NutePrune: Efficient Progressive Pruning with Numerous Teachers for Large Language Models

Anonymous ACL submission

Abstract

The considerable size of Large Language Models (LLMs) presents notable deployment challenges, particularly on resource-constrained hardware. Structured pruning, offers an effective means to compress LLMs, thereby reducing storage costs and enhancing inference speed for more efficient utilization. In this work, we study data-efficient and resource-efficient structure pruning methods to obtain smaller yet still powerful models. Knowledge Distillation is well-suited for pruning, as the intact model can serve as an excellent teacher for pruned students. However, it becomes challenging in the context of LLMs due to memory constraints. To address this, we propose an efficient progressive Numerous-teacher pruning method (NutePrune). NutePrune mitigates excessive memory costs by loading only one intact model and integrating it with various masks and LoRA modules, enabling it to seamlessly switch between teacher and student roles. This approach allows us to leverage numerous teachers with varying capacities to progressively guide the pruned model, enhancing overall performance. Extensive experiments across various tasks demonstrate the effectiveness of NutePrune. In LLaMA-7B zero-shot experiments, NutePrune retains 97.17% of the performance of the original model at 20% sparsity and 95.07% at 25% sparsity.

1 Introduction

Large Language Models (LLMs) excel in language tasks (OpenAI, 2023; Touvron et al., 2023; Thoppilan et al., 2022; Scao et al., 2022), but their substantial size poses deployment and inference challenges (Frantar et al., 2022). Techniques like model pruning (Molchanov et al., 2016), knowledge distillation (Jiao et al., 2019), and quantization (Dettmers et al., 2023) have been proposed to address computational demands. The exploration of LLM pruning, especially structured pruning (Frantar and Alistarh, 2023), holds great significance. Structured pruning reduces model size by removing coherent parameter groups, cutting inference costs on standard hardware. But it is more challenging than unstructured pruning in retaining the capabilities of LLMs (Hoefler et al., 2021). Existing methods either adopt data-efficient approaches, causing a performance decline (Ma et al., 2023), or require extensive post-training to recover model performance (Xia et al., 2023). In this work, we investigate efficient methods to prune the model to higher sparsity without significant performance decline.

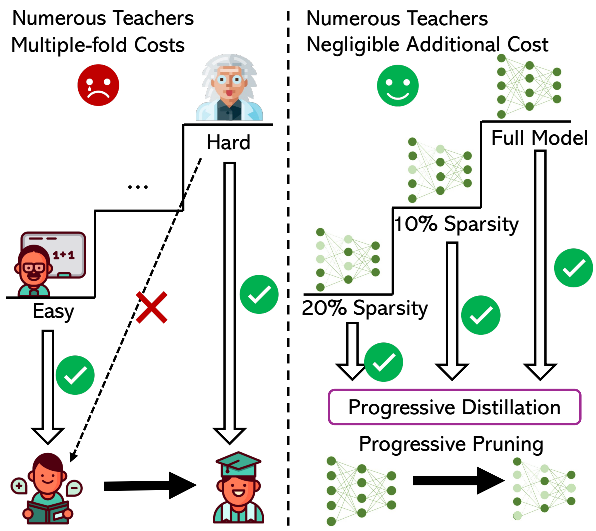

Figure 1: The advantage of our NutePrune. **Left**: Progressive distillation guides the student with teachers from easy to hard to avoid large capacity gap harming learning. But it suffers from multiple-fold costs of loading numerous teachers. **Right**: Our NutePrune leverages models with varying sparsity, enabling progressive distillation with negligible additional cost.

Knowledge distillation (KD) aims to train a more compact student model with supervision from a larger teacher model (Sanh et al., 2019; Gou et al., 2021). It's widely adopted and proven highly effective in the field of LLMs. Progressive learning, utilizing intermediate teachers with a reduced gap in capabilities, has been demonstrated to improve performance in KD (Xiang et al., 2020). Previous work has shown that pruning with a distillation objective can improve performance (Xia et al., 2022). Distillation is particularly suitable for pruning since the full original model inherently serves as an excellent teacher for the pruned model (Sanh et al., 2020), which can offer a more detailed supervisory signal than conventional supervised training, enhancing the effectiveness of pruning with limited data (Lagunas et al., 2021).

However, applying this method in the realm of LLMs proves challenging. Given the vastness of an LLM, loading it onto GPUs consumes a substantial amount of memory. Introducing an additional teacher model requires twice the memory, making it impractical with limited memory resources. Furthermore, relying on a single teacher may not be the best practice (Liu et al., 2020; Wu et al., 2021). With the increasing gap of sparsity between teacher and student, the capacity gap is also widening, which toughens distillation. Employing multiple teachers with varying capacities can enhance the transfer of knowledge to students (Yuan et al., 2021). However, when it comes to the distillation of LLMs, memory consumption of multiple teachers becomes an even more pressing concern.

Method	NutePrune	LLM-Pruner	KD
GPU Memory (GB)	28.7	35.4	42.1

Table 1: GPU memory consumption during pruning.

In this paper, we address the above challenges with an efficient progressive **Nu**merous-**te**acher pruning method (NutePrune). Our motivation is demonstrated in Figure 1. NutePrune aims to diminish the capacity gap between the full teacher model and the highly sparse student, thereby alleviating the difficulty of distillation (Su et al., 2021; Mukherjee et al., 2023; Xiang et al., 2020). Instead of relying solely on a single full teacher, we instruct the student with many teachers with varying sparsity. To achieve this, we formulate pruning as a optimization problem where we learn masks to prune sub-modules while updating model parameters through LoRA (Hu et al., 2021). Specially, we load an intact model, serving dual roles as both a teacher and a student. In teacher mode, we incorporate the original model with collected frozen low-sparsity masks and corresponding LoRA modules. And in student mode, we incorporate it with learnable high-sparsity masks and LoRA modules. Since the masks and LoRA modules are highly parameter efficient, we collect and leverage numerous modules with different sparsity to incorporate numerous teachers and progressively prune the student. And as shown in Table 1, this novel strategy remains highly memory efficient. Our contributions can be summarized as follows:

- We propose a novel distillation method that progressively guide the student using numerous teachers with varying sparsity to narrow the capacity gap. Through progressive KD, we achieve higher model sparsity without significant performance decline on limited data.
- Our NutePrune only loads one intact model and switch it between teacher and student modes by incorporating various masks and LoRA modules. This novel efficient distilling method for pruning enables using numerous teachers and introduces no extra memory cost, which is especially critical for LLMs.
- Extensive experiments, including LLaMA-1/2/3 with varying sizes and Mistral, demonstrate the effectiveness of our approach across perplexity, commonsense reasoning, MMLU, and BBH.

2 Related Works

Pruning Type	Speedup	No Support	No Index
Unstructured		✓	
Semi-Structured	✓		
Structured	✓✓	✓	✓

Table 2: Structured pruning yield most significant speedup without any special hardware support or aditional index storage.

Pruning for LLMs For LLMs, SparseGPT (Frantar and Alistarh, 2023) and WANDA (Sun et al., 2023) employ unstructured pruning methods, while N:M sparsity (Zhou et al., 2021) is considered semi-structured. Despite the effectiveness of these methods, their intricate structures do not yield significant inference speedup on standard hardware (Frantar and Alistarh, 2023) and they need to store additional indexes. As compared in Table 2, structured pruning offers significant advantages, resulting in increased focus on this field in recent works.

CoFi (Xia et al., 2022) and nn pruning (Lagunas et al., 2021) are proposed for smaller language models like BERT (Devlin et al., 2018), often designed for specific tasks. CoFi loads both the teacher and student models, which is impractical for LLMs. Sheared-LLaMA (Xia et al., 2023) proposes pruning LLMs using a dynamic pre-training method, enhancing performance through extensive data and training resources.

However, concerns persist regarding limited memory and training resources for LLMs. In a pioneering effort, LLM-Pruner (Ma et al., 2023) prunes LLMs in one-shot and utilizes LoRA (Hu et al., 2021) for fine-tuning. LoRAPrune (Zhang et al., 2023) employs iterative pruning, replacing gradients on full weights with gradients on LoRA to calculate group importance. Compresso (Guo et al., 2023) leverages LoRA and elaborately designed prompts for training and inference. Meanwhile, LoRAShear (Chen et al., 2023) employs LoRA and a dynamic fine-tuning scheme to recover knowledge.

Knowledge Distillation (KD) for LLMs KD (Hinton et al., 2015) has emerged as a vital technique to reduce inference costs while maintaining performance quality in the context of LLMs. Prior work of KD (Taori et al., 2023; Fu et al., 2023) mostly focuse on black-box KD, using teacher's generations to fine-tune the student. With the rise of open-source LLMs (Zhang et al., 2022; Touvron et al., 2023), interest in white-box KD is growing. White-box KD, leveraging teacher weights and logits, provides richer supervision signals, enhancing language abilities (Agarwal et al., 2023; Gu et al., 2023; Wen et al., 2023). Despite progress on small language models, significant performance gaps between large and small models persist (Achiam et al., 2023; Anil et al., 2023).

Progressive knowledge distillation (Xiang et al., 2020) has proven effective by using intermediate teachers to bridge the capacity gap with LLMs, especially in scenarios reliant on data generated by multiple teachers (Mukherjee et al., 2023). Orca (Mukherjee et al., 2023) first learns from easier examples from ChatGPT and then from harder ones from GPT-4, enhancing performance for smaller students in KD. However, applying white-box KD to LLMs poses challenges due to substantial memory requirements for loading both teacher and student models. This challenge becomes even more difficult when attempting to load multiple teachers.

3 Methodology

In this section, we first introduce how our NutePrune enables efficient knowledge distillation for structured pruning in 3.1. Then, to narrow capacity gap during distillation, we introduce the progressive knowledge distillation method that collects and incorporates numerous teachers in 3.2. The overview framework is illustrated in Figure 2.

3.1 Efficient Distillation for Structured Pruning

We formulate structure pruning as a constrained optimization problem where we simultaneously learn masks to prune the structure and update the model to recover ability. To mitigate memory consumption, we utilize LoRA for model updates, making pruning the process of training these masks and LoRA parameters.

Learning masks to control the pruned structure Three types of structure are pruned: attention heads, FFN intermediate dimensions, and hidden dimensions. We achieve this by learning masks $\mathbf{z}_{head}, \mathbf{z}_{int}, \mathbf{z}_{hid} \in \{0, 1\}$. Formally, the multi-head attention module $\mathrm{MHA}(x)$ and feed-forward networks $\mathrm{FFN}(x)$ of layer l are pruned as:

$$\mathrm{MHA}^l(X) = \mathbf{z}_{hid} \cdot \sum_{h=1}^{N_{head}} \mathbf{z}_{head}^{l,h} \,\mathrm{Att}^{l,h}(X). \quad (1)$$

$$\mathrm{FFN}^l(X) = \mathbf{z}_{hid} \cdot W_D^l \left(\mathbf{z}_{int}^l \cdot W_U^l(X) \cdot W_G^l(X)\right) \quad (2)$$

where $\mathrm{Att}()$ is the attention module and activation is omitted. W_D, W_U, W_G are down projection, up projection, and gating projection.

During mask training, we calculate the remaining size to obtain the expected sparsity $\hat{s}$:

$$\begin{aligned}\hat{s}(\mathbf{z}) &= \frac{1}{M} \cdot 4 \cdot d_h \cdot \sum_{l}^{L} \sum_{h}^{N_{head}} \sum_{k}^{d} \mathbf{z}_{head}^{l,h} \mathbf{z}_{hid}^{k} \\ &+ \frac{1}{M} \cdot 3 \cdot \sum_{l}^{L} \sum_{i}^{d_{int}} \sum_{k}^{d} \mathbf{z}_{int}^{l,i} \mathbf{z}_{hid}^{k},\end{aligned} \quad (3)$$

where M denotes full model size. L is number of layers. $d_h, N_{head}, d, d_{int}$ are head dimension, number of head, hidden dimension, and intermediate dimension, correspondingly.

All masking variables are learned as real numbers in $[0, 1]$ during training. We follow (Louizos et al., 2017; Guo et al., 2023) and employ the augmented L_0 regularization, which is detailed in Appendix A.

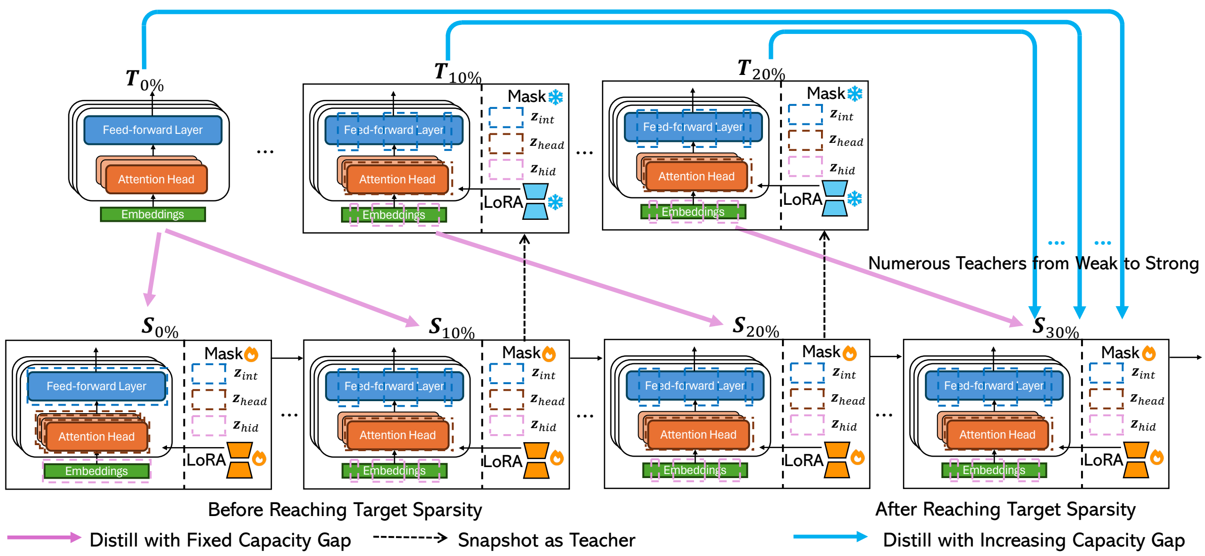

Figure 2: The overall framework of NutePrune. The pruned model is frozen and incorporated with learnable masks and LoRA. During pruning, the model is guided by numerous teachers. Before pruned to the target sparsity (e.g. 30%), it learns from teachers with a fixed capacity gap. Once the target sparsity is achieved, it continues to learn from all previous teachers from weak to strong. All these teachers are derived from snapshots of the student model itself. Since only the mask and LoRA modules are snapshotted, the additional memory cost is negligible.

Updating parameters with LoRA Considering massive memory usage during full fine-tuning for LLMs, we incorporate lightweight LoRA (Hu et al., 2021) modules into LLM weights to update parameters during pruning.

An incorporated module W' is consisted of the original weight $W : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and sequential LoRA weights parallel to W:

$$W'(X) = W(X) + W_B\left(W_A(X)\right), \quad (4)$$

where $W_A : \mathbb{R}^n \rightarrow \mathbb{R}^r, W_B : \mathbb{R}^r \rightarrow \mathbb{R}^m$ and $r \ll m, n$. During training, W is frozen and only W_A and W_B are learnable.

Efficient distillation Instead of simultaneously loading two massive models into memory, we propose to incorporate the frozen and intact model M with different lightweight masks and LoRA modules for the teacher and the student. Formally, let $\mathbf{I} = \{\mathbf{z}, \mathbf{W}_A, \mathbf{W}_B\}$ denotes the set of all masks and LoRA modules which is highly parameter efficient ($|\mathbf{I}| \ll |\mathbf{M}|$). By incorporating $\mathbf{I}$ into $\mathbf{M}$, we obtain $\mathbf{M_I}$. The objective of knowledge distillation is the KL-divergence (Van Erven and Harremos, 2014) between teacher's and student's output probability distributions p:

$$\mathcal{L}_{KL} = D_{KL}(p(\mathbf{M}_{\mathbf{I}_S}, x), p(\mathbf{M}_{\mathbf{I}_T}, x)), \quad (5)$$

where x denotes training data. $\mathbf{I}_S$ and $\mathbf{I}_T$ denote the lightweight modules of student and teacher.

Additionally, intermediate layers of a teacher model can serve as effective targets for training a student model (Chen et al., 2021). This objective can be formulated as:

$$\mathcal{L}_{layer} = \sum_{l}^{L} \mathrm{MSE}(\mathbf{h}_l(\mathbf{M}_{\mathbf{I}_S}, x), \mathbf{h}_l(\mathbf{M}_{\mathbf{I}_T}, x)), \quad (6)$$

where $\mathbf{h}_l$ is the hidden embedding of the l-th layer. Therefore, the overall objective is:

$$\mathcal{L} = \mathcal{L}_{KL} + \alpha_1 \mathcal{L}_{layer} + \alpha_2 \mathcal{L}_0, \quad (7)$$

where α_1, α_2 are hyperparameters to control the importance of different loss terms.

3.2 Progressive Knowledge Distillation with Numerous Teachers

All teachers are collected from the snapshot of students as the dotted line illustrated in Figure 2. To narrow the capacity gap between the intact teacher and high sparsity students, we leverage a novel progressive knowledge distillation (PKD) method for pruning. It consists of two stages when pruning a model from 0% sparsity as illustrated in Figure 3.

Before reaching target sparsity The sparsity of pruned model gradually increase from 0 to t. To narrow the sparsity gap, we set a fixed gap value g and make the pruned model S guided by teachers T whose sparsity $\hat{s}(T)$ is approximately g less than $\hat{s}(S)$: $\hat{s}(T) = \hat{s}(S) - g$. These teachers are

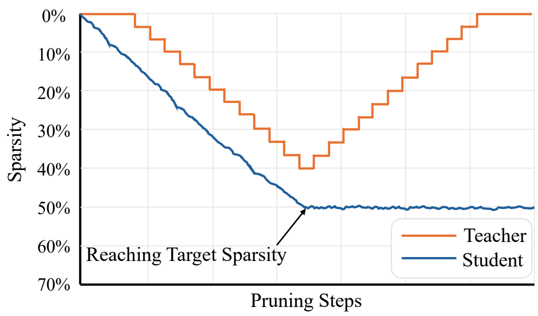

Figure 3: Illustration of the sparsity of teacher and student models during pruning. Take the example with the target sparsity $t = 50\%$ and sparsity gap $g = 10\%$.

snapshots of previous students. The original intact model serves as the teacher for student $\hat{s}(S) < g$.

To avoid collecting too many teachers, we only collect teachers with an interval of i. Therefore, for any teacher with sparsity $\hat{s}(T)$, it is responsible for guiding a student set within a range of sparsity. We use $\rightarrow$ to denote the relationship in which a teacher distills knowledge to students.

$$T \rightarrow \{S | \hat{s}(T) + g < \hat{s}(S) < \hat{s}(T) + g + i\}. \quad (8)$$

And the intact model $\mathbf{M} = T_0$ is responsible for the early students whose sparsity is less than $g + i$:

$$T_0 \rightarrow \{S | \hat{s}(S) < g + i\}. \quad (9)$$

After reaching target sparsity When the pruned model reaches the target sparsity t, we proceed to the second stage of PKD. The model undergoes distillation by all preceding teachers, with a reduction of sparsity in the teachers. This gradual process guides the model's learning trajectory from weaker to stronger knowledge and from easier to more challenging concepts. Throughout this stage, the sparsity of the pruned model $\hat{s}$ remains close to the target sparsity t, while the masks $\mathbf{z}$ and LoRA modeuls $\mathbf{W}_A, \mathbf{W}_B$ are continually optimized.

To receive sufficient instruction from the best model (the intact model $\mathbf{M}$), the teacher model is maintained as $\mathbf{M}$ during the final period.

3.3 Post Fine-tuning

After the pruning phase, to obtain better performance, we undergo a post fine-tuning stage following LLM-Pruner (Ma et al., 2023). We fix the masks and only fine-tune LoRA modules on the Standford Alpaca (Taori et al., 2023) dataset.

4 Experiments

4.1 Experimental Setup

Datasets To assess the zero-shot ability of LLMs, we perform zero-shot classification tasks on seven commonsense reasoning benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy (Clark et al., 2018), ARC-challenge (Clark et al., 2018), and OpenBookQA (OBQA) (Mihaylov et al., 2018). We evaluate the general capcability of LLMs on the perplexity metric with WikiText (Merity et al., 2016). Additionally, to evaluate the in-context learning ability, We report the results on 5-shot MMLU (Hendrycks et al., 2020), and 3-shot BBH (Suzgun et al., 2022).

Models NutePrune is applicable across various models of different sizes. We assess the performance of NutePrune on the LLaMA-1 family (7B/13B) (Thoppilan et al., 2022), LLaMA-2 (Touvron et al., 2023) family (7B/13B), LLaMA-3-8B and Mistral-7B (Jiang et al., 2023).

Baselines Considering the benefits of inference acceleration, we focus on structured pruning. We first replicate conventional methods: Magnitude pruning (MaP) (Li et al., 2018), Movement Pruning (MvP) (Sanh et al., 2020), and WANDA (Sun et al., 2023). For recent open-source methods, we implement LLM-Pruner (Ma et al., 2023) and Compresso (Guo et al., 2023) and conduct detailed comparisons. For more recent works that are not publicly available, we assess NutePrune using the same settings as theirs. This includes LoRAPrune (Zhang et al., 2023) and LoRAShear (Chen et al., 2023).

Implementation details For pruning stage, we sample 20,000 sentences from the C4 (Raffel et al., 2020) dataset with a length of 512 tokens. We train with AdamW optimizer, a batch size of 16, and learning rates of 0.1 for masks and 0.001 for LoRA. We prune the model for 7 epochs and a linear sparsity schedule for target sparsity warmup: 4 epochs for 20% sparsity and 1 epoch for 50%. The sparsity gap between the teacher and student g is 10% and the snapshot interval i of teachers is 1%. After pruning, we post fine-tune the pruned model on the Alpaca dataset (Taori et al., 2023) for 3 epochs. All experiments are conducted on one A100 GPU (80G).

Ratio	Method	WikiText2↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.	⋆Avg.
0%	LLaMA-7B	5.68	73.18	78.35	72.99	67.01	67.45	41.38	42.40	63.25	66.39
20%	LLM-Pruner	9.96	59.39	75.57	65.34	61.33	59.18	37.12	39.80	56.82	59.01
	LoRAPrune	-	57.98	75.11	65.81	59.90	62.14	34.59	39.98	56.50	-
	†**NutePrune**	**8.02**	63.21	76.55	67.96	66.69	63.72	38.05	40.00	**59.46**	**63.03**
20% Tuned	MaP	12.67	60.00	76.12	65.43	60.93	60.31	37.80	39.80	57.20	60.05
	MvP	10.52	64.50	73.50	62.50	61.80	62.42	36.95	37.80	57.07	58.76
	WANDA	-	65.75	74.70	64.52	59.35	60.65	36.26	39.40	57.23	-
	LLM-Pruner	8.57	69.54	76.44	68.11	65.11	63.43	37.88	40.00	60.07	61.94
	LoRAPrune	-	65.82	79.31	70.00	62.76	65.87	37.69	39.14	60.05	-
	LoRAShear	-	70.17	76.89	68.69	65.83	64.11	38.77	39.97	60.63	-
	Compresso	10.38	73.64	75.08	64.77	67.72	66.12	37.54	40.40	60.75	62.60
	‡**NutePrune**	8.04	72.69	76.71	68.99	65.51	65.49	38.48	40.20	61.15	63.57
	NutePrune	**7.65**	74.56	77.04	70.01	65.67	65.78	37.97	39.20	**61.46**	**64.39**
25%	†**NutePrune**	9.04	68.10	75.35	66.75	62.04	58.08	36.77	39.00	58.01	61.72
25% Tuned	‡**NutePrune**	-	65.84	76.17	66.69	64.56	61.49	37.03	39.20	58.71	63.12
	NutePrune	7.85	68.99	77.20	67.90	65.04	63.76	37.80	40.20	60.13	63.78
50%	LLM-Pruner	98.10	52.32	59.63	35.64	53.20	33.50	27.22	33.40	42.13	40.94
	LoRAPrune	-	51.78	56.90	36.76	53.80	33.82	26.93	33.10	41.87	-
	†**NutePrune**	**17.45**	62.29	67.95	53.03	57.06	45.45	30.03	36.60	**50.35**	**53.14**
50% Tuned	MaP	33.18	39.69	66.81	42.49	50.67	49.32	30.63	31.40	44.43	46.33
	MvP	27.62	59.94	63.06	40.98	55.64	44.07	26.79	31.80	46.04	46.23
	WANDA	-	50.90	57.38	38.12	55.98	42.68	34.20	38.78	45.43	-
	LLM-Pruner	22.76	61.47	68.82	47.56	55.09	46.46	28.24	35.20	48.98	48.97
	LoRAPrune	-	61.88	71.53	47.86	55.01	45.13	31.62	34.98	49.71	-
	LoRAShear	-	62.12	71.80	48.01	56.29	47.68	32.26	34.61	50.39	-
	Compresso	59.73	60.09	66.70	39.31	51.93	48.82	27.82	33.40	46.87	47.43
	‡**NutePrune**	16.72	62.20	69.91	53.87	57.77	46.59	31.74	35.80	51.13	53.94
	NutePrune	**13.20**	62.26	71.00	55.88	57.54	51.68	32.17	34.40	**52.13**	**54.91**

† only prunes the model by training masks without incorporating LoRA modules.
‡ prunes the model by co-training the masks and LoRA modules but without post fine-tuning on Alpaca.
⋆ includes results with the newer version of *lm-evaluation-harness*. See Appendix B for detail.

Table 3: Performance (%) of the compressed LLaMA-7B models.

Ratio	Method	WikiText2↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	⋆Avg.
0%	LLaMA-2-7B	5.47	77.74	79.11	75.97	69.06	76.35	46.33	44.20	66.97
20%	LLM-Pruner	12.94	50.55	75.46	67.18	65.67	67.38	38.14	38.40	57.54
	NutePrune	**8.74**	77.06	76.66	70.56	65.59	71.97	42.58	42.40	**63.83**
50%	LLM-Pruner	24.47	54.13	68.06	46.71	51.54	50.97	25.85	34.00	47.30
	NutePrune	**12.94**	66.24	70.83	57.04	59.51	58.46	31.97	34.00	**54.01**

Table 4: Performance (%) of the compressed LLaMA-2-7B models.

Ratio	Method	WikiText2↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	⋆Avg.
0%	Mistral-7B	5.25	83.73	82.26	81.05	74.19	80.89	53.84	43.80	71.39
20%	LLM-Pruner	7.50	77.52	78.13	73.64	69.46	72.59	46.41	41.80	65.65
	NutePrune	**7.06**	78.29	80.41	75.57	68.82	76.35	45.05	42.20	**66.67**
50%	LLM-Pruner	30.51	62.48	66.59	48.00	56.51	52.61	28.07	29.80	49.15
	NutePrune	**12.29**	63.64	72.63	57.95	61.25	62.46	35.41	33.80	**55.31**

Table 5: Performance (%) of the compressed Mistral-7B models.

4.2 Results

Zero-shot performance Table 3 demonstrates PPL and zero-shot performances on commonsense reasoning tasks for compressed LLaMA-7B models. The reported results include experiments for 20%, 25% and 50% sparsity levels, covering scenarios with and without parameter tuning.

The average performance of NutePrune consistently outperforms previous methods across all settings. For pruning without tuning, NutePrune outperforms LLM-Pruner by 2.64%/8.22% at 20%/50% sparsity, underscoring its ability to derive a more effective pruned structure compared to other methods. For pruning with LoRA contrained, NutePrune improves from 59.46%/50.35% to 61.46%/52.13% at 20%/50% sparsity, indicating

Ratio	Method	WikiText2↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	⋆Avg.
0%	LLaMA-3-8B	6.14	81.04	80.85	79.18	73.40	80.13	53.16	44.60	70.34
20%	LLM-Pruner	10.06	71.50	77.97	70.49	68.75	72.35	42.83	38.40	63.18
	NutePrune	**9.51**	78.65	79.76	73.74	70.09	76.22	45.90	43.40	**66.82**
50%	LLM-Pruner	27.37	41.59	67.46	46.53	55.64	49.71	27.22	31.60	45.68
	NutePrune	**21.72**	65.20	68.72	52.30	58.64	53.41	29.44	35.00	**51.82**

Table 6: Performance (%) of the compressed LLaMA-3-8B models.

Ratio	Method	Avg. Zero-Shot (%)	WikiText2↓	MMLU (5-shot)	BBH (3-shot)
0%	LLaMA-13B	67.63	5.62	46.90	37.72
20%	LLM-Pruner	65.76	6.95	29.57	15.77
	NutePrune	**67.51**	**6.55**	**39.37**	**31.99**
0%	LLaMA-2-13B	68.80	5.30	54.86	39.53
20%	LLM-Pruner	65.87	7.25	31.26	25.20
	NutePrune	**67.39**	**6.86**	**45.78**	**31.22**

Table 7: Performance of the compressed LLaMA-13B and LLaMA-2-13B models with 20% sparsity.

co-training with LoRA could help recover model capability damaged by pruning. And with additional post fine-tuning on Alpaca, notably, it retains 97.17% of the performance of the original model at 20% sparsity and 95.07% at 25% sparsity.

Table 4, 5 and 6 further demonstrates performances for compressed LLaMA-2-7B, Mistral-7B and LLaMA-3 models. At the same sparsity, multi-query attention models experience a more significant performance decline. Nevertheless, NutePrune consistently outperforms LLM-Pruner, proving our method is applicable across various models. Noticeable improvements are observed at higher sparsity levels, proving the effectiveness of our PKD in mitigating the capacity gap during distillation.

Pruning of larger model We assess larger models: LLaMA-13B and LLaMA-2-13B with 20% sparsity. To evaluate the ability of these stronger models, we further assess their in-context learning ability with MMLU and BBH (Brown et al., 2020). As demonstrated in Table 7, our approach yield an average zero-shot commonsense reasoning performance of 67.51% and 67.39%, which is only slightly lower than the full model and much higher than LLM-Pruner. It also outperforms LLM-Pruner in terms of PPL in WikiText2. For in-context learning ability, NutePrune achieves a score of 36.37 MMLU and 31.99 BBH in LLaMA-13B, and 45.78 MMLU and 31.22 BBH in LLaMA-2-13B. The slight decline in performance compared to the full model is acceptable, indicating that NutePrune maintains sufficient in-context learning capability. Additionally, when compared to LLM-Pruner, our advantages are clearly evident.

Inference latency We test the inference latency by generating from 64 tokens to 256 tokens on vLLM (Kwon et al., 2023), which is a fast and widely deployed library for LLM inference and serving. The results are presented in Table 8. NutePrune achieves latency savings of 11% and 29% at sparsity levels of 20% and 50%. While LLM-Pruner save slightly more latency due to its predefined neater structure, it comes at the cost of reduced flexibility in tailoring. As sparsity increases, the difference becomes negligible.

Method	20%	50%
0% Baseline	3.06	
LLM-Pruner	2.63(-14%)	2.17(-29%)
NutePrune	2.72(-11%)	2.18(-29%)

Table 8: Inference latency of pruned LLaMA-7B.

Training cost We report the memory and latency cost on different settings in Table 9. For extra GPU memory cost of PKD, NutePrune snapshot lightweight modules (masks and LoRA) of numerous teachers into CPU. Only one teacher module is loaded onto the GPU when needed, resulting in negligible memory cost compared with KD. In terms of extra time cost, compared with supervised training, KD requires one extra forward pass of teacher model, which is inevitable and cost 18.0% extra latency. When snapshoting a teacher or switching to a new teacher, due to the extremely low frequency of operations, the time can be ignored. Introducing $\mathcal{L}_{layer}$ requires additional 32% memory which is also efficient compared to conventional KD.

Progressive	KD	$\mathcal{L}_{layer}$	Memory	Latency
			27.68	3.67
	✓		28.67	4.33
✓	✓		28.69	4.33
✓	✓	✓	38.00	5.52

Table 9: Training cost measured by average GPU memory (GB) and per step latency (s/iter).

4.3 Ablation Study

We validate the effectiveness of NutePrune and investigate which properties make for a good NutePrune. Results are average zero-shot performance with tuning but without post fine-tuning, unless otherwise stated.

Effectiveness of PKD To validate progressive knowledege distillation (PKD) in our NutePrune, we conduct ablation studies on various learning strategies. We eliminate the progressive schedule and adopt standard KD, where the intact model serves as the teacher throughout. Subsequently, we exclude the entire distillation procedure and employ the standard generative language model loss, specifically next-token prediction, to train masks and LoRA modules. The results presented in Table 10 demonstrate the critical role of KD in enhancing performance, with further improvements achieved through PKD. This phenomenon is particularly pronounced at higher sparsity.

Progressive	KD	20%	50%
✓	✓	**63.57**	**53.94**
	✓	63.19	52.73
		59.98	41.77

Table 10: NutePrune and variants at 20%/50% sparsity.

Two stages of PKD PKD includes one stage before reaching target sparsity and the other stage after that. Different progressive schedules are adopted. To assess the effectiveness of them, we conducted an ablation study at 50% sparsity under two training settings, as shown in Table 11: training masks only and co-training masks with LoRA. In a stage without a progressive schedule, the intact model serves as the teacher. For the masks-only scenario, adopting either stage 1 or 2 alone yields significant improvements over KD. And for co-training, significant improvement is observed when both stages are adopted simultaneously.

Stage 1	Stage 2	Avg.(%) masks-only	Avg.(%) co-train
✓	✓	**53.14**	**53.94**
✓		52.31	52.79
	✓	52.40	52.53
		51.83	52.73

Table 11: Performance of two stages of PKD.

Sparsity gap and interval of teachers During stage 1, the sparsity gap between teacher and student model is an important hyperparameter. As shown in Table 12, a 10% gap is deemed appropriate to prevent a gap that is too small, as it may result in insufficient guidance, or a gap that is too large, which would toughing distillation. And when taking snapshots of students as teachers, it is preferable to save as many teachers as possible to facilitate more comprehensive training. However, it comes with extra costs. As demonstrated in Table 13, selecting an interval of 1% leads to significant improvement over the 10% interval, and the associated extra storage is acceptable.

Sparsity Gap	5%	10%	20%
Avg.(%)	53.62	**53.94**	53.04

Table 12: Performance of various sparsity gap.

Snapshot Interval	CPU Storage	Avg.(%)
1% (ours)	728MB	**53.94**
10%	73MB	53.27

Table 13: Storage and performance of various intervals.

5 Conclusion

In this work, we propose NutePrune as a novel efficient progressive structured pruning method for LLMs. Our well-designed techniques minimize the memory cost of KD, enabling NutePrune to utilize numerous teachers to mitigate the capacity gap between teacher and student and improve the quality of distillation. We show the effectiveness of NutePrune across various base models on diverse metrics. This work contributes to structured pruning techniques for LLMs, particularly in resource-constrained scenarios.

6 Limitations

Recent work (Ma et al., 2023; Xia et al., 2023) proves that using extensive data for post-training could substantially enhance the performance, but it comes with a substantial increase in computational costs. We target on pruning on resource-constraint scenarios and leave pruning with extensive data for future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. 2021. Cross-layer distillation with semantic calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7028–7036.

Tianyi Chen, Tianyu Ding, Badal Yadav, Ilya Zharkov, and Luming Liang. 2023. Lorashear: Efficient large language model structured pruning and knowledge recovery. *arXiv preprint arXiv:2310.18356*.

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. *arXiv 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. *arXiv preprint arXiv:1803.05457*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Song Guo, Jiahang Xu, Li Lyna Zhang, and Mao Yang. 2023. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015*.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1):10882–11005.

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. *arXiv preprint arXiv:2106.09685*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

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*.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. 2021. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*.

Guiying Li, Chao Qian, Chunhui Jiang, Xiaofen Lu, and Ke Tang. 2018. Optimization based layer-wise magnitude-based pruning for dnn compression. In *IJCAI*, volume 330, pages 2383–2389.

Yuang Liu, Wei Zhang, and Jun Wang. 2020. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

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

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2:13.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Weiyue Su, Xuyi Chen, Shikun Feng, Jiaxiang Liu, Weixin Liu, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie-tiny: A progressive distillation framework for pretrained transformer compression. *arXiv preprint arXiv:2106.02241*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

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.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tim Van Erven and Peter Harremos. 2014. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. 2023. f-divergence minimization for sequence-level knowledge distillation. *arXiv preprint arXiv:2307.15190*.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2021. One teacher is enough? pre-trained language model distillation from multiple teachers. *arXiv preprint arXiv:2106.01023*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*.

Liuyu Xiang, Guiguang Ding, and Jungong Han. 2020. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 247–263. Springer.

Fei Yuan, Linjun Shou, Jian Pei, Wutao Lin, Ming Gong, Yan Fu, and Daxin Jiang. 2021. Reinforced multi-teacher selection for knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14284–14291.

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

Mingyang Zhang, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, Bohan Zhuang, et al. 2023. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*.

A Detailed L_0 Regularization

To get the learnable masks $\mathbf{z}$, L_0 regularization introduces a sampling strategy as augmentation during training. $\mathbf{z}$ is a real number in $[0, 1]$ and is obtained from:

$$\begin{aligned}
&\mathbf{u} \sim U(0,1) \\
&\mathbf{s} = \text{sigmoid}\left(\frac{1}{\beta}\log\frac{\mathbf{u}}{\mathbf{1}-\mathbf{u}} + \log\alpha\right) \\
&\tilde{\mathbf{s}} = \mathbf{s} \times (r - l) + l \\
&\mathbf{z} = \min(1, \max(0, \tilde{\mathbf{s}})),
\end{aligned} \tag{10}$$

where $\mathbf{u}$ is uniformly sampled between 0 to 1. α is the parameter to be learned and β is a hyperparameter. l, r is often -0.1 and 1.1 to ensure most $\mathbf{z}$ are either 0 or 1 after training.

To prevent models from drastically converging to different sizes, we follow (Wang et al., 2019) to use this Lagrangian term:

$$\mathcal{L}_0 = \lambda_1 \cdot (\hat{s} - t) + \lambda_2 \cdot (\hat{s} - t)^2, \tag{11}$$

where λ_1 and λ_2 are both learnable. This loss term $\mathcal{L}_0$ will impose $\hat{s}$ to gradually converge to target sparsity t.

B Zero-shot Performance with Newer Version

lm-evaluation-harness released a new version in June 2023 to assess the zero-shot performance of LLaMA [1]. This update addressed a tokenization bug specific to LLaMA, resulting in higher and more accurate performance results compared to the older version. Despite these improvements, current state-of-the-art reports continue to reference the older version. Consequently, we conducted experiments using both the new and old versions, and the detailed results for the new version are presented in Table 14.

C Pruning at Higher Sparsity

To demonstrate the effectiveness of NutePrune at higher sparsity, we conducted experiments at 70% sparsity in Table 14.

D Pruned Structure

To gain insights into the pruned model, we present a detailed overview of the pruned structure at sparsity levels of 20% and 50%. The original hidden dimension is 4096, with a number of heads set at 32 and an intermediate dimension of 11008. Tables 16 and 17 reveal several observations. Notably, NutePrune tends to avoid pruning the hidden dimension, which aligns with the observation that pruning it may result in significant performance degradation (Ma et al., 2023). Regarding heads and intermediate dimensions, NutePrune tends to prune the the last few layers. This observation differs from LLM-Pruner, which asserts the importance of the last layers. Further analysis of this phenomenon is left for future work.

[1] https://github.com/EleutherAI/lm-evaluation-harness/pull/531

Ratio	Tune	Method	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	⋆Avg.
0%		LLaMA-7B	75.11	79.16	76.21	69.85	75.29	44.71	44.40	66.39
20%		LLM-Pruner	57.49	76.06	69.53	63.93	67.17	38.05	40.80	59.01
		†**NutePrune**	70.21	76.93	71.66	68.27	71.09	40.44	42.60	**63.03**
20%	✓	MaP	64.43	76.82	67.42	63.61	66.92	36.95	44.20	60.05
		MvP	64.56	75.19	64.71	64.09	66.12	38.65	38.00	58.76
		LLM-Pruner	67.37	77.86	71.47	65.90	69.57	39.59	41.80	61.94
		Compresso	73.21	75.90	66.90	68.90	69.99	41.47	41.80	62.60
		‡**NutePrune**	73.79	77.37	72.27	67.48	72.77	38.91	42.40	63.57
		NutePrune	75.38	78.02	72.97	67.40	73.82	40.36	42.80	**64.39**
25%		†**NutePrune**	71.53	76.50	70.60	65.98	69.11	39.93	38.40	61.72
25%	✓	‡**NutePrune**	72.91	77.42	70.34	68.11	70.92	41.55	40.60	63.12
		NutePrune	74.95	77.75	71.27	67.40	71.25	41.81	42.00	63.78
50%		LLM-Pruner	46.48	61.10	36.87	51.78	35.10	27.65	27.60	40.94
		†**NutePrune**	65.38	69.04	55.08	61.33	55.72	30.80	34.60	**53.14**
20%	✓	MaP	43.33	67.46	44.27	54.78	52.19	30.46	31.80	46.33
		MvP	60.00	63.11	41.73	56.04	47.10	27.05	28.60	46.23
		LLM-Pruner	57.89	69.97	50.06	52.64	49.66	28.58	34.00	48.97
		Compresso	61.31	66.32	40.73	52.41	51.18	27.65	32.40	47.43
		‡**NutePrune**	67.25	70.67	56.64	59.83	57.07	31.74	34.40	53.94
		NutePrune	67.52	71.60	58.64	60.14	59.72	32.94	33.80	**54.91**

Table 14: Zero-shot performance of the compressed LLaMA models in the new version of lm-evaluation-harness. **Bold** denotes the best average performance at the same setting.

Ratio	Method	WikiText2↓	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	⋆Avg.
0%	LLaMA-7B	5.68	75.11	79.16	76.21	69.85	75.29	44.71	44.40	66.39
70%	LLM-Pruner	56.33	47.28	60.83	31.66	50.75	39.56	24.83	28.80	40.53
	NutePrune	**34.30**	62.08	62.30	39.43	51.46	42.17	26.19	30.20	**44.83**

Table 15: Performance (%) of the compressed LLaMA-7B models at 70% sparsity.

E Generated Examples

We present generated examples from our pruned model using NutePrune at 20% sparsity. We provide examples of three types: without tuning (w/o tune), with tuning but without post-finetuning (w/ tune), and with tuning and post fine-tuning (w/ tune + post FT). The results are displayed in Table 18.

# Hidden Dim	4070							
Layer	1	2	3	4	5	6	7	8
# Head	23	22	30	22	29	27	30	28
# Intermediate Dim	5832	7820	9169	9187	8967	9163	9186	9112
Layer	9	10	11	12	13	14	15	16
# Head	30	30	31	30	32	27	30	30
# Intermediate Dim	9261	9165	9303	9695	10005	10258	10417	10564
Layer	17	18	19	20	21	22	23	24
# Head	30	29	26	25	23	21	16	21
# Intermediate Dim	10715	10759	10785	10790	10808	10778	10729	10707
Layer	25	26	27	28	29	30	31	32
# Head	14	15	6	7	11	8	7	9
# Intermediate Dim	10568	10366	10054	9403	8519	7297	6588	5164

Table 16: Detailed structure of compressed 20% LLaMA.

# Hidden Dim	4021							
Layer	1	2	3	4	5	6	7	8
# Head	24	21	27	18	26	21	20	20
# Intermediate Dim	3980	6216	7120	6590	5889	5731	5283	4944
Layer	9	10	11	12	13	14	15	16
# Head	25	21	27	26	23	21	25	21
# Intermediate Dim	4879	4563	4590	5040	5469	5832	6215	778
Layer	17	18	19	20	21	22	23	24
# Head	23	23	21	19	16	15	6	13
# Intermediate Dim	7706	8121	8005	8091	8232	7705	6879	6134
Layer	25	26	27	28	29	30	31	32
# Head	7	8	2	5	7	5	3	8
# Intermediate Dim	4912	3762	3107	2527	2357	2314	2931	2622

Table 17: Detailed structure of compressed 50% LLaMA.

Model	Generated Sentences
w/o Tune	Neural Sequential Model, especially transformers, are used to model the human brain. The model is based on the assumption that the brain is a system of interconnected modules, each of which is responsible for a specific function. The modules are connected by synapses, which are the synaptic connections between neurons. The synapsis are responsible for the transfer of information between the modules. The information is transferred in the form of a signal, which is a combination of the input signal and the output signal. The input signal is the signal that is received by the module, while the output is the output of the module. The output signal is then passed on to the next ...
	AI can create a logo in seconds . It can also create a logo that is more creative than a human designer. AI can also create logos that are more creative and unique than a human can. AI is also able to create logos in a short time. AI has the ability to create logotypes. AI will be able to create a logo for a company in a short period of time. A logo is a symbol that is used to identify a company. A logo can be used to identify the company. AI also has the ability of creating logos that can be used in different ways. AI creates logos that have ...
	What's great about the holiday season ? The holiday spirit, of course! But it's also a time for giving. And giving is what we do best at the Chamber. We give you the opportunity to give back to your community. We've got a great list of charitable organizations that are looking for your help. And we've made it easy for you to give. Just click on the link below and you'll be taken to the website of the charity of your choice. You can make a donation online, or you can mail a check. Either way, you're giving back to the community. And that's what the ...
w/ Tune	Neural Sequential Model, especially transformers, are used to model the human brain. The model is a neural network that is trained to recognize patterns in data. The network is trained by feeding it data and then testing it on new data. This is a form of machine learning. The neural network is a type of artificial neural network. The term neural network refers to a group of neurons that are connected together. The neurons are connected by synapses. The synapsis are the connections between the neurons. The connections are called weights. The weights are used by the network to determine how much influence each neuron has on the other neurons in the network ...
	AI can create a logo in seconds , but it can't design a logo that will work for a company. AI is great for creating a logo, but not for designing a logo. A logo is a visual representation of a company's brand. It's a visual symbol that represents the company' logo. It is a symbol that is used to identify a company and its products. A good logo is one that is memorable and easy to recognize. A great logo is memorably and easy recognizable. A bad logo is not memorable or easy to recognize. A poorly designed logo is also not memorably or easy recognized. A ...
	What's great about the holiday season is that it's a time for giving. And what better way to give than to give the gift of a new home? Whether you're buying a home for the first time or you've been in the market for a while, there are a few things you should know before you start looking. 1. Know your budget. This is the most important thing to consider when you'll be looking for a home. You'll want to know how much you can afford to spend on a home, and you'd want to make sure you don't go over your budget when you start searching for homes ...
w/ Tune + Post FT	Neural Sequential Model, especially transformers, is a popular model for deep learning. It is a model that can be trained on large datasets. It is used for tasks such as image classification, natural language processing, and other tasks. It is an effective model for tasks such as image classification and natural language processing. It is also used for tasks like natural language understanding and natural film fiction. It is able to learn from large datasets and can be trained for various tasks. It can be used for tasks that require a large amount of data. It is capable of learning from large datasets. It is used to train neural networks for tasks such image classification ...
	AI can create a logo in seconds. AI Logo Designer. A Logo AI Can Create. A logo is a key element of any business. It helps to create a unique identity and helps to stand out from the crowd. A good logo can help to build trust and credibility. It can also help to increase the sales. Alogoai.com is a AI logo design service. We can design logos for any business, with the latest technology. We provide high quality logos at affordable prices. Logo design service provided by Alogo AI. Logos designed by AI are more accurate and effective. Logs designed by ...
	What's great about the holiday season is the opportunity to give back to the community. Whether it's donating to a local charity or giving to a needy family, it'll make you feel good to know that you're making a difference. Here are some ideas to get started. Volunteer at a local organization. If you have some free time, consider volunteering at a nonprofit organization. You can help with the holidays by helping with the decorations, helping with food preparation, or even helping with some of the administrative tasks. You'll feel great knowing that you are making a contribution to the organization. Donate ...

Table 18: Generated Examples from the Compressed LLaMA-7B at 20% sparsity