Shears: Unstructured Sparsity with Neural Low-rank Adapter Search

J. Pablo Muñoz[*]
Intel Labs
pablo.munoz@intel.com

Jinjie Yuan[*]
Intel Corporation
jinjie.yuan@intel.com

Nilesh Jain
Intel Labs
nilesh.jain@intel.com

Abstract

Recently, several approaches successfully demonstrated that weight-sharing Neural Architecture Search (NAS) can effectively explore a search space of elastic low-rank adapters (LoRA), allowing the parameter-efficient fine-tuning (PEFT) and compression of large language models. In this paper, we introduce a novel approach called **Shears**, demonstrating how the integration of cost-effective sparsity and a proposed Neural Low-rank adapter Search (NLS) algorithm can further improve the efficiency of PEFT approaches. Results demonstrate the benefits of Shears compared to other methods, reaching high sparsity levels while improving or with little drop in accuracy, utilizing a single GPU for a pair of hours.

1 Introduction

Large language models (LLMs) exhibit impressive capabilities in comprehensive language understanding, as evidenced by their remarkable zero-shot generation across various tasks. However, supervised fine-tuning is often employed to unlock their true potential in real-world applications. Fine-tuning is essential for tailoring performance to domain-specific or proprietary data, bridging the gap between general language understanding and task-specific precision. Recently, parameter-efficient fine-tuning (PEFT) (Ding et al., 2022) has emerged as a crucial strategy for efficiently boosting the performance of LLMs in domain-specific tasks.

In addition to fine-tuning, increasing model parameters is another critical strategy to improve model performance. LLMs have produced impressive achievements as they scale to significant sizes, such as PaLM with 540 billion parameters (Chowdhery et al., 2022). The projection for future models suggests a continuous escalation in parameter count, anticipating improved performance. However, this trend also underscores the growing demands on computing devices. As model parameters increase, so does the computational complexity, necessitating more powerful hardware and infrastructure. In this context, the importance of model compression becomes particularly evident. Model compression is a crucial strategy to mitigate these challenges and make LLMs more accessible and deployable across a broader spectrum of devices.

Motivated by the significance of PEFT and model compression, this paper introduces a novel approach called **Shears**, showing the effective integration of PEFT and model compression to optimize the LLM performance with a high sparsity level. In the proposed methodology, we initiate the process by employing a zeroth-order sparse approach to induce sparsity in the LLM. Subsequently, we introduce elastic low-rank adapters into the sparsified model and apply Neural Low-rank adapter Search (NLS) to train a super-adapter network. Finally, a search algorithm is employed to identify the optimal adapter configuration. The contributions of this work can be summarized as follows:

1. We propose a practical solution combining model compression and PEFT, manifested in cost-effective sparsity and the proposed neural low-rank adapter search.

2. Our approach features three well-designed steps, i.e., unstructured sparsification, super-adapter training, and sub-adapter search. The proposed approach effectively obtains sparse fine-tuned LLMs that reduce inference time.

3. Experiments and ablation studies to confirm that our approach can produce models that maintain high accuracy while significantly increasing their sparsity levels.

[*]Co-first authors.

The content of this paper uses the following outline: Section 2 discusses the algorithms Shears uses. Section 3 describes the three stages and details of our practical solution. Section 4 presents results with several models on a variety of tasks. We offer concluding remarks in Section 5, and due to space limitations, we provide more details and a *Related Work* section in the Appendix.

2 Preliminaries

2.1 Sparsification

Our approach introduces sparsity into LLMs using a zeroth-order and cost-effective algorithm. In our experiments, we utilized the Wanda algorithm (Sun et al., 2023), which calculates weight importance based on weights, and activations and then leverages this information for unstructured pruning. Specifically, given a weight matrix $\boldsymbol{W}$ and input feature activations $\boldsymbol{X}$, Wanda computes the weight importance $\boldsymbol{S}$ as the element-wise product of the weight magnitude and the norm of input activations, formulated as follows:

$$\boldsymbol{S} = |\boldsymbol{W}| \cdot \|\boldsymbol{X}\|_2. \tag{1}$$

Wanda compares the weight importance scores within each row in $\boldsymbol{W}$. After obtaining the importance information, the algorithm zeroes out the less critical weights according to the specified sparsity level. The sparsification approach efficiently obtains a model with any level of unstructured sparsity desired before training.

2.2 Low-Rank Adaptation

Recently, PEFT technology has emerged as a solution to address the challenges of fine-tuning large-scale models. Among PEFT approaches, Low-Rank Adaptation (LoRA) (Hu et al., 2022) has shown notable efficacy in fine-tuning Transformer-based models for downstream NLP tasks. LoRA constraints the update for a pre-trained weight, $\boldsymbol{W}_0 \in \mathbb{R}^{d\times k}$, by a low-rank decomposition $\boldsymbol{W}_0 + \Delta\boldsymbol{W} = \boldsymbol{W}_0 + \boldsymbol{B}\boldsymbol{A}$, where $\boldsymbol{B} \in \mathbb{R}^{d\times r}$, $\boldsymbol{A} \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$. Throughout the training process, $\boldsymbol{W}_0$ remains frozen and does not undergo gradient updates, while only the parameters of $\boldsymbol{A}$ and $\boldsymbol{B}$ are trained. For the linear projection, $\boldsymbol{H} = \boldsymbol{W}_0\boldsymbol{X}$, the forward pass with LoRA is formulated as follows:

$$\boldsymbol{H} = \boldsymbol{W}_0\boldsymbol{X} + \Delta\boldsymbol{W}\boldsymbol{X} = \boldsymbol{W}_0\boldsymbol{X} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{X}, \tag{2}$$

where $\boldsymbol{A}$ is initialized with a random Gaussian while $\boldsymbol{B}$ is initialized with zeros, ensuring $\Delta\boldsymbol{W} = \boldsymbol{B}\boldsymbol{A}$ is zero at the beginning of training. Inspired by this approach, this paper integrates elastic LoRA adapters into Neural Architecture Search.

3 Methodology

In this section, we delve into the proposed approach, Shears. Figure 1 illustrates the overview of the Shears pipeline. As depicted in the figure, the method comprises three key steps: i) Unstructured Sparsification, ii) Super-Adapter Training, and iii) Sub-Adapter Search. Through these steps, the model undergoes sparsification and neural low-rank adapter search while preserving a performance comparable to the fine-tuned model from the original model. Next, we discuss the details of each step.

3.1 Unstructured Sparsification

As illustrated in step 1 of Figure 1, Shears employs a sparsification metric to zero out the less essential weights of the given LLM. As mentioned in Section 2.1, we apply the Wanda algorithm (Equation 1) in our main experiments. However, in theory, Shears could utilize other algorithms, e.g., movement sparsity (Sanh et al., 2020) or SparseGPT (Frantar and Alistarh, 2023). The pruned weights $\boldsymbol{W_p}$ are kept frozen throughout the subsequent stages of the overall pipeline. In this step, we factor in the cost of obtaining the weight importance structure. When using Wanda, only a tiny subset of inputs needs to forward pass to get the unstructured importance measurements instead of more sophisticated approaches that require weight updates and training iterations. The reader can find further details about the Wanda algorithm in its paper (Sun et al., 2023). Obtaining $\boldsymbol{W_p}$ for a model with seven billion parameters takes less than five minutes on a single GPU, as utilized in our experiments.

3.2 Super-Adapter Training

Subsequently, a weight-sharing super-adapter network is generated using the space of low-rank adapters. Shears does not make the original model weights $\boldsymbol{W}$ elastic as opposed to the elastic configurations of the adapters. The super-adapter network is then fine-tuned for a particular task through Neural Low-Rank Adapter Search (NLS), which we discuss next.

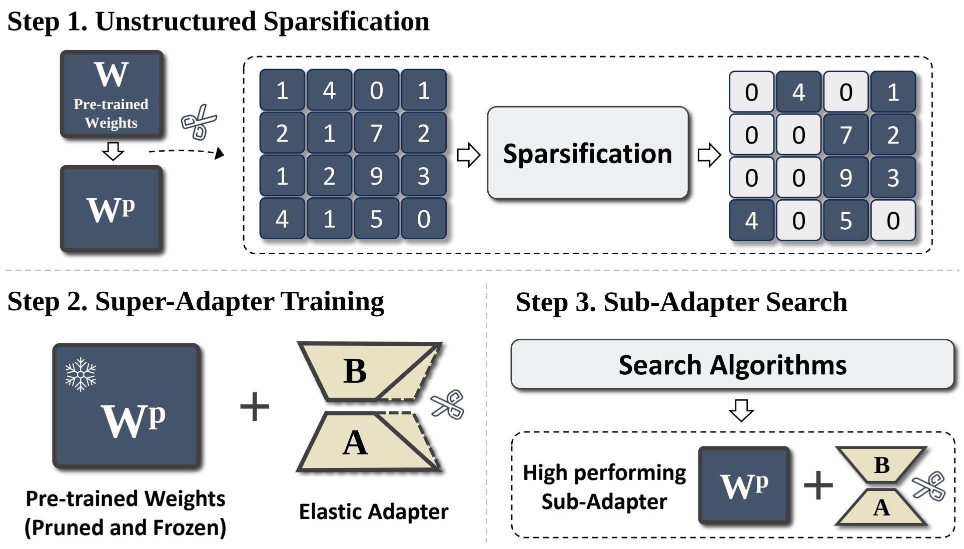
Step 1. Unstructured Sparsification
W
Pre-trained Weights
Wp
1 4 0 1
2 1 7 2
1 2 9 3
4 1 5 0
Sparsification
0 4 0 1
0 0 7 2
0 0 9 3
4 0 5 0
Step 2. Super-Adapter Training
Wp
+
B
A
Pre-trained Weights (Pruned and Frozen)
Elastic Adapter
Step 3. Sub-Adapter Search
Search Algorithms
High performing Sub-Adapter
Wp
+
B
A

Figure 1: Shears workflow. Initially, Shears employs a zeroth-order pruning algorithm to induce sparsity in the given LLM. Subsequently, the framework generates a super-adapter network trained by activating subnetworks within the search space of elastic adapters. Finally, Shears yields sub-adapter networks that exhibit high performance.

Neural Low-Rank Adapter Search (NLS) An elastic low-rank adapter can have numerous possible configurations. NLS leverages the mechanisms inherited from Neural Architecture Search (NAS) to activate adapter configurations and proceed with the forward and backward passes to fine-tune the possible sub-adapters. After fine-tuning the super-adapter network, which takes a pair of hours in a single GPU (further details in Section 4), Shears discovers a configuration that yields comparable accuracy on the target task.

3.3 Sub-Adapter Search

Identifying an optimal sub-adapter configuration can be an expensive endeavor. Although the search space of elastic adapter configurations is significantly smaller than if we also include subnetworks derived from the pre-trained weights of the LLM, the number of possible configurations for the adapters is still considerable. Sampling and evaluating these configurations can demand a significant amount of time. We can employ several approaches to explore search spaces of neural network configurations, such as evolutionary search using the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002) or a variation like RNSGA-II (Deb and Sundar, 2006). However, the cost of this type of search in LLM is prohibitive. To address this, we employ two alternatives. First, we extract a sub-adapter configuration using a heuristic. Then, suppose the performance of this configuration falls short of the desired outcome, a well-designed hill-climbing algorithm can be utilized to search for better configurations. Concretely, Shears can initiate a hill-climbing algorithm from the sub-adapter configuration found with the heuristic to explore its neighborhood and discover potentially improved configurations. This search approach is significantly less expensive than other search strategies, e.g., evolutionary search. Formally, the heuristic strategy, initially proposed in LoNAS (Muñoz et al., 2024a), to obtain a reference subnetwork configuration approximately at the center of the search space is as follows:

$$\textbf{Shears-Heuristic}_{l_i} \leftarrow \textbf{Shears-Maximal}_{l_i}[c], \text{ s.t. } c = \left\lfloor \frac{n}{2} \right\rfloor, \tag{3}$$

where c represents the index of the elastic width (rank of the adapter) configuration for the adapter l_i, chosen from a total of n possible elastic configurations at that adapter. This heuristic provides a (weak) indication of the performance of smaller sub-adapter networks.

4 Experiments

Shears is implemented by extending BootstrapNAS (Muñoz et al., 2022) and OpenVINO's Neural Network Compression Framework[1]. We explore the benefits of Shears by generating and fine-tuning super-adapter networks for various LLMs. The following sections detail our experimental setup and the analysis of the results.

[1]https://github.com/openvinotoolkit/nncf

Table 1: Sparsity and test accuracy (%) comparison of Shears with other LLM-Adapter approaches. The baseline results are those reported by Hu et al. (2023). Shears models have high accuracy while significantly increasing model sparsity.

LLM	Method	Sparsity	Datasets | Accuracy(%)				Average
			GSM8K	AQuA	MAWPS	SVAMP	
GPT-3.5	Zero-shot CoT	-	56.4	38.9	87.4	69.9	70.4
LLaMA$_{7B}$	Prefix	-	24.4	14.2	63.4	38.1	35.0
	Series	-	33.3	15.0	77.7	**52.3**	44.6
	Parallel	-	35.3	18.1	82.4	49.6	46.4
	LoRA	-	**37.5**	18.9	79.0	52.1	**46.9**
	Shears	**40%**	36.8	19.7	**83.2**	47.7	**46.9**
	Shears	**50%**	36.1	**22.0**	78.6	44.5	45.3
LLaMA$_{13B}$	Prefix	-	31.1	15.7	66.8	41.4	38.8
	Series	-	44.0	**22.0**	78.6	50.8	48.9
	Parallel	-	43.3	20.5	81.1	**55.7**	50.2
	LoRA	-	47.5	18.5	**83.6**	54.6	51.1
	Shears	**40%**	**48.3**	21.3	83.2	55.2	**52.0**
	Shears	**50%**	45.1	**22.0**	83.2	53.3	50.9

4.1 Experimental Setup

Datasets. Following the work of LLM-Adapters (Hu et al., 2023) [2], we assess the performance of Shears across a diverse range of tasks, including four math reasoning datasets (GSM8K (Cobbe et al., 2021), AQUA (Ling et al., 2017), MAWPS (Lan et al., 2022) and SVAMP (Patel et al., 2021)) and eight commonsense reasoning datasets (BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC (Clark et al., 2018) and OBQA (Mihaylov et al., 2018)). Leveraging GPT-3.5, the LLM-Adapters team generated high-quality, unified datasets for training while compiling several math or commonsense datasets. Additionally, we conduct evaluations of Shears on the original GSM8K training dataset for comparison with the work of Kurtic et al. (2023).

Models. We validate our approach using the LLaMA-series (Touvron et al., 2023) and MPT-series (MosaicML, 2023) language models. Specifically, we generate Shears super-adapter networks from LLaMA$_{7B}$[3], LLaMA$_{13B}$[4], and MPT$_{7B}$[5]. LLaMA (Touvron et al., 2023) models are popular autoregressive text generation models that have obtained outstanding results compared to larger language models. MPT (MosaicML, 2023) is an open-source model developed to get similar performance as LLaMA but available for commercial use.

Baselines. We compare Shears against PEFT approaches like Prefix (Li and Liang, 2021), Series (Houlsby et al., 2019), Parallel (Pfeiffer et al., 2020), and LoRA (Hu et al., 2022), using their results reported by LLM Adapters (Hu et al., 2023). In the case of the GSM8K dataset, we also compare Shears against the results obtained by Kurtic et al. (2023), which uses full fine-tuning.

More details about the experiment implementation are included in Appendix B.

4.2 Comparison to LLM-Adapters

4.2.1 Math Reasoning

Table 1 shows the comparison of Shears with various adapter approaches. We fine-tune the sparsified super-adapter network in this experimental scenario utilizing the 10K unified math dataset. Then, the test accuracy on four math reasoning test datasets of the heuristic subnetwork is reported. As shown in the table, Shears successfully generates subnetworks with higher sparsity levels while demonstrating improvements or marginal drops in accuracy. At a sparsity level of 40% [6] for LLaMA$_{7B}$, Shears shows performance comparable to PEFT approaches without sparsity. Meanwhile,

[2]https://github.com/AGI-Edgerunners/LLM-Adapters
[3]https://huggingface.co/yahma/llama-7b-hf
[4]https://huggingface.co/yahma/llama-13b-hf
[5]https://huggingface.co/mosaicml/mpt-7b

[6]The actual sparsity is marginally lower than the value in the table (approximately less than 0.5%), attributed to the introduction of additional parameters for the adapter.

Table 2: Sparsity and test accuracy (%) comparison of Shears with other LLM-Adapter approaches on commonsense reasoning datasets. The result of zero-shot[1] is derived from Touvron et al. (2023), and the result of zero-shot[2] is from LLM-Pruner (Ma et al., 2023). LLM-Pruner employs prompts different from those used by Touvron et al. (2023) for zero-shot evaluation since they do not provide the prompts they used. Almost all results of the PEFT baselines are obtained from Hu et al. (2023), except for the LoRA baseline in the 15k train dataset, which we experimented with the official implementation.

LLM	Train Set Size	Method	Sparsity	Datasets | Accuracy(%)								Average
				BoolQ	PIQA	SIQA	HellaSwag	WinoG	ARC-e	ARC-c	OBQA	
GPT-3.5	-	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA_{7B}	-	Zero-shot[1]	-	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2	66.1
	-	Zero-shot[2]	-	73.2	78.4	32.9	73.0	67.0	67.5	41.4	42.4	59.5
	15k	LoRA*	-	62.6	75.3	67.9	52.9	58.6	**79.2**	58.3	71.2	65.8
		Shears	**40%**	**65.5**	**76.0**	**71.2**	**56.8**	**65.6**	79.0	**62.6**	**76.4**	**69.1**
		Shears	**50%**	62.5	75.7	69.7	54.8	65.7	75.1	59.5	72.6	66.9
	170k	Prefix	-	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
		Series	-	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
		Parallel	-	67.9	76.4	**78.8**	69.8	**78.9**	73.7	57.3	75.2	72.3
		LoRA	-	**68.9**	**80.7**	77.4	78.1	78.8	**77.8**	61.3	74.8	74.7
		Shears	**40%**	67.0	79.9	76.6	**80.1**	78.6	76.9	**62.3**	**77.8**	**74.9**
		Shears	**50%**	67.3	79.1	77.5	73.3	77.7	74.4	57.9	72.8	72.5

for LLaMA_{13B}, a higher sparsity level of 50% can be attained while maintaining satisfactory performance. It is noteworthy that with a sparsity of 40%, Shears outperforms all PEFT approaches, even surpassing their performance in the absence of any sparsity.

4.2.2 Commonsense Reasoning

To further understand Shears' generalizability to other tasks, we fine-tune LLaMA_{7B} using the unified commonsense dataset from LLM-Adapters (Hu et al., 2023) using subsets of 15k and 170k samples and evaluate Shears' models at different levels of sparsity on commonsense reasoning datasets. As shown in Table 2, at 40% sparsity, Shears obtains models that outperform the baselines, and at 50% sparsity, it obtains competitive models, demonstrating the benefits and generalizability of the proposed approach.

4.3 Comparison to Full Fine-Tuning: MPT with GSM8K

In addition to comparing with other PEFT methods, we conducted experiments to compare Shears and full fine-tuning. We conduct experiments on a single math reasoning dataset, the GSM8K dataset (Cobbe et al., 2021), generating the MPT_{7B} super-adapter network. GSM8K can be challenging to

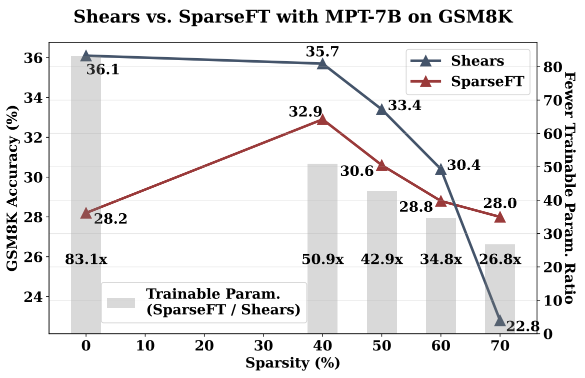

Figure 2: Comparison of Shears and *Sparse Fine-tuning* (SparseFT) (Kurtic et al., 2023) on the GSM8K test dataset.

LLMs that have not been fine-tuned for this particular task. Figure 2 shows a comparison of Shears and recent work by Kurtic et al. (2023), *Sparse Fine-Tuning* (SparseFT). This work employs SparseGPT (Frantar and Alistarh, 2023) and full fine-tuning using a novel knowledge distillation strategy. In the case of Shears, we adopt a more efficient approach leveraging unstructured sparsity and only fine-tuning the elastic adapters, which means that Shears incorporates fewer trainable parameters, reducing training and memory costs. SparseFT uses FP32 precision for tuning the whole model weights and employs a knowledge distillation strat-

Table 3: Comparison of non-zero parameters. Acc. represents the average accuracy across all math test datasets.

LLM	Method	Sparsity	Accuracy(%)	Non-zero Param.
LLaMA$_{7B}$	LoRA	-	46.9	6.7B
	Shears	50%	45.3	**3.5B**
LLaMA$_{13B}$	LoRA	-	51.1	13.0B
	Shears	50%	50.9	**6.7B**

egy with a more knowledgeable teacher. At the same time, Shears utilizes FP16 precision for pre-trained weights, and the training process does not involve knowledge distillation. As shown in the figure, our approach, Shears, outperforms SparseFT across sparsity levels from 0% to 60%, which indicates that Shears produces models with similar sparsity but higher accuracy. However, at a sparsity level of 70%, SparseFT yields higher accuracy but involves the high cost of fine-tuning all the weights in the original model.

4.4 Benefits of Sparse Models

Table 3 shows the benefits of the high-performing models within the Shears super-adapter network. Shears obtains a model with 50% sparsity that contains $1.91\times$ fewer non-zero parameters with minor drops in accuracy. Notably, the model from Shears maintains the adapters unmerged, while the vanilla LoRA adapters are merged with the original model. Since the bulk of the model sparsity is concentrated in the frozen weights, combining the adapters will reduce the sparsity levels. Furthermore, benefiting from sparsity, Shears still exhibits notable inference acceleration while maintaining accuracy or experiences only a marginal decrease compared to the vanilla LoRA.

4.5 Ablation Studies

Tables 4 and 5 illustrate the test accuracy comparison for ablation studies conducted on diverse methods, considering sparsity and various LLMs. The findings indicate that LLaMA$_{7B}$ and MPT$_{7B}$ can only effectively handle the challenging downstream datasets with fine-tuning, emphasizing the pivotal role of fine-tuning in these tasks. In the supervised fine-tuning setup, Shears demonstrates some benefits, whether applied to models with or without sparsity. Specifically, LoRA and Shears perform similarly in the experimental group without sparsity. However, with 50% sparsity, Shears outperforms LoRA significantly, highlighting its efficacy in enhancing model performance under sparsity conditions. This observation underscores that for sparsified models, employing Shears allows for a more substantial maximization of model performance in the supervised fine-tuning setup.

4.6 Sub-Adapter Configuration Search

Table 6 demonstrates the accuracy range of the search space of sub-adapter configurations. Since the sparsified weights of the model remain frozen, the search for the configuration of the attached adapter in Shears is significantly smaller than the search space in general neural architecture search. Studies indicate a narrow accuracy range, with the difference in accuracy between the minimal and the maximal sub-adapter configuration being only a single accuracy percentage point. The heuristic obtained in O(1) already gives us a reliable indication of the quality of the sub-adapters around the mid-configuration space. If the user has the budget, a more refined sub-adapter configuration can be searched using a cost-effective hill-climbing strategy that is cheaper than other methods, e.g., evolutionary search with RNSGA-II.

5 Conclusion

This paper presents **Shears**, a practical and novel solution for real-world applications to sparsifying weight-sharing super-networks of elastic adapters (super-adapters). By incorporating elastic LoRA adapters into the sparsified base model, Shears can fine-tune LLMs without sacrificing the sparsity obtained from the original model weights and produces sparse models with improvements or minor drops in accuracy and a fraction of the cost compared to other approaches. The increase in sparsity can result in significant speedup when using runtimes that take advantage of these patterns. Ablation studies show that combining sparsified models with elastic low-rank adapters yields better results than using LoRA adapters alone. Models and code are available at https://github.com/IntelLabs/Hardware-Aware-Automated-Machine-Learning.

Ethical Considerations and Limitations

The significant size of recent large language models has brought challenges for fine-tuning and deployment. Users with proprietary data must spend

Table 4: Ablation studies for LLaMA$_{7B}$. For a fair comparison, all ablation experiments with LoRA and NLS tuning applied the same adapter target modules (**Q**, **K**, **V**, **Up**, and **Down**).

Method	**Sparsity**	**Datasets | Accuracy(%)**				**Average**
		GSM8K	**AQuA**	**MAWPS**	**SVAMP**	
LLaMA$_{7B}$:						
w/o tune	-	11.0	24.8	3.4	2.9	10.5
w/ LoRA tune	-	**37.5**	**18.9**	79.0	**52.1**	46.9
w/ NLS tune (**Shears** w/o sparsity)	-	37.3	18.5	**82.8**	49.4	**47.0**
Pruned LLaMA$_{7B}$:						
w/o tune	50%	2.5	8.7	13.0	6.5	7.7
w/ LoRA tune	50%	33.8	18.1	**79.0**	42.3	43.3
w/ NLS tune (**Shears**)	50%	**36.1**	**22.0**	78.6	**44.5**	**45.3**

Table 5: Ablation studies for MPT$_{7B}$. Experiments with LoRA and NLS tuning applied the same adapter target modules (**Q**, **K**, **V**, **O**, **Up**, and **Down**). Shears outperforms LoRA with and without the sparsification step.

Method	**Sparsity**	**Test Accuracy**
MPT$_{7B}$:		
w/o tune	-	2.7
w/ LoRA tune	-	35.5
w/ NLS tune (**Shears** w/o sparsity)	-	**36.1**
Pruned MPT$_{7B}$:		
w/o tune	40%	2.9
w/ LoRA tune	40%	33.0
w/ NLS tune (**Shears**)	40%	**35.7**
w/o tune	50%	2.4
w/ LoRA tune	50%	31.8
w/ NLS tune (**Shears**)	50%	**33.4**

considerable time and resources adjusting LLMs' weights to improve their performance on custom tasks. In a world with limited resources, it is an ethical concern to find approaches that reduce the requirements of training and fine-tuning LLMs. Although Shears significantly reduces this process's requirements, more work is needed to address this issue. There is also the need for more research on the inherent limitations of LLMs. Their results and decisions should be carefully audited when they can affect customers' lives, who might need to be made aware of the depths and gaps in understanding that LLM researchers still have. Our goal is to make these models more efficient. However, efficiency is not the end of the story, and the above limitations should be considered when using or sharing LLMs.

Table 6: Comparison of various sub-adapter networks and the method used to obtain them from the LLaMA$_{7B}$ + Shears super-adapter network. Accuracy represents the average accuracy across all math test datasets.

Method	**Sparsity**	**Sub-Adapter**	**Accuracy (%)**
LoRA	-	-	46.9
Shears	50%	Maximal	44.5
		Heuristic	45.3
		Hill-climbing	**45.9**
		RNSGA-II	45.7
		Minimal	43.5

Acknowledgments

We are grateful to Michael Beale from Intel Labs, who helped us set up the infrastructure for sharing our models during the review stage and the final release and guided us through open-sourcing our compressed models. We also thank the anonymous reviewers for their insightful suggestions, which helped us improve the paper.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2020. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek

Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

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. In *NAACL*.

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*, abs/1803.05457.

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

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Kalyanmoy Deb and J. Sundar. 2006. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 635–642, New York, NY, USA. Association for Computing Machinery.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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

Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Eldar Kurtic, Denis Kuznedelev, Elias Frantar, Michael Goin, and Dan Alistarh. 2023. Sparse finetuning for inference acceleration of large language models. *arXiv preprint arXiv:2310.06927*.

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2022. Mwptoolkit: an open-source framework for deep learning-based math word problem solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 13188–13190.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*.

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. In *Conference on Empirical Methods in Natural Language Processing*.

NLP Team MosaicML. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

J. Pablo Muñoz, Nikolay Lyalyushkin, Yash Akhauri, Anastasia Senina, Alexander Kozlov, and Nilesh Jain. 2022. Enabling nas with automated super-network generation. In *Practical Deep Learning in the Wild, AAAI*.

J. Pablo Muñoz, Jinjie Yuan, Yi Zheng, and Nilesh Jain. 2024a. Lonas: Elastic low-rank adapters for efficient large language models. In *The 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

J. Pablo Muñoz, Yi Zheng, and Nilesh Jain. 2024b. EFTNAS: Searching for efficient language models in first-order weight-reordered super-networks. In *The 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

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

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

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*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. 2023. Neural architecture search: Insights from 1000 papers.

Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas S. Huang, Xiaodan Song, Ruoming Pang, and Quoc V. Le. 2020. Bignas: Scaling up neural architecture search with big single-stage models. *CoRR*, abs/2003.11142.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. 2023. Autopeft: Automatic configuration search for parameter-efficient fine-tuning. *arXiv preprint arXiv:2301.12132*.

A Related Work

Neural Architecture Search (NAS) Given a set of possible deep learning architecture configurations, a NAS algorithm discovers high-performing configurations. They often update the model weights, yielding a trained model ready for deployment. Research in NAS has increased dramatically in the past few years (White et al., 2023), making these techniques highly popular with practitioners engaged in model optimization and compression. One-shot weight-sharing neural architecture search has been demonstrated to be a practical class of NAS algorithms with savings in memory and additional storage since they construct a super-network that contains a large number of subnetworks (Cai et al., 2020; Yu et al., 2020). In this case, the objective of the NAS algorithm is to train and identify an outstanding subnetwork, frequently representing a compressed version of the original model. Our approach, Shears, differs from traditional NAS in that we do not attempt to find a better, more efficient neural architecture using the original model as a reference. Shears freezes the original model and attaches elastic low-rank adapters, directing the NAS mechanisms only to these adapters, termed neural low-rank adapter search (NLS).

Elastic Adapters PEFT (Ding et al., 2022) has become a popular method for fine-tuning large models. Recently, there has been work on making the adapters in PEFT elastic, aiming to find the optimal adapter configuration through a search process. AutoPEFT (Zhou et al., 2023) automatically applies elastic serial adapters, parallel adapters, and prefix-tuning into the small language model like BERT to identify the optimal adapter class and its configuration within these elastic modules. LoNAS (Muñoz et al., 2024a) introduces elasticity to the low-rank adapters and pre-trained weights in LLM, enabling them to adopt various configurations. This feature effectively generates a search space conducive to exploring using weight-sharing neural architecture search (NAS). In our approach, Shears only makes the LoRA adapters of the sparsified model elastic, ingeniously combining both model sparsification and elastic adapters to elicit optimal performance in the sparsified model.

Sparsity and Pruning Pruning the weights of a neural network is a popular technique for model compression. The most common approach of element-wise pruning uses the *magnitude* of the weights and a thresholding function that zeroes out the weights below a threshold. Weight *magnitude pruning* is ineffective when applied to LLMs (Frantar and Alistarh, 2023). One possible reason is the existence of outlier features when models reach several billion parameters (Dettmers et al., 2022). Alternative approaches have been proposed to measure the importance of the weights. For instance, first-order approaches use several iterations to update the weights, e.g., Movement Pruning (Sanh et al., 2020) and SparseGPT (Frantar and Alistarh, 2023). These approaches have also improved weight-sharing NAS (Muñoz et al., 2024b). Unfortunately, using weight updates for LLM pruning requires a significant computational cost. Recently, efficient approaches have been proposed to achieve high degrees of sparsity with a single forward pass of N samples. For example, Wanda (Sun et al., 2023) is a simple but effective sparsification method that determines which parameters to zero out by the importance of weights based on both the weights and the activations. LLM-Pruner (Ma et al., 2023) is proposed to compress LLMs in a task-agnostic manner (Ma et al., 2023). This approach produces good zero-shot results after applying structured pruning on the targe LLM. Unlike these approaches, Shears is designed for specific task fine-tuning scenarios, which can obtain higher levels of unstructured sparsity while improving or with minor drops in accuracy by combining unstructured sparsity with neural low-rank adapter search (NLS).

Sparsity and Fine-Tuning SparseFT (Kurtic et al., 2023) uses SparseGPT (Frantar and Alistarh, 2023) to sparsify the model and then fine-tunes all the weights of the model using a novel knowledge distillation technique (see section 4.3). Unlike SparseFT, Shears does not use knowledge distillation and fine-tunes only a tiny set of weights in elastic low-rank adapters. Our approach necessitates updating only a fraction of the total parameters, thereby reducing memory and computing demands during training while enhancing accuracy.

B Hyperparameters

The hyperparameters of our approach under different LLMs are listed in Table 7, Table 8, and Table 9.

Table 7: Hyperparameters for LLaMA-series models with the math reasoning dataset.

Model	LLaMA$_{7B}$	LLaMA$_{7B}$	LLaMA$_{13B}$	LLaMA$_{13B}$
Sparsity	40%	50%	40%	50%
Epoch	4	3	3	3
Batch size	16	16	16	16
Learning rate	3e-4	3e-4	3e-4	3e-4
LoRA alpha	64	64	64	64
LoRA target modules	Q, K, V, Up, Gate, Down	Q, K, V, Up, Down	Q, K, V, Up, Down	Q, K, V, Up, Down
Low-rank Search Space	[32, 24, 16]	[32, 24, 16]	[32, 24, 16]	[32, 24, 16]

Table 8: Hyperparameters for LLaMA$_{7B}$ with the commonsense reasoning dataset.

Train set size	15k	15k	170k	170k
Sparsity	40%	50%	40%	50%
Epoch	3	3	3	5
Batch size	16	16	16	16
Learning rate	3e-4	3e-4	3e-4	3e-4
LoRA alpha	64	64	64	64
LoRA target modules	Q, K, V, Up, Down	Q, K, V, Up, Gate, Down	Q, K, V, Up, Gate, Down	Q, K, V, Up, Down
Low-rank Search Space	[32, 24, 16]	[32, 24, 16]	[32, 24, 16]	[32, 24, 16]

Table 9: Hyperparameters for MPT$_{7B}$ with GSM8K.

Sparsity	40%	50%	60%	70%
Epoch	4	5	5	8
Batch size	16	16	16	16
Learning rate	5e-4	3e-4	3e-4	3e-4
LoRA alpha	64	64	64	64
LoRA target modules	Q, K, V, O, Up, Down	Q, K, V, O, Up, Down	Q, K, V, O, Up, Down	Q, K, V, O, Up, Down
Low-rank Search Space	[32, 24, 16]	[32, 24, 16]	[32, 24, 16]	[32, 24, 16]