Sparsity-Aware Prompt Tuning: A Simple and Effective Way to Fine-tune High-Sparsity LLMs

Yuxin Zhang[1], Weizhong Huang[1], Yuexiao Ma[1], Yunshan Zhong[1], Xiawu Zheng[1], Rongrong Ji[1]
[1]Xiamen University, China
Corresponding author: rrji@xmu.edu.cn

Pruning has recently demonstrated promising results in alleviating the heavy parameter burden and computational cost of Large Language Models (LLMs). However, the missing of sparsity-friendly fine-tuning significantly limits the performance of high-sparsity LLMs. While LoRA serves as the most popular fine-tuning approach for dense LLMs, it is naturally incompatible with unstructured sparsity since the merging operation condenses the weight matrix, thereby eliminating the benefits of sparsity. In this paper, we introduce **S**parsity-aware **P**rompt **T**uning (**SPT**), a simple and effective fine-tuning approach specifically tailored for sparse LLMs. Instead of fine-tuning the remaining weights or adding extra adaptors, SPT aims to learn soft prompts to compensate for pruned LLMs, enabling them to generate more desired content. Pruning occurs gradually during fine-tuning, with the prompt length proportional to the sparsity ratio assigned to each layer. This gradual imposition of pruning allows the output deviation caused by pruning to be efficiently mitigated through sparsity-aware prompt tuning. Experimental results demonstrate that SPT significantly enhances the performance of sparse LLMs across extensive model architectures, parameter sizes, and tasks, particularly at high sparsity ratios. For instance, fine-tuning an 80% sparse LLaMA-V2-13B produced by SparsGPT for just 2.5 hours, SPT improves the zero-shot performance from 47.39% to 55.27%, outperforming its LoRA baseline by 2.55%, while using only 6.5% of the trainable parameters compared to the latter. This will deliver a 3.14× end-to-end inference speed-up using the DeepSparse inference engine.

1. Introduction

Large Language Models (LLMs), characterized by parameters in the billions, such as GPT [1, 2] and LLaMA [3, 4], have emerged with transformative prominence in a wide array of language and multimodal tasks. Although the vast number of parameters in LLMs contributes to their superior performance, this immense scale also renders them inefficient and memory-intensive for deployment. For instance, the GPT-175B model [5] consumes 320G of memory when loading parameters in FP16 precision and necessitates no fewer than five A100-80G GPUs for inference [6]. To address this challenge, recent interest has surged towards the compression of LLMs through parameter quantization [7–14] and pruning [6, 15–19] in a bid to preserve performance while significantly reducing memory consumption and computational overheads.

This paper concentrates on the pruning of LLM parameters, *i.e.*, LLM sparsity, which receives increasing research and industrial efforts performance promise [6, 15, 17, 20] and increased support from practical hardware [21–23]. Recent developments in LLM sparsity have introduced various pruning metrics, such as the weight-activation-product [15] or the hessian-inversion [6] to identify unimportant weights that minimally impact the output. Additionally, it has been demonstrated that optimizing the layer-wise sparse budget can further aid in performance recovery [17, 19]. These remarkable advancements facilitate satisfying performance retention for compressing LLMs at moderate sparsity levels, *e.g.*, 50%. Nevertheless, such performance guarantee will quickly diminish as the sparsity rate increases, reaching random levels at 70% sparsity or beyond [17]. Enabling high sparsity is crucial due to its significant memory reduction and substantial speedup gains. For

Third Conference on Parsimony and Learning (CPAL 2026).

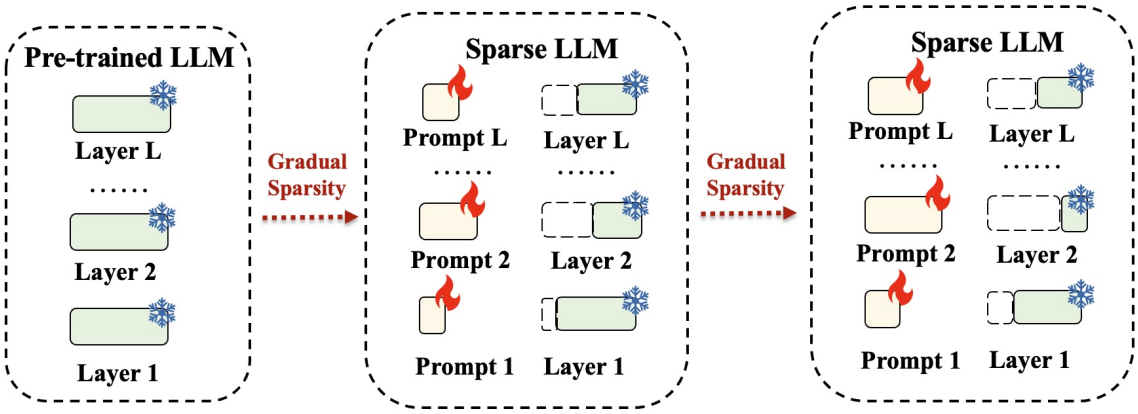

Figure 1: Illustration of our proposed Sparsity-aware Prompt Tuning (SPT). It imposes sparsity on the pre-trained LLMs in an incremental manner. With each escalation of the sparsity rate, a layer-wise learnable prompt is designated according to the quantity of weights rendered sparse in each respective layer. This progressive sparsity alongside fine-tuning enables the pre-trained LLM to effectively achieve the desired sparsity rate, while concurrently preserving the original performance.

instance, a speed-up spike from 1.77× to 4.15× can be fulfilled when increasing the sparsity level of LLaMA-7b from 50% to 90% [21]. Consequently, addressing the challenge of high-sparsity LLMs is a crucial task that requires resolution in both industrial and academic spheres.

Reflecting upon the historical progression of traditional network sparsity, there has been a wide-scale recognition that fine-tuning–*i.e.*, retraining the compressed model on the original training set–plays a pivotal role in recovering the performance of sparse models, especially at high levels of sparsity [24–26]. As such, it is intuitively appropriate to fine-tune sparse LLMs to address the aforementioned obstacle. However, two significant challenges remain formidable. First, full fine-tuning of sparse LLMs necessitates an exhaustive collection of training data and attendant computational burden, making it impractical in resource-constrained scenarios. On the other hand, while previous endeavors demonstrated encouraging performance using Low-Rank Adapter (LoRA) [27] for sparse LLM fine-tuning [15, 17], the merging operation of LoRA naturally conflicts with sparse tensors, eliminating its inference benefits.

To this end, we propose a simple and effective, Sparsity-aware Prompt Tuning (SPT), which is specifically tailored for fine-tuning sparse LLMs. SPT tackles the aforementioned obstacles through two complementary advancements: **(1) Progressive Sparsity Allocation:** SPT gradually introduces sparsity to LLMs in a step-by-step manner. This approach avoids the irreversible performance degradation caused by the simultaneous removal of excessively large weights. **(2) Sparsity-aware Prompt Refinement:** Following each pruning step, SPT enhances the output of sparse LLMs by incorporating sparsity-aware prompts into the input embeddings. The length of these prompts is designed to align with the layerwise sparsity levels. This incremental imposition of deviation in the output induced by sparsification allows for timely and efficient mitigation through sparsity-aware prompt tuning.

We embark on extensive experiments to evaluate the efficacy of SPT across a broad spectrum of representative LLMs, encompassing LLaMA-V1 [28], LLaMA-V2 [3], OPT [29], and Vicuna [30] with sizes varying from 7 billion to 70 billion parameters. Our results reveal that SPT significantly amplifies the performance of sparse LLMs, particularly at extreme sparsity, even reaching up to 90%. It is pertinent to highlight that SPT boasts remarkable efficiency, with the additional parameter burden being less than 0.1% of the original LLMs, such that the entire fine-tuning procedure can executed within very few hours. For an illustrative example, by fine-tuning LLaMA-V1-7B pruned by SparseGPT [6] at 70% sparse ratio, SPT reduces the perplexity evaluated on WikiText-2 from 27.00 to 11.15, using only two hours on a single NVIDIA A100 GPU, achieving 2.55× practical speedup evaluated on the NeuralMagic DeepSparse engine [21].

2. Related Work

2.1. LLM Sparsity

Parallel to conventional network sparsity [24, 31], the progression of LLM sparsity manifests in both structured and unstructured manner. The former can only attain a lower compression rate, and necessitate massive computing resources for re-training the compressed model [20, 32]. Thus, we concentrate on LLM unstructured sparsity which holds the performance at higher sparsity, which delivers practical speedups corroborated by the DeepSparse engine [21]. The key problem in unstructured LLM sparsity falls into acquiring the optimal sparse mask. For example, SparseGPT [6] amalgamates the Hessian inverse for pruning and subsequent weight adjustments, while Wanda [15] realizes a sparse LLM model via a criterion defined by the product of absolute weight values and their corresponding activations, intending to maintain outliers [11] that arise in LLMs. DSnoT [16] further optimizes the sparse mask through training-free mask updates reliant on reconstruction error. Despite these advancements, the question of efficiently fine-tuning sparse LLMs to enhance their performance has achieved limited progress. Sun *et al.*[15] made a preliminary attempt by utilizing LoRA fine-tuning, thus improving performance under a 50% sparsity rate. Xu *et al.* [18] proposed employing traditional prompt learning to fine-tune models combing quantization and sparsity. SPP [33] leverages learnable column and row matrices to optimize sparse LLM weights, also with moderate sparse levels. Differently, our proposed SPT in this paper embarks on the inaugural exploration into extreme-high LLMs sparsity via progressively learning layer-wise prompts in a incremental pruning manner.

2.2. Parameter-efficient Fine-tuning

LLMs, trained on extensive, heterogeneous data, typically necessitate fine-tuning for the transfer of accrued knowledge to a variety of downstream tasks. In contrast to full fine-tuning that consumes a lot of training resources, Parameter-Efficient Fine-Tuning (PEFT) freezes the majority of pre-trained model parameters, yet manifests comparable proficiency in downstream tasks in a cheap manner. Predominant PEFT strategies encompass Adapters [27, 34–38] and prompt-tuning [39–44]. Adapters endeavor to incorporate lightweight parameter modules to adjust the model's capability for handling downstream tasks. The Low-rank Adapter (LoRA) [27], a prime exemplar method, integrates trainable rank decomposition matrices into pre-trained weights, exhibiting impressive fine-tuning capacity across a multitude of LLMs and downstream tasks. Prompt tuning, alternatively, appends trainable prompt tokens to the input embeddings [40, 41] or intermediate layers [39, 42] of LLMs. This approach allows the model to assimilate bias information pertinent to the task, paving the way for efficient transfer learning on downstream tasks. In our work, we designed a novel per-layer incremental prompt that yielded the most substantial performance improvement for sparse LLMs when compared to LoRA and the traditional prompt tuning. We leave it to future work to explore the trade-offs inherent to other PEFT techniques.

3. Methodology

3.1. Revisiting LLM Sparsity

Initially, we review the problem formulation and advances in LLMs sparsity. The objective is to eliminate superfluous weights in pre-trained LLMs, thereby alleviating significant computational and parameter burdens. To this avail, extant research concentrates primarily on fabricating custom metrics, identifying and synchronously eliminating negligible weights [6, 15, 16], or administering layer-specific sparsity ratios to bolster performance [17, 19]. Regrettably, despite the efficacy of these methods in obtaining sparse LLMs in a matter of minutes, a swift performance deterioration transpires, with degradation plummeting to stochastic levels for sparsity ratios surpassing 70%.

This idiosyncrasy is well recognized within the traditional sparsity literature [24, 26, 45]: irrespective of the pruning strategy deployed, fine-tuning the sparse network on the original training data is

highly necessitated to recuperate its performance. For sparse LLMs, however, this prospect seems improbable owing to their voracious demands for training data and resources. Sun *et al.* [15] explored strategies such as fine-tuning with LoRA [27] or full fine-tuning, which moderately revives performance improvement, but nonetheless necessitates considerable computational resources and performs sub-optimally at high sparsity ratios. As a result, the conundrum of fine-tuning highly sparse LLMs in a parameter-efficient manner continues to pose a daunting challenge warranting urgent attention in the research area.

3.2. Incremental Sparsity

Through the above review of LLM sparsity research, we first highlight a commonality that weights are pruned in a one-shot fashion. This methodology, broadly adopted in conventional sparse models such as ResNet-50 [46] with 25.6M parameters, has indeed demonstrated effectiveness [24, 47, 48]. However, we posit that for LLMs, with their immense volume of parameters, it is nearly impossible to effectively recover from the pruning error caused by one-shot removal of parameters at the billion level. By way of illustration, applying 70% sparsity rate to LLaMA-V2-70B [3] equates to removing 49B weights in a single sweep - a pruning equivalent to expunging 20 entire ResNet-50. Therefore, restoring output discrepancy due to such quantity of weight elimination is a formidable task, even if the network undergoes full fine-tuning, needless to say pruning relatively unimportant weights solely by manual design metrics without re-training [6, 15].

Based on this analysis, we propose to replace the one-shot sparsity with incremental sparsity, a concept originating from traditional sparsity literature that is especially practical under conditions of high sparsity rates [26, 49]. In particular, the global sparsity is allocated in a progressive manner, with the initial sparsity S^i typically being 0% and eventually reaching the target sparsity S^f as follows:

$$S^t = S^f + \left(S^i - S^f\right)\left(1 - \frac{t}{T}\right)^3, \;\; t \in \{1,\;\; 2,\;\; ...,\;\; T\}\,, \tag{1}$$

where P^t is the sparsity rate at training step t in the fine-tuning phase and T is the total pruning step. Given the global sparsity ratio, we utilized OWL [17] to decide the layer-wise sparsity ratio $S^t = [S_1^t, S_2^t, ..., S_L^t]$. Adhering to this progressive sparsity schedule promotes a smoother invocation of weight removal. In other words, the target of fine-tuning is decomposed into errors precipitated by each incremental pruning, recovering a minor fraction of weights each time. We deem such incremental sparsity critically pivotal in fine-tuning sparse LLMs, given the magnitude of removed weights eclipsing that found in conventional scenarios, as previously alluded to.

Note that, one necessary phase involved in the incremental sparsity schedule is to retrain the sparse weights once upon weight removal. However, fully fine-tuning the sparse LLMs requires an extensive volume of training data and consequent computational load, therefore not applicable in consideration of resource-constrained scenarios.

3.3. Sparsity-aware Prompt Tuning.

We introduce sparsity-aware prompt tuning, which enables efficient fine-tuning of the sparse LLM in accordance with the gradual sparsity schedule. Given an incremental layer-wise sparsity ratio $S^t = [S_1^t, S_2^t, ..., S_L^t]$ at step t, we apportion learnable prompts $P^t = [P_1^t, P_2^t, ..., P_L^t]$ for each layer, where each $P_l^t \in \mathbb{R}^{K^t \times C}$, K^t depicts the prompt length, and C corresponds to the hidden dimension of transformer layers. The prompt length K^t assigned to each layer is proportional to the sparsity ratio S^t as per the equation

$$K_l^t = \lfloor \frac{S_l^t - S_l^{t-1}}{S_i^t - S_i^{t-1}} \cdot K_o \rfloor, \;\; i = \arg\min_j (S_j^t - S_j^{t-1}), \tag{2}$$

where $l = 1, 2, ..., L$ and K_o signifies the minimum prompt length, which we set as 5. In this manner, layers with significant weight removal receive a longer prompt, enabling efficient recovery of pruning loss.

With the l-th layer utilized as exemplification, we denote the input N-length tokens as $O_l \in \mathbb{R}^{N\times C}$. These serve as a representation of the input instruction and the previously generated response. The adaptable, learnable prompt is concatenated with O_l as $[P_l;\ O_l^t] \in \mathbb{R}^{(K_l+N)\times C}$. Assuming the model is in the process of generating the $(M+1)$-th output tokens on top of $[P_l;\ T_l^t]$ at the l-th layer, denoted as $I_l \in \mathbb{R}^{1\times C}$. The queries, keys, and values within the attention mechanism are transferred as indicated below:

$$Q_l = \mathrm{Linear}_{\mathrm{q}}(I_l\ ), \tag{3}$$

$$K_l, V_l = \mathrm{Linear}_{\mathrm{kv}}(\ [P_l;\ T_l^t;\ I_l]\ ), \tag{4}$$

where Linear_{qkv} denotes the Q,K,V layers within attention module with sparse weights. Subsequently, the attention scores and the layer output are computed as follows:

$$A_l = \mathrm{softmax}(Q_l K_l^T/\sqrt{C}), \tag{5}$$

$$O_l^o = \mathrm{Linear}_{\mathrm{o}}(A_l V_l), \tag{6}$$

where $Linear_o$ denotes the sparse linear projection layer and O_l^o denotes the output. During the incremental sparsity, we freeze all the sparsified weights but only optimize the introduced prompts. Thus, the insights acquired via P_l^t can efficaciously guide the original input embedding tokens to render precise outputs following weight elimination in LLMs.

Discussion. The adapter choice for fine-tuning sparse LLMs during incremental pruning may incorporate the Low-rank adapter (LoRA) [27] and Prompt tuning at the input embeddings [18]. We further explore, delineate, and juxtapose their attributes with the SPT we have proposed below.

LoRA has been substantiated as a potent tool in boosting the performance of sparse LLMs at moderate degrees of sparsity [15]. Albeit, the additional parameter load imposed by LoRA is roughly tenfold compared to that of the prompt. Further, the primary objective of LoRA is to simulate dense weight updating; however, even comprehensive fine-tuning fails to effectively revive performance, largely due to overfitting the limited training data [15]. Conversely, prompt tuning endows the model with a bias to facilitate the generation of accurate outputs following the sparsification of weights. At each progressive pruning timepoint, the newly introduced prompts can effectively be learned to compensate for the output deviation induced by weight removal, whilst concurrently preserving the pre-trained LLM weights from updates.

Prompt tuning at the input embeddings functions as a traditional PEFT practice to adapt LLMs for downstream tasks [40, 41]. Notwithstanding, the acquisition of the prompt before the initial layer falls short of efficaciously capturing each layer's individual pruning errors. This understanding finds solid backing from previous approaches underscoring the need for layer-wise reconstruction in sparse LLMs [6, 16]. Contrarily, our proposed SPT uniquely assigns learnable prompts at each layer in tandem with the sparsity degree, thus mitigating the pruning error in a more adaptive manner.

4. Experiment

In this section, we quantitatively examine the effectiveness of the proposed SPT for fine-tuning sparse LLMs. Initially, Section 4.1 summarily delineates the experimental settings. Subsequently, in Section 4.2, we conduct experiments for sparsifying representative LLMs to sweep the efficacy of SPT over a wide spectrum of sparse rates and benchmarks. Conclusively, we undertake performance analysis to ablate the components embodied in SPT.

5. Experiment

In this section, we quantitatively examine the effectiveness of the proposed SPT for fine-tuning sparse LLMs. Initially, Section 4.1 summarily delineates the experimental settings. Subsequently, in Section 4.2, we conduct experiments for sparsifying representative LLMs to sweep the efficacy of SPT over a wide spectrum of sparse rates and benchmarks. Conclusively, we undertake performance analysis to ablate the components embodied in SPT.

Table 1: Zero-shot performance comparison for fine-tuning LLMs at 70% sparsity. The average is calculated among seven zero-shot datasets. We adopt SparseGPT [6] as the pruning baseline.

Method	Wiki-2↓	HellaSwag	Winogrande	BoolQ	OBQA	PIQA	ARC-e	ARC-c	Average
LLaMA-V1-7B[50]	5.63	56.94	69.93	75.11	34.2	78.67	75.25	41.89	61.71
Baseline	27.54	34.58	56.43	64.80	16.8	64.25	45.24	23.12	43.60
w. LoRA	13.54	42.77	60.14	63.88	21.0	69.86	55.39	26.96	48.86
w. Prompt-tuning	16.61	40.17	60.30	58.38	**22.4**	68.06	53.24	26.96	47.07
w. SPT (ours)	**11.15**	**44.86**	**63.22**	**64.07**	20.8	**70.46**	**55.47**	**27.13**	**49.57**
Vicuna-13B[30]	5.98	76.5	79.8	76.1	70.1	72.8	47.6	57.2	68.59
Baseline	21.95	38.52	61.01	73.30	17.8	67.30	53.91	27.90	48.53
w. LoRA	10.37	47.09	64.80	71.77	25.4	73.67	65.66	32.85	54.46
w. Prompt-tuning	13.73	44.98	60.62	74.07	21.6	70.13	62.16	30.89	52.06
w. SPT (ours)	**9.15**	**49.03**	**65.19**	**76.82**	**28.2**	**72.96**	**66.25**	**35.15**	**56.23**
LLaMA-V2-13B[50]	4.88	60.06	72.22	80.55	35.2	79.11	79.42	48.46	65.00
Baseline	20.57	36.90	61.64	66.02	21.0	67.57	52.61	25.94	47.38
w. LoRA	10.07	47.21	64.01	66.51	24.8	72.80	63.05	30.63	52.72
w. Prompt-tuning	11.24	44.85	61.96	72.29	22.4	71.11	60.35	29.01	51.71
w. SPT (ours)	**8.72**	**50.17**	**67.01**	**70.86**	**27.0**	**73.88**	**64.81**	**33.19**	**55.27**
OPT-13B[29]	9.86	52.43	65.04	65.93	27.2	75.84	67.13	32.94	55.22
Baseline	20.26	40.94	61.40	62.94	21.0	69.10	52.44	25.94	47.68
w. LoRA	18.74	43.90	60.85	56.88	23.4	73.50	60.40	27.99	49.56
w. Prompt-tuning	21.46	43.36	61.40	59.85	24.0	72.47	59.85	27.30	49.74
w. SPT (ours)	**16.63**	**44.16**	**61.96**	**66.27**	**25.2**	**74.86**	**61.27**	**28.13**	**51.69**
LLaMA-V2-70B[50]	3.32	64.77	77.90	83.73	37.2	82.21	82.74	54.44	69.00
Baseline	9.46	50.98	75.45	80.06	30.0	75.24	73.57	40.61	60.84
w. LoRA	7.05	57.11	73.48	75.66	32.6	78.78	76.56	45.65	62.55
w. Prompt-tuning	9.01	55.43	72.14	73.12	30.8	77.01	75.54	41.43	60.77
w. SPT (ours)	**5.99**	**59.19**	**74.32**	**80.12**	**33.1**	**79.64**	**77.91**	**46.58**	**64.40**

5.1. Experimental settings

Foundation models and Evaluating benchmarks. We base our experiments on sparsifying the most representative open-source LLMs, including LLaMA-V1 [50], LLaMA-V2 [3], Vicuna [30], and OPT [29], with the parameter size ranging from 7 to 70 billion. Following prior studies [6, 7, 9, 10], we assess the performance of sparse LLMs in language modeling and zero-shot question-answering tasks. For language modeling, we report the perplexity of language generation experiments on separate validation sets derived from WikiText-2 [51]. For question answering, we test six tasks using lm-eval-harness [52] framework: BoolQ [53], RTE [54], HellaSwag [55], Winogrande [56], ARC Easy and Challenge [57], and OpenBookQA [58].

Implementation Details and Baselines. We primarily base our experiments on fine-tuning sparse LLMs obtained by SparseGPT, a state-of-the-art LLM pruning method that performs especially better at high sparse ratios. In addition, we also evaluate the efficacy of SPT upon other methods including Magnitude [24] and Wanda [15] to test its scalability. We compare SPT with representative PEFT methods that have been proven to be effective for sparse LLM fine-tuning including LoRA [15], Prompt-tuning [18], and SPP [33]. We implement SPT in PyTorch [59] and use the HuggingFace Transformers library [60] for handling models and datasets. In particular, we set the steps of incremental sparsity $T = 10$ and minimum prompt length $K_o = 5$. We fine-tune the sparse LLMs for a total 1000 training iterations, where we equally divided the total iterations into each progressive pruning step. The training set are construct from 400 samples containing 2048 token from the C4[61] dataset. All experiments are conducted on NVIDIA A100 GPUs with 80GB of memory.

5.2. Quantitative Results

Main Results. We first provide the quantitative results evaluated on various tasks of different methods when fine-tuning LLMs at a high sparsity rate of 70%, as detailed in Table 1. Regardless

Table 2: WikiText-2 perplexity performance of SPT for fine-tuning sparse LLaMA-V1-7B and LLaMA-V2-13B at varying sparsity rates. We adopt SparseGPT [6] as the pruning baseline.

	LLaMA-V1-7B					LLaMA-V2-13B				
Method	50%	60%	70%	80%	90%	50%	60%	70%	80%	90%
Baseline	7.20	10.40	27.00	167.55	3912.78	6.01	8.24	20.57	103.52	1321.03
w. LoRA	7.20	9.02	11.91	28.38	184.83	5.95	7.11	10.07	20.87	89.11
w. Prompt-tuning	7.17	9.13	16.61	45.63	1669.27	5.89	7.12	11.01	29.01	398.53
w. SPP	7.00	8.75	11.32	26.98	101.99	5.87	7.01	9.72	20.22	78.76
w. SPT (ours)	**6.85**	**8.02**	**11.15**	**22.13**	**79.70**	**5.85**	**6.67**	**8.72**	**14.64**	**50.13**

Table 3: Zero-shot performance comparison for fine-tuning LLMs at 60% structured sparsity. We adopt LLM-Pruner [20] as the pruning baseline.

Method	ARC	HellSwag	MMLU	TruthfulQA	Average
LLM-Pruner	39.2	67.0	24.9	40.6	42.9
w. LoRA	46.8	65.2	23.9	46.2	45.5
w. APT	45.4	71.1	**36.9**	46.6	50.0
w. SPT (ours)	**46.9**	**73.6**	36.7	**48.0**	**51.3**

of the perplexity in Wiki-2 or the zero-shot datasets' evaluations, our SPT has shown significant advantages. For instance, when fine-tuning the sparse LLaMA-V1-7B [28] pruned by SparseGPT, SPT reduces the perplexity from 27.54 to 11.15, whereas prompt-tuning only manages a reduction to 16.61. The same conclusion applies to other tasks, LLMs, and pruning methods[1]. Such results demonstrate the efficacy of our proposed layer-wise prompt learning that independently solve the pruning error within each layer, instead of only inserting the prompt at the first embedding. In comparison to LoRA, SPT consistently exhibits superior performance in LLMs with a range of 7$\sim$70B parameters, which only requires a fairly small amount of additional parameters as we shown in the next subsection. For example, by fine-tuning sparse LLaMA-V2-13B, SPT achieves 55.27% average zero-shot accuracy on seven tasks, while LoRA yields a lower 52.72%. Therefore, the performance advantages of SPT over other fine-tuning methods are evident.

Varying Sparsity Rates. The comparison results in Table 2 delineate the perplexity performance of fine-tuning LLaMA-V1-7B and LLaMA-V2-13B models across varying degrees of sparsity. We observe that SPT facilitates a substantial enhancement in performance for LLMs at all sparsity rates of 50% to 90%. Particularly, this improvement becomes increasingly evident as the sparsity level grows against SPP, LoRA, and prompt-tuning (*e.g.*, SPT decreases the perplexity of 90% sparse LLAMA-V2-13B from 1321.03 to 50.13, while SPP, prompt-tuning, and LoRA only yield 101.99, 398.53 and 89.11, respectively). These results demonstrate the effectiveness of SPT for fine-tuning sparse LLMs at high sparsity, mostly due to its advancement on replacing traditional one-shot removal of billion weights with incremental pruning guided by layer-wise prompt-tuning.

Structured Sparsity. Notwithstanding the previous results on unstructured sparsity, we extend our experiment to assess the performance of SPT on structured sparsity within LLMs. Table 3 provides a comparative analysis against SOTA LLM structured sparsity methods, including LLM-Pruner [20], APT [62]. Remarkably, SPT continues to exhibit superior performance in the realm of structured sparsity, achieving an average precision of 51.3 across four downstream tasks, surpassing LoRA and APT by margins of 1.3 and 5.8, respectively. This underscores SPT's superior generalizability when applied to sparsity at other granularities.

5.3. Efficiency Analysis

We proceed to conduct an efficiency analysis of our proposed SFT in comparison to other PEFT methods when fine-tuning sparse LLMs. Table 4 depicts the comparison while fine-tuning LLaMA-

[1]We also provide the quantitative results of applying SPT to fine-tune sparse LLMs pruned by other metrics including Wanda [15] and Magnitude [24] in the supplementary material.

Table 4: Efficiency Comparison of different methods when fine-tuning LLaMA-V1-7B at 70% sparsity. We adopt SparseGPT [6] as the pruning baseline.

Method	Wiki-2 Perplexity	Tuned Params	Training Time	Inference Latency(ms)	Inference Speedup	Throughput (tokens/s)
LLaMA-V1-7B	5.63	0	0	291.45	1.00×	3.43
Baseline	27.00	-	-	111.92	2.60×	8.93
w. LoRA	11.91	20M	4h	126.87	2.29×	7.88
w. Prompt-tuning	16.61	4M	4h	117.52	2.48×	8.50
w. SPP	11.32	19.6M	12h	111.92	2.60×	8.93
w. SPT (ours)	**11.15**	**1.3M**	**2.5h**	**114.16**	**2.55×**	**8.75**

V1-7B at 70% sparsity, where the inference latency, throughput are evaluated on the NeuralMagic's DeepSparse engine [21]. SPT maintains an absolute performance advantage in such high sparsity, outperforming LoRA, prompt-tuning, and SPP by 0.75, 5.46, and 0.17 PPL on WikiText-2, respectively. Despite such robust performance, SPT remarkably upholds highly efficient fine-tuning and deploying efficiency. Specifically, the additional parameter burden introduced by SPT and prompt-tuning is substantially less than that of LoRA and SPP, resulting in a significantly expedited fine-tuning speed (2.5 hours for SPT and 4 hours for LoRA). Such lightweight fine-tuning structure paves the way for SPT to achieve better deployment acceleration effects than LoRA on the DeepSparse engine [21] (2.55× and 2.29× speedup for SPT and LoRA, respectively). These findings galvanize the effectiveness of SPT as a plug-and-play method for compressing LLMs towards practical deployment.

5.4. Performance Study

In this part, we conduct performance analysis of SPT, which encompasses its specific modules and hyper-parameter settings on the impact of fine-tuning accuracy and effectiveness.

Ablation Studies. Firstly, we research every portion of SPT's contribution to the enhancement of fine-tuning performance, including the non-uniform layer-wise sparsity budget from OWL, and incremental sparsity. Table 5 exhibits the quantitative results for diminishing these compents from SPT or adding them to LoRA [27] and Prompt-tuning [18]. It can be referred to that both factors bring about performance improvement for other types of adapters including LoRA and prompt-tuning, validating the universality of our proposed fine-tuning paradigm. Importantly, SPT still maintains performance advantages under evenly fine-tuning setting, providing a powerful endorsement of our proposed sparsity-aware prompt learning's targeted advantage for LLMs sparsity.

Table 5: Ablation studies of our proposed SPT when fine-tuning LLaMA-V1-7B at 70% sparsity. We adopt SparseGPT [6] as the pruning baseline.

Method	Incremental Sparsity	Non-uniform Sparsity	Wiki-2 Perplexity
LoRA	✗	✗	13.54
LoRA	✓	✓	11.88
Prompt-tuning	✗	✗	16.61
Prompt-tuning	✓	✓	14.72
SPT	✗	✗	11.78
SPT	✗	✓	11.31
SPT	✓	✗	11.60
SPT	✓	✓	11.15

Schedule of Incremental Sparsity. Figure 2 (left) presents the impact of the frequency of applying sparsity and the variation in training iterations on fine-tuning performance. Sparse application once parallels traditional one-shot sparsity, leading to unsatisfactory performance due to the wholesale removal of massive weights as previously discussed. Conversely, incremental sparsity can lead to a notable enhancement in fine-tuning performance, with the requirement of training steps proportional to the frequency sparsity applied. Therefore, we opt for a setting of $T = 10$ and 1000 training steps towards both efficient and effective fine-tuning.

Prompt Length. Moreover, Figure 2 (right) explores the influence of differing prompt lengths on performance. Intuitively, an increasing prompt length tends to contribute towards performance

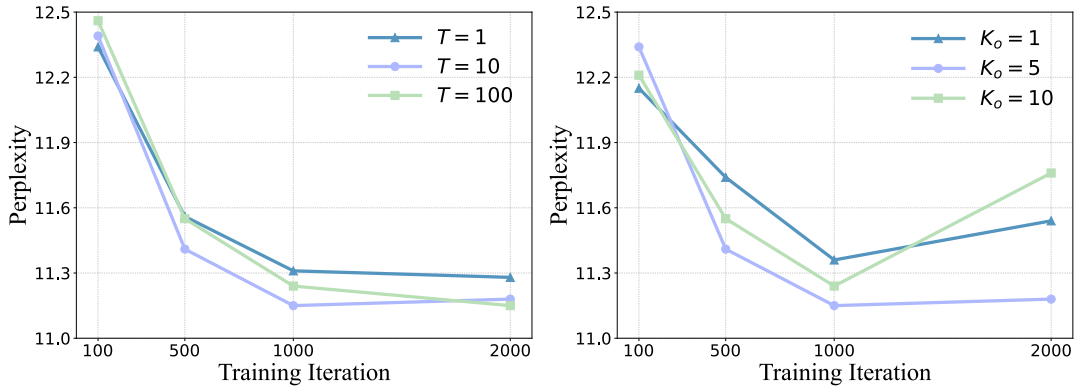

Figure 2: (**left**) Effect of the incremental sparsity schedule (T) and (**right**) minimum prompt length (K_o) of each layer when fine-tuning LLaMA-V1-7B at 70% sparsity.

advancement. However, it also necessitates larger number of training steps to sufficiently train the introduced prompts. Of note, endlessly augmenting the prompt length does not imply a perpetual performance gain, largely due to the over-fitting to fine-tuning data that does not fit the original pre-training scenario of LLM, *e.g.*, performance even drops when increasing training iterations from 1000 to 2000 for $P_o = 10$. Therefore, we moderately set P_o as 5 to attain a compromise between efficiency and performance.

6. Limitation

We further discuss several uncharted limitations of SPT, which warrant further investigation. Firstly, despite the potency and efficiency of the integrated layer-wise prompt learning, it provokes additional parameters and computational expenditure during the inference stage. Though these additional expenditures are minuscule compared to the original LLM, a potentially fruitful direction for future exploration could involve developing fine-tuning methods that introduce no ancillary parameters, harnessing techniques such as reparameterization [63]. Secondly, Subsequently, while SPT's supremacy over existing methods in fine-tuning highly sparse LLM is pronounced, yet it falls short of achieving an entirely lossless compression to the original model. A collective objective within the field to put greater emphasis on achieving lossless LLM sparsifycation at high ratios to enable more practical deployment.

7. Conclusion

A conspicuous bottleneck for LLMs resides in their overwhelming memory and computational burden. Network sparsity illuminates potential solutions to this limitation, albeit with disconcerting performance under high sparsity ratios. In this paper, we introduce SPT, a potent and efficient fine-tuning methodology designed to augment the performance of sparse LLMs through sparsity-aware prompt tuning. Our approach sequentially imposes sparsity on the pre-trained LLM, during which we add an incrementally learnable prompt to restore its performance each time pruning is instigated. Such prompt is executed in a layer-wise fashion, with its length proportional to the amount of pruned weights within each layer, therefore timely and precisely mitigate the output derivation at each pruning step. Extensive experimental validation demonstrated SPT's capacity to efficiently and significantly revitalize the performance of sparse LLMs. Fine-tuning an 80% sparse LLaMA-V2-13B produced by SparsGPT for just 2.5 hours, SPT is able to improve the zero-shot performance from 47.39% to 55.27%, outperforming LoRA by 2.55%, while using only 6.5% of the trainable parameters compared to the latter. The resultant sparse LLaMA-V2-12B will deliver a 3.14× end-to-end inference speed-up using the DeepSparse inference engine.

8. Acknowledgement

This work was supported by National Key R&D Program of China (No.2022ZD0118202), the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No. U21B2037, No. U22B2051, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, No. 62002305 and No. 62272401), and the Natural Science Foundation of Fujian Province of China (No.2021J01002, No.2022J06001).

References

[1] Chatgpt. `https://chat.openai.com`, 2023.

[2] OpenAI. Gpt-4 technical report, 2023.

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

[4] Meta. Llama3. `https://github.com/meta-llama/llama3`, 2024.

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

[6] Elias Frantar and Dan Alistarh. Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, 2023.

[7] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

[8] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[9] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training compression for generative pretrained transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

[10] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[11] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[12] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning (ICML)*, pages 38087–38099. PMLR, 2023.

[13] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[14] Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

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

[16] Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*, 2023.

[17] Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*, 2023.

[18] Zhaozhuo Xu, Zirui Liu, Beidi Chen, Yuxin Tang, Jue Wang, Kaixiong Zhou, Xia Hu, and Anshumali Shrivastava. Compress, then prompt: Improving accuracy-efficiency trade-off of llm inference with transferable prompt. *arXiv preprint arXiv:2305.11186*, 2023.

[19] Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. *arXiv preprint arXiv:2402.16880*, 2024.

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

[21] NeuralMagic. Deepsparse. `https://github.com/neuralmagic/deepsparse`, 2021.

[22] Vithursan Thangarasa, Abhay Gupta, William Marshall, Tianda Li, Kevin Leong, Dennis DeCoste, Sean Lie, and Shreyas Saxena. Spdf: Sparse pre-training and dense fine-tuning for large language models. In *Uncertainty in Artificial Intelligence*, pages 2134–2146. PMLR, 2023.

[23] Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. *arXiv preprint arXiv:2309.10285*, 2023.

[24] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems* (*NeurIPS*), pages 1135–1143, 2015.

[25] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pages 4340–4349, 2019.

[26] S Liu, DC Mocanu, ARR Matavalam, Y Pei, and M Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. arxiv. *arXiv preprint arXiv:1901.09181*, 2019.

[27] 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.

[28] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

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

[30] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org* (*accessed 14 April 2023*), 2023.

[31] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems* (*NeurIPS*), pages 598–605, 1989.

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

[33] Xudong Lu, Aojun Zhou, Yuhui Xu, Renrui Zhang, Peng Gao, and Hongsheng Li. Spp: Sparsity-preserved parameter-efficient fine-tuning for large language models. In *International Conference on Machine Learning* (*ICML*), 2024.

[34] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[35] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[36] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*, 2020.

[37] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

[38] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in Neural information processing systems*, 30, 2017.

[39] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[40] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[41] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

[42] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

[43] Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*, 2021.

[44] Renrui Zhang, Zhengkai Jiang, Ziyu Guo, Shilin Yan, Junting Pan, Hao Dong, Peng Gao, and Hongsheng Li. Personalize segment anything model with one shot. *arXiv preprint arXiv:2305.03048*, 2023.

[45] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations* (*ICLR*), 2019.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pages 770–778, 2016.

[47] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems* (*NeurIPs*), 33: 20744–20754, 2020.

[48] Yuxin Zhang, Mingbao Lin, Fei Chao, Yan Wang, Ke Li, Yunhang Shen, Yongjian Wu, and Rongrong Ji. Lottery jackpots exist in pre-trained models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (*TPAMI*), 2023.

[49] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. In *International Conference on Learning Representations Workshop (ICLRW)*, 2017.

[50] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

[52] Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 2021.

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

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

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

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

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

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

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8026–8037, 2019.

[60] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

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

[62] Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. Apt: Adaptive pruning and tuning pretrained language models for efficient training and inference. In *International Conference on Machine Learning (ICML)*, 2024.

[63] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021.

A. Appendix

Authors may wish to optionally include extra information (complete proofs, additional experiments and plots) in the appendix, which should be placed after bibliographies and submitted together with the main body of the paper. There will be no separate supplementary material submission. The main text should be self-contained; reviewers are not obliged to look at the appendices when writing their review comments.