knowledge distillation not only achieves substantially higher performance than supervised fine-tuning but also maintains significantly greater behavioral consistency between the pruned and original models. To assess behavioral consistency, we constructed an evaluation set by randomly sampling 100 test cases from each of four domains: MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), IFeval (Zhou et al., 2023), and Humaneval (Chen et al., 2021). The average accuracy on this set is reported as the consistency score. Using the Qwen2.5-14B-Instruct model, we conducted comprehensive comparative experiments under varying pruning ratios. As illustrated in Figure 6, both SFT and KD exhibit improved performance of the pruned models as training progresses, indicating a gradual recovery of capabilities. However, the behavioral consistency of SFT-trained models fails to improve effectively with additional steps; indeed, under the 1/8 pruning ratio, it further declines. This suggests that SFT leads to overfitting during recovery, causing the pruned model to diverge from the original model’s behavior. In contrast, KD ensures a steady increase in consistency throughout training, effectively preserving behavioral alignment with the original model.

F Gumbel Scores Optimization dynamics

Figure 7 illustrates the optimization trajectory of the Gumbel scores during the mask search phase. In Figure 7a, we present the evolution of layer-wise

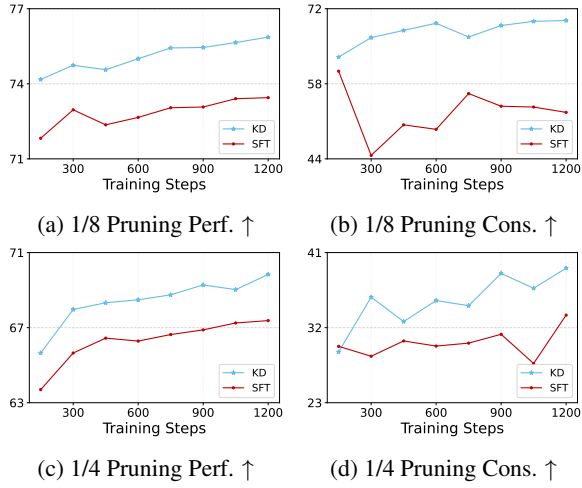


Figure 6: Performance and consistency during training. Despite both improving performance, KD effectively improves the consistency while SFT deteriorates consistency in some case.

scores throughout the search process. It can be seen that as the search progresses, certain layers initially selected for retention exhibit a decrease in scores and are eventually pruned. In contrast, other layers that demonstrate greater importance are identified and retained in later stages of the search. Figure 7b displays the layers retained during the forward pass, sampled by the proposed Gumbel-TopK sampler. Notably, due to our gradual strategy, the number of retained layers decreases progressively. In the early phases, the Gumbel-TopK sampler efficiently explores the mask search space, allowing nearly every layer to be potentially pruned. In later stages, as the scores converge and the sampling temperature decreases, the mask selection stabilizes, indicating that an optimal mask has been identified.

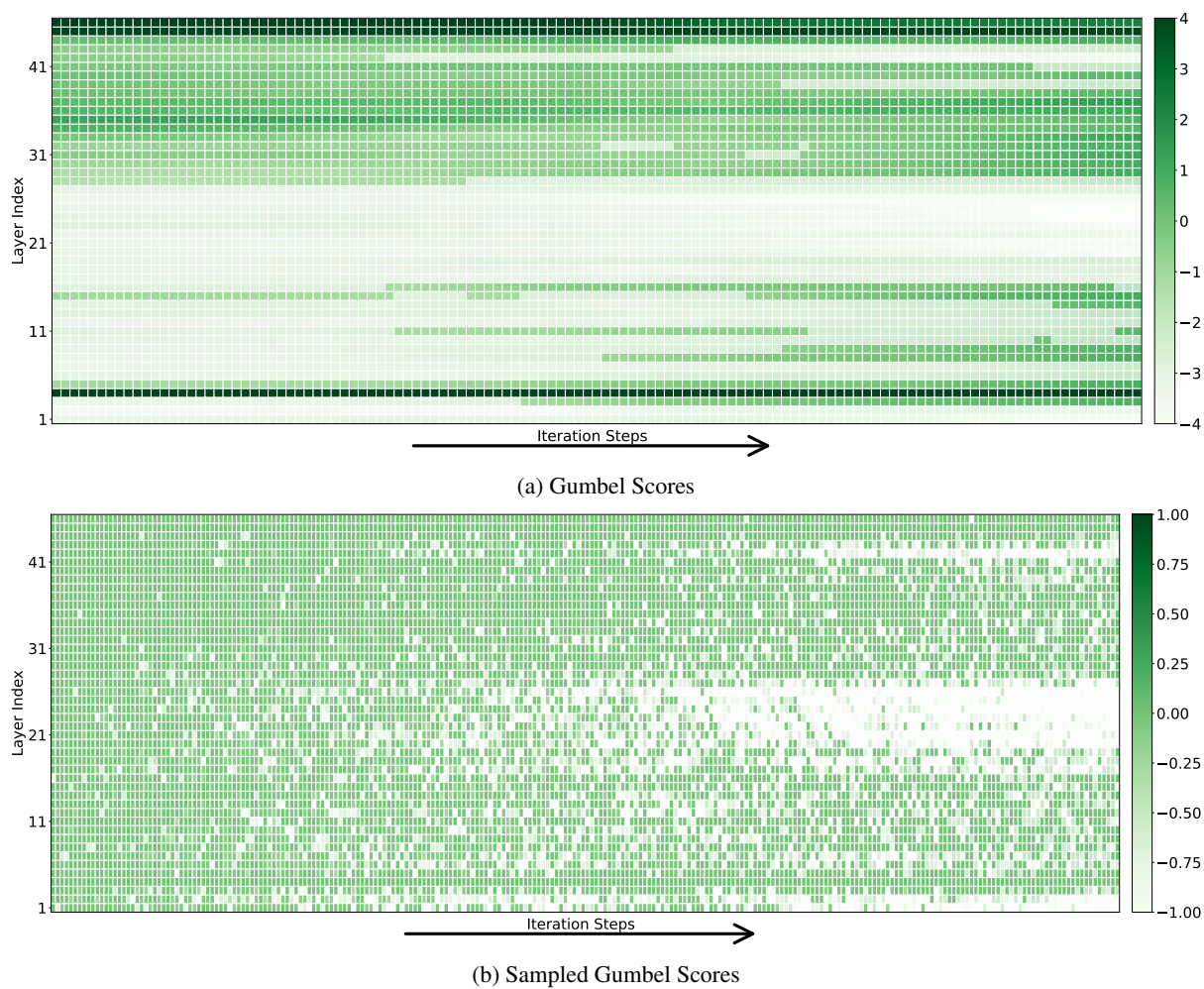


Figure 7: Gumbel scores optimization trajectory on Qwen2.5-14B-Instruct. (a): The Gumbel scores of each layer in each iteration step, the shadowed blocks indicate pruned layers. (b) The Sampled Gumbel scores during each forward pass.