SQFT: Low-cost Model Adaptation in Low-precision Sparse Foundation Models

Anonymous ACL submission

Abstract

Large pre-trained models (LPMs), such as large language models, have become ubiquitous and are employed in many applications. These models are often adapted to a desired domain or downstream task through a fine-tuning stage. This paper proposes SQFT, an end-to-end solution for low-precision sparse parameter-efficient fine-tuning of LPMs, allowing for effective model manipulation in resource-constrained environments. Additionally, an innovative strategy enables the merging of sparse weights with low-rank adapters without losing sparsity and accuracy, overcoming the limitations of previous approaches. SQFT also addresses the challenge of having quantized weights and adapters with different numerical precisions, enabling merging in the desired numerical format without sacrificing accuracy. Multiple adaptation scenarios, models, and comprehensive sparsity levels demonstrate the effectiveness of SQFT. **We make SQFT's fine-tuned models available to reviewers for reproducing our results at:** https://anonymous.4open.science/r/sqft_examples-71C7

1 Introduction

Despite several limitations, such as hallucinations and a significant computational footprint, large pre-trained, foundation, or frontier models have become integral to numerous applications, including language understanding and code generation. These models are trained with extensive corpora on thousands of graphics processing units (GPUs), resulting in outstanding zero-shot performance across various tasks and datasets. However, it is frequently the case that they must be adapted to improve their performance on new tasks or data.

Low-rank adapters (LoRA) (Hu et al., 2022) have demonstrated their effectiveness in model adaptation. However, when LoRA is combined with model compression techniques, e.g., sparsity or quantization, several challenges prevent merging these adapters into a single compressed and fine-tuned model, as illustrated in Figure 1. These challenges stem from two primary reasons: **i)** merging dense adapters causes the loss of sparsity in the base model, and **ii)** adapter merging cannot be achieved due to different numerical precisions.

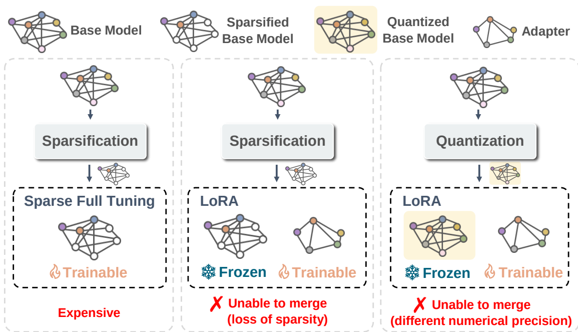

Figure 1: Limitations of existing approaches for fine-tuning sparse and quantized models. Full fine-tuning is expensive. Low-rank adapters (LoRA) for Parameter-efficient Fine-tuning (PEFT) on sparse or quantized models cannot easily merge with the compressed weights due to loss of previously induced sparsity or different numerical precision.

This paper introduces **SQFT**, an end-to-end compression and model adaptation solution for large pre-trained models (LPMs) that alleviates the limitations above. SQFT is designed to sparsify, quantize, and fine-tune large models and can instantiate efficient pipelines that streamline compression techniques. Within the SQFT framework, we propose **Sparse** Parameter-**E**fficient **F**ine-**T**uning (**SparsePEFT**), a strategy to address the adapter merging problem for sparse and quantized model, resulting in more effective high-performing models. Furthermore, SQFT also benefits from weight-sharing techniques applied to traditional parameter-efficient fine-tuning (PEFT) techniques and incorporates insights from state-of-the-art compression techniques. Throughout this paper, we discuss the

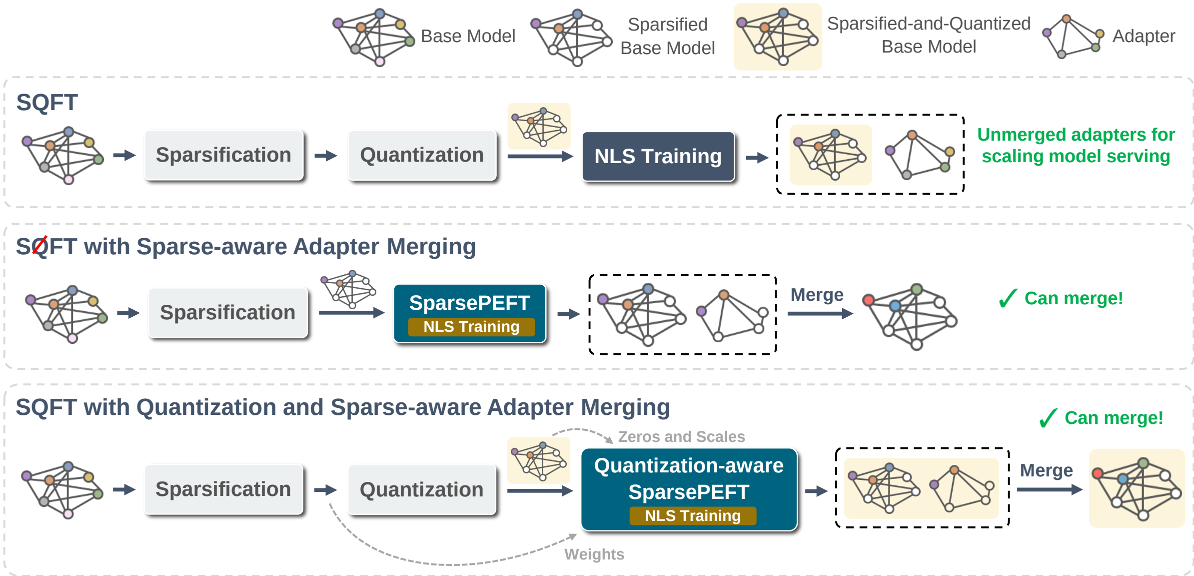

Figure 2: SQFT Overview. Several pipeline configurations can be activated to efficiently fine-tune large models while addressing several limitations of existing approaches.

following contributions:

1. An end-to-end model adaptation solution, SQFT, designed for efficient low-cost configurable pipelines tailored for large pre-trained models with low numerical precision and sparsity.

2. SparsePEFT, a component of SQFT, addresses several limitations in existing parameter-efficient fine-tuning approaches for sparse and quantized models, including the reduction in the cost of fine-tuning, the effective merging of adapters into the sparse model without the loss of sparsity, and the effective merging of components that operate in different numerical precision.

3. Extensive experiments demonstrate the effectiveness of SQFT across different foundation models, sparsity levels and adaptation scenarios.

This paper is organized as follows: Section 2 describes the stages in the proposed end-to-end solution, SQFT. Section 3 discusses SQFT's evaluation, and we finalize with some concluding remarks in Section 4. Due to page limits, we include a Related Work section, and additional results in the Appendix.

2 Methodology

SQFT fine-tunes large pre-trained models (LPMs) in an efficient multi-stage approach that includes (1) Sparsification, with an optional reduction in the numerical precision, i.e., Quantization, (2) Fine-tuning with Neural Low-rank Adapter Search (NLS), (3) Sparse Parameter-Efficient Fine-Tuning (SparsePEFT) with optional (4) Quantization-awareness. Figure 2 illustrates the alternative LPM compression and model adaptation pipelines that SQFT can instantiate. In the following sections, we discuss the details of each stage and the benefits of accelerating inference and model serving.

2.1 Sparsification and Quantization Stage

As shown in Figure 2, at the beginning of all possible pipeline configurations, SQFT employs an effective method to induce sparsity in the model. For a given weight matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, with entries $w_{i,j}$ s.t. $\boldsymbol{W} = (w_{i,j}), 1 \leq i \leq m, 1 \leq j \leq n$, an arbitrary scoring function, Ψ, is assigned to the proposed solution. This function determines the relative importance of $w_{i,j}$ compared to the other weights in $\boldsymbol{W}$. Ψ can be formulated in various ways. For instance, $\Psi(\boldsymbol{W}) = |\boldsymbol{W}| \cdot \|\boldsymbol{X}\|_2$, where $\boldsymbol{X}$ represents sampled feature input activations, as proposed by Sun et al. (2023). However, it is important to highlight that the proposed end-to-end model fine-tuning solution, SQFT, can utilize any other scoring function. Leveraging the scores from Ψ and a desired level of sparsity, s, we derive the sparsified weight, denoted as $\boldsymbol{W}^p$, with a sparsity pattern $S\{\boldsymbol{W}^p\} = \{(i,j) \mid \boldsymbol{W}^p_{i,j} \neq 0, 1 \leq i \leq m, 1 \leq j \leq n\}$, s.t. $|S\{\boldsymbol{W}^p\}| \leq |S\{\boldsymbol{W}\}|$.

It has been demonstrated that LPMs can tolerate higher sparsity levels compared with the previous generations of smaller transformer-based models (Frantar and Alistarh, 2023). Our experiments confirm these observations (Section 3). Once SQFT has induced sparsity in the pre-trained weights, $\boldsymbol{W}^p$ enables an optional reduction in their numeric precision. Given the sparsified weights, SQFT applies layer-wise one-shot quantization (Nagel et al., 2020; Frantar et al., 2022a; Wang et al., 2020; Frantar et al., 2022b). Utilizing a selection from state-of-the-art post-training quantization approaches, SQFT identifies the low-precision sparsified weights, denoted as $\widehat{\boldsymbol{W}}^p$, that given an input $\boldsymbol{X}$, minimize $argmin_{\widehat{\boldsymbol{W}}^p} ||\boldsymbol{W}^p\boldsymbol{X} - \widehat{\boldsymbol{W}}^p\boldsymbol{X}||_2^2$.

Reducing the numerical precision and inducing sparsity on weights frequently decrease the model's accuracy, requiring fine-tuning to improve performance.

2.2 Fine-tuning with Neural Low-rank Adapter Search (NLS)

Given the sparse quantized weights, $\widehat{\boldsymbol{W}}^p$, SQFT recovers any drops in accuracy induced by the compression schema and fine-tunes these weights for a specific downstream task. As shown in Figure 2, SQFT employs Neural Low-rank Adapter Search (NLS) (Munoz et al., 2024a) instead of vanilla Low-rank Adapters (LoRA) (Hu et al., 2022), and fine-tunes sparse and quantized model. To justify using NLS, traditional LoRA adapters require assigning the values for several hyperparameters, including their rank r, and the subset of modules where these adapters will be placed. Determining these hyperparameters can be a challenging endeavor. To alleviate this limitation, SQFT extends NLS' weight-sharing techniques to facilitate the discovery of optimal adapter configurations from a space of elastic adapter configurations. In other words, instead of having a fixed value for the rank, r, we enable elastic configurations, $C = [c_1, \ldots, c_n]$, s.t., $r \leftarrow c_i$ depending on the activation of the corresponding sub-adapter.

2.3 SparsePEFT

Fine-tuning the sparse quantized model with adapters effectively improves the model's performance on a downstream task. However, as illustrated in the middle and right part of Figure 1, a challenge arises when dealing with sparse or quantized weights and dense adapter weights: merging them will i) result in the loss of sparsity on the model's weights or ii) be unable to merge due to different numerical precisions. Aiming to address the first limitation, we propose an effective strategy, Sparse Parameter-Efficient Fine-Tuning (SparsePEFT), to make adapters sparsity-aware. As depicted in Figure 3, SparsePEFT applies a binary mask $\boldsymbol{M}$ derived from the initial sparsification of $\boldsymbol{W}$. This mask is used to sparsify the adapters matrix (denoted as $\boldsymbol{BA}$) into $\boldsymbol{L}^p$. The process can be formulated as:

$$\boldsymbol{L}^p = (\boldsymbol{BA}) \odot \boldsymbol{M}, \tag{1}$$

which is activated during the fine-tuning process for sparsity awareness. SparsePEFT enables the merging of the sparsified weights $\boldsymbol{W}^p$ and the adapter weight $\boldsymbol{L}^p$ without sacrificing the sparsity induced early in the compression pipeline as follows,

$$\boldsymbol{W}^p \leftarrow \boldsymbol{W}^p + \boldsymbol{L}^p. \tag{2}$$

In addition to preserving sparsity, SparsePEFT demonstrates comparable (even better) accuracy compared to fine-tuning with dense adapters. Extensive experimental findings substantiate the advantages of SparsePEFT, as detailed in Section 3.

Although SparsePEFT can effectively preserve the model's sparsity, it presents additional challenges when merging with quantized models, the second limitation we discussed before, which is primarily attributed to the need for the adapter and pre-trained weights to possess identical numerical precision. In the following subsection, we explore a pipeline variation for SQFT that facilitates the integration of sparse quantized weights. This approach aims to address both challenges mentioned above while improving the overall efficiency of the resulting model.

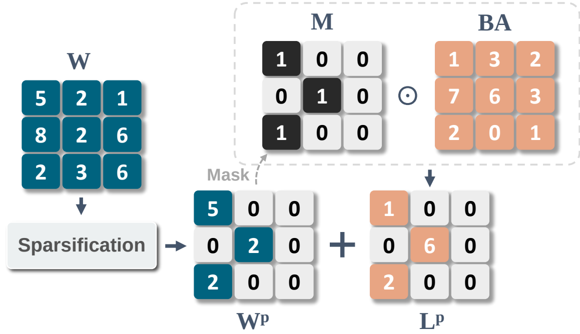

Figure 3: Sparse Parameter-efficient Fine-tuning (SparsePEFT). A binary mask is obtained from the sparsified weights and applied to the adapters, allowing for the later merge without loss of sparsity.

2.4 Quantization-aware SparsePEFT

Building upon the concept of SparsePEFT, we propose Quantization-aware SparsePEFT (QA-SparsePEFT), an extension of SparsePEFT for sparse quantized models. QA-SparsePEFT integrates quantization awareness into SparsePEFT. In most common quantization schemes, the zero point and scales for the target quantized tensor can be determined during the quantization process (e.g., GPTQ (Frantar et al., 2022a)). Within the framework of QA-SparsePEFT, the zeros, and scales of the sparse quantized weights $\widehat{\boldsymbol{W}}^{p}$ are shared with the adapter. The adapters can be quantized smoothly with the shared fixed zeros and scales, enabling quantization-aware fine-tuning. Formally, given the sparsified pre-trained weight $\boldsymbol{W}^{p}$, sparsified adapter weight $\boldsymbol{L}^{p}$ obtained from SparsePEFT, zeros $\boldsymbol{z}$ and scales $\boldsymbol{s}$ from the quantization of $\boldsymbol{W}^{p}$, the quantization process in the proposed QA-SparsePEFT can be formulated as:

$$\widehat{\boldsymbol{W}}^{p}_{m} = \text{round}\left(\text{clamp}\left(\frac{(\boldsymbol{W}^{p}+\boldsymbol{L}^{p})-\boldsymbol{z}}{\boldsymbol{s}}, Q_n, Q_p\right)\right), \tag{3}$$

where $\widehat{\boldsymbol{W}}^{p}_{m}$ denotes the sparse quantized (merged) weight, $Q_n = -2^{n-1}$ and $Q_p = 2^{n-1} - 1$ (n represents the bit-width of the quantized values). Dequantization is the inverse as follows:

$$\tilde{\boldsymbol{W}}^{p}_{m} = \widehat{\boldsymbol{W}}^{p}_{m} \times \boldsymbol{s} + \boldsymbol{z}, \tag{4}$$

which applies $\boldsymbol{z}$ and $\boldsymbol{s}$ to approximate $\boldsymbol{W}^{p}_{m}$. Through QA-SparsePEFT, we can obtain the fine-tuned, sparsified low-precision resulting model. Moreover, SQFT with QA-SparsePEFT can run the NLS stage using this schema, which allows us to merge the adapters once an optimal configuration has been discovered.

2.5 Model Serving and Inference Acceleration

Accelerating model serving and inference through sparsification and quantization techniques has shown significant efficacy across various hardware platforms and kernels, demonstrating remarkable speedups. However, for PEFT with a sparsified or quantized model (as shown in Figure 1), the addition of adapter models introduces the computational overhead during inference due to their non-mergeability. SparsePEFT (QA-SparsePEFT) allows adapters to be merged into the sparse (quantized) model, which can reduce adapters' redundancy and computational overhead, leading to more streamlined inference processes. Moreover, quantization techniques further enhance acceleration by reducing the model size and computational complexity, but balancing the trade-off between acceleration and maintaining competitive accuracy is essential.

In summary, SQFT and its SparsePEFT strategy bring the benefits of adapter merging and maintaining accuracy on sparse or quantization scenarios. The choice between the sparsity level and whether to apply quantization depends on the specific deployment scenario (e.g., task requirements and resource constraints), including the trade-off between model performance, inference speed, and memory efficiency. In the next section, we will delve into further empirical studies to fully understand the strengths and weaknesses of each approach in different settings.

3 Experimental Results

We evaluate SQFT on several state-of-the-art large pre-trained models and datasets. Next, we discuss the setup for our experimental analysis.

3.1 Setup

Models SQFT is evaluated on two state-of-the-art models, including Llama-3-8B[1], Phi-3-Mini-4K-Instruct[2]. To study it more comprehensively, we aim to explore SQFT across different models, scales, and settings.

Datasets and Settings Aligned with other works in the LPMs compression and fine-tuning spaces, SQFT is validated on three experimental settings: **1)** Grade School Math 8K (GSM8K) (Cobbe et al., 2021), **2)** Math reasoning with instruction tuning (following LLM-Adapters (Hu et al., 2023)), including 3 math reasoning datasets: GSM8K, Math Word Problems (MAWPS) (Koncel-Kedziorski et al., 2016), Simple Variations on Arithmetic Math word Problems (SVAMP) (Patel et al., 2021), and **3)** Commonsense reasoning datasets: Boolean Questions (BoolQ) (Clark et al., 2019), Physical Interaction: Question Answering (PIQA) (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), Large-scale Winograd Schema Challenge (WinoGrande) (Sakaguchi et al., 2021), AI2 Reasoning Challenges (Arc-e, Arc-c) (Clark et al., 2018), and Open Book Question Answering (OBQA) (Mihaylov et al., 2018).

The evaluations of our experiments are conducted utilizing *lm-eval-harness* (Gao et al., 2023)

[1]https://huggingface.co/meta-llama/Meta-Llama-3-8B
[2]https://huggingface.co/microsoft/Phi-3-mini-4k-instruct

Table 1: Results for adapting **Llama-3-8B** to GSM8K. The criterion for mergeable is that there should be no loss in either accuracy or sparsity before and after merging. The evaluation used the default configuration for *lm-eval-harness* (Gao et al., 2023).

Model	Sparsity	Method	Mergeable	Final Precision (Base + Adapter / Base)	GSM8K Test Accuracy(%)
Llama-3-8B	0%	w/o tune	-	FP16	50.0
		w/o Quantization			
	50%	w/o tune	-	FP16	12.5
		LoRA	✗	FP16 + FP16	50.6
		Shears	✗	FP16 + FP16	52.2
		SQFT + SparsePEFT (Ours)	✓	FP16	**52.5**
		Quantization			
		w/o tune	-	**INT4**	7.0
		GPTQ + LoRA	✗	INT4 + FP16	48.9
		SQFT (Ours)	✗	INT4 + FP16	50.0
		SQFT + QA-SparsePEFT (Ours)	✓	**INT4**	**50.2**

in both setting 1 and 3, while following the evaluation from LLM-Adapters in setting 2. We present a comparative analysis of the results obtained from our various pipelines and also compare with vanilla LoRA (Hu et al., 2022), Shears (Munoz et al., 2024a) (a parameter-efficient fine-tuning method for sparse models), and GPTQ + LoRA. For fair comparison, all methods are run in the same environment and with the same configuration. SQFT employs the implementation of Wanda (Sun et al., 2023) as default method for sparsification, and GPTQ in Huggingface [3] for quantizing the LPMs and adapters.

Reference configuration Unless stated in the results, we report a reference configuration for SQFT. This configuration is obtained utilizing the heuristic proposed in Munoz et al. (2024b). The heuristic is intuitive and straightforward, activating the configuration with the median of each set of elastic values per module. Spending additional cycles to search the space of configurations might yield even more competitive results, presented in Table 4. Next, we discuss experimental results and studies conducted using SQFT.

3.2 Main Results

3.2.1 Fine-tuning Llama-3 on GSM8K

We begin our evaluation with Llama-3B-8B, assessing its accuracy in a dense mode and after inducing 50% sparsity without fine-tuning on the GSM8K dataset. Subsequently, we execute various pipelines of SQFT. As described in Table 1, for Llama-3-8B at the 50% sparsity level, SQFT recovers the model's accuracy from 12.5% to 52.5% without employing quantization, while allowing for the merging of adapters without sacrificing sparsity (SparsePEFT) and incorporating quantization into the pipeline results in a minor drop in accuracy to 50.2% when enabling the adjustment to merge adapters (QA-SparsePEFT).

More importantly, SQFT with SparsePEFT and QA-SparsePEFT exhibit comparable performance to their corresponding non-mergeable approaches. These results suggest that SQFT with SparsePEFT (QA-SparsePEFT) effectively addresses the limitation of the merging problem encountered when fine-tuning adapters into sparse models (or sparse and quantized models) without any degradation in accuracy. Furthermore, the comparison between LoRA and SQFT with SparsePEFT (or Shears), and between GPTQ + LoRA and SQFT with QA-SparsePEFT without adapter merging, highlights the superior performance of NLS (elastic rank) compared with LoRA (fixed rank). We also explore the performance of a broader range of sparsity levels and conduct more detailed ablation experiments in this experimental setting, which can be found in Sections 3.4 and 3.6, respectively.

3.2.2 Math Reasoning with Instruction Tuning for Phi-3

In addition to Llama-3 on GSM8K, we also investigated the performance of SQFT with the Phi-3 model. Since the Phi-3-series models released by Microsoft are the instruction models currently best-suited for a chat prompt, we evaluate SQFT on three math reasoning datasets for instruction tuning. Table 2 presents the test accuracy for our approaches and baselines. Interestingly, in the full-precision mode (*w/o Quantization*), our proposed SparsePEFT not only achieves the highest average accuracy (77.3%) compared to other approaches

[3] https://huggingface.co/blog/gptq-integration

Table 2: Results for **Phi-3-Mini-4K-Instruct** with math instruction tuning. *Mergeable* means that merging the dense adapters with the sparse weights is possible without losing the induced sparsity levels or affecting the desired low numerical precision.

Model	Sparsity	Method	Mergeable	Final Precision (Base + Adapter / Base)	Datasets | Accuracy(%) GSM8K	MAWPS	SVAMP	Average
Phi-3-Mini-4K-Instruct	0%	w/o tune	-	FP16	64.7	84.5	85.4	78.2
	50%	*w/o Quantization*						
		w/o tune	-	FP16	38.9	64.7	66.8	56.8
		LoRA	✗	FP16 + FP16	62.5	90.3	77.8	76.9
		Shears	✗	FP16 + FP16	62.3	90.8	76.1	76.4
		SQFT + SparsePEFT (Ours)	✓	FP16	61.9	91.2	78.7	**77.3**
		Quantization						
		w/o tune	-	**INT4**	33.4	56.7	64.2	51.4
		GPTQ + LoRA	✗	INT4 + FP16	60.3	89.5	74.8	74.9
		SQFT (Ours)	✗	INT4 + FP16	60.3	90.8	75.6	**75.5**
		SQFT + QA-SparsePEFT (Ours)	✓	**INT4**	60.4	90.8	72.9	74.7

but also uniquely allows for the merging of adapters and sparse weights without any loss of sparsity. This result is achieved without needing an expensive search and by utilizing the heuristic detailed in Section 3.1. However, in quantization mode, the accuracy of SQFT + QA-SparsePEFT (mergeable) is marginally lower compared to the non-mergeable approaches (74.7% vs. 74.9%/75.7%). This result suggests there may be a need to balance the trade-off between accuracy and efficiency. Fortunately, SQFT + QA-SparsePEFT results in a merged fine-tuned quantized model, eliminating the overhead associated with dense adapters.

3.2.3 Fine-tuning Phi-3 on Commonsense Reasoning

Besides the mathematical domain of the first two experimental settings, we also explore SQFT in other areas, e.g., commonsense reasoning. We apply SQFT to fine-tuning the Phi-3 model on a set of unified commonsense training datasets with 83K samples for fine-tuning from BoolQ, PIQA, HellaSwag, WinoGrande, Arc-e, Arc-c, and OBQA. Table 3 compares the test accuracy of the evaluated approaches. SQFT obtains a competitive configuration with Shears, LoRA, and GPTQ + LoRA. However, SQFT has the additional benefit of allowing for the merging without losing the previously induced sparsity, both in full-precision and quantized modes. It is worth noting that SQFT with QA-SparsePEFT shows super competitiveness here, i.e., the most efficient model with high accuracy (among all full-precision and quantized cases).

3.3 Hill-climbing to Better Configurations

The results presented in the previous sections employ the simple heuristic (as detailed in Section 3.1) to obtain a reference configuration from the NLS search space. However, superior configurations can be discovered with an additional budget. We apply a well-designed hill-climbing search algorithm (Algorithm 1 in Appendix), which starts from the configuration derived from the heuristic and explores its neighboring configurations in a hill-climbing matter based on their validation accuracy. For this purpose, we employed the validation sets from Arc-e, Arc-c, and OBQA, as other datasets do not provide a validation set. As demonstrated in Table 4, a more optimal configuration can be discovered, outperforming the default adapter configuration obtained from the heuristic. Exploring further the search space of elastic adapter ranks produces richer adapter distributions as depicted in Figure 4. More importantly, the test set results re-

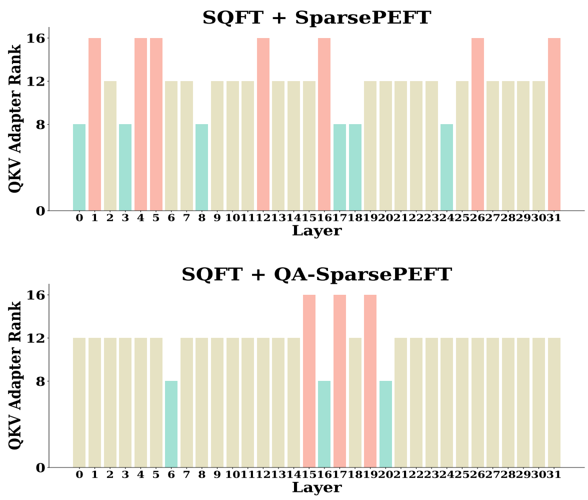

Figure 4: The adapter rank distribution of the optimal configurations obtained from the hill-climbing search algorithm (**Phi-3-Mini-4K-Instruct** with commonsense reasoning).

Table 3: Results for **Phi-3-Mini-4K-Instruct** with commonsense reasoning. SQFT obtains competitive fine-tuned models with an additional benefit over Shears and LoRA applied to low-precision weights, i.e., SQFT's adapters can be efficiently merged into the weights without any loss of precision or accuracy. We are reporting a reference submodel for SQFT obtained the heuristic detailed in 3.1, which means that, as shown in Table 4, with an additional cost, SQFT can discover submodels with even higher performance.

Model	Sparsity	Method	Mergeable	Final Precision (Base + Adapter / Base)	BoolQ	PIQA	HellaS	WinoG	Arc-e	Arc-c	OBQA	Average
					Datasets | Accuracy(%)							
Phi-3-Mini-4K-Instruct	0%	w/o tune	-	FP16	86.1	80.3	78.5	73.7	83.2	57.5	46.8	72.3
		w/o Quantization										
	50%	w/o tune	-	FP16	82.5	75.9	69.9	69.1	76.9	50.9	43.4	66.9
		LoRA	✗	FP16 + FP16	85.6	79.1	75.8	71.5	79.6	53.2	49.4	70.6
		Shears	✗	FP16 + FP16	85.2	78.9	75.7	72.6	80.1	53.3	50.4	**70.9**
		SQFT + SparsePEFT (Ours)	✓	FP16	84.0	78.8	75.5	72.1	80.1	53.5	48.6	70.4
		Quantization										
		w/o tune	-	**INT4**	81.4	75.2	68.5	68.2	75.9	50.3	40.2	65.7
		GPTQ + LoRA	✗	INT4 + FP16	85.3	79.1	75.3	72.5	79.5	54.6	47.2	70.5
		SQFT (Ours)	✗	INT4 + FP16	85.1	79.0	75.4	71.2	79.6	54.1	48.8	70.5
		SQFT + QA-SparsePEFT (Ours)	✓	**INT4**	83.7	80.1	74.1	73.6	80.1	55.1	48.2	**70.7**

Table 4: Hill-climbing searching results for **Phi-3-Mini-4K-Instruct** with commonsense reasoning.

Model	Sparsity	Method	Sub-Adapter	Validation Datasets | Accuracy(%) Arc-e	Arc-c	OBQA	Average	Test Datasets | Accuracy(%) BoolQ	PIQA	HellaS	WinoG	Arc-e	Arc-c	OBQA	Average
Phi-3-Mini-4K-Instruct	50%	SQFT + SparsePEFT	Heuristic	79.3	50.8	47.4	59.2	84.0	78.8	**75.5**	**72.1**	80.1	53.5	48.6	70.4
			Hill-climbing	**80.2**	**51.8**	**47.6**	**59.9**	**84.3**	**78.9**	75.4	72.0	80.1	**54.3**	**49.4**	**70.6**
		SQFT + QA-SparsePEFT	Heuristic	80.0	51.5	45.4	59.0	**83.7**	**80.1**	**74.1**	73.6	80.1	55.1	48.2	70.7
			Hill-climbing	**80.4**	**53.5**	**46.2**	**60.0**	83.6	79.7	**74.1**	**73.7**	80.1	**56.2**	**48.8**	**70.9**

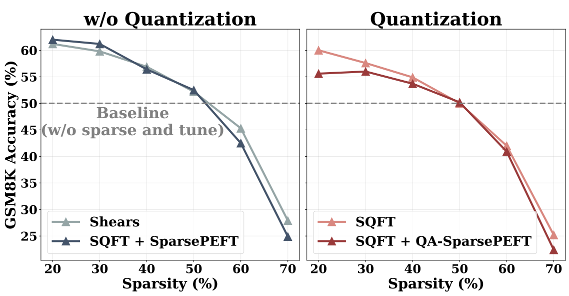

Figure 5: Comparison of various sparsity levels for Llama-3-8B with GSM8K. SQFT achieves similar performance as Shears but with the added benefit of merging adapters with different numerical precision.

veal a significant improvement in the performance of the Arc-c and OBQA datasets, which suggests that an appropriate validation set can assist in identifying the optimal adapter configuration.

3.4 Exploring a Broader Range of Sparsity Levels

All our previous experiments employ 50% sparsity as it is moderate and mild. In this section, we explored the behavior of SQFT in a broader range of sparsity levels. As shown in Figure 5, the model's accuracy experiences a significant drop between a sparsity of 60% and 70%. We denote this range as the critical sparsity threshold, representing the boundary at which the model's performance begins to degrade notably. Through our recovery downstream fine-tuning strategy, models with up to 50% sparsity (even with quantization) can achieve comparable performance with the original dense model (represented by the baseline in the figure) on the downstream task. This 50% sparsity can be defined as the optimal sparsity level, as it represents the point of balance where the model maintains high performance while achieving computational efficiency. Moreover, there is little difference in accuracy between our mergeable approaches and non-mergeable methods, which illustrates the effectiveness of our proposed SparsePEFT.

3.5 Cost Analysis of Pipeline Configurations

The different versions of SQFT's pipelines incur various costs that allow users to choose based on their fine-tuning budget. Table 6 details the characteristics of each pipeline configuration, e.g., whether we can merge the adapters, the precision of the based model and the adapters, and the cost of each configuration. Two assumptions are

Table 5: Ablation studies for LoRA vs. NLS (**Llama-3-8B** with GSM8K). Compared to LoRA, NLS demonstrates significantly better accuracy performance across all possible pipelines of SQFT and different sparsity levels.

Model	Sparsity	Method	Mergeable	Final Precision (Base + Adapter / Base)	Fine-tune Approach	GSM8K Test Accuracy(%)
Llama-3-8B	30%	Shears	✗	FP16 + FP16	LoRA	58.2
					NLS	**59.8**$_{+1.6}$
		SQFT + SparsePEFT (Ours)	✓	FP16	LoRA	60.0
					NLS	**61.2**$_{+1.2}$
		SQFT (Ours)	✗	INT4 + FP16	LoRA	56.7
					NLS	**57.6**$_{+0.9}$
		SQFT + QA-SparsePEFT (Ours)	✓	INT4	LoRA	54.8
					NLS	**56.0**$_{+1.2}$
	50%	Shears	✗	FP16 + FP16	LoRA	50.6
					NLS	**52.2**$_{+1.6}$
		SQFT + SparsePEFT (Ours)	✓	FP16	LoRA	50.6
					NLS	**52.5**$_{+1.9}$
		SQFT (Ours)	✗	INT4 + FP16	LoRA	48.9
					NLS	**50.0**$_{+1.1}$
		SQFT + QA-SparsePEFT (Ours)	✓	INT4	LoRA	48.2
					NLS	**50.2**$_{+2.0}$
	70%	Shears	✗	FP16 + FP16	LoRA	25.5
					NLS	**27.9**$_{+2.4}$
		SQFT + SparsePEFT (Ours)	✓	FP16	LoRA	22.1
					NLS	**24.9**$_{+2.8}$
		SQFT (Ours)	✗	INT4 + FP16	LoRA	24.2
					NLS	**25.2**$_{+1.0}$
		SQFT + QA-SparsePEFT (Ours)	✓	INT4	LoRA	**22.6**$_{+0.2}$
					NLS	22.4

made regarding model storage, inference speedup, or memory: merging is better than unmerging due to the overhead from the unmerged adapters, and quantization mode is better than full-precision mode. As for accuracy, the mergeable method we propose is competitive with the previous non-mergeable method. Regarding the fine-tuning time, our mergeable method is slightly slower than the non-mergeable method due to the additional mask and adapter calculations. In summary, SQFT with SparsePEFT is the best choice for full-precision mode because it eliminates the adapter's additional path without sacrificing accuracy. Suppose memory usage during fine-tuning is a priority for the quantization mode. In that case, vanilla SQFT (first configuration in Figure 2) is the best choice because it only requires the quantized model with little overhead of different precision adapters. Otherwise, SQFT with QA-SparsePEFT is better because it can ultimately produce a most efficient model that will be of great benefit at deployment time.

3.6 Ablation Studies - LoRA vs NLS

As shown in Table 5, the ablation studies across 30%, 50%, and 70% sparsity highlight the benefits of elastic adapters (NLS), which enhance the performance of SQFT, further reducing the gap to the dense or non-quantized models while enjoying the advantages of sparsity or quantization.

Table 6: Cost analysis for different pipelines (**rank**). ID 1, 2, 3, and 4 represent LoRA/Shears, SQFT + SparsePEFT, SQFT, and SQFT + QA-SparsePEFT, respectively.

ID	**1**	**2**	**3**	**4**
Mergeable	✗	✓	✗	✓
Final Precision	FP16 + FP16	FP16	INT4 + FP16	INT4
Model Storage (↓)	$1 > 2 > 3 > 4$			
Fine-tuning Time (↓)	$1 \approx 3 < 2 \approx 4$			
Fine-tuning Memory (↓)	$3 < 1 \approx 2 \approx 4$			
Inference Speedup (↑)	$4 > 3 > 2 > 1$			
Inference Memory (↓)	$4 < 3 < 2 < 1$			
Accuracy (↑)	$1 \approx 2 > 3 \approx 4$			

4 Conclusion

Large pre-trained models often require fine-tuning to downstream target tasks and compression to utilize them in resource-constrained environments. This paper presents SQFT, a low-cost fine-tuning solution for low precision and sparse foundation models. SQFT solves challenges when merging sparse (and quantized) base models and dense (with different numerical precision) adapters without losing the induced sparsity in the base model while delivering high-performing fine-tuned models. We make a few SQFT's fine-tuned models available to reviewers for reproducing our results at: https://anonymous.4open.science/r/sqft_examples-71C7

Limitations and Ethical Considerations

Large pre-trained models have gained popularity and are the base of many applications. However, these models are often used indiscriminately with little analysis of their potential failures and consequences. SQFT solely focuses on these large models' efficient fine-tuning and compression. However, users of SQFT should also consider the limitations of these models before deployment in environments where they can cause harm or conflict. Although compressing and fine-tuning these models on a particular downstream task would make them perform better, more studies are needed regarding the effects of this specialization.

We demonstrate SQFT on several pre-trained models. The benefits obtained from the proposed solution might transfer smoothly to other transformer-based models. However, there might also be models and datasets in which additional considerations must be taken. For instance, in our current experiments, we have noticed that in the case of OpenELM-1.1B (Mehta et al., 2024), fine-tuning on math reasoning datasets, e.g., GSM8K, does not result in high accuracy, and more experimentation is needed. There is also the case in which a pre-trained model might have been trained on a particular benchmark, a form of data contamination, which is difficult to confirm since often the details of the training data are not shared publicly (Zhang et al., 2024). In these cases, inducing sparsity might result in a drop in accuracy on that particular benchmark.

Due to the many unknowns and complexity of current large models, it is essential to take measures to prevent their use in sensitive applications. With insights obtained by the research community in the years to come, understanding the intricacies of these models will help us use them beneficially and safely.

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*.

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. *Preprint*, arXiv:2110.14168.

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*.

Tim Dettmers and Luke Zettlemoyer. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022a. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. 2022b. Optimal Brain Compression: a framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 36.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Masafumi Hagiwara. 1994. A simple and effective method for removal of hidden units and weights. *Neurocomputing*, 6(2):207–218. Backpropagation, Part IV.

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. *J. Mach. Learn. Res.*, 22(1).

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*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Sachin Mehta, Mohammad Sekhavat, Qingqing Cao, Max Horton, Yanzi Jin, Frank Sun, Iman Mirzadeh, Mahyar Najibikohnehshahri, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. 2024. Openelm: An efficient language model family with open training and inference framework.

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*.

J. Pablo Munoz, Jinjie Yuan, and Nilesh Jain. 2024a. Shears: Unstructured sparsity with neural low-rank adapter search. *The 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2024)*.

J. Pablo Munoz, Jinjie Yuan, Yi Zheng, and Nilesh Jain. 2024b. LoNAS: Elastic low-rank adapters for efficient large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10760–10776, Torino, Italia. ELRA and ICCL.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

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.

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.

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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng. 2020. Towards accurate post-training network quantization via bit-split and stitching. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9847–9856. PMLR.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*.

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

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

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*.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. 2024. A careful examination of large language model performance on grade school arithmetic. *Preprint*, arXiv:2405.00332.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2023. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning. *Preprint*, arXiv:2305.18403.

Appendix

A Related Work

Generative pre-trained models often based on the Transformer architecture (Vaswani et al., 2017) require the application of compression techniques to reduce their significant computational cost and to address challenges, e.g., related to memory bandwidth. Classic compression techniques like pruning and quantization have been adapted to the age of LPMs, removing inefficiencies that cannot be tolerated when dealing with billions of parameters. We discuss them in more detail next.

Pruning Inducing sparsity, either by zeroing out weights or activations or removing network elements, can improve the efficiency of LPMs during inference, provided that they are executed on a runtime that can exploit sparse patterns. Pruning has a long history (LeCun et al., 1989), but with the advent of LPMs, traditional methods(Hoefler et al., 2021), e.g., Magnitude Pruning (Hagiwara, 1994), have been replaced by new approaches that are suited for the challenges of these models. In particular, due to their large number of parameters. SparseGPT (Frantar and Alistarh, 2023) proposes a one-shot pruning method for transformer-based models that trade minimal accuracy drop for increasing sparsity levels. The method approaches LPMs' pruning layer-wise with an efficient weight reconstruction algorithm that incrementally prunes the weight matrix elements. Wanda (Sun et al., 2023) proposes a more straightforward approach that does not require weight updates, computing a score using the weight magnitude and the norm of input activations. This approach obtains better results than SparseGPT. Recently, BESA (Xu et al., 2024) improves over SparseGPT and Wanda by targeting individual transformer blocks and allocating sparsity per layer using a differentiable method. These approaches induce sparsity on pre-trained models and are evaluated on zero-shot benchmarks. Our end-to-end solution, SQFT, focuses on further adapting the sparsified models to new tasks or datasets.

Quantization In the era of large pre-trained foundation/frontier models (LPMs), quantization approaches have evolved to address the challenges of scale and memory bandwidth. Due to the high cost of retraining these models to recover accuracy degradation, special consideration has to be taken when incorporating compression techniques, like quantization-aware training in foundation models. Post-training, one-shot quantization methods have prevailed, obtaining quantized versions of large models in hours. LLM.Int8() was among the first Int8 quantization procedures for large-scale transformer-based PLMs (Dettmers et al., 2022). Using vector-wise quantization and mixed-precision decomposition, LLM.Int8() demonstrated that it can effectively confront the outliers that emerge in activations, which makes traditional quantization methods fail in models with more than 6.7B parameters. In a contemporary work, after running thousands of experiments with various large pre-trained models, it was demonstrated that 4-bit parameters can reach optimal performance compared to other bit-precisions in the 3 to 16-bit range (Dettmers and Zettlemoyer, 2023). ZeroQuant (Yao et al., 2022) quantizes GPT-3 models, obtaining a reduction in latency up to 4.16x by utilizing group-wise quantization for weights, token-wise quantization for activations, and layer-by-layer knowledge distillation. SmoothQuant (Xiao et al., 2023) makes activations easier to quantize by smoothing them and compensating this operation with a transformation of the weights, resulting in improved results over ZeroQuant and LLM.Int8(). GPTQ is another good representative of one-shot quantization approaches designed especially for LPMs (Frantar et al., 2022a). GPTQ builds on the learnings from Optimal Brain Quantization (OBQ) (Frantar et al., 2022b) and applies layer-wise quantization to the full-precision weights of a base LPM. We incorporate GPTQ as the default quantization method in SQFT's pre-fine-tuning stage.

Parameter-efficient Fine-tuning (PEFT) Due to their large number of parameters, it is too costly to fine-tune pre-trained large models. Updating all their weights to improve their performance in a downstream task might require devices with large memory capacity. PEFT techniques attempt to address this challenge by avoiding the update of all weights in the pre-trained model. For instance, low-rank (LoRA) adapters (Hu et al., 2022) use a fraction (often less than 1%) of additional weights to adapt the model to a new task. LoRA adapters, $\boldsymbol{B}$ and $\boldsymbol{A}$, are utilized to reparameterize a linear projection, $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$, keeping the weights, $\boldsymbol{W}$, frozen and updating only the low-rank adapter matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$, i.e., $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{X}$.

Algorithm 1 Hill-climbing Search Algorithm

Input: Number of turns T, Number of neighbors N, Neighbor step size S, Number of evaluation samples M, Heuristic configuration c_h, Validation dataset $\mathcal{D}$
Output: Optimal configuration c^*

$c_a \leftarrow c_h$ ▷ *Initialize anchor with the heuristic configuration*
$V \leftarrow \{c_h\}$ ▷ *Initialize the set of visited configurations*
$\mathcal{D}_M \leftarrow$ Sample($\mathcal{D}$, M) ▷ *Create a proxy dataset by randomly sampling M samples from $\mathcal{D}$*
for $t \leftarrow 1$ to T **do**
  $\mathcal{C} \leftarrow$ Neighbor-sample(c_a, N, S) - $\mathcal{V}$ ▷ *Sample N unvisited S-step neighbor configs*
  $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{C}$ ▷ *Add the sampled configurations to the set of visited configurations*
  $c_m \leftarrow$ MaxAcc(Eval($\mathcal{D}_M$, $\mathcal{C}$)) ▷ *The config with the maximum accuracy on proxy data*
  if $Acc(c_m) > Acc(c^*)$ **then**
    $c_a \leftarrow c_m$ ▷ *Update anchor configuration if the new configuration has higher accuracy*
  end if
end for
$c^* \leftarrow c_a$ ▷ *The optimal configuration is the final anchor configuration*
return c^*

Recently, Shears proposed Neural Low-rank Adapter Search (Munoz et al., 2024a) and demonstrated that LoRA adapters can be made elastic to allow for the application of weight-sharing schemes and keeping the original weights of the model frozen and compressed, e.g., inducing sparsity before the fine-tuning stage. However, a challenge that emerges is that merging the dense adapters with the sparse weights results in the overall loss of sparsity. LoRAPrune has attempted to address this challenge by using the weights and gradients of the LoRA adapters to remove elements in the model's weights (Zhang et al., 2023). As demonstrated in the main sections of the paper, SQFT proposes an alternative method for merging the dense adapters with a minimal drop in accuracy.

B Hyperparameters

The hyperparameters used in our main experiments are shown in Table 7.

C Hill-climbing search algorithm

We propose Algorithm 1 to start from the reference configuration (Section 3.1) and systematically explore its neighbors. Table 4 in the main paper shows the benefits of using any available budget to execute this algorithm and discover better-performing models.

D Additional Sparsity Levels and Ablation Studies for Llama-3 on GSM8K

We conducted additional experiments and ablations studies with different sparsity levels and compared the underlying NLS approach to LoRA. Table 8 shows that up to high sparsity levels, SQFT delivers high-performing models.

Table 7: Hyperparameters used in our experiments. For all approaches with NLS, we explored several manually designed search spaces and identified the optimal configuration for each pipeline. Note that in our experiments involving GSM8K and math instruction tuning, we conducted trials over 3 or 4 epochs and reported the best results achieved. Interestingly, SQFT with QA-SparsePEFT often necessitates extended training periods to exploit its quantization-aware capabilities fully.

Model	Task	Sparsity	Method	Epoch	Batch size	Learning rate	Adapter rank	Adapter alpha	Adapter target modules
Llama-3-8B	GSM8K	50%	LoRA	3	16	3e-4	32	64	Q, K, V, Up, Down
			Shears	3	16	3e-4	32,28,24,20,16	64	Q, K, V, Up, Down
			SQFT + SparsePEFT	3	16	3e-4	48,32,16	64	Q, K, V, Up, Down
			GPTQ + LoRA	3	16	3e-4	32	64	Q, K, V, Up, Down
			SQFT	3	16	3e-4	40,32,24	64	Q, K, V, Up, Down
			SQFT + QA-SparsePEFT	4	16	3e-4	48,32,16	64	Q, K, V, Up, Down
Phi-3-Mini-4K-Instruct	Math	50%	LoRA	3	16	3e-4	32	64	Qkv
			Shears	3	16	3e-4	48,40,32,24,16	64	Qkv
			SQFT + SparsePEFT	3	16	3e-4	48,32,16	64	Qkv
			GPTQ + LoRA	3	16	3e-4	32	64	Qkv
			SQFT	3	16	3e-4	32,28,24,20,16	64	Qkv
			SQFT + QA-SparsePEFT	4	16	3e-4	32,24,16	64	Qkv
Phi-3-Mini-4K-Instruct	CS	50%	LoRA	3	16	1e-4	16	32	Qkv
			Shears	3	16	1e-4	16,12,8	32	Qkv
			SQFT + SparsePEFT	3	16	1e-4	16,12,8	32	Qkv
			GPTQ + LoRA	3	16	1e-4	16	32	Qkv
			SQFT	3	16	1e-4	16,12,8	32	Qkv
			SQFT + QA-SparsePEFT	3	16	1e-4	16,12,8	32	Qkv

Table 8: Ablation studies for various sparsity levels (**Llama-3-8B** with GSM8K).

Model	Sparsity	Method	Mergeable	Final Precision (Base + Adapter / Base)	Fine-tune Approach	GSM8K Test Accuracy(%)
Llama-3-8B	0%	w/o tune	-	FP16	-	50.0
	20%	*w/o Quantization*				
		w/o tune	-	FP16	-	47.5
		Shears	✗	FP16 + FP16	LoRA	58.7
					NLS	**61.2**$_{+2.5}$
		SQFT + SparsePEFT	✓	FP16	LoRA	60.3
					NLS	**62.0**$_{+1.7}$
		Quantization				
		w/o tune	-	INT4	-	36.6
		SQFT	✗	INT4 + FP16	LoRA	57.8
					NLS	**60.0**$_{+2.2}$
		SQFT + QA-SparsePEFT	✓	INT4	LoRA	54.7
					NLS	**55.6**$_{+0.9}$
	30%	*w/o Quantization*				
		w/o tune	-	FP16	-	40.9
		Shears	✗	FP16 + FP16	LoRA	58.2
					NLS	**59.8**$_{+1.6}$
		SQFT + SparsePEFT	✓	FP16	LoRA	60.0
					NLS	**61.2**$_{+1.2}$
		Quantization				
		w/o tune	-	INT4	-	30.3
		SQFT	✗	INT4 + FP16	LoRA	56.7
					NLS	**57.6**$_{+0.9}$
		SQFT + QA-SparsePEFT	✓	INT4	LoRA	54.8
					NLS	**56.0**$_{+1.2}$
	40%	*w/o Quantization*				
		w/o tune	-	FP16	-	31.6
		Shears	✗	FP16 + FP16	LoRA	56.9
					NLS	56.9
		SQFT + SparsePEFT	✓	FP16	LoRA	**57.9**$_{+1.5}$
					NLS	56.4
		Quantization				
		w/o tune	-	INT4	-	20.1
		SQFT	✗	INT4 + FP16	LoRA	54.9
					NLS	54.9
		SQFT + QA-SparsePEFT	✓	INT4	LoRA	53.4
					NLS	**53.7**$_{+0.3}$
	50%	*w/o Quantization*				
		w/o tune	-	FP16	-	12.5
		Shears	✗	FP16 + FP16	LoRA	50.6
					NLS	**52.2**$_{+1.6}$
		SQFT + SparsePEFT	✓	FP16	LoRA	50.6
					NLS	**52.5**$_{+1.9}$
		Quantization				
		w/o tune	-	INT4	-	7.0
		SQFT	✗	INT4 + FP16	LoRA	48.9
					NLS	**50.0**$_{+1.1}$
		SQFT + QA-SparsePEFT	✓	INT4	LoRA	48.2
					NLS	**50.2**$_{+2.0}$
	60%	*w/o Quantization*				
		w/o tune	-	FP16	-	-
		Shears	✗	FP16 + FP16	LoRA	39.9
					NLS	**45.3**$_{+5.4}$
		SQFT + SparsePEFT	✓	FP16	LoRA	40.7
					NLS	**42.5**$_{+1.8}$
		Quantization				
		w/o tune	-	INT4	-	-
		SQFT	✗	INT4 + FP16	LoRA	40.1
					NLS	**42.0**$_{+1.9}$
		SQFT + QA-SparsePEFT	✓	INT4	LoRA	37.6
					NLS	**40.9**$_{+3.3}$
	70%	*w/o Quantization*				
		w/o tune	-	FP16	-	-
		Shears	✗	FP16 + FP16	LoRA	25.5
					NLS	**27.9**$_{+2.4}$
		SQFT + SparsePEFT	✓	FP16	LoRA	22.1
					NLS	**24.9**$_{+2.8}$
		Quantization				
		w/o tune	-	INT4	-	-
		SQFT	✗	INT4 + FP16	LoRA	24.2
					NLS	**25.2**$_{+1.0}$
		SQFT + QA-SparsePEFT	✓	INT4	LoRA	**22.6**$_{+0.2}$
					NLS	22.4