Late Breaking Results: Dynamically Scalable Pruning for Transformer-Based Large Language Models

Junyoung Lee[1], Shinhyoung Jang[1], Seohyun Kim[1], Jongho Park[1]
Il Hong Suh[2] Hoon Sung Chwa[1] and Yeseong Kim[1]
[1]DGIST, [2]Coga Robotics
{lolcy3205, shinhyoung.jang, selenium, psyractal1123, chwahs, yeseongkim}@dgist.ac.kr, ihsuh@coga-robotics.com

***Abstract*—We propose Matryoshka, a novel framework for transformer model pruning, enabling dynamic runtime controls while maintaining competitive accuracy to modern large language models (LLMs). Matryoshka incrementally constructs submodels with varying complexities, allowing runtime adaptation without maintaining separate models. Our evaluations on LLaMA-7B demonstrate that Matryoshka achieves up to 34% speedup and outperforms the quality of state-of-the-art pruning methods, providing a flexible solution for deploying LLMs.**

***Index Terms*—Large Language Model, Depth-Pruning, Real-Time Management**

I. Introduction

Pruning [1] has emerged as one of the most promising approaches for reducing the computational cost of large models by removing redundant components. Traditionally, pruning techniques are categorized into *structured pruning*, which removes entire structural components such as blocks or channels, and *unstructured pruning*, which selectively zeros out individual weight parameters based on certain criteria, such as magnitude. While unstructured pruning can achieve finer granularity in parameter reduction, its reliance on sparsity-aware custom kernels often limits its actual efficiency gain on real-world hardware. On the other hand, structured pruning yields models with regular structures more suited for efficient inference, but it often causes significant accuracy degradation, particularly when large components such as decoder blocks are removed. Even though prior work has proposed various techniques to address this trade-off between efficiency and accuracy, a critical gap persists: most pruning techniques are static, designed for one-time optimization, and lack the flexibility to adapt to dynamic runtime constraints often required in system designs. Despite recent efforts for various model optimizations, a fundamental challenge remains: *how can the pruning approaches can offer dynamic adaptability to runtime constraints without requiring separate models for every deployment scenario?*

In this work, we propose Matryoshka, a framework that enables transformer models to dynamically scale their computational complexity during runtime while preserving competitive accuracy. Drawing inspiration from the nested structure of Matryoshka dolls, the framework constructs submodels of decreasing complexity that are embedded within the full model, allowing seamless transitions between configurations without maintaining separate models. Achieving this requires addressing three core challenges: mitigating the disruption to model representations caused by pruning tightly coupled transformer blocks, enabling smooth transitions between configurations to avoid abrupt changes in model behavior, and supporting real-time selection of optimal configurations based on runtime constraints such as latency and throughput.

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2022-0-00991, 1T-1C DRAM Array Based High-Bandwidth, Ultra-High Efficiency Processing-in-Memory Accelerator). This work was also supported by Cogarobotics.

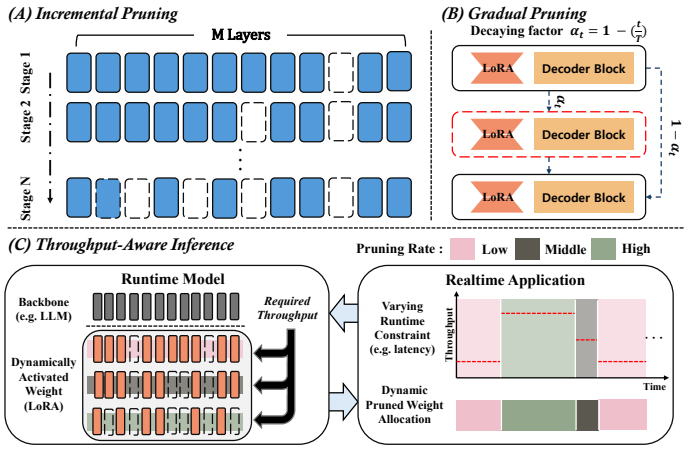

Fig. 1. Overview of Proposed Method

To tackle these challenges, Matryoshka integrates three key innovations: (i) *Incremental Pruning* progressively fine-tunes the model as blocks are removed, minimizing cumulative accuracy loss. (ii) *Gradual Pruning* deactivates blocks through a smooth transition mechanism, maintaining stable intermediate representations. (iii) *Throughput-Aware Inference Algorithm* dynamically selects the most suitable configuration during runtime.

II. Methods

The core idea of Matryoshka is to guide pruning and fine-tuning in a way that avoids abrupt shifts in the model weight distributions, thereby minimizing knowledge loss. Figure 1 illustrates the training and inference pipeline of our framework. During training steps, blocks are gradually removed based on their relative importance, determined by their contribution to the feature space representations. As each block is pruned, the remaining model is fine-tuned using a low-rank adaptation (LoRA) mechanism [2] to recover the knowledge lost during pruning. This process is repeated iteratively until the model achieves a hierarchy of submodels of decreasing complexity, all embedded within the full model. We observed that fine-tuning the model on insufficient datasets, such as Alpaca, leads to suboptimal model quality. To address this, we employ a dataset shuffling technique that combines multiple datasets from diverse sources. For inference, the system employs a throughput-aware inference algorithm to dynamically select the optimal submodel configuration based on runtime constraints such as throughput requirements.

Incremental Pruning Existing block-level pruning methods remove entire layers simultaneously, which can result in significant disruption to the learned representations of the original model. To mitigate this, we propose the *Incremental Pruning* technique, which identifies and removes blocks iteratively based on their contribution to the model feature space. Specifically,

978-3-9826741-0-0/DATE25/© 2025 EDAA

TABLE I
SUMMARY OF ACCURACY COMPARISON

Pruning Rate	Models	PPL ↓		Zero-shot Task Performance (Accuracy) ↑							
		Wiki2	PTB	Avg	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
0%	LLaMa-7B	12.6	22.1	66.3	75.0	78.7	76.2	69.9	75.3	44.7	44.4
20% (5.5B)	LLM-Pruner	17.6	**30.1**	61.8	66.2	**77.6**	**71.4**	66.1	**70.5**	**39.3**	41.2
	Ours-Mixed(Ours-Alpaca)	**13.6(17.5)**	**26.0(30.1)**	**62.4**	**73.6**	74.6	70.6	**69.3**	67.6	39.2	**41.6**
35% (4.5B)	LLM-Pruner	24.2	40.7	55.5	62.9	**72.8**	62.3	62.7	**57.4**	33.0	**37.6**
	Ours-Mixed(Ours-Alpaca)	**19.8(24.1)**	**37.4(40.5)**	**55.7**	**70.5**	70.2	**62.4**	**68.9**	49.1	**32.8**	36.4

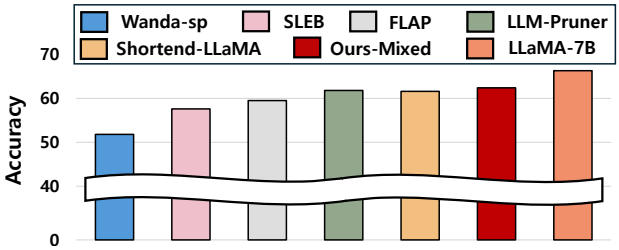

Fig. 2. Zero-shot Performance on 20% Pruned LLaMA-7B

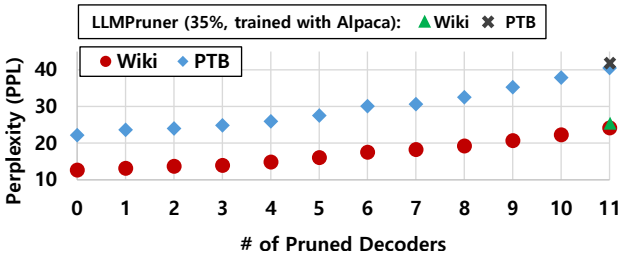

Fig. 3. PPL Changes over Pruning Levels (Fine-tuned with Alpaca Dataset)

for a given block, we calculate its importance using a cosine similarity metric between the output features of the original model (with the block) and those of the pruned model (without the block). The cosine similarity quantifies the preservation of feature information. Blocks with the lowest importance scores are pruned first, followed by a fine-tuning step using LoRA weights to compensate for the knowledge loss.

Gradual Pruning Removing transformer layers abruptly leads to significant degradation in model generality. To address this, we propose *Gradual Pruning*, which can be used along with the incremental pruning to smoothly deactivate a block's contributions over several fine-tuning steps. During this process, a linear decaying factor α_t, defined as: $\alpha_t = 1 - \frac{t}{T}$, is applied to the output of the block being pruned, where t is the current step and T is the total number of steps. As α_t decreases from 1 to 0, the output of the target block is progressively reduced to zero, allowing the downstream layers to adapt to its absence.

Dataset Shuffling Removing layers can lead to a reduction in the mutual information shared across the model's layers and a loss of domain information that individual layers can otherwise retain. To address these challenges, we shuffle multiple datasets during training so that the model is exposed to a diverse range of domains and enhance its generalization capabilities.

Throughput-Aware Inference The pruned model may operate under varying runtime constraints, such as fluctuating throughput and latency requirements, while maintaining quality metrics such as perplexity (PPL). The framework leverages the nested submodel structure to dynamically activate the appropriate configuration based on system demands, without needing separate models or incurring significant memory overhead. This is feasible as the LoRA adapters, constituting less than 1% of the original model size, can be preloaded alongside the full model in systems like GPUs.

III. EXPERIMENTAL RESULTS

We evaluate the proposed Matryoshka framework on the LLaMA-7B model; but it is noteworthy that the larger models are known to have larger rooms for the prunining as LLMs are often overparameterized. We fine-tuned models with LoRA ranks of 8. We implemented Matryoshka based on PyTorch 2.0.1 and evaluated it on NVIDIA 3090 GPUs.

Accuracy: Figure 2 presents the average accuracy results of representative zero-shot tasks for the proposed approach compared to existing pruning methods at a 20% pruning level. The results show that Matryoshka consistently outperforms the baseline across a range of zero-shot benchmarks. For instance, at a 20% pruning level, Matryoshka achieves an 62.4% accuracy on average, 0.6% higher than the state-of-the-art width-pruning algorithm, LLM-Pruner [3]. Table 1 summarizes the detailed comparison results for zero-shot tasks along with the PPL, including the results of using different datasets, i.e., the mixed (created with dataset shuffling) and standard Alpaca datasets. The results show that Matryoshka offers a higher quality both for the PPL and zero-shot accuracy, e.g., achieving a PPL of 13.6 at 20% pruning (19.8 at 35% pruning), which is notably lower than the PPL of 17.6 achieved by the baseline.

Tradeoff: Matryoshka facilitates inference by dynamically adjusting the pruning level, optimizing runtime in direct proportion to the pruning percentage, albeit with an associated accuracy tradeoff. For instance, a 20% pruning level reduces execution time by approximately 23% compared to the original model, while a 35% pruning level achieves a reduction of up to 34%. Figure 3 illustrates the PPL results for the model fine-tuned on the Alpaca dataset, comparing the performance of a 35% pruned model with LLMPruner fine-tuned on the same dataset. The results demonstrate that inference quality remains relatively stable until six decoder layers are pruned, beyond which the accuracy impact becomes relatively higher. The adaptability of Matryoshka enables the use of multiple submodels generated during pruning, providing flexibility to balance runtime efficiency and model quality.

IV. CONCLUSIONS

In this paper, we propose Matryoshka, a robust framework for scalable and adaptive transformer model pruning, addressing the challenges of runtime constraints in diverse deployment scenarios. By combining Incremental and Gradual Pruning with a dynamic inference algorithm, the framework balances computational efficiency and accuracy across a nested hierarchy of submodels. In our evaluation on LLaMA-7B, we show that Matryoshka achieves significant runtime reductions while outperforming the model performance against the existing pruning methods, offering flexible controls for latency requirements.

REFERENCES

[1] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models, 2024.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[3] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models, 2023.